Reference Guide

mTROPOLIS™

Version 2.0

mTropolis Reference Guide

Email: mtropolis@quark.com

Web: http://www.quark.com

Trademark Information

Third-Party Companies & Products

Third-party companies and products are mentioned for informational purposes only and are neither an endorsement nor a recommendation. Quark assumes no responsibility with regard to the selection, performance, or use of these products. All understandings, agreements, or warranties, if any, take place between the vendors of these products and their respective users.

Introduction

Introduction

The *mTropolis Reference Guide* is your best resource for answering "How does it work?" questions about mTropolis. The *Reference Guide* provides detailed coverage of the mTropolis™ authoring environment — including all menu items, windows, palettes, and other tools — as well as full documentation of all features available for use in your authored projects.

If you're new to mTropolis, you should start with the *mTropolis Developer Guide*. The *Developer Guide* provides detailed coverage of mTropolis concepts, tutorials, and examples — everything you'll need to build a foundation of knowledge to apply mTropolis quickly and effectively.

If you're already familiar with mTropolis, you should start by reviewing the *mTropolis Quick Reference*. This 16-page booklet summarizes most of the features — modifiers, system messages and commands, object attributes, Miniscript syntax, and Miniscript functions — available for use in your authored projects.

Welcome!

Welcome to mTropolis. mTropolis is a visual, object-oriented authoring system for creating networked, interactive, multimedia applications. Designed for the demanding requirements of consumer CD-ROM developers, mTropolis is now available for mainstream multimedia authoring. mTropolis combines a rich set of features and effects, a time-saving visual interface, and a behavior-oriented programming system to enable rapid development of high-performance multimedia projects. Whether you're creating presentations, interactive product brochures, entertainment or educational CD-ROMs, computer-based training, or some other category of new media application, mTropolis will accelerate your development process and ease collaboration with other development team members. Projects created in mTropolis may be distributed on CD-ROM, DVD, or the Internet, and will run on Microsoft Windows, Mac OS, Netscape Navigator, and Internet Explorer platforms.

Installing mTropolis

To install the mTropolis authoring environment, player, and supporting components on your Macintosh system, mount the mTropolis CD-ROM and follow the instructions in the "Read Me First!" file at the top level of the disc.

To install the mTropolis player, mPire™ plug-in, and other components on your Windows system, mount the mTropolis CD-ROM and follow the instructions in the "README.WRI" file at the top level of the disc.

Please read the "Release Notes 2.0" file installed with mTropolis for late-breaking information about this release.

Starting mTropolis

To start mTropolis, double-click the mTropolis icon. The first time mTropolis is launched, you will be asked for your name, organization, and registration number. Please enter this information to personalize and enable your copy of mTropolis.

At regular intervals, mTropolis will check to see if another copy of the software with the same registration number is running elsewhere on the network. If another copy with the same registration number is detected, an alert will appear and mTropolis will quit. To avoid this inconvenience, please ensure that your copy of mTropolis is installed and run on only one machine at any time, as specified in your *mTropolis End User License Agreement*.

The default installation of mTropolis includes the *mTropolis Info* interactive title, which will appear every time you start mTropolis. To disable *mTropolis Info*, simply remove this file from the **Startup** subfolder of your **mPlugins** folder.

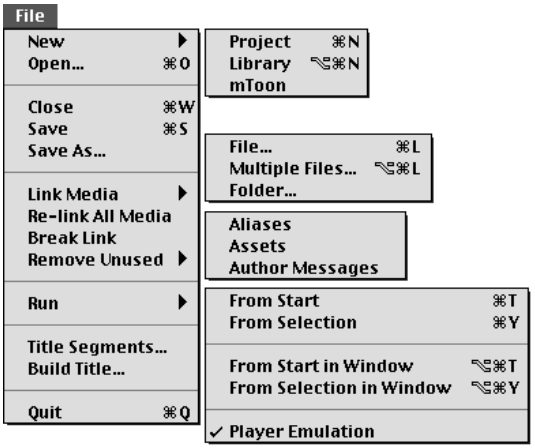
Learning mTropolis

For a comprehensive "roadmap" of resources that you can use to learn more about mTropolis in ways that suit your individual learning style, please refer to the Introduction in the *mTropolis Developer Guide*.

1. File Menu

1. File Menu

The **File** menu provides options for: creating, opening, closing and saving projects, creating and opening libraries, creating, opening, closing and saving mToons, linking external media to projects, breaking links to external media, removing unused objects from projects, switching projects from edit mode to run-time mode, building titles, and quitting mTropolis.



*The **File** menu*

New, Open, Save, and Close for Projects

Titles created with the mTropolis authoring environment are referred to as *projects* during the development phase. A mTropolis project file contains programming and formatting information as well as information on media elements that have been linked to the project (but it does not contain copies of the media files themselves).

When mTropolis is launched, or when a new project is created by choosing **New ➛ Project** from the **File** menu, mTropolis creates an untitled project with a single untitled section (Untitled Section), subsection (Untitled Subsection), shared scene (Untitled Shared Scene) and scene (Untitled Scene). See "Structure Window Overview" in Chapter 8 for more information on the components of a mTropolis project.

Although the components of a project can also be viewed from two other windows (the Structure window and the Layers window), the Layout window is the default view when mTropolis creates a new project or opens an existing project.

More than one project can be opened simultaneously, although your system's RAM imposes a practical limit on the number of projects that can be open at once. See "Equipment and Memory Required" in the Introduction of the *mTropolis Developer Guide* for suggested hardware requirements for mTropolis.

Creating a new project

Choose **New ➛ Project** from the **File** menu (⌘-N). A new, untitled project is created and an empty project Layout window appears.

Opening an existing project

Choose **Open** from the **File** menu. A standard file selection dialog box appears.

Closing a Project

Make sure the Layout window is in the foreground. Choose **Close** from the **File** menu (⌘-W). If changes have been made to the project since the last save, an alert appears asking if you want to save your changes before closing the project.

Choose **Don't Save**, **Cancel**, or **Save.** If **Save** is chosen and the project has been saved previously, mTropolis re-saves the project under the existing name. If the project hasn't been saved previously, a standard file dialog box appears, requesting a name and a destination before saving.

!!! Note: Alternatively, projects can be closed by clicking in the close box in the Layout window's title bar.

Saving a Project

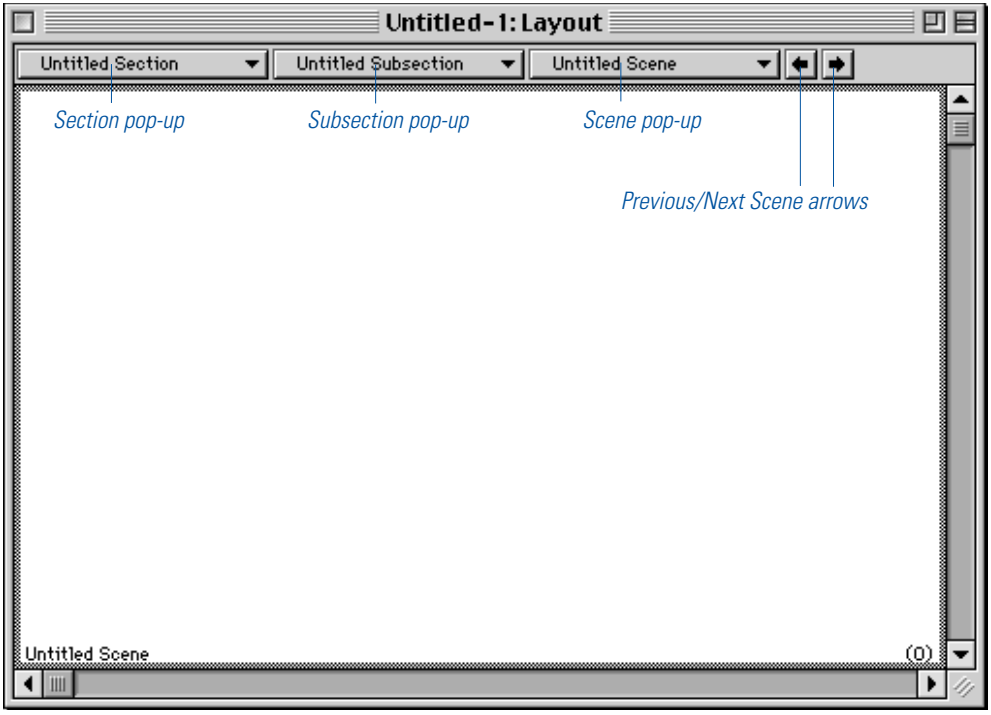
Choose **Save** from the **File** menu (⌘-S). If the project has been saved previously, mTropolis re-saves the project under the existing name. If the project hasn't been saved previously, a standard file dialog box appears, requesting a name and a destination before saving. Any subsequent changes to the project are saved under the selected file name.

Note: Saving a project saves its structural elements, their associated modifiers, and all links to external media. Media files are not saved with the project. They are stored outside the mTropolis environment.

Renaming a project

Choose **Save As** from the **File** menu to save the current project under a new name. A standard file dialog box appears, requesting a new name and location before saving.

Note: Saving a project using **Save As** reclaims the space used by deleted scenes, elements, and assets that were previously linked and then dragged to the **Asset** palette's trash can. Periodically using **Save As** during the development of a project can help keep the project's file size smaller.



A new, empty Layout window

New, Open, and Save for mTropolis Libraries

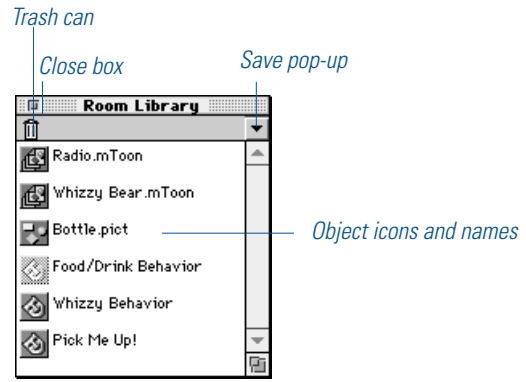
Libraries can be used to store project components (sections, subsections, scenes, elements, and modifiers) in a file that is separate from the project. When a new library is created, or an existing library is opened, a **Library** palette appears on the screen. mTropolis components can be dragged to and from this palette. Multiple libraries can be open at once.

Libraries allow portions of projects to be distributed to development team members for selective editing. Libraries can also be used to archive project components for easy use in other projects.

!!! Note: Projects cannot be stored in libraries, though any other mTropolis component, including entire sections, subsections, and scenes can be stored.

Creating a new library

Choose **New** ➛ **Library** from the **File** menu (⌘-Option-N). A new, untitled library appears.



A ***Library*** palette

Opening an existing library

Choose **Open** from the **File** menu to display a standard file selection dialog box. Choose a library file from the dialog box. The **Library** palette appears.

Adding items to a library

Drag sections, subsections, scenes, elements, or modifiers from a project and drop them into the **Library** palette. The component added to the library is represented by an icon along with its name.

!!! Note: Modifiers that are stored in libraries retain their settings. However, some modifiers may have dependencies; that is, they may be dependent upon additional information to function correctly. For example, Vector Motion modifiers are dependent upon Vector Variables. If a functioning Vector Motion modifier is placed in a library and moved to another project, it will have to be reconfigured to reference a new Vector Variable.

Deleting items from a library

Drag the item to the **Library** palette's trash can, in the upper left corner of the palette. The next time the library is saved, it is saved without the deleted items.

Saving a library

Click the **Library** palette's pop-up arrow. A pop-up menu appears.

Choose **Save** from the pop-up menu to save the library under its current name. If the library hasn't been saved previously, a standard file dialog box appears, requesting a name and a destination before saving. Choose **Save As** to save the library under a new name.

Closing a library

Close a library by clicking the close box in the **Library** palette's title bar. If changes have been made to the library since it was last saved, an alert appears, asking if you want to save your changes before closing the library.

New and Open for mToons — the mTropolis Animation Format

mTropolis includes support for proprietary, cel-based animation format called mToon files. An mToon is a series of images compiled by mTropolis into a single file. Any animation created in a 3D or 2D program that has been saved as, or converted to, PICT or PICS files can be imported by the mToon editor and processed as an mToon.

mToons are cel-based and very flexible. In addition to being able to specify the playback rate and duration of an mToon, a selected cel or range of cels in the mToon can be specified for playback. These ranges can be accessed during run-time, opening new creative possibilities to the multimedia author. For example, a character's sitting, walking, and running motions can be compiled into a single mToon linked to an element. During run-time, messengers or Miniscript can be used to specify which cels or range of cels in the animation to play back according to predefined conditions.

Single or multiple PICS or PICT files can be imported by the mToon editor. Ranges of cels within each animation can be defined and named. The mToon can be tested within the editor, optionally compressed and then saved as an mToon. Existing mToons can also be opened and edited in this window.

Creating a new mToon

To create a new mToon:

Choose **New ➛ mToon** from the **File** menu to open the mToon editor window. The mToon editor window appears.

Opening an existing mToon

Choose **Open** from the **File** menu to display a standard file selection dialog box. Choose an mToon file from the dialog box. The mToon editor window appears, displaying the mToon frames. If source files for the mToon have changed since the mToon was last saved, a dialog box appears. See "mToon Update Features" in this chapter for more information about the relationship between mToons and their source files.

The mToon editor window

The mToon editor window is used to create and modify mToons. This window displays a single cel of the mToon animation, along with a number of controls. Controls for this window are described below.

Note: Standard **Cut**, **Copy**, **Duplicate**, **Paste**, and **Undo** menu items from the **Edit** menu can also be used while working in the mToon editor window.

Registration point

By default, the registration point of each cel appears in its upper left corner. The registration point is the point used to align cels when the **Align and Trim Cels** mToon menu option (later in this chapter) is selected. Use the Selection tool to drag the registration point to a new location, or hold down the Option key and click in the mToon window to change the cel's registration point.

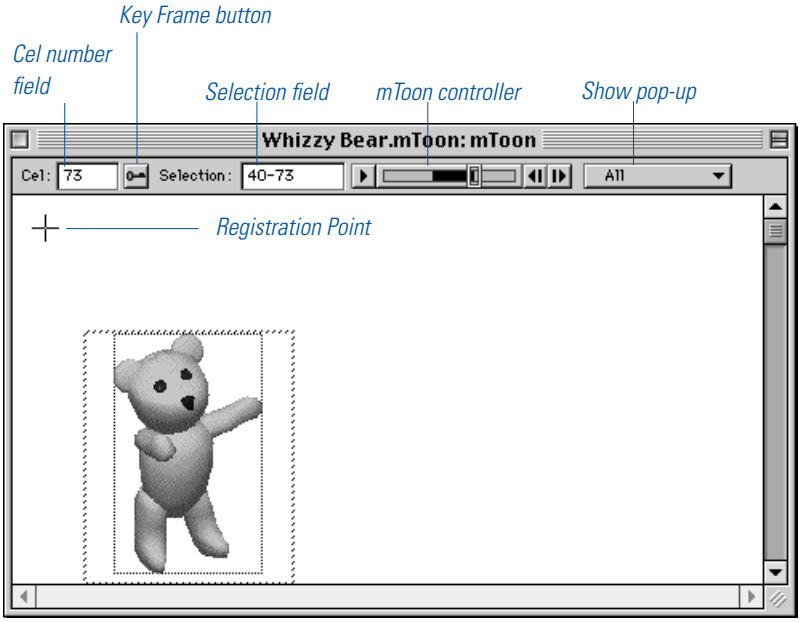
Cel number field
This field displays the number of the current cel.

Key Frame button
Select this button to make the currently-selected frame a key frame. This button is only available if the mToon compression options are configured to use temporal compression (see "Temporal Compression Check Box" in this chapter). If temporal compression is not selected, every frame is considered a key frame.

Selection field
This field displays the range of cels that has been selected for editing.

mToon controller
The controller is used to preview the animation, as well as to select a range of cels for editing. Use the step buttons to step forward or backward through the animation one cel at a time. Hold down the Shift key while stepping through cels to select a range of cels. Or, hold down the Shift key while dragging the slider to select a range of cels.

Show pop-up menu
Options in this menu work in conjunction with the controller. Choose **All** to play all of the cels in the animation. To limit playback to a specific range of cels, select the name of the range from the **Show** pop-up menu.



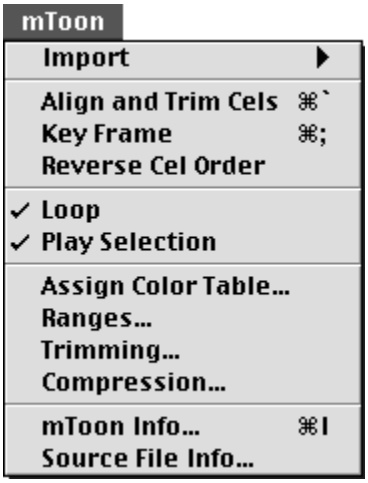
The mToon Editor window

The mToon Menu

When the mToon Editor window is open, the **mToon** menu is available from the mTropolis menu bar.

Cels from PICT files, PICS files, QuickTime video files, or existing mToons can be imported by the mToon editor by using the options in the **Import** submenu. Other items in this menu, including play controls, editing tools, and file info options, relate specifically to the cels in the mToon Editor window.

mToon menu options are described below.



*The **mToon** menu*

Import

Choose a menu item from the **Import** submenu in the **mToon** menu to import cels from a single file, multiple files, or a folder of files for processing. PICT, PICS, QuickTime, and mToon files can be imported.

Files within folders are imported according to the order assigned to them by the Finder. If any cels are currently in the mToon editor window, cels from the newly-imported file(s) are inserted sequentially after the last cel. The **Edit** menu's **Cut**, **Copy**, and **Paste** functions can be used to rearrange the order of individual cels after they have been imported.

Tip: When creating source files in an external program, such as a 3D rendering package, save the animation frames as individual PICTs and compile them in the mToon editor. Saving files this way allows changed frames to be imported much more easily than if they are rendered to PICS files (wherein multiple images are stored as a single file).

Align and Trim Cels

Choose **Align and Trim Cels** from the **mToon** menu (⌘-` [Command-left single quote]) to align the registration points of cels in the animation. At the same time, the background area of cels is trimmed based on the background color of the cells. By default, this color is white, but can be changed using the **Trimming** dialog box (see "Trimming" later in this chapter).

Key Frame

Choose **Key Frame** from the **mToon** menu to make the frame currently displayed in the mToon window a key frame. This option is only available if the mToon compression options are configured to use temporal compression (see "Temporal Compression Check Box" in this chapter). If temporal compression is not selected, every frame is considered a key frame. Choosing this menu option is the same as choosing the key frame button on the mToon window.

Reverse Cel Order

Choose **Reverse Cel Order** from the **mToon** menu to reverse the order of cels in the mToon. By default, the order of all cels in the animation is reversed. If a range of cels has been selected with the mToon controller, the order of only those cels is reversed.

mToon menu play controls — Loop and Play Selection

Two options in the mToon menu allow the author to control the way an animation is played in the mToon editor window.

1. Check the **Loop** menu toggle to repeatedly play the entire animation.
2. Check the **Play Selection** menu toggle to play a selected series of cels in an animation.
3. Check both the **Play Selection** and **Loop** menu toggles to loop a selected range of cels.

Assign palette

Each animation created in the mToon editor can have its own custom palette.

To assign a palette:

1. Choose **Assign Palette** from the **mToon** menu. A standard file dialog box appears.
2. Choose the CLUT (color look-up table) file to be assigned to the animation.

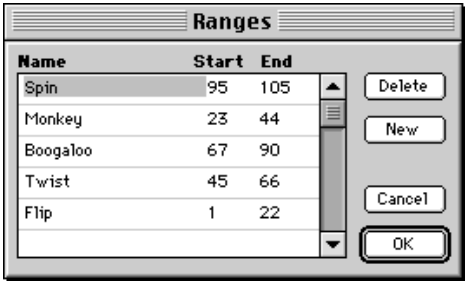
By default, the mToon editor is set to display 256 colors. Use the **Compression** dialog box (see "Compression" earlier in this chapter) to change this setting.

Animations will be dithered to the system palette if no palette is assigned.

To be used in a project, mToons (and their custom palettes, if any) must first be linked to elements in the project. To display an mToon created with a custom color palette, place a Color Table modifier in the scene the mToon is in and choose the table from the modifier's color table pop-up menu. The effect of the new CLUT will now be visible in runtime. To view the effect of a color table while in edit mode, choose a color table from the **Preview Color Table** submenu in the **View** menu.

Ranges

Choose this option to define sequences of cels in an mToon as named ranges. The **Ranges** dialog box appears. mToon ranges can be accessed by messengers or Miniscript during run-time. To play a named mToon range, send a **Play** command to the mToon with the desired range name as accompanying data. See "Played/Play" in Chapter 13.



*The **Ranges** dialog box*

Tip: To simplify programming, order your animations in a logical progression of cels whenever possible. For example, if several characters are to perform the same actions (running, jumping, etc.), design the cel ranges of these actions in the same pattern to reuse as much of the programming of ranges as possible.

To define a new range of cels:
While in the mToon Editor window, select a range of cels to be defined by holding down the Shift key while dragging the slider on the controller. The section of the bar corresponding to the selected cels is highlighted.

Choose **New Range** from the mToon editor's **Show** pop-up menu to display the **New Range** dialog box and type the name of the new range. Alternatively, choose **Ranges** from the **mToon** menu to display the **Ranges** dialog box.

Components of the **Ranges** dialog box are described below.

Delete button
To delete a range, highlight the range name and click this button.

New button
To create a new range from the **Ranges** dialog box, click this button. An untitled range will be added to the bottom of the list. To define the length of the new range, enter new values in the Start and End fields.

Name field
Enter the new range's name or edit the name of an existing range in this text field.

End field
The last cel of the range appears in this field.

Start field
The first cel of the range appears in this field.

Cancel button
Click this button to close the **Ranges** dialog box and disregard any changes.

OK button
Click this button to dismiss the **Ranges** dialog box and keep any changes.

Note: Use the Tab key to move through the fields in the **Ranges** dialog box.

Trimming

When the **Align and Trim Cels** menu item is selected, in addition to aligning the cels, the mToon editor automatically trims the white space from the background of each cel. This process optimizes the playback speed of an animation.

To specify a new color to be trimmed, or to turn Trim background information off, choose **Trimming** from the **mToon** menu. The **Trimming** dialog box appears.

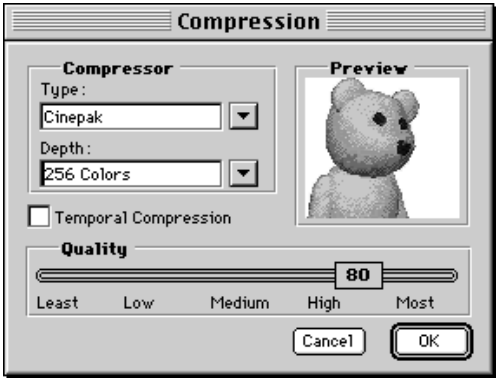
*The **Trimming** dialog box*

Trim background information
To turn default trimming off, uncheck the "Trim background information" check box.

Trim Color check box
To choose a new trim color, click and hold on the swatch. The cursor will change to an eyedropper. Choose the background color to be trimmed from the animation and release the mouse button. Click **OK** to confirm changes. Choose **Align and Trim Cels** from the **mToon** menu (⌘-') to trim the newly-selected background color from the cels.

Compression

mToons can be compressed for optimal playback during run-time. mTropolis accesses the software compressors supplied with QuickTime. Choose **Compression** from the mToon menu to access the **Compression** dialog box (see below). Components of this dialog box are described below.



*The mToon **Compression** dialog box*

Type pop-up menu

This pop-up specifies the compression method to be used. The visual area displays a preview of the effect of the selected compression method.

Available compression methods (codecs) will vary according to the version of QuickTime installed on the author's system, but standard QuickTime options are described below:

- **Animation:** Use the Animation compressor for compression of images that were originally in digital form (animation and computer generated content) and were not obtained by video tape. The Animation compressor employs a compression algorithm developed by Apple based on run-length encoding techniques. The animation compressor works in a lossy or a lossless mode, and supports both spatial and temporal compression. This compressor can play back images at up to 30 fps at full screen resolution; the performance and compression ratios you achieve depend on the type of image you are using.
- **Cinepak:** Use the Cinepak compressor when compressing 16-bit and 24-bit video for playback from CD–ROM disks. This compressor attains higher compression ratios, better quality, and faster playback speeds than Video compression. For best results, use the Cinepak compressor on raw source data that has not been previously compressed with a highly lossy compressor. Decompression is much faster than compression with Cinepak, and the data rate for playback can be defined by the end user.
- **Component Video:** A compression algorithm designed for use when capturing video. This codec is of limited use as an mToon compression method.
- **Graphics:** Specially designed for 8 bit (256 color) stills where compression ratio (file size) matters more than compression speed. Generally this choice will reduce an image to half the size of a file compressed using the animation compressor, but will take twice as long to do it.
- **None:** By choosing None, mToons remain uncompressed.
- **Photo-JPEG:** JPEG (Joint Photographic Experts Group) is an international standard for compressing still images. Use the Photo compressor for images that contain smooth transitions, or that do not contain a high percentage of edges or other sharp detail. Most natural images fall into this category. For this type of 24-bit image, the Photo compressor produces a reconstructed image that is virtually indistinguishable from the original image

at a compression ratio of 10:1. Compression time is equal (or very nearly equal) to decompression time.

- **Video:** Use the Apple Video compressor for capture and compression of analog video, high-quality playback for hard disk, and moderate quality playback from CD–ROM. This compressor supports both spatial and temporal compression, and can playback at rates of 10 fps or more. Data can be later recomposed or recompiled for higher compression ratios. The Apple Video compressor allows recompression with minimal or no quality degradation.
- **mFactory Animation:** This codec is Quark's own cross-platform, QuickTime-independent compression method for mToons. This codec works on 8-bit (256 color) and 16-bit (thousands of colors) mToons. Since it is supplied by Quark, and is not part of the QuickTime distribution, it can be used in either Mac OS or Windows titles without QuickTime having to be licensed or installed on target machines.

  When the **Depth** pop-up menu is set to 256 colors (8-bit) this compressor is nearly identical to QuickTime's Animation compressor and uses a lossless, run-length encoding (RLE) scheme.

  When the **Depth** pop-up menu is set to thousands of colors (16-bit) the "Quality" slider becomes adjustable. Set to a Quality of 100, the mToon is compressed using a lossless, run-length encoding (RLE) scheme. When set to values lower than 100, a lossy compression algorithm is used. The algorithm used is similar to the lossy compression used in the QuickTime Animation compressor.

  The **Build Title** dialog box contains options for automatically converting uncompressed or QuickTime Animation codec compressed mToons to the mFactory™ Animation codec at build time. See "mToon Conversion Options Section" later in this chapter.

Note: Some compression methods use delta compression, where one frame is differenced from a previous frame. When random access to an individual cel is made during run-time, a lag in access time results, since the accessed cel must be recreated from one or more previous cels.

Depth menu

Sets the number of colors used by a new mToon. The default color depth is 256 colors. Some compression types only allow certain color depths.

Temporal compression check box

Check this box to perform temporal compression on the mToon. By default this option is off. Temporal compression will shrink the file size of the mToon, but individual frames are no longer randomly accessible. mToons that are not temporally compressed may play faster than ones that are. Note that this option is not available for all compression settings. When this option is selected, the **Key Frame** button on the mToon Editor window becomes active (see "Key Frame Button" earlier in this chapter).

Compression quality slider

The position of this slider affects the quality of the chosen compression method. Lower quality settings result in smaller file sizes and faster mToons, but introduce artifacts.

Cancel

Click **Cancel** to ignore changes to compression settings.

OK
Click **OK** to accept changes made to compression settings.

Tip: Larger animations that are loaded from a CD will need to be compressed to load more quickly. Files that need to run very quickly can be left at a compression of none, however, beyond 1 MB in size, they will take more time to load. In the prototype stage, test various compression methods to determine which is appropriate for the specific animation you are building.

mToon Info

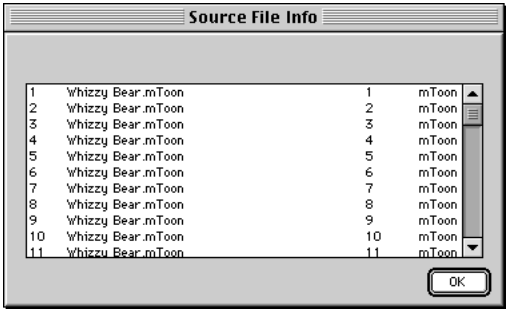
Choose **mToon Info** from the **mToon** menu to display the **mToon Info** dialog box.

The **mToon Info** dialog box displays information about the current animation, including the full path to the file, the file size, number of cels, compression settings, and color depth.

*The **mToon Info** dialog box*

Source File Info

Choose **Source File Info** from the **mToon** menu to display information about the source files that are linked to the current animation. The **Source File Info** dialog box appears. More information about the relationship between mToons and their source files can be found in "mToon Update Features" earlier in this chapter. Components of the **Source File Info** dialog box are described below.



*The **Source File Info** dialog box*

The source files for the cels of an animation are displayed in a list. The list elements, from left to right, are:

Cel column
The cel number assigned to the source file.

Original file name
The name of the original source file.

Index number from animation file
If the source file is a PICS, mToon, or QuickTime file, the index number of the cel is displayed in this column.

File format
The file format of the imported file is listed in this column.

Saving mToons

mToons can be saved using standard procedures.

Saving a new mToon

With the mToon window selected as the current window, choose **Save** from the **File** menu (⌘-S). If the mToon has been saved previously, mTropolis re-saves the mToon under the existing name. If the mToon hasn't been saved previously, a standard file dialog box appears, requesting a name and a destination before saving. Any changes made to the mToon are saved under the file name.

Renaming an mToon

With the mToon window selected as the current window, choose **Save As** from the **File** menu to save the mToon under a new name. A standard file dialog box appears, requesting a new name and a destination before saving.

mToon update features

Information about the source files from which an mToon animation is created is stored inside the mToon file. This information can be seen by selecting **Source File Info** from the mToon menu. If source files have been changed, the mToon editor can be used to easily recompile the mToon from the altered source files. Open the mToon file and mToon Editor window by choosing **Open** from the **File** menu. If you choose an mToon file that has a changed source file, mTropolis displays an alert saying that source files have changed and they are being updated. The mToon editor appears with the updated animation cels. Choose **Save** from the **File** menu to recompress the mToon with the changed cels.

Opening Projects, Libraries, and mToons

Previously-saved projects, libraries, or mToon files can be opened using standard procedures. Choose **Open** from the **File** menu (⌘-O) to open an existing project, library, or mToon.

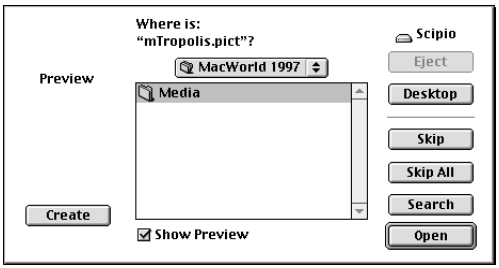
!!! Note: Double-clicking a project, library, or mToon from the Finder will open the chosen file and will also launch mTropolis if it isn't already launched.

Re-linking external media files

When a project, library, or mToon is saved, mTropolis saves the links (path names) to its external media files. If any media files were deleted, moved, or renamed since the project, library, or mToon was last saved, mTropolis automatically searches for the media.

This search starts from the location of the project file and continues down through the folder and disk volume hierarchy. Initially, mTropolis searches for media contained in the first scene and its elements. When the project opens and navigates to the scene containing missing media, mTropolis searches for those files during run-time or edit mode.

If the name of a media file has changed since the project was last opened, the search will be unsuccessful and a file dialog box is displayed.



Re-linking a missing media file

When this dialog box appears, the following options are available:

1. The **File** dialog box can be used to navigate to the missing file. Once a file is chosen, the **Open** button changes to **Re-link.** Click **Re-link** to link the chosen file to the project, replacing the original link.

2. Click the **Search** button to display a folder selection dialog box. Choose a folder in which to search for the media. That folder and all its subfolders will be searched. If the file is found, it is automatically re-linked to the project. If the file is not found, the **File** dialog box is displayed again.

3. Click the **Skip** button to ignore the need for this file and return to the project. If the file was previously linked to an element, its thumbnail (if available) is used in place of the actual media. Even if the project is saved, this media file will again be searched for the next time the project is opened. To remove any reference to the file, it must be deleted from the **Asset** palette.

4. Click the **Skip All** button to ignore the need for this file and any others that may also be missing. This option is otherwise identical to the **Skip** option.

Link Media — Attaching External Media Files to Elements

mTropolis stores the path to external media files and refers to these files as required. The **Link Media** submenu in the **File** menu is used to establish the path or "link" between the project and external media files.

The **Link Media** command can be used to link media to a selected element in any of editing views in mTropolis — the Structure, Layout, and Layers windows — or multiple files can be linked to the **Asset** palette. Use **Link Media ➛ File** (⌘-L) to link a single file to a selected element. Use **Link Media ➛ Multiple Files** (⌘-Option-L) to link a number of media files to the **Asset** palette. Use **Link Media ➛ Folder** to link all supported media files found in a folder to the **Asset** palette.

Instructions on linking media to elements in each of the mTropolis editing views can be found elsewhere in this document (as described below). The rest of this section contains important general information about file linking in mTropolis. New users should read "File names for Linked Media" and "Media Types Supported by mTropolis" later in this chapter before proceeding.

- For instructions on how to create and link media to elements in the structure view, see "Creating New Sections, Subsections, Scenes, and Elements" in Chapter 8.
- For instructions on linking media in the layout view, see "Linking Media to an Element in the Layout Window" in Chapter 9.
- For instructions on linking media in the layers view, see "Linking Media to Elements in the Layers Window" in Chapter 10.
- For information on linking media to the **Asset** palette, see "Linking Media to the Asset Palette" in Chapter 11.

File names for linked media

Internally, mTropolis keeps track of media assets by their file names, not by their locations on the hard disk. Therefore, any media file that is linked to a mTropolis project should have a

unique file name (not just a unique location on the disk). If a media file is linked to a mTropolis project that already contains an asset with the same file name, mTropolis will replace the previous asset with the new one.

For example, suppose you have two images you want to use as buttons, and they both have the same file name (for example, myButton.pict). Even if the files are located in different directories (one is on the desktop and another is in a folder named "Media" on the hard disk), mTropolis will consider them to be the same file when linked to the project. If the "myButton.pict" file from the desktop is linked to an element first and sometime later the other "myButton.pict" file from the "Media" folder is linked to a different element, the original "myButton.pict" file will be replaced with the new one.

To avoid this type of problem, ensure that all assets used in the mTropolis title have unique names before linking them. In our example, the "myButton.pict" files could be renamed to "myButton1.pict" and "myButton2.pict" to avoid confusion.

Media types supported by mTropolis

The following media formats can be linked to mTropolis elements:

- **Images:** PICT files, including PICT files saved with JPEG compression (but JPEG compression is not preserved when the PICT is written into a built title file). Note that mTropolis does not retain alpha-channel information from 32-bit images.
- **Animation:** mToons, proprietary animation format in mTropolis. See "New and Open for mToons — the mTropolis Animation Format" earlier in this chapter. mToons can be created from source files in PICT, PICS, or QuickTime formats.
- **Sound:** AIFF files of any bit-depth and sampling rate, though compressed sounds are not currently supported. mTropolis can make use of AIFF sound markers when playing sounds, as described in "Played/Play" in Chapter 13.
- **Video:** QuickTime movies can be linked to elements in edit mode. Multitrack QuickTime movies can be created with the MovieTrax™ application (described in Appendix A, "MovieTrax") and manipulated with the Track Control Modifier (Chapter 12). Video for Windows files (AVI files) can be used in titles built for Windows platforms as described in "Make Movies External Check box" later in this chapter, but AVI files cannot be imported directly into the mTropolis editing environment.
- **Panoramic Video:** QuickTime VR panorama and object movie files can be linked to a mTropolis project. Currently, QuickTime VR movies always play "direct to screen"; they are not part of the mTropolis layer order and ink effects cannot be applied to them.
- **Color Tables:** To provide appropriate colors for 8-bit images, mTropolis supports color tables (palettes) in the CLUT (Color Look Up Table) format supported by many imaging applications, including Adobe Photoshop. CLUT files can only be linked to the Color Table modifier or to the **Asset** palette — they cannot be linked directly to graphic elements. See "Color Table Modifier" in Chapter 12.

Re-link All Media

Choose **Re-link All Media** from the **File** menu to cause mTropolis to search for any missing media assets in the entire project. The search technique used and the re-linking options are the same as described in "Re-Linking External Media Files" earlier in this chapter, except that the entire project (not just the first scene) is searched.

Break Link

Select **Break Link** from the **File** menu to break an element's link to a media file while leaving all other information related to the element intact. To use this feature:

- Select the element. Use Shift-click to select multiple elements.
- Choose **Break Link** from the **File** menu. All aspects of the selected elements, including their frames and modifiers, will remain intact, but they are no longer linked to their media files. Note also that the media files continue to be linked to the project and their thumbnails remain visible in the **Asset** palette. The **Remove Unused ➛ Assets** menu option can be used to remove such assets from the project.

Remove Unused

The **Remove Unused** cascading menu option can be used to remove unused aliases, assets, and author messages from the current project. The three options are described in more detail below.

Remove Unused Aliases

Choose **Remove Unused ➛ Aliases** from the **File** menu to remove any aliased modifiers that are not actually used in the current project. To identify which aliases are unused, mTropolis must load the entire project into memory. If parts of the project are not yet loaded, mTropolis displays a warning that the project will be searched. Once the unused aliases have been identified, they are removed from the project and their names and icons disappear from the **Alias** palette.

Remove Unused Assets

Choose **Remove Unused ➛ Assets** from the **File** menu to remove media assets that have been linked to the current project but are not used anywhere in the project's structural hierarchy. To identify which assets are unused, mTropolis must load the entire project into memory. If parts of the project are not yet loaded, mTropolis displays a warning that the project will be searched. Once the unused assets have been identified, they are removed from the project and their names and thumbnails disappear from the **Asset** palette.

Remove Unused Author Messages

Choose **Remove Unused ➛ Author Messages** from the **File** menu to remove author messages that have been created in the project but are not actually used by any of the modifiers in the project. To identify which author messages are unused, mTropolis must load the entire project into memory. If parts of the project are not yet loaded, mTropolis displays a warning that the project will be searched. Once the unused author messages have been identified, they are removed from the project and their names are removed from the "Author Messages" section of the **Message/Command** and **When** modifier pop-up menus. The unused

author messages are also removed from the Author Messages window, along with any empty author message groups.

Run — Switching Between Edit and Run-time Modes

When mTropolis is launched, the program is in edit mode. Authoring occurs in *edit mode,* but the project can be previewed by entering run-time mode. In *run-time mode,* the project is shown as it will appear to end users.

Running a project from its first scene

Choose **Run ➛ From Start** from the **File** menu (⌘-T) to switch to run-time mode and preview the project from its first scene. This scene is the first scene in the first subsection in the first section you see in the Structure window.

Normally, the project is displayed over a black background. To keep the mTropolis interface visible during run-time, press ⌘-Option-T instead. Note that mTropolis palettes are hidden and that the **Section**, **Subsection**, and **Scene** pop-up menus at the top of the Layout window are unavailable, indicating that mTropolis is in run-time mode.

Press ⌘-T again to return to the scene where ⌘-T was originally pressed. Optionally, use ⌘-. (Command-period) to re-enter edit mode with the current run-time scene displayed in the Layout window.

Running a project from a specific scene

Choose **Run ➛ From Selection** from the **File** menu (⌘-Y) to switch to run-time and preview the project from the current scene.

Normally, the project is displayed over a black background. To keep the mTropolis interface visible during run-time, press ⌘-Option-Y instead. Note that mTropolis palettes are hidden and that the **Section**, **Subsection**, and **Scene** pop-up menus at the top of the Layout window are unavailable, indicating that mTropolis is in run-time mode.

Press ⌘-Y again to return to the scene where ⌘-Y was originally pressed. Optionally, use ⌘-. (Command-period) to re-enter edit mode with the current run-time scene displayed in the Layout window.

Player Emulation

The **Player Emulation** menu toggle controls the way in which the mTropolis editor's run-time mode functions. When this option is checked (the default), the editor's run-time mode emulates the behavior of the stand-alone mTropolis player applications. The project behaves much as it would if it were built into a title file and run with a mTropolis player. However, the following differences and issues should be noted:

- If their values are changed, variables located *above* the scene level (that is, contained in the Subsection, Section, or Project components) always retain their changed values upon return to edit mode. Other variables (such as variables at the scene level and lower) that change during run-time are reset to their original values. The behavior of variables in run-time mode can be a source of confusion when testing a mTropolis project. For more information on this topic, see "Variable Persistence" in Chapter 13.
- Any changes to the attributes of scenes are retained upon returning to edit mode. Attributes of other components that change in

run-time are reset to their original values upon return to edit mode.

- If the project has not been saved prior to entering run-time mode, mTropolis saves the current project to a temporary file so that the project can be fully restored upon returning to edit mode.
- When entering run-time mode, all objects that have been loaded into memory (any scenes that have been edited, media assets, thumbnail versions of assets, etc.) are purged from memory. Upon returning to edit mode, these objects are reloaded to restore the editing environment to its previous state.

  However, it is possible that due to insufficient memory, mTropolis will not be able to restore the structure view to the same state in which it was left. In this case, an alert is displayed and the structure view appears with all of its open/close triangles closed. Note that no project data is lost in this situation — the structure view is just reset to its simplest state.

Run-time behavior when player emulation is disabled

When the **Player Emulation** option is not checked, changes made in run-time mode are persistent (they are retained upon returning to edit mode). In this mode, persistent changes include:

- Components created by the **Clone** command become part of the project in edit mode.
- Components destroyed by the **Kill** command are no longer present in the project and cannot be recovered.
- Attributes (such as position, size, visible, layer, parent, asset, etc.) changed during run-time remain changed in edit mode. Note that this feature can be useful as an editing technique — for example, a Miniscript modifier could be used to accurately set the positions and layers of numerous elements. The script could then be removed.
- Draggable elements moved in run-time retain their new positions in edit mode. Similarly, elements that have moved via vector motion retain their new positions.
- The values of *any* variables that are changed in run-time remain changed in edit mode. The behavior of variables in run-time mode can be a source of confusion when testing a mTropolis project. For more information on this topic, see "Variable Persistence" in Chapter 13.

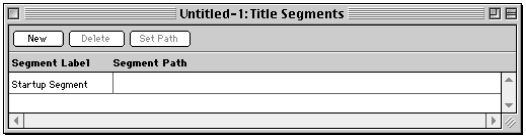
Note: Switching **Player Emulation** off causes an alert to appear. This alert notifies the author that all changes made in run-time mode will be preserved. To eliminate this alert's reappearance, click the **Don't Warn Again** button when the alert appears.

Title Segments

Choose **Title Segments** to display the Title Segments window. Use this window to create and manage title segments — groups of project sections that are written to the same file when the project is built into a stand-alone title (see "Build Title" in this chapter).

By default, a mTropolis project is built into a single title file that can be "played" by a mTropolis player. However, large projects may result in a single title file that is too large to fit on the desired distribution media. For example, the project may need to be split up for distribution on multiple CDs. Use the Title Segments window to create a title that consists

of multiple title segment files. Components of this window are described below.



The Title Segments window

New button

Click this button to create a new title segment. A new segment label with a default name appears in the "Segment Label" list.

Delete button

Click this button to delete the currently highlighted title segment. A segment label or segment path must be highlighted for this option to be active.

Set Path button

Click this button to choose a folder into which this title segment will be written when the title is built. A standard folder selection dialog box appears. When a folder is selected, its path appears in the "Segment Path" list.

Segment Label list

This list shows the names of currently defined title segments. Click a name to edit the name or to select it for the **New, Delete,** or **Set Path** operations. Note that the segment labeled "Startup Segment" cannot be edited or deleted — all mTropolis titles have a startup segment.

Segment Path list

Items in this list show the path to the location in which the corresponding title segment will be written. There are two ways to change the path:

- Choose the path and edit it from the keyboard. The path must be entered in proper Mac OS path syntax (that is, a colon separated list of drive/folder names).
- Choose the path or corresponding segment label and click the **Set Path** button. A standard folder selection dialog box appears. When a folder is chosen, its path appears in the list.

Assigning sections to title segments

By default, all sections in a project are assigned to the Startup Segment. When they are created, new title segments are not associated with any sections in the project. Sections must be assigned to new title segments for those segments to be created during the "Build Title" process. To assign a section to a title segment:

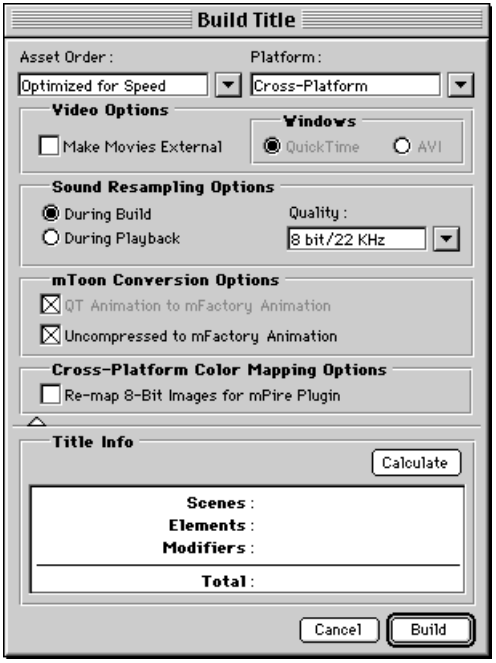
- Open the Structure window (if it is not already open) by choosing **Structure Window** from the **View** menu (⌘-2).
- In the Structure window, choose the section to be reassigned, then choose **Object Info** from the **Object** menu. The **Section Info** dialog box appears. You can also double-click the section to display this dialog box.

*The **Section Info** dialog box*

- Use the **Segment Label** pop-up menu to choose the segment for this section. A new segment can be created from this pop-up menu by choosing **New Segment Label.** A prompt appears for entering the new label.
- Click **OK** to confirm the new title segment association and dismiss the **Section Info** dialog box.

Build Title

When the project is complete, it can be built into a stand-alone title for Mac OS or Windows platforms. Select **Build Title** from the **File** menu to display the **Build Title** dialog box. Components of this dialog box are described below.



*The **Build Title** dialog box*

Asset Order pop-up menu

Select an optimization option from this menu. These options control the way in which media assets are written into the built title file or files. Options include:

- **Optimized for Speed:** Select this option to place an instance of each media asset in the order in which it is used in the project. For every scene in which an asset appears, a copy of the asset is written to the built title file. For example, if an asset appears in three scenes, it will be written to the title file (or files, in the case of segmented titles) three times.

  This option creates a title with optimal performance, at the expense of disk space. To conserve space, individual assets can be set to write only once each time they appear in a segment. A check box in the **Asset Info** dialog box controls this behavior. See "Build Options Section" in Chapter 5.

- **Optimized for Space:** Choose this option to place only the first instance of an asset in the project. Choosing this option makes the built title smaller than it might otherwise be when built for speed. However, scenes may take longer to load as media used in multiple places in the project must be loaded from a single location in the built title file (or files, if multiple segments are written).

Platform pop-up menu

Use this pop-up to choose the type of platform for which the title will be built. Options are:

- **Cross-Platform:** Choose this option to build a title that can be played by any mTropolis player. The stand-alone Mac OS and Windows players, as well as the Mac OS and Windows mPire plug-in players can be used to play the title. The title file is built using Windows

media formats and byte ordering. When run on a Mac OS platform, some translation from these Windows formats occurs, causing a small decrease in performance.

This is the preferred built title format for distributing network titles with the mPire plug-in players. See "Delivering mPire Titles" in Chapter 16.

Note: Not all features are available when using Windows versions of the mTropolis player — see Appendix C, "Cross-Platform Authoring Issues."

- **Mac OS:** Choose this option to build a title that can only be played by the mTropolis players for Mac OS platforms. The title is written with Mac OS media formats and byte ordering, insuring highest performance on Mac OS platforms.

- **Windows:** Choose this option to build a title that can only be played by the mTropolis players for Windows platforms. Note that not all mTropolis features are available in titles built for Windows — see Appendix C, "Cross-Platform Authoring Issues."

Video Options

The Video Options section is enabled when building titles for Windows platforms (when "Windows" is selected in the **Platform** pop-up menu). These options control the way QuickTime and Video for Windows files are handled when building Windows titles.

Make Movies External check box

Check the **Make movies external** check box to save movies as external files instead of including them in the title segment file(s). The video files are saved in the folders of the segments that require them. If multiple segments of the title require the same movie, the movie file is duplicated and placed in each appropriate folder.

Note: If this option is selected when building a Windows title, QuickTime movies are automatically written in QuickTime's "playable on Windows" format.

Windows buttons

These buttons become active when building a title for Windows only platforms:

- **QuickTime:** Click this button (the default) to use the same QuickTime files used when authoring the title in the final build.

- **AVI:** Click this button to replace the QuickTime files used when authoring the title with AVI (Video for Windows) versions in the final build. To use this option, you must have AVI versions of the QuickTime files — mTropolis does not automatically convert video between these formats. Put the AVI files in the same location(s) as their QuickTime counterparts before starting the build title process. These AVI files must have the same names as the QuickTime files, except for a ".AVI" extension. For example, the file "myMovie.mov" must have an AVI version named "myMovie.AVI".

Note: Quicktime VR panorama and object movies cannot be converted to AVI format.

Sound Resampling Options

The Sound Resampling Options section is available when building titles for Cross-Platform or Windows distribution. These options control the way sound media files are handled in titles run with Windows versions of the mTropolis player.

During build
Resampling during the build process results in all sounds in the project being resampled to conform to the bit resolution and sample rate selected in the **Quality** menu. The sounds are upsampled or downsampled as they are written to the title files. As a result, they may take up more or less disk space than their original versions, but Windows will never have to perform sound conversions during run-time.

During playback
Resampling during playback causes all sounds to be written to the title file at their native quality. However, no matter what their actual bit resolution and sample rate, they are resampled as specified by the **Quality** menu when they are played. Thus, the Windows player must perform sound conversions during run-time. Depending upon the speed of the playback machine and the number of conversions being performed, this conversion may result in reduced performance. Note also that if the playback machine does not support the specified sample rate, the next best quality will be used.

Quality
Use this pop-up menu to specify a bit resolution and sample rate for use with the "During build" and "During Playback" sound options. Options are:

- **16 bit/44 KHz:** The highest quality sound supported by mTropolis, "CD quality." Sounds written in this format require the maximum amount of disk space.
- **16 bit/22 KHz:** High-quality, 16-bit sound.
- **8 bit/22 KHz:** The default medium-quality, 8-bit sound.
- **8 bit/11 KHz:** The lowest-quality sound supported by mTropolis. Sounds written in this format require the least disk space.
- **Dynamic:** This special **Quality** option is only available when "During playback" has been selected as the sound resampling option. When this option is enabled, sounds are written to the title file at their native quality.

  At run-time, they are played at their native quality, except when multiple sounds are made to play at the same time. When multiple sounds are played simultaneously, the quality of the first sound played is used to resample any others. For example, if an 8-bit, 22KHz sound is playing, then a 16-bit, 44KHz sound is made to play at the same time, Windows will "downsample" the 16-bit sound (adversely affecting its quality). Similarly, if the 16-bit, 44KHz sound started playing first, the 8-bit 22KHz sound would be "upsampled."

  In addition to matching the bit-depth and sample rate of sounds, this behavior also causes the sounds' stereo or mono qualities to be matched. That is, if a stereo sound plays first, any simultaneously played mono sounds are converted to stereo (and vice-versa).

  Thus, this setting can cause unpredictable sound performance on Windows and should only be used in projects where the programmer has carefully controlled the order in which sounds of differing qualities might be played.

mToon Conversion options section
Two check boxes in this section of the **Build Title** dialog box control build-time compression options for uncompressed mToons and mToons compressed with the 8-bit QuickTime Animation codec. These options affect both Mac OS and Windows builds:

- **QT Animation to mFactory Animation check box:** When this check box is selected

(the default), mToons compressed with the 8-bit and 16-bit (256 colors and thousands of colors) QuickTime Animation codec are recompressed using the mFactory Animation codec when written into the built title file. The mFactory Animation codec has the advantages of not requiring QuickTime to be installed and providing a cross-platform compression method for mToons. In general, it offers better performance than the QuickTime Animation compressor. See "Compression" earlier in this chapter.

- **Uncompressed to mFactory Animation check box:** When this check box is selected (the default), 8-bit and 16-bit (256 colors and thousands of colors) mToons saved without compression are recompressed using the lossless mFactory Animation codec when written into the built title file. The compressed mToons take up less space than uncompressed mToons and have comparable playback performance. See "Compression" earlier in this chapter.

Cross-Platform Color Mapping options

This section of the **Build Title** dialog box, active only when building for cross-platform distribution, has one option. Check the "Re-map 8-Bit Images for mPire Plug-in" check box to automatically re-map the colors in any 8-bit graphics used in the project to the Netscape palette. The colors are re-mapped using a closest match approximation and any color table assets in the project are replaced with the Windows Netscape palette.

This option is provided mostly as a convenience feature. Since the Mac OS 8-bit system palette is very similar to the Netscape palette, images mapped to the Mac OS palette can be easily converted to the Netscape palette with a minimum of artifacts. Therefore, when authoring a title for distribution as a 256 color mPire title, 8-bit images that use the "Mac OS 8-bit" palette can be used in the mTropolis editing environment (instead of having to use images already mapped to the Netscape palette). Then select the "Re-map 8-Bit Images" option and build the title. The converted images will display the same in the mPire player as they did in the mTropolis editor.

Note that 8-bit images that use palettes other than the Mac OS palette may not appear acceptable when converted to the Netscape palette by mTropolis. In such cases, it may be necessary to use an image editing application to create new versions of the 8-bit images that use the Windows Netscape palette (which can be found on the mTropolis CD–ROM) or the Mac OS 8-bit system palette.

Title Info section

Click the triangle at the bottom of the **Build Title** dialog box to display the Title Info section.

Click the **Calculate** button to calculate and display the number of scene, element, and modifier components present in the project.

Cancel button

Click **Cancel** to dismiss the **Build Title** dialog box and abort the build title process.

Build button

Click the **Build** button to begin compiling the project into a title. If there are title segments that do not have "segment paths" defined (see "Title Segments" earlier in this chapter), a standard folder selection dialog box appears for each title segment. Select a folder to be used to contain that segment's files. Note that once a location has been

selected, it is remembered for future builds (the Title Segments window can be used to change the path for the segment).

Files created by the Build Title process

mTropolis creates the following types of files during the Build Title process:

- **Startup Segment File:** All mTropolis projects have a Startup Segment file. This file is written to the folder specified in the Title Segments window.

  For cross-platform builds, this file is given the name of the project plus the file extension ".mfx". For example, when the project named "myProject" is built into a cross-platform title, the title's startup segment is named "myProject.mfx".

  For Mac OS builds, this file is given the name of the project plus the file extension ".mfm". For example, when the project is named "myProject", the title's startup segment is named "myProject.mfm".

  For Windows builds, this file is given the name of the project plus the file extension ".mfw". For example, when the project is named "myProject", the title's startup segment is named "myProject.mfw".

  These file name extensions are summarized in "mTropolis File Name Extensions and File Types" later in this chapter.

  Cross-platform startup segments have the icon shown below.

A cross-platform startup segment icon

  Mac OS and Windows startup segments have similar icons with a small "M" (for Mac OS) or "W" (for Windows) superimposed.

- **Other Segment Files:** If any title segments (other than the Startup Segment) were defined and associated with project sections, corresponding data segment files are written. They are written to folders as specified in the Title Segments window.

  For cross-platform builds, these files are given the name of the project, appended with a number (starting at 2) that represents the order of the data segment, plus the extension ".mxx". For example, if the project named "myProject" is built into a multiple-segment title, the data segments are named "myProject2.mxx", "myProject3.mxx", etc.

  For Mac OS builds, these files are given the same types of names, but with the extension ".mxm" ("myProject2.mxm").

  For Windows builds these files are given the same types of names, but with the extension ".mxw" ("myProject2.mxw").

  These file name extensions are summarized in "mTropolis File Name Extensions and File Types" later in this chapter.

  Cross-platform data segments have the icon shown below:

A cross-platform data segment icon

  Mac OS and Windows data segments have similar icons with a small "M" (for Mac OS) or "W" (for Windows) superimposed.

- **Video Folder:** If the **Make movies external** check box was marked in the **Build Title**

dialog box, a folder named "Video" is created for each segment that needs a video file. The folder resides in the same folder as its corresponding segment file. The "Video" folder contains the externalized video files, each named with a number and an extension (".mov" for QuickTime files, ".AVI" for Video for Windows files).

mTropolis features not supported in titles built for Windows platforms

Note that not all mTropolis features are available on the Windows platform in this release. See the mTropolis release notes for specific information about differences between titles built for Windows and Mac OS platforms.

Playing mTropolis Title Files

To play titles built with mTropolis, you need the following files:

- Segment files and folders created by the Build Title process.
- The appropriate version of mTropolis Player for the target machine. The mTropolis CD–ROM contains three Mac OS players — 68K-only version, Power Mac OS-only version, and a version for any Mac OS. There are two versions of the Windows player — "MTPLAY16.EXE" for Windows 3.1, and "MTPLAY32.EXE" for Windows 95/NT.
- The accompanying mPlugins folder (for Mac OS) or subdirectory (for Windows) for the player. If you are deploying a title for both Windows 3.1 and Windows 95/NT players, the contents of their "MPLUGINS" directories should be combined into a single "MPLUGINS" directory.

These files and "top-level" folders should be put in the same folder (for Mac OS or Windows 95) or subdirectory (for Windows 3.1).

Running the title

A mTropolis title can be played in one of four ways:

- Launching a mTropolis player with a startup segment in the same folder/directory causes that startup segment to be run. That is, the player automatically runs the startup segment located in the same directory. This is the recommended configuration for a mTropolis-built title.

Note: Only one startup segment should be placed in the player's folder/directory. If multiple startup segments are present in this directory, there is no way to determine which one will be run.

- Launching a mTropolis player without a startup segment present starts the player without playing a title. A startup segment can be choosen manually. On Mac OS, choose **Open** from the **File** menu. A file selection dialog box appears. Use the dialog box to choose a startup segment to be run. On Windows, a file selection dialog box appears automatically. Use the dialog box to choose a startup segment file (a ".MFW" or ".MFX") to be run.
- Double-clicking or running a startup segment file causes that title to run with a mTropolis player, if one is available. If multiple copies of the mTropolis player are installed, there is no way to tell which one will run.
- Dragging a startup segment file on a mTropolis player causes that title to run with that player executable.

During run-time, you can quit the mTropolis player by pressing ⌘-Q on Mac OS or ALT-F followed by **x** on Windows.

Customizing title file names, location, and icons

The mTropolis player executable and the startup segment can be renamed or moved to another folder/directory with the following restrictions:

- The player's mPlugins folder and the title's Video folder (if it has one) must also be moved to the same folder as the player. These folders must not be renamed.
- Other segment files (if defined) cannot be moved or renamed — the exact paths set in the **Title Segments** dialog box are used by the player to locate segment files other than the startup segment.

Tip: Since the startup segment and player can be moved together to another location, frequently-accessed scenes can be assigned to the startup segment, which can then be copied to the end user's hard disk for fastest access.

Mac OS or Windows resource editors can be used to customize the mTropolis player icon. A custom title with single-icon launching can be created by changing the name and icon of the mTropolis player (though both the player and startup segment must still reside on the end user's hard disk).

Deploying multiple-disc titles

Because the mTropolis player code may be loaded dynamically during run-time, the mTropolis player and its mPlugins directory must always be available on a mounted volume. For multiple-disc titles, this means that the player must be copied to the end user's hard disk. The startup segment and its Video folder (if it has one) must be copied to the same location as the player if the title is to run automatically.

To minimize the size of the startup segment, ensure that all sections in your project are assigned to other segments. Alternatively, use the startup segment on the hard disk to hold sections that will be accessed frequently or that contain high-bandwidth media.

mTropolis file name extensions and file types

The table below summarizes mTropolis file name extensions and their corresponding Mac OS file types.

Description	Extension	Mac OS File Type
Cross-platform startup segment	.mfx	MFmx
Cross-platform data segment	.mxx	MFxx
Mac-only startup segment	.mfm	MFmm
Mac-only data segment	.mxm	MFxm
Windows-only startup segment	.mfw	MFmw
Windows-only data segment	.mxw	MFxw

File name extensions for mTropolis built title files

Quitting mTropolis

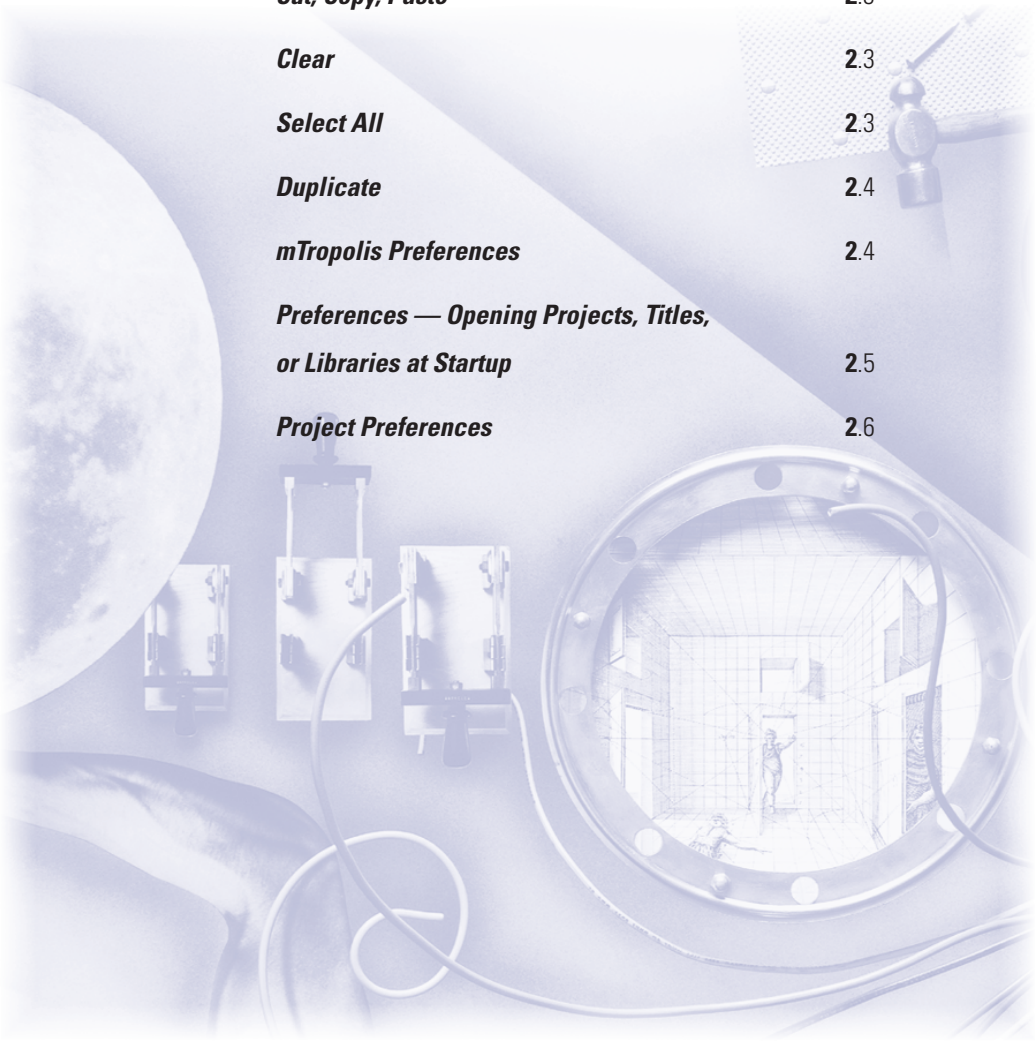
Choose **Quit** from the **File** menu (⌘-Q) to exit the mTropolis environment.

If changes have been made to the project since the last save, an alert appears, asking if you want to save your changes before closing the project.

Choose **Don't Save**, **Cancel**, or **Save.** If **Save** is chosen and the project has been saved previously, mTropolis re-saves the project under the existing name. If the project hasn't been saved previously, a standard file dialog box appears, requesting a name and a destination before saving.

!!! Note: If multiple projects are open, mTropolis quits each one in sequence.

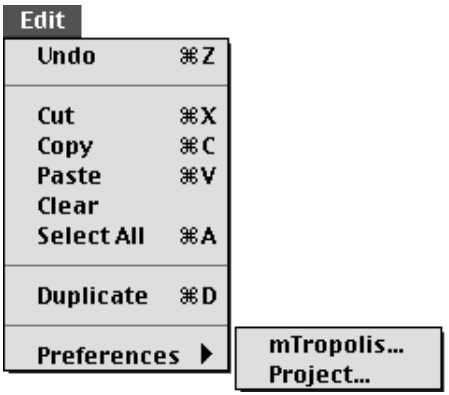
2. Edit Menu



2. Edit Menu

The **Edit** menu contains project editing functions such as **Undo**, **Cut**, **Copy**, and **Paste.** It also provides access to preferences for mTropolis and for the current project.

Most of the project editing functions are available in three project windows in mTropolis: Layout, Structure and Layers. Editing functions that are not available for the current selection are dimmed.



*The **Edit** Menu*

Undo

The **Undo** menu item restores the project to its state prior to the last change. To undo the last operation, choose **Undo** from the **Edit** menu (⌘-Z).

Cut, Copy, Paste

mTropolis has a clipboard where project components or text selections can be temporarily stored using the **Edit** menu's **Cut** and **Copy** menu items. The **Paste** menu item is used to place the clipboard's contents on selected locations in the current project, or on selected locations in another project. The menu items are context sensitive and do not allow invalid operations to be performed. For example, an element cannot be pasted at the project level.

Text selections, elements, modifiers, behaviors, project components, sections, subsections, and scenes can be cut, copied, and pasted. Items can be cut, copied, and pasted between project windows.

If the clipboard is empty, **Paste** is unavailable. To cut, copy, or paste an item:

1. Select the desired item with the **Tool** palette's Selection tool.
2. Choose **Cut**, **Copy**, or **Paste** from the **Edit** menu. Alternatively, you can use the keyboard shortcuts shown below:

Operations	Shortcut
Cut	⌘-X
Copy	⌘-C
Paste	⌘-V

Clear

The **Edit** menu's **Clear** command can be used to delete any item from the project, with the exception of items in palettes, which are deleted by dragging them to a palette trash can. Cleared objects are deleted without being copied to the clipboard. To clear an item from the project:

1. Select the item to be cleared in any window.
2. Choose **Clear** from the **Edit** menu. The item is deleted from the project.
3. You may also use the **Delete** key to clear items.

Select All

Use the **Select All** command (⌘-A) to select all of the items in an active window, including those not currently visible. This command can be used to quickly select a large group of items. After all items have been selected, individual items can be removed from the selection group by Shift-clicking them.

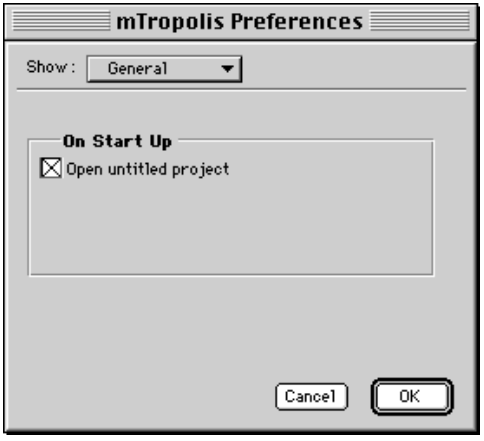
Duplicate

The **Edit** menu's **Duplicate** command creates a copy of a selected item. This option combines the functionality of **Copy** and **Paste** without affecting the contents of the clipboard. To duplicate an item:

1. Select the item with the **Tool** palette's Selection tool.
2. Choose **Duplicate** from the **Edit** menu (⌘-D). The duplicated item automatically appears near the original.

mTropolis Preferences

Choose **mTropolis** (**Edit** ➛ **Preferences**) to set preferences for mTropolis. The **mTropolis Preferences** dialog box appears. This dialog box can be used to change a number of different preference types. Use the **Show** pop-up menu to select different preference types. The dialog box changes to show applicable settings. These settings are described below.



General ***mTropolis Preferences*** dialog box

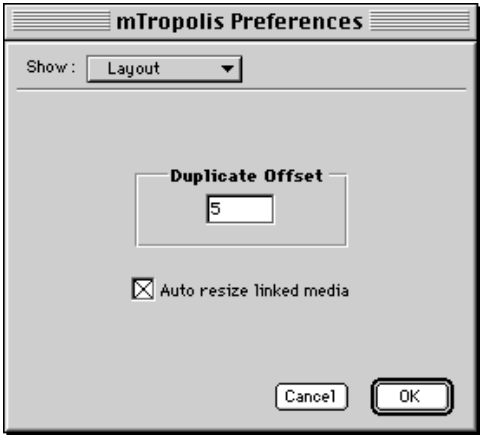
General

By default the **mTropolis Preferences** dialog box opens to display general mTropolis preferences. Components of this dialog box are described below.

Open untitled project on start up check box
When this box is checked, mTropolis opens a new, untitled project on startup. When deselected, mTropolis launches without creating an untitled project.

Layout

To set layout preferences, choose Layout from the **Show** pop-up menu in the **mTropolis Preferences** dialog box.

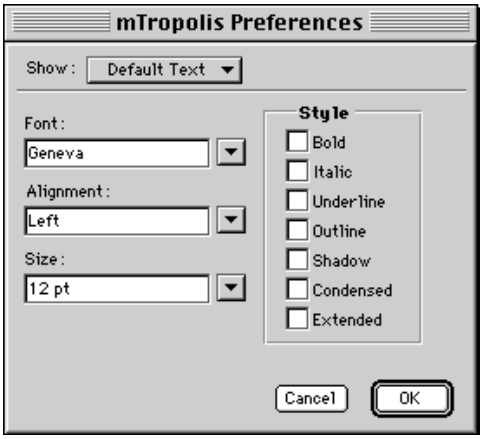


Layout ***mTropolis Preferences*** dialog box

Duplicate Offset field
Use this field to specify the offset for new elements created by choosing **Duplicate** from the **Edit** menu. By default, elements duplicated in the Layout window are offset 5 pixels to the right and 5 pixels below the original.

Auto resize linked media check box
When checked, this option automatically resizes elements to the screen dimensions of linked external media. The default is on.

Default text
To set preferences for the default type used in text elements, choose **Default Text** from the **Show** pop-up menu in the **mTropolis Preferences** dialog box.



*Default Text in the **mTropolis Preferences** dialog box*

Font pop-up menu
Choose mTropolis default text font from this pop-up menu.

Alignment pop-up menu
Choose mTropolis default text alignment from this pop-up menu.

Size pop-up menu
Choose mTropolis default text font size from this pop-up menu.

Style check boxes
Choose mTropolis default text font style by clicking the appropriate check boxes.

Preferences — Opening Projects, Titles, or Libraries at Startup

mTropolis can be configured to automatically open your favorite projects, titles, or libraries at start up. This feature is not controlled by the **mTropolis Preferences** dialog box, but is mentioned here because it is similar to other **mTropolis** and **Project Preferences.**

Any project, title, or library file put inside the **Startup** folder, found inside the mTropolis **mPlugins** folder (found in the same directory as your mTropolis editor/player) will be opened automatically whenever that copy of mTropolis is launched.

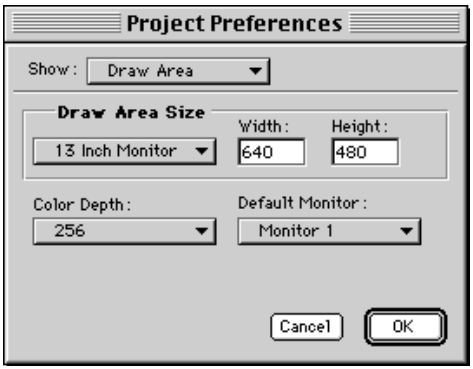
To have files open automatically whenever mTropolis is launched:

1. In the Mac OS Finder, locate the folder in which the mTropolis editor is installed.
2. Open the **mPlugins** folder found in that folder.
3. Open the **Startup** folder found inside the **mPlugins** folder. If the **Startup** folder does not exist, it can be created by selecting **New Folder** from the Finder's **File** menu.
4. Copy or move any mTropolis project, built title, or library files that you want to automatically open into the **Startup** folder.
5. When you launch mTropolis, these files will open automatically.

There is also a project preference that allows you to specify library files that will open whenever a specific project is opened. See the Libraries section in this chapter for details.

Project Preferences

Choose **Project** (**Edit** ➛ **Preferences**) to set preferences for the current project. The **Project Preferences** dialog box appears. This dialog box can be used to change a number of preference types. Use the **Show** pop-up menu to select preference types. The dialog box changes to show applicable settings, described below.



*Draw Area in the **Project Preferences** dialog box*

Draw Area

To set the draw area preferences, choose **Draw Area** from the **Show** pop-up menu in the **Project Preferences** dialog box. The draw area is the visible area of the project when viewed in run-time mode.

Draw Area Size pop-up menu
Select the size of the monitor on which the title will be displayed. Options include 9-Inch Monitor, 12-Inch Monitor, 13-Inch Monitor, 15-Inch Monitor, Current Monitor, or Custom.

Draw Area Size fields
These fields automatically display the draw area size of the monitor selected from the **Size** pop-up menu.

To specify a custom draw area size, choose **Custom** from the pop-up menu and enter new pixel values in these fields.

Color Depth pop-up menu
Choose the color depth of the project from this pop-up menu. Options include B/W (black and white), 256 colors (8-bit color), thousands (16-bit color), or millions (32-bit color).

This setting controls the bit depth of the off-screen buffer for the project. It has the following implications for media in built titles:

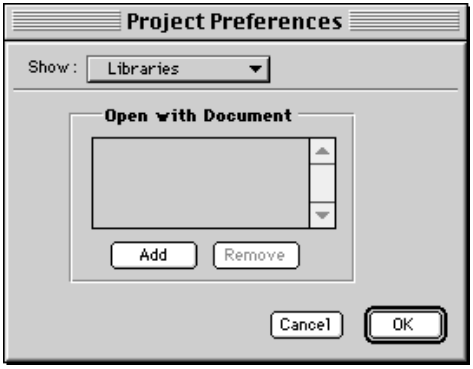
- When the project is built into a title file, PICT and mToon assets that have *greater* bit depths than the project are resampled to the specified bit depth (in an 8-bit title, 24-bit PICTs would be "downsampled" to 8-bit).
- Assets that have a bit depth *less* than the project are not resampled, thus preserving space in the built title file. For example, in a 16-bit title (thousands of colors), 8-bit (256 color) PICTs are written to the built title file as 8-bit graphics. Note that mTropolis displays 4-bit (16 color) and 8-bit (256 color) graphics with their correct colors in 16-bit and 32-bit titles without having to use Color Table modifiers. Color information is retrieved from the media file itself.
- QuickTime assets are not resampled at build time, though they will display at the specified color depth unless they are set to display Direct to Screen (see Direct to Screen check box in Chapter 5).

End users of the built title should set their monitors to this bit-depth for the title to display properly.

The supportsBitDepth System attribute can be used to check which resolutions a Mac OS monitor supports at run-time. The monitorBitDepth System attribute can be used to change the current color depth of a Mac OS monitor. See Chapter 15 for more information on attributes.

Default Monitor
Select the project's default display monitor (on systems with multiple monitors installed).

Libraries
To set the libraries preferences, choose Libraries from the **Show** pop-up menu in the **Project Preferences** dialog box. Use these preferences to specify any libraries that should be opened whenever the current project is opened.



*Libraries in the **Project Preferences** dialog box*

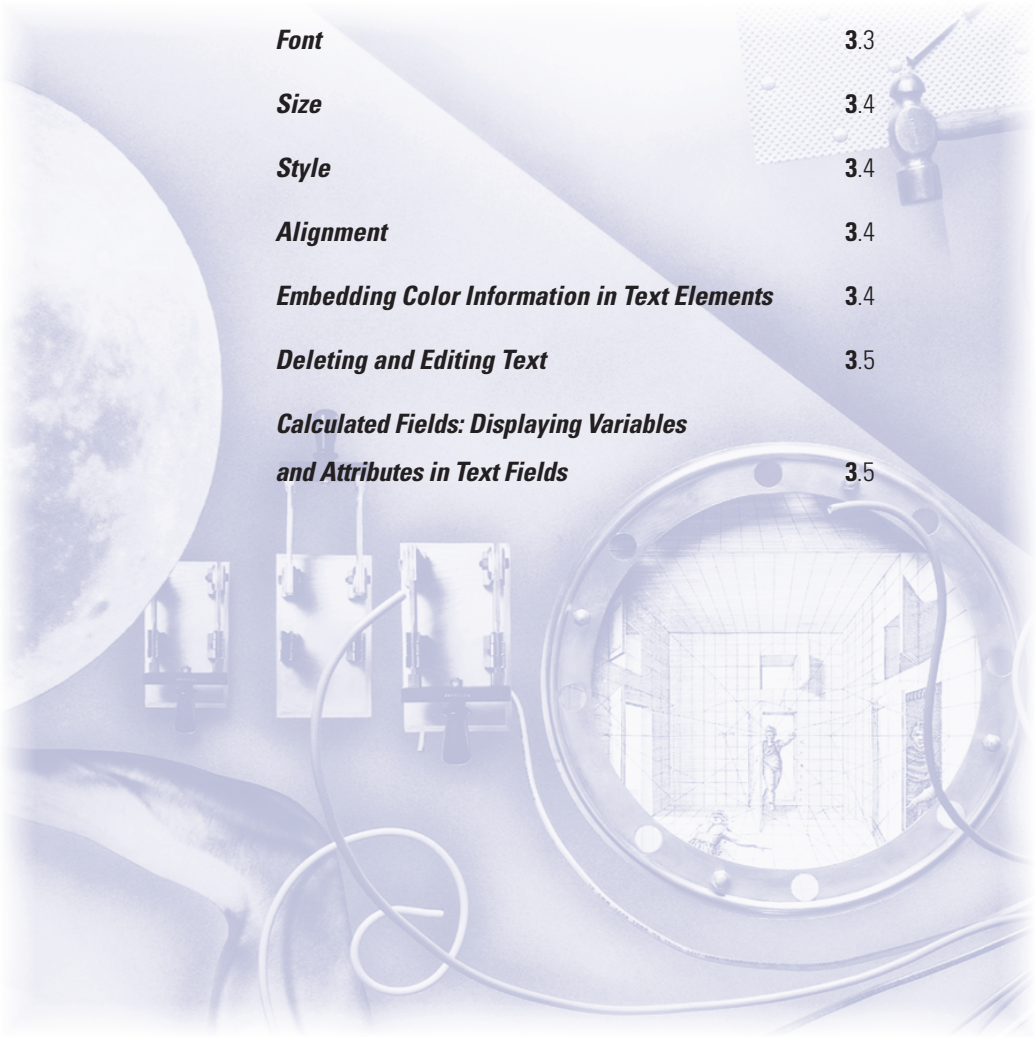
Open with document list
The library or libraries to be opened with the project are listed here.

Add button
Click the **Add** button to display a standard file dialog box. The libraries you select will be added to this list and will be opened with the project.

Remove button
To prevent a library from opening with the project, select it and click the **Remove** button.

There is also a way to specify that libraries, projects, and built-titles automatically open whenever mTropolis is launched. See "Preferences — Opening Projects, Titles, or Libraries at Startup" in this chapter.

3. Format Menu

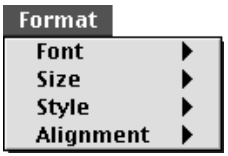


3. Format Menu

Options in the **Format** menu can be used to edit the characteristics of text in text elements. These options are available when the Text tool is clicked in a text object in the Layout window. These formatting options are applied only to the currently selected text in a text element. Multiple formatting options can be applied to a single text element. See the discussion below for more information about how these formatting options appear in built titles.

When a text element is first created, it sets up text formatting as specified in the Default Text selection (**Edit ➛ Preferences ➛ mTropolis**). See "Default Text" in Chapter 2. Text formatting can be changed from this default using the **Format** menu options.

The Text tool (the A icon found in the **Tool** palette) can be used to create text elements. See "Text Tool" in Chapter 11 for more information on creating text elements.



*The **Format** menu*

How mTropolis Handles Embedded Formatting in Text Elements

Text formatting applied to portions of a text element using the **Format** menu options is referred to as "embedded formatting."

Though the **Format** menu provides a convenient way to apply formatting to a text element, the following limitations should be kept in mind:

- Embedded formatting is not supported in titles built for the Windows platform. Note, however, that embedded formatting is still useful for creating text elements if those elements will be converted to bitmaps when the project is built into a title. See "Convert Text to Bitmap Check Box" in Chapter 5. Remember that text elements that are converted to bitmaps cannot be edited, cannot be changed via the text attribute, and cannot contain calculated fields.
- When Text Style modifiers (Chapter 12) are applied in run-time mode, they override all embedded formatting in a text element. When Text Style modifiers are disabled in run-time mode, the embedded formatting is not restored.
- When Graphic modifiers (Chapter 12) are applied in run-time mode, they override any embedded color formatting in a text element. When Graphic modifiers are removed, embedded color formatting is restored.
- When creating text elements that will not be converted to bitmaps, it is best to avoid using the options in the **Format** menu. Use Text Style modifiers and Graphic modifiers (Chapter 12) to create the desired formatting effects. These modifiers work on both Mac OS and the Windows platform. They also work when the contents of a text element change.

Format menu options are described below.

Font

Choose items from the **Font** submenu to change the font of highlighted text within a text element. mTropolis has access to any font currently installed on the system.

To change the font used by a section of a text element:
Select the Text tool in the **Tool** palette. The mouse cursor changes to an insertion bar.

1. Click the text element. The insertion bar appears at the start of the text in the text element.
2. Click and drag over the block of text to highlight the text to be modified.
3. Choose **Font** in the **Format** menu. Note that the name of the font currently used by the selected text has a check mark beside it in the **Font** submenu.
4. Choose a font from the submenu. The block of text changes to the specified style.

!!! Note: A similar procedure is used for most of the text formatting options described in this chapter.

Size

Choose items from the **Size** submenu to change the point size of highlighted text within a text element. The text in a text element can be set to one size, or blocks of text can be set to different sizes.

Font sizes are displayed in points. There are 72 points per 1 inch. Font sizes displayed as outlined text in the **Size** submenu represent installed screen font sizes. These sizes produce a better image on-screen.

To change the font size used in a text element:

1. The size of currently highlighted text is indicated by a check mark beside the size in the **Size** submenu.
2. To change the size of text, first highlight a specific block of text within an element and choose a **Size** option from the **Format** menu.

Style

Choose items from the **Style** submenu to change the style of one or more selected text elements, or highlighted text within a text element. The text in a text element can be set to one style, or blocks of text within a text element can be set to different styles.

To change the style used in a text element:

1. The style of currently highlighted text is indicated by a check mark beside the style name in the **Style** submenu.
2. To change the style of text, first highlight a specific block of text within an element and choose a **Style** option from the **Format** menu.

Alignment

Choose items from the **Alignment** submenu to align text within a text element. The options are **Left**, **Center**, or **Right** alignment. These options act on the entire contents of the text element.

The alignment of currently highlighted text is indicated by a check mark beside the alignment in the **Alignment** submenu.

To align text in a text element:

1. Choose the Text tool in the **Tool** palette.
2. Select the text element.
3. Choose **Alignment** in the **Format** menu.
4. Choose **Left**, **Center**, or **Right** from the submenu.

Embedding Color Information in Text Elements

Though this option is not listed in the **Format** menu, embedded foreground colors can be applied to a text element.

To assign an embedded color in a text element:

1. Choose the Text tool in the **Tool** palette.
2. Select the text element. The insertion bar appears at the start of the text in the text element.
3. Click and drag over the block of text to highlight the text to be modified.
4. Select a foreground color from the **Tool** palette's color swatch. See "Foreground and Background Colors" in Chapter 11.
5. The color of the highlighted text is changed.

Deleting and Editing Text

To delete text within an element:

1. Select the Text tool in the **Tool** palette.
2. Click and drag over a block of text to highlight it.
3. Press the **Delete** key (or choose **Cut** or **Clear** from the **Edit** menu). The text is cleared from the text element. Its frame and modifiers remain intact.

Calculated Fields: Displaying Variables and Attributes in Text Fields

A text element can be configured to display the contents of a variable or attribute during run-time. Whenever the field receives the **Update Calculated Fields** command, the contents of the field update to display the current value of the variable. Use the following technique:

1. Create a text element that contains the name of the variable to be displayed enclosed in `<` and `>` symbols as follows:

   `<variablename>`

2. The fields of Compound Variables can be displayed by using the same syntax used to access them in Miniscript. Use a period to separate the variable name from the desired field as follows:

   `<variable.field>`

   Multiple values can be displayed by enclosing the name of each variable in its own < > symbols. Variable names can also be mixed with regular text.

   Similarly, the values of object attributes can be displayed by enclosing a valid Miniscript object reference expression to the desired attribute in `<` and `>` symbols:

   `<object.attribute>`

3. For example, to display a String Variable named `user` that contains an end user's name and an Integer Variable named `score` that contains the end user's game score, the following text element entry could be used:

   ```
   <user> current score is: <score>
   ```

   The variables to be displayed must be within the scope of the text element (in other words, they must be located on or somewhere above the text element in the structural hierarchy).

   Similarly, the value of an attribute (such as the size of the current scene) could be displayed with the following text element entry:

   ```
   The size of the current scene is:
   <scene.size>
   ```

Continuing our example, the `user` and `score` variables might be placed on the scene.

4. In run-time mode, the initial values contained in the specified variables are substituted in place of the < > enclosed variable names. For example, upon switching to run-time mode, our text element might now show:

   ```
   Keith current score is: 1005
   ```

5. To update the text element to display the current contents of the specified variables, use a messenger to send the **Update Calculated Fields** command to the text element. See "Update Calculated Fields (Message/Command Menu Only)" in Chapter 13.

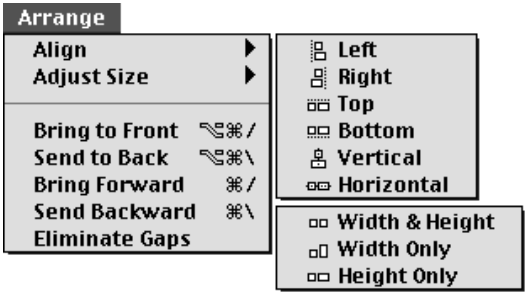
The text element updates to show the new values of the specified variables.

Note: Even though the visible contents of a calculated field change, the text element's text attribute does not update. It continues to contain the original syntax used to define the calculated field. Continuing our example, the end user sees `Keith current score is: 1005`, but the string contained in the text element's text attribute is still `<user>` current score is: `<score>`.

4. Arrange Menu

4. Arrange Menu

The **Arrange** menu provides options for aligning graphic and text elements along a specified edge, and for giving elements a common size. Options to change an element's layer order, and to eliminate gaps in the layer order, are also available in this menu.



Arrange Menu

Features of the **Arrange** menu are described below.

Align

Choose **Align** to line up two or more selected elements along a specified edge. The alignment options are **Left**, **Right**, **Top**, **Bottom**, **Vertical**, and **Horizontal.** Objects are aligned relative to the first selected element. To align elements:

1. Choose the Selection tool in the **Tool** palette.
2. Shift-click to select the elements to be aligned, or drag a marquee around the elements.
3. Choose an alignment option from the **Align** submenu. The elements align relative to the first selected element.

Adjust Size

Choose an option from the **Arrange** menu's **Adjust Size** submenu to give two or more selected elements a common size. The sizing options are:

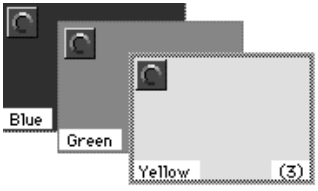
1. **Width & Height:** Both the widths and heights of elements change to the width and height of the largest selected element.
2. **Width Only:** The widths of the elements change to the width of the largest selected element.
3. **Height Only:** The heights of the elements change to the height of the largest selected element.

