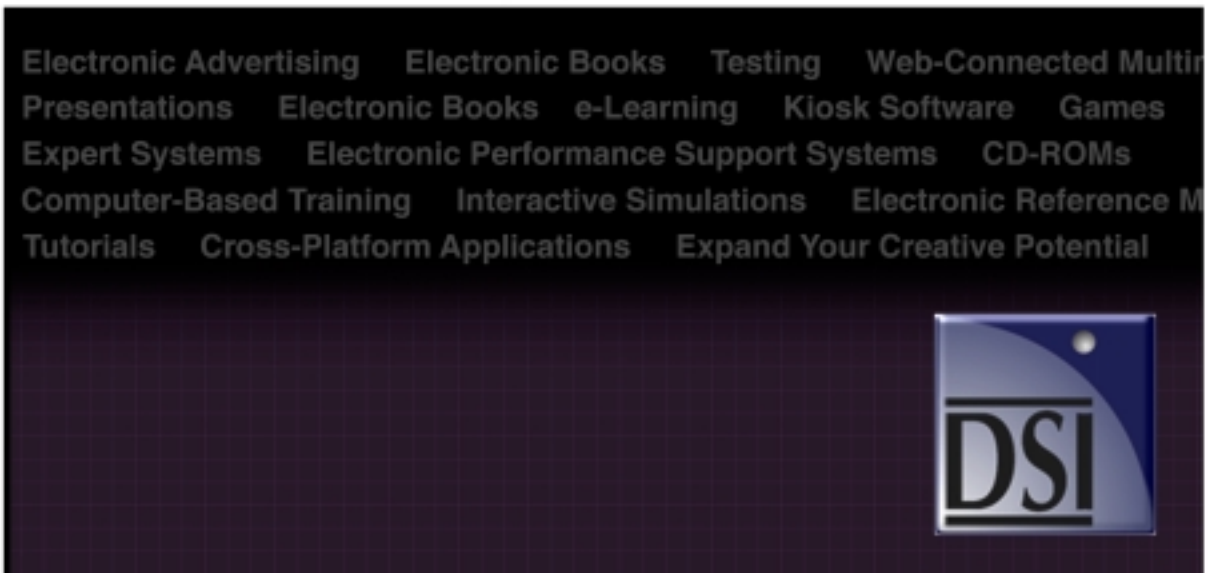
REFERENCE MANUAL

Multimedia Authoring System



onViz™ Reference Manual
Version 1.0b2 for use with Apple Macintosh Computers

Licensing & Copyright Agreement

Copyright © 2001 by Discovery Systems International, Inc. All rights reserved. Except as permitted under copyright law, no part of this reference manual may be reproduced or distributed in any form or by any means without the prior written permission of Discovery systems.

onViz program, Interior of the Application Map, Group Maps and Condition Window ©1987-2001 by Discovery Systems International, Inc. all right reserved.

Portions of this program are licensed from Synergistic Applications, Inc.

Limitations on Warranty and Liability

Neither Discovery Systems International, Inc., nor its distributors and dealers, make any warranties or representations, either expressed or implied, as to the software and documentation, including without limitation, the condition of the software and the implied warranties of its merchantability or fitness for a particular purpose. Discovery Systems shall not be liable for any lost profits or any direct, indirect, incidental, consequential or other damages suffered by licensee or others resulting from the use of the program or arising out of any breach of warranty.

Trademarks and other Copyrights

onViz is a trademark of Discovery Systems International, Inc.

Apple, Macintosh, Finder, QuickTime, and MacinTalk are registered trademarks of Apple Computer, Inc.

Windows is a trademark of Microsoft Corp.

Credits

Reference Manual writing: Joe Slagle, Janice Tocher

Cover design: Visual Resources

onViz programming: Paul Satterlee, Roger Kittelson, Bill Appleton

Our thanks to our users for their continued support and excellent suggestions for enhancements to Discovery Systems' product line.

This release of onViz is dedicated to our fond memories of Scott Didlake

Welcome to onViz

onViz is a rapid development tool for creating stand-alone multimedia and courseware applications that can run on Macintosh, Windows, Intranets, and the Web.

Originally started as an upgrade to Discovery Systems' CourseBuilder product, we decided to completely rework the product from the ground up. While we kept some of the features that made CourseBuilder popular, there is so much more "under the hood" that it simply isn't the same program.

onViz is the result of many years of development and evolution based on user comments, feedback, and wishlists.

Our goal is to let you meet your teaching, training, and presentation needs using advanced and efficient technology. Feedback from our users helps us continue to create products which meet your needs – and we are always responsive to your suggestions. One of our greatest pleasures is seeing the results of your efforts, whether it is an application on radon detection, a discussion of the finer points of basketball, an application teaching management skills, whatever your vision may lead you to create...

Thank you for adding onViz to your repertoire of software. Please let us know how we can be of service to you - give us a call at 888.284.5389 or email us at support@discoverysystems.com.

onViz v1.0b2 Documentation

As onViz continues to evolve toward its final release, so does this documentation. You will note that several sections are incomplete. However, there is a wealth of information in this Reference Manual to assist you in using onViz and incorporating its powerful features into your own interactive creation. Feel free to contact us for details or further information on any of onViz' feature documented or undocumented.

Table of Contents

Contents

Chapter 1

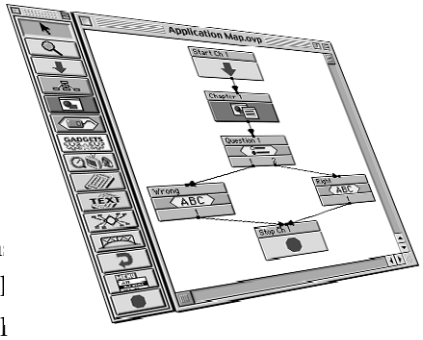
Introduction

Covered in this Chapter:

- Welcome to onViz!
- Application Considerations
- onViz File Types
- System Requirements
- Target Machines and Application Requirements

Welcome to onViz!

Welcome to the onViz world of interactive computing! Get ready to learn how to create exciting, interactive courseware, presentations, surveys, games, or any other multimedia application you can imagine!



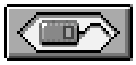
onViz' visual metaphor for creating interactive applications frees you from the need of learning complicated programming or scripting languages, and lets you focus instead on content. Structure your application with the ea creating a flow chart. Use onViz' built in editors to create l quality graphics and sophisticated animations and special effects. Ask questions, score and capture replies, and even create custom reports. onViz gives you the tools to create your own applications as quickly as you can imagine them. Let onViz help you "Turn on your vision!"

The onViz Metaphor

If you've ever used a map to plan a trip, then you may appreciate the metaphor around which onViz is designed. When planning a trip, you flatten out the map and choose a particular route that will take you through one or more states until you reach your destination. Similarly, onViz allows you to map out the route your audience will take through the applications you create. In some of your applications, you may take all of your users down a straight path to the same destination, while in others their paths may diverge based on the answers they give and the choices they make. Straightforward or complex, the onViz Application Map allows you to plan your users' journeys visually, without the need to learn complex scripting or programming languages.

From within the onViz Application Map, you'll organize your application into a series of 'States' (icons) that are connected with Routes. Some of onViz States are:

Outputs, which display text, graphics, and animation.



Inputs, which capture user responses in the form of keyboard or mouse.

Multimedia States, for incorporating content such as movies or sound files.

Gadgets, which allow you to perform a variety of tasks such as sending email, connecting to the Internet, changing cursors, or setting report options.

Simply drag and drop these States onto the Application Map, add your content, and connect them with Routes, and you're on your way to creating impressive interactive multimedia presentations.

Application Considerations

Before launching onViz, take some time to think about the application you want to create. There are some considerations you may want to keep in mind while setting up your project, such as:

- What type of application is it (informational, inspirational, survey, e-learning, testing, electronic book)?
- Who is my audience (number, age, computer proficiency)?
- What is their computing environment (operating system, average processor speed, network/Internet connection)?

Factors such as these will influence the way you set up your application, including:

- required system configuration;
- if it is cross-platform;
- if it is a single, self-contained file or a top level application with supporting files;
- if it is delivered from a local drive, CD-ROM, network, or Internet;
- if it is customizable by the user.

If your application is informational, you may not be concerned with security issues such as verifying your user's identity or protecting application resources from prying eyes. On the other hand, even a kiosk application may include copyrighted materials that should be safeguarded against unauthorized access. By saving your application as a single self-contained file, you can protect its media assets. Likewise, a multiple file application can have its media assets secured by delivering them over a network or the Internet.

The number of users in your audience may affect several aspects of your new application. If your application is to be used for training, you may want to consider issuing user IDs and passwords, and setting up a process that will automatically send you an email with the users' scores and statistics.

Your audience's age and computer proficiency will influence your application's content. Should your tone be formal or informal? How thoroughly should menus, icons, and other navigational aids and computing terms be explained? How large should you make your text and graphics? An elderly audience may have trouble reading text smaller than twelve points, while children may have trouble clicking small icons.

Your audience's computing environment will be a major determinant in setting up your project. Does your audience use Windows or the Macintosh OS, or both? onViz gives you several options for creating cross-platform applications. How powerful are your audience's computers? A good practice is to target your application to your audience's weakest computers - design for the 'lowest common denominator.'

The way in which your audience's computers are connected and the types of removable media they support are also factors in determining your options for delivering the application. If your audience's computers are networked, and/or connected to the Internet, you can

take advantage of onViz' features that let you select which files you want to run locally, and which you want to deliver remotely.

However, if your audience is not on a network or not connected to the Internet, you may need to investigate other types of removable media your audience can access. Many computers have CD-ROM drives but no floppy drives, or vice-versa, while many use other forms of removable media. These factors can affect whether your application is single file, multiple file, or a hybrid that has integrated files below a certain size.

This chapter covers the difference between types of onViz documents, and other options determined using the Project Setup dialogs, such as target system requirements. Although several issues discussed above are not covered in detail until the next chapter, Getting Started and Delivering, it is still helpful to be aware of them while going through the Project Setup dialogs.

onViz Files

The files you create with onViz are referred to as top-level applications. Future versions of onViz will also support saving files as supporting documents that can be linked to your top-level application.

Top-level Applications

onViz offers two methods for setting up top-level applications: single-file or multiple-file. A single-file application has the advantage of containing all of its media assets, such as images, sounds, fonts, and movie clips. This method gives you the convenience of being able to distribute your application as a single file with no chance of supporting files being lost or not installed correctly. While this method may be fine for some onViz projects, others may be better suited for distribution aqs a multiple-file application.

A multiple-file application has its media assets contained within support folders.

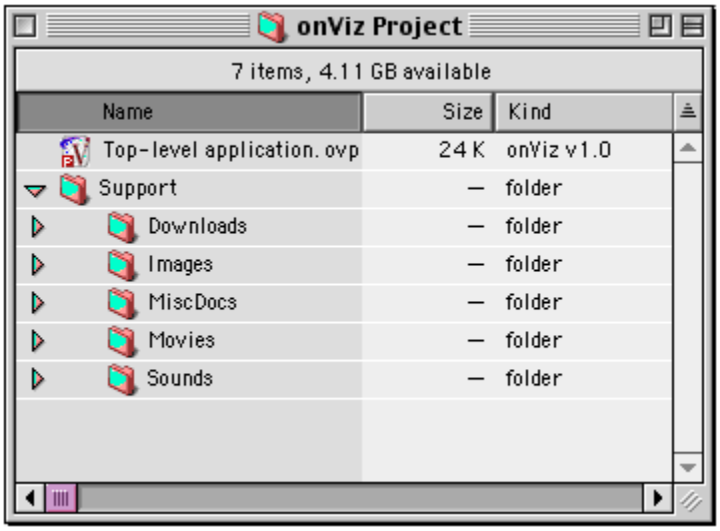


Figure 1.1: onViz' project folder hierarchy.

When an onViz project is set up in this way, onViz creates a hierarchy of folders that can be accessed by the top-level application. In this manner, the overall file size of a large project can be kept to a minimum, as all of its onViz files are drawing from a common library of media assets, rather than each having its own. In addition, since onViz' supporting files are common to both Macintosh and Windows, setting up a multiple-file application give greater flexibility in creating cross-platorm applications.

When distributing a multiple-file application, its support folder must be either distributed with it, or available 1x when the user runs the application.

During development, onViz saves all of the application's media assets in its support folders. By doing so, all of these files can be easily accessed for changes and updates. For instance, if a graphic containing your company's phone number is used in several places throughout your application, and the phone number changes, you need only update the original graphic in the support folders, and the graphic is updated wherever it appears in your application.

The decision to create either a single-file or a multiple-file application is made within the Build Target dialog - giving the flexibility of being able to save out many different delivery configurations from one single project.

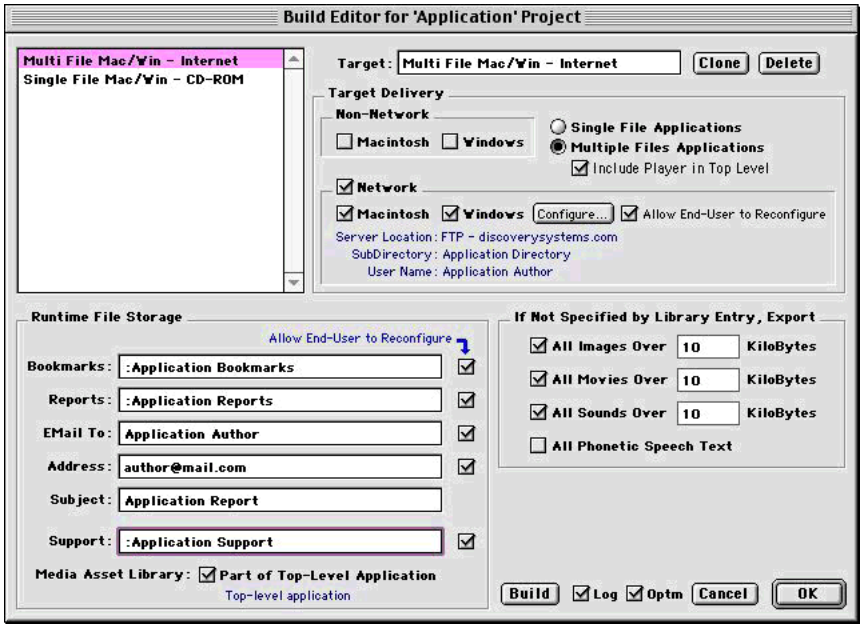


Figure 1.2: The onViz Build Target editor allows you to set up your project with a variety of delivery options.

Accessed from the **File** menu, the Build Target dialog lets you choose your application's target platform, whether or not it will be delivered over a network, and whether or not your end-users will have the ability to customize certain settings, such as report and bookmark file paths and email information. When building a multiple-file application, this dialog also gives you the option to integrate media assets below a designated file size. This option is particularly useful for network delivery; you can integrate many small, repetitively used files, such as fonts or icons, into the top-level application, and then deliver its supporting documents and large multimedia files over the network.

System Requirements

Authoring

- Macintosh OS 8.6 or above
- At least 8 MB free RAM*
- Carbon Lib 1.3.1
- QuickTime 3 or above
- Program requires 2.5 MB disk space*

*actual requirements dependent on application being created

Runtime

Macintosh:

- Macintosh OS 8.6 or above
- At least 8 MB free RAM
- Carbon Lib 1.3.1
- QuickTime*

Windows:

- Windows 95 or later
- Direct X*
- Sound Card*
- QuickTime†

* Required only if your application includes sound files

† Required only if your application includes QuickTime movies, QuickTime VR, or MP3 files

Target Machines and Application Requirements

When setting up a new onViz top-level application, you can specify certain runtime attributes for your application. Two such attributes are target screen size and color depth, chosen from dropdown menus in onViz' Project Setup dialogs.

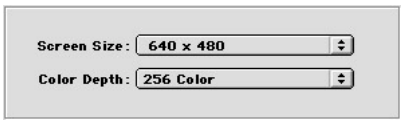


Figure 1.3: Settings for target display attributes.

These options activate features within onViz' development environment that help you design your application with these target attributes in mind. For example, choosing **a s creen size of 640 x 480** displays a 640 x 480 Target Screen Size boundary area in your Output Presentation windows and the onViz Image Editor. Choosing **256 Color** from the **Color Depth** dropdown menu causes onViz to display a warning whenever a 16 or 32-bit bitmap image is created or imported into onViz. In order to determine what options to choose for these settings, you need to determine the configuration of your target audience's computers, on which your application will be run; these computers are your application's target machines. While you can make these selections during initial project setup through the Setup Helper, these options can also be changed at any time in development of your project through the **Project Attributes** menu item.

You can set up your onViz application to check your users' computer display settings and installed components to ensure proper running of the application.

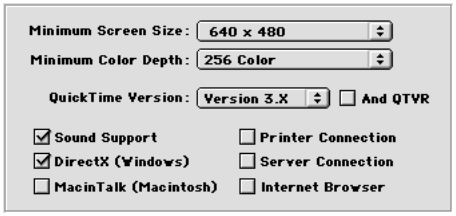


Figure 1.3: Settings in the Setup Helper to set minimum system and hardware requirements for playback of your application.

These attributes and technologies are chosen in the Project Setup Helper.

