MacBALSA

Version Aleph

Marc H. Brown

Preface

This manuscript and the accompanying MacBALSA software are predicated on the thesis that computer algorithms are best understood through interactive simulations: that "a picture is worth a thousand words;" that a moving picture is worth much more; and that the ability to interact with a movie is invaluable. The intent is to exploit the capabilities of the computer as a communications medium, bringing to life concepts presented in conventional textbooks, and transforming you, the reader, from a passive observer into an active participant in the study of a variety of fundamental and important algorithms and data structures.

The manuscript contains documentation for using the software, and the software contains the MacBALSA runtime environment with interactive simulations from a variety of fundamental algorithms in the realms of sorting, computational geometry, graphs, and binpacking. Both are preliminary and not ready for wide-spread distribution; they are available by contacting me directly.

What I hope will be ready in the near future for general distribution is an "exploratorium for algorithms:" a book containing exercises that utilize animated algorithms within the MacBALSA environment. The exploratorium will not replace a standard textbook, but rather will complement one. The software will contain interactive simulations of textbook algorithms (as in the software accompanying this manuscript), and the exploratorium book will contain exercises that make use of the software. The software could be used by an instructor during a lecture (in a lecture hall with Macintoshes or appropriate projection equipment), showing computer "videotapes" in lieu of static diagrams on blackboards or viewgraphs; by students for homework or during a laboratory session; or by an individual on his own.

The algorithms I intend to cover are from the domains of sorting, searching, computational geometry, graphs, and fundamentals. The complete list is as follows:

- **Sorting:** selection sort, insertion sort, bubble sort, shaker sort, Shellsort, quicksort, radix sorting, and heapsort.
- **Searching:** array methods, binary search trees, balanced trees, and hashing.
- **Computational Geometry:** closed paths, point inclusion, convex hulls, range searching, line intersection.
- **Graphs:** depth-first and breadth-first traversals, union-find, minimal spanning tree, transitive closure, and matching.
- **Fundamentals:** elementary data structures such as arrays, lists, matrices, stacks, queue, and trees, and recursive and dynamic programming techniques.

The accompanying software contains an almost complete set of computational geometry and sorting algorithms, as well as a taste of the graph algorithms, that the envisioned exploratorium will contain.

It's possible, though unlikely, that the exploratorium will include the C source code for the algorithms (requiring the use of the Symantec's LightSpeed C compiler), thereby enabling a reader to modify the algorithms, and even code his own, using the MacBALSA displays. Regardless, it will be possible, as it currently is, for a user to create his own exercises and videotapes.

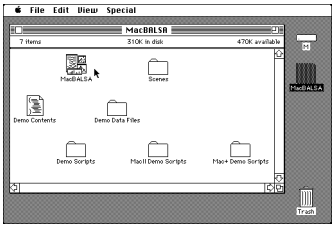
The algorithms and visualizations in MacBALSA are, for the most part, independent of the implementation appearing in any specific textbook. Most of the algorithms have been coded based on the implementation appearing in Robert Sedgewick's *Algorithms, 2nd Edition* (Addison-Wesley, 1988), and you are encouraged to consult the text for more information. Many of the designs of the visualizations are based on joint work with Bob in the "electronic classroom" project at Brown University in 1983. Bob's support and assistance (especially for later serving as my Ph.D. thesis advisor) are gratefully acknowledged.

Finally, I welcome your feedback. This project is under active development and you have an opportunity to mold it. How would you make use of the envisioned exploratorium? Why wouldn't you use it? What visualizations were not effective? Why not? Which algorithms should be covered and which should not? Would you be willing to review (read: be a guinea pig) the exploratorium manuscript and software when it's available?

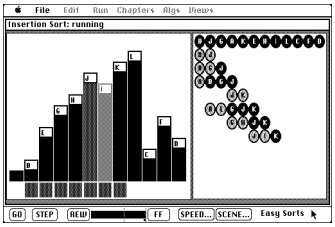
Marc H. Brown
Box 186 Palo Alto, CA 94302
July 1989

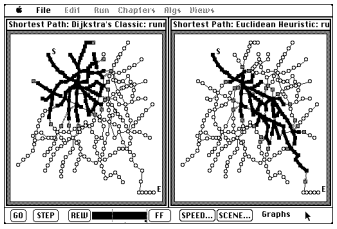
Getting Started

Insert the accompany disk into a Macintosh and double-click on the MacBALSA application. MacBALSA is configured so that it will continuously run a *videotape* showing a sampling of the animated algorithms it contains.