To give elements a common size:

1. Choose the Selection tool in the **Tool** palette.
2. Shift-click to select the elements to be resized, or drag a marquee around the elements.
3. Choose an adjustment option from the **Adjust Size** submenu.

Changing the layer order of elements

The layer order of text and graphic elements is the order in which the elements are drawn on the screen. When new elements are added to a scene, they draw on top of previous elements. Drawing elements on top of each other creates the illusion that the two-dimensional (X,Y) screen has a depth (Z) dimension. The effect is popularly called 2.5D.



The effect of the mTropolis layer order

In the Layout window, the layer order of an element is represented by a number in parentheses shown in the bottom right corner of the element. The higher the number, the closer it appears to the viewer.

Options in the **Arrange** menu provide a quick and easy way of changing the position of elements within the layer order, or moving elements to the back or front of the layer order. These options are described below.

Changing the layer order of single elements

The **Arrange** menu layer ordering options produce the following effects when applied to single elements:

- **Bring to Front** (⌘-Option-/): The selected element is moved to the top of the layer order. It will draw on top of the other elements.
- **Send to Back** (⌘-Option-\): The selected element is moved to the bottom of the layer order. The element will draw underneath all other elements.
- **Bring Forward** (⌘-/): The selected element is moved one position up in the layer order.
- **Send Backward** (⌘-\): The selected element is moved one position down in the layer order.

To move an element in the layer order:

1. Choose the Selection tool in the **Tool** palette.
2. Select the element to be moved.
3. Choose an item from the **Arrange** menu. The selected element is moved in the layer order and its layer order number is updated.

Changing the layer order of multiple elements

Applied to multiple elements, the **Arrange** menu layer ordering options produce the following effects:

- **Bring To Front** or **Send To Back:** Selected multiple elements can be moved using the Bring to Front/Send to Back commands. They retain their relative order when placed.

  When using the **Bring To Front** command, any existing gaps between the selected elements are eliminated when they are placed.

  When using the **Send to Back** command, any existing gaps between the selected elements are retained when they are placed. If there are insufficient spaces to accommodate the number of elements being moved, existing elements are automatically moved forward the appropriate number of spaces in the layer order.

  These commands are not available if all elements in a scene are selected, if a scene is selected, or if no elements have been selected.
- **Bring Forward** or **Send Backward:** Used with multiple-selected items, **Bring Forward** moves the selected items one layer forward in the layer order. The **Send Backward** command moves the selected items one layer backward in the layer order. Any existing gaps among the selected elements are eliminated when they are placed.

  If there is an element in front of or behind the elements to be moved, their positions are swapped. For example, when elements whose layer order numbers are 1, 2, 3 are moved forward one layer, the element previously in position 4 will be moved to layer number 1.

Additional notes about layer ordering

Layer order numbers begin with the scene. The layer order number of a scene is 0 and cannot be changed.

All the layer order editing options in the **Arrange** menu can be used in any of mTropolis' editing views.

The layer order can also be changed by editing the element's layer order number in the **Element Info** dialog box, by using the editing options in the Layers window, or by using the **Object Info** palette.

Each element on the shared scene also has a layer number. These elements' layers must be adjusted on the shared scene.

Elements on the active scene with identical layers to an element on the shared scene are drawn above the shared scene element.

Eliminating gaps in the layer order

The **Eliminate Gaps** (**Arrange** menu) item can be used to remove unused layers between elements in a scene.

During the process of creating elements and changing their layer orders, gaps in their numbering sequence may be created. At times these gaps may be useful. For example, an author may deliberately leave gaps between elements on a shared scene to accommodate other elements that will appear there during run-time.

On the other hand, unnecessary gaps can add to the amount of scrolling required to view elements in the Layers window. These gaps can be removed using the **Eliminate Gaps** menu item.

To eliminate all gaps in the Layout or Layers window:

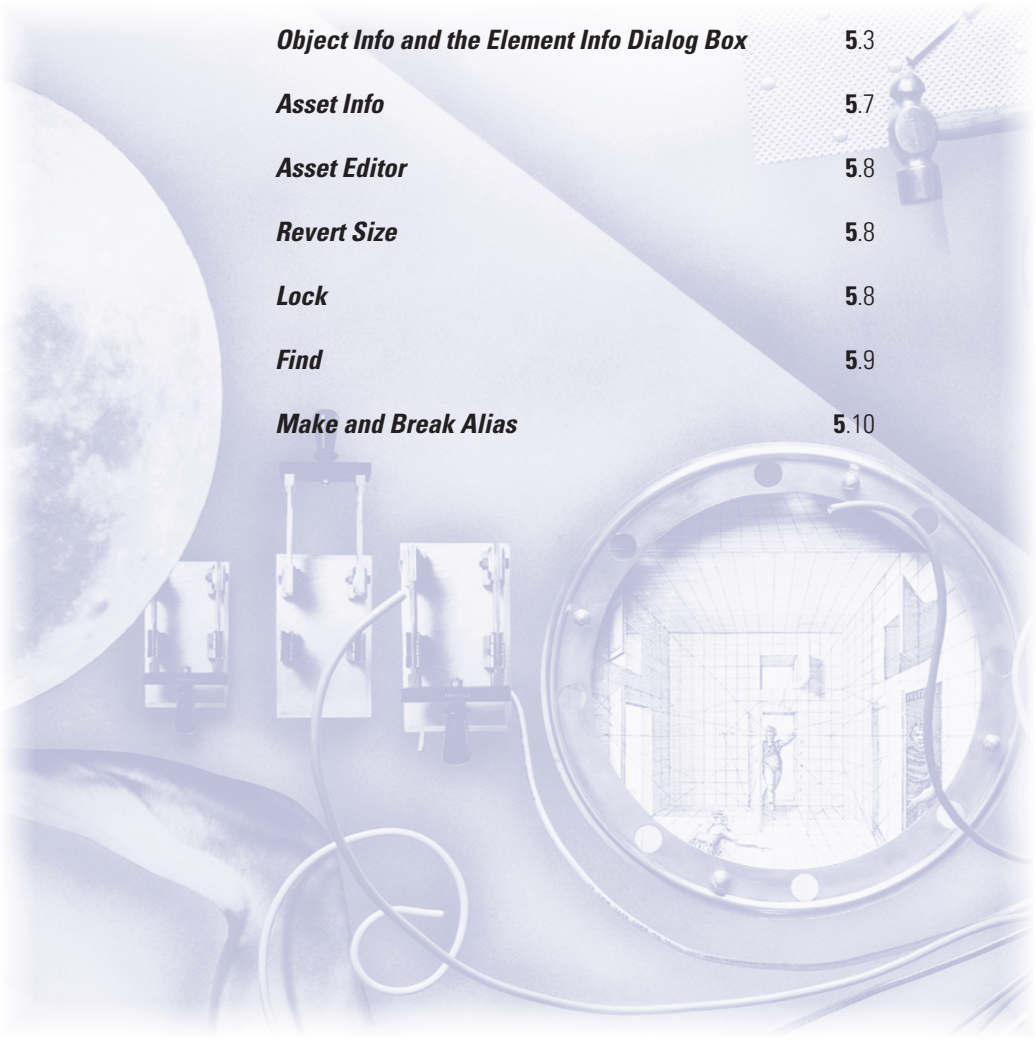
Select the scene and choose **Eliminate Gaps** from the **Arrange** menu. The gaps are eliminated and the layer order of all elements within the scene is changed.

Gaps can also be eliminated between specifically selected elements in any view.

To eliminate specific gaps:

Hold down the Shift key or marquee around the selected the elements. Choose **Eliminate Gaps** from the **Arrange** menu. The layer gaps between the selected elements are eliminated and the layer order of the elements is changed.

5. Object Menu



5. Object Menu

The **Object** menu contains options for creating and managing project components.

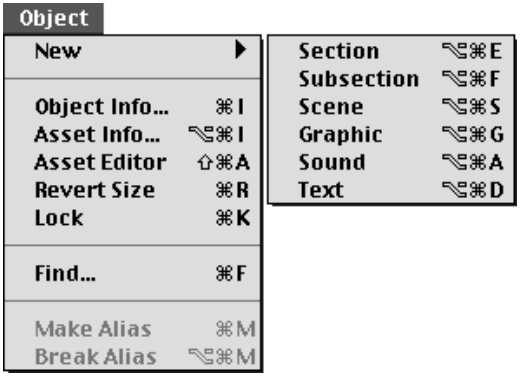
The first section of the menu contains commands for creating new project components. These options can be used in any editing view (except for **New ➛ Sound**, which can only be used in the structure view).

The second section of the menu contains commands for obtaining information about assets, editing assets, resizing elements, and locking elements.

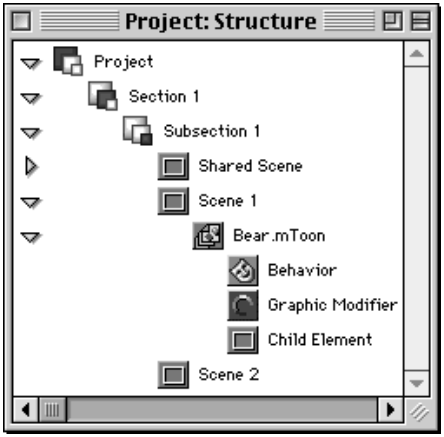
The **Find** option can be used to find project components based on many different search criteria.

The last section of the menu contains an option for creating modifiers that share the same settings (**Make Alias**) and another that removes this functionality (**Break Alias**).

Object menu options are described in detail in this chapter.



*The **Object** menu*



The Structure window, showing the components of a typical project

New — Adding Sections, Subsections, Scenes, and Elements to a Project

A mTropolis project has a structural hierarchy that begins at the project level, and descends through sections, subsections, scenes, and elements. The options in the **New** cascading menu can be used to add components to a mTropolis project.

The **New** ➛ **Section** (⌘-Option-E), **New** ➛ **Subsection** (⌘-Option-F), **New** ➛ **Scene** (⌘-Option-S), **New** ➛ **Graphic** (⌘-Option-G), **New** ➛ **Sound** (⌘-Option-A, available only when a component is selected in the Structure window) and **New** ➛ **Text** (⌘-Option-D) menu items in the **Object** menu provide a quick and convenient way to add a new component to a project while working in one of mTropolis' three editing views.

See Chapter 8, "Structure Window," Chapter 9, "Layout Window," or Chapter 10, "Layers Window" for information on using **Object** menu items to add components to a project in these views.

Object Info and the Element Info Dialog Box

Choose the **Object Info** option (⌘-I) in the **Object** menu to display a configuration dialog box for the currently selected object.

Object Info for modifiers

If a modifier is selected, the modifier's configuration dialog box is displayed. Modifier configuration dialog boxes are described for each modifier in Chapter 12, "Modifier Reference."

Object Info for structural elements

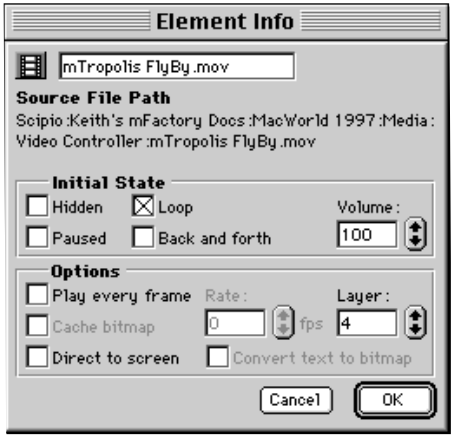
If this option is chosen when a Section object is selected, the **Section Info** dialog box appears. See "Assigning Sections to Title Segments" in Chapter 1. Project and subsection objects do not have configuration dialog boxes.

Object Info for elements

If an element is selected, the **Element Info** dialog box is displayed. This dialog box is used

to set the initial states of graphic, sound, and text elements. The **Element Info** dialog box can also be opened by double-clicking the element. See "Tool Palette" in Chapter 11 for complete information on creating elements.

Controls in the **Element Info** dialog box are described below. Note that options in the **Element Info** dialog box vary according to the type of media that has been linked to the selected element. Not all fields are available in all **Element Info** dialog boxes.



*The **Element Info** dialog box, shown for a QuickTime element*

Element Type icon
These icons identify the element's type:

A graphic element that has not been linked to media.

A graphic element that has been linked to a PICT.

A graphic element that has been linked to an mToon, the proprietary animation format in mTropolis.

A graphic element linked to a QuickTime movie.

A text element.

A sound element.

Element Name field
By default, the name of the linked external media appears in this text field. A new name for the element can be entered without changing the name of the media asset (the media file to which the element is linked).

Source File path
This area displays the location of external media that has been linked to the element.

Hidden check box
Click Hidden to initially hide an element during run-time. The element can be made visible by sending it a **Show** or **Play** command (see "Shown/Show" and "Played/Play" in Chapter 13). Hidden elements do not respond to end user mouse clicks, as described in Chapter 13.

Loop check box
By default, movies and mToons will loop (play continuously). Uncheck this option to have them play only once.

Paused check box
The **Paused** check box is specific to movies, mToons, and sounds. Click **Paused** to initially pause a video, mToon, or sound at run-time. Paused movies and mToons display their first frame, but do not continue playing until told to do so. Similarly, paused sounds do not play until told to do so. Elements can be made to unpause by sending them a **Play**, **Unpause**, or **Toggle Pause** command. See "Played/Play," "Unpaused/Unpause," and

"Toggle Pause (Message/Command Menu Only)" in Chapter 13.

Back & Forth check box
When checked, this option allows video files to play back and forth between their start and end positions. The **Loop** check box must also be checked for this option to be available.

Volume field
Set the initial volume level of a sound or QuickTime element to a percentage of the sound file's original volume. Volume levels can also be accessed by messengers and by Miniscript.

Play Every Frame check box
When checked, this option forces playback of every frame of a QuickTime movie. Usually, QuickTime movies drop frames to ensure proper synchronization between the video and audio portions. However, for movies without sound, this check box can be clicked to ensure that the movie's playback speed is adjusted to the speed of the system.

Maintain Rate check box
Unlike QuickTime movies, which may drop frames to match their specified frame rate, mToon always show all of their cels. This mToon option, unchecked by default, controls the way in which an mToon plays its cels:

- When this option is checked, the length of time that each cel in the mToon is actually displayed on screen may vary as the mToon attempts to play at the frame rate specified in the Rate field. On slower machines, this behavior may cause an mToon to play unevenly as the frames slow down when the system is busy then rapidly speed up to attempt to match the designated frame rate.
- When this option is not checked (the default), mTropolis attempts to make the length of time that each cel is displayed on screen constant. As a result, the mToon may not be able to maintain the specified frame rate, but the animation's speed will appear consistent. This behavior is useful when the display of each cel for a consistent duration is more important than the overall frame rate.

Cache Bitmap check box
Checking **Cache Bitmap** stores an element's contents in memory as a bitmap graphic, optimizing screen redraw time.

This option can be useful when ink or image effects have been applied to an image. For example, gradient effects require significant processing time to be produced. An element that contains a Gradient modifier (set to apply itself on Parent Enabled or Scene Started) with the **Cache Bitmap** check box option selected will be stored in memory — with the effect already applied — where it can be accessed as required, greatly decreasing processing time.

This option is the default for text elements. When this option is checked, text elements are rendered to a bitmap during run-time and that bitmap is used to display the text. The bitmap is updated only when the text changes. In most cases, this option results in the most versatile and fastest display of text.

However, if a text element contains more text than the element can display at one time, scrolling the text element by sending it one of the **Scroll** commands (described in Chapter 13) will not reveal more text. The cached bitmap contains an image of only the originally visible text. Unchecking the **Cache Bitmap** option causes the text element

to be refreshed each time it receives a **Scroll** command, revealing previously-hidden text.

When text is not cached, not all ink effects are supported. On Mac OS, un-cached text can only have the "Copy" ink effect applied. On Windows, un-cached text can have only the "Copy" or "Background Transparent" ink effect applied.

Note: This option is limited by the memory available to the application in run-time mode.

Direct to Screen check box
By default, elements are drawn in a layer order, creating a 2.5D effect. Checking the **Direct to Screen** option improves element redraw time by drawing an element on top of other elements in a scene. The element is no longer part of the mTropolis layer order. When set to play direct to screen, an element's layer order number is shown in square brackets in the Layout window (for example, [01] instead of the usual parentheses, (01)).

The **Direct to Screen** option is a performance optimization that fundamentally changes the way an element displays on screen. Results may not always be pleasing. For example, draggable elements displayed direct to screen may exhibit "flashing" or other updating problems when they are dragged. mToons that contain cels of different sizes may display "trails" as they play. Some ink effects may not work properly with this option enabled. Such side-effects are a result of the direct to screen behavior.

This option is most useful for movies and mToons that simply play in one place without moving and without having other elements move over or under them.

Note: By default, QuickTime movies are played direct to screen, though this option can be changed on Mac OS. On Windows, QuickTime movies always play direct to screen, regardless of the **Direct to Screen** check box setting.

Note: QuickTime VR movies are always drawn direct to screen.

Layer field
Change the layer order number of an element by entering a new number here. See "Changing the Layer Order of Elements" in Chapter 4 for more information on mTropolis layers.

Rate field
Specify the desired play rate of an mToon, in frames per second, by entering a value here. Optionally, use the arrow keys to change the value. The default value is 15 frames per second.

Convert Text to Bitmap check box
Check this box to convert a text element to a bitmap when the project is built into a title (see "Build Title" in Chapter 1). The text becomes a bitmap picture of the text and behaves exactly like a PICT element. Hence, machines that run the title do not need to have installed the fonts that are used in this text element and all embedded formatting is preserved. This option is useful for cross-platform applications where the font used in the text element is not available on all platforms.

Text converted to a bitmap cannot be changed during run-time mode. The following restrictions apply to text elements that have this option enabled:

- They do not have a "text" attribute. The contents of the element cannot be read or

written. In addition, they do not have other text-specific attributes such as "clickedLine" and "lineCount."

- They cannot be made editable. Sending them the **Enable Editing** command has no effect. They do not have an "editable" attribute.
- They cannot be used as "calculated fields" to display mTropolis variables.
- If the text element contains more text than the element can display at one time, scrolling the text element by sending it one of the **Scroll** commands will not reveal more text (see Chapter 13). The converted bitmap contains an image of only the originally visible text.

Depending on the effects applied to text elements, they may be written as either 1-bit or 8-bit bitmaps when built into a title. One-bit bitmaps take very little space to store in a title (about the same space as a regular text element). Eight-bit bitmaps use significantly more space in a title file.

The text element will be converted to a 1-bit bitmap in the following situations:

- When the text element has a Graphic modifier applied to it. A single foreground and background color are applied over the entire element.
- When the text element has a Text Style modifier applied to it.
- If no text style or Graphic modifiers are applied to the element, and the entire text element uses black as the text color.

The text element is converted to an 8-bit bitmap when the text element contains no text style or Graphic modifiers and embedded formatting has been used to change the color of any part of the text to a color other than black. See Chapter 3, "Format Menu," for more information about embedded formatting.

OK button
Click **OK** to save the settings and close the dialog box.

Cancel button
Click **Cancel** to close the dialog box without making any changes.

Asset Info

Choose **Asset Info** from the **Object** menu (⌘-Option-I) to display general information about the asset linked to the currently-selected element. The **Asset Info** dialog box appears. Components of this dialog box are described below.

Open Editor button
Click the **Open Editor** button to launch the application that created the asset. See "Asset Editor" in this chapter.

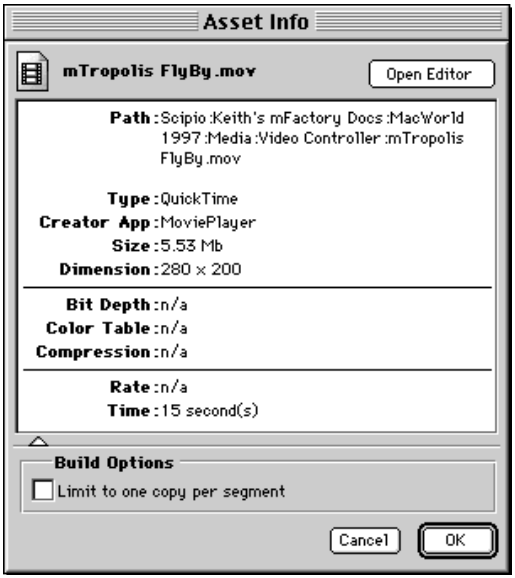
Build Options section
Click the triangle at the bottom of the **Asset Info** dialog box to display the Build Options section of the dialog box.

The **Limit to one copy per segment** check box in this section can be used to modify the way the asset is written when the project is built into a title. By default, assets are written as specified by the option selected in the **Build Title** dialog box's **Asset Order** pop-up menu (see Chapter 1). When this check box is checked, the asset is included only once in the first title segment in which it is used.

This option is useful for including a large asset only once in a built title. Note that this option takes effect only when the "Build for Speed" build title option is selected. In titles built using the "Build for Space" option, all assets are only written once, so this option becomes redundant.

Asset Editor

Choose **Asset Editor** from the **Object** menu (Shift-⌘-A) to launch the application that created the asset linked to the currently-selected element (if that application is available). Once launched, the editing application becomes the active application. The asset editor can also be launched from the **Asset Info** dialog box.



*The **Asset Info** dialog box*

Revert Size

Choose **Revert Size** from the **Object** menu (⌘-R) to resize selected elements to the original dimensions of their linked external media.

Lock

Choose **Lock** from the **Object** menu (⌘-K) to prevent a selected element from being moved with the mouse while in edit mode.

When an element is locked it cannot be resized or moved with the mouse. Media cannot be linked to an element that is locked. All other functions and capabilities of an element remain intact while it is locked. Elements can always be deleted by pressing the **Delete** key or by choosing **Clear** from the **Edit** menu, even when they are locked.

!!! Note: By default, scenes are locked when they are first created. Also, media can be linked to scenes using the **Link Media ➝ File** option, even when they are locked. However, other methods of changing media associated with the scene element (such as dragging from the **Asset Palette**) do not work when the scene is locked.

To lock an element while in edit mode:

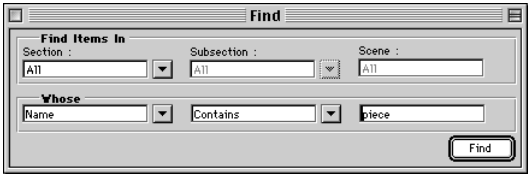
- Select an element to be locked.
- Choose **Lock** from the **Object** menu (⌘-K). The element is now locked.

To unlock an element while in edit mode:

- Select a locked element.
- Choose **Lock** from the **Object** menu (⌘-K). The element is unlocked.

Find

Choose **Find** from the **Object** menu to search for components in a project by name. The **Find** dialog box appears. The **Find** dialog box has two components: Find Item In and Whose.



*The **Find** dialog box*

Find Items In

Use the pop-up menus in this section to define the portion of the project to search. Individual sections, subsections, and scenes can be specified. The default is to search in the entire project.

Whose

Use the pop-up menus in this section to define search criteria. The leftmost pop-up menu defines the type of search to perform. Options include **Name**, **Asset**, and **Alias:**

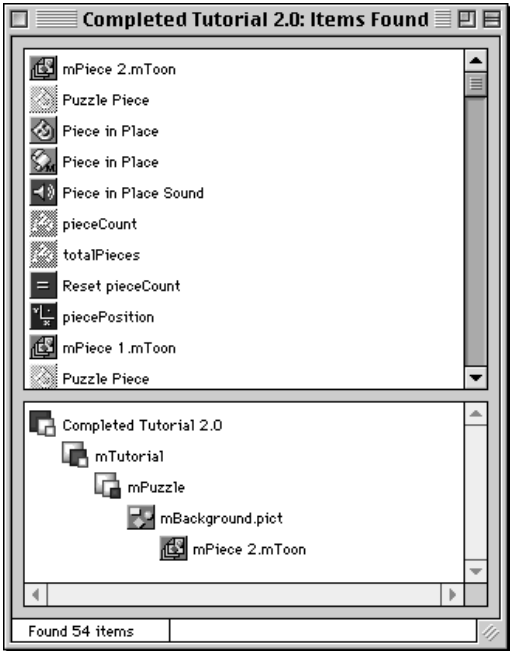
- Choose **Name** (the default) to search for a project component based on characters in the component's name. Enter the string to be searched for in the rightmost pop-up menu. Alternatively, drag an element from the project and drop it in the rightmost pop-up menu — the pop-up menu changes to reflect the dropped element's name. The middle pop-up menu can be used to choose the conditions under which the string will be found. Options include Contains (the default), Starts with, Ends with, Is, Is not, and Doesn't contain. Click the **Find** button to start the search.
- Choose **Asset** to search for project components that make use of a specified asset (including modifiers that "link" to assets, such as the Sound Effect modifier and Color Table modifier). The middle pop-up menu changes to "Is" (the only choice for this type of search). The rightmost pop-up menu changes to "Drop asset here." Drag an asset from the **Asset Palette** and drop it on the rightmost pop-up menu. Click the **Find** button to start the search.
- Choose **Alias** to search for copies of a specified alias. This option is useful because individual copies of an alias used in a project can be given unique names. Thus, it may not be obvious from the editing views which aliases are copies of the same master. The middle pop-up menu changes to "Is" (the only choice for this type of search). The rightmost pop-up menu changes to "Drop alias here." Drag an alias from the project and drop it on the rightmost pop-up menu. Click the **Find** button to start the search.

Find Button

Click **Find** to start the search. If items are found, they appear in the Items Found window.

Items Found Window

The Items Found window displays items located by a find operation. The top portion of the window shows a list of found items and their icons in alphabetical order. Choose an item to see its location in the lower portion of the Items Found window.



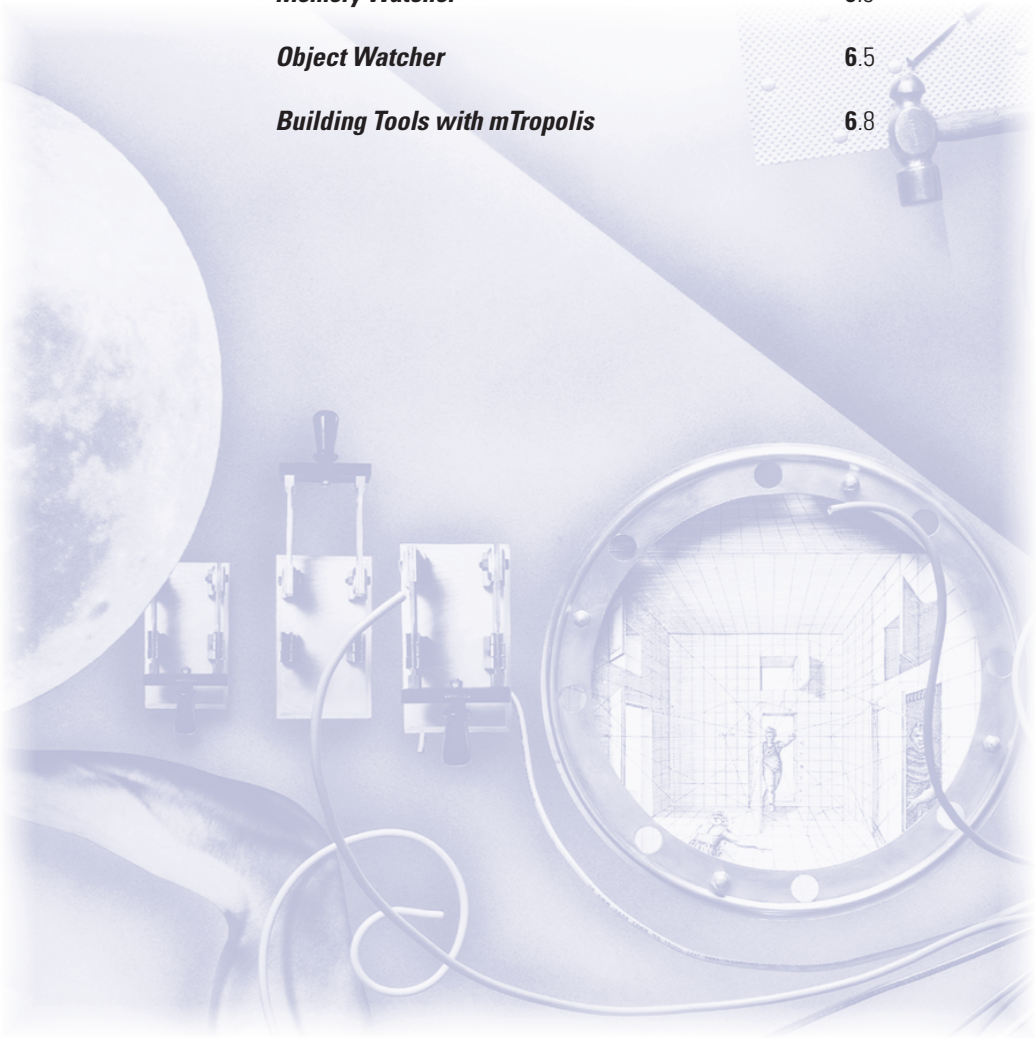
The Items Found window

The lower portion of the Items Found window shows the path of the found item selected in the top portion. Double-click the found item to make all open windows in the project "sync" to its location. The item will be chosen in the project. If the found item exists in a palette (such as the **Alias** or **Asset** palette), the palette is displayed and scrolled to the location of the found item.

Make and Break Alias

The **Make Alias** option in the **Object** menu (⌘-M) turns a selected modifier into an alias. An alias is a productivity tool that is used to more easily manage modifiers with identical settings. The **Break Alias** option breaks the relationship between an instance of an alias and the master copy of the alias. Aliases reside on the **Alias** palette, which can be viewed by selecting **Alias** palette from the **View** menu. See "Alias Palette" in Chapter 11 for a complete description of aliases, the **Alias** palette, and the **Make Alias** and **Break Alias** options.

6. Tools Menu



6. Tools Menu

The **Tools** menu provides access to additional mTropolis editor functions built with MOM (in a language such as C or C++) or with mTropolis itself. Options in the **Tools** menu represent mTropolis components that are not part of the "built-in" editor functionality.

Quark provides three tools with mTropolis that are installed by default, the mPack Guide, the Memory Watcher, and the Object Watcher. New tools can be added by end users or by third parties.

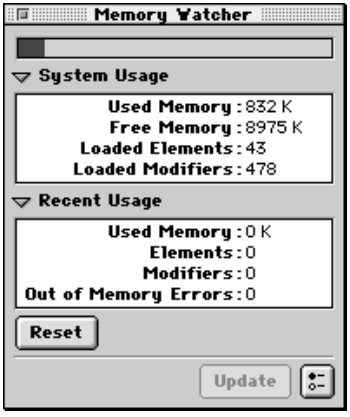
The first part of this chapter describes the tools that are installed with mTropolis. The second part describes some of the features that are useful when building new tools using mTropolis. Tools can also be created with the mFactory Object Model (MOM). See the *MOM Reference Guide* (found on the mTropolis CD–ROM) for details.

mPack™ Guide

Choose **mPack Guide** from the **Tools** menu to display the mPack Guide tool. This tool contains interactive documentation for all of the mPacks included with mTropolis. mPack documentation is also included as an "About" component inside each mPack. See Chapter 9, "mPacks" of the *mTropolis Developer Guide* for more information about mPacks and mPack documentation.

Memory Watcher

Choose **Memory Watcher** from the **Tools** menu to display the Memory Watcher tool. This tool can be used to see how much memory all the open mTropolis projects are using. The Memory Watcher is a floating palette that is active both in edit mode and in runtime mode.



The Memory Watcher tool

Components of the Memory Watcher tool are described below.

Memory Usage Graph

The bar indicator at the top of the Memory Watcher window gives a graphical view of how much memory is available in mTropolis' memory partition. The dark part of the bar indicates the amount of memory currently used, the light part of the bar indicates remaining free memory.

System Usage Display

The System Usage display can be alternately shown and hidden by clicking the "System Usage" turn-down triangle. Fields shown in this display are:

- **Used Memory:** The amount of memory currently used (in Kilobytes).
- **Free Memory:** The amount of memory (in Kilobytes) free in the mTropolis memory partition.

!!! Note: The sum of the used and free memory values will be less than the memory partition set in the mTropolis **Get Info** dialog box (set in the Mac OS Finder). The Memory Watcher measures the memory available to projects, which does not include the memory consumed by the mTropolis application itself.

- **Loaded Elements:** The total number of media elements currently loaded.
- **Loaded Modifiers:** The total number of modifiers currently loaded.

Recent Usage Display

The Recent Usage display can be alternately shown and hidden by clicking the "Recent Usage" turn-down triangle. This display is like the trip odometer in a car — it displays the amount of change to various parameters since it was last reset. Click the Memory Watcher's **Reset** button to cause all of these fields to be reset to zero. As projects are open and run, the numbers update to show how much additional memory has been used since the last reset.

Note: To accurately measure how much memory is being used by a project, the **Reset** button must be clicked before opening the project to be measured. This is necessary because elements and modifiers are loaded as soon as the project is opened, not just when it is run. Objects are also loaded whenever the end user opens a scene in the structure view. In fact, the element and modifier counts will often shrink when the project is run because all objects not on the current scene and shared scene will be unloaded.

Fields shown in this display are:

- **Used Memory:** The change in the amount of memory (in Kilobytes) used since the last reset.
- **Elements:** The change in the number of objects loaded since the last reset.
- **Modifiers:** The change in the number of modifiers loaded since the last reset.
- **Out of Memory Errors:** The number of memory errors that have occurred since the last reset. This count increases each time there is not enough memory available for mTropolis to satisfy a particular request. Such errors are not always fatal. When out of memory errors occur, mTropolis frees memory by unloading unnecessary assets or elements (if possible). In such cases no alert dialog box appears, but the project's performance may suffer if it runs out of memory frequently.

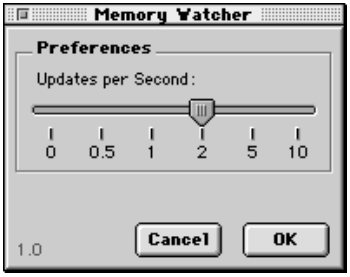
Reset button

Click this button to reset the fields in the Recent Usage display to zero.

Update button

Click this button to force the Memory Watcher tool's display fields to update. This button is only active if the tool's "Updates per Second" preference has been set to 0.

Preferences button
Click the icon button in the lower-right corner of the Memory Watcher tool to display the tool's Preferences section.



The Memory Watcher Preferences display

The Memory Watcher has just one preference:

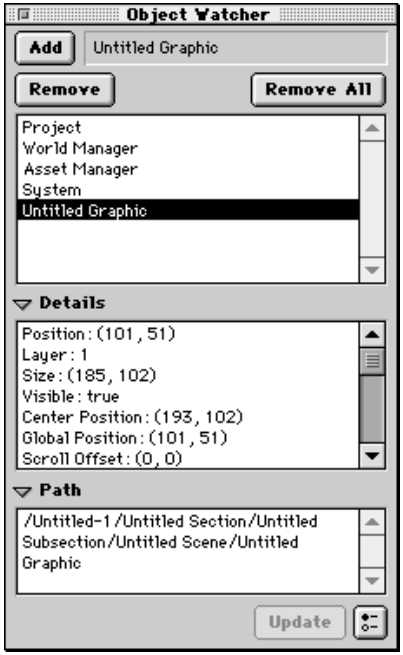
- **Updates per Second:** Use this slider to choose how many times per second the Memory Watcher updates its displays. Choices are [0, 0.5, (that is, one Update every 2 seconds) 1, 2, 5, and 10]. Selecting 0 causes the displays to update only when the tool's **Update** button is clicked.

Click **OK** to accept the new preferences. Click **Cancel** to dismiss the preferences section without any changes.

Note: When closed, the Memory Watcher stores its preferences [in the Mac OS Preferences folder] in a file named "mTropolis Mem Watcher Prefs."

Object Watcher

Choose **Object Watcher** from the **Tools** menu to display the Object Watcher tool. This tool can be used to see the current attributes of an object. This tool is particularly useful for watching the contents of variable modifiers, but the basic attributes of any mTropolis object, including structural components, can be watched as well.



The Object Watcher tool

Components of the Object Watcher tool are described below.

Add button and Name field
Click this button to add the currently selected object(s) to the object list. Once added to the object list, the name of the object can be clicked to show information about that object in the Object Watcher's Details and Path fields. Note that mTropolis must be in edit mode and there must be an object selected (in any of the mTropolis editing views) for the **Add** button to be active.

The name of the currently selected object (or "All Selected Objects," if multiple objects are selected) is displayed in the text field to the right of the **Add** button. If nothing is selected, this field is blank and the **Add** button is disabled.

Run-time Object Watching
During run-time mode, any object already in the object list can be watched by clicking on its name in the list. Because objects can be created during run-time (using the **Clone** command or clone attribute), you may want to watch an object that does not yet exist in edit mode. The Object Watcher mPack contains behaviors that can be used to communicate with the Object Watcher during run-time and add new objects to the object list. This is useful if the project clones elements and you want to watch one of the cloned elements. See Chapter 9, "mPacks" of the *mTropolis Developer Guide* for more information on mPacks.

Remove button
Click this button to remove the object currently selected in the object list.

Remove All button
Click this button to remove all objects from the object list. Note that the structural components "Project," "World Manager," "Asset Manager," and "System" always show up in this list unless the Object Watcher preferences have been changed so that "invisible components" are not shown. See "Object Watcher Preferences" in this chapter.

Object List
This section of the Object Watcher shows all of the objects that have been added by the end user. It also shows four components by default — the Project, World Manager, Asset Manager, and System components.

For most components, only the object's name appears in this list. However, the value contained by simple variables is shown directly in the list.

To see the value of complex variables or the attributes of an element, click the name of the desired object to highlight it. When selected, more details about that object appear in the tool's Details and Path displays.

Double-click the name of an object in this list to open that object's configuration dialog box (if it has one). This feature only works when the project is in edit mode.

If an object in the list is moved or deleted, its name in the list will change to "[moved or deleted object]". Click this name and then click **Remove** to delete it from the list.

Details display
The Details display can be alternately shown and hidden by clicking the "Details" turndown triangle. This display shows further information about the object currently selected in the Object List. The contents of this display vary depending on the type of object selected:

- Simple variables show their values (also shown in the object list).
- Compound Variables show the names and values of each of their fields.
- List variables show the value of each item in the list.
- Elements show the values of their position, layer, size, visible, centerPosition, and globalPosition attributes. In addition, media-specific attributes such as cel and range (for mToons) are displayed depending on the type of media linked to the element.
- Modifiers other than variables show the value of their ClassID attribute. This string reflects the modifier's general type (either the general name of the modifier or "InternalMod" for many built-in modifiers).

Path display

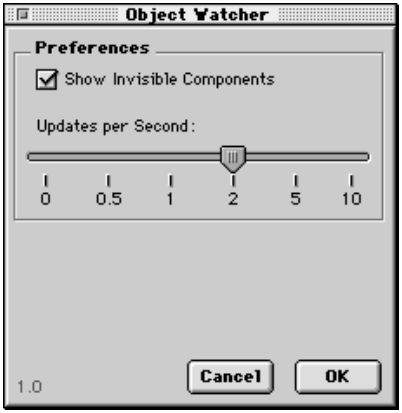
The Path display can be alternately shown and hidden by clicking the Path turn-down triangle. This display shows the path to the currently selected object (that is, its position in the project's structural hierarchy). This path is displayed in the same path syntax used in the object reference variable. See "Object Path Field" in Chapter 12. The Path display is useful for differentiating between multiple objects with the same names.

Update button

Click this button to force the Object Watcher tool's displays to update. This button is active only if the tool's "Updates per Second" preference has been set to 0.

Preferences button

Click the icon button in the lower-right corner of the Object Watcher tool to display the tool's preferences section.



The Object Watcher Preferences display

The Object Watcher has the following preferences:

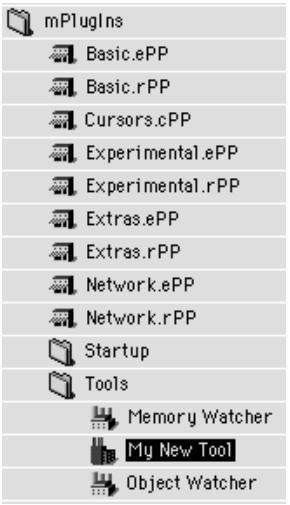
- **Show Invisible Components:** Check this check box (the default) to display four of the usually invisible components (the Project, World Manager, Asset Manager, and System components) in the Object List. When this check box is unchecked, these four components are not shown in the list.
- **Updates per Second:** Use this slider to choose how many times per second the Object Watcher updates its displays. Choices are [0, 0.5, (that is, one Update every 2 seconds) 1, 2, 5, and 10]. Selecting 0 causes the displays to update only when the tool's **Update** button is clicked.

Click **OK** to accept the new preferences. Click **Cancel** to dismiss the preferences section without any changes.

Note: When closed, the Object Watcher stores its preferences in a file in the (Mac OS Preferences folder named) "mTropolis Obj Watcher Prefs".

Building Tools with mTropolis

End users can build their own tools and add them to mTropolis tools menu. This section describes the steps necessary to add a tool to mTropolis and features that are useful in the creation of new mTropolis tools.



Using the Finder to add a new tool to the Tools subfolder of the mPlugins folder

Creating tools and adding them to the Tools Menu
Take the following steps to create a new tool with mTropolis:

1. Create a new project and build your new tool in edit mode. Your tool will probably use many of the features described below in "mTropolis Features Relating to Tools." At the very least, the tool should use the Window Prefs modifier to run inside of a window (such as a floating palette). Use run-time mode to test and debug your tool.
2. Save the finished tool as a mTropolis project file (**File ➛ Save** or **File ➛ Save As**), or as a built title (using **File ➛ Build Title**).
3. In the Finder, copy or move the tool's project file or built title file to the **Tools** folder, found inside the mTropolis editor's **mPlugins** folder (mPlugins is found in the same folder as mTropolis).
4. The next time mTropolis is launched, your new tool will be available from the **Tools** menu, just like the Memory Watcher and Object Watcher tools. Choose the name of your tool from the menu to open the tool.

*A user-created tool added to the **Tools** menu*

The tool opens and runs. Note that the mTropolis editor remains active and visible, but the tool itself is in run-time mode and behaves much like a regular built title.

Note: Tools opened in this way automatically open in run-time mode. So a tool saved as a project file will run just the same as a tool saved as a built title. Tools saved as projects will tend to have smaller file sizes than tools saved as built titles, because any media used by the tool is not saved with the title file — the tool gets any media used through links. However, tools saved as built titles are easier to send to other end users, since the media is bundled inside the built title file. When installing "project" tools in other copies of mTropolis on other machines, you will also need to copy any media files used by that project. This may also cause the **Re-link Media** dialog box to appear if the media files cannot be automatically found by mTropolis. Note also that "project" tools can be opened in the mTropolis editor, allowing others to see your source code and/or modify your tools. Using tools saved as built title files avoids these problems.

mTropolis features relating to tools

The following features are useful in the creation of mTropolis tools:

Window Prefs modifier

This modifier allows mTropolis projects to be run inside of a window, instead of taking over the entire screen. This feature is essential for the creation of useful tools. A tool that runs full-screen is of limited use, because you cannot simultaneously access other mTropolis editor windows. The Window Prefs modifier is described in Chapter 12.

This modifier must be applied directly on the project component. Use the structure view (choose **View ➛ Structure Window** or press ⌘-2) to drop the Window Prefs modifier onto the project (the topmost icon in the structure view).

It is often useful to set the project's Draw Area preferences (choose **Edit ➛ Preferences ➛ Project**) the same as the "Window Size" settings in the Window Prefs modifier. When the Draw Area preferences are changed, mTropolis will offer to automatically resize the scenes in your project. The resized scenes (when viewed in the Layout window) will be the same size as the visible area of the tool.

mPacks

A number of the mPack libraries included with mTropolis contain behaviors and objects that are useful in the creation of tools. The "Buttons" mPack contains pre-built buttons that emulate the function and appearance of standard Mac OS buttons. The "Scrollbars & Sliders" mPack contains elements that emulate the behavior of standard Mac OS scrollbars and slider.

Attributes

Tools are often used to read or write the attributes of the objects in another project. Tools can access any of an object's attributes, but the following attributes are especially useful in the creation of tools:

- **activeProject:** Tools are commonly used to read and write the attributes of other projects that are open for editing with mTropolis. However, because mTropolis projects might have any name and could be opened in any order, it would be difficult to distinguish between the various projects that might be open.

This system attribute, described in Chapter 15, is useful because it contains an object reference to the project component of the currently active project. Usually, this project is the one currently being edited. For example, a project becomes the active project when the end user chooses an object in one of that project's editing views (in edit mode) or when a project is being manipulated in run-time mode (such as when buttons are being clicked, or text fields are being edited).

As an example, a tool could be programmed to close the currently active project by setting that project's close attribute to true. The tool can recognize the currently active project by referencing the `system.activeProject` attribute. So to close that project, have the tool execute the following Miniscript statement:

```
set system.activeProject.close to true
```

- **asset:** The asset attribute, described in Chapter 15, can be used to link media to elements of a project open in edit mode (just like choosing the **File ➛ Link Media** menu option). See "Using the Asset Attribute to Dynamically Link Media" in Chapter 15 for details.
- **"Attribute" attributes:** These attributes, listed in Chapter 15, can be used to retrieve the names and values of an object's attributes. They are useful for creating tools like the Object Watcher, which display the current settings of an object's attributes.
- **clone:** This attribute, described in Chapter 15, can be used to create new instances of mTropolis objects. Setting an object's clone attribute to true creates a new instance of that object. The clone is created as the "next" object of the original.

The clone attribute can also be used to clone an object and put that new clone in a specific location. This feature is often used by wizard-type tools. For example, a wizard might customize one of its own scenes in response to end user input, then copy that scene into a different project, "automating" the authoring process.

For example, suppose that a tool has a scene named "wizardScene" that is customized by the tool. Once customized, the tool could create a copy of that scene in another project (such as, the activeProject or some other project created by the tool). When the following Miniscript statement is executed, "wizardScene" will be cloned into the currently active project as the last scene in that project's currently active subsection:

```
set wizardScene.clone to \
  system.activeProject.worldManager.\
  currentScene.parent
```

- **close:** This project attribute, described in Chapter 15, can be used to close projects that are open for editing in mTropolis.
- **kill:** The kill attribute, described in Chapter 15, can be used to delete objects in a mTropolis project. The effect of killing an object in edit mode is similar to selecting **Edit ➛ Clear.**
- **newProject:** This system attribute, described in Chapter 15, can be used by tools to create new projects. Setting this attribute to true creates a new project, just as if the end user had selected the **File ➛ New ➛ Project** menu option.
- **openEditDialog:** This attribute of elements and modifiers, described in Chapter 15, can be used to open an object's configuration dialog box, just as if the object had been double-clicked by the end user.

- **playerEmulation:** This project attribute, described in Chapter 15, can be used to toggle the state of a project's Player Emulation setting, just like the **File ➛ Run ➛ Player Emulation** menu toggle.
- **project[*n*]:** This System attribute, described in Chapter 15, contains object references to all currently open projects. For example, the first opened project could be accessed by a tool with the expression `system.project[1]`.
- **projectCount:** This System attribute, described in Chapter 15, contains the number of currently open projects.
- **running** and **runninginWindow:** These attributes, described in Chapter 15, can be written to make a project start or stop running, or determine whether a project is currently in run-time mode.
- **save** and **saveAs:** These project attributes, described in Chapter 15, can be used to save or save and rename a project, just like choosing the **File ➛ Save** or **File ➛ Save As** menu options.
- **selection[*n*]:** This project attribute, described in Chapter 15, contains object references to all the currently selected objects. For example, a tool could access the first selected object in the active project with the expression `system.activeProject.selection[1]`, and the second selected object with the expression `system.activeProject.selection[2]`.

  Note that if Shift-click is used to select multiple items, their order in the selection list will be the order in which they are clicked, not their order in the structure view.
- **selectionCount:** This project attribute, described in Chapter 15, contains the number of currently selected objects.
- **Targeting attributes:** These attributes, shown in Chapter 15, can be used to build object reference expressions, without having to know the names of the objects themselves. They are useful for getting information about a project's components or for "traversing" the project's structural hierarchy. For example, the following Miniscript statement reads the number of modifiers the currently selected object contains and stores that number in the Integer Variable "myInt":

```
set myInt to system.activeProject.\
selected [1].childModCount
```

7. View Menu

7. View Menu

The **View** menu contains options for configuring the view of the project in edit mode.

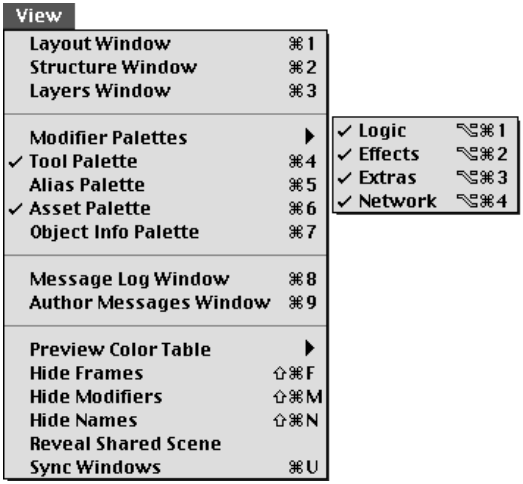
mTropolis has three editing views: the Layout window, the Structure window, and the Layers window. The options in the first part of the menu allow switching between these three views. Although editing views are discussed briefly in this chapter, they are described in detail in the following chapters:

- Chapter 8, “Structure Window”
- Chapter 9, “Layout Window”
- Chapter 10, “Layers Window”

The palettes listed in the second part of the menu contain tools for editing elements and modifiers. They are described in Chapter 11, “Palette Reference.” The modifier palettes are described in Chapter 12, “Modifier Reference.”

The Message Log window can be accessed from the **View** menu. This window displays the messages and commands that have been sent and received between components during run-time.

The **View** menu also contains options for altering the view of the current scenes in the Layout and Layers windows.



*The **View** menu*

Layout Window, Structure Window, and Layers Window — Selecting an Editing View

The Layout, Structure, and Layers windows in mTropolis provide three different ways of viewing and editing a project.

- The Layout window provides a WYSIWYG (what you see is what you get) view of the project on a per scene basis. It is the best view for laying out graphic and text elements on the screen. See Chapter 9, "Layout Window," for more information on this view.
- The Structure window is used primarily for altering the structure of the project. New sections, subsections, scenes, elements, or modifiers can be added to the structure, deleted, or rearranged. Some authors prefer to begin a project by building its structure in the Structure Window. See Chapter 8, "Structure Window," for more information on this view.
- The Layers window can be used to edit the elements and scenes in a subsection. The order of a subsection's scenes can be changed and the layer order of each scene's elements can be edited. New elements can also be added and configured with modifiers in this view. See Chapter 10, "Layers Window," for more information on this view.

To open the view windows:

- Choose **Layout Window** from the **View** menu (⌘-1).
- Choose **Structure Window** from the **View** menu (⌘-2).
- Choose **Layers Window** from the **View** menu (⌘-3).

Any of these windows can be closed by clicking the close box in the window's title bar.

!!! Note: The structure and layer views may be closed, but closing the layout view closes the project. The Layout window shortcut (⌘-1) only works if the window is open but concealed.

Working with the three views

All three views can be on the screen at once, but only one view can be active at a time. The currently active view is highlighted. Click a view window to make it active.

Copying and pasting between projects

Items in a view in one project can be copied and pasted into any view in another project. For example, an entire scene could be copied from the layout view in one project and pasted into the structure view in a second project. Similarly, an element and its modifiers could be copied from the layers view of one project and pasted into the layers view of a second project.

To copy components from one project to another:
Open both projects.

1. Choose a view in the first project.
2. Click the item to be copied, or select multiple items by using Shift-click or by dragging the cursor for a marquee selection.
3. Choose **Copy** from the **Edit** menu (⌘-C).
4. Choose a view in the second project.
5. Select a destination in the second project's view.
6. Choose **Paste** from the **Edit** menu (⌘-V) while the second project is still selected. The item appears in the second project.

Note: With the exception of the project component, any component can also be copied into another project by simply dragging and dropping it onto a destination in the new project.

Displaying Palettes

Palettes can be opened by selecting their corresponding menu options from the **View** menu or by pressing the command keys shown below:

- **Modifier Palettes ➛ Logic** (⌘-Option-1).
- **Modifier Palettes ➛ Effects** (⌘-Option-2).
- **Modifier Palettes ➛ Extras** (⌘-Option-3).
- **Modifier Palettes ➛ Network** (⌘-Option-4).
- **Tool Palette** (⌘-4).
- **Alias Palette** (⌘-5).
- **Asset Palette** (⌘-6).
- **Object Info Palette** (⌘-7).

For a complete description of these palettes, see Chapter 11.

Message Log Window

Choose **Message Log Window** from the **View** menu (⌘-8) to display the Message Log window, which shows the messages and commands that have been sent and received between components during run-time. This window is useful for debugging mTropolis projects.

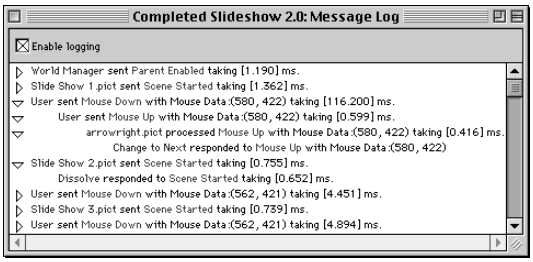
In addition to displaying the messages passed between components, the Message Log also displays any error messages that may be generated during run-time.

Check the window's **Enable Logging** check box to display information about the messages generated during run-time mode upon returning to edit mode.

The Message Log window also acts as an error log. Even if logging is not enabled, this window is automatically displayed when an error occurs during run-time. The text of the error message is displayed in the Message Log window. Individual error messages are described in "Error Messages" in this chapter.

Selecting a message displayed in the Message Log causes mTropolis to synchronize any currently-open editing windows with the message log and makes the "sender" or "recipient" shown in that line the current selection. Double-clicking a message displayed in the Message Log opens the "sender" or "recipient" object's configuration dialog box.

The Message Log displayed below shows those components that have received and responded to a number of mouse messages.



*The **Message Log** window*

Using the Message Log

Components of the Message Log window are described below.

Enable Logging check box

Check **Enable Logging** to display the path of messages and commands that have been passed to selected components during run-time execution. Then use the Structure window (⌘-2) to choose the components to be monitored. See "Selecting Messages and Commands to be Displayed by the Message Log Window" later in this chapter.

Sender

The sender of a message or command is displayed in red. Usually, this is the name of a messenger that sent a message.

End user mouse actions that cause messages to be generated (such as Mouse Up) are shown as being sent by a component called "User."

When Player Emulation is enabled (the **File ➛ Run ➛ Player Emulation** menu option is checked) any object that has been created by the **Clone** command, deleted with the **Kill** command, or reparented (by changing the value of its "parent" attribute) is shown with the name "<Moved or deleted object>." (See Chapter 13.) When Player Emulation is on, scenes are unloaded from memory upon returning to edit mode, causing the names of cloned or killed objects to be lost. Turning off Player Emulation prevents scenes from automatically being unloaded and preserves the names of cloned or killed objects. However, keep in mind that when Player Emulation is turned off, changes made during run-time are persistent — cloned objects will be present and killed objects will be permanently deleted upon returning to edit mode.

Message/Command sent or received

Messages and commands that are passed between objects are displayed in green.

"With" data

The Message Log shows the value of any data sent with a message or command. The value is shown in standard Miniscript syntax. See "Literal Values" in Chapter 14.

Recipient

Recipients of messages or commands are displayed in purple. The log shows the name of the component that received the message.

When Player Emulation is enabled (the **File ➛ Run ➛ Player Emulation** menu option is checked) any object that has been created by the **Clone** command, deleted with the **Kill** command, or reparented (by changing the value of its "parent" attribute) is shown with the name "<Moved or deleted object>" as described previously under "Sender." (See Chapter 13 for more information.)

New thread
Each new thread of messages is indicated by a left-aligned message. Lines of text "cascade" to the right, as messages and commands are passed to each new object in the project.

Open and close triangles
Click these triangles to show and hide the message path generated by each sender and recipient. Option-click a triangle to open or close it and all other triangles it contains.

Aligned messages
Recipients that share the same parent are vertically aligned.

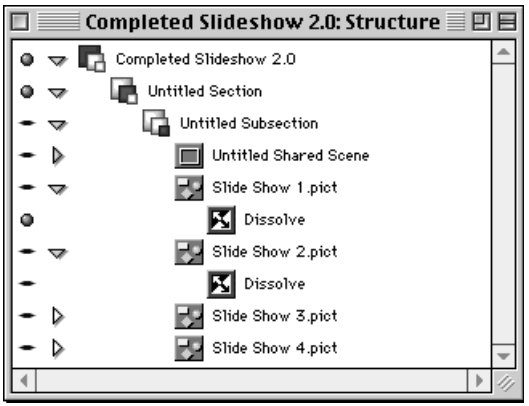
Time required to send or process message
The time it takes for each message to execute is also displayed in the Message Log window. Total execution time is displayed for each new message thread. A time is also displayed for each point in the message path. This time shows how long it takes the receiver to process or respond to the message. The total of these times is roughly equal to the total execution time. Note that the actual numbers rarely add exactly — a certain amount of processor overhead results in small discrepancies.

Time display is useful for message execution optimization. It is possible to track how long a message is taking to execute, or to analyze specific points in the message path for optimal performance. Altering the message options for a given messenger (see "Message Options" in Chapter 13) may result in faster execution times.

Choosing messages and commands to be displayed by the Message Log window
To enable the display of messages passed between components in the project:

- Ensure that the Message Log window is displayed (choose **View ➛ Message Log** or ⌘-8).
- Check the Message Log window's **Enable Logging** check box.
- Ensure that the Structure window is visible (choose **View ➛ Structure window** or ⌘-2). Note the buttons that appear to the far left of each component shown in the Structure window. By default, these buttons are deselected and look like flattened circles. Click a button to toggle message logging for the component that button is aligned with. The button expands into a "ball" to indicate that logging is enabled.

  To enable (or disable) logging for a component and all of its children (all components lower in the hierarchy), Option-click its logging button.
- Switch to run-time mode (⌘-T or ⌘-Y), test the project, and then return to edit mode (⌘-T).
- After run-time execution, the messages and commands that have been passed to the selected components appear in the Message Log window.
- Click a message displayed in the Message Log to synchronize any currently-open editing windows with the Message Log and choose the "sender" or "recipient" shown in that line. Double-click a message displayed in the Message Log to open the "sender" or "recipient" object's Configuration dialog box (if it has one).



Choosing messages to be logged. In this example, the first Dissolve modifier has its logging button enabled while the second one does not. Logging is also enabled for the project and "Untitled Section" components

Tip: When a modifier that has logging enabled is made into an alias (⌘-M), logging will always be enabled for that modifier and all of its copies.

Error Messages

The Message Log window displays error messages generated by mTropolis. Even if logging is not enabled, the Message Log window is automatically displayed (upon returning to edit mode) when an error occurs during run-time.

Short descriptions of each error message can be found below, in alphabetical order.

Note: Custom error or notification messages can be sent to the Message Log window using the debug variable. See "The Debug Variable" in Chapter 14.

Auto screen fade: bit depths higher than 256 colors currently not supported.
The WorldManager.autoScreenFade attribute (see Chapter 15) has been set to true in a project that supports thousands or millions of colors (as set in the **Edit ➛ Preferences ➛ Project** dialog box). This attribute only works with 256 color (8-bit) projects. As a result, automatic screen fades will not appear.

Bad target for send message
Messages can only be sent to structural elements (subsections, scenes), graphical elements, and modifiers. Messages cannot be sent to non-structural components such as the WorldManager, AssetManager, and System. Check the message's target destination.

Can't add
An addition operation was attempted on data types that cannot be added. Usually, the data type of operands must be the same for them to be added. Floating-point values can be added to integer values and vice-versa — mTropolis automatically handles the data type conversion in this case. To "add" (append) string values to one another, use the concatenation operator (&), described in Chapter 14.

Can't compare these
A relational operation was attempted on data types that can't be compared. Note that the only relational operators that work with strings are the = and ≠ (typed as Option-= or <>) operators — strings cannot be compared with the less than (<) or greater than (>) operators.

Can't divide
A division operation was attempted on data types that cannot be divided.

Can't do boolean operation
A boolean operation was attempted on inappropriate data types. Not all data types have boolean equivalents. See "Definition of True" in Chapter 14.

Can't find element
The referenced element cannot be found. Check if the element has been renamed, moved, or deleted.

Can't find mod
The referenced modifier cannot be found. Check if the modifier has been renamed, moved, or deleted.

Can't get attribute of component
An attempt was made to read an attribute that does not exist or apply to the specified component. See "Lists of Attributes by Component Type" in Chapter 15 for tables of valid attributes.

Can't make datatype — data conversion error messages
These error messages are displayed when a variable or literal value can't be converted to the desired data type. The various data conversion Miniscript functions are described in "Miniscript Functions" in Chapter 14. Specific error messages are:

Can't make boolean
Can't make floating-point
Can't make integer
Can't make list
Can't make point
Can't make range
Can't make string
Can't make vector

Can't multiply
A multiplication operation was attempted on data types that cannot be multiplied.

Can't negate
A negation operation was attempted on data types that cannot be negated.

Can't set attribute of component
An attempt was made to set an attribute that does not exist or apply to the specified component. See "Lists of Attributes by Component Type" in Chapter 15 for tables of valid attributes.

Can't subtract
A subtraction operation was attempted on data types that cannot be subtracted. Usually, the data type of operands must be the same for them to be subtracted. Floating-point values can be subtracted from integer values and vice-versa — mTropolis automatically handles the data type conversion in this case.

Change parent: Not allowed for scenes, subsections, sections, or project
An attempt was made to set the "parent" attribute of a structural element. While most components have a "parent" attribute, the parent attribute for structural elements (such as scenes, subsections, sections, and projects) is read-only. It cannot be set.

Changing scene: Pending clone ignored
Objects in a scene can only be cloned while that scene is active. If the scene is changed before a previously-sent **Clone** command can be acted on (before mTropolis actually constructs the new clone), this error message is generated. Once the scene is exited, all **Clone**

commands that have not yet been processed are ignored.

Changing scene: Pending kill ignored
Objects in a scene can only be killed while the scene is active. If the scene is changed before a previously-sent **Kill** command can be acted on (before mTropolis actually deletes the object), this error message is generated. When the scene is exited, all **Kill** commands that have not yet been processed are ignored.

Cloning or killing: Not allowed for scenes, subsections, sections, or project
A **Clone** or **Kill** command was sent to a structural element such as a scene. The **Clone** and **Kill** commands only operate on objects within a scene.

Function error
A generic function error has occurred. Check that function syntax and data types of any parameters are correct. See "Miniscript Functions" in Chapter 14.

Index out of range
An attempt was made to reference a list item that does not exist. A List Variable can only be subscripted with a value between 1 and the total number of elements in the list. Note that the total number of elements in a list is stored in the list variable's "count" attribute. See Chapter 12 for more information.

Message thread too deep
The message thread queue is overloaded. To avoid overloading the queue, turn off the "Immediate" check box in the "Message Options" section of the messenger modifier's configuration dialog box. When this check box is disabled, the messenger will only send messages once the current thread has ended.

No code
A Miniscript modifier was executed, but it contains no written script.

Project link failed — cannot link from an unsaved project
An Open Project modifier (see Chapter 12) was triggered before the currently-open project was saved. Save the project and try running it again.

Project link failed — cannot link to an unsaved project
An Open Project modifier (Chapter 12) was triggered before the project that it tried to open was saved. Save the referenced project and try running the current project again.

Project link failed — cannot link to itself
An Open Project modifier (Chapter 12) was triggered and that modifier was configured to open the currently-open project. A project cannot open itself.

Project link failed — unable to open file
An Open Project modifier (Chapter 12) was triggered, but failed to open the specified project. Check to see that the specified project exists.

Set current shared scene: incorrect object type
An attempt was made to set the WorldManager.sharedScene with an object other than a scene.

Set current shared scene: can't set to nonstructural value
An attempt was made to set the WorldManager.sharedScene attribute with a nonstructural object (one of the special mTropolis components — AssetManager, System, or WorldManager — that is not displayed in the structure view).

Set current scene: can't set to nonstructural value
An attempt was made to set the WorldManager.currentScene attribute (Chapter 15) to a non-structural object (one of the special mTropolis components — AssetManager, System, or WorldManager — that is not displayed in the structure view).

Set current scene: incorrect object type
An attempt was made to set the WorldManager.currentScene attribute (Chapter 15) to an object other than another scene.

Set parent attribute: cannot be a child of target parent
An attempt was made to change a component's "parent" attribute to an invalid target. For example, an element cannot be made a child of one of its current children ("set element.parent to element.mychild"). Similarly, an element cannot be made a child of itself ("set element.parent to element").

Set parent attribute: target parent doesn't exist or is not a structural object
An attempt was made to reparent an object to a target that cannot contain it. For example, a graphic element cannot be the child of a sound element.

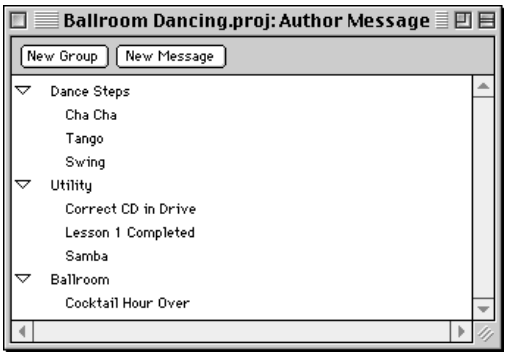
Stack overflow
An attempt was made to evaluate an expression that is too complex. Try breaking the expression into smaller parts by storing intermediate values in variables.

Text: Incompatible setting
An attempt was made to alter or make editable a text element that was converted to a bitmap. A text element is converted to a bitmap when the "Convert text to bitmap" option is checked in its **Element Info** dialog box. See "Convert Text to Bitmap Check Box" in Chapter 5.

Zero division error
A division operation was attempted with a denominator of zero (such as "3/0"). The result of this operation is not a number.

Author Messages Window

Choose **Author Messages Window** from the **View** menu to display the Author Messages window.



The Author Messages window

Author messages are one way to activate modifiers. They can be created using the **Message** and **Message/Command** pop-up menus in modifier dialog boxes, or in the Author Messages window. See "Author Messages" in Chapter 13 for a complete description.

The Author Messages window can be used to create and manage author messages from a central location. Author messages can be created, edited, deleted, or placed in groups from this window. Controls for this window are described below.

New Message button

Click this button to create a new author message. An untitled message appears in the Author Messages window. Double-click the new message to change its name.

Once created, a new author message appears as an item in the Author Messages window and as a submenu item under Author Messages in each modifier's **When** pop-up menu or **Message/Command** pop-up menu.

New Group button

Click this button to create an untitled author message group. An untitled group appears in the Author Messages window.

Once created, a new group appears as an item in the Author Messages window and as a submenu item under Author Messages in each modifier's **When** pop-up menu or **Message/Command** pop-up menu. This feature can be used to group sets of related author messages together.

Author message groups can contain both author messages and other author message groups.

Open and close triangle

To expand or collapse the contents of author message groups, click their triangles.

To delete author messages:
Select either an author message or a group from the list and press the **Delete** key.

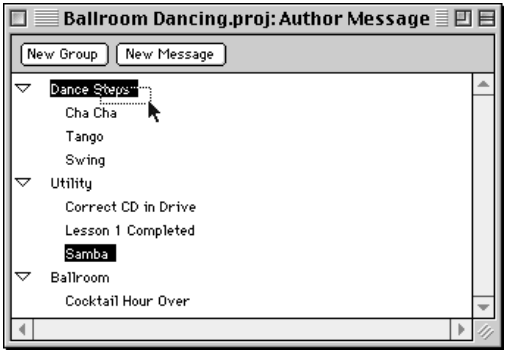
Note: Deleting an author message from the window deletes the author message from the list but does not delete occurrences of this message in the project. The author message is no longer available from the **Message/Command** pop-up menu. However, if that author message was previously selected in a modifier dialog box, that modifier will continue to send or listen for the author message until its dialog box is opened and reconfigured.

To edit or rename messages and groups:
To edit the name of an author message or to rename a group, double-click its name in the Author Messages window and enter a new name.

Note: Author messages must have unique names, even if they reside in different author message groups. They also must not duplicate the names of built-in messages (such as "Mouse Up"). If the name of the new author message is already in use, mTropolis displays an alert and asks for a different author message name.

To move author messages and groups into new groups:
Standard drag and drop methods can be used to move items in the Author Messages window.

- Choose the author message or group to be moved and drag it to a new group. An outline of the dragged object appears.
- Drag the object to its destination. When the targeted destination group highlights, release the mouse button.



Moving an author message to a new group in the Author Messages window. Here, the author message "Samba" is being moved from the "Utility" group to the "Dance Steps" group

To reorder author messages in the dialog box:

- Choose the author message to be moved and drag it to a new location. A dotted, horizontal insertion bar appears between items, indicating where the item will appear when dropped.

Using View Menu Options in Edit Mode

The last set of **View** menu items controls the appearance of the layout and layers views. The menu items described below can be used to remove visual clutter from the editing view, or to simulate the view of the project in run-time.

Preview color table

Select a color table name from this cascading menu to see the effect of a color table (that has been previously linked to the project) during edit mode. Select **Mac OS 8bit** to return the display to its default color scheme. Regardless of the setting in this menu, Color Table modifiers show their effects when they are activated in run-time mode (see "Color Table Modifier" in Chapter 12).

Hide/Show Frames

The **Frames** menu toggle can be used to show and hide the frames around elements. Choose **Frames** from the **View** menu (⌘-Shift-F). The default is on.

When **Frames** are hidden, **Modifiers** and **Names** are automatically hidden as well. These items will be disabled in the **View** menu.

When **Frames** are shown, the **Modifiers** and **Names** options are enabled. These options can then be toggled individually.

Hide/Show Modifiers

To hide or show the modifier and behavior icons on elements (in the Layout or Layers windows only — the structure view always shows modifiers), use the **Modifiers** menu toggle in the **View** menu (⌘-Shift-M). The default is on.

Note: This option is not available when **Frames** is toggled off.

Hide/Show Names

To hide or show element names, use the **Names** menu toggle in the **View** menu (⌘-Shift-N). The default is to show names.

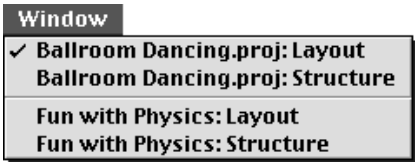
Note: This option is not available when **Frames** is toggled off.

Reveal Shared Scene
Use the **Reveal Shared Scene** menu toggle to show or hide the shared scene with the scene currently displayed in the Layout window. The default is off. When this option is on, the scene appears as it would during run-time.

Sync Windows
Choose **Sync Windows** (⌘-U) to make the selected component of one window visible in all other editing windows.

Window Menu Options

The **Window** menu displays a list of the editing windows that are currently open. Windows are listed by project name and then window name. Choose an item from the menu to make that window the active one. A check mark appears next to the name of the active window.



*The **Window** menu. Here, two projects are open and each has both its Layout and Structure windows open. The checked menu item indicates the currently active window*

8. Structure Window

8. Structure Window

This chapter provides information on working in the Structure window, one of the three editing views in mTropolis. The other views are described in Chapter 9, “Layout Window” and Chapter 10, “Layers Window.”

To open the Structure window: Choose **Structure Window** from the **View** menu (⌘-2). The Structure window appears as the active window.

All three editing windows can be open at once. Any of the open windows can be made active by clicking it. Optionally, use the window’s corresponding keyboard command to display it or make it active: Layout window ⌘-1, Structure window ⌘-2, Layers window ⌘-3.

Structure Window Overview

mTropolis projects have a hierarchical structure. The highest level of the structure is the project itself. The hierarchy then descends to sections, subsections, scenes, and finally to elements. Modifiers can be placed anywhere in the structure hierarchy.

The expanded view of a default project in the Structure window reflects this hierarchy. Components of the Structure window are described below.

Element levels

Each kind of element in the project's hierarchy is shown on its own indented position. The default view of a project shows four levels of elements: project, section, subsection, and scene.

Structural elements

Structural elements can contain and therefore can provide structure for other elements.

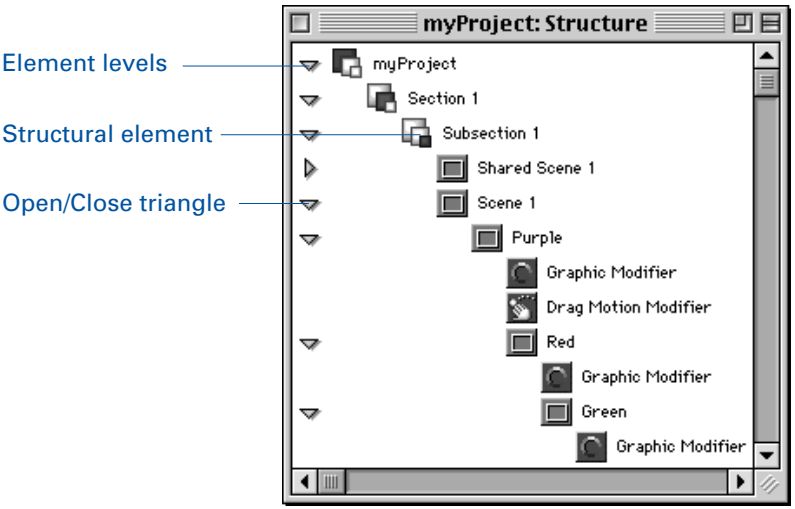
A structural element that contains other elements or modifiers is called the "parent" of the first level of items it contains. For example, the project is the parent of its sections and each scene is the parent of its elements. Modifiers in the first level below an element are called "children" of that element. Children of an element are "siblings" of one another.

Behaviors are a special kind of modifier that can contain other modifiers. Modifiers within behaviors are the children of that behavior. See "Behavior Modifier" in Chapter 12.

Open and close triangles

Along the left of the Structure window are open and close triangles for collapsing or expanding those structural elements in the project that contain other components.

Click on a triangle to toggle its element or behavior between open (pointing down) and closed (pointing to the right). Option-click a



The Structure window

triangle to open or close all levels of items within the element or behavior.

When new scenes, elements, or behaviors are added to the Structure window, they do not have a triangle associated with them. This is because other components have not yet been placed in them.

Open/close triangles are shown as either shaded or filled with white. Shaded triangles indicate that media associated with that component have been loaded into memory. White-filled triangles indicate that the component contains media that have not yet been loaded by mTropolis. Closing a triangle usually causes it to turn white, indicating that mTropolis has unloaded the corresponding component's media. Note that the scene being viewed in the Layout window cannot be unloaded from memory.

The hierarchy of elements in a project affects the way that messages are passed between its components. See "Message Paths" in Chapter 13.

Renaming elements and modifiers

When project components are created, they are given a default name. This name is shown next to the item's icon in the structure view. Although mTropolis stores unique, internal identification numbers for each item in a project, making it possible to give items identical names, it is good practice to give items unique names. Items can be renamed in the Structure window.

To rename an item:

1. Click its name once and wait briefly, or click the items icon and press the Return key. The text highlights.
2. Type the new name.
3. Alternatively, double-click an element or modifier. Its dialog box will appear on the screen. Enter the new name in the **Name** field of the dialog box and click **OK.** The name of a component can also be changed by entering a new name for the selected element in the **Object Info** palette (see "Object Info Palette" in Chapter 11).

!!! Note: Projects cannot be renamed in the Structure window. The name of a project is the same as its file name. Use the **Save As** menu item in the **File** menu to rename a project.

Managing the Structure Window

An elaborate project can have many sections and subsections with many scenes. mTropolis provides keyboard commands for limiting the display of the Structure window to selected levels in the hierarchy.

Viewing different levels of the structure hierarchy

The commands described below move a selected item to the top of the Structure window. This item becomes the view's new "root." All items above this object disappear from view, making the items below it easier to examine.

Click the desired item to select it before using the commands described below:

- **⌘-Option-Down Arrow:** This keyboard command confines the Structure window to display a selected item. The element is placed at the top of the Structure window, showing its modifiers below it.
- **⌘-Option-Up Arrow:** This keyboard command displays the next level above the current level. The parent of the chosen element appears at the top of the Structure window.

- ⌘-Shift-Option-Up Arrow: This keyboard command displays all levels above the currently-selected level, all the way up to the project level.

Controlling open and close triangles

Use the following keyboard commands as alternatives to clicking the expand and collapse triangle buttons:

- ⌘-Right Arrow: This keyboard command expands the selected component(s).
- ⌘-Left Arrow: This keyboard command collapses the selected component(s).
- ⌘-Option-Left Arrow: This keyboard command collapses the entire hierarchy under the selected component(s).
- ⌘-Option-Right Arrow: This keyboard command expands the entire hierarchy under the selected component(s).

Note: To hide or show modifiers in the Structure window, use the **Hide/Show Modifiers** menu toggle in the **View** menu.

Stepping through the Structure window

Use the following keyboard commands to step from component to component up and down the list in the structure view.

- **Tab** or **Down Arrow:** Use these keys to move down through the list.
- **Shift-Tab** or **Up Arrow:** Use these keys to move up through the list.

Adding Components to a Project in the Structure Window

When a new project is created, it has a single section, a subsection, a shared scene, and a first scene named "Untitled Scene."

New elements can be added to a project in the Structure window using the **New Section**, **New Subsection**, **New Scene**, **New Graphic**, **New Sound**, and **New Text** options in the **Object** menu.

Creating new sections, subsections, scenes, and elements

To insert a new section, subsection, scene, or element in a project, select an item and choose the desired option from the **Object** menu.

Where the new item appears in the project's hierarchy depends upon both the item selected in the window and the option chosen from the menu:

- If the selected component is the same type as the item to be created, the new item appears next to the selected component.
- If the selected component is a different type from the item created, the new item appears as a child of the nearest component that can be a parent.

For example, if a section is selected when **New Graphic** is chosen from the **Object** menu, the graphic appears as a child in the first scene below the section.

To create a new section:

1. Choose an item in the structure view.
2. Choose **New Section** from the **Object** menu (⌘-Option-E). A new, untitled section is added to the project.

To create a new subsection:

1. Select an item in the structure view.
2. Choose **New Subsection** from the **Object** menu (⌘-Option-F). A new, untitled subsection is added to the project.

Note: When a subsection is created, a shared scene is automatically created with it.

To create a new scene:

1. Select an item in the structure view.
2. Choose **New Scene** from the **Object** menu (⌘-Option-S). A new, untitled scene is added to the project.

To create a new graphic element:

1. Select an item in the structure view.
2. Choose **New Graphic** in the **Object** menu (⌘-Option-G). A new graphic element named "Untitled Graphic" is added to the project.

Note: New graphic elements created in the Structure window appear in the top left corner of their scene when viewed in the Layout window. If more than one graphic element is added to a scene in the Structure window, the elements will overlap in the Layout window. Switch to the layout view to resize and position the new elements in the scene. Graphic elements created in the Structure window are 50 × 50 pixels in the Layout window.

Note: Graphic elements cannot be added at the project, section, or subsection levels.

To link an external media file to a graphic element:

1. Select a graphic element in the structure view.
2. Choose **Link Media** from the **File** menu (or use ⌘-L). A standard file dialog box appears.
3. Navigate to the media file. Only valid media files are listed (QuickTime movies, mToons, and PICTs).
4. Double-click the file name, or highlight the file name and choose **Link.** The element's icon changes to the icon for the media type. The file name of the external media appears to the right of the icon.

Note: Any media in existing elements can be changed simply by linking new media to the element. Optionally, drag and drop new media on the element from the **Asset** palette (see "Asset Palette" in Chapter 11).

Note: By default, media that are added to the project appear in the Layout window at the screen size of the external media file. This behavior can be changed by unchecking the **Auto Resize Linked Media** check box in the **Application Preferences** dialog box (see "Auto Resize Linked Media Check Box" in Chapter 2).

To create a new sound element:

1. Select an item in the structure view.
2. Choose **New Sound** from the **Object** menu (⌘-Option-A). A new sound element is added to the project.

To link an external sound file to a sound element:

1. Select a sound element in the structure view.
2. Choose **Link Media** from the **File** menu (⌘-L). A standard file dialog box appears.

3 Navigate to the sound file. Only valid sound files are listed (AIFF files).

4 Double-click the file name, or highlight the file name and choose **Link.** The file name of the external sound file appears in the text field to the right of the icon.

Note: Sounds cannot be added at the project, section, or subsection level. Sound elements cannot be added to a project from the Layout or Layers windows — they can only be added in the structure view. Sound Effect modifiers, however, can be added to any level of the project hierarchy. See "Sound Effect Modifier" in Chapter 12.

To create a new text element:

1 Select an item in the structure view.

2 Choose **New Text** from the **Object** menu (⌘-Option-D). A new text element is added to the project.

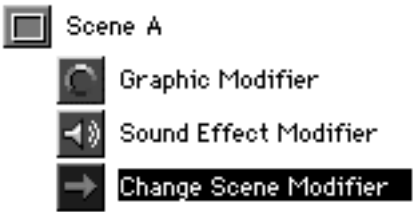
Note: New text elements created in the Structure window appear in the top left corner of their scene when viewed in the Layout window. If more than one new text element is added to a scene in the Structure window, the elements will overlap in the Layout window. Use the **Object Info** palette to precisely position the new text element, or switch to the layout view to resize and position the new elements in the scene. Text elements created in the Structure window are 50 × 50 pixels in the Layout window.

Changing the Order of Components in the Structure Window

Messages are passed to components in the project hierarchy according to their descending order in the Structure window. The messaging order of components can be changed by moving them into new positions.

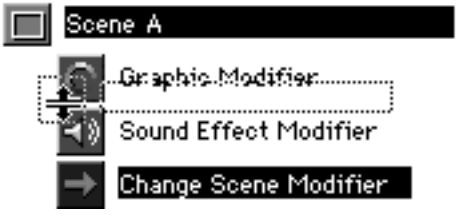
The following example illustrates the method for moving a modifier from one position in a scene to another. The method is the same for any component in a project.

The following figure shows the original order of a scene's three modifiers:



To reverse the order of the Sound Effect modifier and the Change Scene modifier:

- Click and drag the Change Scene modifier until the element is highlighted. The cursor changes to an insertion bar when the Change Scene modifier is positioned between the other two modifiers. The highlight box (a dotted line rectangle) provides visual feedback for the insertion point of the objects as shown in the figure below:



- Release the mouse button. The modifier moves to its new position.

Note: Use Shift-click to multiple-select items to be moved.

When using this method to change the order of elements, be careful that another element is not highlighted when the first element is dropped. Dropping an element on top of another element creates a parent/child relationship, affecting the order in which messages are passed to those objects. Parent/child relationships are discussed in more detail in "Message Passing Among Elements" in the next section.

Message Passing Among Elements

When elements are first created, they are positioned below the scene in the project's structural hierarchy. These elements are referred to as children of that scene. This hierarchical arrangement, combined with the messaging order of elements and modifiers in a scene, determines the path of messages that are passed through the scene.

For example, a message sent to a scene from a messenger is first passed to any modifiers on that scene according to their messaging order (the descending order in which they appear in the Structure window). From there, the message is passed to the first element in the scene and its modifiers, and then to the next element, and so on. When there are no further modifiers or elements in the scene, the path of the message comes to an end.

Effects of parent/child relationships

The hierarchical relationship between elements in a scene can be changed, changing the path of messages. When an element is made a child of another element in a scene, messages are passed from the element to its child element(s) before being passed to the next element in the scene.

New parent/child relationships between elements can be created in the Structure window using the drag and drop methods outlined in "Creating New Parent/Child Relationships in the Structure Window" in the next section. Alternatively, the relationship between elements in a scene can be changed using the parent/child tool in the Layout window. See "Parent/Child Tool" in Chapter 11.

The parent/child relationship among elements has an additional effect; because the position of a child element is calculated relative to the position of its parent, a child element will move when its parent is moved.

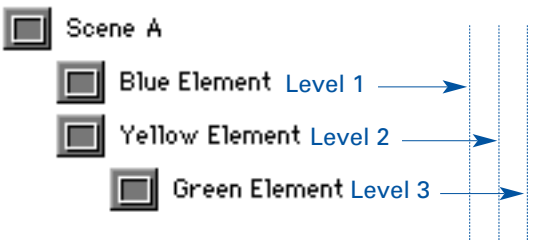
More information on the path of messages through a project can be found in "Message Paths" in Chapter 13.

Creating new parent/child relationships in the Structure window

The following section illustrates the drag and drop method used to create new parent/child relationships among elements in the Structure window.

To make one element the child of another, drag and drop the first element onto the second element. The second element will highlight as the first element is dragged over it. In the following figure, the Green Element is dropped on the Yellow Element:

The Green Element becomes a child of the Yellow Element. The result of this move, as seen in the Structure window, appears in the figure below:



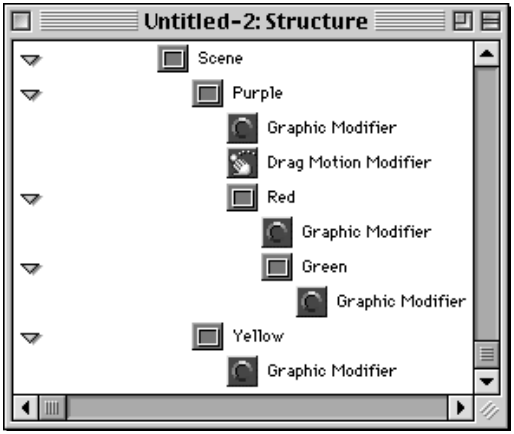
Note: The Green Element in the previous diagram is indented relative to the Yellow Element, indicating their new parent/child relationship. Parent/child relationships can also be made in the Layout window using the Parent/Child Tool (see "Parent/Child Tool" in Chapter 11).

Note: Multiple elements can be made children of another element. Use Shift-click to multiple-select elements to be moved.

Note: Graphic elements cannot be made children of sound elements.

Appearance of parent/child relationships in the Structure window

The Structure window below shows a series of child elements and their modifiers. Relationships between the levels shown in the following figure are described below.



Relationships between components in the structure hierarchy

Level 1

By default, any element that is placed in a scene appears on the first level of elements in the scene. The element's modifiers appear below it, indented to the right. In this example, the elements Purple and Yellow are children of the Scene.

Level 2

The element Red has been made a child of element Purple. Therefore, Red appears on the next element level in the scene, which is indented relative to the Purple element. The Red element's Graphic modifier appears below it, indented to the right.

Level 3
The Green element is a child of the Red element. It appears on the next element level, indented relative to the Red element, along with Red's Graphic modifier. Green's Graphic modifier appears below it, indented yet another step to the right.

9. Layout Window

9. Layout Window

This chapter provides information on working with the Layout window, one of three editing views in mTropolis. The other views are described in Chapter 8, “Structure Window” and Chapter 10, “Layers Window.”

Layout Window Overview

The Layout window is the default view when mTropolis creates a new project.

