reference

GUIDE•

mTropolis Reference Guide

© 1995 mFactory, Inc. All rights reserved.

This publication, or parts thereof, may not be reproduced, stored in a retrieval system, or transmitted in any form, by any method, for any purpose, without the prior written permission of mFactory, Inc. ("mFactory").

For conditions of use and permission to use these materials for publication in other than the English language, contact mFactory.

mFactory Trademarks

mFactory, the mFactory logo, mTropolis, mFusion, and "Move The World" are trademarks of mFactory, Inc.

Third-Party Trademarks

All brand names, product names or trademarks belong to their respective holders.

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

To the extent this manual ("Manual") was purchased in connection with the purchase of mFactory Software, the mFactory License Agreement sets forth all applicable warranty and damage provisions concerning the Manual.

To the extent purchased independent of any mFactory Software, however, mFactory hereby states that it: (i) makes no representations or warranties with respect to the accuracy or completeness of the contents of this publication or the information contained herein; (ii) specifically disclaims all warranties of merchantability, fitness for particular purpose and non-infringement of third party intellectual property rights, and their equivalents under the laws of any jurisdiction; (iii) assumes no responsibility for any errors or omissions in this publication or the information contained herein; and (iv) disclaims responsibility for any injury or damage resulting from the use of the information set forth in publication.

mFactory further reserves the right to, at any time and without notice: (i) make changes to the information contained in this publication; (ii) discontinue the use, support or development of any products or information described or otherwise set forth in this publication; and (iii) discontinue or decide not to release any of the products described in this publication. The maximum amount of damages for which mFactory will be liable arising from the use of this publication or the information described herein is the purchase price, if any, for this publication. In no event shall mFactory be liable for any loss of profit or any other commercial damages including, but not limited to, special incidental, sequential or other similar type damages.

The warranty and remedies set forth above are exclusive and in lieu of all other, oral or written, expressed or implied. No mFactory dealer, agent or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitations or exclusions may not apply to you.

Contents

Chapter 3. Edit Menu

Chapter 8. Structure Window

Chapter 9. Layout Window

Chapter 10. Layers Window

Chapter 11. Palette Reference

Chapter 12. Modifier Reference

Chapter 13. Modifier Pop-Up Menus and Message Reference

Chapter 14. Miniscript Modifier

Chapter 15. Glossary

Appendix A. MovieTrax

Appendix B. mTropolis Object Model (MOM) User's Guide

Index

Chapter 1. Getting Started

mFactory's mTropolis™ is a development system for authoring multimedia titles and applications. Sophisticated features such as message passing and reusability of components are built into the environment, offering the author new creative possibilities and significant productivity advantages.

mTropolis' object technology is presented in a graphical authoring environment. This approach changes the focus to the media and logic of the title, rather than its low-level code. All creative work is done visually, without intricate and time-consuming programming. mTropolis provides facilities for creating components, linking them to media and altering these components with tools and modifiers. An organizational structure is built into the environment, providing a simple framework for managing components within the title. Although complete titles can be created with the basic package, additional software components for specialized titles can be purchased from mFactory or third parties. mFactory also offers an Application Program Interface (API) called mFactory Object Model (MOM™) for multimedia programmers who want to extend the power of mTropolis by writing their own modifiers using programming languages such as C or C++.

For an overview of the authoring process and a hands-on introduction to mTropolis, including a step-by-step tutorial and sample projects, consult the *mTropolis Developer Guide*.

The rest of this chapter outlines platform requirements, mTropolis installation instructions and the structure of this reference manual.

EQUIPMENT AND MEMORY REQUIRED

The minimum configuration recommended for authoring in mTropolis is a 68040 or Power Macintosh with:

- 6 MB application RAM (for 8-bit color) or 8MB application RAM (for 24-bit color)
- System 7.x
- QuickTime 2.x (included)
- Sound Manager 3.x (included)
- double-speed CD-ROM drive

INSTALLATION

To install the mTropolis editor and player on Macintosh, follow the instructions in the "Read Me First!" file on the mTropolis CD-ROM. Don't forget to complete your registration card and mail it to mFactory.

Installing the Windows Runtime Player

To install the mTropolis player on Windows, follow the instructions in the README.WRI file on the mTropolis CD-ROM

Release Notes

Check the mTropolis release notes in the "Read Me First!" file for late-breaking information about features in this release.

Learning mTropolis

This book describes mTropolis features in detail.

The *mTropolis Developer Guide* provides an overview of mTropolis functionality and an introduction to object-oriented design. Chapter 7, "Tutorial", of the *mTropolis Developer Guide* is a step-by-step guide to creating your first mTropolis project.

CONVENTIONS USED IN THIS MANUAL

Illustrations have been included to help you find specific information quickly.

Step-by-step instructions are shown as bullet text. For example:

- Choose **Open** from the File menu.
- Select the graphic tool in the tool palette.

The names of menu items, buttons, check boxes and radio buttons are capitalized and set in bold type. For example:

- …the **Duplicate** menu item…
- …the **Cancel** button…

The names of messages, dialogs and interface fields are capitalized. For example:

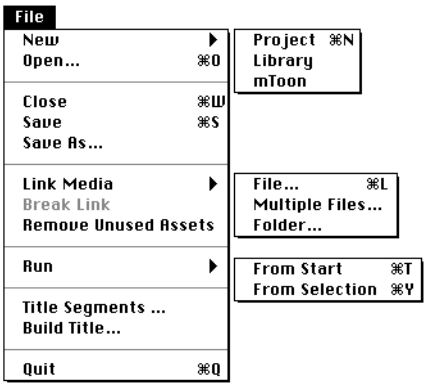
- …the Modifier's Name field…
- …the Element Info dialog…

The names of commands are capitalized and italicized. For example:

- …sends a *Play Forward* command.

For keyboard shortcuts, the command key is written as "⌘".

Chapter 2. File Menu



The File menu provides options for:

- creating, opening, closing and saving projects
- creating and opening libraries
- creating, opening, closing and saving mToons
- linking external media to projects
- breaking links to external media
- switching projects from edit mode to run-time mode
- building titles
- quitting mTropolis.

CREATING, OPENING, AND SAVING mTROPOLIS PROJECTS

Titles created with the mTropolis authoring environment are referred to as projects during the development phase.

When mTropolis is first launched, or when a new project is created by selecting **New-Project** from the File menu, mTropolis creates an untitled project with a single untitled section (Untitled Section 1), subsection (Untitled Subsection 1), shared scene (Untitled Shared Scene) and scene (Untitled Scene 1) as shown in Figure 2.1. See "Structure Window Overview" on page 8.1 for more information on the components of a mTropolis project.

Although the components of a project can also be viewed from two other windows, the structure window and the layers window, the layout window is the default view when mTropolis creates a new project or opens an existing project.

More than one project can be opened simultaneously, although your system's RAM imposes a practical limit on the number of projects that can be open at once. See "Equipment and Memory Required" on page 1.1 for suggested hardware requirements for mTropolis.

Creating a New Project

- Choose **New-Project** from the File menu (or use ⌘-N). A new, untitled project is created and an empty project layout window (Figure 2.1) appears.

Opening an Existing Project

- Select **Open** from the File menu. A standard file selection dialog appears.

Closing a Project

- Make sure the layout window is in the foreground. Choose **Close** from the File menu (or use ⌘-W). If changes have been made to the project since the last save, an alert appears, asking if you want to save your changes before closing the project.
- Choose **Don't Save**, **Cancel**, or **Save**. If Save is chosen and the project has been saved previously, mTropolis re-saves the project under the existing name. If the project hasn't been saved previously, a standard file dialog appears, requesting a name and a destination before saving.
- *Note: Alternatively, projects can be closed by clicking in the close box in the layout window's title bar.*

Saving a Project

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

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

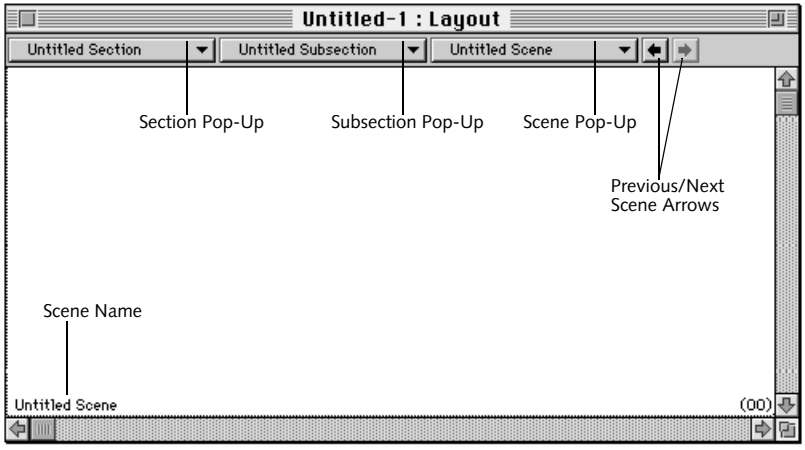


Figure 2.1 A new, empty layout window

Renaming a Project

- Choose **Save As** from the File menu to save the current project under a new name. A standard file dialog appears, requesting a new name and location before saving.
- *Note: Saving a project using Save As reclaims the space used by assets that were previously imported and then dragged to the asset palette's trashcan. Periodically using Save As during the development of a project can help keep the project's file size smaller.*

CREATING AND MODIFYING mTROPOLIS LIBRARIES

Libraries can be used to store project components (e.g., sections, subsections, scenes, elements and modifiers) in a file that is separate from the project. When a new library is created, or an existing library is opened, a library palette appears on the screen (Figure 2.2). Project components can be dragged to and from this palette. Multiple libraries can be open at once.

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

- *Note: Projects cannot be stored in libraries.*

Creating a New Library

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

Adding Items to a Library

- Drag sections, subsections, scenes, elements or modifiers from a project and drop them into the library palette. The component added to the library is represented by an icon along with its name.
- *Note: Modifiers that are stored in libraries retain their settings. However, some modifiers may have dependencies; that is, they may be dependent upon additional information to function correctly. For example, vector motion modifiers are dependent upon vector variables. If a functioning vector motion modifier is placed in a library and moved to another project, it will have to be reconfigured to reference a new vector variable.*

Deleting Items from a Library

- Drag the item to the library palette's trash can, in the upper left corner of the palette.

Figure 2.2 A library palette

The trash can bulges to indicate that it contains deleted items.

Saving a Library

- Click on the library palette's pop-up arrow. A pop-up appears.
- Choose **Save** from the pop-up to save the library under its current name. If the library hasn't been saved previously, a standard file dialog appears, requesting a name and a destination before saving. Choose **Save As** to save the library under a new name.
- *Note: Libraries should be saved with descriptive file names (e.g., "Frog Motions Library").*

Closing a Library

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

CREATING AND MODIFYING mTOONS—THE mTROPOLIS ANIMATION FORMAT

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

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

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

Creating a New mToon

To create a new mToon:

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

The mToon Editor Window

The mToon editor window (Figure 2.3) is used to create and modify mToons. This window displays a single cel of the mToon animation, along with a number of controls. Controls for this window are described below.

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

Registration Point

By default, the registration point of each cel appears in its upper left corner. The registration point is the point used to align cels when the **Align and Trim Cels** mToon menu option (see page 2.6) is selected. Use the selection tool to drag the registration point to a new location, or hold down the Option key and click on the screen to change the cel's registration point.

Cel Number Field

This field displays the number of the current cel.

Keyframe Button

Select this button to make the currently-selected frame a keyframe. This button is only available if the mToon compression options are configured to use temporal compression (see "Temporal Compression Check Box" on page 2.10). If temporal compression is not selected, every frame is considered a keyframe.

Selection Field

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

mToon Controller

The controller is used to preview the animation, as well as to select a range of cels for editing. Use the step buttons to step forward or backward through the animation one cel at a time.

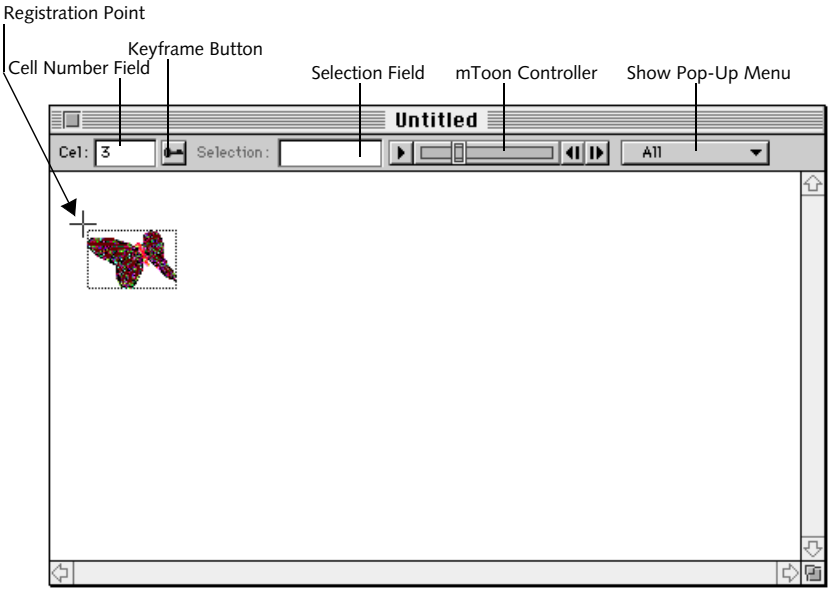


Figure 2.3 The mToon Editor Window

Hold down the Shift key while stepping through cels to select a range of cels. Or, hold down the Shift key while dragging the slider to select a range of cels.

Show Pop-Up Menu

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

THE mTOON MENU

When the mToon editor window is open, the mToon menu (Figure 2.4) is available from the mTropolis menu bar.

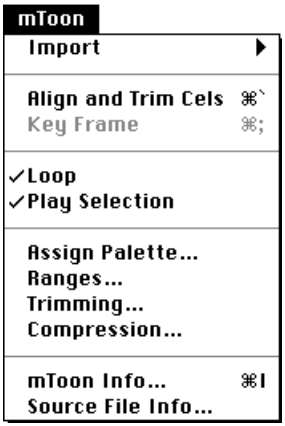


Figure 2.4 The mToon menu

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

mToon menu options are described below.

Import

Choose a menu item from the **Import** submenu in the mToon menu to import cels from a single file, multiple files, or a folder of files for processing.

- *Note: Files within folders will be imported according to the order assigned to them by the Finder.*
- *Note: If any cels are currently in the mToon editor window, cels from the newly-imported file(s) will be inserted sequentially after the last cel.*

Hot Tip

If your characters are to be rendered in a 3D package, render them as PICTs and compile them in the mToon editor. This will allow you to insert any changed frames much more easily than if they are rendered to PICS.

Align and Trim Cels

Choose **Align and Trim Cels** from the mToon menu (or use ⌘-‘ (Command-left single quote)) to align the registration points of cels in the animation. At the same time, the background area of cels is trimmed based on the cels’ background color. By default, this color is white, but can be changed using the Trimming dialog (see “Trimming” on page 2.8).

mToon Menu Play Controls

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

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

Assign Palette

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

To assign a palette:

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

By default, the mToon editor is set to display 256 colors. Use the Compression dialog (see "Compression" on page 2.9) to change this setting.

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

To be used in a project, mToons (and their custom palettes, if any) must first be linked to elements in the project. To display an mToon created with a custom color palette, place a color table modifier in the mToon's scene and select the table from the modifier's color table pop-up.

The effect of the new CLUT will now be visible in runtime. To view the effect of a color table while in edit mode, select a color table from the **Preview Color Table** submenu in the View menu.

Ranges

Select this option to define sequences of cels in an mToon as named ranges. The Ranges dialog (Figure 2.5) appears. mToon ranges can be accessed by messengers or Miniscript during runtime.

Hot Tip

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

To define a new range of cels:

- While in the mToon editor window, select a range of cels to be defined by holding down the Shift key while dragging the slider on the controller. The section of the bar corresponding to the selected cels is highlighted.
- Choose **New Range** from the mToon editor's Show pop-up to display the New Range dialog and type the name of the new range. Alternatively, choose **Ranges** from the mToon menu to display the Ranges dialog (Figure 2.5).

Components of the Ranges dialog are described below.

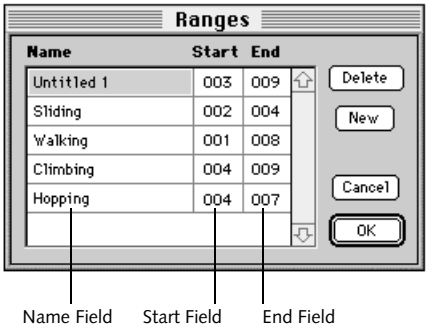


Figure 2.5 The Ranges dialog

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

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

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

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

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

- *Note: Use the Tab key to move through the fields in this dialog.*

Hot Tip

To the extent that memory will allow, preload animations on a "parent enabled" message. Preloading allows them to play as quickly as possible and their cels and ranges can be switched instantaneously. See "Preload Media (Message/Command Menu Only)" on page 13.10.

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

To specify a new color to be trimmed, or to turn Trim background information off, choose **Trimming** from the mToon menu. The Trimming dialog (Figure 2.6) appears.

Figure 2.6 The Trimming dialog

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

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

Compression

mToons can be compressed for optimal playback during runtime. mTropolis accesses the software compressors supplied with QuickTime. Choose **Compression** from the mToon menu to access the Compression dialog (Figure 2.7). Components of this dialog are described below.

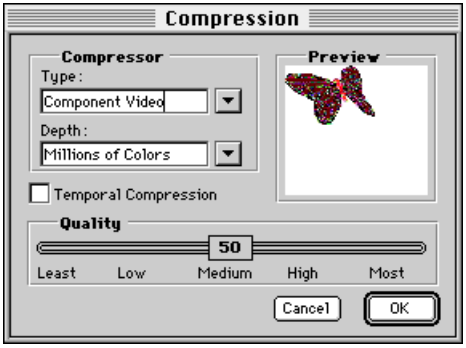


Figure 2.7 The Compression dialog

Type Pop-Up Menu

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

Compression methods will vary according to the version of QuickTime installed on the author's system, but standard QuickTime options are described below. Some of the following descriptions are reprinted, with the permission of Adobe Systems Incorporated, from the *Adobe Premier 3.0 User Guide*:

- **Animation**: Use the Animation compressor for compression of images that were originally in digital form (animation and computer generated content) and were not obtained by videotape. The Animation compressor employs a compression algorithm developed by Apple based on run-length encoding techniques. The animation compressor works in a lossy or a lossless mode, and supports both spatial and temporal compression. This compressor can play back images at up to 30 fps at full screen resolution; the performance and compression ratios you achieve depend on the type of image you are using.
- **Cinepak**: Use the Cinepak compressor when compressing 16-bit and 24-bit video for playback from CD-ROM disks. This compressor attains higher compression ratios, better quality, and faster playback speeds than Video compression. For best results, use the Cinepak compressor on raw source data that has not been previously compressed with a highly lossy compressor. Decompression is much faster than compression with Cinepak, and the data rate for playback can be defined by the user.
- **Graphics**: Specially designed for 8 bit (256 color) stills where compression ratio (file size) matters more than compression speed. Generally this choice will reduce an image to half the size of a file compressed using the animation compressor, but will take twice as long to do it.
- **None**: By selecting None, mToons remain uncompressed.

- **Photo-JPEG**: JPEG (Joint Photographic Experts Group) is an international standard for compressing still images. Use the Photo compressor for images that contain smooth transitions, or that do not contain a high percentage of edges or other sharp detail. Most natural images fall into this category. For this type of 24 bit image, the Photo compressor produces a reconstructed image that is virtually indistinguishable from the original image at a compression ratio of 10:1. Compression time is equal (or very nearly equal) to decompression time.
- **Video**: Use the Apple Video compressor for capture and compression of analog video, high-quality playback for hard disk, and moderate quality playback from CD-ROM. This compressor supports both spatial and temporal compression, and can playback at rates of 10 fps or more. Data can be later recomposed or recompiled for higher compression ratios. The Apple Video compressor allows recompression with minimal or no quality degradation.

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

Depth Menu

Sets the number of colors used by a new mToon. The default color depth is 256 colors.

Temporal Compression Check Box

Check this box to perform temporal compression on the mToon. By default this option is off. Temporal compression shrinks the mToon's file size, but individual frames are no longer randomly accessible. mToons that are *not* temporally compressed may play faster than ones that are. Note that this option is not available for all compression Type settings. When this option is selected, the keyframe button on the mToon editor window becomes active (see "Keyframe Button" on page 2.5).

Compression Quality Slider

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

Cancel

Click Cancel to ignore changes to compression settings.

OK

Click OK to accept changes made to compression settings.

Hot Tip

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

Figure 2.8 The mToon Info dialog

mToon Info

Choose **mToon Info** from the mToon menu to display the mToon Info dialog (Figure 2.8).

The mToon Info dialog displays information about the current animation.

Source File Info

Choose **Source File Info** from the mToon menu to display information about the source files that are linked to the current animation. The Source File Info dialog (Figure 2.9) appears. Components of this dialog are described below.

Path Display

The path to the source file of the selected cel is displayed here.

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

Cel Column

The cel number assigned to the source file.

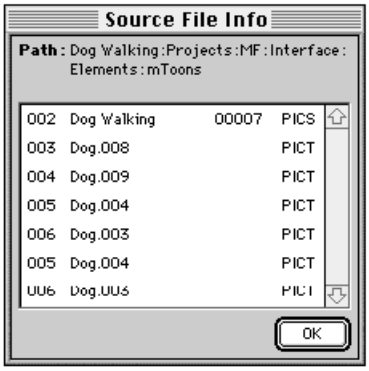


Figure 2.9 The Source File Info dialog

Original File Name

The name of the original source file.

Index number of PICS File

If the source file is a PICS file, its index number is displayed in this column.

File Format

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

SAVING mTOONS

mToons can be saved using standard procedures.

Saving a New mToon

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

Renaming an mToon

- Choose **Save As** from the File menu to save the mToon under a new name. A standard file dialog appears, requesting a new name and a destination before saving.

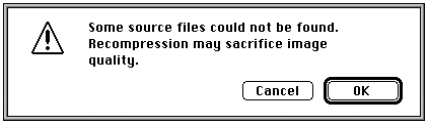
OPENING PROJECTS, LIBRARIES AND mTOONS

Previously-saved projects, libraries, or mToons can be opened using standard procedures.

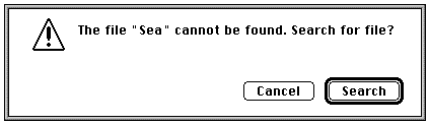
- Choose **Open** from the File menu (or use ⌘-O) to open an existing project, library, or mToon.
- *Note: Double-clicking on a project, library, or mToon from the Finder will open the selected file and will also launch mTropolis if it isn't already launched.*

Re-Linking External Media Files

When a project, library, or mToon is saved, mTropolis saves the links (path names) to its external media files. If any media files were deleted, moved, or renamed since the project, library, or mToon was last saved, mTropolis displays an alert dialog when it is reopened. The alert for mToons:



The alert for projects and libraries:



When this dialog appears, the "missing" media can be re-linked to the project or library as follows:

- Click **Search** to start a file search. If all the files are located, mTropolis reestablishes the links to the project or library. If mTropolis cannot find a file, a standard file dialog appears (Figure 2.10), asking you to locate the missing file.

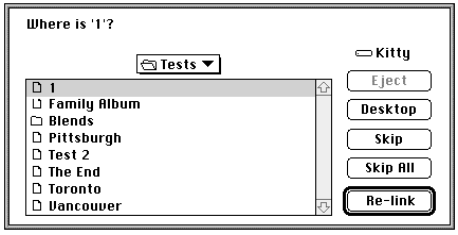


Figure 2.10 Re-linking a missing file

- If you locate the missing file, select the filename and click **Re-link**.
- If you cannot locate the file, click **Skip** to ignore just this file, or **Skip All** to ignore this file and all other missing files. Although mTropolis will not be able to access the missing file, the missing graphic elements will be replaced with their low resolution thumbnails by default. All aspects of the project, including the element frames,

names of external media files and modifiers, will remain in the project.

If the **Save Thumbnails** check box has been toggled off in the Project Preferences dialog, the frames and names of the missing media and all of their associated modifiers will remain in the project; however, the graphic elements will be empty. To alter the project's preferences settings, the Project Preferences dialog can be displayed by selecting **Preferences-Project** from the Edit menu.

- If more files need to be re-linked, mTropolis continues to search in the same location until all of the media is re-linked, or until it cannot locate another file.

Saving the project after re-linking to files saves the project with links to the new file locations.

LINKING EXTERNAL MEDIA FILES TO ELEMENTS

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

The Link Media command can be used to link media to a selected element in any of mTropolis' editing views—the structure, layout, and layers windows—or multiple files can be linked to the asset palette. Use **Link Media - File** (or ⌘-L) to link a single file to a selected element. Use **Link Media - Multiple Files** or **Link Media - Folder** to link groups of files to the asset palette.

- For instructions on how to create and link media to elements in the structure view, see "Creating New Sections, Subsections, Scenes and Elements" on page 8.4.
- For instructions on linking media in the layout view, see "Linking Media to an Element in the Layout Window" on page 9.3.
- For instructions on linking media in the layers view, see "Linking Media to Elements in the Layers Window" on page 10.6.
- For information on linking media to the asset palette, see "Linking Media to the Asset Palette" on page 11.10.

BREAK LINK

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

- Select the element. Use Shift-Click to select multiple elements.
- Choose **Break Link** from the File menu. All aspects of the selected elements, including their frames and modifiers, will remain intact, but they are no longer linked to their media files.

SWITCHING TO RUNTIME MODE

When mTropolis is first started, the program is in *edit mode*. Authoring occurs in edit mode, but the project can be previewed by

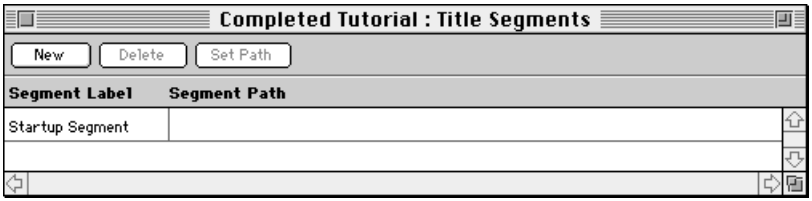


Figure 2.11 The Title Segments window

entering *runtime mode*. In runtime mode, the project is shown as it will appear to users.

Running a Project from its First Scene

- Choose **Run-From Start** from the File menu (or use ⌘-T) to switch to runtime mode and preview the project from its first scene. This scene is the first scene in the first subsection in the first section you see in the structure window.
- Normally, the project is displayed over a black background. To keep the mTropolis interface visible during runtime, press ⌘-Option-T instead.
- Press ⌘-T again to return to the scene where ⌘-T was originally pressed. Optionally, use ⌘-. (Command-period) to re-enter edit mode with the current runtime scene displayed in the layout window.

Running a Project from a Specific Scene

- Choose **Run-From Selection** from the File menu (or use ⌘-Y) to switch to runtime and preview the project from the current scene.
- Normally, the project is displayed over a black background. To keep the mTropolis interface visible during runtime, press ⌘-Option-Y instead.
- Press ⌘-Y again to return to the scene where ⌘-Y was originally pressed. Optionally, use ⌘-. (Command-period) to re-enter runtime mode with the current runtime scene displayed in the layout window.

TITLE SEGMENTS

Select **Title Segments** to display the Title Segments window (Figure 2.11). Use this window to create and manage *title segments*—groups of project sections that are written to the same file when the project is built into a standalone title (see "Build Title" on page 2.16).

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

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

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

Set Path Button
Select this button to select a folder in which this title segment will be written when the title is built. A standard folder selection dialog appears. When a folder is selected, its path appears in the "Segment Path" list.

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

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

- Select the path and edit it from the keyboard. The path must be entered in proper Macintosh path syntax (i.e., a colon separated list of drive/folder names).
- Select the path or corresponding segment label and click the Set Path button. A standard folder selection dialog appears. When a folder is selected, its path appears in the list.

Assigning Sections to Title Segments

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

- Open the structure window (if it is not already open) by selecting **Structure Window** from the View menu (or press ⌘-2).
- In the structure window, select the section to be reassigned, then select **Element Info** from the Object menu. The Section Info dialog (Figure 2.12) appears. You can also double-click on the section to display this dialog.

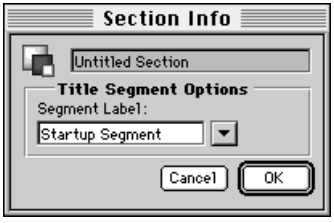


Figure 2.12 The Section Info dialog

- Use the Segment Label pop-up menu to select the segment for this section. A new segment can be created from this pop-up by selecting **New Segment Label**. A prompt appears for entering the new label.

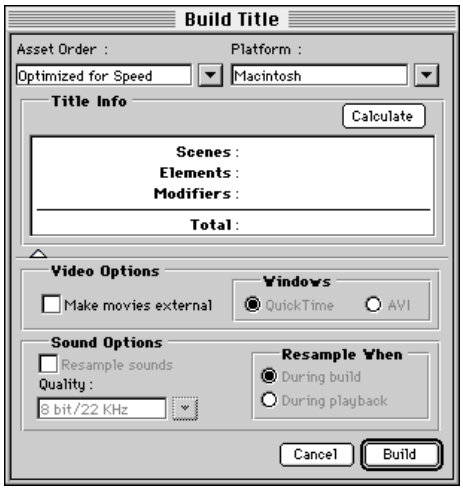


Figure 2.13 The Build Title dialog

- Click OK to confirm the new title segment association and dismiss the Section Info dialog.

BUILD TITLE

When the project is complete, it can be built into a stand-alone title for Macintosh or Windows platforms. Select **Build Title** from the File menu to display the Build Title dialog (Figure 2.13). Components of this dialog are described below.

Asset Order Pop-Up

Select an optimization option from this menu. Options include:

- **Optimized for Speed**: Select this option to place each instance of the asset in the order in which they are used by the project.
- **Optimized for Space**: Select this option to place only the first instance of an asset in the project.

Platform Pop-Up Menu

Use this pop-up to select the type of platform for which the title will be built. Options are "Macintosh" (the default) and "Windows". mTropolis title files will be built for the selected platform. Note that not all mTropolis features are available in titles built for Windows. See "mTropolis Features not Supported in Titles Built for Windows Platforms" on page 2.18.

Calculate Button/Title Info Area

Click the **Calculate** button in the dialog's File Size section to calculate and display the number of scene, element, and modifier components present in the project.

Video and Sound Options

Select the triangle in the lower-left part of the Build Title dialog to reveal build options for video and sound.

Make Movies External

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

Windows Radio Buttons

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

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

Resample Sounds/Resample When/Quality Section

Check the "Resample Sounds" checkbox to convert all sounds in the project to a single bit resolution and sample rate. This option is only available when building titles for Windows platforms. This option is useful because of the default technique that Windows uses to play sounds simultaneously:

- When sounds are played simultaneously under Windows, the quality of the first sound played is used to resample any others. For example, if an 8-bit, 22KHz sound is playing, then a 16-bit, 44KHz sound is made to play at the same time, Windows will "downsample" the 16-bit sound (affecting its quality adversely). Similarly, if the 16-bit, 44KHz sound started playing first, the 8-bit 22KHz sound would be "upsampled". This behavior can be modified by using the other controls in this section, as described below.

Use the "Resample When" radio buttons to specify whether to resample sounds "During build" or "During playback":

- Resampling during the build process results in all sounds in the project being resampled to conform to the bit resolution and sample rate selected in the "Quality" menu. The sounds are upsampled or downsampled as they are written to the title files. As a result, they may take up more or less disk space than their original versions, but Windows will never have to perform sound conversions during runtime.
- Resampling during playback causes all sounds, no matter what their actual bit resolution and sample rate, to be resampled as specified by the "Quality" menu when they are played. The default behavior of Windows (to resample sounds to the format of the sound that started playing first) is overridden. Note that if the playback machine does not support the specified sample rate, the next best quality will be used.

Cancel Button

Select "Cancel" to dismiss the Build Title dialog and abort the build title process.

Build Button

Click the "Build" button to begin compiling the project into a title. If there are title segments that do not have "segment paths" defined (see "Title Segments" on page 2.14), a standard folder selection dialog appears for each title segment. Select a folder to be used to contain that segment's files. Note that once a location has been selected, it is remembered for future builds (the Title Segments window can be used to change the path for the segment).

Files Created by the Build Title Process

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

- **Startup Segment File**: All mTropolis projects have a Startup Segment file. This file is written to the folder specified in the Title Segments window. For Macintosh builds, this file is given the name of the project, appended with "1". For example, for a project named "myProject", the startup segment is written as a file named "myProject1". For Windows builds, this file is given the name of the project truncated to seven characters and appended with "1.MPL". For example, for a project named "myProject", the startup segment is written as a file named "MYPROJE1.MPL". The Startup Segment has the icon shown below.

- **Other Segment Files**: If any title segments (other than the Startup Segment) were defined and associated with project sections, corresponding data segment files are written. They are written to folders as specified in the Title Segments window. For Macintosh builds, these files are given the name of the project, appended with a number (e.g., "myProject2", "myProject3", etc.). For Windows builds these files are given the name of the project, truncated as necessary to fit the name and segment number into eight characters, appended with the segment number and ".MPX" (e.g., "MYPROJE2.MPX", "MYPROJE3.MPX").

- **Video Folder**: If the "Make movies external" checkbox was selected in the Build Title dialog, a folder named "Video" is created for each segment that needs a video file. The folder resides in the same folder as its corresponding segment file. The "Video" folder contains the externalized video files, each named with a number and an extension (".mov" for QuickTime files, ".AVI" for Video for Windows files).

mTropolis Features not Supported in Titles Built for Windows Platforms

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

PLAYING MTROPOLIS TITLE FILES

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

- Segment files and folders created by the Build Title process.
- The appropriate version of mTropolis Player for the target machine. The mTropolis CD-ROM contains three Macintosh players—68K-only version, Power Macintosh-only version, and a version for any Macintosh. There are two versions of the Windows player—"MTPLAY31.EXE" for Windows 3.x, and "MTPLAY95.EXE" for Windows 95.
- The accompanying resource folder (for Macintosh) or subdirectory (for Windows) for the player. If you are deploying a title for both Windows 3.1 and Windows 95 players, the contents of their "RESOURCE" directories should be combined into a single "RESOURCE" directory.

These files and "top-level" folders should be put in the same folder (for Macintosh or Windows 95) or subdirectory (for Windows 3.x).

Running the Title

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

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

  Note that only one startup segment should be placed in the player's folder/directory. If multiple startup segments are present in this directory, there is no way to determine which one will be run.
- Launching a mTropolis player without a startup segment present starts the player without playing a title. A startup segment can be selected manually. On Macintosh, choose **Open** from the File menu. A file selection dialog appears. Use the dialog to select a startup segment to be run. On Windows, a file selection dialog appears automatically. Use the dialog to select a startup segment file (i.e., a ".MPL" file) to be run.
- Double-clicking or running a startup segment file causes that title to be run with a mTropolis player, if one is available. If multiple copies of the mTropolis player are installed, there is no way to tell which one will be run.
- Dragging a startup segment file onto a mTropolis player causes that title to be run with that player executable.

During runtime, the mTropolis player can be quit by pressing ⌘-Q on Macintosh or Ctrl-Q on Windows.

Customizing Title Filenames, Location, and Icons

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

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

Hot Tip

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

Macintosh or Windows resource editors can be used to customize the mTropolis player icon. A custom title with single-icon launching can be created by changing the name and icon of the mTropolis player.

Deploying Multiple-Disc Titles

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

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

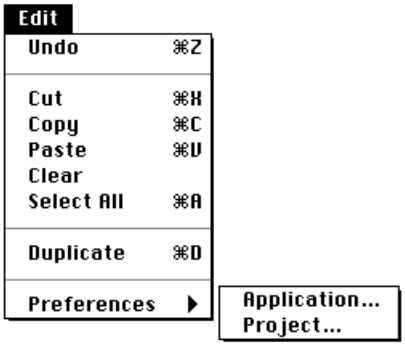
QUITTING mTROPOLIS

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

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

- Choose **Don't Save**, **Cancel** or **Save**. If Save is chosen and the project has been saved previously, mTropolis re-saves the project under the existing name. If the project hasn't been saved previously, a standard file dialog appears, requesting a name and a destination before saving.
- *Note: If multiple projects are open, mTropolis quits each one in sequence.*

Chapter 3. Edit Menu



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

Most of the project editing functions are available in mTropolis' three project windows (layout, structure and layers). Editing functions that are not valid for the current selection are dimmed.

UNDO

The Undo menu item restores the project to its state prior to the last change.

To undo the last operation:

- Choose **Undo** from the Edit menu (or use ⌘-Z).

CUT, COPY, PASTE

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

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

If the clipboard is empty, the **Paste** options appears greyed out in the Edit menu.

To cut, copy, or paste an item:

- Select the desired item with the tool palette's selection tool.
- Choose **Cut, Copy** or **Paste** from the Edit menu. Alternatively, the keyboard shortcuts shown in Table 3.1 can be used:

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

Table 3.1: Edit menu shortcuts

CLEAR

The Edit menu's **Clear** menu item is used to delete any item from the project, with the exception of items in palettes, which are deleted

by dragging them to a palette trash can. Cleared objects are deleted without being copied to the clipboard.

To clear an item from the project:

- Select the item to be cleared in any window.
- Choose **Clear** from the Edit menu. The item is deleted from the project.

SELECT ALL

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

DUPLICATE

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

To duplicate an item:

- Select the item with the tool palette's selection tool.
- Choose **Duplicate** from the Edit menu (or press ⌘-D). The duplicated item automatically appears near the original.

PREFERENCES-APPLICATION

Select **Preferences-Application** from the Edit menu to set preferences for mTropolis. The Application Preferences dialog (Figure 3.1) appears. This dialog can be used to change a number of different preference types. Use the "Show" pop-up menu to select different preference types. The dialog changes to show applicable settings. These settings are described below.

General

By default the Application Preferences dialog opens to general application preferences (Figure 3.1). Components of this dialog are described below.

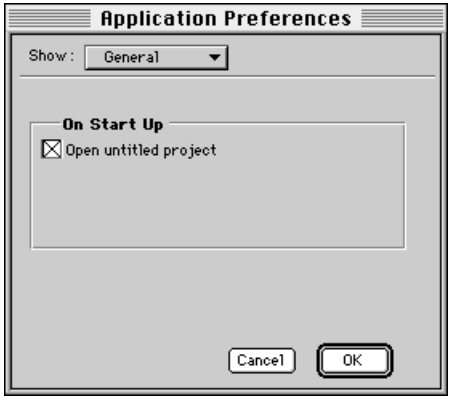


Figure 3.1 General Application Preferences

Open Untitled Project on Start Up Check Box
When selected, mTropolis opens a new, untitled project at start-up. When deselected, mTropolis launches without creating an untitled project.

Layout

To set layout preferences, choose Layout from the Show pop-up in the Application Preferences dialog (Figure 3.2).

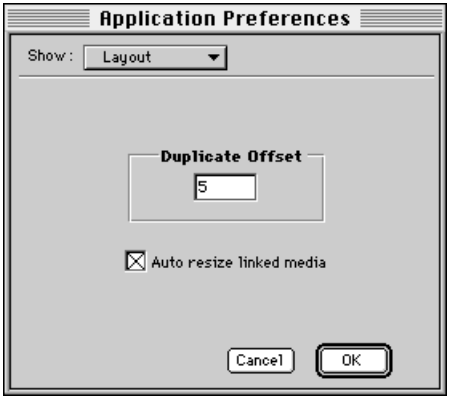


Figure 3.2 Layout Application Preferences

Duplicate Offset Field

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

Auto Resize Linked Media Check Box

This option automatically resizes elements to the screen dimensions of linked external media. The default is on.

Type

To set preferences for the default type used in text elements, choose **Type** from the Show pop-up in the Application Preferences dialog (Figure 3.3).

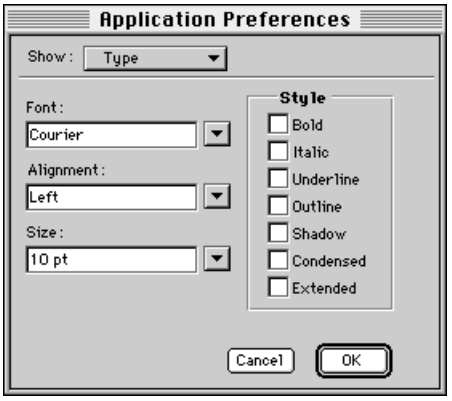


Figure 3.3 Type Application Preferences

Font Pop-Up Menu

Select mTropolis' default text font from the pop-up.

Alignment Pop-Up Menu

Select mTropolis' default text alignment from the pop-up.

Size Pop-Up Menu

Select mTropolis' default text font size from the pop-up.

Style Check Boxes

Select mTropolis' default text font style by clicking on zero or more check boxes.

Libraries

To set library preferences, choose **Libraries** from the Show pop-up in the Application Preferences dialog (Figure 3.4). Use these preferences to specify any libraries that should be opened whenever mTropolis is started.

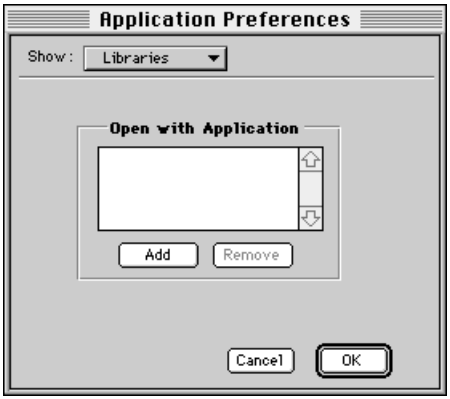


Figure 3.4 Libraries Application Preferences

Open with Application List

The library or libraries to be opened along with the application are listed here. More information on libraries can be found in "Creating and Modifying mTropolis Libraries" on page 2.3.

Add Button

Click this button to display a standard file dialog. Libraries selected from the dialog will be added to the list. These libraries will be automatically opened with the application.

Remove Button

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

PREFERENCES-PROJECT

Select **Preferences-Project** from the Edit menu to set preferences for the current project. The Project Preferences dialog (Figure 3.5) appears. This dialog can be used to change a number of different preference types. Use the "Show" pop-up menu to select different preference types. The dialog changes to show applicable settings. These settings are described below.

Figure 3.5 Draw Area Project Preferences

Draw Area

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

Draw Area Size Pop-Up Menu

Select the size of the monitor on which the title is targeted to be displayed. Options include 9-Inch Monitor, 12-Inch Monitor, 13-Inch Monitor, 15-Inch Monitor, Current Monitor, and Custom.

Draw Area Size Fields

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

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

Color Depth Pop-Up Menu
Select the color depth of the monitor display from this pop-up. Options include B/W (black and white), 16 colors, 256 colors (8-bit), Thousands (16-bit), and Millions (24-bit).

Default Monitor
Select the project's default display monitor.

Libraries

To set the libraries preferences, select Libraries from the Show pop-up in the Project Preferences dialog (Figure 3.6). Use these preferences to specify any libraries that should be opened whenever the project is opened.

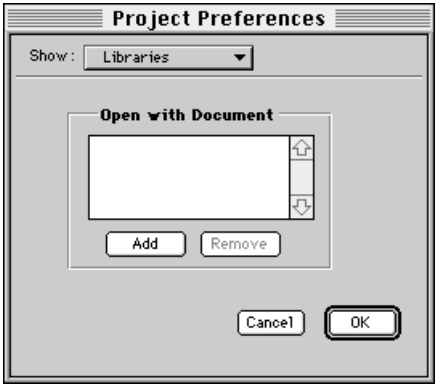


Figure 3.6 Libraries Project Preferences

Open with Document List
The library or libraries to be opened with the project are listed here.

Add Button
Click this button to display a standard file dialog. Selected libraries will be added to this list. These libraries will be opened with the project.

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

Thumbnails

To set the thumbnail preferences, choose Thumbnails from the Show pop-up in the Project Preferences dialog (Figure 3.6).

Figure 3.7 Thumbnails Project Preferences

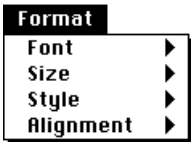
Quality Radio Buttons
Select an option to display thumbnails in the project at high, medium, or low resolution.

Save Thumbnails Check Box
By default, this option is selected, allowing any media files that cannot be found when the project is opened to be replaced with their thumbnails.

Hot Tip

Toggle Save Thumbnails off to decrease the size of project files.

Chapter 4. Format Menu



Options in the Format menu can be used to edit the characteristics of text in text elements. These options are only available when text in a text element has been selected in the layout window.

When a text element is first created, it creates text as specified in the Application Preferences–Type dialog (see "Type" on page 3.3). Text formatting can be changed from this default using the Format menu options.

The text tool (the "A" icon found in the tool palette) can be used to create text elements. See "Text Tool" on page 11.2 for more information on creating text elements.

Format menu options are described below.

FONT

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

To change the font used by a section of a text element:

- Select the text tool in the tool palette. The mouse cursor changes to an insertion bar.
- Click on the text element.The insertion bar appears at the end of the text in the text element.
- Click and drag over the block of text to highlight the text to be modified.
- Choose **Font** in the Format menu. Note that the name of the font currently used by the selected text has a checkmark beside it in the **Font** submenu.
- Select a font from the submenu. The block of text changes to the specified style.
- *Note: A similar procedure is used for most of the text formatting options described in this chapter.*

SIZE

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

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

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

To change the style of text, first highlight a specific block of text within an element and choose a **Size** option from the Format menu.

STYLE

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

The style of currently highlighted text is indicated by a check mark beside the style name in the **Style** submenu.

To change the style of text, first highlight a specific block of text within an element and choose a **Style** option from the Format menu.

ALIGNMENT

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

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

To align text in a text element:

- Select the text tool in the tool palette.
- Click on the text element.
- Choose **Alignment** in the Format menu.
- Choose **Left**, **Center** or **Right** from the submenu.

DELETING AND EDITING TEXT

To delete text within an element:

- Select the text tool in the tool palette.
- Click and drag over a block of text to select it.
- Press the Delete key (or select **Cut** from the Edit menu). The text is cleared from the text element. Its frame and modifiers remain intact.

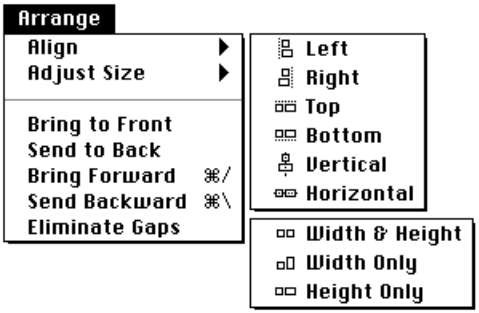
DISPLAYING VARIABLES IN TEXT FIELDS

A text element can be configured to display the contents of a variable during runtime. The steps below describe the method used to display the value of a string variable, however, the value of any variable type can be displayed using the same steps.

- Create a text element that contains the name of the string variable enclosed in ‹ and › symbols.
- Create a string variable somewhere in your project. Double-click the string variable and enter the text to be displayed during runtime in the string variable's text field. Note that the content of a string variable is limited to 255 characters.
- Configure a messenger in the scope of the string variable to send an *Update Calculated Fields* command to the text element. Use the With menu to send the string variable with the *Update Calculated Fields* command. See "Update Calculated Fields (Mes-

sage/Command Menu Only)" on page 13.10.

Chapter 5. Arrange Menu



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

Features of the Arrange menu are described below.

ALIGNING ELEMENTS

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

To align elements:

- Choose the selection tool in the tool palette.
- Shift-Click to select the elements to be aligned, or drag a marquee around the elements.
- Choose an alignment option from the **Align** submenu. The elements align along the edge of the first selected element.

ADJUSTING THE SIZE OF ELEMENTS

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

- **Width & Height**: both the widths and heights of elements change to the width and height of the largest selected element.
- **Width Only**: the widths of the elements change to the width of the largest selected element.
- **Height Only**: the heights of the elements change to the height of the largest selected element.

To give elements a common size:

- Select the selection tool in the tool palette.
- Shift-Click to select the elements to be resized, or drag a marquee around the elements.
- Choose an adjustment option from the **Adjust Size** submenu.

CHANGING THE LAYER ORDER OF ELEMENTS

The layer order of text and graphic elements is the order in which the elements are drawn on the screen. When new elements are added to a scene, they draw on top of previous elements. Drawing elements on top of each other creates the illusion that the two-dimensional (X,Y) screen has a depth (Z) dimension (Figure 5.1). The effect is popularly called "2.5D."

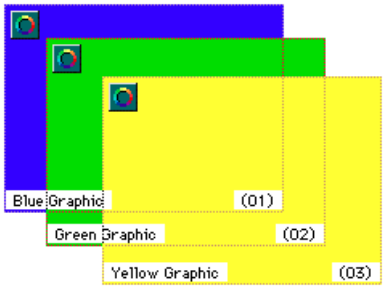


Figure 5.1 The effect of graphic layers

In the layout window, the layer order of an element is represented by a number in parentheses shown on the bottom right corner of the element. The higher the number, the closer it appears to the viewer (Figure 5.1).