If, during development, you change your mind about these attributes, they can be modified through the Project Attributes dialogs, which is accessed from the **File** menu. For more information on the Project Attributes dialogs, see chapter 2, Customizing the Development Environment.

Chapter 2

Starting and Finishing your onViz Project

Covered in this Chapter:

- Launching onViz for authoring
- Using onViz Project Setup
- Setting Project Attributes
- Customizing your Development Environment
- Saving Your Work
- Delivering your application

Launching onViz for Authoring

Launch onViz for authoring by double-clicking the onViz program icon. The first time onViz is launched, the onViz Project Setup dialog appears, which guides you through the initial process of creating an onViz document. The onViz authoring environment can also be opened by dragging and dropping an onViz document onto the onViz program icon. Launching onViz this way bypasses the Project Setup dialogs while opening the chosen document.

Using onViz Project Setup

Upon launching onViz, you'll be presented with the Project Setup dialog. The onViz Project Setup allows you to setup various properties of your interactive application before you get started on it.

If you are working primarily on existing onViz files, you may want to disable the Project Setup at startup. To prevent the Project Setup from displaying when onViz launches, select the **Do not Show onViz Project Setup on Startup** check box.

You can always open an existing onViz file by choosing **Open** from the **File** menu. The Project Setup dialog can be accessed by choosing **New** from the **File** menu.

To make onViz once again display the Project Setup at startup, choose **Authoring Preferences** from the **File** menu, and select the **Display the onViz Project Setup Screen on Startup** check box.

Tip

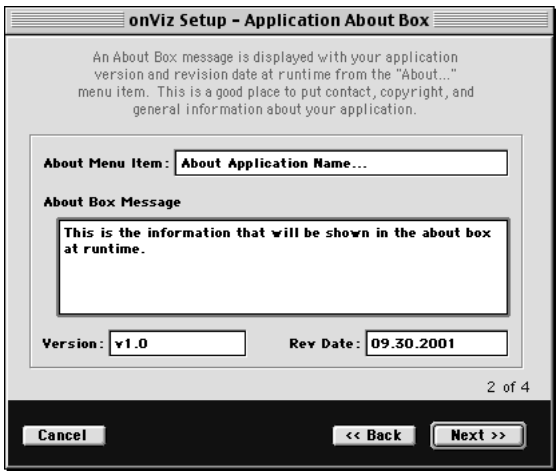
You can bypass most of the Project Setup dialogs by clicking the **Quick Setup** button. If you decide to use Quick Setup, you can access the same application properties found in the Setup Helper in the **Project Attributes** dialog, found in the **File** menu, and the **Target Build** dialog, found in the **Edit** menu.

Creating a New onViz Top-level Application

1. When the onViz Project Setup dialog appears, select the **New Project** radio button. Click **OK** to proceed to the next screen.
2. In the Application Information dialog, enter a name for your new application; the name you enter will be the file name for this top-level application. Enter your name in the Author's Name field. By associating your name with your application, you let your users know who to contact with questions, problems, or suggestions.

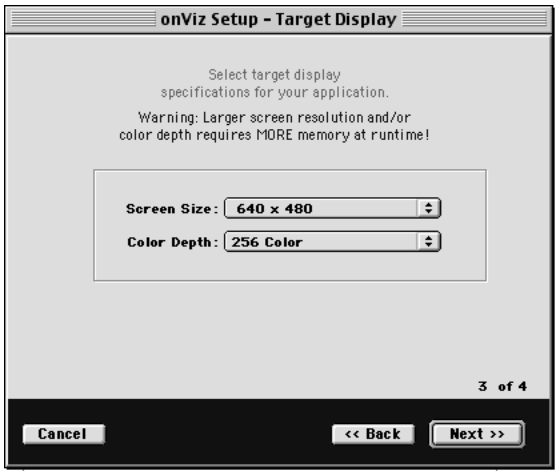
Click **Next** to proceed to the next dialog.

3. The Application About Box dialog contains a field that allows you to enter information about yourself and your application. Enter into the **About Box Message** field any information you want your users to see when choosing **About…** from the Apple menu.

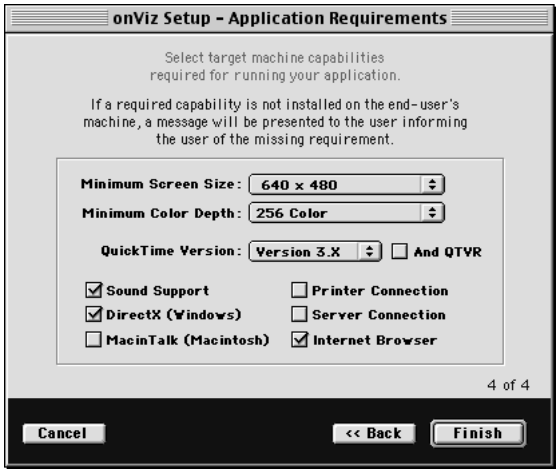


Use the **Next** and **Back** buttons to navigate between the Setup Helper's screens.

4. The Target Machine dialog gives you the opportunity to tailor your application specifically for the computers and monitors on which it will be viewed. The settings chosen here affect several attributes of the development environment. For example, choosing **640 x 480** from the **Screen Size** dropdown menu displays a 640 x 480 Target Screen Size boundary in your Output Presentation windows and in the Image Editor so you can position elements accordingly. Choosing **256 Color** from the **Color Depth** dropdown menu causes onViz to display a warning whenever a 16 or 32-bit bitmap image is created or imported into onViz.

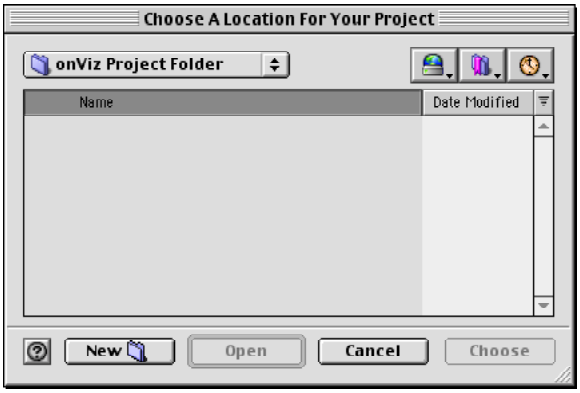


5. In the Application Requirements dialog, you can designate the features you plan to include in your application. For instance, if you plan on making your application connect to the Internet, place a check mark in the Internet Browser check box. Then, when your audience launches your application, it checks their computer for the presence of a Web browser, and notifies them if it cannot be found.



6. Click **Finish** to complete the setup process and to save the application to your hard drive, after which you enter the onViz development environment.

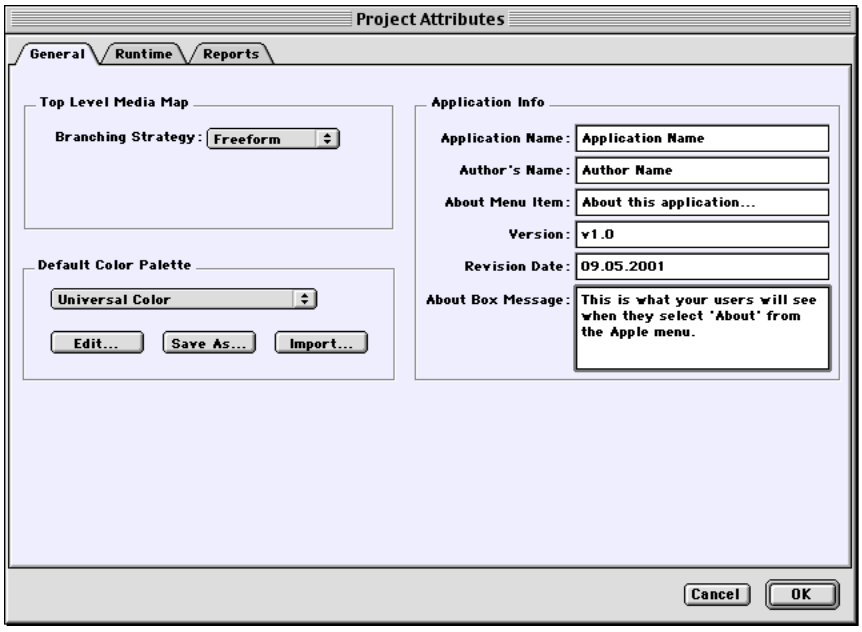
In the Save New Project File dialog, click the **New Folder** button to create a new folder that will contain your project and its support folders.



Customizing the Development Environment

Setting up Project Attributes

Many of the options in the Project Attributes dialog are ones that you can setup in the Project Setup Helper.



Setting up Authoring Preferences

Planning your Application

After completing the Project Setup dialogs and entering the development environment, you will have already made several decisions about the way your application will appear and operate during runtime. Before you begin creating your application, however, you should make just a few more decisions about how your application will function:

- Will it be single or multiple-file?
- Will it be for the Macintosh OS, Windows, or both?
- How will it be delivered?

During development, onViz saves all of the application's media assets in the project support folders. By doing so, all of these source files can be easily accessed for changes and updates. For instance, if a graphic containing your company's phone number is used in several places throughout your application, and the phone number changes, you need only update the original graphic from the support folders, and the graphic is updated wherever it appears in your application.

When development has finished, you save out, or ***build***, all of your source files into a stand-alone application, or ***target***. You can define several targets, each a variation of the same set of source files. Target variations might include a single-file Macintosh application, a single-file Windows application, a multiple-file Macintosh application, or a multiple-file Windows application for network delivery.

Building target applications

Targets are defined using the Build Target dialog, which is accessed from the **File** menu. The Build Target dialog lets you choose your application's target platform, whether it will be single or multiple-file, whether or not it will be delivered over a network, and whether or not your end-users will have the ability to customize certain settings, such as report and bookmark file paths and email information. Single-file targets are stand-alone files that are comprised of the application and all of its media assets. With multiple-file targets, the application is a separate file that must have access to its media assets, which are contained within its support folders.

After your targets are defined, the **Build** command is used to save out the selected targets as final, stand-alone applications. In this manner, the same source files can be used to create several completely different targets. If the targets need to be revised, the changes only need to be made once on the source files, and then the Build command can be issued again.

Even though you can define your targets at any time during the development of your application, you should always keep in mind an idea of how you want to deliver your final product. Forming this concept of your final product will help you make decisions regarding how to save your media assets during development. **For instance**, when you import a sound, movie, or image file, the Library dialogs give you the option to reference the file internally or externally; for this option, you might default to factors such as file size or how often the file is used. However, if you plan in advance to have several multiple-file targets, you can use this

option to your advantage when creating your targets.

When building a multiple-file application, the Build Target dialog gives you the option to export files above a designated file size. This option overrides your library references, and allows you to create a variety of multiple-file targets, each with a different number of files integrated into the application and a different number of files in the support folders. You might build one target, with a small application file size and most of its media assets in its support folders, for distribution on CD-ROM, and another target, with a larger application file size and fewer of its media assets in its support folders, for distribution over a network.

Single-file Applications

The single-file Build Target option works well for relatively small applications, especially if you require that the application be distributed on a floppy disk or some other removable media. Single-file applications have the advantage of having all of their media assets integrated inside of them, which is more secure than multiple-file applications. With a multiple-file application, a user can potentially access its media assets, such as images; if an image contains the answers to a question, the application could be compromised. With its media assets integrated inside it, the single-file application cannot be opened in this way.

To build a single-file application, simply choose **Build/Edit** from the **Target** sub-menu, located under the **File** menu.

Click the **Build** button to compile your source files, including all of the media assets, into a single-file. To make a custom target, select one of the default targets and click the **New** button to make a copy. Rename the new target, then make any necessary changes.

Multiple-file Applications

Considerations for Cross-Platform Applications

Considerations for Web-based Delivery

Chapter 3

The Application Map

Covered in this Chapter:

- The Purpose of the Application Map
- Elements of the Application Map
- onViz States
- onViz Paths
- Map Tools Palette
- Application Map Menus
- Testing your Application

The Purpose of the Application Map

The Application Map allows you to create a visual representation of your ideas. When you plan a project, you create in your mind a rough idea of how its elements should flow. onViz lets you get those ideas out of your head and recreated visually on the Application Map, where you can move the elements around, chart the flow between them, and decide what works and what doesn't.

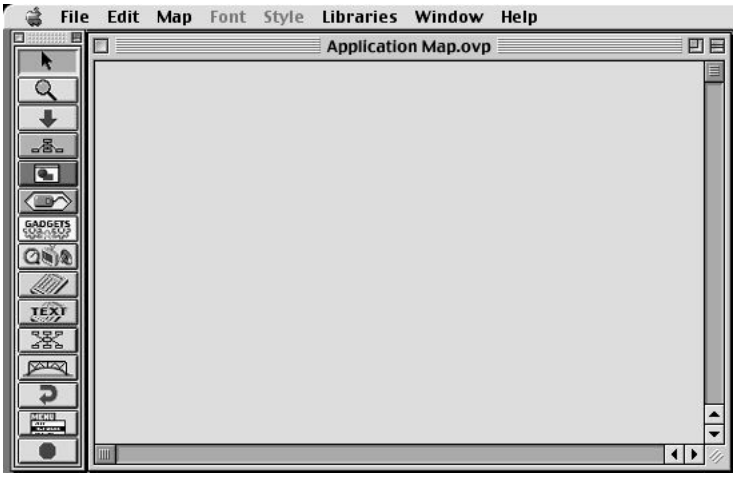


Figure 3.1: onViz' Application Map.

If you're planning a simple course, you might present the subject matter first and then ask some questions, providing positive feedback each time they get one right ("Way to go, genius!") and some gentle coaching when they get one wrong ("Umm, perhaps you should re-read the material."). You probably haven't written your subject matter yet, or decided what questions to ask, but you may already have a great idea for a layout that will make your project informative and interesting at the same time. Once you get your project laid out, you can then go back in and add the content.

With the Application Map, you can choose your project's elements, onViz calls them States, from the Map Tools palette and drag them to a spot on the Application Map. The Map Tools palette has a State for every element you could think of to include in your project: text, graphics, animations, sounds, movies, mouse inputs, and many more. Leave something out? Don't like the order your States are in? Simply drag the States around, or add a new one where you want it. Once your project's elements, or States, are laid out, the Application Map lets you use Paths to direct the route your audience will take through your project. Figure 3.2 shows a possible Application Map, complete with States and Paths, for your simple course.

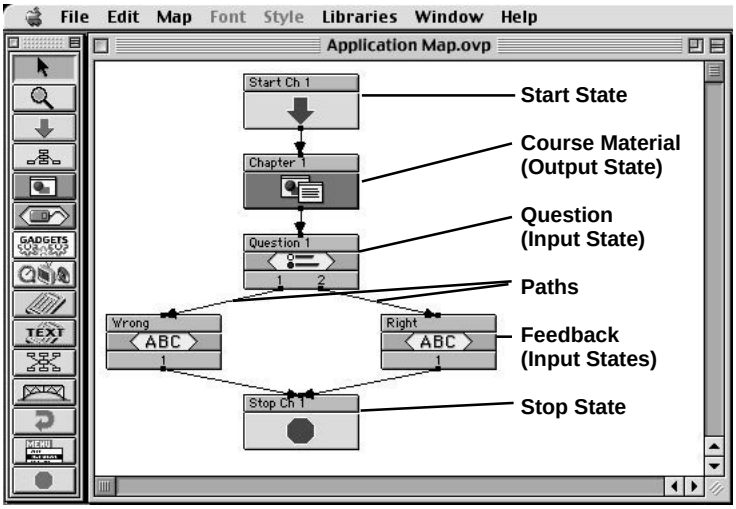


Figure 3.2: An example Application Map, with a Start State, course material, question, right and wrong feedback, a Stop State, and Paths.

Now that you've got your course laid out, the Application Map makes it easy to build upon it. Want to add another set of questions and answers? Simply select the current set, copy it, and paste it back into the work area. Now you have two sets of questions and answers. Repeat the process, this time using the two sets of questions and answers, and now you have four sets! The Application Map supports familiar computing operations such as cut, copy, and paste, which makes it easy to replicate repetitive elements of your project. Think of all the time you can Figure 3.3: Elements of the Application Map, including the Map Tools palette, the work area, the menu bar, States, and Paths.save by building your project in this way.

Elements of the Application Map

The Application Map window is where you create your project. This window contains: the Map Tools Palette, from which you choose the States that comprise your project, the work area, where States and Paths are placed, and the Menu Bar, for selecting functions for the authoring environment. A more detailed explanation of each element follows this introduction.

Work Area

The work area is where you lay out your project's structure. Simply drag and drop the desired States from the Map Tools palette onto the work area, which can hold 64 States vertically and horizontally.

