



An
Introduction
To QuickTime

Session 17a

QUICKTIME QUICK START	5
OVERVIEW	5
ARCHITECTURE	6
MOVIE TOOLBOX	6
IMAGE COMPRESSION MANAGER	7
COMPONENTS	7
QUICKTIME'S BUILT-IN DATA PROCESSING COMPONENTS	7
QUICKTIME BUILT-IN MEDIA HANDLING COMPONENTS	8
QUICKTIME'S BUILT-IN UTILITIES	8
REFERENCES	9
INTRODUCTION TO QUICKTIME	9
MOVIE TOOLBOX	9
COMPONENT MANAGER	9
IMAGE COMPRESSION MANAGER	9
MEDIA HANDLERS	9
 MOVIES	 10
THE MOVIE	10
MOVIE TIME - TIME COORDINATE SYSTEM	11
TIME SCALE	12
MEDIA TIME SCALE	13
PLAYBACK TIME BASE	14
MOVIE STRUCTURE	15
TRACKS AND MEDIA	17
MEDIA REFERENCES	20
DATA REFERENCES	21
GRAPHIC TRANSFORMATION	22
CLIPPING	23
MATRIX	24
LAYERING, TRACK ENABLING AND QUALITY	25
GRAPHICS MODES	26
SOUND TRACKS	28
VOLUME	28
BALANCE	29
REFERENCES	29
MOVIE TOOLBOX	29
SUPPORTED QUICKTIME MEDIA TYPES	29
TRANSFORMATION MATRIX	29
INSIDE MACINTOSH: FILES	29
INSIDE MACINTOSH: MEMORY	29
TECHNOTE 1195 : TAGGED HANDLE DATA REFERENCES IN QUICKTIME 4	29
INTRODUCTION TO SOUND	29
 ASYNCHRONOUS MOVIE LOADING	 30
NON-ASYNCHRONOUS LOAD MODEL:	30
ASYNCHRONOUS LOAD MODEL:	30
REFERENCES	31
QUICKTIME 4.1 – INTRODUCTION TO ASYNCHRONOUS MOVIE LOADING	31
 FAST-START PROGRESSIVE DOWNLOADS	 31
REFERENCES	31
SUPPORT FOR PROGRESSIVE DOWNLOADS	31
 STREAMING	 32
OVERVIEW	32

APPLICATION CONSIDERATIONS	33
TRACK STRUCTURE	34
PREROLL AND PRE-PREROLL	34
REACTING TO CHANGES IN MOVIE CHARACTERISTICS	35
DURATION CHANGES	35
SOUND AND VIDEO CHANGES	36
OTHER PLAYBACK CONSIDERATIONS	36
REFERENCES	36
STREAMING	36
CUSTOMIZING MOVIE CONTROLLERS	36
COMPONENTS	36
THE COMPONENT MANAGER	37
COMPONENTS TYPES	39
REFERENCES	41
COMPONENT MANAGER	41
GRAPHICS IMPORTER / EXPORTER COMPONENTS	41
GRAPHICS IMPORTERS	41
ALPHA CHANNELS	41
MULTIPLE IMAGES	42
GRAPHICS EXPORTERS	42
REFERENCES	43
GRAPHICS IMPORTER COMPONENTS	43
GRAPHICS EXPORTER COMPONENTS	43
EXPORTING IMAGES – DISPATCH 14	43
MOVIE DATA EXCHANGE COMPONENTS	43
IMPORTING AND EXPORTING MOVIE DATA	44
REFERENCES	45
MOVIE DATA EXCHANGE COMPONENTS	45
COMPONENT MANGER	45
FINDING MOVIE EXPORT COMPONENTS – DISPATCH 6	45
THE SEQUENCE GRABBER	45
SEQUENCE GRABBER COMPONENTS	45
PREVIEWING	47
RECORDING	47
CHANNEL COMPONENTS	47
PANEL COMPONENTS	47
REFERENCES	48
SEQUENCE GRABBER	48
SEQUENCE GRABBER CHANNEL COMPONENTS	48
SEQUENCE GRABBER PANEL COMPONENTS	48
VIDEO DIGITIZERS	48
VIDEO OUTPUT COMPONENTS	48
REFERENCES	49
VIDEO OUTPUT COMPONENTS	49
QUICKTIME ATOM CONTAINERS	49
QT ATOM HIERARCHIES	50
REFERENCES	51
QUICKTIME ATOMS	51
MOVIE TOOLBOX:DATA TYPES	51

QTATOMCONTAINER-BASED DATA STRUCTURE DESCRIPTIONS	51
SPRITES	51
MOVIE SPRITES	52
REFERENCES	53
INTRODUCTION TO WIRED MOVIES, SPRITES, AND THE SPRITE TOOLBOX	53
EFFECTS	53
VIDEO EFFECTS IN MOVIES	54
EFFECTS OUTSIDE MOVIES	55
REFERENCES	56
BUILT-IN QUICKTIME VIDEO EFFECTS	56
INTRODUCTION TO QUICKTIME VIDEO EFFECTS	56
CROSS PLATFORM DEVELOPMENT	56
IMPLEMENTATION	56
CROSSPLATFORM ISSUES	56
ENDIAN ISSUES	57
REFERENCES	58
QUICKTIME FOR WINDOWS PROGRAMMERS	58
MAC OS FOR QUICKTIME PROGRAMMERS	58
QUICKTIME FOR JAVA	58
DEVELOPER RESOURCES	59
QUICKTIME DEVELOPERS WEB SITE	59
QUICKTIME WEB SITE	59
APPLE QUICKTIME API DOCUMENTATION	59
SAMPLE CODE	59
QUICKTIME SOFTWARE DEVELOPMENT KIT (SDK)	60
LETTERS FROM THE ICE FLOW	60
QUICKTIME DEVELOPER SERIES BOOKS	60
NEWSGROUPS & MAILING LISTS	61
QUICKTIME TOOLS	61
3 RD PARTY TOOLS	62
ADC	62

QUICKTIME QUICK START

OVERVIEW

QuickTime is a cross platform system-level software package for Macintosh, Windows and Java™ which adds the capability to play movies, synthesize music, display animations, view virtual reality worlds and add multimedia capability to the computer desktop.

QuickTime is implemented as a set of extensions on the Macintosh platform and a dynamic-link library (DLL) on Windows. It can process video data, still images, animated images (also known as sprites), vector graphics, multiple sound channels, MIDI music, 3D objects, virtual reality objects, panoramas and text. The number of data formats QuickTime recognizes is impressive. Currently, more than 70 different formats can be imported or exported and as formats are added, applications created today will work with them automatically.

QuickTime is readily extensible. It is built in a modular fashion made up of many software components installed and accessed through the **Component Manager**. These built-in components handle the most common multimedia tasks. Developers can expand on these capabilities by writing custom components to augmented or completely replace QuickTime's capabilities if desired.

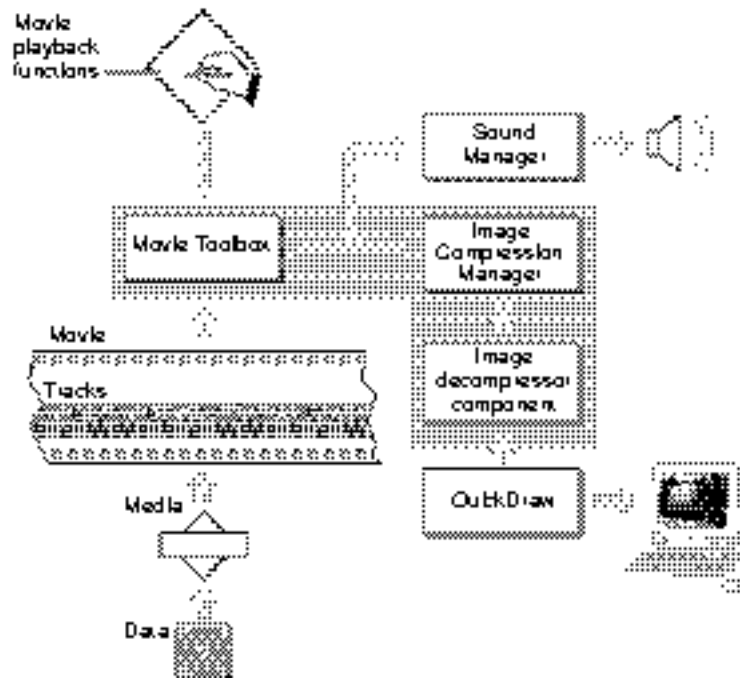
An important concept to remember is the idea of "time-based media" and how it can be manipulated. QuickTime is a generalized way to define time lines and organize information along these time lines.

The QuickTime API is large to say the least, and attempting to understand it all is a daunting task. However, QuickTime's modularity allows you to do many things easily and quickly with as little as a half dozen calls! For example, you can selectively work with the **Movie Toolbox** alone or **Graphics Importers** without needing to visualize all of the underlying components or API calls. You don't need to understand it all to start working with it.

ARCHITECTURE

QuickTime comprises two main managers—the **Movie Toolbox** and the **Image Compression Manager**—plus a set of built-in components.

The diagram shows the relationships between these managers and components for an application playing a movie.



MOVIE TOOLBOX

Applications gain access to the capabilities of QuickTime by calling functions in the Movie Toolbox. The Movie Toolbox lets an application store, retrieve, and manipulate time-based data and provides functions for editing movies.

IMAGE COMPRESSION MANAGER

Single image data may require a large amount of storage space. A sequence of images, like those contained in a QuickTime movie can demand many times as much space. The Image Compression Manager provides a device-independent and driver-independent means of compressing and decompressing images and sequences of images, thereby minimizing the storage requirements for any application that works with images.

In most cases applications use the Image Compression Manager indirectly by calling Movie Toolbox functions or by displaying a compressed picture. However, if an application compresses images or makes movies with compressed images, it may call some Image Compression Manager functions directly.

COMPONENTS

Apple ships a number of built-in components with QuickTime. These built-in components provide essential services to applications and to the managers that make up the QuickTime architecture. The Apple-defined component types include image processors, media handlers, and miscellaneous utilities.

QUICKTIME'S BUILT-IN DATA PROCESSING COMPONENTS PERFORM THESE TASKS:

- **Movie controller components** let applications play movies through a standard user interface.
- **Image compressor components** compress and decompress image data.
- **Image compression dialog components** let the user specify the parameters for compression operations.
- **Image transcoder components** convert compressed files from one format to another.
- **Video digitizer components** let applications control video digitization by external devices.

- **Movie data-exchange components** (also known as Movie Import and Movie Export components) let applications move various types of data into and out of QuickTime movies.
- **Video output components** convert QuickTime movies into video streams.
- **Graphics import components** let applications work with still image files by providing a simple API that works with a wide variety of image file formats.
- **Graphics export components** let applications export still image files using a standard easy to use API.
- **Music components** process and synthesize music tracks in QuickTime movies.
- **Effects and transitions components** implement video synthesis, video filters and video transitions, including the 133 standard SMPTE transitions. These components are implemented as a subcategory of image decompressor components.
- **Preview components** are used by the Movie Toolbox's standard file preview functions to display and create visual previews of file contents.

QUICKTIME BUILT-IN MEDIA HANDLING COMPONENTS:

Media handler components implement the behavior of different track types. They deal with the specifics of how a particular track's media data should be presented to the viewer.

- **Video media handler Component** implements video tracks by calling the Image Compression Manager to display video samples.
- **Sound media handler Component** implements sound tracks by calling the Sound Manager to play sound samples.

QUICKTIME'S BUILT-IN UTILITIES PERFORM THESE TASKS:

- **Clock components** provide timing services for applications that use QuickTime.

- **Standard Sound component** lets an application present the user with a dialog for configuring sound settings.
- **Sequence grabber components** let applications preview and record video and sound data as QuickTime movies.
- **Sequence grabber channel components** handle the acquisition of individual data streams for sequence grabber components. For example, the video sequence grabber channel component acquires video.
- **Text channel components** are sequence grabber channel components for text.
- **Sequence grabber panel components** let sequence grabber components obtain configuration information from the user for a particular sequence grabber channel component.