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

Changing the Layer Order of Single Elements

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

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

To move an element in the layer order:

- Click on the selection tool in the tool palette.
- Select the element to be moved.
- Choose an item from the Arrange menu. The selected element is moved in the layer order and its layer order number is updated.

Changing the Layer Order of Multiple Elements

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

- **Bring To Front** or **Send To Back**: Selected multiple elements can be moved using the

Bring to Front/Send to Back commands. They retain their relative order when placed.

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

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

These commands are not available if all elements in a scene are selected, a scene is selected, or if no elements have been selected.

- **Bring Forward** or **Send Backward**: Used with multiply selected items, **Bring Forward** moves the selected elements one layer forward in the layer order. The **Send Backward** command moves the selected elements one layer backward in the layer order. Any existing gaps among the selected elements are eliminated when they are placed.

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

Additional Notes about Layer Ordering

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

All the layer order editing options in the Arrange menu can be used in any window.

The layer order can also be changed by editing the element's layer order number in the Element Info dialog, by using the editing options in the layers window, or by using the object info palette.

ELIMINATING GAPS IN THE LAYER ORDER

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

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

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

To eliminate gaps in the layout or layers window:

- Select the scene and choose **Eliminate Gaps** from the Arrange menu. The gaps are

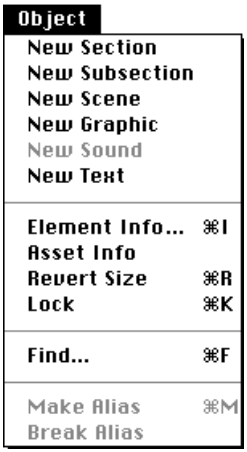
eliminated and the layer order of all elements within the scene is changed.

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

To eliminate specific gaps:

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

Chapter 6. Object Menu



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

- The first section of the menu contains commands for creating new project components. These options can be used in any editing view (except for New Sound, which can only be used in the structure view).
- The second section of the menu contains commands for obtaining information about, resizing, and locking project components.
- The Find option can be used to find project components based on many different search criteria.
- The last section of the menu contains an option for creating modifiers that share the same settings (Make Alias) and another that removes this functionality (Break Alias).

Object menu options are described in detail below.

ADDING SECTIONS, SUBSECTIONS, SCENES AND ELEMENTS TO A PROJECT

A mTropolis project has a structural hierarchy that begins at the project level, and descends through sections, subsections, scenes and elements (Figure 6.1). Modifiers can be added to any level in the project's hierarchy.

The **New Section**, **New Subsection**, **New Scene**, **New Graphic**, **New Sound** and **New Text** menu items in the Object menu provide a quick and convenient way to add a new

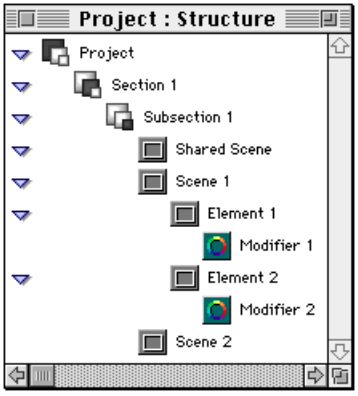


Figure 6.1 Structural view of a project hierarchy

component to a project while working in one of mTropolis' three editing views.

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

THE ELEMENT INFO DIALOG

The **Element Info** option (⌘-I) in the Object menu opens a selected element's Element Info dialog (Figure 6.2). This dialog is used to set the initial states of graphic, sound and text elements. The Element Info dialog can also be opened by double-clicking on the element. See "Tool Palette" on page 11.1 for complete information on creating elements.

- *Note: If this option is chosen when a Section element is selected, the Section Info dialog appears. See "Assigning Sections to Title Segments" on page 2.15.*

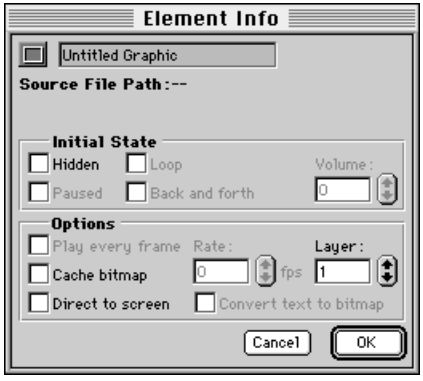


Figure 6.2 The Element Info dialog

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

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


A graphic element that has not been linked to media.


A graphic element that has been linked to a PICT.


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


A graphic element linked to a QuickTime movie.


A text element.


A sound element.

Element Name Field
By default, the name of the linked external media appears in this text field. A new name for the element can be entered without changing the name of the external media file.

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

Hidden Check Box
Select Hidden to initially hide an element during runtime. The element can be made

visible by sending it a *Show* or *Play* command (see "Shown/Show" on page 13.9 and "Played/Play" on page 13.12). Hidden elements do not respond to user mouse clicks, as described in "How mTropolis Generates Mouse Messages" on page 13.6

Loop Check Box
By default, movies and mToons will loop (i.e., play continuously). Uncheck this option to have them play only once.

Paused Check Box
The Paused check box is specific to movies, mToons and sounds. Choose Paused to initially pause a movie, mToon or sound at runtime. It can be made to "unpause" by sending it a *Play, Unpause* or *Toggle Pause* command (see "Played/Play" on page 13.12, "Unpaused/Unpause" on page 13.12, and "Toggle Pause (Message/Command Menu Only)" on page 13.13).

Back & Forth Check Box
When selected, this option allows mToons to play back and forth between the start and end positions.

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

Play Every Frame Check Box
This option forces playback of every frame of a QuickTime movie. For movies without sound, this ensures that the movie's playback speed is adjusted to the speed of the system. Turn this option off for movies with sound. Frames will be dropped to ensure proper sound synchronization.

Cache Bitmap Check Box
Checking Cache Bitmap converts the element into a bitmap graphic, optimizing screen re-draw time.

For example, gradient ink effects require significant processing time to be produced. An element with a gradient ink effect with the Cache Bitmap check box option selected will be stored in memory and called as required, greatly decreasing processing time.

- *Note: This option is limited by the memory available to the application.*

Direct to Screen Check Box
By default, elements are drawn in a layer order, creating a "2.5D" effect. Checking the Direct to Screen option improves element re-draw time by drawing an element on top of other elements in a scene. Elements using this option have a double dash (- -) symbol in their layer order number field.

Hot Tip

The Direct to Screen option decreases RAM use, allowing larger files to play more efficiently. However, if the object is to be moveable, uncheck Direct to Screen, as the re-draw time of draggable objects and animations is slowed.

Layer Field
Change the layer order number of an element by entering a new number here. See "Changing the Layer Order of Elements" on page 5.2 for more information on mTropolis layers.

- *Note: Miniscript can be used to change the layer order number of elements during runtime.*

Rate Field
Specify the rate of an mToon by entering a value here. Optionally, use the arrow keys to change the value.

Convert Text to Bitmap Checkbox
Check this box to convert this text element to a bitmap when the project is built into a title (see "Build Title" on page 2.16). The "text" becomes a bitmap "picture" of the text. Hence, machines that run the title do not need to have installed the fonts that are used in this text element. This option is useful for cross-platform applications where the font used in the text element is not available on all platforms. Text converted to a bitmap cannot be changed during runtime mode; don't use this option with text elements that will be editable by the user, display the value of mTropolis variables, or have their text changed through Miniscript commands. Note also that bitmap text takes up more space in the title, but is faster to display on the screen.

OK Button
Click OK to save the settings and close the dialog.

Cancel Button
Click Cancel to close the dialog without making any changes.

ASSET INFO

Select **Asset Info** from the Object menu option to display general information about the asset linked to the currently-selected element. The Asset Info dialog (Figure 6.3) appears.

Figure 6.3 The Asset Info dialog

REVERT SIZE

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

LOCK

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

When an element is locked it cannot be deleted, resized, or moved with the mouse. Media cannot be linked to an element that is locked. All other functions and capabilities of an element remain intact while it is locked.

Figure 6.4 The Find dialog

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

To lock an element while in edit mode:

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

To unlock an element while in edit mode:

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

FIND

Select **Find** from the Object menu to search for components in a project by name. The Find dialog appears (Figure 6.4). Components of the Find dialog are described below.

Find Items In

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

Whose Section

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

- Select **Name** (the default) to search for a project component based on characters in the component's name. Enter the string to be searched for in the rightmost pop-up. Alternatively, drag an element from the project and drop it in the rightmost pop-up—the pop-up changes to reflect the dropped element's name. The middle pop-up can be used to select the conditions under which the string will be found. Options include **Contains** (the default), **Starts with, Ends with, Is, Is not,** and **Doesn't** contain. Click the "Find" button to start the search.
- Select **Asset** to search for project components that make use of a specified asset (including modifiers that "link" to assets, such

as the sound effect modifier and color table modifier). The middle pop-up changes to "Is" (the only choice for this type of search). The rightmost pop-up changes to "Drop asset here". Drag an asset from the Asset Palette and drop it on the rightmost pop-up menu. Select the "Find" button to start the search.

- Select **Alias** to search for copies of a specified alias. This option is useful because individual copies of an alias used in a project can be given unique names. Thus, it may not be obvious from the editing views which aliases are copies of the same master. The middle pop-up changes to "Is" (the only choice for this type of search). The rightmost pop-up changes to "Drop alias here". Drag an alias from the project and drop it on the rightmost pop-up menu. Select the "Find" button to start the search.

Find Button

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

Items Found Window

The Items Found window (Figure 6.5) displays items located by a find operation. The top portion of the window shows a list of found items and their icons in alphabetical order. Select an item to see its location in the lower portion of the Items Found window.

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

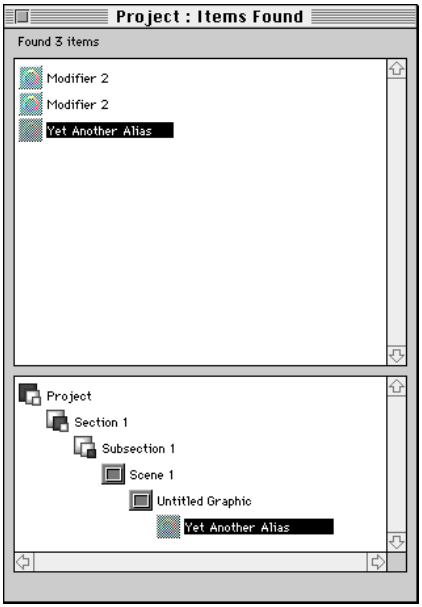


Figure 6.5 The Items Found window

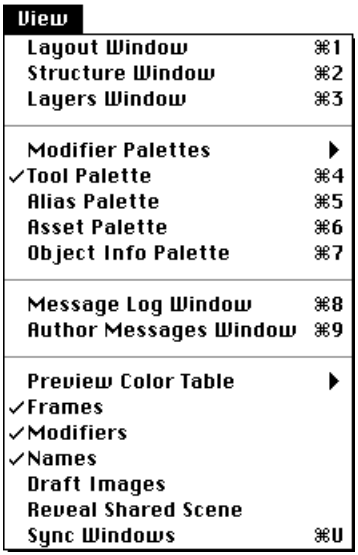
MAKE AND BREAK ALIAS

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

See “Alias Palette” on page 11.7 for a complete description of aliases, the alias palette, and the **Make Alias** and **Break Alias** options (see “Creating Aliases” on page 11.7 and “To “break” an alias:” on page 11.9).

Make and Break Alias

Chapter 7. View Menu



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

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

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

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

The message log window can be accessed from the View menu. This window displays the path of messages and commands that have been sent and received between components during runtime.

The View menu also contains options for altering the view of the current scenes in the layout and layers windows.

SELECTING AN EDITING VIEW

mTropolis’ layout, structure and layers windows provide three different ways of viewing and editing a project.

- The layout window provides a WYSIWYG (what you see is what you get) view of the project on a per scene basis. It is the best view for laying out graphic and text elements on the screen. See Chapter 9, “Layout Window”, for more information on this view.
- The structure window is used primarily for altering the structure of the project. New sections, subsections, scenes, elements, or modifiers can be added to the structure,

deleted, or shuffled. Some authors prefer to begin a project by building its structure in the structure window. Elements or modifiers that span scenes, especially sound elements, can be added and edited in this view. See Chapter 8, "Structure Window", for more information on this view.

- The layers window can be used to edit the elements and scenes in a subsection. The order of a subsection's scenes can be changed and the layer order of each scene's elements can be edited. New elements can also be added and configured with modifiers in this view. See Chapter 10, "Layers Window", for more information on this view.

To open the view windows:

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

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

Working with the Three Views

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

Copying and Pasting between Projects

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

To copy components from one project to another:

- Open both projects.
- Select a view in the first project.
- Click on the item to be copied.
- Choose **Copy** from the Edit menu (or use ⌘-C).
- Select a view in the second project.
- Select a destination in the second project's view.
- Choose **Paste** from the Edit menu (or use ⌘-V) while the second project is still selected. The item appears in the second project.
- *Note: With the exception of "project" icons, any component can also be copied into another project by simply dragging and dropping it onto a destination in the new project.*

WORKING WITH PALETTES

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

- **Modifier Palettes-Group 1** (⌘-Option-1). For a complete description of this palette, see "Modifier Palettes" on page 11.6.
- **Modifier Palettes-Group 2** (⌘-Option-2). For a complete description of this palette, see "Modifier Palettes" on page 11.6.
- **Modifier Palettes-Group 3** (⌘-Option-3). For a complete description of this palette, see "Modifier Palettes" on page 11.6.
- **Tool Palette** (⌘-4). For a complete description of this palette, see "Tool Palette" on page 11.1.
- **Alias Palette** (⌘-5). For a complete description of this palette, see "Alias Palette" on page 11.7.
- **Asset Palette** (⌘-6). For a complete description of this palette, see "Asset Palette" on page 11.9.
- **Object Info Palette** (⌘-7). For a complete description of this palette, see "Object Info Palette" on page 11.11.

MESSAGE LOG WINDOW

The message log window (Figure 7.1) displays the paths of messages and commands that have been sent and received between components during runtime. It also displays various error messages that may be generated during runtime. This window is useful for debugging mTropolis projects.

To display this window, choose **Message Log Window** from the View menu (or use ⌘-8).

Information appears in this window when returning to edit mode after runtime, but only if the window's **Enable Logging** checkbox is checked. For example, the message log displayed in Figure 7.1 shows those components that have received and responded to a *Mouse Down* message, to the point where an

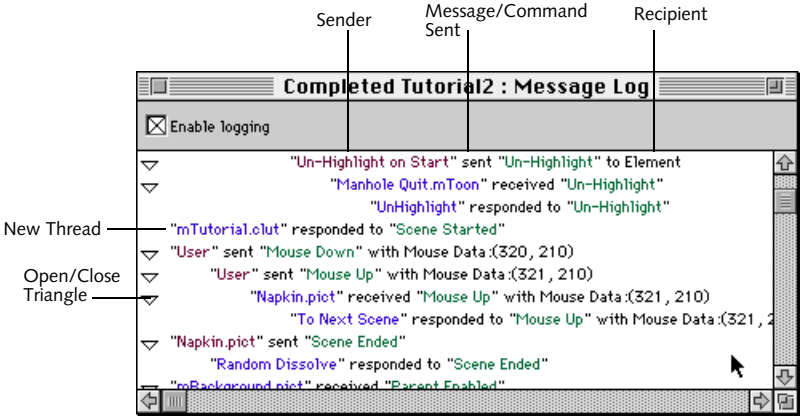


Figure 7.1 A Message Log window

idle is reached. Components of the message log window are described below.

Sender
On color monitors, senders of messages and commands are displayed in red.

Message/Command Sent
On color monitors, messages and commands that are passed between objects are displayed in green.

Recipient
On color monitors, recipients of messages or commands are displayed in purple.

Enable Logging Checkbox
Check **Enable Logging** to display the path of messages and commands that have been passed to selected components during runtime execution. See "Selecting Messages and Commands to be Displayed by the Message Log Window" on page 7.4.

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

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

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

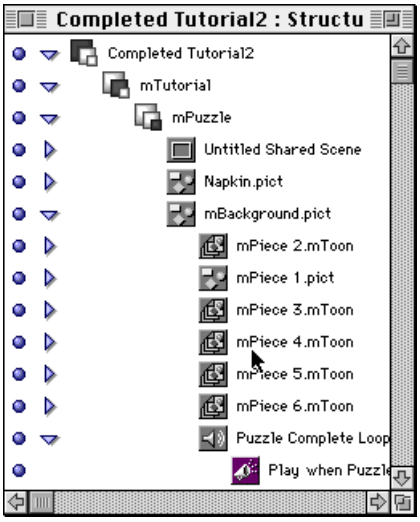


Figure 7.2 Selecting messages to be logged. In this example, all the components shown have their logging buttons enabled.

Selecting Messages and Commands to be Displayed by the Message Log Window

When the message log window's **Enable Logging** checkbox is checked, buttons appear to the far left of each component shown in the *structure* window (Figure 7.2). By default, these buttons are de-selected and look like flattened circles. Click a button to toggle message logging for the component that button is aligned with. The button expands into a "ball" to indicate that logging is enabled.

After a runtime execution, the messages and commands that have been passed to the selected components appear in the message log window.

Error Messages

The Message Log Window also displays certain types of error messages generated by mTropolis.

AUTHOR MESSAGES WINDOW

Select the **Author Messages Window** to display the Author Messages window (Figure 7.3).

Author messages are one way to activate modifiers. They can be created using the Message and Message/Command pop-ups in modifier dialogs, or in the Author Messages window. See "Author Messages" on page 13.6 for a complete description of author messages.

Using the Author Messages Window

To display the Author Messages window:

- Select Author Messages from the View menu. The Author Messages window appears (Figure 7.3).

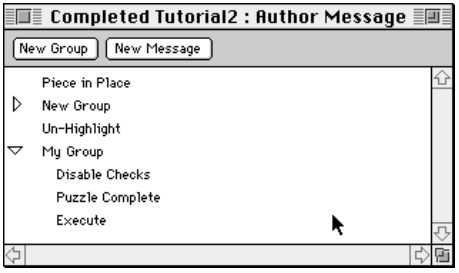


Figure 7.3 The Author Messages window

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

New Message Button

Click on this button to create a new author message. An untitled message appears in the Author Messages window.

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

New Group Button

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

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

Note that author message groups only contain author messages, they cannot contain other author message groups.

Open and Close Triangle

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

To delete author messages:

- Select either an author message or a group from the list and press the **Delete** key. Note that deleting an author message from the window deletes the author message from the

list but does not delete occurrences of this message in the project.

To edit or rename messages and groups:

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

To move author messages and groups into new groups:

Standard drag and drop methods can be used to move items in the Author Messages window.

- Select the author message or group to be moved and drag it to a new group. An outline of the dragged object appears.
- Drag the object to its destination. When the targeted destination group highlights, release the mouse button. Figure 7.4 illustrates this process.

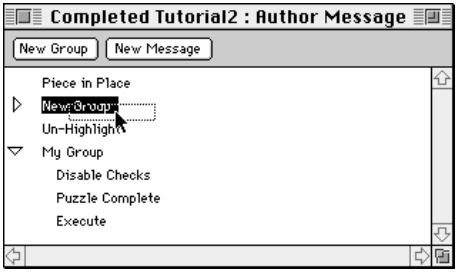


Figure 7.4 Moving an author message to a new group

To reorder author messages in the dialog:

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

USING VIEW MENU OPTIONS IN EDIT MODE

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

Preview Color Table

Select a color table name from this cascading menu to see the effect of a color table (that has been previously linked to the project) during edit mode. Select **Macintosh 8bit** to return the display to its default color scheme. Regardless of the setting in this menu, color table modifiers show their effects when triggered in runtime mode (see "Color Table Modifier" on page 12.24).

Frames

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

When **Frames** is toggled off, both the **Modifiers** and **Names** options are also automatically turned off. These items will appear greyed out in the View menu.

When **Frames** is toggled on, the **Modifiers** or **Names** options are also automatically turned on. These options can then be toggled individually.

Modifiers

To hide or show the modifier and behavior icons on elements (in the layout or structure view), use the **Modifiers** menu toggle in the View menu (or press ⌘-Shift-M). The default is on.

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

Names

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

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

Draft Images

Use the **Draft Images** menu toggle to switch between low- and high-resolution displays of a project's media. When **Draft Images** is toggled on, screen redraw time is reduced, which is especially useful when stepping through scenes in a project in the layout window. This option is also useful when using a less powerful computer for project development. The default is off.

Reveal Shared Scene

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

Sync Windows

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

WINDOW MENU OPTIONS

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

Window Menu Options

Chapter 8. Structure Window

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

To open the structure window:

- Choose **Structure Window** from the View menu (or use ⌘-2). The structure window appears as the active window.

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

STRUCTURE WINDOW OVERVIEW

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

The expanded view of a default project in the structure window (Figure 8.1) reflects this hierarchy. Components of the structure window are described below.

Element Levels

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

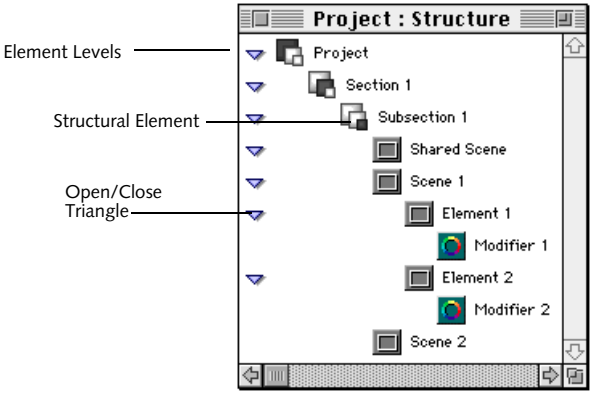


Figure 8.1 The Structure window

Structural Elements

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

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

Behaviors are a special kind of modifier that can contain other modifiers. Modifiers within behaviors are the children of that behavior. See "Behavior Modifier" on page 12.6.

Open and Close Triangles

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

Click on a triangle to toggle its element or behavior between open (pointing down) and closed (pointing to the right). Option-Click a triangle to open or close all levels of items within the element or behavior.

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

Open/close triangles are shown as either shaded or filled with white. Shaded triangles indicate that media associated with that component has been loaded into memory.

White-filled triangles indicate that the component contains media that has not yet been loaded by mTropolis.

The hierarchy of elements in a project affects the way that messages are passed between its components. See "Message Paths" on page 13.23.

Renaming Elements and Modifiers

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

To rename an item:

- Click twice on its name, but do not double-click (i.e., click once, wait briefly, then click again) or click the name and press the Return key. The text highlights.
- Type the new name.
- Alternatively, double-click an element or modifier. Its dialog will appear on the screen. Enter the new name in the **Name** field of the dialog and click **OK**. The name of a component can also be changed by entering a new name for the selected element in the object info palette (see "Object Info Palette" on page 11.11).
- *Note: Projects cannot be renamed in the structure window. The name of a project is the same*

as its filename. Use the Save As menu item in the File menu to rename a project.

MANAGING THE STRUCTURE WINDOW

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

Viewing Different Levels of the Structure Hierarchy

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

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

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

Controlling Open/Close Triangles

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

- **⌘-Right Arrow**: This keyboard command expands the selected component.
- **⌘-Left Arrow**: This keyboard command collapses the selected component.
- **⌘-Option-Left Arrow**: This keyboard command collapses the entire hierarchy under the selected component.
- **⌘-Option-Right Arrow**: This keyboard command expands the entire hierarchy under the selected component.
- *Note: To hide or show modifiers in the structure window, use the Modifiers menu toggle in the View menu.*

Stepping through the Structure Window

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

- **Tab** or **Down Arrow**: Use these keys to move down through the list.
- **Shift-Tab** or **Up Arrow:** Use these keys to move up through the list.
- **Shift-Select**: This keyboard command allows new objects to be added to a selection set.

ADDING COMPONENTS TO A PROJECT IN THE STRUCTURE WINDOW

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

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

Creating New Sections, Subsections, Scenes and Elements

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

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

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

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

To create a new section:

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

To create a new subsection:

- Click on an item in the structure view.
- Choose **New Subsection** from the Object menu (or use ⌘-Option-F). A new, untitled subsection is added to the project.
- *Note: When a subsection is created, a shared scene is automatically created with it.*

To create a new scene:

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

To create a new graphic element:

- Select an item in the structure view.
- Select **New Graphic** in the Object menu (or use ⌘-Option-G). A new graphic element named "Untitled Graphic" is added to the project.
- *Note: New graphic elements created in the structure window appear in the top left corner of their scene when viewed in the layout window. If more than one new graphic element is added to a scene in the structure window, the elements will overlap in the layout window. Switch to the layout view to resize and position the new elements in the scene. Graphic elements created in the structure window are 50 x 50 pixels in the layout window.*

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

To link an external media file to a graphic element:

- Select a graphic element in the structure view.
- Choose **Link Media** from the File menu (or use ⌘-L). A standard file dialog appears.
- Navigate to the media file. Only valid media files are listed (i.e., QuickTime movies, mToons and PICTs).
- Double-click on the file name, or select the file name and select **Link**. The element's icon changes to the icon for the media type. The file name of the external media appears to the right of the icon.
- *Note: Any media in existing elements can be changed simply by linking new media to the element. Optionally, drag and drop new media onto the element from the asset palette (see "Asset Palette" on page 11.9).*
- *Note: By default, media that is added to the project appears in the layout window at the screen size of the external media file. This behavior can be changed by unchecking the* **Auto Resize Linked Media** *checkbox in the Application Preferences dialog (see "Auto Resize Linked Media Check Box" on page 3.3).*

To create a new sound element:

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

To link an external sound file to a sound element:

- Select a sound element in the structure view.
- Choose **Link Media** from the File menu (or use ⌘-L). A standard file dialog appears.
- Navigate to the sound file. Only valid sound files are listed (i.e., snd and AIFF).
- Double-click on the filename, or select the filename and select **Link**. The filename of the external sound file appears in the text field to the right of the icon.
- *Note: Sounds cannot be added at the project, section or subsection level. Sound elements cannot be added to a project from the layout or layers windows—they can only be added in the structure view.*

To create a new text element:

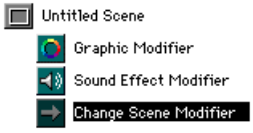
- Select an item in the structure view.
- Choose **New Text** from the Object menu (or use ⌘-Option-D). A new text element is added to the project.
- *Note: New text elements created in the structure window appear in the top left corner of their scene when viewed in the layout window. If more than one new text element is added to a scene in the structure window, the elements will overlap in the layout window. Use the object info palette to precisely position the new text element, or switch to the layout view to resize and position the new elements in the scene. Text elements created in the structure window are 50 x 50 pixels in the layout window.*

CHANGING THE ORDER OF COMPONENTS IN THE STRUCTURE WINDOW

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

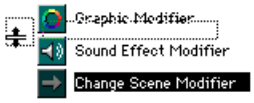
The following example illustrates the method for moving a modifier from one position in a scene to another. The method is the same for any component in a project.

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



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

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



- Release the mouse button. The modifier moves to its new position.
- *Note: Use Shift-Click to select multiple items to be moved.*

When using this method to change the order of elements, be careful that another element is not highlighted when the first element is dropped. Dropping an element on top of another element creates a parent/child relationship, affect the order in which messages are passed to elements in an unintended way. Parent/child relationships are discussed in more detail in "Message Passing among Elements" on page 8.6.

MESSAGE PASSING AMONG ELEMENTS

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

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

Effects of Parent/Child Relationships

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

New parent/child relationships between elements can be created in the structure window using the drag and drop methods outlined in "Creating New Parent/Child Relationships in the Structure Window" on page 8.7. Alternatively, the relationship between elements in a scene can be changed using the parent/child tool in the layout window. See "Parent/Child Tool" on page 11.3.

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

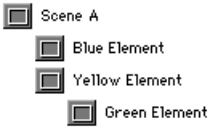
More information on the path of messages through a project can be found in "Message Paths" on page 13.23.

Creating New Parent /Child Relationships in the Structure Window

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

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

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



Note that the Green Element in the previous diagram is indented relative to the Yellow Element, indicating their new parent/child relationship. Parent/child relationships can also be made in the layout window using the Parent/Child Tool (see "Parent/Child Tool" on page 11.3).

- *Note: Multiple elements can be made children of another element. Use Shift-Click to select multiple elements to be moved.*
- *Note: Graphic elements cannot be made children of sound elements.*

Appearance of Parent/Child Relationships in the Structure Window

The structure window shown in Figure 8.2 shows a series of child elements and their modifiers. Relationships between the various

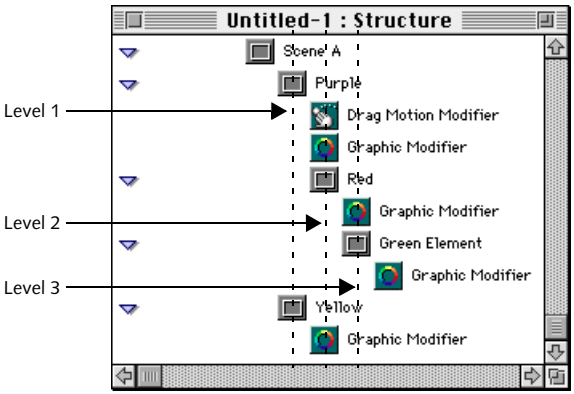


Figure 8.2 Relationships between components in the structure hierarchy.

levels shown in the figure are described below.

Level 1
By default, any element that is placed in a scene appears on the first level of elements in the scene. The element's modifiers appear below it, indented to the right. In this example, the elements Purple and Yellow are children of Scene A.

Level 2
The element Red has been made a child of element Purple. Therefore, Red appears on the next element level in the scene, which is indented relative to the Purple element. The Red element's Graphic Modifier appears below it, indented to the right.

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

Chapter 9. Layout Window

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

To make the layout window active:

- Choose the **Layout Window** option from the View menu (or use ⌘-1). The layout window (Figure 9.1) appears in the foreground.

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

LAYOUT WINDOW OVERVIEW

The layout window is the default view when mTropolis creates a new project (Figure 9.1).

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

The Scene

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

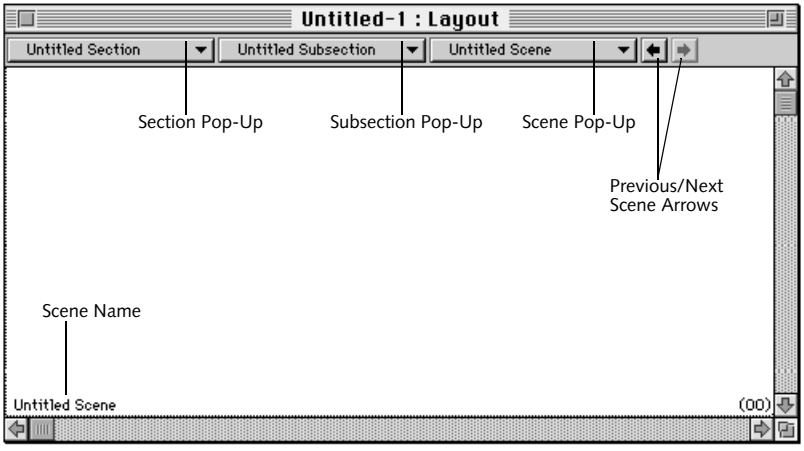


Figure 9.1 A new, empty layout window

Default Properties of Scenes

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

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

Initially, scenes are locked into position in the layout window. When locked, scenes cannot be moved. Unlike most locked elements, however, media can be linked to locked scenes using the **Link Media-File** option in the File menu. To unlock a scene, select the scene and select the **Lock** menu toggle in the Object menu (or use ⌘-K).

Layer Order Numbering of Elements

The first scene in a new project has a layer order of 00. As each element is added to the scene, it is given a unique sequential layer order number that is incremented by 1. The layer order number appears at the bottom right of the element. Elements with higher layer order numbers appear to be "in front" of elements with lower layer order numbers.

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

- Use the layer order options from the Arrange menu. See "Changing the Layer Order of Elements" on page 5.2.
- Drag and drop elements in the layers window. See "The Layer Order Grid" on page 10.2.
- Change the layer order number in the Element Info dialog. See "The Element Info Dialog" on page 6.2.
- Use the Object Info palette of a selected element. See "Object Info Palette" on page 11.11.

ADDING ELEMENTS TO SCENES

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

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

To create a graphic element in the layout window:

- Select the graphic tool from the tool palette. This tool looks like a single rectangle. The cursor changes to a cross hair.
- Click and drag on the scene to create a new graphic element. As you drag, a graphic element frame is created. The name "Untitled

Graphic" appears in the bottom left of the element's frame and its layer order number appears in parentheses at the bottom right.

To create a text element in the layout window:

- Select the text tool from the tool palette. This tool looks like a letter "A". The cursor changes to an "I" beam and a small box.
- Click and drag on the scene to create a new text element. As you drag, a new text element frame is created.
- Click inside the text element. A flashing insertion bar appears.
- Enter the desired text from the keyboard. When finished, choose the selection tool and click outside the text element to commit the text.
- The text is displayed in black on a white background. The name "Untitled Text" appears at the bottom left of the element's frame and its layer order number appears in parentheses at the bottom right.

Modifiers can be added to both graphic and text elements. Graphic elements can be linked to external media (see "Linking Media to an Element in the Layout Window" on page 9.3). Text in text elements can be modified using options from the Format menu (see Chapter 4, "Format Menu").

More information on the graphic and text tools can be found in "Tool Palette" on page 11.1. Elements can also be added to scenes in the structure and layer windows. See "Adding Components to a Project in the Structure Window" on page 8.4 and "Creating Scenes and Elements in the Layers Window" on page 10.3.

Linking Media to an Element in the Layout Window

- Select an existing element and choose **Link Media-File** from the File menu (or use ⌘-L). A standard file dialog appears. Valid media types will be listed.
- Select the desired media file and click **Link**. The media appears within the element's frame. A thumbnail of the media file is automatically placed in the asset palette.

By default, the element will conform to the spatial dimensions of the external media file. Resize the element by dragging on its frame or use the Object Info palette to precisely resize the element. See "Object Info Palette" on page 11.11.

- *Note: There are generally five levels of structural elements in a project's hierarchy: project, section, subsection, scene and element. Of these elements, only scenes (which behave much like graphic elements) and elements can contain external media.*
- *Note: Sounds can only be linked to sound elements, which can only be created in the structure window. See Chapter 8, "Structure Window" for details.*

Other Techniques for Linking Media

Media can also be linked to elements in the structure window, layers window, and asset palette:

- For instructions on how to create and link media to elements in the structure view, see "Creating New Sections, Subsections, Scenes and Elements" on page 8.4.
- For instructions on linking media in the layers window, see "Linking Media to Elements in the Layers Window" on page 10.6.
- The asset palette can also be used to link media, see "Linking Media to the Asset Palette" on page 11.10.

NAVIGATING IN THE LAYOUT WINDOW

The layout window provides a view of a single scene. Use the Section, Subsection and Scene pop-ups at the top of the layout window (Figure 9.1) to move to new locations in a project.

To navigate to a new scene:

- Select the appropriate section from the Section pop-up (if there is more than one section in the project).
- Select the appropriate subsection from the Subsection pop-up (if there is more than one subsection in the project).
- Select the desired scene from the Scene pop-up. mTropolis changes to the selected scene.

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

Adding New Sections, Subsections and Scenes to a Project

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

To create a new section or subsection:

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

When a new subsection is created, a shared scene is also automatically created. See "The Shared Scene" on page 9.5.

To create a new scene:

- Choose **New Scene** from the Scene pop-up. The new scene appears in the layout window. The scene is created with a default name ("Untitled Scene").
- Rename the new scene by double-clicking within its frame to display its Element Info dialog. Enter a new name into the element name text field. When the Element Info dialog is closed, the scene is renamed.
- *Note: Scenes are initially locked into position in the layout window. Locked scenes cannot be moved or resized. However, media can be linked to a locked scene by selecting the **Link Media-File** option in the File menu. To unlock a scene, select the scene and toggle **Lock** off in the Object menu (or use ⌘-K).*

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

- Choose **Object Info Palette** from the View menu. The palette appears.
- Click on the scene.
- Highlight the scene name, found in the top left of the Object Info palette.
- Edit the scene name in the name field.
- Click anywhere outside the name field to commit the change.
- *Note: Sections, subsections, scenes, elements and modifiers can also be renamed in the structure window. See "To rename an item:" on page 8.2.*

THE SHARED SCENE

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

The shared scene precedes all other scenes in the section.

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

To show the shared scene in edit mode:
By default, the shared scene is only visible in runtime mode, but it can be made visible in edit mode.

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

The Shared Scene

Chapter 10. Layers Window

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

To open the layers window:

- Choose **Layers Window** from the View menu (or use ⌘-3). The layers window (Figure 10.1) appears.

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

LAYERS WINDOW OVERVIEW

The layers window can be used to view and edit a subsection of the project at a time. Figure 10.1 shows a subsection from a project. Controls in the layers window are described below.

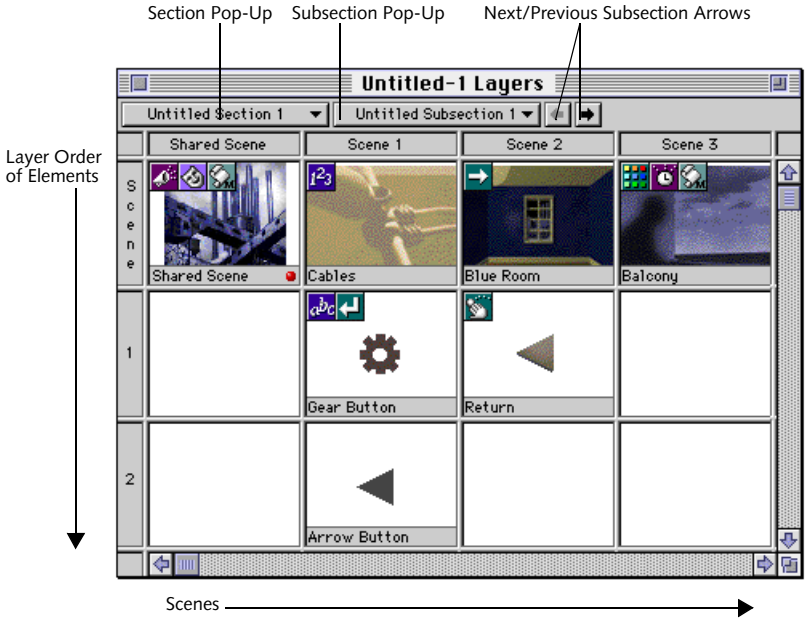


Figure 10.1 The Layers window. Items in the "grid" are child elements of the selected subsection organized horizontally by scene and vertically by the layer order of the elements.

The Layer Order Grid

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

Layer Order

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

Elements drawn on top of each other create the illusion that the two-dimensional (X,Y) screen has a depth (Z) dimension. The effect is commonly called "2.5D" (Figure 10.2).

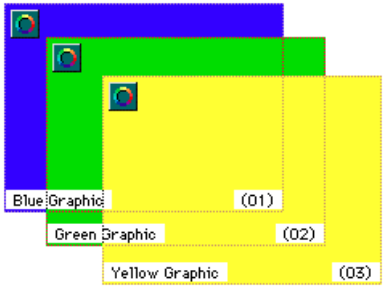


Figure 10.2 The effect of graphic layers

The layer order numbering begins with the scene at 00, and is incremented by 1 as new elements are added to the scene.

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

Scene Order

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

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

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

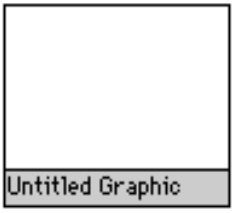
Shared Scenes

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

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

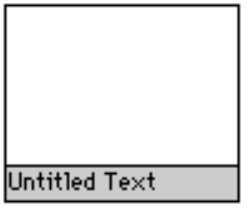
Elements and Modifiers

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



Graphic elements linked to external media (i.e., mToons, QuickTime movies, PICTs) are shown as small "thumbnail" images. The name of the element appears across the bottom. Any modifiers on the element will appear across the top of the thumbnail as shown below.

Text elements are displayed as white icons with the name of the text element at the bottom and any modifiers across the top of the thumbnail. A newly-created text element with the name "Untitled Text" looks like this in the layers window:



Hiding Editing Aids Using View Menu Options

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

- Toggle **Frames** off in the View menu (or use ⌘-Shift-F) to hide element frames, modifiers and names.
- Toggle **Modifiers** off in the View menu (or use ⌘-Shift-M) to hide element modifiers.
- Toggle **Names** off in the View menu (or use ⌘-Shift-N) to hide element names.

CREATING SCENES AND ELEMENTS IN THE LAYERS WINDOW

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

To create a new scene:

- Select the first unoccupied cell in the Scene row (i.e., the first row) of the layout window.
- Choose **New Scene** from the Object menu (or use ⌘-Option-S). A new scene is added after the last scene in the subsection.

A new scene can also be inserted between existing scenes:

- Select an existing scene and choose **New Scene** from the Object menu (or use ⌘-Option-S). The new scene appears in the targeted location. The scene that was in that location moves to the right (i.e., it becomes the "next" scene of the newly-created scene). All later scenes also move one position to the right.

To create a new graphic element:

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

To create a new text element:

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

EDITING IN THE LAYERS WINDOW

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

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

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

Tools that are *not* active in this window include the graphic, crop, text and parent/ child tools of the Tool palette.

Changing the Order of Scenes

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

To change the order of a scene in a subsection:

- Select the scene to be moved.
- Drag and drop the scene over the scene to be displaced. The dropped scene appears in the new location. The scene that previously occupied the targeted location moves one space to the right (i.e., it becomes the

"next" scene of the relocated scene). All other scenes in the subsection move to accommodate the change.

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

Changing the Layer Order of Elements

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

To move an element between other elements:

- Select an element in the layers window.
- Drag and drop the element over the element to be displaced. The dropped element appears in the new location. The element that previously occupied the targeted location moves down one position in the layer order. All later elements in the layer order also move down.
- *Note: Use Shift-Click to select a group of elements to be moved. Also, Option-Drag can be used to move a copy of an element, leaving the original in place.*

To move an element to an empty grid location:

- Click on the element and drag and drop it onto the empty grid location. The element appears in the new grid cell.

Duplicating Scenes or Elements

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

To duplicate a scene or element:

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

Other Methods for Editing the Layer Order of Elements

The Arrange Menu Options

The options for changing the layer order of elements in the Arrange menu (**Bring to Front, Send to Back, Bring Forward, Send Backward**) can be used in all editing windows, including the layers window. See "Changing the Layer Order of Elements" on page 5.2.

The Element Info Dialog

An element's layer order number can be changed in its Element Info dialog. This method can be used in all the editing windows. Double-click on the element or choose **Element Info** from the Object menu (or use ⌘-I) to open the selected element's Info dialog. See "The Element Info Dialog" on page 6.2.

Object Info Palette

The object info palette can be used to view and change the size, position and layer order number of a selected element. It can also be used to change an element's name. To open this palette, select **Object Info Palette** from the View menu (or use ⌘-7). See "Object Info Palette" on page 11.11.

Eliminating Gaps in the Layer Order

If an element is moved from one scene to another, a gap is created in the layer order of elements in the original scene. Figure 10.3 shows what happens when an element is moved into another scene.

Figure 10.3 shows the gap that has been created in Scene 1 by moving the Balcony element from its original position (upper left illustration) into Scene 2 (upper right illustration). This gap can be eliminated by selecting Scene 1 and choosing **Eliminate Gaps** from the Arrange menu. The elements in Scene 1 are moved to eliminate empty layers between elements (lower illustration).

LINKING MEDIA TO ELEMENTS IN THE LAYERS WINDOW

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

To link a an external media file:

- Select an element.
- Choose **Link Media-File** from the File menu (or use ⌘-L). A standard file dialog appears.

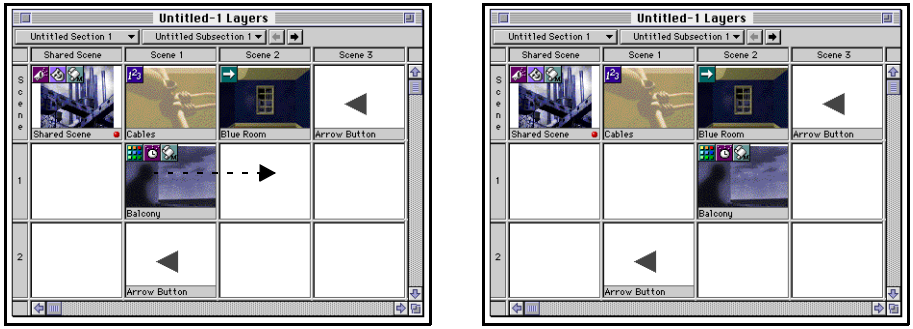
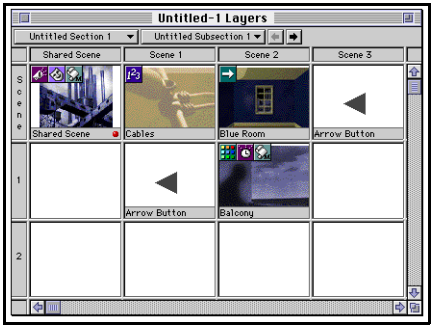


Figure 10.3 Eliminating gaps in the layer order: Before moving Balcony element (upper left); After moving Balcony element (upper right); After applying Eliminate Gaps from the Arrange menu (below).



- Navigate to the external media file. Valid file types appear in the file list (QuickTime movies, mToons and PICTs).
- Double-click on the desired filename or click on the filename and choose **Link**. A new thumbnail representing the media appears in the grid cell. The filename of the linked file replaces the word "Untitled Graphic."
- The element can be renamed by double-clicking it to display its Element Info dialog, where a new name can be entered, or by using the Object Info palette.

Adding Graphic Elements to the Layers Window from the Asset Palette

All media that has been linked to a project is stored in the asset palette. These assets can be dragged and dropped into a project in any view.

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

To add a media element from the asset palette:

- Open the asset palette by selecting **Asset Palette** from the View menu (or use ⌘-6). The asset palette appears.
- Drag the desired media element from the asset palette to an empty grid location. An image representing the media appears in the grid cell.

NAVIGATING IN THE LAYERS WINDOW

Use the Section and Subsection pop-ups at the top of the layers window (Figure 10.1 on page 10.1) to display different subsections of a project.

To navigate to a new section:

- Select the section in which to work from the Section pop-up.
- Select the subsection in which to work from the Subsection pop-up. The layers view displays the scenes of the newly-selected subsection.
- The Next/Previous Subsection arrows can be used to move to other subsections in the section.

Adding New Sections or Subsections to a Project

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

To create a new section or subsection:

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

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

To rename a section or subsection in the structure window:

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

To rename a section or subsection using the object info palette:

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

Chapter 11. Palette Reference

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

When visible, the palettes float over the three editing views (i.e., the layout, structure and layers windows).

- The Tool palette contains tools for creating graphic and text elements in the layout window. Tools on this palette can also be used to size elements and link them in new hierarchical relationships.
- The Modifier palettes contain the modifiers that are supplied with mTropolis.
- The Alias palette stores the originals of modifiers that have been aliased in a project. The alias palette also shows the number of aliases of each original used in the project.
- The Asset palette stores icons and thumbnails of media linked to the project.
- The Object Info palette is used to view and change the name, size, position and layer number of elements. It can also be used to identify the current cel of an mToon and to step through its cels while in edit mode.

TOOL PALETTE

The Tool palette (Figure 11.1) contains tools for creating graphic and text elements, sizing them and linking them in new hierarchical relationships. Only one tool can be selected at a time.

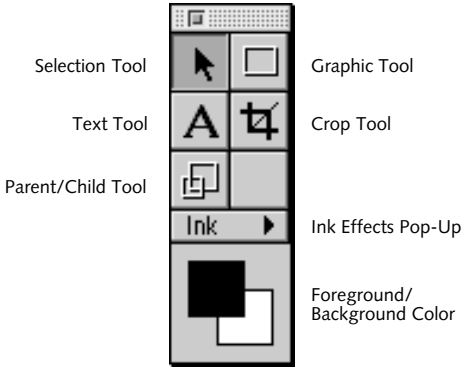


Figure 11.1 The Tool palette

Selection Tool

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

- Select items by single clicking.
- Select multiple items by Shift-clicking or by dragging on the screen to enclose items in a marquee.
- Move items by clicking and dragging.

- Resize elements by clicking and dragging their frames. Note that resizing also works with multiple selections.
- Open an element or modifier dialog by double-clicking on the item.

Graphic Tool

The graphic tool can be used to create new graphic elements in the layout window. New graphic elements can then be linked to media files (see "Linking External Media Files to Elements" on page 2.13).

- *Note: This tool cannot be used in the structure or layers window. To create graphic elements in these views, select* **New Graphic** *from the Object menu.*

To create a graphic element in the layout window:

- Click on the graphic tool. The cursor changes to a cross hair.
- Click and drag on the scene displayed in the layout window to create a new graphic element frame. The default name of the item, "Untitled Graphic," appears in the bottom left of the frame, and its layer order number appears in parentheses at the bottom right.

Graphic element properties can be changed by applying ink effects (see "Ink Effects" on page 11.4 and "Foreground and Background Colors" on page 11.6) or graphic modifiers (see "Graphic Modifier" on page 12.33).