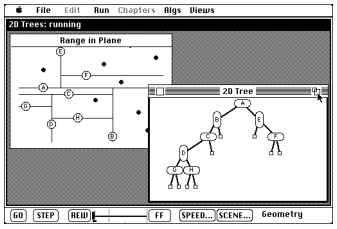


While the videotape is playing, there is a *control panel* at the bottom of the screen. You can stop the videotape at any time by a mouse click or keystroke, and resume it using the GO button in the control panel. The STEP button allows you to advance through the algorithm, one logical unit at a time. The SPEED button allows you to slow down the videotape (and, in some cases, speed it up); the speed is only active during the current "segment" of the videotape.

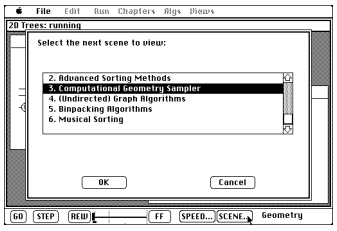




The screen is partitioned into one or more non-overlapping windows, called *algorithm windows*. These are containers for (potentially overlapping) *view windows*. The screen at the side contains two algorithms, each with one view; the screen above contains one algorithm with two views. As we shall see later, you can manipulate view windows following familiar Macintosh paradigms.

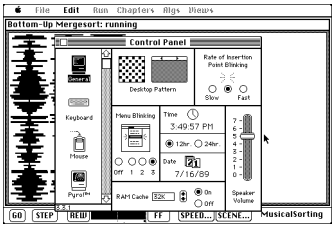


(OK, if you really cannot resist the temptation, go ahead and try it. You'll first need to double-click in a view window to cause it to display its *dressing*. When you are ready, use the GO or STEP button to resume the videotape. Because the control panel is a Macintosh window, you'll need to click twice: once to bring it to the top as the *selected* window, and the second time for it to listen to clicking the GO or STEP button.)



The videotape you are watching is composed of six *scripts*. An abbreviated name of the current script appears at the far right in the control panel. MacBALSA will play the scripts sequentially; you can jump to a different script using the Scene... button in the control panel. (Yes, the button should be renamed Script....) Each script is composed of a number of *segments*. The REW and FF buttons allow you to stop the current segment and return to the previous segment or advance to the next one.

The last script in the default videotape uses the Macintosh's sound capabilities. You can control the volume using the standard Macintosh `ControlPanel` desk accessory.



When you are finished, you can quit the MacBALSA application using the `QUIT` button in the `File` menu. Alternatively, you can stop watching the videotape using the `Stop Reading` button in the `File` menu. In this case, the control panel at the bottom disappears and you will be in the MacBALSA interactive environment. The remainder of this manuscript describes how you make use of the interactive environment.

What Really Happens At Startup

When MacBALSA is invoked, it looks for the file named `DemoContents -machine` (where `machine` is the type of Macintosh currently running: II, Plus, SE, etc.) in the folder containing MacBALSA. If the file is found, MacBALSA assumes it defines a videotape. MacBALSA then plays the videotape, and continues to do so until the user intervenes as we have described. If the videotape file for the machine is not found, MacBALSA looks for the videotape file `DemoContents.` If that isn't found, then MacBALSA looks in the `Scenes` folder for the file `Last.` If that file is found, MacBALSA restores the interactive environment to what is described in `Last.` As you might expect, whenever MacBALSA terminates, it snapshots the environment into `Last.` If none of these files is found, or if you start MacBALSA while holding the mouse button down, MacBALSA initializes the interactive environment with appropriate defaults.

The videotape file `DemoContents` consists of pairs of strings. The first string is the name of the file containing the script; the second string is the name that the user sees when the list of scripts is presented.

The MacBALSA disk also contains a folder `DataFiles` with a number of input files that some of the computational geometry and graph algorithms

in the default videotape reference. Don't change the names of this folder or of the files in it.

The disk does not contain a Macintosh `SystemFolder,` but there's enough room to store one. You'll need one if you wish to run MacBALSA on a Macintosh without a hard disk and with only one floppy.

Fundamental Concepts

A program being animated within MacBALSA is split into the *algorithm* itself and multiple *views* of the algorithm in action. At runtime, MacBALSA allows the user to select which views to use; the algorithm executes without knowing which views have been selected. Conversely, each view executes without knowing which algorithm is driving it. This model leads to clean and modular implementations of both algorithms and views: algorithms do not get involved in any graphics, whereas views are not involved in any algorithm details. This model also facilitates sharing views among many (diverse) algorithms.