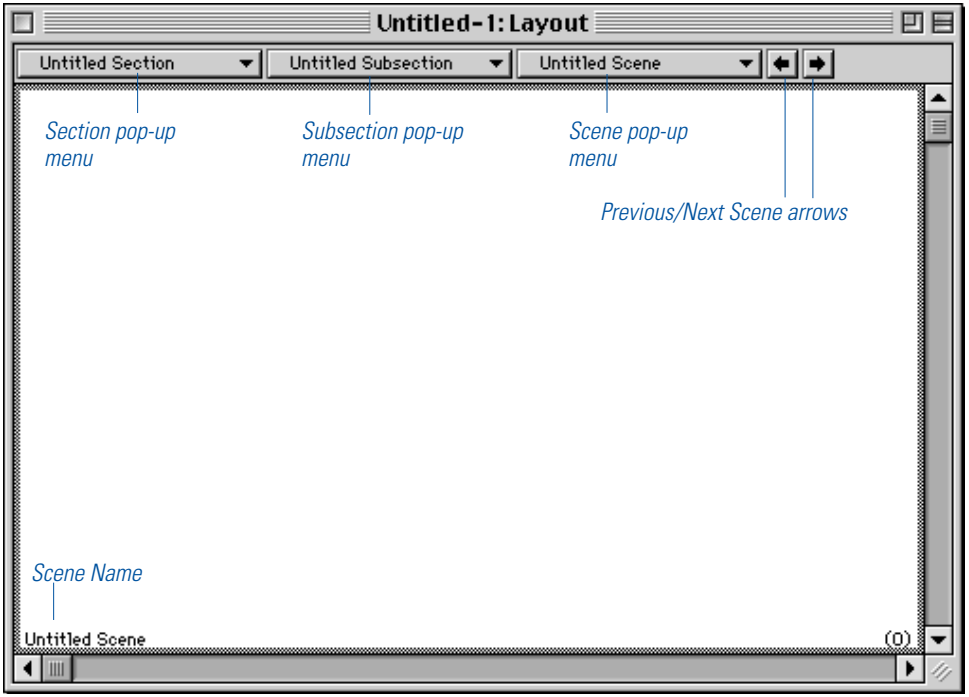
When mTropolis is launched, or when a new project is created, mTropolis creates an untitled project with a single section (with the default name Untitled Section), a single subsection (Untitled Subsection), a shared scene (Untitled Shared Scene), and a scene (Untitled Scene).

To make the Layout window active:
Choose the **Layout Window** option from the **View** menu (⌘-1). The Layout window appears in the foreground.

All three editing windows can be open at once. Any of the open windows can be made active by clicking them. Optionally, use the window's corresponding keyboard command to display it or make it active: Layout window (⌘-1), Structure window (⌘-2), Layers window (⌘-3).

!!! The structure and layer views may be closed, but closing the layout view closes the project. The Layout window shortcut (⌘-1) only works if the window is open but concealed.

Scene
The main workspace found in the center of the Layout window represents the current scene being edited. The current scene is the scene whose name appears in the **Scene** pop-up menu.



A new, empty Layout window

Default properties of scenes

A scene has the standard properties of a graphic element — it is colorless and transparent by default, in the Layout window its boundaries are represented by a frame, and, like other elements, it can be configured with modifiers and linked to external media. Its default settings can also be edited in its **Element Info** dialog box.

The boundaries of the scene appear in the Layout window as a frame. The scene's default size, 640 × 480 pixels, can be changed using the **Object Info** palette. This palette can be displayed by choosing **Object Info Palette** from the **View** menu.

Initially, scenes are locked into position in the Layout window. When locked, scenes cannot be moved. However, media can still be linked to the scene by using the **Link Media ➛ File** option in the **File** menu. To unlock a scene, select the scene and choose the **Lock** menu toggle in the **Object** menu (⌘-K).

Layer order numbering of elements

The first scene in a new project has a layer order of 0. As each element is added to the scene, it is given a unique sequential layer order number that is increased by 1. The layer order number appears at the bottom right of the element. Elements with higher layer order numbers appear to be in front of elements with lower layer order numbers.

The layer order of elements in a scene can be changed using the following methods:

- Use the layer order options from the **Arrange** menu. See "Changing the Layer Order of Elements" in Chapter 4.
- Drag and drop elements in the Layers window. See "The Layer Order Grid" in Chapter 10.
- Change the layer order number in the **Element Info** dialog box. See "Object Info and the Element Info Dialog Box" in Chapter 5.
- Use the **Object Info** palette of a selected element. See "Object Info Palette" in Chapter 11.

Adding Elements to Scenes

Graphic and text elements can be added to scenes using the Layout tools in the **Tool** palette or by using options in the **Object** menu. Graphic media can also be added to a scene by dragging and dropping assets from the **Asset** palette. By default, all elements added to a scene become children of the scene and appear below the scene in the project's structural hierarchy.

As a result, deleting a scene deletes both the scene and any elements and modifiers it contains from the project.

To create a graphic element in the Layout window:

1. Select the Graphic tool from the **Tool** palette. This tool looks like a single rectangle. The cursor changes to a cross hair.
2. Click and drag on the scene to create a new graphic element. As you drag, a graphic element frame is created. The name Untitled Graphic appears in the bottom left of the element's frame and its layer order number appears in parentheses at the bottom right.

To create a text element in the Layout window:

1. Select the Text tool from the **Tool** palette. This tool looks like a letter A. The cursor changes to an I beam and a small box.
2. Click and drag on the scene to create a new text element. As you drag, a new text element frame is created.
3. Click inside the text element. A flashing insertion bar appears.
4. Enter the desired text from the keyboard. When finished, select the Selection tool and click outside the text element to finish the editing process.

The text is displayed in black on a white background. The name Untitled Text appears at the bottom left of the element's frame and its layer order number appears in parentheses at the bottom right.

Modifiers can be added to both graphic and text elements. Graphic elements can be linked to external media (see "Linking Media to an Element in the Layout Window" in Chapter 10). Text in text elements can be modified using options from the **Format** menu (see Chapter 3, "Format Menu").

More information on the Graphic and Text tools can be found in "Tool Palette" in Chapter 11. Elements can also be added to scenes in the Structure and Layer windows. See "Adding Components to a Project in the Structure Window" in Chapter 8 and "Creating Scenes and Elements in the Layers Window" in Chapter 10.

Linking media to an element in the Layout window

1. Select an existing element and choose **Link Media ➛ File** from the **File** menu (⌘-L). A dialog box appears, with valid media types listed.
2. Select the desired media file and click **Link.** The media appears within the element's frame. A thumbnail of the media file is automatically placed in the **Asset** palette.

By default, the element will conform to the spatial dimensions of the external media file. Resize the element by dragging on its frame or use the **Object Info** palette to precisely resize the element. See "Object Info Palette" in Chapter 11.

!!! Note: There are generally five levels of structural elements in a project's hierarchy: project, section, subsection, scene, and element. Of these elements, only scenes (which behave much like graphic elements) and elements can contain external media.

!!! Note: Sounds can only be linked to sound elements, which can only be created in the Structure window. See Chapter 8, "Structure Window" for details.

Other techniques for linking media

Media can also be linked to elements using the Structure window, Layers window, and **Asset** palette:

- For instructions on how to create and link media to elements in the structure view, see "Creating New Sections, Subsections, Scenes, and Elements" in Chapter 8.
- For instructions on linking media in the Layers window, see "Linking Media to Elements in the Layers window" in Chapter 10.
- The **Asset** palette can also be used to link media, see "Linking Media to the Asset Palette" in Chapter 11.

Navigating in the Layout window

The Layout window provides a view of a single scene. Use the **Section**, **Subsection**, and **Scene** pop-up menus at the top of the Layout window to move to new locations in a project.

To navigate to a new scene:

1. Choose the appropriate section from the **Section** pop-up menu (if there is more than one section in the project).
2. Choose the appropriate subsection from the **Subsection** pop-up menu (if there is more than one subsection in the project).
3. Choose the desired scene from the **Scene** pop-up menu. mTropolis changes to the chosen scene.

Optionally, you can use the **Next/Previous Scene** arrows to move to other scenes within the currently-selected subsection.

Adding new sections, subsections, and scenes to a project

The **Section**, **Subsection**, and **Scene** pop-up menus can be used to create new sections, subsections, and scenes.

To create a new section or subsection:

Choose **New Section** or **New Subsection** from the **Section** or **Subsection** pop-up menu. The new section or subsection appears in the Layout window. The name of the new section or subsection is shown on the **Section** or **Subsection** pop-up menu.

When a new subsection is created, a shared scene is also automatically created. See "The Shared Scene" in this chapter.

To create a new scene:

1. Choose **New Scene** from the **Scene** pop-up menu. The new scene appears in the Layout window. The scene is created with the default name Untitled Scene.
2. Rename the new scene by double-clicking within its frame to display its **Element Info** dialog box. Enter a new name in the element name text field. When the **Element Info** dialog box is closed, the scene is renamed.

!!! Note: Scenes are initially locked into position in the Layout window. Locked scenes cannot be moved or resized. However, media can be linked to a locked scene by choosing the **Link Media ➛ File** option in the **File** menu (and, by default, the scene will resize itself to fit the linked media). To unlock a scene, select the scene and toggle **Lock** off in the **Object** menu (⌘-K).

Alternatively, the scene can be renamed using the Object Info palette:

1. Choose **Object Info Palette** from the **View** menu. The palette appears.
2. Select the scene.
3. Highlight the scene name, found in the top left of the **Object Info** palette.
4. Edit the scene name in the name field.
5. Click anywhere outside the name field to commit the change.

!!! Note: Sections, subsections, scenes, elements, and modifiers can also be renamed in the Structure window. See "To rename an item" in Chapter 8.

The Shared Scene

mTropolis automatically adds a shared scene to a new subsection when it is created. The contents of the shared scene are visible in every scene that belongs to the subsection when the project is viewed in run-time mode. Shared scenes are useful for storing background images and the elements and modifiers that are common to all the scenes in a subsection.

By default, the first scene in a subsection is the shared scene. However, any scene in a project may be assigned as the shared scene by using the Shared Scene modifier. See the "Shared Scene Modifier" in Chapter 12.

To show the shared scene in edit mode:
By default, the shared scene is not displayed when editing other scenes. It is usually only visible in run-time mode.

Choose **Reveal Shared Scene** from the **View** menu to toggle the display of the shared scene on and off while editing other scenes. The shared scene is displayed behind other elements in the scene as it will appear in run-time.

Note: By default, shared scenes are locked when they are first created. To unlock a shared scene, select it and toggle off the **Lock** option from the **Object** menu (⌘-K).

10. Layers Window

10. Layers Window

This chapter provides information on working with the Layers window, one of three editing views in mTropolis. The other views are described in Chapter 8, “Structure Window” and Chapter 9, “Layout Window.”

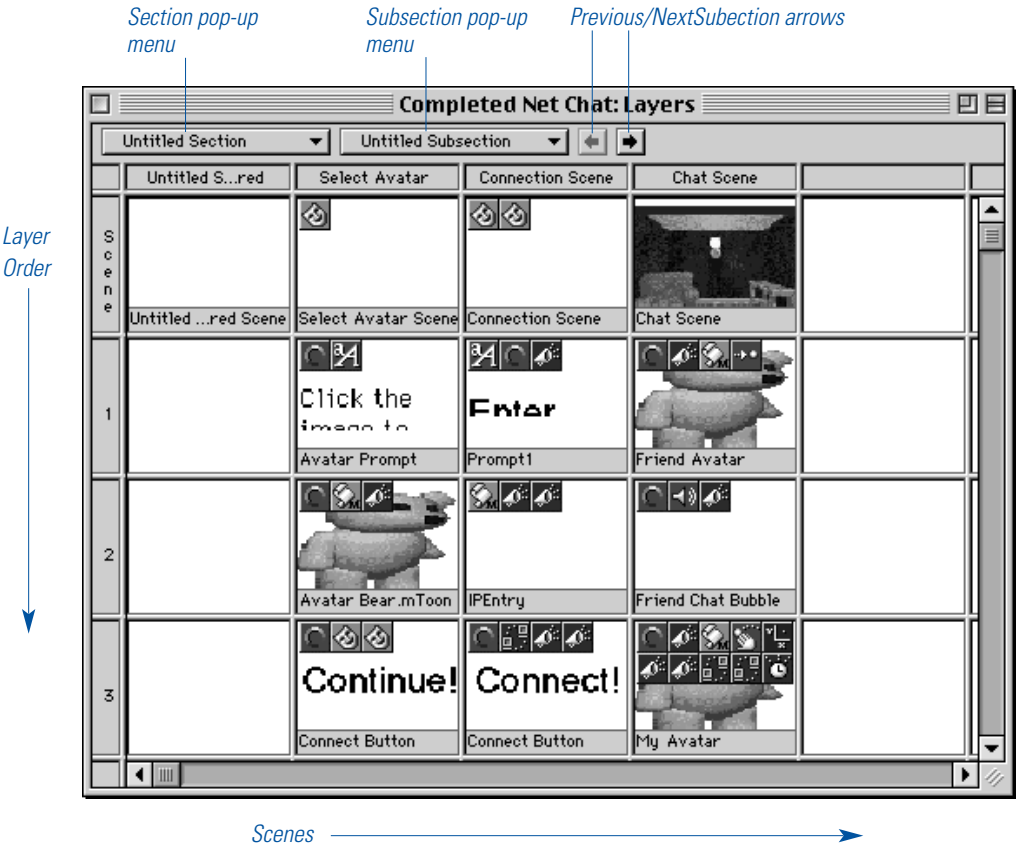
Layers Window Overview

The Layers window can be used to view and edit an entire subsection of the project at a time. This view is useful for adjusting the layer order of elements within a scene or for accessing elements that are invisible in the layout view because of their current position (for instance, outside of the scene) or layer order (for instance, they are hidden by another element).

Controls in the Layers window of a project subsection are described in the figure below.

To open the Layers window:
Choose **Layers Window** from the **View** menu (⌘-3). The Layers window appears.

All three editing windows can be open at once. Any of the open windows can be made active by clicking them. Optionally, use the window's corresponding keyboard command to display it



The Layers window. Items in the grid are child elements of the selected sub-section organized horizontally by scene and vertically by the layer order of the elements

or make it active: Layout window (⌘-1), Structure window (⌘-2), Layers window (⌘-3).

!!! Note: The structure and layer views may be closed, but closing the layout view closes the project. The Layout window shortcut (⌘-1) only works if the window is open but concealed.

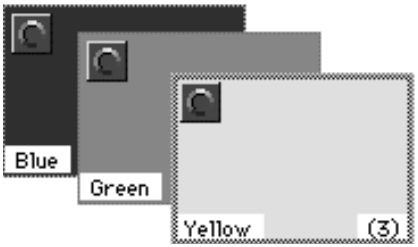
Layer order grid

The Layers window displays all the component elements of a subsection. Thumbnail images in the grid represent individual elements. Each column of the grid represents a scene. The elements in an individual scene are shown in ascending layer order.

Layer order

Each column of the grid shows elements of the corresponding scene in their layer order, that is, the order in which the elements are drawn on the screen. The scene at the top of the grid is drawn first, then the next element in the column, and so on.

Elements drawn on top of each other create the illusion that the two-dimensional (X,Y) screen has a depth (Z) dimension. The effect is commonly called 2.5D.



The effect of the mTropolis layer order

The layer order numbering begins with the scene at 0, and is increased by 1 as new elements are added to the scene.

The order of elements in a scene can be changed in the Layers window by simply dragging and dropping an element's thumbnail image to a new position in its current column.

Scene order

As mentioned previously, each column in the grid represents a scene. The column furthest to the left represents the shared scene. Subsequent scenes are shown in ascending order to the right.

The order of scenes can be important — for example, when the scene change modifier is being used with its Next Scene in Subsection or Previous Scene in Subsection options. The order of scenes can be changed in the Layers window by simply dragging and dropping a scene's thumbnail image to a new position in the Scene row.

When new scenes are created and added to the subsection, they are added as untitled scenes. When a scene is deleted, all other scenes automatically move to fill the space left by the deleted scene. Note that, because the elements of a scene are children of that scene, deleting a scene also deletes all of its elements.

Shared scenes

The leftmost scene shown in the grid is the shared scene. Shared scenes are special scenes that come first in a subsection. They are used to store the elements and modifiers common to all the scenes in a subsection. For example, elements and modifiers that form navigational buttons are often placed in the shared scene.

Note: Any scene that moves into the left-most position in the grid (any scene that is made the very first scene in a subsection) becomes the shared scene of its subsection. Any scene that was previously in that position becomes a regular scene and is no longer shared.

Elements and modifiers

Graphic elements are displayed as thumbnails in the cells that make up the grid. Newly created graphic elements are shown with the default name Untitled Graphic.

Graphic elements linked to external media (for example, mToons, QuickTime movies, PICTs) and text elements that contain text are shown as small thumbnail images. The name of the element appears across the bottom. Any modifiers on the element appear across the top of the thumbnail as shown below.

A graphic element linked to media as seen in the layers view

Hiding editing aids using View menu options

The following options can be used to reduce visual clutter in the Layers window:

- Toggle **Hide/Show Frames** in the **View** menu (⌘-Shift-F) to alternately hide and show element frames, modifiers, and names.
- Toggle **Hide/Show Modifiers** in the **View** menu (⌘-Shift-M) to alternately hide and show modifier icons.
- Toggle **Hide/Show Names** in the **View** menu (⌘-Shift-N) to alternately hide and show element names.

Creating Scenes and Elements in the Layers Window

New scenes and graphic and text elements can be quickly created in the Layers window by using the options in the **Object** menu.

To create a new scene:

Select the first unoccupied cell in the Scene row (the first row) of the Layout window.

Choose **New ➛ Scene** from the **Object** menu (⌘-Option-S). A new scene is added after the last scene in the subsection.

A new scene can also be inserted between existing scenes:

Choose an existing scene and choose **New ➛ Scene** from the **Object** menu (⌘-Option-S). The new scene appears directly to the right of the chosen scene. The scene that was in that location moves to the right (that is, it becomes the next scene of the newly-created scene). All later scenes also move one position to the right.

To create a new graphic element:

Select any grid cell, occupied or unoccupied, under an existing scene in the Layers window.

Select **New ➛ Graphic** from the **Object** menu (⌘-Option-G). The new graphic element appears in the targeted location. If an occupied cell was selected, the selected element and any elements below it are moved down one position in the layer order to accommodate the new graphic element. The newly-created graphic element has the name Untitled Graphic.

To create a new text element:
Choose any grid cell, occupied or unoccupied, under an existing scene in the Layers window.

Select **New** ➛ **Text** from the **Object** menu (⌘-Option-D). A new text element appears in the targeted location. If an occupied cell was selected, the selected element and any elements below it are moved down one position in the layer order. A new text element is called Untitled Text.

Editing in the Layers Window

During project editing, it is often necessary to duplicate scenes, elements, or modifiers, or to move them from one place to another in a subsection. Because the Layers window shows small images of all the media elements in a subsection, it is a convenient window in which to duplicate and relocate components.

Standard project editing tools can be used in the Layers window. The **Cut**, **Copy**, **Paste**, and **Duplicate** options in the **Edit** menu can be used to edit elements and modifiers.

As in other windows, modifiers and elements can be moved in the Layers window using drag and drop techniques. These components can also be configured using standard methods.

Tools that are not active in this window include the Graphic, Crop, Text, and Parent/Child tools of the **Tool** palette.

Changing the order of scenes

The order of scenes in a subsection can be easily changed using the Layers window. Moving a scene also moves all of the elements within the scene (an entire column in the Layers window is moved).

To change the order of a scene in a subsection:

1. Select the scene to be moved.
2. Drag and drop the scene over the scene to be displaced. The dropped scene appears in the new location. All other scenes in the subsection move to accommodate the change.

!!! Note: Use Shift-click to select a group of scenes to be moved. Also, Option-drag can be used to move a copy of a scene, leaving the original in place.

Changing the layer order of elements

The layer order of elements in a scene can be changed by dragging and dropping an element over another element.

To move an element between other elements:

1. Select an element in the Layers window.
2. Drag and drop the element over the element to be displaced. The dropped element appears in the new location. The element that previously occupied the targeted location moves down one position in the layer order. All later elements in the layer order also move down.

!!! Note: Use Shift-click to select a group of elements to be moved. Also, Option-drag can be used to move a copy of an element, leaving the original in place.

To move an element to an empty grid location:
Click the element and drag and drop it onto the empty grid location. The element appears in the new grid cell.

Duplicating scenes or elements

Elements or scenes can be duplicated using the same technique in the Layers window.

To duplicate a scene or element:
Choose **Duplicate** from the **Edit** menu (⌘-D) to duplicate the currently-selected scene, graphic, or text element. A duplicated scene will appear to the right of the original scene. A duplicated graphic or text element will appear below its original in the layer order.

Eliminating gaps in the layer order
When an element is moved from one scene to another using the Layers window, a gap is created in the layer order of elements in the original scene. Such gaps can be eliminated by selecting the scene in which the gap appears and choosing **Eliminate Gaps** from the **Arrange** menu. The elements in the scene are moved to eliminate empty layers between elements.

Other methods for editing the layer order of elements
There are a number of other ways to change the layer order of elements in addition to moving icons in the Layers window.

The Arrange menu options
The options for changing the layer order of elements in the **Arrange** menu (**Bring to Front, Send to Back, Bring Forward, Send Backward**) can be used in all editing windows, including the Layers window. See "Changing the Layer Order of Elements" in Chapter 4.

The Element Info dialog box
An element's layer order number can be changed in its **Element Info** dialog box. This method can be used in all the editing windows. Double-click the element or choose **Element Info** from the **Object** menu (⌘-I) to open the selected element's **Info** dialog box. See "Object Info and the Element Info Dialog Box" in Chapter 5.

Object Info Palette
The **Object Info** palette can be used to view and change the size, position, and layer order number of a selected element. It can also be used to change an element's name. To open this palette, choose **Object Info Palette** from the **View** menu (⌘-7). See "Object Info Palette" in Chapter 11.

Linking Media to Elements in the Layers Window

The **File** menu's **Link Media** option can be used to link external media files to graphic elements in the Layers window.

To link an external media file:

1. Select an element.
2. Choose **Link Media** ➝ **File** from the **File** menu (⌘-L). A dialog box appears.
3. Navigate to the external media file. Valid file types appear in the file list (QuickTime movies, mToons, and PICTs).
4. Double-click the desired file name or choose the file name and then **Link.** A new thumbnail representing the media appears in the grid cell. The file name of the linked file replaces the word "Untitled Graphic."
5. The element can be renamed by double-clicking to display its **Element Info** dialog box, where a new name can be entered, or by using the **Object Info** palette.

Adding graphic elements to the Layers window from the Asset Palette
All media that has been linked to a project is stored in the **Asset** palette. These assets can be dragged and dropped into a project in any view.

Unlike elements that are first created and then linked to media, assets in the palette do not have elements to contain them. However, when an asset is dragged and dropped from the **Asset** palette into the project, a new element is automatically created to contain the media.

To add a media element from the Asset palette:

1. Open the **Asset** palette by selecting **Asset** palette from the **View** menu (⌘-6). The **Asset** palette appears.
2. Drag the desired media element from the **Asset** palette to an empty grid location. A thumbnail image representing the media appears in the grid cell.

Navigating in the Layers Window

Use the **Section** and **Subsection** pop-up menus at the top of the Layers window to display different subsections of a project.

To navigate to a new section:

1. Select the section in which to work from the **Section** pop-up menu.
2. Select the subsection in which to work from the **Subsection** pop-up menu. The layers view displays the scenes of the newly-selected subsection.

The **Next/Previous Subsection** arrows can be used to move to other subsections in the current section.

Adding new sections or subsections to a project

The **Section** and **Subsection** navigation pop-up menus can also be used to create new sections and subsections.

To create a new section or subsection:

Choose **New Section** or **New Subsection** from the **Section** or **Subsection** pop-up menu. The new section or subsection is created with the default name Untitled Section or Untitled Subsection. This section or subsection becomes the current one and its name appears as the label of the pop-up menu.

The name of a new section or subsection can be changed using the Structure window or the **Object Info** palette, as described below.

To rename a section or subsection in the Structure window:

1. Switch to the Structure window by choosing **Structure Window** from the **View** menu (⌘-2).
2. Click the name of the section or subsection to be renamed, pause briefly and click again (or click the name and press **Enter**). The object's name highlights.
3. Type in the new name.

To rename a section or subsection using the Object Info palette:

1. Switch to the Structure window by choosing **Structure Window** from the **View** menu (⌘-2).
2. Display the **Object Info** palette by choosing **Object Info Palette** from the **View** menu (⌘-7).
3. Choose the section or subsection to be renamed in the Structure window.
4. The name field at the top left of the **Object Info** palette highlights.
5. Type the new name in the palette's name field (the upper left text box).
6. Click anywhere outside the name field to commit the change.

11. Palette Reference

11. Palette Reference

This chapter provides information on the palettes used to hold mTropolis tools, modifiers, aliases, and media assets, the basic building blocks of the mTropolis authoring environment. These palettes can be shown or hidden by toggling their menu options in the **View** menu.

When visible, the palettes float over the three editing views (the Layout, Structure, and Layers windows).

The **Tool** palette contains tools for creating graphic and text elements in the Layout window. Tools on this palette can also be used to size elements and link them in new hierarchical relationships.

The **Modifier** palettes contain the modifiers that are in the mPlugins folder.

The **Alias** palette stores the originals of modifiers that have been aliased in a project.

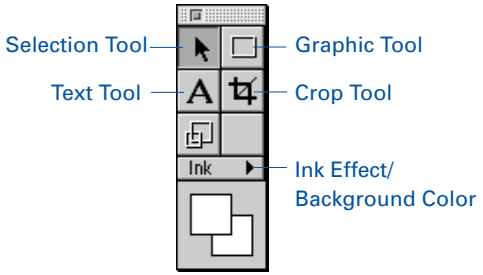
The **Asset** palette stores icons and thumbnails of media linked to the project.

The **Object Info** palette is used to view and change the name, size, position, and layer number of elements. It can also be used to identify the current cel of an mToon and to step through its cels while in edit mode.

Tool Palette

The **Tool** palette contains tools for creating graphic and text elements, sizing them, and linking them in new hierarchical relationships. Only one tool can be selected at a time.

To select a tool, click its icon in the **Tool** palette. The Selection tool is persistent — it remains the active tool until another tool is selected. Click a different tool's icon to make that tool active for a single use. After using the tool, the Selection tool becomes active again. Double-click a tool to make that tool persistent. A small triangle appears in the upper left corner of the tool's icon indicating that it is "sticky" and will remain the active tool until a different tool is selected.



*The **Tool** palette*

Selection tool

The Selection tool is a pointing device. It can be used to:

- Select items by single clicking.
- Select multiple items by Shift-clicking or by drawing a marquee around items on the screen.
- Move items by clicking and dragging. If the Shift key is held down while dragging an element, the element is constrained to move in only the horizontal or vertical direction. If the Option key is held down while dragging an object, a duplicate of the original object is created.
- Resize an element by clicking and dragging its frame to the new desired size. If the Control key is held down while resizing, the element resizes around its center point. If the Shift key is held down while resizing, the element maintains its aspect ratio. If the Option key is held down while resizing, the element resizes in exact multiples of its current size. Note that resizing also works with multiple selections.
- Open an element or modifier configuration dialog box by double-clicking the item.

The Selection tool is the default mTropolis tool. The Selection tool becomes the active tool after using any of the other tools.

Graphic tool

The Graphic tool can be used to create new graphic elements in the Layout window. New graphic elements can then be linked to media files (see "Link Media — Attaching External Media Files to Elements" in Chapter 1).

!!! Note: This tool cannot be used in the Structure or Layers window. To create graphic elements in these views, choose **New** ➛ **Graphic** from the **Object** menu.

To create a graphic element in the Layout window:

1. Click the Graphic tool. The cursor changes to a cross hair.
2. Click and drag on the scene displayed in the Layout window to create a new graphic element frame. The default name of the item, "Untitled Graphic," appears in the bottom left of the frame, and its layer

order number appears in parentheses at the bottom right.

Graphic element properties can be changed by applying ink effects (see "Ink Effects" and "Foreground and Background Colors" in this chapter) or Graphic modifiers (see "Graphic Modifier" in Chapter 12).

3 The Selection tool becomes the active tool, so the new graphic element may be moved or resized immediately.

Text tool A

The Text tool can be used to create new text elements in the Layout window.

!!! Note: This tool can only be used in the Layout window. It cannot be used in the Structure or Layers window. To create text elements in these views, choose **New ➛ Text** from the **Object** menu.

To create a text element in the Layout window:

1 Click on the Text tool. The cursor changes to an "I" beam with a small box.

2 Click and drag on the scene displayed in the Layout window to create a new text element frame of the desired size.

3 A flashing insertion bar appears inside the new text element. Enter the desired text from the keyboard (text can also be entered by pasting). The text is displayed in black on a white background.

4 Click outside the text element to end the text entry. The default name of the element, "Untitled Text," appears in the bottom left of the frame and its layer order number appears in parentheses at the bottom right.

!!! Note: You can also edit a text element by Command-clicking an existing text element, regardless of which tool is selected.

Characters supported by mTropolis

mTropolis supports the ANSI character set. All roman-based characters are supported on all mTropolis platforms. For example, German, Italian, or French text is converted and displayed properly in both Mac OS and Windows titles. mTropolis does not support multi-byte character sets (such as Kanji).

!!! Note: Many commonly-used Mac OS symbols are not in the ANSI standard character set and cannot be displayed properly in Windows. For example, the bullet character (typed by pressing Option-8) found in most Mac OS fonts is not in the ANSI character set. When displayed in Windows, these characters are replaced with the "pipe" character (|). These characters will display in Windows if the text element is converted to a bitmap (see "Convert Text to Bitmap Check Box" in Chapter 5).

The Tab character and tab stops are not fully supported by mTropolis. When typed into a text element, tab characters are converted to spaces. However, if text is pasted from the clipboard into a text element, tabs are preserved. These tab characters may cause unexpected word wrapping when the title is built for Windows platforms. When pasting text from another application into mTropolis, either remove any tabs or replace them with spaces before copying the text.

Tab and newline characters in text elements
Both tab characters and newline characters can be typed in text elements by pressing the Tab and Return keys when using the Text tool. Tab characters can be represented in Miniscript string expressions with the characters `\t`. Similarly, newline characters can be represented with the characters `\n`.

For example, the following Miniscript statement sets the text of a text element to a string that contains both tabs and newline characters:

```
set text to "Item 1\tItem 2\nItem
  3\tItem 4"
```

The contents of the text element will look like:

```
Item 1  Item 2
Item 3  Item 4
```

Changing the appearance of text
Text properties can be changed using the options found in the **Format** menu (see Chapter 3, "Format Menu") or by applying Text Style modifiers (see "Text Style Modifier" in Chapter 12), ink effects (see "Ink Effects" and "Foreground and Background Colors" in this chapter), or Graphic modifiers (see "Graphic Modifier" in Chapter 12).

!!! Note: When a project is built into a title for distribution, fonts are not bundled into the built file. To display properly, any fonts used by the project must be installed on the target system. Note also that for fonts to work on both Mac OS and Windows systems, the same font must be made available on both platforms. Those fonts must have exactly the same names on both platforms. Fonts that will be used for Mac OS and Windows distribution should be licensed from the same vendor.

Crop tool
The Crop tool can be used to reduce the size of the visible area of media in graphic elements or the visible area of text in text elements. The media itself is not resized; rather, a new view of the media is created. The cropped media still consumes the same amount of memory.

To crop an element:
1. Select the Crop tool. The cursor changes to a crosshair icon.
2. Click and drag across the element that you wish to crop. The visible area of the media is cropped to the newly-specified cropping region. Drag on the frame of the element to change the size or shape of the cropping region.

- If the Shift key is held down while dragging the Crop tool, the media is cropped to the union of the old and new cropping regions.
- If the Option key is held down while the Crop tool is dragged, the cropping area can be extended to reveal parts of the media that were previously cropped.
- If the Control key is held down while dragging on the frame of the element with the Crop tool, the cropping region resizes around the element's center

!!! Note: Cropping an element can be changed during run-time using the cropPosition and cropSize attributes described in Chapter 15.

To immediately undo a cropped element:
Choose **Undo** from the **Edit** menu (⌘-Z). The element returns to its pre-cropped size.

Alternatively, select the **Revert Size** option in the **Object** menu (⌘-R) to return the element to its original size.

Parent/Child tool

The Parent/Child tool can be used to alter the hierarchical relationship between elements in a scene.

When elements are created in the Layout window, they are automatically positioned below their scene in the project's structural hierarchy (that is, they are children of the scene). When the hierarchical relationship between a scene and its elements is changed, the path of messages that are passed through the scene is altered. For example, when an element is made a child of another element in a scene, messages will be passed from the parent element to its child element(s) before being passed to the next element in the scene. More information about parent/child relationships and message passing can be found in "Message Passing among Elements" in Chapter 8.

The parent/child relationship among elements has an additional effect; because the layout position of an element is static relative to the position of its parent, a child element will move when its parent is moved. Its position is measured relative to its parent's origin (the upper left corner of the parent element).

The sections below describe techniques for using the Parent/Child tool. When working with this tool, keep in mind that, by default, the scene is the parent of the elements it contains. When using the Parent/Child tool, make sure the scene is not selected.

To make one element the child of another:

1. Click the Parent/Child tool.
2. Click and hold the element targeted as the child.
3. Drag the mouse. A line appears from the center of the element to the location of the cursor.
4. Drag the line to the element targeted as the parent and release the mouse button. The elements are linked. Now when the parent is moved, the child moves with it.

!!! Note: Graphic elements cannot be made children of sound elements. Also, because sound elements have no representation in the Layout window, sound elements cannot be reparented with the Parent/Child tool. Use the structure view to move sound elements to new locations in the structural hierarchy.

To break the parent/child relationship between two elements:

1. Click the Parent/Child tool.
2. Click and drag the child element.
3. A line appears. Release the mouse to drop the line on the scene. The scene becomes the child's new parent, breaking the relationship with the other element.

To make several elements children of one element:

1. If it is currently selected, de-select the scene by Shift-clicking it.
2. Use Shift-click to select multiple elements to be made children. Alternatively, drag a marquee around the elements.
3. Click the Parent/Child tool.

4 Click and drag from one of the selected elements. Lines from all the elements attach to the cursor.

5 Drop the lines on the element targeted as the parent. The targeted element becomes the parent of the multiple-selected elements.

To break the parent/child relationship between a parent and its children:

1 Select one or more of the child elements.

2 Click the Parent/Child tool.

3 Click and drag from any selected child element to the scene. Lines from all selected elements attach to the cursor.

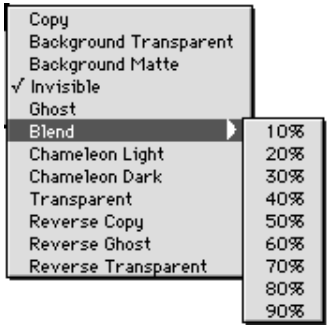
4 Drag and drop the lines on the scene. All the selected elements become children of the scene. The previous parent/child relationship among the elements is broken.

Using the Structure window to modify parent/child relationships

Parent/child relationships can also be created in the Structure window. See "Creating New Parent/Child Relationships in the Structure Window" in Chapter 8.

Ink Effects

Ink effects can be applied by selecting an element and choosing an option from the **Ink** pop-up menu. These options are the same as the ones available in the "Ink:" menu of the **Graphic Modifier** dialog box (see "Graphic Modifier" in Chapter 12). Selecting an ink effect from this pop-up menu actually creates a Graphic modifier on the element if it does not yet contain one. If the element contains several Graphic modifiers, this pop-up menu will affect only the last Graphic modifier on the element. The **Ink** pop-up menu options are described below.



*Ink effects can be applied by selecting an element and choosing an option from the **Ink** pop-up menu*

In titles built for Windows platforms, only the Copy, Background Transparent, Background Matte, and Invisible inks take effect. Other inks have no effect when used in titles built for Windows.

Copy

This is the default ink effect. It displays the graphic element in its original state.

Background Transparent

Background Transparent makes the element's background color transparent. All pixels of the specified color in the element are made transparent. That is, other elements drawn beneath the element will be visible through the transparent areas. The transparent color is specified in the background color swatch (see "Foreground and Background Colors" later in this chapter).

Background Matte
Background Matte is similar to Background Transparent. In addition to adding transparency, however, this ink makes the transparent regions unresponsive to mTropolis events (such as end user mouse clicks or collisions with other elements). Using this ink is a simple way to create objects that behave as if they have a complex border.

Invisible
An element is invisible when this option is applied. Note that this ink effect is not the same as making the element hidden (using the **Hide** command described in Chapter 13). The element is not visible, but it is enabled, has all of its normal attributes, and continues to respond to end user mouse events.

Ghost
Applied to black and white elements, Ghost creates an image that can only be seen when placed over a black background. In color, Ghost draws with the current background color. Since white is default background color in mTropolis, Ghost can be used to make text appear white on a transparent background.

Note: This ink has no effect in titles built for Windows platforms.

Blend
The Blend ink effect makes the colors of overlapping elements blend together. The blend amount can be selected in increments of 10% from 10% to 90% from the cascading menu of blend percentages.

Note: The version of this effect found in the Graphic modifier does not have a cascading menu of blend percentages. Instead, the amount of blending is determined by the selected background color. Selecting a light gray makes the element slightly transparent. Selecting a darker gray makes the element more transparent. Selecting a color other than gray gives a tinted blend.

This ink has no effect in titles built for Windows platforms.

Chameleon Light
With this ink effect, the colors in a graphic on top of a white graphic turn opaque. When placed over a colored graphic, light colors are tinted.

Note: This ink has no effect in titles built for Windows platforms.

Chameleon Dark
With this ink effect, the colors are lightened when the element is on top of another graphic. Placed over white, the graphic becomes transparent. Placed over a colored graphic, dark colors are tinted.

Note: This ink has no effect in titles built for Windows platforms.

Transparent
Applied to black and white elements, this ink creates an image that can only be seen when placed over a black background. In color, Transparent draws with the current background color.

Note: This ink has no effect in titles built for Windows platforms.

Reverse Copy
Reverse Copy reverses the black and white colors: all white pixels will appear black, all black pixels will appear white. Colors are inverted.

Note: This ink has no effect in titles built for Windows platforms.

Reverse Ghost
Black pixels become transparent and white pixels remain white. When a graphic with this ink is placed over a graphic with white pixels, the white in the first graphic becomes transparent.

Note: This ink has no effect in titles built for Windows platforms.

Reverse Transparent
White pixels change to black, and black pixels become transparent.

Note: This ink has no effect in titles built for Windows platforms.

Foreground and Background colors
The foreground and background color swatches are used to change the appearance of elements in the Layout window.

By default, the foreground color of elements is black, and the background color is white. For text elements, the foreground color is the color used when entering text into text elements, and the background color is the color used to draw its field.

To change the foreground/background color of a graphic or text element:

1. Select an element.
2. Click and hold either the foreground or background color box. The cursor changes to an eyedropper tool and a palette appears. If the scene has not been assigned a custom palette, the system palette appears. If the scene has been assigned a custom palette, it will appear.
3. Drag the eyedropper to a color on the palette or to a color in an image on the screen. Release the mouse. The foreground or background color in the element changes to the selected color.

Note: For text elements, the foreground color is the color used for the letters themselves.

Modifier Palettes

The **Modifier** palettes contain built-in modifiers. To display or hide each palette, choose **Modifier Palettes** ➛ **Logic** (⌘-Option-1), **Modifier Palettes** ➛ **Effects** (⌘-Option-2), **Modifier Palettes** ➛ **Extras** (⌘-Option-3), or **Modifier Palettes** ➛ **Network** (⌘-Option-4) from the **View** menu. (These keyboard shortcuts may change if new modifier kits are added to mTropolis.)

Each palette contains a number of modifiers, represented by icons. To see the name of a modifier in the palette, move the cursor over a modifier's icon while holding down the Control key.

Complete descriptions of the **Modifier** palettes and individual modifiers can be found in Chapter 12, "Modifier Reference."

Applying modifiers to project components
Modifiers can be placed on any element or in any behavior in a mTropolis project. The method for placing a modifier is the same in all views:

- Drag a modifier from the **Modifier** palette and drop it on its destination. In the layout and layers views, the modifier appears on the element on which it was dropped. In the

structure view, the modifier appears below and to the right of the element on which it has been dropped.

To add a modifier to a behavior:

1. Drag the modifier over a behavior icon that has already been placed in a project. When the behavior icon highlights, drop the modifier by releasing the mouse.
2. Optionally, double-click the behavior to open it. Drag and drop the modifier in the behavior window. The modifier is added to the behavior.

More information on behaviors can be found in "Behavior" in Chapter 12.

Changing modifier settings

Once a modifier has been placed on a project component, the modifier's effect can be specified by altering the settings in its dialog box.

To view or change the settings in a modifier's dialog box:

1. Double-click the modifier. The modifier's dialog box appears.
2. Change any settings within the modifier dialog box. The new settings take effect when the dialog box is closed. See Chapter 12, "Modifier Reference" and Chapter 13, "Modifier Pop-Up Menus and Message Reference" for more information on modifier settings.

Alias Palette

The **Alias** palette stores the master copies of aliased modifiers. An alias is a productivity tool that is used to globally manage modifiers with identical settings. Choose **Alias Palette** from the **View** menu (⌘-5) to display the **Alias** palette.

An alias is a special copy of a modifier that takes its functionality from the modifier from which it was made. This "master copy" is automatically placed in the **Alias** palette when the alias is made (using the **Make Alias** option from the **Object** menu or press ⌘-M). Additional aliases that refer to this master copy can also be created and placed throughout a project.

The advantage of using aliases is that they can be globally updated from a single source. Changing the settings of an alias changes the settings of both the master copy and all other aliases that refer to the same original throughout the project.

Variables, which are a class of modifiers that store values, are good candidates for aliasing. For example, a variable used to contain the score in a game can be aliased and strategically placed in the project for access by specific messengers or Miniscript modifiers. As the value of this variable changes during the game, the value of its aliases will also change, giving the modifiers efficient access to the updated score. See "Variable Scopes" in Chapter 13. Aliased variables in a scene also maintain their values across scene changes in a way that unaliased ones do not. For more information, see "Persistence of Variables between Scene Changes in Built Titles" in Chapter 13.

Aliasing is particularly powerful when used with the Behavior modifier. If a behavior will be used in many places within the project, consider making it an alias. This ensures that any changes made to any of the modifiers within the behavior will be updated in all copies of the aliased behavior.

Creating aliases

An alias can be created by selecting a modifier that has been placed in the project, and choosing **Make Alias** from the **Object** menu (⌘-M). Additional alias copies can be made by selecting and dragging the icon of the master copy from the **Alias** palette.

To make an alias:

1. Select a modifier. More than one modifier can be selected using Shift-click.
2. Choose **Make Alias** from the **Object** menu (⌘-M). The selected modifier is now an alias and the modifier icon changes to a lighter color to indicate that it is now aliased. A master copy of the modifier is automatically placed in the **Alias** palette.

!!! Note: An alias can also be created simply by dragging a modifier from a component and dropping it in the **Alias** palette window. Choose **Alias Palette** from the **View** menu (⌘-5) to view the master copies of aliases in a project.

Alias palette components

The **Alias** palette shows the master copies of all aliases in the current project. Components of the **Alias** palette are described below.

*The **Alias** palette, containing a number of alias master copies*

Trash can

When all aliases of a master copy have been deleted from the project, drag the master copy to the trash to remove it from the **Alias** palette.

Sort By pop-up menu

Use the **Sort By** pop-up menu to control the way in which mTropolis sorts modifiers on the **Alias** palette. By default, **Name** is chosen and aliases are sorted alphabetically by name. Choose to display the aliases in an unsorted list. The aliases will be roughly in the order they were created, but they may shift as aliases are added and deleted.

Modifier icons

The master copy of an aliased modifier is automatically placed in the **Alias** palette. The palette shows the modifier's icon and name.

Using aliases

Once an alias has been created, its master copy resides on the **Alias** palette. Creating and managing aliases is simple.

To modify an alias:
To modify an alias such that all copies of the alias are updated, double-click any one of the instances of the alias found in the project. The modifier's dialog box appears.

Note: Master copies of an alias cannot be edited (that is, you cannot double-click an alias on the **Alias** palette to display its dialog box). An instance of the alias must be present in the project to edit the aliased modifier.

To create a new alias copy:

1. Drag the desired modifier from the **Alias** palette. A new alias copy attaches to the Selection tool. The master copy remains on the palette.
2. Drop the alias in the desired place in the project.

Note: Aliases can also be duplicated using the standard **Cut**, **Copy**, **Paste**, **Duplicate** menu items or by Option-dragging an alias to a new position.

If an alias from one project is dropped into a different project that already contains the same alias, mTropolis displays an alert asking if you want to replace the alias (**Replace** button) with the new one or keep the existing (**Use Existing** button) alias. To cancel the copy operation, click the **Cancel** button. This feature prevents the accidental replacement of aliases with older or unintentionally different versions. This alert contains a **Don't Warn Again** check box that can be clicked to disable subsequent alerts during the current mTropolis session. Any subsequent replacements will be handled in the same manner as the first (existing aliases will either be replaced or retained depending upon which option was chosen the first time this alert appeared).

To delete an alias from a project component:
Select the alias and press the Delete key (or use **Cut** from the **Edit** menu).

To delete a master copy of an alias:
To delete a master copy from the **Alias** palette, first delete all of its aliases from the project, then drag the master copy to the **Alias** palette's trash can.

To delete the master copy of all unused aliases:
To delete the master copy of all aliases that have no occurrences in the project, choose **File ➛ Remove Unused ➛ Aliases.**

To "break" an alias:
An alias can be "broken" from its relationship to other instances of the alias and to the master copy in the **Alias** palette with the **Break Alias** menu item in the **Object** menu. Once broken from the group, any subsequent changes to the modifier apply only to that specific modifier.

A modifier or behavior that has been removed from the alias chain will retain its current settings until they are changed.

Asset Palette

The **Asset** palette is a visual database of the media that have been linked to a mTropolis project. This palette makes linking, storing, and copying media easy and convenient. Choose **Asset Palette** from the **View** menu to display the **Asset** palette (⌘-6).

Media can be displayed in the **Asset** palette as large or small thumbnail images, or icons and

names. Each item in the palette can be dragged and dropped onto elements.

Option-double clicking an asset in the palette opens its editor. Option-dragging an asset from the palette creates a new element (instead of replacing an element's media).

Asset palette Components

By default, all media files in the **Asset** palette are displayed by large thumbnails. Components of the **Asset** palette are described below.

Trash can

Remove unused media files from the **Asset** palette by dragging them into the trash can. Only media files that are not being used in the project can be removed. The **Remove Unused Asset** menu command may also be used to accomplish this automatically.

Show pop-up menu

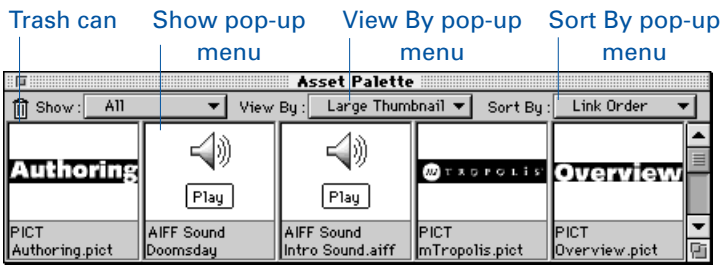
The **Show** pop-up menu allows the display of media files to be restricted to selected types as follows:

- **All:** Displays all types of media linked to the project.
- **PICTs:** Displays PICT files linked to the project.
- **mToons:** Displays mToons linked to the project.
- **Sounds:** Displays AIFF files linked to the project. When displayed by large thumbnail, sounds can be previewed by clicking on the sound thumbnail's **Play** button.
- **QuickTime:** Displays QuickTime movies linked to the project.
- **Color Tables:** Displays CLUT files linked to the project.
- **All Graphics:** Displays all graphic files (PICTs, QuickTime movies, and mToons) that have been linked to the project.

View By pop-up menu

By default, the **Asset** palette displays the media file as a large thumbnail. The **View By** pop-up menu provides an option for reducing the size of these thumbnails, or displaying each media file by its icon and name:

- **Large Thumbnail:** Displays a large thumbnail, the name of the file, and the type of media.

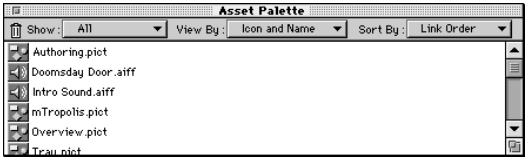


*The **Asset** palette, showing large thumbnails of available media*

- **Small Thumbnail:** Displays the thumbnail at 50% size and removes the text description of the file.

*The **Asset** palette, showing small thumbnails*

- **Icon and Name:** Displays an icon related to the media type, the file's name, its media type, and the number of times it is used in the project.



*The **Asset** palette, showing icons and names of assets*

Sort By pop-up menu

Use the **Sort By** pop-up menu to control the way in which mTropolis sorts assets on the **Asset** palette. By default, **Name** is chosen and assets are sorted alphabetically by name. Choose **None** to display the assets roughly in the order in which they linked to the project. The **Asset** palette may display much more quickly if the assets are unsorted. Assets can also be sorted by **Date Created**, **Date Modified**, **Size**, and **Type**.

Linking media to the Asset palette

1. Choose **File**, **Multiple Files**, or **Folder** from the **Link Media** submenu in **File** menu. A standard file dialog box appears. Files with valid media types are listed. Valid media types are described in "Valid Media File Types and Associated Thumbnails" in this chapter.
2. Double-click the name of the item to be linked or select the name of the item and click the **Link** button. If multiple files are being linked, the file dialog box will reappear after each selection is made. If a folder is to be linked, click the button below the list of item names when the target folder's name appears in the button. A thumbnail of each asset appears in the **Asset** palette.

Media can be dragged and dropped in the various editing views from the **Asset** palette. When media are dropped on a scene in this way, an element is automatically created to contain it.

Replacing media in an element

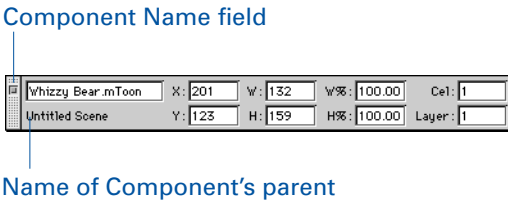
Media in an element can be replaced with new media from the **Asset** palette. To accomplish this, drag and drop the media from the palette over the element.

Viewing source file information

To view details regarding an asset's source file, double-click its image or icon in the **Asset** palette. The **Asset Info** dialog box is displayed. See "Asset Info" in Chapter 5.

Object Info Palette

The **Object Info** palette is used to view and change the size, position, and layer number of elements, or to change the name of any component.



*The **Object Info** palette*

The **Object Info** palette reports the following information about an object:

Component Name field

This text field displays the selected component's name. This field is editable. To change the name of the component, click this field and type a new name.

Name of component's parent

This non-editable text field below the component name displays the name of the component's parent.

X and Y coordinates

These values represent the position of the selected element in pixels relative to its parent. An element's origin is the upper left corner of its rectangular frame and establishes position (0,0) for any children of the element.

The position of an element can be changed by dragging it to a new position in the Layout window, or by entering new numbers in these editable fields.

Width and height

The "W:" field shows the width of the selected element. The "H:" field shows its height. Values are in pixels.

The dimensions of an element can be changed by dragging its boundaries in the Layout window or by entering new numbers in these editable fields.

Relative scale

The "W%:" and "H%:" fields display the percentage to which the image has been scaled, in width and height, respectively, relative to the original image.

The dimensions of an element can be changed by dragging its boundaries in the Layout window or by entering new numbers in these editable fields.

Cel Number

The "Cel:" field identifies the current cel number of a selected mToon. The Cel field will be unavailable for other media types.

The up/down arrow buttons to the right of this field allow stepping backward or forward through the animation from a specified cel. These buttons affect the appearance of the animation in edit mode only. They allow the author to easily preview the animation without impacting how it has been configured to appear in run-time.

Layer order number
The "Layer:" field shows the layer order number of the selected element. Use the up/down arrows to the right of this field to select a new layer order number for the element, or enter a new number in this editable field.

If a new layer order number is assigned to an element, and that number has already been assigned to another element, the layer order numbering of elements is updated to accommodate the change. The previous occupant of the layer moves one step higher in the layer order. Any elements with lower layer order numbers move in a similar way.

12. Modifier Reference

12. Modifier Reference

This chapter describes the modifiers built-in to mTropolis. Chapter 13, “Modifier Pop-Up Menus and Message Reference” describes the Message and Message/ Command pop-up menus found in all modifier dialog boxes. Chapter 14, “Miniscript Modifier,” describes the scripting language used by the Miniscript modifier and “If” messenger.

Modifiers Overview

Modifiers are special mTropolis components that modify the properties of other components in a project.

Modifiers are used by dragging them from one of the **Modifier** palettes and dropping them on the object that they are to modify. Each modifier on a **Modifier** palette has unique capabilities or properties. When a modifier is dropped onto a component, the component assumes these capabilities or properties.

For example, a Gradient modifier has the ability to alter the visual characteristics of graphic elements. When dropped onto a graphic element, the Gradient modifier's capabilities are added to the information that makes up that object, as shown in the figure below.

The Gradient modifier, placed on a graphic element

While some modifiers have the ability to change the visible characteristics of the elements onto which they are placed, other modifiers change invisible characteristics, or properties, of the element that contains them. For example, when a Floating-Point Variable modifier that contains the value 2.5 is placed on an element, the physical representation of the element does not change, but its content, the value 2.5, becomes an intrinsic part of the element that contains it.

All modifiers can be configured (that is, their capabilities can be customized) by changing the default settings in their modifier dialog boxes. In addition, most modifiers can be configured to apply their effects at specific times through a process called messaging.

A message in mTropolis can be as simple as a mouse click, or as complex as an author-defined message that is generated only after specific conditions in the run-time environment have been met. Some messages are generated by mTropolis during run-time and automatically sent to specific components throughout the project, and others can be sent to components from special modifiers called messengers.

For details regarding messaging within the mTropolis environment, and instructions on how to configure modifiers and messengers to respond to messages, see Chapter 13, "Modifier Pop-Up Menus and Message Reference."

Modifier Palettes

The **Modifier** palettes (see figures below) contain mTropolis built-in modifiers. Toggle the display of these palettes on and off by selecting **Modifier Palettes ➛ Logic**, **Modifier Palettes ➛ Effects**, **Modifier Palettes ➛ Extras**, or **Modifier Palettes ➛ Network** from the **View** menu. Note that additional sets of modifiers can also be added to mTropolis. If any optional or third-party modifier "kits" have been installed, menu options for those modifier palettes will also appear in the **Modifier Palettes** cascading menu.

Logic Modifier palette

Effects Modifier palette

Extras Modifier palette

Network Modifier palette

Using modifiers

Modifiers can be applied to project elements by dragging modifier icons from the palette and dropping them on their destination. The modifier becomes a child of that component.

Mac OS-only modifiers

Some of the modifiers in this version of mTropolis work only in mTropolis titles built for the Mac OS — they have no effect on titles built for Windows platforms. These modifiers are marked with a yellow dot in the upper left corner of their icons. Be careful when using these Mac OS-only modifiers in a title that will be built for both platforms.

mPire-only modifiers

Some of the modifiers in this version of mTropolis are intended to be used only with the mPire web-browser plug-in. These modifiers work only when the project is built into a title and played inside of a web browser using the mPire plug-in. These modifiers are marked with an orange dot in the upper left corner of their icons. Attempting to execute an mPire-only modifier in the editor's runtime mode causes an error to appear in the Message Log window. See Appendix D, "Internet Authoring Issues," for more information on the mPire plug-ins.

Names of modifiers

To see the names of modifiers in a palette, hold down the Control key while moving the cursor over the palette.

Some notes about the names of modifiers and their use in this documentation is in order:

- The names of variable modifiers all end with the word "variable" ("Boolean Variable," "String Variable"). In the documentation, variable modifiers are often referred to simply as "variables."
- Messenger modifiers all end with the word "messenger" ("Timer Messenger," "Collision Messenger"). In the documentation, messenger modifiers are often referred to simply as "messengers."
- To accentuate the difference between the Behavior modifier, which can contain other modifiers and behaviors, and all other modifiers that cannot, the Behavior modifier is often referred to simply as a "behavior."
- All other modifiers are referred to by their full names ("Path Motion modifier," "Graphic modifier").

Modifier Types

Modifiers in the **Modifier** palettes can be grouped into the following categories. Modifier icons have color-coded backgrounds that help to identify their category.

- *Effects* modify the characteristics of the elements on which they are placed. Effect icons have a green background.
- *Variables* store data values of various types such as text or integers. Variable icons have a purple background.
- *Messenger*, which send messages, commands and data to specific destinations. Messenger icons have a dark red background.
- *Task* modifiers control general project features such as scene navigation, variable values, and data storage. Icons for these modifiers have a blue-grey background.

- *Services* enable new types of functionality in mTropolis. Services may be required for some other modifiers to take effect. For example, the Network Messaging Service must be added to a project for the network messenger to work properly. Services have a purplish-grey background.
- The *Miniscript* modifier provides access to mTropolis scripting language. Depending upon the script used, a Miniscript modifier may act like an effect, messenger, or both. This modifier has a pale green background.
- The *Behavior* modifier encapsulates groups of modifiers and behaviors. A behavior can respond to messages for enabling and disabling its encapsulated modifiers. This modifier has a pale purple background.

Short descriptions of the modifier types and individual modifiers follow. Complete documentation for each modifier can be found (in alphabetical order) at the end of this chapter.

Effect modifiers

In general, modifiers grouped under the category of effects are used to modify visible characteristics of the elements on which they are placed. Modifiers in this category include:

- **Color Table modifier:** Manages color tables (custom color palettes).
- **Cursor modifier:** Changes the mouse cursor.
- **Drag Motion modifier:** Allows end user to drag an element.
- **Element Transition modifier:** Activates an element transition (for example, fade).
- **Gradient modifier:** Creates color gradients in elements.
- **Graphic modifier:** Modifies graphic properties of elements (such as, color, border shape).
- **Image Effect modifier:** Creates various image effects (inverts colors of an element, adds button bevels).
- **MIDI modifier:** Plays and controls MIDI music files or individual notes.
- **Path Motion modifier:** Also listed as a messenger, this modifier can be used to assign motion paths to elements. Additionally, messages can be sent from any point on the motion path.
- **Point Motion modifier:** Moves an element in a straight line from one location to another.
- **Scene Transition modifier:** Specifies the type of transition that will occur when the scene changes (“Zoom”).
- **Simple Motion modifier:** Initiates a simple motion path (“Down” or “Right”).
- **Sound Effect modifier:** Plays sound effects.
- **Sound Fade modifier:** Decreases or increases the volume of a sound.
- **Sound Panning modifier:** Sets the stereo position of a sound.
- **Text Style modifier:** Modifies text styles.
- **Track Control modifier:** Activates and deactivates individual tracks in a QuickTime movie.
- **Vector Motion modifier:** Initiates vector motion in degrees and inches per second.

Variable modifiers

Variables store integers, strings, and other values. When a variable is dropped onto a component and configured, the value contained within it can be referenced by messengers and sent with messages or commands, or referred to by name and used as variables in Miniscript programs.

Like any other modifier, variables can be given any name. For clarity, you should name variable modifiers to reflect their function. If the variable name will be referenced in a Miniscript modifier, it is good programming practice to make the name a single word (or multiple words separated by underscore characters instead of spaces).

There are two classes of variables: simple and compound. Simple variables store a single value while Compound Variables store multiple values.

Variable modifiers include:

- **Boolean Variable:** Stores a true/false value.
- **Compound Variable:** Use this modifier to create custom Compound Variables that can contain any combination of other variables.
- **Floating-Point Variable:** Stores a floating-point value (for example, 3.14159).
- **Integer Range Variable:** Stores an integer range value (such as 4 thru 8).
- **Integer Variable:** Stores an integer value (7). Possible values range from –32767 to 32767.
- **List Variable:** Stores a list of values of any other data type (exept compound or list).
- **Object Reference Variable:** Stores a reference to any object in a project.
- **Point Variable:** Stores a point value (25,45).
- **String Variable:** Stores a string (“Bob Brown”).
- **Vector Variable:** Stores a vector value in degrees and inches per second (33˚ 15.6).

Variable scopes

Where a variable modifier is placed defines its scope (the group of components that can access the variable). Put simply, a variable modifier is accessible to all descendants of its parent. For example, a variable modifier placed on a section is available to any modifier on the section, and all “descendants” of that section.

A variable modifier whose parent is the project is called a global variable. Global variables can be accessed by any modifier in the entire project. See “Variable Scopes” in Chapter 13. For a more localized scope, variables can be aliased and placed on specific objects anywhere in the project.

Messenger modifiers

Messenger modifiers are used to conditionally send specific kinds of information to elements and modifiers.

Messenger modifiers include:

- **Boundary Detection Messenger:** Sends messages or commands after detecting collisions with its element’s parent.
- **Collision Messenger:** Sends messages or commands after detecting collisions with elements.
- **If Messenger:** Sends messages and commands after a condition has been met.
- **Keyboard Messenger:** Detects and responds to keystrokes.

- **Media Cue Messenger:** Sends messages based upon the play position of an element's time-based media. For example, a message can be sent when an mToon starts playing a certain range.
- **Messenger:** The basic messenger modifier. Sends messages and commands in response to an incoming message.
- **Net Messenger:** Similar to the regular messenger modifier, but can send messages to other mTropolis projects over a network (including the Internet).
- **Panorama Messenger:** Detects end user mouse actions over QuickTime VR panorama movies.
- **Path Motion modifier:** Also listed as an effect, this modifier can be used to assign motion paths to elements. Additionally, messages can be sent from any point on the motion path.
- **Timer Messenger:** Sends a message after a given time has elapsed.

Task modifiers

Task modifiers control project features such as scene navigation and data manipulation. Further information on the task modifiers listed below can be found throughout this chapter. Modifiers in this category include:

- **Cache URL modifier:** Stores the contents of a URL in the web browser's cache when used with the mPire web-browser plug-in.
- **Change Scene modifier:** Changes from one scene to another.
- **Fetch Net Text modifier:** Retrieves text from a network URL and stores it in a String Variable.
- **File modifier:** Used to write and read external text files.
- **Navigation modifier:** Changes from one scene to any other scene in a project. Scenes can be selected by name or by certain relative criteria. This modifier provides a superset of the Change Scene modifier's functionality.
- **Open Application modifier:** Launches another application or document.
- **Open Project modifier:** Opens other title or project files.
- **Open URL modifier:** Opens a new URL when used with the mPire web-browser plug-in.
- **Panorama Navigation modifier:** Changes the current view displayed by a QuickTime VR panorama.
- **Post Net Text modifier:** Sends text to a network URL.
- **Print File modifier:** Sends a PICT or BMP file to a printer.
- **Return modifier:** This modifier works in association with a Change Scene modifier with its "add to return list" option selected. The Return modifier returns to the last scene which was added to the return list.
- **Save and Restore modifier:** Writes and reads data values in an external file.
- **Set modifier:** Changes the value of a variable or incoming data to a specified value.
- **Shared Scene modifier:** This modifier changes a standard scene into the shared scene.
- **Window Prefs modifier:** This Mac OS-only modifier can be used to create titles that run in a window, instead of taking over the entire screen.

Services

Services enable special features in mTropolis projects. In this version of mTropolis, there is just one service:

- **Net Messaging Service:** Enables network functionality that allows the use of the network messenger. See "Net Messaging Service" in this chapter.

Behavior modifier

A behavior is a special component in the mTropolis environment. It can be used to encapsulate (contain) groups of modifiers and other behaviors.

Behaviors can be used to group collections of modifiers that work in close concert. Each collection can be enabled or disabled with messages, creating "super modifiers" that provide more complex operations than single modifiers alone.

Like modifiers, behaviors are used by dragging and dropping them onto elements. Modifiers and other behaviors can then be dropped onto them, arranged and configured for use.

The power of behaviors lies in the fact that they can be made "switchable." That is, they can be turned on or off with messages. When a behavior is switched off, all of the modifiers it encloses are disabled. When a behavior is switched on, individual behaviors or modifiers within a behavior can then be activated by incoming messages. This feature allows the author to create and control components with very sophisticated capabilities.

Once they have been configured, the capabilities of fully functioning behaviors can be given to any other component simply by copying and pasting the onto the component.

Behaviors can also be aliased and placed on multiple elements in a project. Since any change made to an alias is automatically made to all other aliases of the same object, aliasing a behavior allows the author to save significant authoring time while providing complete control over multiple elements that share the same capabilities. See "Alias Palette" in Chapter 11.

Behaviors, like all other components in a project, can also be stored in libraries and saved for future use in any other project. See "New, Open, and Save for mTropolis Libraries" in Chapter 1.

Miniscript modifier

Miniscript is a complete scripting language embedded in a modifier. The Miniscript modifier allows you to use a scripting language to create customized modifiers that can:

- Get and set object attributes and variable values.
- Send messages and commands.
- Evaluate mathematical functions.
- Parse and edit string data.
- Evaluate relational expressions and perform conditional branching.

A complete description of the Miniscript language can be found in Chapter 14, "Miniscript Modifier."

Adding Modifiers to Components

Modifiers can be added to components by dragging and dropping them from the **Modifier** palettes.

Modifiers can be placed on objects while in any of the three views in mTropolis: the Layout window, the Structure window or the Layers window. The method for adding modifiers is the same in each case: click, drag, and drop.

Once added to a component, modifiers can be customized by changing the settings in their dialog boxes.

To add a modifier to an element in the Layout window:

1. Click and drag a modifier icon from the **Modifier** palette. An outline of the icon attaches to the cursor.
2. Drop the modifier on the target element. The modifier icon appears in the top left corner of the element (or after any icons already present on the element).
3. Modifiers already present in a project can be moved to other components by dragging and dropping. They can be copied to other components by Option-dragging.

To remove a modifier from an element:
Click on the modifier icon to be removed. Press the Delete key. The modifier icon (and its associated functionality) is removed from the element.

Note: Modifiers can be cut, copied, pasted, or duplicated in any of mTropolis' editing views.

To customize a modifier:
Double-click the icon of a modifier that is attached to some component (that is, not an icon in a modifier palette). The modifier's dialog box appears.

Configuring Modifier Dialog Boxes

The rest of this chapter contains detailed descriptions of the modifiers and their dialog boxes. This section describes features shared by all of the modifier dialog boxes.

Each of the modifiers in mTropolis has settings that can be edited through its dialog box. Display a modifier's configuration dialog box by double-clicking a modifier icon attached to a component.

Common modifier dialog box controls

The figure on the following page shows a typical modifier dialog box, for the **Drag Motion Modifier**. All the modifier dialog boxes are similar and share a number of common fields. These fields are described below. Complete documentation for specific modifiers is found later in this chapter.

Modifier icon
The icon associated with the modifier appears in the upper left corner of the dialog box.

Name field
This editable text field shows the name of the modifier. When first created, modifiers have a default name that is just the modifier type (such as "Graphic Modifier," "Messenger").

Enter a new name for the modifier here. It is good practice to use descriptive names, such as "Blue on mouse down." Descriptive naming makes it easy to find the modifier while in the Structure window. Variable modifiers that will be referenced by Miniscript are easiest to use if they are given a single-word name.

Apply (or Execute, or Enable) When pop-up menu
This pop-up menu displays the message that, when received by the modifier during run-time, causes it to apply, execute, or enable its effect. In other words, this is the message that acts as an "On" switch for the modifier.

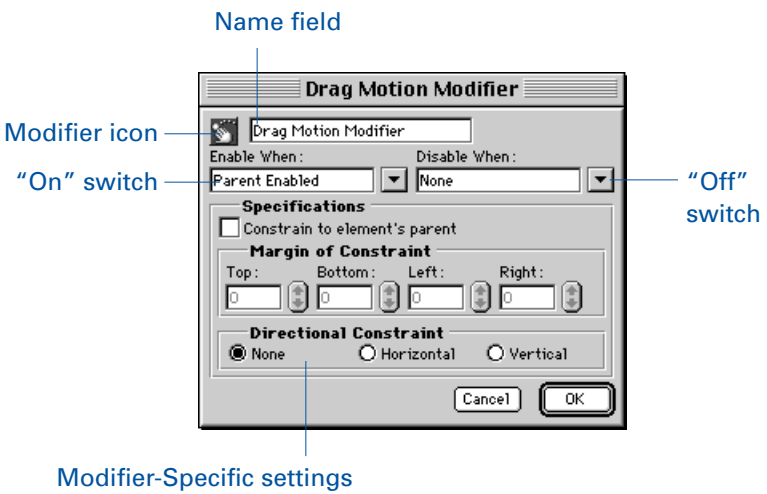
Click the down arrow button to the right of this field to display menu options. Select an item from the list, or simply type in an option and mTropolis will attempt to match it to an item in the menu. If an item cannot be matched, an alert appears, asking if you wish to create a new author message.

For a detailed description of items in this pop-up menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Remove (or Terminate, or Disable) When pop-up menu
This pop-up menu displays the message that, when received by the modifier during run-time, causes it to remove, terminate, or disable its effect. In other words, this is the message that acts as an "Off" switch for the modifier.

Click the down arrow button to the right of this field to display menu options, or simply type in an option and mTropolis will attempt to match it to an item on the menu. If an item cannot be matched, an alert appears, asking if you wish to create a new author message.

For a detailed description of items in this pop-up menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.



*A typical modifier dialog box, the **Drag Motion Modifier** dialog box*

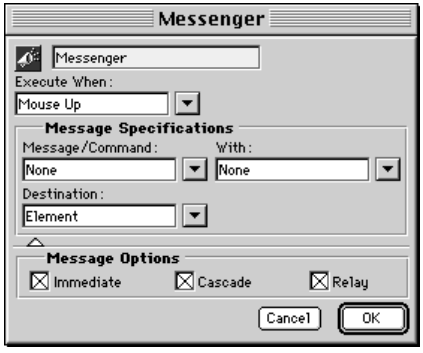
Modifier-Specific settings
Many modifier dialog boxes have settings that can be used to specify the particular action or effect to be applied.

Configuring Messenger Modifiers

Some modifiers are used to send specific kinds of information to other elements and modifiers. Modifiers in this group are called messengers. Although each has unique functionality, there are some similarities among them. The next figure shows a typical messenger dialog box.

All messenger dialog boxes share the same pop-up menus in their Message Specifications section:

Message/Command pop-up menu
Use this pop-up menu to choose the message or command to be sent when the messenger is activated. For a complete discussion of this menu, see "When Pop-Up Menu and Message/ Command Pop-Up Menu Options" in Chapter 13.



*A typical **Messenger** dialog box. The "Message Specifications" and "Message Options" sections are common to all Messenger dialog boxes*

With pop-up menu
Use this menu to choose a value to be sent with the message or command. The choices on this pop-up menu are **None**, **Incoming Data**, and any variables to which the modifier has access. These menu items are described in more detail below:

- **None:** This is the default selection. No value is sent with the message.
- **Incoming Data:** Choose this option to configure the messenger to send the incoming data (that is, the value that arrives with the message that has been configured to trigger the messenger) with the outgoing message or command.
- **Variables:** To send the value of a variable modifier with the outgoing message, choose its name from the submenus available in the second section of the **With** pop-up menu. All variable modifiers currently available to a messenger from its position in the project's hierarchy are shown. These variable modifiers are only accessible to those modifiers or Miniscripts in their "scope."
- **Constant Data Value:** To send a constant data value, highlight the With text field and type the value to be sent. The value should be entered with the same syntax as if it were being used in a Miniscript modifier. The syntax of the expression entered is checked when **OK** is clicked. Any appropriate mTropolis data type can be sent.
- **mToon Ranges:** If the current scene contains an mToon that has named ranges defined, those ranges can be selected from the "Toon Ranges" submenu.

Destination pop-up menu
Use this menu to choose the destination for the message or command sent from this modifier. For a complete discussion of this menu, see "The Destination Pop-Up Menu" in Chapter 14.

Message options
All **Messenger** dialog boxes have a Message Options section that can be revealed by clicking the triangle at the bottom of the dialog box. Options in this section are:

Cascade check box
By default, the message "cascades" from the targeted destination to all components in the hierarchy below it. Uncheck this option to configure the messenger to send its message only to the modifiers in a targeted component. "Cascade off" is equivalent to the path of an environment message sent by mTropolis.

!!! Note: This option has no effect when sending a command. Commands act only on the targeted element.

Immediate check box
By default, messages or commands are sent immediately when the messenger is activated. Uncheck this box to configure the messenger to send its message or command only after the current thread of messages has ended. This option can be used to avoid system overload if the message queue becomes too deep.

Relay check box
By default, messages travel from component to component in the hierarchy activating any and all modifiers that respond to the message. Uncheck this box to activate only the first modifier in the path of the message that is configured to respond. In effect, the first modifier to respond to the message "swallows" the message.

!!! Note: This option has no effect when sending a command. Commands act only on the targeted element.

Behavior

A behavior is a special component in the mTropolis environment. It is used to encapsulate (contain) other modifiers and behaviors.

Behaviors can be used to hierarchically group collections of modifiers that work in close concert. Each collection can be enabled or disabled with messages, creating "super modifiers" that provide more complex operations than single modifiers alone.

Like modifiers, behaviors are used by dragging and dropping them onto components. Modifiers and other behaviors can then be dropped onto them, and arranged and configured for use. The power of behaviors lies in the fact that they are switchable, that is, they can be turned off or on with messages.

When a behavior is switched off, all of the modifiers and behaviors it encloses cannot be activated by incoming messages. When a behavior is switched on, individual behaviors or modifiers within a behavior can be activated by incoming messages. This feature allows the author to create and control components with highly sophisticated capabilities.

Once they have been configured, the capabilities of fully functioning behaviors can be given to any other component, simply by copying and pasting the behavior onto another component.

Behaviors can also be aliased and placed on other elements in a project. Since any change made to an alias is automatically made to all other aliases of the same object, aliasing a behavior allows the author to save significant authoring time while providing complete control over multiple elements that share the same capabilities. See "Alias Palette" in Chapter 11.

Behaviors, like all other components in a project, can also be stored in libraries and saved for future use in any other project. See "New, Open, and Save for mTropolis Libraries" in Chapter 1.

Note: Behaviors are often used simply as a tool to organize an element's modifiers into groups.

Behaviors and components

Behaviors, and all the modifiers they contain, are always associated with a single structural component.

A structure view of an element that contains behaviors

To illustrate, in the figure above, Element 1 is the parent of Behavior 1 and Messenger 1, and Behavior 1 is the parent of Behavior 2 and grandparent of Messenger 2. Both messengers and behaviors are, however, associated with Element 1, which is the first structural component at the root of the modifier hierarchy.

This association is important to remember when targeting the recipient of a message from messengers within behaviors.

For example, in the figure above, when "Element's Parent" is the destination chosen for the message sent from Messenger 1, it will send its message to its associated element's parent, Scene A. This is because Element 1 is the first element at the root of the modifier hierarchy, and Scene A is this element's parent.

Using behaviors

To create a new behavior:

- Drag a Behavior modifier from the **Modifier** palette (or a library) to an element.
- Open the behavior by double-clicking it.
- Add modifiers to the behavior by dragging them into the Behavior window.
- Optionally, add modifiers to a behavior simply by dragging and dropping the modifier icons onto a closed behavior icon. When the behavior is opened, modifiers in the Behavior window can be repositioned by dragging them.

Message passing within behaviors

As long as a behavior is switched "on," all messages are passed to all modifiers and behaviors inside the behavior. Any modifier or behavior configured to respond to these messages will be executed.

Messages are passed to modifiers and behaviors within a behavior according to their message passing order. This order is visible in the Structure window and inside the Behavior window. For more information on changing

the execution order of components in a behavior, see "Messaging Order Numbers of Modifiers" later in this chapter.

Components of the Behavior dialog box
Controls in the **Behavior** dialog box are described below.

Modifier's Name field
This editable text field can be used to change the behavior's name.

Modifier icon
This icon identifies the modifier's type.

Switchable check box
When checked, this check box allows the selection of enable and/or disable messages for the behavior. Check this box to create behaviors whose component modifiers are active only after the receipt of the specified "Enable When" message and are deactivated after the receipt of the specified "Disable When" message. Note that non-switchable behaviors (that is, behaviors that do not have this check box checked) are always enabled.

Enable When pop-up menu
This pop-up menu can be used to select the message that enables this behavior. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13. This menu is only available when the **Switchable** check box is checked.

When the enabling message is received by the behavior, all modifiers inside that behavior receive the "Parent Enabled" message and become active.

Note: With the exception of the Parent Enabled message, any message that is used to enable a switchable behavior is not broadcast to the components it contains. To enable a behavior and activate the modifiers it contains on a single message, use Parent Enabled.

Disable When pop-up menu
This pop-up menu can be used to select the message that disables this behavior. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13. This menu is only available when the **Switchable** check box is checked.

When the disabling message is received by the behavior, all modifiers inside that behavior receive the "Parent Disabled" message and become inactive.

Names of modifiers
The name of a any modifier, shown below its icon, can be changed in the behavior window by clicking on the name and entering a new one.

Messaging Order Numbers of modifiers
These numbers, shown in parentheses to the right of a modifier's icon, denote the messaging order of modifiers within a behavior.

The messaging order of components can be changed by clicking the Messaging Order number. Hold down the mouse until a pop-up menu appears. Select **Do Sooner** or **Do Later** to move the modifier one position forward or backward in the messaging order.

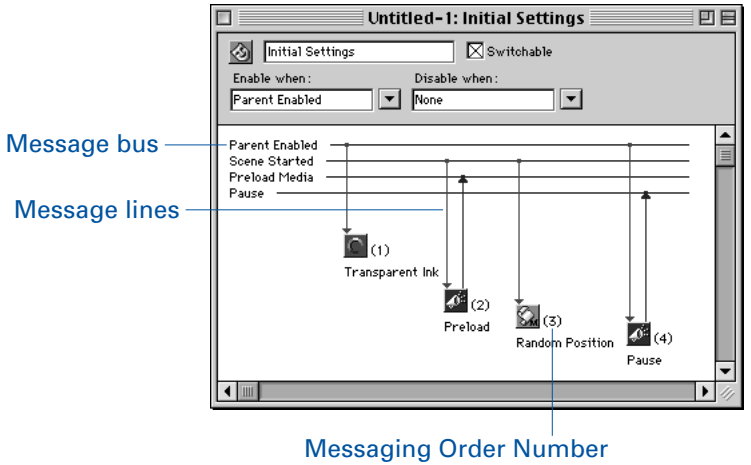
Alternatively, the messaging order of modifiers and behaviors in a behavior can be changed by dragging and dropping them into new positions in the behavior via the Structure window.

Message bus
All messages received and/or sent by modifiers in a behavior are shown in this list. The horizontal lines extending from the messages in this list represent a message "bus" that the modifiers are "listening" to. Note that this display is not a timeline. Click on a message name to highlight its line so that the path of the message in the behavior can be seen more easily.

Message lines
These lines show the paths of messages to and from modifiers within the behavior. These lines connect to the message bus to show which modifiers receive and/or send certain messages.

On color monitors, vertical message lines to and from modifiers and messages are displayed in three different colors:

- Green message lines are drawn from the message bus to a modifier, indicating a message that executes, applies, or enables the modifier.
- Blue message lines are drawn from a modifier or behavior to the message bus, indicating that the modifier sends this message when activated.
- Purple message lines are drawn from the message bus to a modifier to indicate that the message terminates, removes, or disables a modifier. These lines are also drawn with an "open" (unfilled) arrow head.

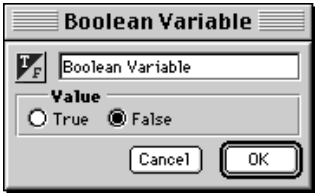


Behavior Modifier *dialog box*

Boolean Variable

The Boolean Variable modifier stores a boolean (true/false) value. It is useful for keeping track of objects or properties that have only two states. For example, whether an end user has touched a certain object could be stored in a Boolean Variable modifier; depending on the value of that variable, the end user may or may not be able to do something else in the project.

Components of the **Boolean Variable** dialog box (see figure above) are described below:



*The **Boolean Variable** dialog box*

Variable's Name field
This editable text field can be used to change the variable's name. If the variable will be referenced in a Miniscript modifier, it is good programming practice to make the name a single word.

Modifier icon
This icon identifies the variable modifier's type.

Value buttons
Enter the variable's initial value here by selecting either the **True** or **False** button. The default is **False.** The value can be changed during run-time. See "Setting Values of Variable Modifiers" in Chapter 14 for details regarding getting and setting the value of this variable modifier.

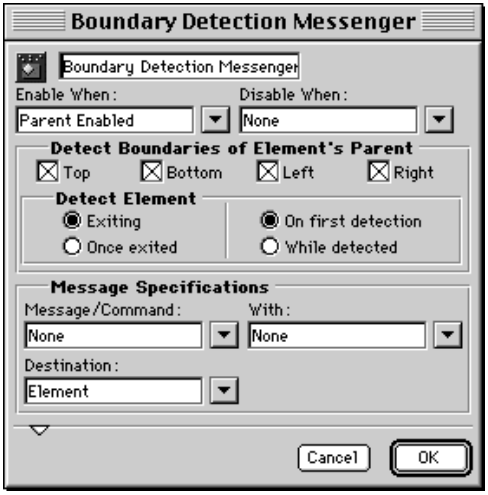
Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Boundary Detection Messenger

The Boundary Detection Messenger detects the relative position of elements. It can be configured to send a message when an object has left or touched the frame of its parent. For information regarding parent/child relationships, see "Parent/Child Tool" in Chapter 11.

Components of the **Boundary Detection Messenger** dialog box are described below:



*The **Boundary Detection Messenger** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Enable When pop-up menu
Use this pop-up menu to select the message that enables boundary detection. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Disable When pop-up menu
Use this pop-up menu to select an optional message that disables boundary detection. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Detect Boundaries of Element's Parent check boxes
These four check boxes (**Top**, **Bottom**, **Left**, and **Right**) specify the boundaries of the parent element that generate messages when the child touches them. By default, all boundaries are sensitive.

Note: The individual boundaries of the element's parent are not clipped by each other. For example, the left boundary is the line that runs vertically along the left side of the parent element. This boundary extends for the entire height of the screen. Therefore, if a Boundary Detection modifier is configured to detect the boundary of just one side of the parent element (for example only the **Left** check box is selected), a child element could move in some other direction (up past the top boundary of the parent) and, sometime later, move past the left boundary triggering the messenger even though the child is not "inside" the parent.

Detect Element section
The buttons in this section control how the child element's boundary crossing is detected:

- **Exiting:** Click **Exiting** to send the specified message or command when the element is exiting its parent's boundary, but there is still contact between the child and the boundary.
- **Once Exited:** Click **Once Exited** to send the messenger's specified message or command when the element has completely left its parent's frame.
- **On First Detection:** Click **On First Detection** to send the specified message or command when the edge of the element first touches the frame of its parent (if **Exiting** is checked) or when the element first exits (if **Once Exited** is checked).
- **While Detected:** Click **While Detected** to send the specified message or command each time the child moves and is touching (if **Exiting** is checked) or outside of (if **Once Exited** is checked) the parent's frame.

Message specifications
Use this section to specify the message to be sent when a boundary contact is detected.

See "Configuring Messenger Modifiers" or "Message Specifications" in this chapter for a complete description of the controls in this section.

Message options
Click the triangle at the bottom of the dialog box to display the Message Options section. See "Message Options" in this chapter for a complete description of the controls in this section.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

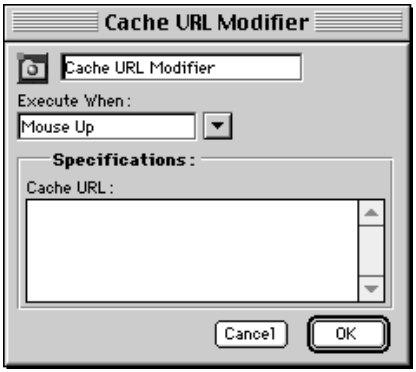
Cache URL Modifier

The Cache URL modifier retrieves data from a specified URL and stores it in the web browser's cache for quick retrieval via the Open URL modifier (described in this chapter).

There is no guarantee that the cached file will be in the browser's cache when the file is needed, as the browser may have flushed it. Browser cache settings are maintained by the browser itself and individual end users may have different cache settings.

Note: This modifier has no effect when used in the mTropolis editor's run-time mode or in the standard mTropolis players. It will only take effect once the project has been built into a title and run in the mPire plug-in player. When this modifier is activated in the editor's run-time mode, an error message is displayed in the Message Log.

Components of the **Cache URL Modifier** dialog box are described below.



*The **Cache URL Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to select the message that causes the data at the specified URL to be to be retrieved and stored in the network browser's cache. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Once the data at the specified URL is successfully cached, mTropolis sends a Net Operation Completed message to the modifier's parent. See "Net Operation Completed" in Chapter 14.

Cache URL Modifier Attribute	Type	Get	Set	Description
stop	Boolean		•	Set this attribute to true to cause caching-in-progress to stop.
url	String	•	•	Gets or sets the URL that will be opened by the modifier, just as if it had been set in the "Cache URL" field. For example: `set mycache.url to \ "http://www.mfactory.com/"`

Unique Attributes of the Cache URL modifier

If caching cannot be completed, mTropolis sends a Net Operation Failed message to the modifier's parent. See "Net Operation Failed" in Chapter 13.

If the caching process is stopped (by setting the modifier's stop attribute to true), mTropolis sends a Net Operation Stopped message to the modifier's parent. See "Net Operation Stopped" in Chapter 13.

Cache URL field
Enter a URL in this text field. The contents of the path field must be a valid URL. The syntax of this path is not checked by the modifier. The URL itself is not resolved until the modifier receives the specified "Execute When" message.

Any protocol supported by the Netscape browser can be accessed by this modifier (http, file, ftp, mailto, news, etc.). For file access protocols such as ftp, the end user will need a valid username and password if the referenced server is protected.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

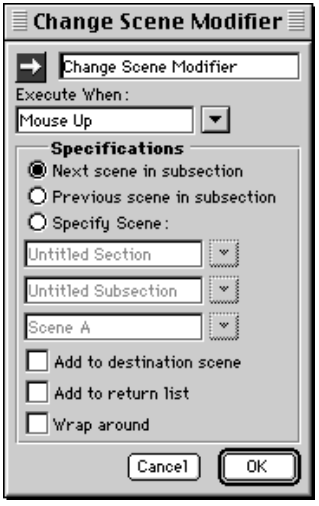
OK button
Click **OK** to accept changes made to this modifier.

Miniscript-accessible attributes of the Cache URL modifier
The Cache URL modifier has a number of attributes that can be accessed via Miniscript. For general information about using attributes, see Chapter 15, "Attributes." The table below describes unique attributes of the Cache URL modifier in a format similar to that found in "Lists of Attributes by Component Type" in Chapter 15.

Change Scene Modifier

The Change Scene modifier executes a scene change when triggered in run-time mode. The previous scene, next scene, or a specific scene in the project can be selected. Note that the order of scenes in a subsection can be easily changed in the Layers window — see "Changing the Order of Scenes" in Chapter 10. A more complex and flexible modifier for changing scenes is described in Navigation Modifier later in this chapter.

Components of the **Change Scene Modifier** dialog box are described below:



*The **Change Scene Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that triggers the scene change. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Specifications section
This section of the dialog box can be used to select the scene to display when the Change Scene modifier is triggered.

Specifications buttons
Choose the type of scene:

- **Next Scene in Subsection:** Click this option to change to the next scene in the current subsection.
- **Previous Scene in Subsection:** Click this option to change to the previous scene in the current subsection.
- **Specify Scene:** Click this option to activate the **Section**, **Subsection**, and **Scene** pop-up menus. Use the pop-up menus to select a specific scene anywhere in the project.

Specifications check boxes
Choose scene change options:

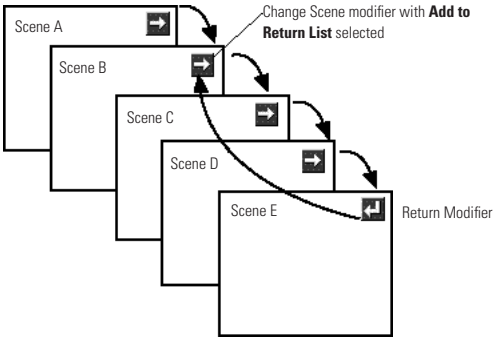
- **Add to Destination Scene:** Check this check box to perform a special scene change in which the currently displayed scene is still visible after the change to the new scene. That is, the current scene acts like a shared scene for this scene change. Note that checking this option automatically checks the **Add to Return List** option as well.

  When this option is checked and a scene change is triggered, the Scene Deactivated message is sent to the "original" scene. See "Scene Deactivated" in Chapter 13. If the Return modifier is used to return from the new scene to the original scene, the Scene Reactivated message is sent to the original scene. See "Scene Reactivated" in Chapter 13.