Text Tool

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

- *Note: This tool cannot be used in the structure or layers window. To create text elements in these views, select* **New Text** *from the Object menu.*

To create a text element in the layout window:

- Click on the text tool. The cursor changes to an "I" beam and a small box.
- Click and drag on the scene displayed in the layout window to create a new text element frame of the desired size.
- Click inside the text element. A flashing insertion bar appears.
- Enter the desired text from the keyboard (text can also be entered by pasting). The text is displayed in black on a white background.
- Click on the selection tool, or another tool in the tool palette, to end text entry. The default name of the item, "Untitled Text," appears in the bottom left of the frame and its layer order number appears in parentheses at the bottom right.

Text properties can be changed using the options found in the Format menu (see Chapter 4, "Format Menu") or by applying text style modifiers (see "Text Style Modifier" on page 12.76), ink effects (see "Ink Effects" on page 11.4 and "Foreground and Background

Colors" on page 11.6), or graphic modifiers (see "Graphic Modifier" on page 12.33).

- *Note: When a project is built into a title for distribution, fonts are not bundled into the build file. To display properly, any fonts used by the project must be installed on the target system.*

Crop Tool

The crop tool can be used to reduce the size of the visible area of media in graphic elements or the visible area of text in text elements. The media itself is not resized; rather, a new view of the media is created.

To crop an element:

- Select the crop tool. The cursor changes to a crop tool icon.
- Click and drag from the corner of the graphic element. The visible area of the media is cropped.

To immediately undo a cropped element:

- Choose Undo from the Edit menu (or use ⌘-Z). The element returns to its pre-cropped size.
- *Note: Alternatively, select the Revert Size option in the Object menu (⌘-R) to return the element to its original size.*

Parent/Child Tool

The parent/child tool can be used to alter the hierarchical relationship between elements in a scene.

When elements are created in the layout window, they are automatically positioned below their scene in the project's structural hierarchy (i.e., they are children of the scene). When the hierarchical relationship between a scene and its elements is changed, the path of messages that are passed through the scene is altered. For example, when an element is made a child of another element in a scene, messages will be passed from the parent element to its child element(s) before being passed to the next element in the scene. More information about parent/child relationships and message passing can be found in "Message Passing among Elements" on page 8.6.

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

The sections below describe techniques for using the parent/child tool. When working with this tool, keep in mind that, by default, the scene is the parent of the elements it contains. When using the parent/child tool, make sure the scene is deselected.

To make one element the child of another:

- Click on the parent/child tool.
- Click and hold on the element targeted as the child.
- Drag the mouse. A line appears.
- Drag the line onto the element targeted as the parent and release the mouse button.

The elements are linked. Now when the parent is moved, the child moves with it.

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

To break the parent/child relationship between two elements:

- Click on the parent/child tool.
- Click and drag on the child element.
- A line appears. Release the mouse to drop the line on the scene. The scene becomes the child's new parent, breaking the relationship with the other element.

To make several elements children of one element:

- If it is currently selected, de-select the scene.
- Use Shift-Click to select multiple elements to be made children. Alternatively, drag a marquee around the elements.
- Click on the parent/child tool.
- Click and drag from one of the selected elements. Lines from all the elements attach to the cursor.
- Drop the lines onto the element targeted as the parent. The targeted element becomes the parent of the multiply-selected elements.

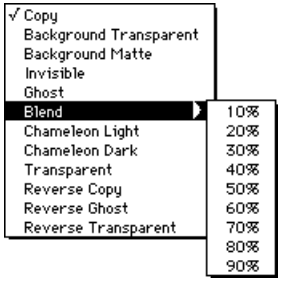
To break the parent/child relationship between a parent and its children:

- Select one or more of the child elements.
- Click on the parent/child tool.
- Click and drag from any selected child element to the scene. Lines from all the elements attach to the cursor.
- Drag and drop the lines on the scene. All the elements become children of the scene. The previous parent/child relationship among the elements is broken.

Using the Structure Window to Modify Parent/Child Relationships

Parent/child relationships can also be created in the structure window. See "Creating New Parent/Child Relationships in the Structure Window" on page 8.7.

Ink Effects



Ink effects can be applied by selecting an element and choosing an option from the Ink pop-up. These options are the same as the ones available in the "Ink:" menu of the graphic modifier dialog (see "Graphic Modifier" on page 12.33). The Ink pop-up dialog options are described below.

Copy

This is the default ink effect. It displays the graphic element in its original state.

Background Transparent
Background Transparent makes the element's background color transparent. All pixels of the specified color in the element are made transparent. That is, other elements drawn beneath the element will be visible through the transparent areas. The color is specified in the background color swatch (see "Foreground and Background Colors" on page 11.6).

Background Matte
Background Matte is similar to Background Transparent. In addition to adding transparency, however, this ink makes the transparent regions unresponsive to mTropolis events. Using this ink is a simple way to create objects that behave as if they have a complex border.

Invisible
An element is invisible when this option is applied.

Ghost
Applied to black and white elements, Ghost creates an image that can only be seen when placed over a black background. In color, Ghost draws with the current background color. Since white is mTropolis' default background color, Ghost can be used to make text appear white on a transparent background.

Blend
The blend ink effect makes the colors of overlapping elements blend together. The blend amount can be selected, in increments of 10% from 10% to 90%, from the cascading menu of blend percentages.

Chameleon Light
With this ink effect, the colors in a graphic on top of a white graphic turn opaque. When placed over a colored graphic, light colors are tinted.

Chameleon Dark
With this ink effect, the colors are lightened when the element is on top of another graphic. Placed over white, the graphic becomes transparent. Placed over a colored graphic, dark colors are tinted.

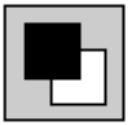
Transparent
Applied to black and white elements, this ink creates an image that can only be seen when placed over a black background. In color, Transparent draws with the current background color.

Reverse Copy
Reverses the black and white colors: all white pixels will appear black, all black pixels will appear white. Colors are inverted.

Reverse Ghost
Black pixels become transparent and white pixels remain white. When a graphic with this ink is placed over a graphic with white pixels, the white in the first graphic becomes transparent.

Reverse Transparent
White pixels change to black, and black pixels become transparent.

Foreground and Background Colors



The foreground and background color swatches are used to change the appearance of elements in the layout window.

By default, the foreground color of elements is black, and the background color is white. For text elements, the foreground color is the color used when entering text into text elements, and the background color is the color used to draw its field.

To change the foreground/background color of a graphic or text element:

- Select an element.
- Click and hold on either the foreground or background color box. The cursor changes to an eyedropper tool and a palette appears. If the scene has not been assigned a custom palette, the system palette appears. If the scene has been assigned a custom palette, this palette will appear.
- Drag the eyedropper to a color on the palette *or* to a color in an image on the screen. Release the mouse. The foreground or background color in the element changes to the selected color.
- *Note: For text elements, the foreground color is the color used for the letters themselves.*

MODIFIER PALETTES

The modifier palettes contain mTropolis' built-in modifiers. To display or hide each palette, choose **Modifier Palettes-Group 1** (⌘-Option-1), **Modifier Palettes-Group 2** (⌘-Option-2), or **Modifier Palettes-Group 3** from the View menu.

Each palette contains a number of modifiers, represented by icons. To see the name of a modifier in the palette, move the cursor over a modifier's icon while holding down the Control key.

Complete descriptions of the modifier palettes and individual modifiers can be found in Chapter 12, "Modifier Reference".

Applying Modifiers to Project Components

Modifiers can be placed on any element or in any behavior in a mTropolis project. The method for placing a modifier is the same in all views:

- Drag a modifier from the modifier palette and drop it on its destination. In the layout and layers views, the modifier appears on the element on which it was dropped. In the structure view, the modifier appears below and to the right of the element on which it has been dropped.

To add a modifier to a behavior:

- Drag the modifier over a behavior icon that has already been placed in a project. When the behavior icon highlights, drop the modifier by releasing the mouse.
- Optionally, double-click on the behavior to open it. Drag and drop the modifier into the behavior window. The modifier is added to the behavior.

More information on behaviors can be found in "Behavior" on page 12.13.

Changing Modifier Defaults

Once a modifier has been placed on a project component, the modifier's effect can be specified by altering the default settings in its dialog.

To view or change the settings in a modifier's dialog:

- Double-click on the modifier. The modifier's dialog box appears.
- Change any settings within the modifier dialog. The new settings take effect when the dialog is closed. See Chapter 12, "Modifier Reference" and Chapter 13, "Modifier Pop-Up Menus and Message Reference" for more information on modifier settings.

ALIAS PALETTE

The alias palette stores the master copies of aliased modifiers. An alias is a productivity tool that is used to globally manage modifiers with identical settings. Select **Alias Palette** from the View menu (or use ⌘-5) to display the alias palette (Figure 11.2)

An alias is a special copy of a modifier; it takes its functionality from the modifier from which it was made. This "master copy" is automatically placed in the alias palette when the alias is made (using the **Make Alias** option from the Object menu or press ⌘-M). Additional aliases that refer to this master copy can also be created and placed throughout a project.

The advantage of using aliases is that they can be globally updated from a single source. Changing the settings of an alias changes the settings of both the master copy and all other aliases that refer to the same original throughout the project.

Variables, which are a class of modifiers that store values, are good candidates for aliasing. For example, a variable used to contain the score in a game can be aliased and strategically placed in the project for access by specific messengers or Miniscript modifiers. As the value of this variable changes during the game, the value of its aliases will also change, giving the modifiers efficient access to the updated score. See "Variable Scopes" on page 13.25.

Creating Aliases

An alias can be created by selecting a modifier that has been placed in the project, and choosing **Make Alias** from the Object menu (or use ⌘-M). Additional alias copies can be made by selecting and dragging the icon of the master copy from the alias palette.

To make an alias:

- Select a modifier. More than one modifier can be selected using Shift-click.
- Choose **Make Alias** from the Object menu (or use ⌘-M). The selected modifier is now an alias and the modifier icon appears dimmed. A master copy of the modifier is automatically placed in the alias palette.
- *Note: An alias can also be created simply by dragging a modifier from a component and*

dropping it in the alias palette window. Choose **Alias Palette** *from the View menu (or use* ⌘*-5) to view the master copies of aliases in a project.*

Alias Palette Components

The alias palette (Figure 11.2) shows the master copies of all aliases in the current project. Components of the alias palette are described below.

Trash Can

When all aliases of a master copy have been deleted from the project, drag the master copy to the trash to remove it from the alias palette.

Modifier Icons

The master copy of an aliased modifier is automatically placed in the alias palette. Shown on the palette are the modifier's icon and name.

Using Aliases

Once an alias has been created, its master copy resides on the alias palette. Creating and managing aliases is simple.

To modify an alias:

- To modify an alias such that all copies of the alias are updated, double-click any one of the instances of the alias found in the project. The modifier's dialog appears.
- *Note: Master copies of an alias cannot be edited (i.e., you cannot double-click an alias on the alias palette to display its dialog). An instance of the alias must be present in the project to edit the aliased modifier.*

To create a new alias copy:

- Drag the desired modifier from the alias palette. A new alias copy attaches to the selection tool. The master copy remains on the palette.
- Drop the alias in the desired place in the project.
- *Note: Aliases can also be duplicated using the standard* **Cut/Copy/Paste/Duplicate** *menu*

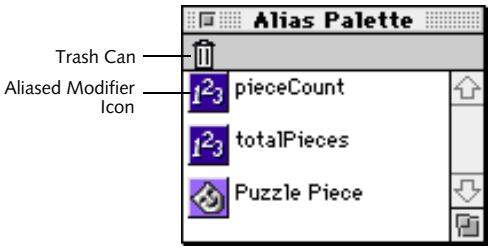


Figure 11.2 The Alias palette, containing a number of alias master copies.

items or by Option-dragging an alias to a new position.

To delete an alias from a project component:

- Select the alias and press the Delete key (or use **Cut** from the Edit menu).

To delete a master copy of an alias:

- To delete a master copy from the alias palette, first delete all of its aliases from the project, then drag the master copy to the trash alias palette's trash can.

To "break" an alias:

- An alias can be "broken" from its relationship to other instances of the alias and to the master copy in the alias palette with the **Break Alias** menu item in the Object menu. Once broken from the group, any subsequent changes to the modifier apply only to that specific modifier.

  A modifier or behavior that has been removed from the alias chain will retain its current settings until they are changed.

ASSET PALETTE

The asset palette is a visual database of the media that have been linked to a mTropolis project. This palette makes linking, storing, and copying media easy and convenient. Select **Asset Palette** from the View menu to display the asset palette (or use ⌘-6).

Media can be displayed in the asset palette as large or small thumbnail images, or icons and names. Each item in the palette can be dragged and dropped onto elements.

Asset Palette Components

By default, all media files in the asset palette (Figure 11.3) are displayed by large thumbnails. Components of the asset palette are described below.

Trash Can

Remove unused media files from the asset palette by dragging them into the trash can. Only media files that are not being used in the project can be removed.

Show Pop-Up Menu

The Show pop-up allows the display of media files to be restricted to selected types as follows:

- **All**: Displays all types of media linked to the project.

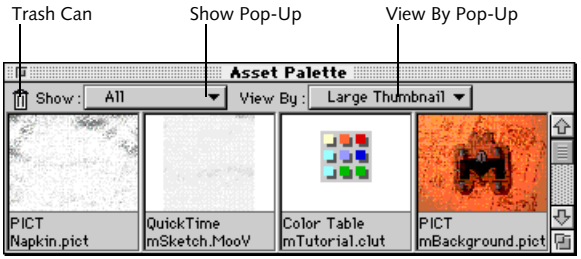


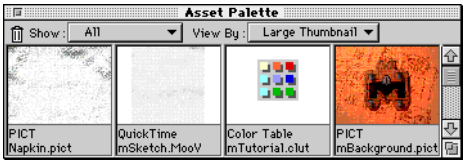
Figure 11.3 The Asset palette, showing thumbnails of available media

- **PICTs**: Displays PICT files linked to the project.
- **mToons**: Displays mToons linked to the project.
- **Sounds**: Displays AIFF and snd files linked to the project. When displayed by large thumbnail, sounds can be previewed by clicking on the sound thumbnail's Play button.
- **QuickTime**: Displays QuickTime movies linked to the project.
- **Color Tables**: Displays CLUT files linked to the project.
- **All Graphics**: Displays all graphic files (PICTs, QuickTime movies and mToons) that have been linked to the project.

View By Pop-Up Menu

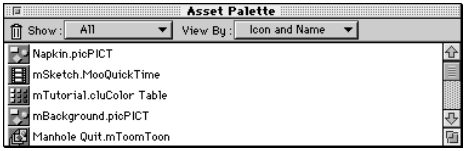
By default, the asset palette displays the media file as a large thumbnail. The View By pop-up provides an option for reducing the size of these thumbnails, or displaying each media file by its icon and name:

- **Large Thumbnail**: Displays a large thumbnail, the name of the file, the type of media and the number of times it is used in the project.



- **Small Thumbnail**: Displays the thumbnail at 50% size and removes the text description of the file.

- **Icon and Name**: Displays an icon related to the media type, the file's name, its media type and the number of times it is used in the project.



Linking Media to the Asset Palette

- Choose **File, Multiple Files** or **Folder** from the **Link Media** submenu in File menu. A standard file dialog appears. Files with valid media types are listed. Valid media types are described in "Valid Media File Types and Associated Thumbnails" on page 11.11.
- Double-click on the name of the item to be linked or click on the name of the item and click on **Link**. If multiple files are being linked, the file dialog will reappear after each selection is made. If a folder is to be linked, click on the button below the list of item names once the target folder's name appears in the button. A thumbnail of each asset appears in the asset palette.

Media can be dragged and dropped into the various views from the asset palette. When media is dropped onto a scene in this way, an element is automatically created to contain it.

Replacing Media in an Element

Media in an element can be replaced with new media from the asset palette. To accomplish this, drag and drop the media from the palette over the element.

Viewing Source File Information

To view details regarding an asset's source file, double-click on its image or icon in the asset palette. The Asset Info dialog is displayed. See "Asset Info" on page 6.4.

Valid Media File Types and Associated Thumbnails

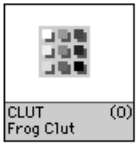
The following list shows the types of external media files supported by mTropolis, along with examples of their thumbnails.

Still image in PICT format

Animation in mTropolis mToon animation format

Movie in QuickTime format

Sounds in AIFF or snd format



Color look-up table (CLUT)

OBJECT INFO PALETTE

The object info palette (Figure 11.4) is used to view and change the size, position and layer number of elements, or to change the name of any component. The Object Info palette reports the following information about an object:

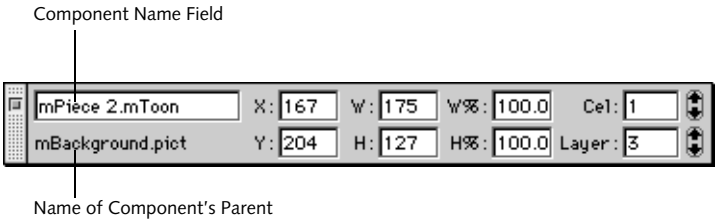


Figure 11.4 The Object Info palette

Object Info Palette

Component Name Field
This text field displays the selected component's name. This field is editable. To change the name of the component, click in this field and type a new name.

Name of Component's Parent
This non-editable text field below the component name displays the name of the component's parent. In Figure 11.4, "mBackground.pict" is the parent of "mPiece 2.mToon".

X and Y Coordinates
These values represent the position of a selected element in pixels relative to its parent. The origin of elements is their upper left corner.

The position of an element can be changed by dragging it to a new position in the layout window, or by entering new numbers in these editable fields.

Width and Height
The "W:" field shows the width of the selected element. The "H:" field shows its height. Values are in pixels.

The dimensions of an element can be changed by dragging its boundaries in the layout window or by entering new numbers in these editable fields.

Relative Scale
The "W%:" and "H%:" fields display the percentage to which the image has been scaled, in width and height, respectively, relative to the original image.

The dimensions of an element can be changed by dragging its boundaries in the layout window or by entering new numbers in these editable fields.

Cel Number
The "Cel:" field identifies the current cel number of a selected mToon.

The up/down arrow buttons to the right of this field allow stepping backward or forward through the animation from a specified cel. These buttons affect the appearance of the animation in edit mode only. They allow the author to easily preview the animation without impacting how it has been configured to appear in runtime.

Layer Order Number
The "Layer:" field shows the layer order number of the selected element. Use the up/down arrows to the right of this field to select a new layer order number for the element, or enter a new number in this editable field.

If a new layer order number is assigned to an element, and that number has already been assigned to another element, the layer order numbering of elements is updated to accommodate the change. The previous occupant of the layer moves one later in the layer order. Any elements with later layer order numbers also move in a similar way.

Chapter 12. Modifier Reference

This chapter describes the mTopolis' built-in modifiers. Chapter 13, "Modifier Pop-Up Menus and Message Reference" describes the Message and Message/Command pop-up menus found in all modifier dialogs. Chapter 14, "Miniscript Modifier", describes the scripting language used by the Miniscript modifier.

MODIFIERS OVERVIEW

Modifiers are special mTropolis components that modify the properties of other components in a project.

Modifiers are used by dragging them from one of the modifier palettes and dropping them on the object that they are to modify. Each modifier on the modifier palettes has unique capabilities or properties. When a modifier is dropped onto a component, the component assumes these capabilities or properties.

For example, a gradient modifier has the ability to alter the visual characteristics of graphic elements. When dropped onto a graphic element, the gradient modifier's capabilities are added to the information that makes up that object, as shown in Figure 12.1.

While some modifiers have the ability to change the visible characteristics of the elements onto which they are placed, other modifiers change invisible characteristics, or *properties,* of the element that contains them. For example, when a floating-point variable modifier that contains the value 2.5 is placed on an element, the physical representation of the element does not change, but its content, the value 2.5, becomes an intrinsic part of the element that contains it.

All modifiers can be configured (i.e., their capabilities can be customized) by changing the default settings in their modifier dialogs. In addition, most modifiers can be configured to apply their effects at specific times through a process called messaging.

A message in mTropolis can be as simple as a mouse click, or as complex as an author-defined message that is generated only after specific conditions in the runtime environment have been met. Some messages are generated by mTropolis during runtime and automatically sent to specific components throughout the project, and others can be sent to compo-

Figure 12.1 The gradient modifier, placed on a graphic element

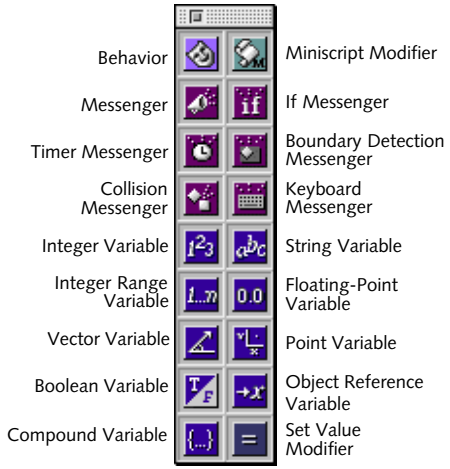


Figure 12.2 Modifier Palette-Group 1

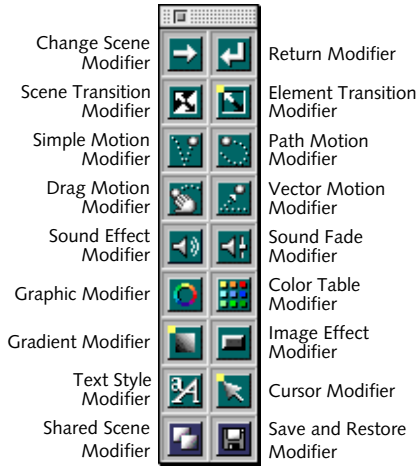


Figure 12.3 Modifier Palette-Group 2

nents from special modifiers called messengers.

For details regarding messaging within the mTropolis environment, and instructions on how to configure modifiers and messengers to respond to messages, see Chapter 13, "Modifier Pop-Up Menus and Message Reference".

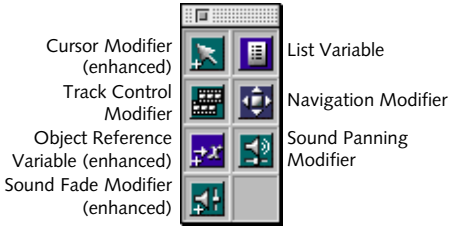


Figure 12.4 Modifier Palette-Group 3

MODIFIER PALETTES

The modifier palettes (Figure 12.2, Figure 12.3, and Figure 12.4) contain mTropolis' built-in modifiers. Toggle the display of these palettes on and off by selecting **Modifier Palette-Group 1**, **Modifier Palette-Group 2**, or **Modifier Palette-Group 3** from the View menu.

Modifiers can be applied to project elements by dragging modifier icons from the palette and dropping them on their destination.

Macintosh-only Modifiers

Some of the modifiers in this version of mTropolis work only in mTropolis titles built for the Macintosh—they have no effect in titles built for Windows platforms. These modifiers are marked with a yellow dot in the upper left corner of their icons. Be careful

when using these Macintosh-only modifiers in a title that will be built for both platforms.

Names of Modifiers

To see the names of modifiers in a palette, hold down the Control key while moving the cursor over the palette.

Some notes about the names of modifiers and their use in this documentation is in order:

- The names of variable modifiers all end with the word "variable" (e.g., "boolean variable", "string variable"). In the documentation, variable modifiers are often referred to simply as "variables".
- Messenger modifiers all end with the word "messenger" (e.g., "timer messenger", "collision messenger"). In the documentation, messenger modifiers are often referred to simply as "messengers".
- To accentuate the difference between the behavior modifier, which can contain other modifiers and behaviors, and all other modifiers that cannot, the behavior modifier is often referred to simply as a "behavior".
- All other modifiers are referred to by their full names (e.g., "path motion modifier", "graphic modifier").

MODIFIER TYPES

Most of the modifiers in the modifier palettes can be grouped into one of five categories. The icons for the modifiers have color-coded backgrounds that identify their category:

- Effects, which further modify the characteristics of the elements on which they are placed. Effect icons have a green background. See "Effect Modifiers" on page 12.4 for brief descriptions of the modifiers in this group.
- Variables, which store data values of various types such as text or integers. Variable icons have a purple background. See "Variable Modifiers" on page 12.5 for brief descriptions of the modifiers in this group.
- Messengers, which send messages, commands and data to specific destinations. Messenger icons have a dark red background. See "Messenger Modifiers" on page 12.6 for brief descriptions of the modifiers in this group.
- The Miniscript modifier is a special type of modifier that provides access to mTropolis' scripting language. Depending upon the script used, a Miniscript modifier may act like an effect, messenger, or both. This modifier has a pale green background.
- The behavior modifier is a special type of modifiers used to encapsulate groups of modifiers and behaviors. Optionally, a behavior can respond to messages for enabling and disabling its encapsulated modifiers. This modifier has a pale purple background.

Short descriptions of the modifier types and individual modifiers follow. Complete documentation for each modifier's dialog can be found at the end of this chapter.

Effect Modifiers

In general, modifiers grouped under the category of effects are used to modify visible characteristics of the elements on which they are placed. Modifiers in this category include:

- **Change Scene Modifier**: Changes from one scene to another within a single subsection. See "Change Scene Modifier" on page 12.20.
- **Color Table Modifier**: Manages color tables (i.e., custom color palettes). See "Color Table Modifier" on page 12.24.
- **Cursor Modifier**: Changes the mouse cursor. See "Cursor Modifier" on page 12.27.
- **Drag Motion Modifier**: Allows user element dragging. See "Drag Motion Modifier" on page 12.28.
- **Element Transition Modifier**: Activates an element transition (e.g., Fade). See "Element Transition Modifier" on page 12.29.
- **Gradient Modifier**: Creates color gradients in elements. See "Gradient Modifier" on page 12.32.
- **Graphic Modifier**: Modifies graphic properties of elements (e.g., color). See "Graphic Modifier" on page 12.33.
- **Image Effect Modifier**: Creates various image effects (e.g., inverts colors of an element). See "Image Effect Modifier" on page 12.37.
- **Navigation Modifier**: Changes from one scene to any other scene in a project. See "Navigation Modifier" on page 12.50.
- **Path Motion Modifier**: Controls the motion paths of elements. See "Path Motion Modifier" on page 12.56.
- **Return Modifier**: This modifier works in association with a change scene modifier with its "add to return list" option selected. The return modifier returns the scene to the last scene which was added to the return list. See "Return Modifier" on page 12.60.
- **Save and Restore Modifier**: Writes and reads values to an external file. See "Save and Restore Modifier" on page 12.61.
- **Scene Transition Modifier**: Specifies the type of transition that will occur when the scene changes (e.g., Zoom). See "Scene Transition Modifier" on page 12.63.
- **Set Value Modifier**: Changes the value of a variable or incoming data to a specified value. See "Set Value Modifier" on page 12.65.
- **Simple Motion Modifier**: Initiates a simple motion path (e.g., Down & Right). See "Simple Motion Modifier" on page 12.68.
- **Sound Effect Modifier**: Plays sound effects. See "Sound Effect Modifier" on page 12.70.
- **Sound Fade Modifier**: Decreases or increases the volume of a sound. See "Sound Fade Modifier" on page 12.71.

- **Sound Panning Modifier**: Sets the stereo position of a sound. See "Sound Panning Modifier" on page 12.73.
- **Text Style Modifier**: Modifies text styles (e.g., italics). See "Text Style Modifier" on page 12.76.
- **Track Control Modifier**: Activates and deactivates individual tracks in a QuickTime movie. See "Track Control Modifier" on page 12.80.
- **Vector Motion Modifier**: Initiates vector motion in degrees and inches per second. See "Vector Motion Modifier" on page 12.83.

Variable Modifiers

Variables store integers, strings and other values. When a variable is dropped onto a component and configured, the value contained within it can be referenced by messengers and sent with messages or commands, or referred to by name and used as variables in Miniscript programs.

Like any other modifier, variables can be given any name. For clarity, name variable modifiers to reflect their function. If the variable name will be referenced in a Miniscript modifier it is good programming practice to make the name a single word (or multiple words separated by underscore characters instead of spaces).

There are two classes of variables: simple and compound. Simple variables store a single value while compound variables store multiple values.

Variable modifiers include:

- **Boolean Variable**: Stores a true/false value. See "Boolean Variable" on page 12.17.
- **Compound Variable:** Use this modifier to create custom compound variables that can contain any combination of other variables.
- **Floating Point Variable**: Stores a floating point value (e.g., 3.14159). See "Floating Point Variable" on page 12.31.
- **Integer Variable**: Stores an integer value (e.g., 7). Possible values range from -32767 to 32767. See "Integer Variable" on page 12.39.
- **Integer Range Variable**: Stores an integer range value (e.g., 4…8). See "Integer Range Variable" on page 12.40.
- **List Variable**: Stores a list of values of any other data type. See "List Variable" on page 12.43.
- **Object Reference Variable:** Stores a reference to any object in a project. See "Object Reference Variable" on page 12.53.
- **Point Variable**: Stores a point value (e.g., (25,45)). See "Point Variable" on page 12.59.
- **String Variable**: Stores a string (e.g., "Bob Brown"). See "String Variable" on page 12.75.

- **Vector Variable**: Stores a vector value in degrees and inches per second (e.g., (33° 15.6)). See "Vector Variable" on page 12.82.

Variable Scopes

Where a variable modifier is placed defines its scope, that is the group of components that can access it. Put simply, a variable modifier is accessible to all descendants of its parent. For example, a variable modifier placed on a section is available to any modifier on the section, and all "descendants" of that section. (A parent's descendant includes any object that exists anywhere "under" it, no matter how many layers deep.)

A variable modifier whose parent is the project is called a *global variable*. Global variables can be accessed by any appropriate modifier in the entire project. See "Variable Scopes" on page 13.25. For a more localized scope, variables can be aliased and placed on specific objects anywhere in the project.

Messenger Modifiers

Messenger modifiers are used to conditionally send specific kinds of information to elements and modifiers.

Messenger modifiers include:

- **Boundary Detection Messenger**: Sends messages or commands after detecting collisions with its element's parent. See "Boundary Detection Messenger" on page 12.18.
- **Collision Messenger**: Sends messages or commands after detecting collisions with elements. See "Collision Messenger" on page 12.22.
- **If Messenger**: Sends messages and commands after a condition has been met. See "If Messenger" on page 12.35.
- **Keyboard Messenger**: Detects and responds to keystrokes. See "Keyboard Messenger" on page 12.41.
- **Messenger**: Sends messages and commands. See "Messenger" on page 12.47.
- **Timer Messenger**: Sends a message after a given time has elapsed. See "Timer Messenger" on page 12.78.

Behavior Modifier

A behavior is a special component in the mTropolis environment. It can be used to encapsulate (i.e., contain) groups of modifiers and other behaviors.

Behaviors can be used to group collections of modifiers that work in close concert. Each collection can be enabled or disabled with messages, creating "super modifiers" that provide more complex operations than single modifiers alone.

Like modifiers, behaviors are used by dragging and dropping them onto elements. Modifiers and other behaviors can then be dropped into them, arranged and configured for use.

The power of behaviors lies in the fact that they can be made "switchable". That is, they can be turned on or off with messages. When

a behavior is switched off, all of the modifiers it encloses are disabled. When a behavior is switched on, individual behaviors or modifiers within a behavior can then be activated by incoming messages. This feature allows the author to create and control components with very sophisticated behaviors and capabilities.

Once they have been configured, the capabilities of fully functioning behaviors can be given to any other component simply by copying and pasting the variable onto the component.

Behaviors can also be aliased and placed on multiple elements in a project. Since any change made to an alias is automatically made to all other aliases of the same object, aliasing a behavior allows the author to save significant authoring time while providing complete control over multiple elements that share the same capabilities. See "Alias Palette" on page 11.7.

Behaviors, like all other components in a project, can also be stored in libraries and saved for future use in any other project. See "Creating and Modifying mTropolis Libraries" on page 2.3.

A complete description of the behavior modifier dialog can be found in "Behavior" on page 12.13.

Miniscript Modifier

Miniscript is a simple scripting language embedded in a modifier. The Miniscript modifier allows you to use a scripting language to create customized modifiers that can:

- Get and set element attributes.
- Send messages and commands.
- Evaluate mathematical functions.
- Evaluate relational expressions and perform conditional branching.

A complete description of the Miniscript language can be found in Chapter 14, "Miniscript Modifier". A complete description of the Miniscript modifier dialog can be found in "Miniscript Modifier" on page 12.49.

ADDING MODIFIERS TO COMPONENTS

Modifiers can be added to components by dragging and dropping them from the modifier palettes.

Modifiers can be placed on objects while in any of mTropolis' three views: the layout window, the structure window or the layers window. The method for adding modifiers is the same in each case: select, drag and drop.

Once added to a component, modifiers can be customized by changing the settings in their dialogs.

To add a modifier to an element in the layout window:

- Click and drag a modifier icon from the modifier palette. A copy of the icon attaches to the cursor.
- Drop the modifier on the target element. The modifier icon appears in the top left corner of the element (or to the right of any icons already present on the element).

- Modifiers already present in a project can be moved to other components by dragging and dropping. They can be copied to other components by Option-dragging.

To remove a modifier from an element:

- Click on the modifier icon to be removed. Press the Delete key. The modifier icon, and its associated functionality, is removed from the element.
- *Note: Modifiers can also be cut, copied, pasted or duplicated in any of mTropolis' views.*

To customize a modifier:

- Double-click a modifier icon attached to an element. The modifier's dialog appears. See "Configuring Modifier Dialogs" on page 12.9 for descriptions of common modifier dialog controls. Complete descriptions of each modifier dialog can be found at elsewhere in this chapter.

CONFIGURING MODIFIER DIALOGS

The rest of this chapter contains detailed descriptions of the modifier dialogs. This section describes features shared by all of the modifier dialogs.

Each of the modifiers in mTropolis has settings that can be edited through its modifier dialog. Display a modifier's configuration dialog by double-clicking on a modifier icon attached to a component.

Common Modifier Dialog Controls

Figure 12.5 shows a typical modifier dialog, the Drag Motion dialog. All the modifier dialogs are similar and share a number of common fields. These fields are described below. Complete documentation for specific modifiers is found later in this chapter.

Modifier Icon

The icon associated with the modifier appears in the upper left corner of the dialog. This icon can be used to identify the type of modifier dialog being displayed if the modifier's default name has been changed.

Name Field

This editable text field shows the name of the modifier. When first created, modifiers have a default name that is just the modifier type (e.g., "Graphic Modifier", "Messenger").

Enter a new name for the modifier here. It is good practice to use descriptive names, such as "Blue on mouse down." Descriptive naming also makes it easy to find the modifier while in the structure window. Variable modifiers that will be referenced by Miniscript, are easiest to use if they are given a single word name.

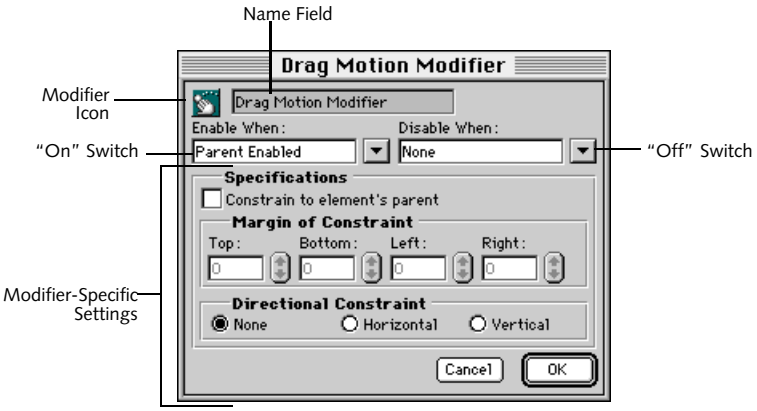


Figure 12.5 A typical modifier dialog, the Drag Motion dialog

Apply (or Execute, Enable) When Pop-Up
This pop-up menu displays the message that, when received by the modifier during runtime, causes it to apply, execute, or enable its effect. In other words, this is the message that acts as an "On" switch for the modifier.

Click the down arrow button to the right of this field to display menu options. Select an item from the list, or simply type in an option and mTropolis will attempt to match it to an item in the menu. If an item cannot be matched, an alert appears, asking if you wish to create a new author message.

For a detailed description of items in this pop-up, see "When Pop-Up and Message/ Command Pop-Up Options" on page 13.5.

Remove (or Terminate, Disable) When Pop-Up
This pop-up menu displays the message that, when received by the modifier during runtime, causes it to remove, terminate, or disable its effect. In other words, this is the message that acts as an "Off" switch for the modifier.

Click the down arrow button to the right of this field to display menu options or simply type in an option and mTropolis will attempt to match it to an item on the menu. If an item cannot be matched, an alert appears, asking you if you wish to create a new author message.

For a detailed description of items in this pop-up, see "When Pop-Up and Message/ Command Pop-Up Options" on page 13.5.

Modifier-Specific Settings
Many modifier dialogs have settings that can be used to specify the particular action or effect to be applied.

The options for some modifiers are affected by their location in the project's hierarchy. For example, the variables available to a messenger are dependent upon where the variable modifier is placed in relation to the messenger. See "Variable Scopes" on page 13.25.

CONFIGURING MESSENGER MODIFIERS

Some modifiers are used to send specific kinds of information to other elements and modifiers. Modifiers in this group are called messengers. Although each has unique functionality, there are some similarities among them. Figure 12.6 shows a typical messenger dialog.

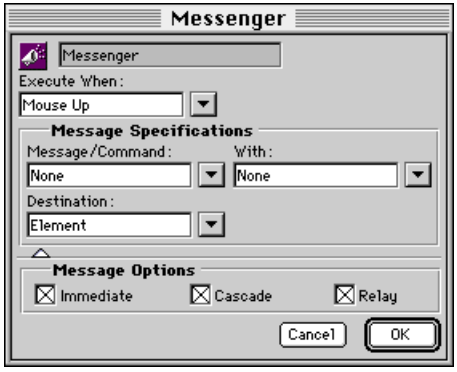


Figure 12.6 A typical messenger dialog. The "Message Specifications" and "Message Options" fields are common to all messenger dialogs.

All messenger dialogs share the same pop-ups in their Message Specifications section:

Message/Command Pop-Up Menu

Use this pop-up to select the message or command to be sent when the messenger is activated. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

With Pop-Up Menu

Use this menu to select a value to be sent with the message or command. The choices on this pop-up are None, Incoming Data and any variables to which the modifier has access. These menu items are described in more detail below:

- **None**: This is the default selection. No value is sent with the message.
- **Incoming Data**: Choose this option to configure the messenger to send the incoming data (i.e., the value that arrives with the message that has been configured to trigger the messenger), with the outgoing message or command.
- **Variables**: To send the value of a variable modifier with the outgoing message, select its name from the submenus available in the second section of the With pop-up menu. All variable modifiers currently available to a messenger from its position in the project's hierarchy are shown. These variable modifiers are only accessible to those modifiers or Miniscripts in its "scope."
- **Constant Data Value**: To send a constant data value, highlight the With text field and type the value to be sent. The value should be entered with the same syntax as if it were being used in a Miniscript modifier. The syntax of the expression entered is checked when OK is clicked. Any appropriate mTropolis data type can be sent.
- **Toon Ranges**: If the current scene contains an mToon that has named ranges defined, those ranges can be selected from the "Toon Ranges" submenu.

Destination Pop-Up Menu

Use this menu to select the destination for the message or command sent from this modifier. For a complete discussion of this menu, see"The Destination Pop-Up Menu" on page 13.21.

Message Options

All messenger dialogs have a Message Options section that can be revealed by clicking the triangle at the bottom of the dialog. Options in this section are:

Cascade Checkbox

By default, the message "cascades" from the targeted destination to all components in the hierarchy below it. Uncheck this option to configure the messenger to send its message only to the modifiers in a targeted component. "Cascade off" is equivalent to the path of an environment message sent by mTropolis.

- *Note: This option has no effect when sending a command. Commands act only on the targeted element.*

Immediate Checkbox
By default, messages or commands are sent immediately when the messenger is activated. Uncheck this box to configure the messenger to send its message or command only after the current thread of messages has ended. This option can be used to avoid system overload if the message thread queue becomes too deep.

Relay Checkbox
By default, messages travel from component to component in the hierarchy activating any and all modifiers that respond to the message. Uncheck this box to activate only the *first* modifier in the path of the message that is configured to respond. In effect, the first modifier to respond to the message "swallows" the message.

- *Note: This option has no effect when sending a command. Commands act only on the targeted element.*

BEHAVIOR

A behavior is a special component in the mTropolis environment. It is used to encapsulate (i.e., contain) other modifiers and behaviors.

Behaviors are used to hierarchically group collections of modifiers that work in close concert. Each collection can be enabled or disabled with messages, creating "super modifiers" that provide more complex operations than single modifiers alone.

Like modifiers, behaviors are used by dragging and dropping them onto components. Modifiers and other behaviors can then be dropped into them, and arranged and configured for use. The power of behaviors lies in the fact that they can be switchable, that is, they can be turned off or on with messages.

When a behavior is switched off, all of the modifiers and behaviors it encloses cannot be activated by incoming messages. When a behavior is switched on, individual behaviors or modifiers within a behavior can be activated by incoming messages. This feature allows the author to create and control components with highly sophisticated capabilities.

Once they have been configured, the capabilities of fully functioning behaviors can be given to any other component, simply by copying and pasting the behavior onto another component.

Behaviors can also be aliased and placed on other elements in a project. Since any change made to an alias is automatically made to all other aliases of the same object, aliasing a behavior allows the author to save significant authoring time while providing complete control over multiple elements that share the same capabilities. See "Alias Palette" on page 11.7.

Behaviors, like all other components in a project, can also be stored in libraries and saved for future use in any other project. See "Creating and Modifying mTropolis Libraries" on page 2.3.

- *Note: Behaviors are often used simply as a tool to organize an element's modifiers into groups.*

Behaviors and Components

Behaviors, and all the modifiers they contain, are always associated with a single structural component.

To illustrate, in the figure above, Element 1 is the parent of Behavior 1 and Messenger 1, and Behavior 1 is the parent of Behavior 2 and Messenger 2. Both messengers and behaviors are, however, associated with Element 1, which is the first structural component at the root of the modifier hierarchy.

This association is important to remember when targeting the recipient of a message from messengers within behaviors.

For example, in the figure above, when "Element's Parent" is the destination chosen for the message sent from Messenger 1, it will send its message to its associated element's parent, Scene A. This is because Element 1 is the first element at the root of the modifier hierarchy, and Scene A is this element's parent.

Using Behaviors

To create a new behavior:

- Drag a behavior modifier from the modifier palette (or a library) to an element.
- Open the behavior by double-clicking it.
- Add modifiers to the behavior by dragging them into the behavior window.
- Optionally, add modifiers to a behavior simply by dragging and dropping the modifier icons onto a closed behavior icon. When the behavior is first opened, modifiers in the behavior window can be repositioned by dragging them.

Message Passing within Behaviors

As long as a behavior is switched "on," all messages are passed to all modifiers and behaviors inside the behavior. Any modifier or behavior configured to respond to these messages will be executed. For example, if Parent Enabled were sent to the behavior depicted in Figure 12.7, the Transparent Ink and Pause modifiers would be executed but the other modifiers would not.

Messages are passed to modifiers and behaviors within a behavior according to their message passing order. This order is visible in the structure window and inside the behavior window. For more information on changing the execution order of components in a behavior, see "Messaging Order Numbers of Modifiers" on page 12.15.

Components of the Behavior Dialog

Controls in the behavior dialog are described below.

Modifier's Name Field

This editable text field can be used to change the behavior's name.

Modifier Icon

This icon identifies the modifier's type.

Switchable Checkbox

When selected, this checkbox allows the selection of enable and/or disable messages for the behavior. Select this box to create behaviors whose component modifiers are active only after the receipt of the specified "Enable When" message and are deactivated after the receipt of the specified "Disable When" message. Note that non-switchable behaviors (i.e., behaviors that do not have this checkbox checked) are always enabled.

Enable When Pop-Up Menu

This pop-up menu can be used to select the message that enables this behavior. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5. This menu is only available when the Switchable checkbox is checked.

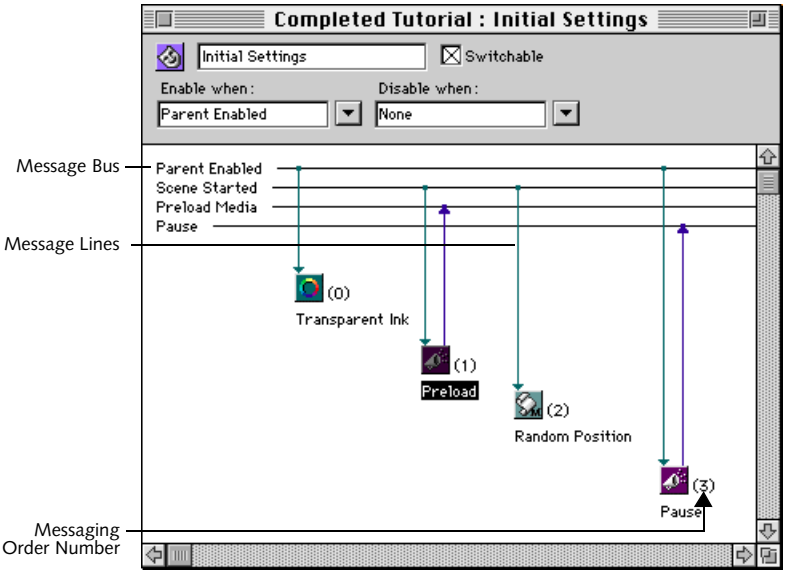


Figure 12.7 Behavior modifier dialog

- *Note: With the exception of the Parent Enabled message, any message that is used to enable a switchable behavior is not broadcast to the components it contains. To enable a behavior and activate the modifiers it contains on a single message, use Parent Enabled. This message is sent by mTropolis after a scene starts.*

Disable When Pop-Up Menu

This pop-up menu can be used to select the message that disables this behavior. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5. This menu is only available when the Switchable checkbox is checked.

Names of Elements

The name of a component modifier, shown below its icon, can be changed in the behavior window by clicking on the name and entering a new one.

Messaging Order Numbers of Modifiers

These numbers, shown in brackets to the right of a component modifier's icon, denote the messaging order of modifiers within a behavior. For example, in the behavior shown in Figure 12.7. the Transparent Ink and Pause modifiers act on the same message, but Transparent Ink has a lower messaging order number than Pause and will be executed first.

The messaging order of components can be changed by clicking on the Messaging Order

number. Hold down the mouse until a pop-up appears. Select Do Sooner or Do Later to move the modifier one position forward or backward in the messaging order.

Alternatively, the messaging order of modifiers and behaviors in a behavior can be changed by dragging and dropping them into new positions in the behavior from the *structure* window.

Message Bus
All messages acted on and sent by modifiers in a behavior are shown in this list. The horizontal lines extending from the messages in this list represent a message "bus" that the modifiers are "listening" to. Note that this display is not a timeline. Click on a message name to highlight its bus so that the path of the message in the behavior can be seen more easily.

Message Lines
These lines show the paths of messages to and from modifiers within the behavior. These lines connect to the message bus to show which modifiers receive and send certain messages.

On color monitors vertical message lines to and from modifiers and messages are displayed in three different colors:

- Green message lines are drawn *from* the message bus *to* a modifier, indicating the message that execute, applies, or enables the modifier.
- Blue message lines are drawn *from* a modifier or behavior *to* the message bus, indicating that the modifier sends this message when activated.
- Purple message lines are drawn *from* the message bus *to* the modifier to indicate that the message terminates, removes, or disables. These lines are also shown with an "open" (i.e., unfilled) arrow head.

For example, in Figure 12.7, the "Preload" messenger is activated by a *Scene Started* message. Upon activation, it sends a *Preload Media* message.

BOOLEAN VARIABLE

The boolean variable modifier stores a boolean (i.e., true/false) value. It is useful for keeping track of objects or properties that have only two states. For example, whether a user has touched a certain object could be stored in a boolean variable modifier; depending on the value of that variable modifier, the user may or may not be able to do something else in the project.

Components of the boolean variable dialog are described below:

Variable's Name Field
This editable text field can be used to change the variable's name. If the variable will be referenced in a Miniscript modifier, it is good programming practice to make the name a single word.

Modifier Icon
This icon identifies the variable modifier's type.

Value Buttons
Enter the variable's initial value here by selecting either the True or False radio button. The default is False. The value can be changed during runtime. See "Setting Values of Variable Modifiers" on page 14.5 for details regarding getting and setting the value of this variable modifier.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

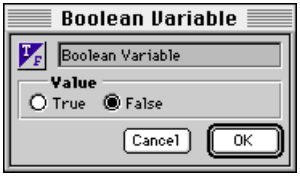


Figure 12.8 The Boolean Variable dialog

BOUNDARY DETECTION MESSENGER

The boundary detection messenger detects the relative position of elements. It can be configured to send a message when an object has left or touched the frame of its parent. For information regarding parent/child relationships, see "Parent/Child Tool" on page 11.3.

Components of the boundary detection messenger dialog (Figure 12.9) are described below:

Modifier's Name Field

This editable text field can be used to change the modifier's name.