REFERENCES

INTRODUCTION TO QUICKTIME

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/INMAC/INTROS/xxIntroductions.htm>

MOVIE TOOLBOX

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmMTFundamentals.htm>

COMPONENT MANAGER

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmComponentMgr.htm>

IMAGE COMPRESSION MANAGER

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmImageCompMgr.htm>

MEDIA HANDLERS

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmMHFundamentals.htm>

MOVIES

THE MOVIE

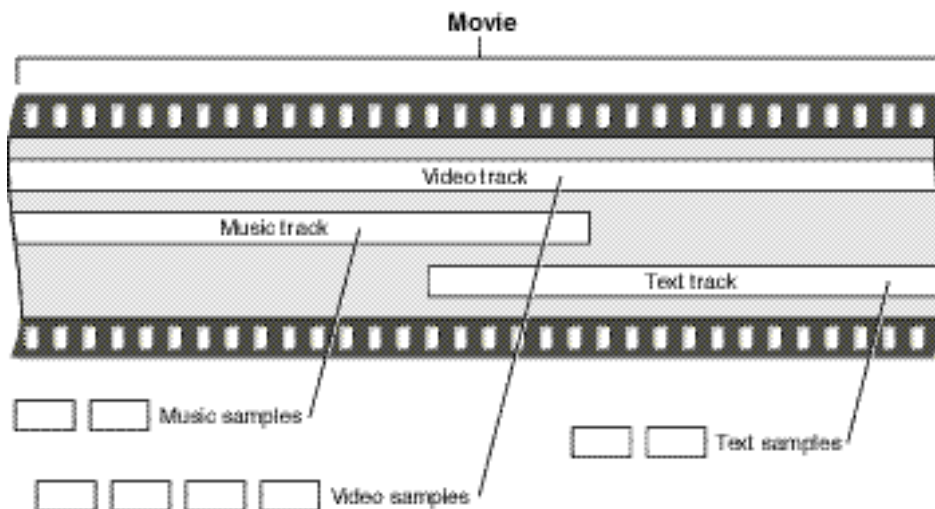
QuickTime uses the metaphor of a movie to describe time-based data. Any time-based data can be organized as a movie (audio, video or both). Movies are containers that hold all of the information needed to organize data in time, but not the data itself.



The term container is not a specific or special term in QuickTime, but is used in the general sense suggesting a place where data is stored and accessed.

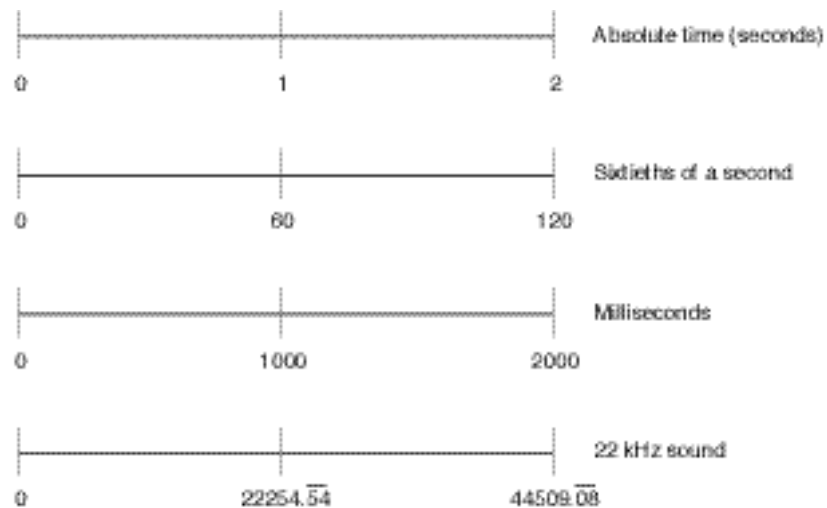
Movies are made up of data streams called **Tracks** and each track references and organizes a sequence of data of the same type in a time-ordered way. Tracks contain Media Structures that reference the actual data. Media is organized within a track and chunks of media data are called media samples.

A movie file typically contains a movie structure and its media bundled together so you can download or transport everything together. Movies can also contain references to media not stored locally, for example a URL. You can combine these together, having some data references to local media and other references to media stored elsewhere.

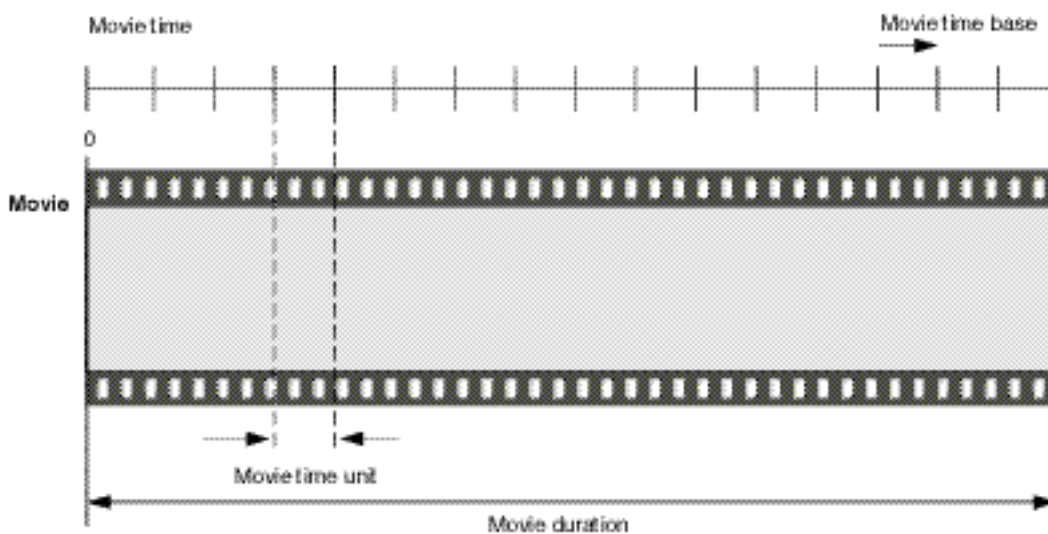


MOVIE TIME - TIME COORDINATE SYSTEM

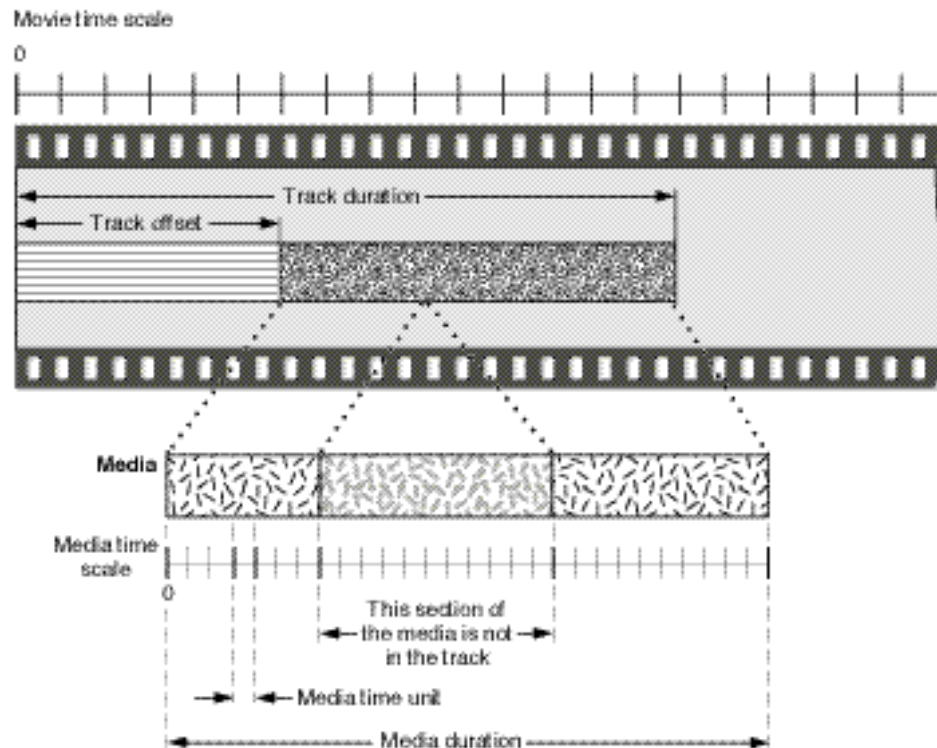
QuickTime movies organize media along the time dimension. To manage this time dimension QuickTime defines a time coordinate system. This coordinate system locks movies and media data structures to a common measurement, the second. Each time coordinate system establishes a time scale and this time scale establishes the translation between real time and movie time. Time scales are marked in time units of so many units per second.



The time coordinate system also defines duration. This is the length of a movie or media structure in terms of time units. Therefore, a particular time in a movie can be identified by the number of time units elapsed to that point.



Each track in a movie contains a time offset and duration. These attributes determine when a track begins playing and for how long. Each media structure also has its own time scale, which determines the default time units for data samples of that media type.



TIME SCALE

The time scale is the master ruler for a movie. Every event within a movie is measured and located by the movie's time scale and this time scale is expressed in so many units per second. The duration of a movie element is the number of time scale units from its beginning to its end. Each track in a movie also has an offset that is specified in time scale units. This offset is the beginning point for the track. A track that begins when the movie begins has an offset of 0.

Every movie has a time coordinate system, but these systems may differ from movie to movie.

The time scale in a movie's time coordinate system should be a convenient number of fractions of a second. The scale number should be one that makes it easy to translate movie times into other time scales.

A movie time scale of 600 (the default time scale for a newly created movie) can translate a playback rate of 24 frames per second. With a time scale of 600 and a playback rate of 24 frames per second, each frame would be displayed for 25 time scale units ($600 / 24 \text{ frames} = 25$). In this same way 600 works well for 30, 50 and 60 frames per second. Media with high sample rates (such as digital audio, which can have a sample rate such as 44,100) are specified as so many samples per time scale unit.

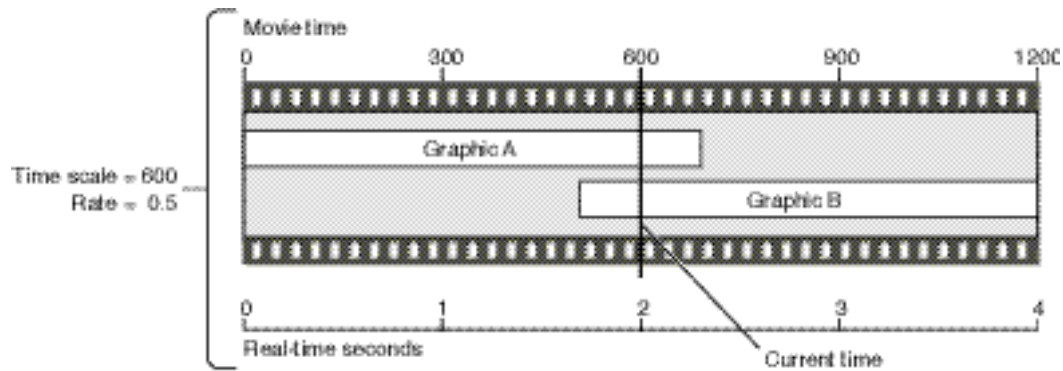
A movie's duration is therefore the total number of its time scale units from end to end. A track in a movie can have smaller duration if it doesn't extend to the end of the movie, and media chunks that tracks reference usually have even shorter duration's.

MEDIA TIME SCALE

Each chunk of media that is referenced by a track has its own time scale determined by its sample rate. QuickTime will automatically translate between a movie's time scale and the time scales of its various media; an application doesn't need to do this.

For example, consider a movie containing a single video track, a single audio track and a text track. The movie has a duration of 2 seconds and a default time scale of 600. The video track starts at the beginning of the movie (offset 0) and goes to the end (duration 1200), at 30 frames per second (a media time scale of 30). The audio track starts at the beginning and goes to the end (offset 0, duration 1200), at 44,100 samples per second (media time scale of 44100). The text track contains one single sample, the title that appears for 0.25 seconds into the movie and lasts for 1 second (offset 150, duration 600, media time scale 1).

If an edit action is performed in the middle of the movie (for example a copy operation) 0.5 seconds in duration (300 in movie time scale), QuickTime would correctly copy each track using the appropriate medial time scale of 15 video frames, 22,050 audio samples and 1 text sample.



The above diagram outlines a movie with a time scale of 600 running at a Rate of 0.5. The movie's duration is 1200 and contains a transition between two tracks (Graphic A and Graphic B) at time 600.

Because the movie's duration is 1200 with a time scale of 600 it would normally take 2 seconds to play (Rate = 1). However, the playback time base of the example is 0.5, so the movie is running at half speed and will play for 4 seconds.

The current time shown is 600, so the transition from Graphic A to Graphic B is in progress. Because the playback rate is 0.5, it has taken 2 real-time seconds to reach this current time. The viewer has seen 2 seconds of A and will now see 2 seconds of B.