  Tip: The **Add to Destination Scene** option can be used to simulate dialog boxes and alerts. Put elements and artwork that comprise the dialog box or alert on the destination scene and ensure that this option is checked. When the scene is changed, it looks as though the dialog box has appeared "in front" of the previous scene. The Return modifier can be used to "dismiss" the dialog box or alert.

- **Add to Return List:** Check this check box to make the current scene the one returned to by the next Return modifier in a series of scenes. For example, in the figure below, Change Scene modifiers have been placed on each of the scenes. Each of these modifiers has been configured to change to the next scene. Scene B's scene Change Modifier has the **Add to Return List** option selected. A Return modifier has been placed on Scene E. When this modifier receives a message that will activate it during runtime, Scene E will change to Scene B (the arrows shows the sequence of scenes).



*The **Return List***

- **Wrap Around:** By default, the first and last scene in a subsection have no connection in the scene order. That is, the last scene in a subsection has no next scene and the first scene in a subsection has no previous scene.

  When this check box is marked, a change to the "Next Scene," executed from the last scene in a subsection, "wraps around" the list of scenes and changes to the first scene in the subsection.

- Similarly, a change to the "Previous Scene," executed from the first scene in a subsection, "wraps around" the list of scenes and changes to the last scene in the subsection.

Cancel button

Click **Cancel** to ignore changes made to this modifier.

OK button

Click **OK** to accept changes made to this modifier.

Collision Messenger

The Collision Messenger is used to detect when elements are in the same space. The Collision Messenger sends a message when its element touches another element.

Tip: Collision detection requires a significant amount of processor overhead. Occasionally, mTropolis may seem to "miss" detecting a collision between elements, especially when an element is moving quickly. These missed collisions happen because of interrupts by the operating system. To improve performance of the Collision Messenger, consider setting the gameMode attribute (see Chapter 15) to true. Enabling gameMode disables many system functions such as file and network access, giving more processing time to the mTropolis title.

Components of the **Collision Messenger** dialog box are described below.

*The **Collision Messenger** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Enable When pop-up menu
Use this pop-up menu to select the message that enables collision detection. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Disable When pop-up menu
Use this pop-up menu to select an optional message that causes collision detection to be disabled. For a complete discussion of this menu, see "When Pop-Up Menu and Message/ Command Pop-Up Menu Options" in Chapter 13.

Collide With pop-up menu
Choose the type of elements whose collision will trigger a message. There are two options.

- **Any Element:** Choose this option to send a message upon collision with any element.
- **All Except Parent(s):** Choose this option to send a message upon collision with any element except the messenger's parent.

Detect Layer check boxes
By default, the collision messages are generated for collisions with elements in any layer.

- **Front:** Deselect the **Front** check box to disable collisions with objects with a higher layer order than the element (that is, those objects that are "in front" of the element).
- **Behind:** Deselect the **Behind** check box to disable collisions with objects with a lower layer order than the element (those objects that are "behind" the element).

Note: If both boxes are deselected, messages will never be generated, since each object has a unique layer order number.

Detect Elements section
The buttons in this section control when messages are sent during the collision process:

- **On First Contact:** Click this option to send the specified message or command when an object enters or makes contact with the element's frame.
- **While in Contact:** Click this option to send the message or command repeatedly while a moving object is in contact with the element.
- **Exiting:** Click this option to send the message or command when a colliding object leaves the element's frame.

Message Specifications Section
Use this section to specify the message to be sent when a boundary collision is detected. See "Configuring Messenger Modifiers" or "Message Specifications" in this chapter for a description of the common controls in this section.

The **Collision Detection** dialog box has a number of unique message specifications that control the destination of messages sent by the Collision Messenger:

To Collision Element(s) button
When this option is clicked, messages are sent only to the objects that collide with the element.

First Element Only check box
This option is only available when the **To collision element(s)** button is clicked. When the **First Element Only** check box is chosen, a message is sent only to the first object that collides with the modified element.

To Other Destination button
Click this option to send collision-generated messages to other destinations. A **Destination** pop-up menu is activated. The selected destination is the only recipient of messages generated by collisions with the element.

Message options
Click the triangle at the bottom of the dialog box to display the Message Options section. See "Message Options" in this chapter for a complete description of the controls in this section.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

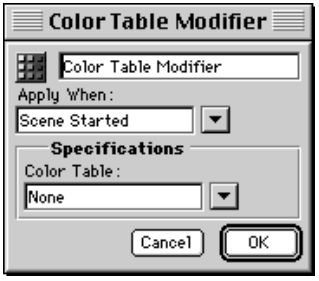
Color Table Modifier

Often, 8-bit graphic media (256 color media) are designed using a color table other than the standard Mac OS system color table. Though any number of color tables can be used in a project, only one color table can be displayed at a time. The Color Table modifier allows you to define which color table is to be active at any one time.

Note: Color Table modifiers are only necessary when building 256 color (8-bit) titles. In higher bit-depth titles, mTropolis displays graphics properly without needing Color Table modifiers. In this case, color information is retrieved from the media files themselves and correctly displayed in thousands or millions of colors. See "Color Depth Pop-Up Menu" in Chapter 2 for more information about the color bit-depth of mTropolis titles.

A complete description of mTropolis color table handling follows the description of the Color Table modifier.

Components of the Color Table Modifier dialog box
Components of the **Color Table Modifier** dialog box are described below:



*The **Color Table Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Apply When pop-up menu
Use this pop-up menu to choose the message that applies the Color Table modifier. For a

complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Color Table pop-up menu
Use this pop-up menu to choose the name of a color table file that has been linked to the current project. New color tables can be linked by choosing **Link Color Table.** A standard file dialog box appears. Choose a CLUT file to be linked.

A set of CLUT files that contain commonly-used color tables (Windows System Palette, Grayscale Palette, etc.) can be found in the Color Tables folder of the mTropolis distribution.

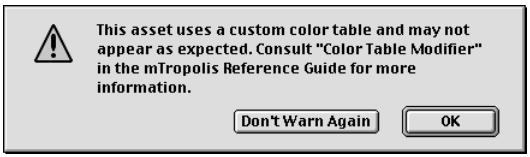
Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

How mTropolis Handles Color Tables

By default, mTropolis projects use an 8-bit offscreen buffer. That is, projects are designed to be run on 8-bit (256 color) displays. This option is a project preference that can be changed (see "Color Depth Pop-Up Menu" in Chapter 2). In a 256-color project, color tables (also called color palettes) are used to define which 256 colors — out of the total range of visible colors — can be used.

When an 8-bit image that uses a color table other than the Mac OS 8-bit system color table is linked to a 256 color project, the alert shown below is displayed:



This alert appears because, by default, mTropolis uses the Mac OS 8-bit system color table to display images. Images that use custom color tables may appear wildly different when displayed using the Mac OS system color table.

Applying color tables in run-time mode

To make the image display properly, put a Color Table modifier in the same scene as the image. Link the appropriate color table to the modifier and configure the modifier as described below. When activated during runtime, the chosen color table will be applied to the scene and the graphic will display properly.

Applying color tables in edit mode

By default, mTropolis uses the Mac OS 8-bit color table to display graphics during edit mode. If other color tables have been linked to the project (through the use of the Color Table modifier or **Asset** palette), those color tables can be used to display graphics in edit mode by choosing them from the **View ➛ Preview Color Table** cascading menu. See "Preview Color Table" in Chapter 7.

Creating custom color tables

To create a custom color table for an 8-bit image, use an image editing application to extract the color table and save it as a CLUT file. To save a CLUT file from Adobe Photoshop, use the following steps:

1 Launch Photoshop and open the desired 8-bit image.

2 Choose **Color Table** from Photoshop's **Mode** menu. The **Color Table** dialog box appears, showing the custom color table used by the image.

3 Click the **Color Table** dialog box's **Save** button. A file selection dialog box appears. Enter a name for the color table and click **Save.** The color table is saved as a CLUT file, suitable for use with mTropolis.

To use a custom color table with multiple images or entire projects, it is usually best to create the color table first, then create images using only the colors in that color table. Alternatively, existing graphics can be re-mapped to the custom color table. Programs such as Equilibrium's DeBabelizer include features for creating optimal color tables from multiple 8-bit images.

Color tables and projects that use thousands or millions of colors

When linked to a project that supports thousands or millions of colors (see "Color Depth Pop-Up Menu" in Chapter 2), 16 color (4-bit) and 256 color (8-bit) images display properly without having to use the Color Table modifier. mTropolis retrieves color information from the media file and displays the image with its correct colors.

Compound Variable

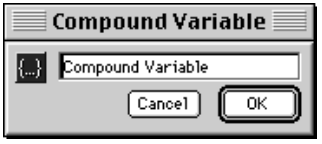
The Compound Variable modifier can be used to organize a number of other mTropolis variables into a single, user-defined variable. The fields of custom Compound Variables created with this modifier can be accessed from Miniscript in the same way as mTropolis' built-in compound data types. See "Variables" and "Accessing the Fields of Compound Variables" in Chapter 14.

To use the Compound Variable modifier, place it on a component, then drag variables onto its icon. The variables will seem to "disappear" into the Compound Variable. To view the contents of the Compound Variable, switch to the structure view and toggle the Compound Variable's open/close triangle to reveal the variables that have been placed inside.

The name given to the Compound Variable is its "top-level" name. The names of the individual variables put inside the Compound Variable become the names of the variable's fields. See the example below.

!!! Note: Compound Variables can be nested inside other Compound Variables to create complex data hierarchies. Like any other variable, the Compound Variable modifier can also be sent or received as data by any messenger in its scope.

Components of the **Compound Variable** dialog box are described below.



*The **Compound Variable** dialog box*

Variable's Name field

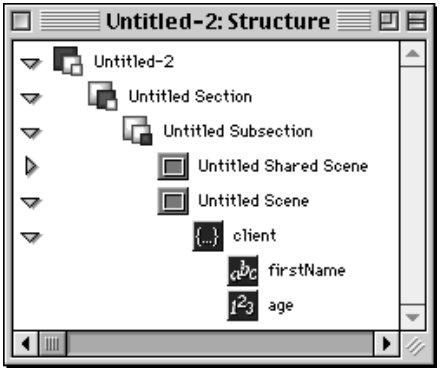
This editable text field can be used to change the variable's name. This name becomes the "top-level" name of the Compound Variable. Any variables dropped into the Compound Variable become "fields" of the custom Compound Variable. See the example below.

Modifier icon
This icon identifies the modifier's type.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Example
Consider the structure view of the Compound Variable shown in the figure below. The Compound Variable has been given the name "client." There are two variables that have been placed inside the Compound Variable — a String Variable and an Integer Variable. The names of these variables, "name" and "age," become the names of the Compound Variable's fields.



Structure view of example Compound Variable

Values in the Compound Variable can be accessed through Miniscript statements. For example, the entire compound value can be accessed by using its "top-level" name in a Miniscript statement:

```
send "NewClient" with client
```

The values of individual variables within the Compound Variable can be accessed just like the fields of "built-in" Compound data types using the "." syntax. In our example, the Miniscript expression `client.firstname` resolves to the string value contained within the "firstName" String Variable. Similarly, the expression `client.age` resolves to the integer value contained in the "age" Integer Variable.

These values could be used with any type of Miniscript statement. The following examples illustrate just a few of the possibilities:

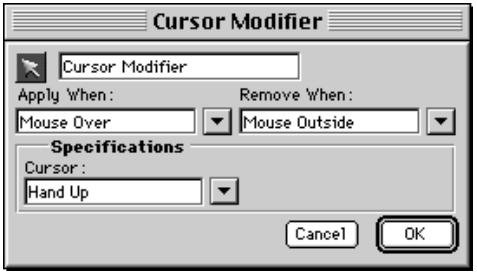
```
Set the value of a variable from
one of the fields:

set mytextvar to client.firstName

Change the value of a field:
set client.age to 27

Perform a comparison using a field:
if client.age < 21 then \
  send "No Cocktails for You"

Set the value of an attribute
from a field:

set mytextelement.text to \
  "Good job, " & client.firstName
```

Miniscript statements are described in Chapter 14, “Miniscript Modifier.”

Cursor Modifier

The Cursor modifier changes the active mouse cursor on receipt of a specified message. Its effects are only visible during run-time.

Components of the **Cursor Modifier** dialog box are described below:



*The **Cursor Modifier** dialog box*

Modifier’s Name field
This editable text field can be used to change the modifier’s name.

Modifier icon
This icon identifies the modifier’s type.

Apply When pop-up menu
Use this pop-up menu to choose the message that causes the cursor to change. The default is Mouse Over, which makes the cursor change when it is moved over the element. The cursor remains changed until the message specified in the **Remove When** pop-up menu is received, or another Cursor modifier is activated.

Note: The newly-specified cursor is not actually applied immediately after the receipt of the message specified in this pop-up menu. The cursor must be moved before the change takes effect. When the Cursor modifier is applied upon receipt of a message such as Mouse Over, this behavior is not noticeable. However, in situations where this behavior is a problem, the cursor can be forced to change without having to wait for end user mouse actions. See “refreshCursor” in Chapter 15.

For a complete discussion of this menu, see “When Pop-Up Menu and Message/Command Pop-Up Menu Options” in Chapter 13.

Remove When pop-up menu
Use this pop-up menu to choose the message that causes the cursor to revert to the default system cursor. Note that when multiple Cursor modifiers are applied sequentially, the cursor does not revert to the previously-applied cursor, but always reverts to the system cursor.

For a complete discussion of this menu, see “When Pop-Up Menu and Message/Command Pop-Up Menu Options” in Chapter 13.

Cursor pop-up menu
Use this pop-up menu to choose a cursor to be changed to when the specified “Apply When” message is received. This list contains the names of many different cursor selections.

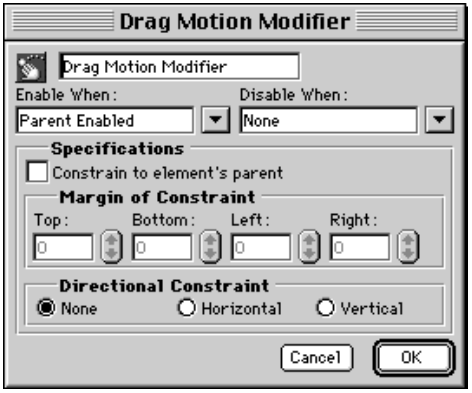
Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Drag Motion Modifier

The Drag Motion modifier allows an element to be dragged around the screen by the end user during run-time.

Components of the **Drag Motion Modifier** dialog box are described below:



*The **Drag Motion Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Enable When pop-up menu
Use this pop-up menu to choose the message that enables dragging of the element. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Disable When pop-up menu
Use this pop-up menu to choose the message that disables dragging of the element. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Specifications section
Use the controls in this section of the dialog box to set drag motion options.

Constrain to Element's Parent check box
Check this box to make the object draggable only within the limits of its parent's frame.

Margin of Constraint fields
These options are available when "Constrain to Element's Parent" is selected. They set a "margin" inside the frame of the element's parent that constrains drag motion even more. Margins can be set for the Top, Bottom, Left, and Right sides of the parent frame. Margins are measured in pixels from the parent's borders. For example, if 10 is entered into the "Top:" margin of constraint field, the dragged element can never be dragged closer than 10 pixels from its parent's top border.

Directional Constraint buttons
Use these options to constrain drag motion to a single direction. Options are:

- **None:** The default. Choose this option to allow dragging in any direction.
- **Horizontal:** Choose this option to allow horizontal dragging only.
- **Vertical:** Choose this option to allow vertical dragging only.

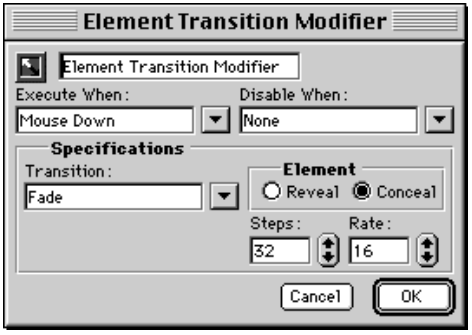
Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Element Transition Modifier

The Element Transition modifier can be used to creatively hide or reveal an element. Like the Scene Transition modifier (discussed later in this chapter), the **Transition** pop-up menu lists optional effects, such as fade and rectangular iris.

Components of the **Element Transition Modifier** dialog box are described below:



*The **Element Transition Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that enables the element transition. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Note that mTropolis sends a Transition Started message when an element transition begins and a Transition Ended message when the transition has finished. See "Transition Started" and "Transition Ended" in Chapter 13.

Disable When pop-up menu
Use this pop-up menu to choose the message that disables the element transition. Note that a transition cannot be cancelled while it is being played. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Specifications section

Use the controls in this section to select the type of element transition.

Transition pop-up menu
Select a transition type from this menu. Options include:

- **Fade**
- **Rectangular Iris**
- **Zoom**
- **Oval Iris**

Element buttons
The transition can reveal a hidden element or hide a visible element:

- **Reveal Element:** Choose this option to reveal a currently hidden element. If this option is chosen for a visible element, the element will be hidden at the start of the transition.
- **Conceal Element:** Choose this option to hide a currently visible element.

Steps field
Use this field to set the number of steps between the start and finish of the transition. The greater this number, the finer (and slower) the transition.

Rate field
Use this field to select the speed of the transition in steps per second. At a rate of 60, a transition set to 60 steps would take 1 second.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

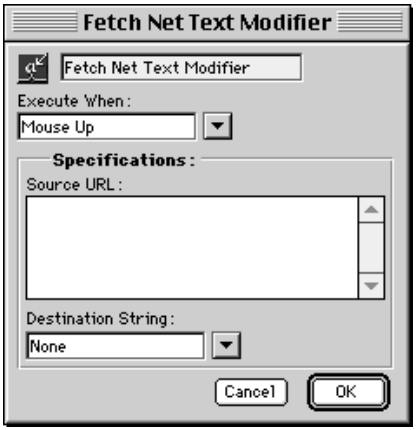
OK button
Click **OK** to accept changes made to this modifier.

Fetch Net Text Modifier

The Fetch Net Text modifier retrieves text from a specified URL and stores it in a String Variable.

This modifier is intended for use with URLs that contain plain ASCII text. While URLs that contain HTML can also be fetched (because HTML code is written as plain text), mTropolis itself cannot perform any special operations on raw HTML code (for example, display HTML in a text element with proper formatting). To display the contents of a URL in a browser window, use the Open URL modifier described later in this chapter.

Components of the **Fetch Net Text Modifier** dialog box are described in the folling column.



*The **Fetch Net Text Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that causes text to be fetched from the specified URL. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Once the text at the specified URL is successfully stored in the specified variable, mTropolis sends a Net Operation Completed message to the modifier's parent. See "Net Operation Completed" in Chapter 13.

If text cannot be fetched from the specified URL, mTropolis sends a Net Operation Failed message to the modifier's parent. See "Net Operation Failed" in Chapter 13.

If the fetching process is stopped (by setting the modifier's stop attribute to true), mTropolis sends a Net Operation Stopped message to the

modifier's parent. See "Net Operation Stopped" in Chapter 13.

Specifications section

Use the controls in this section to specify the source URL and destination string for fetched text.

Source URL field

Enter a URL in this text field. The contents of the path field must be a valid URL. For example:

```
http://www.mydomain.com/mytext.txt
```

The syntax of this path is not checked by the modifier. The URL itself is not resolved until the modifier receives the specified "Execute When" message.

Only the http and ftp protocols are supported by this modifier. For the ftp protocol, the end user will need a valid user name and password if the referenced server is protected.

Destination String pop-up menu

Use this pop-up menu to select the String Variable in which the fetched text will be stored. Options include:

- **None:** This is the default. No String Variable is selected. Text will not be fetched.
- **Incoming Data:** Choose this option to store the fetched text in a String Variable sent as incoming data (that is, the value that arrives with the message that has been configured to trigger the Fetch Net Text modifier). This option will only work properly if a String Variable is sent as incoming data — fetched text cannot be stored in a literal string (for example "my literal text string") or non-String Variable (such as an Integer Variable).
- **String Variable:** To store fetched text in a String Variable, choose its name from the submenus available in the second section of the **Destination String** pop-up menu. Any String Variable modifier within the "scope" of the Fetch Net Text modifier can be chosen.

Cancel button

Click **Cancel** to ignore changes made to this modifier.

OK button

Click **OK** to accept changes made to this modifier.

Miniscript-accessible attributes of the Fetch Net Text modifier

The Fetch Net Text modifier has a number of unique attributes that can be accessed via Miniscript. For general information about using attributes, see Chapter 15, "Attributes." The table below describes unique attributes of the Fetch Net Text modifier in a format similar to that found in "Lists of Attributes by Component Type" in Chapter 15.

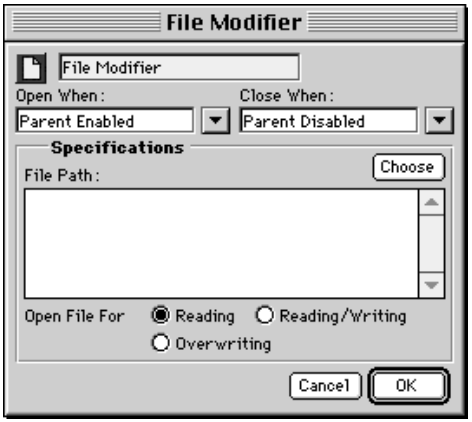
Fetch Net Text Modifier/Attribute	Type	Get	Set	Description
stop	Boolean		•	Set this attribute to true to stop a Fetch Net text operation in progress. See "Net Operation Stopped" in Chapter 13.
url	String	•	•	Gets or sets the URL that will be fetched by the modifier, just as if it had been set in the "Source URL" field.

Unique attributes of the Fetch Net Text modifier

File Modifier

The File modifier can be used to open, read, write, and close ASCII text files.

Components of the **File Modifier** dialog box are described below.



*The **File Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Open When pop-up menu
Use this pop-up menu to choose the message that causes the specified file to be opened. If the file does not already exist, it will be created. The new file's Mac OS type is specified by the macType attribute (which defaults to "TEXT") and its creator is specified by the macCreator attribute (which defaults to "ttxt"). For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Close When pop-up menu
Use this pop-up menu to choose the message that causes the specified file to be closed. Files are also closed automatically when the File modifier is unloaded from memory (when the scene changes). For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Specifications section
Use the controls in this section to define the file to be operated on and the mode of operation (read, read/write, or overwrite).

File Path field
Use this field to specify the full path to the file to open. Enter a path as described below, or click the **Choose** button to choose a file using a standard file dialog box.

Paths specified in this way are not resolved until run-time, so a file that does not currently exist can be specified. Note also that the full path name is limited to a maximum of 255 characters.

- **Entering the Text of an Absolute or Relative Path:** Enter an absolute or relative path to a file in this field. This path must be specified using the standard colon-delimited Mac OS path syntax. The path is translated as appropriate for the current platform at run-time.

  An absolute path can be specified by starting with the name of the hard disk volume and continuing down through the disk hierarchy. For example:

  ```
  My HD:My Folder:MyFile
  ```

A path relative to the location of the currently running project or title can be specified by using colons to move up the filesystem hierarchy. For example, to indicate a project in the same folder as the currently running project, use the syntax:

```
:MyFile
```

To indicate a file in the next folder or volume up the hierarchy, use the syntax:

```
::MyFile
```

Similarly, any number of colons can be used to move up to the desired level of the filesystem hierarchy. Once the desired level has been specified, folders down from that location can be specified. For example, to specify a file named "MyFile" found within a folder named "Projects" which is three folders up from the currently-running project, use the syntax:

```
:::Projects:MyFile
```

There are some special "tokens" that can be used in the File Path field. For example, the token `<Startup Segment Folder>` can be used to specify the folder that contains the currently-running title. For example, the following file specification would open the mTropolis title named "Title2," found in the currently-running project's startup segment folder:

```
<Startup Segment Folder>:MyFile
```

The tokens `<Title Folder>`, `<Preferences Folder on Macintosh>`, `<Boot Drive>`, and `<Player>` can be used as well.

Open File For buttons

Use these buttons to choose between different modes of opening the file. Note that the selected button can be overridden by setting the modifier's openMode attribute (described in the table below) to a different value. Choices are:

- **Reading:** Select this option to open the file for reading only. sets the index attribute to 1.
- **Reading/Writing:** Select this option to open the file for reading and writing sets the index attribute to 1.
- **Overwriting:** Select this option to open the file for "overwriting." Opening a file in this way causes the file to be opened, its length to be set to 0 (causing its previous contents to be lost), and the index attribute to be set to 1.

Cancel button

Click **Cancel** to ignore changes made to this modifier.

OK button

Click **OK** to accept changes made to this modifier.

Miniscript syntax for File modifiers

Files opened with the File modifier can be manipulated through Miniscript statements and the File modifier's attributes, as described in the table below. Some examples of common file operations follow. The examples assume that the File modifier is named "filemod":

Reading the first line from a file

Assuming that the File modifier is set to open a previously-specified file for reading, read the first line from a file by executing the following Miniscript statements:

```
 -- Set delimiter to newline:
set filemod.chunkDelimiters to "\n"

-- Read the line if there are no
```

```
-- errors:
if not filemod.error then
  set myString to filemod.readChunk
end if
```

Note that filemod.index is now positioned at the start of the next chunk and filemod.lastDelimiter is now set to the last delimiter read.

Reading a specific number of characters
In this example, we'll use attributes to manually specify the file to be read and open the file for reading (overriding any setting in the File modifier's dialog box), then read the second 100 characters from the file:

```
-- Specify the file to be opened.
In
-- this case the
-- file is named data.txt and
resides -- in the same folder
-- as the title:
set filemod.filePath to \
"<Title Folder>:data.txt"

-- Open the file for reading:
-- set filemod.openMode to "read"

-- Check for errors and read the
characters:
if not filemod.error and \
filemod.length >= 200 then
  set filemod.index to 101
  set myString to filemod.read[100]
end if
```

If no errors occurred, and if the file had a length of 200 characters or greater, the String Variable "myString" will now contain the second 100 characters in the file (characters 101 through 200).

Writing a string list to a file
In this example, we assume that there is a list of string type named "list" within the File modifier's scope and that there are two Miniscript modifiers on the element. The first Miniscript contains the script:

```
-- Specify the file to be opened:
set filemod.filePath to \
"<Title Folder>:data.txt"

-- Open the file for reading and writing:
set filemod.openMode to "readwrite"

if not filemod.error then
  set list.forAllDo to next
end if
```

The second Miniscript (the "next" modifier referred to in the previous script) should be set to execute on receipt of the "Do" message and contains the script:

```
-- Append each incoming string to
the file:
set filemod.append to incoming
```

Using special text characters with the File modifier
Two special text characters are especially useful with the File modifier:

- The newline character ("\n") is often used to separate lines of text in a text file.
- The tab character ("\t") is useful when reading or writing tab-delimited files.

For example, consider a tab-delimited file where each record contains three fields (representing someone's first, middle, and last name separated by tabs) per line (delimited with a newline character), as follows:

```
Keith  R.  Crosley
William  J.  Clinton
```

The following Miniscript, put on a text element, could be used to scan through each field. Configure the Miniscript modifier to execute on Mouse Up:

```
-- Set delimiters to be tabs or
newlines:
set filemod.chunkDelimiters to
"\t\n"

-- Read a chunk and display it in
the text element:
set text to filemod.readChunk
```

Miniscript-accessible attributes of the File modifier

The File modifier has a number of unique attributes that can be accessed via Miniscript. For general information about using attributes, see Chapter 15, "Attributes." The table below describes unique attributes of the File modifier in a format similar to that found in "Lists of Attributes by Component Type" in Chapter 15.

File Modifier Attribute	Type	Get	Set	Description
append	String		•	Appends characters to the end of the file. The file index is reset to the end of the file when the append operation is complete.
chunkCount	Integer	•		The number of chunks (as defined by the settings of the chunkDelimiters attribute) in the file. Empty chunks are counted.
chunkDelimiters	String	•	•	The character(s) used to delimit chunks in the file. The default is the newline character ("\n") which is always interpreted correctly regardless of the type of newlines (for example, Mac OS, Windows, or Unix) in the file.
close	Boolean		•	Set to true to close the file. Files are also closed when the File modifier is unloaded from memory (as when the scene changes).
error	Integer	•		Contains an error code for the last operation on the file variable. Values can be: 0 - no error 1 - generic file error 2 - file not found 3 - can't write to file 4 - can't read from file 5 - can't open file 6 - can't open file for writing 7 - end of file
fileName	String	•		The name of the file.
filePath	String	•	•	The full path to the file. Set this attribute to a null string ("") to open the standard open file dialog box. If the dialog box is cancelled, the path remains "."

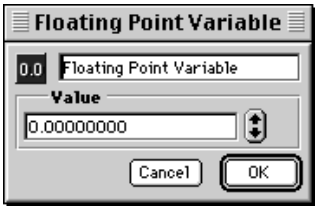
File Modifier Attribute	Type	Get	Set	Description
index	Integer	•	•	The current read or write position of the file. The index starts at 1 and is measured in characters.
lastDelimiter	String	•		The last delimiter character matched by the File modifier. This attribute is useful when the chunkDelimiters attribute contains a string with multiple characters.
length	Integer	•		The length, in characters, of the entire file.
macCreator	String	•	•	Specifies the Mac OS creator type for the file. Must be 4 characters long and defaults to "ttxt."
macType	String	•	•	Specifies the Mac OS type for the file. Must be 4 characters long and defaults to "TEXT."
openMode	String	•	•	This attribute controls the mode in which the file is opened. Values can be: "read" — open for reading "readwrite" — open for reading and writing "overwrite" — truncate file length to 0 before opening for reading and writing "default" — open the file according to the File modifier's current setting. "closed" — if the file has not been read, or written according to the mode in which it was opened. Sets the index to 0.
read[*n*]	String	•		Reads the next n characters from the file, starting from the current index. Resets index after reading.
readChunk	String	•		Reads the next chunk (as defined in chunkDelimiters) starting from the current index. Resets index after the chunk is read. An empty string is returned if the chunk is empty.
readToEnd	String	•		Reads from the current index to the end of the file.
write	String		•	Set this attribute to a string to write that string to the file starting at the current index. Note that the write operation overwrites any characters that may already be present starting at the index.

Unique attributes of the File modifier

Floating-Point Variable

Floating-Point Variables store floating-point values. Floating-point values have a range of $\pm 10^9 - 1$ and accuracy to 8 places. Floating-point values are useful in mathematical computations.

Components of the **Floating-Point Variable** dialog box are described below:



*The **Floating-Point Variable** dialog box*

Variable's Name field
This editable text field can be used to change the variable's name. If the variable will be referenced in a Miniscript modifier, it is good programming practice to make the name a single word.

Modifier icon
This icon identifies the variable modifier's type.

Value field
Enter the variable's initial value here or use the up/down arrow buttons to set the value by 0.5 increments. The value is checked for validity when the **OK** button is clicked. The value can be changed during run-time. See "Setting Values of Variable Modifiers" in Chapter 14 for details regarding getting and setting the value of this variable modifier.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

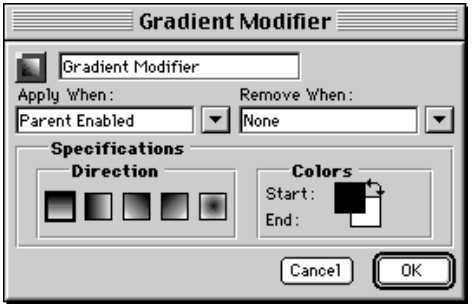
OK button
Click **OK** to accept changes made to this modifier.

Gradient Modifier

The Gradient modifier produces a color gradient between two colors on any graphic element that has not been linked to an external file (such as an "empty" graphic element).

!!! Note: The Gradient modifier is currently supported only on the Mac OS (as indicated by the yellow dot in the icon's upper left corner). This modifier will have no effect in a title built for the Windows platform.

Components of the **Gradient Modifier** dialog box are described below:



*The **Gradient Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Apply When pop-up menu
Use this pop-up menu to select the message that applies the Gradient modifier. For a complete discussion of this menu, see "When Pop-Up Menus and Message/Command Pop-Up Menu Options" in Chapter 13.

Remove When pop-up menu
Use this pop-up menu to select the message that removes the Gradient modifier. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Direction buttons
Click one of the sample **Gradient** buttons to select a direction/style for the Gradient.

Start and End color boxes
These boxes display the start and end colors of the Gradient. Click and hold on a square, and drag to a color on the palette that appears. To reverse the direction of the Gradient, click the arrow that points to both color squares.

!!! Note: Regardless of the bit depth of your project, Gradients produced with this modifier are always 8-bit Gradients. Some Gradients, especially those between very different colors, may look "banded" or dithered.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

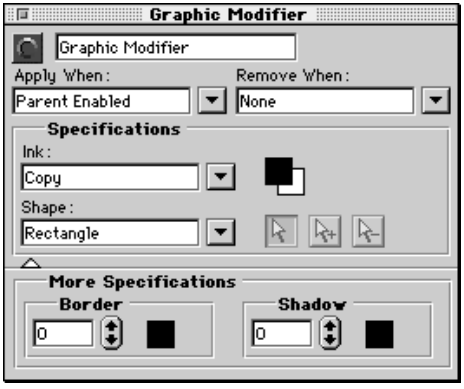
OK button
Click **OK** to accept changes made to this modifier.

Graphic Modifier

The Graphic modifier applies basic shapes, color, ink effects, borders, and shadows to graphic elements. It can also be used to define the matte color and borders of elements. Elements can have multiple Graphic modifiers, but the effects of only one Graphic modifier are active at once.

!!! Note: Some of the effects made possible by this modifier can also be chosen using tools on the **Tool** palette. See "Ink Effects" and "Foreground and Background Colors" in Chapter 11.

Components of the **Graphic Modifier** dialog box are described below:



*The **Graphic Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Apply When pop-up menu
Use this pop-up menu to choose the message that applies the Graphic modifier. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Options Menu" in Chapter 13.

Remove When pop-up menu
Use this pop-up menu to choose the message that removes the Graphic modifier. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Options" in Chapter 13.

Specifications section
Use the controls in this section to select the type of graphic effect to apply.

Ink pop-up menu
The ink effect changes the way a graphic element is displayed on the screen. Different inks have varying effects, depending on the color(s) of the object itself, and the object beneath it. Options in this pop-up menu are the same as those described in "Ink Effects" in Chapter 11. The foreground/background color boxes to the right of this menu work the same way as those described in "Foreground and Background Colors" in Chapter 11.

Shape pop-up menu
Use this menu to create or edit the "hot" region of the object. Areas not enclosed in the shape do not respond to mouse messages. Available shapes are:

- **Rectangle:** The default shape. The graphic completely fills its rectangular frame.
- **Round Rectangle:** The graphic takes a rectangular shape with rounded corners.
- **Oval:** The graphic takes an oval shape. If the frame of the graphic is square, the shape looks like a perfect circle.
- **Star:** The graphic takes a five-pointed star shape.
- **Polygon:** The Polygon option allows an author-definable irregular shape around an object within the element. To define the polygon region, use the tools to the right of the **Shape** pop-up menu. Note that polygon shapes should be created within the original rectangular bounding box. Polygon points created outside the rectangular border are ignored and the element is instead clipped to its original boundary.

Polygon shape tools
These tools are active when Polygon is chosen in the **Shape** pop-up menu:


Point Selection tool: Use this tool to drag a polygon point to a new location.


Add Point tool: Use this tool to add a new point to the polygon. Click on the polygon outline to add a new point.


Delete Point tool: Use this tool to delete a point on the polygon. Click on a polygon point with this tool to delete the point.

More specifications section
Click the white triangle at the bottom of the **Graphic Modifier** dialog box to toggle the display of these options:

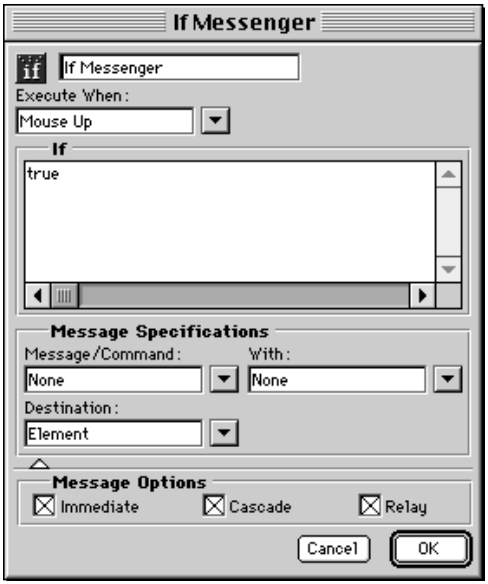
Border field
Use this field to specify the size, in pixels, of a border around the defined shape of the element. Use the color box to the right of this field to select a color for the border.

Shadow field
This option adds a shadow effect to the object of the color chosen in the box to the right.

If Messenger

The If Messenger is very similar to the standard messenger modifier, except that it only sends its message or command if a specified condition is true.

The messenger evaluates a boolean expression, entered in Miniscript syntax in the dialog box's "If" scrolling text field. This expression is evaluated when the messenger receives its "Execute When" message. If the expression evaluates to "true," the message or command specified in the Message Specifications section is sent. Otherwise, no message is sent. Components of the **If Messenger** dialog box are described below:



*The **If Messenger** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that causes the "If" expression to be evaluated. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

If expression scrolling text field
Enter a boolean expression in this field; use standard boolean and relational operators, reserved variables, functions and/or author-defined variables. This expression must be in proper Miniscript syntax. See "Miniscript Modifier" in Chapter 14. The syntax of the expression is checked when **OK** is clicked.

If the expression evaluates to true (see "Definition of True" in Chapter 14), the message specified in the Message Specifications section is sent. Some sample if expressions follow:

- Checking the value of an Integer Variable: The following expression evaluates to true if the end user-defined variable count is equal to 5:

```
count=5
```

- Checking the value of an attribute: The following expression evaluates to true if the width of the element that contains the If Messenger is greater than 10:

```
width > 10
```

- Checking the value of a Boolean Variable: The following expression evaluates to true

if the end user-defined Boolean Variable userCanDo contains true:

```
userCanDo
```

- Checking the value of a String Variable: The following expression evaluates to true if the end user-defined String Variable userName contains the string "super":

```
userName = "super"
```

- The value of "incoming" data (that is, data sent "With" the message that triggers the If Messenger) can also be checked. Suppose that the If Messenger is configured to trigger on "Mouse Up." Since mouse messages arrive with point data containing the position of the mouse action, we could check to see if the click occurred at a particular point (for example). The following expression evaluates to true if the Mouse Up occurred at point (100, 100):

```
incoming = (100,100)
```

Message specifications
Use this section to specify the message to be sent if the expression in the "If" scrolling text field evaluates to true.

See "Configuring Messenger Modifiers" or "Message Specifications" in this chapter for a complete description of the controls in this section.

Message options
Click the triangle at the bottom of the dialog box to display the Message Options section. See "Message Options" or "Message Options" in this chapter for a complete description of the controls in this section.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

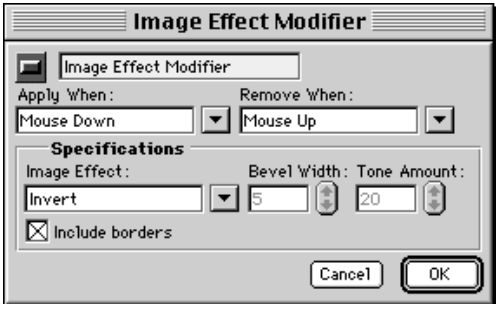
OK button
Click **OK** to accept changes made to this modifier.

Image Effect Modifier

The Image Effect modifier can apply one of a number of predefined graphic effects to an element. These effects are particularly useful when creating elements that represent clickable buttons.

Tip: Use multiple Image Effect modifiers to create a bevel-edged 3D-button effect: one to show the regular state of the button, one to show it depressed and one to reset it to its regular state.

Components of the **Image Effect Modifier** dialog box are described below:



*The **Image Effect Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Apply When pop-up menu
Use this pop-up menu to choose the message that applies the effect. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Remove When pop-up menu
Use this pop-up menu to select the message that removes the effect. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Specifications section
Use the controls in this selection to choose the effect to be applied.

Image Effect pop-up menu
Use this pop-up menu to choose an image effect. Options include:

- **Invert:** This option causes the color of the element to invert, that is, the opposite color of the element will be applied.
- **Selected Bevels:** This option applies a 3D beveled edge effect to the element. The width of the bevel edge and the tone of the color to be applied to the edge is specified in the Bevel Width and Tone Amount fields.
- **Deselected Bevels:** This option applies an indented 3D beveled edge effect to the element. The width of the bevel edge and the tone of the color to be applied to the edge are specified in the Bevel Width and Tone Amount fields.
- **Tone Down:** This option darkens the tone of the element's color according to the value specified in the Tone Amount field.
- **Tone Up:** This option lightens the tone of the element's color according to the value specified in the Tone Amount field.

Bevel Width field
Use this field to choose a width of the bevel, in pixels, when a bevel effect is selected. The up/down buttons increase or decrease the bevel size by 1 pixel. This field is active only when a bevel effect is selected.

Tone Amount field
Use this field to choose the severity of the selected effect. The up/down arrow buttons increase or decrease the tone in increments of 1%. This field is active for all effects except Invert.

Include Borders check box
Choose this option to apply the image effect to an element's border and/or shadow effect (if it has one). See "Border Field" and "Shadow Field" in this modifier entry.

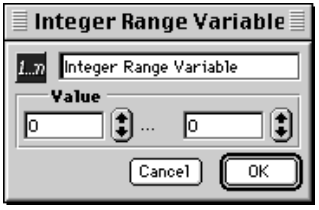
Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Integer Range Variable

An Integer Range Variable is used to hold an integer range value. These values are especially useful for storing ranges of cels in an "mToon" animation. See "The mToon Menu" in Chapter 1 for more information on mToons.

Components of the **Integer Range Variable** dialog box are described below.



*The **Integer Range Variable** dialog box*

Variable's Name field
This editable text field can be used to change the variable's name. If the variable will be referenced in a Miniscript modifier, it is good programming practice to make the name a single word.

Modifier icon
This icon identifies the variable modifier's type.

Value fields
Enter the variable's initial values here. The input field on the left represents the start value. The input field on the right represents the end value. Highlight a field and enter a value or use the up/down arrow buttons to change the value of the field.

The value is checked for validity when the **OK** button is clicked.

Note: If this range is to be used with an mToon, the start and end values should be less than the number of cels in the animation being controlled.

The values can be changed during run-time. See "Setting Values of Variable Modifiers" in Chapter 14 for details regarding getting and setting the value of this variable modifier.

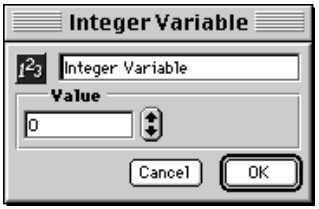
Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Integer Variable

The Integer Variable modifier holds integer values. Integer Variables are "long integers" with a range of –2147483648 to +2147483647. Integer Variables are useful as counters or in mathematical computations.

Components of the **Integer Variable** dialog box are described below:



*The **Integer Variable** dialog box*

Variable's Name field
This editable text field can be used to change the variable's name. If the variable will be referenced in a Miniscript modifier, it is good programming practice to make the name a single word.

Modifier icon
This icon identifies the variable modifier's type.

Value field
Enter the variable's initial value here or use the up/down arrow buttons to set the value. The value is checked for validity when the **OK** button is clicked. The value can be changed during run-time. See "Setting Values of Variable Modifiers" in Chapter 14 for details regarding getting and setting the value of this variable modifier.

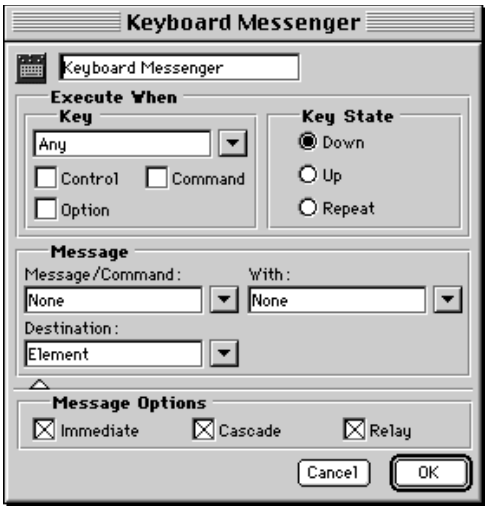
Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Keyboard Messenger

The Keyboard Messenger generates messages when a specified keyboard event, is detected.

The **Keyboard Messenger** dialog box is described below.



*The **Keyboard Messenger** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When section
The controls in this section choose the type of keyboard event to be detected. Choose a key and key state as described below.

Key section
In the pop-up menu field, enter the key that causes this messenger to execute. Alternatively, use the pop-up menu to select one of the special keys described below:

- **Any:** Any keypress activates the modifier.
- **Return:** The main keyboard "Return" or "Enter" key is not to be confused with the "Enter" key found on the numeric keypad. On Mac OS, this is the key that is marked

"Return." On Windows, this is the same key, often marked "Enter."

- **Enter:** The "Enter" key found on the numeric keypad.
- **Tab**: The "Tab" key.
- **Backspace**: The "Backspace" or "Delete" key, not to be confused with "Del," the forward delete key.
- **Arrow Left**: The left arrow key.
- **Arrow Right**: The right arrow key.
- **Arrow Up**: The up arrow key.
- **Arrow Down**: The down arrow key.
- **Escape**: The "Escape" or "Esc" key.
- **Home**: The "Home" key.
- **End**: The "End" key.
- **Help**: On Mac OS, the "Help/Ins" key. On Windows, this key is mapped to F1 (because F1 is commonly used as a keyboard shortcut for displaying help). There is currently no mapping to the Windows "Insert" key.
- **Page Up**: The "Page Up" key.
- **Page Down**: The "Page Down" key.
- **Del**: The forward delete key, marked "Del" or "Delete."

Check the **Control**, **Command**, or **Option** key check box, if desired. Note that both Command and Option are mapped to the "Alt" key on Windows platforms.

Key State buttons

Select the type of key event to be detected. The Keyboard Messenger can detect the state of the key being pressed. The default state is key down, but the other choices allow for greater functionality:

- **Down**: The message or command is sent when the key is first pressed down.
- **Up**: The message or command is sent when the key is released after having been pressed.
- **Repeat**: The message or command is repeatedly sent as long as the key is held down (that is, while key repeat is activated). Note that this selection does not generate a message when the key is first pressed.

Message section

Use the controls in this section to specify the message to be sent when a keyboard event is detected.

Message/Command pop-up menu

Use this pop-up menu to choose the message or command to be sent when the messenger is activated. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

With pop-up menu

Use this menu to select a value to be sent with the message or command. The standard choices are available, but the "Incoming Data" menu option is slightly different when used with this modifier. When the specified keyboard event causes this modifier to execute, the value of the key that was pressed is sent to the Keyboard Messenger as incoming data. Select "Incoming Data" to send that key value with the selected outgoing message. Note that this key value can only be accessed by the Miniscript modifier. The value cannot be properly interpreted by other modifiers. The Miniscript "incoming" keyword can be used to access the key value where it is interpreted by Miniscript as a string. For example, the following Miniscript statement

would display the pressed key in a text element named “mytext”:

```
set mytext.text to incoming
```

Destination pop-up menu
Use this menu to select the destination for the message or command sent from this modifier. For a complete discussion of this menu, see “The Destination Pop-Up Menu” in Chapter 13.

Message options
Click the triangle at the bottom of the dialog box to display the Message Options section. See “Message Options” in this chapter for a complete description of the controls in this section.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

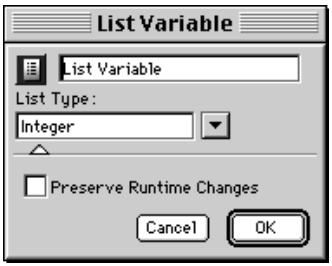
OK button
Click **OK** to accept changes made to this modifier.

List Variable

The List Variable modifier can be used to store a list of values of the same data type. Values can be added to and deleted from the list dynamically via Miniscript statements. Other Miniscript statements can be used to sort the list, randomly shuffle the individual list elements, return the number of elements in a list, or return randomly-selected values from the list.

Note: The **List Variable** dialog box is used only for naming a List Variable and declaring the data type of its elements. Adding values to a list or otherwise manipulating its contents must be performed through Miniscript statements as described below The contents of a list cannot be seen or set in any of mTropolis’ editing views.

Components of the **List Variable** dialog box are described below.



*The **List Variable** dialog box*

Variable’s Name field
This editable text field can be used to change the variable’s name. If the variable will be referenced in a Miniscript modifier, it is good programming practice to make the name a single word.

Modifier icon
This icon identifies the variable modifier’s type.

List Type pop-up menu
Use this pop-up menu to select the data type of the list elements.

Preserve Run-Time Changes check box
Click the open/close triangle at the bottom of the **List Variable** dialog box to reveal the **Preserve Run-Time Changes** check box. When this box is checked, changes to the contents of the List Variable made during run-time mode are retained upon returning to edit mode, when Player Emulation is off

(unchecked — see "Player Emulation" in Chapter 1). In addition, the values in the list are preserved when the project is built into a title.

When the option is unchecked (the default), the contents of the List Variable will be lost — if the variable is located at the scene level or lower — when switching to or from run-time mode. In addition, when built into a title the List Variable is written as an empty list. This behavior is the same regardless of whether Player Emulation is enabled or disabled. List Variables are different from all other variables in this respect (other types of variables always retain their values when Player Emulation is off).

Note: Unchecking the **Preserve Run-Time Changes** check box causes the values in the list to be lost immediately.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Miniscript syntax for list variables

The individual elements of a list can be accessed and manipulated through Miniscript statements as described below.

Basic list syntax
Individual elements of a list can be referenced by using the following syntax:

```
listname[n]
```

where *listname* is the name of the list, as specified in its name field and *n* is an integer from 1 to the total number of list elements. For example, to retrieve the value of the third element of the integer List Variable `myIntList` and store it in the variable `myInt`, use the statement:

```
set myInt to myIntList[3]
```

The literal value for an entire list is represented by a comma-separated list of values enclosed in curly brackets:

$$\{value_1,\ value_2,\ \ldots,\ value_n\}$$

where the value of the *n*th element is specified by $value_n$.

Creating the contents of the entire list
To create and set the contents of an entire list at once, use the syntax:

$$\texttt{set}\ listname\ \texttt{to}\ \{v_1,\ v_2,\ \ldots,\ v_n\}$$

where *listname* is the name of the list and the comma-separated list of values enclosed in curly brackets are the values to be assigned to list elements 1 through *n*. Note that when this syntax is used, any previous values held by the list are lost — the list's values and total number of elements change to conform to the specified list.

For example, the following statement creates a four-element integer list:

```
set myIntList to {10, 20, 30, 100}
```

After execution of this statement, the first element of `myIntList` will contain 10, the second will contain 20, the third will contain 30, and the fourth element (the last one in the list) will contain 100.

Changing the value of a list element
To change a single value in a list, use the set statement to assign a value to the desired element number in the list:

```
set listname[n] to value
```

where *listname* is the name of the list, *n* is the element's position in the list (starting at 1), and *value* is a data value of the correct type. A new list element is created at the specified position in the list.

If *n* exceeds the total number of elements in the list (that is, the *n*th element does not exist), the *n*th element is created with the specified value, but intermediate elements are also created and assigned a default value of 0 (or null for string values). For example, consider the following statement, executed when `myIntList` contains no elements:

```
set myIntList[3] to 10
```

List element 3 will be created and assigned the value 10, but elements 1 and 2 will also be created and assigned the default value of 0.

Inserting or adding a list element

Instead of changing the value of an existing list element, a new value can be inserted at any position in the list, causing the current element at that position (and any subsequent elements) to move one position later in the list. To insert an element, use the set statement with the list's insert attribute as follows:

```
set listname.insert[n] to value
```

where *listname* is the name of the list, *n* is the desired position of the new element and *value* is the data value to be stored in the new list element. For example, consider the following Miniscript statements that refer to an integer List Variable, `myIntList`:

```
-- create a three-element list
set myIntList to {10, 20, 30}
-- insert a new element at the second
-- position
set myIntList.insert[2] to 100
```

After executing these statements, `myIntList` would contain the list {10, 100, 20, 30}.

Deleting a list element

To delete an element from a list and return the deleted value (as if "popping" the element out of the list), reference the list's delete attribute in a Miniscript expression with the following syntax:

```
listname.delete[n]
```

where *listname* is the name of the list and *n* is the number of the element to be deleted from the list. The element is removed from the list, any later elements in the list are moved to one position earlier in the list to fill the gap, and the expression resolves to the value that was stored in the deleted element. For example, consider the following Miniscript statements that refer to a String Variable, `myString`, and a string List Variable, `myList`:

```
-- create a three-element string list
set myList to {"one", "two", "three"}

-- remove the second element from
-- the list
set myString to myList.delete[2]
```

After executing these commands, myString would contain the string value "two" and myList would contain the list `{"one", "three"}`.

Note: Attempting to delete an item that doesn't exist (for example, attempting to delete element 4 of a 2-element list) causes an error that halts Miniscript execution.

Returning the number of elements in a list
The count attribute of a List Variable stores the current number of elements in the list. Therefore, the Miniscript expression:

```
listname.count
```

where *listname* is the name of the desired list, resolves to the number of elements in the list. For example, the number of list elements could be retrieved with a statement such as:

```
set myInt to myIntList.count
```

Because the count attribute is writable, the number of list elements can also be changed by setting count to a new value. For example:

```
set myIntList.count to 10
```

Note that, if the new value of count is less than the previous value, the list is truncated, but remaining elements retain their previously stored values. If the new value of count is greater than the previous value, new elements are created past the end of the previous list. These new elements contain the default value of 0 (or null for strings).

Returning a randomly-selected value from a list
To return a randomly-selected value from a list, reference the list's random attribute:

```
listname.random
```

where *listname* is the name of the list. For example, to store a value, chosen at random from the list `myStringList`, into the String Variable `myString`, use the statement:

```
set myString to myStringList.random
```

Sorting the values in a list
The values in a list can be sorted in three different ways using the sortAscend, sortDescend, and shuffle list attributes.

To sort the values in a list in ascending order (that is, the first element contains the smallest value and the last element contains the greatest value), reference the list's sortAscend attribute:

```
listname.sortAscend
```

where *listname* is the name of the list to be sorted. This expression resolves to an integer value that is the total number of elements in the list.

Similarly, the list can be sorted in descending order (the first element contains the greatest value and the last element contains the smallest value) by referencing the list's sortDescend attribute:

```
listname.sortDescend
```

where *listname* is the name of the list to be sorted. Again, the expression returns an integer that represents the total number of elements in the list.

For example, consider the following Miniscript statements that refer to an Integer Variable, `myInt`, and an integer list, `myIntList`:

```
-- create a four-element list
set myIntList to {30, 40, 10, 20}
-- sort the list and return the
-- number of elements
set myInt to myIntList.sortDescend
```

After executing these statements, `myInt` contains the value 4, and `myIntList` contains the list `{40, 30, 20, 10}`.

Randomizing the order of list elements
List elements can be "unsorted" into a random order by referencing the list's shuffle attribute:

```
listname.shuffle
```

where *listname* is the name of the list to be shuffled. The expression returns an integer that represents the total number of elements in the list.

Looping over the items in a list — forAllDo
The forAllDo attribute of a list can be used to implement a type of "looping" in mTropolis. Often, it is necessary to perform the same action repetitively, once for each item in a list. The forAllDo attribute makes such tasks easy.

When the list variable's forAllDo attribute is set to an object reference, mTropolis sends a number of "Do" messages to the specified object. One Do message is sent for each item in the list. Each Do message is sent with the value of the corresponding list item as accompanying data. In this way, a modifier can be configured to perform multiple actions based on the values in a list.

As sent by mTropolis, the Do message does not "cascade" or "relay." It is sent only to the specified object and no others. See "Do" in Chapter 13.

The forAllDo attribute is useful when you want to display all of the values in a list variable, as the following simple example illustrates:

- Drop a List Variable on an empty scene. Give the List Variable the name `myList` and choose "String" from its list type menu.
- Create a text element on the scene. Give it the name `myText`.
- Drop a Miniscript modifier on the text element. Configure it to execute on the Do message (found in the "Object" submenu of the **Execute When** pop-up menu. Enter the following statements in its script field:

```
-- On each Do message, append the
-- incoming text
set append to " " & incoming
```

- Now drop a Miniscript modifier on the scene. Configure it to execute on Scene Started and enter the following statements in its script field:

```
-- Create the contents of the list
set myList to {"cat", "dog", "cow", \
  "horse"}

-- Send Do messages to myText
set myList.forAllDo to myText
```

- Now run the project by pressing ⌘-T. You should see the text element containing the values of the list (`"cat dog cow horse"`).

Summary of Miniscript-accessible attributes of the List Variable
The table below summarizes the unique attributes of the List Variable. Complete descriptions of these attributes are provided above.

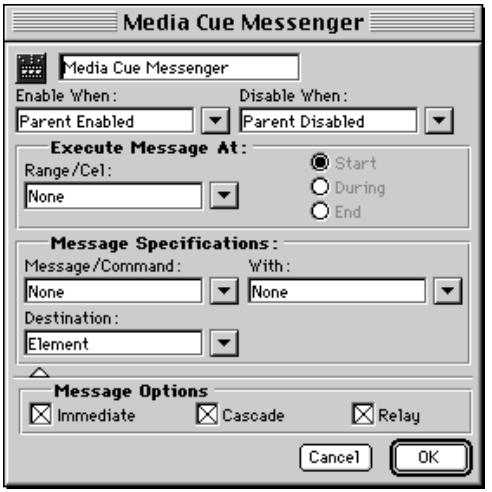
List Variable Attribute	Type	Get	Set	Description
count	Integer	•	•	Sets or returns the total number of items in the list.
delete[*n*]	List Type	•		Deletes the nth item from the list and returns that value.
forAllDo	Object		•	Set to an object reference to send Do messages to that object, accompanied by individual data values from the list.
insert[*n*]	List Type		•	Inserts a new item at the nth position in the list.
random	List Type	•		Returns a random item from the list.
shuffle	Integer	•		Shuffles the list items into a random order and returns the number of items in the list.
sortAscend	Integer	•		Sorts the list items in ascending order and returns the number of items in the list. Lists of object reference type cannot be sorted.
sortDescend	Integer	•		Sorts the list items in descending order and returns the number of items in the list. Lists of object reference type cannot be sorted.

Unique attributes of the List Variable

Media Cue Messenger

The Media Cue Messenger can be used to send messages at specified times when mToon, sound, or QuickTime media are playing. When activated, the Media Cue Messenger monitors its element for a specified media cue point, such as the start or end of a range. When that point is reached, the messenger sends its message.

Components of the **Media Cue Messenger** dialog box are described below:



*The **Media Cue Messenger** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Enable When pop-up menu
Use this pop-up menu to choose the message that causes the Media Cue Messenger to start checking the element for the media condition selected in the "Execute Message At:" section. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Disable When pop-up menu
Use this pop-up menu to choose the message that causes the Media Cue Messenger to stop checking for the specified media condition.

"Execute Message At" section
The controls in this section of the dialog box specify the condition under which a message is sent.

Range/Time pop-up menu
Use this pop-up menu to choose a range that, when played, triggers a message. Valid entries are:

- **Named Toon Range:** If the Media Cue Messenger is on an mToon element that has named ranges defined, they appear in the "mToon Ranges" cascading menu item found in this pop-up menu.
- **Named QuickTime Range:** If the Media Cue Messenger is on a QuickTime element that has named ranges defined, they appear in the "QuickTime Ranges" cascading menu item found in this pop-up menu.
- **Named Sound Marker:** If the Media Cue Messenger is on a sound that has sound markers defined, they appear in the "Sound Markers" cascading menu item found in this pop-up menu. Note that sound markers are single points in a sound file, not ranges. Therefore, when choosing a sound marker as the "Range/Time" selection, use only the **Start** or **End** buttons. The specified message is sent when the sound plays at the specified marker.

- **Integer Range or Integer Variable**: Any Integer Range Variables in the Media Cue Messenger's scope can be selected from the pop-up menu.

  If the Media Cue Messenger is on an mToon, the integer range is interpreted as a range of cels in the mToon. If the messenger is on a QuickTime movie, the integer range is interpreted as a range of QuickTime timevalues (see "timevalue" in Chapter 15).

  Alternatively, an Integer Variable can be selected to specify a single-cel "range." Note that if a single cel is specified, the **Start**, **During**, and **End** button selections all have the same effect.

- **Constant Range of Cels**: A literal integer range can be entered in the "Range/Time" field. Highlight the field and type an integer range in proper Miniscript syntax (`2 thru 6`). More information on the syntax for literal values can be found in "Literal Values" in Chapter 14.

  If the Media Cue Messenger is on an mToon, the integer range is interpreted as a range of cels in the mToon. If the messenger is on a QuickTime movie, the integer range is interpreted as a range of QuickTime timevalues (see "timevalue" in Chapter 15).

  Alternatively, a single-cel "range" can be specified by entering an integer (2) in this field. Note that if a single cel is specified, the **Start**, **During**, and **End** button selections all have the same effect.

Start button
Click this button to send a message when:

- for mToons: the first cel in the range specified in the "Range/Time" field (the "start" value of the integer range) is played.
- for QuickTime: the first timevalue in the range specified in the "Range/Time" field is played.
- for sounds: the specified sound marker is played.

During button
Click this button to send a message whenever an mToon cel within the specified range is played. This button has no effect when used with sound markers.

End button
Click this button to send a message when:

- for mToons: the last cel in the range specified in the "Range/Time" field (the cel that corresponds to the "end" value of the integer range) is played.
- for QuickTime: the last timevalue in the range specified in the "Range/Time" field is played.
- for sounds: the specified sound marker is played.

Message specifications
All messenger modifiers have the same controls in the Message Specification section of their dialog boxes:

Message/Command pop-up menu
Use this pop-up menu to choose the message or command to be sent when the messenger is activated. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

With pop-up menu
Use this menu to choose a value to be sent with the message or command. The choices on this pop-up menu are None, Incoming

Data and any variables to which the element has access. These menu items are described in more detail below:

- **None:** This is the default selection. No value is sent with the message.
- **Incoming Data:** Choose this option to configure the messenger to send the incoming data (the value that arrives with the message that has been configured to trigger the messenger), with the outgoing message or command.
- **Variables:** To send the value of a variable modifier with the outgoing message, choose its name from the submenus available in the second section of the **With** pop-up menu. All variable modifiers currently available to a messenger from its position in the project's hierarchy are shown. These variable modifiers are only accessible to those modifiers or Miniscripts in its "scope." See "Variable Scopes" in Chapter 14.
- **Constant Data Value:** To send a constant data value, highlight the With text field and type the value to be sent. The value should be entered with the same syntax as if it were being used in a Miniscript modifier. The syntax of the expression entered is checked when **OK** is clicked. Any appropriate mTropolis data type can be sent.

Destination pop-up menu
Use this menu to select the destination for the message or command sent from this modifier. For a complete discussion of this menu, see "The Destination Pop-Up Menu" in Chapter 13.

Message options
All messenger dialog boxes have a Message Options section that can be revealed by clicking the triangle at the bottom of the dialog box. Options in this section are:

Cascade check box
When this box is checked, the message "cascades" from the targeted destination to all components in the hierarchy below it. Uncheck this option to configure the messenger to send its message only to the modifiers in a targeted element. "Cascade off" is equivalent to the path of an environment message sent by mTropolis.


Note: This option has no effect when sending a command.

Immediate check box
By default, messages or commands are sent immediately when the messenger is activated. Uncheck this box to configure the messenger to send its message or command only after the current thread of messages has ended. This option can be used to avoid system overload if the message queue becomes too deep.

Relay check box
By default, messages travel from element to element in the hierarchy activating any and all elements that respond to the message. Uncheck this box to activate only the first modifier in the path of the message that is configured to respond.


Note: This option has no effect when sending a command.

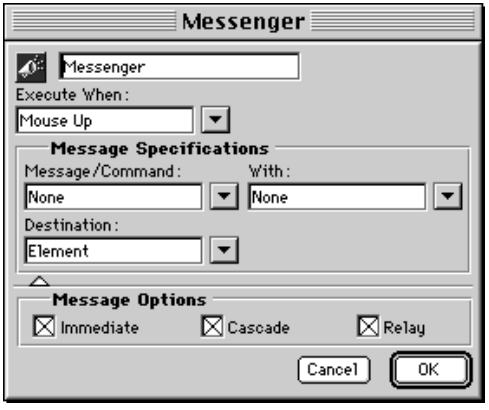
Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Messenger

The Messenger modifier (sometimes referred to as simply a "messenger") sends a message or command after it has received a specified message.

Components of the **Messenger** dialog box are described below.



*The **Messenger** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that causes the messenger to activate. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Message specifications
All Messenger modifiers have the same controls in the Message Specification section of their dialog boxes:

Message/Command pop-up menu
Use this pop-up menu to choose the message or command to be sent when the messenger is activated. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

With pop-up menu
Use this menu to choose a value to be sent with the message or command. The choices on this pop-up menu are None, Incoming Data, and any variables to which the element has access. These menu items are described in more detail below:

- **None:** This is the default selection. No value is sent with the message.
- **Incoming Data:** Choose this option to configure the messenger to send the incoming data (the value that arrives with the message that has been configured to trigger the messenger), with the outgoing message or command.
- **Variables:** To send the value of a variable modifier with the outgoing message, choose its name from the submenus available in the second section of the With pop-up menu. All variable modifiers currently available to a messenger from its position in the project's hierarchy are shown. These variable modifiers are only accessible to those modifiers or Miniscripts in its "scope." See "Variable Scopes" in Chapter 14.

- **Constant Data Value:** To send a constant data value, highlight the With text field and type the value to be sent. The value should be entered with the same syntax as if it were being used in a Miniscript modifier. The syntax of the expression entered is checked when **OK** is clicked. Any appropriate mTropolis data type can be sent.

Destination pop-up menu
Use this menu to select the destination for the message or command sent from this modifier. For a complete discussion of this menu, see "The Destination Pop-Up Menu" in Chapter 13.

Message options
All **Messenger** dialog boxes have a Message Options section that can be revealed by clicking the triangle at the bottom of the dialog box. Options in this section are:

Cascade check box
When this box is checked, the message "cascades" from the targeted destination to all components in the hierarchy below it. Uncheck this option to configure the messenger to send its message only to the modifiers in a targeted element. "Cascade off" is equivalent to the path of an environment message sent by mTropolis.

Note: This option has no effect when sending a command.

Immediate check box
By default, messages or commands are sent immediately when the messenger is activated. Uncheck this box to configure the messenger to send its message or command only after the current thread of messages has ended. This option can be used to avoid system overload if the message queue becomes too deep.

Relay check box
By default, messages travel from element to element in the hierarchy activating any and all elements that respond to the message. Uncheck this box to activate only the first modifier in the path of the message that is configured to respond.

Note: This option has no effect when sending a command.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

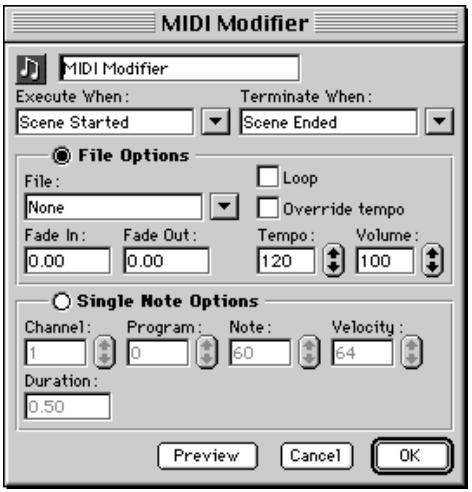
MIDI Modifier

The MIDI modifier can be used to play MIDI files and single MIDI notes.

System requirements for the MIDI modifier
On Mac OS, this modifier requires that QuickTime 2.5 (or greater) and the QuickTime Musical Instruments be installed. The modifier plays to the QuickTime General MIDI software synthesizer. If QuickTime Musical Instruments are not installed, MIDI playback will not work.

The MIDI modifier also works on Windows machines that have MIDI hardware (a MIDI card or soundcard that supports General MIDI) installed. MIDI is sent to the currently active MIDI output device.

Components of the **MIDI Modifier** dialog box are described below.



*The **MIDI Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to select the message that causes the MIDI file or note to begin playing. When this message is received and a MIDI file is being played, the file fades in from zero volume if a Fade In time has been selected. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Terminate When pop-up menu
Use this pop-up menu to choose an optional message that causes the MIDI file or note to stop playing. When this message is received and a MIDI file is being played, the file fades to zero volume (if a Fade Out time has been selected) and stops playing. When this message is received and a single MIDI note is being played, the note is sent a MIDI "note off" message and stops playing. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

File options section/ button
Click the **File Options** button to play a MIDI file. Controls in this section become active.

File field
Enter the name of a MIDI file to be played in this field. Alternatively, choose **Link File** from this menu to link display a standard file dialog box that can be used to select a MIDI file.

All files with the Mac OS type "Midi" or whose file names end in ".mid" are displayed. All three MIDI file formats (type 0, type 1, and type 2) can be read, though type 2 files (designed to contain multiple pieces of music) are interpreted as multitrack files. MIDI files in which the times are SMPTE-based instead of bar and beat based cannot be used.

Loop check box
Click this check box to cause the MIDI file to play looped. By default, the file loops from its beginning to the end of the longest track. Alternate loop points can be specified by including MIDI Marker events, named "loopstart" and "loopend" (case is important) in the file. These events can be placed in any track, though it is recommended that they appear in the meter track of a multitrack MIDI file.

Override Tempo check box
Click this check box to play the file at a tempo other than that defined in the file. The Tempo field becomes active.

Tempo field
Use this field to select a tempo, in beats per minute, to use when playing the MIDI file. This option is only available when the **Override Tempo** check box is selected. The default value shown in the field is the first tempo found in the MIDI file. Valid ranges for the tempo are 10 to 240 beats per minute.

Fade In field
Use this field to specify a time, in seconds.hundredths over which the volume of the MIDI file will fade in (from silence to full volume) when it starts playing. This feature works by scaling the MIDI velocity values in the file.

Fade Out field
Use this field to specify a time, in seconds.hundredths over which the volume of the MIDI file will fade out (from full volume to silence) before it stops playing. This feature works by scaling the MIDI velocity values in the file.

Volume field
Use this field to specify the initial volume of the MIDI file. Values can range from 0 to 100, representing a percentage of full volume (for example, 0 is silence, 100 is full volume). The default is 100. This feature works by scaling the MIDI velocity values in the file.

Single note options section/button
Click the **Single Note Options** button to play a single MIDI note. Controls in this section become active.

Channel field
Use this field to select the MIDI channel on which the note is sent. Valid values for this field are 1 through 16. Note that QuickTime does not distinguish between channels, but the Apple MIDI Driver (which sends MIDI information to external MIDI devices) does.

Program field
Use this field to select the sound to be played. The QuickTime Musical Instruments and other General MIDI devices have a different sound assigned to each program number. On external MIDI devices, this parameter selects the "patch" (synthesizer program) to be played. Valid values are 1 through 127.

Note field
Use this field to select the MIDI note number to be played. Valid values are from 0 (the lowest note) to 127 (the highest note). The default is note number 60, which corresponds to middle C.

Velocity field
Use this field to select a velocity for the note to be played. The velocity parameter often controls the volume of the selected sound, but may also affect its timbre (the tone of the sound), duration, or other attributes. Many MIDI keyboards (and other MIDI instruments) send different velocity values depending on how fast the keys are struck. Valid values are from 0 (lowest velocity) to 127 (highest velocity). The default is 64.

Duration field
The duration of the note in seconds.hundredths. Note that the sound may end before this time has elapsed due to the programming of the sound being played. The sound can also be stopped before this time has elapsed if the "Terminate When" message is received.

Stop button
Click this button to stop any MIDI notes being previewed.

Preview button
Click this button to preview the selected MIDI file (including specified "Fade In" time) or note.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK Button
Click **OK** to accept changes made to this modifier.

Miniscript-accessible attributes of the MIDI modifier
The MIDI modifier has many attributes that can be accessed via Miniscript. For general information about using attributes, see Chapter 15, "Attributes." The table below describes attributes for the MIDI modifier in a format similar to that found in "Lists of Attributes by Component Type" in Chapter 15.

MIDI Mod Attribute	Type	Get	Set	Description
loop	Boolean		•	When true, the MIDI file plays looped
muteTrack[*n*]	Boolean		•	Set to true to mute the MIDI track specified by n (for example, `set mutetrack[5] to true mutes track 5`). If no track is specified, all tracks are affected.
noteChannel	Integer		•	The MIDI channel (1–16) on which to play the next single note.
noteDuration	Float		•	The duration (in seconds) of the next single note.
noteNum	Integer		•	The MIDI note number (0–127) for the next single note.
noteProgram	Integer		•	The MIDI program number (0–127) for the sound to be played for the next single note.
noteVelocity	Integer		•	The MIDI velocity (0–127) for the next single note.
pan[*n*]	Integer		•	The stereo panning for a MIDI channel (if n is specified — for example pan[4] refers to the balance of channel 4) or all channels (if n is omitted). Values range from –100 (full left) through 0 (center) to 100 (full right).
playNote	Boolean		•	Set this attribute to any value to cause the note specified by the settings in the Single Note section of the dialog box to be played. The "note" attributes can be used to change the note to be played. The modifier must be configured to play a single note (rather than a file) for this attribute to work.

MIDI Mod Attribute	Type	Get	Set	Description
reset	Boolean		•	Set this attribute to any value to reset all changeable attributes of the MIDI file back to their initial values.
soloTrack[*n*]	Boolean		•	Set to true to play only the MIDI track specified by n, muting all other tracks.
tempo	Integer	•	•	The tempo used to play the MIDI file, in beats per minute. When retrieved, the tempo at the current position of the MIDI file, or the override tempo value (if set), is returned. Set this attribute to override tempos specified in the file.
transpose	Integer		•	A transposition value in halfsteps. Transposition affects only those tracks for which the "transposing" attribute is true. Values are –24 (transpose down 2 octaves) to 24 (transpose up two octaves).
transposing[*n*]	Boolean		•	Set to true to enable transposing for the channel specified by n (1–16). By default, all channels except channel 10 have this value set to true. If no track is specified, all tracks are affected.
volume	Integer	•	•	The volume of the MIDI file, specified as a percentage of full volume (0–100).

Unique attributes of the MIDI modifier

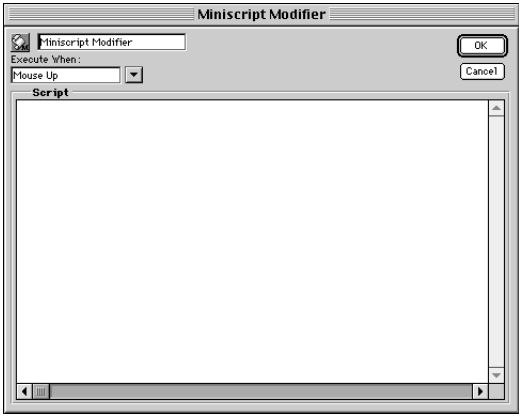
Miniscript Modifier

Miniscript is a complete scripting language embedded in a modifier. The Miniscript modifier allows you to use a scripting language to create customized modifiers that can:

- Get and set object attributes and variable values.
- Send messages and commands.
- Evaluate mathematical functions.
- Evaluate relational expressions and perform conditional branching.
- Parse and edit string data.

A complete description of the Miniscript language can be found in Chapter 14, "Miniscript Modifier."

Components of the **Miniscript Modifier** dialog box are described below.



*The **Miniscript Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that causes the Miniscript modifier to execute. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Script Text field
Enter the desired Miniscript script in this field. The syntax of the script is checked and any object references in the script are resolved when the **OK** button is clicked. A complete description of the Miniscript language can be found in Chapter 14, "Miniscript Modifier."

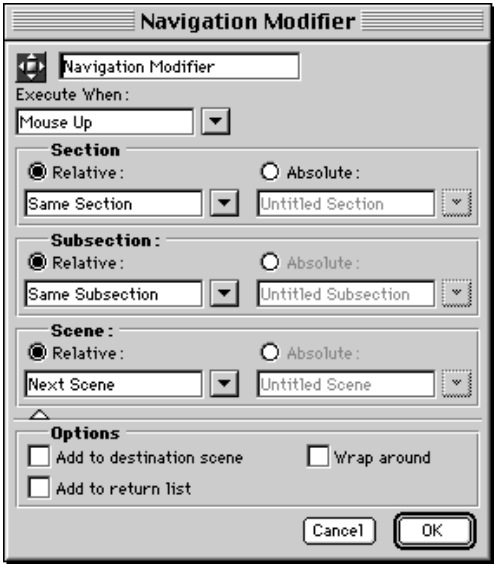
Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Navigation Modifier

The Navigation modifier executes a scene change when triggered in run-time mode. Any scene in any section and subsection of the project can be selected by name or by certain "relative" criteria. Note that this modifier is a "superset" of the functionality provided by the Change Scene modifier (described in this chapter).

Components of the **Navigation Modifier** dialog box are described below.



*The **Navigation Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that causes the scene to change. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Section options
Use the controls in this section of the dialog box to choose the section that contains the scene to be changed to. Options include:

- **Relative Button:** Click this button to activate the **Relative** pop-up menu. This menu can be used to choose a section by its position relative to the current section. Choose from "Same Section," "Previous Section," "Next Section," "First Section," and "Last Section." Note that the order of sections can be changed in the Structure window. See "Changing the Order of Components in the Structure Window" in Chapter 8.

- **Absolute Button:** Click this button to activate the **Absolute** pop-up menu. This menu can be used to choose a section by name.

Subsection options
Use the controls in this section of the dialog box to choose the selection that contains the scene to be changed to. Options include:

- **Relative Button:** Click this button to activate the **Relative** pop-up menu. This menu can be used to choose a subsection by its position relative to the current subsection. choose from "Same Subsection," "Previous Subsection," "Next Subsection," "First Subsection," and "Last Subsection." The "Corresponding Subsection (by Index)" can be chosen to designate the subsection in the new section that has the same position in the subsection order as the current subsection. The "Corresponding Subsection (by Name)" option can be chosen to designate the subsection in the new section that has the same name as the current subsection. Note that the order of subsections can be changed in the Structure window. See "Changing the Order of Components in the Structure Window" in Chapter 8.

- **Absolute Button:** Click this button to activate the **Absolute** pop-up menu. This menu can be used to choose a subsection

by name. This option is available only if the Section has also been specified by name (that is, the Section option's **Absolute** button is also clicked).

Scene options

Use the controls in this section of the dialog box to select the scene to be changed to. Options include:

- **Relative Button:** Click this button to activate the **Relative** pop-up menu. This menu can be used to choose a scene by its position relative to the current scene. Choose from "Same Scene," "Previous Scene," "Next Scene," "First Scene," and "Last Scene." The "Corresponding Scene (by Index)" can be selected to designate the scene in the new subsection that has the same position in the scene order as the current scene. The "Corresponding Scene (by Name)" option can be chosen to designate the scene in the new subsection that has the same name as the current scene. Note that the order of scenes can be changed in the Structure window. See "Changing the Order of Components in the Structure Window" in Chapter 8.
- **Absolute Button:** Click this button to activate the **Absolute** pop-up menu. This menu can be used to choose a scene by name. This option is available only if the Subsection has also been specified by name (the Subsection option's **Absolute** button is also clicked).

Options check boxes

Click the open/close triangle at the bottom of the **Navigation Modifier** dialog box to reveal navigation options:

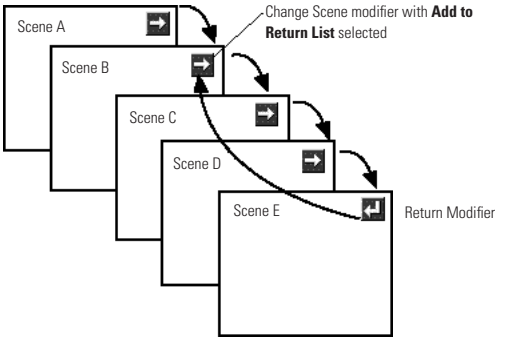
- **Add to Destination Scene:** Mark this check box to perform a special scene change in which the currently displayed scene is still visible after the change to the new scene. That is, the current scene acts like a shared scene for this scene change. Note that selecting this option automatically chooses the "Add to Return List" option as well.

  When this option is checked and a scene change is triggered, the Scene Deactivated message is sent to the "original" scene. See "Scene Deactivated" in Chapter 13. If the Return modifier is used to return from the new scene to the original scene, the Scene Reactivated message is sent to the original scene. See "Scene Reactivated" also in Chapter 13.

  Tip: The "Add to Destination Scene" option can be used to simulate dialog boxes and alerts. Put elements and artwork that comprise the dialog box or alert on the destination scene and ensure that this option is checked. When the scene is changed, it looks as though the dialog box has appeared "in front" of the previous scene. The Return modifier can be used to "dismiss" the dialog box or alert.

- **Add to Return List:** Check this check box to make the current scene the one returned to by the next Return modifier in a series of scenes. For example, in the figure below, Change Scene modifiers have been placed on each of the scenes. Each of these modifiers has been configured to change to the next scene. Scene B's scene change modifier has the Add to Return List option selected. A Return modifier has been placed on Scene E. When this modifier receives the message that will activate it during run-time, Scene E will change to Scene B (the arrows show the sequence of scenes).



The Return list

- **Wrap Around**: By default, the first and last scene in a subsection have no connection in the scene order. That is, the last scene in a subsection has no next scene and the first scene in a subsection has no previous scene.

  When this check box is checked, a change to the "Next Scene," executed from the last scene in a subsection, "wraps around" the list of scenes and changes to the first scene in the subsection.

  Similarly, a change to the "Previous Scene," executed from the first scene in a subsection, "wraps around" the list of scenes and changes to the last scene in the subsection.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Net Messaging Service

The Net Messaging Service is a modifier that enables and disables the use of network messages.

When enabled, the Net Messaging Service establishes an end point of communication using the TCP protocol and begins listening for messages destined for mTropolis. By default, the mTropolis network messaging service uses port 5480 for listening. The service checks for connections and/or packets every 100 milliseconds. When incoming messages arrive, this service dispatches them to the local mTropolis title.

Note: Because listening for network packets can be computationally expensive, consider disabling this service when it is not required. Note also that TCP/IP must be installed for this service to work properly.

Components of the **Net Messaging Service** dialog box are described below.

*The **Net Messaging Service** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Enable When pop-up menu
Use this pop-up menu to choose the message that enables network messaging. The default for this menu is Project Started because this modifier is commonly dropped directly on the project component. If the Net Messaging Service is placed elsewhere in the structural hierarchy, the default "Enable When" message should be changed (because Project Started is sent only to the project component itself). When the specified message is received, the service begins listening for packets.

For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

If the Net Messaging Service cannot be enabled, or if the maximum number of network connections has been exceeded, mTropolis sends a Network Failed message to the modifier's parent. See "Network Failed" in Chapter 13.

Disable When pop-up menu
Use this pop-up menu to choose the message that disables network messaging. The default is None. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Attributes of the Network Messaging Service
The Network Messaging Service modifier has several attributes that can be accessed via Mini-script. For general information about using attributes, see Chapter 15, "Attributes." The table below describes attributes for the Net Messaging Service in a format similar to that found in "Lists of Attributes by Component Type" in Chapter 15.

Net Messaging Service Attributes	Type	Get	Set	Description
listeningPort	Integer	•	•	The port being used to receive TCP/IP messages. The default is 5480. To use a different port, change this attribute before activating the service.
localIP	String	•		The IP address of the local machine for this session. Note that IP addresses may be different each network session.
maxConnections	Integer	•	•	The maximum number of simultaneous connections (incoming and outgoing) allowed by the title. The default is 64 for Mac OS TCP. The default is variable and system-dependent for Mac OS using Open Transport and Windows platforms. This attribute can be set, but if the requested maximum connections exceeds the number allowed by the system, maxConnections will be set to the maximum number of connections allowed by the system.
sourceIP	String	•		The IP address of the sender of the most recently received message.
sourcePath	String	•		The path to the Net Messenger that sent the most recently received message.
sourceParentPath	String	•		The path to the parent of the Net Messenger that sent the most recently received message.
timeoutPeriod	Integer	•	•	The timeout period, in seconds, that the Net Messaging Service uses when trying to connect to the host IP address. The default is 20 seconds. When this time period has elapsed and connections cannot be made or a message cannot be sent, the Connection Timed Out error message is sent. See “Connection Timed Out” in Chapter 13.

Attributes of the Network Messaging Service

Net Messenger

The Net Messenger is a variation on the standard messenger modifier that can send mTropolis messages to other projects over a TCP-based LAN or over the Internet.

Note: The Net Messaging Service must be enabled for messages to be sent or received over a network.

Components of the **Net Messenger** dialog box are described below.

*The **Net Messenger** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that causes the messenger to activate. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Message specifications
Use the controls in this section to specify the message to be sent by the Net Messenger:

Message/Command pop-up menu
Use this pop-up menu to choose the message or command to be sent when the messenger is activated. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

With pop-up menu
Use this menu to select a value to be sent with the message or command. The choices on this pop-up menu are None, Incoming Data and any variables to which the element has access. These menu items are described in more detail below:

- **None**: This is the default selection. No value is sent with the message.
- **Incoming Data**: Choose this option to configure the messenger to send the incoming data (the value that arrives with the message that has been configured to trigger the messenger), with the outgoing message or command.
- **Variables**: To send the value of a variable modifier with the outgoing message, select its name from the submenus available in the second section of the **With** pop-up menu. All variable modifiers currently available to a messenger from its position in the project's hierarchy are shown. These variable modifiers are only accessible to those modifiers or Miniscripts in its "scope." See "Variable Scopes" in Chapter 13.

- **Constant Data Value**: To send a constant data value, highlight the With text field and type the value to be sent. The value should be entered with the same syntax as if it were being used in a Miniscript modifier. The syntax of the expression entered is checked when **OK** is clicked. Any appropriate mTropolis data type can be sent.

Destination pop-up menu
Use this menu to select the destination for the message or command sent from this modifier. For a complete discussion of this menu, see "The Destination Pop-Up Menu" in Chapter 13.

Most destinations are resolved locally (that is, in the sender's project). To be more specific, during run-time, the targeted object is searched for in the sender's project. The absolute path to that object is then sent across the network to the remote project(s) and applied there. See "Order of Execution for the Net Messenger" below.

The behavior of the Active Scene and Shared Scene destinations should be noted. At any given time, the Active Scene or Shared Scene in the sender's project may be different than those in the destination project. During run-time, these destinations are resolved for each host that has been targeted. As a result, the message being sent may arrive at a different scene for each host and the result of sending the message may vary.

In addition to the usual options, the Net Messenger's "Destination" menu contains an "By Reference" cascading option. This option provides easy access to all object reference variables in the messenger's scope. During run-time, the object reference contained in the variable is resolved for each targeted host. That is, the message is sent to the object "pointed at" by the specified Object Reference Variable, not to the Object Reference Variable itself.

Host pop-up menu
Use this menu to select the IP address of the destination project(s). The following options are available:

- **Local**: This is the default selection. No network host has been selected and the messenger behaves like a normal messenger, sending messages only to the local project.
- **Variables**: The contents of a String Variable can be used to store a host address. Select the name of a variable from the submenus available in the second section of the **Host** pop-up menu. All String Variables currently available to a messenger from its position in the project's hierarchy are shown. See "Variable Scopes" in Chapter 13.

  The contents of the string must be an IP address in the correct format (for example, `"230.22.34.56"`). The value contained in the variable is not resolved until run-time.

  Alternatively, a host's name may be stored in a String Variable as a hostname. For example:

  ```
  myhost
  ```

  Entering a host's name in this way will only be resolvable within a LAN. The domain name is assumed to be the same as that of the sender. On Mac OS platforms, the correct domain name must be entered in the "Implicit Search Path" field of the TCP/IP control panel for this to work successfully.

To target a host at another domain, use a fully qualified domain name. For example:

```
myhost.mfactory.com
```

- **IP Address Entered as a Constant**: To enter an IP address as a constant, highlight the Host text field and type the address as a string (for example, enclosed in quotes such as `"230.22.34.56"` or as a hostname such as `"myhost.mfactory.com"`). Since this address is not resolved until run-time, mTropolis does not know whether or not the address actually exists. Only the syntax of the string is checked when **OK** is clicked.