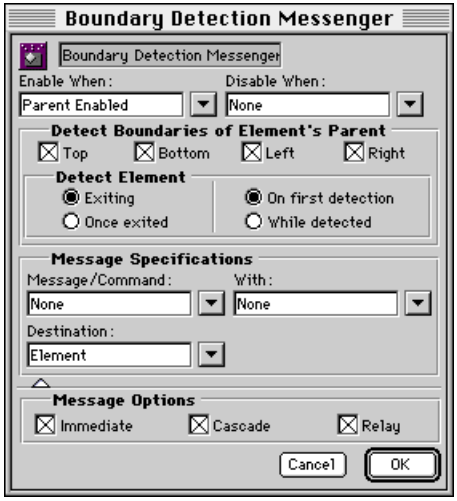


Figure 12.9 The Boundary Detection messenger dialog.

Modifier Icon

This icon identifies the modifier's type.

Enable When Pop-Up Menu

Use this pop-up menu to select the message that enables boundary detection. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Disable When Pop-Up Menu

Use this pop-up menu to select an optional message that causes boundary detection to be disabled. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Detect Boundaries of Element's Parent Checkboxes

These four checkboxes (Top, Bottom, Left, and Right) specify the boundaries of the parent element that generate messages when the child touches them. By default, all boundaries are sensitive.

Detect Element Section

The radio buttons in this section control how the child element's boundary crossing is detected:

- **Exiting**: Choose Exiting to send the specified message or command when the element is exiting its parent's boundary (i.e., there is still contact between the child and the boundary).
- **Once Exited**: Choose Once Exited to send the messenger's specified message or command when the element has completely left its parent's frame.

- **On First Detection**: Choose On First Detection to send the specified message or command when the edge of the element first touches the frame of its parent (if "Exiting" is checked) or when the element first exits (if "Once exited" is checked).
- **While Detected**: Choose While Detected to send the specified message or command each time the child moves and is touching (if "Exiting" is checked) or outside of (if "Once exited is checked) the parent's frame.

Message Specifications

Use this section to specify the message to be sent when a boundary contact is detected.

See "Configuring Messenger Modifiers" on page 12.10 or "Message Specifications" on page 12.47 for a complete description of the controls in this section.

Message Options

Click the triangle at the bottom of the dialog to display the Message Options section. See "Message Options" on page 12.11 or "Message Options" on page 12.48 for a complete description of the controls in this section.

Cancel Button

Click Cancel to ignore changes made to this modifier.

OK Button

Click OK to accept changes made to this modifier.

CHANGE SCENE MODIFIER

The change scene modifier executes a scene change when triggered in runtime mode. The previous scene, next scene, or a specific scene in the project can be selected. Note that the order of scenes in a subsection can be easily changed in the layers window—see "Changing the Order of Scenes" on page 10.4. A more complex and flexible modifier for changing scenes is described in "Navigation Modifier" on page 12.50

Components of the change scene modifier dialog (Figure 12.10) are described below:

Modifier's Name Field

This editable text field can be used to change the modifier's name.

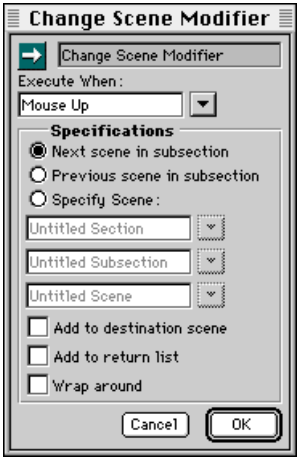


Figure 12.10 The Change Scene modifier dialog

Modifier Icon

This icon identifies the modifier's type.

Execute When Pop-Up Menu

Use this pop-up menu to select the message that triggers the scene change. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Specifications Section

This section of the dialog can be used to select the scene to display when the change scene modifier is triggered.

Specifications Radio Buttons

Select the type of scene:

- **Next Scene in Subsection**: Choose this option to change to the next scene in the current subsection.
- **Previous Scene in Subsection**: Choose this option to change to the previous scene in the current subsection.
- **Specify Scene**: Choose this option to activate the section, subsection and scene pop-ups. Use the pop-ups to select a specific scene anywhere in the project.

Specifications Check Boxes

Select scene change options:

- **Add to Destination Scene**: Select this checkbox to perform a special scene change in which the currently displayed scene is still visible after the change to the new scene. That is, the current scene acts like a shared scene for this scene change.

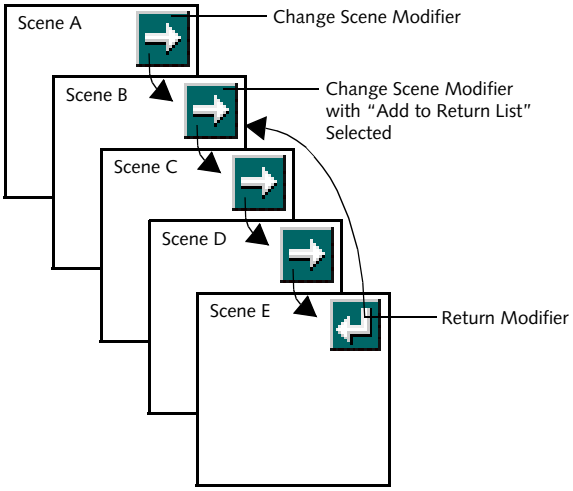


Figure 12.11 The Return List

- **Add to Return List**: Select this checkbox to make the current scene the one returned to by the next return modifier in a series of scenes. For example, in Figure 12.11, change scene modifiers have been placed on each of the scenes. Each of these modifiers has been configured to change to the next scene. Scene B's scene change modifier has the Add to Return List option selected. A return modifier has been placed on Scene E. When this modifier receives a message that will activate it during runtime, Scene E will change to Scene B (the arrows shows the sequence of scenes).
- **Wrap Around**: By default, the first and last scene in a subsection have no connection in the scene order. That is, the last scene in a subsection has no next scene and the first scene in a subsection has no previous scene.

  When this checkbox is selected, a change to the "Next Scene", executed from the last scene in a subsection, "wraps around" the list of scenes and changes to the first scene in the subsection.

  Similarly, a change to the "Previous Scene", executed from the first scene in a subsection, "wraps around" the list of scenes and changes to the last scene in the subsection.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

COLLISION MESSENGER

The collision messenger is used to detect when elements are in the same space. The collision messenger sends a message when its element touches another element.

Components of the collision messenger dialog (Figure 12.12) are described below.

Modifier's Name Field
This editable text field can be used to change the modifier's name.

Modifier Icon
This icon identifies the modifier's type.

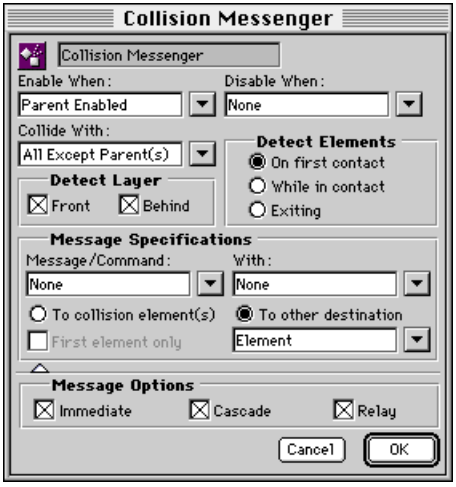


Figure 12.12 The Collision Messenger dialog

Enable When Pop-Up Menu
Use this pop-up menu to select the message that enables collision detection. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Disable When Pop-Up Menu
Use this pop-up menu to select an optional message that causes collision detection to be disabled. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Collide With Pop-Up Menu
Select the type of elements whose collision will trigger a message. There are two options.

- **Any Element**: Choose this option to send a message upon collision with any element.
- **All Except Parent(s)**: Choose this option to send a message upon collision with any element except the messenger's parent.

Detect Layer Checkboxes
By default, the collision messages are generated for collisions with elements in any layer.

- **Front**: Deselect the Front checkbox to disable collisions with objects with a higher layer order than the element (i.e., those objects that are "in front" of the element).
- **Behind**: Deselect the Behind checkbox to disable collisions with objects with a lower layer order than the element (i.e., those objects that are "behind" the element).

- *Note: If both boxes are deselected, messages will never be generated, since each object has a unique layer order number.*

Detect Elements Section
The radio buttons in this section control when messages are sent during the collision process:

- **On First Contact**: Choose this option to send the specified message or command when an object enters or makes contact with the element's frame.
- **While in Contact**: Choose this option to send the message or command repeatedly while a moving object is in contact with the element.
- **Exiting**: Choose this option to send the message or command when a colliding object leaves the element's frame.

Message Specifications Section

Use this section to specify the message to be sent when a boundary collision is detected. See "Configuring Messenger Modifiers" on page 12.10 or "Message Specifications" on page 12.47 for a description of the common controls in this section.

The collision detection dialog has a number of unique message specifications that control the destination of messages sent by the collision modifier:

To Collision Element(s) Radio Button
When this option is selected, messages are sent only to the objects that collide with the element.

First Element Only Checkbox
This option is only available when the "To collision element(s)" radio button is selected. When the First Element Only checkbox is selected, a message is sent only to the first object that collides with the modified element.

To Other Destination Radio Button
Select this option to send collision-generated messages to other destinations. A destination pop-up is activated. The selected destination is the only recipient of messages generated by collisions with the element.

Message Options
Click the triangle at the bottom of the dialog to display the Message Options section. See "Message Options" on page 12.11 or "Message Options" on page 12.48 for a complete description of the controls in this section.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

COLOR TABLE MODIFIER

Many projects are designed using an 8-bit color table other than the standard system palette. The color table modifier allows you to define which color table is to be active at any one time.

To use a custom palette for a project, follow these steps:

- Create your image(s) and produce a color palette for all the images to be dithered. Dither the image(s) to the palette(s).
- Export a custom palette(s) from Adobe Photoshop (click the Save button in the Color Table dialog displayed from the Mode menu) or similar application.

The resulting file can be linked to the asset palette. Use the **Link Media-File** option of the File menu.

Components of the color table modifier dialog (Figure 12.13) are described below:

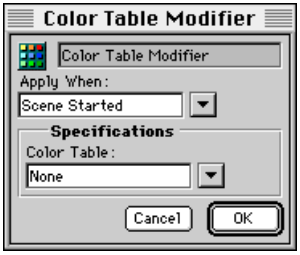


Figure 12.13 The Color Table modifier

Modifier's Name Field
This editable text field can be used to change the modifier's name.

Modifier Icon
This icon identifies the modifier's type.

Apply When Pop-Up Menu
Use this pop-up menu to select the message that applies the color table modifier. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Color Table Pop-Up Menu
Use this pop-up to select the name of a color table file that has been linked to the current project. A new color table can be linked by choosing Link Color Table. A standard file dialog appears.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

COMPOUND VARIABLE

The compound variable modifier can be used to organize a number of other mTropolis variables into a single, user-defined compound variable. The fields of custom compound variables created with this modifier can be accessed from Miniscript in the same way as mTropolis' built-in compound variables. See "Variables" on page 14.3 and "Accessing the Fields of Compound Variables" on page 14.3.

To use the compound variable modifier, place it on a component, then drag variables onto its icon. The variables will seem to "disappear" into the compound variable. To view the contents of the compound variable, switch to the structure view and toggle the compound variable's open/close triangle to reveal the variables that have been placed inside.

The name given to the compound variable is its "top-level" name. The names of the individual variables put inside the compound variable become the names of the variable's fields. See the example below.

Note that compound variables can be nested inside other compound variables to create complex data hierarchies. Like any other variable, the compound variable modifier can also be sent or received as data by any messenger in its scope.

Hot Tip

Use this variable in conjunction with list variables (see "List Variable" on page 12.43) to create various types of multidimensional arrays.

Components of the compound variable dialog are described below.

Variable's Name Field

This editable text field can be used to change the variable's name. This name becomes the "top-level" name of the compound variable. Any variables dropped into the compound variable become "fields" of the custom compound variable. See the example below.

Modifier Icon

This icon identifies the modifier's type.

Cancel Button

Click Cancel to ignore changes made to this modifier.

OK Button

Click OK to accept changes made to this modifier.

Example

Consider the structure view of the compound variable shown in Figure 12.15. The compound variable has been given the name "client". There are two variables that have been placed inside the compound variable—a string variable and an integer variable. The names of these variables, "name" and "age",

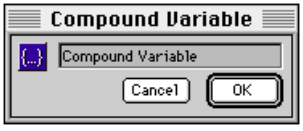


Figure 12.14 The Compound Variable dialog

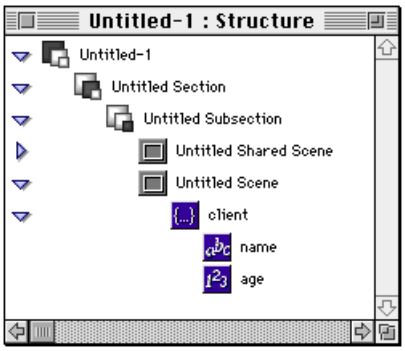


Figure 12.15 Structure view of example compound variable

become the names of the compound variable's fields.

Values in the compound variable can be accessed through Miniscript statements. For example, the entire compound value can be accessed by using its "top-level" name in a Miniscript statement:

```
send "NewClient" with client
```

The values of individual variables within the compound variable can be accessed just like the fields of "built-in" compound variables using the "." syntax. In our example, the Miniscript expression `client.name` resolves to the string value contained within the "name" string variable. Similarly, the expression `client.age` resolves to the integer value contained in the "age" integer variable.

These values could be used with any type of Miniscript statement. The following examples illustrate just a few of the possibilities:

```
-- Set the value of a variable from
-- one of the fields:
set mytextvar to client.name

-- Change the value of a field:
set client.age to 27

-- Perform a comparison using a field:
if client.age < 21 then \
   send "No Cocktails for You"

-- Set the value of an attribute
-- from a field:
set mytextelement.text to \
   "Good job, " & client.name
```

Miniscript statements are described in Chapter 14, "Miniscript Modifier".

CURSOR MODIFIER

•Note: There are two versions of this modifier provided with mTropolis. The version found on Modifier Palette Group 3 is a newer, experimental version. Only the Group 3 Cursor Modifier, with its enhanced functionality, is described below. Both versions of this modifier are Macintosh-only—they have no effect when used under Windows.

The cursor modifier changes the active mouse cursor. Its effects are only visible during runtime.

Components of the cursor modifier dialog (Figure 12.16) are described below:

Modifier's Name Field
This editable text field can be used to change the modifier's name.

Modifier Icon
This icon identifies the modifier's type.

Apply When Pop-Up Menu
Use this pop-up menu to select the message that causes the cursor to change. The default is Mouse Over, which makes the cursor change when it is moved over the element. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

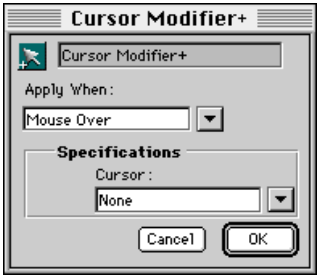


Figure 12.16 The Cursor Modifier dialog

Cursor Pop-Up Menu
Use this pop-up menu to select a cursor to be changed to when the specified "Apply When" message is received. This list contains many standard cursor options.

Using Custom Cursors with the Group 3 Cursor Modifier
The "enhanced" version of this modifier can be used with "custom" cursors as described below:

- **For a Macintosh title:** Save your project file and exit mTropolis. Use a Macintosh resource editor (such as ResEdit) to add a new crsr or CURS resource to the saved project file making sure that you give the cursor a unique name. When your project is next opened in mTropolis, the Cursor pop-up menu will include the name of your custom cursor. Custom cursors can be used just like the built-in cursors.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

DRAG MOTION MODIFIER

The drag motion modifier allows an element to be dragged around the screen by the user during runtime.

Components of the drag motion modifier dialog are described below:

Modifier's Name Field
This editable text field can be used to change the modifier's name.

Modifier Icon
This icon identifies the modifier's type.

Enable When Pop-Up Menu
Use this pop-up menu to select the message that enables dragging of the element. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Disable When Pop-Up Menu
Use this pop-up menu to select the message that disables dragging of the element. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

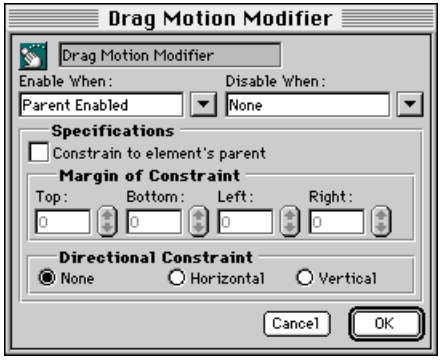


Figure 12.17 The Drag Motion modifier dialog

Specifications Section

Use the controls in this section of the dialog to set drag motion options.

Constrain to Element's Parent Checkbox
Check this box to make the object draggable only within the limits of its parent's frame.

Margin of Constraint Fields
These options are available when "Constrain to Element's Parent" is selected. They set a "margin" inside the frame of the element's parent that constrains drag motion even more. Margins can be set for the Top, Bottom, Left, and Right sides of the parent frame. Margins are measured in pixels.

Directional Constraint Radio Buttons
Use these options to constrain drag motion to a single direction. Options are:

- **None**: The default. Select this option to allow dragging in any direction.
- **Horizontal**: Select this option to allow horizontal dragging only.
- **Vertical**: Select this option to allow vertical dragging only.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

ELEMENT TRANSITION MODIFIER

The element transition modifier can be used to creatively hide or reveal an element. Like the scene transition modifier, the transition pop-up lists optional effects, such as fade and rectangular iris.

- *Note: The element transition modifier is currently supported only on the Macintosh. This modifier will have no effect in a title built for the Windows platform.*

Components of the element transition modifier dialog are described below:

Modifier's Name Field
This editable text field can be used to change the modifier's name.

Modifier Icon
This icon identifies the modifier's type.

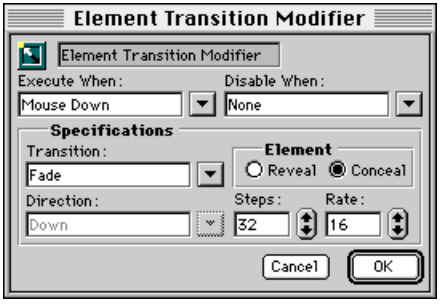


Figure 12.18 The Element Transition modifier dialog

Execute When Pop-Up Menu
Use this pop-up menu to select the message that enables the element transition. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Disable When Pop-Up Menu
Use this pop-up menu to select the message that disables the element transition. Note that a transition cannot be cancelled while it is being played.

Specifications Section

Use the controls in this section to select the type of element transition.

Transition Pop-Up Menu
Select a transition type from this menu. Options include:

- Fade
- Rectangular Iris
- Zoom
- Oval Iris

Element Radio Buttons
The transition can reveal a hidden element or hide a visible element:

- **Reveal Element**: Select this option to reveal a currently hidden element. If this option is selected for a visible element, the element will be hidden at the start of the transition.
- **Conceal Element**: Select this option to hide a currently visible element.

Direction Pop-Up Menu
If the transition chosen has an associated direction, this pop-up becomes visible. Use this menu to select a direction for the transition.

Steps Field
Use this field to set the number of steps between the start and finish of the transition. The greater this number, the finer (and slower) the transition.

Rate Field
Use this field to select the speed of the transition in steps per second. At a rate of 60, a transition set to 60 steps would take 1 second.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

FLOATING POINT VARIABLE

Floating-point variables store floating-point values. Floating-point values have a range of 10^9-1 and accuracy to 8 places. Floating-point values are useful in mathematical computations.

Components of the floating-point variable dialog are described below:

Variable's Name Field
This editable text field can be used to change the variable's name. If the variable will be referenced in a Miniscript modifier, it is good programming practice to make the name a single word.

Modifier Icon
This icon identifies the variable modifier's type.

Value Field
Enter the variable's initial value here or use the up/down arrow buttons to set the value by 0.5 increments. The value is checked for validity when the OK button is clicked. The value can be changed during runtime. See "Setting Values of Variable Modifiers" on page 14.5 for details regarding getting and setting the value of this variable modifier.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

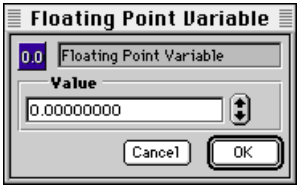


Figure 12.19 The Floating Point variable dialog

GRADIENT MODIFIER

The gradient modifier produces a color gradient between two colors on any graphic element that has not been linked to an external file (i.e., an "empty" graphic element).

- *Note: The gradient modifier is currently supported only on the Macintosh (as indicated by the yellow dot in the icon's upper left corner). This modifier will have no effect in a title built for the Windows platform.*

Components of the gradient modifier dialog (Figure 12.20) are described below:

Modifier's Name Field
This editable text field can be used to change the modifier's name.

Modifier Icon
This icon identifies the modifier's type.

Apply When Pop-Up Menu
Use this pop-up menu to select the message that applies the gradient modifier. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

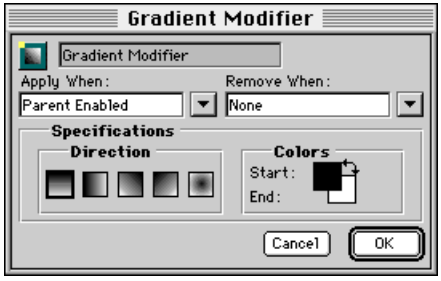


Figure 12.20 The Gradient Modifier dialog

Remove When Pop-Up Menu
Use this pop-up menu to select the message that removes the gradient modifier. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Direction Buttons
Click one of the sample gradient buttons to select a direction/style for the gradient.

Start and End Color Boxes
These boxes display the start and end colors of the gradient. Click and hold on a square, and drag to a color on the palette that appears. To reverse the direction of the gradient, click on the arrow that points to both color squares.

- *Note: Regardless of the bit depth of your project, gradients produced with this modifier are always 8-bit gradients. Some gradients, especially those between very different colors, may look "banded" or dithered.*

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

GRAPHIC MODIFIER

The graphic modifier applies basic shapes, color, ink effects, borders and shadows to graphic elements. It can also be used to define the matte color and borders of elements. Elements can have multiple graphic modifiers, but the effects of only one graphic modifier are active at once.

Note that some of the effects made possible by this modifier can also be selected using tools on the tool palette. See "Ink Effects" on page 11.4, and "Foreground and Background Colors" on page 11.6.

Components of the graphic modifier dialog (Figure 12.21) are described below:

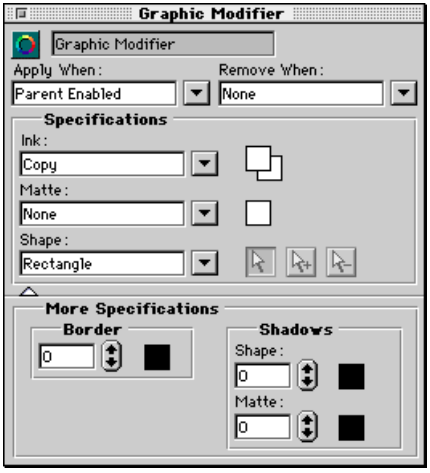


Figure 12.21 The Graphic Modifier Dialog

Modifier's Name Field

This editable text field can be used to change the modifier's name.

Modifier Icon

This icon identifies the modifier's type.

Apply When Pop-Up Menu

Use this pop-up menu to select the message that applies the graphic modifier. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Remove When Pop-Up Menu

Use this pop-up menu to select the message that removes the graphic modifier. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Specifications Section

Use the controls in this section to select the type of graphic effect to apply.

Ink Pop-Up Menu

The ink effect changes the way a graphic element is displayed on the screen. Different inks have varying effects, depending on the color(s) of the object itself, and the object beneath it. Options in this pop-up are the same as those described in "Ink Effects" on page 11.4. The foreground/background color boxes to the right of this menu work the same way as those described in "Foreground and Background Colors" on page 11.6.

Matte Pop-Up Menu

Only applicable to graphic elements with a solid background color, the matte function

makes the specified color transparent. This effect is often used in conjunction with the image shadow option. To select a custom color, choose Custom from the pop-up. Click and hold on the color box to the right of the Matte menu. An eyedropper tool appears. Use the eyedropper tool to select a specific color from a graphic to use as the matte color.

Shape Pop-Up Menu

Use this menu to create or edit the "hot" region of the object. Areas not enclosed in the shape do not respond to mouse messages. Available shapes are:

- **Rectangle**: The default shape. The graphic completely fills its rectangular frame.
- **Round Rectangle**: The graphic takes a rectangular shape with rounded corners.
- **Oval**: The graphic takes an oval shape. If the frame of the graphic is square, the shape looks like a perfect circle.
- **Star**: The graphic takes a five-pointed star shape.
- **Polygon**: The Polygon option allows an author-definable irregular shape around an object within the element. To define the polygon region, use the tools to the right of the Shape pop-up. If the element is all or mostly colored black, change the color while defining the polygon region.

Polygon Shape Tools

These tools are active when Polygon is selected in the Shape pop-up:

Point Selection Tool: Use this tool to drag a polygon point to a new location.

Add Point Tool: Use this tool to add a new point to the polygon. Click on the polygon outline to add a new point.

Delete Point Tool: Use this tool to delete a point on the polygon. Click on a polygon point with this tool to delete the point.

More Specifications Section

Click the white triangle at the bottom of the graphic modifier dialog to toggle the display of these options:

Border Field

Use this field to specify the size, in pixels, of a border around the defined shape of the element. Use the color box to the right of this field to select a color for the border.

Shadows Shape Field

This option adds a shadow effect to the object of the color chosen in the box to the right.

Shadows Matte Field

This option adds a shadow effect to the object relative to the matte function, using the color chosen in the box to the right. The matte function must be enabled for this option to take effect. This option will add a shadow effect to the object of the color chosen in the box to the right.

IF MESSENGER

The "if" messenger is very similar to the standard messenger modifier, except that it only sends its message or command if a specified condition is true.

The messenger evaluates a boolean expression, entered in Miniscript syntax in the dialog's "If" scrolling text field. This expression is evaluated when the messenger receives its "Execute When" message. If the expression evaluates to "true", the message or command specified in the Message Specifications section is sent. Otherwise, no message is sent.

Components of the "if" messenger dialog (Figure 12.22) are described below:

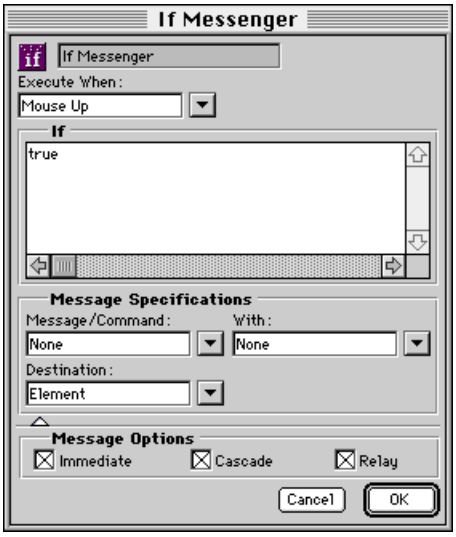


Figure 12.22 The If Messenger dialog

Modifier's Name Field
This editable text field can be used to change the modifier's name.

Modifier Icon
This icon identifies the modifier's type.

Execute When Pop-Up Menu
Use this pop-up menu to select the message that causes the "If" expression to be evaluated. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

If Expression Scrolling Text Field
Enter a boolean expression in this field; use standard boolean and relational operators, reserved variables, functions and/or author-defined variables. This expression must be in proper Miniscript syntax. See "Miniscript Modifier" on page 14.1. The syntax of the expression is checked when OK is clicked.

If the expression evaluates to true (see "Definition of True" on page 14.10), the message specified in the Message Specifications section is sent. Some sample 'if' expressions follow:

- The following expression evaluates to true if the user-defined variable `cel` is equal to 5:

  ```
  cel=5
  ```

- The following expression evaluates to true if the user-defined boolean variable `userCanDo` contains true:

  ```
  userCanDo
  ```

- The following expression evaluates to true if the user-defined string variable `user-Name` contains the string "super":

```
userName = "super"
```

Message Specifications
Use this section to specify the message to be sent if the expression in the "If" scrolling text field evaluates to true.

See "Configuring Messenger Modifiers" on page 12.10 or "Message Specifications" on page 12.47 for a complete description of the controls in this section.

Message Options
Click the triangle at the bottom of the dialog to display the Message Options section. See "Message Options" on page 12.11 or "Message Options" on page 12.48 for a complete description of the controls in this section.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

IMAGE EFFECT MODIFIER

The image effect modifier can apply one of a number of predefined graphic effects to an element. These effects are particularly useful when creating elements that represent clickable buttons.

Components of the image effect modifier dialog are described below:

Modifier's Name Field

This editable text field can be used to change the modifier's name.

Modifier Icon

This icon identifies the modifier's type.

Apply When Pop-Up Menu

Use this pop-up menu to select the message that applies the effect. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

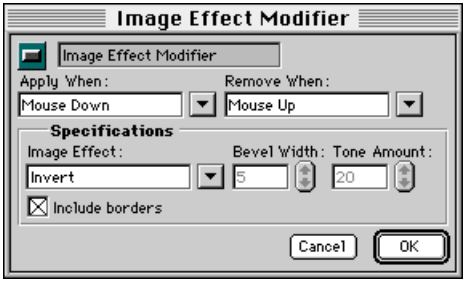


Figure 12.23 The Image Effect modifier dialog

Remove When Pop-Up Menu

Use this pop-up menu to select the message that removes the effect. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Specifications Section

Use the controls in this section to select the effect to be applied.

Image Effect Pop-Up Menu

Use this pop-up to select an image effect. Options include:

- **Invert**: This option causes the color of the element to invert, that is, the opposite color of the element will be applied.
- **Selected Bevels**: This option applies a 3D beveled edge effect to the element. The width of the bevel edge and the tone of the color to be applied to the edge is specified in the Bevel Width and Tone Amount fields.
- **Deselected Bevels**: This option applies an indented 3D beveled edge effect to the element. The width of the bevel edge and the tone of the color to be applied to the edge are specified in the Bevel Width and Tone Amount fields.
- **Tone Down**: This option darkens the tone of the element's color according to the value specified in the Tone Amount field.
- **Tone Up**: This option lightens the tone of the element's color according to the value specified in the Tone Amount field.

Bevel Width Field
Use this field to select a width of the bevel, in pixels, when a bevel effect is selected. The up/down buttons increase or decrease the bevel size by 1 pixel. This field is active only when a bevel effect is selected.

Tone Amount Field
Use this field to select the severity of the selected effect. The up/down arrow buttons increase or decrease the tone in increments of 1%. This field is active for all effects except Invert.

Include Borders Checkbox
Select this option to apply the image effect to an element's border and/or shadow effect (if it has one). See "Border Field" on page 12.34 and "Shadows Shape Field" on page 12.34.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

Hot Tip

Use multiple image effect modifiers to create a bevel-edged 3D-button effect: one to show the regular state of the button, one to show it depressed and one to reset it to its regular state.

INTEGER VARIABLE

The integer variable modifier holds integer values. Integer variables are "long integers" with a range of ±2147483648. Integer variables are useful as counters or in mathematical computations.

Components of the integer variable dialog (Figure 12.24) are described below:

Variable's Name Field
This editable text field can be used to change the variable's name. If the variable will be referenced in a Miniscript modifier, it is good programming practice to make the name a single word.

Modifier Icon
This icon identifies the variable modifier's type.

Value Field
Enter the variable's initial value here or use the up/down arrow buttons to set the value. The value is checked for validity when the OK button is clicked. The value can be changed during runtime. See "Setting Values of Variable Modifiers" on page 14.5 for details regarding getting and setting the value of this variable modifier.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

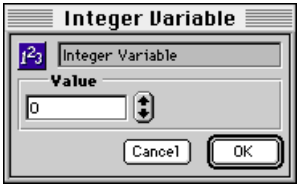


Figure 12.24 The Integer Variable dialog

INTEGER RANGE VARIABLE

An integer range variable is used to hold an integer range value. These values are especially useful for storing ranges of cels in an "mToon" animation. See "The mToon Menu" on page 2.6 for more information on mToons.

Components of the integer range dialog are described below.

Variable's Name Field
This editable text field can be used to change the variable's name. If the variable will be referenced in a Miniscript modifier, it is good programming practice to make the name a single word.

Modifier Icon
This icon identifies the variable modifier's type.

Value Fields
Enter the variable's initial values here. The input field on the left represents the start value. The input field on the right represents the end value. Highlight a field and enter a value or use the up/down arrow buttons to change the value of the field.

The value is checked for validity when the OK button is clicked.

- *Note: If this range is to be used with an mToon, the start and end values should be less than the number of cels in the animation being controlled.*

The values can be changed during runtime. See "Setting Values of Variable Modifiers" on page 14.5 for details regarding getting and setting the value of this variable modifier.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

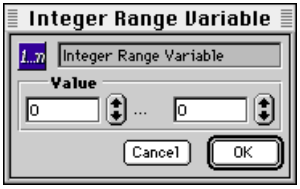


Figure 12.25 The Integer Range dialog

KEYBOARD MESSENGER

The keyboard messenger generates messages when a keyboard event, such as a keypress, is detected.

The keyboard messenger dialog (Figure 12.26) is described below.

Modifier's Name Field

This editable text field can be used to change the modifier's name.

Modifier Icon

This icon identifies the modifier's type.

Execute When Section

The controls in this section select the type of keyboard event to be detected. Select a key and key state:

Figure 12.26 The Keyboard Messenger dialog

Key

In the text field, enter the key or key combination that activate this messenger. Select the Control, Command or Option key checkbox, if desired.

Key State Radio Buttons

Select the type of key event to be detected. The keyboard messenger can detect the state of the key being pressed. The default state is key down, but the other choices allow for greater functionality:

- **Down**: The message or command is sent when the key is first pressed down.
- **Up**: The message or command is sent when the key is released after having been pressed.
- **Repeat**: The message or command is repeatedly sent while the key continues to be held down (i.e., while key repeat is activated). Note that this selection does not generate a message when the key is *first* pressed.

Message Specifications

Use this section to specify the message to be sent when a keyboard event is detected.

See "Configuring Messenger Modifiers" on page 12.10 or "Message Specifications" on page 12.47 for a complete description of the controls in this section.

Message Options

Click the triangle at the bottom of the dialog to display the Message Options section. See "Message Options" on page 12.11 or "Mes-

sage Options" on page 12.48 for a complete description of the controls in this section.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

LIST VARIABLE

•Note: In version 1.0 of mTropolis, this modifier is provided in a "beta" version—features and documentation may not be complete. In version 1.0, this modifier has no effect when used in titles built for the Windows platform.

The list variable modifier is a compound variable that can be used to store a list of values of the same data type. Values can be added to and deleted from the list dynamically via Miniscript statements. Other Miniscript statements can be used to sort the list, randomly shuffle the individual list elements, return the number of elements in a list, or return randomly-selected values from the list.

Note that the list variable dialog is used only for creating a list variable, naming it, and declaring the data type of its list elements. Adding values to a list or otherwise manipulating its contents must be performed through Miniscript statements (see "Miniscript Syntax for List Variables" on page 12.43 and Chapter 14, "Miniscript Modifier"). The contents of a list cannot be seen or set in any of mTropolis' editing views.

Figure 12.27 The List Variable dialog

Components of the list variable dialog (Figure 12.27) are described below.

Variable's Name Field
This editable text field can be used to change the variable's name. If the variable will be referenced in a Miniscript modifier, it is good programming practice to make the name a single word.

Modifier Icon
This icon identifies the variable modifier's type.

List Type Pop-Up Menu
Use this pop-up menu to select the data type of the list elements.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

Miniscript Syntax for List Variables

The individual elements of a list can be accessed and manipulated through Miniscript statements as described below.

Basic List Syntax
Individual elements of a list can be referenced by using the following syntax:

```
listname[n]
```

where *listname* is the name of the list, as specified in its name field and *n* is an integer from 1 to the total number of list elements. For example, to retrieve the value of the third element of the integer list variable `myIntList`

and store it in the variable `myInt`, use the statement:

```
set myInt to myIntList[3]
```

An entire list is represented by a comma-separated list of values enclosed in curly brackets:

```
{value1, value2, ..., valuen}
```

where the value of the *n*th element is specified by $value_n$.

Creating the Contents of the Entire List

To create and set the contents of an entire list at once, use the syntax:

```
set listname to {v1, v2, ..., vn}
```

where *listname* is the name of the list and the comma-separated list of values enclosed in curly brackets are the values to be assigned to elements 1 through *n*. Note that when this syntax is used, any previous values held by the list are lost—the list's values and total number of elements change to conform to the specified list.

For example, the following statement creates a four-element integer list:

```
set myIntList to {10, 20, 30, 100}
```

After execution of this statement, the first element of `myIntList` will contain 10, the second will contain 20, the third will contain 30, and the fourth element (the last one in the list) will contain 100.

Changing the Value of a List Element

To change a single value a list, use the **`set`** statement to assign a value to the desired element number in the list:

```
set listname[n] to value
```

where *listname* is the name of the list, *n* is the element's position in the list (starting at 1), and *value* is a data value of the correct type. A new list element is created at the specified position in the list. Note that if *n* exceeds the total number of elements in the list (i.e., the *n*th element does not exist), the *n*th element is created with the specified value, but intermediate elements are also created and assigned a default value of 0 (or null for string values). For example, consider the following statement, executed when `myIntList` contains no elements:

```
set myIntList[3] to 10
```

List element 3 will be created and assigned the value 10, but elements 1 and 2 will also be created and assigned the default value of 0.

Inserting a List Element

Instead of changing the value of an existing list element, a new value can be inserted at any position in the list, causing the current element at that position (and any subsequent elements) to move one position later in the list. To insert an element, use the **`set`** statement with the list's `insert` attribute as follows:

```
set listname.insert[n] to value
```

where *listname* is the name of the list, *n* is the desired position of the new element and *value*

is the data value to be stored in the new list element. For example, consider the following Miniscript statements that refers to an integer list variable, `myIntList`:

```
-- create a three-element list
set myIntList to {10, 20, 30}

-- insert a new element at the
-- second position
set myIntList.insert[2] to 100
```

After executing these statements, `myIntList` would contain the list `{10, 100, 20, 30}`.

Deleting a List Element

To delete an element from a list and return the deleted value (i.e., as if "popping" the element out of the list), reference the list's `delete` attribute in a Miniscript expression with the following syntax:

```
listname.delete[n]
```

where *listname* is the name of the list and *n* is the number of the element to be deleted from the list. The element is removed from the list, any later elements in the list are moved to one position earlier in the list to fill the gap, and the expression resolves to the value that was stored in the deleted element. For example, consider the following Miniscript statements that refer to a string variable, `myString`, and a string list variable, `myList`:

```
-- create a three-element string list
set myList to {"one", "two", "three"}

-- remove the second element from
-- the list
set myString to myList.delete[2]
```

After executing these commands, `myString` would contain the string value `"two"` and `myList` would contain the list `{"one", "three"}`.

Returning the Number of Elements in a List

The `count` attribute of a list variable stores the current number of elements in the list. Therefore, the Miniscript expression:

```
listname.count
```

where *listname* is the name of the desired list, resolves to the number of elements in the list. For example, the number of list elements could be retrieved with a statement such as:

```
set myInt to myIntList.count
```

Because the `count` attribute is writable, the number of list elements can also be changed by setting `count` to a new value. For example:

```
set myIntList.count to 10
```

Note that, if the new value of count is less than the previous value, the list is truncated, but remaining elements retain their previously stored values. If the new value of count is greater than the previous value, new elements are created past the end of the previous list. These new elements contain the default value of 0 (or null for strings).

Returning a Randomly-Selected Value from a List

To return a randomly-selected value from a list, reference the list's random attribute:

```
listname.random
```

where *listname* is the name of the list. For example, to store a value, chosen at random

from the list `myStringList`, into the string variable `myString`, use the statement:

```
set myString to myStringList.random
```

Sorting the Values in a List

The values in a list can be sorted in three different ways using the `sortAscend`, `sortDescend`, and `shuffle` list attributes.

To sort the values in a list in ascending order (i.e., the first element contains the smallest value and the last element contains the greatest value), reference the list's `sortAscend` attribute:

```
listname.sortAscend
```

where *listname* is the name of the list to be sorted. This expression resolves to an integer value that is the total number of elements in the list.

Similarly, the list can be sorted in descending order (i.e., the first element contains the greatest value and the last element contains the smallest value) by referencing the list's `sortDescend` attribute:

```
listname.sortDescend
```

where *listname* is the name of the list to be sorted. Again, the expression returns an integer that represents the total number of elements in the list.

For example, consider the following Miniscript statements that refer to an integer variable, `myInt`, and an integer list, `myIntList`:

```
-- create a four-element list
set myIntList to {30, 40, 10, 20}

-- sort the list and return the number
-- of elements
set myInt to myIntList.sortDescend
```

After executing these statements, `myInt` contains the value 4, and `myIntList` contains the list `{40, 30, 20, 10}`.

Randomizing the Order of List Elements

List elements can be "unsorted" into a random order by referencing the list's `shuffle` attribute:

```
listname.shuffle
```

where *listname* is the name of the list to be shuffled. The expression returns an integer that represents the total number of elements in the list.

MESSENGER

The messenger modifier (sometimes referred to as simply a "messenger") sends a message or command after it has received a specified message.

Components of the Messenger dialog are described below.

Modifier's Name Field

This editable text field can be used to change the modifier's name.

Modifier Icon

This icon identifies the modifier's type.

Execute When Pop-Up Menu

Use this pop-up menu to select the message that causes the messenger to activate. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

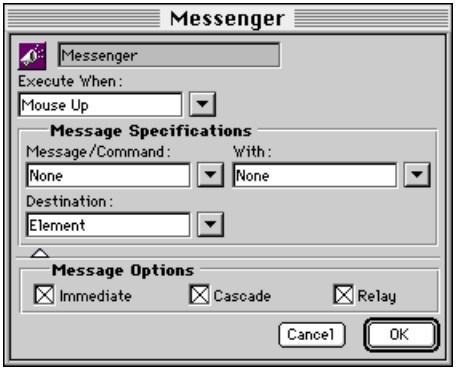


Figure 12.28 The Messenger dialog

Message Specifications

All messenger modifiers have the same controls in the Message Specification section of their dialogs:

Message/Command Pop-Up Menu

Use this pop-up to select the message or command to be sent when the messenger is activated. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

With Pop-Up Menu

Use this menu to select a value to be sent with the message or command. The choices on this pop-up are None, Incoming Data and any variables to which the element has access. These menu items are described in more detail below:

- **None**: This is the default selection. No value is sent with the message.
- **Incoming Data**: Choose this option to configure the messenger to send the incoming data (i.e., the value that arrives with the message that has been configured to trigger the messenger), with the outgoing message or command.
- **Variables**: To send the value of a variable modifier with the outgoing message, select its name from the submenus available in the second section of the With pop-up menu. All variable modifiers currently available to a messenger from its position in the project's hierarchy are shown. These variable modifiers are only accessible to those modifiers or Miniscripts in its

"scope." See "Variable Scopes" on page 13.25.

- **Constant Data Value**: To send a constant data value, highlight the With text field and type the value to be sent. The value should be entered with the same syntax as if it were being used in a Miniscript modifier. The syntax of the expression entered is checked when OK is clicked. Any appropriate mTropolis data type can be sent.

Destination Pop-Up Menu

Use this menu to select the destination for the message or command sent from this modifier. For a complete discussion of this menu, see"The Destination Pop-Up Menu" on page 13.21.

Message Options

All messenger dialogs have a Message Options section that can be revealed by clicking the triangle at the bottom of the dialog. Options in this section are:

Cascade Checkbox

When this box is checked, the message "cascades" from the targeted destination to all components in the hierarchy below it. Uncheck this option to configure the messenger to send its message only to the modifiers in a targeted element. "Cascade off" is equivalent to the path of an environment message sent by mTropolis.

- *Note: This option has no effect when sending a command.*

Immediate Checkbox

By default, messages or commands are sent immediately when the messenger is activated. Uncheck this box to configure the messenger to send its message or command only after the current thread of messages has ended. This option can be used to avoid system overload if the message thread queue becomes too deep.

Relay Checkbox

By default, messages travel from element to element in the hierarchy activating any and all elements that respond to the message. Uncheck this box to activate only the *first* modifier in the path of the message that is configured to respond.

- *Note: This option has no effect when sending a command.*

Cancel Button

Click Cancel to ignore changes made to this modifier.

OK Button

Click OK to accept changes made to this modifier.

MINISCRIPT MODIFIER

Miniscript is a simple scripting language embedded in a modifier. The Miniscript modifier allows you to use a scripting language to create customized modifiers that can:

- Get and set element attributes.
- Send messages and commands.
- Evaluate mathematical functions.
- Evaluate relational expressions and perform conditional branching.

A complete description of the Miniscript language can be found in Chapter 14, "Miniscript Modifier".

Components of the Miniscript modifier dialog (Figure 12.29) are described below.

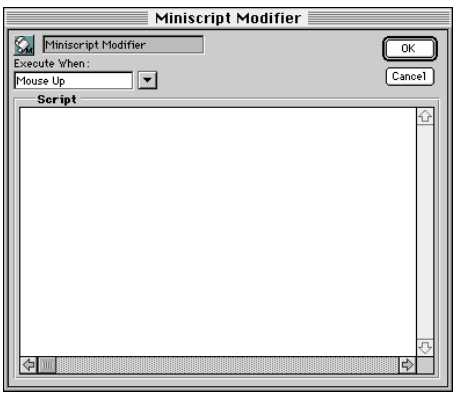


Figure 12.29 The Miniscript modifier dialog

Modifier's Name Field
This editable text field can be used to change the modifier's name.

Modifier Icon
This icon identifies the modifier's type.

Execute When Pop-Up Menu
Use this pop-up menu to select the message that causes the Miniscript modifier to execute. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Script Text Field
Enter the desired Miniscript script in this field. The syntax of the script is checked when the OK button is clicked. A complete description of the Miniscript language can be found in Chapter 14, "Miniscript Modifier".

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

NAVIGATION MODIFIER

•Note: In version 1.0 of mTropolis, this modifier is provided in a "beta" version—features and documentation may not be complete. In version 1.0, this modifier has no effect when used in titles built for the Windows platform.

The navigation modifier executes a scene change when triggered in runtime mode. Any scene in any section and subsection of the project can be selected by name or by certain "relative" criteria. Note that this modifier is a "superset" of the functionality provided by the change scene modifier (described on page 12.20).

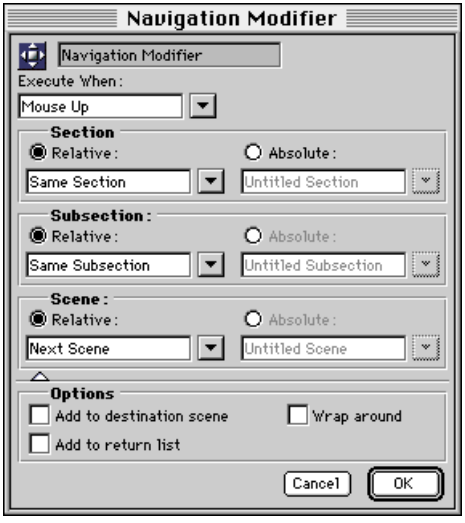


Figure 12.30 The Navigation Modifier dialog

Components of the navigation modifier dialog (Figure 12.30) are described below.

Modifier's Name Field

This editable text field can be used to change the modifier's name.

Modifier Icon

This icon identifies the modifier's type.

Execute When Pop-Up Menu

Use this pop-up menu to select the message that causes the scene to change. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Section Options

Use the controls in this section of the dialog to select the section that contains the scene to be changed to. Options include:

- **Relative Radio Button**: Select this button to activate the Relative pop-up menu. This menu can be used to select a section by its position relative to the current section. Select from "Same Section", "Previous Section", "Next Section", "First Section", and "Last Section". Note that the order of sections can be changed in the structure window. See "Changing the Order of Components in the Structure Window" on page 8.6.
- **Absolute Radio Button**: Select this button to activate the Absolute pop-up menu. This menu can be used to select a section by name.

Subsection Options

Use the controls in this section of the dialog to select the subsection that contains the scene to be changed to. Options include:

- **Relative Radio Button**: Select this button to activate the Relative pop-up menu. This menu can be used to select a subsection by its position relative to the current subsection. Select from "Same Subsection", "Previous Subsection", "Next Subsection", "First Subsection", and "Last Subsection". The "Corresponding Subsection (by Index)" can be selected to designate the subsection in the new section that has the same position in the subsection order as the current subsection. The "Corresponding Subsection (by Name)" option can be selected to designate the subsection in the new section that has the same name as the current subsection. Note that the order of subsections can be changed in the structure window. See "Changing the Order of Components in the Structure Window" on page 8.6.
- **Absolute Radio Button**: Select this button to activate the Absolute pop-up menu. This menu can be used to select a subsection by name. This option is available only if the Section has also been specified by name (i.e., the Section option's Absolute radio button is also selected).

Scene Options

Use the controls in this section of the dialog to select the scene to be changed to. Options include:

- **Relative Radio Button**: Select this button to activate the Relative pop-up menu. This menu can be used to select a scene by its position relative to the current scene. Select from "Same Scene", "Previous Scene", "Next Scene", "First Scene", and "Last Scene". The "Corresponding Scene (by Index)" can be selected to designate the scene in the new subsection that has the same position in the scene order as the current scene. The "Corresponding Scene (by Name)" option can be selected to designate the scene in the new subsection that has the same name as the current scene. Note that the order of scenes can be changed in the structure window. See "Changing the Order of Components in the Structure Window" on page 8.6.
- **Absolute Radio Button**: Select this button to activate the Absolute pop-up menu. This menu can be used to select a scene by name. This option is available only if the Subsection has also been specified by name (i.e., the Subsection option's Absolute radio button is also selected).