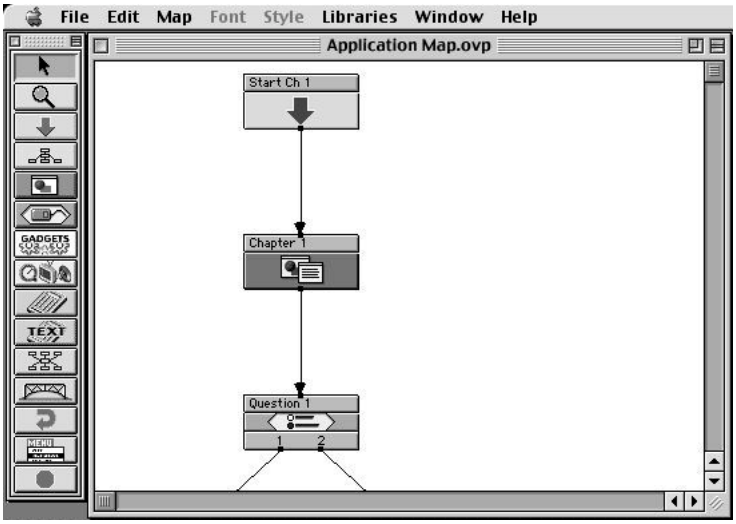


Figure 3.3: Elements of the Application Map, including the Map Tools palette, the work area, the menu bar, States, and Paths.

States

States, accessed from the Map Tools palette, are the basic building blocks of onViz, containing your project's content and performing its activities. Represented by rectangular icons, States have an Entrance Point on their upper edge where Paths enter, and one or more Exit Points on their lower edge where Paths leave. The top third of the State icon is the InfoCenter. Here you can enter the State's Name, and select options to determine the specific functions or runtime behavior of the State. The lower two-thirds of the State icon is the Presentation, which provides access to tools for adding content and modifying the display viewed by your audience.

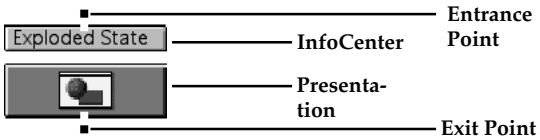


Figure 3.4: An exploded view of a State, showing its Entrance point, InfoCenter, Presentation, and Exit point.

Paths

Paths, represented by arrows, determine the sequence in which each State's activities are performed (application flow). You create a Path by clicking on a State's Exit Point and holding the mouse button down while dragging towards the desired State. Release the mouse button anywhere over the State's icon and the Path automatically connects to its Entrance.

There are two types of Paths, Unconditional and Conditional. Unconditional Paths simply allow the flow of your project from one State to the next. Conditional Paths allow you to use formulas and variables in determining the flow of your project between States.

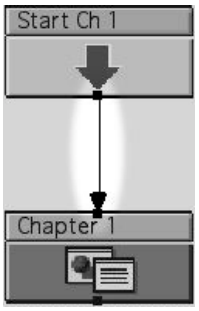


Figure 3.5: A Path connecting two States.

Map Tools Palette

The Map Tools palette contains the States you use to create your project. Simply drag and drop the desired States from the Map Tools palette onto the work area.

Figure 3.6: The Map Tools palette.

States

States are selected from the Map Tools palette, and contain the content and perform the activities that make up your interactive project. There are thirteen types of States:

- Start
- Output
- Gadget
- Calculator
- Junction
- Return
- Stop
- Group
- Input
- Multimedia
- Text Calculator
- Bridge
- Menu

For more information on each State type, refer to the Map Tools section below, and to each State type's chapter later in the manual.

Each type of tate is represented by a rectangular icon containing a unique symbol (Figure 3.7). When placed on the Application Map, this symbol offers a visual cue as to how the State functions in the project. As you gain experience with onViz, you'll be able to determine how a project flows from simply looking at how these icons are arranged and connected on the Application Map.

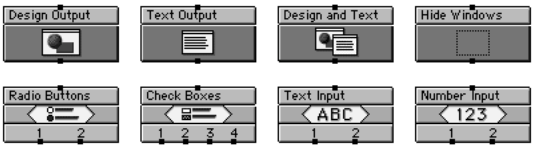


Figure 3.7: A State's icon helps demonstrate its function. Top row (Outputs): Design, Text, Design and Text, and Hide Windows. Bottom Row (Inputs): Radio Buttons, Check Boxes, Enter Text Input, and Enter Numbers Input.

To add a State to the Application Map, simply drag and drop the desired State from the Map Tools palette onto the work area. Alternately, you can click the desired State in the Map Tools palette, turning the pointer into a star as it passes over the work area, and then click in the work area where you want it to be positioned (Figure 3.8).

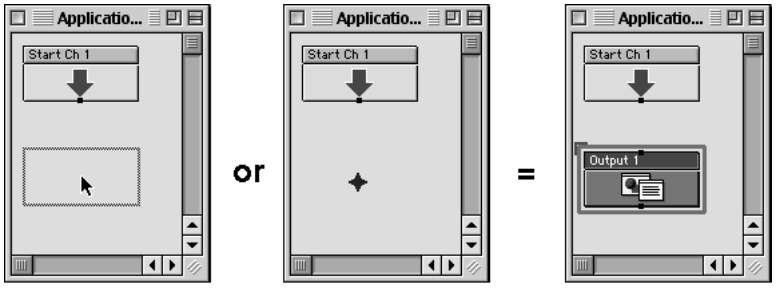


Figure 3.8: States can be dragged from the Map Tools palette into the work area (left), or, a State can be added to the work area by clicking its icon in the Map Tools palette (the cursor will change into a star as it passes over the work area), and clicking in the work area where you want it to go.

States can be selected with a single mouse click; once selected, they are highlighted with a blue outline (Figure 3.9). Additional States may be selected by holding down the shift key while clicking on the State icon. Multiple States can be selected by using the Pointer tool to drag a selection marquee around the desired States. All States in the current Application Map can be selected by choosing **Select All States** from the **Edit** menu.

Once selected, States can be repositioned by dragging them to a new location on the Application Map. Any connected Paths maintain their connections with the repositioned States. Operations such as Cut, Copy, nd Paste can also be performed on selected States.

Except where noted, all States contain the following components:

Figure 3.9: An unselected State (left) and a selected State (right).

Entrance

A State's Entrance is the destination point for incoming Paths, and is where the project's flow of activity enters a State.

Exit

A State's Exit is the point of departure for outgoing Paths. The project's flow of activity leaves a State's Exit and continues to the Entrance of the next State, as determined by either their Path connections or branching strategy.

InfoCenter

A State's InfoCenter is typically used to set window attributes and runtime options for that particular type of State. Because every type serves a different function and has different capabilities, they all have different InfoCenters. For more information on a specific type of InfoCenter, refer to that State's chapter. A State's InfoCenter is accessed by double-clicking the rectangular area in the upper 1/3 of its icon.

Tip

When using the Pointer to select States with the selection marquee, the entire State icon must be enclosed within the marquee. This level of precision is a feature that prevents the unintended selection of States and Paths on complex Application Maps. To select each State icon the marquee touches, hold down the Option key while dragging the marquee.

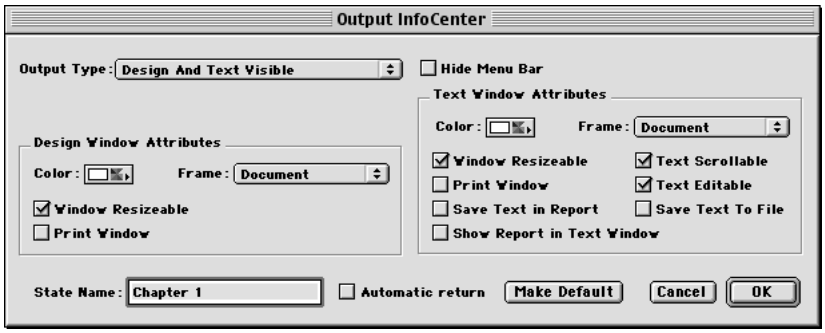


Figure 3.10: An Output State InfoCenter.

While a State's name can be changed from within its InfoCenter window, it can also be changed without leaving the Application Map. To change a State's name from the Application Map window, click once on its name, wait for the Name field to become a black rectangle with a red outline, then type the new name. Pressing the Return or Enter keys, or clicking anywhere outside the Name field, makes the new name take effect (Figure 3.11).

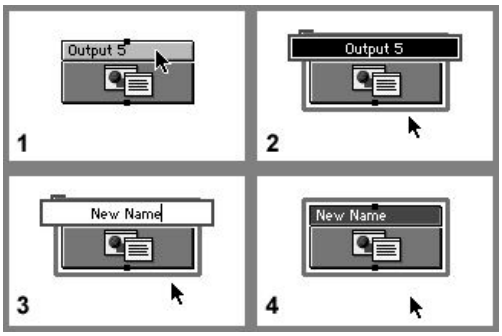


Figure 3.11: Changing a State's name in the Application Map.
1) Click once in the name field.
2) Wait for the name field to become highlighted (black rectangle with a red outline).
3) Type the new name.
4) Click anywhere outside the name field.

Presentation Window

A State's Presentation window is used to create and modify the content displayed by the State, or functions that are carried out when the application flows through it. The Presentation window is accessed by double-clicking the rectangular area in the lower 2/3 of the State's icon. Only three States include a Presentation window: Group, Output, and Input.

The Group Presentation window is used to design and organize the flow of activity for the States within each Group.

The Output Presentation window is used to design the text, graphics, and animation included in your project (Figure 3.12). Typically, this window is also where you set up the playing of other media components, such as sound clips and QuickTime movies.

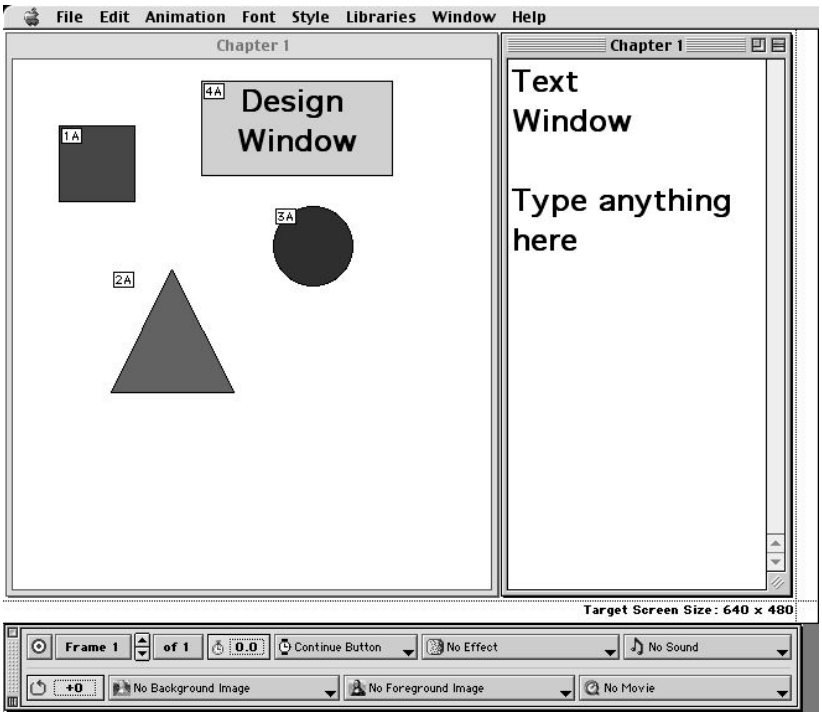


Figure 3.12: An Output Presentation Window. Note the design window on the left, the text window on the right, and the Frame Control palette at the bottom.

The Input Presentation window is used for setting the size and position of the question dialog on the screen. When you open this window for the first time, you get a dialog box prompting you to select an Output graphic to aid in the positioning of the question dialog.

Tip

A State's Presentation window can also be accessed by selecting it and then choosing **Presentation** (⌘E) from the **Map** menu.

Paths

Paths, represented by arrows, determine the sequence in which the project's activities are performed. There are two types of Paths, Unconditional and Conditional. Unconditional Paths simply direct the flow oyour project from one State to the next. Conditional Paths allow you to set up situations to determine the flow of your project between States. These situations may be comparing variables, checking the current score, etc. Typically, when using Conditional Paths, you have two or more Paths leaving the same Exit: one Unconditional and one or more Conditional. Any States following a Conditional Path are not carried out unless the Path's conditions are satisfied.

Paths are created by clicking and dragging from one State's Exit P Et to the EntrancePoint of another State, and are only functional if they are properly connected at both ends (beginning at an Exit and ending at an Entrance). Paths can be used to connect any two States within the same Group. Improperly connected Paths are shown as blue arrows. If the project's flow of activity tries to follow one of these improperly connected Paths, onViz displays a "Dead End Route Reached" error message.

Tip

You can use a Bridge to connect two States not in the same Group, or to connect to a different onViz document.

A Path can be selected by clicking on its line with a single mouse click; once selected, the end points of the Path are designated by drag handles (Figure 3.13). Additional Paths may be selected by holding down the shift key while clicking on the Paths' line.

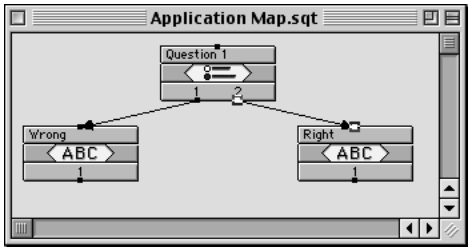


Figure 3.13: An unselected Path (left) and a selected Path (right). Note the handles at each end of the selected Path.

Multiple Paths can be selected by using the Pointer tool to drag a selection marquee around the desired Paths. All Paths in the current Application Map can be selected by choosing **Select All Paths** from the **Edit** menu.

Tip

When using the Pointer to select Paths with a selection marquee, the entire Path must be enclosed within the marquee. This level of precision is required to prevent the unintended selection of Paths and States on complex Application Maps. To select all Path elements the marquee touches, hold down the Option key while dragging the marquee.

It is possible for several Paths to leave a State by the same Exit Point. If all of these Paths are Unconditional, onViz selects one at random for its Exit. This type of routing is fine if you truly want the project to flow in a random fashion. However, if you want to direct the project's flow of activity based on a series of events or variables, use Conditional Paths. For more control, you might use a single Unconditional Path and several Conditional Paths leaving from the same Exit Point. If connected in this manner, onViz begins evaluating the Conditional Paths and takes the first one whose conditions have been satisfied. If no conditions are satisfied, the application flow follows the Unconditional Path.

When initially created, all Paths are Unconditional, and are shown as a solid black arrows. To create a Conditional Path, double-click any point of a Path to bring up its Conditional Path Info window (Figure 3.14), fill in a Condition line, and click **OK** to close the window and return to the Application Map.

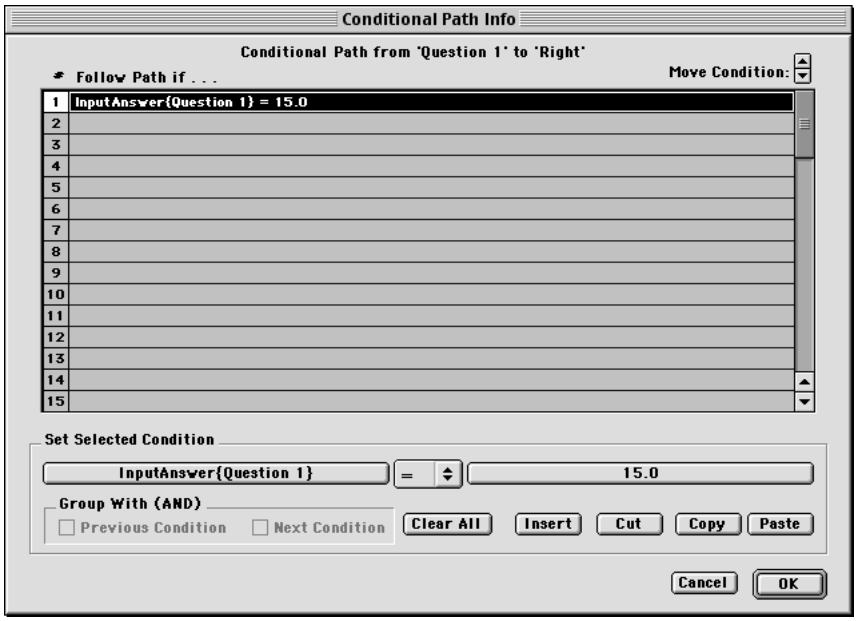


Figure 3.14: The Conditional Path Info window.

The new Conditional Path is shown as a solid red arrow, a visual cue that the Path contains criteria that must be met.

For more information, see chapter 15, Paths.

Tip

A Path's Info window can also be accessed by selecting it and then choosing **Path Info-Center** (⌘I) from the **Map** menu.

Map Tools Palette

The Map Tools palette contains States used for the creation of your interactive project. Simply drag and drop the desired States from the Map Tools palette onto the work area.

Several State categories (Group, Output, Input, and Gadget) contain variations that can be accessed via contextual menus. You access these contextual menus by holding down the Control key while clicking on the State icon. Choose the desired type of State from the menu; your mouse pointer will turn into a star. Then click in the work area where you want it to be positioned.

Tip

By default, States are automatically named when they are placed on the Application Map. This Auto-Naming feature can be turned off in the Authoring Preferences menu. Regardless of whether or not this feature is turned on, best practice is to give each State a unique, meaningful name, as Variables and Bridge Targets are based on the named State.

The Map Tools palette includes:

Pointer – allows Application Map elements to be placed, selected, and moved.

Zoom – allows you to shrink or magnify the work area in order to see more or less of the Application Map. Press the Zoom Tool button, then click in the work area to zoom out. Option–click in the work area to zoom back in and return to Normal Map View for editing.