Optionally associated with each algorithm and view are *parameters* the user can manipulate. For example, the parameters for a text compression algorithm might be the name of file to be compressed and the name of a database of word frequencies. Parameters for a typical view used by this algorithm might include what colors and stipple patterns to use for graphically representing the original and compressed text.

Algorithms and views communicate by *interesting events*—procedure calls to the MacBALSA event-dispatcher placed in the algorithm source code. Interesting events identify fundamental algorithm operations. For example, many sorting algorithms are based on the interesting events "Compare" and "Exchange," indicating when an algorithm examines an element of the array being sorted and when an algorithm exchanges two elements of the array, repectively. When the MacBALSA event-dispatcher is called by the algorithm, it invokes each view on the screen, passing the name of the event and any arguments the algorithm specified. Each view updates itself to reflect the algorithm operation. Interesting events are not only important concepts to programmers preparing animations, but they are also important to users because events are the abstraction through which algorithm execution is controlled.

The Runtime Environment

The MacBALSA environment comprises three basic facilities. It is a *manager* for selecting and manipulating the algorithms and views; an *interpreter* for controlling and analyzing algorithm execution; and an *assistant* for saving and restoring the environment, and for authoring and replaying scripts.

- As a manager, MacBALSA allows the user to tile the screen with *algorithm windows*, and to associate with each algorithm window an algorithm from among those in the system, and one or more views from among those that are meaningful for the algorithm. Each view is displayed in its own window on the screen, contained within its algorithm window; *view windows* are, to a first approximation, conventional Macintosh overlapping windows and can be manipulated as such. Finally, the manager allows the user to observe and manipulate the parameters associated with each algorithm and view.

- As an interpreter, MacBALSA allows the user to control algorithm execution based on the interesting events. The user can single-step, set breakpoints, and specify the amount of time each event should consume. In other words, three quantities are associated with each event: a *step* flag ("is this event in the set of event that constitute the next 'step' of the algorithm?"), a *stop* amount ("how many of this event should occur before a 'stop' occurs?"), and a *cost* ("how much 'time' should each occurence of this event take to execute?"). The cost is most useful when running multiple algorithm simultaneous, and for representing different models of computation. The facilities for analyzing the algorithm are counts of the number of times each event has occurred and the total cost consumed by all occurences of each event.

- As an assistant, MacBALSA has facilities for *scenes* and *scripts*. A scene is recorded as a textual description of the environment (i.e., which algorithms and views have been chosen; window locations; the value of parameters to each; and the step, stop and cost associated with each event in each algorithm). A script is a collection of scenes, recorded each time the user starts running an algorithm. A script can function as a "virtual videotape;" however, it is more powerful than a conventional VCR because the "viewer" has control during playback over the collection of views and the interpreter values for steps, stops, and costs.

MacBALSA is used in a "setup and run" style of interaction. That is, the user creates a scene, either interactively using the manager facilities or by

restoring a one from a file, and then runs the scene. While the algorithm is running, the user can suspend the execution, change aspects of the scence, and then resume it or abort the remainder of the execution.

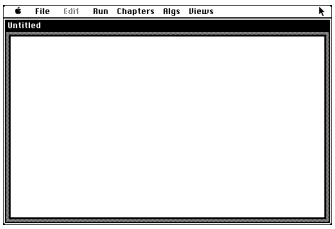
Thus, MacBALSA has three modes: *setup*, *running*, and *suspended*. During setup mode, all aspects of the scene can be set. During running mode, the user just watches the execution. During suspended mode, only *presentation* and *temporal* aspects of the scene can be changed. The presentation aspects concern the size, location, and parameters associated with the view windows. The temporal aspects concern the step, stop, and cost values associated with each interesting event. The aspects of the scene that cannot be changed while the system is in the suspended mode, the *structural* aspects, are which algorithm is in each algorithm window, and the parameters specified for each one.

A Tutorial

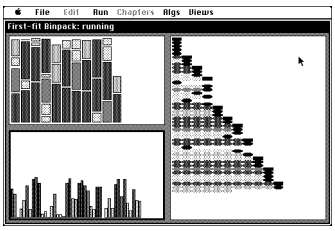
This chapter walks you through a demonstration of using the MacBALSA interactive environment. The first part is concerned with the basic features of the system, and the second part describes the advanced features.