Options Check Boxes

Click the open/close triangle at the bottom of the Navigation Modifier dialog to reveal navigation options:

- **Add to Destination Scene**: Select this checkbox to perform a special scene change in which the currently displayed scene is still visible after the change to the new scene. That is, the current scene acts like a shared scene for this scene change.

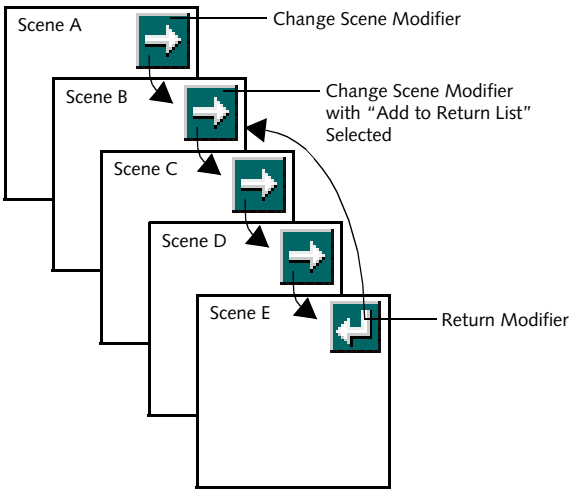


Figure 12.31 The Return List

- **Add to Return List**: Select this checkbox to make the current scene the one returned to by the next return modifier in a series of scenes. For example, in Figure 12.31, change scene modifiers have been placed on each of the scenes. Each of these modifiers has been configured to change to the next scene. Scene B's scene change modifier has the Add to Return List option selected. A return modifier has been placed on Scene E. When this modifier receives the message that will activate it during runtime, Scene E will change to Scene B (the arrows show the sequence of scenes).
- **Wrap Around**: By default, the first and last scene in a subsection have no connection in the scene order. That is, the last scene in a subsection has no next scene and the first scene in a subsection has no previous scene.

  When this checkbox is selected, a change to the "Next Scene", executed from the last scene in a subsection, "wraps around" the list of scenes and changes to the first scene in the subsection.

  Similarly, a change to the "Previous Scene", executed from the first scene in a subsection, "wraps around" the list of scenes and changes to the last scene in the subsection.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

OBJECT REFERENCE VARIABLE

•Note: There are two versions of this modifier provided with mTropolis. The version found on Modifier Palette Group 3 is a newer, experimental version. Only the Group 3 Object Reference Variable, with its enhanced functionality, is described below. The Group 3 version is Macintosh-only—it has no effect when used in a title built for Windows.

The object reference variable stores a reference to another mTropolis object. It acts as a "pointer" to another mTropolis object.

The object reference variable can be configured to store an initial value. It can also be configured to update its reference automatically upon receipt of a message. When the specified message is received, the object reference variable stores a reference to the parent of the messenger that sent the activating message.

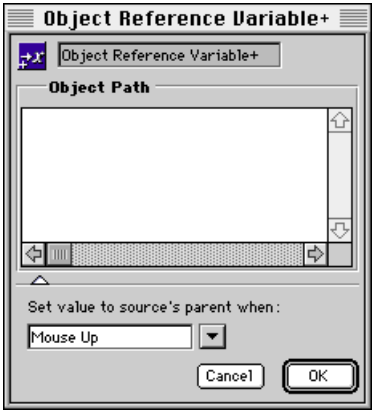


Figure 12.32 The Object Reference Variable dialog

Once a value is stored, the referenced component can be targeted as the recipient for messages sent from Miniscript or used as a value in other operations. See "Miniscript Syntax for Object Reference Variables" on page 12.54.

Components of the object reference variable dialog (Figure 12.32) are described below.

Variable's Name Field

This editable text field can be used to change the variable's name. If the variable will be referenced in a Miniscript modifier, it is good programming practice to make the name a single word.

Modifier Icon

This icon identifies the variable modifier's type.

Object Path Field

This field can be used to set an initial value for the object reference variable. This field is also updated whenever the value stored in the object reference variable changes. This field specifies the "path" to an object in a mTropolis project using the syntax described below:

- The path is described using a syntax similar to that used to specify directory names in Unix. The full path to an object, starting from the project level would be:

  ```
  /project/subsection/section/scene/element
  ```

  where *project* is the actual name of the project, *subsection* is the name of the subsection, etc., all the way to *element* which is

the name of the specific element being referenced.

- As implied above, the slash character (/) separates levels in the mTropolis structure hierarchy.
- To reference objects by their position relative to the object reference variable, the full pathname can be omitted. For example, to target an object on the same level of the structure hierarchy as the object reference variable itself, simply use the objects name, for example:

  ```
  mybehavior
  ```

- To specify structure levels *above* the current position, use "`..`" to move up a level and "`/`" to separate levels. For example, the path:

  ```
  ../myelement
  ```

  would cause "myelement" to be searched for at the scene level.

The syntax for this field is checked when the OK button is clicked. If the syntax is not in the correct format, an error alert appears. Note that, while the syntax is checked, the actual existence of the object specified in this field is *not* checked.

Set Value to Source's Parent When Pop-Up Menu
Click the open/close triangle at the bottom of the object reference dialog to reveal the "Set value to source's parent when:" pop-up menu. Use this menu to select a message that causes the value of the variable to change to the parent of the messenger that sent the activating message.

- *Note: As mTropolis has no "parent," environment messages originating from mTropolis cannot be used to set the value of this variable. For example, suppose "Mouse Up" is selected in the "Set Value to Source's Parent" menu. A user mouse click that generates a Mouse Up message will not cause the value of this variable to change. However, a "Mouse Up" message sent by a messenger modifier would trigger the value change.*

For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

Miniscript Syntax for Object Reference Variables

The value contained by an object reference variable can be accessed through Miniscript as described below. A general overview of Miniscript can be found in Chapter 14, "Miniscript Modifier".

Sending Messages
The object referenced by the object reference variable can be accessed as the "object" attribute of the object reference variable. Therefore, to send a message to the object at which

the object reference variable is pointing, use the following syntax:

```
send "message" to objectvar.object
```

where *message* is the message or command to be sent and *objectvar* is the name of the object reference variable being accessed. For example, to send the *Play* message to the object referenced by an object reference variable named "pointer1", use the statement:

```
send "Play" to pointer1.object
```

Changing the Value of an Object Reference Variable

To change the reference held by an object reference variable, use the following syntax:

```
set objectvar to object
```

where *objectvar* is the name of the object reference variable being accessed and *object* is a mTropolis object specified by name, by relative position (see Table 14.2 on page 14.9 for a list of "building blocks" that can be used to specify elements), or by the "object path field" syntax. A number of examples follow:

```
-- point to the parent of the element
-- that contains the variable:
set myObjectRef to element.parent

-- point to the scene:
set myObjectRef to scene

-- point to the message source's
-- parent:
set myObjectRef to source's parent

-- point to the message source's
-- element:
set myObjectRef to source's element

-- point to an element by name:
set myObjectRef to myBigBlueButton

-- change the value using path
-- syntax:
set myObjectRef to \
  "/myProj/mySect/mySub/Scene/Bob"
```

Accessing Attributes of the Referenced Object

Any attributes of the object pointed to by the object reference variable can be targeted using the syntax:

```
objectvar.object.attribute
```

where *objectvar* is the name of the object reference variable and *attribute* is the name of an attribute of the referenced object. See "Element Attributes" on page 14.17 for more information on attributes.

For example, the width of the referenced object can be retrieved:

```
set myInt to myObjectRef.object.width
```

Similarly, the attributes of a referenced variable can be changed:

```
set myObjectRef.object.position \
    to (40, 50)
```

PATH MOTION MODIFIER

The path motion modifier can be used to add motion along a path to an element. Any child elements of the element modified by the path motion modifier also move along the path.

This modifier is especially useful when used with mToons. It can be used to specify which cels are visible at each point along the path. Multiple path motion modifiers can be used to define many paths for a single animation. Each of these modifiers can be configured to respond to different messages, allowing the element to move on a specific path according to specific conditions.

The motion path is stored in the modifier itself. Once a path is defined, it can be used again by dragging a copy onto another Graphic element, movie or mToon.

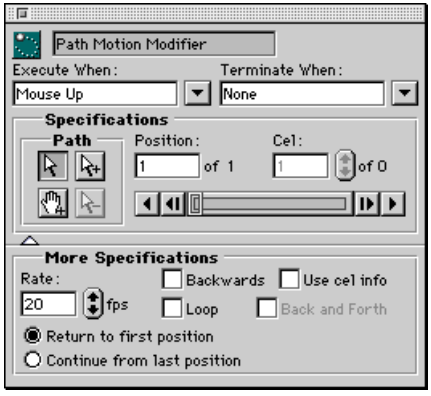


Figure 12.33 The Path Motion modifier dialog

Components of the path motion modifier dialog (Figure 12.33) are described below:

Modifier's Name Field
This editable text field can be used to change the modifier's name.

Modifier Icon
This icon identifies the modifier's type.

Execute When Pop-Up Menu
Use this pop-up menu to select the message that enables the path motion. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Terminate When Pop-Up Menu
Use this pop-up menu to select the message that stops the path motion. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Specifications Section

The controls in this section allow you to create the motion path and specify animation cels for individual positions.

Path Tools

Selection Arrow: The selection arrow allows you to move any position on the motion path. When a position is selected, the modifier's associated element will automatically move to that position on the path.

Add Position Tool: Use this tool to define a new position on the path. Clicking on the scene after

the last selected point adds a position to the path.

To add a position between two points on an existing path, select the first of the two points with the Selection Arrow, and click on the scene before the next point. (If the Option key is depressed while the Selection Arrow with "+" sign is selected, the select arrow appears, allowing a point to be selected and/or moved.)

Drag Path Tool: This tool allows you to Mouse Down and drag an element on its scene to define a motion path. When the mouse is released the points of the modifier's path are automatically drawn. (If the Option key is depressed while the hand with the "+" sign is selected, the select arrow appears, allowing a point to be selected and/or moved.)

Delete Position Tool: This tool allows you to select and delete positions anywhere along the motion path. The Cel and Position fields work together, allowing the author to assign a specific cel of an animation to a selected position on the motion path.

Position Field

This field shows the number of the position currently being edited of the total positions on the path. The step controls below this field allow you to step forward or back through the positions of the motion path one at a time.

Cel Field

This field shows the number of the current cel and the total number of cels in the animation. Use the arrow controls to scroll through the cels of the animation. Each point on the motion path can have a different cel associated with it.

Animation Controller Bar

These basic controls allow you to step, or play through the positions forward or backward. The arrows to the right and left allow you to preview the motion path forward and backwards while the dialog is open.

To use this feature, the compression method used to compress the animation must allow cels to be randomly accessible.See "Compression" on page 2.9.

More Specifications Section

Click the white triangle at the bottom of the path motion dialog to toggle the display of more options for path animations:

Rate Field

Select the rate, in frames per second, at which the animation will be played over the path. This setting overrides any previous rate setting in the element's Element Info dialog.

Return to First Position Radio Button

Select this option to reset the element to its original starting position whenever the path motion restarts or is looped.

Continue from Last Position Radio Button

Select this option to continue path motion from the last position of the path whenever path motion restarts or is looped.

Backwards Checkbox
Select this box to play through the motion positions from the last position to the first position.

Use Cel Info
Select this box to use the cel positions defined with the "Cel" field in the dialog's Specifications section.

When unselected, cels will be mapped sequentially to each position. If the number of cels is less than the number of positions on the path, the animation sequence will loop. If the number of cels is greater than the number of positions on the path, the number of cels displayed will be divided among the number of defined positions.

Loop Checkbox
Select this box to make the path motion loop. That is, the element moves along the motion path continuously from beginning to end positions.

Back & Forth Checkbox
Select this option to play the path motion from start to end, then backward from end to start.

POINT VARIABLE

The point variable modifier stores a pair of integer values, commonly used for storing and/or setting the screen location of elements. For example, the position of an element can be changed during runtime by placing multiple point variables on the element with different values in each. New positions can then be set using the Miniscript modifier

Note that the location of an element is measured in pixels relative to its parent's origin (the upper left corner of the parent).

Components of the point variable dialog (Figure 12.34) are described below:

Variable's Name Field
This editable text field can be used to change the variable's name. If the variable will be referenced in a Miniscript modifier, it is good programming practice to make the name a single word.

Modifier Icon
This icon identifies the variable modifier's type.

Value Fields
Enter the variable's initial value here or use the up/down arrow buttons to set the value. There are two fields:

- X: The horizontal value.
- Y: The vertical value.

The value is checked for validity when the OK button is clicked. The value can be changed during runtime. See "Setting Values of Variable Modifiers" on page 14.5 for details regarding getting and setting the value of this variable modifier.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

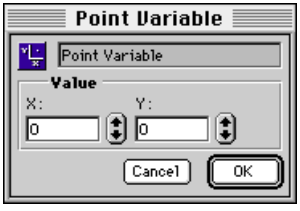


Figure 12.34 The Point Variable dialog

RETURN MODIFIER

Upon receipt of a specified message, the return modifier executes a scene change to the first scene in the "scene return list". See the Figure 12.11 on page 12.21 for a graphical explanation of the return list. Note that once a scene has been returned to, it is removed from the return list and the next scene in the list (if any) becomes the next scene to be returned to.

Components of the return modifier dialog (Figure 12.35) are described below:

Modifier's Name Field
This editable text field can be used to change the modifier's name.

Modifier Icon
This icon identifies the modifier's type.

Execute When Pop-Up Menu
Use this pop-up menu to select the message that triggers the scene return. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

Figure 12.35 The Return Modifier dialog

SAVE AND RESTORE MODIFIER

The save and restore modifier can be used to save data stored in a mTropolis variable to a file and restore that data upon receipt of specified messages. Any previously created mTropolis variable of any type (including user-defined compound variables) can be saved and restored. This sort of functionality is essential for game titles that are designed to be played over multiple sessions. Using an elaborate compound variable (see "Compound Variable" on page 12.25) with this modifier allows the author to save and restore complicated states.

The file can be saved to a number of preset locations or the user can be prompted to select a location for the file.

Components of the save and restore modifier dialog (Figure 12.36) are described below.

Modifier's Name Field
This editable text field can be used to change the modifier's name.

Figure 12.36 The Save and Restore Modifier dialog.

Modifier Icon
This icon identifies the modifier's type.

Save When Pop-Up Menu
Use this menu to select the message that causes the selected variable to be saved. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Restore When Pop-Up Menu
Use this menu to select the message that causes the selected variable to be restored (i.e., read from the previously saved file). The specified mTropolis variable is set to the values contained in the save file. Note that if a "restore" operation is attempted before a "save" operation has been performed (i.e., a file with the specified name and location does not yet exist), the current value of the specified variable is not changed. For a complete discussion of the options in this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Data to Save/Restore Pop-Up Menu
Use this pop-up menu to select the variable that contains the data to be saved and restored. When saving, the selected variable is written to the specified file. When restoring, the data value(s) last saved to the file are placed into this variable. All variable modifiers currently available to a messenger from its position in the project's hierarchy are shown. These variable modifiers are only accessible to those modifiers or Miniscripts in its "scope." See "Variable Scopes" on page 13.25.

One other option, "None" is also available. This option is the default. If None is selected

as the data value and a save or restore operation is triggered, nothing happens—the specified save/restore file remains unchanged.

File Location Pop-Up Menu

Use this pop-up to select the location in which the file (specified in the "File Name:" pop-up) will be saved. Options are:

- **Title Folder**: Select this option to save the data file in the application's title folder. This option is ignored, and replaced with the "Ask User" option when the title is built for Windows platforms.
- **Preferences**: Select this option to save the data file in the Macintosh "Preferences" folder, found in the Macintosh "System Folder". This option is ignored, and replaced with the "Ask User" option when the title is built for Windows platforms.
- **Ask User**: Select this option to display a file selection dialog in which the user selects the path and file to be written to or loaded from. The filename specified in the "File Name" field becomes the "default" filename that the user can change.
- *Note: When a title is built for Windows platforms, this modifier always uses the "Ask User" option, even if a different option was selected in the File Location pop-up.*

File Name Field

Use this field to specify the name of the file to be saved in the location selected in the "File Location" pop-up. The default name is **save.dat**.

Cancel Button

Click Cancel to ignore changes made to this modifier.

OK Button

Click OK to accept changes made to this modifier.

SCENE TRANSITION MODIFIER

The scene transition modifier adds a special transition effect to a scene. Transition effects are often desired when changing from one scene to the next. The transition pop-up lists options, such as dissolve, slide, push, etc. The transition becomes visible only on the next scene change.

Components of the scene transition modifier dialog (Figure 12.37) are described below:

Modifier's Name Field
This editable text field can be used to change the modifier's name.

Modifier Icon
This icon identifies the modifier's type.

Enable When Pop-Up Menu
Use this pop-up menu to select the message that enables the scene transition. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

This modifier is commonly activated on a Scene Started message. However, other messages could be chosen to activate a transition. For example, when a user clicks an element to turn right, a push right could be enabled, while when clicking an element to turn left, a push left could be enabled.

Disable When Pop-Up Menu
Use this pop-up menu to select the message that disables the scene transition. Note, however, that a transition cannot be disabled while it is being played.

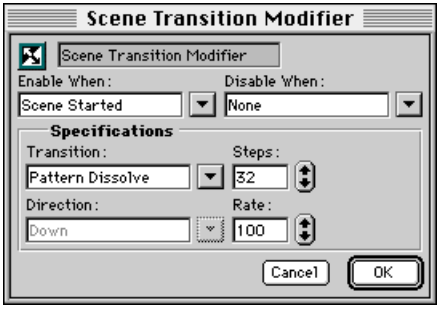


Figure 12.37 The Scene Transition modifier

Specifications Section

Use the controls in this section to select the type of transition.

Transition Pop-Up Menu
This pop-up contains a list of all available transition types. Options include:

- Pattern Dissolve
- Random Dissolve
- Fade
- Push
- Slide
- Wipe
- Zoom

Steps Field
Use this field to specify the number of steps between the start and finish of the transition.

The greater this number, the finer (and slower) the transition.

Direction Pop-Up
Some transitions (e.g., Slide, Push and Wipe) have associated directions. Use this pop-up to select a direction for the transition effect.

Rate Field
Use this field to select the speed of the transition. Values for this field can range from 0 (slowest) to 100 (fastest). Note that the speed setting is processor dependent—different computers will render the transition at different speeds, based on the speed of the computer.

- *Note: On faster computers, even medium speed settings may make the transition happen too quickly to be noticed. If a transition seems not to be working, try lowering the rate to a very low setting (e.g., 0 or 1) and increasing the number of steps. Then incrementally increase the rate until the effect appears at the desired speed.*

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

SET VALUE MODIFIER

The set value modifier can be used to change the value of a variable, or the value of incoming data, to a new value on receipt of a specified message. This modifier is useful for simple functions, such as resetting scores to 0 when leaving a scene. It can also be used in more complex operations, such as setting the value of a aliased variable to the value of another variable during runtime.

Using the set value modifier to change the value of a variable is equivalent to the Miniscript assignment statement:

```
set variableName to expression
```

where *variableName* is the name of a valid variable in the current project and *expression* is a Miniscript expression that evaluates to the proper type for storage in the variable. However, the set value modifier executes faster, makes projects easier to understand, and should be used in preference to single-line Miniscript assignment statements.

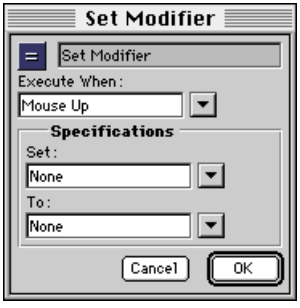


Figure 12.38 The Set Value Modifier dialog

Components of the set value modifier dialog (Figure 12.38) are described below:

Modifier's Name Field

This editable text field can be used to change the modifier's name.

Modifier Icon

This icon identifies the modifier's type.

Execute When Pop-Up Menu

Use this pop-up menu to select the message that triggers the set operation. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Specifications Section

The controls in this section specify the variable or property to be set and the new value.

Set Pop-Up Menu

Select the data or variable to be changed from this pop-up. Options include:

- **None**: This is the default selection. No data or variable is changed on receipt of the specified message.
- **Incoming Data**: Choose this option to set the incoming data (i.e., the value that arrives with the message that triggers the set value modifier) to the value entered in the To field.
- **Variable**: Any variable modifier in the set value modifier's scope can be specified to be

changed. The variable modifiers will be visible in the Set pop-up. Variables are only available to those modifiers or Miniscripts in its scope. For information regarding scoping rules, see "Variable Scopes" on page 13.25.

To Pop-Up Menu

Use this menu to select the data or variable to which the item specified in the Set pop-up is to be changed. The choices on this pop-up are None, Incoming Data and any variables to which the element has access. A data value can be sent by highlighting the To text field and typing in a value. These menu items are described in more detail below:

- **None**: This is the default selection. No set operation is performed (i.e., the selected value is not changed).
- **Incoming Data**: Choose this option to configure the messenger to set the item in the Set field to the incoming data (i.e., the value that arrives with the message that has been configured to trigger the messenger).
- **Variables**: To set the item in the Set field to a variable, select its name from the submenus available in the second section of the To pop-up menu. All variable modifiers currently available to a messenger from its position in the project's hierarchy are shown. These variable modifiers are only accessible to those modifiers or Miniscripts in its "scope." See "Variable Scopes" on page 13.25.
- **Constant Data Value**: To set the item in the Set field to a constant data value, highlight the To text field and type the value to be sent. The value should be entered with the same syntax as if it were being used in a Miniscript modifier. The syntax of the expression entered is checked when OK is clicked. Any appropriate mTropolis data type can be sent.

Cancel Button

Click Cancel to ignore changes made to this modifier.

OK Button

Click OK to accept changes made to this modifier.

SHARED SCENE MODIFIER

The shared scene modifier can be used to specify a scene to become the shared scene. Note that there is always one, and only one, shared scene in a subsection. The new shared scene replaces the current shared scene. The displaced shared scene becomes a "regular" scene—the first scene in the subsection. Any other scenes in the subsection "move" in the scene order to accommodate the new first scene. Note, however, that the current scene (the one visible to the user) does not change except to display the new shared scene as its background. More information about shared scenes can be found in "The Shared Scene" on page 9.5 and "Shared Scenes" on page 10.2.

Components of the shared scene modifier dialog (Figure 12.39) are described below:

Modifier's Name Field
This editable text field can be used to change the modifier's name.

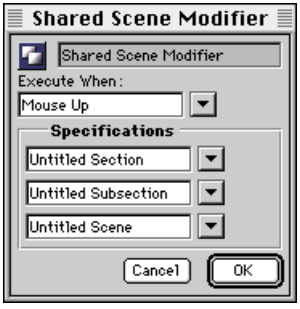


Figure 12.39 The Shared Scene Modifier dialog.

Modifier Icon
This icon identifies the modifier's type.

Execute When Pop-Up Menu
Use this pop-up menu to select the message that causes the new shared scene to be set. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Specifications Section

Select the scene to become the shared scene from the pop-up menus in this section. The first pop-up selects a section, the second selects a subsection, and the third selects the scene for specifying the new shared scene.

The default is the current section, current subsection and current scene. Note that the scene default should *always* be changed because if the current scene is targeted to become the shared scene, nothing happens (i.e., the shared scene is not changed). Note that this is true when any shared scene modifier (no matter where it is located) attempts to make the current scene the shared scene.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

SIMPLE MOTION MODIFIER

The simple motion modifier applies pre-defined motion paths to elements, including text, graphics, mToons, and video elements. Simple directional motion or random movement can be selected.

Components of the simple motion modifier dialog (Figure 12.40) are described below:

Modifier's Name Field
This editable text field can be used to change the modifier's name.

Modifier Icon
This icon identifies the modifier's type.

Execute When Pop-Up Menu
Use this pop-up menu to select the message that enables the element motion. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

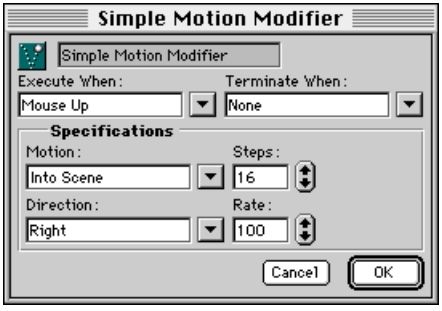


Figure 12.40 The Simple Motion modifier dialog

Terminate When Pop-Up Menu
Use this pop-up menu to select the message that stops the element motion. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Specifications Section

Use the controls in this section to customize the simple motion.

Motion Pop-Up Menu
Use this pop-up menu to select a basic motion type. Options include:

- **Into Scene**: The object moves into the scene to its current position as defined in the layout window.
- **Out of Scene**: The object moves off the scene from its current position as defined in the layout window.
- **Random Bounce**: The object bounces around at random angles within the frame of its parent.

Steps Field
Use this field to set the number of positions between start and end points of the motion path. The greater the number, the finer and slower the animation. This field is not available for all motion types.

Direction Pop-Up Menu
If the motion chosen has an associated direction, this pop-up becomes visible. Use this menu to select a direction for the simple motion.

Rate Field
Use this field to set a speed of the motion. Note that this speed is not absolute, but varies depending on the speed of the playback computer.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

SOUND EFFECT MODIFIER

The sound effect modifier can be used to play sound effects in response to a message.

Both AIFF and Macintosh system snd files can be played. Note that snd files can only be played one at a time, but multiple AIFF files can be played simultaneously.

Components of the sound effect modifier dialog (Figure 12.41) are described below:

Modifier's Name Field
This editable text field can be used to change the modifier's name.

Modifier Icon
This icon identifies the modifier's type.

Execute When Pop-Up Menu
Use this pop-up menu to select the message that starts the sound effect. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

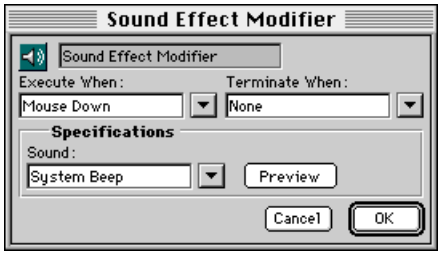


Figure 12.41 The Sound Effect Modifier dialog

Terminate When Pop-Up Menu
Use this pop-up menu to select the message that terminates the sound effect. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Sound Pop-Up Menu
Use this pop-up to select a sound that has been linked to the project. Alternatively, select Link Sound from this menu to link a new sound. A standard file dialog appears.

The current system beep of your computer is the default setting of this modifier. Note however, that the system beep is different on different machines, so don't rely on this sound being the same during runtime on other machines.

Preview Button
Click this button to preview the sound currently selected in the Sound pop-up.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

SOUND FADE MODIFIER

•Note: There are two versions of this modifier provided with mTropolis. The version found on Modifier Palette Group 3 is a newer, experimental version. Only the Group 3 Sound Fade Modifier, with its enhanced functionality, is described below. The Group 3 version is Macintosh-only—it has no effect when used in a title built for Windows.

The sound fade modifier decreases or increases the volume of a sound element. For example, this modifier can be used to obscure the abrupt halt of background audio by fading it to zero volume before the sound ends.

Note that this modifier only effects sound elements and video elements that play audio (e.g., QuickTime elements). This modifier has no effect when attached to graphic elements, even if that graphic element has a *sound effect* modifier (see page 12.70) attached.

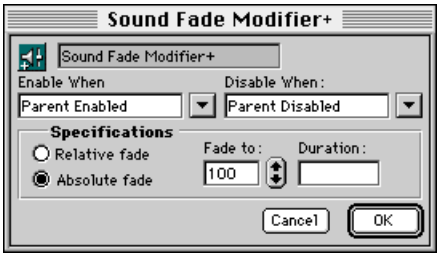


Figure 12.42 The Sound Fade Modifier dialog

Components of the sound fade modifier dialog (Figure 12.42) are described below:

Modifier's Name Field
This editable text field can be used to change the modifier's name.

Modifier Icon
This icon identifies the modifier's type.

Enable When Pop-Up Menu
Use this pop-up menu to select the message that starts the sound fade. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Disable When Pop-Up Menu
Use this pop-up menu to select the message that stops the sound fade. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Relative Fade Radio Button
Select this button to specify a sound fade relative to the sound's current volume. This fade is specified as a percentage from 0 to 10,000% in the "Fade to:" field.

Absolute Fade Radio Button
Select this button to specify a sound fade to an absolute volume. This fade is specified as a percentage from 0 to 100% in the "Fade to:" field.

Fade To Field
Use this field to specify the volume level to which the sound fades. The meaning of this field changes depending upon which radio button is selected:

- If the Relative Fade button is selected, this field specifies a percentage of the sound's *current* volume. Valid values range from 0 (no sound) through 100 (no change to current sound) to 10000 (increase the current sound level by 100 times). Note that the sound can only fade up to its full volume level, no matter what this value is set to (i.e., the sound is never *amplified* by mTropolis).
- If the Absolute Fade button is selected, this field specifies a percentage of the sound's *full* volume. Valid values range from 0 (no sound) to 100 (sound played at full volume). If a value greater than 100 is entered in this field, the sound plays at full volume and the value will be shown as 100 the next time this dialog is opened.

Duration Field
Use this field to specify the duration of the fade in `minutes:seconds.hundredths` format.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

SOUND PANNING MODIFIER

•Note: In version 1.0 of mTropolis, this modifier is provided in a "beta" version—features and documentation may not be complete. In version 1.0, this modifier has no effect when used in titles built for the Windows platform.

The sound panning modifier changes the stereo position (i.e., the left/right balance) of a sound element or the sound in a video (e.g., QuickTime) element. Note that this modifier cannot modify a sound being played by the sound effect modifier, only sound elements and the sound portion of video elements can be modified. Sound elements can only be added in the structure view—for details, see "To create a new sound element:" on page 8.5.

Components of the sound fade modifier dialog are described below.

Modifier's Name Field
This editable text field can be used to change the modifier's name.

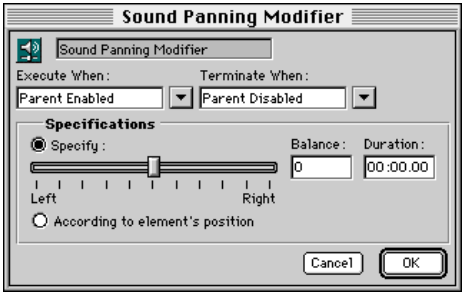


Figure 12.43 The Sound Panning Modifier dialog

Modifier Icon
This icon identifies the modifier's type.

Execute When Pop-Up Menu
Use this pop-up menu to select the message that activates the sound panning. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Terminate When Pop-Up Menu
Use this pop-up menu to select the message that ends the sound panning. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Specifications Section

Use the controls in this section to specify the type of pan to be performed.

Specify Radio Button
Select this radio button to set a specific panning position for the sound. When this option is selected, three other interface elements become active:

- **Left/Right Slider**: Use this slider to select the pan position of the element. The Balance text field updates to reflect the position of the slider.
- **Balance Field**: This field can be used to enter a precise pan position. A value of -100 indicates full left, 0 indicates center, and 100 indicates full right. The Left/Right slider updates to reflect the value entered in this field.

- **Duration Field**: Use this field to specify the time it takes for the sound to pan from its current position in the stereo field to the position specified by the Left/Right slider or Balance field. This value should be entered in `minute:second.millisecond` format. The default, 00:00.00 means that the pan position changes instantaneously.

According to Element's Position Radio Button
Select this radio button to deactivate the Left/Right, Balance, and Duration controls and instead specify the pan position of the sound according to the element's left/right position in the scene. When the element is positioned at the left edge of the scene, its sound is panned full left. When the element is positioned at the right edge of the scene, its sound is panned full right. The panning changes smoothly as the element is positioned between the two extremes.

Hot Tip

Use this option in conjunction with the drag motion modifier (page 12.28) to create video or sound elements whose sound panning changes in real time as they are moved about the screen by the user. Note that sounds, since they don't have a physical representation in the layout view, inherit "position" information from the element that is their parent.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

STRING VARIABLE

String variable modifiers hold ASCII text values. String variables can hold up to 32K bytes of text. They are useful for storing user names, and other text values that may change during runtime.

Components of the string variable dialog (Figure 12.44) are described below:

Variable's Name Field
This editable text field can be used to change the variable's name. If the variable will be referenced in a Miniscript modifier, it is good programming practice to make the name a single word.

Modifier Icon
This icon identifies the variable modifier's type.

Value Text Field
Enter the variable's initial value here. The value is checked for validity when the OK button is clicked. The value can be changed during runtime. See "Setting Values of Variable Modifiers" on page 14.5 for details regarding getting and setting the value of this variable modifier.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

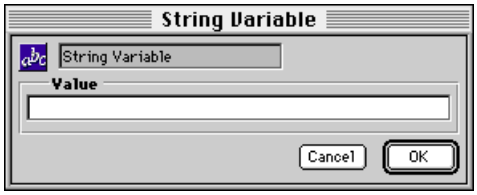


Figure 12.44 The String Variable dialog

TEXT STYLE MODIFIER

The text style modifier can be used to change the settings of all text in a text element. Like the graphic modifier, multiple text style modifiers can be added to a text element, but the effects of only one text style modifier can be applied at once.

The default text style of a text element can be changed using the options in the Format menu. See Chapter 4, "Format Menu".

Components of the text style modifier dialog are described below:

Modifier's Name Field
This editable text field can be used to change the modifier's name.

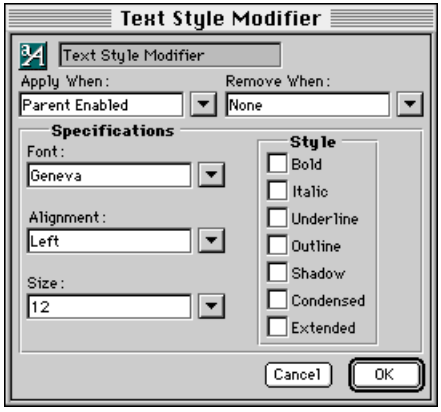


Figure 12.45 The Text Style modifier dialog

Modifier Icon
This icon identifies the modifier's type.

Apply When Pop-Up Menu
Use this pop-up menu to select the message that causes the specified text style to be applied. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Remove When Pop-Up Menu
Use this pop-up menu to select an optional message that causes the text effects to be removed. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Specifications Section

Use the controls in this section to select the text style to be applied.

Font Pop-Up Menu
Use this pop-up to select the font to be applied. All fonts installed on the current system are shown in the menu.

- *Note: When a project is built into a title for distribution, fonts are not bundled into the build file. To display properly, any fonts used by the project must be installed on the target system.*

Alignment Pop-Up Menu
Use this pop-up to select the alignment of the text. Options are Left, Center or Right, relative to the frame of the text element.

Size Pop-Up Menu
Use this pop-up to specify the size of the text, in points.

Text Style Checkboxes
Select text style options, if desired, by selecting any of these checkboxes.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

TIMER MESSENGER

The timer messenger can be used to send a message after a given time has elapsed. The message can be sent once or repeatedly.

Components of the timer messenger dialog (Figure 12.46) are described below:

Modifier's Name Field
This editable text field can be used to change the modifier's name.

Modifier Icon
This icon identifies the modifier's type.

Execute When Pop-Up Menu
Use this pop-up menu to select the message that causes the timer to start. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

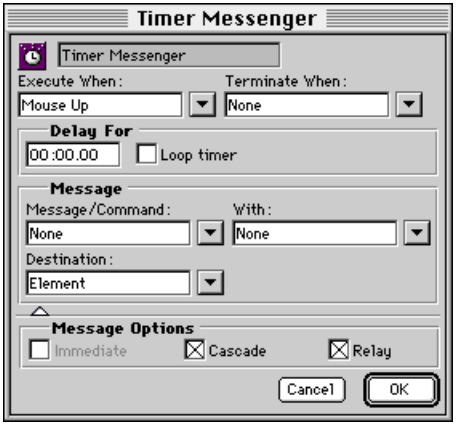


Figure 12.46 The Timer messenger dialog

Terminate When Pop-Up Menu
Use this pop-up menu to select an optional message that causes the timer to stop. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Delay For Section
Enter the duration of the timer in this section's text field using the format `minutes:seconds.hundredths`. The timer starts when it receives the message specified by the "Execute When" pop-up menu. After the specified time has elapsed, the message specified by the "Message" section is sent.

Loop Timer Checkbox
By default, the timer runs once and sends its message once at the end of the "Delay For" time. Select the "Loop timer" checkbox to make the timer restart and send its message repeatedly until the timer is stopped by the message specified in the "Terminate When" pop-up.

Message Specifications
Use this section to specify the message to be sent when the "Delay For" time has elapsed.

See "Configuring Messenger Modifiers" on page 12.10 or "Message Specifications" on page 12.47 for a complete description of the controls in this section.

Message Options
Click the triangle at the bottom of the dialog to display the Message Options section. See "Message Options" on page 12.11 or "Mes-

sage Options" on page 12.48 for a complete description of the controls in this section.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

TRACK CONTROL MODIFIER

•Note: In version 1.0 of mTropolis, this modifier is provided in a "beta" version—features and documentation may not be complete. In version 1.0, this modifier has no effect when used in titles built for the Windows platform.

The track control modifier can be used to manipulate the video and audio tracks within a QuickTime movie element. QuickTime movies consist of tracks—individual video and audio elements that can be manipulated separately, or in unison, by this modifier. Tracks can be turned on, turned off, or toggled between the two states using this modifier.

All QuickTime movies have at least one track. However, QuickTime movies can be made that consist of multiple tracks. mTropolis includes a separate Macintosh application called MovieTrax that can be used to create your own multitrack QuickTime movies for use in mTropolis projects. See Appendix A, "MovieTrax", for a description of this application.

Components of the track control modifier dialog (Figure 12.47) are described below.

Modifier's Name Field

This editable text field can be used to change the modifier's name.

Modifier Icon

This icon identifies the modifier's type.

Execute When Pop-Up Menu

Use this pop-up menu to select the message that triggers the track control modifier. For a complete discussion of this menu, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Track Selection Section

Use the controls in this section to specify the track or tracks to be affected by the operation specified in the Track Options section.

All Tracks Radio Button

Select this button to specify all tracks in the movie.

All Visual Radio Button

Select this button to specify all video tracks in the movie.

All Audio Radio Button

Select this button to specify all audio tracks in the movie.

Use Incoming Data Radio Button

Select this button to specify tracks via the data accompanying the message that triggers

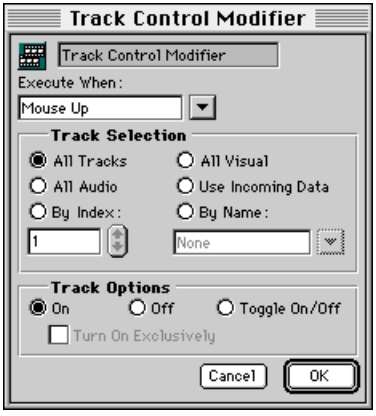


Figure 12.47 The Track Control Modifier dialog

the track control modifier. Acceptable data types are a string (that contains the name of the track to be selected), an integer (that contains the index of the track to be selected), or an integer range (that specifies a range of track indices to be selected).

By Index Radio Button
Select this button to specify a single track by its index number. The By Index field becomes active. Enter an index number in the field or use the arrows to select one. Note that it is possible to select an index that is not actually present in the movie.

By Name Radio Button
Select this button to specify a single track by its name. The By Name pop-up menu becomes active. The names of tracks that have names assigned to them appear as the menu options.

Track Options Section

Use the controls in this section to specify the type of operation to perform on the selected track(s).

On Radio Button
Select this button to enable the selected tracks (i.e., turn them "on").

Off Radio Button
Select this button to disable the selected tracks (i.e., turn them "off").

Toggle On/Off Radio Button
Select this button to enable any selected tracks that are currently disabled and disable any selected tracks that are currently enabled.

Turn On Exclusively Checkbox
When the "On" radio button is selected, this checkbox can also be selected to turn *off* all QuickTime tracks except for the ones the modifier is configured to turn on.

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

VECTOR VARIABLE

The vector variable modifier stores angle and magnitude values for use with the vector motion modifier (see "Vector Motion Modifier" on page 12.83).

Variable's Name Field

This editable text field can be used to change the variable's name. If the variable will be referenced in a Miniscript modifier, it is good programming practice to make the name a single word.

Modifier Icon

This icon identifies the variable modifier's type.

Value Fields

Enter the variable's initial values here or use the up/down arrow buttons to set the values by 0.5 increments. There are two fields:

- **Angle**: This is the vector's angle, measured in degrees counter-clockwise from horizontal (e.g., 0 degrees is at "3 o'clock", 90 degrees is at "12 o'clock").
- **Magnitude**: This is the vector's magnitude. The vector motion modifier interprets this magnitude as velocity in inches per second.

The value is checked for validity when the OK button is clicked. The value can be changed during runtime. See "Setting Values of Variable Modifiers" on page 14.5 for details regarding getting and setting the value of this variable modifier.

Cancel Button

Click Cancel to ignore changes made to this modifier.

OK Button

Click OK to accept changes made to this modifier.

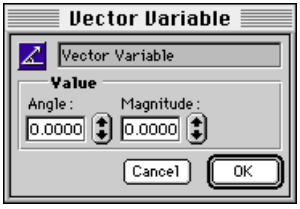


Figure 12.48 The Vector Variable dialog

VECTOR MOTION MODIFIER

The vector motion modifier applies a motion path to an element using the angle and magnitude values specified in a vector variable (see “Vector Variable” on page 12.82).

The vector motion of the element is tied to the specified vector variable. If the values in the vector variable are changed, the motion of the element changes to match the new values. The values of the vector variable modifier may be changed during runtime, providing many possibilities for an object’s vector motion. For example, a graphic element could be configured to move in a straight line toward the user’s cursor and then change direction as the mouse’s location changes.

Components of the vector motion modifier dialog (Figure 12.49) are described below:

Modifier’s Name Field
This editable text field can be used to change the modifier’s name.

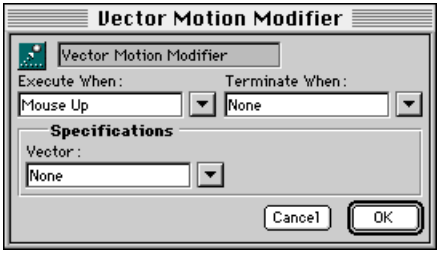


Figure 12.49 The Vector Motion modifier dialog

Modifier Icon
This icon identifies the modifier’s type.

Execute When Pop-Up Menu
Use this pop-up menu to select the message that initiates the vector motion. For a complete discussion of this menu, see “When Pop-Up and Message/Command Pop-Up Options” on page 13.5.

Terminate When Pop-Up Menu
Use this pop-up menu to select the message that terminates the vector motion. For a complete discussion of this menu, see “When Pop-Up and Message/Command Pop-Up Options” on page 13.5.

Vector Pop-Up Menu
Use this pop-up to select a previously defined vector variable as the model for the vector motion. Any variables currently available to the vector motion modifier from its position in the project’s hierarchy will be visible in this pop-up (see “Variable Scopes” on page 13.25).

- *Note: A vector variable must be selected in this pop-up for any vector motion to happen.*

Cancel Button
Click Cancel to ignore changes made to this modifier.

OK Button
Click OK to accept changes made to this modifier.

Chapter 13. Modifier Pop-Up Menus and Message Reference

This chapter describes menus that are common to many of mTropolis' modifier dialogs. These menus include:

- **When Pop-Up Menu**: The menu used to select the message to apply/execute/enable or remove/terminate/disable the effects of a modifier. See "The 'When' Pop-Up Menu" on page 13.1.
- **Message/Command Pop-Up Menu**: The menu used to select the message or command to be sent from messengers. See "The Message/Command Pop–Up Menu" on page 13.2.
- **With Pop-Up Menu**: The menu used to specify the value to be sent with incoming messages or commands. See "The "With" Pop-Up Menu" on page 13.20.
- **Destination Pop-Up Menu**: The menu used to specify the destination for the message or command to be sent from messengers. See "The Destination Pop-Up Menu" on page 13.21.

The following topics are also covered in this chapter:

- "mTropolis Messages and Commands" on page 13.2.
- "Message Paths" on page 13.23.
- "Variable Scopes" on page 13.25

The settings specific to a modifier's capabilities are documented in the previous chapter, Chapter 12, "Modifier Reference".

THE 'WHEN' POP-UP MENU

Most modifier dialogs contain pop-up menus used to specify the messages that activate (or sometimes, deactivate) the modifier when they are received. These menus are referred to as When pop-up menus (Figure 13.1).

Each When pop-up in a modifier's dialog is labeled with a phrase that describes the action of the modifier when the selected message is received. This phrase helps to clarify the use of a When pop-up for each particular modifier. For example, "Execute When", "Terminate When", and "Apply When" are common When menu labels.

- *Note: Variable modifiers do not have When pop-ups as the value contained within the variable automatically becomes a part of the component on which it is placed.*

All When menus contain the same options. Types of messages are described in "mTropolis Messages and Commands" on page 13.2. For information about individual items in the When pop-up, see "When Pop-Up and Mes-

sage/Command Pop-Up Options" on page 13.5.

THE MESSAGE/COMMAND POP–UP MENU

Messenger modifier dialogs have a Message Specifications section, used to select the message to be sent when the messenger is activated, any extra data sent with the message, and the destination of the message.

The first menu in this section is the Message/Command pop-up menu (Figure 13.2). This menu lists all of the possible messages and commands that can be sent to elements in the project. They types of messages and commands that can be sent are described in the next section. For information about individual items in the When pop-up, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

mTROPOLIS MESSAGES AND COMMANDS

There are three types of options available in the Message/Command menu: Environment messages, author messages, and commands. The When pop-up menu displays both author messages and environment messages that can be used to activate or deactivate modifiers. The three types of messages and commands are described below.

Environment Messages

In runtime mode, changes in the mTropolis environment, such as user mouse clicks, transitions, or scene changes, are detected by mTropolis' engine and reported to components of a project as messages. These messages are called *environment messages*. Environment messages are said to be generat-

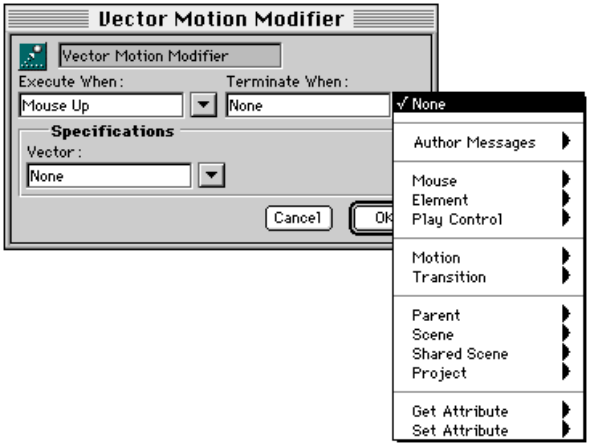


Figure 13.1 Typical mTropolis modifier dialog with When menus

ed “by mTropolis” or “by the mTropolis environment”.

Environment messages can be used to activate or deactivate modifiers by selecting them from the When pop-up menu. In this way, runtime events can be made to trigger modifiers. For example, the image effect modifier on the blue element in Figure 13.3 is configured to be applied by the environment message Mouse Down and removed by the environment message Mouse Up.

Sending Environment Messages from Messengers

In addition to being generated by runtime events, environment messages can also be sent from author-configured messengers. Select an environment message from the Message/Command pop-up menu to send an environment message to a specific element, just as if the runtime event associated with that message had occurred.

For example, consider the “Blue Square” graphic element in Figure 13.3. Its has an image effect modifier assigned to it. This modifier will be applied when the element receives a Mouse Down message. Therefore, the effect will be applied in runtime mode when a user starts to click on the element with the mouse. However, the image effect could also be triggered (without the user pressing the mouse button) by configuring a messenger to send a Mouse Down message to the Blue Square element.

Figure 13.2 Typical messenger dialog with Message/Command menu

This ability to send any environment message from a messenger gives the author flexible control over objects in a mTropolis project.

For information about individual environment messages, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

Author Messages

A message that has been created by the author of a mTropolis project is called an *author message*. Author messages are never sent by the mTropolis environment, they are only sent by messengers configured by the author. As with any other message, author messages can be used to activate or deactivate modifiers.

Previously-created author messages can be sent by selecting the message from the Author Messages submenu of the Message/Command pop-up. New author messages can be created simply by highlighting the text in the When or Message/Command text fields and entering new text. A dialog appears asking if you want to create the new author message. Once created, the new author message appears in the When menus of all modifiers and the Message/Command menu of all messengers.

Because modifiers respond to strings of words but not their meaning, the text of an author message is irrelevant. As long as the message that is sent to a modifier is the same as the message it is configured to receive, the modifier will be activated. For example, a change scene modifier could be configured to respond to the author message "Jump up!" When this message is received the change scene modifier activates and changes to the next scene in the project.