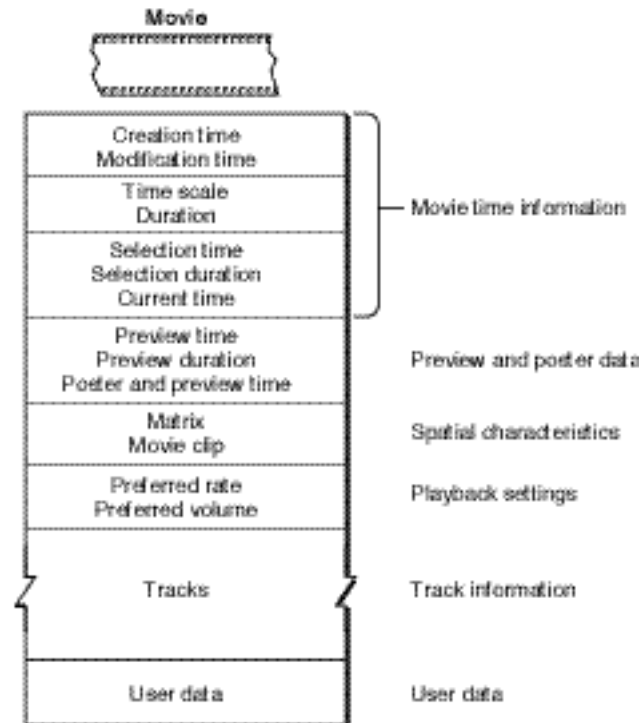
MOVIE STRUCTURE

Movies act as QuickTime's central data structure and as this concept evolved, the movie structure acquired a number of basic elements:

- There is only one movie structure with one set of software tools to create, edit, and run movies.
- The movie is a container and does not contain any media. Instead, movies contain references, which tell QuickTime how to find chunks of media elsewhere.
- These references are organized according to a master timeline.
- Movies may contain multiple tracks. Each track can access and present a specific kind of media. Therefore, movies can present several different media chunks at the same time.

A movie contains general information about handling its media (such as decompression parameters) and QuickTime uses this information as a basis for selecting the most effective processing technique.

Movies also contain fields that perform global housekeeping roles for the whole movie.



A movies global information includes:

- Creation date and time and last modification date and time.
- Movies time scale and duration.
- The current time - This changes as the movie plays. When a movie is saved to a movie file the current time is also saved and the movie will start playing where it left off the next time it's viewed.
- Selection information - This information is saved from the last time the movie was edited. Selection information consists of a start time and duration for the current selection and is expressed in movie time scale units.

- Active movie segment - This is defined by a start time and a duration in movie time scale units. The default active segment is the entire movie, but an application can set it to some other smaller region if limiting the movies playback is required. QuickTime will not process the movie outside the active segment unless instructed to do so.
- Movies preview start time and duration - You can set the preview to be any short segment of the movie.
- The current time that defines the movies poster - This is a "frame shot" from a movie to be used as a poster. A movie poster is used as its default preview if a preview has not been set.
- The transformation matrix and clipping region for the movie - These parameters control how QuickTime displays the visual elements in a movie.
- The preferred rate and preferred volume for movie playback - QuickTime will use these parameters unless overridden by an application.
- A general area for user data - This data can be any mixture of text and binary information. You can use this field to store copyright, credits etc.

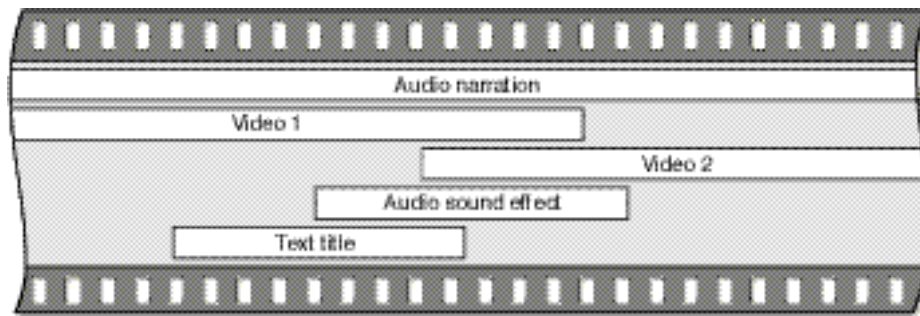
QuickTime movies and tracks are private data structures; you don't access their fields directly.

TRACKS AND MEDIA

The audiovisual parts of a QuickTime movie are its tracks. Each enabled track tells QuickTime how to fetch and present chunks of digital media.

Tracks are a familiar concept from audio technology where tracks can be mixed and overlapped to create different sound sequences. These sound sequences can then create a single sound output. QuickTime uses tracks in movies to combine data of many different types into a single user experience.

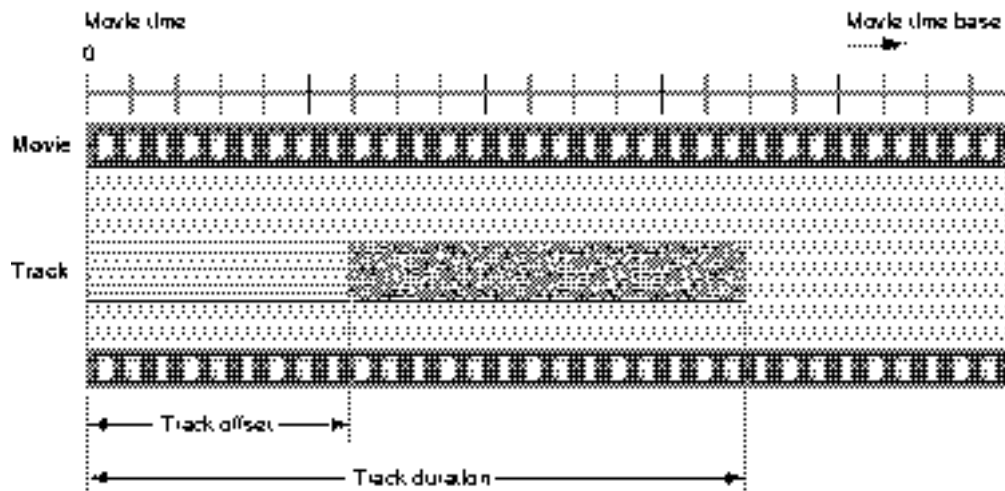
A movie may contain many tracks, each track accessing one media type. The type of media determines the type of track, which can range over all the data types QuickTime handles.



All the tracks in a movie use the movie's time coordinate system. The media the tracks access may use various time scales. For example, a video track may access 30 frames a second while an audio track may access 22,050 samples per second. QuickTime takes care of the coordination between the media time scale and the movie time scale.

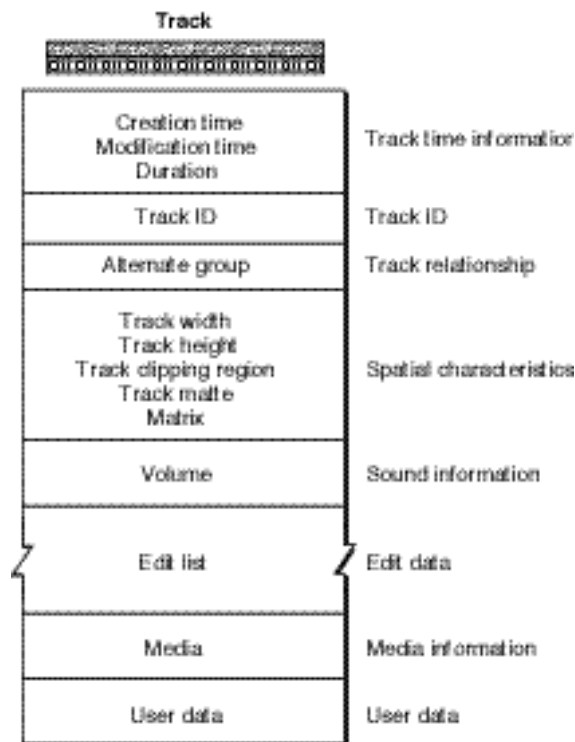
Each track begins at the beginning of the movie, but the tracks data might not begin until some time other than 0. This is called the track offset. The initial blank time is represented in an audio track by silence and in a video track by no image.

Tracks have a duration and each track may have a different duration. Some of these duration's may be shorter than the movie's duration. A movie's duration will always equal the duration of the longest track.



Tracks contain a list of references, which identify portions of the media the track uses. These references are called edit lists. This allows flexible access to media; a track can play the underlying media data in any order and any number of times.

Tracks also store several fields of housekeeping information specific to the track itself.

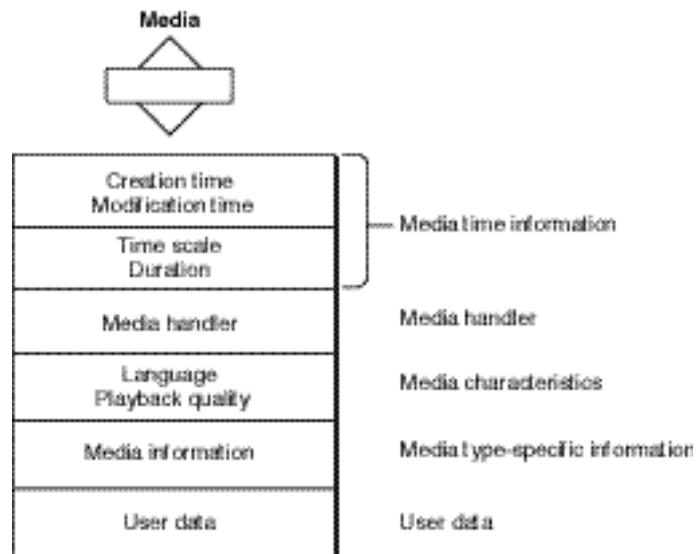


This information includes:

- Creation date and time and last modification date and time.
- A permanently assigned Track ID - an application can use this to locate the track.
- An optional alternate group ID - can be used to construct a movie with alternate tracks of the same media type. For example, multiple sound tracks in a different language. Giving these tracks a different track ID but the same alternate group ID allows QuickTime to easily relate tracks together.
- Visual presentation information if the track is for visual media - This consists of width, height, clipping region and transformation matrix.
- Volume and stereo balance values if the track is an audio track.
- User data, such as a user-readable name for the track.

MEDIA REFERENCES

As with movies and tracks, QuickTime media are referenced by private data structures and QuickTime provides access to functions which let you work with this information.



QuickTime stores the following information in each media reference:

- Creation date and time and last modification date and time.
- Media's time coordinate system, time scale and duration - this is usually different from the time coordinate system of the movie that plays the media. An example of this would be a movie with a time scale of 600 including video media with a time scale of 30 fps. QuickTime will automatically keep the two systems in sync.
- Media handler specification. A reference to the QuickTime component that must be used to interpret and display the media content.
- Media information - including a specification of where the content is stored, what type it is, how it is compressed.
- Language and playback quality.
- User data. A general area for user data such as copyright information and credits.

Media references are small as they only point to the media which will actually be used to play and are included in the movie structure. To keep everything manageable when a movie and all its media are packed together into a single movie file, the media is usually compressed.

QuickTime will not only find and interpret media chunks but will also decompress them on the fly.

DATA REFERENCES

Movies obtain the actual media they play from several sources outside the movie structure.

The most common of these sources are:

- Macintosh or Windows files on disk or CD-ROM.
- Network sources, accessed through URLs.

Movies access media through QuickTime data references. When a movie accesses a chunk of media, QuickTime uses a function such as `GetDataHandler` to return a data handler component that can interpret the chunk.

All data handlers identify their data with data references. Data references specify the location of the data and which data handler is able to interpret the data. A data reference consists of the following two components:

```
Handle dataRef
OSType dataHandlerSubType
```

The `dataRef` value specifies the actual data reference. This is a handle to the information that identifies the data to be used. The type of information stored in the handle depends on the data reference type (`dataHandlerSubType`).

For example, if your application is loading a movie from a handle, this data reference handle would contain a handle to the movie data.

The `dataHandlerSubType` value specifies the type of data reference and can be one of the Macintosh OSTypes through which media can be accessed. The type of information stored in the handle depends on this data reference type. For example, for an alias data reference you set this to `rAliasType ('alis')`, indicating that the reference is an alias.

Listed below are all the available data reference types:

- 'alis' (`rAliasType`) Data reference is a Macintosh alias handle. An alias handle contains information about the file. For more information refer to the references provided at the end of this section.
- 'hndl' (`HandleDataHandlerSubType`) Data reference is a Macintosh handle whose data is the handle containing the data. In addition to specifying movie data in memory, this handle may contain data reference extensions containing the file type, file name, MIME type etc. as described in Technote 1195. For more information refer to the reference provided at the end of this section.
- 'rsrc' (`ResourceDataHandlerSubType`) Data reference is a Macintosh alias handle. However, appended to the end of the alias handle is the resource type (stored as a OSType) and ID (stored as a 16-bit signed integer) that identifies the resource within the specified file. Both these values must be big-endian format.
- 'url' (`URLDataHandlerSubType`) Data reference is a handle whose data is a C-string (null-terminated) specifying a URL. A 'url' data reference can have data reference extensions too, so the actual size of a url data reference may be larger than the length of the string. QuickTime supports the 'file', 'http', 'ftp' URL types as well as the 'rtsp' type in special cases when opening movies.