When a host's name is successfully resolved, a Host Name Found message is sent to the Net Messenger's parent. Similarly, if mTropolis cannot resolve a host name, a Host Name Not Found message is sent to the messenger's parent. See "Modifier Messages" in Chapter 13 for descriptions of these and other response messages that may be generated by the Net Messenger.

Message options

All **Messenger** dialog boxes have a Message Options section that can be revealed by clicking the triangle at the bottom of the dialog box. Options in this section are:

Cascade check box
When this box is checked, the message "cascades" from the targeted destination to all components in the hierarchy below it. Uncheck this option to configure the messenger to send its message only to the modifiers in a targeted element. "Cascade off" is equivalent to the path of an environment message sent by mTropolis.

Note: This option has no effect when sending a command.

Immediate check box
By default, messages or commands are sent immediately when the messenger is activated. Uncheck this box to configure the messenger to send its message or command only after the current thread of messages has ended. This option can be used to avoid system overload if the message queue becomes too deep.

Relay check box
By default, messages travel from element to element in the hierarchy activating any and all elements that respond to the message. Uncheck this box to activate only the first modifier in the path of the message that is configured to respond.

Note: This option has no effect when sending a command.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Order of execution for the Net Messenger

When activated, the Net Messenger performs the following tasks:

- "Execute When" message is received and activates the Net Messenger.
- If any remote hosts have been targeted:

1. The destination is resolved locally if a standard destination has been selected.
2. If any data has been specified in the **With** pop-up menu, it is retrieved.

3 The IP address of the hosts are retrieved from the String Variable specified in the **Host** pop-up menu.

4 If Active Scene or Shared Scene has been chosen as the destination, the relative path is sent to the Net Messaging Service which then sends the path to the host's project where the path is resolved.

5 The message or command specified in the **Message/Command** pop-up menu and any data specified in the **With** pop-up menu are sent to the Net Messaging Service which then sends them to the remote project(s).

- If no remote hosts have been targeted (for example, the **Host** pop-up menu is set to Local):

1 The chosen Destination is resolved locally.

2 Any data specified in the **With** pop-up menu is retrieved.

3 The chosen message and data are sent to the appropriate destination within the project.

Object Reference Variable

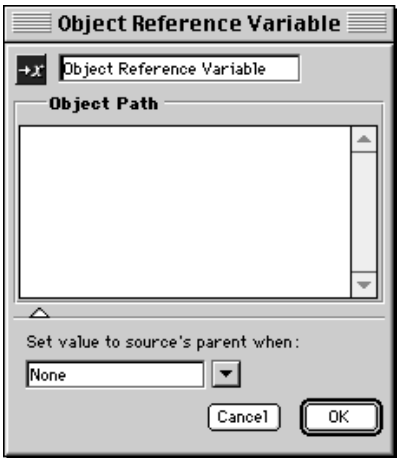
The Object Reference Variable stores a reference to another mTropolis object. It acts as a "pointer" to another mTropolis component.

The Object Reference Variable can be configured to store an initial value. It can also be configured to update its reference automatically upon receipt of a message. When the specified message is received, the Object Reference Variable stores a reference to the parent of the messenger that sent the activating message (that is, it stores the value of `source's parent`).

Once a value is stored, the referenced component can be targeted as the recipient for messages sent from Miniscript or used in other operations where a value of object reference type can be used. See "Miniscript Syntax for Object Reference Variables" below.

Object Reference Variables have another useful property: they can be used to reference a component even if that component has been unloaded from memory (that is, the referenced component resides on a scene that has been unloaded from memory). Whenever the Object Reference Variable is referred to, the component for which it stores a reference is reloaded.

Components of the **Object Reference Variable** dialog box are described below.



*The **Object Reference Variable** dialog box*

Variable's Name field

This editable text field can be used to change the variable's name. If the variable will be

referenced in a Miniscript modifier, it is good programming practice to make the name a single word.

Modifier icon
This icon identifies the variable modifier's type.

Object Path field
This field can be used to set an initial value for the Object Reference Variable. This field is also updated whenever the value stored in the Object Reference Variable changes. This field specifies the "path" to an object in a mTropolis project using the syntax described below:

- The path is described using a syntax similar to that used to specify directory names in the Unix operating system. The full path to an element, starting from the project level would be:

`/project/subsection/section/scene/element`

  where *project* is the actual name of the project, *subsection* is the name of the subsection, etc., all the way to *element* which is the name of the specific element being referenced.

- As implied above, the slash character (/) separates levels in the mTropolis structure hierarchy.

- To reference objects by their position relative to the Object Reference Variable, a relative pathname can be used. To specify structure levels above the current position, use ".." to move up a level and "/" to separate levels. For example, if the Object Reference Variable is in an element whose parent is the scene, the path:

  `../../myelement`

  would cause "myelement" to be searched for at the scene level.

  Similarly, to refer to a modifier that is a sibling of the object reference variable, a path such as:

  `../mymodifier`

  would start a search within the variable's parent for a modifier named "mymodifier."

  A special token, `<project>`, can be used to refer to the project component instead of the project's actual name. Using `<project>` prevents the object reference path from becoming invalid if the name of the project is changed (if the project is saved with a different file name). For example, an object might be specified by:

  `/<project>/mysection/mysubsection/myscene/myelement`

The syntax for the object path field is checked when the **OK** button is clicked. If the syntax is not in the correct format, an error alert appears. Note that, while the syntax is checked, the actual existence of the object specified in this field is not checked. The value of this field is not resolved to an object reference until the variable is referenced in run-time mode.

Set Value to Source's Parent When pop-up menu
Click the open/close triangle at the bottom of the **Object Reference** dialog box to reveal the **Set Value To Source's Parent When**: pop-up menu. Use this menu to select a message that causes the value of the variable to change to the parent of the messenger that sent the activating message.

Note: As mTropolis has no "parent," environment messages originating from mTropolis cannot be used to set the value of this variable. For example, suppose "Mouse Up" is chosen in the

"Set Value to Source's Parent" menu. An end user mouse click that generates a Mouse Up message will not cause the value of this variable to change. However, a "Mouse Up" message sent by a Messenger modifier would trigger the value change.

For a complete discussion of this menu, see "When Pop-Up Menu and Message/ Command Pop-Up Menu Options" in Chapter 13.

Differences between specifying an Object path versus the Set value to source's parent when pop-up menu
When a path is entered into the Object Path Field, the path is not resolved to an object reference until the Object Reference Variable is accessed during run-time. If the specified object is moved before the variable is referenced, the path will not be resolvable. However, if the object reference is established by the message specified in the **Set Path To Source's Parent** pop-up menu, the reference will always be current, even if the object moves.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Miniscript Syntax for Object Reference Variables

The value contained by an Object Reference Variable can be accessed through Miniscript as described below. A general overview of Miniscript can be found in Chapter 14, "Miniscript Modifier."

Sending messages to the referenced object
The component referenced by the Object Reference Variable can be accessed as the "object" attribute of the Object Reference Variable. That is, to send a message to the object at which the Object Reference Variable is pointing, use the following syntax:

```
send "message" to objectvar.object
```

where *message* is the message or command to be sent and *objectvar* is the name of the Object Reference Variable being accessed. For example, to send the **Play** command to the object referenced by an Object Reference Variable named `pointer1`, use the statement:

```
send "Play" to pointer1.object
```

Why is the "Object" attribute necessary?
New users of the Object Reference Variable are sometimes confused by the need for the "object" attribute. The "object" attribute is necessary because both the object stored in the variable, and the Object Reference Variable itself are valid targets for mTropolis messages and Miniscript expressions.

Consider the following Miniscript statement:

```
send "Set Me" to myObjectRef
```

where "Set Me" is an author message and "myObjectRef" is the name of an Object Reference Variable. This command sends the "Set Me" message directly to the Object Reference Variable, because the variable is a valid destination for the message. For example, "Set Me" may be the message that triggers a change in the Object Reference Variable's value. The message is not passed on to the object referred to by the variable.

The statement:

```
send "Set Me" to myObjectRef.object
```

sends the "Set Me" message to the component referred to by myObjectRef, where it might have a completely different effect.

Changing the value of an Object Reference Variable

To change the reference held by an Object Reference Variable, use the following syntax:

```
set objectvar to object
```

where *objectvar* is the name of the Object Reference Variable being accessed and *object* is a mTropolis object specified by name, by relative position (see Chapter 14 for a list of "building blocks" that can be used to specify objects), or by the "object path field" syntax. A number of examples follow:

```
-- point to the parent of the
-- element that
-- contains the variable:
set myObjectRef to element.parent

-- point to the scene:
set myObjectRef to scene

-- point to the message source's
-- parent:
set myObjectRef to source's parent

-- point to the message source's
-- element:
set myObjectRef to source's element

-- point to an element by name:
set myObjectRef to myBigBlueButton

-- change the value using a string
-- that contains a path specification:
set myObjectRef to \
  "/myProj/mySect/mySub/Scene/Bob"

-- The variable's "path" attribute
-- could also be used
-- to set the value using a
string:
set myObjectRef.path to \
  "/myProj/mySect/mySub/Scene/Bob"
-- The path could also be set using a
-- String Variable
-- instead of a string constant. In
-- this case,
-- the "path" attribute must always
-- be used:
set myObjectRef.path to myString
```

Clearing the value of the Object Reference Variable

The Object Reference Variable can be "reset" so that it no longer refers to an object by setting its value to the special value none. For example:

```
set myObjectRef to none
```

Alternatively, the "path" attribute of the Object Reference Variable can be set to a null string. For example:

```
set myObjectRef.path to ""
```

Detecting an empty Object Reference Variable

An Object Reference Variable that does not refer to an object or that refers to an invalid object (an object that does not exist) has the value none. In addition, its "fullpath" attribute contains a null string.

For example, suppose myObjectRef has been configured to point to an element that does not exist. The following script could be used to send a message based on this fact:

```
if myObjectRef = none then \
  send "Invalid Reference"
```

Alternatively, the "fullpath" attribute can be examined:

```
if myObjectRef.fullpath = "" then \
   send "Invalid Reference"
```

Accessing attributes of the referenced object

Any attributes of the object pointed to by the Object Reference Variable can be targeted using the syntax:

```
objectvar.object.attribute
```

where *objectvar* is the name of the Object Reference Variable and *attribute* is the name of an attribute of the referenced object. See Chapter 15, "Attributes," for more information on attributes.

For example, the width of the referenced object can be retrieved:

```
set myInt to
myObjectRef.object.width
```

Similarly, the attributes of a referenced object can be changed. For example:

```
set myObjectRef.object.position to \
(40,50)
```

Attributes of the Object Reference Variable

The Object Reference Variable has a number of attributes that can be accessed via Miniscript. For general information about using attributes, see Chapter 15, "Attributes." The table below describes attributes for the Object Reference Variable in a format similar to that found in "Attribute Descriptions" in Chapter 15.

Object Reference Variable Attribute	Type	Get	Set	Description
fullPath	String	•		The absolute, resolved path to the referenced object.
object	Object	•	•	The referenced object.
path	String	•	•	The path to the object as specified in the Object Path field. Changing this attribute changes the value displayed within the **Object Reference Variable** dialog box.

Attributes of the Object Reference Variable

Open Application Modifier

The Open Application modifier can be used to launch another application from a mTropolis title. Either an application (an executable file) or a document file can be specified. If a document is specified, the operating system launches the correct application to view that document (if such an application is available) just as if the end user had opened that document.

Components of the **Open Application Modifier** dialog box are described below.

*The **Open Application Modifier** dialog box*

Execute When pop-up menu
Use this pop-up menu to select the message that causes the specified project/title to be opened and the current one to be closed. For a complete discussion of this menu, see "When Pop-Up Menu and Message/ Command Pop-Up Menu Options" in Chapter 13.

If the application or document specified in the Document/Application field cannot be found, mTropolis sends an Open Application Failed message to the modifier's parent. See "Open Application Failed" in Chapter 13. Note that if an error occurs after the modifier finds the application or document to open, this message is not sent.

Specifications section

Use the controls in this section to specify the application to be opened and other options.

Document/Application field
Use this field to specify the full path to the application or document to open. Enter a path as described below, or click the **Choose** button to select an application or document file using a standard file dialog box.

Paths specified in this way are not resolved until run-time, so a file that does not currently exist can be specified. Note also that the full path name is limited to a maximum of 255 characters.

- **Entering the Text of an Absolute or Relative Path:** Enter an absolute or relative path to an application or document in this field. This path must be specified using the standard colon-delimited Mac OS path syntax. The path is translated as appropriate for the current platform at run-time.

  An absolute path can be specified by starting with the name of the disk volume and continuing down through the disk hierarchy. For example:

  ```
  My CD:My Folder:My mTitle
  ```

  A path relative to the location of the currently running project or title can be specified by using colons to move up the filesystem hierarchy. For example, to indicate an application in the same folder as the currently running project, use the syntax:

  ```
  :MyApplicationName
  ```

To indicate a file in the next folder or volume up the hierarchy, use the syntax:

```
::MyApplicationName
```

Similarly, any number of colons can be used to move up to the desired level of the filesystem hierarchy. Once the desired level has been specified, folders down from that location can be specified. For example, to specify a document named "My Document" found within a folder named "Docs" which is two folders up from the currently-running project, use the syntax:

```
:::Docs:My Document
```

There is one special "token" that can be used in the Document/Application field. The token `<Startup Segment Folder>` can be used to specify the folder that contains the currently-running title. For example, the following file specification would open the document named "Readme," found in the currently-running project's startup segment folder:

```
<Startup Segment Folder>:Readme
```

Choose button

Click this button to display a standard file selection dialog box. Choose the application or document to be opened. Once selected, the absolute path to the chosen file appears in the Document/Application field.

Put application Into background/foreground buttons
Use these buttons to select how the launched application is run:

- **Background Button:** Click this button (the default) to launch the application and run it in the background. The mTropolis title continues to be visible.
- **Foreground Button:** Click this button to launch the application and run it in the foreground. The specified application appears in front of the currently-running mTropolis title.

Quit mTropolis after launching app check box
Check this check box to force the mTropolis title to quit as soon as it has launched the specified application. Note that this setting has no effect in the mTropolis editor's runtime mode.

Attributes of the Open Application modifier

The Open Application modifier has a number of attributes that can be accessed via Miniscript. For general information about using attributes, see Chapter 15, "Attributes." The table below describes attributes for the Open Application modifier in a format similar to that found in "Lists of Attributes by Component Type" in Chapter 15.

Open Application Modifier Attribute	Type	Get	Set	Description
background	Boolean	•	•	When this attribute is set to true, the specified application is opened in the background.
filePath	String	•	•	The contents of the Document/Application path field.
quit	Boolean	•	•	When this attribute is set to true, the mTropolis player quits after launching the specified application.

Attributes of the Open Application Modifier

Open Project Modifier

The Open Project modifier can be used to open a different project or title file during run-time. The currently-running title project or title is closed, while the newly-opened project or title starts running from its beginning.

When used with the mPire plug-in player, this modifier can be used to open mTropolis titles that reside at remote network locations by specifying a URL instead of a standard file path.

In the mTropolis editor, this modifier can be used to open both project files (the editor's file format) and built title files (finished mTropolis titles created with the **File ➛ Build Title** option). However, in a built title being run with a mTropolis player application, this modifier can only open other built title files. Typically this modifier would be used to open other project files during the development stage of a project. However, when it comes time to distribute a built title that contains this modifier, the modifier's "Project/Title Path" field must be updated to refer to the built title version of the project that needs to be opened.

Because this modifier closes the current project, any project that this modifier is added to must be saved (choose **Save** or **Save As** from the **File** menu) before switching to run-time mode for testing.

Components of the **Open Project Modifier** dialog box are described on the following page.

*The **Open Project Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that causes the specified project/title to be opened and the current one to be closed. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Specifications section
Use the controls in this section to specify the project or title file to be opened.

Path or URL field
Use this field to specify the full path to the project or title file to open. Enter a path as described below, or click the **Choose** button to choose a project or title file using a standard file dialog box.

Paths specified in this way are not resolved until run-time, so a file that does not currently

exist can be specified. Note also that the full path name is limited to a maximum of 255 characters.

- **Entering the Text of an Absolute or Relative Path:** Enter an absolute or relative path to a mTropolis project or title file in this field. This path must be specified using the standard colon-delimited Mac OS path syntax. The path is translated as appropriate for the current platform at run-time.

  An absolute path can be specified by starting with the name of the hard disk volume and continuing down through the disk hierarchy. For example:

  ```
  My HD:My Folder:My mTitle
  ```

  A path relative to the location of the currently running project or title can be specified by using colons to move up the filesystem hierarchy. For example, to indicate a project in the same folder as the currently running project, use the syntax:

  ```
  :My mProject
  ```

  To indicate a file in the next folder or volume up the hierarchy, use the syntax:

  ```
  ::My mProject
  ```

  Similarly, any number of colons can be used to move up to the desired level of the filesystem hierarchy. Once the desired level has been specified, folders down from that location can be specified. For example, to specify a project named "My Project" found within a folder named "Projects" which is two folders up from the currently-running project, use the syntax:

  ```
  :::Projects:My Project
  ```

  There is one special "token" that can be used in the Project/Title Path field. The token `<Startup Segment Folder>` can be used to specify the folder that contains the currently-running title. For example, the following file specification would open the mTropolis title named "Title2," found in the currently-running project's startup segment folder:

  ```
  <Startup Segment Folder>:Title2
  ```

- **Entering a URL:** To specify a mTropolis title that resides at a network location, enter a URL in this field. The contents of the path field must be a valid URL that contains a mTropolis title file. For example:

  ```
  http://www.mydomain.com/MyTitle.mfx
  ```

  The syntax of this path is not checked by the modifier. The URL itself is not resolved until the modifier receives the specified "Execute When" message.

  Only the http protocol is supported by this modifier. Note also that only built titles can be opened remotely — project files cannot be opened in this way. Opening a title file over the network only works in the mPire plug-in player. Executing this modifier in the mTropolis editor's run-time mode causes an error to appear in the Message Log window. The project must be built into a title and run in the mPire plug-in player to test this functionality.

  When a new title is opened by the plug-in player, the original settings of the HEIGHT and WIDTH parameters (as specified in the EMBED tag used to embed the title in the web page) are retained. The new title runs in the same space as was created for the first title. The new title is aligned with the top and left sides of the space. Therefore, if the new title is larger than the original, its right and bottom sides will be clipped. If the new title is smaller, the mPire player fills the

excess space with the background color specified in the EMBED tag or the background color of the page (if no BACKGROUND parameter is specified).

Choose button
Click this button to display a standard file selection dialog box. Choose the project or title file to be opened. Once chosen, the absolute path to the chosen file appears in the Project/Title Path field.

Add To Return List check box
When this option is checked, the project that contains this modifier is added to the "return list." The project can be returned to by executing a Return modifier in the newly-opened project.

If this option is selected, when the newly-opened project closes in response to the Close Project command, it will return to the project with the Return modifier. See "Close Project (Message/Command Menu Only)" in Chapter 13.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Open project attributes

The Open Project modifier has a number of attributes that can be accessed via Miniscript. For general information about using attributes, see Chapter 15, "Attributes." The table below describes attributes for the Open Project modifier in a format similar to that found in "Lists of Attributes by Component Type" in Chapter 15.

Open Project Modifier Attribute	Type	Get	Set	Description
addToReturnList	Boolean	•	•	The setting of the **Add To Return List** check box. This attribute is true if the box is checked.
filePath	String	•	•	The contents of the "Path or URL" field.

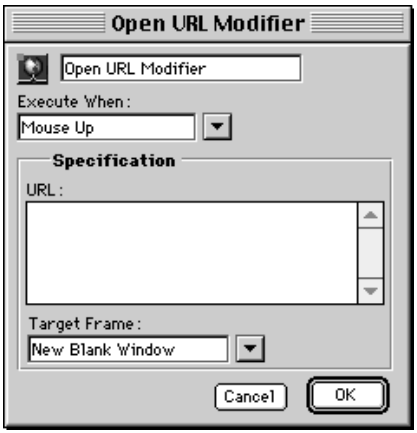
Attributes of the Open Project Modifier

Open URL Modifier

The Open URL modifier opens a new URL from within a mTropolis title running in the mPire plug-in player. The URL can be opened within the current browser window (replacing the currently-displayed mPire title), a new browser window, or a specified frame (when used within a page designed using HTML frames).

!!! Note: This modifier has no effect when used in the mTropolis editor's run-time mode or in the standard mTropolis players. It will only take effect once the project has been built into a title and run in the mPire plug-in player. When this modifier is activated in the editor's run-time mode an error message is displayed in the Message Log.

Components of the **Open URL Modifier** dialog box are described below.



*The **Open URL Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that causes the specified URL to be opened. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Specifications section
Use the controls in this section to specify the URL to be opened and where that URL should be displayed.

URL field
Enter a URL in this text field. The contents of the path field must be a valid URL. For example:

```
http://www.mfactory.com/
```

The syntax of this path is not checked by the modifier. The URL itself is not resolved until the modifier receives the specified "Execute When" message.

Any protocol supported by the Netscape browser can be accessed by this modifier (http, file, ftp, mailto, news, etc.). For file access protocols such as ftp, the end user will need a valid user name and password if the referenced server is protected.

Target Frame pop-up menu
Use this pop-up menu to choose where the new URL is to be displayed. Options include:

New Blank Window: Choose this option to display the contents of the specified URL in a new browser window. This option is the default.

Same Frame: Choose this option to display the contents of the specified URL in the same frame or window as the currently-displayed mPire title. When the new URL is loaded, the mPire plug-in player and its contents are purged from memory along with the currently-displayed web page. To return to the page, the end user must use the browser's navigation controls.

- **Parent Frame:** Choose this option to display the contents of the specified URL in the parent frame of the frame containing the currently-displayed mPire title.
- **Topmost Frame:** Choose this option to display the contents of the specified URL in the topmost frame of the current frame set.
- **User-specified frame name:** A target frame can also be specified by name. Highlight the text of the **Target Frame** pop-up menu and type the name of the frame, enclosed in quotation marks (`"myframe"`). The contents of the specified URL will be displayed in the specified frame.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Miniscript-accessible attributes of the Open URL modifier

The Open URL modifier has a number of unique attributes that can be accessed via Miniscript. For general information about using attributes, see Chapter 15, "Attributes." The table below describes attributes for the Open URL modifier in a format similar to that found in "Lists of Attributes by Component Type" in Chapter 15.

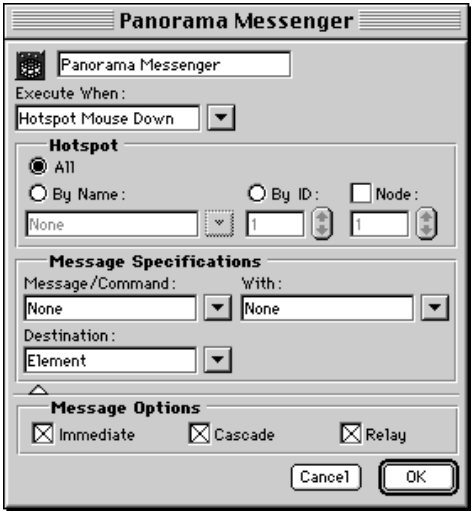
Open URL Modifier Attribute	Type	Get	Set	Description
targetFrame	String	•	•	Specifies the frame in which the URL will be displayed. Possible values are "New Blank Window," "Same Frame," "Parent Frame," "Topmost Frame," or an end user-specified frame name.
url	String	•	•	Gets or sets the URL that will be opened by the modifier, just as if it had been set in the "URL" field. For example: `set urlmod.path set myopenurlmod.url to \ …"http://www.mfactory.com"`

Attributes of the Open URL modifier

Panorama Messenger

The Panorama Messenger sends a message or command in response to end user mouse messages on hotspots or other navigation events within a QuickTime VR panorama movie. This modifier only takes effect when attached to a QuickTime VR panorama movie.

Components of the **Panorama Messenger** dialog box are described below.



*The **Panorama Messenger** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that causes the messenger to activate. This menu has a different set of options than the standard **When** pop-up menu. Note also that these messages are only generated by end user mouse actions over panoramic movies; they cannot be sent to panoramic movies by the author. Options include:

- **None:** When this option is chosen, the modifier will never be executed.
- **Hotspot Mouse Down:** Choose this option to execute the messenger when the mouse button is pressed down and the mouse cursor is over a hotspot in the panorama movie. The "Hotspot" section of the dialog box can be used to select the hotspot that listens for this message. This message arrives with the ID (an integer) of the hotspot as incoming data.
- **Hotspot Mouse Over:** Choose this option to execute the messenger when the mouse passes into a hotspot region. The "Hotspot" section of the dialog box can be used to select the hotspot that listens for this message. This message arrives with the ID (an integer) of the hotspot as incoming data.
- **Hotspot Mouse Outside:** Choose this option to execute the messenger when the mouse leaves (passes out of) a hotspot region. The "Hotspot" section of the dialog box can be used to select the hotspot that listens for this message. This message arrives with the ID (an integer) of the hotspot as incoming data.
- **Panning:** Choose this option to execute the messenger when the panorama is being panned. The panning message is sent continuously (one message each time the screen updates) until panning stops. This message arrives with the new pan angle (a floating-point value representing 0 to 360 degrees) as incoming data.

- **Zooming:** Choose this option to execute the messenger when the panorama is being zoomed (when the field of view is changing). The zooming message is sent continuously (one message each time the screen updates) until zooming stops. This message arrives with the new zoom angle (a floating-point value representing field of view from 0 to 90 degrees) as incoming data.
- **Leaving Node:** Choose this option to execute the messenger when the end user navigates from one node in the QuickTime VR movie to another. This option only has effect when used with multi-node QuickTime VR movies (sometimes called scenes in Apple's QuickTime VR documentation). This message arrives with the ID (an integer) of the node to which the end user is navigating as incoming data.

Hotspot section

Use the controls in this section to specify a hotspot when the modifier is configured to execute on a hotspot message such as Hotspot Mouse Down.

All button
Click this button to listen for hotspot messages generated by any hotspot in the movie.

By Name button and pop-up menu
Click this button to listen for hotspot messages generated only by the hotspot named in the accompanying pop-up menu. Any named hotspots in the movie appear in the pop-up menu.

By ID button and field
Click this button to listen for hotspot messages generated only by the hotspot with the ID number selected in the accompanying field. Each hotspot in a movie has a unique ID number with a value between 1 and 255. To use this feature, you must make note of which hotspots map to which ID numbers when creating hotspots in your QuickTime VR movie.

Node check box and field
Click this check box to specify in which node the hotspot specified in the "By Name:" or "By ID" section is found. Leaving this box unchecked (the default) causes the messenger to listen for hotspot messages from all hotspots (with the specified name or ID) across all nodes.

Message specifications

All Messenger modifiers have the same controls in the Message Specification section of their dialog boxes:

Message/Command pop-up menu
Use this pop-up menu to select the message or command to be sent when the messenger is activated. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

With pop-up menu
Use this menu to select a value to be sent with the message or command. The choices on this pop-up menu are None, Incoming Data, and any variables to which the element has access. These menu items are described in more detail below:

- **None**: This is the default selection. No value is sent with the message.
- **Incoming Data**: Choose this option to configure the messenger to send the incoming data (the value that arrives with the message that has been configured to trigger the messenger), with the outgoing message or command. Incoming data values for each of the special panorama

messages are described in the "Execute When pop-up menu" section.

- **Variables**: To send the value of a variable modifier with the outgoing message, select its name from the submenus available in the second section of the **With** pop-up menu. All variable modifiers currently available to a messenger from its position in the project's hierarchy are shown. These variable modifiers are only accessible to those modifiers or Miniscripts in its "scope." See "Variable Scopes" in Chapter 13.
- **Constant Data Value**: To send a constant data value, highlight the With text field and type the value to be sent. The value should be entered with the same syntax as if it were being used in a Miniscript modifier. The syntax of the expression entered is checked when **OK** is clicked. Any appropriate mTropolis data type can be sent.

Destination pop-up menu
Use this menu to choose the destination for the message or command sent from this modifier. For a complete discussion of this menu, see "The Destination Pop-Up Menu" in Chapter 13.

Message options
All **Messenger** dialog boxes have a Message Options section that can be revealed by clicking the triangle at the bottom of the dialog box. Options in this section are:

Cascade check box
When this box is checked, the message "cascades" from the targeted destination to all components in the hierarchy below it. Uncheck this option to configure the messenger to send its message only to the modifiers in a targeted element. "Cascade off" is equivalent to the path of an environment message sent by mTropolis.


Note: This option has no effect when sending a command.

Immediate check box
By default, messages or commands are sent immediately when the messenger is activated. Uncheck this box to configure the messenger to send its message or command only after the current thread of messages has ended. This option can be used to avoid system overload if the message queue becomes too deep.

Relay check box
By default, messages travel from element to element in the hierarchy activating any and all elements that respond to the message. Uncheck this box to activate only the first modifier in the path of the message that is configured to respond.


Note: This option has no effect when sending a command.

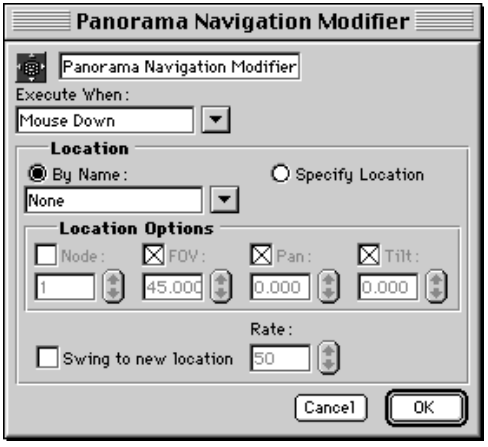
Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Panorama Navigation Modifier

The Panorama Navigation modifier can be used to control the display of QuickTime VR panorama movies. Upon receipt of a message, the modifier changes the view shown in the movie from the currently-displayed location to a named location in the movie or a location specified by node, pan, tilt, and field of view parameters. This modifier only takes effect when attached to a QuickTime VR panorama movie.

Components of the **Panorama Navigation Modifier** dialog box are described below.



*The **Panorama Navigation Modifier** dialog box*

Execute When pop-up menu
Use this pop-up menu to choose the message that causes the Panorama Navigation modifier to execute. Upon the receipt of this message, the QuickTime VR movie updates to show the view defined in the Location section of the **Panorama Navigation Modifier** dialog box. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

!!! Note: Because end user mouse events are used to navigate QuickTime VR movies, it is not usually possible to execute this modifier based on mouse events such as "Mouse Up." To make a QuickTime VR panorama unresponsive to end user mouse events and instead pass those events to mTropolis, use the movieClick attribute, described in Chapter 15.

Location section

Use the controls in this section to specify the new view shown in the QuickTime VR movie.

By Name button and pop-up menu
Click this button (the default) to specify a new location by name. The pop-up menu displays a list of all locations defined in the movie. Locations can be defined and named using the Panorama Location Namer utility, found in the QuickTime Utilities folder of the mTropolis distribution and described in Appendix B, "Panorama Location Namer."

Specify Location button
Click this button to specify a new location using QuickTime VR parameters. The Location Options check boxes become enabled:

- **Node Check Box and Field:** Check this check box (the default) to specify the node to be displayed. If this box is not checked, the current node is used.
- **Pan Check Box and Field:** Check this check box (the default) to specify the pan angle to be displayed. This value is a floating-point number representing degrees from 0 to 360. If this box is not checked, the current panning is used.

- **Tilt Check Box and Field:** Check this check box (the default) to specify the tilt angle to be displayed. This value is a floating-point number representing degrees from -90 to 90. If this box is not checked, the current tilt is used.
- **FOV Check Box and Field**: Check this check box (the default) to specify the field of view to be displayed. This value is a floating-point number representing degrees from 0 to 90 (0 is fully zoomed in, 90 is fully zoomed out).

Swing to New Location check box
Check this check box (unchecked by default) to make the view "swing" (animate) to its new location. If this box is unchecked, the view changes instantly to the new settings. When this box is checked, the Rate field also becomes enabled.

Rate field
Use this field to specify the speed of the "swing" animation as the QuickTime VR view changes to its new settings. Valid values range from 1 (slowest) to 100 (fastest). The default is 50. The speed of the actual transition may vary depending upon the speed of the target machine.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Path Motion Modifier

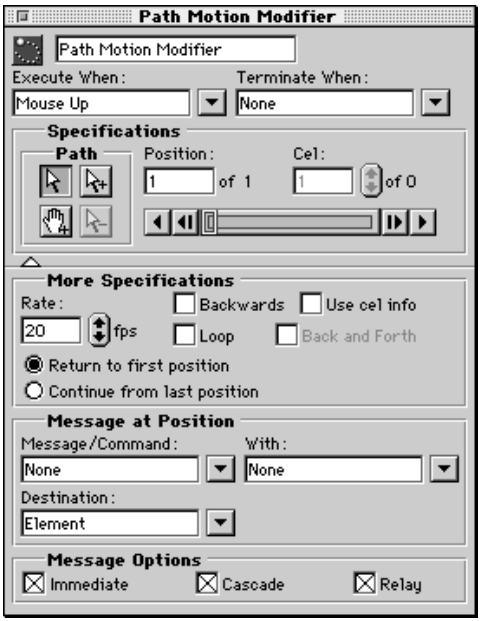
The Path Motion modifier can be used to add motion along a path to an element. Any child elements of the element that contains the Path Motion modifier also move along the path.

This modifier is especially useful when used with mToons. It can be used to specify which cels are visible at each point along the path. Multiple Path Motion modifiers can be used to define many paths for a single animation. Each of these modifiers can be configured to respond to different messages, allowing the element to move on a specific path according to specific conditions.

The motion path is stored in the modifier itself. Once a path is defined, it can be used again by dragging a copy onto another Graphic element, movie or mToon.

In addition to defining motion paths, this modifier can be used to send messages from any point along the motion path.

Components of the **Path Motion Modifier** dialog box are described below:



*The **Path Motion Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that enables the path motion. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Terminate When pop-up menu
Use this pop-up menu to choose the message that stops the path motion. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Specifications section

The controls in this section allow you to create the motion path and specify animation cels for individual positions.

Path tools
A number of tools aid in the creation of the motion path:

Selection Arrow: The selection arrow allows you to move any position on the motion path. When a position is selected, the modifier's associated element will automatically move to that position on the path.

Add Position Tool: Use this tool to define a new position on the path. Clicking on the scene after the last selected point adds a position to the path.

To add a position between two points on an existing path, select the first of the two points with the Selection Arrow, choose Add Position tool, and click on the scene before the next point. If the Option key is depressed while the Add Position tool is selected, the Select Arrow appears, allowing a point to be selected and/or moved.

Drag Path Tool: This tool allows you to Mouse Down and drag an element on its scene to define a motion path. When the mouse is released, the points of the modifier's path are automatically drawn. If the Option key is depressed while the Drag Path tool is selected, the Select Arrow appears, allowing a point to be selected and/or moved.

Delete Position Tool: This tool allows you to select and delete positions anywhere along the motion path. If the

Option key is depressed while the Delete Position tool is selected, the Select Arrow appears, allowing a point to be selected and/or moved.

Position field
This field shows the number of the current position on the path. The step controls below this field allow you to step forward or backward through the positions of the motion path one at a time.

The Position and Cel fields work together, allowing the author to assign a specific cel of an animation to a selected position on the motion path.

Cel field
This field shows the number of the current cel and the total number of cels in the animation. Use the arrow controls to scroll through the cels of the animation. Each point on the motion path can have a different cel associated with it.

Animation Controller bar
These basic controls allow you to step, or play through the positions forward or backward. The arrows to the right and left allow you to preview the motion path forward and backward while the dialog box is open.

To use this feature, the compression method used to compress the animation must allow cels to be randomly accessible. See "Compression" in Chapter 1.

More specifications section
Select the triangle at the bottom of the path motion dialog box to toggle the display of more options for path animations:

Rate field
Select the rate, in frames per second, at which the animation will be played over the path. This setting overrides any previous rate setting in the element's **Element Info** dialog box.

Return to First Position button
Click this option to reset the element to its original starting position when the end of the path is reached.

Continue from Last Position button
Click this option to continue path motion from the last position of the path whenever path motion restarts.

Backwards check box
Check this box to play through the motion positions from the last position to the first position.

Use Cel info
Check this box to use the cel positions defined with the "Cel" field in the dialog box's Specifications section.

When unselected, cels will be mapped sequentially to each position. If the number of cels is less than the number of positions on the path, the animation sequence will loop. If the number of cels is greater than the number of positions on the path, the number of cels displayed will be divided among the number of defined positions.

Loop check box
Check this box to make the path motion loop. That is, the element moves along the motion path continuously from beginning to end positions.

Back & Forth check box
Check this option to play the path motion from start to end, then backward from end to start.

Message at position section
Use the controls in this section to select a message to be sent when the element is at the current position in the motion path (the position shown in the "Position" field of the **Path Motion Modifier** dialog box). A different message can be selected for each point in the motion path. The default is to send no message.

Message/Command pop-up menu
Use this pop-up menu to select the message or command to be sent when the current path motion point is reached. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

With pop-up menu
Use this menu to choose a value to be sent with the message or command. The choices on this pop-up menu are None, Incoming Data, and any variables to which the element has access. These menu items are described in more detail below:

- **None:** This is the default selection. No value is sent with the message.
- **Incoming Data:** Choose this option to configure the messenger to send the incoming data (the value that arrives with the message that has been configured to trigger the messenger), with the outgoing message or command.
- **Variables:** To send the value of a variable modifier with the outgoing message, choose its name from the submenus available in the second section of the **With** pop-up menu. All variable modifiers currently available to a messenger from its position in the project's hierarchy are shown. These variable modifiers are only accessible to those modifiers or Miniscripts in its "scope." See "Variable Scopes" in Chapter 13.
- **Constant Data Value:** To send a constant data value, highlight the With text field and type the value to be sent. The value should be entered with the same syntax as if it were being used in a Miniscript modifier. The syntax of the expression entered is checked when **OK** is clicked. Any appropriate mTropolis data type can be sent.

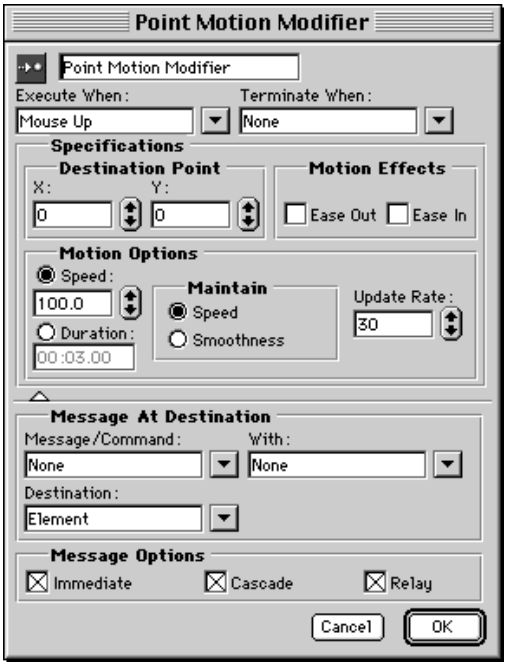
Destination pop-up menu
Use this menu to choose the destination for the message or command sent from this modifier. For a complete discussion of this menu, see "The Destination Pop-Up Menu" in Chapter 13.

Message options
The standard message options are available for controlling the behavior of the selected message. See "Message Options" in this chapter for a complete description of the controls in this section.

Point Motion Modifier

The Point Motion modifier moves an element in a straight line from its current position to a new position.

Components of the **Point Motion Modifier** dialog box are described on the following page.



*The **Point Motion Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to select the message that enables the Point Motion. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Terminate When pop-up menu
Use this pop-up menu to choose the message that stops the Point Motion. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Specifications section
Use the controls in this section to adjust various Point Motion parameters.

Destination point
Use the fields in this section to specify the point that the element moves to. This position is specified relative to the origin of the element's parent.

- **X Field**: The new horizontal position.
- **Y Field**: The new vertical position.

Motion effects
Use the check boxes in this section to apply various special effects to the element's Point Motion.

- **Ease Out**: Check this check box to cause the element to speed up as it begins its point motion. The element accelerates from stopped to its maximum speed. If the **Speed** button is clicked, the specified speed will be used as the maximum speed. If the **Duration** button is clicked, the maximum speed is computed taking into account the fact that the element begins moving slowly.
- **Ease In**: Check this check box to cause the element to slow down as it approaches the destination point. The element decelerates from its maximum speed to a complete stop.

Motion options section
Use the controls in this section to specify the appearance of the element's Point Motion.

Speed button and field
Click this button (the default) to specify the maximum speed of the element in pixels per second. The default is 100.0.

Duration button and field
Click this button to specify a duration for the Point Motion instead of a constant speed. The maximum speed of the element is computed based on this value, any motion effect that may be selected, and the element's distance from the destination point.

Maintain options
The following options are used to control how mTropolis handles Point Motion updates in the event that the requested update rate (see below) cannot be maintained:

- **Maintain Speed Button**: When this option is clicked (the default), the updated position of the element is calculated according to how much time has passed since the last update. The Point Motion may appear less smooth than intended, but the overall speed of the element will be maintained. This option is always selected if the **Duration** button is clicked.
- **Maintain Smoothness Button**: When this option is clicked, the position of the element is always changed by a fixed number of pixels. As a result, the specified speed of the motion may not be maintained but the motion will always appear smooth. This option cannot be chosen if the **Duration** button is clicked.

Update Rate field
The number of times per second that the position of the element is updated. The default is 30 updates per second. If the requested rate cannot be achieved, mTropolis adjusts the motion based on the settings in the "Maintain" section.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Message at Destination section
Click the open/close triangle at the bottom of the **Point Motion Modifier's** dialog box to reveal a standard **Message Specifications** dialog box. The specified message will be sent when the element reaches the destination specified in the X and Y fields.

Attributes of the Point Motion modifier

The Point Motion modifier has a number of unique attributes that can be accessed via Miniscript. For general information about using attributes, see Chapter 15, "Attributes." The table below describes the attributes in a format similar to that found in "Lists of Attributes by Component Type" in Chapter 15. Note that setting any of the attributes described below causes an immediate change in the Point Motion of the element.

Related reading

The following related topics may also be of interest: "Vector Motion Modifier" in this chapter, "Motion and Transition Messages" in Chapter 13, and "Position" in Chapter 15.

Point Motion Modifier Attribute	Type	Get	Set	Description
destination	Point	•	•	The destination for the element as set in the modifier dialog box's "X" and "Y" destination fields.
duration	Float	•	•	The duration for the motion in seconds.
speed	Float	•	•	The desired speed for the motion, in pixels per second.
updateRate	Integer	•	•	The desired number of motion updates per second.

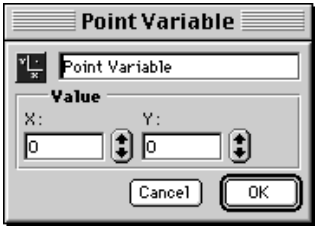
Attributes of the Point Motion modifier

Point Variable

The Point Variable modifier stores a value of point data type (a pair of integer values). Points are commonly used for storing and/or setting the screen location of elements. For example, the position of an element can be changed during run-time by placing multiple Point Variables on the element with different values in each. New positions can then be set using the Miniscript modifier.

Note: The location of an element is measured in pixels relative to its parent's origin (the upper left corner of the parent).

Components of the **Point Variable** dialog box are described below:



*The **Point Variable** dialog box*

Variable's Name field
This editable text field can be used to change the variable's name. If the variable will be referenced in a Miniscript modifier, it is good programming practice to make the name a single word.

Modifier icon
This icon identifies the variable modifier's type.

Value fields
Enter the variable's initial value here or use the up/down arrow buttons to set the value. There are two fields:

- **X**: The horizontal value.
- **Y**: The vertical value.

Note: Y-coordinates in mTropolis are *screen coordinates*; zero is at the top of the screen and values increase as you move down to the bottom of the screen. This differs from standard mathematical (Cartesian) coordinates, in which the Y-value *decreases* in the downward direction.

The value is checked for validity when the **OK** button is clicked. The value can be changed during run-time. See "Setting Values of Variable Modifiers" in Chapter 14 for details regarding getting and setting the value of this variable modifier.

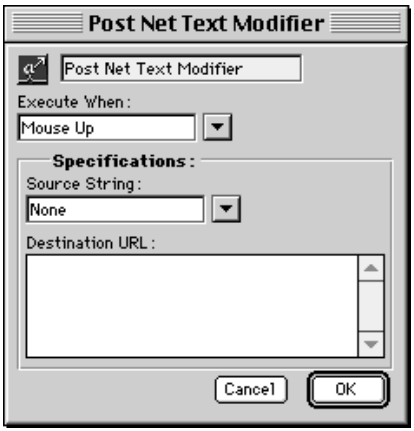
Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Post Net Text Modifier

The Post Net Text modifier sends a string value to a specified URL. What the URL does with the posted string depends upon the network protocol being used and the actual contents of that URL.

The components of the **Post Net Text** modifier dialog box are described below.



*The **Post Net Text Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to select the message that causes the text to be posted to the specified URL. For a complete discussion of this menu, see "When Pop-Up Menu and Message/ Command Pop-Up Menu Options" in Chapter 13.

When the text has been successfully posted to the specified URL, mTropolis sends a Net Operation Completed message to the modifier's parent. See "Net Operation Completed" in Chapter 13.

If text cannot be posted to the specified URL, mTropolis sends a Net Operation Failed message to the modifier's parent. See "Net Operation Failed" in Chapter 13.

If the posting process is stopped (by setting the modifier's stop attribute to true), mTropolis sends a Net Operation Stopped message to the modifier's parent. See "Net Operation Stopped" in Chapter 13.

Specifications section

Use the controls in this section to specify the source text to be posted and its destination URL.

Source String pop-up menu
Use this pop-up menu to specify the string to be posted. Options include:

- **None**: This is the default. No String Variable is selected. Text will not be posted.
- **Incoming Data**: Choose this option to post any text sent as incoming data (the value that arrives with the message that has been configured to trigger the Post Net Text modifier).
- **String Variable**: To post text contained in a String Variable, choose its name from the submenus available in the second section of the **Destination String** pop-up menu. Any String Variable modifier within the "scope" of the Post Net Text modifier can be chosen.
- **Constant String Value**: To send a constant string value, highlight the Source String text field and type the string to be sent. The value should be entered with the same syntax as if it were being used in a Miniscript modifier (the string must be enclosed in quotation marks). The syntax of the expression entered is checked when **OK** is clicked.

Destination URL field
Enter the URL to which text will be posted in this text field. The contents of this field must be a valid URL. For example:

```
http://www.mydomain.com/mStrings.cgi
```

The syntax of this path is not checked by the modifier. The URL itself is not resolved until the modifier receives the specified "Execute When" message.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Miniscript-accessible attributes of the Post Net Text modifier

The Post Net Text modifier has a number of unique attributes that can be accessed via Miniscript. For general information about using attributes, see Chapter 15, "Attributes." The table below describes the attributes in a format similar to that found in "Lists of Attributes by Component Type" in Chapter 15.

Post Net Text Modifier Attributes	Type	Get	Set	Description
stop	Boolean		•	Set to true to stop net text posting. Once stopped, Net Operation Stopped message is sent to the modifier's parent. See "Net Operation Stopped" in Chapter 13.
url	String	•	•	Gets or sets the URL that will be accessed by the modifier, just as if it had been set in the "Destination URL" field.

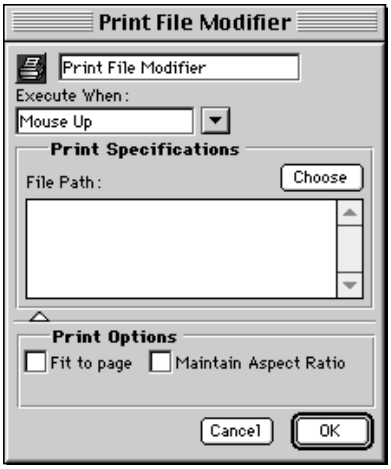
Attributes of the Post Net Text modifier

Print File Modifier

The Print File modifier can be used to print a specified bitmap graphics file from a mTropolis project. The chosen file is external to the project, and must be manually included with built title files.

The Print File modifier prints PICT files on Mac OS platforms and BMP (Windows Bitmap) files on Windows platforms.

Components of the **Print File Modifier** dialog box are described below.



*The **Print File Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that causes the specified file to be printed. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Print specifications section
Use this section to choose the file to printed.

File Path field
Use the File Path field to specify the full path to the PICT or BMP file. Enter a path as described below, or click the **Choose** button to select a project or title file using a standard file dialog box.

Paths specified in this way are not resolved until run-time, so a file that does not currently exist can be specified. Note also that the full path name is limited to a maximum of 255 characters.

- **Entering the Text of an Absolute or Relative Path**: Enter an absolute or relative path to the file to be printed in this field. This path must be specified using the standard colon-delimited Mac OS path syntax. The path is translated as appropriate for the current platform at run-time.

  An absolute path can be specified by starting with the name of the disk volume and continuing down through the disk hierarchy. For example:

  ```
  My CD:My Folder:My PICT
  ```

  A path relative to the location of the currently running project or title can be specified by using colons to move up the filesystem hierarchy. For example, to indicate a file in the same folder as the currently running project, use the syntax:

```
:My PICT
```

To indicate a file in the next folder or volume up the hierarchy, use the syntax:

```
::My PICT
```

Similarly, any number of colons can be used to move up to the desired level of the filesystem hierarchy. Once the desired level has been specified, folders down from that location can be specified. For example, to print a file named "My PICT" found within a folder named "Print Files" which is two folders up from the currently-running project, use the syntax:

```
:::Print Files:My PICT
```

There is one special "token" that can be used in the File Path field. The token `<Startup Segment Folder>` can be used to specify the folder that contains the currently-running title. For example, the following file specification would print the file "My PICT," found in the currently-running project's startup segment folder:

```
<Startup Segment Folder>:My PICT
```

Choose button
Click this button to display a standard file selection dialog box. Choose the PICT or BMP file to be printed. Once selected, the absolute path to the chosen file appears in the File Path field.

Print Options section
Click the open/close triangle at the bottom of the **Print File Modifier** dialog box to display this section. Use the controls to specify various printing options.

Fit to Page check box
Check this check box to force the printed content to be scaled to fit the page.

Maintain Aspect Ratio
Check this check box to cause the printed content to keep its original aspect ratio when scaled. This option is only relevant if the "Fit to Page" check box is also checked.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

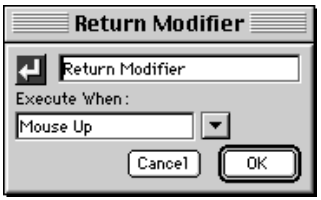
OK button
Click **OK** to accept changes made to this modifier.

Miniscript-accessible attributes of the Print File modifier
The Print File modifier has a number of unique attributes that can be accessed via Miniscript. For general information about using attributes, see Chapter 15, "Attributes."

Return Modifier

Upon receipt of a specified message, the Return modifier executes a scene change to the first scene in the "scene return list." Note that on returning to a scene, it is removed from the return list and the next scene in the list (if any) becomes the next scene to be returned to.

Components of the **Return Modifier** dialog box are described below:



*The **Return Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to select the message that triggers the scene return. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Save and Restore Modifier

The Save and Restore modifier can be used to save data stored in a mTropolis variable to a file and restore that data upon receipt of specified messages. Any previously created mTropolis variable of any type (including end user-defined Compound Variables) can be saved and restored. This sort of functionality is essential for titles that are designed to be used over multiple sessions. Using the Compound Variable (see "Compound Variable" in this chapter) with this modifier allows the author to save and restore complicated states.

The file can be saved to a number of preset locations or the end user can be prompted to choose a location for the file.

Components of the **Save and Restore Modifier** dialog box are described below.

*The **Save and Restore Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Save When pop-up menu
Use this menu to choose the message that causes the chosen variable to be saved. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Restore When pop-up menu
Use this menu to select the message that causes the selected variable to be restored (read from the previously saved file). The specified mTropolis variable is set to the values contained in the saved file. Note that if a "restore" operation is attempted before a "save" operation has been performed (a file with the specified name and location does not yet exist), the current value of the specified variable is not changed. For a complete discussion of the options in this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Variable to Save/Restore pop-up menu
Use this pop-up menu to choose the variable to be saved and/or restored. When saving, the selected variable is written to the specified file. If the modifier is configured to restore, this field must be used to select a variable. The data saved in the specified file is restored into the specified variable. Selections are described in more detail below:

- **None**: This is the default selection. No data can be saved or restored.
- **Incoming Data**: Choose this option to configure the modifier to save the incoming data (the value that arrives with the message that has been configured to trigger the messenger), to the specified file. This selection cannot be used to restore data.
- **Variables**: Choose the name of a variable from the submenus available in the second section of this pop-up menu. When the modifier is triggered to save, the value contained within a selected variable is written to the specified file. When the modifier is triggered to restore, data from the specified file is placed in the selected variable. All variable modifiers currently available to a messenger from its position in the project's hierarchy are shown. See "Variable Scopes" in Chapter 13.
- **Constant Data Value**: Though a constant data value can be entered into this field, constants cannot be saved to files nor can data be restored into a constant. Using the field in this way is the equivalent of selecting "None."

File Location section
Use the controls in this section to specify the location of the file to be saved or restored.

Path Options pop-up menu and File Name field
Use this pop-up menu to select the location in which the file (specified in the "File Name:" field) will be saved. Options are:

- **Ask User**: Choose this option to display a file selection dialog box in which the end user selects the path and file to be written to or loaded from. The file name specified in the "File Name" field becomes the "default" file name that the end user can change.
- **Title Folder**: Choose this option to save or restore the data file in the same location as the title startup segment. See "Build Title" in Chapter 1 for more information about startup segments and built titles.
- **Mac Prefs Folder**: Choose this option to save or restore the data file in the Mac OS "Preferences" folder, found in the Mac OS "System Folder." This option is ignored and replaced with the "Ask User" option if the title is run onr Windows platforms.

Use the File Name field to enter the name of the file to be saved. The default is `save.dat`.

Specify Path button and pop-up menu
Click the **Specify Path** button to specify a String Variable that contains a path, or to specify the path as a literal string.

- **String Variable**: To specify a String Variable that contains an absolute or relative path to the save/restore file, simply choose it from the pop-up menu. When the modifier is executed, the String Variable must contain a valid path string as described below.
- **Text of Absolute or Relative Path**: To enter an absolute or relative path to a folder, highlight the text of this field and enter a path. This path must be specified using the standard colon-delimited Mac OS path syntax.

  An absolute path can be specified by starting with the name of the hard disk volume and continuing down through the disk hierarchy. For example:

  ```
  "My HD:My Folder:My Subfolder"
  ```

- A path relative to the location of the startup segment can be specified by using colons to move up the filesystem hierarchy. For example, to indicate a folder in the same folder as the startup segment, use the syntax:

  ```
  ":My Folder"
  ```

  To indicate a folder in the next folder/volume up the hierarchy, use the syntax:

  ```
  "::My Folder"
  ```

  Similarly, any number of colons can be used to move up to the desired level of the filesystem hierarchy. Once the desired level has been specified, folders down from that location can be specified. For example, to specify a folder named “Scores” found within a folder named “Data” which is two folders up from the startup segment, use the syntax:

  ```
  ":::Data:Scores"
  ```

  Three “tokens” can be used at the beginning of the file specification. The token `<boot drive>` can be used to denote the root directory of the drive that booted the computer. The tokens `<title folder>` or `<startup segment folder>` can be used to denote the folder in which the title’s startup segment resides.

Note: Paths specified this way are not resolved until run-time, so a folder that does not currently exist can be specified.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

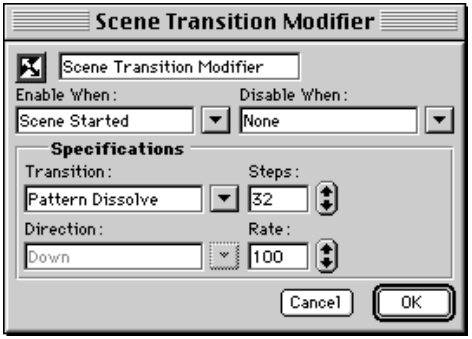
OK button
Click **OK** to accept changes made to this modifier.

Scene Transition Modifier

The Scene Transition modifier adds a special transition effect to a scene. Transition effects are often desired when changing from one scene to the next. The **Transition** pop-up menu lists options such as dissolve, slide, push, etc. The transition becomes visible only on the next scene change. Note that time-based media (video, mToon, or sound elements) in the new scene do not begin playing until the transition effect has finished.

The Scene Transition affects only the “bounding box” defined by media elements within the scene. For example, if the scene contains only a single PICT element in the center of the screen (that does not take up the entire scene) and there is no media linked to the scene itself, the transition will appear to affect only the PICT element.

Components of the **Scene Transition Modifier** dialog box are described below:



*The **Scene Transition Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Enable When pop-up menu
Use this pop-up menu to choose the message that enables the scene transition. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

This modifier is commonly activated on a Scene Started message. However, other messages could be chosen to activate a transition. For example, when an end user clicks an element to turn right, a push right could be enabled, while clicking an element to turn left could enable a push left.

mTropolis sends a Scene Transition Ended message when a scene transition is complete. See "Scene Transition Ended" in Chapter 13.

Disable When pop-up menu
Use this pop-up menu to choose the message that disables the scene transition. Note, however, that a transition cannot be disabled while it is being played.

Specifications section
Use the controls in this section to choose the type of transition.

Transition pop-up menu
This pop-up menu contains a list of all available transition types. Options include:

- Pattern Dissolve
- Random Dissolve
- Fade
- Push
- Slide
- Wipe
- Zoom
- *Steps Field*

Use this field to specify the number of steps between the start and finish of the transition. The greater this number, the finer (and slower) the transition.

Direction pop-up menu
Some transitions (such as Slide, Push, and Wipe) have associated directions. Use this pop-up menu to select a direction for the transition effect.

Rate field
Use this field to select the speed of the transition. Values for this field can range from 1 (slowest) to 100 (fastest). Note that the speed setting is processor dependent — different

computers will render the transition at different speeds, based on the speed of the computer.

!!! Note: On faster computers, even medium speed settings may make the transition happen too quickly to be noticed. If a transition seems not to be working, try lowering the rate to a very low setting (0 or 1) and increasing the number of steps. Then incrementally increase the rate until the effect appears at the desired speed.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Set Modifier

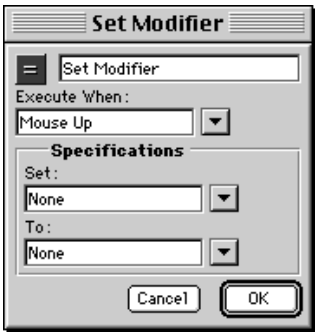
The Set modifier can be used to change the value of a variable, to a new value on receipt of a specified message. This modifier is useful for simple functions, such as resetting scores to 0 when leaving a scene. It can also be used in more complex operations, such as setting the value of an aliased variable to the value of another variable during run-time.

Using the Set modifier to change the value of a variable is equivalent to the Miniscript assignment statement:

```
set variableName to expression
```

where *variableName* is the name of a valid variable in the current project and *expression* is a Miniscript expression that evaluates to the proper type for storage in the variable.

Components of the **Set Modifier** dialog box are described below:



*The **Set Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that triggers the set operation. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Specifications section
The controls in this section specify the variable to be set and the new value.

Set pop-up menu
Choose the variable to be changed from this pop-up menu. Options include:

- **None**: This is the default selection. No variable is changed on receipt of the specified message.

- **Incoming Data**: This option is equivalent to selecting "None."
- **Variable**: Any variable modifier in the Set modifier's scope can be specified to be changed. The variable modifiers will be visible in the **Set** pop-up menu. Variables are only available to those modifiers or Miniscripts in their scope. For information regarding scoping rules, see "Variable Scopes" in Chapter 13.

To pop-up menu
Use this menu to select the data or variable to which the item specified in the **Set** pop-up menu is to be changed. The choices on this pop-up menu are None, Incoming Data and any variables to which the element has access. A constant data value can be sent by highlighting the field and typing in a value. The menu items are described in more detail below:

- **None**: This is the default selection. No set operation is performed (the chosen value is not changed).
- **Incoming Data**: Choose this option to configure the messenger to set the item in the Set field to the incoming data (the value that arrives with the message that has been configured to trigger the messenger).
- **Variables**: To set the item in the Set field to a variable, choose the variable from the submenus available in the second section of the **To** pop-up menu. All variable modifiers currently available to a messenger from its position in the project's hierarchy are shown. These variable modifiers are only accessible to those modifiers or Miniscripts in their "scope." See "Variable Scopes" in Chapter 13.
- **Constant Data Value**: To set the item in the Set field to a constant data value, highlight the "To" text field and type the value to be sent. The value should be entered with the same syntax as if it were being used in a Miniscript modifier. The syntax of the expression entered is checked when **OK** is clicked. Any appropriate mTropolis data type can be sent.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

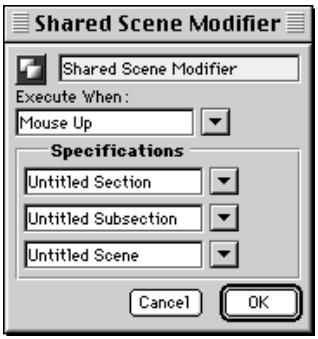
OK button
Click **OK** to accept changes made to this modifier.

Shared Scene Modifier

The Shared Scene modifier can be used to specify a scene to become the shared scene. Note that by default there is always one, and only one, shared scene in a subsection.

The new shared scene replaces the current shared scene. The displaced shared scene becomes a "regular" scene — the first regular scene in the subsection. Any other scenes in the subsection "move" in the scene order to accommodate the new first scene. Note, however, that the currently active scene (the one visible to the end user) does not change except to display the new shared scene as its background. More information about shared scenes can be found in "The Shared Scene" in Chapter 9 and "Shared Scenes" in Chapter 10.

Components of the **Shared Scene Modifier** dialog box are described below:



*The **Shared Scene Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that causes the new shared scene to be set. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Specifications section
Select the scene to become the shared scene from the pop-up menus in this section. The first pop-up menu selects a section, the second selects a subsection, and the third selects the scene for specifying the new shared scene.

The default is the current section, current subsection, and current scene. Note that the scene default should always be changed because if the current scene is targeted to become the shared scene, nothing happens (the shared scene is not changed). Note that this is true when any Shared Scene modifier (no matter where it is located) attempts to make the current scene the shared scene.

To revert to the "automatic" use of the first scene in each subsection as the shared scene, use the "autoShareScene" attribute, described in chapter 15.

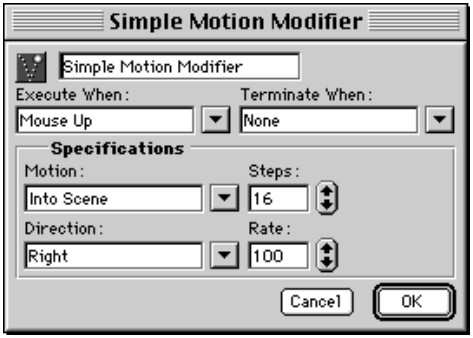
Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Simple Motion Modifier

The Simple Motion modifier applies predefined motion effects to elements. Simple directional motion or random movement can be selected.

Components of the **Simple Motion Modifier** dialog box are described below:



*The **Simple Motion Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that enables the element motion. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Terminate When pop-up menu
Use this pop-up menu to choose the message that stops the element motion. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Specifications section
Use the controls in this section to customize the simple motion.

Motion pop-up menu
Use this pop-up menu to select a basic motion type. Options include:

- **Into Scene**: The object moves into the scene to its current position as defined in the Layout window.
- **Out of Scene**: The object moves off the scene from its current position as defined in the Layout window.
- **Random Bounce**: The object bounces around at random angles within the frame of its parent.

Steps field
Use this field to set the number of positions between start and end points of the motion path. The greater the number, the finer and slower the animation. This field is not available for all motion types.

Direction pop-up menu
If the motion chosen has an associated direction, this pop-up menu becomes visible. Use this menu to select a direction for the simple motion.

Rate field
Use this field to set a speed of the motion. Note that this speed is not absolute, but varies depending on the speed of the playback computer.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

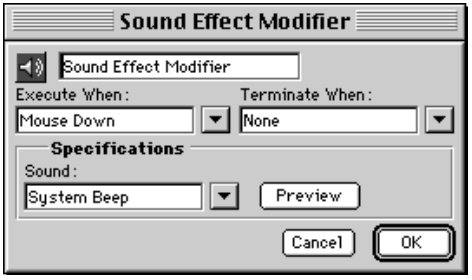
Sound Effect Modifier

The Sound Effect modifier can be used to play sound effects in response to a message.

Sound files can be linked to the project and played from this modifier. Note that the Sound Effect modifier is not the same as a sound element. The Sound Effect modifier is designed for playing a sound once in response to a specific message. Sound elements are more flexible — they can contain modifiers, they can be looped, and they respond to play control commands. The Sound Effect modifier, however, is very useful for setting up simple sound effects.

The Sound Effect modifier can also be used to play a sound across scene changes. Once triggered, Sound Effect modifiers continue to play until the sound ends or the "terminate when" message is received.

Components of the **Sound Effect Modifier** dialog box are described below:



*The **Sound Effect Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that starts the sound effect. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Terminate When pop-up menu
Use this pop-up menu to choose the message that terminates the sound effect. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Sound pop-up menu
Use this pop-up menu to choose a sound that has been linked to the project. Alternatively, choose Link Sound from this menu to link a new sound. A standard file dialog box appears.

The current system beep of your computer is the default setting of this modifier. Note however, that the system beep is different on different machines, so don't rely on this sound being the same during run-time on other machines. Choosing "System Beep" does not cause the actual sound media being used as the system beep to be linked to the mTropolis project. Also, the system beep is a special system sound that does not play in the same way as other sounds — if the system beep is triggered multiple times in quick succession, the "beeps" do not play in tandem (as other mTropolis sounds would), but play one after another.

Preview button
Click this button to preview the sound currently selected in the **Sound** pop-up menu.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

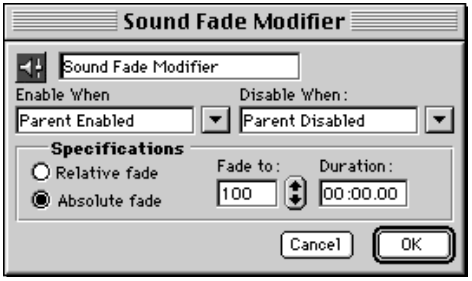
OK button
Click **OK** to accept changes made to this modifier.

Sound Fade Modifier

The Sound Fade modifier smoothly decreases or increases the volume of a sound element. For example, this modifier can be used to obscure the abrupt halt of background audio by fading it to zero volume before the sound ends.

Note that this modifier only affects sound elements and Quicktime elements that play audio. This modifier has no effect when attached to graphic elements, even if the graphic element has a Sound Effect modifier attached.

Components of the **Sound Fade Modifier** dialog box are described below:



*The **Sound Fade Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Enable When pop-up menu
Use this pop-up menu to choose the message that starts the sound fade. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Disable When pop-up menu
Use this pop-up menu to choose the message that stops the sound fade. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Relative Fade button
Click this button to specify a sound fade relative to the sound's current volume. This fade is specified as a percentage from 0 to 10,000% in the "Fade to:" field.

Absolute Fade button
Click this button to specify a sound fade to an absolute volume. This fade is specified as a percentage from 0 to 100% in the "Fade to:" field.

Fade To field
Use this field to specify the volume level to which the sound fades. The meaning of this field changes depending upon which button is clicked:

If the **Relative Fade** button is clicked, this field specifies a percentage of the sound's current volume. Valid values range from 0 (no sound) through 100 (no change to current sound) to 10000 (increase the current sound level by 100 times). Note that the sound can only fade up to its full volume level, no matter what this value is set to (the sound is never amplified by mTropolis).

If the **Absolute Fade** button is clicked, this field specifies a percentage of the sound's full volume. Valid values range from 0 (no sound) to 100 (sound played at full volume). If a value greater than 100 is entered in this field, the sound fades to its full volume and the

value will be shown as 100 the next time this dialog box is opened.

Duration field
Use this field to specify the duration of the fade in minutes:seconds.hundredths format.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

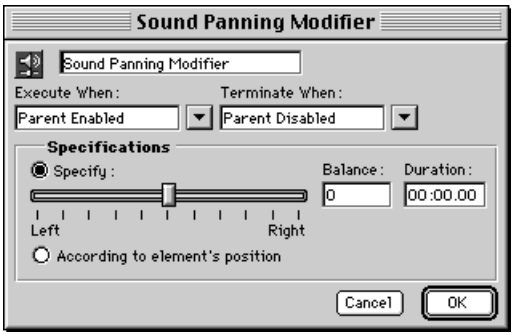
OK button
Click **OK** to accept changes made to this modifier.

Sound Panning Modifier

The Sound Panning modifier changes the stereo position (the left/right balance) of a sound element or the audio in a QuickTime element. Note that this modifier cannot modify a sound being played by the Sound Effect modifier. Note also that sound elements can only be added in the structure view — for details, see "To create a new sound element:" in Chapter 8.

On Mac OS, all sounds can be made to pan. On Windows platforms, only stereo sounds can be made to pan. Mono sounds do not pan.

Components of the **Sound Panning Modifier** dialog box are described below.



*The **Sound Panning Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that activates the sound panning. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Terminate When pop-up menu
Use this pop-up menu to choose the message that ends the sound panning. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Specifications section
Use the controls in this section to specify the type of pan to be performed.

Specify button
Click this button to set a specific panning position for the sound. When this option is selected, three other interface elements become active:

- **Left/Right Slider**: Use this slider to select the pan position of the element. The Balance text field updates to reflect the position of the slider.
- **Balance Field**: This field can be used to enter a numeric pan position. A value of –100 indicates full left, 0 indicates center, and 100 indicates full right. The Left/Right slider updates to reflect the value entered in this field.
- **Duration Field**: Use this field to specify the time it takes for the sound to pan from its current position in the stereo field to the position specified by the Left/Right slider or Balance field. This value should be entered in minute:second.hundredths format. The default, 00:00.00 means that the pan position changes instantaneously.

According to Element's Position button
Click this button to deactivate the Left/Right, Balance, and Duration controls and instead specify the pan position of the sound according to the element's left/right position in the scene. When the element is positioned at the left edge of the scene, its sound is panned full left. When the element is positioned at the right edge of the scene, its sound is panned full right. The panning changes smoothly as the element is positioned between the two extremes.

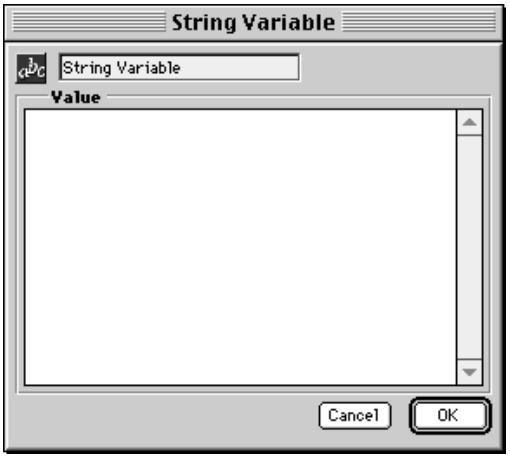
Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

String Variable

String Variable modifiers hold ASCII text values. They are useful for storing end user names, and other text values that may change during run-time.

Components of the **String Variable** dialog box are described below:



*The **String Variable** dialog box*

Variable's Name field
This editable text field can be used to change the variable's name. If the variable will be referenced in a Miniscript modifier, it is good programming practice to make the name a single word.

Modifier icon
This icon identifies the variable modifier's type.

Value Text field
Enter the variable's initial value here. The value is checked for validity when the **OK** button is clicked. The value can be changed during run-time. See "Setting Values of Variable Modifiers"

in Chapter 14 for details regarding getting and setting the value of this variable modifier.

Note: Although the String Variable can contain large amounts of text, the Value field can only accept up to 256 characters.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Text Style Modifier

The Text Style modifier can be used to change the settings of all text in a text element. Like the Graphic modifier, multiple Text Style modifiers can be added to a text element, but the effects of only one Text Style modifier can be applied at once.

The default text style of a text element can be changed using the options in the **Format** menu. The Text Style modifier, **Format** menu options, and cross-platform text issues are discussed in detail in Chapter 3, "Format Menu."

Components of the **Text Style Modifier** dialog box are described below:

*The **Text Style Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Apply When pop-up menu
Use this pop-up menu to choose the message that causes the specified text style to be applied. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Remove When pop-up menu
Use this pop-up menu to choose an optional message that causes the text effects to be removed. For a complete discussion of this menu, see "When Pop-Up Menu and Message/ Command Pop-Up Menu Options" in Chapter 13.

Specifications section
Use the controls in this section to select the text style to be applied.

Font pop-up menu
Use this pop-up menu to choose the font to be applied. All fonts installed on the current system are shown in the menu.

Note: When a project is built into a title for distribution, fonts are not included in the title file. To display properly, any fonts used by the project must be installed on the target system, or text elements must be converted to bitmaps.

Alignment pop-up menu
Use this pop-up menu to select the alignment of the text. Options are Left, Center, or Right, relative to the frame of the text element.

Size pop-up menu
Use this pop-up menu to specify the size of the text, in points.

Text Style check boxes
Select text style options, if desired, by checking any of these check boxes.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Timer Messenger

The Timer Messenger can be used to send a message after a given time has elapsed. The message can be sent once or repeatedly.

One common use for the Timer Messenger is to implement "looping" behavior in mTropolis. If the modifier's **Loop Timer** check box is checked, the messenger sends its message repeatedly at the specified time interval. The message is sent repeatedly from the time the timer is started (when it receives the message specified in its **Execute When** pop-up menu) until the timer is stopped (when it receives the message specified in its **Terminate When** pop-up menu). Other modifiers in the project can be configured to perform their actions when they receive the message sent by the Timer Messenger. Thus, the actions occur repeatedly.

Components of the **Timer Messenger** dialog box are described below:

*The **Timer Messenger** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that causes the timer to start. For a complete

discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Terminate When pop-up menu
Use this pop-up menu to choose an optional message that causes the timer to stop. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Delay For section
Enter the duration of the timer in this section's text field using the format minutes:seconds.hundredths. The timer starts when it receives the message specified by the **Execute When** pop-up menu. After the specified time has elapsed, the message specified by the "Message" section is sent.

Loop Timer check box
By default, the timer runs once and sends its message once at the end of the "Delay For" time. Check the **Loop Timer** check box to make the timer restart and send its message repeatedly until the timer is stopped by the message specified in the **Terminate When** pop-up menu.

Message specifications
Use this section to specify the message to be sent when the "Delay For" time has elapsed.

See "Configuring Messenger Modifiers" or "Message Specifications" in this chapter for a complete description of the controls in this section.

Message options
Click the triangle at the bottom of the dialog box to display the Message Options section. See "Message Options" in this chapter for a complete description of the controls in this section.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

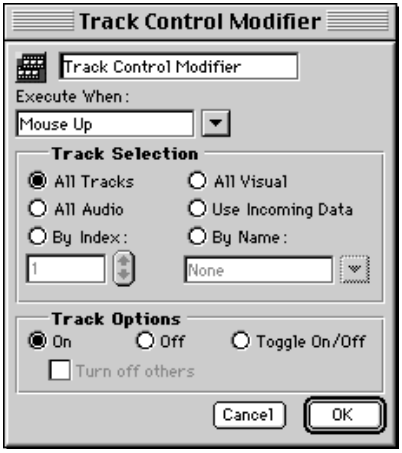
Track Control Modifier

The Track Control modifier can be used to manipulate the video and audio tracks within a QuickTime movie element. QuickTime movies consist of tracks — individual video and audio segments that can be manipulated separately, or in unison, by this modifier. Tracks can be turned on, turned off, or toggled between states using this modifier.

All QuickTime movies have at least one track. However, QuickTime movies can be made that consist of multiple tracks. mTropolis includes a separate Mac OS application called MovieTrax that can be used to create your own multitrack QuickTime movies for use in mTropolis projects. See Appendix A, "MovieTrax," for a description of this application.

Note: Because of limitations in the current version of QuickTime, this modifier is not fully supported under Windows. See "Behavior of the Track Control Modifier on Windows Platforms" in this chapter.

Components of the **Track Control Modifier** dialog box are described below.



*The **Track Control Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that triggers the Track Control modifier. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Track Selection section
Use the controls in this section to specify the track or tracks to be affected by the operation specified in the Track Options section.

All Tracks button
Click this button to specify all tracks in the movie.

All Visual button
Click this button to specify all video tracks in the movie.

All Audio button
Click this button to specify all audio tracks in the movie.

Use Incoming Data button
Click this button to specify tracks via the data accompanying the message that triggers the Track Control modifier. Acceptable data types are a string (that contains the name of the track to be selected), an integer (that contains the index of the track to be selected), or an integer range (that specifies a range of track indices to be selected).

By Index button
Click this button to specify a single track by its index number. The By Index field becomes active. Enter an index number in the field or use the arrows to choose one. Note that it is possible to choose an index that is not actually present in the movie.

By Name button
Click this button to specify a single track by its name. The **By Name** pop-up menu becomes active. The names of tracks that have names assigned to them appear as the menu options.

Track Options section
Use the controls in this section to specify the type of operation to perform on the selected track(s).

On button
Click this button to enable the selected tracks (turn them "on").

Off button
Click this button to disable the selected tracks (turn them "off").

Toggle On/Off button
Click this button to enable any selected tracks that are currently disabled and disable any selected tracks that are currently enabled.

Turn off others check box
When the **On** button is selected, this check box can also be checked to turn off all QuickTime tracks except for the ones the modifier is configured to turn on.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Behavior of the Track Control Modifier on Windows Platforms

QuickTime for Windows does not support multitrack QuickTime movies in the same way as QuickTime for Mac OS. Specifically, QuickTime for Windows at most one audio track of a QuickTime movie at any time.

When the Track Control modifier is used in titles run on Windows platforms, the following restrictions apply:

- Only one video track is recognized. The video track with the lowest QuickTime layer number is the only one recognized. The MovieTrax utility can be used to change the order of QuickTime tracks. See "List Window" in Appendix A.
- Track index numbers may change as a result of multiple video tracks being ignored. For example, consider a movie with two video tracks and two audio tracks. Suppose these tracks were layered with MovieTrax such that video track 1 is the front-most track, followed by video track 2, audio track 1, and audio track 2.

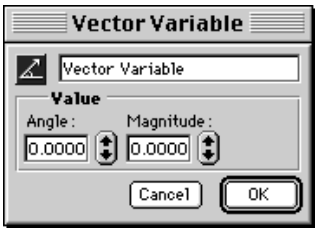
  On Mac OS, these tracks could be accessed by index number as indexes 1 through 4 (video track 1 is at index 1, video track 2 is at index 2, audio track 1 is at index 3, and audio track 2 is at index 4). However, when built for Windows, video track 2 will be ignored. Therefore, the index numbering of the track changes such that video track 1 is at index 1, audio track 1 is at index 2, and audio track 2 is at index 3.

  As a result, the Track Control modifier will behave differently under Windows. In our example, track index 4 could be turned on or off in a Mac OS title, but that track index would be unavailable on the Windows title.
- Similarly, track names may become confused if there are multiple video tracks in the movie. mTropolis simply maps track names to index numbers.
- When used in titles built for Windows platforms, this modifier is best used with QuickTime movies that have only one video track and multiple audio tracks. For such movies, the Track Control modifier will work identically on both Mac OS and Windows.

Vector Variable

The Vector Variable modifier stores angle and magnitude values for use with the Vector Motion modifier (see "Vector Motion Modifier" in this chapter).

Components of the **Vector Variable** dialog box are described below:



*The **Vector Variable** dialog box*

Variable's Name field
This editable text field can be used to change the variable's name. If the variable will be referenced in a Miniscript modifier, it is good programming practice to make the name a single word.

Modifier icon
This icon identifies the variable modifier's type.

Value fields
Enter the variable's initial values here or use the up/down arrow buttons to set the values by 0.5 increments. There are two fields:

- **Angle**: This is the vector's angle, measured in degrees counter-clockwise from horizontal (for example, 0 degrees is at "3 o'clock," 90 degrees is at "12 o'clock").
- **Magnitude**: This is the vector's magnitude. The Vector Motion modifier interprets this magnitude as velocity in inches per second.

The value is checked for validity when the **OK** button is clicked. The value can be changed during run-time. See "Setting Values of Variable Modifiers" in Chapter 14 for details regarding getting and setting the value of this variable modifier.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

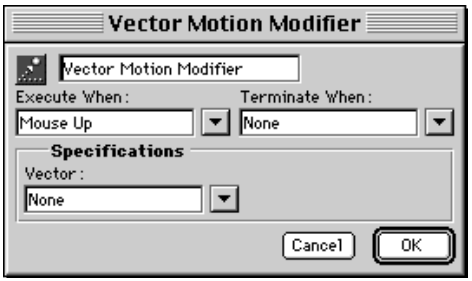
OK button
Click **OK** to accept changes made to this modifier.

Vector Motion Modifier

The Vector Motion modifier applies a motion effect to an element using the angle and magnitude values specified in a Vector Variable (see "Vector Variable" above).

The vector motion of the element is tied to the specified Vector Variable. If the values in the Vector Variable are changed, the motion of the element changes to match the new values. The values of the Vector Variable modifier may be changed during run-time, providing many possibilities for an object's vector motion. For example, a graphic element could be configured to move in a straight line toward the end user's cursor and then change direction as the mouse's location changes.

Components of the **Vector Motion Modifier** dialog box are described below:



*The **Vector Motion Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Execute When pop-up menu
Use this pop-up menu to choose the message that initiates the vector motion. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Terminate When pop-up menu
Use this pop-up menu to choose the message that terminates the vector motion. For a complete discussion of this menu, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in Chapter 13.

Vector pop-up menu
Use this pop-up menu to choose a previously defined Vector Variable as the model for the vector motion. Any variables currently available to the Vector Motion modifier from its position in the project's hierarchy will be visible in this pop-up menu (see "Variable Scopes" in Chapter 13).

Note: A Vector Variable must be selected in this pop-up menu for any vector motion to happen.

Cancel button
Click **Cancel** to ignore changes made to this modifier.

OK button
Click **OK** to accept changes made to this modifier.

Window Prefs Modifier

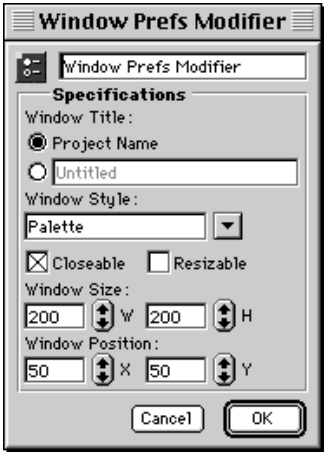
The Window Prefs modifier can be used to create projects that run in a window. A number of standard window styles can be specified. The window title, default size, and default position can also be specified.

The Window Prefs modifier must be placed at the project level and must not be enclosed in a behavior. Use the structure view to place this modifier directly on the project component.

When run in the mTropolis editor or a Mac OS version of the mTropolis player, a project that contains this modifier runs inside of a window instead of taking over the entire screen.

Note: This modifier is Mac OS only. It has no effect in titles run in Windows versions of the mTropolis player. In its current implementation, this modifier is intended for use in the creation of mTropolis tools (titles that assist in the creation of other mTropolis projects). While it can be used to create Mac OS titles that run in a window (instead of taking over the entire screen), it is not intended as a general solution for creating cross-platform, windowed mTropolis titles.

Components of the **Window Prefs Modifier** dialog box are described below:



*The **Window Prefs Modifier** dialog box*

Modifier's Name field
This editable text field can be used to change the modifier's name.

Modifier icon
This icon identifies the modifier's type.

Specifications section
Use the controls in this section to control the type of window created by the Window Prefs modifier.

Window Title buttons
The title of the window (displayed in the window's title bar) can be set in one of two ways:

- **Project Name Button**: Click this button (the default) to display the name of the project (the file name under which the project is saved) in the window's title bar.
- **Text Field and Button**: To give a window a title other than the project name, click the button next to the unavailable text field. Type the desired window title into the text field.

Window Style pop-up menu
Use this pop-up menu to choose the type of window in which the project will run. The following styles are available:

- **Palette**: Choose this option to create a movable "palette" window. This type of window has a narrow title bar along its top.
- **Sidebar Palette**: Choose this option to create a movable "palette" window with a narrow drag bar along its side. This type of window does not display a title.
- **Document**: Choose this option to create a standard document window with a wide title bar along its top.
- **Modal**: Choose this option to create a modal window without a title bar. This type of window cannot be moved and does not display a title. Modal windows are "blocking" windows. When a modal window is displayed, only that window can be selected and worked with until it is dismissed.
- **Movable Modal**: Choose this option to create a modal window that can be moved around the screen. Modal windows are "blocking" windows. When a modal window is displayed, only that window can be selected and worked with until it is dismissed.
- **Borderless**: Choose this option to create a non-movable window without a border. This type of window cannot be moved and does not display a title. This window is not modal and allows other windows to be selected and come to the front of the display.

- **Floating Borderless**: Choose this option to create a non-movable window without a border that always "floats" at the front of the screen. This type of window does not display a title. This window is not modal, but always appears in front of other windows, even when another window is active.

Closeable check box
Check this check box (the default) to create a window that can be closed by clicking its **Close** button. Note that not all window styles can have **Close** buttons. When a window style that does not support **Close** buttons is chosen, this check box is unavailable.

Resizable Check box
Check this check box (unchecked by default) to create a window that can be resized. Note that not all window styles can be resized. When a window style that does not support resizing is selected, this check box is unavailable. Note that selecting this option does not add resize controls to a window; resizing must be performed using the windowSize attribute (described in Chapter 15)

Window Size fields
Use these fields to set the window's initial size:

- **W Field**: Enter the window's default width, in pixels, in this field. The default is 200.
- **H Field**: Enter the window's default height, in pixels, in this field. The default is 320.

Window Position fields
Use these fields to set the window's initial position:

- **X Field**: Enter the window's default horizontal position in this field. This position is measured in pixels from the left edge of the screen. The default is 50.
- **Y Field**: Enter the window's default vertical position in this field. This position is measured in pixels from the top edge of the screen. The default is 50.

Related project attributes

The project component has attributes that control many of the same features as the Window Prefs modifier. The windowMaxSize, windowMinSize, windowPosition, windowSize, windowStyle, and windowTitle project attributes are described in Chapter 15, "Attributes."

13. Modifier Pop-Up Menus and Message Reference

13. Modifier Pop-Up Menus and Message Reference

This chapter describes menus that are common to many of the modifier dialog boxes found in MTropolis. These menus include:

• When Pop-Up Menu, Message/Command Pop-Up Menu, With Pop-Up Menu, and Destination Pop-Up Menu.

The following topics are also covered in this chapter:

• “mTropolis Messages and Commands,” “Message Paths,” and “Variable Scopes.”

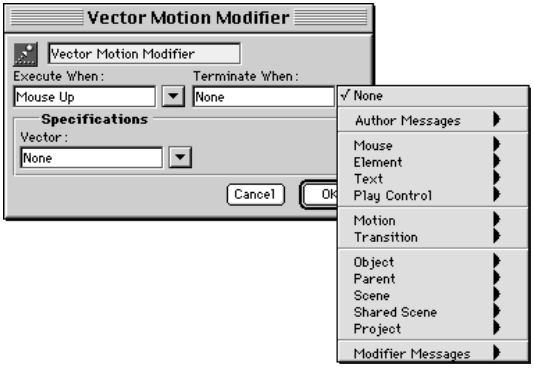
The "When" Pop-Up Menu

Most **Modifier** dialog boxes contain pop-up menus used to specify the messages that activate (or sometimes, deactivate) the modifier when they are received. These menus are referred to as **When** pop-up menus.

Each **When** pop-up menu in a modifier's dialog box is labeled with a phrase that describes the action of the modifier when the selected message is received. This phrase helps to clarify the use of a **When** pop-up menu for each particular modifier. For example, "Execute When," "Terminate When," and "Apply When" are common When menu labels.

Variable modifiers do not have **When** pop-up menus as the value contained within the variable automatically becomes a part of the component on which it is placed.