Commands

In addition to environment and author messages, the Message/Command menu allows the selection of *commands*. Commands are im-

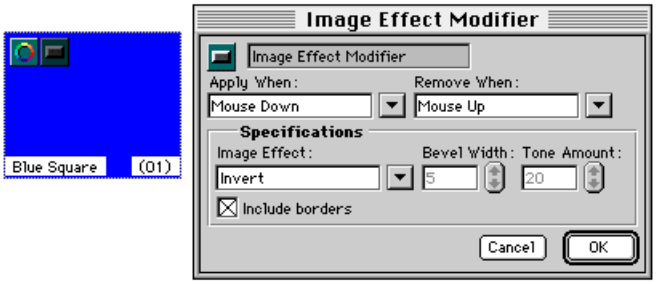


Figure 13.3 Activating and deactivating a modifier with environment messages.

perative instructions that are acted on by elements immediately upon receipt.

Commands are different from messages in the following ways:

- A command constitutes a demand that the recipient perform some action, and it cannot be ignored. Commands are primarily used to control elements directly. For example, sending the *Play* command to an element causes any time-sensitive media (e.g., a QuickTime movie) linked to the element to start playing.
- Unlike messages, commands are not used to activate modifiers and are therefore not available in the When pop-up menu.
- Like author messages, commands are never sent from the mTropolis environment. They are always generated by author-configured messengers.
- Unlike either type of message, commands are sent to and acted on by a single, "targeted" destination. Commands do not "cascade" or "relay" through the scene.
- Although commands are targeted only at a single destination, some commands have associated messages that are generated by mTropolis when an element has responded to a command. These messages report the new state of an element. For example, when an element is sent the *Play* command, mTropolis generates a Played message that is sent to the element and its modifiers. These "message/command pairs" are described in "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.
- Command names are shown in *italic* type—in both the Message/Command menu and this manual—to differentiate them from messages.

For information about individual commands, see "When Pop-Up and Message/Command Pop-Up Options" on page 13.5.

WHEN POP-UP AND MESSAGE/ COMMAND POP-UP OPTIONS

The When and Message/Command menus are organized into a number of submenus that open to reveal the names of specific message and commands that can be sent. Each of the following sections details a different submenu:

• "Get and Set Attribute Commands" on page 13.18

AUTHOR MESSAGES

Previously-defined author messages can be selected from the Author Messages submenu. To create a new author message, highlight the text in the Message/Command text field and enter the name of the new message. A prompt appears, asking if you want to create the new author message. Author messages can also be created in the Author Messages Window (see "Author Messages Window" on page 7.5).

How mTropolis Handles Author Messages

By default, author messages sent from messengers "cascade" from the targeted destination to every component below it in the project's hierarchy. This feature allows many modifiers to be activated by a single message.

For example, a messenger could be configured to send an author message called "SetSpeed" with a value of 10 to a scene. Messengers on mToons in the scene can be configured to respond to this author message by setting the frame rate of their animations with the incoming data. When the author message "Set Speed" and the value 10 is received by the scene, it is automatically broadcast from the scene to its elements and modifiers, causing the messengers on the mToons to collectively set the frame rates of their animations to 10.

Messages can be more precisely targeted by altering the default message options in the messenger dialog. See "Message Options" on page 13.24 or page 12.11.

Using Author Messages in the When Pop-Up

Select an author message from this menu to specify the message that activates or deactivates the modifier.

Using Author Messages in the Message/Command Pop-Up

Select an author message to be sent to a targeted destination from this pop-up.

MOUSE MESSAGES

Choose Mouse in the When pop-up or the Message/Command pop-up to reveal the submenu options related to this item (Figure 13.4).

How mTropolis Generates Mouse Messages

During runtime, mTropolis sends mouse messages in response to a user's mouse actions. When the mouse action is detected, the appropriate messages are sent to the element under the mouse cursor and passed to its modifiers.

If elements that are layered over each other are each configured with modifiers to respond to the same mouse messages, only the element closest to the viewer (i.e., the element with the highest layer order number) under the user's mouse receives mouse messages from mTropolis.

Hidden elements neither receive nor do they act on mouse messages sent by mTropolis. However, modifiers on a hidden element that act on mouse messages can be triggered by sending mouse messages to the hidden element from a messenger modifier.

Using Mouse Messages with the When Pop-Up

Select a mouse message in a When menu to activate or deactivate the modifier when the message is received, either from mTropolis or from a messenger.

Using Mouse Messages with the Message/Command Pop-Up

Select a mouse message in a Message/Command menu to send the mouse message to a targeted destination as if the message had been generated by the user's mouse actions. Note that sending one of these messages does not actually cause the mouse cursor (or the mouse itself!) to move.

Mouse Message Descriptions

Mouse message options found in the When and Message/Command pop-up menus are described below:

Mouse Down

mTropolis sends a Mouse Down message to an element when the mouse button is pressed down and the mouse cursor is over that element.

Mouse Up

mTropolis sends a Mouse Up message to an element when the mouse button has been pressed down over that element and then released anywhere.

Mouse Up Inside

mTropolis sends a Mouse Up Inside message to an element when the mouse button has been pressed down over that element and then released *inside the frame* of that element.

Mouse Up Outside

mTropolis sends a Mouse Up Outside message to an element when the mouse button has been pressed down over that element and then released *outside the frame* of that element.

Mouse Over

mTropolis sends the Mouse Over message to an element when the mouse cursor passes over the frame of that element in an inward direction.

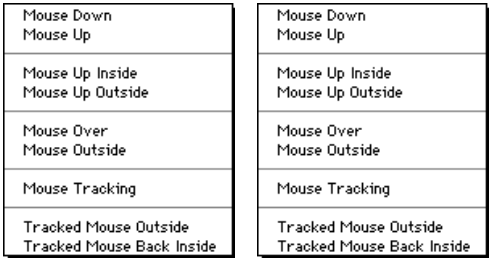


Figure 13.4 Mouse messages in the When and Message/Command pop-up menus

Mouse Outside

mTropolis sends the Mouse Outside message to an element when the mouse cursor leaves the frame of an element.

Mouse Tracking

mTropolis sends the Mouse Tracking message to an element repeatedly after the mouse button has been pressed over that element and is being held down and moved (i.e., the mouse cursor is being dragged) over the element. The message is sent until the mouse button is released.

Tracked Mouse Outside

mTropolis sends the Tracked Mouse Outside message to an element when Mouse Tracking has started (i.e., after the user has started a drag motion over the element), and the mouse cursor is dragged outside the frame of that element.

Tracked Mouse Back Inside

mTropolis sends the Tracked Mouse Back Inside message to an element after the Tracked Mouse Outside condition has been met (i.e., after the user has started a drag motion over the element, but continued dragging outside the element's frame) and the mouse cursor is dragged back inside that element's frame.

ELEMENT MESSAGES AND COMMANDS

Choose Element in the When or Message/Command menu to reveal the submenu items described below (Figure 13.5).

How mTropolis Handles Element Messages and Commands

Most of the Element submenu items in the Message/Command menu are commands. Commands are not sent by mTropolis, they can only be sent from messengers.

With the exception of Edit Done, the element environment messages listed in the When pop-up are only sent by mTropolis when it detects an element that has responded to a command. mTropolis notifies the element of

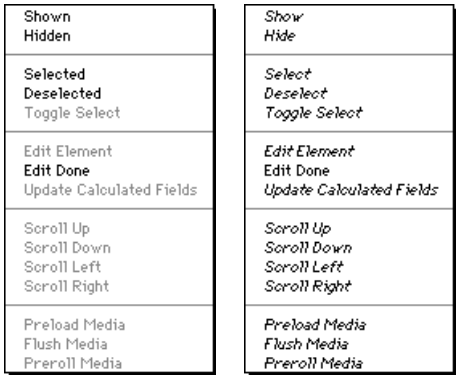


Figure 13.5 Element messages in the When and Message/Command pop-ups

its changed state by sending the environment message to the element and its modifiers.

mTropolis sends the Edit Done environment message to an element and its modifiers when the user has clicked off of an editable text element during runtime. Send the Edit Done message to an element to make it act just as if the message had been generated by the user's mouse actions.

Using Element Messages with the When Pop-Up
Select an element message in a When menu to activate or deactivate the modifier when the message is received, either from mTropolis or from a messenger.

Using Element Messages and Commands with the Message/Command Pop-Up
With the exception of Edit Done, every item on the Message/Command pop-up's Element submenu is a command. Select an element message or a command to send to a targeted destination from this menu.

Element Option Descriptions

The following element-related options are available in the When and Message/Command windows. Note that some options are available in only one menu. Other options are described as related message/command pairs. The message, found in the When menu (e.g., "Shown"), is described first, followed by the command (e.g., "*Show*"), found in the Message/Command menu, that (when received by an element) causes the message to be generated.

Shown/Show
mTropolis sends a Shown message to an element when it becomes shown (in response to a *Show* command). Send the *Show* command to a hidden element to make it visible.

Hidden/Hide
mTropolis sends a Hidden message to an element when it becomes hidden (in response to a *Hide* command). Send the *Hide* command to a visible element to make it invisible (and unresponsive to user mouse events).

Selected/Select
mTropolis sends a Selected message to an element when it receives a *Select* command. Send the *Select* command to an element to set some author-defined state to "on". *Select* commands can only be sent by messengers—the mTropolis environment never sends *Select*.

- *Note: The Select command has nothing to do with user mouse actions. See "Mouse Messages" on page 13.6 for messages sent in response to user mouse motions and clicks.*

Deselected/Deselect
mTropolis sends a Deselected message to an element when it receives a *Deselect* command. Send the *Deselect* command to an element to set some author-defined state to "off". *Deselect* commands can only be sent by messengers—the mTropolis environment never sends *Deselect*.

- *Note: The Deselect command has nothing to do with user mouse actions. See "Mouse Messages" on page 13.6 for messages sent in response to user mouse motions and clicks.*

Toggle Select (Message/Command Menu Only)
Send the *Toggle Select* command to an element that has previously received the *Select* or *Deselect* commands to reverse the element's current selection state. When a "selected" element receives *Toggle Select*, mTropolis sends a Deselected message to the element. When a "deselected" element receives Toggle Select, mTropolis sends a *Select* message to the element.

Edit Element (Message/Command Menu Only)
The *Edit Element* command can be used to control the "editability" of text elements. Send *Edit Element* to a non-editable text element to make it editable (i.e., text in the element can be changed by users during runtime).

Edit Done
mTropolis sends the Edit Done message to an editable text element when the user finishes text entry by clicking outside the text element. Edit Done can also be sent from a messenger to make the text element respond just as if the user had finished text entry, making the text element non-editable.

Update Calculated Fields (Message/Command Menu Only)
Send the *Update Calculated Fields* command to a text field that contains a variable (see "Displaying Variables in Text Fields" on page 4.2) to update the field so that it shows the current value contained in the variable.

Scroll Up (Message/Command Menu Only)
Send the *Scroll Up* command to move the content of an element up within its frame. Specify the amount in pixels that the content is to move by sending a value using the "With" field. As the content of the element moves, it will be cropped by the frame of the element.

Scroll Down (Message/Command Menu Only)
Send the *Scroll Down* command to move the content of an element down within its frame. Specify the amount in pixels that the content is to move by sending a value using the "With" field. As the content of the element moves, it will be cropped by the frame of the element.

Scroll Left (Message/Command Menu Only)
Send the *Scroll Left* command to move the content of an element left within its frame. Specify the amount in pixels that the content is to move by sending a value using the "With" field. As the content of the element moves, it will be cropped by the frame of the element.

Scroll Right (Message/Command Menu Only)
Send the *Scroll Right* command to move the content of an element right within its frame. Specify the amount in pixels that the content is to move by sending a value using the "With" field. As the content of the element moves, it will be cropped by the frame of the element.

Preload Media (Message/Command Menu Only)
Send the *Preload Media* command to an element to load that element's associated media file into system memory (RAM) for optimal playback. This command can only be sent to elements that contain mToons, video, and audio media. The *Flush Media* command can

be used to remove preloaded media from memory.

- *Note: This option is limited by the memory available to the project at runtime. Use caution when sending this command as exceeding the available memory on machines that play your title can cause unpredictable behavior, including program crashes. Pay attention to the size of media you are attempting to preload. For example, it is not a good idea to attempt to preload a 40 megabyte QuickTime video.*

Flush Media (Message/Command Menu Only)
Send the *Flush Media* command to an element to flush its associated media from memory. This command is useful for removing media that was previously loaded into memory using the *Preload Media* command.

Preroll Media (Message/Command Menu Only)
Send the *Preroll Media* command to an mToon, video, or sound element to perform caching and initialization of the media so that it can respond more quickly to the next *Play* command. This command is especially useful with sound elements which usually must take time to load from disk into a memory buffer before they actually begin playing. Note that once the media has been played, it is no longer "prerolled".

PLAY CONTROL MESSAGES AND COMMANDS

The following section describes the items available in the Play Control submenu of the When pop-up and Message/Command pop-ups (Figure 13.6). These items are related to "playing" the contents of an element.

How mTropolis Handles Play Control Messages and Commands

Most of the items in the Play Control submenu are message/command pairs. With the exception of At First Cel and At Last Cel, the Play Control environment messages listed on the When pop-up are only sent by mTropolis when it detects an element that has responded to a corresponding Play Control command. Commands are not sent by mTropolis, they can only be sent from messengers.

mTropolis sends At First Cel or At Last Cel environment messages to an mToon element

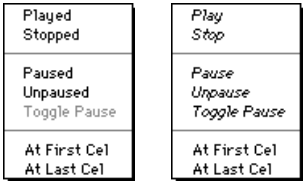


Figure 13.6 Play control messages in the When and Message/Command pop-up menus

and its modifiers when the animation reaches its first or its last cel.

Using Play Control Messages with the When Pop-Up
Select a Play Control message in a When menu to activate or deactivate the modifier when the message is received, either from mTropolis or from a messenger.

Using Play Control Messages and Commands with the Message/Command Pop-Up
Select a Play Control message or command in a Message/Command menu to send the message or command to a targeted destination.

Play Control Option Descriptions

Play control options found in the When and Message/Command pop-up menus are described below. Note that many of these options are described as related message/command pairs. The message, found in the When menu (e.g., "Played"), is described first, followed by the command (e.g., "*Play*"), found in the Message/Command menu, that (when received by an element) causes the message to be generated.

Played/Play
mTropolis sends a Played message to an element when it starts playing (in response to a *Play* command). Send the *Play* command to an element to show the element (if it is hidden) and start playing that element's associated media.

- *Note: Though not every element has media associated with it (e.g., text elements), Play can be sent to any element type, causing the element to be shown (if it is hidden) and receive a Played message.*

Stopped/Stop
mTropolis sends a Stopped message to an element when it stops playing (in response to a *Stop* command). Send the *Stop* command to an element to stop playing that element's associated media and hide the element. To "stop" an element from playing without hiding it, use the *Pause* command.

- *Note: Though not every element has media associated with it (e.g., text elements), Stop can be sent to any element type, causing the element to be hidden and receive a Stopped message.*

Paused/Pause
mTropolis sends a Paused message to an element when its media pauses playing (in response to a *Pause* command). Send the *Pause* command to an element to temporarily stop playing that element's media, without hiding the element.

- *Note: Though not every element has media associated with it (e.g., text elements), Pause can be sent to any element type, causing it to receive a Paused message.*

Unpaused/Unpause
mTropolis sends an Unpaused message to an element when it unpauses playing (in response to an *Unpause* command). Send the *Unpause* command to an element to start playing that element's paused media once again.

- *Note: Though not every element has media associated with it (e.g., text elements), Unpause can be sent to any element type, causing it to receive an Unpaused message.*

Toggle Pause (Message/Command Menu Only)
Send the *Toggle Pause* command to an element that has previously received the *Pause* command or is currently playing to reverse the element's current pause state. That is, when a paused element receives *Toggle Select*, mTropolis starts playing its media and sends an Unpaused message to the element. When a currently-playing element receives *Toggle Select,* mTropolis pauses the media and sends a *Paused* message to the element.

At First Cel
mTropolis sends an At First Cel message to an mToon element when the animation reaches its first cel.

This message can also be sent to a targeted element to trigger any modifiers that act on At First Cel just as if the first cel in the animation had been reached. Note, however, that the animation does not actually change to the first cel when this message is received from a messenger.

At Last Cel
mTropolis sends an At Last Cel message to an mToon element and when the animation reaches its last cel.

This message can also be sent to a targeted element to trigger any modifiers that act on At Last Cel just as if the last cel in the animation had been reached. Note, however, that the animation does not actually change to the last cel when this message is received from a messenger.

MOTION AND TRANSITION MESSAGES

Choose Motion or Transition in the When pop-up or the Message/Command pop-up to reveal the submenu options described below (Figure 13.7).

How mTropolis Handles Motion and Transition Messages
These messages are sent by mTropolis when it detects elements or scenes in motion. The motion or transition environment messages are sent to the element and its modifiers.

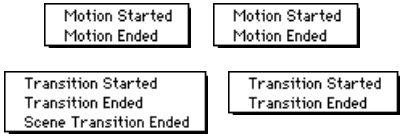


Figure 13.7 Motion and Transition messages in the When and Message/Command pop-up menus

Using Motion and Transition Messages with the When Pop-Up

Select a motion or transition message in a When menu to activate or deactivate the modifier when the message is received, either from mTropolis or from a messenger.

Using Motion and Transition Messages with the Message/Command Pop-Up

Select a motion or transition message in a Message/Command menu to send the message to a targeted destination just as if the mTropolis environment had generated the message. Note, however, that sending one of these messages does not actually make motion or a transition happen.

Motion Started

mTropolis sends a Motion Started message to an element and its modifiers when an element starts moving (e.g., when the user drags the element or the element starts simple, path, or vector motion).

Motion Ended

mTropolis sends a Motion Ended message to an element when its motion ends.

Transition Started

mTropolis sends a Transition Started message to an element when an element transition starts. Note that this message refers only to *element* transitions, not scene transitions (see "Scene Messages" on page 13.15).

Transition Ended

mTropolis sends a Transition Ended message to an element when its element transition ends. Note that this message refers only to *element* transitions, not scene transitions (see "Scene Messages" on page 13.15).

Scene Transition Ended (When Menu Only)

mTropolis sends a Scene Transition Ended message to a scene after the scene has begun (i.e., after the Scene Started message) and any scene transition assigned to that scene (see "Scene Transition Modifier" on page 12.63) has finished playing. This message is useful because many scene transitions, such as "fade", create scenes that are not yet visible when the Scene Started message is sent. Note that this message is always sent, even if the scene does not contain a scene transition modifier. In this case, Scene Started is sent to the scene followed immediately by Scene Transition Ended.

PARENT MESSAGES

Choose Parent in the When pop-up or the Message/Command pop-up to reveal the submenu options described below (Figure 13.8).

How mTropolis Handles Parent Messages

mTropolis sends the Parent Enabled message to the scene when the scene starts. Unlike most environment messages, this message

Figure 13.8 Parent messages in the When and Message/Command menus

cascades through the scene, activating *all* modifiers and behaviors in a scene that are waiting for this message when the scene begins.

The Parent Enabled message is also sent to the components of a switchable behavior modifier when the behavior is switched on (i.e., when it receives the message specified in its Enable When field—see "Behavior" on page 12.13).

The Parent Disabled message is sent to the scene when the scene ends (by changing to another scene, exiting a mTropolis runtime application, or re-entering edit mode from runtime mode in the editor). This message also cascades through the scene, activating *all* modifiers and behaviors in a scene that are waiting for this message when the scene ends.

The Parent Disabled message is also sent to the components of a switchable behavior modifier when the behavior is switched off (i.e., when it receives the message specified in its Disable When field—see "Behavior" on page 12.13).

Using Parent Messages with the When Pop-Up
Select a parent message in a When menu to activate or deactivate the modifier when the message is received, either from mTropolis or from a messenger.

Using Parent Messages with the Message/Command Pop-Up
Select a parent message in a Message/Command menu to send the message to a targeted destination just as if the mTropolis environment had generated the message. Note, however, that sending one of these messages does not actually enable or disable the recipient's parent.

Parent Enabled
mTropolis sends this message to an element when its parent is enabled. Elements are enabled when the scene starts; behaviors are enabled when they receive the message specified in their Enable When field.

Parent Disabled
mTropolis sends this message to an element when its parent is disabled. Elements are disabled when the scene ends; behaviors are disabled when they receive the message specified in their Disable When field.

SCENE MESSAGES

Choose Scene in the When pop-up or the Message/Command pop-up to reveal the submenu options described below (Figure 13.9).

How mTropolis Handles Scene Messages
When navigating a multimedia title, actions often take place at the beginning and end of a

Figure 13.9 Scene messages in the When and Message/Command pop-up menus

scene. The scene messages are related to this functionality.

mTropolis sends the scene messages described below to the scene, its child elements and every modifier and behavior that has been placed on them.

Using Scene Messages with the When Pop-Up
Select a scene message in a When menu to activate or deactivate the modifier when the message is received, either from mTropolis or from a messenger.

Using Scene Messages with the Message/Command Pop-Up
Select a scene message in a Message/Command menu to send the message to a targeted destination just as if the mTropolis environment had generated the message. Note, however, that sending one of these messages does not actually cause the current scene to start, end, deactivate, or reactivate.

Scene Started
mTropolis sends a Scene Started message to all elements of the scene, including the scene itself, the moment a scene begins.

Scene Ended
mTropolis sends a Scene Ended message to all elements of the scene, including the scene itself, when the scene ends (e.g., when a change scene modifier is executed).

Scene Deactivated
mTropolis sends this message to the scene and all of its elements when a change scene or navigation modifier is executed with the Add to Destination Scene checkbox selected (see "Change Scene Modifier" on page 12.20). This message is sent in lieu of the Scene Ended message.

Scene Reactivated
mTropolis sends this message to the scene when returning to that scene (via the return modifier—see "Return Modifier" on page 12.60) and that scene was previously changed *from* (via the change scene or navigation modifier) with the Add to Destination Scene checkbox selected. In other words, when the scene is "reactivated" after having a new active scene "layered" over it.

SHARED SCENE MESSAGES

Choose Shared Scene in the When pop-up or the Message/Command pop-up to reveal the submenu options described below (Figure 13.10).

How mTropolis Handles Shared Scene Messages
Unlike most environment messages, mTropolis sends shared scene messages to the

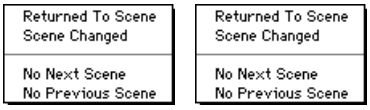


Figure 13.10 Shared scene messages in the When and Message/Command pop-up menus

shared scene, its child elements, and every modifier that has been placed on them.

Using Shared Scene Messages with the When Pop-Up

Select a shared scene message in a When menu to activate or deactivate the modifier when the message is received, either from mTropolis or from a messenger.

Using Shared Scene Messages with the Message/Command Pop-Up

Select a shared scene message in a Message/Command menu to send the message to a targeted destination just as if the mTropolis environment had generated the message. Note, however, that sending one of these messages does not actually cause the properties of the shared scene to change.

Returned To Scene

mTropolis sends a Returned to Scene message to the shared scene and its elements and modifiers when a return modifier is executed in the project during runtime.

Scene Changed

mTropolis sends a Scene Changed message to the shared scene, its elements and modifiers at the start of each scene during runtime.

No Next Scene

mTropolis sends a No Next Scene message to the shared scene, its elements and modifiers when a scene begins and that scene has no scenes following it (i.e., the scene is the last in the scene order of its subsection).

No Previous Scene

mTropolis sends a No Previous Scene message to the shared scene, its elements and modifiers when a scene begins and that scene has no scenes before it (i.e., the scene is the first in the scene order of its subsection).

Hot Tip

No Next Scene and No Previous Scene can be used to show and hide scene navigation buttons located in the shared scene.

PROJECT MESSAGES

Choose Project in the When pop-up or the Message/Command pop-up to reveal the submenu options described below (Figure 13.11).

How mTropolis Handles Project Messages and Commands

Project messages and commands relate to an entire project:

- The *Close Project* command can be sent to the project to quit a mTropolis project in runtime.

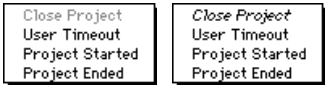


Figure 13.11 Project messages in the When and Message/Command pop-up menus

- The User Timeout message is generated when there has been no user activity for a specified time. The timeout duration can be changed using the *Set Attribute* command (see "Get and Set Attribute Commands" on page 13.18).
- The Project Started and Project Ended messages are sent to the project component when a project begins or ends.

Using Project Messages with the When Pop-Up
Select a project message in a When menu to activate or deactivate the modifier when the message is received, either from mTropolis or from a messenger.

Using Project Messages with the Message/Command Pop-Up
Select a project message in a Message/Command menu to send the message to a targeted destination just as if the mTropolis environment had generated the message.

Close Project (Message/Command Menu Only)
Send the *Close Project* command to the project to quit a mTropolis runtime project.

User Timeout
mTropolis sends a User Timeout message to the shared scene, its elements and modifiers when there have been no user actions for a specified time. The timeout duration can be changed using the *Set Attribute* command (see "Get and Set Attribute Commands" on page 13.18).

Project Started
mTropolis sends a Project Started message to the project component, and all of its modifiers, when the project starts. This message does not cascade any further through the project.

Project Ended
mTropolis sends a Project Ended message to the project component, and all of its modifiers, when the project ends (e.g., when the user quits the title). This message does not cascade any further through the project.

GET AND SET ATTRIBUTE COMMANDS

Choose Get Attribute or Set Attribute in the Message/Command pop-up to reveal the submenu options described below. All of these options are commands; they are not available in the When menu. Commands are not sent by mTropolis, they can only be sent from messengers.

Element Attributes

Every element in a project has inherent attributes that allow the media to which the element is linked to be displayed or controlled. These are accessible via the *Get Attribute* and *Set Attribute* commands listed in the Message/Command pop-up. Note that the Miniscript modifier can also be used to get and set attributes, including advanced attributes that cannot be selected from the Message/Command menu. See "Setting Values of Element Attributes" on page 14.6.

The items listed in the Get Attributes and Set Attributes submenus vary according to the media type associated with the element on which the messenger is placed. For example, a messenger placed on a PICT has access to the

graphic element attributes such as height, width, position and layer order number.

Some set operations require a value to be sent with a command. For example, the *Set width* command requires a value, in pixels, to be used as the new element width. Use the With menu (see "The "With" Pop-Up Menu" on page 13.20) to specify the value to be sent.

- *Note: When using these commands, use the correct variable type to contain the value you are getting or setting. For example, use a point variable to set the (x, y) position of an element.*

The *Get Attributes* and *Set Attributes* commands can also be used in more complex operations. For example, these commands can be used with variables to store and retrieve the attributes of elements that change over time. For example, a messenger on a draggable element can be configured to get its position when it receives a Scene Started message. The position of the element can be stored in a point variable, and later accessed by a second messenger that is configured to reset the element's position upon receipt of a Scene Ended message. During runtime, regardless of where the user has dragged the element, it will be returned to its original location when the scene ends.

Get and Set Attribute Option Descriptions

The following Get Attribute and Set Attribute options are available in the Message/Command menu. Note that different options are available depending on the type of media associated with the element being modified by the messenger.

The figures below show the Get Attribute submenu options that are available for graphic elements (e.g., PICTs), text elements, video elements (e.g., QuickTime), mToons, sound elements, and projects. The Set Attribute submenu options are identical to the options shown below, except that the word "Set" replaces the word "Get" in the command name.

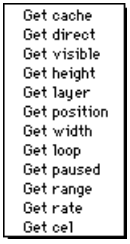
Information on individual attributes can be found in "Element Attributes" on page 14.17.

Get/Set Attribute Options by Media Type

These options are available for graphic elements such as PICTs:

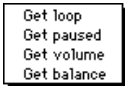
These options are available when targeting a text element:

These options are available when targeting an mToon:

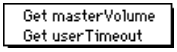


These options are available when targeting a QuickTime movie:

These options are available when targeting a sound element:



The following options are available when a messenger targets the project component itself. Such a modifier can only be placed in the structure view:



THE "WITH" POP-UP MENU

Use the With pop-up menu to select a value to be sent with the message or command selected in the Message/Command menu.

The choices on this pop-up are None, Incoming Data and any variables to which the element has access. A data value can be sent by highlighting the With text field and typing in a value. These menu items are described in more detail below:

- **None**: This is the default selection. No value is sent with the message.
- **Incoming Data**: Choose this option to configure the messenger to send the incoming data (i.e., the value that arrives with the message that has been configured to trigger the messenger), with the outgoing message or command.
- **Variables**: To send the value of a variable modifier with the outgoing message, select its name from the submenus available in the second section of the With pop-up menu. All variable modifiers currently available to a messenger from its position in the project's hierarchy are shown. These variable modifiers are only be accessible to those modifiers or Miniscripts in its "scope." See "Variable Scopes" on page 13.25.
- **Constant Data Value**: To send a constant data value, highlight the With text field and type the value to be sent. The value should be entered with the same syntax as if it were being used in a Miniscript modifier.

The syntax of the expression entered is checked when OK is clicked. Any appropriate mTropolis data type can be sent.

THE DESTINATION POP-UP MENU

Use the Destination pop-up menu to select a destination for the message or command selected in the Message/Command menu.

The default for this menu is “Element”, meaning that the message or command is sent to the element on which the messenger is placed. The destination can be changed by selecting a new option from the menu (Figure 13.12).

Destinations can be chosen by name or by relative position in the project hierarchy. The ability to target a relative rather than an absolute destination for a message allows the elements of a project to be changed without having to reconfigure the messengers that target them. This feature is particularly useful when the components of a project are to be used in other projects. More information about hierarchical relationships between elements can be found in “Message Passing among Elements” on page 8.6.

Destination Option Descriptions

Choose an item from the Destination pop-up to indicate where in the project the message or command is to be sent. Note that, by default, messages are sent not only to the targeted destination, but also to all the components

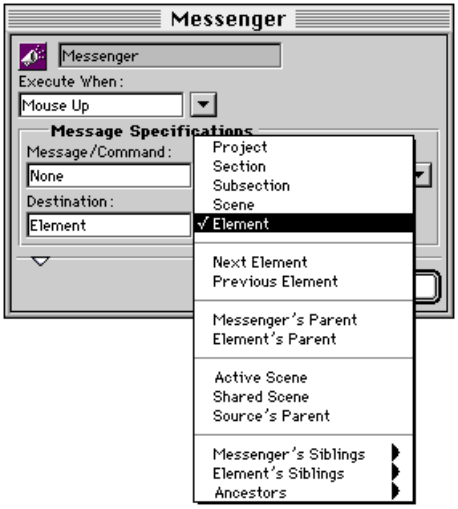


Figure 13.12 Typical messenger dialog with Destination menu

within the targeted destination (i.e., they "cascade" and "relay"). This behavior can be changed using the Message Options section of the messenger dialog (see "Message Options" on page 13.24 or page 12.11).

Destination options are described below. Note that only the default behavior of the message is described:

Project
Messages directed to the project are sent only to modifiers on the project level.

- *Note: Use the structure view to see or place modifiers at the project level.*

Section
The section destination refers to the section on which the messenger lies. Every item contained in the section (i.e., all subsections, scenes, and components of those scenes) will also receive the message. Although possible, sending a Mouse Up, or equally common message, to a section is not recommended. The system will run more efficiently if this type of message is configured to be sent directly to the element to be affected.

Scene
The scene destination refers to the scene of which the messenger is a part. Directing a message to a scene results in all elements on the scene receiving the message, including the scene itself. This destination is useful when multiple elements in a scene need to be sent the same message.

Element
Element is the default selection. The message or command is sent to the messenger's element itself.

Next Element
Choose Next Element to send a message or command to the next sibling of the element associated with the messenger (i.e., the next element in the messaging order of elements in a component).

Previous Element
Choose Previous Element to send a message or command to the previous sibling of the element associated with the messenger (i.e., the previous element in the messaging order of elements in a component).

Messenger's Parent
The parent of the messenger receives the message or command. The parent of a messenger is its "container", whether that container is an element or behavior.

Element's Parent
The parent of the element associated with the messenger receives the message or command.

Active Scene
Choose Active Scene to send a message to the scene visible to the user when the message is sent. By default, all elements and their modifiers on the active scene, as well as the active scene and its modifiers receive the message.

Shared Scene
Choose Shared Scene to send a message to all elements and their modifiers on the shared scene, as well as the scene and its modifiers.

For additional information regarding shared scenes, see “Shared Scenes” on page 10.2.

Source’s Parent
The source’s parent is the parent of the messenger that sent the message that activated the current messenger. This destination is invalid if the messenger is configured to respond to an environment message sent by mTropolis.

Messenger’s Siblings
The names of the messenger’s siblings appear on the Messenger’s Siblings submenu. Target any one of these modifiers by choosing its name from this list.

Element’s Siblings
The names of the siblings of the element associated with the messenger appear on the Element’s Siblings submenu. Target any one of these elements directly by choosing its name from this list.

Ancestors
The names of all ancestors of the messenger (that is, parents further up the structure hierarchy) appear in the Ancestors submenu. Target any of these destinations directly by choosing its name from this list.

MESSAGE PATHS

By default, messages sent from messengers are sent to their target destination and then they continue down through the items that destination contains. In this way, a single message can be used to activate many modifiers in components throughout a project.

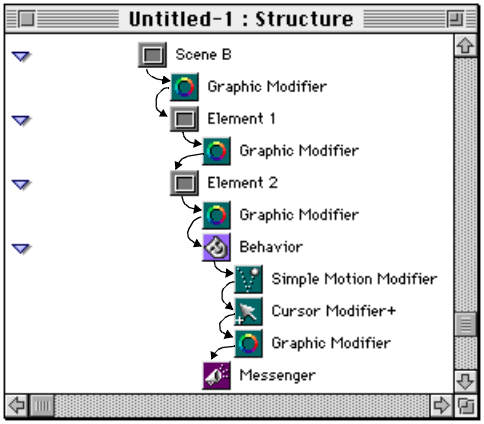


Figure 13.13 Messaging order. The arrows show the path of a cascading and relaying message sent to Scene B.

Figure 13.13 shows a sample scene as it would appear in the structure window. Each element in this window appears on its own level, in a hierarchy from left to right. The descending order of modifiers and components within these components is called their “messaging order.” The arrows trace the default path of a message targeted to Scene B. Notice how the message descends each level of the structure hierarchy and that the message could be activating any of the modifiers in the scene.

A message is said to “Cascade” if it travels through the structure hierarchy from component to component, level to level. A message is said to “Relay” if it continues from modifier to modifier, activating as many modifiers as possible. This path is the default for messages sent from a messenger.

For example, in Figure 13.13, when Scene B receives the message it is relayed to its graphic modifier. From there the message "cascades" to Element 1, the next level in the structure hierarchy. The element then relays the message to Element 1's graphic modifier. The message cascades once again to the next component in the scene, Element 2. The message is then relayed from Element 2 to every remaining modifier in the scene. When the last item in the scene has received the message, its path is complete.

The default path of messages sent by messengers can be changed as described in "Message Options" on page 13.24.

Default Path of Messages Sent by the mTropolis Environment

Note that the path of messages sent to elements from the mTropolis environment (e.g., user mouse click messages) is slightly different than the default for messages sent by messengers. mTropolis-sent environment messages do not cascade, but they do relay. In other words, the message is sent to the target element, activates any modifiers on that element that are set to respond to the message, and then stops.

Message Options

All messenger dialogs have a Message Options section that can be revealed by clicking the triangle at the bottom of the dialog. This dialog can be used to more precisely target and time messages sent from a messenger.

Options in this section are:

Cascade Checkbox

By default, the message "cascades" from the targeted destination to all components in the hierarchy below it. Uncheck this option to configure the messenger to send its message only to the modifiers in a targeted element. "Cascade off" is equivalent to the path of an environment message sent by mTropolis.

- *Note: This option has no effect when sending a command. Commands act only on the targeted element.*

Immediate Checkbox

By default, messages or commands are sent immediately when the messenger is activated. Uncheck this box to configure the messenger to send its message or command only after the current thread of messages has ended. This option can be used to avoid system overload if the message thread queue becomes too deep.

Relay Checkbox

By default, messages travel from element to element in the hierarchy activating any and all elements that respond to the message. Uncheck this box to activate only the *first* modifier in the path of the message that is configured to respond. In effect, the first modifier to respond to the message "swallows" the message.

- *Note: This option has no effect when sending a command. Commands act only on the targeted element.*

VARIABLE SCOPES

Variable modifiers (variables) can be selected from the With menu (see "The "With" Pop-Up Menu" on page 13.20) and referenced in the text of Miniscript modifiers, on the condition the messenger modifier exists in the variable's *scope*.

That is, a variable modifier's scope determines which modifiers can "see" and make use of that variable modifier. The scope of the variable is determined by its position in the project's structural hierarchy.

A variable is accessible to all descendants of its parent.

Illustrating Variable Modifier Scopes

As stated previously, the scope of a variable is determined by its position in the project's structural hierarchy. For example, a variable modifier placed on a section is available to any modifier within that section, and all the section's "descendants". An element's descendants include any objects found anywhere "under" it in the hierarchy, no matter how many layers deep.

A variable modifier whose parent is the project (such as Variable Modifier 1 in Figure 13.14) is said to be *global*. Since all components of a project are its descendants, any variable placed at the project level can be accessed by any appropriate modifier in the entire project.

For example, an integer variable modifier that keeps track of the user's current score would probably need to be updated by many different modifiers throughout the title. Placing the variable on the project ensures that any modifier that may use the variable has access to it.

To create a global variable:

- Use the structure view to display the project's hierarchy.
- Drag a variable modifier from its palette and drop it on the project icon (the topmost icon in the structure hierarchy). For more information on using the structure window, see "Adding Components to a Project in the Structure Window" on page 8.4.

For a more localized scope, variable modifiers can be placed in specific components anywhere in the project.

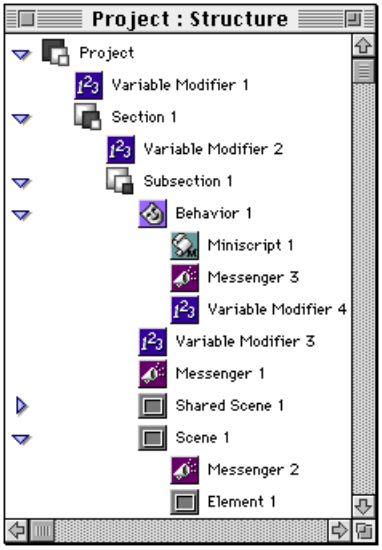


Figure 13.14 Variable scopes

Figure 13.14 shows a number of variables with *local* scopes. For example, following the scoping rule, Messenger 1, Messenger 2 and Miniscript 1 shown in the figure have access to the following variable modifiers:

- Messenger 1 could access Variable Modifier 1, Variable Modifier 2, and Variable Modifier 3.
- Messenger 2 could access Variable Modifier 1, Variable Modifier 2, and Variable Modifier 3.
- Miniscript 1 could access Variable Modifier 1, Variable Modifier 2 and Variable Modifier 4.

To summarize, a variable modifier can be used by any modifier that:

- Makes use of variable modifiers (e.g., messengers, Miniscript), and
- Is a descendant of the variable modifier's parent (i.e., it is within the variable modifier's scope).

Chapter 14. Miniscript Modifier

The Miniscript modifier can be used to access mTropolis' built-in scripting language. Miniscript is a simple scripting language that can be used to:

- Access and change element attributes, such as the screen position of elements.
- Perform mathematical operations.
- Get and set the value of variable modifiers that lie within the scope of the Miniscript modifier.
- Perform comparisons and conditional branching using relational operators and "if" statements.
- Send multiple messages and/or commands to components of a project.

This chapter describes Miniscript syntax, statements, operators, and miscellaneous functions. It also contains lists of reserved words and element attributes that can be retrieved and/or set.

THE MINISCRIPT MODIFIER DIALOG

Miniscript modifiers can be added to elements (and behaviors) just like any other modifier. Simply drag the Miniscript modifier from the modifier palette and drop it on the element to be modified. Double-click the Miniscript modifier on an element to display its dialog.

Figure 14.1 shows a Miniscript modifier dialog with a single-line script entered into it. Components of the Miniscript modifier dialog are described below.

Editable Name Text Box
Like all other mTropolis modifier dialogs, the Miniscript dialog has a text entry field that can be used to give the modifier a unique and descriptive name.

Execute When Pop-Up Menu
Use this pop-up menu to select the message that triggers the modifier. When this message is received, the Miniscript commands in the "Script" text box are executed.

Script Text Box
Use the Script text box to edit the script associated with the modifier. Scripts can have a maximum of 32,768 characters in them. By default, this field is blank. Figure 14.1 shows a simple script that changes the value of a floating-point variable modifier ("myFloat") to 4.555.

OK Button
The script is checked for syntax and then compiled when the OK button is clicked. If mTropolis detects any syntax or other errors during compilation, the error is flagged and the compilation process is aborted. A flashing text cursor is placed in front of the offending statement and an alert dialog appears, describing the error.

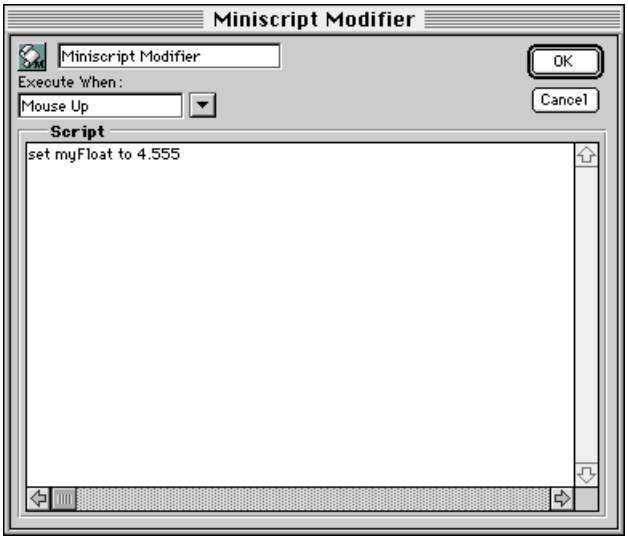


Figure 14.1 The Miniscript modifier dialog

BASIC MINISCRIPT SYNTAX

Miniscript uses a "natural language" syntax that makes the scripting language quick and easy to learn. Writing a script involves using a combination of reserved words, variable names, component names, and literal values (numbers, strings, etc.). An example is the script:

```
set myFloat to 4.555
```

"myFloat" refers to an existing floating-point variable modifier. The literal value 4.555 is the value that the variable modifier will assume.

Basic rules of Miniscript syntax are described below.

Comments

Any text following two dashes (`--`) is considered to be a comment, and is ignored when the script is compiled. Use comments to annotate and clarify scripts. In the following example, the first line is a comment:

```
-- Add two to myFloat:
set myFloat to myFloat+2.0
```

In the next example, the text "-- Initialize X" is a comment:

```
set x to 5 -- Initialize X
```

Case Sensitivity

Miniscript is *not* case sensitive. Miniscript statements can be written in uppercase, lowercase, or mixed case. In this manual, most statements and keywords are written in low-

ercase. To improve readability, element and variable names are sometimes written in mixed case (e.g., `myFloat`). However, since variable names are not case sensitive, the names `myfloat` and `MyFloat` are interpreted as the same by Miniscript.

Note that the text of string values *is* case sensitive (e.g., an uppercase string such as `"MY TEXT"` is not equivalent to the string `"my text"`).

Continuation Character

Each line of a script is a complete command terminated by a return character. To write a single command as a number of lines, add a backslash character ("\") to the end of each continued line. For example, the following two lines are a single Miniscript command:

```
send "myAuthorMessage" with myFloat \
      to element's parent
```

Note that in the second line shown above, the leading spaces are ignored by Miniscript; the line does not have to be indented. Spaces and tabs can be used to improve the readability of scripts.

Variables

The values of variables in a mTropolis project can be accessed and set using Miniscript. mTropolis supports two types of variables: *simple* variables and *compound* variables. Simple variables contain a single value. Simple variables include the boolean, floating-point, integer, and string variable types. Compound variables contain multiple *fields* that each contain separate values. Compound variables include the integer range, list, point, and vector variable types. See Chapter 12, "Modifier Reference", for descriptions of the individual variable modifiers.

Using Variable Names in Scripts

Most variables can be referenced in a script by simply using the variable's name (as set in the variable modifier's editable name text box). The only exceptions are variable names that contain periods and/or spaces. If a variable name contains these characters, the name must be enclosed in single quotes when used in a script. For example, an integer variable with the name `myInt` could be referenced in a script as follows:

```
set myInt to 5
```

However, a variable with a name such as `my Wonderful Integer` must be enclosed in single quotes as shown below:

```
set 'my Wonderful Integer' to 5
```

Note that variables should be created *before* their names are referenced in a script. Writing Miniscript statements that use variable names that have not yet been created can lead to unpredictable results. Also, keep in mind that variables can only be used in a script if they are within the Miniscript modifier's scope. See "Variable Scopes" on page 13.25.

Accessing the Fields of Compound Variables

The fields of compound variables can be referenced by using a period followed by the name of the field. For example, the following command sets the "start" value of an integer range variable to 4:

```
set myIntegerRange.start to 4
```

The fields in each type of compound variable are listed in the descriptions of each variable modifier found in Chapter 12, "Modifier Reference". They are also described in "Setting the Value of Compound Variables" on page 14.5

Note that the syntax is the same for accessing the user-created fields of custom "compound variables" created with the compound variable modifier (see "Compound Variable" on page 12.25).

Literal Values

Literal values for most mTropolis data types can be used in scripts. Each mTropolis data type has a different syntax as shown in Table 14.1.

Data Type	Syntax	Example
Boolean	**true** or **false**	true
Floating-point	*n.n*	60.547
Integer	*n*	5
Integer Range	**(***n_{start}* **thru** *n_{end}***)** parentheses are optional	(1 thru 10)
List	**{***n_1, n_2, ..., n_{last}***}** items in a list can have any data type except list	{10, 15, 30}
Point	**(***n_x,n_y***)**	(240,320)
String	**"***text***"**	"mFactory"
Vector	**(***n_{angle}*° *n.*$n_{magnitude}$**)** the degrees symbol can be typed by pressing Shift-Option-8 on Macintosh	(60° 2.5)

Table 14.1. Miniscript Syntax for Literal Values. An "n" in the table indicates a numeral.

Element Names

Elements can be the targets of messages sent by Miniscript statements. Additionally, all elements in a mTropolis project have attributes whose values can be retrieved or modified using Miniscript.

Most elements can be referenced in a script by simply using the element's name (as set in the element's Element Info dialog). The only exceptions are element names that contain periods and/or spaces. If an element name contains these characters, the name must be enclosed in single quotes when used in a script (e.g., `'my Element'`).

Element *attributes* (see "Element Attributes" on page 14.17) can be referenced using a syntax similar to that used to reference the fields of compound variables. Add a period, followed by the attribute's name, to the element name. For example, the following command sets the "width" of an element:

```
set myElement.width to 120
```

Message and Command Names

Messages and commands can be sent to elements using the **send** statement. Message or command names must be enclosed in double quotation marks. For example:

```
send "myAuthorMessage" to element
```

SET—THE MINISCRIPT ASSIGNMENT STATEMENT

The Miniscript **set** statement can be used to change the values of variables or element attributes. It is Miniscript's version of an assignment operator. The basic form of this statement is:

```
set variable to value
```

where *variable* is the name of a variable or element attribute to be set and *value* is an expression of the proper type for *variable*.

Setting Values of Variable Modifiers

Variable modifiers give elements the ability to store many different types of data. Miniscript can be used to alter these values.

The **set** statement can be used to modify the values of both simple and compound variable types.

Setting the Value of Simple Variables

Setting the value of a simple variable is easy. The following examples show the syntax used to set each type of simple variable:

- Boolean variable:

  ```
  set myBoolean to true
  ```

- Floating-point variable:

  ```
  set myFloat to 4.555666
  ```

- Integer variable:

  ```
  set myInteger to 888
  ```

- String variable:

  ```
  set myString to "mFactory"
  ```

Setting the Value of Compound Variables

The fields of compound variable modifiers can be set individually or together as a group. To set the value of a field individually, use the syntax:

```
set variable.field to value
```

where *variable* is the name of the compound variable, *field* is the name of the field to be set and *value* is an expression of the proper type for *field*. Alternatively, the following syntax can be used for a more "natural language" style:

```
set the field of variable to value
```

The following examples show the syntax used to set various types of compound variables:

- Integer range variables have start and end fields. To set the start field individually:

  ```
  set myIntegerRange.start to 4
  ```

  To set the end field individually:

  ```
  set myIntegerRange.end to 9
  ```

  To set both fields:

  ```
  set myIntegerRange to (4 thru 9)
  ```

- List variables can store any number of values that are accessed with a slightly different syntax than most other variables. See "Miniscript Syntax for List Variables" on page 12.43.

- Point variables have an x and a y field. To set the x field individually:

  ```
  set myPoint.x to 15
  ```

  To set the y field individually:

  ```
  set myPoint.y to 30
  ```

  To set both fields:

  ```
  set myPoint to (15,30)
  ```

- Vector variables have an angle and a magnitude field. To set the angle field individually:

  ```
  set myVector.angle to 45
  ```

  To set the magnitude field individually:

  ```
  set myVector.magnitude to 1.41
  ```

  To set both fields:

  ```
  set myVector to (45°1.41)
  ```

- *Note: The ° symbol ("degrees") can be typed by pressing Shift-Option-8 on Macintosh.*

- User-created compound variables, made with the Compound Variable modifier, can have any number of user-defined fields. See "Compound Variable" on page 12.25 for complete documentation and examples.