GRAPHIC TRANSFORMATION

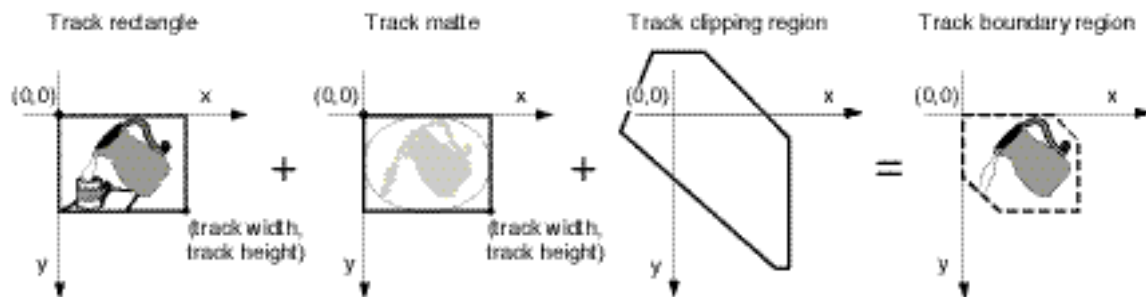
When a movie containing visual data is played, the movie's data is gathered from the appropriate tracks and media structures and as this data is passed though QuickTime on its way to the outside world

certain actions may be performed on these chunks of data before they are finally presented to the user.

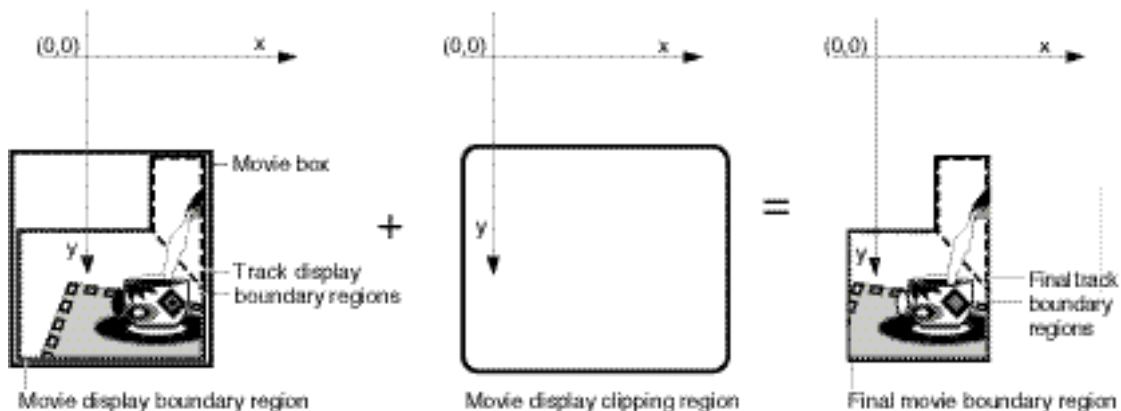
CLIPPING

Clipping is the process of limiting the spacial boundaries of an image before it's presented to the user. QuickTime lets you clip each graphic track separately, then clip the movie as a whole.

An image in a video or graphics track has a track rectangle, defined by a track coordinate system anchored to the upper-left corner of the track's image. When you apply a track clipping region to the track rectangle, their intersection becomes the track boundary region. This is the portion of the tracks image the viewer will see.



Movies as a whole can also be clipped. The movie boundary region is the union of all the clipped track regions. It is the total space needed to contain any track image. By applying a movie clipping region this space can be cut back to the clipped movie boundary region. These spaces are all defined in terms of the movie coordinate system.



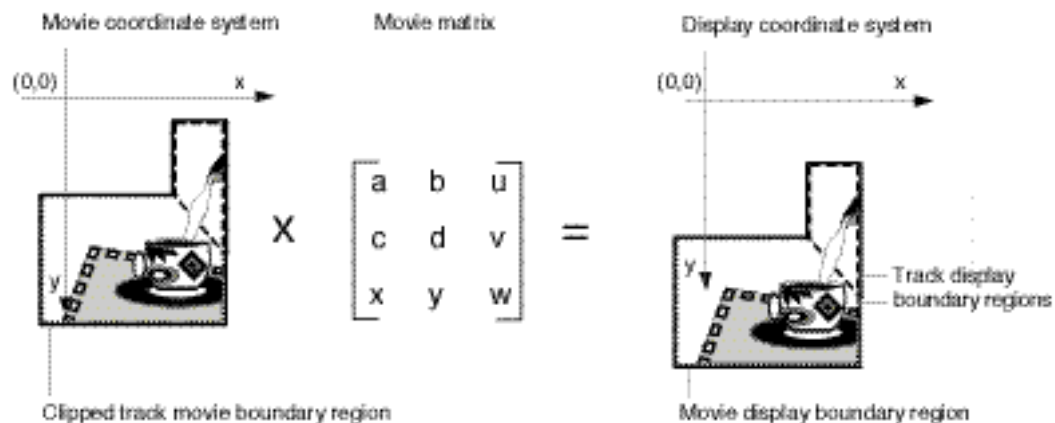
MATRIX

Matrix transformations allow you to slide images around, shrink or rotate them and translate their coordinates from one system to another. A transformation matrix defines how to map points from one coordinate system into another coordinate system.

QuickTime uses nine-element matrices to translate among coordinate systems. The diagram below is referred to as the **identity matrix** and performs no transformations.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Each matrix defines how measurements in one system must be calculated to become measurements in another system. The movie matrix for example translates between the movie coordinate system and the display coordinate system. The display coordinate system is the same as that of the graphics world in which the movie is played.



Matrix operations also let you slide, stretch, shrink and rotate images. When matrix operations are combined with clipping operations they give you complete control over how graphics and video images are displayed by QuickTime movies.

Transformation matrices used by the Movie Toolbox contain the following data types:

[0] [0] Fixed	[1] [0] Fixed	[2] [0] Fract
[0] [1] Fixed	[1] [1] Fixed	[2] [1] Fract
[0] [2] Fixed	[1] [2] Fixed	[2] [2] Fract



A common pitfall when first using some QuickTime APIs is to pass integers or floats to **Fixed** parameters, or use them in MatrixRecords. The Fixed format has 16 bits to the left and 16 bits to the right of the binary point. Fixed format numbers range from -32,768 to approximately 32,768. The value of 1.0 Fixed is represented by the integer value of 65536 [0x00010000].

For more information regarding transformations refer to the reference provided at the end of this section.

LAYERING, TRACK ENABLING AND QUALITY

Layering refers to how tracks can overlap one another, layering several images to produce what the user sees. Think of the tracks in a movie as if they were painted on overlapped layers of film. What is presented to the user is the sum of the contents in all the tracks that are enabled at any given instant. Track layers are numbered from -32,768 through 32,767. Lower numbered layers are shown on top of higher-numbered layers. This layering order is important mainly when multiple video tracks are combined through graphics modes.

QuickTime movies can contain many tracks or a series of grouped alternate tracks which may be enabled or disabled as required to present the user with the desired experience.

By enabling one track out of a group, you can present different levels of quality in a movie or select from multiple content. You could layer an 8-bit sound track with an English dialog track or enable a higher quality 16-bit audio track with a Spanish dialog track.

QuickTime lets you store quality information for media in an 8-bit quality setting that is part of the media's description structure. Bits 6 and 7 of the setting encode a relative quality number ranging from 0 to 3 where higher quality values indicate larger sample sizes. You could use this functionality if for example a movie contained two alternate sound tracks, one referencing 8-bit sound media and the other

referencing 16-bit sound media. The 8-bit media would be assigned a quality value of `mediaQualityNormal` (0x0040) and the 16-bit media would be assigned a quality value of `mediaQualityBetter` (0x0080). QuickTime would play the 16-bit media only if the user's configuration could handle 16-bit sound; otherwise it would play the 8-bit media.

Additionally QuickTime uses the low-order bits 0 through 5 of the quality setting to indicate pixel depths at which the media should be played. The quality bits correspond to depths of 1, 2, 4, 8, 16, and 32 bits of data.

GRAPHICS MODES

When images are overlapped, QuickTime must decide for each pair of pixels (one in the bottom image and one in the top) what pixel value the user should see. QuickTime bases its decision on the graphics mode. QuickTime supports all the Macintosh QuickDraw transfer modes plus four alpha channel modes of its own.

Commonly used modes:

- copy
- blend
- dither
- transparent
- alpha
- alpha blend
- premultiplied white alpha
- premultiplied black alpha

Copy Mode - The default mode for still images and video layers, in this mode QuickTime simply displays the top image and ignores the lower layers.



Blend Mode – This mode averages the numeric values of each color for each corresponding pixel in two overlapped images. You can move the average toward one layer or the other by applying weighting factors.

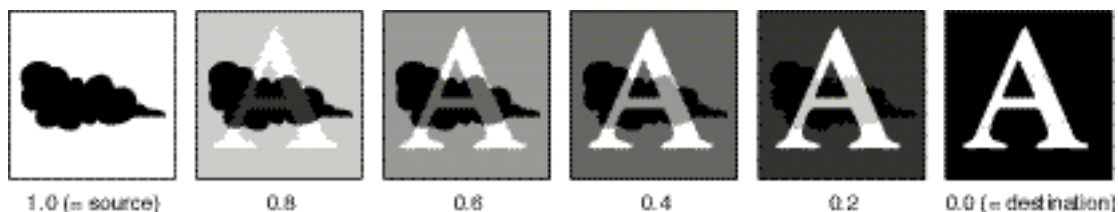


Dither Mode – You can use this mode to approximate more colors than are available on the screen. For example, you could dither a 32-bit image on an 8-bit display. Dithering can produce a grainy image and will increase rendering time.

Transparent Mode – This mode replaces a pixel in the lower layer with the overlapping pixel from the upper layer, but only if the upper-layer pixel's color is not equal to a specified background color. This mode can be used to perform matte or blue screen techniques.

Alpha Modes – An alpha value in the color description of an image defines the image's opacity. Opacity is the degree to which an image obscures images in lower layers. An alpha value of 1 (0xFF for 32-bit color) means the image covers the low one completely. 0 would mean the image is invisible and the lower layer shows completely through. By changing the alpha value you can make the image fade in and fade out over a background without affecting its coloring.

Alpha values may also be used to blend two images. The alpha blending mode will use the alpha value of the first image to determine how much of it appears in the resulting image and will use 1 minus this alpha value to determine how much of the second image appears in the resulting image. The result of blending produces an image with a total opacity of 1, but it constitutes a blend of two semi-opaque images.



QuickTime can also perform premultiplication by either white or black backgrounds. Pre-multiplied with white means that the color components of each pixel have already been blended with a white pixel, based on their alpha channel value. What this means is the image has already been combined with a white background. To combine the image with a different background color, QuickTime must first remove the white from each pixel and then blend the image with the actual background pixels. Images are often pre-multiplied with white as this reduces the appearance of jagged edges around objects.

Pre-multiplied with black is the same as pre-multiplied with white, except the background color that the image has been blended with is black instead of white.



Although you pass these new alpha channel graphics modes to QuickTime in the same way as you would traditional QuickDraw transfer modes, these modes are not supported by QuickDraw and will cause unpredictable results if passed to QuickDraw routines.

SOUND TRACKS

QuickTime movies may have multiple sound tracks which can be layered and enabled just like video and graphics tracks. QuickTime will mix separate music and voice tracks to accompany video, or you can enable one track out of a set of alternate tracks recorded in different languages. As with graphics, sound media may be transformed by manipulating their track properties.

VOLUME

Every QuickTime movie stores a preferred volume value for the whole movie. Each audio track also stores a preferred volume value for the track. Track volume allows you to adjust the loudness of one track relative to another while movie volume allows you to specify the loudness of all the tracks mixed together. When a movie is loaded, QuickTime sets its current track and movie volumes to the preferred values of the movie.

Volumes in QuickTime are represented as a 16-bit fixed value. This value has a range of -1.0 to +1.0. The high-order 8 bits contain the integer portion of the value and the low-order 8 bits contain the fractional part. Positive values denote volume settings with 1.0 corresponding to the maximum volume on the user's computer.

Negative values are silent but retain their magnitude. By toggling the sign of the volume setting, you can very easily perform a mute function and turn off the sound and then turn it back on at its previous level.

BALANCE

Tracks and media also have their own balance setting. Balance controls the relative levels of sound between a computers left and right speakers. When working with a monaural source, the balance setting controls the loudness of each speaker. With a stereo source the balance setting governs the relative emphasis of the right and left channels. When a movie is saved, these setting are preserved and stored in the movie file.

The balance values are represented by a 16-bit fixed-point number ranging from -1.0 to +1.0. The high-order 8 bits contain the integer portion of the value and the low-order 8 bits contain the fractional part. Negative values move the balance control to the left while positive values move the balance to the right.