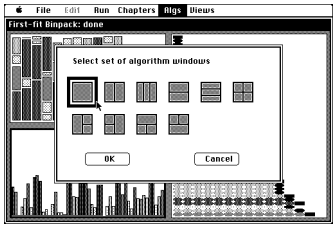
Basic Features

Step 1. Select a chapter. We'll pick the `Binpacking` chapter. MacBALSA restores the scene to what it was the last time we were in the the `Binpacking` chapter. Because this is the first time that we are visiting `Binpacking` on this disk, the default algorithm, `Untitled` is selected.

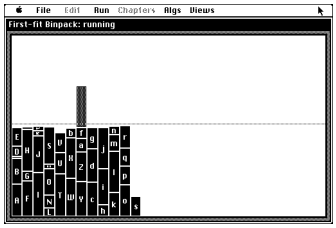


Digression for the impatient reader: Pick an algorithm from the `Algs` menus, and then `Go` from the `Run` menu. Stop it with a mouse click or keystroke, or slow it down using the `Speed...` item in the `Run` menu. Resume using `Go` again. Cancel the execution with `Reset` in the `Run` menu, or give it different data using `Parms...` in the `Algs` menu. Add views using `Windows...` in the `Views` menu. End digression.

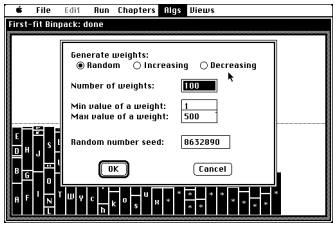




Step 2a. Choose a configuration of algorithm windows. Use the `Windows...` command in the `Algs` menu. For now, we'll run a single algorithm.

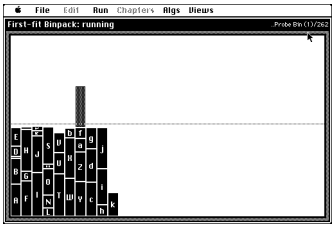


Step 2b. Decide which algorithm goes in each algorithm window. Select the algorithm window by clicking on it (its banner becomes black), and then pick an algorithm from the `Algs` menu. We'll pick `First-fitBinpacking.` This binpacking algorithm examines the bins from left to right, and inserts the new weight in the first bin it finds with enough room for it.

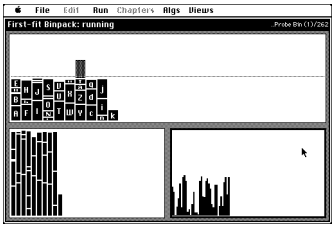


Step 3. Specify data for the algorithm to process. Use the `Parms...` command in the `Algs` menu. In general, algorithms have reasonable default parameters, so this step isn't essential before running an algorithm for the first time. For binpacking algorithms, we can specify the number of weights to generate, the range of the weights (1 is the minimum, and 1000 is the maximum), a seed for the random number generator to use (0 causes the generator to use the Macintosh clock as its seed), and whether the weights should be sorted in increasing or decreasing order.

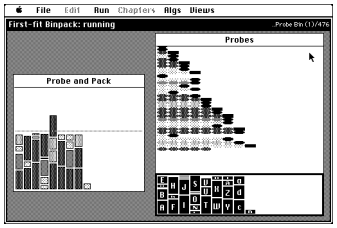
The algorithm is now ready to be run using the commands in the `Run` menu. The `Go,` `Trace,` and `Reset` commands are probably the most useful run-styles at this point. You can double-click on the algorithm window, and tracing information appears at the far right. The tracing information is the name of the most recent event generated in the algorithm, a parenthesized number, followed by a slash and then another number. The rightmost number reports the number of events that the algorithm has generated. The parenthesized number is the number of times the current event has happened since the last time it was reported in the algorithm banner.



Step 4. View Windows. We can change views by selecting a view (its border is highlighted) and then picking a new view from the `View` menu. Alternatively, we can create more view windows using the `Windows...` command in the `Views` menu. Some views cannot show the algorithm unless the view has been open since the algorithm began running; such views display an appropriate warning.

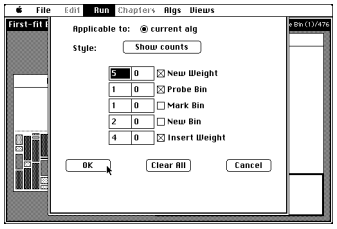


Step 5. View Parameters. Use `Parms...` in the `Views` menu. Most views do not have any parameters. The screen to the side shows a few instantiations of the `Packing` view, each with a some different view parameter settings.