Setting Values of Element Attributes

Element attributes can be set with a syntax similar to that used to set the values of compound variable fields. To set the value of an attribute, use the syntax:

```
set element.attribute to value
```

where *element* is the name of the element, *attribute* is the name of the attribute to be set and *value* is an expression of the proper type for *attribute*. Alternatively, the following syntax can be used for a more "natural language" style:

```
set the attribute of element to value
```

Note that in the statements above, the word "element" is optional since **set** assumes that

the script is referencing its own element's fields unless otherwise specified. Therefore, the statements above are equally valid written as:

```
set attribute to value
set the attribute to value
```

Some element's attributes have multiple fields, such as the position attribute, which accepts a point value. To access the fields of a compound value, simply add another period and the name of the field. For example, to set the "x" component of an element's "position" attribute, use the statement:

```
set element.position.x to value
```

Descriptions of Element Attributes

For lists of element attributes and their descriptions, see "Element Attributes" on page 14.17.

Setting Variables and Attributes to Incoming Values

Messages are often sent with an accompanying data value. The keyword **incoming** can be used with the **set** command to access data sent with the message that activates the Miniscript modifier. For example:

```
set myFloat to incoming
set myToon.width to incoming.x
```

The incoming data value could also be set to some new value. For example:

```
set incoming to myFloat
```

Note that incoming data can be of any type. If a **set** operation is attempted between items with different types, the value of the set "target" does not change. No error message or warning is generated.

THE SEND STATEMENT—SENDING MESSAGES AND COMMANDS

The **send** statement can be used to send environment messages, author messages, or commands from a script. Any number of messages can be sent from a single script, eliminating the need for multiple messengers. Any message or command described in Chapter 13, "Modifier Pop-Up Menus and Message Reference", can be sent with the **send** statement.

Basic Use of Send

The most simple form of the **send** statement sends a message to the element to which the script is attached:

```
send "Message"
```

The message or command enclosed in double quotes is sent to the element that contains the Miniscript modifier. Note that even if the Miniscript modifier is located inside a behavior, the message is still sent to the element that the behavior is associated with.

See Chapter 13, "Modifier Pop-Up Menus and Message Reference", for the names and descriptions of messages and commands that can be sent.

Specifying the Destination for a Message

Messages or commands can be sent to destinations other than the element to which the script is attached. Use the **to** keyword as shown below:

```
send "Message" to destination
```

where *destination* is an explicit or relative destination described by using the building blocks shown in Table 14.2.

For example, to send the author message **myMessage** to the parent of the element to which the script is attached, use the statement:

```
send "myMessage" to element.parent
```

To send the same message to an element that is a sibling of the element, simply use the name of the destination element. For example:

```
send "myMessage" to Frog
```

The destination can be any expression, composed of the building blocks shown in Table 14.2, that resolves to an object. For example:

```
send "Msg" to source.parent.parent
```

sends the specified message to the parent of the parent of the messenger that triggered the Miniscript modifier. Note that, as with attributes and compound variables, you can use the "dot" syntax shown above or the "natural language" syntax. For example, the previous send statement could also be written as:

```
send "Msg" to source's parent's parent
```

Sending Values with Messages

A value can also be sent along with the message or command using the **with** keyword:

```
send "Msg" with value to destination
```

where *value* is an expression of the appropriate type to accompany the message or command sent. For example:

```
send "Scroll Up" with 5
send "MoveFrog" with myPoint
```

The **incoming** keyword can also to used to access data that was received by the Miniscript modifier. For example:

```
send "New Left Edge" with incoming \
    to parent
```

If **incoming** contains a compound value, its fields can be accessed using the standard syntax:

```
send "add Value" with \
  incoming.magnitude to scene
```

Changing the Message Options

Just like messages sent from messengers, messages sent with the **send** statement can be sent with special options. By default, messages are set to "cascade", "relay", and are sent immediately. See "Message Options" on page 13.24 for more information on these options and message passing in general.

The **options** keyword can be used to change the default behavior of the message sent by a **send** statement. The **options** keyword must be followed by one or more option selections. Valid selections are **cascade**, **relay**, **immediate**, **all**, and **none**. The **options** keyword can also be shortened to **opt**.

The default option is **all**, meaning that the message will cascade, relay, and be sent immediately. When a single option is specified,

Destination	Syntax
Project	`project`
Section	`section`
Subsection	`subsection`
Scene	`scene`
Element	`element` (or simply omit as element is the default destination)

Element's parent	`element.parent`
	or
	`element's parent`

Miniscript modifier's parent	`parent`

Source's parent
(The *source* is the modifier that generated the message that triggered the Miniscript modifier.)
	`source.parent`
	or
	`source's parent`

Active scene	`activeScene`
Shared scene	`sharedScene`

Next element	`element.next`
	or
	`element's next`

Next modifier (either on the same element or in the same behavior that contains the Miniscript modifier)	`next`

Previous element	`element.previous`
	or
	`element's previous`

Previous modifier (either on the same element or in the same behavior that contains the Miniscript modifier)	`previous`
Element's sibling	*`elementname`* (i.e., use the actual name of the element's sibling)
Parent's sibling	*`elementname`* (i.e., use the actual name of the parent's sibling)

Table 14.2. Building blocks for targeting elements in Miniscript expressions

the remaining options are turned off. The **none** option can be used to turn all options off.

For example, to send a message with all options turned off:

```
send "myMsg" to parent options none
```

To send a message that only cascades:

```
send "myMsg" to element \
    options cascade
```

To send a message that cascades and relays, but is not immediate:

```
send "myMsg" to scene opt \
    cascade relay
```

THE IF STATEMENT—CONDITIONAL BRANCHES OF EXECUTION

The **if** statement can be used to evaluate an expression and take certain actions if that expression is true. The general syntax is:

```
if expression then statement
```

The Miniscript statement represented by *statement* will be executed if *expression* evaluates to true. For example, suppose variable a has the value 2 and variable b has the value 3 when the following **if** statement is executed:

```
if a<b then send "myMessage"
```

Because a is less than b, the message will be sent.

Multiple statements can be executed by using the syntax:

```
if expression then
statement1
...
statementn
end if
```

The **else if** and **else** keywords can be used to specify other conditions or statements that should be executed. Use the syntax:

```
if expression1 then
statements
else if expression2 then
statements
else if expressionn then
statements
else
statements
end if
```

Each expression is evaluated in order. If an expression is true, its corresponding statements are executed. Any following conditions are skipped and execution continues with the next Miniscript statement in the script. Statements associated with the **else** section are executed if no other condition is satisfied.

For example, the following script executes the first **set** statement if the angle field of myVector is greater than 30, otherwise, the second **set** statement is executed:

```
if myVector.angle > 30 then
set myVector.magnitude to rnd(25)
else
set myVector.magnitude to rnd(10)
end if
```

Definition of True

The expression evaluated by the **if** statement can be any type of expression. Usually, you want to evaluate a boolean expression

(which simply evaluates to either true or false). However, other data types have "true" and "false" conditions. The definition of true or false for different data types is as follows:

- Boolean values are either true or false.
- Integer and floating-point: 0 is false, any other value is true.
- Strings and compound values: these values are always false.

MINISCRIPT OPERATORS

Operators are used to create expressions. The set of Miniscript operators is described below.

Parentheses ( )

Parentheses are used to group expressions and to override the natural order of operator evaluation—expressions in parentheses are always evaluated first. For example, in the statement:

```
set c to (a+b)/e
```

The expression `(a+b)` is evaluated first, then the result is divided by `e`.

See "Operator Precedence" on page 14.12 for information on the default order of operator evaluation.

Mathematical Operators

Miniscript supports the following mathematical operators, described in order of descending precedence.

Exponentiation (^)

The caret (^) is the exponentiation operator. `A^B` is equal to `A` raised to the `B` power.

Multiplication (*)

The asterisk (*) is the multiplication operator. For example, `2*5` evaluates to 10.

Division (/)

The forward slash (/) is the division operator. For example, `6/2` evaluates to 3.

Div (Integer Division)

The **`div`** operator performs a special "integer" type of division where any remainder is dropped. For example, `4.0 div 3.0` evaluates to 1.

Mod (Modulus)

The **`mod`** operator performs a modulus operation. It returns the remainder when the first operand is divided by the second operand. For example, `9 mod 5` evaluates to 4.

Addition (+)

The plus sign (+) is the addition operator. For example, `2 + 4` evaluates to 6.

Subtraction and Negation (-)

The minus sign (-) is the subtraction and negation operator. For example, 4 - 2 evaluates to 2. The statement **`set c to -c`** changes the sign of the value in variable c.

String Concatenation (&)

The string concatenation operator (&) can be used to combine the contents of strings. This operator accepts two string operands and returns a single string that consists of the first operand, followed by the second operand. For example, the statement:

```
"mTropolis" & " rules"
```

returns the single string:

```
"mTropolis rules"
```

Boolean Operators

Boolean operators return either true or false boolean values. Different variable types have different "true" and "false" states—see "Definition of True" on page 14.10 for more information.

And

The **and** operator returns true when both of its operands are true. For example, if boolean variable `a` contains true and boolean variable `b` contains true, the expression `a and b` evaluates to true.

Not

The **not** operator is the boolean inverse operator. It inverts the boolean value of its single operand. That is, the expression **not true** evaluates to false and the expression **not false** evaluates to true.

Or

The **or** operator returns true when either of its operands is true. For example, the expression `0 or 1` evaluates to true.

Relational Operators

Relational operators can be used to test the relationship between two values. They return true or false based upon the relationship of their operands.

Equal To (=)

The equal sign (=) is the relational "equal to" operator. It returns true if both its operands are equal, otherwise it returns false. For example, `2 = 4` returns false.

Greater Than (>)

The greater than operator (>) returns true if the first operand has a greater value than the second operand. For example, `4 > 2` returns true.

Greater Than or Equal To (>= or ≥)

The "greater than or equal to" operator (>= or ≥, which can be typed on Macintosh by pressing option-period) returns true if the first operand is greater than or equal to the second operand. For example, both `4 ≥ 3` and `3 ≥ 3` return true.

Less Than (<)

The less than operator (<) returns true if the first operand has a lesser value than the second operand. For example, `2 < 4` returns true.

Less Than or Equal To (<= or ≤)

The "less than or equal to" operator (<= or ≤, which can be typed on Macintosh by pressing option-comma) returns true if the first operand is less than or equal to the second operand. For example, both `2 ≤ 3` and `3 ≤ 3` return true.

Not Equal To (<> or ≠)

The "not equal to" operator (<> or ≠, which can be typed on Macintosh by pressing option-=) returns true if its operands are not equal. For example, `3 ≠ 5` returns true.

Operator Precedence

Operators are evaluated in order of their precedence. Operators with a higher precedence

are evaluated before those with a lower precedence. Operators with equal precedence are evaluated from left to right, as they appear in the expression. Table 14.3 shows the precedence of mTropolis' operators.

Precedence Level	Operators
1 (highest)	( ), expressions inside parentheses are evaluated first.
2	not
3	^
4	*, /, div, mod
5	+, -
6	>, >=, ≥, <, <=, ≤
7	=, <>, ≠, and, or

Table 14.3. Operator precedence

SPECIAL ENVIRONMENT VARIABLES

Two read-only environment variables can be referenced from within Miniscript.

The Mouse Variable

The **mouse** environment variable holds a point value that contains the current position of the mouse cursor. For example, to move an element's origin point to the current mouse position, use the statement:

```
set element.position to mouse
```

The Ticks Variable

The **ticks** environment variable holds an integer value that contains the number of time ticks (defined as 1/60 second) that have elapsed since boot time. For example, to obtain the number of seconds that have elapsed since boot time, use the statement:

```
set myFloat to ticks/60
```

MINISCRIPT FUNCTIONS

Miniscript contains a number of built-in functions for performing mathematical operations, data type conversions, and other common operations. Functions can be used anywhere an expression or value can be used. For example, the **cos** function calculates the cosine of the specified angle:

```
set myFloat to cos(30)
```

Functions can also be nested. For example:

```
set myFloat to cos(exp(rnd(45)))
```

mTropolis assumes all angles are expressed in degrees.

Functions are described in alphabetical order, below. Note that some functions have multiple names that can be used interchangeably. These functions are listed with their names separated by commas. For example, to return the arctangent of an angle, either the **atn** or **arctangent** function can be used.

abs

The abs function returns the absolute value of its argument. The abs function is useful for tracking element movement. Use it to convert coordinate differences (negative or positive) into distances (always positive).

Syntax

abs(*numericExpression*)

Example

```
set myDistance to \
abs(positionB.x - positionA.x)
```

See Also
"sgn" on page 14.16

atn, arctangent

The atn function returns the angle, in degrees, whose tangent is *numericExpression*. The returned value is an angle between -90 and +90 degrees.

Syntax
atn(*numericExpression*)

Example
```
set myvector.angle to atn(45)
```

See Also
"tan, tangent" on page 14.17

cos, cosine

The cos function returns the cosine of *numericExpression*, which represents an angle measured in degrees. The return value is a floating-point number.

Syntax
cos(*numericExpression*)

Example
```
set myvector.angle to cos(45)
```

See Also
"sin, sine" on page 14.16
"tan, tangent" on page 14.17

cosh

The cosh function returns the hyperbolic cosine of *numericExpression*, which represents an angle measured in degrees. The return value is a floating-point number.

Syntax
cosh(*numericExpression*)

Example
```
set myFloat to cosh(2.2222)
```

See Also
"cos, cosine" on page 14.14
"sinh" on page 14.16
"tanh" on page 14.17

exp

The exp function returns the value of e raised to the power of *numericExpression*, where e is the natural number 2.71828 (i.e., the "natural" exponential function). The return value is a floating-point number.

Many natural processes involve quantities that increase or decrease at a rate proportional to their size. For example, a culture of bacteria will grow at a rate proportional to its mass. The value of an investment bearing interest that is continually compounded will increase at a rate proportional to that value.

Syntax
exp(*numericExpression*)

Example
```
set mynumber to exp(999.999)
```

See Also
"ln" on page 14.14

ln

The ln function returns the natural logarithm, log to the base e, of *numericExpression* where e is the natural number 2.71828. The return value is a floating-point number.

Syntax
ln(*numericExpression*)

Example
```
set myFloat to ln(3.1111)
```

See Also
"exp" on page 14.14
"log" on page 14.15

log

The log function returns the logarithm to the base 10 of *numericExpression*. The return value is a floating-point number.

This function is useful in calculating the pH of solutions, in heat conduction as well as for financial calculations.

Syntax
log(*numericExpression*)

Example
```
set myFloat to log(100000)
```

See Also
"ln" on page 14.14

num2str, numToString

The num2str function converts a numeric value to its string equivalent.

Syntax
num2str(*numericExpression*)

Example
```
set mystring to num2str(2.67)
```

See Also
"str2num, stringToNum" on page 14.17

polar2rect

The polar2rect function converts polar coordinates (angle, radius) to rectangular coordinates (x,y). This function accepts an argument of vector type. The return value is of point type.

Syntax
polar2rect(*vector*)
polar2rect(*angle°radius*)

Example
```
set myPoint to polar2rect(myVector)
```

See Also
"rect2polar" on page 14.15

rect2polar

The rect2polar function converts rectangular coordinates (x,y) to polar coordinates (angle, radius). This function accepts an argument of point type. The return value is of vector type (angle, magnitude).

Syntax
rect2polar(*point*)
rect2polar(*x,y*)

Example
```
set myVector to rect2polar(myPoint)
```

See Also
"polar2rect" on page 14.15

round

The round function rounds the floating-point value *numericExpression* to the nearest whole number. The return value is of integer type.

Syntax
round(*numericExpression*)

Example
```
set myInteger to round(2.71828)
```

See Also
"trunc" on page 14.17

rnd, random

The rnd function returns a random value from 0 to the value specified by *numericExpression*-1. The return value is an integer. This function is useful for randomizing messages in game play and positions on a Scene.

Syntax
rnd(*numericExpression*)

Example
```
set mynumber to rnd(100)
```

sgn

The sgn function returns the sign of *numericExpression*. If the value of *numericExpression* is less than 0, the return value is -1. If the value of *numericExpression* is greater than 0, the return value is 1. If *numericExpression* equals 0, the return value is 0.

Syntax
sgn(*numericExpression*)

Example
```
set mynumber to sgn(tan(rnd(26)))
```

See Also
"abs" on page 14.13

sin, sine

The sin function returns the sine of *numericExpression*, which represents an angle measured in degrees. The return value is a floating-point number.

Syntax
sin(*numericExpression*)

Example
```
set myvector.angle to sin(45)
```

See Also
"cos, cosine" on page 14.14
"tan, tangent" on page 14.17

sinh

The sinh function returns the hyperbolic sine of *numericExpression*, which represents an angle measured in degrees. The return value is a floating-point number.

Syntax
sinh(*numericExpression*)

Example
```
set myFloat to sinh(3.2222)
```

See Also
"cosh" on page 14.14
"sin, sine" on page 14.16
"tanh" on page 14.17

sqrt

The sqrt function returns the square root of *numericExpression*. The return value is a floating-point number.

Syntax
sqrt(*numericExpression*)

Example
```
set mynumber to sqrt(999.999)
```

str2num, stringToNum

The str2num function converts a string value to a floating-point value.

Syntax
str2num(*stringExpression*)

Example
```
set mynumber to str2num("1.5")
```

See Also
"num2str, numToString" on page 14.15

tan, tangent

The tan function returns the tangent of *numericExpression*, which represents an angle measured in degrees. The return value is a floating-point number.

Syntax
tan(*numericExpression*)

Example
```
set myvector.angle to tan(45)
```

See Also
"cos, cosine" on page 14.14
"sin, sine" on page 14.16
"tanh" on page 14.17

tanh

The tanh function returns the hyperbolic tangent of *numericExpression*, which represents an angle measured in degrees. The return value is a floating-point number.

Syntax
tanh(*numericExpression*)

Example
```
set myFloat to tanh(3.2222)
```

See Also
"cosh" on page 14.14
"sinh" on page 14.16
"tan, tangent" on page 14.17

trunc

The trunc function truncates the floating-point value *numericExpression*. That is, any decimal portion is removed. The return value is of integer type.

Syntax
trunc(*numericExpression*)

Example
```
set myInteger to trunc(2.71828)
```

See Also
"round" on page 14.15

ELEMENT ATTRIBUTES

As mentioned previously, all elements in a mTropolis project have attributes whose values can be retrieved and/or modified using Miniscript. Element attributes can be referenced using a syntax similar to that used to reference the fields of compound variables (see "Accessing the Fields of Compound Variables" on page 14.3) as described below.

Element Attribute Syntax

In general, an element attribute can be specified by adding a period, followed by the attribute's name, to the element name. For example, the current width of the element named `myPict` could be set to the value contained in `myInt` using the statement:

```
set myPict.width to myInt
```

A slightly different syntax can be used to refer to the attributes of the element on which the Miniscript modifier resides. Instead of using the element's name, the element can be referred to as `element`. For example:

```
set element.width to myInt
```

Alternatively, the "`element.`" part can simply be omitted because the element on which the Miniscript modifier resides is the default "target" for both messages and attribute references. For example, the following statement is equivalent to the previous one:

```
set width to myInt
```

- *Note: There is one special exception to this syntax. The "parent" attribute, when specified without the "element." prefix, specifies the Miniscript modifier's parent (i.e., the element or behavior that contains that script).*

Note also that the next and previous keywords can be used to specify element siblings using a syntax similar to that shown in Table 14.2 on page 14.9.

Attribute Descriptions

The tables below (Table 14.4 through Table 14.9) show the attributes that can be accessed for each type of media element. Also shown is the attribute's corresponding data type and two columns labeled "R" and "W".

- If a bullet appears in an attribute's "R" column, the attribute is *readable*. For these attributes, using the syntax *element.attribute* in a script returns the current value of *attribute* (which has the data type shown in the "Type" column) for the specified *element*.
- If a bullet appears in an attributes's "W" column, the attribute is *writable*. The value of the attribute can be changed by using the **`set`** statement (e.g., **`set myToon.paused to true`**). Changing the value of an attribute causes that aspect of the element's behavior or appearance to change instantly.

Note that not all element types support every attribute. Consult Table 14.4 through Table 14.9 for lists of attributes supported by each element type. In addition to the attributes shown below, keep in mind that the `project`, `section`, `subsection`, `scene`, `element`, and `parent` "building blocks" described in Table 14.2 are essentially attributes of each element though they are not shown in the tables below.

Individual attributes are described in alphabetical order, below.

- asset: The name of the media asset (e.g., the media asset's name as shown in the Asset Palette) that is currently associated with the element. Changing the value of this at-

tribute changes the media contained in the element. The new asset must be of the correct media type for the element (i.e., you cannot change the type of the element by assigning it a new asset of a different type). Note that an asset name is *not* a filename—the media must have been linked to the current project before it can be used with the asset attribute. Note also that, for assets to be dynamically changed in a built title, they must actually be used in a scene (so the build title process knows to export the assets to the finished title file). Simply create a scene that is never actually shown as a placeholder for such assets.

- cache: When true, the contents of the element are cached in memory.
- cel: The animation cel currently displayed in the element.
- celcount: The total number of cels in the animation.
- clickedline: For editable text elements, the line number where the cursor was last positioned. The first line of a text element is line number 1. This attribute is always 1 for non-editable elements.
- controllerclick: If this attribute is set to true, user mouse clicks on the movie controller are sent only to QuickTime, they are not passed to the mTropolis QuickTime element.
- direct: When true, the element is drawn "direct to screen". See "Direct to Screen Check Box" on page 6.3. When false, the element is drawn in its regular position in the layer order.
- duration: The length of a movie in QuickTime timevalues. To find the length of a movie in seconds, divide this value by the movie's timescale attribute.
- editable: When true, the text element can be edited in runtime mode by the user. When false, the text cannot be edited.
- globaloffset: A point that contains the position of the element with respect to the origin of the scene (i.e., the upper left hand corner of the scene). The x field contains the number of pixels to the right of the scene origin. The y field contains the number of pixels down from the scene origin. See also "position" on page 14.20.
- height: The height of the element (i.e., its "y" size) in pixels.
- layer: The layer order number of the element. See "Layer Order" on page 10.2.
- line[*n*]: The text contained in line number *n* of the text element (where *n* is an integer from 1 to linecount). For example, to set the value of the second line of text element "myText", use the statement:

```
set myText.line[2] to "mFactory"
```

- linecount: The total number of lines of text in the text element.
- loop: When true, the time-based media in the element starts playing again from its beginning when it reaches its end. When false, the media plays only once.

- loopbackforth: When true, the time-based media in the element plays backward when it reaches its end, then plays forward again when it reaches its beginning.
- mastervolume: A volume control for the entire project. Valid values range from 0 (silence) to 7 (full volume).
- movieclick: If this attribute is set to true, user mouse clicks on the QuickTime movie are not sent to mTropolis, but are handled by QuickTime itself.
- name: The name of the element, as set in its Element Info dialog box. See "Element Name Field" on page 6.2.
- objectID: A unique ID assigned to each element by mTropolis.
- pan: The sound panning (i.e., left/right balance of the sound in the stereo field) parameter. Valid values range from -100 to 100. Pan value -100 is full left, 0 is centered, 100 is full right.
- paused: When true, the element is in its paused state. When false, the element is not paused. See "Paused/Pause" on page 13.12.
- playeveryframe: When true, every frame in the QuickTime movie will be played regardless of the rate (audio may be dropped or go out-of-sync). When false, frames may be dropped to display the movie at its proper rate with audio in sync.
- position: A point that contains the position of the element with respect to the origin of its parent (i.e., the upper left hand corner of its parent). The x field contains the number of pixels to the right of the parent's origin. The y field contains the number of pixels down from the parent's origin. Note that this attribute will be different from the value of the globaloffset attribute if the element has a parent other than the scene. See "globaloffset" on page 14.19.
- range: The range of cels to be played in an animation. The range of timevalues to be played in a QuickTime movie.
- rate: For mToons, this value is the playback speed in frames per second. For QuickTime, this value is a rate multiplier where the movie's default speed is represented by 1. Higher values make the movie play faster; lower values make the movie play slower. Negative values make the movie play in reverse. For example, setting this value to -2 would make the movie play backward at twice its normal speed.
- showcontroller: When true, the QuickTime movie's standard play controls are visible. When false, the controller is hidden.
- size: A point value that contains the width and height of the element. The x field contains the width of the element, in pixels. The y field contains the height of the element, in pixels.
- timescale: The rate of the QuickTime movie in QuickTime time values per second. To compute the movie's length in seconds, di-

vide the movie's duration attribute by this value.

- timevalue: The current time in the QuickTime movie (in QuickTime-specific units). To compute the current time in seconds, divide this value by the movie's timescale attribute.
- trackdisable: Set this attribute to an integer (representing a QuickTime track number) or a string (representing a QuickTime track name) to cause the specified QuickTime track to be disabled.
- trackenable: Set this attribute to an integer (representing a QuickTime track number) or a string (representing a QuickTime track name) to cause the specified QuickTime track to be enabled.
- usertimeout: A length of time after which the User Timeout message is sent. This value is measured in 1/60 second intervals. For example, to generate User Timeout messages at intervals of 1 second, set this attribute to 60.
- visible: When true, the element is visible. When false, the element is hidden.
- volume: An integer value representing a percentage of the sound's full volume. A value of 0 indicates that the sound will be completely silent when played. A value of 100 indicates that the sound will play at full volume.
- width: The width of the element (i.e., its "x" size) in pixels.

Shared Attributes of 'Graphical' Elements

The following attributes (Table 14.4) are common to graphic elements, mToon elements, sound elements, text elements, QuickTime elements, and mTropolis modifiers.

Graphical Attributes	Type	R	W
asset	string		•
cache	bool	•	•
direct	bool	•	•
globaloffset	point	•	
height	int	•	•
layer	int	•	•
name	string	•	•
objectID	int	•	
position	point	•	•
size	point	•	•
visible	bool	•	•
width	int	•	•

Table 14.4. Shared attributes of graphic, mToon, sound, text, QuickTime elements, and mTropolis modifiers

mToon Attributes

The following attributes (Table 14.5) are unique attributes for mToon elements.

mToon Attribute	Type	R	W
cel	int	•	•
celcount	int	•	
loop	bool	•	•
paused	bool	•	•

Table 14.5. Attributes of mToon elements

mToon Attribute	Type	R	W
range	range	•	•
rate	int	•	•

Table 14.5. Attributes of mToon elements

Project Attributes

The following attributes (Table 14.6) are unique to the project component.

Project Attributes	Type	R	W
mastervolume	float	•	•
usertimeout	int	•	•

Table 14.6. Attributes of the project

Text Attributes

The following attributes (Table 14.7) are unique to text elements.

Text Attribute	Type	R	W
clickedline	int	•	
editable	bool	•	•
line[*n*]	string	•	•
linecount	int	•	
paused	bool	•	•
rate	int	•	•
text	string	•	•

Table 14.7. Attributes of text elements

Sound Attributes

The following attributes are unique to sound elements. Note that, for sound elements, "positioning" attributes such as position, size, and globaloffset are inherited from the element's parent (because sound elements have no "physical" representation in the layout view).

Sound Attribute	Type	R	W
pan	int	•	•
paused	bool	•	•
volume	int	•	•

Table 14.8. Attributes of sound elements

QuickTime Attributes

The following attributes are unique to QuickTime elements.

QuickTime Attribute	Type	R	W
controllerclick	bool	•	•
duration	int	•	
loop	bool	•	•
loopbackforth	bool	•	•
movie	int	•	•
movieclick	bool	•	•
pan	int	•	•
paused	bool	•	•
playeveryframe	bool	•	•
range	range	•	•
rate	float	•	•
showcontroller	bool	•	•
timescale	int	•	•
timevalue	int	•	•
trackdisable	int/string		•
trackenable	int/string		•
volume	int	•	•

Table 14.9. Attributes of QuickTime elements

RESERVED WORDS

All Miniscript statement keywords, function names, operators, and element attributes described in the preceding sections of this chapter are "reserved" words. These words should not be used to name project components or variables. While it is possible to use these reserved words outside of Miniscript to name project components, it is strongly recommended they not be used so as to avoid confusion and possible syntax or logic errors when writing scripts.

Chapter 15. Glossary

Alias

An alias is a special "copy" of a modifier that refers to the settings of the master copy it was made from for its functionality. When an alias is made, its master copy is automatically placed in the alias palette. Any changes to the settings of an alias update the settings of both the master copy in the palette and all other aliases in the project that refer to it.

Asset

A media file that has been linked to a project is called an asset. mTropolis compatible files include: PICT, AIFF and snd files, mToons (mTropolis' animation format) and QuickTime movies.

Author Message

A message that has been scripted by the author is called an author message. An author message can be sent from a messenger and it can be specified as the message to activate or deactivate a modifier.

Behavior

An encapsulated set of modifiers with an "on/off" switch is a behavior. Behaviors allow complex interactions to be created and organized.

Broadcasting

Broadcasting is the means by which messages are automatically passed from the recipient of a message to its descendants. *See* Descendant.

Cel

A single frame in an mToon. *See* mToon.

Child

All components directly contained by another component are considered its children. For example, a graphic modifier and an element transition modifier contained by the same element are that element's children.

Commands

Instructions sent from messengers that are directly implemented by an element.

Descendant

Any component that is contained by another component is called a descendant. For example, all items in a project are the descendants of the element "project."

Element

An element is one of mTropolis' basic building blocks. All elements in a project are structural, i.e., they may contain and therefore provide structure for other items. Structural elements include: project, section, subsection, scene and element. Of these compo-

nents, only scenes and elements may contain media. Media elements have pop-up editors that set their basic display properties. *See* Shared Scene and Scene.

Environment Message

Changes in mTropolis' runtime environment (e.g., a user's mouse click or a scene change) are detected by mTropolis' engine and automatically sent as messages to the components of a project. These messages are called environment messages.

Hierarchy

A basic organizational principle in mTropolis' authoring environment is the hierarchy. A project has a hierarchical structure composed of sections, subsections, scenes and elements. A section acts as a parent of its subsections, a subsection acts as the parent of its scenes, and a scene acts the parent of its elements. Elements may also be parents of other elements. This structure helps organize a project, and provides a pathway for message passing through the project.

Ink

A color's "ink" determines how it mixes the foreground and background colors of the image's pixels on the screen. For example, some ink effects mix the foreground and background colors with the element's color, effectively tinting the image.

Layer Order

Layer order is the order in which text and graphic elements are drawn on the screen. When new elements are added to a scene, they draw on top of previous elements. mTropolis keeps an internal list of elements in a scene and the order in which they are drawn. This layer order can be changed using tools in the authoring environment. Drawing elements on top of each other creates the illusion that the two-dimensional (X,Y) screen has a Z (depth) dimension. The effect is popularly called "2.5D."

Library

A library is a separate file where project components can be stored. Libraries can be used to distribute portions of a project to development team members for editing, or to archive project components for use in other projects. Within the mTropolis environment, an open library is represented as a palette. With the exception of the project element, any component can be dragged and dropped in and out of this palette.

Linking

Linking is the procedure used to establish a path from a mTropolis project to external media files. Media that has been linked to a project appears in the asset palette. Once linked, media can be dragged and dropped into the project in any view.

Messages

Messages are the means by which software components in a project communicate. Messages are signals that are used to activate or deactivate modifiers in a project. Both the

mTropolis engine and messenger modifiers pass messages between components.

mFusion Technology

The object-oriented technology underlying mFactory's mTropolis. Features of object technology such as messaging and reusability of components are built into mTropolis tools.

Miniscript

Miniscript is a simple scripting language embedded in a modifier. Its syntax is loosely based upon AppleScript, Apple's scripting language. It is a mTropolis message-handling tool and is used for mathematical operations, conditional execution of messages, and for "containing" multiple operations.

mToon

An mToon is mTropolis' proprietary animation format. It is a series of images compiled from the mToon editor into a single file by mTropolis. An animation created in a 2D or 3D program which has been saved as a PICS or PICT file can be processed as an mToon.

mToons are similar to digital video (such as QuickTime movies) in being a series of images, but differ in the way they are controlled during playback. While digital movies are time-based (only their duration can be specified), mToons are more flexible. Besides having the ability to play cels for a specified duration, mTropolis can play a selected cel or a specified range of cels. This precise control over the mToon makes them especially useful for animated sequences. For example, a rabbit's sitting, walking and running motions can be compiled into a single mToon linked to an element. The mToon now has a "library" of actions that can be individually accessed and played back.

Modifier

Modifiers are used by dragging and dropping them onto elements. The capabilities or properties of a modifier become a part of the element or behavior on which they are placed. In the case of modifiers within behaviors, these capabilities or properties become a part of the element with which they are associated.

Modifiers have dialogs that allow the author to alter their default settings.

MOM

mFactory's Object Model (MOM) is an application programming interface for multimedia programmers who want to extend the power of mTropolis by writing their own modifiers and mTropolis features in languages such as C or C++.

Palette

A floating menu containing project tools and/or resources is called a palette. For example, the asset palette contains the media assets that have been linked to a project.

Parent

All components in a project, with the exception of the project, have one parent. The component that directly contains another component is referred to as its "parent." For

example, a project is the parent of sections, a section is the parent of a subsections, a subsection is the parent of scenes, and a scene is the parent of elements. An element may also be the parent of other elements.

Project

A project is a component that holds an entire title within it. The children of a project are sections. A project can contain modifiers that modify the actions or characteristics of the entire title. Titles created in the mTropolis authoring environment are referred to as projects during the development phase. Projects cannot contain other projects.

Scene

The structural component one level below a subsection. Scenes can contain elements that are visible to the user in runtime mode. The elements that are added to a scene draw on top of it and are attached to it. Deleting the scene deletes all the elements and modifiers attached to it. *See* Element.

Scope

Where a modifier is placed in a project determines its scope — that is, the group of objects that may "see" it and that may access its value in mTropolis. All descendents of a variable's parent are in its scope.

Section

A section is an element that can be used to organize parts of a title into groups that logically belong together. For example, a title developer for a travel game might put everything to do with Africa under a single section. Sections are parents to subsections (*See* Subsection). As with a project, a section can contain modifiers. Sections cannot contain other sections.

Shared Scene

Special scenes that appear behind all other scenes in a subsection are shared scenes. Shared scenes are used to store elements and modifiers common to all the scenes in a subsection. For example, a background image placed on the shared scene forms the backdrop of all subsequent scenes. Elements and modifiers, such as navigational buttons, that are common to all scenes in a section are placed in the shared scene. *See* Element *and* Scene.

Sibling

All components that share the same parent are referred to as "siblings". For example, a graphic modifier and an element transition modifier contained by the same element are called siblings.

Structure

The structure is literally the way a project is organized. mTropolis projects consist of structural components and modifiers that interact through a messaging system. A modifier can change the component to which it is attached or another component by issuing a message that is interpreted and acted upon by the recipient.

Subsection

A subsection is a component that can be used to more finely divide the content of a section, literally like a subsection in a book. As with a project and section, a subsection can contain modifiers.

Subsections cannot contain other subsections or sections.

Title

A compiled, playback version of a project is called a title.

Appendix A. MovieTrax

This appendix describes MovieTrax, a stand-alone Macintosh application bundled with mTropolis that allows simple editing and manipulation of QuickTime video files.

- *Note: The version of MovieTrax provided with this release of mTropolis is a "beta" version. Program features and documentation may not be complete. Features are subject to change in future releases.*

WHAT IS MOVIETRAX?

MovieTrax is a QuickTime editing application that allows the editing of *tracks* within a QuickTime file. Tracks are the basic components of a QuickTime movie that contain different types of media to be played. QuickTime supports many different types of tracks including video, audio, text, MIDI, sprite, and timecode tracks. Tracks are independent of one another and a movie can contain multiple tracks of the same type. For example, different video tracks in a movie may have different durations, compression encodings, bit depths, start times, etc.

MovieTrax can be used to create new multitrack movies from component media or to edit existing ones. Basic temporal and spatial editing options are also supported, including the creation of ranges in a movie.

Using QuickTime element attributes (see "Element Attributes" on page 14.17) or the track control modifier (see "Track Control Modifier" on page 12.80), mTropolis can manipulate multitrack QuickTime movies. For example, the track control modifier can be used to activate or deactivate individual tracks in a movie.

STARTING MOVIETRAX

To start MovieTrax, double-click the MovieTrax icon. The MovieTrax menus appear. MovieTrax menu options are described in the sections below.

FILE MENU

Options in the File menu can be used to open, import, and save QuickTime movies for editing with MovieTrax.

New Movie

Select this option to open a new, empty List window.

Open Movie

Select this option to open an existing QuickTime file. A standard file dialog appears. Select a QuickTime movie to open. If the movie has not previously been opened by MovieTrax, a new list window appears, listing the tracks the movie contains. Movies that have previously been opened by MovieTrax open

to show the configuration of windows visible when the movie was last saved.

Note that other types of component media can also be opened and converted into QuickTime movies using this option. Valid files are displayed in the file selection dialog and the "Open" button changes to "Convert" button when they are highlighted. Click "Convert" to open the media file and convert it to a QuickTime movie. The media becomes available in MovieTrax as a QuickTime track.

Import Movie

Select this option to import the tracks from a QuickTime file directly into the currently-active movie (i.e., the movie associated with the currently-active view).

Export Tracks

Select this option to save only the currently-selected tracks as a new QuickTime file. A standard file selection dialog appears.

Close

Select this option to close the currently-active window.

Close Project

Select this option to close all open windows and their associated files.

Save

Select this option to save the currently-active movie with its current filename. If the movie has not yet been saved, a standard file selection dialog appears. Enter a name for the movie and select "Save". Select the "Make movie self-contained" checkbox to create a movie that is independent of its original source files.

Save As

Select this option to save the currently-active movie with a new name. A standard file selection dialog appears. Enter a name for the movie and select "Save". Select the "Make movie self-contained" checkbox to create a movie that is independent of its original source files.

Quit

Select this option to quit MovieTrax. If there are any unsaved or changed movies open, MovieTrax asks if they should be saved before quitting.

EDIT MENU

MovieTrax supports the following standard edit options.

Undo

Select this option to undo the last operation. Note that not all operations can be undone.

Cut

Select this option to cut the current selection and place it on the clipboard. Note that tracks cannot be cut.

Copy

Select this option to copy the current selection to the clipboard. Note that tracks cannot be copied.

Paste

Select this option to paste the contents of the clipboard.

Select All

Select this option to select all tracks in the currently-active window.

MOVIE MENU

Options in the Movie menu affect all tracks in the current movie.

Movie Info

Select this option to display information about the currently-active movie.

Play/Pause

Select this option to play the currently-active movie. Note that if the Spatial view is not open, the movie plays, but the video portion is not visible.

When a movie is playing, this menu option changes to Pause. Select it to stop the currently-playing movie.

Display Time Values/Display Standard Time

Use this menu toggle to change the time format shown in the Movie Controller between QuickTime "time values" and hours:minutes:seconds.hundredths format.

Loop

Select this menu toggle to cause the movie to loop from the beginning (i.e., repeatedly play through and restart from the beginning) when it plays.

Back and Forth

Select this menu toggle to cause the movie to loop back and forth (i.e., repeatedly play forward to the end then backward to the beginning) when it plays.

Play Every Frame

Select this menu toggle to play the movie without dropping frames. Note that audio does not play when this option is checked.

Play Selection Only

Select this menu toggle to play only the currently-selected range when Play is activated.

New Range

Select this command to specify a new range. The New Range dialog (Figure A.1) appears. Enter a name for the range (if desired), and a start and end time in QuickTime units. Note that if a range selection has been made in the Movie Controller, start and end times for that range are shown in the dialog. Click "OK" to create the new range. The new range becomes available in the "Range" pop-up menu on the movie controller. Click cancel to dismiss the dialog without creating a new range.

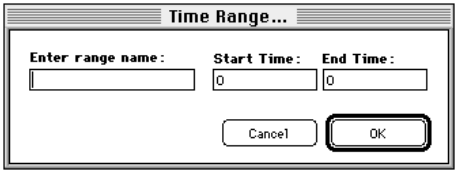


Figure A.1 The Range dialog

QuickTime ranges can be used with the mTropolis *Play* command by typing the range name into the "With" field of a messenger. Unlike mToon ranges, the ranges will not automatically appear as items in the "With" pop-up menu.

TRACK MENU

These options can be used to configure the behavior and appearances of tracks within a movie. Note that many of these commands are only accessible when the Spatial window is open and a track has been selected in that window.

Track Info

Select this option to display information about the currently-selected track (the track can be selected in the List, Spatial, or Temporal window). The Track Info dialog appears. A number of information sections, with their own open/close triangles, are shown. Click the triangles to open or close that information section.

The "Options" section contains three checkboxes that can be set by the user. These options control various QuickTime track defaults:

- **Enabled**: Check this box (the default) to set the track as initially enabled when used in mTropolis.
- **Preload**: Check this box to set the track as initially preloaded when used in mTropolis. Note that preloading of media is limited by the memory available on the machine that plays the media. By default, this box is unchecked.
- **Cache hints**: Hints are pieces of information about how the track will play. Check this box to cache hint information when the track is initially loaded into mTropolis. The next time that track is played, performance may be improved (at the expense of using extra memory to store the hints). By default, this box is unchecked.

Half Size

Select this option to display the track currently selected in the Spatial window at half its normal size.

Normal Size

Select this option to display the track currently selected in the Spatial window at normal size (the default).

Double Size

Select this option to display the track currently selected in the Spatial window at double its normal size.

Bring to Front

Select this option to bring the track currently selected in the Spatial view to the front of any other tracks that it overlaps.

Send to Back

Select this option to send the track currently selected in the Spatial view to the back of any other tracks that it overlaps.

Bring Forward

Select this option to bring the track currently selected in the Spatial view one position forward in the "layer order" of tracks.

Send Backward

Select this option to send the track currently selected in the Spatial view one position back in the "layer order" of tracks.

Align

The options in this cascading menu can be used to align two or more tracks in the spatial view. The following options are available when two or more tracks are selected in the Spatial window:

- **Left**: Select this option to align the left edges of the selected tracks. This option is only available for tracks selected in the Spatial window.
- **Right**: Select this option to align the right edges of the selected tracks. This option is only available for tracks selected in the Spatial window.
- **Top**: Select this option to align the top edges of the selected tracks. This option is only available for tracks selected in the Spatial window.
- **Bottom**: Select this option to align the bottom edges of the selected tracks. This option is only available for tracks selected in the Spatial window.

The following options are available when two or more tracks are selected in the Temporal window:

- **Start**: Select this option to align the start times of the selected tracks.
- **End**: Select this option to align the end times of the selected tracks.

Clip

The options in this cascading menu can be used to clip the visible area of the track currently selected in the Spatial window. Options are:

- **Crop**: Select this option to turn the selected track's resize handles (marked by red dots in the Spatial view) into cropping handles (marked by green dots). Moving the handles changes the "frame" of the track, without resizing its contents.
- **Paste**: This option is only available when a PICT image is on the clipboard. Select this option to paste the PICT image onto the current track as a clipping region. The PICT is converted to black and white—white pixels indicate a clipped region, while black pixels reveal the image underneath.
- **Invert**: Select this option to invert the clipping region of the currently-selected track.
- **Clear**: Select this option to clear the clipping region of the currently-selected track.

Matte

The options in this cascading menu can be used to specify a matte for the currently-selected track. A matte is similar to a clipping region, except that a matte has degrees of transparency and may also impart color tinting to the track. Options are:

- **Paste**: This option is only available when a PICT image is on the clipboard. Select this option to paste the PICT onto the current track as a matte. White areas of the PICT are completely opaque while black areas are completely transparent, showing the image underneath. Grayscale and colored pixels impart a tint to the image based upon their lightness.
- **Clear**: Select this option to remove a matte from the currently-selected track.

Ink

The options in this cascading menu can be used to specify an ink effect for the track currently-selected in the Spatial window. Options are:

- **Normal**: Select this option (the default) to display the selected track in its default state. This option is similar to mTropolis' "Copy" ink effect.
- **Bkgd Transparent**: Select this option to make the "Op Color" (selected with the "Set Op Color" option, described below) transparent in the selected track.
- **Blend**: Select this option to make the "Op Color" transparent based on lightness values in the selected track. This option is similar to the mTropolis "Blend" ink effect.
- **Set Op Color**: Select this option to display a color selection dialog. Select a color to be used in the Background Transparent and Blend options described above.

Set Volume

Select this option to select the volume of the selected audio track. The Set Volume dialog appears. Use the volume slider to select a value from 0 (silent) to 100 (full volume). Click "OK" to accept the change. Click "Cancel" to dismiss the dialog without changing the audio's volume. This option is available only when an audio track is selected in the List or Temporal windows.

Set Balance

Select this option to select the stereo balance of the selected audio track. The Set Balance dialog appears. Use the volume slider to select a value from -100 (full left) to 100 (full right). Click "OK" to accept the change. Click "Cancel" to dismiss the dialog without changing the audio's stereo position. This option is available only when an audio track is selected in the List or Temporal windows.

Set Start Time

Select this option to set the start time of the selected track to the time indicated by the Movie Controller. This option is available only when a track is selected in the Spatial or Temporal window.

VIEW MENU

Options in this menu can be used to show and hide the various MovieTrax display windows. Options and the windows they select are described below.

List Window

Select this menu toggle to show and hide the List window (Figure A.2). The List window shows the tracks in a movie and lists the name, type and layer number of tracks. The layer number of tracks is similar to the layer order number of elements in mTropolis. Tracks with a lower layer number appear "in front" of other tracks in the Spatial view.

Other features of this window include:

- Tracks can be selected by clicking on their names. Use ⌘-Click to select multiple items.
- Selected tracks can be deleted by pressing the Delete key.

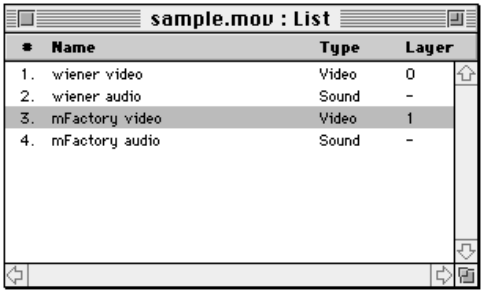


Figure A.2 The List window, shown here from a movie that contains four tracks

- Tracks can be dragged and dropped into other windows.

Spatial Window

Select this menu toggle to show and hide the Spatial window (Figure A.3). The Spatial window shows the visual portion of a movie. Tracks can be repositioned in this window by dragging and dropping.

Other features of this window include:

- As a track is repositioned in the window, its x, y coordinates in pixels (with respect to the left and top edges of the window) are reported at the bottom of the window.
- Tracks can be selected by clicking them. Multiple tracks can be selected using ⌘-Click and Shift-Click.
- When a track is selected, red dots appear on its corners. These dots represent resize handles. Drag the handles to scale the track. Width and height of the track, in pixels, is reported at the bottom of the window.
- Selected tracks can be deleted by pressing the Delete key.
- When the mouse is over a track, its name is shown at the bottom of the window.

Temporal Window

Select this menu toggle to show and hide the Temporal window (Figure A.4). The Temporal window shows the start and end times of tracks. The time position of a track can be changed by dragging it within the window.

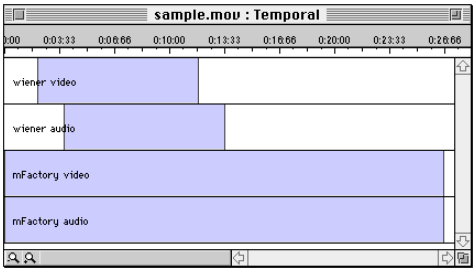


Figure A.4 The Temporal window, shown here from a movie with four tracks. The shaded part of the bar represents the extent of the track.

The start and end times of the track are shown at the bottom of the window as the track is dragged.

Other features of this window include:

- A time scale, in hours:minutes:seconds:hundredths is shown at the top of the window. The scale of the view can be changed by clicking the zoom out and zoom in tools at the bottom left corner of the window.
- Tracks can be selected by clicking on their shaded portions. Use ⌘-Click to select multiple items.
- Selected tracks can be deleted by pressing the Delete key.
- If a range of time values have been selected in the Movie Controller (by Shift-dragging the Controller's slider), the range is displayed as a yellow highlight in the time scale.

Figure A.3 The Spatial Window, shown here from a movie with two video tracks.

Movie Controller

Select this menu toggle to show and hide the Movie Controller (Figure A.5). The Movie Controller is similar to the standard QuickTime controller, including volume, play/pause, position slider, step, and loop controls. In addition, a time format toggle and Range pop-up menu have been added.

Features of the Movie Controller include:

- Select the volume icon to display a volume control slider.
- Select the play/pause icon to play or pause the movie.
- Drag the slider to set the current time of the movie. Shift-drag the slider to select a range of time values.
- Click the step controls to step backward or forward through the movie one frame at a time.
- Select the loop control to toggle between "play once" and "looped" play modes.
- Select the time format control to toggle the display of time between "time value" and "hour:minute:second:hundredths" mode.
- The Range pop-up can be used to select previously-defined ranges or specify new ones. Select "New Range" to display the New Range dialog (see "New Range" on page A.3). If a range of times has been selected by Shift-dragging the slider, those values will be shown as the defaults in the New Range dialog.