REFERENCES

MOVIE TOOLBOX

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/INMAC/QT/iqMovieToolbox.htm>

SUPPORTED QUICKTIME MEDIA TYPES

<http://www.apple.com/quicktime/specifications.html>

TRANSFORMATION MATRIX

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/INMAC/INTROS/xxIntroductions.17.htm#pgfid=26240>

INSIDE MACINTOSH: FILES

<http://developer.apple.com/techpubs/macos8/Files/AliasManager/aliasmanager.html>

INSIDE MACINTOSH: MEMORY

<http://developer.apple.com/techpubs/macos8/OSSvcs/MemoryManager/memorymanager.html>

TECHNOTE 1195 : TAGGED HANDLE DATA REFERENCES IN QUICKTIME 4

<http://developer.apple.com/technotes/tn/tn1195.html>

INTRODUCTION TO SOUND

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmsound.htm>

ASYNCHRONOUS MOVIE LOADING

QuickTime supports the asynchronous loading of movies. This method is the preferred mechanism for instantiating movies located on a network or other slow link. In fact, asynchronous movie loading can be used in all movie-loading situations with QuickTime taking care of which load model to use.

NON-ASYNCHRONOUS LOAD MODEL:

`NewMovieFrom... <playable> <complete>`

ASYNCHRONOUS LOAD MODEL:

`NewMovieFrom... <incomplete> <playable> <complete>`

There are a number of advantages to using the asynchronous model. To the user, it can mean a more responsive system. The difference in user experience centers on the time before the movie data has actually been located and processed. To the developer, it can mean more time to do things while a movie loads in cases where it can take some time for enough data to arrive to allow the movie to play. For example, very large movie files over a slow network connection.

Instead of a fully formed movie being immediately available and playable, `NewMovieFrom...` functions will return a valid movie, but that movie may be completely empty. For example, it may contain no tracks, user data etc. At some point, when the movie resource is found or the file has been imported this information becomes available and the movie is 'playable'.

To enable asynchronous loading when opening a movie, an additional flag (`newMovieAsyncOK`) is passed to one of the `NewMovieFrom...` functions. Without this flag, calling one of the `NewMovieFrom...` functions returns either a fully formed movie or an error.

During the time when the movie state is `kMovieLoadStateLoading` the movie cannot be worked with. To determine when the movie becomes `kMovieLoadStatePlayable` use the `GetMovieLoadState` function, this call will return a value indicating the current movie state.

- `kMovieLoadStateLoading` – QuickTime instantiating the movie.
- `kMovieLoadStatePlayable` – movie fully formed and can be played.

- `kMovieLoadStateComplete` – all media data is available.
- `kMovieLoadStateError` – movie loading failed.

REFERENCES

QUICKTIME 4.1 – INTRODUCTION TO ASYNCHRONOUS MOVIE LOADING
http://developer.apple.com/techpubs/quicktime/qtdevdocs/REF/QT41_HTML/QT41WhatsNew-81.html

FAST-START PROGRESSIVE DOWNLOADS

QuickTime supports progressive downloads of movies, which allow part of a movie to be displayed before all of its data has been received over a network or other slow link. QuickTime is able to instantiate the movie and start playing it before the entire file has arrived. This is known as Fast-Start movie playback.

A Fast-Start download is a simple file transfer. Any QuickTime movie saved using the Fast-Start option can be played before all of its data has been received.

QuickTime movies can be Fast-Started using:

- HTTP (Hyper Text Transport Protocol)
- FTP (File Transfer Protocol)

HTTP uses TCP/IP protocol to ensure that all movie packets are delivered, retransmitting if necessary. HTTP allows the client to store the data locally and Fast-Start the movie after enough data has arrived.

Applications using the movie controller component automatically get support for progressive downloads. Applications can also use the `GetMaxLoadedTimeInMovie` function to determine whether a movie is being progressively downloaded and, if so, to see how much of the movie has already been downloaded. If all of the movie has been downloaded `GetMaxLoadedTimeInMovie` will return the duration of the entire movie.

REFERENCES

SUPPORT FOR PROGRESSIVE DOWNLOADS
<http://developer.apple.com/techpubs/quicktime/qtdevdocs/REF/reflMovieToolbox.11.htm#pgfid=24160>

STREAMING

OVERVIEW

Streaming involves sending movies from a server to a client over a network such as the Internet. The server breaks the movie into packets that can be sent over the network. At the receiving end, the packets are reassembled by the client and the movie is played as it comes in.

QuickTime movies can be real-time streamed using:

- RTP (Realtime Transport Protocol)

Streaming is different from Fast-Start download in that the client plays the movie as it comes in over the network, rather than downloading the entire movie and playing it as it is being downloaded. A streaming movie is never actually downloaded; its packets are played as they come in, then discarded.

QuickTime supports RTP streaming of video, audio, text, and MIDI.

RTP is used for real-time streaming. Movie packets are sent in real time, so that a one-minute movie is sent over the network in one minute. The packets are time-stamped, so they can be displayed in time-synchronized order. Because packets are sent in real time, RTP streaming works with live content in addition to pre-recorded movies and can even carry a mixture of the two. RTP can be used for live transmission and multicast, this allows the user to view long movies or continuous transmissions without having to store more than a few seconds of data locally. Using RTP transmission under RTSP control, a user can skip to any point in a movie on a server without downloading the intervening material.

RTP streams can be incorporated in a movie using streaming tracks.

A streaming track is a track in a QuickTime movie that contains the URL of the streaming content. A QuickTime movie that contains streaming tracks can also include non-streaming tracks whose media exist on the client's computer. This allows a live transmission, or data stored on the Internet, to be incorporated into a movie along with material stored on locally or distributed over CD-ROM.

RTP uses UDP protocol, which doesn't attempt to retransmit lost packets. This allows multicasts as well as live streams, both cases where retransmission would not be practical.

APPLICATION CONSIDERATIONS

An application receives streaming content by opening a movie and playing it. In general, opening a streaming movie is like opening any QuickTime movie. You can open a streaming movie by opening a movie file that contains streaming tracks, a session description file (.SDP) or a URL.

If you open a streaming movie from a URL or an SDP file, the Movie Toolbox will call the appropriate movie importer. If you open a movie file that contains streaming tracks, stream importers will be called.



Applications that can already play QuickTime movies need to take the following points into account to reliably support real-time streaming movies:

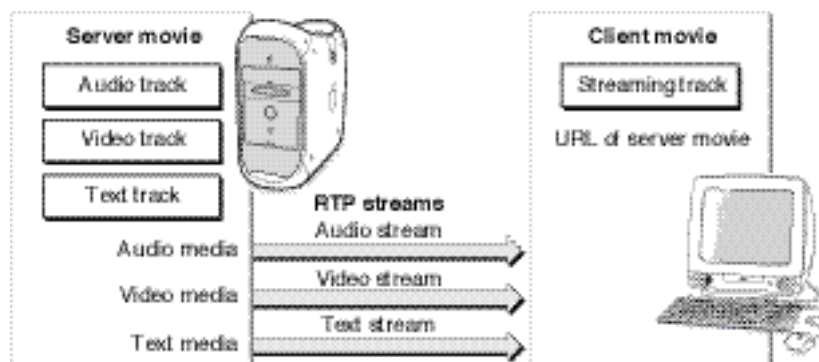
- Open movies with high-level Movie Toolbox calls or a movie importer.
- Do not assume that the track structure of the movie you play reflects the track structure of the original movie; a streaming track can contain sound, video, text, MIDI, or all of these.
- Use a movie controller to play the movie, or make sure to Pre-Preroll in addition to Prerolling a movie to set up any streams before playing the movie.
- Display status messages returned by the movie controller.
- Be prepared to deal with network errors when your application plays a movie.
- Be prepared for movie characteristics to change dynamically; the height, width, duration, and whether or not the movie has audio or video can all change as the movie plays.
- Do not assume that the movie will begin immediately.

TRACK STRUCTURE

Unlike other QuickTime movies, streaming movies consist of two distinct movie files—one on the server and the other on the client machine—often with different track structures. You sometimes need to distinguish between the server movie and the client movie.

The media of each track in the server movie is transmitted as a separate RTP stream. On the client side, multiple RTP streams can be combined into a single streaming track.

If you open a movie from a URL, for example, the movie importer may create a client movie that contains only a streaming ('strm') track. Unlike most other QuickTime track types, a streaming track can contain multiple media streams of different types. A streaming track in a client movie may contain the URL of a server movie with audio, video, text, or MIDI tracks.



The client movie file never contains audio or video media from the server movie; they are displayed and discarded. If a client movie is saved, the saved movie contains information such as the URL of the server movie and the currently displayed movie time. The client movie can also contain local tracks and local media, but the media samples in the server movie remain on the server except when playing.

PREROLL AND PRE-PREROLL

Before any movie is played, QuickTime needs to allocate buffers and open appropriate media handlers. This process is called prerolling the movie. Before a streaming movie can be played, additional steps need to be taken, such as establishing RTP streams between the client and the server. This setup process is called pre-prerolling. Pre-prerolling is performed automatically when a streaming movie is played using a movie controller. If your application uses movie controllers to play

movies, you do not need to take any special steps to pre-preroll a streaming movie.

If you are playing movies using the Movie Toolbox (without a Movie Controller), you will need to pre-preroll the movie to set up the network connections for a streaming movie before you can preroll or play the movie. Pre-prerolling does nothing unless a movie contains streaming content, so it's safe to do for all movies.

Pre-prerolling can be done either synchronously or asynchronously, depending on whether you specify a completion procedure. If called asynchronously, control will return almost immediately even if the movie contains streaming content. This allows your application to perform other tasks while awaiting the completion of the pre-prerolling. You will need to periodically to grant time to the Movie Toolbox for the task of pre-prerolling when using it asynchronously. When pre-prerolling is finished, your completion procedure will be called. In the simplest case, the completion procedure will want to preroll and start the movie.

REACTING TO CHANGES IN MOVIE CHARACTERISTICS

One feature of streamed movies is that their characteristics may change dynamically during playback. For example, when you open a movie from a URL you may not know the actual height and width of the movie, its duration or how many streams it contains. You also might not know if the movie has a sound track, or whether it is an audio-only movie.

QuickTime will assign default values to these characteristics if they are unknown.

QuickTime can notify your application when the movie characteristics become known or are changed and in most cases, your application will want to reflect such changes. Changes in the movie size and other movie characteristics are announced and can be handled in a action filter procedure.

DURATION CHANGES

The duration of a streaming movie or a streaming track may not be initially known. A track or movie whose duration is not known is assigned an indefinite duration: 0x7FFFFFFF . If you do not treat this as

a special case, a streaming movie will appear to your application as a very long movie. Once the pre-preroll process is complete, QuickTime should know the actual duration of the streams. Duration changes are also handled in the application's action filter procedure.

If the movie contains live content, it may not have a specific duration. In this case, the duration remains indefinite: 0x7FFFFFFF . You might want to set a timeout in your code that detects the fact that QuickTime has not adjusted the duration from 0x7FFFFFFF after a few seconds of movie play and use such a timeout to notify the viewer that the movie is a live broadcast.

SOUND AND VIDEO CHANGES

Whether a streaming movie has sound or is sound only (no video) may not be initially known or may change dynamically. If you have an action filter procedure and set the movie hints, your applications filter procedure will be called when the sound or video characteristics change.

OTHER PLAYBACK CONSIDERATIONS

Streaming movies only play back at a rate of 1. Other playback rates, such as playing backward, are not supported when streaming in real-time.

REFERENCES

STREAMING

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/REF/Streaming.htm>

CUSTOMIZING MOVIE CONTROLLERS

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/INMAC/OTC/imMovieContComp.8.htm#pgfid=965>

COMPONENTS

A component can be thought of as a black box, a piece of code providing a specific set of services to a client. QuickTime is highly modular in that it is made up of over 200 different components, each performing a specific job and responding to a defined programming interface. This modularity not only make it simpler to program for QuickTime but allows QuickTime to be extended through the addition of new components providing new and or specific services. These

components provide essential services to your application and to the managers that make up the QuickTime architecture.

THE COMPONENT MANAGER

The Component Manager provides services that allow you to find, open, and access specific components.

Components of a particular type are found using three main criteria which are part of a `ComponentDescription` structure.

- **Type of component service** - this is the `componentType` field within the component description and identifies the type of component. For example, graphics exporter components all have a component type 'grev' represented by the `GraphicsExporterComponentType` constant.
- **Specific service type** - this is the `componentSubType` field within the component description and identifies the subtype of the component. Different subtypes of a component type may support additional features or provide interfaces that extend beyond the standard routines for a given component type. For example, the graphics exporter for PNG files has a component subtype of 'PNGf' represented by the `kQTFil eTypePNG` constant.
- **Component manufacturer** - this is the `componentManufacturer` field within the component description. This field allows for further differentiation between individual components.

Additionally, specific information about a particular component can be specified by using the `componentFlags` and `componentFlagsMask` fields of the component description record.