Zooming out of the Application Map is helpful for viewing your entire map to track its flow. Zooming out is also useful for printing a large Application Map, as onViz only prints the area of the map visible on the screen (Figure 3.15).

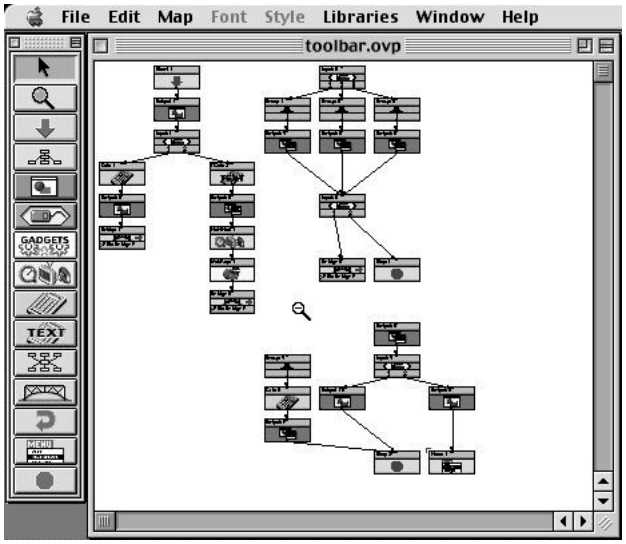


Figure 3.15: Zooming out of a very large Application Map.

Tip

You must be in **Normal Map View** in order to create new States or to access an InfoCenter or Presentation window. You can not access these windows if you are zoomed in or out.

Zoom functions are also available from the **Map** menu:

Zoom In – (⌘])

Zoom Out – (⌘ [)

Start – Begins application or group flow in all Freeform Application Maps. The Start State is one of five in the larger category of States known as Flow States. Flow States direct the sequence of execution from one State to another, and can also direct the sequence of execution to another onViz document or to another program.

For more information on Flow States, see chapter 5.

Group – Provides a means to organize the States used in your Project. Also allows for automating the way your project flows. By holding down the control key while choosing from the Map Tools palette, you can access a contextual menu containing the five variations of Groups:

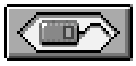
- Freeform
- Linear
- One Per Row
- Random Pool
- List Access

For more information on Groups, see chapter 12.

Output – Presents information in the form of multimedia elements: graphics, text, animation, sound, and movies. By holding down the control key while choosing from the Map Tools palette, you can access a contextual menu containing the four variations of Outputs:

- Design Window
- Design and Text
- Text Window
- Hide Windows

For more information on Outputs, see chapter 6.



Input – Provides templates for seeking input from your user. These templates include requesting input in the form of text or number answers, presenting a selection from multiple choice responses, or clicking on-screen buttons. Inputs lets you score your users' responses, as well as letting you track and capture their scores, number of attempts, and response time for use as variables.

By holding down the control key while choosing from the Map Tools palette, you can access a contextual menu containing the five variations of Inputs:

- Mouse Bays
- Enter Text
- Enter Numbers
- Radio Buttons
- Checkboxes

For more information on Inputs, see chapter 9.

Gadget – Contains a variety of functions to enhance your application. By holding down the control key while choosing from the Map Tools palette, you can access a contextual menu containing the nine variations of Gadgets:

- Bookmark
- Cursor Display
- Print Options
- Report Options
- Restart Application
- Send Email
- Show Web Page
- Monitor and Sound
- Timer

For more information on Gadgets, see chapter 11.

Multimedia – Used for playback of multimedia components, including:

- Play Sound
- Play Movie
- Overlay Image

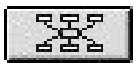
For more information on Multimedia States, see chapter 10.

Calculator – Assigns values to numeric variables.

For more information on Calculators, see chapter 14. For more information on Variables, see chapter 13.

Text Calculator – Assigns values to text variables.

For more information on Calculators, see chapter 14. For more information on Variables, see chapter 13.



Junction – Brings multiple paths together so that branching decisions can be made.

Bridge – Moves or jumps the project's flow of activity to a different location in your project, or launches another program. Bridges are one of five in the larger category of States known as Flow States. Flow States direct the sequence of execution from one State to another, and can also direct the sequence of execution to another onViz document or to another program.

For more information on Flow States, see chapter 5.

Return – Used in combination with a Bridge and Return to return the project's flow of activity to the exit of the last Bridge. Returns are one of five in the larger category of States known as Flow States. Flow States direct the sequence of execution from one State to another, and can also direct the sequence of execution to another onViz document or to another program.

For more information on Flow States, see chapter 5.

Menu – Creates a custom menu that appears in your project's menu bar. Menus are one of five in the larger category of States known as Flow States. Flow States direct the application flow from one State to another, and can also direct it to another onViz document or to another program.

For more information on Flow States, see chapter 5.

Stop State – Ends the application when reached in the top level Application Map, and exits a group when used within a Group. The Stop State is one of five in the larger category of States known as Flow States. Flow States direct the application flow from one State to another, and can also direct it to another onViz document or to another program.

For more information on Flow States, see chapter 5.

Menu Bar

In addition to standard Macintosh functions, the Application Map Menu Bar provides support for special Application Map activities. Menus include File, Edit, Map, Libraries, and Window.

File Menu

Contains all of the functions associated with file access and maintenance.

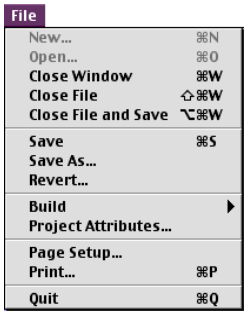


Figure 3.16: The File menu.

New (⌘N) – Brings up the Setup Helper to create a new onViz document or project.

Open (⌘O) – Opens a previously created onViz file for additional editing.

Close Window (⌘W) – While within the top-level work area window, **Close Window** closes the current onViz file.

Close File (⇧⌘W) – Closes the current onViz file.

Close File and Save (⌥⌘W) – Closes the current onViz file.

Save (⌘S) – Saves the current onViz file under its present name, overwriting the previous version.

Save As – Saves the current onViz file, with the options of renaming it, saving it in a different location, and saving it as a onViz application or document.

Revert – Reverts the current onViz file to its previously saved version.

Build – Contains submenu options for building your onViz target application. Options include:

- **All…**– Builds all defined target applications
- **Editor…**– Displays the Build Editor to define a target application
- **Build for Technical Support…**– Automatically copies the project and all assigned library elements into a folder titled **Tech Support Build**. Since no unused elements are copied to the support folder, choosing this menu option can also be used to optimize the project.
- A list of all 'Target Builds' that have been defined - this lists the names of all the build configurations you have created for this project.

Project Attributes – Provides options for setting up runtime attributes for your application including top level application flow, display and report options.

Page Setup – Provides options for printing.

Print (⌘P) – Prints the currently visible level of the Application Map.

Quit (⌘Q) – Exits onViz, with the option of saving changes to your project.

Tip

While within a Group or Output Presentation window, **Close Window** closes the current window and returns you to the Application Map window.
While within the Sprite Library or Graphics Editor window, **Close Window** closes the Presentation window and returns you to the Output Presentation window.

Edit Menu

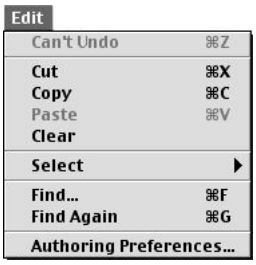


Figure 3.17: The Edit menu.

Contains all of the functions related to the selection and manipulation of States and Paths.

Cut (⌘X) – Removes the selected information from the Application Map and places it on the Clipboard.

Copy (⌘C) – Copies the selected information from the Application Map and places it on the Clipboard.

Paste (⌘V) – Places information from the Clipboard onto the Application Map.

Clear – Removes the selected information from the Application Map without placing it on the Clipboard.

Select – Contains submenu options for selecting all the Application Map elements of a specific type. Includes:

- All Paths and States (⌘A)
- All States
- All Paths
- All Unconditional Paths
- All Conditional Paths

Find (⌘F) – Currently not implemented

Find Again (⌘G) – Currently not implemented

Authoring Preferences – Currently incomplete

Map Menu

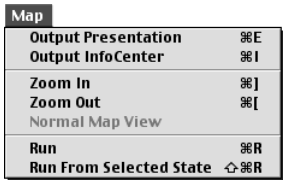


Figure 3.18: The Map menu.

Contains functions that allow you to navigate, modify, and test your project.

Presentation (⌘E) – Brings up the Presentation window of the currently selected State.

InfoCenter (⌘I) – Brings up the InfoCenter window of the currently selected State or Path.

Zoom In (⌘]) – Magnifies the work area in order to see less of the work area**?**

Zoom Out (⌘[) – Shrinks the work area in order to see more of the work area**?**

Normal Map View – Resets the work area to its default magnification.

Run (⌘R) – Begins runtime execution from the first State in the top level of the Application Map.

Run From Selected State (⇧⌘R) – Begins runtime execution of the currently selected State.

Libraries Menu

Figure 3.19: The Libraries menu.

Allows access to your project's libraries, so that they can be easily shared and edited.

For more information on libraries, see chapter 4.

Bridge Target Library (⌘1) – Displays a library of available Bridge Targets, with the ability to create new Bridge Targets and edit existing ones.

Cursor Library (⌘2) – Displays a library of available cursors, with the ability to create new cursors and edit existing ones.

Custom Variables Library (⌘3) – Displays a library of available Variables, with the ability to create new Variables, edit existing Variables, or to import Variables from a separate onViz file.

Font Library (⌘4) – Displays a library of available fonts, with the ability to add new Fonts and edit the attributes of existing fonts.

Image Library (⌘5) – Displays a library of available images, with the ability to create new images, edit existing images, convert existing images to different colors, resolutions, and/or formats, or import images from a file.

Movie Library (⌘6) – Displays a library of available movies, with the ability to add new movies, edit the attributes of existing movies, or to play them.

Sound Library (⌘7) – Displays a library of available sounds, with the ability to create new sounds, edit t he attributes of existing sounds, convert existing sounds to other playback rates and/or formats, or to record new sounds.

Window Menu

Allows you to either display or hide all of the palettes associated with a specific window. Not all palettes are available at all times.

Map Tools Palette (⌘M) – Toggles (Hides/Displays) the Map Tools palette, the only palette available within the Application Map window.

Testing your Application

The Application Map also allows you to test your project without leaving the authoring environment. If you choose **Run** (⌘R) from the **Map** menu, onViz starts your project from the first Start State. If you select a State and then choose **Run From Selected State** (⌘⇧R) from the **Map** menu, onViz starts your project from whichever State is selected. **Run From Selected State** is particularly helpful in lengthy projects, where you may want to test a single State, such as a Design Output containing animation, without running the entire project.

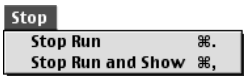


*Figure 3.20: When testing your application, choose **Stop Run** (⌘ .) from the **Stop** menu to return to the Application Map. Choose **Stop Run and Show** (⌘ ,) from the **Stop** menu to return to the Application Map with the State being carried out at the time of interruption selected.*

Application execution stops when:

- a Stop is encountered in the top-level Application Map, signifying that the end of the project has been reached.
- you choose **Stop** (⌘ .) from the **Stop** menu (Figure 3.20).
- you choose **Stop Run and Show** (⌘ ,) from the **Stop** menu (you are returned to the Application Map with the State being carried out at the time of interruption selected).
- a "Dead end route reached" dialog is displayed, indicating that onViz has encountered a State with no Path or an unconnected Path.

onViz makes it easy to test your interactive project and fix any discrepancies you have found, as much of the testing takes place within the Application Map, where any errors can be detected visually. Ultimately, this approach allows you to develop your projects more quickly.

Chapter 4

onViz Libraries

Covered in this Chapter:

- Libraries Overview
- Library Considerations
- The Bridge Target Library
- The Cursor Library
- The Custom Variables Library
- The Font Library
- The Image Library
- The Movie Library
- The Sound Library

Libraries Overview

onViz' libraries provide you with the flexibility needed to structure your project in a way that best suits your needs, for development and for delivery. onViz' libraries let you set up the way media assets play within your application, create custom cursors, define custom variables, designate display fonts, and set up bridge actions. If your project contains several documents that all use the same sounds, digital movies, and images, its overall file size could be quite large if every document contained its own copy of the sound, digital movie and image files. Libraries let all of your project's documents draw their media assets from a common source, thereby reducing file size.

And, as development time is often a consideration, Libraries save you time. If you've created a background and then decide to change it's colors or appearance, changing the image once, updates the image every place it is used in the application. If you create custom variables or cursors for your top-level application, you don't have to duplicate your efforts to access these in your supporting documents. onViz lets you store them in libraries where they are available to all of your project's supporting documents.

Simply open one of the libraries from the **Libraries** menu and you have instant access to all of the media assets in that particular category (Figure 4.1). onViz has seven libraries:

- Bridge Target Library
- Cursor Library
- Custom Variables Library
- Font Library
- Movie Library
- Sound Library

Figure 4.1: The Libraries menu.

This chapter deals specifically with making entries in the libraries. See specific chapters for how to use the library components in your application.

Library Considerations

Resident vs. Disk Referenced Files

During development, media assets are referenced by your application as files in designated support folders in your project folder. When you build your target application, you have the option of integrating media assets into your application (internal library), or letting them exist as separate files saved to a disk (external).

Because all files are external during development, it is easy to update your application content. If, for example, you're working on a project containing a company's logo, and the company changes names, you can simply replace the old logo file with the new one and the change is automatically updated throughout your project. Or, want a different button sound? Simply replace the sound file in your Sound Support folder with a new one of the same name and the sound is updated. When you're ready to distribute your application, you have the option to import any or all media files into your application. Any assets not imported must be accessible to your application in a support folder or in a designated server location, if delivering over a network.

If you are delivering your application on CD-ROM, you may want to import all assets to make a single, self-contained file. In this way, any sound, movie, and image files cannot be opened or copied.

Deleting Library Entries

One important point to keep in mind when using libraries: changing an entry in a library changes it for the entire application. Also, if you delete a library entry, it is no longer be available to any of your documents. This is important to remember, especially with custom variables and bridge targets. You may have a conditional path dependent on a variable value – if you delete the variable, your path may no longer operate as expected. If your application is set to follow a bridge target and it is no longer in the Bridge Library, your user will receive a "Cannot find Bridge Target" message at that point in the application.

Tip

Note: deleting a library entry does not delete the file from the project support folder, it just removes the reference from the onViz project. Hold down the option (⌥) key to delete both the library reference and the file from the support folder.

Naming Conventions

onViz' Libraries allow your project to include a top-level application with many supporting documents, each of which shares common media assets. Each supporting document also contains States of its own, with names like "Output 1," "Input 2," "Group 1," etc. With so many onViz files, States, and media assets, it's easy to see how you might get lost if you don't establish a naming convention at the beginning of your development. For instance, when creating a new Bridge Target, its library name defaults to the name of the state to which you are bridging. If you don't rename the states on your application map, as well as your Bridge Target entries, you could wind up with several Bridge Targets named Output 2. The more supporting documents you have, the greater the potential for confusion.

However, with just a little thought up front, you can manage the naming of your States, media assets, and onViz files in such a way as to know exactly what they are from their name

references. Try to use unique, meaningful names that you'll remember later. You may know what's inside "Group 1" when you're creating your top-level application, but, two days later when you're working on your fourth supporting document, you'll be more likely to remember it if you'd named it "Chapter 1 Questions."

Library Dropdown Menus

Most of the libraries (except for the Cursor Library and the Custom Variables Library) offer the option to include their entries in dropdown menus for quick access (Figure 4.2). If, for instance, when creating a Bridge Target, you do not deselect the option to **Show in Bridge Drop-Downs**, your new Bridge Target is conveniently placed in a list at the bottom of every Bridge dropdown menu. These dropdown menus are a handy way to access your library entries without having to actually open the library.

When developing small projects, you may decide to use the **Show in Bridge Drop-Downs** option for all library entries. In larger projects, however, where the list of entries could get very long, you may want to be more selective about which entries are included in the dropdown menus. To keep the menu more manageable, include the entries you'll use often, and access those that you'll use only use once or twice from the library dialog.

The Bridge Target Library

You will often want your project's application flow to jump to specific places within your application, to other onViz documents, and occasionally, to another application altogether. The Bridge Target Library is used to set up pointers that tell your application where you want it to go, and whether you want to return after the bridge.

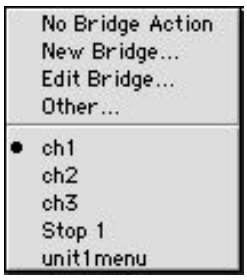


Figure 4.2: An example of the Libraries dropdown menus: the Bridge Target dropdown menu. The selections at the top of the menu allow quick access to Library functions, while the bottom of the menu lets you conveniently access library entries.

Bridge Target Library entries are called from Bridge States, Smart Sprites, Custom Menus, the Timer Gadget, and from various options within the Gadget State. Once set up, the same bridge target entry can be used anywhere you can call a bridge.

While onViz' other library entries are common to all documents, the Bridge Target Library is a bit different, in that each document contains its own Bridge Target Library. The states available as targets are the states within that document.