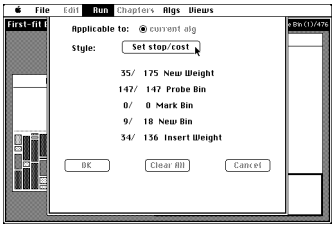


Step 6. View Window Management. View windows, unlike algorithm windows, can be moved around and resized. To do so, double-click on the window to get it to display (or erase) its *dressing*. As expected, a view window is moved by dragging the banner, and it is removed using the close box in the left part of the banner. Unlike standard Macintosh guidelines, the window is resized by picking up the magnet in the right part of the banner and touching it to any corner or side to be moved. Double-clicking the magnet causes it to act like a standard Macintosh zoom box.

Advanced Features



Step 7. Controlling Execution. MacBALSA's interpreter allows you to control execution in terms program-specific units, namely the interesting events with which the algorithm has been annotated. The `Stops...` command in the `Run` menu displays a dialog box for observing and specifying cost value, stop value, and stepping flag of each event. In the `Run` menu, `Go` stops at the next stop-point; `GoGo` pauses at stop-points; `Step` stops at the next step-point; and `Trace` pauses at step-points. `Reset` terminates the execution of the algorithms.

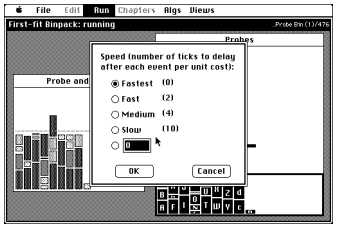


Step 8. Analyzing Algorithm Performance. To precisely measure an algorithm's performance, we can find out the number of times each interesting event has occurred by issuing the `ShowCounts` button in the previous dialog. The dialog is changed as shown here, and the numbers in the left are the number of times each event has occurred (the right column will be explained in moment). In practice, interesting events correspond to an algorithm's fundamental operations, and these numbers are often of immediate interest to many people studying an algorithm's performance.

It is reasonable to wonder why we might specify a cost for each event. Intuitively, the cost is a measure of how much computer "time" the interpreter should allocate for the event. This is useful for simulating an algorithm on two different models of computation: for example, in one model, data movement might be expensive relative to data access, whereas in another model, these operations might have the same relative cost.

MacBALSA schedules the multiple algorithms in a round-robin fashion. During each time-slice, an algorithm executes until it reaches an event and then that event is broadcast by MacBALSA to all views so they can update themselves. If the cost for the event is W, then that algorithm misses its next W-1 time-slices.

Two additional terms are helpful for a complete understanding of MacBALSA's scheduler: the *event count* of an algorithm is the number of events that have occurred, and the *virtual time* is the weighted sum of the events. It follows directly from MacBALSA's scheduling algorithm that while multiple algorithms are running, their virtual times are identical. However, their event counts may not be. It is the virtual time, not the event count, that is displayed as part of the algorithm window dressing. The right column of numbers in the figure above shows how much each event contributes to an algorithm's virtual time.



Step 9. Speed. The `Speed...` button in the `Run` menu lets us slow down the animation by specifying how long MacBALSA should wait per unit cost after each event. For example, if the speed factor is 3, and event `Foo` has a cost of 5, MacBALSA will pause 15 ticks (each tick is 1/30th of a second) after each event `Foo` happens.

Miscellaneous Features

The `File` menu contains a number of commands that are quite useful. The `Quit` button causes MacBALSA to terminate. The `SaveScene...` button records the current scene into a file, and `RestoreScene...` reads a scene from a file. The scene file is textual and the format is self-explanatory.

The `StartAuthoring...` command causes MacBALSA to start recording a script: each time an algorithm begins running, the scene is automatically saved as a *segment* of the script file. The script can be replayed using the `StartReading...` command. While the script is replaying, many of the commands for control view window management and the interpreter are available, as we saw in the Getting Started chapter. The script file is textual and is essentially a concatenation many scene files, with some extra information inserted to denote the start and end of each segment.

Menus

MacBALSA has six menus: `File`, `Edit`, `Run`, `Chapters`, `Algs`, and `Views` (from left to right). We now consider the contents of each one.

`File` ***Menu***

There are a number of experimental and debugging items in this menu that are not listed below.