- **componentFlags** - These flags can be used to indicate the presence of features or capabilities in a given component and can be used to further narrow the search criteria for a particular component. If you use the `componentFlags` field in a component search, use the `componentFlagsMask` field to indicate which flags are to be considered in the search.
- **componentFlagsMask** - These flags indicates which flags in the `componentFlags` field are relevant to a particular component search operation. For each flag in the `componentFlags` field that is to be considered your application should set the corresponding bit in the

`componentFlagsMask` field to 1. For example, to look for a component with a specific control flag that is set to 0, set the appropriate bit in the `componentFlags` field to 0 and the same bit in the `componentFlagsMask` field to 1. To look for a component with a specific control flag that is set to 1, set the bit in the `componentFlags` field to 1 and the same bit in the `componentFlagsMask` field to 1. To ignore a flag, set the bit in the `componentFlagsMask` field to 0.



There is an Ice Floe Engineering Technical Note which expands on the above discussion as it applies to Movie Export Components. Refer to the reference at the end of the Movie Data Exchange Components section of this document.

The `FindNextComponent` function returns the component identifier for the next registered component that meets the selection criteria specified by your application. You specify the selection criteria in a `ComponentDescription` structure.

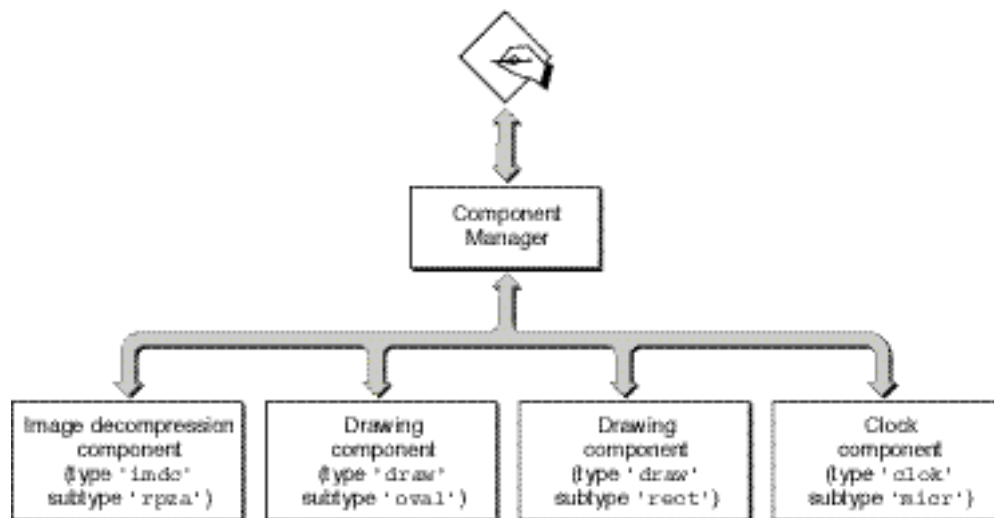
You use the `OpenComponent` function to gain access to a specified component by using the previously obtained component identifier.

If you are interested only in using a component of a particular type-subtype and you do not need to specify any other characteristics of the component, use the `OpenDefaultComponent` or `OpenDefaultComponent` function and specify only the component type and subtype—the Component Manager then selects a component for you and opens a connection to that component.

When a component is opened through the component manager an instance of that component is created. This establishes a communications channel with the component and allocates memory for the tasks to be performed.

When you are done with a component instance that you opened, you should close it using `CloseComponent`. This will terminate the communication channel and release memory used by the component.

This diagram shows the relationship between an application, the Component Manager, and components.



COMPONENTS TYPES

QuickTime components fall into several categories:

- **Movie Controllers** – displays movies and provides playback controls.
- **Media Handlers** – interpret and manipulate media sample data.
- **Video Digitizers/Sequence Grabbers** – convert analog video data to digital form. Allows the ability to obtain digitized data from external sources.
- **Data Exchange** – import and export data from non-movie sources and non-movie formats.
- **Compressors/Decompressors** (Codecs) – provide compression and decompression services for media (such as sounds and image sequences).
- **Transcoders** – translate data from one compressed format to another.
- **Video Output** – send video to devices (such as DV Cameras) that are not recognized as displays by the computer.

- **Graphics Importers** – display still images obtained from data in various file formats.
- **Graphics Exporters** – store still images in the same format that graphics importers handle.
- **Preview** - create and display previews.
- **Tween** – perform interpolation between values of various data types.
- **Effects** – provide real-time effects and transitions.
- **Text Channel** – imports and exports text between movies and external text handling applications such as word processors.
- **Clock** – generate timing information and schedule time-based callback events.
- **Real-time streaming** – allow you to receive movies and live video in real time without downloading large files.

In some cases you may want to modify, extend or replace QuickTime's built-in components to accomplish a specific goal.

Media Presentation – You could write your own movie controller, graphics importer, and sounds component or music synthesizer.

Media Capture – You may want to create a new Video Digitizer, image compressor or image compressor dialog component.

Media Handling – QuickTime provides extendable component frameworks that you can modify to provide custom handlers for video, sound, text, sprite, 3D, vector graphics and tween media. They include base components for media handling, effects, and transitions and musical instruments.

Utility Tasks – These include data exchange components, sequence grabber channel and panel components, video digitizers, video output components, image transcoders, data handlers, streaming components, clock components, music components and note allocators. These can be replaced if you want to alter specific areas of QuickTime's behavior.

REFERENCES

COMPONENT MANAGER

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmComponentMgr.htm>

GRAPHICS IMPORTER / EXPORTER COMPONENTS

QuickTime's graphics import and export components allow you to work with image data regardless of its file format or compression method.

GRAPHICS IMPORTERS

Graphics importer components provide a standard method for applications to open and display still images contained within graphics documents. Graphics importer components allow you to work with any type of supported image data, regardless of the file format or compression used in the file.

You can use the graphics importer component functions to obtain a graphics importer component instance that can read graphics data from a particular file or area of memory. For example, `GetGraphicsImporterForFile` will attempt to locate and open a graphics importer component that can be used to draw the specified file. The component can then be used to find some characteristics of the image such as its dimensions by using `GraphicsImportGetNaturalBounds` and can be drawn by simply calling `GraphicsImportDraw`.

ALPHA CHANNELS

Some Quicktime supported file formats, such as TIFF, PNG and Photoshop, support alpha channels; some, such as JPEG and TGA, do not. However, even if a file format supports alpha channels, not every file will contain one.

You can use graphics importers to find out whether a file has an alpha channel by calling `GraphicsImportGetImageDescription` and looking at the depth field in the image description. By convention, a depth of 24 means RGB without alpha, whereas 32 means ARGB.

QuickTime will not synthesize an opaque alpha channel for an image that does not have one; if the depth is 24, the first byte of each pixel will probably be zero. Developers working with alpha channels should only depend on these values if the depth is 32.



All depths of PNG files can have transparency — for example, a 4-bit indexed-colour PNG file can have an opaqueness value associated with each colour index. QuickTime translates these files to 32-bit ARGB in order to preserve the transparency information.

MULTIPLE IMAGES

Graphics Importers also support image formats containing multiple images in a single file. For example, TIFF files can support multiple images, Photoshop files can contain multiple layers and FlashPix files can contain multiple resolutions. You can use `GraphicsImportGetImageCount` to find out how many images are in a file, and `GraphicsImportSetImageIndex` to select a particular image.

GRAPHICS EXPORTERS

Graphics exporter components provide a standard interface for exporting graphics to image files in a variety of formats. QuickTime selects a graphic exporter component based on the desired output format, such as GIF or PNG.

The image input for an export operation can come from a QuickDraw Picture, a `GWorld` or `PixMap`, a QuickTime graphics importer component instance, or a segment of compressed data described by a QuickTime image description. In the last case, the compressed data may be accessed via a pointer, handle, file or other data reference.

The output image for an export operation can be stored in a handle, file, or other data reference. Different file formats support a wide range of configurable features, such as depth, resolution, and compression quality.

To use a graphics exporter, you must first open a graphics exporter component instance, specify the source of the input image, its destination, set whatever parameters you want and then actually do the export. For example, if you want to write a `GWorld` out to a PNG image file you would use `OpenADefaultComponent` to open an instance of the `kQTFileTypePNG` graphics exporter component, `GraphicsExportSetInputGWorld` to associate the input `GWorld` with the export operation, `GraphicsExportSetOutputFile` to associate the output `FSSpec` with the export operation and `GraphicsExportDoExport` to perform the export.

Once you're finished with component instances you can simply close them using `CloseComponent`.

REFERENCES

GRAPHICS IMPORTER COMPONENTS

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmlImporter.htm>

GRAPHICS EXPORTER COMPONENTS

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmExporter.htm>

EXPORTING IMAGES – DISPATCH 14

<http://developer.apple.com/quicktime/icefloe/dispatch014.html>

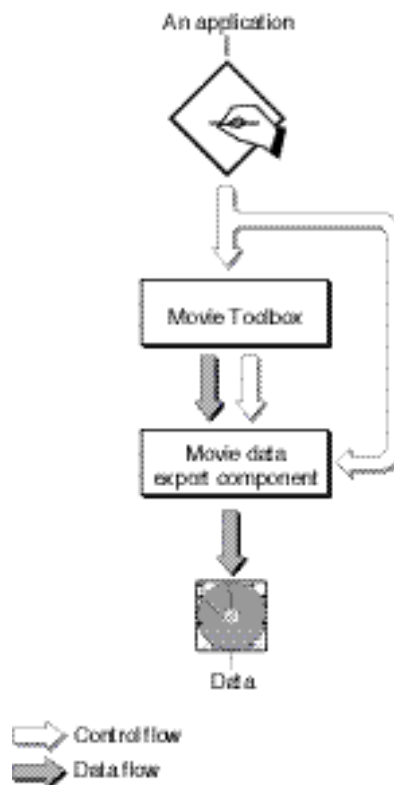
MOVIE DATA EXCHANGE COMPONENTS

Movie data exchange components allow applications to place various types of data into a QuickTime movie or extract data from a movie in a specified format.

Movie data import components support in-place file importing with no extra work needed by the QuickTime application developer. The `NewMovieFrom...` calls can open GIF, JPEG, AVI, SMIL etc. files automatically and translate the nonmovie data formats into the QuickTime movie data format. For example, in the case of an animated GIF, the movie data import component might convert the images into frames in a QuickTime movie.

Movie data export components convert movie data into other formats, so that the data can be used by other applications. For example, a movie data export component might allow an application to extract the sound track from a QuickTime movie in AIFF format. The extracted sound track may then be manipulated by applications that are not QuickTime-aware.

The diagram below shows how movie data export components are used by a QuickTime-aware application.



IMPORTING AND EXPORTING MOVIE DATA

Applications can start a data import or export operation by calling the Movie Toolbox. There are several Movie Toolbox functions that allow you to specify a data import or data export component. For example, the `PasteHandleIntoMovie` and `ConvertFileToMovieFile` functions allow you to specify a movie data import component. The `PutMovieIntoTypedHandle` and `ConvertMovieToFile` functions allow you to specify a movie data export component.

All of these functions select a component for you automatically if you do not specify one yourself.

REFERENCES

MOVIE DATA EXCHANGE COMPONENTS

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmDataExchange.htm>

COMPONENT MANGER

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmComponentMgr.htm>

FINDING MOVIE EXPORT COMPONENTS – DISPATCH 6

<http://developer.apple.com/quicktime/icefloe/dispatch006.html>

THE SEQUENCE GRABBER

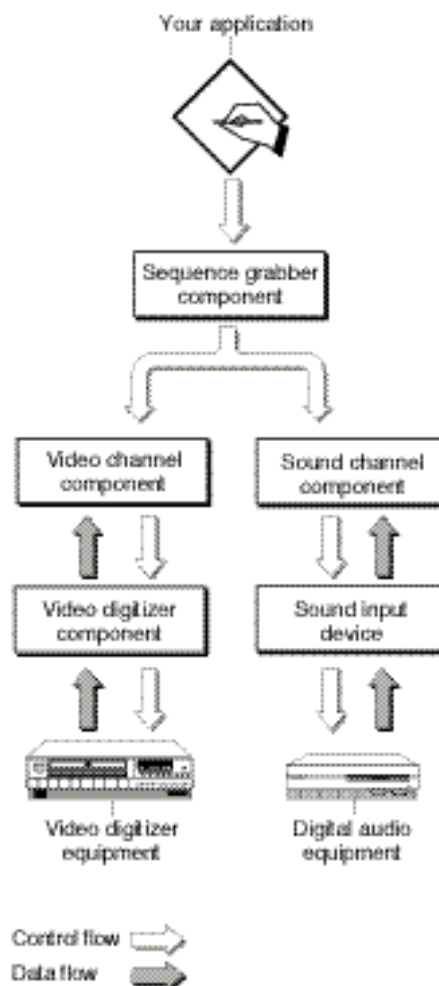
SEQUENCE GRABBER COMPONENTS

Sequence Grabber components allow applications to obtain digitized data from external sources, such as video capture boards. The digitized data can be previewed, saved as a QuickTime movie, or both. Sequence grabber components allow applications to capture audio and video easily, without concern for the details of how the data is acquired.