All When menus contain the same options. Types of messages are described in "mTropolis Messages and Commands" in this chapter. For information about individual items in the **When** pop-up menu, see "When Pop-Up and Message/Command Pop-Up Options" also in this chapter.



*Typical mTropolis **Modifier** dialog box with **When** menus*

The Message/Command Pop-Up Menu

Messenger modifier dialog boxes have a Message Specifications section, used to select the message to be sent when the messenger is activated, any extra data sent with the message, and the destination of the message.

The first menu in the Message Specifications section is the **Message/Command** pop-up menu. This menu lists all of the possible messages and commands that can be sent to elements in the project. The types of messages and commands that can be sent are described in the next section. For information about individual items in the **When** pop-up menu, see "When Pop-Up and Message/Command Pop-Up Options" below.

*Typical **Messenger** dialog box with **Message/Command** menu*

mTropolis Messages and Commands

There are three types of options available in the Message/Command menu: environment messages, author messages, and commands. The **When** pop-up menu displays both author

messages and environment messages that can be used to activate or deactivate modifiers. The three types of messages and commands are described below.

Environment messages

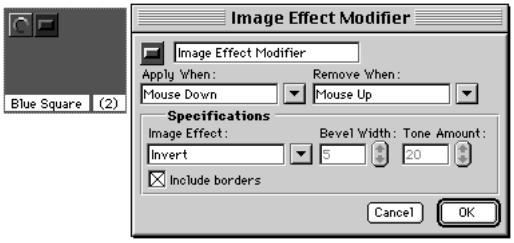
In run-time mode, changes in the mTropolis environment, such as end user mouse clicks, transitions, or scene changes, are detected by mTropolis' engine and reported to components of a project as messages. These messages are called environment messages. Environment messages are said to be generated "by mTropolis" or "by the mTropolis environment."

Environment messages can be used to activate or deactivate modifiers by selecting them from the **When** pop-up menu. In this way, run-time events can be made to trigger modifiers. For example, the Image Effect modifier on the blue element in the following figure is configured to be applied by the environment message Mouse Down and removed by the environment message Mouse Up.

Sending environment messages from messengers

In addition to being generated by run-time events, environment messages can also be sent from author-configured messengers. Choose an environment message from the **Message/Command** pop-up menu to send an environment message to a specific element, just as if the run-time event associated with that message had occurred.

For example, consider the "Blue Square" graphic element in the figure below, which has an Image Effect modifier assigned to it. This modifier will be applied when the element receives a Mouse Down message. Therefore, the effect will be applied in run-time mode when an end user starts to click on the element with the mouse. However, the image effect could also be triggered (without the end user pressing the mouse button) by configuring a messenger to send a Mouse Down message to the Blue Square element.



Activating and deactivating a modifier with environment messages

This ability to send any environment message from a messenger gives the author flexible control over objects in a mTropolis project.

For information about individual environment messages, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options"below.

Author messages

A message that has been created by the author of a mTropolis project is called an author message. Author messages are never sent by the mTropolis environment; they are only sent by messengers configured by the author. As with any other message, author messages can be used to activate or deactivate modifiers.

Previously-created author messages can be sent by selecting the message from the Author Messages submenu of the **Message/Command** pop-up menu. New author messages can be created simply by highlighting the text in the When or Message/Command text fields and entering new text. A dialog box appears asking if you want to create the new author message. Once created, the new author message appears in the

When menus of all modifiers and the Message/Command menu of all messengers.

Because modifiers respond to strings of words but not their meaning, the text of an author message is irrelevant. As long as the message that is sent to a modifier is the same as the message it is configured to receive, the modifier will be activated. For example, a Change Scene modifier could be configured to respond to the author message "Jump up!" When this message is received, the Change Scene modifier activates and changes to the next scene in the project.

Commands

In addition to environment and author messages, the Message/Command menu allows a choice of commands. Commands are imperative instructions that are acted on by elements immediately upon receipt.

Commands are different from messages in the following ways:

- A command constitutes a demand that the recipient perform some action, and it cannot be ignored. Commands are primarily used to control elements directly. For example, sending the **Play** command to an element causes any time-based media (such as a QuickTime movie) linked to the element to start playing.
- Unlike messages, commands are not used to activate modifiers and are therefore not available in the **When** pop-up menu.
- Like author messages, commands are never sent from the mTropolis environment. They are always generated by author-configured messengers.
- Unlike either type of message, commands are sent to and acted on by a single, "targeted" destination. Commands do not "cascade" or "relay" through the scene.
- Although commands are targeted only at a single destination, some commands have associated messages that are generated by mTropolis when an element has responded to a command. These messages report the new state of an element. For example, when an element is sent the **Play** command, mTropolis generates a Played message that is sent to the element and its modifiers. These "message/command pairs" are described in "When Pop-Up Menu and Message/Command Pop-Up Menu Options" below.
- Command names are shown in *italic¡* type in the **Message/Command** menu to differentiate them from messages.

For information about individual commands, see "When Pop-Up Menu and Message/Command Pop-Up Menu Options" in this chapter.

When Pop-Up Menu and Message/Command Pop-Up Menu Options

The **When** and **Message/Command** pop-up menus are organized into a number of submenus that open to reveal the names of specific message and commands that can be sent. Each of the following sections details a different submenu: "Author Messages;" "Mouse Messages;" "Element Messages and Commands;" "Text Messages and Commands;" "Play Control Messages and Commands;" "Motion and Transition Messages;" "Object Messages and Commands;" "Parent Messages;" "Scene Messages;" "Shared Scene Messages;" "Project Messages and Commands;" "Modifier Messages;" and "Get and Set Attribute Commands."

Author Messages

Previously-defined author messages can be chosen from the Author Messages submenu.

To create a new author message, highlight the text in the Message/Command text field and enter the name of the new message. A prompt appears, asking if you want to create the new author message. Alternatively, choose the **Author Messages ➛ New Author Message** option from the **Message/Command** pop-up menu. The **New Author Message** dialog box appears. Enter the name of the new author message and click **OK.**

Author messages can also be created in the Author Messages Window (see "Author Messages Window" in Chapter 7).

How mTropolis handles author messages

By default, author messages sent from messengers "cascade" from the targeted destination to every component below it in the project's hierarchy. This feature allows many modifiers to be activated by a single message.

For example, a messenger could be configured to send an author message called "SetSpeed" with a value of 10 to a scene. Messengers on mToons in the scene can be configured to respond to this author message by setting the frame rate of their animations with the incoming data. When the author message "Set Speed" and the value 10 is received by the scene, it is automatically broadcast from the scene to its elements and modifiers, causing the messengers on the mToons to collectively set the frame rates of their animations to 10.

Messages can be more precisely targeted by altering the default message options in the **Messenger** dialog box. See "Message Options" in this chapter and Chapter 12.

Using author messages in the When pop-up menu

Choose an author message from this menu to specify the message that activates or deactivates the modifier.

Using author messages in the Message/Command pop-up menu

Choose an author message to be sent to a targeted destination from this pop-up menu.

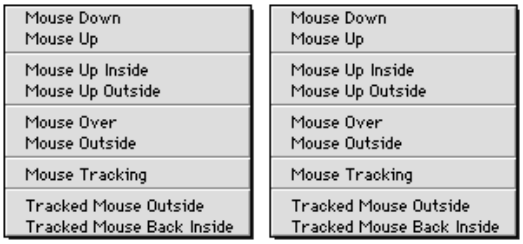
New Author Message

Choose this option from the **When** or **Message/Command** pop-up menu to display the **New Author Message** dialog box. Enter the name of the desired author message and click **OK.** The new author message can now be chosen in either pop-up menu. See "Author Messages Window" in Chapter 7 for more information on creating and managing author messages.

Note: Author messages must have unique names, even if they reside in different author message groups. They also must not duplicate the names of built-in messages (such as Mouse Up).

Mouse Messages

Choose Mouse in the **When** pop-up menu or the **Message/Command** pop-up menu to reveal the submenu options related to this item.



*Mouse messages in the **When** and **Message/Command** pop-up menus*

How mTropolis generates mouse messages

During run-time, mTropolis sends mouse messages in response to an end user's mouse actions. When the mouse action is detected, the appropriate messages are sent to the element under the mouse cursor and passed to its modifiers.

If elements that are layered over each other are each configured with modifiers to respond to the same mouse messages, only the element closest to the viewer (that is, the element with the highest layer order number) under the end user's mouse receives mouse messages from mTropolis.

Hidden elements neither receive nor do they act on mouse messages sent by mTropolis. However, modifiers on a hidden element that act on mouse messages can be triggered by sending mouse messages to the hidden element from a messenger modifier.

Note: For performance reasons, mouse messages such as Mouse Over and Mouse Outside are only sent when the end user moves the mouse cursor over an element. Messages are not generated when moving elements pass under a stationary cursor. See the description of "refreshCursor" in Chapter 15 for information on detecting such events.

Using mouse messages with the When pop-up menu

Choose a mouse message in a **When** menu to activate or deactivate the modifier when the message is received, either from mTropolis or from a messenger.

By default, the cursor changes to "hand up" when moved over an element that is sensitive to mouse messages. This behavior can be changed using the Cursor modifier (see Chapter 12).

Using mouse messages with the Message/Command pop-up menu

Choose a mouse message in a **Message/Command** menu to send the mouse message to a targeted destination as if the message had been generated by the end user's mouse actions. Note that sending one of these messages does not actually cause the mouse cursor (or the mouse itself!) to move.

Data sent with mouse messages

When sent by the mTropolis environment, all mouse messages are sent with point data indicating the screen position at which the event occurred. Mouse messages sent by the author can also be sent with point data to provide a simulated position for the author-generated mouse event.

Mouse message descriptions

Mouse message options found in the **When** and **Message/Command** pop-up menus are described below:

Mouse Down

mTropolis sends a Mouse Down message to an element when the mouse button is pressed down and the mouse cursor is over that element. mTropolis sends point data

with this message, indicating the position at which the event occurred.

Mouse Up
mTropolis sends a Mouse Up message to an element when the mouse button has been pressed down over that element and then released anywhere. mTropolis sends point data with this message, indicating the position at which the event occurred.

Mouse Up Inside
mTropolis sends a Mouse Up Inside message to an element when the mouse button has been pressed down over that element and then released inside the frame of that element. mTropolis sends point data with this message, indicating the position at which the event occurred.

Mouse Up Outside
mTropolis sends a Mouse Up Outside message to an element when the mouse button has been pressed down over that element and then released outside the frame of that element. mTropolis sends point data with this message, indicating the position at which the event occurred.

Mouse Over
mTropolis sends the Mouse Over message to an element when the mouse cursor passes over the frame of that element in an inward direction. mTropolis sends point data with this message, indicating the position at which the event occurred.

Mouse Outside
mTropolis sends the Mouse Outside message to an element when the mouse cursor leaves the frame of an element. mTropolis sends point data with this message, indicating the position at which the event occurred.

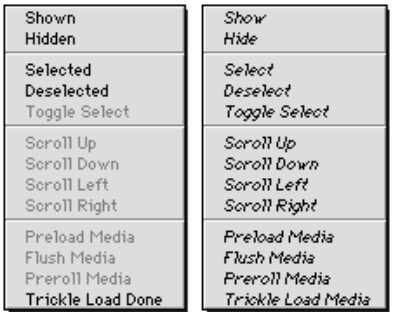
Mouse Tracking
mTropolis sends the Mouse Tracking message to an element repeatedly after the mouse button has been pressed over that element and is being held down and moved (that is, the mouse cursor is being dragged). The message is sent until the mouse button is released. mTropolis sends point data with this message, indicating the position at which the event occurred.

Tracked Mouse Outside
mTropolis sends the Tracked Mouse Outside message to an element when Mouse Tracking has started (after the end user has started a drag motion over the element), and the mouse cursor is dragged outside the frame of that element. mTropolis sends point data with this message, indicating the position at which the event occurred.

Tracked Mouse Back Inside
mTropolis sends the Tracked Mouse Back Inside message to an element after the Tracked Mouse Outside condition has been met (after the end user has started a drag motion over the element, but continued dragging outside the element's frame) and the mouse cursor is dragged back inside that element's frame. mTropolis sends point data with this message, indicating the position at which the event occurred.

Element Messages and Commands

Choose Element in the **When** or **Message/Command** pop-up menu to reveal the submenu items described below.



*Element messages in the **When** and **Message/Command** pop-up menus*

How mTropolis handles element messages and commands

All of the Element submenu items in the **Message/Command** pop-up menu are commands. Commands are not sent by mTropolis, they can only be sent from messengers.

The element environment messages listed in the **When** pop-up menu are only sent by mTropolis when it detects an element that has responded to a command. mTropolis notifies the element of its changed state by sending the environment message to the element and its modifiers.

Using element messages with the When pop-up menu

Choose an element message in a **When** pop-up menu to activate or deactivate the modifier when the message is received, either from mTropolis or from a messenger.

Using element commands with the Message/Command pop-up menu

Every item on the **Message/Command** pop-up menu's Element submenu is a command. Select an element message or a command to send to a targeted destination from this menu.

Element Message/Command descriptions

The following element-related messages and commands are available in the **When** and **Message/Command** windows. Note that some options are available in only one menu. Other options are described as related message/command pairs. The message, found in the **When** menu (for example, "Shown"), is described first, followed by the command (**Show**), found in the **Message/Command** pop-up menu, that (when received by an element) generates the message.

Shown/Show

mTropolis sends a Shown message to an element when it becomes shown (in response to a **Show** command). Send the **Show** command to a hidden element to make it visible.

Hidden/Hide

mTropolis sends a Hidden message to an element when it becomes hidden (in response to a **Hide** command). Send the **Hide** command to a visible element to make it invisible (and unresponsive to end user mouse events). To create invisible elements that do respond to end user mouse events, consider applying the "Invisible" ink effect described in Chapter 11.

Selected/Select
mTropolis sends a Selected message to an element when it receives a **Select** command. Send the **Select** command to an element to set some author-defined state to "on." **Select** commands can only be sent by messengers — the mTropolis environment never sends **Select.**

!!! Note: The **Select** command has nothing to do with end user mouse actions. See "Mouse Messages" in this chapter for messages sent in response to end user mouse motions and clicks.

Deselected/Deselect
mTropolis sends a Deselected message to an element when it receives a **Deselect** command. Send the **Deselect** command to an element to set some author-defined state to "off." **Deselect** commands can only be sent by messengers — the mTropolis environment never sends **Deselect.**

The **Deselect** command has nothing to do with end user mouse actions. See "Mouse Messages" in this chapter for messages sent in response to end user mouse motions and clicks.

Toggle Select (Message/Command Menu Only)
Send the **Toggle Select** command to an element that has previously received the **Select** or **Deselect** commands to reverse the element's current selection state. When a "selected" element receives **Toggle Select**, mTropolis sends a Deselected message to the element. When a "deselected" element receives **Toggle Select**, mTropolis sends a Selected message to the element.

Scroll Up (Message/Command Menu Only)
Send the **Scroll Up** command to move the content of an element down within its frame. That is, this command emulates the behavior of the up-pointing arrows commonly found on scrolling interface elements. Specify the amount in pixels that the content is to move by sending a value using the "With" field. As the content of the element moves, it will be cropped by the frame of the element.

Scroll Down (Message/Command Menu Only)
Send the **Scroll Down** command to move the content of an element up within its frame. That is, this command emulates the behavior of the down-pointing arrows commonly found on scrolling interface elements. Specify the amount in pixels that the content is to move by sending a value using the "With" field. As the content of the element moves, it will be cropped by the frame of the element.

Scroll Left (Message/Command Menu Only)
Send the **Scroll Left** command to move the content of an element right within its frame. That is, this command emulates the behavior of the left-pointing arrows commonly found on scrolling interface elements. Specify the amount in pixels that the content is to move by sending a value using the "With" field. As the content of the element moves, it will be cropped by the frame of the element.

Scroll Right (Message/Command Menu Only)
Send the **Scroll Right** command to move the content of an element left within its frame. That is, this command emulates the behavior of the right-pointing arrows commonly found on scrolling interface elements. Specify the amount in pixels that the content is to move by sending a value using the "With" field. As the content of the element moves, it will be cropped by the frame of the element.

Preload Media (Message/Command Menu Only)
Send the **Preload Media** command to an element to load that element's associated media

file into system memory (RAM) for optimal playback. This command can only be sent to elements that contain mToons, video, and audio media (the equivalent for still graphics is the "cache bitmap" setting — see "Cache Bitmap Check Box" in Chapter 5).

Preloading begins immediately upon receipt of this message. Note that other mTropolis processes are stopped while loading occurs. See the description of **Trickle Load Media** later in this chapter for a preloading technique that uses fewer system resources, at the expense of longer preloading time. When the element's media is completely loaded, its "preloaded" attribute (see Chapter 15) is set to true.

For mToons and QuickTime movies, **Preload Media** can be used to preload a range instead of the entire media asset. To preload a range:

- for mToons: send the **Preload Media** command with an accompanying mToon range. This range can be selected from the "mToon Ranges" portion of the **With** pop-up menu, or it can be entered as a literal integer range that represents the range of mToon cels to be loaded (`1 thru 10`). Alternatively, a single cel can be preloaded by sending an integer (`5`) with the **Preload Media** command.

  When preloading is completed, the mToon's preloaded attribute is set to true.

- for QuickTime: send the **Preload Media** command with an accompanying QuickTime range. This range can be selected from the "QuickTime Ranges" portion of the **With** pop-up menu, or it can be entered as a literal integer range that represents the range of QuickTime timevalues to be loaded (for example, `0 thru 1000`). Sending a single integer (for example, `250`) with the **Preload Media** command causes mTropolis to preload the movie from that timevalue all the way to the end of the movie.

  When preloading is completed, mTropolis sets the movie's preloaded attribute to true.

The **Flush Media** command can be used to remove preloaded media from memory. mTropolis will also automatically unload preloaded media when the scene changes or when a low memory situation occurs. However, the priority of preloaded media can be changed as described in "flushPriority" in Chapter 15.

Note: This option is limited by the memory available to the project at run-time. Use caution when sending this command as exceeding the available memory on machines that play your title can cause unpredictable behavior. Pay attention to the size of media you are attempting to preload. For example, on most machines, it is not a good idea to attempt to preload a 40 megabyte QuickTime video.

Flush Media (Message/Command Menu Only)
Send the **Flush Media** command to an element to flush its associated media from memory. This command is useful for removing media that was previously loaded into memory using the **Preload Media** command.

Preroll Media (Message/Command Menu Only)
Send the **Preroll Media** command to an mToon, video, or sound element to perform caching and initialization of the media so that it can respond more quickly to the next **Play** command. This command is especially useful with sound elements which usually must take time to load from disk into a memory buffer before they actually begin playing.

Note that once the media has been played, it is no longer "prerolled."

Prerolling may also be useful when a sound or the sound in a QuickTime element "stutters" before beginning to play smoothly.

Trickle Load Done/Trickle Load Media
mTropolis sends a Trickle Load Done message to an element when the media associated with that element has been completely loaded into memory using the trickle loading process. Send the **Trickle Load Media** command to an element to start the trickle loading process for that element's media.

Trickle loading is a special way of preloading media during idle time (times when the system is not otherwise busy processing mTropolis events). Standard preloading blocks all other processes while loading takes place. For example, mToons in a scene may stop playing during a long preload. Because it occurs in the "background," trickle loading does not have this problem — though the trickle loading process may still produce a noticeable slowdown.

For mToons and QuickTime movies, trickle loading can be used to trickle load a range instead of the entire media asset. To trickle load a range:

- for mToons: send the **Trickle Load Media** command with an accompanying mToon range. This range can be selected from the "mToon Ranges" portion of the **With** pop-up menu, or it can be entered as a literal integer range that represents the range of mToon cels to be loaded (for example, `1 thru 10`). Alternatively, a single cel can be trickle loaded by sending an integer (for example, `5`) with the **Trickle Load Media** command.

  Once trickle loading is completed, mTropolis sends the Trickle Load Done message with accompanying data that contains the range of cels (as an integer range) or cel (as an integer) that was trickle loaded.

- for QuickTime: send the **Trickle Load Media** command with an accompanying QuickTime range. This range can be selected from the "QuickTime Ranges" portion of the **With** pop-up menu, or it can be entered as a literal integer range that represents the range of QuickTime timevalues to be loaded (for example, `0 thru 1000`). Sending a single integer (for example, `250`) with the **Trickle Load Media** command causes mTropolis to trickle load the movie from that timevalue all the way to the end of the movie.

  When trickle loading is completed, mTropolis sends the Trickle Load Done message with accompanying data that contains the range (as an integer range) that was trickle loaded.

The trickleReadSize Asset Manager attribute (see Chapter 15), can be used to tune the trickle loading process. The trickleEnabled Asset Manager attribute (also in Chapter 15) can be used to enable and disable trickle loading for the entire project.

The **Trickle Load Media** command is otherwise similar to the **Preload Media** command described above.

Text Messages and Commands

All of the Text submenu items in the **Message/Command** menu are commands. Commands are not sent by mTropolis; they can only be sent from messengers. These commands are used to control the state of text elements.

*Text messages in the **When** and **Message/Command** pop-up menus*

mTropolis sends the Edit Done environment message to an element and its modifiers when the end user has clicked off of an editable text element during run-time.

Using text messages with the When pop-up menu

Choose an element message in a When menu to activate or deactivate the modifier when the message is received, either from mTropolis or from a messenger.

Using text messages and commands with the Message/Command pop-up menu

Every item on the **Message/Command** pop-up menu's Text submenu is a command. Choose a text message or a command to send to a targeted destination from this menu.

Text Message/Command descriptions

The following text-related options are available in the When and Message/Command windows. Note that some options are available in only one menu.

Enable Editing (Message/Command menu only)
The **Enable Editing** command can be used to make a text element editable (that is, the contents of the element can be changed by the end user in run-time mode). Send **Enable Editing** to a text element to make it editable. Upon receipt of this command, the text element becomes editable, and an insertion point appears in the text. The text element remains editable until the end user clicks outside of the text element to indicate that text entry is complete, or until the text element receives a **Disable Editing** command. To make a text element editable without immediately putting an insertion point in the text, use the "editable" attribute (described in Chapter 15).

!!! Note: When an element is being edited, any active Graphic modifiers are ignored and the text is displayed with a black foreground, white background, and "Copy" ink effect. The element is restored to its original appearance when editing is finished.

Disable Editing (Message/Command menu only)
The **Disable Editing** command can be used to return editable text elements to their non-editable state. Send **Disable Editing** to a text element that is currently being edited to return it to its default state (there is no insertion point in the text). When an element receives the **Disable Editing** command, mTropolis also sends an Edit Done message to the element.

Edit Done (When pop-up menu only)
mTropolis sends the Edit Done message to an editable text element when the end user finishes text entry by clicking outside the text element. This message is also sent to an element that receives the **Disable Editing** command.

Update Calculated Fields (Message/Command Menu Only)
Send the **Update Calculated Fields** command to a text element to update any calculated fields displayed in the element. Any placeholders (variable names enclosed by the < and > symbols) are updated to show the current value of the specified variable. See "Calculated Fields — Displaying Variables in Text Fields" in Chapter 3.

Play Control Messages and Commands

The following section describes the items available in the Play Control submenu of the **When** pop-up menu and **Message/Command** pop-up menu. These items are related to "playing" the contents of an element.

*Play control messages in the **When** and **Message/Command** pop-up menus*

How mTropolis handles play control messages and commands

Most of the items in the Play Control submenu are message/command pairs. With the exception of At First Cel and At Last Cel, the Play Control environment messages listed on the **When** pop-up menu are only sent by mTropolis when it detects an element that has responded to a corresponding Play Control command. Commands are not sent by mTropolis, they can only be sent from messengers.

mTropolis sends At First Cel or At Last Cel environment messages to an mToon or QuickTime element and its modifiers when the animation or movie reaches its first or its last cel (or last timevalue for QuickTime).

Using play control messages with the When pop-up menu

Choose a Play Control message in a **When** menu to activate or deactivate the modifier when the message is received, either from mTropolis or from a messenger.

Using play control messages and commands with the Message/Command pop-up menu

Select a Play Control message or command in a **Message/Command** menu to send the message or command to a targeted destination.

Play control Message/Command descriptions

Play control options found in the **When** and **Message/Command** pop-up menus are described below. Note that many of these options are described as related message/command pairs. The message, found in the **When** menu (Played), is described first, followed by the command (**Play**), found in the **Message/Command** menu, that (when received by an element) causes the message to be generated.

Played/Play

mTropolis sends a Played message to an element when it starts playing (in response to a **Play** command). Send the **Play** command to an element to show the element (if it is hidden) and start playing that element's associated media.

By default, the **Play** command starts the element's media playing from its beginning. However, this behavior can be changed by sending **Play** with a data value using the **With** pop-up menu. The type of value that can be sent depends upon the media type being played, as described below:

- **mToons:** When sending **Play** to an element that contains an mToon with predefined ranges, the **With** pop-up menu can be used to select an mToon range by name. Range names can be found under the "mToon Ranges" cascading menu item. For information about creating mToon ranges, see "Ranges" in Chapter 1. When the element

receives the **Play** command, only the cels in the specified range are played.

Alternatively, the **Play** command can be sent from Miniscript with the range name as accompanying (with) data. Note that, when used this way, the range name should not be sent as a string (that is, it does not have to be enclosed in quotation marks) but is instead simply referenced. For example, the following Miniscript statement sends the **Play** command to the mToon named "myToon" so that only the range named "myRange" is played:

```
send "Play" with myRange to myToon
```

!!! Note: If an mToon is linked to the project and then removed, any ranges that mToon contained continue to show up in the "mToon Ranges" cascading menu.

- **QuickTime**: When sending **Play** to an element that contains a QuickTime movie with predefined ranges, the **With** pop-up menu can be used to select a QuickTime range by name. Range names can be found under the QuickTime Ranges cascading menu item. QuickTime ranges can be created using the MovieTrax utility included on the mTropolis CD–ROM. For information about creating QuickTime ranges, see New Range in Appendix A. When the element receives the **Play** command, only the specified range is played.

  Alternatively, the **Play** command can be sent from Miniscript with the range name as accompanying (with) data. Note that, when used this way, the range name should not be sent as a string (it does not have to be enclosed in quotation marks) but is instead simply referenced. For example, the following Miniscript statement sends the **Play** command to the movie named "myMovie" so that only the range named "myRange" is played:

  ```
  send "Play" with myRange to myMovie
  ```

  !!! Note: If a QuickTime movie is linked to the project and then removed, any ranges that movie contained continue to show up in the "QuickTime Ranges" cascading menu.

- **Sound**: When sending **Play** to an element that contains an AIFF sound file with predefined markers, the **With** pop-up menu can be used to select a marker by name. Marker names can be found under the "Sound Markers" cascading menu item. When the element receives the **Play** command, the sound starts playing from the selected marker point to the next marker or the end of the sound file, whichever comes first. Sound markers (sometimes also called cue points) can be added to AIFF files using an external sound editing application.

  Alternatively, the **Play** command can be sent from Miniscript with the sound marker name as accompanying ("with") data. Note that, when used this way, the range name should not be sent as a string (it does not have to be enclosed in quotation marks) but is instead simply referenced. For example, the following Miniscript statement sends the **Play** command to the sound element named "mySound" so that it starts playing from the "myMarker" sound marker:

  ```
  send "Play" with myMarker to mySound
  ```

  If multiple sound files and sound markers are being used in a project, ensure that all of the markers are given unique names. When mTropolis loads a sound file that contains a marker with the same name as a marker

from a sound file already linked to the project, only the previously-linked marker shows up in the "Sound Markers" cascading menu.

!!! Note: also that if a sound is linked to the project and then removed, any sound markers that sound contained continue to show up in the "Sound Markers" cascading menu.

Though not every element has media associated with it (for example, text elements), **Play** can be sent to any element type, causing the element to be shown (if it is hidden) and receive a Played message.

Stopped/Stop
mTropolis sends a Stopped message to an element when it stops playing (in response to a **Stop** command). Stopped is also sent to sound elements when the sound reaches its end (and the sound is not set to loop). Send the **Stop** command to an element to stop playing that element's associated media and hide the element. To "stop" an element from playing without hiding it, use the **Pause** command.

!!! Note: Though not every element has media associated with it (text elements), **Stop** can be sent to any element type, causing the element to be hidden and receive a Stopped message.

Paused/Pause
mTropolis sends a Paused message to an element when its media pauses playing (in response to a **Pause** command). Paused is also sent to mToon and QuickTime elements to stop playing when they reach their end (and they are not set to loop). Send the **Pause** command to an element to temporarily stop playing that element's media, without hiding the element.

Unpaused/Unpause
mTropolis sends an Unpaused message to an element when it unpauses playing (in response to an **Unpause** command). Send the **Unpause** command to an element to start playing that element's paused media once again. The media begins playing from its paused position.

Toggle Pause (Message/Command Menu Only)
Send the **Toggle Pause** command to an element that has previously received the **Pause** command or is currently playing to reverse the element's current pause state. That is, when a paused element receives **Toggle Pause**, mTropolis starts playing its media and sends an Unpaused message to the element. When a currently-playing element receives **Toggle Pause**, mTropolis pauses the media and sends a Paused message to the element.

At First Cel
mTropolis sends an At First Cel message to an mToon element when the animation reaches its first cel. mTropolis sends this message to a QuickTime element when the movie reaches its starting timevalue.

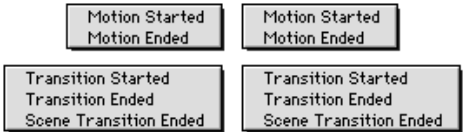
This message can also be sent to a targeted element to trigger any modifiers that act on At First Cel just as if the first cel in the animation had been reached. Note, however, that the animation does not actually change to the first cel when this message is received from a messenger.

At Last Cel
mTropolis sends an At Last Cel message to an mToon element when the animation reaches its last cel. mTropolis sends this message to a QuickTime element when the movie reaches its ending timevalue.

This message can also be sent to a targeted element to trigger any modifiers that act on At Last Cel just as if the last cel in the animation had been reached. Note, however, that the animation does not actually change to the last cel when this message is received from a messenger.

Motion and Transition Messages

Choose Motion or Transition in the **When** pop-up menu or the **Message/Command** pop-up menu to reveal the submenu options described below.



*Motion and Transition messages in the **When** and **Message/Command** pop-up menus*

How mTropolis handles motion and transition messages

These messages are sent by mTropolis when it detects elements in motion or when a transition starts or stops. The motion or transition environment messages are sent to the parent of the modifier that triggered the motion or transition where they relay and cascade. For example, if a Simple Motion modifier is directly on an element, the element and all of its descendants receive the Motion Started message when the modifier is enabled. However, if the Simple Motion modifier is inside of a behavior located on the element. Only the behavior and all of its descendants receive the message. Other modifiers at the element level would not receive the Motion Started message.

This feature allows the messages associated with multiple Motion or Transition modifiers on a single element to be differentiated. Each Motion modifier can be put into its own behavior. The appropriate motion or transition messages will be sent to the individual behaviors themselves, not to the entire element.

Using motion and transition messages with the When pop-up menu

Select a motion or transition message in a **When** menu to activate or deactivate the modifier when the message is received, either from mTropolis or from a messenger.

Using motion and transition messages with the Message/Command pop-up menu

Select a motion or transition message in a **Message/Command** menu to send the message to a targeted destination just as if the mTropolis environment had generated the message. Note, however, that sending one of these messages does not actually make motion or a transition happen.

Motion and transition message descriptions

The following motion and transition messages are available in the When and Message/Command pop-up munes.

Motion Started

mTropolis sends a Motion Started message to the parent of a Motion modifier when an element starts moving in response to that Motion modifier being enabled (when the end user drags the element or the element starts simple, path, point, or vector motion).

Motion Ended
mTropolis sends a Motion Ended message to the parent of a Motion modifier when an element stops moving.

Transition Started
mTropolis sends a Transition Started message to the parent of an Element Transition modifier when that transition starts. Note that this message refers only to Element Transitions, not Scene Transitions (see "Scene Transition Ended" later in this chapter).

Transition Ended
mTropolis sends a Transition Ended message to the parent of an Element Transition modifier when that transition has finished. Note that this message refers only to Element Transitions, not Scene Transitions (see "Scene Messages" later in this chapter).

Scene Transition Ended
mTropolis sends a Scene Transition Ended message to a scene after the scene has begun (after the Scene Started message) and any Scene Transition assigned to that scene (see "Scene Transition Modifier" in Chapter 12) has finished playing. This message is useful because many scene transitions, such as "fade," create scenes that are not yet visible when the Scene Started message is sent. Scene Started is sent to the scene followed immediately by Scene Transition Ended.

Object Messages and Commands

Choose Object in the **When** or **Message/Command** pop-up menu to reveal the submenu options described below.

*Object messages in the **When** and **Message/Command** pop-up menus*

How mTropolis handles object messages and commands

Most of the items in the Object submenu are message/command pairs. The Cloned and Killed environment messages listed on the **When** pop-up menu are only sent by mTropolis when it detects an element that has responded to a corresponding **Clone** or **Kill** command. Commands are not sent by mTropolis, they can only be sent from messengers. The Do message is a special environment message generated by the List Variable.

Using object messages with the When pop-up menu

Select an Object message in a **When** pop-up menu to activate or deactivate the modifier when the message is received, either from mTropolis or from a messenger.

Using Object messages and commands with the Message/Command pop-up menu

Select an Object message or command in a **Message/Command** pop-up menu to send the message or command to a targeted destination.

Object Message/Command descriptions

Object options found in the **When** and **Message/Command** pop-up menus are described below. Note that many of these options are described as related message/command pairs. The message found in the **When** menu (for example, Cloned) is described first, followed by the command found in the **Message/Command** menu (for example, **Clone**), that (when received by an element) causes the message to be generated.

Cloned/Clone

mTropolis sends a Cloned message to an object when it has been created with the **Clone** command. Send the **Clone** command to an object to create a new instance of that object. The **Clone** command can be sent to any scene, element, or modifier but has no effect when sent to larger structural components (such as subsection, section, and project components). The newly-created object has exactly the same properties as the original, including name, position, appearance, and modifiers with the following exceptions:

- Acloned element occupies the next highest layer order number. Any existing elements in higher layer orders move up one layer to accommodate the new element, if necessary.
- A cloned object becomes the original object's "next" sibling. All existing siblings of the original object move one position later in the message passing order to accommodate the new object. Therefore, the most recent clone of an element could be targeted in Miniscript expressions as `element.next`. If there are two clones of an object, the first clone could be addressed as `element.next.next`, and the second clone (the most recent one) as `element.next`.
- Internally, each mTropolis object has a unique ID. Therefore, though clones have the same "name" as the original objects, they do not automatically receive messages targeted at the original by name. They are completely autonomous objects.

!!! Note: If the run-time **Player Emulation** option is turned off (that is, if the **File ➝ Run ➝ Player Emulation** menu toggle is unchecked), clones are "persistent" between run-time and edit modes. That is, if a clone is created while examining a mTropolis project in run-time mode, the clone will exist as a new component upon returning to edit mode. The **Kill** command, described below, could be used to remove unwanted clones in run-time. Cloning is *not* persistent if **File ➝ Run ➝ Player Emulation** is checked (the default).

Killed/Kill

mTropolis sends the Killed message to an object just before it is deleted from a project in response to the **Kill** command. Send the **Kill** command to an object to delete it from the project in run-time mode. The object and all of its children are removed from the project. **Kill** is most useful for deleting clones of objects, not original object, though it can be used to delete any object. The **Kill** command can be sent to any element or modifier. It has no effect when sent to larger structural components (such as scene, subsection, section, and project components).

!!! Note: Like the **Clone** command, the results of **Kill** are persistent between run-time and edit mode if the run-time **Player Emulation** option is turned off (if the **File ➝ Run ➝ Player Emulation** menu toggle is unchecked). That is, upon

returning to edit mode, the killed element is no longer present in the project. Use caution (and the **File ➝ Save** or **File ➝ Save As** menu options) when testing projects that use the **Kill** command.

Do

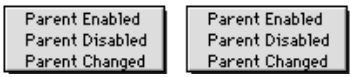
mTropolis sends a Do message to an object when a list's forAllDo attribute has been set to that object (using a Miniscript set statement). This message is used to implement "looping" in mTropolis. See "List Variable" in Chapter 12 for more information about the forAllDo attribute.

The Do message is sent once for each item in the list (for example, if the list has 5 items in it, the Do message is sent 5 times). Each Do message is sent with an accompanying data value. This data value is the value contained at the corresponding position in the list. For example, the value in the first position of the list is sent with the first Do message, the value in the second position of the list is sent with the second Do message, etc.

As sent by mTropolis, this message does not "cascade" or "relay." It is sent only to the specified object and no others. Therefore, if the Do message is sent to an element, only the first modifier that is configured to listen for the Do message receives the message.

Parent Messages

Choose Parent in the **When** pop-up menu or the **Message/Command** pop-up menu to reveal the submenu options described below.



*Parent messages in the **When** and **Message/Command** pop-up menus*

How mTropolis handles parent messages

mTropolis sends the Parent Enabled message to the scene when the scene starts. Unlike most environment messages, this message cascades through the scene, activating all modifiers and behaviors that are waiting for this message when the scene begins.

The Parent Enabled message is also sent to the components of a switchable behavior modifier when the behavior is switched on (when it receives the message specified in its Enable When field — see "Behavior" in Chapter 12).

The Parent Disabled message is sent to the scene when the scene ends (by changing to another scene, exiting a mTropolis run-time application, or re-entering edit mode from run-time mode in the editor). This message also cascades through the scene, activating all modifiers and behaviors that are waiting for this message when the scene ends.

The Parent Disabled message is also sent to the components of a switchable behavior modifier when the behavior is switched off (when it receives the message specified in its Disable When field — see "Behavior" in Chapter 12).

The Parent Changed message is sent to a component when its "parent" attribute has been changed.

Using parent messages with the When pop-up menu

Select a parent message in a **When** menu to activate or deactivate the modifier when the message is received, either from mTropolis or from a messenger.

Using parent messages with the Message/Command pop-up menu

Select a parent message in a **Message/Command** menu to send the message to a

targeted destination just as if the mTropolis environment had generated the message. Note, however, that sending one of these messages does not actually enable or disable the recipient's parent. In general, these messages should not be sent by the author.

Parent message descriptions

Parent Enabled

mTropolis sends this message to an element when its parent is enabled. Elements are enabled when the scene starts; behaviors are enabled when they receive the message specified in their Enable When field.

Parent Disabled

mTropolis sends this message to an element when its parent is disabled. Elements are disabled when the scene ends; behaviors are disabled when they receive the message specified in their Disable When field.

Parent Changed

mTropolis sends this message to an element when it is reassigned to a new parent by changing the value of its "parent" attribute. See "parent" in Chapter 15.

Scene Messages

Choose Scene in the **When** pop-up menu or the **Message/Command** pop-up menu to reveal the submenu options described below.

*Scene messages in the **When** and **Message/Command** pop-up menu*

How mTropolis handles scene messages

When navigating a multimedia title, actions often take place at the beginning and end of a scene. The scene messages are related to these events.

mTropolis cascades the scene messages described below to the scene, its child elements, and every modifier and behavior that has been placed on them.

Using scene messages with the When pop-up menu

Select a scene message in a **When** menu to activate or deactivate the modifier when the message is received, either from mTropolis or from a messenger.

Using scene messages with the Message/Command pop-up menu

Select a scene message in a **Message/Command** menu to send the message to a targeted destination just as if the mTropolis environment had generated the message. Note, however, that sending one of these messages does not actually cause the current scene to start, end, deactivate, or reactivate.

Scene message descriptions

Scene Started

mTropolis sends a Scene Started message to all elements of the scene, including the scene itself, the moment a scene begins.

Scene Ended

mTropolis sends a Scene Ended message to all elements of the scene, including the scene itself, when the scene ends (when a Change Scene modifier is executed).

Scene Deactivated

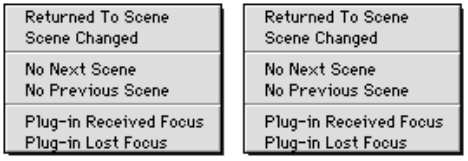
mTropolis sends this message to the scene and all of its elements when a Change Scene or

Navigation modifier is executed with the **Add to Destination** Scene check box checked (see "Change Scene Modifier" in Chapter 12). This message is sent in lieu of the Scene Ended message.

Scene Reactivated
mTropolis sends this message to the scene when returning to that scene (via the Return modifier — see "Return Modifier" in Chapter 12) and that scene was previously changed from (via the Change Scene or Navigation modifier) with the **Add to Destination Scene** check box checked. In other words, this message is sent when the scene is "reactivated" after having a new active scene "layered" over it. This message is sent in lieu of the Scene Started message.

Shared Scene Messages

Choose Shared Scene in the **When** pop-up menu or the **Message/Command** pop-up menu to reveal the submenu options described below.



*Shared scene messages in the **When** and **Message/Command** pop-up menus*

How mTropolis handles shared scene messages

Unlike most environment messages, mTropolis cascades shared scene messages to the shared scene, its child elements, and every modifier that has been placed on them.

Using shared scene messages with the When pop-up menu

Select a shared scene message in a **When** menu to activate or deactivate the modifier when the message is received, either from mTropolis or from a messenger.

Using shared scene messages with the Message/Command pop-up menu

Select a shared scene message in a **Message/Command** menu to send the message to a targeted destination just as if the mTropolis environment had generated the message. Note, however, that sending one of these messages does not actually cause the properties of the shared scene to change.

Shared scene message descriptions

Returned To Scene
mTropolis sends a Returned to Scene message to the shared scene and its elements and modifiers when a Return modifier is executed in the project during run-time.

Scene Changed
mTropolis sends a Scene Changed message to the shared scene, its elements and modifiers at the start of each scene during run-time.

No Next Scene
mTropolis sends a No Next Scene message to the shared scene, its elements and modifiers when a scene begins and that scene has no scenes following it (the scene is the last in the scene order of its subsection).

No Previous Scene
mTropolis sends a No Previous Scene message to the shared scene, its elements and modifiers when a scene begins and that scene has no scenes before it (the scene is the first in the scene order of its subsection).

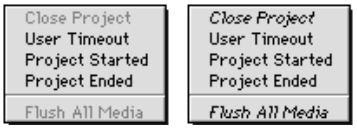
Tip: No Next Scene and No Previous Scene can be used to show and hide scene navigation buttons located in the shared scene.

Plug-In Received Focus
This message, sent only by the mPire web-browser plug-in, is sent to the shared scene if the end user clicks inside the mPire title's draw area after it previously lost the cursor "focus" (see "Plug-In Lost Focus" in this chapter).

Plug-In Lost Focus
This message, sent only by the mPire plug-in player, is sent to the shared scene when the end user clicks in a different frame or browser window, causing the mPire title to lose the cursor "focus."

Project Messages and Commands

Choose Project in the **When** pop-up menu or the **Message/Command** pop-up menu to reveal the submenu options described below.



*Project messages in the **When** and **Message/Command** pop-up menu*

How mTropolis handles project messages and commands

Project messages and commands relate to an entire project:

- The **Close Project** command can be sent to the project to quit a mTropolis project in run-time.
- The User Timeout message is generated when there has been no end user activity for a specified time. The timeout duration can be changed using the **Set Attribute** command (see "Get and Set Attribute Commands" in this chapter).
- The Project Started and Project Ended messages are sent to the project component when a project begins or ends.
- The **Flush All Media** command can be sent to the project to globally remove media from memory.

Using project messages with the When pop-up menu

Choose a project message in a **When** menu to activate or deactivate the modifier when the message is received, either from mTropolis or from a messenger.

Using project messages and commands with the Message/Command pop-up menu

Choose a project message or command in a Message/Command menu to send the message to a targeted destination just as if the mTropolis environment had generated the message.

Project message descriptions

Close Project (Message/Command menu only)
Send the **Close Project** command to the project to quit a mTropolis run-time project.

User Timeout
mTropolis sends a User Timeout message to the shared scene and the active scene, as well as all of their elements and modifiers when there have been no end user actions for a specified time. The timeout duration can be changed using the **Set Attribute** command or Miniscript to change the value of the userTimeout attribute (see "Get and Set Attribute Commands" in this chapter and "userTimeout" in Chapter 15).

!!! Note: While end user mouse clicks are interpreted as end user actions and cause the timeout timer to be reset, end user mouse motions do not.

Project Started
mTropolis sends a Project Started message to the project component, and all of its modifiers, when the project starts. This message does not cascade any further through the project.

Project Ended
mTropolis sends a Project Ended message to the project component, and all of its modifiers, when the project ends (when the end user quits the title). This message does not cascade any further through the project.

Flush All Media (Message/Command menu only)
Send the **Flush All Media** command to the project to remove all loaded media from memory. Sent by itself, this command causes all media currently loaded to be unloaded from memory. This command can be sent with an accompanying integer that represents a flushPriority value. When sent with an accompanying value, media are flushed only if their element's flushPriority attribute is less than or equal to the accompanying value. For more information, see "flushPriority" in Chapter 15.

Modifier Messages

Choose **Modifier Messages** in the **When** pop-up menu or the **Message/Command** pop-up menu to reveal the submenu options described below.

How mTropolis handles Modifier Messages
The **Modifier** messages are status messages sent by some mTropolis modifiers. Each message has its own, unique criteria for being sent as described below. Not all modifiers send such messages.

Using Modifier Messages with the When pop-up menu
Choose a modifier message in a **When** menu to activate or deactivate the modifier when the message is received.

Using Modifier Messages with the Message/Command pop-up menu
Choose a modifier message in a **Message/Command** menu to send the message to a targeted destination just as if the mTropolis environment had generated the message.

Modifier Messages submenus options in the When and Message/Command pop-up menus

Net Messaging modifier messages

Network Failed

The Net Messaging Service (Chapter 12) can report this error message. The Net Messaging Service sends this message when network startup has failed or when network resources have been depleted and no connections can be accepted or initiated. This message is sent to the parent of the Net Messaging Service.

Connection Timed Out

This message is sent to the parent of the Net Messenger when the Net Messaging Service has failed to connect to the targeted host. The message is sent after the timeout period of the network messaging service has elapsed (see the description of the timeoutPeriod attribute in "Attributes of the Network Messaging Service" in Chapter 12). A string that contains the host name or IP address of the targeted host, is sent as incoming data.

Host Name Found

This message is sent to the parent of the Net Messenger when a host name is successfully resolved (during a network messaging attempt). A string, that contains the host name, is sent as incoming data. For more information about host names, see "Host Pop-Up Menu" in Chapter 12.

Host Name Not Found

This message is sent to the parent of the Net Messenger when an attempt is made to send a message to an invalid host name (that is, the host name cannot be resolved by the DNS server) or if there are other DNS problems. A string that contains the host name that could not be resolved is sent as incoming data. For more information about host names, see "Host Pop-Up Menu" in Chapter 12.

Connection Opened

This message is sent to the parent of the net messenging service the first time a Net Messenger successfully connects to a specifiec host. A string that contains the host address, is sent as incoming data.

Connection Closed

This message is sent to the parent of the Net Messaging Service when the connection to its targeted host is closed (for example, when the targeted project has been closed or the targeted project's network service has been disabled). A string that contains the host address is sent as incoming data.

Max Connections Exceeded

This message is sent to the parent of the Net Messenger when the maximum number of simultaneously open connections has been reached and a new outgoing connection request has been received. The maximum number of connections is controlled by the maxConnections attribute of the Net Messaging Service (see "Attributes of the Network Messaging Service" in Chapter 12).

For example, suppose that maxConnections is set to 5 and that there are already 5 open network connections. A network message sent to a new address (an address for which a connection is not already open) may or may not cause this error message to be sent. If all 5 connections are busy (waiting for an acknowledgment for data just sent), the error message will be sent. If the connections are not all busy, the connection which has had no activity for the longest time will be closed to accommodate the new request and the network message will be successfully sent, without generating the Max Connections Exceeded message.

Net Operation modifier messages

The Fetch Net Text and Post Net Text modifiers can report the following messages.

Net Operation Completed
This message is sent to the parent of the modifier when the network operation is successfully completed.

Net Operation Failed
This message is sent to the parent of the modifier if the network operation could not be completed. For example, this message would be generated if the specified URL does not exist.

Net Operation Stopped
This message is sent to the parent of the modifier when the modifier's stop attribute is set to true. See "Miniscript Accessible Attributes of the Fetch Net Text Modifier" in Chapter 12.

Open Application modifier messages

The Open Application modifier (Chapter 12) can report the following error message.

Open Application Failed
This message is sent to the parent of the Open Application modifier when the specified document or application cannot be found. If an error occurs after the modifier finds the application or document, this message is not sent.

Get and Set Attribute Commands

Choose **Get Attribute** or **Set Attribute** in the **Message/Command** pop-up menu to reveal the submenu options described below. All of these options are commands; they are not available in the **When** menu. Commands are not sent by mTropolis, they can only be sent from messengers.

Element attributes

Every element in a project has inherent attributes that allow the element or its linked media to be controlled and modified. Many attributes are accessible via the **Get Attribute** and **Set Attribute** commands listed in the **Message/Command** pop-up menu. Note that the Miniscript modifier can also be used to get and set attributes, including advanced attributes that cannot be selected from the **Message/Command** menu. See "Setting Values of Element Attributes" in Chapter 14.

The items listed in the Get Attributes and Set Attributes submenus vary according to the media type associated with the element on which the messenger is placed. For example, a messenger placed on a PICT has access to the graphic element attributes such as height, width, position, and layer order number.

Using the Get Attribute command

To retrieve the current value of an attribute, choose one of the **Get Attribute** commands from the Get Attribute submenu. The **With** pop-up menu must be used to select a variable in which to store the attribute's value. When the element receives the selected **Get Attribute** command, the chosen variable is set to the attribute's value. Note that the variable must be of the same data type as the attribute being retrieved (for example, the **Set range** command must be accompanied by an Integer Range Variable). The data types of attributes are listed in the descriptions of individual attributes (see "Attribute Descriptions" in Chapter 15).

Using the Set Attribute command

To set the value of an attribute, choose one of the **Set Attribute** commands from the Set Attribute submenu. These commands require the new value of the attribute to be sent with the command. This value must be of the same data type as the attribute being set. For example, the **Get width** command requires an integer value, representing pixels, to be used as the new element width. Use the **With** pop-up menu (see "The 'With' Pop-Up Menu" later in this chapter) to specify the value to be sent. The data types of attributes are listed in the descriptions of individual attributes (see "Attribute Descriptions" in Chapter 15).

The **Get Attribute** and **Set Attribute** commands can also be used in more complex operations. For example, these commands can be used with variables to store and retrieve the attributes of elements that change over time. For example, a messenger on a draggable element can be configured to get its position when it receives a Scene Started message. The position of the element can be stored in a Point Variable, and later accessed by a second messenger that is configured to reset the element's position upon receipt of a Scene Ended message. During run-time, regardless of where the end user has dragged the element, it will be returned to its original location when the scene ends.

Get and Set Attribute command descriptions

The following Get Attribute and Set Attribute command are available in the Message/Command menu. Note that different options are available depending on the type of media associated with the element being modified by the Messenger.

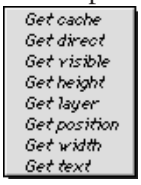
The figures below show the Get Attribute submenu options that are available for graphic elements (such as PICTs), text elements, video elements (such as QuickTime), mToons, sound elements, and projects. The Set Attribute submenu options are identical to the options shown below, except that the word "Set" replaces the word "Get" in the command name.

Information on individual attributes can be found in "Attribute Descriptions" in Chapter 15.

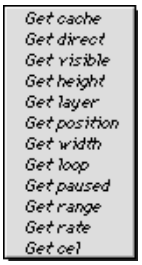
Get/Set attribute options by media type

These options are available for graphic elements such as PICTs:

These options are available for text elements:

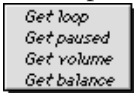


These options are available for mToons:



These options are available for QuickTime movies:

These options are available for sound elements:



The following options are available when a messenger is attached to the project component itself. Such a modifier can only be placed in the structure view:

The "With" Pop-Up Menu

Use the **With** pop-up menu to select a value to be sent with the message or command chosen in the **Message/Command** menu.

The choices on this pop-up menu are None, Incoming Data, and any variables to which the element has access. A data value can be sent by highlighting the With text field and typing in a value. These menu items are described in more detail below:

- **None:** This is the default selection. No value is sent with the message.
- **Incoming Data:** Choose this option to configure the messenger to send the incoming data (the value that arrives with the message that has been configured to trigger the messenger), with the outgoing message or command.

  Variables: To send the value of a variable modifier with the outgoing message, choose its name from the submenus available in the second section of the **With** pop-up menu. All variable modifiers currently available to a messenger from its position in the project's hierarchy are shown. These variable modifiers are only accessible to those modifiers or Miniscripts in "scope." See "Variable Scopes" later in this chapter.
- **Constant Data Value**: To send a constant data value, highlight the With text field and type the value to be sent. The value should be entered with the same syntax as if it were being used in a Miniscript modifier. The syntax of the expression entered is checked when **OK** is clicked. Any appropriate mTropolis data type can be sent.

The Destination Pop-Up Menu

Use the **Destination** pop-up menu to choose a destination for the message or command selected in the **Message/Command** menu.

The default for this menu is "Element," meaning that the message or command is sent to the element on which the messenger is placed. The destination can be changed by selecting a new option from the menu.

*Typical **Messenger** dialog box with **Destination** menu*

Destinations can be chosen by name or by relative position in the project hierarchy. The ability to target a relative rather than an absolute destination for a message allows the elements of a project to be changed without having to reconfigure the messengers that target them. This feature is particularly useful when the components of a project are to be used in other projects. More information about hierarchical relationships between elements can be found in "Message Passing among Elements" in Chapter 8.

Destination option descriptions

Choose an item from the **Destination** pop-up menu to indicate where in the project the message or command is to be sent. Note that, by default, messages are sent not only to the targeted destination, but also to all the components within the targeted destination (that is, they "cascade" and "relay"). This behavior can be changed using the Message Options section of the **Messenger** dialog box (see "Message Options" in Chapter 12.

Destination options are described below. Note that only the default behavior of the message is described (messages are assumed to be set to "cascade" from one level of the hierarchy to another and to "relay" from one modifier to another).

!!! Note: Do not send cascaded messages to a project, section, or subsection unless absolutely necessary. Every object "below" the target of a cascaded message must be loaded into memory to process the message. In a large project, section, or subsection this may cause significant performance and memory problems. Always uncheck the "cascade" message option when sending a message to a component with many descendants.

Project

The project destination refers to the topmost structural level of the mTropolis hierarchy. The project component and every component contained within the project (all sections, subsections, scenes, and components of those scenes) receives the message.

Note: Use the structure view to see or place modifiers at the project level.

Section
The section destination refers to the section in which the messenger resides. The section component and every item contained in the section (all subsections, scenes, and components of those scenes) receives the message.

Note: Use the structure view to see or place modifiers at the section level.

Subsection
The subsection destination refers to the subsection in which the messenger resides. The subsection component and every item contained in the subsection (all scenes and components of those scenes) receives the message.

Note: Use the structure view to see or place modifiers at the subsection level.

Scene
The scene destination refers to the scene of which the messenger is a part. Directing a message to a scene results in all elements on the scene receiving the message, including the scene itself. This destination is useful when multiple elements in a scene need to be sent the same message.

Element
Element is the default selection. The message or command is sent to the element that contains the messenger. All modifiers on the element receive the message. The message then cascades to any child elements.

Next Element
Choose Next Element to send a message or command to the next sibling of the element associated with the messenger (the next element in the messaging order of elements in a component).

Previous Element
Choose Previous Element to send a message or command to the previous sibling of the element associated with the messenger (the previous element in the messaging order of elements in a component).

Messenger's Parent
The parent of the messenger receives the message or command. The parent of a messenger is its "container," whether that container is an element or behavior.

Element's Parent
The parent of the element associated with the messenger receives the message or command.

Active Scene
Choose Active Scene to send a message to the scene visible to the end user when the message is sent. By default, all elements and their modifiers on the active scene, as well as the active scene and its modifiers receive the message.

Shared Scene
Choose Shared Scene to send a message to all elements and their modifiers on the shared scene, as well as the shared scene and its modifiers. For additional information regarding shared scenes, see "Shared Scenes" in Chapter 10.

Source's Parent
The source's parent is the parent of the messenger that sent the message that activated the current messenger. This destination is invalid if the messenger is configured to respond to an environment message sent by mTropolis.

Messenger's Siblings
The names of the messenger's sibling modifiers appear on the Messenger's Siblings submenu. Target any one of these modifiers by choosing its name from this list.

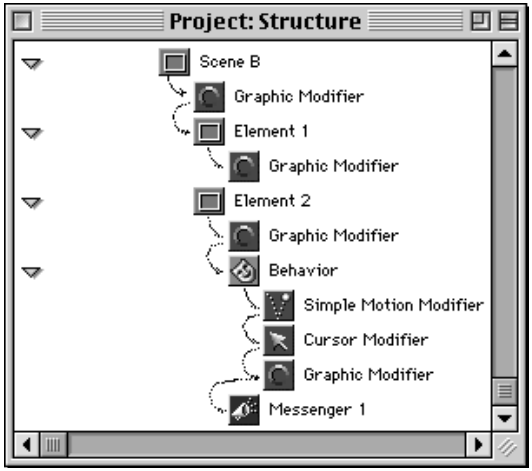
Element's Siblings
The names of the sibling elements of the element associated with the messenger appear on the Element's Siblings submenu. Target any one of these elements directly by choosing its name from this list.

Ancestors
The names of all ancestors of the messenger (that is, components further up the structure hierarchy) appear in the Ancestors submenu. Target any of these components directly by choosing its name from this list.

Message Paths

By default, messages sent from messengers are sent to their target destination and then they continue down through the child components that destination contains. In this way, a single message can be used to activate many modifiers in components throughout a project.

The figure below shows a sample scene as it would appear in the Structure window. Each element in this window appears on its own level, in a hierarchy from left to right. The descending order of modifiers and components within these components is called their "messaging order." The arrows trace the default path of a message targeted to Scene B. Notice how the message descends each level of the structure hierarchy and that the message could be activating any of the modifiers in the scene.



Messaging order. The arrows show the path of a cascading and relaying message sent to Scene B

A message is said to "Cascade" if it travels through the structure hierarchy from component to component, structure level to structure level. A message is said to "Relay" if it continues from modifier to modifier, activating as many modifiers as possible. This path is the default for messages sent from a messenger.

For example, in the figure above, when Scene B receives the message it is relayed to its Graphic modifier. From there the message "cascades" to Element 1, the next level in the structure hierarchy. The element then relays the message to Element 1's Graphic modifier. The message cascades once again to the next component in the scene, Element 2. The message is then relayed from Element 2 to every remaining modifier in the scene. When the last item in the scene has received the message, its path is complete.

The default path of messages sent by messengers can be changed as described in "Message Options" in this chapter.

Default path of messages sent by the mTropolis environment

Note that the path of messages sent to elements from the mTropolis environment (end user mouse click messages) is slightly different than the default for messages sent by messengers. mTropolis-sent environment messages typically do not cascade, but they do relay. In other words, the message is sent to the target element, activates any modifiers on that element that are set to respond to the message, and then stops.

Message Options

All **Messenger** dialog boxes have a Message Options section that can be revealed by clicking the triangle at the bottom of the dialog box. This dialog box can be used to more precisely target and time messages sent from a messenger.

Options in this section are:

Immediate check box

By default, messages or commands are sent immediately when the messenger is activated. Uncheck this box to configure the messenger to send its message or command only after the current thread of messages has ended. This option can be used to avoid system overload if the message queue becomes too deep.

Cascade check box

By default, the message "cascades" from the targeted destination to all components in the hierarchy below it. Uncheck this option to configure the messenger to send its message only to the modifiers in a targeted element. "Cascade off" is equivalent to the path of most environment messages sent by mTropolis.

!!! Note: This option has no effect when sending a command. Commands act only on the targeted element.

Relay check box

By default, messages travel from object to object in the hierarchy activating any and all objects that respond to the message. Uncheck this box to activate only the first modifier in the path of the message that is configured to respond. In effect, the first modifier to respond to the message "swallows" the message.

!!! Note: This option has no effect when sending a command. Commands act only on the targeted element.

Variable Scopes

Variable modifiers (or simply "variables") can be chosen from the **With** menu (see "The 'With' Pop-Up Menu" in this chapter) and referenced in the text of Miniscript modifiers, if the messenger or Miniscript modifier exists in the variable's scope. That is, a variable modifier's scope determines which modifiers can "see" and make use of that variable modifier.

The scope of the variable is determined by its position in the project's structural hierarchy. *A variable is accessible to all descendants of its parent.*

Illustrating variable modifier scopes

A variable can be accessed by any descendant of the variable's parent. For example, a variable modifier placed on a section is available to any modifier on that section, and all the section's "descendants." An object descendants include any objects found "under" it in the hierarchy, no matter how many layers deep.

A variable modifier whose parent is the project is said to be *global*. Since all components of a project are its descendants, any variable placed at the project level can be accessed by any appropriate modifier in the entire project.

For example, an Integer Variable modifier that keeps track of the end user's current score would probably need to be updated by many different modifiers throughout the title. Placing the variable on the project ensures that any modifier that may use the variable has access to it.

To create a global variable:

1. Use the structure view to display the project's hierarchy.
2. Drag a variable modifier from its palette and drop it on the project icon (the topmost icon in the structure hierarchy). For more information on using the Structure window, see "Adding Components to a Project in the Structure Window" in Chapter 8.

For a more localized scope, variable modifiers can be placed in specific components anywhere in the project. It is a descendant of the variable modifier's parent (that is, it is within the variable modifier's scope).

Variable persistence

When testing a project, keep in mind that variables will behave differently depending upon whether the **Run ➛ Player Emulation** menu toggle (found in the **File** menu) is checked or unchecked.

When the **Player Emulation** toggle is *checked* (the default), variables at the scene level of the structural hierarchy and lower revert to their original values upon returning to edit mode. Variables located higher than the scene level (in Subsection, Section, and Project components) retain any changes that are made to them during run-time mode.

When the **Player Emulation** toggle is *unchecked,* the values of all variables are persistent between run-time and edit mode. That is, when a variable's value is changed in run-time mode, the value remains changed upon returning to edit mode.

Note: There is one exception to this behavior. List variables located at the scene level or lower always lose their values upon switching between run-time and edit mode unless they are specifically configured to preserve run-time changes. See "Preserve Run-Time Changes Check Box" in Chapter 12.

Persistence of variables between scene changes in built titles

In a built title, variables return to their "original" values (the values they had when the title was built) *whenever the scene in which they reside is loaded*. This behavior can be confusing as variables behave differently in the mTropolis editing environment depending upon the current **Player Emulation** setting.

Consider a scene that contains an Integer Variable named "myInt". myInt is configured to have an original value of 3. Suppose that some end user mouse event causes the variable's value to be changed to 5. Now suppose that a scene change occurs (for example, the end user navigates to the next scene) and then, sometime later, the project returns to the original scene. The value of myInt will be different depending upon how the project is being run:

- If the project is being run in the mTropolis editor with **Player Emulation** *unchecked,* the

value of myInt will still be 5 upon returning to the original scene. The value stays the same because in run-time mode scenes are only loaded once. They are not unloaded from memory when the scene is changed, nor are they reloaded when a scene is displayed multiple times.

- If the project is being run in the mTropolis editor with **Player Emulation** *checked* (the default), or if the project is made into a built title and run by the mTropolis player, myInt will contain 3 upon returning to the original scene. The value is reset to 3 because the original scene is unloaded from memory when the scene changes and then reloaded into memory upon returning to the original scene. Variables are reset to their original values whenever the scene in which they reside is loaded.

This discrepancy can cause confusion when authoring complex titles. Note also that there are two techniques to make changes to variables persistent across scene changes:

- Use the **Object ➛ Make Alias** menu option to alias variables in a scene that need to retain their values between scene changes. Aliased variables are not unloaded from memory between scene changes and, hence, will be persistent.
- Put variables that need to retain their values across scenes into the subsection, section, or project components. Recall that variables at the project level are considered global.

Messages & Commands

Mouse Message	Destination	Incoming Data
Mouse Down	Element	mouse position
Mouse Up	Element	mouse position
Mouse Up Inside	Element	mouse position
Mouse Up Outside	Element	mouse position
Mouse Over	Element	mouse position
Mouse Outside	Element	mouse position
Mouse Tracking	Element	mouse position
Tracked Mouse Outside	Element	mouse position
Tracked Mouse Back Inside	Element	mouse position

Element Message/Command	Destination	With Data
Shown/*Show*	Element	none
Hidden/*Hide*	Element	none
Selected/*Select*	Element	none
Deselected/*Deselect*	Element	none
Toggle Select	Element	none
Scroll Up	Element	number of pixels (default 20)
Scroll Down	Element	number of pixels (default 20)
Scroll Left	Element	number of pixels (default 20)
Scroll Right	Element	number of pixels (default 20)
Preload Media	mToon, QuickTime, Sound Element	range or cel/timevalue
Flush Media	mToon, QuickTime, Sound Element	none
Preroll Media	mToon, QuickTime, Sound Element	none
Trickle Load Done/ *Trickle Load Media*	mToon, QuickTime, Sound Element	range or cel/timevalue

Text Message/Command	Destination	With Data
Enable Editing	Text Element	none
Edit Done/*Disable Editing*	Text Element	none
Update Calculated Fields	Text Element	none

Messages & Commands

Play Control Message/Command	Destination	With Data
Played/*Play*	Element	range or sound marker
Stopped/*Stop*	Element	none
Paused/*Pause*	Element	none
Unpaused/*Unpause*	Element	none
Toggle Pause	Element	none
At First Cel	Element	none
At Last Cel	Element	none

Motion/Transition Message	Destination	Incoming Data
Motion Started	Motion Modifier's Parent	none
Motion Ended	Motion Modifier's Parent	none
Transition Started	Transition Modifier's Parent	none
Transition Ended	Transition Modifier's Parent	none
Scene Transition Ended	Scene	none

Object Message/Command	Destination	Incoming Data
Cloned/*Clone*	Element	none
Killed/*Kill*	Element	none
Do	Targeted Object (by list.*forAllDo*)	list item

Parent Message	Destination	Incoming Data
Parent Enabled	Element	none
Parent Disabled	Element	none
Parent Changed	Element	none

Scene Message	Destination	Incoming Data
Scene Started	Scene	none
Scene Ended	Scene	none
Scene Deactivated	Scene	none
Scene Reactivateed	Scene	none

Messages & Commands

Shared Scene Message	Destination	Incoming Data
Returned To Scene	Shared Scene	none
Scene Changed	Shared Scene	none
No Next Scene	Shared Scene	none
No Previous Scene	Shared Scene	none
Plug-in Received Focus	Shared Scene	none
Plug-in Lost Focus	Shared Scene	none

Project Message/Command	Destination	With Data
Close Project	Project	none
User Timeout	Shared Scene & Active Scene	none
Project Started	Project	none
Project Ended	Project	none
Flush All Media	Project	maximum *flushPriority*

Modifiers Message	Destination	Incoming Data
Open Application Failed	Modifier's Parent	none
Network Failed	Net Messaging Service's Parent	none
Host Name Found	Modifier's Parent	host address
Host Name Not Found	Modifier's Parent	host name
Connection Opened	Net Messaging Service's Parent	host address
Connection Closed	Net Messaging Service's Parent	host address
Connection Timed Out	Modifier's Parent	host address
Max Connections Exceeded	Modifier's Parent	none
Net Operation Completed	Modifier's Parent	none
Net Operation Failed	Modifier's Parent	none
Net Operation Stopped	Modifier's Parent	none

Attribute Command	Destination	With Data
Set Attribute	Element	new value
Get Attribute	Element	variable

14 Miniscript Modifier

14. Miniscript Modifier

The Miniscript modifier can be used to access the scripting language built into mTropolis. Miniscript is a simple scripting language that can be used to:

- Access and change element attributes, such as the screen position of elements.
- Perform mathematical operations.
- Get and set the value of variable modifiers that lie within the scope of the Miniscript modifier.
- Perform comparisons and conditional branching using relational operators and “if” statements.
- Send multiple messages and/or commands to components of a project.

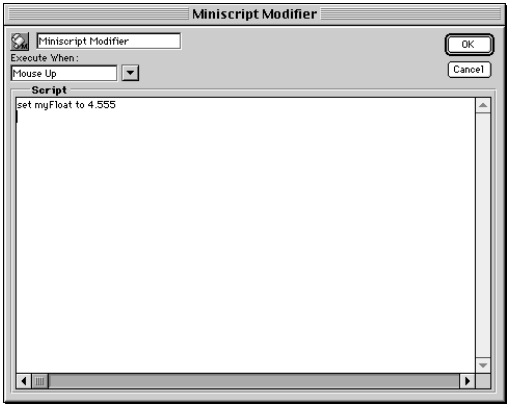
This chapter describes Miniscript syntax, statements, operators, and miscellaneous functions.

Miniscript expressions can also be used in the If Messenger (see Chapter 12).

The Miniscript Modifier Dialog Box

Miniscript modifiers can be added to elements (and behaviors) just like any other modifier. Simply drag the Miniscript modifier from the **Modifier** palette and drop it on the element to be modified. Double-click the Miniscript modifier on an element to display its dialog box.

The **Miniscript Modifier** dialog box below shows a single-line script entered in it. Components of the **Miniscript Modifier** dialog box are described below.



*The **Miniscript Modifier** dialog box*

Editable Name field

Like all other mTropolis modifier dialog boxes, the **Miniscript Modifier** dialog box has a text entry field that can be used to give the modifier a unique and descriptive name.

Execute When pop-up menu

Use this pop-up menu to select the message that triggers the modifier. When this message is received, the Miniscript commands in the Script field are executed.

Script field

Use the Script field to edit the script associated with the modifier. Scripts can have a maximum of 32,768 characters in them. By default, this field is blank.

OK button

The script is checked for syntax and then compiled when the **OK** button is clicked. In addition, references to other mTropolis objects are resolved. If mTropolis detects any syntax or other errors during compilation, the error is flagged and the compilation process is aborted. A flashing text cursor is placed in front of the offending statement and an alert dialog box appears, describing the error.

Basic Miniscript Syntax

Miniscript uses a "natural language" syntax that makes the scripting language quick and easy to learn. Writing a script involves using a combination of reserved words, variable names, component names, and literal values (numbers, strings, etc.). An example is the script:

```
set myFloat to 4.555
```

"myFloat" refers to an existing Floating-Point Variable modifier within the scope of the Miniscript modifier (Chapter 12). The literal value 4.555 is the value that the variable modifier will assume.

Basic rules of Miniscript syntax are described below.

Comments

Any text following two dashes (--) is considered to be a comment, and is ignored when the script is compiled. Use comments to

annotate and clarify scripts. In the following example, the first line is a comment:

```
-- Add two to myFloat
set myFloat to myFloat+2.0
```

In the next example, the text "-- Initialize X" is a comment:

```
set x to 5 -- Initialize X
```

Case sensitivity

Miniscript is generally not case sensitive. Miniscript statements can be written in uppercase, lowercase, or mixed case. In this manual, most statements and keywords are written in lowercase. To improve readability, element, variable, and attribute names are sometimes written in mixed case (as in, myFloat). However, since variable names are not case sensitive, the names myfloat and MyFloat are interpreted the same by Miniscript.

!!! Note: The text of string values is also case insensitive. When performing comparisons between strings, an uppercase string such as "MY TEXT" is equivalent to the lowercase string "my text."

Continuation character

Each line of a script is a complete command terminated by a return character. To write a single command as a number of lines, add a backslash character (\) to the end of each continued line. For example, the following two lines are a single Miniscript command:

```
send "myAuthorMessage" with myFloat \
  to element's parent
```

!!! Note: In the second line shown above, the leading spaces are ignored by Miniscript; the line does not have to be indented. Spaces and tabs can be used to improve the readability of scripts.

Variables

The values of variables in a mTropolis project can be accessed and set using Miniscript. mTropolis supports two types of variables: simple variables and Compound Variables. Simple variables contain a single value. Simple variables include the Boolean, Floating-Point, Integer, and String Variable types. Compound Variables contain multiple fields that each contain separate values. Compound Variables include the Integer Range, List, Point, and Vector Variable types. See Chapter 12, "Modifier Reference," for descriptions of the individual variable modifiers.

Using variable names in scripts

Most variables can be referenced in a script by simply using the variable's name (as set in the variable modifier's editable name text box). The only exceptions are variable names that contain periods and/or spaces or names that begin with numbers or other non-alphabetical characters. When referring to a variable with such a name in Miniscript, the name must be enclosed in single quotes. For example, an Integer Variable with the name myInt could be referenced in a script as follows:

```
set myInt to 5
```

However, a variable with a name such as "my Wonderful Integer" must be enclosed in single quotes as shown below:

```
set 'my Wonderful Integer' to 5
```

Variable creation and scripts

Variables should be created before their names are referenced in a script. Writing Miniscript statements that use variable names not yet created can lead to unpredictable results. If, while writing a Miniscript, you do refer to a variable that does not yet exist, take the following steps to avoid any run-time errors:

- When finished writing the script, click **OK** to dismiss the **Miniscript Modifier** dialog box.
- Add the correct type of variable to the appropriate place in the project. Give the variable the exact name used in the script.
- Double-click the Miniscript modifier icon to re-open the script. Click **OK** to cause the variable and object references within the script to be evaluated. This ensures that references to your variable are properly resolved.

Keep in mind that variables can only be used in a script if they are within the Miniscript modifier's scope. See "Variable Scopes" in Chapter 13.

Variable and component naming

Variables should be given unique names (names that are not used by other components in the project) to ensure that Miniscript references to those variables resolve to the variable and not to some other component. For example, if a variable is given the same name as a Miniscript modifier that attempts to access that variable, an object reference to the Miniscript modifier might be returned instead of the variable's value. Also, variable and component names should never include a forward slash (/) which can confuse object references.

Accessing the fields of Compound Variables

The fields of Compound Variables can be referenced by using a period followed by the name of the field. For example, the following command sets the start value of an Integer Range Variable to 4:

```
set myIntegerRange.start to 4
```

The fields in each type of Compound Variable are listed in the descriptions of each variable modifier found in Chapter 12, "Modifier Reference." They are also described in "Setting the Value of Compound Variables" in Chapter 14.

Note: The syntax is the same for accessing the end user-created fields of custom "Compound Variables" created with the Compound Variable modifier (see "Compound Variable" in Chapter 12).

Literal Values

Literal values for most mTropolis data types can be used in scripts. Each mTropolis data type has a different syntax as shown below.

String equivalents of data types

All mTropolis data types also have string equivalents. Any mTropolis data type can be used in a string expression and the data is automatically converted to its string equivalent. This feature makes it easy to display mTropolis values in text elements or put values into String Variables.

Data Type	Syntax	Example
boolean	true or false	true
integer	*n*	578
string	"*text*"	"mFactory"
integer range	(*n* thru *n*)	(5 thru 75)
floating-point	*n.n*	109.2756
vector†	(*n*°*n.n*)	(90°2.5)
point	(*n*,*n*)	(640,480)
object reference	*object*[.*object*[...]]	myScene.myElement.myMod
list	{*n*, *n*, ..., *n*}	{"dog", "cat", "bird"}

Miniscript syntax for literal values. An "n" in the table indicates a numeral

For example, without automatic string conversion it would be difficult to display the contents of an Integer Range Variable in a text element. The num2str function and the concatenation operator would have to be used as follows:

```
set text to num2str(range.start) & \
  "thru" & num2str(range.end)
```

However, because mTropolis data types automatically convert themselves to strings, we can simply use:

```
set text to range
```

String equivalents for individual mTropolis data types are described below.

Boolean string equivalents
Boolean values become the strings "`true`" or "`false`."

Compound string equivalents
Compound Variables are converted into strings that represent the variable's structure: "`Compound Name:\n\t childName,: childValue, ...\n\t childName`$_n$`: childValue`$_n$" For example, a component variable called "myVar" that contain an Integer Variable "myInt" whose value is 42 and a Point Variable "myPoint" whose value is (10, 10) has the string equivalent (if displayed in a text element):

```
myVar:
 myInt: 42
 myPoint: (10, 10)
```

Floating-point string equivalents
Floating-point values are converted to text values that always contain a decimal point (leading or trailing zeros are added if necessary). For example, the floating-point value 123. becomes "123.0," the value .5 becomes "0.5," and the value 123.45 becomes "123.45."

Integer string equivalents
Integer values are simply converted into strings without alteration. For example, the integer 123 becomes "123."

Integer range string equivalents
Integer range values are converted into strings of the form:

"n_{start} thru n_{end}"

For example, the range (1 thru 10) becomes the string "1 thru 10." Note that parentheses are dropped.

List string equivalents
Values in a list are converted into a string that contains the appropriate string equivalents for each item separated by newline characters ("\n"):

"$item_1$\n$item_2$\n...$item_n$"

For example, an integer list defined as:

```
mylist = {123, 456, 789}
```

has the string equivalent:

```
"123\n456\n789"
```

Note that there is no trailing newline character.

Object reference string equivalents
Object reference values are converted into strings that contain the slash-delineated mTropolis path to that object. For example, an object reference to myElement (an element located in the current scene) would have the string equivalent:

```
"Project/Section/Subsection/Scene/
myElement"
```

Point string equivalents
Point values are converted into strings of the form:

"n_x, n_y"

Note the space after the comma and that parentheses are dropped. For example, the point (10,20) becomes the string "10, 20."

String equivalents of strings
String values are already strings, so they need no "conversion" to string type. Strings are used "as is" in string expressions.

Vector string equivalents
Vector values are converted into strings of the form:

"n_{deg} deg, $n.n_{magnitude}$"

Note that the degrees symbol is converted to "deg" and parentheses are dropped. For example, the vector (90°1.0) becomes the string "90 deg, 1.0."

Element names
Elements can be the targets of messages sent by Miniscript statements. Additionally, most components in a mTropolis project have attributes whose values can be retrieved or modified using Miniscript.

Most elements can be referenced in a script by simply using the element's name (as set in the element's **Element Info** dialog box). The only exceptions are element names that contain periods and/or spaces. If an element name contains these characters, the name must be enclosed in single quotes when used in a script 'my Element'.

To be referred to by name, an element must be within the Miniscript modifier's scope. See "Specifying the Destination for a Message" in Chapter 14 for more information.

Object attributes (see Chapter 15, "Attributes") can be referenced using a syntax similar to that used to reference the fields of Compound Variables. Add a period, followed by the attribute's name, to the element name. For example, the following command sets the "width" of an element:

```
set myElement.width to 120
```

Complete descriptions of mTropolis component attributes can be found in Chapter 15, "Attributes."

Message and command names
Messages and commands can be sent to elements using the **send** statement. Message or command names must be enclosed in double quotation marks. For example:

```
send "myAuthorMessage" to element
```

Names of built-in messages and commands can be found in Chapter 13, "Modifier Pop-Up Menus and Message Reference." The send statement is described in more detail on in "The Send Statement — Sending Messages and Commands" in Chapter 14.

Set — The Miniscript Assignment Statement

The Miniscript `set` statement can be used to change the values of variables or element attributes. It is Miniscript's version of an assignment operator. The basic form of this statement is:

```
set variable to value
```

where *variable* is the name of a variable or element attribute to be set and *value* is an expression of the proper type for variable.

Setting values of variable modifiers

Variable modifiers give elements the ability to store many different types of data. Miniscript can be used to alter these values.

The `set` statement can be used to modify the values of both simple and Compound Variable types as described below.

Setting the value of simple variables

Setting the value of a simple variable is easy. The following examples show the syntax used to set each type of simple variable:

- Boolean Variable:

```
set myBoolean to true
```

- Floating-Point Variable:

```
set myFloat to 4.555666
```

- Integer Variable:

```
set myInteger to 888
```

- String Variable:

```
set myString to "mFactory"
```

Setting the value of Compound Variables

The fields of Compound Variable modifiers can be set individually or together as a group. To set the value of a field individually, use the syntax:

```
set variable.field to value
```

where *variable* is the name of the Compound Variable, *field* is the name of the field to be set, and *value* is an expression of the proper type for field. Alternatively, the following syntax can be used for a more "natural language" style:

```
set the field of variable to value
```

The following examples show the syntax used to set various types of Compound Variables:

- Integer Range Variables have start and end fields. To set the start field individually:

```
set myIntegerRange.start to 4
```

To set the end field individually:

```
set myIntegerRange.end to 9
```

To set both fields:

```
set myIntegerRange to (4 thru 9)
```

- List Variables can store any number of values that are accessed with a slightly different syntax than most other variables. See "Miniscript Syntax for List Variables" in Chapter 12.
- Point Variables have an x and a y field. To set the x field individually:

```
set myPoint.x to 15
```

To set the y field individually:

```
set myPoint.y to 30
```

To set both fields:

```
set myPoint to (15,30)
```

- Vector Variables have an angle and a magnitude field. To set the angle field individually:

```
set myVector.angle to 45
```

To set the magnitude field individually:

```
set myVector.magnitude to 1.41
```

To set both fields:

```
set myVector to (45°1.41)
```

Note: The ° symbol (degrees) can be typed by pressing Shift-Option-8 on Mac OS.

- End user-created Compound Variables, made with the Compound Variable modifier, can have any number of end user-defined fields. See "Compound Variable" in Chapter 12 for complete documentation and examples.

Setting values of element attributes

Element attributes can be set with a syntax similar to that used to set the values of Compound Variable fields. To set the value of an attribute, use the syntax:

```
set element.attribute to value
```

where *element* is the name of the element, *attribute* is the name of the attribute to be set and *value* is an expression of the proper type for attribute. Alternatively, the following syntax can be used for a more "natural language" style:

```
set the attribute of element to
value
```

Note that in the statements above, the word "element" is optional since the set assumes that the script is referencing its own element's fields unless otherwise specified. Therefore, the statements above are equally valid written as:

```
set attribute to value
```

Some elements' attributes have multiple fields, such as the position attribute, which accepts a point value. To access the fields of a compound value, simply add another period and the name of the field. For example, to set the "x" component of an element's position attribute, use the statement:

```
set element.position.x to value
```

Descriptions of object attributes

For lists of object attributes and their descriptions, see "Attribute Descriptions" in Chapter 15.

Setting variables and attributes to incoming values

Messages are often sent with an accompanying data value (specified in the **With** pop-up menu of a messenger modifier or using the `with` keyword in a Miniscript `send` statement). The keyword `incoming` can be used with the `set` command to access data sent with the message that activates the Miniscript modifier. For example:

```
set myFloat to incoming
set myToon.width to incoming.x
```

Setting incoming to a new value

The incoming data value could also be set to some new value if a variable was sent as the incoming data. For example:

```
set incoming to myFloat
```

Note: Incoming data can be of any type. If a set operation is attempted between items with different types, the value of the set "target" does not change and an error message may be generated.

Constant values (for example, the position data sent with mouse messages such as Mouse Up) cannot be modified in this way. When incoming represents a constant value, attempting to set incoming to some new value will cause a "can't set" error to be displayed in the Message Log window.

Variables in mTropolis are passed by reference (as opposed to being passed by value). Therefore, if a message is sent with a variable as accompanying data, and then the value of `incoming` is changed, the actual value stored in the variable will be changed. For example, suppose a Miniscript modifier is configured to execute upon receipt of the message "myMessage" and that it contains the following script:

```
set incoming to incoming + 1
```

A messenger somewhere else in the project could be configured to send "myMessage" with a variable value (such as, an Integer Variable named "myInt"). When the Miniscript modifier is triggered by this message, it will add 1 to the value of myInt. So, if myInt previously contained the integer value 5, it now contains 6.

The Send Statement — Sending Messages and Commands

The `send` statement can be used to send environment messages, author messages, or commands from a script. Any number of messages can be sent from a single script, eliminating the need for multiple messengers. Any message or command described in Chapter 13, "Modifier Pop-Up Menus and Message Reference," can be sent with the `send` statement.

Basic use of send

The most simple form of the send statement sends a message to the element to which the script is attached:

```
send "Message"
```

The message or command enclosed in double quotes is sent to the element that contains the Miniscript modifier. Note that even if the Miniscript modifier is located inside a behavior, the message is still sent to the element that the behavior is associated with.

See Chapter 13, "Modifier Pop-Up Menus and Message Reference," for the names and descriptions of messages and commands that can be sent.

Note: The message being sent must already exist before it can be used in a Miniscript statement. If you attempt to write a Minscript `send` statement that contains a message that has not yet been created, an alert appears when the Miniscript modifier's **OK** button is clicked. Use the **Execute When** pop-up menu or the Author Messages window to create the new author message before using it in a `send` statement.

Specifying the destination for a message

Messages or commands can be sent to destinations other than the element to which the script is attached. Use the `to` keyword as shown below:

```
send "Message" to destination
```

where *destination* is an explicit or relative destination described by using the building blocks shown in the table on the next page. Expressions created from these building blocks are expressions of object reference type.

For example, to send the author message myMessage to the parent of the element to which the script is attached, use the statement:

```
send "myMessage" to element.parent
```

To send the same message to an element that is a sibling or ancestor of the element, simply use the name of the destination element. For example:

```
send "myMessage" to Frog
```

The destination can be any expression that resolves to an object. Such expressions can be composed using the building blocks shown in the table on the next page and any attributes of object reference type (see Chapter 15, "Attributes"). For example:

```
send "myMessage" to \
  source.parent.parent
```

sends the specified message to the parent of the parent of the messenger that triggered the Miniscript modifier. Note that, as with attributes and Compound Variables, you can use the "dot" syntax shown above or the "natural language" syntax. For example, the previous send statement could also be written as:

```
send "Message" to source's parent's
\     parent
```

Sending values with messages

A value can also be sent along with the message or command using the `with` keyword:

```
send "Message" with value to \
  destination
```

where *value* is an expression of the appropriate type to accompany the message or command sent. For example:

```
send "Scroll Up" with 5

send "MoveFrog" with myPoint

send "Play" with myRange to myToon

send "Mouse Up" with (100,100) to \
  element.next
```

The `incoming` keyword can also to used to access data that was received by the Miniscript modifier (as "with" data). For example:

```
send "New Left Edge" with \
  incoming to parent
```

If incoming contains a compound value, its fields can be accessed using the standard syntax:

```
send "Add Value" with \
  incoming.magnitude to scene
```

Changing the message options

Just like messages sent from messengers, messages sent with the send statement can be sent with special options. By default, messages set to "cascade" or "relay" are sent immediately.

See “Message Options” in Chapter 13 for more information on these options and message passing in general.

The `options` keyword can be used to change the default behavior of the message sent by a `send` statement. The `options` keyword must be followed by one or more option selections. Valid selections are `cascade`, `relay`, `immediate`, `all`, and `none`. The `options` keyword can also be shortened to `opt`.

The default option is `all`, meaning that the message will cascade, relay, and be sent immediately. When a single option is specified, the remaining options are turned off. The `none` option can be used to turn all options off.

For example, to send a message with all options turned off:

```
send "myMessage" to parent options none
```

To send a message that only cascades:

```
send "myMsg" to element options cascade
```

To send a message that cascades and relays, but is not immediate:

```
send "myMsg" to scene opt cascade relay
```

Target[†]	Description
project	Project containing the Miniscript modifier
section	Section containing the Miniscript modifier
subsection	Subsection containing the Miniscript modifier
scene	Scene containing the Miniscript modifier
activeScene	Currently active scene (the scene the user currently sees; not necessarily the Scene containing the Miniscript modifier)
sharedScene	Currently active shared scene
element	Element containing the Miniscript modifier (this is the default target; it can be omitted when accessing element attributes or sending messages to the element)
parent	Immediate parent of the Miniscript modifier
element.parent	Parent of the element containing the Miniscript modifier
source	Modifier that sent the message that triggered the Miniscript modifier
source.parent	Immediate parent of the modifier that sent the message that triggered the Miniscript modifier
source.element	Element containing the modifier that sent the message that triggered the Miniscript modifier
this	The Miniscript modifier itself

[†]Building blocks for targeting components in Miniscript. The “Structure Attributes” described in Chapter 15 are also useful in constructing destinations for messages

The If Statement — Conditional Branches of Execution

The `if` statement can be used to evaluate an expression and take certain actions if that expression is true. The general syntax is:

```
if expression then statement
```

The Miniscript statement represented by *statement* will be executed if *expression* evaluates to true. For example, suppose variable `a` has the value 2 and variable `b` has the value 3 when the following `if` statement is executed:

```
if a<b then send "myMessage"
```

Because `a` is less than `b`, the message will be sent.