`SaveScene...`	Store the current scene into a file.
`RestoreScene...`	Restore a scene from a file. This item is only valid in setup mode.
`StartAuthoring...`	Start creating a script and store it into a file. This item is only valid in setup mode. While a script is under construction, this item changes to `Stop Authoring.`
`StartReading...`	Start replaying a script stored in a file. This item is only valid in setup mode. While a script is replaying, there is a control panel at the bottom of the screen and this item changes to `Stop Reading.`
`Quit`	Exit the MacBALSA application. The current scene is stored in the file `Last` in the `Scenes` folder. It is restored the next time MacBALSA is entered, unless the mouse button is held down while MacBALSA is starting, or the file `Demo Contents` exists in the same folder that contains the MacBALSA application.

`Edit` ***Menu***

This menu contains the standard Macintosh commands for text editing, such as cut, copy, paste, and undo. Currently, no MacBALSA views support text or graphics editing.

Run ***Menu***

MacBALSA allows execution to be controlled in terms program-specific units, namely the interesting events with which the algorithm has been annotated. For each event, we can specify whether it should be used as a stop-point or as a step-point. A stop-point is analogous to a breakpoint in conventional debuggers: when the specified event has occurred the specified number of times, the interpreter pauses execution. A step-point is a generalization of a conventional debugger's notion of "step to the next line." That is, when we issue the "single step" command, the interpreter advances to the next event that is specified as a step-point.

`Reset` Abort all algorithms currently running. The system will then be in setup mode.

`Go` Run the algorithms, stopping at the next stop-point.

`GoGo` Run the algorithms, but just pause at the next stop-point.

`Step` Run the algorithms, stopping at the next step-point.

`Trace` Run the algorithms, but just pause at the next step-point.

`Stops...` Display a dialog with the cost, stop, and step values for each event. The `ShowCounts` button in the dialog causes the dialog to display the number of times each event has happened and the weighted total of each event.

`Speed...` Display a dialog for setting how much time to pause after each event has happened per unit cost.

`Chapters` ***Menu***

The items in this menu are the names of chapters. Each chapter is essentially just a way to group similar algorithms so that the `Algs` menu does not become unwieldy. The checked item is the selected chapter. The items in this menu are valid only in setup mode.

When a new chapter is selected, the current scene is stored in a file, and the scene from the last time the chapter was exited is restored. The files containing the chapter scenes are located in the `Scenes` folder; the file names are the names of the chapters.

`Algs` ***Menu***

`Parms...` Display a dialog with the parameters for the selected algorithm. This item is valid only if the selected algorithm has parameters. While algorithms are running, this item becomes `Show Parms...,` indicating that the values of the parameters may be observed, but they may not be modified.

`Windows...` Display a dialog showing ways to arrange new algorithm windows. Currently, selecting a new configuration reinitializes each algorithm window to the `Untitled` algorithm. This item is valid only in setup mode.

The remaining items in this dialog are the names of the algorithms that are valid for the selected chapter. The checked item is the name of the selected algorithm. Selecting one of them will install it into the currently selected algorithm window. These items are valid only in setup mode.

`Views` ***Menu***

`Parms...` Display a dialog with the parameters for the selected view. This item is valid only if the selected view has parameters.

`Windows...` Display a dialog showing ways to arrange view windows for the selected algorithm. Currently, each algorithm can have at most four view windows at any time. Unfortunately, there is no way to just create a single new view window, other than by specifying a new arrangement in this dialog.

The remaining items in this menu are the names of the views that are valid for the selected algorithm. The checked item is the name of the selected view. Selecting one of these items will install it into the currently selected view window. Some views will not update until the next time the algorithm is run; other views are more robust and will display valid information even if installed after an algorithm has begun executing.

Known Bugs

- Selecting the current chapter causes all of the window to disappear and the menu choices to be dim.

  Remedy: Exit MacBALSA and delete the file `foo` in the `Scenes` folder, where `foo` is the name of the chapter.

- The "videotape" file `DemoContents` cannot be replayed from within the MacBALSA environment.

  Remedy: None. The individual scripts can be read however. They reside in the folder `DemoScenes.`

- The term "scene" is used to designate scripts comprising the videotape.

  Remedy: None. Wait for the next version.

More Info...

If you'd like to be notified of updates to MacBALSA, or to the availability of the "exploratorium" described in the Preface, complete the following information:

Name _____

Company/School _____

Address _____

City _____ State/Province _____

Zip/Postal Code _____ Country _____

FAX _____

Electronic Mail _____

and return it to:

Marc H. Brown
Box 186
Palo Alto, CA 94302

If you have any comments you'd like me know about, please use the back side.