These components rely on the services of lower-level components, called channel components, to obtain digitized data from different sources. For example, a single sequence grabber component could provide an application with video and audio data, using the services of different channel components for audio and video. The channel components, in turn, may rely on the services of still lower-level components, such as video digitizer components.

While the entire process of capturing digital data from different sources is quite complex a QuickTime developer is shielded from these complexities by using the higher-level Sequence Grabber API.

The diagram below shows the relationships among a QuickTime application, a sequence grabber component, and channel components.



Sequence grabber components are standard components that are managed by the Component Manager.

You can use sequence grabber components in two ways: to play digitized data for the user or to save captured data in a QuickTime movie. The process of displaying data that is to be captured is called previewing; saving captured data in a movie is called recording. You can use previewing to allow the user to prepare to make a recording. If you do so, your application can move directly from the preview operation to a record operation, without stopping the process.

PREVIEWING

Previewing captured data involves playing the data for the user as it is captured. For video data, this means displaying the video images on the computer screen. For audio data, this means playing the sound through the computer's sound system.

RECORDING

During a record operation, a sequence grabber component collects the data it captures and formats that data into a QuickTime movie. During a record operation, the sequence grabber can also play the captured data for the user. However, the sequence grabber tries to prevent the playback from interfering with the quality of the recording process.

CHANNEL COMPONENTS

Sequence grabber channel components, also known simply as channel components are used by higher-level sequence grabber components, and act to isolate the sequence grabber from the details of working with actual data types. Channel components may, in turn, depend on the services of still lower-level components, such as video digitizer components. QuickTime application programmers use the services of a sequence grabber channel component but usually do not have to interact with them directly.

PANEL COMPONENTS

A sequence grabber panel component creates a settings dialog box, which has settings that affect the behavior of a channel component. For example, a dialog might let the user control the frame capture rate of a video digitizer and the image quality of an image compressor.

Sequence grabber panel components are never called directly by an application. QuickTime application developers indirectly use panel components by calling the sequence grabber component. The sequence grabber component uses a panel component to obtain user preferences for configuration of channel components.

REFERENCES

SEQUENCE GRABBER

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmSeqGrabComp.htm>

SEQUENCE GRABBER CHANNEL COMPONENTS

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmSeqGrabChanComp.htm>

SEQUENCE GRABBER PANEL COMPONENTS

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmSeqGrabPanComp.htm>

VIDEO DIGITIZERS

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmVideoDigComp.htm>

VIDEO OUTPUT COMPONENTS

Video output components allow you to send QuickTime video to devices that are not recognized as displays. These components are used directly by QuickTime applications to allow the user to send movie output to these external devices. For example, Digital Video (DV) and Firewire capable cameras.

QuickTime applications can use a video output components by selecting the component, configuring it, and associating it with a `GWORLD`.

A video output component receives QuickTime video data and delivers the data to a video output device for display. If the incoming data is in a format that the video output device can display directly, the video output component can simply send the data to the video output device. If the incoming data cannot be displayed directly, the video output component must use a transfer codec or decompressor component to convert the data to a format that the video output device can display.

A video output device has one or more display modes. The characteristics of each mode determine how video is displayed. When any software displays video on a video output device, it must select which of the device's display modes to use.

The characteristics of a display mode include.

- The height of the displayed image, in pixels
- The width of the displayed image, in pixels
- The horizontal resolution of the display, in pixels per inch
- The vertical resolution of the display, in pixels per inch
- The refresh rate of the display, in Hertz
- The pixel type of the display
- A text description of the display mode

The characteristics can also include a list of decompressor components required for the mode and provided specifically for the video output device. If a video output device cannot directly display any of the pixel formats supported by QuickTime, the vendor of the device must provide one or more special decompressors to convert supported pixel formats to a format the device can display. If a video output device can display one or more of the pixel formats supported by QuickTime, the Image Compression Manager can use standard decompressors that are included with QuickTime, and the list of special decompressor components can be empty.

REFERENCES

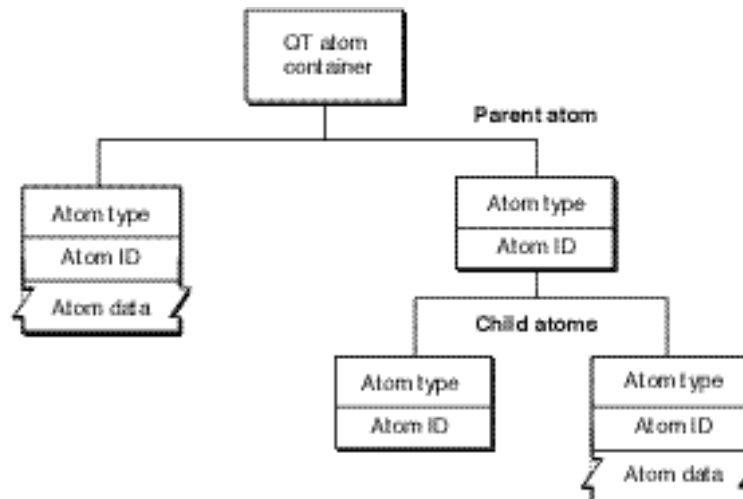
VIDEO OUTPUT COMPONENTS

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmVidOutComp.htm>

QUICKTIME ATOM CONTAINERS

QuickTime stores most of its data using a special memory structure called atoms. You can think of atoms as the basic data containers inside QuickTime. Movies and their tracks are organized as atoms.

A newly created atom is like the root of a tree structure. Each subsequent atom is contained within it and can either contain some data or another atom. If an atom contains other atoms it is referred to as a parent atom, and the atoms it contains are referred to as child atoms. If an atom contains only data instead of another atom, it is called a leaf atom.



For cross-platform purposes, all data in a QT atom is expected to be in big-endian (Macintosh) format.

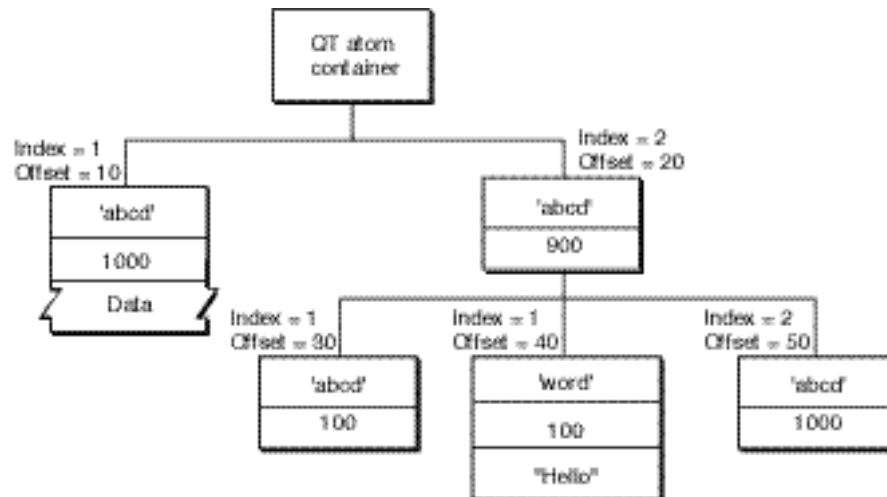
You use QuickTime calls to create and manipulate QT atoms. Each atom has a type code that determines the kind of data stored in it. By storing data this way, the number of data structures and complexity you need to deal with is greatly minimized. Because the data is typed, you can ignore data that's not of interest when interpreting atom data.

QT ATOM HIERARCHIES

QT atoms can nest indefinitely forming hierarchies that are easy to pass from one process to another.

You can search and manipulate QT atoms, as well as search through QT atom hierarchies until you get to the leaf atoms, then read the leaf atom's data from its various fields.

The diagram below shows a QT atom container that has two child atoms. The first child atom (offset = 10) is a leaf atom that has an atom type of 'abcd' , an ID of 1000, and an index of 1. The second child atom (offset = 20) has an atom type of 'abcd' , an ID of 900, and an index of 2. Because the two child atoms have the same type, they must have different IDs. The second child atom is also a parent atom of three atoms.



The first child atom (offset = 30) has an atom type of 'abcd' , an ID of 100, and an index of 1. It does not have any children, nor does it have data. The second child atom (offset = 40) has an atom type of 'word' , an ID of 100, and an index of 1. The atom has data, so it is a leaf atom. The second atom (offset = 40) has the same ID as the first atom (offset = 30), but a different atom type. The third child atom (offset = 50) has an atom type of 'abcd' , an ID of 1000, and an index of 2. Its atom type and ID are the same as that of another atom (offset = 20) with a different parent.

REFERENCES

QUICKTIME ATOMS

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/REF/refMovieToolbox.b.htm#pgfid=16143>

MOVIE TOOLBOX:DATA TYPES

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmMTData.htm>

QTATOMCONTAINER-BASED DATA STRUCTURE DESCRIPTIONS

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/REF/refAppD.htm>

SPRITES

A sprite is a graphical object that is specified once, then animated by making calls to special QuickTime routines. Most other animation techniques are based on bitmaps or vectors that must be constantly redefined. Sprites consume much less memory because they are defined only once.

A QuickTime sprite is a freestanding animated graphic. If the user's screen is like a stage, sprites are like actors that can show up anywhere, move around, and change form.

The advantage of sprites is that they are easy to control and don't use much memory. You define each sprite once and then animate it with simple commands, instead of defining dozens of byte-gobbling bitmaps, one for each location.

In QuickTime, sprites are defined by means of QT atoms. A sprite can be created from almost any data source: a bitmap, an MPEG video file, etc. Once a sprite is created from whatever source and placed in a sprite track, it is controlled from then on by QuickTime calls.

Each sprite has properties that describe its location and appearance at a given time. During the course of an animation, your code makes calls to QuickTime that modify a sprite's properties.

MOVIE SPRITES

A group of sprites can live in either a movie track or a sprite world. In a movie, a sprite can appear alone or over a video track and can act as a graphic or as an interactive button.

Sprites within movies live in a sprite track and are animated by the QuickTime sprite media handler.

As with sprites created in a sprite world, sprites in a movie's sprite track have properties that define their locations, images, and appearance. However, your code mostly uses a different process and different QuickTime calls to create and animate a sprite track.

A sprite track is defined by one or more key frame samples, each followed by any number of override samples. Each key frame sample contains all of the images used by the sprites, and different sprites may share image data.

A good example of movie sprites are the penguins shown in the diagram below where the arrows are wired sprites that act as buttons.



REFERENCES

INTRODUCTION TO WIRED MOVIES, SPRITES, AND THE SPRITE TOOLBOX
<http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmWiredIntro.htm>

EFFECTS

QuickTime provides built-in support for the 133 standard video effects defined by the Society of Motion Picture and Television Engineers (SMPTE), plus another 24 effects created by Apple. For a complete description of the available effects, see the reference at the end of this section.

You can use these effects to dress up the transitions between QuickTime movie frames or still images. The inputs to an effect or transition can be frames of a QuickTime movie or they can be graphics worlds containing images of any type.

QuickTime provides three basic kinds of video effects:

- Filter effects (tinting, embossing, blurring)
- Transition effects (wipes, irises, dissolves, etc.)
- Generator effects (flames, rippling water, etc.)