- For more information on using Bridge States, and using Bridge Targets in Custom Menus, see chapter 5, Flow States.
- For more information on using Bridge Targets with Smart Sprites, see chapter 8, Animation.
- For more information on using Bridge Targets with Timers and Gadget options, see chapter 11, Gadget States.

The Bridge Target Library entries can be accessed in the following ways:

- Choosing **Bridge Target Library** (⌘1) from the **Libraries** menu.
- Opening a Bridge State InfoCenter and choosing from the **Follow:** or **On Error, Follow:** dropdown menus.
- Opening a Menu State InfoCenter and choosing a Bridge option from the **Take Action** dropdown menu.
- Opening a Design Output Presentation window, selecting a **Mouse Action**, and choosing **Bridge To** from the **Go To** dropdown menu.
- Through several of the Gadget States, which have an **On Error, Follow:** dropdown menu that accesses the Bridge Target Library, or through the Timer Gadget State, which allows you to choose a Bridge Target for **Timer Expiration** and **Timer Warning**.

The Bridge Target Library Dialog

The Bridge Target Library Dialog displays a list of all bridge entries that have been set up for the document (Figure 4.3). A check mark in front of the name indicates that the name will be included in the Bridge Target dropdown menu. This checkmark can be toggled off and on by clicking in front of the name.

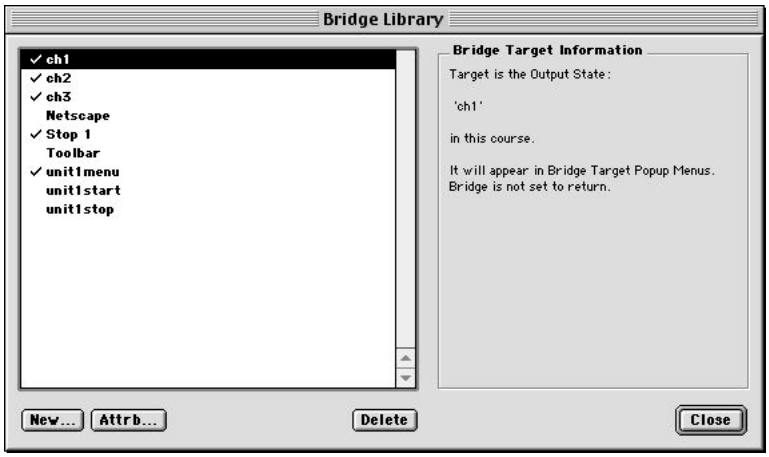


Figure 4.3: The Bridge Target Library. Note the information presented on the right for the selected library entry on the left.

When you click on a library entry, the right-hand side of the dialog provides information about the bridge entry, such as where it will go, and if the Bridge is set to Return.
The Bridge Editor dialog is opened by selecting a library entry and clicking the **Attrb** (Attributes) button, or by double-clicking an entry name.

The Bridge Target Editor Dialog

The Bridge Target Editor dialog varies depending on the option chosen from the **Bridge Type:** dropdown menu, which contains the following Bridge types (Figure 4.4):

- State in This onViz File
- Application
- Application and Document

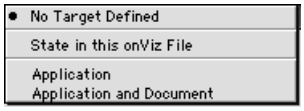


*Figure 4.4: The Bridge Target Editor's **Bridge Type:** dropdown menu.*

State in This onViz File

When the State in This onViz File Bridge Type is chosen, the Bridge Target Editor dialog displays the following selections (Figure 4.5):

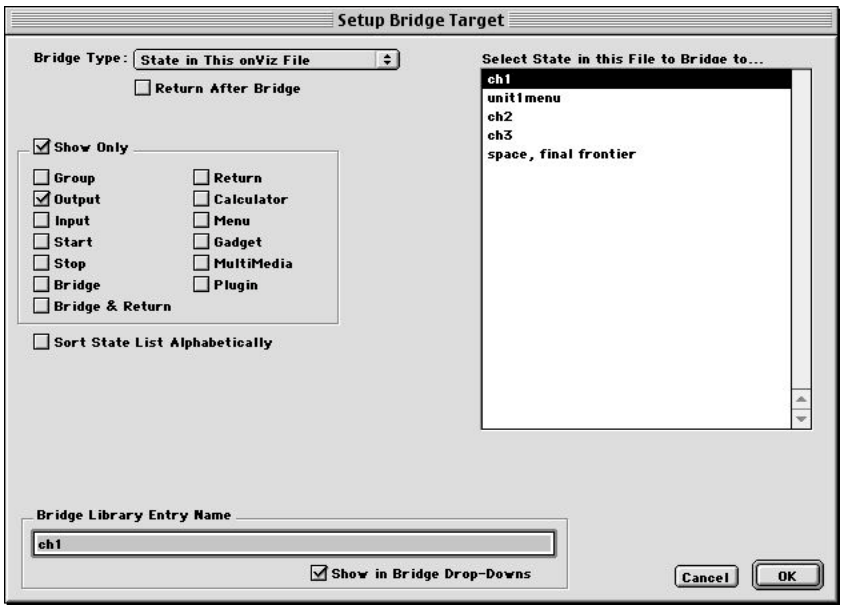


*Figure 4.5: The Bridge Target Editor, with **State in This onViz File** selected from the **Bridge Type:** dropdown menu.*

- **Return After Bridge –** If this option is selected, and the application flows through either a State set with an **Automatic Return** or through a Return State, flow is routed back to the place where the bridge was originally invoked. If you invoked the bridge in a Bridge State, the application will return to the Exit of the Bridge State. If you invoked the bridge from an Output frame, the application will return to the same frame of animation.
- **Select Target State in this File** – Lists the States in the current onViz document, from which you can choose a Bridge Target. By default, this list is sorted according to State type. To sort this list alphabetically, select the **Sort State List Alphabetically** check box.
- **Show Only** – Limits the States displayed in the list to only the types of States checked. This option is helpful if you have a large number of States in your project.
- **Bridge Library Entry Name** – By default, the Bridge Library entry is the same name as the State to which you're bridging; however, you may uncheck this box to give the entry a different name, if desired. The **Show in Bridge Drop-Downs** option allows the Bridge Target's library entry to be accessed from the Bridge dropdown menus.

Application

When the Application Bridge Type is chosen, the Bridge Target Editor dialog displays the following selections (Figure 4.6):

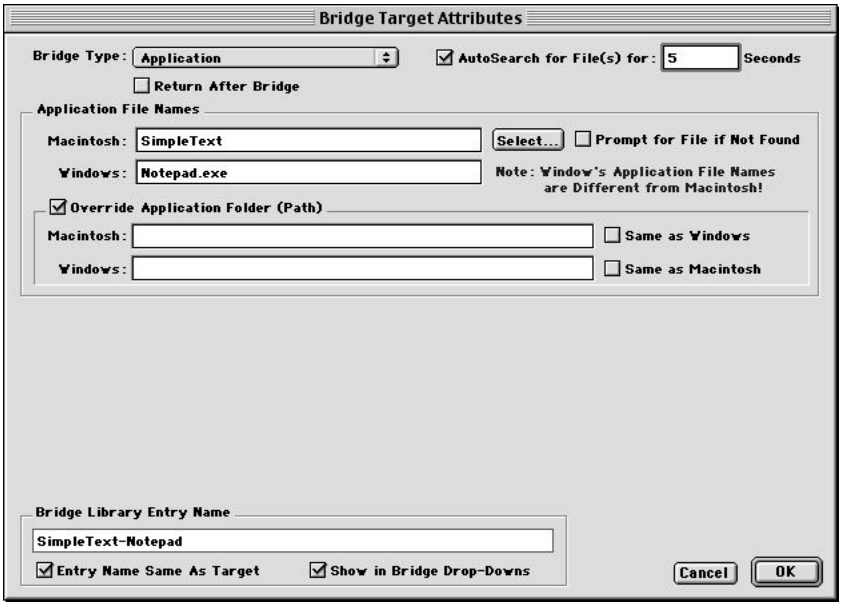


*Figure 4.6: The Bridge Target Editor, with **Application** selected from the **Bridge Type:** dropdown menu.*

- AutoSearch for File(s) for: x Seconds **- Select this option if you want onViz to search for the specified application outside the MiscDocs Support folder. Typically, if the application is on the local computer, a search will be very quick - 3 - 5 seconds. However, if one of the mounted volumes is a server, the search may take considerably. You have the option of designating the amount of time that your onViz application will spend searching for the designated application.**

- **Return After Bridge** – Select this option to keep your onViz application running after opening the target application. If this option is not selected, your onViz application will exit as if it had encountered a Stop State. When your application flows through a Bridge to an Application set to Return After Bridge, both applications will be running simultaneously. If your user's computer does not have enough memory to run both applications simultaneously, a "Not Enough Memory" error message may be displayed and the application being bridged to will not open.
- **Application File Names** – Clicking **Select** brings up a dialog that allows you to navigate to the desired application, whose name is then filled in to this field. If this application is for both Macintosh and Windows delivery, make sure you type in the appropriate name for the Windows application that will be opened.

A copy of the application, or an alias pointing to it, must be in the MiscDocs folder as designated in the Project Attributes Support Folders tab. Make sure you include both the Macintosh AND Windows applications. onViz provides options to override this search path.

- **Prompt for File if Not Found -** Select this option if you want your onViz application to display a 'Get File' dialog so your user can search for the designated application.
- **Override Support Folder (Path)** – If the target application will not be stored in the MiscDocs Support location (for example, stored on a server), you must specify the file path in this field. This location overrides the MiscDocs support location set up when you build your target application. For cross-platform development, you can set different locations for Macintosh and Windows.
- **Bridge Library Entry Name** – You can give your library entry a unique name or let onViz default the name based on the bridge target. If the **Entry Name Same as Target Name** option is selected, the bridge library entry name is the same as the name of the application to which you are bridging. Deselect this option if you want a different name for your bridge entry. The **Show in Bridge Drop-Downs** option allows the Bridge Target's library entry to be accessed from the Bridge dropdown menus.

Application and Document

All features of this selection are identical to the Application option, with this exception: you can also specify a document, in the **Document File Name** field, to open with the application. A copy of the document to be opened, or an alias pointing to it, must also be included in the Misc Docs folder.

Creating a New Bridge Target Entry or Editing an Existing Entry

1. Choose **Bridge Target Library** (⌘1) from the **Libraries** menu.

   The Bridge Target Library window is opened.

2. Click the **New** button. To edit an existing Bridge Target, select a Bridge Target from the list, and click the **Attrb** button or simply double-click the name in the list.

   The Bridge Target Editor dialog opens, with a **Bridge Type:** dropdown list of bridging options.

3. Select an option from the **Bridge Type:** dropdown list, and complete the fields associated with the chosen Bridge Type. See the above section on the Bridge Target Editor dialog for an explanation of these fields.
4. Click **OK** to close the Bridge Target Editor dialog and return to the Bridge Target Library.

   Your new Bridge Target now appears in the Bridge Library list.
5. Click **Close** to close the Bridge Target Library.

Deleting a Bridge Target

1. Select a Bridge Target from the list, and click the **Delete** button.

   A warning message appears, which asks: "Deleting this entry from the Bridge Target Library cannot be undone, delete it anyway?" Hold down the Option key while clicking the **Delete** button to delete the entry without showing this warning dialog.
2. Click **OK** to dismiss the warning message and complete deleting the Bridge Target.

   You are returned to the Bridge Library, which no longer displays the deleted Bridge Target entry.
3. Click **Close** to close the Bridge Target Library.

Tip

Note that deleting a Bridge Target Entry removes the entry and all reference to this entry that might have been set up to direct application flow.

The Cursor Library

You can change the cursor appearance in your application by sending the application through a Gadget State, by using a Mouse Action over a Smart Sprite, or by selecting a Mouse Bays Input type. The Cursor Library contains a collection of default cursors, a series of editable cursors, and empty locations where you can create your own custom cursors. Once created in the Cursor Library, any custom cursors are also available for use within the application's supporting documents.

With the Cursor Library, you can:

- Select default cursors for cross-platform development – default cursors cannot be edited.
- Select or edit a custom cursor.
- Create a new custom cursor.

The Cursor Library entries can be accessed in the following ways:

- Choosing **Cursor Library** (⌘2) from the **Libraries** menu.
- Selecting a Cursor Gadget.
- Opening a Mouse Bays Input type InfoCenter and clicking the **Runtime Feature**'s **Cursor Change** dropdown menu.
- Opening an Output State's Presentation window, selecting one of the **Mouse Action** check boxes in the Sprite Attributes palette, and clicking the **Cursor Change** dropdown menu.

Note that cursors can only be edited in the Cursor Library or when accessed from the Cursor Gadget.

For more information on the Cursor Gadget State, see chapter 11, Gadget States.

For more information on Smart Sprites, see chapter 6, Output States, and chapter 8, Animation Support.

For more information on the Mouse Bays Input Type, see chapter 9, Input States.

The Cursor Library Dialog

The Cursor Library dialog displays the following selections (Figure 4.7):

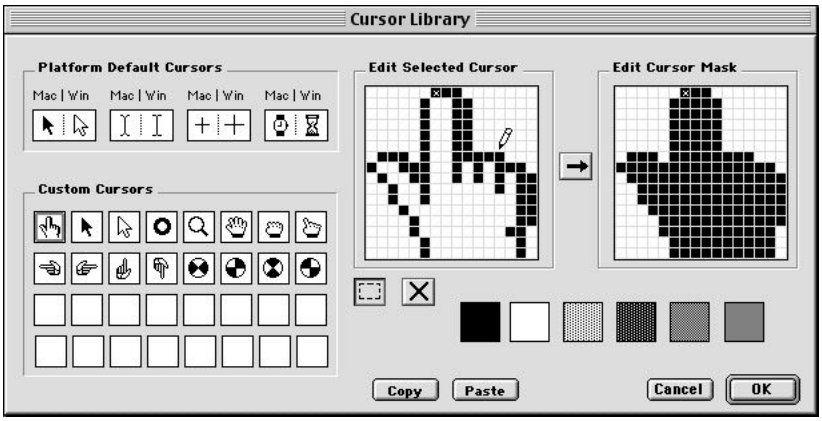


Figure 4.7: The Cursor Library dialog.

- **Platform Default Cursors** – The Platform Default Cursors section contains the four default cursors for Macintosh and Windows. The Macintosh default cursors are available for copying and pasting with the **Copy** and **Paste** buttons.
- **Custom Cursors** – The Custom Cursors section contains sixteen sample cursors and sixteen blank boxes for creating custom cursors. The sample cursors can be edited, or you can copy and paste them into the blank boxes to create new custom cursors while preserving the appearance of the existing ones.
- **Edit Selected Cursor** – The Edit Selected Cursor section contains grids for editing the selected cursor and its cursor mask. The patterns below the Edit fields are for testing your custom cursors.

Creating a New Custom Cursor

1. Open the Cursor Library dialog by choosing **Cursor Library** (⌘2) from the **Libraries** menu.
2. Select one of the empty Custom Cursor thumbnail boxes.

   You can also start with one of the existing Custom Cursors, and edit it to your preference (Figure 4.8):

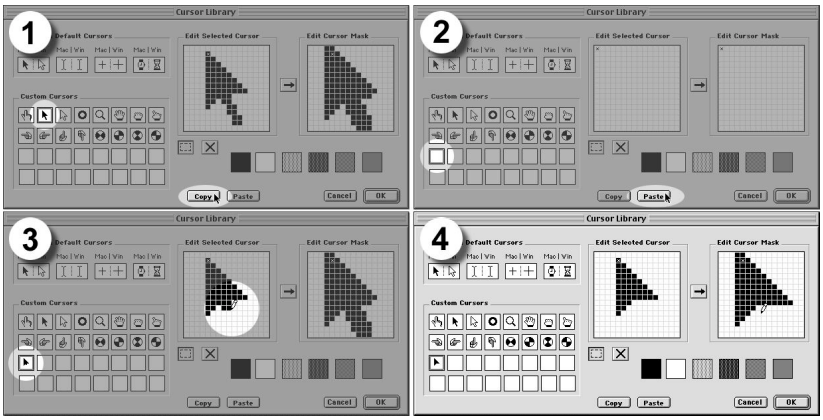


*Figure 4.8: Copying a Custom Cursor for editing. 1) Select an existing cursor and click **Copy**. 2) Choose an empty thumbnail box and click **Paste**. Edit the new Custom Cursor. 4) Edit the new Custom Cursor's cursor mask.*

   a) Select the Custom Cursor that most closely matches the new cursor you want to create.

   b) Click the **Copy** button

   c) Select one of the empty Custom Cursor thumbnail boxes.

   d) Click the **Paste** button

   A duplicate of the chosen Cursor is now available for editing.

3. Using the Pencil, create or edit your new cursor in the bitmap grids on the right.

   Draw your new cursor by clicking white squares to make them black, or clicking black squares to make them white. Hold down the mouse button to draw a continuous line. As you draw, the pattern is reflected in the cursor's thumbnail box on the left.

4. When you've finished drawing your new cursor, create your Cursor Mask (Figure 4.9).

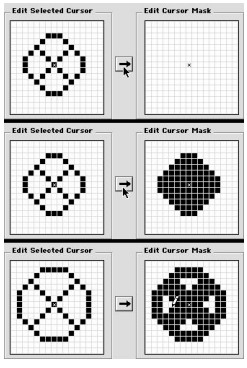


Figure 4.9: A cursor without its cursor mask (top); after using the arrow button to create an initial cursor mask (middle); editing the cursor mask (bottom).