Ranges

Select this menu toggle to show and hide the Ranges window (Figure A.6). The Ranges window lists any previously-defined ranges by name, start time value, and end time value. Double-click on a range name to display the Time Range dialog (Figure A.1), which can be used to change the name, start time, and end time of the selected range.

WINDOW MENU

The Window menu lists the names of all open MovieTrax windows. The current window is indicated with a checkmark. Select any win-

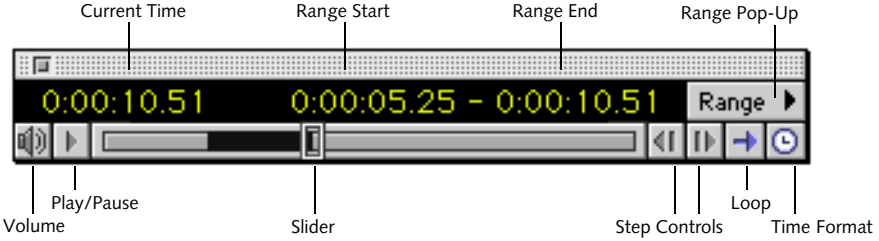


Figure A.5 The Movie Controller

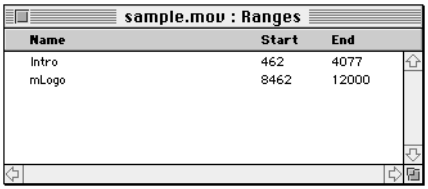


Figure A.6 The Ranges dialog

dow name from the menu to make that window the current one. The selected window moves to the "front" of the screen.

Appendix B. mTropolis Object Model (MOM) User's Guide

This appendix contains introductory documentation for the mTropolis Object Model (MOM). MOM Reference documentation can be found as online documentation.

- *Note: The version of MOM provided with this release of mTropolis is a "beta" version. Program features and documentation may not be complete. Features are subject to change in future releases.*

ABOUT THIS DOCUMENT

Although the examples referred to in this document use the Metrowerks CodeWarrior development environment, they could be modified to work with other environments (e.g., Symantec, MPW). We will be providing explicit support for such environments, including Windows development environments, in future releases of MOM.

We assume you have a working understanding of mTropolis, and make frequent reference to the accompanying API and examples. Make sure that these files are accessible while working through this reference.

This document is still under construction. We have attempted to document the major methods and functions used by the examples. Future versions of this reference will document the other major methods and functions, various supplementary methods and functions, along with more sophisticated examples that better illustrate the scope of MOM's capabilities.

INTRODUCTION

The mFactory Object Model (MOM) is a mechanism by which users can programmatically extend the functionality of mTropolis, using a language such as C or C++. With MOM, new components can be built to be seamlessly and efficiently integrated with the mTropolis environment from the underlying engine to the visual interface.

As a "glue" between the mTropolis environment and customized MOM components, mFusion defines a base class, MComponent, from which MOM components can be derived; mFusion also provides services to allow MOM components to communicate with the mTropolis engine, mTropolis modifiers and any other MOM components. Reflecting the true object orientation of mTropolis, a MOM component can be a derived class of MComponent or another MOM component with no inherited fragile base class.

Seamlessly integrated with the mTropolis environment, MOM components interact with the mTropolis engine and modifiers in the

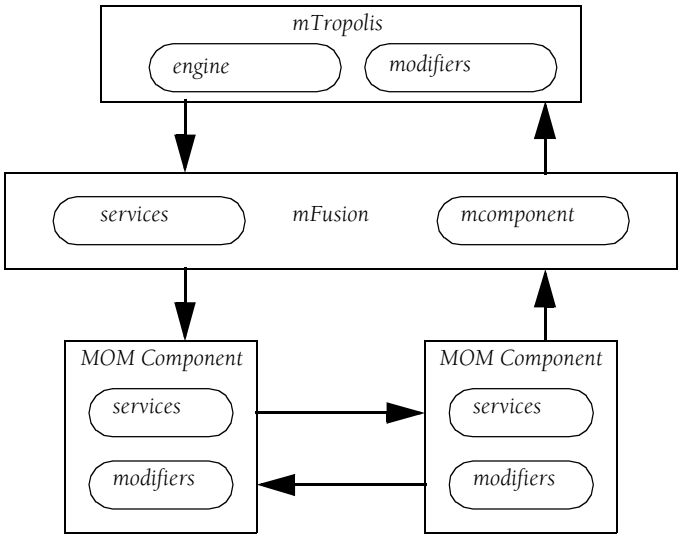


Figure B.1 Relationships among mTropolis, mFusion, and MOM components

same fashion as mTropolis modifiers interact with the engine by sending messages and accessing attributes. MOM components receive messages from, send messages to, and access attributes of the mTropolis engine, mTropolis modifiers (the messenger modifier, for example), and other MOM components. The relationship between the mTropolis environment and MOM is depicted in Figure B.1.

GETTING STARTED

This section describes the pieces that comprise the MOM v1.0b1 release and details MOM-related CodeWarrior issues.

The MOM v1.0b1 Folder

The 'Read Me First' document describes how to install the 'MOM v1.0b1' folder from the CD-ROM. Open this folder to display the following window shown in Figure B.2.

The "MOM" folder contains the header files, prefix file and libraries that comprise the MOM API. The major files in this folder are "MFusion.h" (the primary header file), "MMetro.h" (the Metrowerks CodeWarrior

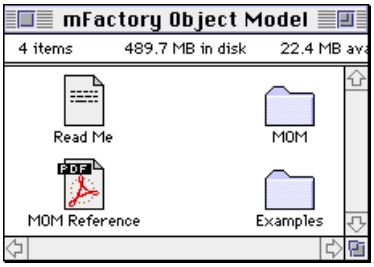


Figure B.2 MOM folders

prefix file), “MOM68K.Lib” (the 680x0 library), and “MOMPPC.Lib” (the PowerPC library). These files, along with the others in the MOM folder, are discussed later in this document.

The second folder, “Examples”, contains three folders: “Kits”, “mTropolis Projects”, and “Source” (see Figure 2). The Kits folder contains the “DemoKit.e68” and “DemoKit.ePP” kits. A kit is a group of MOM components assembled into a single file. The naming convention for the MOM kits is shown in the suffix of the kit: “e” stands for editor; “68” and “PP” indicate the target platforms: 680x0 and PowerPC. The PowerPC version of all the examples is assembled into the “DemoKit.ePP” kit, and the 680x0 versions into the “DemoKit.e68” kit. In order to run the sample mTropolis projects, the appropriate kit needs to be copied (or aliased) to mTropolis’ “Resources” folder prior to launching mTropolis.

The Examples folder also contains the “mTropolis Projects” and “Source” folders. mTropolis Projects contains sample mTropolis projects that demonstrate each example MOM component. The Source folder contains the complete source code, along with the CodeWarrior project, for each example. Five MOM examples are selected to represent the various functional aspects of MOM components. The five examples are Beep, DrawBits, SimpleMessenger, Interface, and Gravity/Momentum.

The Beep example is the very first introduction to writing a MOM component. It presents the general structure of a MOM component, as well as the general interaction with the mTropolis engine via message sending and attribute accessing. The Beep component beeps a number of times when it receives a trigger event. The trigger event is editable in the dialog, while the number of beeps is both editable in the dialog in the mTropolis editor and modifiable via the Miniscript in runtime.

The Interface example presents the MOM dialog editing functionality. It explains the basic building blocks of dialog editing, such as popup menus, buttons, checkboxes, text fields, etc.

The Gravity and Momentum examples illustrate the client-server relationship between MOM components, as well as the timer-based multitask management of MOM components.

The DrawImage example presents a special object-oriented class inheritance mechanism called a “Funk”, which is used to replace the mTropolis element drawing routine with one defined by the MOM developer.

The SimpleMessenger example explains how a MOM component sends a mTropolis message to its element.

The other two items in the MOM folder are “Read Me” and “MOM Reference” documentation (in Adobe Acrobat format).

The ‘Examples’ folder also contains a ‘mTropolis Projects’ and ‘Source’ folder. ‘mTropolis Projects’ contains sample mTropolis projects that demonstrate each example. ‘Source’ con-

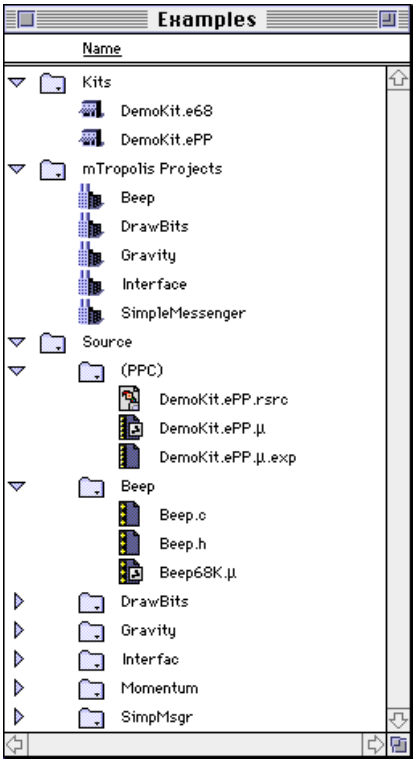


Figure B.3 Contents of the Examples folder

tains the complete source code, along with the CodeWarrior project, for each example.

The other two items in the 'MOM v1.0b1' folder are 'Read Me First' and 'MOM Reference', an Adobe Acrobat file with online documentation for MOM functions.

CodeWarrior Issues — 680x0 Processor

All the examples were built with Metrowerks' Gold CW5.5 version of CodeWarrior. If you are using a different version, some adjustments might be required.

When assembled into a kit, a 680x0 MOM class consists of a code resource (of type 'XCMP') and any supporting resources (such as DITLs, DLOGs, etc.). When using CodeWarrior to build these classes, a few default project preferences must be changed (outlined below).

The language preferences (Figure B.4) are the factory settings with the exception of the prefix file, which is set to 'MMetro.h' (the MOM prefix file, located in the 'MOM v1.0b1:MOM' folder).

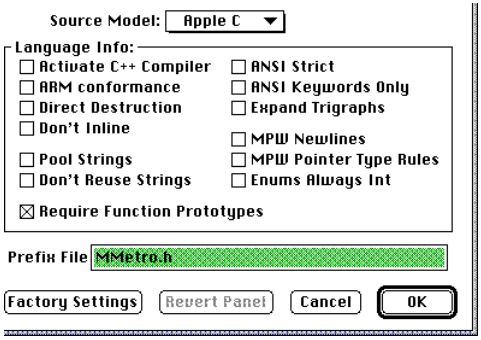


Figure B.4 Language preferences

The processor preferences (Figure B.5) are the factory settings with the exception of the 'Code Model' pop-up, which is set to 'Small',

and the '68020 Codegen' and '4-Byte Ints' check boxes, which are both checked on.

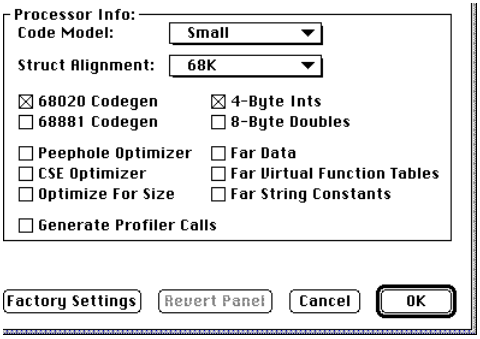


Figure B.5 Processor preferences

The linker preferences (Figure B.6) are the factory settings with the exception of the 'Link Single Segment' check box, which is checked on.

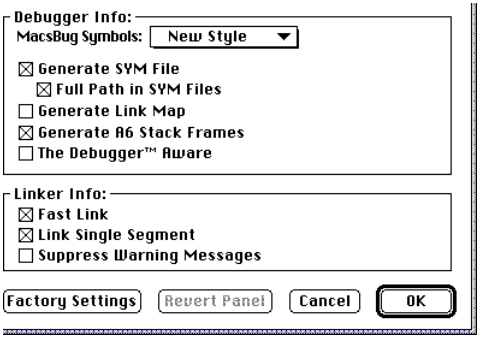


Figure B.6 Linker preferences

The project preferences (Figure B.7) are the factory settings with a few exceptions:

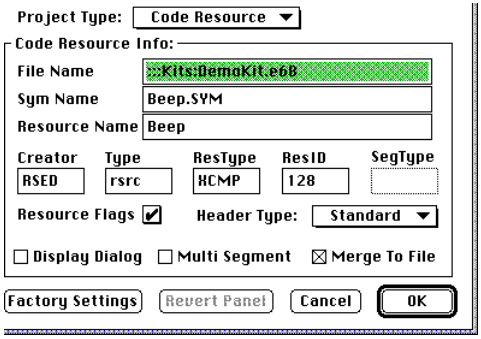


Figure B.7 Project preferences

- 'Project Type' is set to 'Code Resource';
- 'File Name' specifies the name and location of the kit (the kit, even if it's just an empty file, must already exist in the specified location prior to building the code resource);
- 'Sym Name' specifies the symbol file name;
- 'Resource Name' specifies the code resource's name (MOM assigns a unique identifier to your class, referred to as a MOM ID, based upon this resource name);
- 'ResType' is set to 'XCMP';
- 'ResID' specifies the code resource ID (which must be unique for each code resource in the kit); and
- 'Merge To File' is checked on.

The access paths preferences (Figure B.8) are the factory settings with the exception of the addition of the 'MOM' path in the 'User' access path list.

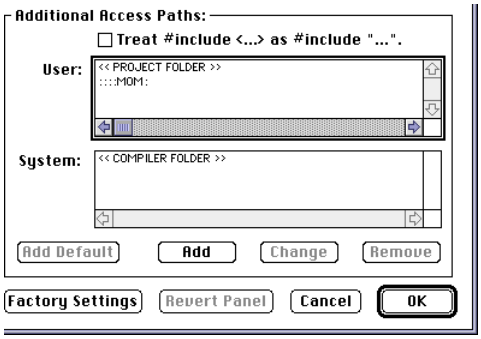


Figure B.8 Access Paths preferences

CodeWarrior Issues — PowerPC Processor

All the examples were built with Metrowerks' Gold CW5.5 version of CodeWarrior. If you are using a different version, some adjustments might be required.

When assembled into a kit, a PowerPC MOM class consists of a data segment and any supporting resources (such as DITLs, DLOGs, etc.). When using CodeWarrior to build these classes, a few default project preferences must be changed (outlined below).

The language preferences (Figure B.9) are the factory settings with the exception of the prefix file, which is set to 'MMetro.h' (the MOM prefix file, located in the 'MOM v1.0b1:MOM' folder).

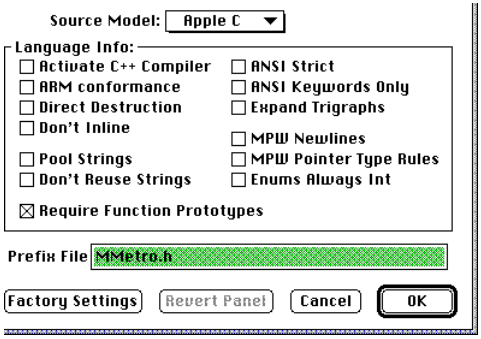


Figure B.9 Language preferences

The processor preferences (Figure B.10) are the factory settings with the exception of the 'Peephole Optimization' checkbox, which is not checked.

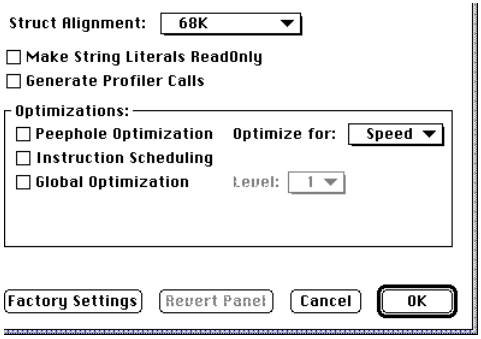


Figure B.10 Processor preferences

The linker preferences (see Figure 10) are the factory settings with the exception of the 'Main' field, which is blank.

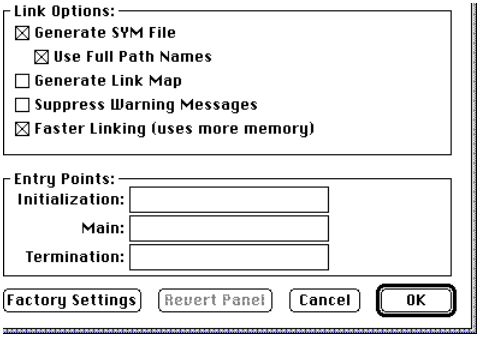


Figure B.11 Linker preferences

The PEF preferences (Figure B.12) are the factory settings with the exception of the 'Export Symbols' pop-up, which is set to 'Use .exp file', and the 'Expand Uninitialized Data' checkbox, which is checked on.

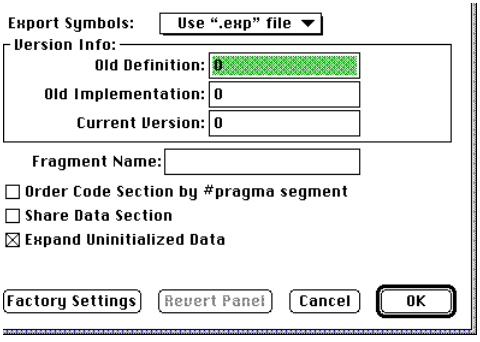


Figure B.12 PEF preferences

The project preferences (Figure B.13) are the factory settings with the exceptions of the 'File Name' field, which specifies the name and location of the kit, the 'Creator' field, which is set to 'MfMf', and the 'Type' field, which is set to 'MFco'.

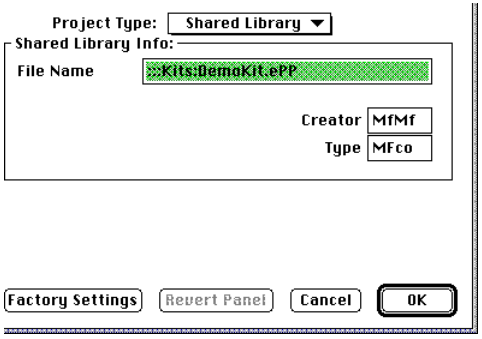


Figure B.13 Project preferences

The access paths preferences (Figure B.14) are the factory settings with the exception of the addition of the 'MOM' path in the 'User' access path list.

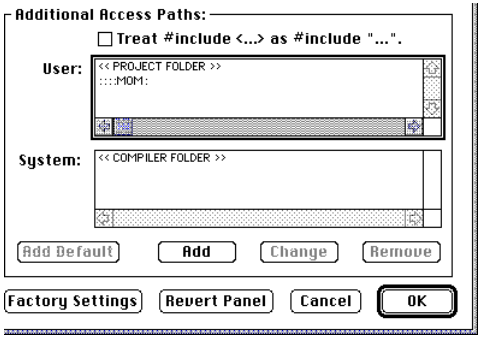


Figure B.14 Access Paths preferences

DEVELOPING MOM COMPONENTS

Derived from mFusion's base class, MComponent (or any other MOM component), a

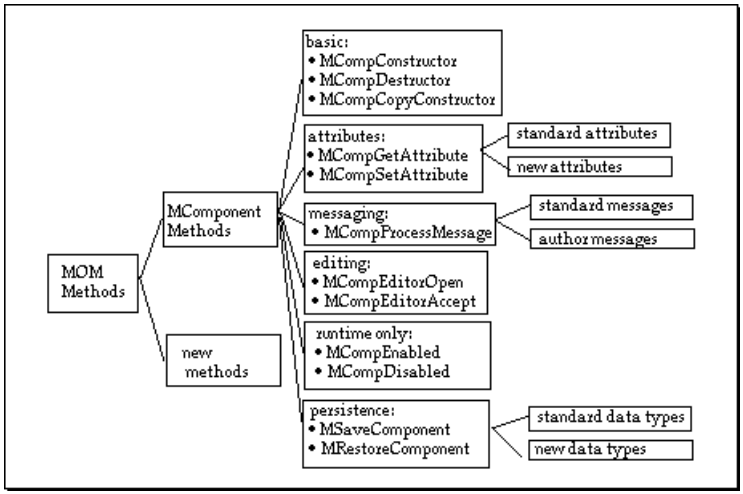


Figure B.15 MComponent methods

MOM component overrides some of the MComponent methods and introduces new methods. The MComponent class has methods for creating and destroying objects, for accessing attributes, for processing messages or events, and for editing mTropolis dialogs. The MComponent's methods are categorized in Figure B.15.

Sending messages and accessing attributes are among the primary underlying mechanisms by which mTropolis objects interact with each other in the mTropolis environment. mTropolis provides three processing levels for object interaction:

- transaction-based messaging via modifiers, e.g. the messenger modifier
- scripting via the miniscript modifier
- function calls via mFusion services APIs

A MOM component can implement methods that directly correspond to each of the above levels. For example, MProcessingMessage implements messaging between MOM components and the mTropolis environment, while MGetAttribute and MSetAttribute enable access to the attributes of MOM components. New methods can also be introduced by a MOM component that can be called directly by other MOM components.

Figure B.16 shows the correspondence between mTropolis functionality and MOM component methods.

Proper usage of the layers is important to maintaining system performance. Processing a message gives a MOM component the greatest amount of flexibility in its response to an event. Messaging is high in overhead (relative to the other layers) and lowest in perfor-

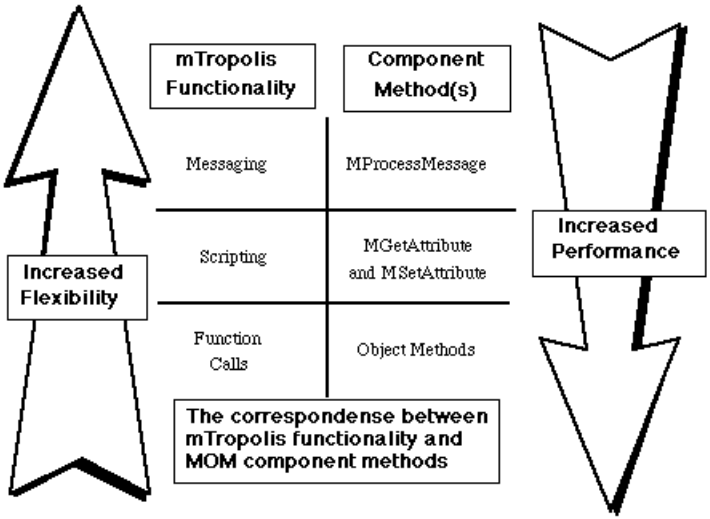


Figure B.16 mTropolis functionality and corresponding MOM methods

mance, so it is best to use it sparingly. Direct method calls are least flexible in their functionality, but provide the greatest performance. Scripting provides a compromise between the other two. Scripting has less overhead than messaging, but does have more processing than direct method calls.

MOM Component Methods

As a derived class from MComponent, a MOM component must override certain methods; to realize its specific functionality, a MOM component can override additional methods and introduce new methods of its own.

The Initialization Method

The function 'CompMainName' is the entry point of a MOM component into mTropolis and is called when mTropolis launches and walks through the MOM kits located in the Resource folder. A MOM component is required to provide its CompMainName defined in mmactype.h for the Macintosh platform and mwintype.h for Windows.

The following macros are used in 'CompMainName', where the 'initInfo' is referred to in some of the macros:

- **MDefineComponent**: MDefineComponent initializes basic class information, such as the size of its data structure.

- **MInheritClass**: MInheritClass specifies the parent class from which this class is derived.
- **MDefineMethod**: MDefineMethod is used to register methods being overridden or newly introduced.
- **MConnectCommonServices**: MConnectCommonServices connects the MOM component to a variety of common, useful services provided by MOM, such as dialog handling, file I/O, list, string handling and standard mathematical functions.
- **MEndComponentDef**: MEndComponentDef declares that there will be no additional calls.

The following table lists the commonly-used methods that a MOM component encounters.

Method Name	Method Functionality	Notes
basic object management		
MCompConstructor	to initialize the primary data structure of the component including allocating memory and setting initial values.	Required: A MOM component must override this method.
MCompDestructor	to dispose of the primary data structure of the component	Required: same as above
MCompCopyConstructor	to provide duplication of the primary data structure	Required: same as above
MCompCopy	to duplicate the primary data structure, used primarily in creating aliases	Required: same as above
attribute accessing		
MCompGetAttribute	to provide information about the MOM component to mTropolis or other MOM components	Conditional: A MOM component that deals with events should override this, see the section on Events, otherwise, a MOM component should override if it adds new attributes that are accessible by others.
MCompSetAttribute	to set the new values of the attributes	Suggested: A MOM component should override if it adds new attributes that are accessible by others.

Table B.1: Commonly-used MOM methods

message passing		
MCompProcessMessage	to process messages sent from mTropolis and other MOM components	Suggested: A MOM component should override this if it processes mTropolis messages.
dialog editing		
MCompEditorOpen	to initialize editor dialog item information.	Suggested: A MOM component should override this if it allows editing through the dialog
MCompEditorAccept	to accept the settings from the dialog by retrieving the settings from the dialog items and saving to the component's data structure.	Suggested: same as above
runtime only		
MCompEnabled	to initialize the component's data and relationship with the runtime environment (runtime only).	Suggested: A MOM component should override this if it allows editing through the dialog
MCompDisabled	to dispose the data and break the relationship with the environment as set in MCompEnabled.	Suggested: same as above
save and restore		
MCompGetSaveInfo	to calculate the total size of data the component needs to save to the persistence, which can be in a format of files or streams.	Suggested: A MOM component that needs to keep any of its data persistent should override this.
MCompSaveComponent	to save all the data the component needs to be kept persistent, must be consistent with MCompGetSaveInfo.	Suggested: same as above
MCompRestoreCompnent	to restore the saved data from a persistent format, must be consistent with what's saved in MCompSaveComponent.	Suggested: same as above

Table B.1: Commonly-used MOM methods

new methods	as needed	Optional: A MOM component can add as many attributes as it needs, including none.

Table B.1: Commonly-used MOM methods

MOM Component Attributes

mTropolis defines a set of standard attributes that are accessed by MOM, and any MOM component can override the standard attributes or introduce new attributes by overriding the MComponent methods 'MGetAttribute' and 'MSetAttribute'.

At the programmable level, MOM provides additional flexibility by allowing the construction of complex attributes. The different parts of a complex attribute can be accessed through a mechanism of attribute selection. A MOM attribute can be fully specified by the following three factors:

- the attribute name, e.g. attribName
- the attribute selector, e.g. selector
- the attribute value, e.g. dataValue

attribute specification: name

MOM attributes have names, which are 16-byte sequence. All 16 bytes must be defined, short names are padded with '\0'. MOM attribute names can be case sensitive when directly used in component methods calls such as MGetAttribute and MSetAttribute; however, since Miniscript uses lowercase, any attribute a MOM component allows Miniscript to access should use lowercase.

attribute specification: selector

As an example, a QuickTime movie track modifier utilizes the selector in implementing an attribute called "length". A movie track can contain multiple videos tracks and multiple audio tracks, and each video track or audio track can be identified by either the index or the name of the track. The movie track modifier thus needs to determine whether to apply the length attribute to one particular video track, to one particular video track, or to all of the tracks. The modifier uses the selector to make this choice.

A MOM component can specify the selector in any way it desires, as long as objects (other MOM components or mTropolis) with which the MOM component communicate follow the same specification.

attribute specification: value

The specification of the value varies according to whether the attribute is to be retrieved or modified, that is, the specification of dataValue varies in MGetAttribute() and MSetAttribute(), and should be agreed upon by both the caller and the implementor. In updating an attribute, the caller of MSetAttribute() needs to fully specify dataValue, both the type and the value. MOM's implementation of MSetAttribute() should check the type of the dataValue for the given attribute.

In attribute retrieval, on the other hand, the caller of MGetAttribute(), can either specify the expected data type or set the data type to kMPreferredCopy. In the former case, MOM's implementation of MGetAttribute() should try to convert the value into the requested type, and returns kMUnableToComplyCompErr when it cannot provide an attribute in the requested data type. In the latter case, however, MOM's implementation of MGetAttribute() should just return its default data type.

A MOM component accesses mTropolis engine's attributes using the component service function call MGetElementAttribute(), while a MOM component accesses another MOM's attributes using MGetAttribute().

kMEventsAttrib

In mTropolis, the behavior modifier graphically shows components' events, including:

- executing events: the events that trigger the component's runtime execution, for example, the execute event for the Miniscript modifier
- terminating events: the events that stops the component's runtime execution, for example, the disable event for the sound panning modifier
- sending events: the events that the component generates and sends to other objects, for example the event the messenger sends when it gets executed

Any MOM component that receives events for execution or termination or sends events is required to override the method MGetAttribute. The behavior pane requests the events attribute according to the following conventions:

(1) the behavior pane sets the data type of selector to MInteger.

(2) the highest byte of the value of the selector contain the type of the event requested, values defined in mevent.h are:

- kMEventExecute
- kMEventTerminate
- kMEventSend

(3) the low 3-bytes of the value of the selector contains the index (starting at 1) of the event of the given type. The MOM should return kMUnableToComplyCompErr when the index exceeds what it expects. For example, if a MOM component has only one triggering event, it should return kMUnableToComplyCompErr when the index is set greater than one.

The following is an example of a MOM component that responds to two execution events, two termination events and one sending event, and here's how it handles the kMEventsAttrib in MGetAttribute method overriding:

```
static MErr MCompGetAttribute(YourComp *self, MomID attribName, MDataType *selector,
        MDataType *dataValue )
{
   // must support the Events attribute for the Behavior pane
   if (MCmpMomID( attribName, kMEventsAttrib)  && selector &&
        selector->f_type.f_type == kMInteger )
   {
          switch( selector->f_integer.f_value & ~kMEventMask)
         {
      case kMEventExecute :
      {
         if ((selector->f_integer.f_value & kMEventMask) == 1)
         {
              MCopyEvent( dataValue->f_event, self->f_executeEvent1);
             return kMNoCompErr;
         }
         else if ((selector->f_integer.f_value & kMEventMask) == 2)
         {
              MCopyEvent( dataValue->f_event, self->f_executeEvent2);
             return kMNoCompErr;
         }
         else
            return kMUnableToComplyCompErr;
      }

      case kMEventTerminate:
      {
         if ((selector->f_integer.f_value & kMEventMask) == 1)
         {
              MCopyEvent( dataValue->f_event, self->f_terminateEvent1);
             return kMNoCompErr;
         }
         else if ((selector->f_integer.f_value & kMEventMask) == 2)
         {
              MCopyEvent( dataValue->f_event, self->f_terminateEvent2);
             return kMNoCompErr;
         }
         else
            return kMUnableToComplyCompErr;
      }

      case kMEventSend:
      {
         if ((selector->f_integer.f_value & kMEventMask) == 1)
         {
              MCopyEvent( dataValue->f_event, self->f_sendEvent);
             return kMNoCompErr;
         }
         else
            return kMUnableToComplyCompErr;
      }

      default :
         return kMUnableToComplyCompErr;
            }
   }
   else
      return kMUnableToComplyCompErr;
}
```

Messaging in MOM

Messaging in mTropolis can be carried out in three ways:

- use a messenger modifier by specifying the message to send, the "with data", the destination of the recipient of the message, and the message passing options.
- use a Miniscript modifier, e.g.,

```
send "greeting" to element \
     with "hello world"
```

  sends an author message "greeting" to the element containing the Miniscript modifier along with the string "hello world".

- use the mFusion core service within a MOM component, i.e., MSendMessage(), by initializing the mTropolis message structure and sending the message to the desired recipient.

A MOM component can receive messages by overriding the MComponent methods MCompProcessMessage, and send messages to any object desired in the project using MSendMessage().

Specifying a MOM message

All MOM messages contain the following information. Some fields must be specified and others may be left unspecified and will default to their predefined values:

- sender: the originator of the message. The specification of the sender is mandatory for messages originated from any object within mTropolis rather than from the system such as a mouse event.
- target: the recipient of the message. The specification of the target is obviously mandatory. The message target can be specified through various representations, as defined in mdata.h. Currently supported is specifying the target via a MObjectPtr.
- event: the identifier of the event or command sent. The specification of the event is also mandatory. The event ID for MOM messages can be either a standard event ID defined in mTropolis or an author message defined by users in editor mode.
- withdata: any data accompanying with the message. The specification of "withdata" is strictly optional. The default is "no with data".
- options: the message passing options. The specification of messaging options is optional, as the default is 0, meaning the message passing is immediate, cascaded and relayed. Please refer to the mTropolis reference guide for definitions of immediate, cascaded and relayed message passing. Options defined are:

  kMQueueMessageMask: setting this flag means the message is not immediate

  kMNarrowcastMessageMask: setting this flag means the message is not cascaded

  kMAbsorbMessageMask: setting this flag means the message is not relayed

- The above options are orthogonal to each other, thus are to be combined using the logic-or operation.

mFusion provides core services to create the message and send it:

- **MInitSimpleMessage**: create a simple message structure
- **MInitMessage**: create a message structure
- **MSendMessage**: send the message

MOM Data Types

mFusion provides the definition and implementation of data types that are used by mFusion services and MOM components. The definition of MOM data types includes the structure definition and basic management routines (macros or function calls) for the creation, disposal and duplication of the data types. The implementation of MOM data types enables data exchange between MOM components and mFusion to be uniform and consistent in all aspects, including basic object management, dialog editing, message sending, attribute access, and persistence.

MOM ID

A MOM ID is a unique 16-byte sequence (of type 'MomID') that identifies a MOM class. Your own component's MOM ID is the first 16 characters of its 'XCMP' code resource's name, for 680x0 components, and the first 16 characters of whatever 'ComponentName' has been defined as in your component's header file. If the code resource name is less than 16 bytes long, it will be padded by the system. A space denotes the end of the MOM ID.

MOM IDs are used for several important tasks. First, a MOM component must specify the parent class from which it is derived, and the parent class is specified by its MOM ID. Second, mTropolis uses the MOM ID to fetch certain resources from the kit, such as its cursor, dialog, and string resources. These resources thus must have the same name as their component's code resource. Finally, when a component communicates with another, it uses the MOM ID of that component. For example, a physical property modifier refers to a gravity service's MOM ID when registering with that service.

The following macros are used in the handling of MOM ID's:

- **C2Mom**: converts a 'C' string to a MOM ID, and pads this ID up to 16 bytes
- **MCmpMomID**: compares two MOM ID's, returning true if they're the same
- **CopyMomID**: copies one MOM ID to another

MDataType

MDataType is the primary MOM data structure for MOM components. It is a union of many MOM data structures, such as MRect and MString, whose first field is f_type to identify the type. This first field makes MDataType a self-identifying polymorphic structure. mFusion provides macros for the initialization, disposal and duplication of MDataTypes. For every MDataType, the macros are constructed according to the following model (taking MInteger as an example):

- **MInitInteger**: initialize a MDataType with initial value(s)

- **MCopyInteger**: duplicate one instance of MDataType to another of the same type
- **MDisposeInteger**: dispose a MDataType

mFusion also provides services for the saving and restoring of MDataTypes:

- **MSizeofValue**: calculate the size of data to be saved from a MDataType
- **MWriteValue**: save a MDataType to a persistence (a file or stream)
- **MReadValue**: restore a MDataType from a persistence

A MOM component should use MDataType and associated macros for better performance and portability of MOM components, especially the data that is saved to the project.

Dynamic Data Types: MString and MPointer

Among the currently supported MDataTypes, MString and MPointer manage the dynamic and variable length memory allocation. A MOM component needs to understand how they manage the dynamic buffers to prevent memory related problems including memory leaks. MString and MPointer are essentially the same data structure, except that MStringPtr is primarily used to store a variable length string, and MPointer to store any type of variable length data structure. Their data structures are defined as follows (defined in mdata.h):

```
typedef struct {
   short f_type;  //kMStringPtr
   void* f_owner;
   long  f_bufSize;  // allocated buffer size of f_ptr
   long  f_strLen; // current data or string length in f_ptr,
                   //excluding null-terminator if any
   char  *f_ptr;
} MString;
```

```
typedef struct {
   short f_type;  //kMPointer
   void* f_owner;
   long  f_bufSize;  // allocated buffer size of f_ptr
   long  f_dataLen;  // current data length in f_ptr
   void  *f_ptr;
} MPointer;
```

The field f_owner is used to indicate who allocates the memory in f_ptr and thus is responsible for disposing it. A MOM component should dispose the memory in f_ptr only if f_owner is itself. The field f_bufSize indicates the size of allocated buffer. f_dataLen in MPointer is required to be set to indicate the length of the data in f_ptr. f_strLen in MString is the length of the string data, excluding the null terminator. It is required to set f_strLen, while it is not required to append the null terminator to f_ptr. Thus when a MOM component receives a MString data type, it should NOT assume the string data in f_ptr is null terminated.

When a MOM component restores a MString or MPointer, mFusion fully restores it by allocating as much memory for f_ptr as when it was saved and reads in as many bytes in f_ptr as were saved. For example, when a MOM component saves a MString whose f_bufSize is 256 and f_strLen is 16, in restoring the component, mFusion allocates a buffer of 256 bytes long and reads 16 bytes from the persistence.

MOM Funk

As stated in earlier sections, MComponent serves as the base class from which all MOM components are derived. However, mFusion provides a mechanism for a MOM component to be inherited from another class within

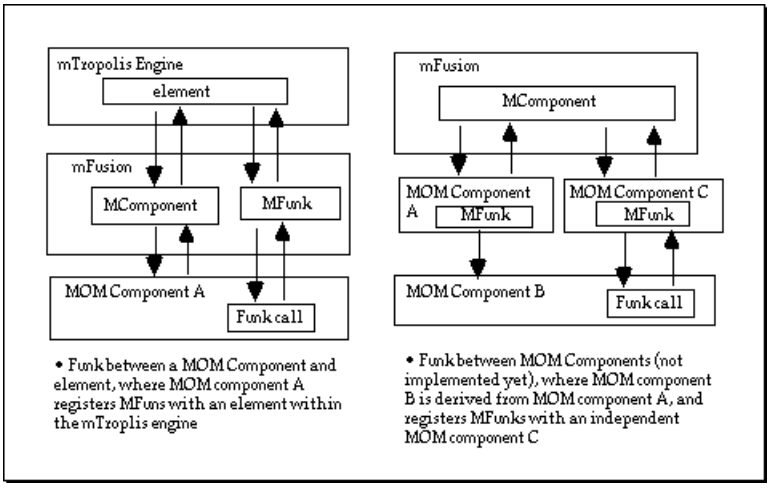


Figure B.17 Illustrating MOM Funk

the mTropolis engine that is not the superclass of MComponent. Such a mechanism is called a Funk.

The purpose of a Funk is twofold: one is to allow simple and efficient multiple class inheritance, another is to allow MOM components to customize some essential functionality within mTropolis for their special needs and high performance. For example, a MOM component might want to provide a special graphical effect on the element transition, it can then use the buffered drawing funk, a predefined mTropolis funk, by accessing the GWorld and directly drawing onto the GWorld buffer.

Although currently not implemented, Funk can be extended to any two classes, including two MOM components, where either class is not derived from another. Such an extension will provide a general mechanism of multiple class inheritance among MOM components.

Besides providing the multiple class inheritance for flexibility and performance, Funk introduces a new level of interaction between MOM components and the mTropolis engine. With a Funk, a MOM component can communicate with the mTropolis engine in the following three ways:

- send messages
- access attributes
- use funks

The relationship among MOM components, the mTropolis engine and Funks are depicted in Figure B.17.

A MOM component that uses the Funk mechanism needs to first initialize the funk

data structure, and then register the funk with the desired element within the mTropolis engine. The element will then call the registered funk for a particular purpose. A MOM component needs to unregister the funk from the element when it wants to stop the element from calling the funk.

The MFunk data structure is defined in mom.h as follows:

```
typedef struct MFunk {
   short f_code;  // the funk type or id
   void* f_comp;  // your component
   short f_slot;  // your component slot
   short f_index; //the index of the
            //function to be called in your component slot
   long  f_ID;    // ID number of this particular Funk
   long  f_data;  // User-defined refcon for this Funk
   long  f_peckingOrder;   // Where this Funk stands
                           // relative to its peers
} MFunk ;
```

mFusion provides macros for initializing, registering and unregistering a funk to an element:

- **MInitFunk**: initialize a mFunk structure
- **MRegisterFunk**: register a funk with an element
- **MUnregisterFunk**: unregistered a funk from an element

Currently defined and implemented funk types are listed in the following:

- **kBufferedDrawFunk**: to override the buffered drawing
- **k2DPredrawFunk**: to override the 2D shape border

MOM Services

mFusion provides a number of core services for the development of MOM components. Currently released services include those shown in Table B.2.

MComponent Methods	MComponent methods that a MOM component can override
Component Service	services for managing MOM services, and client-server relationship between MOM components; services for registering and unregistering Funk; services for initializing and sending messages
Debug Service	services for posting messages in the mTropolis message log for the purpose of debugging
Edit Service	services for dialog editing for MOM components
File Service	services for saving and restoring MOM components
List Service	services for managing a dynamic general-purpose list data structure.
Math Service	wrappers for ANSI Math library function calls
Memory Service	services for managing pointer-based and handle-based memory
Miscellaneous Service	services that do not fit anywhere else
String Service	wrappers for ANSI String library function calls
Task Service	services for managing timer tasks

Table B.2: Core services for MOM components

MOM Error Handling

mFusion service calls are implemented as function macros, and don't directly return the status, rather the service call sets a global variable g_err to the status. A MOM component can then access g_err by using the macro MError() to retrieve the return status of the last service call. The following is an example:

```
static MErr _getMyElementSize(YourComp *self)
{
   MErr        ret;
   MPoint2D    pos;
   MObjectRef  *element = nil;

   MGetElement(&element);
   if ((ret = MError()) != kMNoCompErr)
      return ret;

   MInitPoint2D(pos, 0, 0);
   MGetElementAttribute(element, kMSizeAttrib, nil,
(MDataType *)&pos);
   if ((ret = MError()) != kMNoCompErr)
      return ret;

   //other things to take care
   return kMNoCompErr;
}
```

USER INTERFACE GUIDELINES

The set of modifiers that accompanies mTropolis has incorporated, wherever possible, a consistent approach to layout. The following is a breakdown of each dialog element along with images of existing modifier dialogs. When designing anything with a user interface, it is suggested that you follow these guidelines.

Icon and Name

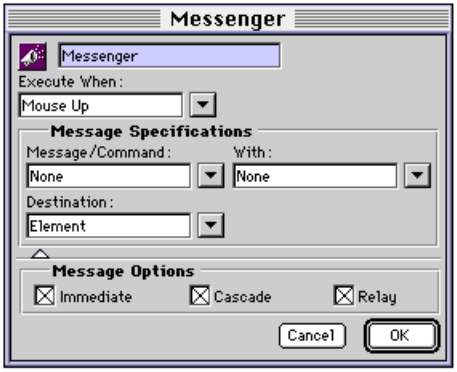
The modifier's icon is positioned in the upper left corner of the dialog, followed by its name field.

Message Pop-ups

In the upper portion of the dialog are the message pop-ups, if they exist, which cause the specifications of the modifier to be executed and/or terminated. If there are two message pop-ups, they are placed side by side; this usually helps to determine the total width of the dialog.

Pop-ups in General

The label for each pop-up is placed above it, capitalized, given a colon and left-aligned. Type-in pop-ups have been used throughout to allow the user to type a choice directly, to tab, as well as to cut and paste between pop-ups. In instances where a new menu item may be created by the user, the name may be keyed in directly to generate this new item. We also intend to provide search ahead functionality to increase productivity.



Specifications Enclosure

The specifications of the modifier are positioned below the message pop-ups and are enclosed by a beveled border. This area encompasses all that will be executed when the message is received. In the case of a messenger, the 'Message/Command' pop-up is placed first, and the 'With' pop-up is placed to its right. The 'Destination' pop-up is placed directly under the 'Message/Command' pop-up. In other instances, the choices are

broken into logical groups, and placed within sub-enclosures if necessary, for the sake of clarity. Any additional options which require an enclosure separate from Specifications are labeled in as clear a manner as possible.

Checkboxes and Radio Buttons

When displayed horizontally, checkboxes and radio buttons are distributed evenly in the space allowed. When they are displayed vertically, space is conserved as much as possible by placing them tightly together.

Arrow Controls

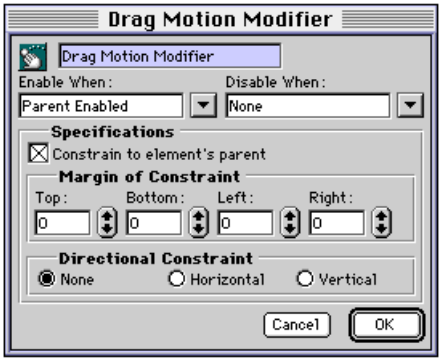
An arrow control is placed to the right of the field it controls.

Drop-down Sections

To conserve real estate, in some cases drop-down sections have been incorporated to hide less frequently used options. This can also be a way of incrementally introducing complexity. Another enclosure can be used here to contain the additional options. The triangles which rest on the dividing line act as a lever to hide and reveal this drop-down section. When the dialog is fully expanded, the triangle points upward and when the dialog is collapsed, it points downward.

OK and Cancel Buttons

The OK and Cancel buttons are positioned to the right and aligned with the rightmost object.



Modifier Names

Modifiers themselves are named, wherever possible, in a simple and clear manner. For example, Change Scene Modifier, Scene Transition Modifier, and Simple Motion Modifier all attempt to convey their intended usage through their names. There are other cases where a straightforward name was too limiting and a more abstract one was chosen. For example, Behavior has been used to describe a modifier which encapsulates other modifiers and provides an all-in-one reusable "behavior".

Naming Conventions Within Dialogs

An effort has been made to label pop-up fields consistently within all modifier dialogs. Beginning with the message pop-ups at the top of many modifiers, three models currently exist for naming. The first, Enable When and Disable When can be used when the message received will not actually cause the

specifications to be executed, but will simply make the modifier receptive to that eventual action. For example, in the case of the Drag Motion Modifier, the modifier may be enabled on the receipt of a Parent Enabled message, however nothing will occur until the user physically drags the element.

The second, Execute and Terminate, will actually cause the specifications to be carried out upon the receipt of a message. The third model, Apply and Remove, has exactly the same meaning only different wording has been chosen to more closely map to the specifications of certain mods. For example, in the case of the Graphic Modifier, Apply and Remove sounded more appropriate for graphic effects than Execute and Terminate. In the case of the Sound Fade Modifier, Execute and Terminate sounded more appropriate.

The term Specifications has been chosen to encompass the items that will be executed when the message is received. In the case of Variables, the term Value is used to encompass the values which are stored.

Icon Design

Icons are used in mTropolis as a visual aid but, unlike tool icons, are not intended to be literally and directly interpreted. Because so many of the concepts behind the modifiers are complex, the usual approach to icon design (a design which not only aids in memorization but also in gleaning the intended purpose) cannot be implemented. Instead, the icons are meant to aid the user in distinguishing one modifier from another once an understanding of its functionality has been attained. Taking into consideration the potential for a large number of mods to be developed in the future, grouping is an important method of organization. Color has been used to establish distinct groups of modifiers. Gray scale icons have been used almost exclusively in combination with a solid color background to provide a clean and consistent appearance. The following is a breakdown of each modifier group and a sample of some of the mods in that group.

Messengers

A dark red has been chosen to identify the group of mods known as messengers. A secondary, unifying theme has been added to assist in identifying them. Broken lines help to indicate action and perhaps, communication or broadcasting.

Effects

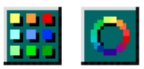
These mods have been grouped together with a dark green background. Their similarity lies in their ability to cause a graphical or sound effect to be applied (generally speaking, of course). The Gradient, Graphic and Sound mods fit nicely into this category whereas the Change Scene modifier and Return Modifier could conceivably, be placed in a group of their own.

Motion

Although these mods reside within the Effects group, they are unified by their dotted motion lines. This helps to define them as a family within a group. When more mods are created that are motion based, another group may be established.

Use of Color

Modifier icons are designed in gray scale almost exclusively. Where color is absolutely necessary to convey the functionality of the modifier, it has been used. For example, in the case of the Color Table Modifier, it has helped to convey its intended purpose whereas gray scale may have caused confusion. As well, in the case of the Graphic modifier, color plays a key role in the functionality of this modifier and was therefore essential in the design of its icon.



Variables

These mods are distinguished by a dark blue background as well as mostly numerical or alphabetical icons.

Mods Without a Group

Neither the Miniscript modifier nor the Behavior fit into a category as of yet. If other mods are designed that share enough traits to define a category, a new group will be formed. Until then, pale pastel colors have been used to indicate their "solitary" status.

Functions

With the introduction of new modifiers to the existing collection, occasionally a new group is necessary. The Save/Restore modifier and the Shared Scene modifier are recent additions to the mTropolis collection and share some characteristics with the existing Set Value modifier. This new group has been called Functions and a dark gray has been used to distinguish them.

Identifying New Modifiers

For the latest modifiers, a red dot in the corner has been used to indicate that they are very new to the collection and are still in an experimental stage.

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z