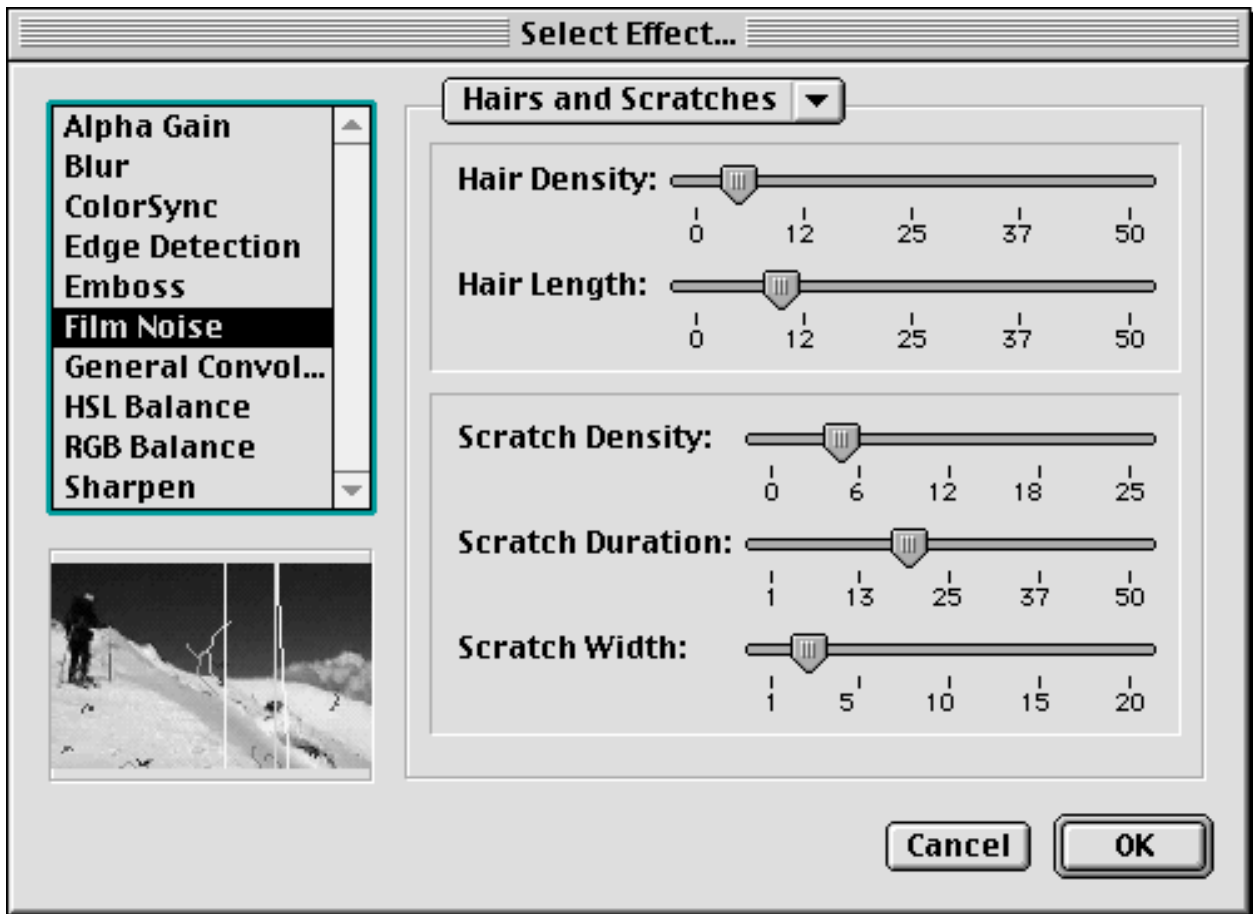
Because visual effects are calculated and executed at runtime, they are compact to download. Also, you don't need to predict the exact appearance of the movie when they execute.

VIDEO EFFECTS IN MOVIES

Video effects are implemented as QuickTime components. To use an effect component in a QuickTime movie, you add an effect track to the movie. Each effect track has two parts: an effect description and an input map. The effect description specifies which effect component will be used; the input map specifies which of the movie's other tracks will be used as content sources for the effect.

For each built-in effect, QuickTime provides a dialog box that you can display to let the user set its parameters. This dialog box is shown in the diagram below.

Applications can use the `QTCreateStandardParameterDialog` function to display the standard parameters dialog box and allow a user to choose an effect, adjust the settings for that effect, and see a preview of the effect with the selected settings.



EFFECTS OUTSIDE MOVIES

You can also apply a video effect outside a movie—for example, a pair of still images. To do this you define a transition between two graphics worlds.

Besides setting up the effect, you must provide code to run it. Because effect components are varieties of image decompressors, the code to execute an effect outside a movie is the same code you would use to decompress and display an image.

REFERENCES

BUILT-IN QUICKTIME VIDEO EFFECTS

<http://developer.apple.com/techpubs/quicktime/qt4beta/REF/refEffects.3d.htm#pgfid=324>

INTRODUCTION TO QUICKTIME VIDEO EFFECTS

<http://developer.apple.com/techpubs/quicktime/qt4beta/REF/refEffects.2.htm#pgfid=12185>

CROSS-PLATFORM DEVELOPMENT

IMPLEMENTATION

QuickTime is truly multiplatform; its application-programming interface is essentially the same on Macintosh and Windows. Code written using QuickTime APIs will compile and run correctly on both platforms. Additionally, QuickTime for Java lets you write Java code and access QuickTime on both Macintosh and Windows.

On Macintosh, QuickTime is implemented as a set of system extensions that will run under System 7.5.5 or later on Power Macintosh computers.

QuickTime on Windows 95, 98 and NT is implemented as a dynamic-link library supporting PC compatibles with a 66MHz 80486 or better processor. It also supports Microsoft DirectDraw and DirectSound for improved performance.

QuickTime for Java is implemented as a set of classes and methods that give you access to QuickTime's functionality. QuickTime for Java requires a Java Virtual Machine Version 1.1 or greater and requires QuickTime itself.

CROSS-PLATFORM ISSUES

Although Macintosh developers working with QuickTime on Windows will feel surprisingly at home, there are a number of issues Windows developers new to QuickTime should be aware of.

Graphics Environments – QuickTime draws to the screen by using QuickDraw. QuickDraw depends on a graphics structure called a graphics port. Windows uses the device context (DC) for similar purposes, but a DC is specific to a particular device such as a window

or printer while a graphics port is global to all drawing operations at a given time.

Data Containers – Although Windows uses the concept of resources, they are less important to the system's software architecture than their Macintosh counterparts. These structured items of data can be loaded on demand to help determine a program's behavior.

Data Type Codes - QuickTime uses four-character codes to identify items such as track types, media types and component types. Internally these are simply 32-bit integers. In source code a four-character string in single quotes, such as 'abcd' represents them.

File Forks – Macintosh files consist of two separate sections logically joined under a single file name. The first section is called the data fork and consists of a single stream of bytes intended to be read sequentially; this corresponds to what Windows generally treats as a file. The second section is known as the resource fork and contains individual resources that are accessed by means of a four-character resource type and integer resource IDs.

Messaging – Macintosh events are similar in concept to Windows messages and carry essentially the same information. However, a Windows message is directed to a specific window while Macintosh OS events are addressed globally to the currently running application.

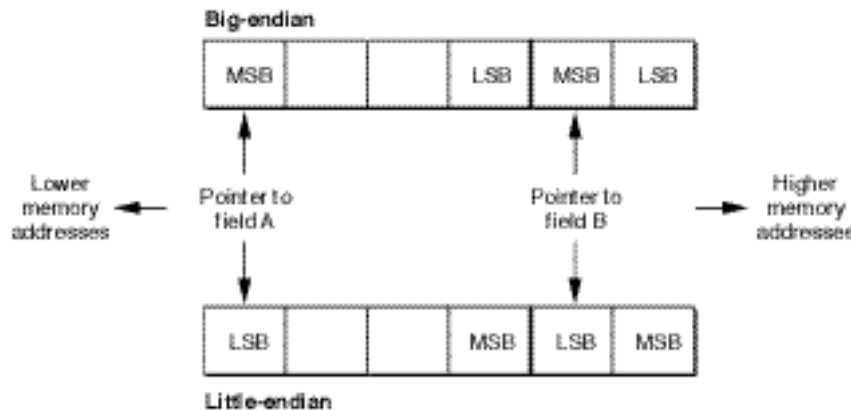
Windows Registration – QuickTime designates a window by a pointer to a Macintosh window record. This window record contains its own graphics port. On the Windows platform a window is normally designated by an `HWND` handle. Before QuickTime for Windows can draw in a window, the window must be registered with the QuickTime Media Layer using the `CreatePortAssociation` API.

File Typing – Every Macintosh file is stamped with a four-character file type and a four-character creator signature. The type and creator signatures identify the application program the file belongs to. Windows uses the three-character DOS file extensions.

String Formats – While Windows uses C-style strings (terminated by a null character), QuickTime uses strings in Pascal form (preceded by a 1-byte length count). QuickTime provides conversion utilities for these two formats, `CopyPascalStringToC` and `CopyCStringToPascal`.

ENDIAN ISSUES

Multibyte data fields can be referenced a couple of ways in memory. One known as big-endian consists of field addresses pointing to the most significant byte while the second, known as little-endian contain field addresses pointing to the least significant byte.



Native QuickTime software uses big-endian (MSB) addressing (because of its origins on Mac OS) while Windows uses little-endian (LSB).

For the most part QuickTime takes care of the endian conversions for you automatically. You may need to use conversion utility APIs only when you add private multibyte data to a structure QuickTime will not know about, or when you interpret media data yourself.

Utility functions for byte flipping can be found in the `Endian.h` header file that comes with the QuickTime Interfaces and Libraries.

REFERENCES

QUICKTIME FOR WINDOWS PROGRAMMERS

http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/tp_rm_qtforwindows.htm

MAC OS FOR QUICKTIME PROGRAMMERS

http://developer.apple.com/techpubs/quicktime/qtdevdocs/INMAC/MACWIN/tp_macosqt_oview.htm

QUICKTIME FOR JAVA

http://developer.apple.com/techpubs/quicktime/qtdevdocs/Java/tp_java_digest.htm

DEVELOPER RESOURCES

Apple provides numerous resources to help engineers learn about QuickTime technologies to make your job easier. This includes documentation, sample code, development kits, technical support, software seeding, and testing/compatibility.

QUICKTIME DEVELOPERS WEB SITE

You will find many useful links, cross-platform tools, resources and suggestions for starting points here.

<http://developer.apple.com/quicktime/index.html>

QUICKTIME WEB SITE

If you want to download QuickTime, check out the hot picks, movie trailers and find links to QuickTime content, content creation and authoring, this is where you want to start.

<http://www.apple.com/quicktime/>

APPLE QUICKTIME API DOCUMENTATION

Includes information on QuickTime for Macintosh and Windows, QuickTime VR, and the Sound Manager. Also links to Ice Floe engineering documents, Technotes and Q&As related to QuickTime.

<http://developer.apple.com/techpubs/quicktime/quicktime.html>

SAMPLE CODE

Contains QuickTime samples ranging from the basics of viewing movies to video capture, components, importers/exporters to wired movies and sprites.

http://developer.apple.com/samplecode/Sample_Code/QuickTime.htm

QUICKTIME SOFTWARE DEVELOPMENT KIT (SDK)

The QuickTime SDK provides everything you need to create applications, interactive content, and more. The SDK includes Macintosh and Windows versions of QuickTime Pro and bundles the Interfaces and Libraries, Sample Code, Technical Documentation and Tools on one convenient CD. You can also choose an SDK bundle that includes a QuickTime book with CD-ROM authored by members of Apple's QuickTime team.

<http://developer.apple.com/products/qt4sdk.html>

LETTERS FROM THE ICE FLOE

These are technical notes from the frozen home of the QuickTime Engineering team. Each document contains some useful tips on taking best advantage of QuickTime.

<http://developer.apple.com/quicktime/icefloe/index.html>

QUICKTIME DEVELOPER SERIES BOOKS

Discovering QuickTime:

An Introduction for Windows and Macintosh Programmers

by George Towner

Academic Press/Morgan Kaufmann

ISBN 0120596407

Written for programmers, multimedia designers, and everyone interested in the latest media technology, this book gives you a step-by-step introduction to QuickTime programming, from movies and animation to streaming video on the Internet. The CD-ROM in the back provides working applications, sample code, and the essential programming resources you need to get started.

http://www.mkp.com/books_catalog/0-12059-640-7.asp

QuickTime for Java: A Developer Reference

by Tom Maremaa and William Stewart

Academic Press/Morgan Kaufmann

ISBN 0123054400

This book is an essential quick reference for the QuickTime and Java programmer. It provides the reader with a wealth of programming examples as well as a handy reference that provides an in-depth, class-by-class description of the API. The authors are part of the original QuickTime engineering team that pioneered and developed QuickTime for Java. A CD-ROM at the back of the book provides working sample code and other resources, so you can get started right away building your own Java applications and applets.

http://www.mkp.com/books_catalog/0-12305-440-0.asp

QuickTime for Web: A Hands-On Guide

by Steven Gulie

Morgan Kaufmann

ISBN 012471255X

This is the complete guide to creating QuickTime content and putting it on the Web. It covers everything-from the right way to embed a movie in a Web page to the best techniques for combining scrolling text, Flash animation, MP3 audio, live streams, and virtual reality-in an engaging and easy to follow style.

Written for multimedia authors, Web heads, and anyone who wants to include sound or video on a website, this book provides clear, detailed, and often humorous guidance. This book contains a wealth of information you won't find anywhere else.

http://www.mkp.com/books_catalog/catalog.asp?ISBN=0-12471-255-X

NEWSGROUPS & MAILING LISTS

Mailing lists run by Apple & others. Includes QuickTime-API, QuickTime-Talk, QuickTime-VR and QuickTime-Java.

<http://lists.apple.com/>

QUICKTIME TOOLS

Authoring and Development tools provided by Apple. These include Dumpster, Hint track profiler, MakeRefMovie and others.

<http://developer.apple.com/quicktime/quicktimeintro/tools/index.html>

THIRD-PARTY TOOLS

There are hundreds of third-party applications and hardware solutions for capturing, editing, and integrating QuickTime content (including QuickDraw 3D and QuickTime VR).

<http://www.apple.com/quicktime/authoring/thirdpartyapps.html>

ADC

The Apple Developer Connection programs offer easy access to technical and business resources for Apple platform developers anywhere in the world. Join one of our membership programs for the benefits and services you need to develop, distribute, and market your products. Some benefits and services are available only when bundled with a membership. You may also purchase some individual developer products and services on as-needed basis.

<http://developer.apple.com/membership>