Use the arrow button to transfer a filled-in outline of your design to the **Edit Cursor Mask** grid. The black in the mask bitmap is a map that controls how your cursor will overlay backgrounds. While creating your mask, periodically pass your cursor over the six patterned test boxes below the bitmap grids to see how your cursor will appear over different backgrounds (Figure 4.10). If you want a white edge around your cursor (like the standard arrow cursor has), add some black around the edge of your mask. If you want portions of your cursor to be transparent, such as a doughnut shape, make the black mask pixels white. Experiment with your creation until you have your cursor looking as you want.

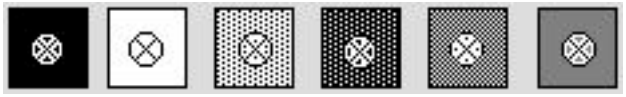


Figure 4.10: The cursor from Fig. 4.11 as it appears when passed over the six patterned test boxes.

5. Click the large X button, and place an X where you want your new cursor's click point to be. This will be the active point for your cursor.
6. Click **OK** to close the Cursor Library and save your changes. Hit Cancel to close the Cursor Library and discard any changes.

The Custom Variables Library

While onViz has a number of built-in variables, as you get into your application development you may find instances to use custom variables. This chapter deals with creating these custom variables. See chapter 13, Variables, to learn about how custom variables are used in your application, and when to use a Text Custom Variable vs. a Numeric Custom Variable.

The Custom Variables Library entries can be accessed in the following ways:

- Choosing **Custom Variables Library** (⌘3) from the **Libraries** menu.
- Selecting Custom Variable in a Calculator State.
- Choosing the Custom Variable tab from a Conditional Path's InfoCenter, or when selecting a variable for substitution in a text object.

The Custom Variable Library Dialog

The Custom Variable Library dialog displays two selections: the **Custom Variables** list on the left, and a panel on the right that displays information about the selected custom variable. At the bottom of the window are buttons for creating new custom variables, importing custom variables, or deleting existing custom variables.

The **Custom Variables** list can be limited to display only Numeric Variables, only Text Variables, or all variables, depending on the selection made from the **Show:** dropdown menu. When a custom variable is selected from the list, the panel on the right displays the **Selected Variable Name**, **Initial Value**, and **Description**. If a numeric variable is selected from the list, the panel on the right also contains the variable's **Display Format**.

Creating a New Custom Numeric Variable

1. Open the Custom Variables Library dialog by choosing **Custom Variable Library** (⌘3) from the **Libraries** menu (Figure 4.11).

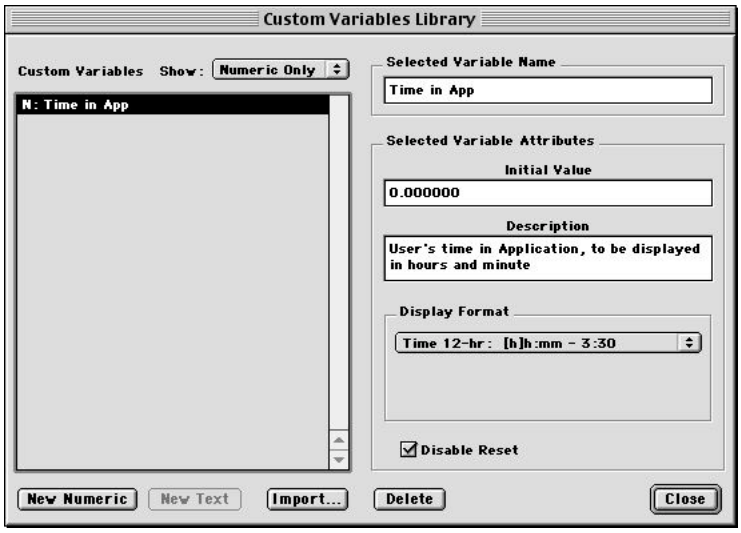


*Figure 4.11: The Custom Variables dialog, with **Numeric Only** chosen from the **Show:** dropdown menu.*

2. From the **Show:** dropdown menu, choose either **All** or **Numeric Only**. **Numeric Only** filters out any existing text variables and disables the **New Text** button, while **All** displays all existing variables and allows you to create both variable types.
3. Click the **New Numeric** button.

   A new numeric variable called Untitled appears in the **Custom Variables** list. The **N:** appearing before the name designates this as a numeric variable. The name Untitled also appears in the **Selected Variable Name** field.
4. In the **Selected Variable Name** field, type a unique, meaningful name for your new variable. If you name it according to its function, it may be easier to identify when the time comes to use it again. As you type the new name, it is reflected in the **Custom Variables** list.
5. If you wish, you can enter an **Initial Value** for the new variable. This value is the "starting point" for this variable. The variable's value can be displayed with variable substitution, or used as a comparison in a Conditional Path. Once defined, a variable's value can be changed with a Numeric Calculator.
6. Enter a **Description** for your new variable. Using this field is a good method for documenting your work, as the description can provide information about what the variable does in the application.
7. Choose a **Display Format** for your numbers:
   - **Fixed Decimal Points** – Your numbers can be displayed with the minimum possible decimals, or with one, two, or a specified number of decimal points. Designating zero decimal points displays the value of the variable rounded to the nearest whole number.
   - **Integers** – Your numbers can be rounded to the nearest whole number, or truncated to display the whole number portion of the value only (does not round).
   - **Scientific Notation** – Displays your number in Scientific Notation format. For example 10,000 is displayed as 1.0E+3 and 0.001 is displayed as 1.0E-6.
   - **Time** – Your numbers can be displayed in one of six different 12 and 24-hour time formats. You will find this format useful when displaying current time or time spent in the application. For example, the value of the General Variable **Application's Time So Far** is the number of seconds that your user has been in the application. You could display this variable to your user, but perhaps displaying the time in hours, minutes and seconds would be more appropriate. By creating a custom variable, you can use a calculator to automatically translate the seconds and display in the format you prefer.
   - **Date** – Your numbers can be displayed in one of seventeen different date, month, and year formats. You can create a custom variable and use a calculator to convert one of onViz' general date variable formats to display a format you prefer.
   - **Currency** – Your numbers can be displayed as dollars and cents.

- **Random** – A numeric variable can generate random numbers between 0 and 32767. If this option is chosen, Min and Max fields appear, allowing you to specify a range from which your random numbers are selected.

8. **Show Commas** – Inserts commas every three decimal places when displaying this custom variable.
9. **Disable Reset** – There is an option within the Calculator States that resets variables to the **Initial Value** set in the Custom Variable Library. Check the **Disable Reset** option for any variables you do NOT want reset.
10. Click **New Numeric** to create another custom numeric variable. Otherwise, click **Close** to close the Custom Variable Library.

You can edit any of the existing variables in the library by clicking on the variable in the list and changing its name, initial value or description.

Creating a New Custom Text Variable

1. Open the Custom Variables Library dialog by choosing **Custom Variable Library** (⌘3) from the **Libraries** menu (Figure 4.12).

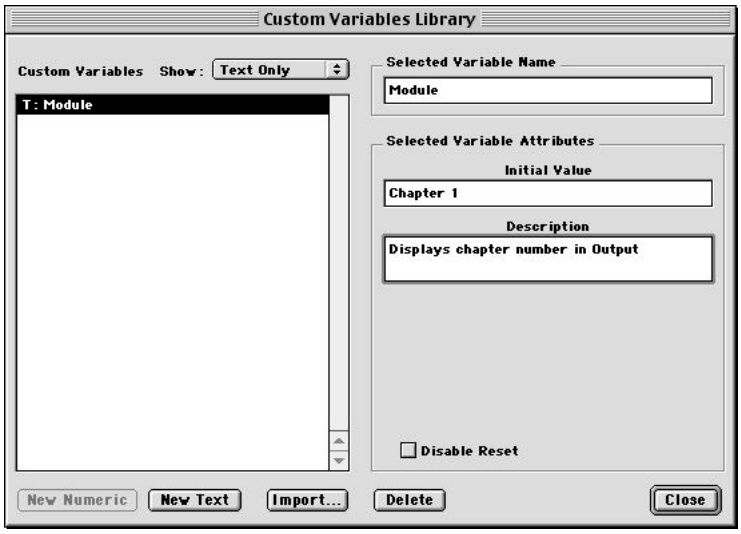


*Figure 4.12: The Custom Variables dialog, with **Text Only** chosen from the **Show:** dropdown menu.*

2. From the **Show:** dropdown menu, choose either **All** or **Text Only**. **Text Only** filters out any existing numeric variables and disables the **New Numeric** button, while **All** displays all existing variables and allows you to create both variable types.
3. Click the **New Text** button.

   A new text variable called Untitled appears in the **Custom Variables** list. The **T:** appearing before the name designates this as a text variable. The name Untitled also appears in the **Selected Variable Name** field.
4. In the **Selected Variable Name** field, type a unique, meaningful name for your new variable. If you name it according to its function, it may be easier to identify when the

time comes to use it again. As you type the new name, it is reflected in the **Custom Variables** list.

5. You may choose to enter an **Initial Value** for the new variable. In the case of a Text Variable, this value is the text that is displayed when this Variable is used for substitution or for comparison in a Conditional Path. Once defined, a Text Variable value can be changed with a Text Calculator. For example, if you are designing an application with multiple modules and you want to use a common background for each module, yet display the chapter name, you could create a Text Variable named "Module" with an initial value of "Chapter 1." Create another Text Variable named "Chapter 2." As your user goes from the first module to the second, you can run the application through a Text Calculator to change the text variable "Module" to "Chapter 2."
6. Enter a **Description** for your new variable. This is a good method for documenting your work. If you are working with a team of developers, providing a description gives information about what the variable does in the application.
7. There is an option within the Calculator States that resets variables to the **Initial Value** set in the Custom Variable Library. If you do NOT want to allow your custom variable to be reset, check the **Disable Reset** option.
8. Click **New Text** to create another custom text variable. Otherwise, click **Close** to close the Custom Variable Library.

You can edit any of the existing variables in the library by clicking on the variable in the list and changing its name, initial value or description.

Deleting Variables

Selecting **Delete** removes the custom variable from the library and all reference to the variable in the application. Be certain that you indeed want to delete a variable, as there is no Undo for this option.

Click **Close** to close the Custom Variable Library and save all changes.

The Font Library

The Font Library allows you set up the fonts used in your application for both Macintosh and Windows operating systems. Setting up fonts in the Font Library helps you avoid conflicting font ID issues that may occur when your application is played on a computer that may have a different ID assigned to the designated font.

In addition to selecting and naming the Macintosh and Windows font, you can designate a substitute display font in case the computer running your application does not have the designated font installed.

Note that the name displayed in the Font Library and Font dropdown menu can be the same as the font selected, or you may wish to give the entry a name such as Main Heading. In this manner you can select the font library item for all headings. If, in your development, you decide that instead of Helvetica for your headings you would prefer Times, just go to the library item and select a different font. All your headings are now displayed in Times.

Tip

To add a new font to your project, the font must be installed on your computer.

The Font Library entries can be accessed in the following ways:

- By choosing **Font Library** (⌘4) from the **Libraries** menu.
- By opening an Input State InfoCenter and clicking the **Font:** dropdown menu under the **Attributes** tab.
- By opening an Output State Presentation window, bringing the Text Output window to the foreground (⌘⇧T), and then selecting **New**, **Edit Current Font**, or **Other** from the **Font** menu.
- By opening the Image Editor, selecting one of the Text tools and clicking in the design area, and then selecting **New**, **Edit Current Font**, or **Other** from the **Font** menu.

Tip

Note: An alert is displayed if you are importing text or images that contain fonts not set up in the Font Library. You will have the option to automatically add them to the library.

The Font Library Dialog

The Font Library Dialog displays a list of all Font Library entries that have been set up for the application (Figure 4.13). A check mark in front of the name indicates that the name will be included in the Font dropdown menus. This check mark can be toggled off and on by clicking in front of the name.

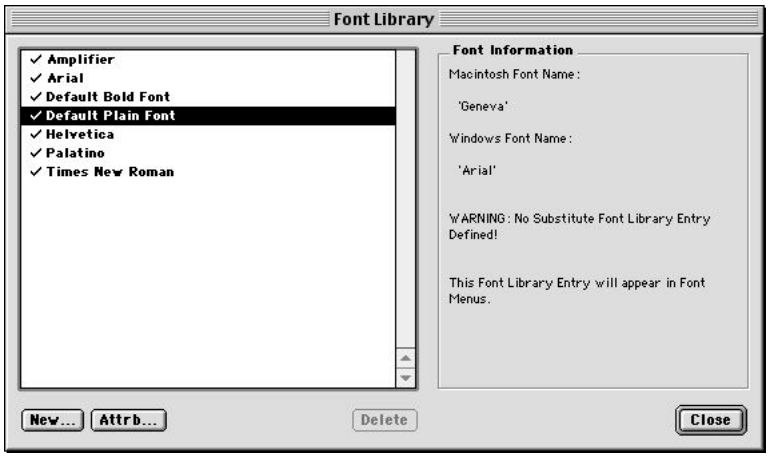


*Figure : 4.13 The Font Library dialog. Note the **Font Information** displayed on the right for the entry selected on the left.*

When you click on a library entry, the right-hand side of the dialog provides information about it, such as its Macintosh and Windows names and whether it is set to appear in the Font dropdown menus.

At the bottom of the dialog are the **New** button and the **Attrb** (Attributes) button, both of which open the Font Editor dialog. The **New** button is used to setup a new Font Library entry. The **Attrb** button is used to edit an existing Font Library entry.

onViz contains two default font entries: Default Plain Text and Default Bold Text. Both of these entries are preset with system fonts typically installed on the Macintosh and Windows. Plain Text is set with Geneva for the Macintosh font and Arial for the Windows font. Bold Text is set for Chicago as the Macintosh font and System for the Windows font.

The Font Editor Dialog

The Font Editor dialog displays the following selections (Figure 4.14):

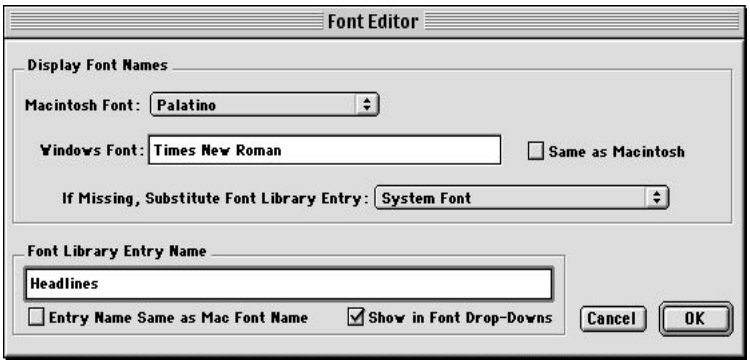


Figure 4.14: The Font Editor dialog.

- **Display Font Names** – The Display Font Names section contains a **Macintosh Font:** dropdown menu, a **Same as Windows** check box, a **Windows Font:** field, and a **Same as Macintosh** check box. If the font is named the same on Windows computers as it is on Macintosh computers, leave the check mark in the **Same as Macintosh** check box to have the **Windows Font:** field automatically filled in.
  If you want to use a different display font on Windows (for example, Helvetica on the Macintosh and Arial on Windows), deselect **Same as Macintosh** check box and enter the name of the Windows font. Ensure that Windows fonts are named in this field exactly the same as they are by the Windows operating system.
- The **If Missing, Substitute Font Library Entry** dropdown menu allows you to select an alternative Font Library entry in case your selected font is not installed on the users computer. It is a good idea to set up a default Font Library entry with fonts that are installed with the System software, such as Times/Times New Roman (for serif fonts) or Geneva/Arial (for sans-serif fonts).
- **Font Library Entry Name** – The Font Library Entry Name section includes the options to make the font's **Entry Name Same as Mac Font Name** and to **Show in Font Drop-Downs**. If **Entry Name Same as Mac Font Name** is selected, the font's library entry name is the same as the name of the actual Macintosh font. Deselect this option if you

want your font's library entry name to be different than its Macintosh font name. The **Show in Font Drop-Downs** option allows the font's library entry to be accessed from the Font dropdown menus.

Adding a New Font or Editing an Existing Font Library Entry

1. Choose **Font Library** (⌘4) from the **Libraries** menu.

   The Font Library dialog is opened.

2. To create a new Font Entry, click the **New** button. To edit an existing Font Entry, select a Font from the list, and click the **Attrb** button, or double-click on the font entry.

   The Font Editor dialog is opened.

3. Using the **Choose From Installed Fonts:** dropdown menu, select a new font for your project.

4. Using the **Substitute Font** dropdown menu, select a font that can be used if the designated font is not installed on the user's computer.

5. Once a font has been selected, the **Macintosh Font:** field automatically fills in. If the font is named the same on Windows computers as it is on Macintosh computers, leave the check mark in the **Same as Macintosh** check box to have the **Windows Font:** field automatically filled in.