Multiple statements can be executed by using the syntax:

```
if expression then
  statement1
  statement2
  ...
  statementn
  end if
```

The `else if` and `else` keywords can be used to specify other conditions or statements that should be executed. Use the syntax:

```
if expression1 then
 statement(s)
else if expression2 then
 statement(s)
else if expressionn then
 statement(s)
else
 statement(s)
end if
```

Each expression is evaluated in order. If an expression is true, its corresponding statements are executed. Any following conditions are skipped and execution continues with the next Miniscript statement in the script. Statements associated with the `else` section are executed if no other condition is satisfied.

For example, the following script executes the first `set` statement if the angle field of myVector is greater than 30, otherwise, the second `set` statement is executed:

```
if myVector.angle > 30 then
 set myVector.magnitude to rnd(25)
else
 set myVector.magnitude to rnd(10)
end if
```

Definition of true

The expression evaluated by the `if` statement can be any type of expression. Usually, you want to evaluate a boolean expression (which simply evaluates to either true or false). However, other data types have "true" and "false" conditions. The definition of true or false for different data types is as follows:

- Boolean values are either true or false.
- Integer and floating-point: 0 is false, any other value is true.
- Strings and compound values: these values are always false.

Miniscript Operators and Expressions

Operators are used to create expressions. The set of Miniscript operators is described below.

Parentheses ( )

Parentheses are used to group expressions and to override the natural order of operator evaluation — expressions in parentheses are always evaluated first. For example, in the statement:

```
set c to (a+b)/e
```

The expression `(a+b)` is evaluated first, then the result is divided by `e`.

See "Operator Precedence" in Chapter 14 for information on the default order of operator evaluation.

Mathematical operators

Miniscript supports the following mathematical operators, described in order of descending precedence.

Exponentiation (^)

The caret (^) is the exponentiation operator. `A^B` is equal to `A` raised to the `B` power.

Multiplication (*)

The asterisk (*) is the multiplication operator. For example, `2*5` evaluates to `10`.

Division (/)

The forward slash (/) is the division operator. For example, `6/2` evaluates to `3`.

Div (Integer Division)

The `div` operator performs a special "integer" type of division where any remainder is dropped. For example, `4.0 div 3.0` evaluates to `1`.

Mod (Modulus)

The `mod` operator performs a modulus operation. It returns the remainder when the first operand is divided by the second operand. For example, `9 mod 5` evaluates to `4`.

Addition (+)

The plus sign (+) is the addition operator. For example, `2 + 4` evaluates to `6`. If one (or both) of the operands is of floating-point type, the result is of floating-point type. If both operands are integers, the result is of integer type.

Addition can also be used with 2 point operands or 2 vector operands. Adding two point values (for example, `(10,10) + (15,10)`) sums the corresponding x and y fields of each point and returns the resulting point (`(25,20)` in our example). The operation $point_1$ + $point_2$ is equivalent to the expression:

($point_1$.x+$point_2$.x, $point_1$.y+$point_2$.y)

Adding two vector values (such as `(10°2.0) + (20°1.0)`) performs a vector addition on the two values (resulting in `(13.3296°2.98985)`) in our example.

Subtraction and Negation (–)

The minus sign (–) is the subtraction and negation operator. For example, `4 - 2` evaluates to `2`. The statement `set c to -c` changes the sign of the value in variable `c`.

Subtraction can also be used with 2 point operands of 2 vector operands. Subtracting two point values (such as `(10,10) – (15,10)`) subtracts the corresponding x and y fields of each point and returns the resulting point (`(–5,0)` in our example). The operation *point1 – point2* is equivalent to the expression:

(*point1*.x–*point2*.x, *point1*.y–*point2*.y)

Subtracting two vector values (such as `(10°2.0) - (20°1.0)`) performs a vector subtraction on the two values (resulting in `(0.293519°,1.02994°)`) in our example.

String Concatenation (&)

The string concatenation operator (&) can be used to combine the contents of strings. This operator accepts two string operands and returns a single string that consists of the first operand, followed by the second operand. For example, the expression:

```
"mTropolis " & "rules"
```

returns the single string:

```
"mTropolis rules"
```

Boolean operators

Boolean operators return either true or false boolean values. Different variable types have different "true" and "false" states — see "Definition of True" in this chapter for more information.

And

The `and` operator returns true when both of its operands are true. For example, if Boolean Variable `a` contains true and Boolean Variable `b` contains true, the expression `a and b` evaluates to true.

Not

The `not` operator is the boolean inverse operator. It inverts the boolean value of its single operand. That is, the expression `not true` evaluates to false and the expression `not false` evaluates to true.

Or

The `or` operator returns true when either of its operands is true. For example, the expression `0 or 1` evaluates to true.

Relational operators

Relational operators can be used to test the relationship between two values. They return true or false based upon the relationship of their operands.

Equal To (=)

The equal sign (=) is the relational "equal to" operator. It returns true if both its operands are equal, otherwise it returns false. For example, `2 = 4` returns false.

Greater than (>)

The greater than operator (>) returns true if the first operand has a greater value than the second operand. For example, `4 > 2` returns true.

Greater than or equal to (>= or ≥)

The "greater than or equal to" operator (>= or ≥, which can be typed as Option-period on Mac OS) returns true if the first operand is greater than or equal to the second operand. For example, both `4 ≥ 3` and `3 ≥ 3` return true.

Less than (<)

The less than operator (<) returns true if the first operand has a lesser value than the second operand. For example, `2 < 4` returns true.

Less than or equal to (<= or ≤)

The "less than or equal to" operator (<= or ≤, which can be typed as Option-comma on Mac OS) returns true if the first operand is less than or equal to the second operand. For example, both `2 ≤ 3` and `3 ≤ 3` return true.

Not equal to (<> or ≠)
The "not equal to" operator (<> or ≠, which can be typed as Option-= on Mac OS) returns true if its operands are not equal. For example, `3 ≠ 5` returns true.

Object reference expressions

Though it's not an operator *per se*, the "dot" character syntax is used to build object reference expressions.

What are object reference expressions?
Any component in a mTropolis project is called an "object." The way in which we refer to objects in Miniscript statements is by writing object reference expressions. Object reference expressions consist of the names of objects, their attributes, and keywords separated by the dot character (also known as a period, ".").

For example, the send statement sends a message to an object. It has the general form:

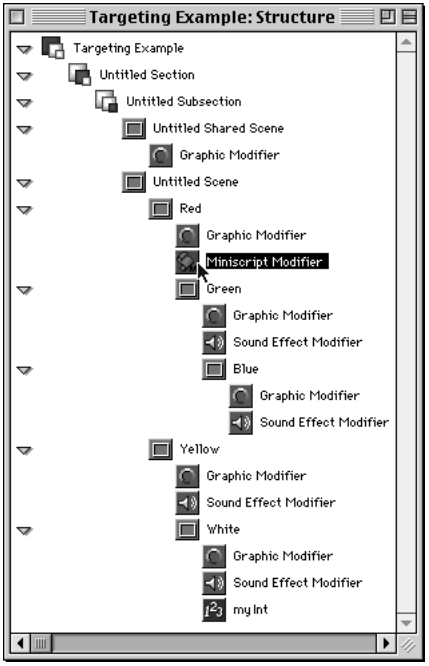
```
send "message" to destination
```

Where *destination* is an object reference expression. This object is defined relative to the location of the Miniscript modifier that contains this statement. We can use building blocks (discussed in this chapter) and targeting attributes (discussed in Chapter 15) to define the destination object. Suppose we wanted to send a message named "myMessage" to the Miniscript modifier's parent object (which might be an element or a behavior). The send statement becomes:

```
send "myMessage" to parent
```

Scoping: What objects can a Miniscript modifier see?
Because object reference expressions in a Miniscript are defined relative to the location of the script itself, it is important to understand which objects can be targeted simply by name and which objects need more precise definition.

Consider the Miniscript modifier on the element named "Red" in the figure below:



Example structure view. The element named "Red" contains a Miniscript Modifier (highlighted in this figure). Statements within that script can use object reference expressions to target various objects in the project

- The Miniscript modifier can refer to its parent element by name (`Red`) or with the keyword `element`. Recall that this is also the default destination for the send statement.

For example, the following three statements are equivalent:

```
send "myMessage" to Red
send "myMessage" to element
send "myMessage"
```

- The Miniscript modifier can refer to the immediate children of its element by name. In this example, the script can refer to the element `Green`, which is an immediate child of `Red`. For example:

  ```
  send "myMessage" to Green
  ```

- The Miniscript modifier can refer to siblings of its element by name. In this example, the script can refer to the element Yellow, which is a sibling of Red. For example:

  ```
  send "myMessage" to Yellow
  ```

- The Miniscript modifier can refer to any object above itself in the structural hierarchy by name or by the keywords shown in the building blocks table earlier in this chapter. For example, the script could send a message to the scene itself by either of the following statements:

  ```
  send "myMessage" to scene
    -- send by keyword
  send "myMessage" to 'Untitled Scene'
    -- send by name
  ```

  Note that the name of the scene was enclosed in single quotes in the second statement because the name contains a space. Names of objects that contain delimiter characters such as space and dot must be enclosed in single quotes to avoid ambiguity.

  Similarly, the Miniscript modifier contained by `Red` could send a *Close Project* command directly to the project component using the statement:

  ```
  send "Close Project" to project
  ```

- Objects in other locations must be defined more precisely. Suppose we want to send a message directly to element `White`, which is a child of element `Yellow`. A message sent to `Yellow` with the default message options will eventually "cascade" to element `White`, but a command such as **Hide** will not (because commands are acted on only by the targeted element and do not cascade). Because the script can target element `Yellow` directly, we can use the dot syntax to travel down the structural hierarchy to element `White`. For example, the following statement could be used to send the **Hide** command to `White`:

  ```
  send "Hide" to Yellow.White
  ```

  Similarly, we could send a message directly to `Blue`, which is a child of element `Green` (which is an immediate child of `Red`) using the statement:

  ```
  send "myMessage" to Green.Blue
  ```

Using object reference expressions to refer to object attributes and variables

Object references are useful not only as the destinations for messages and commands. They can also be used to access the attributes of mTropolis objects or variable modifiers contained by objects. To access an attribute of an object, use the general syntax:

```
object.attribute
```

anywhere that you would use a value with the same type as the value contained by that attribute. O*bject* is an object reference to the

desired object and *attribute* is the name of the desired attribute. This expression will return the value contained in *attribute*. Individual attributes are described in more detail in Chapter 15, "Attributes."

Some examples follow. Consider the Miniscript modifier on the element named "Red" in the figure above:

- All graphic elements have a "width" attribute that contains an integer value that represents their current width in pixels. This attribute can be read or written. Changing the value of an element's width attribute causes the width of the element to change. To make the width of element `Red` the same as the width of element `Yellow`, we would execute the statement:

  ```
  set element.width to Yellow.width
  ```

  The expression `element.width` is an object reference expression that returns the width of element `Red`. The element keyword refers to object `Red` because the Miniscript modifier is contained by `Red`. The expression `Yellow.width` is an object reference expression that returns the width of element `Yellow`. `Yellow` can be referred to by name, because it is a sibling of `Red` as described previously.

  Note that this statement could be written more simply by omitting the "element" part because references to attributes or variables are assumed to be attributes or variables of the Miniscript modifier's element unless otherwise specified:

  ```
  set width the Yellow.width
  ```

  We could do the opposite — set the width of element `Yellow` to be the same as element `Red` — with the statement:

  ```
  set Yellow.width to width
  ```

- Suppose we wanted to store the current width of element `Red` in the Integer Variable `myInt` (contained by element `White`). Because of the scoping rules described previously, we cannot refer to `White` by its name alone. However, we can refer to `White` as `Yellow.White`. Therefore, the script can refer to `myInt` as `Yellow.White.myInt` as follows:

  ```
  set Yellow.White.myInt to width
  ```

- Object reference expressions that refer to attributes can be used anywhere that their types are appropriate, even in other types of expressions. For example, suppose we want to compute the area (in pixels) of element `Blue` and store that value in `myInt`.

  The area of Blue can be computed as Blue's width (`Blue.width`) multiplied by Blue's height (`Blue.height`). Because Blue is not an immediate child of Red, we'd have to refer to it as `Green.Blue` (`Green` *is* an immediate child of `Red`). We also know that `myInt` can be referenced as `Yellow.White.myInt` as in the previous example. So the following statement could be used:

  ```
  set Yellow.White.myInt to \
   Green.Blue.width * Green.Blue.height
  ```

  This statement is legal because `Green.Blue.width` returns an integer as does `Green.Blue.height`. Those values are multiplied together using the `*` operator, and the resulting integer value is then stored in `myInt`.

Operator precedence

Operators are evaluated in order of their precedence. Operators with a higher precedence are evaluated before those with a lower precedence. Operators with equal precedence are evaluated from left to right, as they appear in the expression. The following table shows the precedence of mTropolis' operators.

Operator	Description	Precedence
( )	Expression Grouping	1 (first)
not	Boolean NOT	2
^	Exponentiation	3
*	Multiplication	4
/	Division	4
div	Integer Division	4
mod	Integer Modulus	4
+	Addition	5
-	Subtraction or Unary Negation	5
>	Greater Than	6
>= or ≥	Greater Than or Equal To	6
<	Less Than	6
<= or ≤	Less Than or Equal To	6
=	Equal To	7
<> or ≠	Not Equal To	7
and	Boolean AND	8
or	Boolean OR	9 (last)
&	String Concatenation	-

Operator precedence

Special Environment Variables

Two read-only environment variables (`mouse` and `ticks`), and one write-only environment variable (`debug`) can be referenced from within Miniscript.

The Mouse Variable

The `mouse` environment variable holds a point value that contains the current position of the mouse cursor. For example, to move an element's origin point to the current mouse position, use the statement:

```
set element.position to mouse
```

The ticks variable

The `ticks` environment variable holds an integer value that contains the number of time ticks (defined as 1/60 second) that have elapsed since boot time. For example, to obtain the number of seconds that have elapsed since boot time, use the statement:

```
set myFloat to ticks/60
```

The debug variable

Set the `debug` environment variable to an expression (of any data type) to display that data in the Message Log window. The Message Log window will automatically display itself and the debug message upon returning to edit mode. More information about message logging can be found in "Message Log Window" in Chapter 7. For example, to display the string "Hello World" in the Message Log window, execute the script:

```
set debug to "Hello World"
```

The message is visible in the Message Log on returning to edit mode from run-time mode.

This environment variable can also be used to check the values of variables or attributes. For example, to display the contents of the Point Variable myPoint in the Message Log, execute the script:

```
set debug to myPoint
```

Miniscript Functions

Miniscript contains a number of built-in functions for performing mathematical operations, data type conversions, string manipulation, and other common operations. Functions can be used anywhere an expression or value can be used. For example, the `cos` function calculates the cosine of the specified angle:

```
set myFloat to cos(30)
```

Functions can also be nested. For example:

```
set myFloat to cos(exp(rnd(45)))
```

Functions are described in alphabetical order, below. Note that some functions have multiple names that can be used interchangeably. These functions are listed with their names separated by commas. For example, to return the arctangent of an angle, either the function name `atn` or `arctangent` can be used.

abs

The abs function returns the absolute value of its argument. The abs function is useful for tracking element movement. Use it to convert coordinate differences (negative or positive) into distances (always positive).

Parameters

numericExpression integer or float

Returns

float

Example

```
set myDistance to \
   abs(positionB.x – positionA.x)
```

See also

"`sgn`" in this chapter.

ASCIItoChar

The ASCIItoChar function returns a string that contains the character whose ASCII code corresponds to the integer argument.

Parameters

asciiValue integer

Returns

string

Example

```
set myString to ASCIItoChar(107)
```

See also

"CharToASCII" in this chapter.

arctangent, atn

The atn function returns the angle, in degrees, whose tangent is numericExpression. The returned value is an angle between –90 and +90 degrees.

Parameters

numericExpression integer or float

Returns

float

Example

```
set myVector.angle to atn(45)
```

See also

"tangent" in this chapter.

CanGet

The CanGet function returns true if the specified object has the readable attribute whose name is supplied in *attributeString*.

Parameters

objectReference	object
attributeString	string

Returns

boolean

Example

```
if CanGet(myObjRef.object, "width")
then
   set myInt to
myObjRef.object.width
end if
```

See also

"CanSet" below.

CanSet

The CanSet function returns true if the specified object has the writable attribute whose name is supplied in *attributeString*.

Parameters

objectReference	object
attributeString	string

Returns

boolean

Example

```
if CanSet(myObjRef.object, \
   "width") then
   set myObjRef.object.width to 200
end if
```

See also

"CanGet" above.

Character

The character function returns a substring of characters within a string. If the *indexOrRange* parameter is an integer, this function returns the single character at that index. If the *indexOrRange* parameter is an integer range, this function returns the range of characters from the starting index to the ending index.

Indices start at 1 (one), and negative indices count from the end of the string.

Parameters

i*ndexOrRange*	integer
range string	string

Returns

string

Example

Suppose that the variable `string` contains `"the quick brown fox"`:

```
mytext = Character(1, string)
-- returns "t"
mytext = Character(2 thru 6, string)
-- returns "he qu"
mytext = Character(-1 thru -7, string)
-- returns "own fox"
```

See also

"Paragraph" and "Word" in this chapter.

CharToASCII

The CharToASCII function returns the ASCII value (an integer) of the first character in *string*.

Parameters

Character	string

Returns
integer

Example

```
set charcode to CharToASCII("k")
-- returns 107
```

See also
"ASCIItoChar" in this chapter.

Chunk

The Chunk function returns a substring within *string,* delimited by the characters specified by any of the characters in the *delimiters* string.

The *indexOrRange* parameter defines which chunk is returned. If *indexOrRange* is an integer, the chunk at that index is returned. If *indexOrRange* is an integer range, that range of chunks is returned. If the specified index does not contain any chunks, an empty string is returned. Consecutive runs of delimiters are not ignored, so an empty chunk will be returned as an empty string.

Indices start at 1 (one), and negative indices count from the end of the string.

Parameters

indexOrRange	integer or range
string	string
delimiters	string

Returns
string

Example
Suppose that the variable `string` contains `"the quick brown fox"`:

```
set a to Chunk(1 thru 2, string, "ei")
-- returns "the qu"
set a to Chunk(1, string, "ei")
-- returns "th"
```

See also
"ChunkCount" and "ChunkRange" below.

ChunkCount

The ChunkCount function returns the number of substrings (an integer) within *string* that are delimited by any of the characters in *delimiters*.

Parameters

string	string
delimiters	string

Returns
integer

Example
Suppose that the variable `string` contains `"the quick brown fox"`:

```
set count to ChunkCount(mystring, "f")
-- returns 2
```

See also
"Chunk" and "ChunkRange" in this chapter.

ChunkRange

The chunkRange function returns an integer range that describes the range of characters that make up the substring defined by *indexOrRange* and *delimiters*.

If the specified index does not contain any chunks, a range of (0 thru 0) is returned. Consecutive runs of delimiters are not ignored.

Indices start at 1 (one), and negative indices count from the end of the string.

Parameters

indexOrRange	integer or range
string	string
delimeters	string

Returns
string

Example
Suppose that the variable `string` contains `"the quick brown fox"`:

```
set a to ChunkRange(1, string, "q")
-- returns (1 thru 4)
set a to ChunkRange(1 thru 3,\
 string, " ")
-- returns (1 thru 15)
```

See also
"Chunk" and "ChunkCount" above.

CompareStrings

The CompareStrings function returns 0 if *string1* is identical to *string2*, -1 if *string1* alphabetically precedes *string2*, or 1 if *string1* alphabetically follows *string2*.

Parameters

string1	string
string2	string
[caseSensitive]	boolean

Returns
boolean

Cosh

The cosh function returns the hyperbolic cosine of numericExpression, which represents an angle measured in degrees. The return value is a floating-point number.

Parameters

numericExpression	integer or float

Returns
float

Example

```
set myFloat to cosh(2.2222)
```

See also
"cosine," "sinh," and "tanh" in this chapter.

cosine, cos

The cos function returns the cosine of numericExpression, which represents an angle measured in degrees. The return value is a floating-point number.

Parameters

numericExpression	integer or float

Returns
float

Example
set myvector.angle to cos(45)

See also
"sine" and "tangent" in this chapter.

DeleteString

The DeleteString function removes a range of characters from a string and returns the result. If the *substringRange* parameter is an integer, this function deletes the single character at that index. If the *substringRange* parameter is an integer range, this function deletes the range of characters from the starting index to the ending index.

Indices start at 1 (one), and negative indices count from the end of the string.

Parameters

substringRange	range
string	string

Returns
float

Example

```
set a to DeleteString(4 thru 9, \
 "the quick brown fox")
-- returns "the brown fox"
```

See also
InsertString in this chapter.

EqualStrings

The EqualStrings function returns true if *string1* is equal to *string2*. Otherwise, the function returns false. The optional *caseSensitive* parameter is a boolean value that, if set to true, causes the comparison to be case sensitive. By default, *caseSensitive* is false and the comparison is not case sensitive.

Parameters

string1,	string
string2,	string
[caseSensitive]	boolean

Returns
boolean

Example

```
if EqualStrings(element.text, "Keith,"
\  true) then
   send "Hey cool name" to scene
end if
```

See also
"CompareStrings" in this chapter.

exp

The exp function returns the value of e raised to the power of numericExpression, where e is the natural number 2.71828 (the "natural" exponential function). The return value is a floating-point number.

Many natural processes involve quantities that increase or decrease at a rate proportional to their size. For example, a culture of bacteria will grow at a rate proportional to its mass. The value of an investment bearing interest that is continually compounded will increase at a rate proportional to that value.

Parameters

numericExpression	integer or float

Returns
boolean

Example
set myNumber to exp(999.999)

See also
"ln" in this chapter.

FindStringCount

The FindStringCount function returns the number of occurrences of *stringToFind* in *stringToSeach*. The optional *caseSensitive* parameter is a boolean value that, if set to true, causes the search to be case sensitive. By default, *caseSensitive* is false and the search is not case sensitive.

Parameters

stringToFind	string
stringToSearch	string
[caseSensitive]	boolean

Returns

integer

Example

Suppose that the variable `string` contains `"Eels sleep in the deep."`:

```
set a to FindStringCount("eep," string)
-- returns 2
```

See also

"FindStringRange" below.

FindStringRange

The FindStringRange function returns the range of characters that make up the first occurrences of *stringToFind* in *stringToSearch*. If string1 is not found, the range (0 thru 0) is returned.

The optional *caseSensitive* parameter is a boolean value that, if set to true, causes the search to be case sensitive. By default, *caseSensitive* is false and the search is not case sensitive.

If the optional *index* parameter is present, the function returns the range of the "indexth" occurrence of *stringToFind* in *stringToSearch*. If the *index* value is greater than the number of occurrences, (0 thru 0) is returned. Negative indices count from the end of the string.

Parameters

stringToFind	string
stringToSearch	string
[*caseSensitive*]	boolean
[*index*]	integer

Returns

range

Example

Suppose that the variable `string` contains `"Eels sleep in the deep."`:

```
set a to FindStringRange("eep," string,\
  true, 2)
-- returns (20 thru 22)
```

See also

"FindStringCount" above.

InsertString

The InsertString function inserts *stringToInsert* into *destinationString* before the character at *index* (an integer) and returns the result. If *index* is greater than the length of *destinationString*, *stringToInsert* is appended to the end of *destinationString*. Negative indices count from the end of *destinationString*.

Parameters

stringToInsert	string
destinationString	string
index	integer

Returns
string

Example

```
set text to InsertString("quick ," \
 "the brown fox," 4)
```

See also
"ReplaceString" in this chapter.

ln

The ln function returns the natural logarithm, log to the base e, of *numericExpression* where e is the natural number 2.71828. The return value is a floating-point number.

Parameters

numericExpression	nteger or float

Returns
float

Example

```
set myFloat to ln(3.1111)
```

See Also
"exp" and "log" in this chapter.

log

The log function returns the logarithm to the base 10 of *numericExpression*. The return value is a floating-point number.

This function is useful in calculating the pH of solutions, in heat conduction as well as for financial calculations.

Parameters

numericExpression	intrger or float

Returns
float

Example

```
set myFloat to log(100000)
```

See also
"ln" above.

Lowercase

The Lowercase function converts the contents of *string* into all lowercase characters and returns the result.

Parameters

string	string

Returns
string

Example

```
set mystring to Lowercase(mystring)
```

See Also
"Uppercase" in this chapter.

NumToString, num2str

The num2str function converts a numeric value to its string equivalent. Note that, in many cases, this function is not necessary as most mTropolis data types have string equivalents as described in "String Equivalents of Data Types" in this chapter.

Parameters

numericExpression	integer or float

Returns
string

Example

```
set mystring to num2str(2.67)
```

See also
"StringToNum, str2num" in this chapter.

Paragraph

The Paragraph function returns an entire paragraph within a string. If the *indexOrRange* parameter is an integer, this function returns the "indexth" paragraph in the string. If the *indexOrRange* parameter is an integer range, this function returns the range of paragraphs from the starting index to the ending index.

Indices start at 1 (one), and negative indices count from the end of the string.

Paragraphs are delimited by whichever characters (return or newline characters) are appropriate for the current platform. Consecutive runs of delimiters are not ignored, so a blank line (that is, an empty "paragraph") will be returned as an empty string.

Parameters

indexOrRange	integer or range
string	string

Returns
string

Example

```
-- return the first paragraph in
-- longString:
set myString to Paragraph(1, \
  longString)

-- return the second and third
-- paragraphs in longString:
set mystring to Paragraph(2 thru 3, \
  longString)
```

See also
"Character" and "Word" in this chapter.

ParagraphCount

The ParagraphCount function returns the number of paragraphs in *string*. Paragraphs are delimited by whichever characters (return or newline characters) are appropriate for the current platform. Consecutive runs of delimiters are not ignored, so blank lines (empty "paragraphs") will be counted.

Parameters

string	string

Returns
integer

Example

```
set count to \
   ParagraphCount(longString)
```

See also
"StringLength" and "WordCount" in this chapter.

ParagraphRange

The ParagraphRange function returns an integer range that describes the range of characters that make up the paragraph(s) defined by *indexOrRange*.

The *indexOrRange* parameter defines which paragraph range is returned. If *indexOrRange* is an integer, the range of characters describing the paragraph at that index is returned. If *indexOrRange* is an integer range, the range of characters that makes up that range of paragraphs, is returned. If the specified index does not contain any paragraphs, a range of (0 thru 0) is returned. Paragraphs are delimited by whichever characters (return or newline characters) are appropriate for the current platform. Consecutive runs of delimiters are not ignored.

Indices start at 1 (one), and negative indices count from the end of the string.

Parameters

indexOrRange	integer or range
string	string

Returns
range

Example

```
-- Return the range of characters for
-- paragraph 1:
set myRange to ParagraphRange(1, \
   longstring)

-- Return the range of characters for
-- paragraphs 3 thru 5:
set myRange to ParagraphRange \
  (3 thru 5, longstring)
```

See also
"ChunkRange" and "WordRange" in this chapter.

polar2rect

The polar2rect function converts polar coordinates (angle, radius) to rectangular coordinates (x,y). This function accepts an argument of vector type. The return value is of point type.

Parameters

vector	vector

Returns
point

Example

```
set myPoint to polar2rect(myVector)
```

See also
"rect2polar" below.

random, rnd

The rnd function returns a random value from 0 to the value specified by *limit*-1. The return value is an integer. This function is useful for randomizing messages in game play or the positions of elements in a scene.

Parameters

limit	integer

Returns
integer

Example

```
set mynumber to rnd(100)
```

rect2polar
The rect2polar function converts rectangular coordinates (x,y) to polar coordinates (angle, radius). This function accepts an argument of point type. The return value is of vector type (angle, magnitude).

Parameters
x, y point point

Returns
vector

Example
```
set myVector to rect2polar(myPoint)
```

See also
"polar2rect" above.

ReplaceString
The ReplaceString function replaces the characters in *destinationString* that correspond to the integer range *substringRange* with the characters in *replacementString* and returns the result.

Parameters
substringRange range
destinationString string
replacementString string

Returns
string

Example
Execute the following Miniscript statements as necessary:

```
set substringRange to \
 FindStringRange("Director," text)
set text to \
 ReplaceString(substringRange, \
 text, "mTropolis")
```

See also
"FindStringRange" in this chapter.

round
The round function rounds the floating-point value *numericExpression* to the nearest whole number. The return value is of integer type.

Parameters
numericExpression float

Returns
integer

Example
```
set myInteger to round(2.71828)
```

See also
"trunc" in this chapter.

sgn
The sgn function returns the sign of *numericExpression*. If the value of *numericExpression* is less than 0, the return value is –1. If the value of *numericExpression* is greater than 0, the return value is 1. If *numericExpression* equals 0, the return value is 0.

Parameters
numericExpression integer or float

Returns
float

Example
```
set mynumber to sgn(tan(rnd(26)))
```

See also
"abs" in this chapter.

sine, sin
The sin function returns the sine of *numericExpression,* which represents an angle measured in degrees. The return value is a floating-point number.

Parameters
numericExpression integer or float

Returns
float

Example
```
set myvector.angle to sin(45)
```

See also
"cosine" and "tangent" in this chapter.

sinh
The sinh function returns the hyperbolic sine of *numericExpression,* which represents an angle measured in degrees. The return value is a floating-point number.

Parameters
numericExpression integer or float

Returns
float

Example
```
set myFloat to sinh(3.2222)
```

See also
"cosh," "sine," and "tanh" in this chapter.

sqrt
The sqrt function returns the square root of *numericExpression.* The return value is a floating-point number.

Parameters
numericExpression integer or float

Returns
float

Example
```
set mynumber to sqrt(999.999n
```

StringToNum, str2num
The str2num function converts a string value to a floating-point value.

Parameters
string string

Returns
integer or float

Example
```
set mynumber to str2num("1.5")
```

See also
"NumToString, num2str" in this chapter.

StringLength
The StringLength function returns the number of characters in *string*.

Parameters
string string

Returns
integer

Example
```
if StringLength(password) < 5 then
  send "Need Longer Password"
end if
```

See also
"ChunkCount," "ParagraphCount," and "WordCount" in this chapter.

tangent, tan
The tan function returns the tangent of *numericExpression,* which represents an angle measured in degrees. The return value is a floating-point number.

Parameters
numericExpression integer or float

Returns
float

Example
```
set myvector.angle to tan(45)
```

See also
"cosine", "sine", and "tanh" in this chapter.

tanh
The tanh function returns the hyperbolic tangent of *numericExpression,* which represents an angle measured in degrees. The return value is a floating-point number.

Parameters
numericExpression integer or float

Returns
float

Example
```
set myFloat to tanh(3.2222)
```

See also
"cosh," "sinh," and "tangent" in this chapter.

trunc
The trunc function truncates the floating-point value *numericExpression;* that is, any decimal portion is removed. The return value is of integer type.

Parameters
numericExpression float

Returns
integer

Example
```
set myInteger to trunc(2.71828)
```

See also
"round" in this chapter.

Uppercase

The Uppercase function converts all the characters in *string* to uppercase and returns the result.

Parameters

string	string

Returns
integer

Example

```
set myString to Uppercase(myString)
```

See also
"Lowercase" in this chapter.

Word

The Word function returns a word or range of words within *string*. Words are delimited by space, tab, or newline characters.

The *indexOrRange* parameter defines which words are returned. If *indexOrRange* is an integer, the word at that index is returned. If *indexOrRange* is an integer range, that range of words is returned. If the specified index does not contain any words, an empty string is returned. Consecutive runs of delimiters are ignored.

Indices start at 1 (one), and negative indices count from the end of the string.

Parameters

indexOrRange	integer or range
string	string

Returns
string

Example
Suppose that the variable `string` contains `"the quick brown fox"`:

```
set text to Word(3, string)
-- returns "brown"
set text to Word(2 thru 3, string)
-- returns "quick brown"
```

See also
"Character," "Chunk," and "Paragraph" in this chapter.

WordCount

The WordCount function returns the number of words in *string*. Words are delimited by space, tab, or newline characters. Consecutive runs of delimiters are ignored.

Parameters

string	string

Returns
integer

Example

```
set text to "There are" & \
 num2str(WordCount(myNovel)) & \
 "words in my latest book."
```

See also
"ChunkCount," "ParagraphCount," and "StringLength" in this chapter.

WordRange
The WordRange function returns an integer range that describes the range of characters that make up the word defined by *indexOrRange*.

The *indexOrRange* parameter defines which word range is returned. If *indexOrRange* is an integer, the range of characters describing the word at that index is returned. If *indexOrRange* is an integer range, the range of characters that makes up that range of words, excluding trailing delimiters, is returned. If the specified index does not contain any words, a range of (0 thru 0) is returned. Words are delimited by space, tab, or newline characters. Consecutive runs of delimiters are ignored.

Indices start at 1 (one), and negative indices count from the end of the string.

Parameters

indexOrRange	integer or range
string	string

Returns
range

Example
Suppose that the variable `string` contains `"the quick brown fox"`:

```
set a to wordRange(1, string)
-- returns (1 thru 3)
set a to wordRange(2 thru 3, string)
-- returns (5 thru 15)
```

See also
"ChunkRange" and "ParagraphRange" in this chapter.

Miniscript Errors

Various errors may occur during the execution of the script contained by a Miniscript modifier. The Message Log window displays error messages generated by Miniscript. Even if logging is not enabled, this window is automatically displayed when an error occurs during run-time. The text of the error message is displayed in the Message Log window.

Note: When an error occurs that causes an error message to be generated, execution of the script stops at the statement that caused the error. Statements after the offending statement are not executed.

For more information on individual error messages, see "Error Messages" in Chapter 7.

Reserved Words

All Miniscript statement keywords, function names, operators (described in the preceding sections of this chapter), and attribute names (described in Chapter 15, "Attributes") are "reserved" words. These words should not be used to name project components or variables. While it is possible to use these reserved words outside of Miniscript to name project components, it is strongly recommended you do not do so in order to avoid confusion and possible syntax or logic errors when writing scripts.

Miniscript Syntax

Control Statement	Description
set *variable* to *expression*	Assignment of a value to a variable
set *attribute* to *expression*	Setting an object attribute
send "*message*" [with *data*] [to *target*]	Sending a message (element is the default target)
send "*message*" [with *data*] [to *target*] opt[ions] [i[mmediate]] [c[ascade]] [r[elay]]	Sending a message with specific message options (element is the default target)
if *expression* then *statement*	Single-case, one-line conditional statement

if *expression* then
  statements
end if
	Single-case, multiple-line conditional statement

if *expression* then
  statements
else
  statements
end if
	Two-way conditional branch

if *expression* then
  statements
[else if *expression* then
  statements]
[...]
[else
  statements]
end if
	Multiple-case conditional branch

-- *comment*	Comment (do not execute) to end of line

Data Type	Syntax	Example
boolean	true or false	true
integer	*n*	578
string	"*text*"	"mFactory"
integer range	(*n* thru *n*)	(5 thru 75)
floating-point	*n.n*	109.2756
vector†	(*n*°*n.n*)	(90°2.5)
point	(*n*,*n*)	(640,480)
object reference	*object*[.*object*[...]]	myScene.myElement.myMod
list	{*n*, *n*, ..., *n*}	{"dog", "cat", "bird"}

† The degrees symbol can be typed by pressing Shift-Option-8.

Miniscript Syntax

Operator	Description	Precedence
( )	Expression Grouping	1 (first)
not	Boolean NOT	2
^	Exponentiation	3
*	Multiplication	4
/	Division	4
div	Integer Division	4
mod	Integer Modulus	4
+	Addition	5
-	Subtraction or Unary Negation	5
>	Greater Than	6
>= or ≥	Greater Than or Equal To	6
<	Less Than	6
<= or ≤	Less Than or Equal To	6
=	Equal To	7
<> or ≠	Not Equal To	7
and	Boolean AND	8
or	Boolean OR	9 (last)
&	String Concatenation	-

Target†	Description
project	Project containing the Miniscript modifier
section	Section containing the Miniscript modifier
subsection	Subsection containing the Miniscript modifier
scene	Scene containing the Miniscript modifier
activeScene	Currently active scene (the scene the user currently sees; not necessarily the Scene containing the Miniscript modifier)
sharedScene	Currently active shared scene
element	Element containing the Miniscript modifier (this is the default target; it can be omitted when accessing element attributes or sending messages to the element)
parent	Immediate parent of the Miniscript modifier
element.parent	Parent of the element containing the Miniscript modifier
source	Modifier that sent the message that triggered the Miniscript modifier
source.parent	Immediate parent of the modifier that sent the message that triggered the Miniscript modifier
source.element	Element containing the modifier that sent the message that triggered the Miniscript modifier
this	The Miniscript modifier itself

† See the "Structure Attribute" table in Chapter 15 for additional object targeting options.

Miniscript Syntax

Environment Variable	Type	Get	Set
debug	Any		•
mouse	Point	•	
ticks	Integer	•	

Miniscript Functions

Math Function	Parameters	Returns
abs	*numericExpression*	Float
atn	*numericExpression*	Float
cosh	*numericExpression*	Float
cos	*numericExpression*	Float
exp	*numericExpression*	Float
ln	*numericExpression*	Float
log	*numericExpression*	Float
random	*limit*	Integer
round	*numericExpression*	Integer
sgn	*numericExpression*	Float
sin	*numericExpression*	Float
sinh	*numericExpression*	Float
sqrt	*numericExpression*	Float
tan	*numericExpression*	Float
tanh	*numericExpression*	Float
trunc	*numericExpression*	Integer

Data Conversion Function	Parameters	Returns
AsciiToChar	*asciiValue*	String
CharToAscii	*character*	Integer
NumToString	*numericExpression*	String
StringToNum	*string*	Integer or Float
Lowercase	*string*	String
Uppercase	*string*	String
polar2rect	*vector*	Point
rect2polar	*xyPoint*	Vector

Miniscript Functions

String Chunking Function	Parameters	Returns
Character	*indexOrRange, string*	String
Chunk	*indexOrRange, string, delimiters*	String
ChunkCount	*string, delimiters*	Integer
ChunkRange	*indexOrRange, string, delimiters*	String
Paragraph	*indexOrRange, string*	String
ParagraphCount	*string*	Integer
ParagraphRange	*indexOrRange, string*	Range
StringLength	*string*	Integer
Word	*indexOrRange, string*	String
WordCount	*string*	Integer
WordRange	*indexOrRange, string*	Range

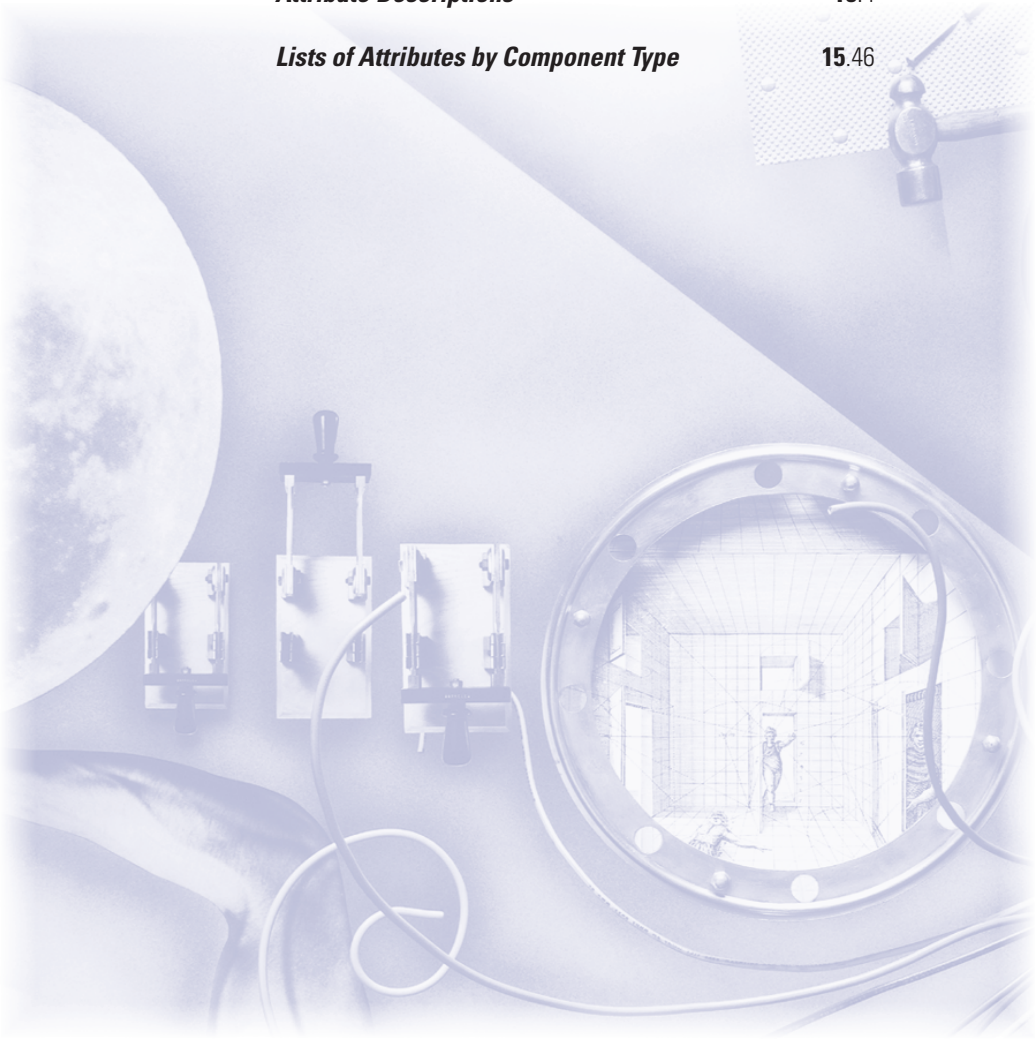
String Editing Function	Parameters	Returns
DeleteString	*substringRange, string*	String
InsertString	*stringToInsert, destinationString, index*	String
ReplaceString	*substringRange, destinationString, replacementString*	String

String Searching Function	Parameters	Returns
FindStringCount	*stringToFind, stringToSearch, [caseSensitive]*	Integer
FindStringRange	*stringToFind, stringToSearch, [caseSensitive], [index]*	Range

String Comparison Function	Parameters	Returns
CompareStrings	*string1, string2, [caseSensitive]*	Boolean
EqualString	*string1, string2, [caseSensitive]*	Boolean

Miscellaneous Function	Parameters	Returns
CanGet	*objectReference, attributeString*	Boolean
CanSet	*objectReference, attributeString*	Boolean

15. Attributes



15. Attributes

Most components of a mTropolis project have attributes whose values can be retrieved and/or modified using Miniscript. Attributes contain values that describe the current state of a mTropolis component. For example, graphic elements have width, height, and position attributes (among others).

Some attributes are “gettable” and can be used in Miniscript expressions. Others are “settable” and can be used as targets for Miniscript set statements. Changing the value of a settable attribute causes an immediate change in the component whose attribute was modified. For example, changing a graphic element’s width attribute causes the width of the element to change to the newly-specified size.

Many of the basic element attributes can also be accessed through the **Message/Command** menu (see “Get and Set Attribute Option Descriptions” in Chapter 13).

Attributes can be referenced using a syntax similar to that used to reference the fields of Compound Variables (see “Accessing the Fields of Compound Variables” in Chapter 14).

Attributes

This chapter describes individual attributes and the syntax used to retrieve and set the value of attributes using Miniscript.

Component attribute syntax

In general, an attribute of any mTropolis component can be specified by adding a period, followed by the attribute's name, to the component's name. For example, the current width of the element named myPict could be set to the value contained in myInt using the statement:

```
set myPict.width to myInt
```

Element attribute syntax

A slightly different syntax can be used to refer to the attributes of the element on which the Miniscript modifier resides. Instead of using the element's name, the element can be referred to as element. For example:

```
set element.width to myInt
```

Alternatively, the "`element.`" part can be omitted because the element on which the Miniscript modifier resides is the default "target" for both messages and attribute references. For example, the following statement is equivalent to the previous one:

```
set width to myInt
```

There is one special exception to this syntax. The "parent" attribute, when specified without the "element." prefix, specifies the Miniscript modifier's parent (the element or behavior that contains that script).

!!! Note also that the relative "building blocks" for specifying targets such as `next`, `previous`, and `parent` can be used to specify components whose attributes are to be read or set. See Chapter 14.

Special components with attributes

In addition to the various media elements and structural components that have attributes, three special components — the AssetManager, WorldManager, and System components, which are not visible in any of the mTropolis editing views — also have attributes:

- The AssetManager is the component that controls media assets. Its attributes describe media assets that have been linked to the project.
- The WorldManager is the component that controls the execution of a project. Its attributes control global project options.
- The System is the component that controls the execution of all open mTropolis projects. Its attributes are mostly hardware related. For example, the `system.ejectCD` attribute can be used to open the computer's CD–ROM drive.

To address its attributes, the System component must always be specified by name. The other two components can be accessed as sub-components of the project, such as `project.WorldManager.currentScene`. Usually, however, these components are accessed by name. These components have no "relative" relationship to any other component in a mTropolis project.

For example, the following Miniscript statement sets the value of the WorldManager's "refreshCursor" attribute:

```
set WorldManager.refreshCursor to true
```

!!! Note that WorldManager may be abbreviated to "wm", and AssetManager may be abbreviated to "am", for easier use in Miniscript.

Attribute Descriptions

Individual attributes are described in alphabetical order, below. For attributes arranged by the type of mTropolis component they belong to, see the tables at the end of this chapter.

In addition to a description, the attribute's component, data type and whether the attribute is gettable or settable is noted. Gettable attributes can be used in Miniscript expressions. For these attributes, using the syntax *component.attribute* in a script returns the current value of attribute (which has the data type listed for the attribute) for the specified component. Settable attributes can be changed by using the `set` statement (for example, `set myToon.paused to true`). Changing the value of an attribute causes that property of the component to change instantly.

In this chapter, some of the attribute names are shown in mixed case for readability (such as autoScreenFade) but remember that Miniscript is not a case-sensitive language. Attribute names can be typed in mixed, lower, or uppercase in Miniscript.

---

abbreviatedDate
System
String
Get

This System attribute contains a string that represents the current date in the form "Wed, Oct 8, 1997."

See also "date" and "longDate" in this chapter.

---

activeProject
System
Object *Editor only*
Get

This System attribute contains an object reference to the currently-active project. Usually, this is the project that corresponds to the frontmost non-floating window.

This attribute is especially useful in the creation of tools. A tool can use this attribute to distinguish between itself and another project being manipulated in edit mode. It cannot be used with projects running in any of the stand-alone mTropolis players.

---

allowQuitKey
Project
Boolean
Set

Set this project attribute to true (the default) to allow ⌘-Q (on Mac OS) or Alt-F4 and Alt-F-X (on Windows) to be used to close the title. Set this attribute to false to disable these **Quit** shortcuts.

The **Close Project** command can also be used to close mTropolis titles. See "Close Project (Message/Command Menu Only)" in Chapter 13.

altKeyDown
System
Boolean
Get

This System attribute contains true when the Option key (on Mac OS) or Alt key (on Windows) is being held down. This attribute is identical to the attribute "optionKeyDown" in this chapter.

See also "controlKeyDown" and "optionKeyDown" in this chapter.

append
Text Elements
String
Set

Set this text element attribute to a string to cause that string to be appended (added to the end of) the current contents of the text element.

Note that, while the same operation can be performed using the text attribute and the concatenation operator (&), for example:

```
set text to text & "appended text"
```

using the append attribute is more efficient, for example:

```
set append to "appended text"
```

asset
All Elements
Object/String
Get Set

When read, this attribute returns an object reference to the media asset that is currently linked to the element. Changing the value of this attribute changes the media contained in the element. When setting this attribute, either an object reference to a new asset or a string that contains the name of the new media asset as it appears in the **Asset** palette can be used.

For example, the following Miniscript statement might be used to set this attribute with a string:

```
set myelement.asset to "mymovie.moov"
```

The attribute could also be set with an object reference. For example, the following statement sets the asset of element "myelement" to be the same as the asset used in the element named "otherelement":

```
set myelement.asset to \
  otherelement.asset
```

The AssetManager component also holds object references to linked media. The following statement sets the element's asset to the second media asset in the AssetManager:

```
set myelement.asset to \
  AssetManager.asset[2]
```

Note that the new asset must be of the correct media type for the element (for instance, you cannot change the type of the element by assigning it a new asset of a different type). When using a string to set this attribute, keep

in mind that an asset name is not a file name — the media must have been linked to the current project before it can be used with the asset attribute.

!!! Note: For assets to be dynamically changed in a built title, they must actually be used in a scene (so the build title process knows to export the assets to the finished title file). Simply create a scene that is never actually shown as a placeholder for such assets.

Using the asset attribute to dynamically link media
In a project designed to run in edit mode (such as a tool or wizard — see Chapter 6, "Tools Menu") the asset attribute can be used to link media to an element in a project. To use the asset attribute this way, set asset to a string that represents a fully-qualified file path (in standard Mac OS `drive:folder:file` syntax) to a media file. For example, the statement:

```
set myelement.asset to \
"My HD:My Media Folder:Image.pict"
```

would cause the PICT file "Image.pict" located in the "My Media Folder" folder of the disk "My HD" to be linked to the element "myelement." If the path cannot be correctly resolved, the mTropolis **Link File** dialog box is is displayed. More information on specifying filepaths in this way can be found in the descriptions of the Open Application modifier and the Save and Restore modifier in Chapter 12.

To display the standard mTropolis **Link File** dialog box so a new asset can be selected by the end user, set the asset attribute to a null string. For example:

```
set myelement.asset to ""
```

As with changing the asset attribute during run-time, the new asset must be of the correct media type for the element. You cannot change the type of element by linking a different media type using the asset attribute. Therefore, for this feature to work properly, the element must already be linked to a media file of the appropriate type (for example, link the element to a QuickTime movie if you want to dynamically link other QuickTime movies to it).

!!! Note: Using the asset attribute to dynamically link media in this way works only in "unbuilt" projects (that is, a mTropolis project file, not a built title file created with the **File ➝ Build Title** menu option). As mentioned above, this feature is intended for the creation of tools and wizards that help build other mTropolis projects.

---

asset[*n*]
Asset Manager
Object
Get

This subscripted AssetManager attribute represents a list of object references to the media assets linked to the project. Valid values for n range from 1 (the first asset in the project) to `AssetManager.count` (the last asset in the project).

The asset[*n*] attribute does not necessarily represent the nth asset in the **Asset** palette because text elements, color tables, and the system color table (the "Mac OS 8bit" color table) are also included in the asset count. In addition, some items in this list may be null object references as "holes" are left in the asset list when assets are removed from a project. Experimentation may be required to find which numbers correspond to which assets.

attributeCount
All Objects
Integer
Get

This attribute contains the number of attributes that the object supports. This attribute is useful in conjunction with the attributeName[*n*] and attributeValue[*n*] attributes.

attributeName[*n*]
All Objects
String
Get

This attribute is a string list that contains the name of each attribute that the object supports. To access individual list elements, the index *n* must be specified. Valid values range from 1 (one) to attributeCount.

attributeString[*n*]
All Objects
String
Get

This attribute is a string list that contains the string equivalent of the current values of each attribute that the object supports (numeric versions of these values are stored in the attributeValue attribute). For example, if attributeValue[5] contains the integer range `2 thru 5`, attributeString[5] contains the string "`2 thru 5.`"

Attributes that are write-only return the string "Error: Write Only." Attributes that are lists and, as such, require selection subscripts (asset[*n*]) return "Error: Requires Selector."

attributeType[*n*]
All Objects
String
Get

This attribute is a string list that describes the data type of each attribute supported by the object. For example, if attributeName[5] is "`position,`" attributeType[5] would be "`Point.`"

Possible values of this attribute are:

- "Boolean"
- "Float"
- "Integer"
- "Object Pointer"
- "Point2D"
- "String"
- "Unknown"

attributeValue[*n*]
All Objects
All Types
Get

This attribute contains a list of the current values of the object's attributes. The value stored in attributeValue[*n*] is the current value of the attribute whose name is stored in attributeName[*n*]. For example, if attributeName[5] is "`position,`" attributeValue[5] would contain the object's current position as a point value (such as`(10,20)`).

autoResetCursor
World Manager
Boolean
Get Set

Set this WorldManager attribute to true to reset the cursor to the standard system arrow whenever a scene starts. The default behavior is for cursors to be persistent until explicitly removed by the Cursor modifier. Note that this attribute will have no effect if the cursorElement attribute (later in this chapter) is being used to make a mTropolis element act as the cursor.

See also "refreshCursor" later in this chapter.

autoScreenFade
World Manager
Boolean
Set

Set this WorldManager attribute to true to enable automatic fades between the scenes of a project. This fade is a built-in, full screen fade. When a scene ends, the entire screen goes dark, the next scene is loaded, any color table switching that activates on Scene Started occurs, then the screen brightens to display the new scene.

This attribute only works with 256 color (8-bit) projects.

When this attribute is set to true, scene transitions (created with the Scene Transition modifier) will appear to be ignored, as they happen while the screen is darkened.

By default, this attribute is set to false, meaning that automatic screen fades are disabled.

autoSharedScene
World Manager
Boolean
Set

By default, this WorldManager attribute is set to true, causing the first scene in a subsection to be considered the shared scene. When set to false, the first scene in a subsection is not automatically set to be the shared scene.

The Shared Scene modifier (see Chapter 12) can be used to specify a scene to be used as the shared scene. Using the Shared Scene modifier implicitly sets this attribute to false.

balance
Sound, Quick Time Elements
Integer
Get Set

Sound panning controls the left/right balance of the sound in the stereo field for a sound or QuickTime element. Valid values range from –100 to 100. Balance value –100 is full left, 0 is centered, 100 is full right.

bytesFree
System
Integer
Get

This System attribute contains the number of bytes currently free in the memory partition allocated to mTropolis.

See also "initialBytesFree" and "bytesLoaded" in this chapter.

bytesLoaded
Asset Manager
Integer
Get

This AssetManager attribute contains the number of bytes used by all assets that are currently loaded into memory.

See also "bytesFree" and "initialBytesFree" in this chapter.

cache
Visual Elements
Boolean
Get Set

When true, the contents of the element are cached in memory. Note that this "graphical element" attribute is not supported by mToon, sound, or QuickTime elements. Only still graphic elements (both unlinked graphic elements and elements linked to PICT files) and text elements can be cached. See "Cache Bitmap Check Box" in Chapter 5.

cel
mToon Elements
Integer
Get Set

The animation cel (frame) currently displayed in an mToon element. The first cel is cel number 1 (one).

celCount
mToon Elements
Integer
Get

The total number of cels in an mToon animation.

centerPosition
Visual Elements
Point
Get Set

An attribute of point type that contains the position of the center of the element with respect to the origin of its parent (that is, the upper left hand corner of its parent). The x field contains the number of pixels that the center is to the right of the parent's origin. The y field contains the number of pixels that the center is down from the parent's origin.

Changing the value of this attribute causes the element to move so that its center is at the newly-specified center position.

childElem[*n*]
All Objects
Ojects
Get

This attribute contains an object reference to each element (graphical, sound, structural, or text element) that is an immediate child of an object. The index *n* is used to specify which child element is being referred to. Valid values for *n* range from 1 (the first child element) to childElemCount (an attribute that contains the total number of child elements).

Using an index that is out of the valid range returns a "null" object reference value. Attempting to reference or target a null object may result in an error being reported in the Message Log window.

See also "childElemCount" and "childMod[*n*]" earlier in this chapter.

childElemCount
All Objects
Integer
Get

The total number of an object's immediate children that are elements. If the object has no element children, this attribute contains 0 (zero).

See also "childElem[*n*]," "childModCount" and "childObjectCount" earlier in this chapter.

childIndex
All Objects
Integer
Get Set

The object's position in its parent's list of elements (if the object is an element) or list of modifiers (if the object is a modifier).

Setting this attribute causes the object to change position in the project structure relative to its siblings. To make an object the first element or modifier in the list, set this attribute to 1 (one). To make an object the last element or modifier in the list, set this attribute to `parent.childElemCount` (if the object is an element) or `parent.childModCount` (if the object is a modifier).

The childIndex attribute cannot be set for section and subsection objects.

One useful application of this attribute is "numbering" the scenes in a subsection because `scene.childIndex` will return 1 for the first scene (usually the shared scene), 2 for the second scene (the first "regular" scene), up to `subsection.childElemCount` (the last scene in the subsection).

See also "childElemCount," "childModCount," and "childObjectIndex" elsewhere in this chapter.

childMod[*n*]
All Objects
Object
Get

This attribute contains an object reference to each modifier that is an immediate child of an object. The index *n* is used to specify which child modifier is being referred to. Valid values for *n* range from 1 (the first child modifier) to childModCount (an attribute that contains the total number of child modifiers).

Using an index that is out of the valid range returns a "null" object reference value. Attempting to reference or target a null object may result in an error being reported in the Message Log window.

See also "childModCount" and "childElem[*n*]" elsewhere in this chapter.

childModCount
All Objects
Integer
Get

The total number of an object's immediate children that are modifiers. If the object has no modifier children, this attribute contains 0 (zero).

See also "childMod[*n*]," "childElemCount," and "childObjectCount" elsewhere in this chapter.

childObject[*n*]
All Objects
Object
Get

This attribute contains an object reference to each object of any type (elements or modifiers) that is an immediate child of an object. The index *n* is used to specify which child is being referred to. Child objects are listed in messaging order. Valid values for *n* range from 1 (the first child to receive a message) to childObjectCount (an attribute that contains the total number of child objects).

Using an index that is out of the valid range returns a "null" object reference value. Attempting to reference or target a null object may result in an error being reported in the Message Log window.

See also "childObjectCount," "childElemCount," and "childModCount" elsewhere in this chapter.

childObjectCount
All Objects
Integer
Get

The total number of an object's immediate children (both elements and modifiers). If the object has no children, this attribute contains 0 (zero).

See also "childObject[*n*]," "childElemCount," and "childModCount" in this chapter.

Object	classID	classType
Project	Project	structure
Section	Section	structure
Subsection	Subsection	structure
MOM element	Element's CompID*	element
QuickTime element	QuickTimeElement	element
mToon	mToonElement	element
PICT element	PICTElement	element
Empty graphic element	GraphicElement	element
Text element	TextElement	element
Sound element	SoundElement	element
MOM variables	Variable's CompID*	variable
MOM modifiers	Modifier's CompID*	modifier
Behavior modifier	BehaviorMod	modifier
Graphic modifier	GraphicMod	modifier
Text Style modifier	TextStyleMod	modifier
Compound Variable	CompoundVariable	variable
Other variables	VariableMod	variable
Other modifiers	InternalMod	modifier
MOM assets	Asset's CompID*	asset
Other assets	Color Table, PICT, QuickTime, Sound, Text	asset
Other objects	InternalObject	object

*A "CompID" is a unique string that identifies a MOM component, such as a modifier. MOM is mTropolis' extension architecture.

String values contained in the classID and classType attributes of mTropolis objects

childObjectIndex
All Objects
Integer
Get

The object's position in its parent's list of objects.

For example, if the object's parent contains 3 modifiers and 3 elements, the second modifier's childObject index would be 2 and the first element's childObjectIndex would be 4.

See also "childObjectCount" and "childIndex" in this chapter.

classID
All Objects
String
Get

A string that describes the "class" to which the object belongs. Possible values are shown in the table on the previous page.

classType
All Objects
String
Get

A string that describes the general "type" of object. Possible values are shown in the table on the previous page.

clearReturnList
World Manager
Boolean
Set

Set this WorldManager attribute to true to clear the scene change return list (used by the Change Scene, Navigation, Return, and Open Project modifiers). As a result, the Return modifier will have no effect until another scene is added to the return list.

clickCount
World Manager
Integer
Get

This WorldManager attribute contains the number of mouse clicks that have occurred within the system-defined double-click interval (controlled by the "Mouse" control panel on both Mac OS and Windows systems). Thus, this attribute can be used to detect multiple clicks. For example, consider the following script, which (when placed in a Miniscript modifier configured to execute on Mouse Up) sends a message called "double click" to the element when the mouse is double-clicked (or triple-clicked, etc.):

```
if WorldManager.clickCount >= 2 then
  send "double click"
end if
```

clickedLine
Text Elements
Integer
Get

The number of the line on which the end user last clicked in a text element. The first line of a text element is line number 1.

Note: Lines are defined by where hard returns are placed in the text, not by text wrapping.

clone
All Objects
Boolean/Object/String
Set

This attribute can be used to dynamically create new instances of mTropolis objects.

Basic use of the clone attribute
Set this attribute to true to create a new instance of the object. Setting the attribute in this way has the same effect as sending the element the **Clone** command. That is, the clone becomes the next sibling of the original object. Note that cloning does not occur immediately; mTropolis sends a Cloned message to the new element when it is created. See "Cloned/Clone" in Chapter 13.

Advanced use of the clone attribute
Set this attribute to an object reference or string (that contains a path to an object specified in standard mTropolis path syntax — see "Object Path Field" in Chapter 12) to cause the source object to be cloned as a child of the specified object. The target object can exist in another open mTropolis project. That is, set the source object's clone attribute in the following way:

```
set sourceObject.clone to targetObject
```

A new instance of the object specified by *sourceObject* is created as a child of *targetObject*.

Note that cloning does not occur immediately. When the attribute is set in this manner, an author message can be optionally sent to the active scene of the project that did the cloning. This message can be used to detect when the cloning operation is finished, which is especially useful when cloning objects across projects. The specific author message sent can be specified by setting the `worldManager.cloneNotify` attribute (described below).

Setting the clone attribute in this way is especially useful in the creation of mTropolis projects that act as "wizards." See Chapter 6, "Tools Menu" for more information on the creation of tools and wizards. An example "wizard" is described in Chapter 11 of the *mTropolis Developer Guide,* "Wizard Authoring Example."

See also "cloneNotify" and "kill" in this chapter.

cloneNotify
World Manager
String
Set

Set this attribute to a string that contains the name of the author message that should be sent when a clone operation (initiated by setting the clone attribute as described above in "Advanced Use of the Clone Attribute") is complete. This author message is sent to the modifiers on the active scene of the project that performed the clone operation.

close
Project
Boolean
Set

Set this project attribute to true to close the project. Note that projects can also be closed by sending the **Close Project** command (Chapter 13) to the project component.

combineRedraws
World Manager
Boolean
Get Set

This WorldManager attribute controls the way in which mTropolis updates the screen. By default, this attribute is set to true. When this attribute is set to true, mTropolis collects all changes to the screen in an offscreen buffer and then updates the changed region of the screen in its entirety. Thus, multiple mToons or other moving elements in a scene will appear to change at the same time. However, if the changing region of the screen grows larger (as would happen if two mToons moved far apart), animations may appear to get slower.

When this attribute is set to false, mTropolis updates changing areas of the screen individually as they change. Thus, multiple animations may appear to update asynchronously, but the overall time to update the screen will appear more constant.

In general, the default behavior (combineRedraws set to true) provides higher performance and a more pleasing effect on most platforms.

commandKeyDown
System
Boolean *Macintosh only*
Get

This System attribute contains true when the Command key (marked "⌘" on Mac OS) is being held down. Note that this attribute always returns false on the Windows platform, as there is no equivalent for the Command key.

See also "altKeyDown," "controlKeyDown," and "optionKeyDown" in this chapter.

controlKeyDown
System
Boolean
Get

This System attribute contains true when the Control key (usually marked Control on Mac OS and Cntl or Ctrl on Windows keyboards) is being held down.

See also "altKeyDown," "commandKeyDown," and "optionKeyDown" in this chapter.

controllerClick
QuickTime Elements
Boolean
Get Set

If this attribute is set to true, end user mouse clicks on the QuickTime movie controller are sent only to QuickTime; they are not passed to the mTropolis QuickTime element.

By default, the QuickTime controller is not visible. See "showController" in this chapter.

count
Asset Manager
Integer
Get

This AssetManager attribute contains the number of assets in the project. Note that the total count of assets includes text elements, color tables, and the system (such as "Mac OS 8-bit") color table.

cropPosition
Visual Elements
Point
Get Set

This attribute contains the position of the element, similar to the position attribute. Setting this attribute, however, moves the element while keeping the asset locked in place. In conjunction with the cropSize attribute (see below), this attribute allows programmatic control over the asset's cropping rectangle.

As an example, create an element linked to a media asset and put a Miniscript modifier with the following script, set to execute on "Mouse Up," onto the element:

```
 set cropPosition to
position+(10,10)
 set cropSize to size-(20,20)
```

In run-time mode, each time the element is clicked, the size of the cropping rectangle is reduced by 10 pixels on a side (because we're subtracting from the element's size), but the region stays centered on the element (because we're moving the cropPosition by an equal amount).

cropSize
Visual Elements
Point
Get Set

This attribute contains the size of the element, similar to the size attribute. Setting this attribute, however, changes the size of the element without scaling the asset (as happens when the size attribute is set). In conjunction with the cropPosition attribute (see above), this attribute allows programmatic control over the asset's cropping rectangle.

See "cropPosition" for an example of using this attribute to crop an element's media.

currentHotspot
QuickTime VR Elements
Integer
Get

An integer that contains the ID number of the QuickTime VR panorama movie hotspot currently under the mouse. These values range from 0 to 255.

currentNodeName
QuickTime VR Elements
String
Get

A string that contains the name (if any) of the currently-displayed node in a QuickTime VR panorama movie.

See also "node" elsewhere in this chapter.

currentScene
World Manager
Object
Get Set

This WorldManager attribute contains an object reference to the currently-visible scene. Reading this attribute is the same as referencing the `activeScene` Miniscript keyword. Changing the value of this attribute causes the scene to change. Note that scenes can also be changed using the Change Scene modifier or Navigation modifier (see Chapter 12).

cursorElement
World Manager
Object/String
Set

This WorldManager attribute can be used to change the cursor from its normal appearance into a graphic element (including an mToon or QuickTime element). Set the cursorElement attribute to an object reference to change the cursor into the specified element. Alternatively, this attribute can be set to a string that contains the name of the element to be used as the cursor. If the specified element is visible, it will appear to jump from its current location to the location of the mouse cursor.

In this mode, the cursor does not respond to the Change Cursor modifier, nor does it automatically change to the "hand" pointer when it is over mouse message-sensitive objects. Mouse movements control the position of the element, but it otherwise retains all of its attributes (including its position in the layer order) and programming.

Cursor elements generate mouse events in the same way as normal cursors, with one exception. The Mouse Over message is generated only when the cursor element is actually above an element in the layer order (that is, when the cursor element's layer order number is greater than the layer order number of the element it passes over).

For example, to use an mToon element named "myToon" as the cursor, execute the script:

```
set WorldManager.cursorElement \
  to myToon
```

To return the cursor to its normal operation, set this attribute to the special keyword "none." For example:

```
set WorldManager.cursorElement to none
```

cursorHotspot
World Manager
Point
Set

This WorldManager attribute, designed for use with the cursorElement attribute (described in this chapter), controls the location of a cursor element's "hotspot" (that is, the point at which end user mouse actions are registered). By default, the hotspot is located at the origin of the element (its upper left corner). The value of cursorHotspot specifies an offset to the right and down from the element's origin. This point becomes the point at which end user mouse actions are registered.

This attribute only works when an element is being used as the cursor (via the cursorElement attribute). It has no effect on standard cursors.

date
System
String
Get

This System attribute contains a string that represents the current date in the form "Oct. 08 1997".

See also "day," "longDate," "month," and "year" in this chapter.

day
System
Integer
Get

This System attribute contains an integer, from 1 to 31, that represents the current day in the month.

See also "dayInWeek," "dayInYear," "month," and "year" in this chapter.

dayInWeek
System
Integer
Get

This System attribute contains an integer, from 1 to 7, that represents the current day of the week. Day 1 is Sunday, day 7 is Saturday.

See also "day," "dayInYear," "month," and "year" in this chapter.

dayInYear
System
Integer
Get

This System attribute contains an integer, from 1 to 366, that represents the current day of the year. Day 1 is January 1st. For most years, day 365 is December 31st. During a leap year, December 31st is day 366.

See also "day," "dayInWeek," "month," and "year" in this chapter.

direct
Visual Elements
Boolean
Get Set

When true, the element is drawn "direct to screen." See "Direct to Screen Check Box" in Chapter 5. When false, the element is drawn in its regular position in the layer order.

duration
QuickTime Elements
Integer
Get

The length of a movie in QuickTime timevalues. To find the length of a movie in seconds, divide this value by the movie's timescale attribute.

editable
Text Elements
Boolean
Get Set

When true, the text element can be edited in run-time mode by the end user. In run-time mode, the cursor changes to an I-beam when passed over the element to indicate that it is editable. When the end user clicks on the element, an insertion point appears in the text just as if the **Enable Editing** command (see Chapter 13) had been sent to the element.

When this attribute is false (the default), the text element does not respond to clicks by becoming editable. However, it still responds to the **Enable Editing** command.

ejectCD
System
Boolean
Set

Set this System attribute to true to eject the CD from the CD-ROM drive. Use this attribute when you need to prompt the end user to insert a new CD in a multiple-disc title. See "Title Segments" and "Deploying Multiple-Disc Titles" in Chapter 1 for more information on creating multiple-disc titles.

element
All Objects
Object
Get

The element ancestor of an object (that is, the nearest element "above" the object as seen in the structure view).

This attribute is most often used as a "building block" in Miniscript object reference expressions. See "Specifying the Destination for a Message," "Building blocks for targeting project components in Miniscript expressions," in Chapter 14.

elementCount
System
Integer
Get

This System attribute contains the number of elements currently loaded anywhere in the mTropolis system, for all open projects.

flush
All Assets
Boolean
Set

Set this asset attribute to true to flush the asset from memory. For example:

```
set myElement.asset.flush to true
```

flushAll
Asset Manager
Integer
Set

Set this AssetManager attribute to an integer value (representing a flushPriority) to cause all media whose flushPriority is less than or equal to that value to be flushed from memory. Setting this attribute is equivalent to sending the *Flush All Media* command. See "Flush All Media (Message/Command Menu Only)" in Chapter 13 and "flushPriority" earlier in this chapter.

flushPriority
All Elements
Integer
Set

This attribute can be used to control the priority with which mTropolis handles preloaded media. To use this feature, set an element's flushPriority attribute to the desired priority value before sending the element a **Preload Media** command. These priorities are integer values starting at 0 — higher values have a higher priority as described below:

- 0–50: These are the "system" priorities assigned automatically by mTropolis when media is loaded. These priorities should not be used with flushPriority.
- 51–74: These are "manual" priorities for use with flushPriority. A higher value means that the media is less likely to be removed from memory. Media loaded with these priorities are automatically flushed when a scene change occurs.
- 75–99: More "manual" priorities for use with flushPriority. Media loaded with these priorities are not automatically flushed when a scene changes. They are persistent until manually removed with a **Flush Media** or **Flush All Media** command.
- 100 and higher: Media preloaded with priorities of 100 and higher are never removed from memory by mTropolis, even if a dangerous (system crashing) low memory situation occurs. Such priorities should be used with great caution.

Once assigned a flushPriority, media can be selectively flushed by sending the **Flush All Media** command to the project with an accompanying data value. All elements with flushPriorities less than or equal to this value are removed from memory. See "Flush All Media (Message/Command Menu Only)" in Chapter 13.

fov
QuickTime VR Elements
Float
Get Set

The current field of view for the QuickTime VR panorama movie, in degrees. Valid values range from 0 (fully zoomed in) to 90 (fully zoomed out).

fullPath
All Objects
String
Get

This element and modifier attribute contains the path to the object's position in the project's structure hierarchy. A slash (/) separates each level of the hierarchy. The project's name is fully spelled out, unlike the path attribute which represents it as "<project>."

This attribute is similar to the fullPath attribute of the Object Reference Variable (see Chapter 12, "Modifier Reference").

---

gameMode
System
Boolean *Macintosh only*
Set

Set this System attribute to true to give a Mac OS title maximum system resources. When gameMode is set to true, the Finder is essentially shut down. This setting is not compatible with titles being run over a network or tools running inside windows. File access and other system maintenance features are shut off while gameMode is active. As a result, this feature should be used with caution.

Set gameMode to false to return the system to normal operation.

This attribute is Mac OS-only. It has no effect when set in titles built for Windows platforms.

---

globalOffset
World Manager
Point
Get Set

This WorldManager attribute represents an offset of the entire project within the drawing region of the project. Changing `WorldManager.globalOffset` from its default of (0,0) causes all visible elements to appear shifted by the specified amount inside the drawing region. This change is persistent across scene changes. This feature can be used to allow end users to view and interact with scenes that are larger than the draw area of the project.

The draw area should not be shifted such that the boundaries of the scene enter the draw area. Attempting to draw in the empty, exposed draw area that results from such a shift may cause unpredictable results.

---

globalPosition
Visual Elements
Point
Get Set

A point that contains the position of the element with respect to the origin of the draw area (for example, the upper left corner of the draw area). The x field contains the number of pixels to the right of the draw area origin. The y field contains the number of pixels down from the draw area origin.

See also "position" in this chapter.

---

height
Visual Elements
Integer
Get Set

The height of the element (its "y" size) in pixels.

Elements resize around their origin (their top left corner), so the bottom boundary of the element moves when this value is changed.

---

hotspotName[*n*]
QuickTime VR Elements
String
Get

This attribute contains the name of each hotspot in a QuickTime VR panorama movie associated with each hotspot ID number, *n*. For example, `hotspotName[10]` is a string that contains the name associated with hotspot ID 10.

hours
System
Integer
Get

This System attribute contains an integer, from 0 to 23, that represents the current hour. Hour 0 is 12 AM, hour 23 is 11 PM.

See also "longTime," "minutes," "seconds," and "time" in this chapter.

initialBytesFree
System
Integer
Get

This System attribute contains the number of bytes of free memory available to projects after mTropolis has started up. This number will be less than the memory allocated to mTropolis in the Finder.

See also "bytesFree" and "bytesLoaded" in this chapter.

kill
All Objects
Boolean
Set

Set this attribute to true to remove the element from the project. Setting this attribute has the same effect as sending the element the Kill command. Note that the element is not killed immediately; mTropolis sends a Killed message to the element just before it is destroyed. The same caveats apply to this attribute as apply to the Kill command. See "Killed/Kill" in Chapter 13.

See also "clone" in this chapter.

layer
Visual Elements
Integer
Get Set

The layer order number of the element. See "Layer Order" in Chapter 10. If an element's layer order number is set to a layer that is already occupied by another element, the elements exchange layer numbers.

length
Text Elements
Integer
Get

The length, in characters, of a text element including all return characters (the total number of all characters in the text element).

See also "lineLength[*n*]" in this chapter.

line[*n*]
Text Elements
String
Get Set

The text contained in line number *n* of the text element (where n is an integer from 1 to lineCount). For example, to set the value of the second line of text element "myText," use the statement:

```
set myText.line[2] to "mFactory"
```

Lines are defined by where hard returns are placed in the text, not by text wrapping.

lineCount
Text Elements
Integer
Get

The total number of lines of text in the text element.

Lines are defined by where hard returns are placed in the text, not by text wrapping.

lineHeight
Text Elements
Integer
Get

The height, in pixels, of the first line in the text element.

lineLength[*n*]
Text Elements
Integer
Get

The length, in characters, of the *n*th line in the text element (where *n* is an integer from 1 to lineCount). Note that this value does not include the return character at the end of the line.

See also "length" elsewhere in this chapter.

load
Scenes
Boolean
Set

Set this scene attribute to true to cause the scene to be loaded into memory if it is not already. See also "unload" later in this chapter.

loadedCount
Asset Manager
Integer
Get

This AssetManager attribute contains the number of assets currently loaded in the project.

lockCD
System
Boolean
Set

Set this System attribute to true to "lock" the CD in the CD–ROM drive. The drive's eject button is disabled until the attribute is set to false again, or the mTropolis title is exited. Use this attribute to prevent end users from accidentally ejecting the CD while running multiple CD titles. Note that a particular CD–ROM drive may not respond to this attribute.

See also "ejectCD" earlier in this chapter.

longDate
System
String
Get

This System attribute contains a string that represents the current date in the form "Wednesday, October 8, 1997".

See also "date," "day," "month," and "year" in this chapter.

longTime
System
String
Get

This System attribute contains a string that represents the current time in the form "00:00:00 PM".

See also "hours," "minutes," "seconds," and "time" in this chapter.

loop
mToon, QuickTime, Sound Elements
Boolean
Get Set

When true, the time-based media (for example, mToon, QuickTime movie, sound) in the element starts playing again from the beginning of the current range when it reaches its end. When false, the media plays only once.

loopBackForth
QuickTime Elements
Boolean
Get Set

When true, the time-based media in the element plays backward when it reaches the end of the current range, then plays forward again when it reaches its beginning.

For QuickTime movies, the "loop" attribute should also be set to true or looping will not continue.

macSndBufferSize
World Manager
Integer *Macintosh only*
Get Set

This Mac OS-only WorldManager attribute controls the size, in bytes, of the buffer used to play a sound. When set to 0 (the default), mTropolis sets the size of this buffer automatically, starting with a size of 32K. Each sound is allocated this much buffer space. Changing this value sets the size of the sound buffer manually. The integer value represents the size of the buffer in bytes. Manually setting the size of the sound buffer to a value larger than 32K (greater than 32*1024 bytes) may improve sound performance in situations where "stuttering" or dropouts occur.

Setting this attribute affects the next sound that is played. It does not affect sounds that are already playing.

See also "winSndBufferSize" in this chapter.

maintainRate
mToon Elements
Boolean
Get Set

When this attribute is true, the mToon's "Maintain Rate" option is enabled. See "Maintain Rate" in Chapter 5.

masterVolume
System
Integer
Get Set

This System attribute acts as a volume control for the entire project. Valid values range from 0 (silence) to 7 (full volume).

This attribute works by setting the overall system volume control, affecting other applications in addition to the mTropolis title.

mediaSize
Visual Elements
Point
Get

The original size of the graphical media linked to the element (that is, the size to which the element would change if **Revert Size** were selected from the Object menu). The x field contains the original width of the element, in pixels. The y field contains the original height of the element, in pixels.

Text elements do not support this attribute.

memErrorCount
System
Integer
Get Set

This System attribute contains the number of memory errors (attempts to allocate memory that failed, attempts to preload media that will not fit in memory, etc.) that have occurred. Note that this attribute is writable, so it can be reset to zero as needed.

minutes
System
Integer
Get

This System attribute contains an integer from 0 to 59 that represents the current minutes into the hour.

See also “hours,” “longTime,” “seconds,” and “time” in this chapter.

modifierCount
System
Integer
Get

This System attribute contains the number of modifiers currently loaded anywhere in the mTropolis system, for all open projects.

monitorBitDepth
System
Integer *set Macintosh only*
Get Set

This System attribute can be used to retrieve or set the current color depth setting (in bits) of the screen on Mac OS systems. This attribute can be read on Windows platforms, but setting it has no effect.

Valid values for this attribute are:
- 1 for "B/W" mode (1-bit black and white).
- 2 for "4" color mode (2-bit color).
- 4 for "16" color mode (4-bit color).
- 8 for "256" color mode (8-bit palette color).
- 16 for thousands of colors (high color).
- 32 for millions of colors (true color).

Note that the color depth of the project itself is set as a project preference. See "Color Depth Pop-Up Menu" in Chapter 2. Color depths supported by the current Mac OS monitor can be found by using the supportsBitDepth attribute described earlier in this chapter.

month
System
Integer
Get

This System attribute contains an integer from 1 to 12, that represents the current month. January is month 1, December is month 12.

See also "day" and "year" in this chapter.

movieClick
QuickTime, QuickTime VR Elements
Boolean
Get Set

For QuickTime videos, this attribute defaults to false. If this attribute is set to true, end user mouse clicks on the QuickTime movie are not sent to mTropolis, but are handled by QuickTime itself. The movie will pause when clicked and play when double-clicked.

For QuickTime VR movies, this attribute defaults to true. If this attribute is set to false, end user mouse actions over the QuickTime VR movie do not let the end user navigate the movie (the default); instead, mouse actions are passed to mTropolis. This feature is useful for temporarily suspending QuickTime VR navigation so that a movie can be made draggable.

name
All Objects
String
Get Set

The name of the mTropolis object, as set in its configuration dialog box. For most elements, this attribute can be read or written; for projects, this attribute is read-only. See "Element Name Field" in Chapter 5.

newProject
System
Boolean *Editor only*
Set

Set this System attribute to true to create a new mTropolis project. Setting this attribute in edit mode is equivalent to selecting **New ➛ Project** from the **File** menu. See "Creating a New Project" in Chapter 1.

This attribute takes effect only in the mTropolis editing environment and is intended for use in the creation of tools and wizards. It has no effect when used with a mTropolis player application.

next
All Objects
Object
Get

This attribute contains an object reference to the object's next sibling of the same type. That is, if the object is an element (such as, graphical, sound, structural, or text element), this attribute refers to its next element sibling. If the object is a modifier, this attribute refers to its next modifier sibling.

If the object does not have a next sibling (that is, it is the last child of its parent), this attribute contains a null object reference.

This attribute is most often used as a "building block" in Miniscript object reference expressions. See "Specifying the Destination for a Message" and "Building blocks for targeting project components in Miniscript expressions," in Chapter 14.

nextObject
All Objects
Object
Get

This attribute contains an object reference to the next object in the structural hierarchy, regardless of its type or position in the hierarchy.

If the object does not have a next object (it is the very last object in the structure view), this attribute contains a null object reference ("none").

nextParentObject
All Objects
Object
Get

This attribute contains an object reference to the next parent object in the structural hierarchy. The next parent object is the next lowest object in the structure view at the same level *or higher* that could be a container for other mTropolis objects.

If the object does not have a next parent object (it is the very last object in the structure view), this attribute contains a null object reference ("none").

nextSibling
All Objects
Object
Get

This attribute contains an object reference to the object's next sibling, regardless of type (the next object at the same structural level, regardless of type).

If the object has no next siblings, this attribute contains a null object reference.

node
QuickTime VR Elements
Integer
Get Set

The ID number of the currently-displayed node in a QuickTime VR panorama movie. Changing this value changes the view displayed in the movie.

numericDate
System
Integer
Get

This System attribute contains an integer value that represents the current date in Unix format. This format has four digits for the year, then 3 digits for the day. For example, "1997123" means the 123rd day of 1997.

See also "date" in this chapter.

numericTime
System
Float
Get

This System attribute contains an integer value that represents the current time in Unix format.

See also "time" in this chapter.

objectID
All Objects
Integer
Get

A unique ID assigned to each object by mTropolis.

open
System
String *Editor only*
Set

Set this System attribute to a string that contains a path and file name (in standard Mac OS colon-delimited format) to open the mTropolis project, title, library, or mToon contained in the specified file. For example:

```
set system.open to \
  "myDisk:myFolder:myProject"
```

openEditDialog
All Objects
Boolean *Editor only*
Set

Set this attribute of elements or modifiers to true to open the object's info dialog box. Setting this attribute is equivalent to selecting the **Object ➝ Object Info** menu option or double-clicking the object. Setting openEditDialog in a running project or a built title has no effect.

This attribute is intended for use in the creation of tools (see Chapter 6, "Tools Menu") and, as such, is only useful for projects running in the mTropolis editor. It cannot be used with projects running in any of the stand-alone mTropolis players.

optionKeyDown
System
Boolean
Get

This System attribute contains true when the Option key (on Mac OS) or Alt key (on Windows) is held down. This attribute is identical to the attribute "altKeyDown" earlier in this chapter.

See also "altKeyDown," "commandKeyDown," and "controlKeyDown" in this chapter.

pan
QuickTime VR Elements
Float
Get Set

For Quicktime VR panorama movie elements, this attribute represents the horizontal pan setting of the currently-displayed view. Valid values range from 0 to 360 degrees.

parent
All Objects
Object
Get Set

This attribute contains an object reference to the object's parent component. Often, this attribute is used like the next and previous keywords to specify a relative destination for a message. However, this attribute can also be set.

Setting the value of this attribute to a new component causes the element to become a child of the newly-specified component. The structure of the project changes to reflect the new parent/child relationship. mTropolis also sends a Parent Changed message to the "re-parented" component (see "Parent Changed" in Chapter 13).

If the run-time Player Emulation option is turned off (that is, if the **File ➛ Run ➛ Player Emulation** menu toggle is unchecked), this change will be persistent between run-time and edit mode. Once an element is re-parented, examining the structure view upon returning to edit mode will show the changed project structure. Reparenting of components is not persistent if **File ➛ Run ➛ Player Emulation** is checked (the default).

For example, suppose a graphic element named "red" is the child of another graphic element (named "blue") in a scene. The "red" element could be made a child of the scene (and thus become a sibling of "blue" instead of blue's child) by executing the following Miniscript, located on the "red" element:

```
set element.parent to scene
```

path
All Objects
String
Get

This element and modifier attribute contains the path to the object's position in the project's structure hierarchy. A slash (/) separates each level of the hierarchy. The project's name is represented by the token "<project>," unlike the fullPath attribute, which fully spells out the project name. This attribute is similar to the path attribute of the Object Reference Variable (see Chapter 12, "Modifier Reference").

paused
mToon, QuickTime, Sound Elements
Boolean
Get Set

When true, the time-based media (such as an mToon, QuickTime, or sound) in the element is in its paused state. When false, the element is not paused. Changing the value of this attribute is equivalent to sending the element the **Toggle Pause** command. There are a number of pause-related messages and commands — see "Paused/Pause," "Unpaused/Unpause," and "Toggle Pause (Message/Command Menu Only)" in Chapter 13.

playerEmulation
Project
Boolean *Editor only*
Get Set

This project attribute contains true when player emulation is on (the **File ➛ Run ➛ Player Emulation** menu toggle is checked). Otherwise, this attribute contains false.

This attribute can also be set. Setting it to true causes player emulation to be turned on for the project, just as if the **File ➛ Run ➛ Player Emulation** menu toggle had been checked. Setting the attribute to false causes player emulation to be turned off for the project. When turned off in this way, mTropolis does not display the "player emulation turned off" warning dialog box, however.

See "Player Emulation" in Chapter 1 for more information about the player emulation menu toggle.

This attribute is intended for use in the creation of tools (see Chapter 6, "Tools Menu") and, as such, is only useful for projects running in the mTropolis editor. It cannot be used with projects running in any of the stand-alone mTropolis players.

playerType
System
String
Get

This System attribute contains a string that describes the type of player that is currently playing the title or project. Possible values are:

- editor: The title is being played back within the mTropolis editing environment.
- player: The title is being played by a stand-alone mTropolis player.
- plug-in: The title is being played within the mPire plug-in player for web browsers.

playEveryFrame
QuickTime Elements
Boolean
Get Set

When true, every frame in the QuickTime movie will be played regardless of the rate (audio may be dropped or go out-of-sync). When false, frames may be dropped to display the movie at its proper rate with audio in sync.

parameterCount
Project
Integer *Plug-in only*
Get

This Project attribute contains the number of parameters that were specified in the <EMBED> HTML tag used with the mPire web-browser plug-in.

parameterName[*n*]
Project
String *Plug-in only*
Get

This Project attribute contains a string list of the names of parameters specified in the <EMBED> HTML tag used with the mPire plug-in web-browser. Valid values for *n* range from 1 to parameterCount.

parameterValue[*n*]
Project
String *Plug-in only*
Get

This Project attribute contains a list that contains string versions of the values assigned to any parameters specified in the <EMBED> HTML tag used with the mPire web-browser plug-in. The value in parameterValue is the value assigned to the parameter whose name is stored in parameterName. Valid values for *n* range from 1 to parameterCount.

position
Visual Elements
Point
Get Set

An attribute of point type that contains the position of the element with respect to the origin of its parent (the upper left hand corner of its parent, position (0,0)). The x field contains the number of pixels to the right of the parent's origin. The y field contains the number of pixels down from the parent's origin.

Changing the value of this attribute causes the element to move to the newly-specified position.

postponeRedraws
World Manager
Boolean
Set

Set this WorldManager attribute to true to suspend updates to the entire screen. While this attribute is set to true, no changes to elements appear on the screen, though elements are still active and may be changing their position or appearance. All changes are drawn to the off-screen buffer only.