   If you want to use a different display font on Windows (for example, Helvetica on the Macintosh and Arial on Windows), deselect **Same as Macintosh** checkbox and enter the name of the Windows font. Note that for some fonts, such as Times, the Macintosh name is Times and the Windows font name is Times New Roman.

6. Give the font a **Font Library Entry Name**.

   For most fonts, you'll probably want the Library Entry Name to be the same as the actual font name, so you can leave the check mark in the **Entry Name Same as Mac Font Name** check box to have this field filled in automatically. However, you might choose to name your Font Library entries according to some other criteria, such as their purpose, for example, 'Headline Type." If so, remove the check mark from the **Entry Name Same as Mac Font Name** check box and type a name for this entry that you consider to be more unique and meaningful.

7. Decide if you want your Font Library Entry to appear in the Font dropdown menus. The Font dropdown menus are a convenient way to access your font listings, but may get too long if you have added a lot of fonts. Leave the **Show in Font Drop-Downs** check box checked if you plan on using the font often.

8. Click **OK** to close the Edit Font dialog and return to the Font Library.

   Your new entry is now in the list of available fonts.

9. Click **Close** to close the Font Library.

To delete a font library entry, select the font entry in the list and click **Delete**. Note that any text assigned to be displayed with this font is now displayed with the System Font.

The Image Library

onViz' Image Library lets your entire project share graphics from a common source. By sharing graphics in this way, you reduce your project's overall file size, while allowing it to display a consistent appearance across all its supporting documents. In addition, the Image Library provides a quick way to update existing images. Once edited, the change is reflected throughout your entire application, rather than having to edit the image in each place it is used.

Using the Image Library, you can:

- Create and delete a new image.
- Edit an existing image.
- Convert an image's file format, color depth, and resolution.

The Image Library entries can be accessed in the following ways:

- Choosing **Image Library** (⌘5) from the **Libraries** menu.
- Opening an Output State and choosing from the foreground/background image dropdown menu or command double-click to create/edit a background image or option command double-click to create/edit a foreground image.
- Opening the sprite palette and choosing from the sprite drop down menu or double-clicking a sprite location.
- Double-clicking a sprite on an animation frame.
- From the Multimedia State **Overlay Image** option.

The Image Library Dialog

The Image Library Dialog displays a list of all images that have been created for this application (Figure 4.15). A check mark in front of the name indicates that the name is included in the Image drop down menus. This check mark can be toggled off and on by clicking in front of the name.

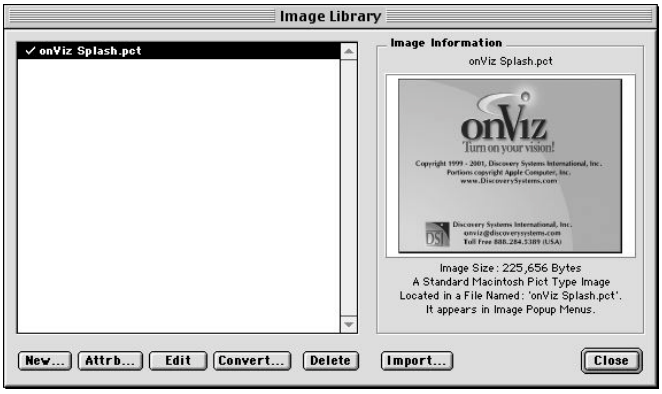


Figure 4.15: The Image Library dialog. Note the check marks to the left of the entry names, which indicate that the image will be listed in the library dropdown menus. Also notice the ***Image Information*** *on the right for the entry selected on the left.*

When you click on a library entry, the right side of the dialog displays a thumbnail of the image and provides information about its size, file name, and file type.

Click **New...**to bring up the Image Attributes dialog (described in the next section) from which you can give the Image Library entry a name and go to the Image Editor to create your new image.

With a library name highlighted you can:

Click **Attrb** (Attributes), or double-clicking an entry name, to open the Image Attributes dialog for the selected image.

Click **Edit**, or double-click the image thumbnail, to open the image in the Image Editor.

Click **Convert** to open a dialog that give options for changing the image's color depth and resolution.

Click **Delete** to remove the image reference from the Image Library.

When you delete an image library entry, you are removing it from your entire project. Make sure your are not using the image somewhere else in your application.

Tip

Hold down the option key to delete both the Image Library entry AND the image file stored in the image support folder.

Clicking **Import** opens media import dialog that allows you to import a single image or multiple images into the Image Library.

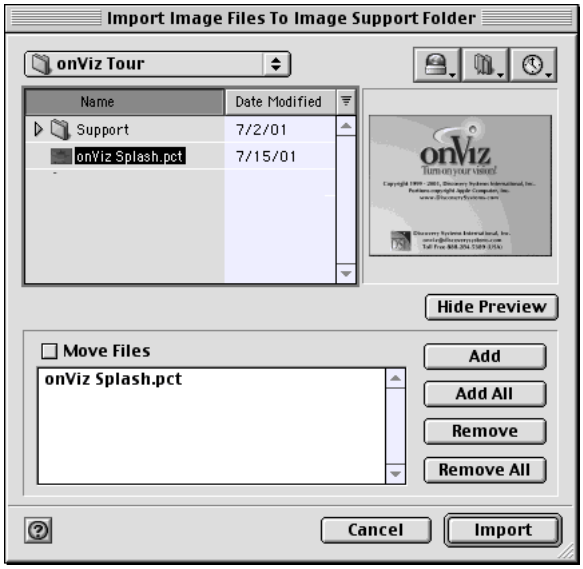


Figure 4.16: The Image Library Import Dialog

The default location for the file import is the last location from which you imported a file. Highlight a file and click add, or double-click a file name to add it to the import list. Click **Add All** to add all the images to the Image Library. A copy of the selected file(s) will be added to the image support folder with a separate library entry created for each imported image. Select **Move Files** if you want to move the file instead of copying it.

The Image Attributes Dialog

The Image Attributes dialog displays the following sections:

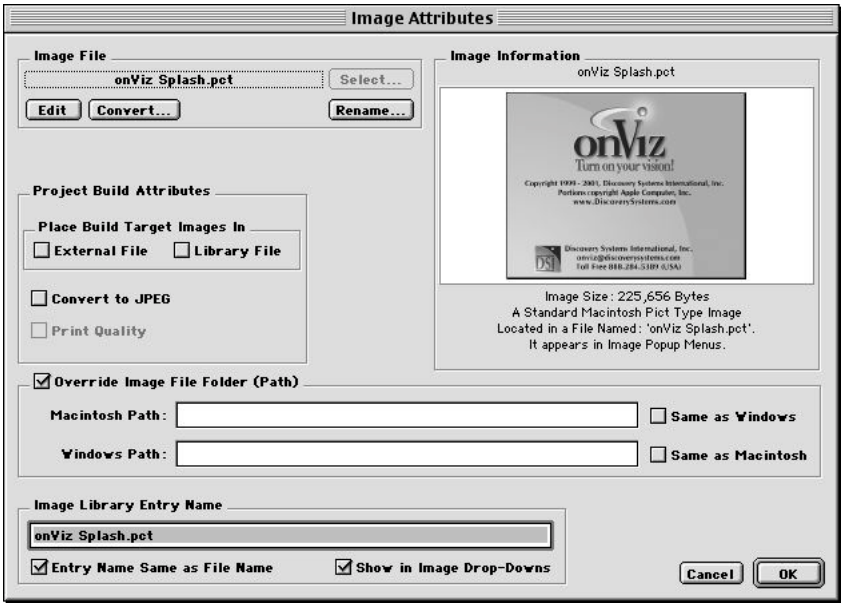


Figure 4.17: The Image Attributes dialog for an internal library entry.

- **Image File** – This group allows you to select and rename existing image files or create new image files for the image library.
  - **Select...** displays a 'Get File' dialog from which you can select a new image to include in the library. Although you can navigate to other locations, the default location that will be opened when clicking **Select...** is the Image folder in the project's Support directory.
  - **Edit/Create** opens the selected image in the Image Editor or, if no image was selected, opens the Image Editor to create a new image.
  - **Rename...** displays an image 'Rename' dialog. Renaming the image from this dialog, also renames the image file stored in the Image Support Folder.
- **Build Project Attributes -** This group includes image settings for your built application. For more information on building target applications, see chapter 2, Planning your application.
  - **Place Build Target Images In:**
    External File always exports the image file to the images support directory when building a multifile application regardless of settings in the Project Build Editor.
    Library File always imports the image file into the application library when building a multifile application regardless of settings in the Project Build Editor.
  - **Convert to JPEG** - Converts the image to a jpeg file when building the project. Default quality setting is 'medium'. Selecting **Print Quality** sets the quality to high.

NOTE: Variable Substitution is not supported in an image that has been converted to JPEG. Also, do not convert images to JPEG that use the Mask Pixels feature.

Tip

Select the **Convert to JPEG** option only for large bitmapped images. Typically object oriented images will become larger when converted to JPEG.

- **Override Image File Folder (Path)**. If the image for this library entry is not in the Image Support location, for example stored on a CD-ROM or on a server, you must specify the file path in this field. This location overrides the Image Support location set up when you build your target application. For cross-platform development, you can set different locations for Macintosh and Windows.
- **Image Information** – The Image Information section shows a thumbnail of the image, and details its current attributes, including its name, file size, file format, and whether or not it appears in the image dropdown menus. You can double-click this image to open the image in the Image Editor.
- **Image Library Entry Name** – The Image Library Entry Name section includes the options to make the image's **Entry Name Same as File Name** and to **Show in Image Drop-Downs**. If **Entry Name Same as File Name** is selected, the image library entry name is the same as the name of the image file. Deselect this option if you want a different name for your image library entry. The **Show in Image Drop-Downs** option allows the image's library entry to be accessed from the Image dropdown menus.

Creating a New Image Library Entry

1. Open the Image Library dialog by choosing **Image Library** (⌘5) from the **Libraries** menu.
2. Click the **New** button, which opens the Image Attributes window.
3. Select whether you want the default for this image to be referenced from an External file or from the Internal Library. Note that you can override this option when you build your application.

   From this point you can either select an image that you already have on disk, or create a new image in the Image Editor.
4. To select an image, click the **Select** button.

   A dialog opens allowing you to navigate to the image file.
5. When you find the file, select it and click **Open**.

   If the image is not already in your Image Support Folder, a dialog is presented that allows you to either move the file or create a copy of the file in your Image Support Folder. Select the option you prefer. After the image has been selected, you can click the **Edit** button to open the image in the Image Editor.
6. To create a new image, give your new image a name in the **Image Library Entry Name** field. The name you type in this field will be the name of the image file in your image support folder.
7. To create or edit another image, reopen the Image Library or select from the Image Editor's Image Controls drop down menu.
   Otherwise, close the Image Editor by clicking the Close Box, or by choosing **Close Window** (⌘W) from the **File** menu.

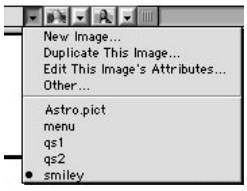


Figure 4.18: The Image Editor's Image Control dropdown menus. Use this menu to create or edit another image, or to reopen the Image Library.

When you close the Image Editor, a file is in the Image Support folder with the same name as your Library Entry Name. This file name will be appended with a .pct extension.

Editing an Existing or Imported Library Entry

1. Open the Image Library dialog by choosing **Image Library** (⌘5) from the **Libraries** menu.
2. Choose the image you'd like to edit from the list on the left.
3. Click the **Edit** button.

   Your chosen image opens inside the Image Editor.
4. Edit the image as you want.
5. To edit another image, reopen the Image Library, or select from the Image Editor controls drop down menu. To close the Image Editor, click the Close Box, or choose **Close Window** (⌘W) from the **File** menu. When you close the Image Editor, you are presented with a dialog that gives you the option to update the original image file or save the updated file with a new name. This dialog contains a **Do not show this dialog again** option. By selecting this option, onViz will always update the image file in the Support folder when closing the Image Editor window.

Converting an Image's File Format or Color Depth

1. Open the Image Library dialog by choosing **Image Library** (⌘5) from the **Libraries** menu.
2. Select the image you'd like to convert from the list on the left.
3. TBA

For more information on the Image Editor and manipulating graphics, see chapter 7, The Image Editor.

Deleting an Entry from the Image Library

1. Open the Image Library dialog by choosing **Image Library** (⌘5) from the **Libraries** menu.
2. Select the image you want to delete from the list on the left.
3. Click the **Delete** button.

   A warning message appears, asking you "Deleting this entry from the Image Library cannot be undone, delete it anyway?" Hold down the Option key when clicking delete to remove the item with no warning dialog.
4. Click **OK** to dismiss the warning message and complete deleting the image.

   The image is removed from the list.
5. Click **Close** to close the Image Library.

Deleting an image library entry does not remove it from the Image Support folder, but it does remove the reference for the current and all supporting documents.

The Movie Library

The Movie Library lets you designate settings for how digital movies and QuickTime VR movies play within your application. These movies can be played in an Output frame or, using the Multimedia State, can play anywhere in your application.

With the Movie Library, you can:

- Create a new Movie library entry.
- Edit the attributes of an existing movie, such as looping the movie, setting the movie window size and screen position, adjusting sound controls, making the movie autostart, providing user controls, and more.
- Play a movie.

Tip

To play mp3 sounds in your onViz application, save the mp3s in the Movie Library

The Movie Library entries can be accessed in the following ways:

- By choosing **Movie Library** (⌘6) from the **Libraries** menu.
- By opening an Output State and choosing from the movie dropdown menu on the Frame Control palette.
- Multimedia State Play Movie option.

The Movie Library Dialog

The Movie Library Dialog displays a list of all movies that have been associated with this application. A check mark in front of the name indicates that the name will be included in the Movie drop down menus. This check mark can be toggled off and on by clicking in front of the name.

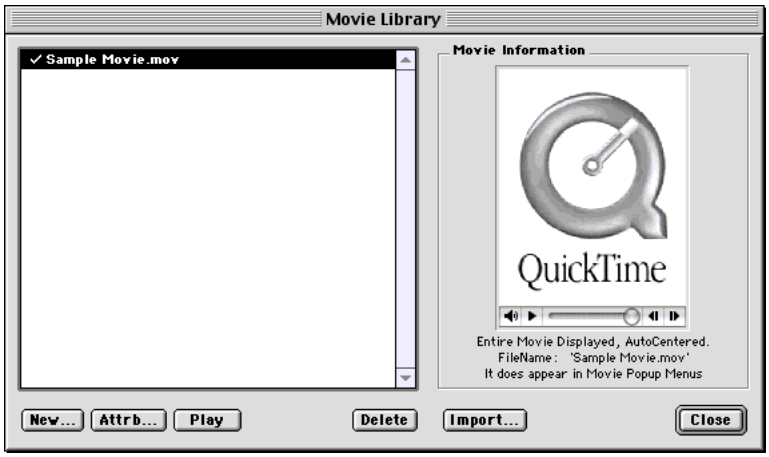


Figure 4.19: The Movie Library Dialog Window

When you click on a library entry, the right-hand side of the dialog displays the movie's attributes, such as its file name, whether it will appear in the drop down menus, and its default target reference (referenced from an external file or the internal library).

At the bottom of the dialog are the **New**, **Attrb** (Attributes), **Play**, **Delete** and **Import** buttons. The **New** button is used to setup a new Movie Library entry. The **Attrb** button is used to edit

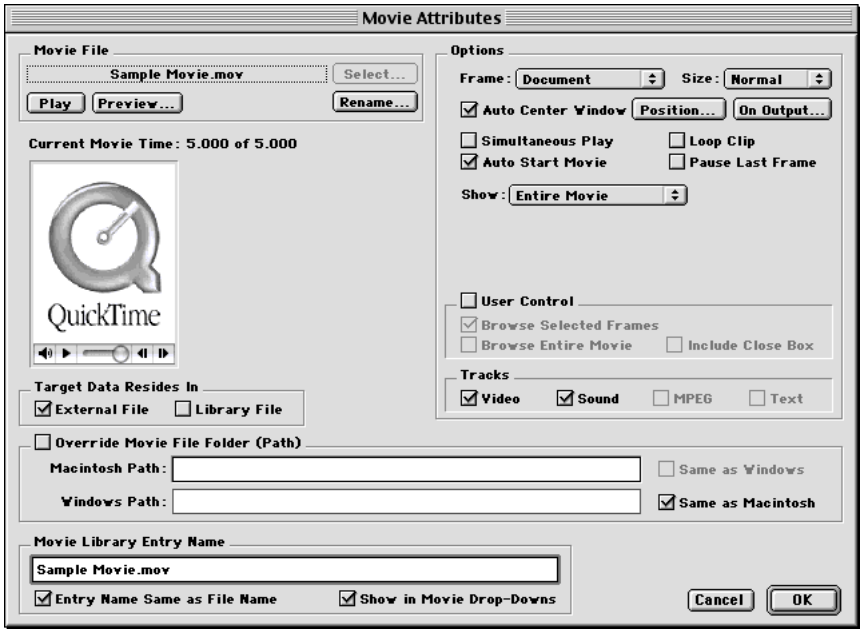
the attributes of an existing Movie Library entry. Both of these buttons open the Edit Movie dialog. The **Play** button is used to preview the selected movie clip, the **Delete** button deletes the selected movie clip entry from the Movie Library, and the **Import** displays the same file import dialog as is displayed when importing images into the Image Library.