Set this attribute to false (the default) to restore the normal operation of the screen and display the current contents of the offscreen buffer.

This attribute is useful when many changes need to be made to the screen, but the author wants them to be revealed simultaneously.

To suspend screen updates for individual elements, use the redraw attribute, described in this chapter.

preload
All Assets
Boolean
Set

Set this asset attribute to true to cause the asset to preload. Setting this attribute to true is equivalent to sending the **Preload Media** command to an element that contains the asset. See "Preload Media (Message/Command Menu Only)" in Chapter 13.

preloaded
All Elements, All Assets
Boolean
Get

This attribute of both elements and assets is true when the asset has been completely preloaded.

previous
All Objects
Object
Get

This attribute contains an object reference to the object's previous sibling of the same type. That is, if the object is an element (graphical, sound, structural, or text element), this attribute refers to its previous element sibling. If the object is a modifier, this attribute refers to its previous modifier sibling.

If the object does not have a previous sibling (it is the first modifier or element child of its parent), this attribute contains a null object reference.

This attribute is most often used as a "building block" in Miniscript object reference expressions. See "Specifying the Destination for a Message" and "Building blocks for targeting project components in Miniscript expressions," in Chapter 14.

previousSibling
All Objects
Object
Get

This attribute contains an object reference to the object's previous sibling, regardless of type (the previous object at the same structural level, regardless of type).

If the object has no previous siblings, this attribute contains a null object reference.

project[*n*]
System
Object
Get

This System attribute contains a list of object references to all open projects. Valid values for *n* range from 1 to projectCount.

projectCount
System
Integer
Get

This System attribute contains the number of currently open projects.

range
mToon, QuickTime Elements
Integer Range
Get Set

The range of cels to be played in an mToon animation. The range of timevalues to be played in a QuickTime movie.

rate
mToon, QuickTime Elements
Integer/Float
Get Set

For mToons, this value is the playback speed in frames per second. Negative values make the mToon play in reverse.

For QuickTime, this value is a rate multiplier where the movie's default speed is represented by 1. Higher values make the movie play faster; lower values make the movie play slower. Negative values make the movie play in reverse. For example, setting this value to –2 would make the movie play backward at twice its normal speed.

For a change in the rate attribute to take effect correctly, either change the rate while the media is playing, or stop/pause the media, change the rate, and then unpause the media (set paused to false or send the **Unpause** or **Toggle Pause** commands). Do not send **Play.** Sending the **Play** command to the element will make it play forward at the default rate, overriding the new rate settings.

redraw
Visual Elements
Boolean
Set

Set this attribute to false to make the element stop showing screen updates as it changes. For example, if this attribute is set to false on an mToon element, the animation will appear to have stopped. Similarly, hiding an element after this attribute has been set to false will make the element insensitive to end user mouse actions, but the image of the element will remain on the screen until other elements move across that region. As they move, they will remove the old image as that part of the screen is redrawn.

Set this attribute to true (the default) to make the element show screen updates once again.

To suspend screen updates for the entire screen, use the postponeRedraws attribute discussed in this chapter.

refreshCursor
World Manager
Boolean
Set

This WorldManager attribute can be used to "refresh" the appearance and messaging status of the mouse cursor. Set this attribute to true to cause the position of the cursor with respect to elements in the scene to be evaluated. This causes mouse messages, such as Mouse Over or Mouse Outside to be sent to elements that should receive those messages. This feature is useful because, for performance reasons, messages such as Mouse Over are only generated when the end user moves the mouse over an element. Such messages are not generated when a moving element moves under a stationary cursor.

regPoint
mToon Elements
Point
Get

The offset (a point value) from the mToon element's origin, to the mToons registration point. See "Align and Trim Cels" in Chapter 1.

running
Project
Boolean/Object *Editor only*
Get Set

This project attribute contains true if the project is running, either inside of a window or full-screen.

Set this attribute to false to cause the project to stop running and return to edit mode.

Set this attribute to true to cause the project to start running (just as if ⌘-T had been pressed). Alternatively, this attribute can be set to an object reference to cause the project to start running from that selection (similar to pressing ⌘-Y to run the project from the current scene).

See "Run — Switching Between Edit and Runtime Modes" in Chapter 1 for information on using mTropolis menu options and command keys to run a project.

This attribute is intended for use in the creation of tools (see Chapter 6, "Tools Menu") and, as such, is only useful for projects running in the mTropolis editor. It cannot be used with projects running in any of the stand-alone mTropolis players.

runningInWindow
Project
Boolean/Object *Editor only*
Get Set

This project attribute contains true if the project is running in a window (running with the mTropolis interface still visible, as when ⌘-Option-T or ⌘-Option-Y is pressed).

Set this attribute to false to cause the project to stop running and return to edit mode.

Set this attribute to true to cause the project to start running in a window (just as if ⌘-Option-T had been pressed). Alternatively, this attribute can be set to an object reference to cause the project to start running in a window from that selection (similar to pressing ⌘-Option-Y to run the project from the current scene).

See "Run — Switching Between Edit and Runtime Modes" in Chapter 1 for information on using mTropolis menu options and command keys to run a project.

This attribute is intended for use in the creation of tools (see Chapter 6, "Tools Menu") and, as such, is only useful for projects running in the mTropolis editor. It cannot be used with projects running in any of the stand-alone mTropolis players.

save
Project
Boolean *Editor only*
Set

Set this project attribute to true to cause the current project to be saved. Setting this attribute is equivalent to selecting the **File ➛ Save** menu option.

This attribute is intended for use in the creation of tools (see Chapter 6, "Tools Menu") and, as such, is only useful for projects running in the mTropolis editor. It cannot be used with projects running in any of the stand-alone mTropolis players.

saveAs
Project
String *Editor only*
Set

Set this project attribute to a string that contains a fully-qualified file path (in standard Mac OS `drive:folder:file` syntax) to cause the project to be saved with that new file name. Setting this attribute is equivalent to selecting the **File ➛ Save As** menu option.

This attribute is intended for use in the creation of tools (see Chapter 6, "Tools Menu") and, as such, is only useful for projects running in the mTropolis editor. It cannot be used with projects running in any of the stand-alone mTropolis players.

scaledSize
Visual Elements
Point
Get Set

This attribute contains the scaled size of the media linked to an element (a point value). By default, this value will be the same as the element's size. However, setting the attribute allows you to scale media without altering the size of the element. If the media is made larger than the element, the media appears cropped.

See also "cropPosition," "cropSize," and "mediaSize" in this chapter.

sceneCount
World Manager
Integer
Get

This WorldManager attribute contains the number of currently loaded scenes in the project.

scrollOffset
Visual Elements
Point
Get Set

A point value that describes the current offset of the media within the frame of the element. Changing the value of this attribute is similar to sending **Scroll Up**, **Scroll Down**, **Scroll Right**, or **Scroll Left** commands to an element These commands are described in Chapter 13.

See also "cropPosition," "cropSize," and "scaledSize" in this chapter.

seconds
System
Integer
Get

This System attribute contains an integer, from 0 to 59 that represents the current seconds into the minute.

See also "hours," "longTime," "minutes," and "time" in this chapter.

selection[*n*]
Project
Object *Editor only*
Get Set

This project attribute contains a list of object references to any currently-selected objects. The object specified by selection[*n*] is the *n*th selected object. Valid values for *n* range from 1 to selectionCount.

This attribute is intended for use in the creation of tools (see Chapter 6, "Tools Menu") and, as such, is only useful for projects running in the mTropolis editor. It cannot be used with projects running in any of the stand-alone mTropolis players.

See also "selectionCount" in this chapter.

selectionCount
Project
Integer *Editor only*
Get

This project attribute contains the number of currently selected objects.

This attribute is intended for use in the creation of tools (see Chapter 6, "Tools Menu") and, as such, is only useful for projects running in the mTropolis editor. It cannot be used with projects running in any of the stand-alone mTropolis players.

See also "selection[*n*]" in this chapter

shared
All Elements
Boolean
Get Set

Set this attribute to true to lock the element (and everything else on its scene) into memory so that the element can be persistent across scene changes. To stop sharing an element, set this attribute to false (the default).

This attribute can be useful with sound elements to make them play across scene and subsection changes. It can also be useful with graphic elements. For example, an mToon that represents the title's "point-of-view" character might have this attribute set to true. The mToon will then be present in every scene in the title.

See also "sharedElemCount" and "sharedElem[*n*]" in this chapter.

sharedElemCount
World Manager
Integer
Get

This WorldManager attribute contains the number of elements that have been shared in the project (that is, the number of elements whose shared attribute is set to true).

See also "shared" and"sharedElem[*n*]" in this chapter.

sharedElem[*n*]
World Manager
Object
Get

This WorldManager attribute contains a list of object references to all the elements that have been shared in the project. Valid values for *n* range from 1 (the first shared element) to sharedElemCount (the last shared element).

For example, to hide the first shared element in the project, execute the following script:

```
send "Hide" to wm.sharedElem[1]
```

See also "shared" and "sharedElemCount" in this chapter.

shiftKeyDown
System
Boolean
Get

This System attribute contains true when the Shift key on the keyboard is being held down.

See also "altKeyDown," "commandKeyDown," "controlKeyDown," and "optionKeyDown" in this chapter.

showController
QuickTime, QuickTime VR Elements
Boolean
Get Set

When true, the QuickTime movie's standard play controls are visible. When false, the controller is hidden. This attribute can only be used with QuickTime movies that are set to play "direct to screen." See "direct" in this chapter and "Direct to Screen Check Box" in Chapter 5.

showCursor
QuickTime VR Elements
Boolean
Get Set

When true (the default), the cursor changes to indicate QuickTime VR panorama hotspots when it passes over them. When false, the cursor does not change.

size
Visual Elements
Point
Get Set

A point value that contains the width and height of the element. The x field contains the width of the element, in pixels. The y field contains the height of the element, in pixels.

startTransition
QuickTime VR Elements
Boolean
Set

Set this QuickTime VR attribute to true to force the panorama movie to begin any pending transition immediately. This happens automatically when the panorama's updateMode attribute is "normal" or "swing" and its pan, tilt, or zoom attributes are changed. However, if updateMode is "none" when those changes are made, the panorama will not update until it is forced to redraw, such as when it is clicked. For example, the following code updates the panorama movie after both its pan and tilt attributes have been set:

```
set updateMode to "none"
set pan to 20
set tilt to 10
set updateMode to "swing"
set startTransition to true
```

staticSmoothing
QuickTime VR Elements
String
Get Set

A string that represents the quality of the anti-aliasing being performed on the QuickTime VR panorama movie. Valid values are:

- full: The highest quality anti-aliasing is being used. This is the default.
- partial: Medium quality anti-aliasing is being used.
- none: No anti-aliasing is being used.

supportsBitDepth[*n*]
System
Boolean *Macintosh only*
Get

This read-only System attribute can be used to query which color depths a Mac OS monitor supports. This attribute has no effect when used in titles built for Windows platforms.

The value of supportsBitDepth[*n*] is true if the Mac OS monitor is capable of displaying graphics in the bit depth specified by n. Valid values for n are:

- 1 for black and white (1-bit color)
- 2 for 4-color mode (2-bit color)
- 4 for 16-color mode (4-bit color)
- 8 for 256-color mode (8-bit palette color)
- 16 for thousands of colors (16-bit high color)
- 32 for millions of colors (32-bit true color)

The monitorBitDepth attribute (described earlier in this chapter) can be used to change the current Mac OS monitor settings. For example, the following script changes the Mac OS display to 8-bit (256 color mode) if it is capable:

```
if System.supportsBitDepth[8] then
  set System.monitorBitDepth to 8
end if
```

text
Text Elements
String
Get Set

The text string displayed in a text element. Changing the value of this attribute causes the text element to update.

textHeight
Text Elements
Integer
Get

The height, in pixels, of the entire amount of text in a text element. Note that this value may be different from the height of the first line multiplied by the number of lines (i.e, `lineHeight * lineCount`) if any of the text lines wrap, or if embedded formatting is used in the text element.

See also “lineHeight” and “lineCount” in this chapter.

ticks
System
Float
Get

This System attribute contains a floating-point value that represents the number of ticks that have elapsed since the computer was started. Ticks are Mac OS time units that equal 1/60th of a second. This value is slightly different from the ticks environment variable in that it contains a floating-point value instead of an integer.

tilt
QuickTime VR Elements
Float
Get Set

The current vertical panning for the QuickTime VR panorama movie, in degrees. Valid values range from –90 (looking straight down) to 90 (looking straight up).

time
System
String
Get

This System attribute contains a string that represents the current time in the form "00:00 PM".

See also "hours," "longTime," "minutes," and "seconds" in this chapter.

timeScale
QuickTime Elements
Integer
Get

The rate of the QuickTime movie in QuickTime timevalues per second. To compute the movie's length in seconds, divide the movie's duration attribute by this value (seconds = duration/timescale).

timeValue
QuickTime Elements
Integer
Get Set

The current time in the QuickTime movie (in QuickTime-specific units). To compute the current time in seconds, divide this value by the movie's timescale attribute.

trackCount
QuickTime Elements
Integer
Get

The number of tracks in a QuickTime movie.

trackDisable
QuickTime Elements
Integer/String
Set

Set this attribute to an integer (representing a QuickTime track number) or a string (representing a QuickTime track name) to disable the specified QuickTime track.

trackEnable
QuickTime Elements
Integer/String
Set

Set this attribute to an integer (representing a QuickTime track number) or a string (representing a QuickTime track name) to enable the specified QuickTime track.

trickleEnabled
Asset Manager
Boolean
Get Set

When this Asset Manager attribute is set to true, trickle loading of assets is enabled. When set to false, no trickle loading can be processed, but new elements can be added to the trickle loading queue (by sending them the **Trickle Load Media** command described in Chapter 13).

See "Trickle Load Done/Trickle Load Media" in Chapter 13.

trickleReadSize
Asset Manager
Integer
Get Set

This Asset Manager attribute controls how many Kilobytes of data are read during a single idle cycle when mTropolis trickle loads media. The default is 16 (16K of data is read each idle cycle). Setting this attribute to a larger value causes mTropolis to read more data with each idle cycle, decreasing total trickle load time but possibly degrading performance (especially on slower machines). Setting this attribute to a smaller value causes mTropolis to read less data with each idle cycle, causing the total trickle load time to increase but reducing the possibility of performance degradation. This attribute cannot be set to a value less than one

See "Trickle Load Done/Trickle Load Media" in Chapter 13.

unload
Scenes
Boolean
Set

Set this scene attribute to true to unload a previously-loaded scene from memory. Note that the current scene cannot be unloaded using this attribute (as such an operation would present the end user with a blank, inoperative screen).

See also "load" in this chapter.

updateMode
QuickTime VR Elements
String
Get Set

A string that represents the way in which the end user's view of a QuickTime VR movie changes when its location is changed (by changing the node, pan, tilt, and fov attributes). Valid values are:

- normal: The movie updates to the new location as soon as the location is set. The view appears to "jump" instantly to the new location.
- none: The movie does not update.
- swing: The view updates with a "swing" transition (that is, it pans smoothly) to the new location. This setting is the default.

userTimeout
Project
Integer
Get Set

This project attribute represents a length of time after which the User Timeout message is sent. This value is measured in 1/60 second intervals. For example, to generate User Timeout messages at intervals of 1 second, set this attribute to 60.

See “User Timeout” in Chapter 13.

visible
Visual Elements, Project
Boolean
Get Set

When true, the element is visible. When false, the element is hidden. Setting this attribute to true is equivalent to sending the **Show** command (see “Shown/Show” in Chapter 13). Setting this attribute to false is equivalent to sending the **Hide** command (see “Hidden/Hide” in Chapter 13).

This attribute has a special behavior for the project component on Mac OS platforms. For projects, this attribute is true when the project window is visible. Setting it to false does not close the project, however, it simply hides the window.

volume
Sound, QuickTime Elements
Integer
Get Set

An integer value representing a percentage of the sound or QuickTime element’s full volume. A value of 0 indicates that the sound will be completely silent when played. A value of 100 indicates that the sound will play at full volume.

volumeIsMounted
System
Boolean
Get

Use this System attribute, in conjunction with the volumeName System attribute, to check whether a CD with a specified volume name is currently in the CD–ROM drive. Set System.volumeName to a string that contains the name of the volume that you want to check for, then examine the contents of System.volumeIsMounted. This attribute will contain true if the specified volume is in the CD–ROM drive, otherwise it will contain false.

See “volumeName” in this chapter for an example of using these attributes.

volumeName
System
String
Set

Use this System attribute, in conjunction with the volumeIsMounted WorldManager attribute, to check whether a CD with a specified volume name is currently in the CD–ROM drive. Set `system.volumeName` to a string that contains the name of the volume that you want to check for, then examine the contents of `system.volumeIsMounted`. This attribute will contain true if the specified volume is in the CD–ROM drive, otherwise it will contain false. For example:

```
set System.volumeName to "myCD"
if System.volumeIsMounted then
  send "correct CD in drive"
else
  send "requested CD not in drive"
end if
```

warpMode
QuickTime VR Elements
String
Get Set

A string that represents the way warping is being used to display the QuickTime VR panorama movie. Valid values are:

- full: Both horizontal and vertical.
- partial: Horizontal warping is being used.
- none: No warping.

width
Visual Elements
Integer
Get Set

The width of the element (its "x" size) in pixels.

Elements resize around their origin (their top left corner), so the right boundary of the element moves when this value is changed.

windowMaxSize
Project
Point *Macintosh only*
Get Set

This project attribute contains the maximum size of the window in which the project is running. This attribute is supported only for projects that have been modified to run in a window with the Window Prefs modifier. The window must also be "resizable" for this attribute to have any effect.

This attribute is only supported on Mac OS platforms.

windowMinSize
Project
Point *Macintosh only*
Get Set

This project attribute contains the minimum size of the window in which the project is running. This attribute is supported only for projects that have been modified to run in a window with the Window Prefs modifier. The window must also be "resizable" for this attribute to have any effect.

This attribute is only supported on Mac OS platforms.

windowPosition
Project
Point *Macintosh only*
Get Set

This project attribute contains the current position (with respect to the upper left corner of the screen) of the window in which the project is running. This attribute is supported only for projects that have been modified to run in a window with the Window Prefs modifier. Setting this attribute causes the window to change its position.

This attribute is only supported on Mac OS platforms.

windowSize
Project
Point *Macintosh only*
Get Set

This project attribute contains the current size of the window in which the project is running. This attribute is supported only for projects that have been modified to run in a window with the Window Prefs modifier. Setting this attribute causes the window to change its size.

This attribute is only supported on Mac OS platforms.

windowStyle
Project
String *Macintosh only*
Set

This project attribute can be used to specify the current style of the window in which the project is running. This attribute is supported only for projects that have been modified to run in a window with the Window Prefs modifier. Setting this attribute to a new window description string causes the window to change its appearance and behavior.

The style can be specified using a string that contains one of the base styles (palette, document, modal, or borderless) and appending any of the optional attributes (closeable, floating, movable, resizable, or sidebar). Not all options can be used with all base types. Valid values for this attribute, with options in parentheses, are shown below:

- palette (closeable | resizable | sidebar)
- document (closeable | resizable)
- modal (movable)
- borderless (floating)

For example, to make the current window a palette-style window that is resizable, execute the following Miniscript:

```
set project.windowStyle to \
  “palette resizable”
```

This attribute is only supported on Mac OS platforms.

windowTitle
Project
String *Macintosh only*
Get Set

This project attribute contains the title of the window in which the project is running. This attribute is supported only for projects that have been modified to run in a window with the Window Prefs modifier. Setting this attribute to a new string causes the window to change its title.

This attribute is only supported on Mac OS platforms.

winSndBufferSize
World Manager
Integer *Windows only*
Get Set

This Windows-only WorldManager attribute controls the size of the buffer used to play sounds. This size is the total amount of sound buffer space available to all sounds. When set to 0 (the default), mTropolis sets the size of this buffer automatically, depending upon the type of sound that is playing:

- for 16-bit/44KHz sounds: 160K
- for 16-bit/22KHz sounds: 88K
- for lower quality formats, including 8-bit/22KHz sounds: 40K

Changing this value sets the size of the sound buffer manually. The integer value represents the size of the buffer in bytes. Manually setting the size of the sound buffer to a larger value may improve sound performance in situations where "stuttering" or dropouts occur.

See also "macSndBufferSize" in this chapter.

year
System
Integer
Get

This System attribute contains an integer that represents the current year (such as `1997`).

See also "day" and "month" in this chapter.

Lists of Attributes by Component Type

The tables below show the attributes that can be accessed for each type of mTropolis component. Also shown are the attribute's corresponding data type and two columns labeled "Get" and "Set."

If a bullet appears in an attribute's "Get" column, the attribute is readable. For these attributes, using the syntax `component.attribute` in a script returns the current value of attribute (which has the data type shown in the "Type" column) for the specified component (an object reference expression that specifies the desired mTropolis object).

If a bullet appears in an attribute's "Set" column, the attribute is writable. The value of the attribute can be changed by using the set statement (for example, `set myToon.paused to true`). Changing the value of an attribute

causes that property of the component to change instantly.

Note that not all component types support every attribute. Consult these tables for lists of attributes supported by each component type. In addition to the attributes shown below, keep in mind that the project, section, subsection, scene, element, and parent "building blocks" described in Chapter 14 are essentially attributes of each element though they may not all be shown in the tables below.

Visual element attributes

The following attributes are common to graphic elements, mToon elements, text elements, QuickTime elements, and QuickTime VR elements.

Visual Element Attribute	Type	Get	Set
cache	Boolean	•	•
centerPosition	Point	•	•
cropPosition	Point	•	•
cropSize	Point	•	•
direct	Boolean	•	•
globalPosition	Point	•	•
height	Integer	•	•
layer	Integer	•	•
mediaSize	Point	•	
position	Point	•	•
redraw	Boolean		•
scaledSize	Point	•	•
scrollOffset	Point	•	•
size	Point	•	•
visible	Boolean	•	•
width	Integer	•	

mToon element attributes

The following attributes are unique to mToon elements.

mToon Element Attribute	Type	Get	Set
cel	Integer	•	•
celCount	Integer	•	
loop	Boolean	•	•
maintainRate	Boolean	•	•
paused	Boolean	•	•
range	Range	•	•
rate	Integer	•	•
regPoint	Point	•	

QuickTime element attributes

The following attributes are unique to QuickTime elements.

QuickTime Element Attribute	Type	Get	Set
balance	Integer	•	•
controllerClick	Boolean	•	•
duration	Integer	•	
loop	Boolean	•	•
loopBackForth	Boolean	•	•
movieClick	Boolean	•	•
paused	Boolean	•	•
playEveryFrame	Boolean	•	•
range	Range	•	•
rate	Integer/ Float	•	•
showController	Boolean	•	•
timeScale	Integer	•	•
timeValue	Integer	•	•
trackCount	Integer	•	
trackDisable	Integer/ String		•
trackEnable	Integer/ String		•
volume	Integer	•	•

QuickTime VR element attributes

The following attributes are unique to QuickTime VR panorama elements.

QuickTime VR Element Attribute	Type	Get	Set
currentHotspot	Integer	•	
currentNodeName	String	•	
fov	Float	•	•
hotspotName[*n*]	String	•	
movieClick	Boolean	•	•
node	Integer	•	•
pan	Float	•	•
showController	Boolean	•	•
showCursor	Boolean	•	•
startTransition	Boolean		•
staticSmoothing	String	•	•
tilt	Float	•	•
transitionSpeed	Integer	•	•
updateMode	String	•	•
warpMode	String	•	•

Sound element attributes

The following attributes are unique to sound elements.

Sound Element Attribute	Type	Get	Set
balance	Integer	•	•
loop	Boolean	•	•
paused	Boolean	•	•
volume	Integer	•	•

Text element attributes

The following attributes are unique to text elements.

Text Element Attribute	Type	Get	Set
append	String		•
clickedLine	Integer	•	
editable	Boolean	•	•
length	Integer	•	
line[*n*]	String	•	•
lineCount	Integer	•	
lineHeight	Integer	•	
lineLength[*n*]	Integer	•	
text	String	•	•
textHeight	Integer	•	

Scene attributes

The following attributes are unique to scene components.

Scene Attribute	Type	Get	Set
load	Boolean		•
unload	Boolean		•

Project attributes

The following attributes are unique to the project component.

Project Attribute	Type	Get	Set
allowQuitKey	Boolean		•
close	Boolean		•
parameterCount	Integer	•	
parameterName[*n*]	String	•	
parameterValue[*n*]	String	•	
playerEmulation‡	Boolean	•	•
running‡	Boolean/ Object	•	•
runningInWindow‡	Boolean/ Object	•	•
save‡	Boolean		•
saveAs‡	String		•
selection[*n*]‡	Object	•	•
selectionCount‡	Integer	•	
userTimeout	Integer	•	•
windowMaxSize†	Point	•	•
windowMinSize†	Point	•	•
windowPosition†	Point	•	•
windowSize†	Point	•	•
windowStyle†	String		•
windowTitle†	String	•	•
visible†	Boolean	•	•

† This attribute is supported on Macintosh systems only.

‡ This attribute is supported in the mTropolis editor only, not in players or plug-ins.

WorldManager attributes

The WorldManager is the mTropolis component that controls the execution of the project. WorldManager attributes control global options.

In Miniscript statements, the `WorldManager` component can be abbreviated as wm, if desired. For example, the following two Miniscript **set** statements are equivalent:

```
set WorldManager.refreshCursor to
true
set wm.refreshCursor to true
```

Note that if there are multiple projects open (as when a tool is running and there is also a project open for editing), there are multiple worldManagers active. In this case, WorldManager should be treated like a project attribute. For example, if two projects are open the autoScreenFade attributes of those two projects could be set independently like so:

```
set system.project[1] \
  .wm.autoSceenFade to true

set system.project[2] \
  .wm.autoScreenFade to false
```

The following attributes are unique to the WorldManager component.

WorldManager Attribute	Type	Get	Set
autoResetCursor	Boolean	•	•
autoScreenFade	Boolean		•
autoSharedScene	Boolean		•
clearReturnList	Boolean		•
clickCount	Integer	•	
cloneNotify	String		•
combineRedraws	Boolean	•	•
currentScene	Object	•	•
cursorElement	Object/ String		•
cursorHotspot	Point		•
globalOffset	Point	•	•
macSndBufferSize†	Integer	•	•
postponeRedraws	Boolean		•
refreshCursor	Boolean		•
sceneCount	Integer	•	
sharedElemCount	Integer	•	
sharedScene	Object	•	•
sharedElement[*n*]	Object	•	
winSndBufferSize	Integer	•	•

† This attribute is supported on Macintosh systems only.

System attributes

The System is the mTropolis component that controls the execution of all open mTropolis projects. Most System attributes control hardware-related options.

The following attributes are unique to the System component.

System Attribute	Type	Get	Set
activeProject‡	Object	•	
altKeyDown	Boolean	•	
bytesFree	Integer	•	
commandKeyDown†	Boolean	•	
controlKeyDown	Boolean	•	
ejectCD	Boolean		•
elementCount	Integer	•	
gameMode†	Boolean		•
initialBytesFree	Integer	•	
lockCD	Boolean		•
masterVolume	Integer	•	•
memErrorCount	Integer	•	•
modifierCount	Integer	•	
monitorBitDepth†	Integer	•	•
newProject‡	Boolean		•
open‡	String		•
optionKeyDown	Boolean	•	
playerType	String	•	
project[*n*]	Object	•	
projectCount	Integer	•	
shiftKeyDown	Boolean	•	
supportsBitDepth[*n*]†	Boolean	•	
volumeIsMounted	Boolean	•	
volumeName	String		•

System time attributes

The following time-related attributes are unique to the system component.

System Time Attribute	Type	Get	Set
abbreviatedDate	String	•	
date	String	•	
day	Integer	•	
dayInWeek	Integer	•	
dayInYear	Integer	•	
hours	Integer	•	
longDate	String	•	
longTime	String	•	
minutes	Integer	•	
month	Integer	•	
numericDate	Integer	•	
numericTime	Float	•	
seconds	Integer	•	
ticks	Float	•	
time	String	•	
year	Integer	•	

†This attribute is supported on Macintosh systems only.
‡This attribute is supported in the mTropolis editor only, not in players or plug-ins.

Any object attributes

The following attributes are common to all mTropolis objects (modifiers, elements, and structural components).

Any Object Attribute	Type	Get	Set
classID	String	•	
classType	String	•	
clone	Boolean/ Object/ String		•
fullPath	String	•	
kill	Boolean		•
name	String	•	•
objectID	Integer	•	•
openEditDialog‡	Boolean		•
path	String	•	

‡ *This attribute is supported in the mTropolis editor only, not in players or plug-ins.*

Any element attributes

The following attributes are common to all mTropolis elements.

Any Element Attribute	Type	Get	Set
asset	Object/ String	•	•
flushPriority	Integer		•
shared	Boolean	•	•
preloaded	Boolean	•	

Any asset attributes

The following attributes are unique to media assets linked to the project.

Any Asset Attribute	Type	Get	Set
flush	Boolean		•
preload	Boolean		•
preloaded	Boolean	•	

AssetManager attributes

The AssetManager is the mTropolis component that controls media assets. This component supports a number of attributes relating to media assets that have been linked to the project.

In Miniscript statements, the `AssetManager` component can be abbreviated as am, if desired. For example, the following two Miniscript set statements are equivalent:

```
set myint to AssetManager.count
set myint to am.count
```

The following attributes are unique to the AssetManager component.

AssetManager Attribute	Type	Get	Set
asset[*n*]	Object	•	
bytesLoaded	Integer	•	
count	Integer	•	
flushAll	Integer		•
loadedCount	Integer	•	
trickleEnabled	Boolean	•	•
trickleReadSize	Integer	•	•

Structure attributes

The following attributes can be used to create object reference expressions. These attributes are common to all mTropolis objects (modifiers, elements, and structural components).

Structure Attribute	Type	Get	Set
childElem[*n*]	Object	•	
childElemCount	Integer	•	
childIndex	Integer	•	•
childMod[*n*]	Object	•	
childModCount	Integer	•	
childObject[*n*]	Object	•	
childObjectCount	Integer	•	
childObjectIndex	Integer	•	•
element	Object	•	
next	Object	•	
nextObject	Object	•	
nextParentObject	Object	•	
nextSibling	Object	•	
parent	Object	•	•
previous	Object	•	
previousSibling	Object	•	

"Attribute" attributes

The following attributes can be used to identify the attributes of any mTropolis object. All mTropolis objects support these attributes.

Attributes Attribute	Type	Get	Set
attributeCount	Integer	•	
attributeName[*n*]	String	•	
attributeString[*n*]	String	•	
attributeType[*n*]	String	•	
attributeValue[*n*]	All Types	•	

Appendix A: MovieTrax

Appendix A: MovieTrax

This appendix describes MovieTrax, a stand-alone Mac OS application bundled with mTropolis that allows simple editing and manipulation of QuickTime video files.

What is MovieTrax?

MovieTrax is a QuickTime editing application that allows the editing of tracks within a QuickTime file. Tracks are the basic components of a QuickTime movie that contain different types of media to be played. QuickTime supports many different types of tracks including video, audio, text, MIDI, sprite, and timecode tracks. Tracks are independent of one another and a movie can contain multiple tracks of the same type. For example, different video tracks in a movie may have different durations, compression encodings, bit depths, start times, etc. Multiple audio tracks in a single QuickTime movie can be used to create "multilingual" movies.

MovieTrax can be used to create new multitrack movies from component media or to edit existing ones. Basic temporal and spatial editing options are also supported, including the creation of ranges in a movie.

Using QuickTime element attributes (see Chapter 15, "Attributes") or the Track Control modifier (see "Track Control Modifier" in Chapter 12), mTropolis can manipulate multitrack QuickTime movies. For example, the Track Control modifier can be used to activate or deactivate individual tracks in a movie.

Starting MovieTrax

To start MovieTrax, double-click the MovieTrax icon, found in the QuickTime utilities folder of the mTropolis distribution. The MovieTrax menus appear. MovieTrax menu options are described in the sections below.

Double-click the MovieTrax icon to launch MovieTrax

File Menu

Options in the **File** menu can be used to open, import, and save QuickTime movies for editing with MovieTrax.

New Movie

Choose this option to create a new QuickTime movie. A new, empty List window is created.

Open Movie

Choose this option to open an existing QuickTime file. A standard dialog box appears. Choose a QuickTime movie to open. If the movie has not previously been opened by MovieTrax, a new List window appears, listing the tracks the movie contains. Movies that have previously been opened by MovieTrax open to show the configuration of windows visible when the movie was last saved.

Note that other types of component media can also be opened and converted into QuickTime movies using this option. Valid files are displayed in the dialog box and the **Open** button changes to **Convert** button when they are highlighted. Click **Convert** to open the media file and convert it to a QuickTime movie. The media becomes available in MovieTrax as a QuickTime track.

Import Tracks
Choose this option to import the tracks from a QuickTime file directly into the currently-active movie (the movie associated with the currently-active view).

Export Tracks
Choose this option to save only the currently-selected tracks as a new QuickTime file. A standard dialog box appears.

!!! Assigned ranges are not exported with tracks. Ranges are only preserved when a movie is saved using the **Save** or **Save As** options.

Close
Choose this option to close the currently-active window.

Close Project
Choose this option to close all the windows and files associated with the currently-active movie.

Save
Choose this option to save the currently-active movie with its current file name. If the movie has not yet been saved, a standard dialog box appears. Enter a name for the movie and click **Save.** Mark the "Make movie self-contained" check box to create a movie that is independent of its original source files.

Save As
Choose this option to save the currently-active movie with a new name. A standard dialog box appears. Enter a name for the movie and click **Save.** Mark the "Make movie self-contained" check box to create a movie that is independent of its original source files.

Quit
Choose this option to quit MovieTrax. If there are any unsaved or changed movies open, MovieTrax asks if they should be saved before quitting.

Edit Menu

MovieTrax supports the following standard edit options.

Undo
Choose this option to undo the last operation. Note that not all operations can be undone.

Cut
Choose this option to cut the current selection and place it on the clipboard.

Copy
Choose this option to copy the current selection to the clipboard.

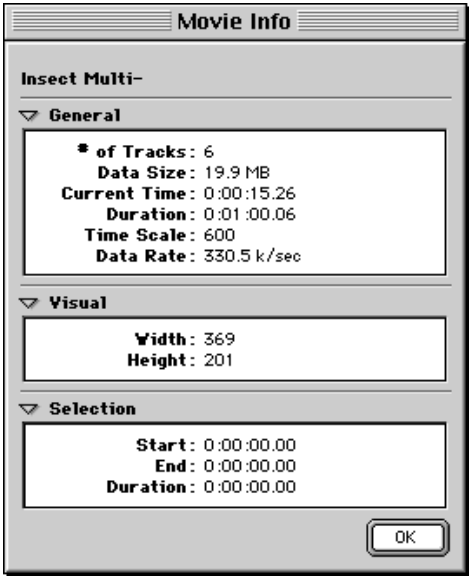
Paste
Choose this option to paste the contents of the clipboard.

Select All
Choose this option to select all tracks in the currently-active window.

Movie Menu

Options in the **Movie** menu affect all tracks in the current movie.

Movie Info
Choose this option to display information about the currently-active movie. The **Movie Info** dialog box is displayed as shown below.



*The **Movie Info** dialog box*

Play/Pause
Choose this option to play the currently-active movie. Note that if the spatial view is not open, the movie plays, but the video portion is not visible.

When a movie is playing, this menu option changes to **Pause.** Select it to stop the currently-playing movie.

Display Time Values/Display Standard Time
Use this menu toggle to change the time format shown in the Movie Controller between QuickTime "time values" and hours:minutes:seconds.hundredths format.

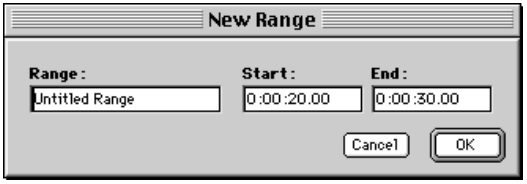
Loop
Choose this menu toggle to cause the movie to loop from the beginning (repeatedly play through and restart from the beginning) when it plays.

Back and Forth
Choose this menu toggle to cause the movie to loop back and forth (repeatedly play forward to the end then backward to the beginning) when it plays.

Play Every Frame
Choose this menu toggle to play the movie without dropping frames. Note that audio does not play when this option is checked.

Play Selection Only
Choose this menu toggle to play only the currently-selected range (as selected from the Movie Controller's **Ranges** pop-up menu) when **Play** is activated.

New Range
Choose this command to specify a new range. The **New Range** dialog box appears. Enter a name for the range (if desired), and a start and end time in QuickTime units. Note that if a range selection has been made in the Movie Controller, start and end times for that range are shown in the dialog box. Click **OK** to create the new range. The new range becomes available in the **Range** pop-up menu on the movie controller. Click **Cancel** to dismiss the dialog box without creating a new range.



*The **New Range** dialog box*

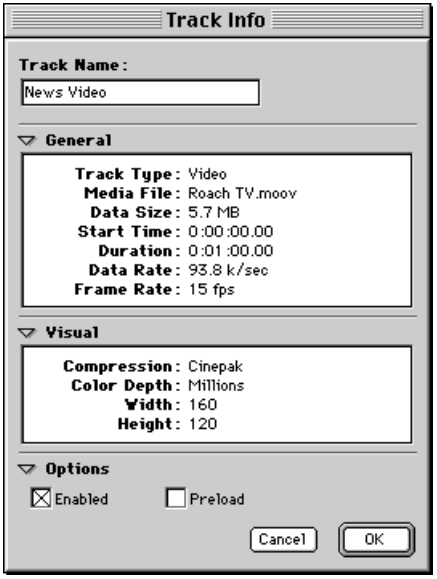
Once defined in a movie, QuickTime ranges can be used in mTropolis as described in "Played/Play" in Chapter 13 and "range" in Chapter 14.

Track Menu

These options can be used to configure the behavior and appearances of tracks within a movie. Note that many of these commands are only accessible when the Spatial window is open and a track has been selected in that window.

Track Info

Choose this option to display information about the currently-selected track (the track can be selected in the List, Spatial, or Temporal window). The **Track Info** dialog box appears. A number of information sections, with their own open/close triangles, are shown. Click the triangles to open or close that information section.



*The **Track Info** dialog box*

The **Options** section contains check boxes that can be set by the end user. These options control various QuickTime track defaults:

- **Enabled:** Check this box (the default) to set the track as initially enabled when used in mTropolis.
- **Preload:** Check this box to set the track as initially preloaded when used in mTropolis. Note that preloading of media is limited by the memory available on the machine that plays the media. By default, this box is unchecked.

Half Size

Choose this option to display the track currently selected in the Spatial window at half its normal size.

Normal Size
Choose this option to display the track currently selected in the Spatial window at normal size (the default).

Double Size
Choose this option to display the track currently selected in the Spatial window at double its normal size.

Bring to Front
Choose this option to bring the track currently selected in the spatial view to the front of any other tracks that it overlaps.

Send to Back
Choose this option to send the track currently selected in the spatial view to the back of any other tracks that it overlaps.

Bring Forward
Choose this option to bring the track currently selected in the spatial view one position forward in the "layer order" of tracks.

Send Backward
Choose this option to send the track currently selected in the spatial view one position back in the "layer order" of tracks.

Align
The options in this cascading menu can be used to spatially align two or more tracks in the spatial view or to align the start and end times of two or more tracks in the Temporal view. Shift-click or ⌘-click tracks in either view to multiple-select. The following options are available when two or more tracks are selected in the Spatial window:

- **Left:** Choose this option to align the left edges of the selected tracks. This option is only available for tracks selected in the Spatial window.
- **Right:** Choose this option to align the right edges of the selected tracks. This option is only available for tracks selected in the Spatial window.
- **Top:** Choose this option to align the top edges of the selected tracks. This option is only available for tracks selected in the Spatial window.
- **Bottom:** Choose this option to align the bottom edges of the selected tracks. This option is only available for tracks selected in the Spatial window.

The following options are available when two or more tracks are selected in the Temporal window:

- **Start:** Choose this option to align the start times of the selected tracks.
- **End:** Choose this option to align the end times of the selected tracks.

Clip
The options in this cascading menu can be used to clip the visible area of the track currently selected in the Spatial window. Options are:

- **Crop:** Choose this option to turn the selected track's resize handles (marked by red dots in the spatial view) into cropping handles (marked by green dots). Moving the handles changes the "frame" of the track, without resizing its contents.
- **Paste:** This option is only available when a PICT image is on the clipboard. Choose this

option to paste the PICT image on the current track as a clipping region. The PICT is converted to black and white — white pixels indicate a clipped region, while black pixels reveal the image underneath.

- **Invert:** Choose this option to invert the clipping region of the currently-selected track.
- **Clear:** Choose this option to clear the clipping region of the currently-selected track.

Matte

The options in this cascading menu can be used to specify a matte for the currently-selected track. A matte is similar to a clipping region, except that a matte has degrees of transparency and may also impart color tinting to the track. Options are:

- **Paste:** This option is only available when a PICT image is on the clipboard. Choose this option to paste the PICT on the current track as a matte. White areas of the PICT are completely opaque while black areas are completely transparent, showing the image underneath. Grayscale and colored pixels impart a tint to the image based upon their lightness.
- **Clear:** Choose this option to remove a matte from the currently-selected track.

Ink

The options in this cascading menu can be used to specify an ink effect for the track currently-selected in the Spatial window. Options are:

- **Normal:** Choose this option (the default) to display the selected track in its default state. This option is similar to mTropolis' "Copy" ink effect.
- **Bkgd Transparent:** Choose this option to make the "Op Color" (selected with the "Set Op Color" option, described below) transparent in the selected track.
- **Blend:** Choose this option to make the "Op Color" transparent based on lightness values in the selected track. This option is similar to the mTropolis "Blend" ink effect.
- **Set Op Color:** Choose this option to display a color selection dialog box. Choose a color to be used in the Background Transparent and Blend options described above.

Set Volume

Select this option to select the volume of the selected audio track. The **Set Volume** dialog box appears. Use the volume slider to select a value from 0 (silent) to 100 (full volume). Click **OK** to accept the change. Click **Cancel** to dismiss the dialog box without changing the audio's volume. This option is available only when an audio track is selected in the List or Temporal window.

Set Balance

Choose this option to select the stereo balance of the selected audio track. The **Set Balance** dialog box appears. Use the volume slider to select a value from –100 (full left) to 100 (full right). Click **OK** to accept the change. Click **Cancel** to dismiss the dialog box without changing the audio's stereo position. This option is available only when an audio track is selected in the List or Temporal window.

Set Start Time

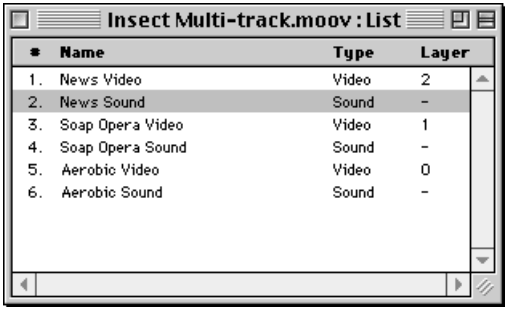
Choose this option to set the start time of the selected track to the time indicated by the Movie Controller. This option is available only when a track is selected in the Temporal window.

View Menu

Options in this menu can be used to show and hide the various MovieTrax display windows. Options and the windows they select are described below.

List window

Choose this menu toggle to show and hide the List window. The List window shows the tracks in a movie and lists the name, type, and layer number of tracks. The layer number of tracks is similar to the layer order number of elements in mTropolis. Tracks with a lower layer number appear "in front" of other tracks in the spatial view.



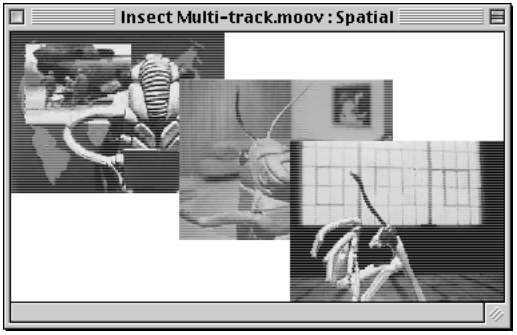
The List window

Other features of this window include:

- Tracks can be selected by clicking their names. Use ⌘-click to select multiple items.
- Selected tracks can be deleted by pressing the Delete key.
- Tracks can be dragged and dropped in other windows.

Spatial window

Choose this menu toggle to show and hide the Spatial window. The Spatial window shows the visual portion of a movie. Tracks can be repositioned in this window by dragging and dropping.



The Spatial window, shown here from a movie with three video tracks

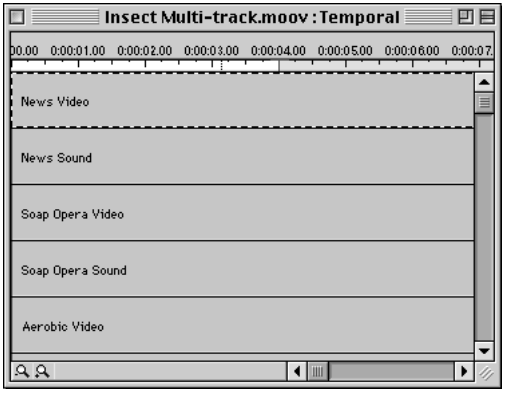
Other features of this window include:

- As a track is repositioned in the window, its x, y coordinates in pixels (with respect to the left and top edges of the window) are reported at the bottom of the window.
- Tracks can be selected by clicking them. Multiple tracks can be selected using ⌘-click or Shift-click.
- When a track is selected, red dots appear on its corners. These dots represent resize handles. Drag the handles to scale the track. Width and height of the track, in pixels, are reported at the bottom of the window.
- Selected tracks can be deleted by pressing the Delete key.
- When the mouse is over a track, its name is shown at the bottom of the window.

To display a track (or multiple tracks) in the Spatial window, you must click "Enabled" in the "Options" section of the **Track Info** dialog box.

Temporal window

Choose this menu toggle to show and hide the Temporal window. The Temporal window shows the start and end times of tracks. The time position of a track can be changed by dragging it within the window. The start and end times of the track are shown at the bottom of the window as the track is dragged.



The Temporal window, shown here from a movie with four tracks. The shaded part of the bar represents the extent of the track

Other features of this window include:

- A time scale, in hours:minutes:seconds:hundredths, is shown at the top of the window. The scale of the view can be changed by clicking the zoom out and zoom in tools at the bottom left corner of the window.
- Tracks can be selected by clicking their shaded portions. Use Shift-click to select multiple items.
- Selected tracks can be deleted by pressing the Delete key.
- If a range of time values have been selected in the Movie Controller (by Shift-dragging the Controller's slider), the range is displayed as a yellow highlight in the time scale.

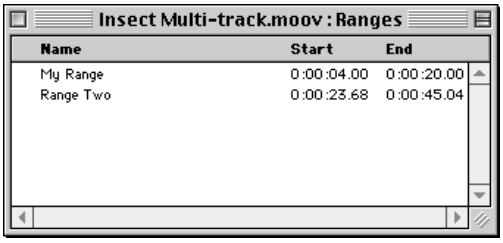
Movie Controller

Choose this menu toggle to show and hide the Movie Controller. The Movie Controller is similar to the standard QuickTime controller, including volume, play/pause, position slider, step, and loop controls. In addition, a time format toggle and **Range** pop-up menu have been added.

Features of the Movie Controller include:

- Choose the volume icon to display a volume control slider.
- Choose the play/pause icon to play or pause the movie.
- Drag the slider to set the current time of the movie. Shift-drag the slider to select a range of time values.
- Click the step controls to step backward or forward through the movie one frame at a time.
- Choose the loop control to toggle between "play once" and "looped" play modes.
- Choose the time format control to toggle the display of time between "time value" and "hour:minute:second:hundredths" mode.
- The **Range** pop-up menu can be used to select previously-defined ranges or specify new ones. Choose **New Range** to display the **New Range** dialog box (see "New Range" in this Appendix). If a range of times has been selected by Shift-dragging the slider, those values will be shown as the defaults in the **New Range** dialog box.

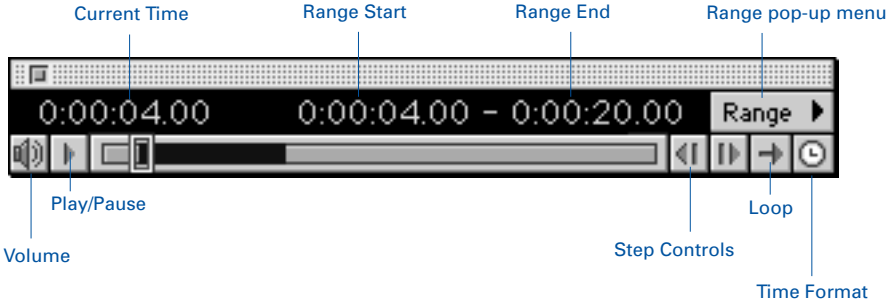
Ranges
Choose this menu to show and hide the Ranges window. The Ranges window lists any previously-defined ranges by name, start time value, and end time value. Double-click a range name to display the **Time Range** dialog box, which can be used to change the name, start time, and end time of the selected range.



The Ranges window

Window Menu

The **Window** menu lists the names of all open MovieTrax windows. The current window is indicated with a check mark. Choose any window name from the menu to make that window the current one. The chosen window moves to the "front" of the screen.



The Movie Controller

Appendix B: Panorama Location Namer

Appendix B: Panorama Location Namer

Panorama Location Namer is a utility that allows named locations to be added to a QuickTime VR panorama movie. Once added to a QuickTime VR panorama, these locations can be used with the Panorama Navigation modifier. (See Chapter 12.)

The Panorama Navigation modifier can be used to control the current view of the QuickTime VR "camera." In this way, mTropolis can control panoramas in addition to the usual end user mouse control over the current view.

When a QTVR panorama that contains named locations is linked to a mTropolis project, these locations can be chosen from the Panorama Navigation modifier's **By Name** pop-up menu. Selecting QTVR locations in this way is often easier than having to enter the Node, FOV, Pan, and Tilt information as numbers.

The features of this simple utility are described below.

Launching Panorama Location Namer

To launch Panorama Location Namer, double-click the Panorama Location Namer icon, found in the **QuickTime Utilities** folder of the mTropolis distribution. The Panorama Location Namer menus appear. Individual menu options are described below.

To launch Panorama Location Namer, double-click the Panorama Location Namer icon

File Menu

Options in the **File** menu can be used to open, close, and save panoramas for editing with Panorama Location Namer.

Open movie

Choose this menu option (⌘-O) to open a panorama for editing. A standard dialog box appears. Note that Panorama Location Namer can only be used with QuickTime VR panorama movies — QuickTime VR object movies are not supported and cannot have named locations added to them. Multiple movies can be open at the same time.

Close

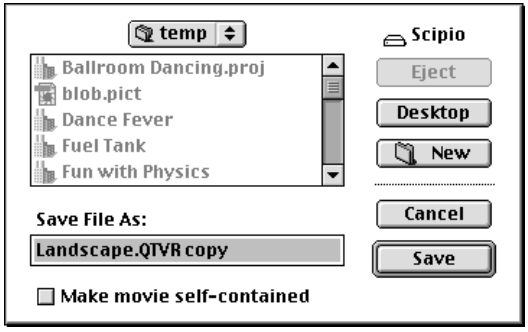
Choose this menu option (or press ⌘-W) to close the currently active panorama.

Save

Choose this option (⌘-S) to save changes to the currently active panorama. New or changed locations are saved with the panorama.

Save As

Choose this option to save the currently active panorama with a new name or location. A file selection dialog box appears.



*The **Panorama Location Namer Save As** dialog box*

Make movie self-contained check box

This check box controls how the new panorama movie is saved. When unchecked (the default), the movie is saved with "dependencies." That is, the new file contains just the new panorama location information with a "link" to the original panorama file. This setting results in a new file with a very small file size, but this file is dependent upon the original panorama movie to supply the actual panorama images. When checked, the panorama is written as a "self-contained" QuickTime movie with no dependencies — all information about the panorama resides in the new file.

Quit

Choose this menu option (⌘-Q) to quit the Panorama Location Namer utility. If any open panoramas contain unsaved changes, Panorama Location Namer asks if you want to save the changes before quitting.

Edit Menu

Panorama Location Namer has an **Edit** menu that contains the standard Mac OS editing options. Note, however, that in the current version of Panorama Location Namer, only the following menu options are fully supported.

Clear

Choose this menu option to delete the currently-selected location (as shown in the Locations window).

Select All

Choose this menu option to choose all of the locations shown in the Locations window.

Panorama Menu

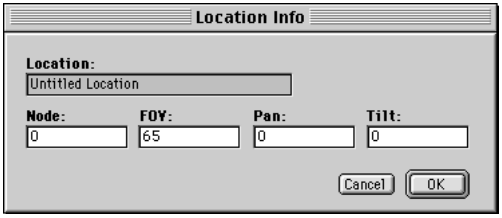
The **Panorama** menu contains options that allow you to manipulate the current QuickTime VR panorama movie.

Panorama info

This menu option is not supported in the current version of Panorama Location Namer.

New location

Choose this menu option (⌘-G) to create a new named location in the panorama movie. The **Location Info** dialog box appears. By default, the new location takes its settings from the current view shown in the Preview window. The **Location** text field is highlighted so the name of the new location can be changed.



*The **Location Info** dialog box being used to create a new location*

To create a new location:

1. Use the Preview window to navigate to the location to which you want to assign a name. See "Changing the View" in this Appendix.
2. The **Location Info** dialog box appears. Type the new name for the location into the **Location** text field.
3. Click **OK** to dismiss the dialog box and create the new location. The new location appears in the Locations window.

Go to location

Choose this menu option (⌘-J) to change the view shown in the Preview window to the location currently selected in the Locations window.

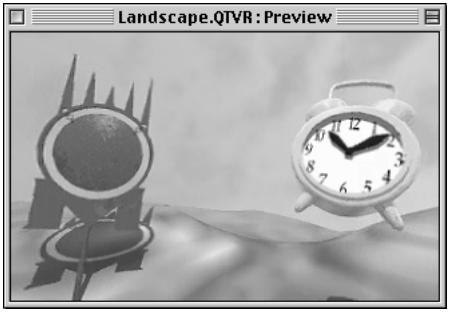
View Menu

The options in this menu can be used to toggle the Panorama Location Namer view windows on and off.

Preview window

Choose this menu toggle (⌘-1) to toggle display of the Preview window on and off.

Navigating in the Preview Window
The standard QuickTime VR navigation controls can be used to set the current view in the Preview window.

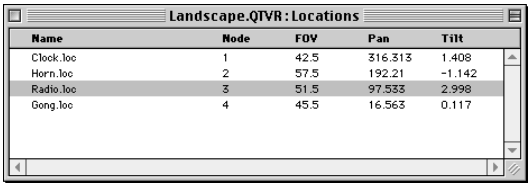


The Panorama Location Namer Preview window

1. Drag the mouse over the Preview window to change the current FOV, Pan, and Tilt parameters.
2. Press the Option key to zoom in.
3. Press the Control key to zoom out.

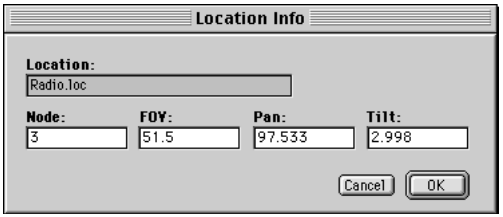
Locations
Choose this menu toggle (⌘-2) to toggle display of the Locations window on and off.

Selecting a location
Click on a location in the Locations window to make that location the currently-selected location.



The Panorama Location Namer Locations window. In this figure, the location named "Radio" is currently-selected

Displaying information about a location
Double-click on a location to display its **Location Info** dialog box. The location's "Node:," "FOV:," "Pan:," and "Tilt:" fields can be changed in this dialog box. Click **OK** to dismiss the **Location Info** dialog box and confirm any changes to the location's parameters.

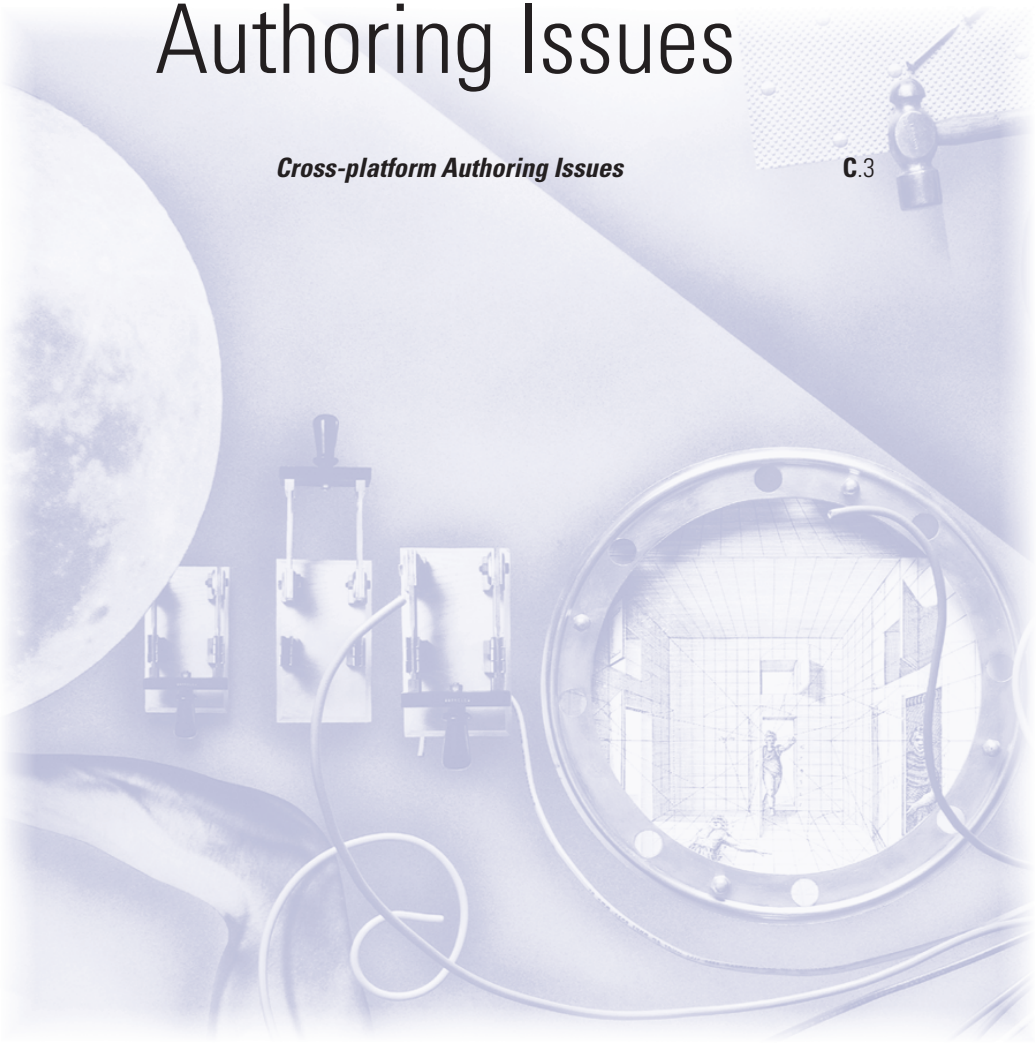


*The **Location Info** dialog box*

Window Menu

Use the menu toggles in this menu to choose the currently-active window. These options are useful when there is more than one panorama movie open.

C. Appendix C: Cross-platform Authoring Issues



C.

Appendix C: Cross-platform Authoring Issues

This appendix describes differences that need to be taken into account when authoring titles for distribution on both Mac OS and Windows platforms.

Cross-platform Authoring Issues

The mTropolis player for Windows has a small set of functional differences from the Mac OS editor and player, as well as several limitations that are not present on the Mac OS platform. Most of these differences and limitations are due to fundamental differences in the underlying platforms. Those inconsistencies that can be addressed by mTropolis will be minimized in future releases. The following sections list differences and limitations of the mTropolis players for Windows.

Graphics, effects, and transitions

- The Oval Iris transition (in the Element Transition modifier) operates as a rectangular transition, and the Fade transition does not operate.
- Only black and white cursors are provided for Windows platforms.
- The Gradient modifier has no effect.
- Scene fade-ins and fade-outs typically occur faster on Windows platforms.
- Ink effects (found in the **Ink** pop-up menu and the **Tool** palette and in the Graphic modifier) are limited to Copy, Background Transparent, Background Matte, and Invisible.
- Tinting (changing the foreground/background color of an imported image with the **Tool** palette or a Graphic modifier) is not supported, except for 1-bit images.
- Tone up, tone down, and bevels (found in the Image Effect modifier) use a pattern rather than being determined by value (true tone down and tone up are not supported).

Keyboard differences

- When a Keyboard Messenger is set to detect control keys, it responds to all control key combinations except those on the numeric keypad, excluding the Enter key.
- Option and Command are mapped to Alt on Windows platforms.
- Alt+certain character keys will cause a system beep if it is not available as a mTropolis shortcut key (such as, Alt-F), however the Keyboard Messenger will still send the message.

mToon

- QuickTime codecs are not supported for mToons on Windows platforms. When building a Windows title, mToons compressed with QuickTime codecs will be treated by default as uncompressed mToons, and will be converted to the mFactory Animation codec. You can work around this by unchecking the "Uncompressed mToons to mFactory Animation" check box in the **Build Title** dialog box.

Sounds

- Mono sounds can't be panned in Windows.
- If a monaural sound is played, any subsequent stereo sounds that are started while the mono sound is still playing will be played mono as well. This mono/stereo state is reset when all currently playing sounds have stopped. This only happens when the Dynamic sound quality option is selected in the **Build Title** dialog box.
- Sound Elements cannot play at the same time as QuickTime sound on Windows.

Text

- Outline, shadow, extended, and condensed text styles are not supported on Windows.
- Only the Text Style modifier can be used to format text cross-platform. Styles applied with the **Format** menu are ignored on Windows.

- A word that is wider than the text element is wrapped while the element is being edited, but clipped when editing is completed.

QuickTime

- QuickTime movies with multiple video tracks are not supported on Windows. For details, see the description of the Track Control modifier in Chapter 12 of the *mTropolis Reference Guide,* "Modifier Reference."
- QuickTime movies are always drawn direct to screen.
- Sounds cannot be played simultaneously with QuickTime video.
- Only one QuickTime audio track can play at a time.
- Building for AVI will not work with projects that contain QuickTime VR assets.

Window preferences

- The Window Prefs modifier has no effect on Windows titles.

Mac OS-only attributes

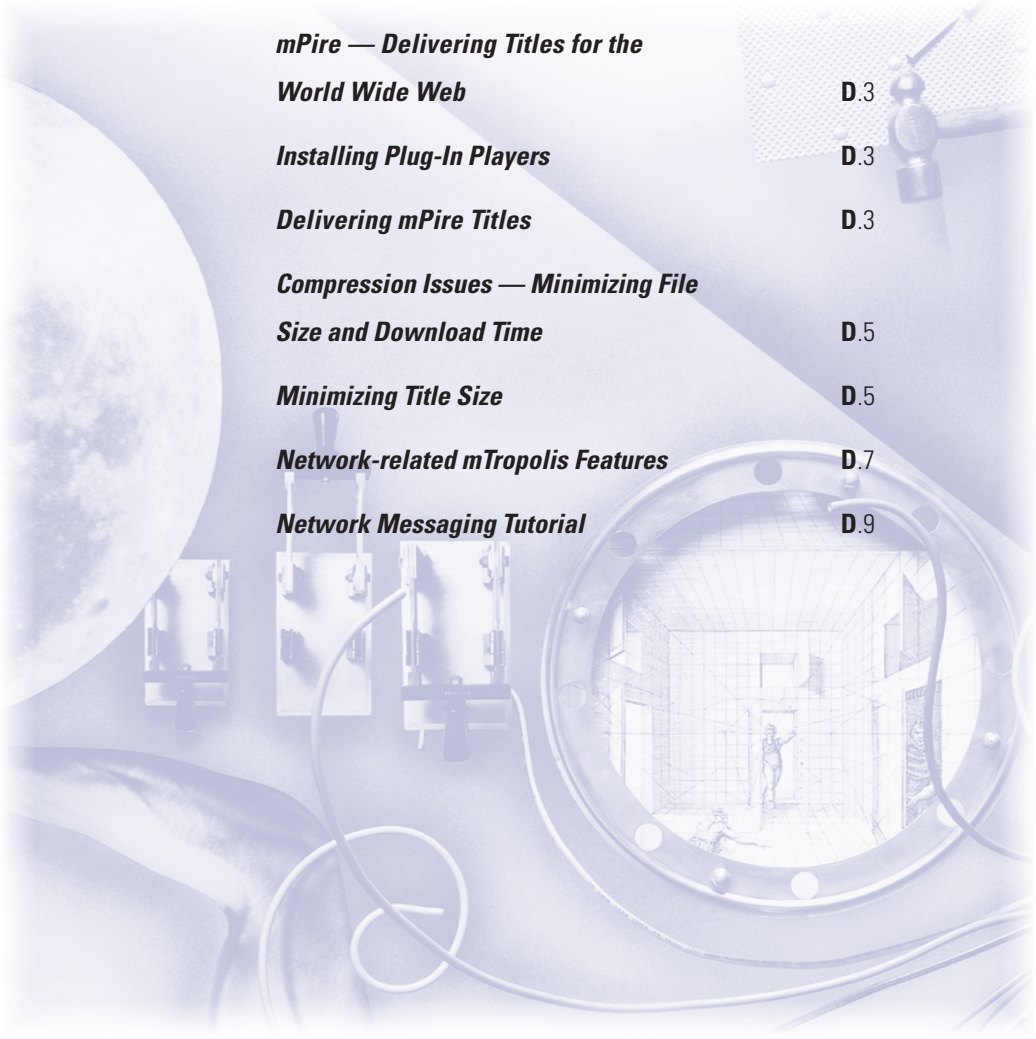
The following attributes (described in Chapter 15) only take effect in titles run on the Mac OS platform. They have no effect in titles run on Windows platforms:

- commandKeyDown
- gameMode
- monitorBitDepth (cannot be set)
- supportsBitDepth[*n*]
- textHeight (does not return accurate value)
- windowMaxSize
- windowMinSize
- windowPosition
- windowSize
- windowStyle
- windowTitle

Media file types

- Media files created on a Windows machine may not be recognized as valid media files by mTropolis. This is because all Macintosh files have a "resource fork" that contains special information about those files. Among other things, the resource fork contains information about a file's type and creator. Files transferred from a Windows machine usually do not have the correct file type. Utilities such as ResEdit, FileTyper, and Snitch can be used to change the file types of such files so that mTropolis correctly recognizes them.

Appendix D: Internet Authoring Issues



Appendix D: Internet Authoring Issues

This appendix describes mPire, the web-browser plug-in version of the mTropolis player and the mTropolis features that can be used to create network-enabled titles.

mPire — Delivering Titles for the World Wide Web

mPire is Quark's platform for delivering interactive multimedia titles over the World Wide Web. The mPire player is a Netscape Navigator browser plug-in that allows mTropolis titles to play inside a web page.

Installing Plug-In Players

End users must have the mPire plug-in installed in their web browsers to view mPire titles. Installation instructions and the mPire plug-in players themselves can be found on the mTropolis CD–ROM.

Delivering mPire Titles

The web server that will be used to deliver your mPire titles must be properly configured to handle the MIME types for mPire titles. Your HTTP server needs to support the following MIME type mappings:

The mPire MIME type is:

```
application/x-mpire
```

Any single-segment mTropolis title can be played with the mPire plug-in players. File extensions for mPire titles are:

- `.mfx` for cross-platform titles,
- `.mfm` for Mac OS-only titles, and
- `.mfw` for Windows-only titles.

Contact your system administrator to update the server. In some cases, end users can configure their own web directories to support new MIME types without having to change the global settings of the server. For example, NCSA servers can access a file called `.htaccess`. This file should contain the lines:

```
AddType application/x-mpire mfx
AddType application/x-mpire mfm
AddType application/x-mpire mfw
```

Other types of HTTP servers may use completely different syntax. Consult your server documentation or system administrator for details.

EMBED tag parameters

Use the HTML EMBED tag to place mPire titles on a web page. The basic syntax is:

```
<EMBED width="width" height="height"
src="file" logo="true|false">
```

where *width* is the width, in pixels, of the mPire title's draw area; *height* is the height, in pixels, of the title's draw area; and *file* is the URL of the mTropolis title file to be displayed. The optional logo tag can be set to either `"true"` (the default) or `"false"`. When this tag is set to true, an mPire logo appears (in the space allocated to the title) while the title is downloading. When the logo parameter is set to false, the logo does not appear.

Note that the mPire plug-in players look for files with the extensions `.mfx`, `.mfm`, and `.mfw`. Your file name should end in one of these three extensions (such as `mytitle.mfx`) depending upon the type of mTropolis title being distributed. See "mTropolis File Name Extensions and File Types" in Chapter 1 for more information about mTropolis title files and their file name extensions.

End user-defined parameters

In addition to the basic EMBED tag parameters, custom parameters can be added to the EMBED tag. These end user-defined parameters can have any HTML-acceptable name and can have any string value.

When the title is running, it can check for the presence of end user-defined parameters using the parameterCount, parameterName[*n*], and parameterValue[*n*] project attributes (described in Chapter 15).

For example, an mPire title could be added to a web page with the following EMBED tag, which adds 2 user-defined parameters:

```
<EMBED width="320" height="200"
src="mytitle.mfx" logo="true"
myParam1="ValueOne" myParam2="ValueTwo">
```

In this case, the title's `project.parameterCount` attribute would contain `6`, `project.parameterName[1]` would contain `"width"`, `project.parameterValue[1]` would contain `"320"`, `project.parameterName[6]` would contain `"myParam2"` and `project.parameterValue[6]` would contain `"valueTwo"`.

Presenting the proper title file with JavaScript

It is simplest to build mPire titles as cross-platform files (".mfx" files). Using cross-platform files allows a single mTropolis title file to work with both Mac OS and Windows browsers. However, because of cross-platform differences, you may want to distribute different Mac OS and Windows files. In this case, you'll need to configure your server or program your HTML pages to identify the type of browser and present the proper title file. The JavaScript shown below demonstrates one way of dealing with this problem.

```
 <SCRIPT language="JavaScript">
   <!-- This handy script figures out what version of the Netscape -->
   <!-- browser the user is running and delivers the right version -->
   <!-- of your mTropolis/mPire built title. -->

 <!-- Is this Navigator for Windows? -->
   if (navigator.appVersion.indexOf("Win") !=-1) {
   <!-- Change the EMBED tag below to work for YOUR Windows file -->
   document.write('<EMBED width="100" height="100" src="wintitle.mfw"><P>')}

<!-- Is this Navigator for Macintosh? -->
   else {if (navigator.appVersion.indexOf("Mac") !=-1) {
   <!-- Change the EMBED tag below to work for YOUR Macintosh file -->
   document.write('<EMBED width="100" height="100" src="mactitle.mfm"><P>')}

<!-- This is some other version. Too bad! -->
   else {document.write("Currently, mPire plug-ins are available only for Macintosh and Windows
   platforms! Sorry.")}
</SCRIPT>
```

If you are distributing separate Mac OS and Windows versions of a title (that is, you are not using the cross-platform title type), the following JavaScript can be used in your HTML page to deliver the appropriate version of your mPire title to end users, based upon which version of the Netscape browser they are using.

Version checking

If the mPire title file specified by the EMBED tag was created for a newer version of the mPire plug-in than the end user currently has installed, an alert appears. This alert informs the end user that the installed version of mPire is too old to display the file and directs him or her to the mFactory web site for the latest version.

Compression Issues — Minimizing File Size and Download Time

Delivering multimedia titles over networks (whether internal "intranets" or the Internet itself) is complicated by the fact that many end users connect to the network at slow speeds. Most dial-up (modem) internet connections are at 14.4 Kbps (thousands of bits per second), 28.8 Kbps, or 33.6 Kbps. While faster connections such as ISDN and cable modems are starting to come into wider use, it is still a good idea to design your mPire titles so that they will be as small as possible.

Smaller files download faster!

The smaller a file is, the faster that file can be downloaded over the network. Keep in mind that end users often grow impatient waiting for a web page to display. Authors must balance the complexity of a title versus the amount of time they think end users will wait. The following table illustrates the smallest amount of time that end users will have to wait to download a file based on their connection speed. Note that actual download times are often significantly longer than these theoretical minimum times.

The next section of this document describes specific authoring techniques for minimizing the size of your mPire titles.

Minimizing Title Size

The following techniques, listed by media type, can be used to minimize the size of your built titles. mTropolis programming takes up only a small portion of a built title file — reducing the size of your source media is the key to reducing the size of your built titles.

Still graphics

Minimize the bit-depth of graphic files. Downsample your source images to 4-bit (16-color) or 8-bit (256-color) PICT files. Because 8-bit graphics have only 256 colors in them, you'll have to

Type of File	File Size	14.4 Kbps	28.8 Kbps	ISDN 64 Kbps	T1 1.5 Mbps
Simple mToon	50 Kbytes	27 sec.	14 sec.	6 sec.	0.3 sec.
Small one-scene	250 Kbytes	2 min.19 sec.	1 min. 9 sec.	31 sec.	1.3 sec.
Project with small QuickTime movie	1.5 MB	14 min. 13 sec.	7 min. 7 sec.	3 min. 12 sec.	8 sec.
Complex multi-scene project	10 MB	95 min.	47 min.	21 min.	55 sec.

Minimum download times for typical mPire titles at different connection speeds

deal with color palette issues (see "Color Table Modifier" in Chapter 12).

When images can be reduced to a single color, don't create them in that color as 8-bit images. Instead, create 1-bit images and tint them using the Graphic modifier.

PICTs can be converted to one-cel mToons so that they can take advantage of various compression codecs. See below for more details.

Draw area size

When repurposing existing titles, consider changing the "Draw Area Size" project preference (**Edit ➛ Preferences ➛ Project**). For our original mPire example titles, we changed the draw area size to quarter-screen (320 by 240 pixels), and resized all media elements to match. This affects any positional logic (the hard-coded piece positions for the mPuzzle tutorial project). There are also implications for motion modifiers — Path Motions must be redrawn, Vector Motions should generally be slowed down (especially if an object bounces around on screen), and Simple Motions should also be slowed down. For our examples, we also reduced the rate for mToons by a small amount.

Note that you must also resize any media in the project to match. Be sure to resize the source images — don't just resize the elements in mTropolis. If you don't resize your sample media, the title will look smaller on the screen, but assets will take up the same amount of space in the built title file.

mToons

Convert PICTs to single-cel, compressed mToons.

To resize an mToon, you must resize the source images used to create the mToon and then recompile the mToon in the mToon editor. Note that you should never throw away source files for mToos, since you can't resize a finished mToon — you have to resize its source and recreate it.

Compress all mToons with the mFactory Animation codec, the only cross-platform codec for mToons in mTropolis. You can either create your mToons with this compression setting or leave them uncompressed and mTropolis can compress them at build time. By default, the "mToon Conversion Options" in the **Build Title** dialog box are set to convert uncompressed, 8-bit mToons to mFactory Animation compression. Note that you can also compress 16-bit mToons with this codec (in lossy or lossless fashion — see the codec descriptions under "Compression" in Chapter 1).

Sound

Minimize the bit-depth and sample rates of sounds. Resample sounds to the lowest acceptable quality. An 8-bit/11 KHz/mono sound is four times smaller than an 8-bit/22 KHz/stereo sound and 16 times smaller than 16-bit/44 KHz/stereo.

Crop looping sounds to the shortest acceptable length, altering your definition of "acceptable" to fit the bandwidth.

Crop any silence from the beginnings and ends of sounds. If sounds have large silent passages inside, cut them out and use the **Pause** and **Unpause** commands, or split the sound into multiple sounds, controlling playback with timers and/or the Media Cue Messenger (see "Media Cue Messenger" in Chapter 12).

Use MIDI files (played with the MIDI modifier, see Chapter 12) instead of AIFF sound files for music. MIDI files are much smaller than audio files. The MIDI modifier can also be used to play single notes as sound effects.

Video

Consider recompressing, resizing, and reducing the frame rate of video files to get the very lowest acceptable data rate.

Experiment with different codecs to see which delivers the lowest data rate with best quality for your specific video files.

If the video has sound, reduce its quality to 8-bit, 11Khz, mono. On Mac OS, QuickTime sound tracks can be compressed using various algorithms such as IMA, uLaw, and MACE.

Text

Use text elements or 1-bit graphic text instead of graphic files. For "live" text, you'll have to deal with display differences and cross-platform font issues. Bitmapped text doesn't take up that much space, though. Unless you've applied multiple colors to different parts of a text element (using the **Format** menu), text with the "Convert Text to Bitmap" option checked is written to the built title file as a 1-bit bitmap.

Other bandwidth-saving tips

The various ink effects in the Graphic modifier can be used to create many interesting effects with low-bandwidth images. However, some of the effects work only on Mac OS.

Use the Graphic modifier and Image Effect modifier to create shapes, buttons, and other objects in your title when that level of simplicity is acceptable. In this way, nice looking buttons can be created without using any source media at all.

Instead of distributing a multi-scene title as a single file, consider breaking the title up by scenes into individual titles (if possible). Put each resulting title onto its own web page and use the Open URL modifier to navigate through them.

Mac OS-only tips

Use any QuickTime codec to compress your mToons. By using low quality settings, you can make your graphics as small as you want. This is an excellent way to achieve small title sizes. Unfortunately, mTropolis Windows titles cannot access the QuickTime codecs in this way. Only the mFactory Animation compressor is available in Windows titles.

Replace your AIFF assets with QuickTime assets, which can contain compressed sound (4:1 IMA, uLaw, or MACE). The current version of QuickTime for Windows does not support compressed sound.

Replace AIFF music with QuickTime, which can contain music (MIDI) tracks.

Network-related mTropolis Features

The following mTropolis features are useful in the creation of titles for use with the mPire player or stand-alone titles that use network messaging.

mTropolis network modifiers

Complete descriptions of modifiers that enable and work with the network functionality in mTropolis can be found in Chapter 12, "Modifier Reference." Relevant modifiers are:

- Cache URL modifier: This mPire-only modifier reads data from a URL and stores it in the web browser's cache.
- Fetch Net Text modifier: This modifier retrieves text from a URL and stores it in a String Variable. It can be used in any mTropolis title.
- Net Messaging Service: This modifier enables mTropolis' network messaging features. It must be enabled before using the Net Messenger, Fetch Net Text modifier, or Post Net Text modifier.
- Net Messenger: This messenger modifier can send mTropolis messages over a network where they can be received by other mTropolis titles.
- Open URL modifier: This modifier opens a specified URL within the web browser.
- Post Net Text modifier: This modifier can send text to a specified URL. It can be used in any mTropolis title.

Cross-platform titles and color mapping options

The default format for mTropolis built title files is to create a "cross-platform" title. Such titles can be played on either Mac OS or Windows platforms. As such, they are a good choice for distribution as mPire titles because either type of computer can view the same title. This option is described in more detail in the description of the **File ➛ Build Title** menu option, found in Chapter 1, "File Menu."

When a cross-platform title is being built, there is a "Re-map 8-Bit Images for mPire Plugin" option that can be selected. Marking this check box automatically re-maps any 8-bit images in the project to the "web-safe" Netscape Navigator color palette. This is very useful if you are creating 256 color titles on machines with 256 color (8-bit) displays. This option also replaces all the color table assets in the resulting built title with the Netscape Navigator palette, thereby ensuring that any Color Table modifiers used in the project will have no effect.

Plug-in related messages

The Plug-in Received Focus and Plug-in Lost Focus messages (Chapter 13) can be used to identify when the end user starts and stops manipulating an mPire title.

Network messaging and network operation messages

Most of the network modifiers send messages that indicate their status as the modifier executes. These messages can be found in the Modifier Messages submenu of the **Message/Command** pop-up menu. These messages are described in detail in "Modifier Messages" in Chapter 13.

PlayerType attribute

The playerType system attribute, described in Chapter 15, can be used to tell if the current title is running in the mPire plug-in, a standalone mTropolis player, or the mTropolis editor.

Network Messaging Tutorial

A network messaging tutorial can be found in Chapter 8 of the *mTropolis Developer Guide,* "Network Messaging Tutorial — Avatar Chat." This tutorial demonstrates the techniques used to build a mTropolis title that sends messages to another title over the network.

Glossary

Alias

An alias is a special copy of a modifier that takes its functionality from the modifier from which it was made. This “master copy” is automatically placed in the **Alias** palette when the alias is made. Additional aliases that refer to this master copy can also be created and placed throughout a project.

The advantage of using aliases is that they can be globally updated from a single source. Changing the settings of an alias changes the settings of both the master copy and all other aliases that refer to the same original throughout the project.

Asset

A media file that has been linked to a project is called an asset. mTropolis compatible files include: PICT still images, AIFF sounds, SND sounds, mToon animations (mTropolis animation format), CLUT color tables, QuickTime video, and QuickTime VR panoramas.

Attribute

A data value that describes the current state of a mTropolis object. For example, graphic elements have attributes that describe their height, width, and layer order number (along with many others).

Author Message

A message that has been created by the author of a mTropolis project is called an author message. Author messages can be sent from messengers in the same way that mTropolis built-in messages and commands can be sent.

Behavior

The Behavior modifier is a special modifier that can be used to encapsulate other mTropolis modifiers. Behaviors can be made “switchable” so that they activate and deactivate upon the receipt of messages. Behaviors allow complex interactions to be created and organized.

Cel

A single frame of an mToon animation.

Child

All components directly contained by another component are considered its children. For example, a Graphic modifier and an Element Transition modifier contained by the same element are children of that element.

Commands

Commands are a special type of message that act directly upon an element or the element's linked media. Unlike other mTropolis messages and author messages, commands act only upon the element to which they are sent. See also Element in Chapter 13.

Component

Any discrete part of a mTropolis project is called a component. Components include structural organizers (such as the project, section, and subsection components), media containers (such as scenes, graphic, and text elements), modifiers (such as messengers, Graphic modifiers, etc.), and non-structural objects (such as the WorldManager, System, and AssetManager components). Components are also referred to as objects.

Descendant

All components contained by another component are called descendants of that container. For example, all items in a project are the descendants of the project component.

Element

Elements are mTropolis components that contain media assets. Scene components are a special type of element that both contain media and act as structural components. Double-click an element to display its **Element Info** dialog box that can be used to configure its default display properties.

Environment Message

Environment messages are the messages that mTropolis sends to the components of a project when it detects changes in the runtime environment. For example, elements are informed of end user mouse clicks by the Mouse Up and Mouse Down environment messages.

Hierarchy

A basic organizational principle in the mTropolis authoring environment is the hierarchy. mTropolis projects have a hierarchical structure composed of sections, subsections, scenes, and elements. A section acts as a parent of its subsections, a subsection acts as the parent of its scenes, and a scene acts as the parent of its elements. Elements can also be parents of other elements. This structure helps organize a project, and provides a pathway for message passing through the project.

Ink

The Graphic modifier's "ink" setting controls how an element's media is displayed on the screen. For example, some ink effects mix the foreground and background colors with the element's color, effectively tinting the image.

Layer Order

Layer order is the order in which elements are drawn on the screen. When new elements are added to a scene, they draw on top of previous elements. mTropolis keeps an internal list of elements in a scene and the order in which they are drawn. This layer order can be changed using tools in the authoring environment. Drawing elements on top of each other creates the illusion that the two-dimensional (X,Y) screen has a depth dimension (Z). This effect is sometimes called "2.5D."

Library

A library is a separate file where mTropolis components can be stored. Libraries can be used to distribute portions of a project to development team members for editing, or to archive project components for reuse in other projects. Within the mTropolis environment, an open library is represented as a palette. With the exception of entire projects, any component can be dragged and dropped to or from a library palette.

Linking

Linking is the procedure used to establish a path from a mTropolis project to external media files. Media that has been linked to a project appears in the **Asset** palette. Once linked, media can be dragged and dropped into the project in any view.

Messages

Messages are the means by which the components of a mTropolis project communicate. Messages are signals that are used to activate or deactivate modifiers in a project. Both the mTropolis engine and messenger modifiers pass messages among components.

Miniscript

Miniscript is a simple scripting language embedded in a modifier. Its syntax is loosely based upon AppleScript, Apple's scripting language. Miniscript statements are executed when the Miniscript modifier that contains them is activated by a message. Miniscript can be used to perform mathematical operations, send messages based on conditional expressions, and send multiple messages or commands from a single modifier.

Modifier

Modifiers are mTropolis components that imbue other mTropolis components with new capabilities or properties. Modifiers are used by dragging and dropping them onto elements. The properties of a modifier become a part of the component on which they are placed. For example, adding the Drag Motion modifier to an element causes that element to become draggable by end users in run-time mode. Double-click modifiers to display their configuration dialog boxes that control modifier options.

MOM

The mFactory Object Model (MOM) is an application programming interface for multimedia programmers who want to extend the power of mTropolis by writing their own modifiers and mTropolis features in languages such as C or C++.

mToon

An mToon is an animation file in Quark's proprietary animation format. A series of individual cels (that is, images from PICT, PICS, QuickTime, or existing mToons) can be compiled into a single mToon file by the mTropolis mToon editor.

mToons can be made to play in their entirety, display a single cel, or play specified ranges of cels. This precise control over mToons makes them especially useful for animated sequences. For example, a character's sitting, walking, and running motions can be compiled into a single mToon linked to an element. The mToon then has a "library" of actions that can be individually accessed and played back.

Object

Any discrete part of a mTropolis project is called an object or component.

Palette

A floating menu containing project tools or resources is called a palette. For example, the **Asset** palette contains the media assets that have been linked to a project.

Parent

The component that directly contains another component is referred to as its "par-

ent." All components in a project (with the exception of the project component itself and non-structural components such as the WorldManager) have one parent. For example, a project is the parent of sections, a section is the parent of its subsections, a subsection is the parent of its scenes, and a scene is the parent of its elements. An element can also be the parent of other elements.

Project

The project is the mTropolis component that holds an entire title within it. The children of a project are sections. A project can contain modifiers that modify the actions or characteristics of the entire title. Titles created in the mTropolis authoring environment are referred to as projects during the development phase. Projects cannot contain other projects. The saved file that represents a mTropolis title in development is called a project file. The name of this file is the name assigned to the project component.

Scene

The structural component one level below a subsection. Scenes can contain elements that are visible to the end user in run-time mode. The elements that are added to a scene draw on top of it and are attached to it. Deleting the scene deletes all the elements and modifiers attached to it. Scenes are like elements in that they can also be linked to media.

Scope

Where a modifier is placed in a project determines its scope — that is, the group of objects that may see it and that can access its value in mTropolis. All descendents of a variable modifier's parent are in its scope.

Section

Section components can be used to organize parts of a title into groups that logically belong together. For example, a title developer for a travel game might put everything to do with Africa under a single section. Sections are parents to subsections. As with a project, a section can contain modifiers. Sections cannot contain other sections.

Shared Scene

Shared scenes are special scenes that appear behind all other scenes in a subsection. Shared scenes can be used to store elements and modifiers common to all the scenes in a subsection. For example, a background image placed on the shared scene forms the backdrop of all subsequent scenes. Elements and modifiers, such as navigational buttons, that are common to all scenes in a section are placed in the shared scene. By default, the very first scene in a subsection is the shared scene.

Sibling

All components that share the same parent are referred to as siblings. For example, a Graphic modifier and an Element Transition modifier contained by the same element are siblings.

Structure

The structure of a project is literally the way the project is organized. mTropolis projects consist of structural components, elements, and modifiers that interact through a messaging system. mTropolis provides a Structure window for examining the hierarchical structure of a project.

Subsection

Subsections are components that can be used to more finely divide the content of a section, much like a subsection in a book. Subsections are the parents of scenes. As with the project and section components, a subsection can contain modifiers. Subsections cannot contain other subsections or sections.

Title

When a mTropolis project is complete, it can be compiled into a playback version called a title using the **Build Title** menu option. See also Project in Chapter 1.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

T

U

V

W

Y

Z


mFactory™


103692