Tip

Hold down the option key to delete both the Movie Library entry AND the movie file stored in the movies support folder.

The Movie Editor Dialog

The Movie Editor dialog displays the following selections:



- **Movie File** – This group allows you to select and rename movie files for your project.
  - **Select...** displays a 'Get File' dialog from which you can select a new movie to include in the library. Although you can navigate to other locations, the default location that will be opened when clicking **Select...** is the movie folder in the project's Support directory. A thumbnail version of the movie appears in the Movie Editor dialog, with the **Current Movie Time** displayed above it.
  - **Rename...** displays a movie 'Rename' dialog. Renaming the movie from this dialog, also renames the mvoie file stored in the Movies support Folder.
  - **Play** plays the movie in the Movie Attributes dialog**.** You can also play the movie withing the Movie Attributes dialog using the standard quicktime controls under the sample display. QuickTime VR movie controls are always active in the Movie Editor.
  - **Preview...** lets you select a background Output and view the movie as it will look at runtime, with your selected border and control options. To exit the movie and return to the Movie Editor dialog, hold down the Command key while pressing the period key (⌘ .), or choose **Close Window** from the **File** menu.
- Current Movie Time **is the time code for the frame of the movie that is currently displaying in the movie window.**

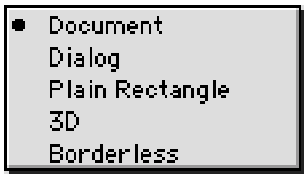
Tip

You can copy the current movie frame by choosing Copy (⌘C) from the Edit menu. This is helpful if you want to include thumbnails of the movies on your output - simply copy the desired movie frame, add the image(s) to the Image library, and select them to display in the output.

- **Target Data Resides In -** This group includes movie settings for your built application. For more information on building target applications, see chapter 2, Planning your application.
  External File always exports the movie to the movie support directory when building a multifile application regardless of settings in the Project Build Editor.
  Library File always imports the movie into the application library when building a multifile application regardless of settings in the Project Build Editor.

  During application development, all movies are referenced from disk and are stored as files in the Movies Support folder. When you're ready to build your application for distribution, you have the option of importing the movies to make them resident in your application, or referencing them from disk. For Internet or server-based delivery of your application, you will have a much smaller file download size by streaming movies from a directory on the server.
- **Options** – The Options section contains the following settings:

  a) **Frame** – Choose a Frame for your movie from the dropdown menu:

  - Document – Rectangular border with a title bar and slight three-dimensional effect.
  - Dialog – Rectangular border embellished with small inside border.
  - Plain Rectangle – Rectangular border with no additional embellishments.
  - 3-D – Rectangular border with a three-dimensional drop shadow.
  - Borderless – No border is placed around the movie.

  b) **Size** – Choose a Size for your movie from the dropdown menu. Options include One Fourth, One Third, One Half, Normal, Double, Triple, Quadruple, or Custom. If the Custom movie size is chosen, a frame appears which you can adjust to the size you want the movie to be.

  c) **Auto Center Window** – Displays the movie frame in the middle of the target screen.

  d) **Position** – Allows you to manually position the movie frame using an Output Window as a background. Selecting this option automatically deselects the **Auto Center Window** option. The first time you select **Position** for a movie, you are presented with the Background Output dialog from which you can select an Output Window as a background for positioning.

  With the positioning window visible, you can drag the movie with the drag handle in the upper left corner to your desired screen position.

  Pressing the Return or Enter keys closes the position window and returns you to the Movie Editor dialog, saving the new position. Holding down the Command key while pressing period (⌘ .) closes the positioning window and returns you to the Movie Editor dialog without saving any position change.

Tip

Good Practice Tip - Resizing a movie window size may adversely affect its performance. You will typically have better results by creating the movie the size you want it to play.

Note that, at runtime, the Movie window is always the front-most window, and is displayed on top of Output Windows, sprites, Multimedia Overlays, and any Input windows.

e) **On Output** – Opens the Background Output dialog, allowing you to choose an Output State to serve as a background for positioning the movie. After choosing a background output, onViz displays the positioning window as described above.

Since QuickTime VR movies are single scene movies, the following options are not applicable to QuickTime VR movies.

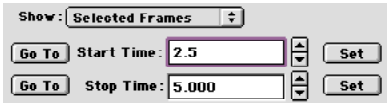
f) **Simultaneous Play** – **Simultaneous Play** – Allows the movie to play while the application continues to run. This option works well in conjunction with Smart Sprites that act as button controls for the movie. Deselecting this option will show the entire movie before the application continues. Note: when selecting the **Simultaneous Play** option, the majority of the computer's processing power will be dedicated to the QuickTime movie. Animation sequences are not as smooth as they are with no QuickTime movie playing.

g) **Auto Start Movie** – Causes the movie to start playing as soon as it is encountered by either the application flowing through a Multimedia State or entering an animation frame with a Movie designated to play. If you deselect this option, be sure to give the user some indication of how to start the movie. You can either display the user controls or include an onscreen message telling the user to double-click the movie window to start it.

h) **Loop Clip** – Causes the movie to keep repeating until the application encounters an Output frame or Multimedia State set to **Stop Any Movie Action** from the Movie drop down list. Selecting this option also automatically selects the **Simultaneous Play** option and deselects the **Pause Last Frame** option. Note that movie settings in the Multimedia State can override this option.

i) **Show:** – Determines how much of the movie will be played:

- Entire Movie – The movie is played in its entirety.
- Selected Frames –Plays only designated frames of the movie. This option adds additional controls to the Edit Movie window that designate a Start Time and a Stop Time.



You can either type specific times in these fields, or navigate to the frame you want to start with the QuickTime control and click the **Set** button beside the **Start Time** field. Navigate to the frame you want the movie to end and click the **Set** button beside the **Stop Time** field. Clicking one of the **Go To** buttons will display the movie frame corresponding to the time set in the respective field.

Preview and **Play** will display the segments of the movie as designated by the Start and Stop times.

- Single Frame – Plays a single designated frame of the QuickTime movie. Controls are same as for Selected Frames, except the dialog displays only the Start Time field. Note that selecting this option automatically selects the **Simultaneous Play**, **Auto Start**, **and Pause Last Frame** options.

j) **User Control** – Displays the standard Apple QuickTime control so the user can manually view segments of the movie. Note that this option is automatically turned on for QuickTime VR movies.

* Browse Selected Frames – User can view any of the frames specified in the **Show** field. This option is only available if Selected Frames was chosen from the **Show:** dropdown menu.
* Browse Entire Movie – User can view the entire movie regardless of what was selected from the Show dropdown menu.
* Include Close Box – Adds a Close Box to the movie frame. Only available if the Document frame type is selected.

k) **Tracks** – Specifies which of the movie's tracks will be played. Options include Video, Sound, MPEG, and Text.

- **Override Movie File Folder (Path)** – These fields are displayed if the **External File** option is selected as the Default Target Reference. If the movie for this library entry is not in the Movie Support location, for example stored on a CD-ROM or server, you must specify the file path in this field. This location overrides the Movie Support location set up when you build your target application. For cross-platform development, this location may be different on Macintosh and Windows.
- **Movie Library Entry Name** – The Movie Library Entry Name section includes the options to make the movie's **Entry Name Same as File Name** and to **Show in Movie Drop-Downs**. If **Entry Name Same as File Name** is selected, the movie's library entry name will be the same as the name of the movie file. Deselect this option if you want your movie's library entry name to be different than its file name. The **Show in Movie Drop-Downs** option allows the movie's library entry to be accessed from the Movie dropdown menus.

Creating a New Movie Library Entry

1. Open the Movie Library dialog by choosing **Movie Library** (⌘6) from the **Libraries** menu.
2. Click the **New** button to create a new movie entry.

   The Get File dialog is displayed, from which you can navigate to the movie you want to reference and click **OK** to select it.
3. The movie file will be added to the Movie Support folder with a .mov extension and the Movie Editor dialog is opened.
4. A thumbnail version of the movie appears in the Edit Movie dialog, with the **Current Movie Time** displayed above it.

5. Set the movie's **Options**. See the above section on the Movie Editor dialog for an explanation of these fields.
6. In the **Movie Library Entry Name** field, you can let onViz default the name based on the movie file name. Deselect this option if you want a different name for your movie entry.
7. To place the movie's library entry in all the movie dropdown menus**,** leave the **Show in Movie Drop-Downs** check box selected.
8. Click **OK** to close the Edit Movie window and return to the Movie Library.
9. Click **Close** to close the Movie Library.

Deleting a Movie Entry from the Movie Library

To delete an entry from the Movie Library, highlight the Movie entry you wish to delete and click the **Delete** button.

Deleting a movie from the Library does not remove it from the Movie Support folder but it does remove the reference for the current and all supporting documents.

The Sound Library

The Sound Library lets you designate settings for how sounds play within your application. These sounds can be played in an Output, or, using the Multimedia State, they can be played anywhere in your application.

Tip

To stream sound from a server, save as a QuickTime movie andaccess through the Movie Library.

You can use the Sound Library to:

- Create a new Sound library entry.
- Edit an existing sound.
- Play a sound.
- Convert a sound's format and other characteristics
- Create Text for Phonetic Speech (text to speech).

Sound types supported include AIFF, WAV, and Macintosh SND. Both AIFF and WAV are supported on Windows, so if you are developing cross-platform applications, it is a good idea to save or convert all sounds to this format.

onViz also supports Phonetic speech on the Macintosh. The Sound Library lets you select voices and add text that will be "spoken" at runtime.

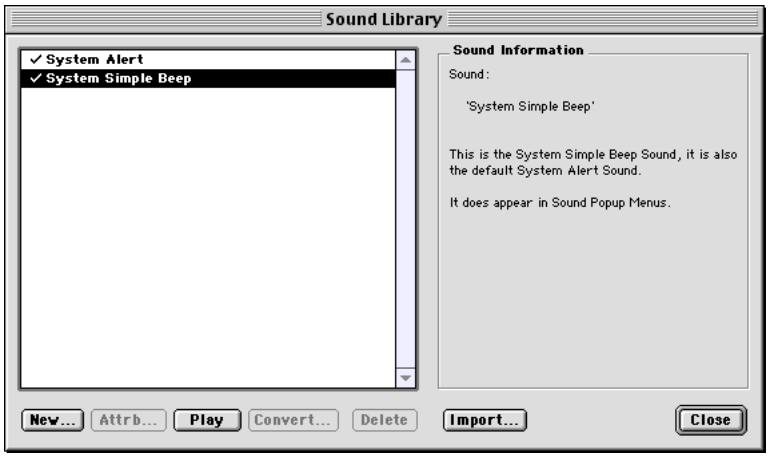
Note: In order for your users to hear the phonetic speech, they must have MacinTalk installed in their system. This option is not currently available for playback on Windows operating system.

The Sound Library entries can be accessed in the following ways:

- Choosing **Sound Library** (⌘7) from the **Libraries** menu.
- Opening an Output State and choosing from the sound dropdown menu on the Frame Control palette.
- Selecting the Multimedia State **Play Sound** option.

The Sound Library Dialog

The Sound Library Dialog displays a list of all sounds that have been associated with this application, as well as two default sounds that are not editable: System Alert and System Simple Beep. A check mark in front of the name indicates that the name will be included in the Movie drop down menus. This check mark can be toggled off and on by clicking in front of the name.



The **System Alert** sound is the sound that your user has assigned as the alert sound in the Sound Control panel and may be different on different computers. The **System Simple Beep** is a single beep and will be identical on all computers.

When you click on a library entry, the right side of the dialog displays the sound's attributes, such as its file name, whether it will appear in the drop down menu, and if its default target reference is referenced from an external file or the internal library. Other attributes displayed include the sound entry's file format, channels, sample rate and size, data size, and duration.

At the bottom of the dialog are the **New**, **Attrb** (Attributes), **Play**, **Convert**, **Delete**, and **Import** buttons. The **New** button is used to setup a new Sound Library entry. The **Attrb** button is used to edit an existing Sound Library entry. Both of these buttons open the Sound Editor dialog. The **Play** button is used to preview the selected sound clip, and the **Delete** button removes the selected sound clip entry from the Sound Library.

The **Convert** button opens the Sound Conversion dialog, where you can convert a sound clip's format, channels, sample rate, and sample size

The **Import** button opens the media import dialog that allows you to import one or multiple sounds into the Sound Library.

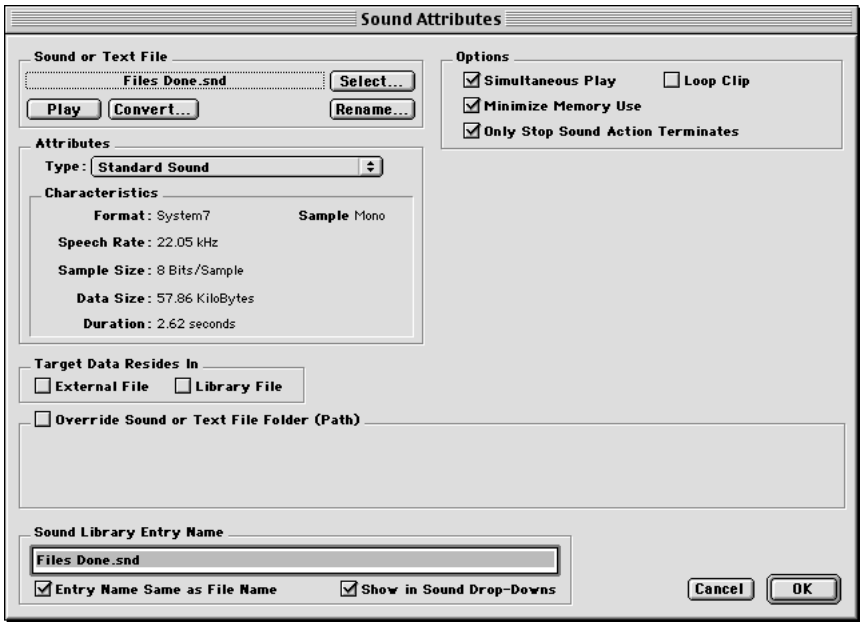
The Sound Editor Dialog

The Sound Editor dialog varies depending on the option chosen from the **Type:** dropdown menu, which contains the following Sound types:

- Standard Sound – Sound files formatted as System 7, Wave, or AIFF files.
- Phonetic Speech (Macintosh Only) – Uses Apple's MacinTalk software to speak text.

Standard Sound

When the Standard Sound type is chosen, the Sound Editor dialog displays the following selections:



- **Sound or Text File** – This group allows you to select and rename existing soundor text files or create new entries for the sound library. NOTE: Text files are used only with phonetic speech (Macintosh runtime only).
  - **Select...** displays a 'Get File' dialog from which you can select a sound or text file to include in the library. Although you can navigate to other locations, the default location that will be opened when clicking **Select...** is the Sounds folder in the project's Support directory. After a sound file is chosen, the **Characteristics** panel displays information about it, including file format, sample rate, sample size, data size, and duration.
  - **Rename...** displays an sound/test file 'Rename' dialog. Renaming the sound or text file from this dialog, also renames the associated file stored in the Sound Support Folder.
  - **Play** allows you to preview the chosen sound.
  - **Convert** displays a dialog from which you can change the sound clip's format, number of channels, sample rate, and sample size. (see Sound Conversion Dialog later in this chapter).
- **Attributes** – The Attributes section contains the **Type:** dropdown men, and a **Characteristics** panel.

  The **Type:** dropdown menu is used to choose between the Standard Sound and Phonetic Speech sound types.
- **Target Data Resides In -** This group includes sound settings for your built application. For more information on building target applications, see chapter 2, Planning your application.

  External File always exports the sound to the sound support folder when building a multifile application regardless of settings in the Project Build Editor.

Library File always imports the sound into the application library when building a multifile application regardless of settings in the Project Build Editor.

During application development, all sounds are referenced from disk and are stored as files in the Sounds Support folder. When you're ready to build your application for distribution, you have the option of importing the movies to make them resident in your application, or referencing them from disk.

- **Options** – The Options section contains the following settings:
  a) **Simultaneous Play** – Allows the sound to play while the application continues to run. Deselecting this option pauses the application until the sound is complete.
  b) **Minimize Memory Use** – This option is particularly beneficial when playing large sound files. onViz loads large sound files into memory in small segments, rather than loading the entire sound in memory. As the first sound segment nears completion, the second segment is loaded, resulting in the sound playing with no interruption. This procedure not only minimizes the amount of memory required, but also allows the sound to begin as soon as it is called by a Multimedia State or Output frame.
  c) **Only Stop Sound Action Terminates** – Once the sound begins playing, it will continue to play until the application encounters a Multimedia State, Output frame, or a Smart Sprite with a mouse action from the Sound dropdown list set to **Silence All Sounds**.
  d) **Loop Clip** – Causes the sound to repeat until the application encounters a Stop Sound Action as described above.
- **Override Sound or Text File Folder (Path)** – These fields are displayed if the **External File** option is selected as the Default Target Reference. If the sound for this library entry is not in the Sound Support location, for example stored on a CD-ROM or server, you must specify the file path in this field. This location overrides the Movie Support location set up when you build your target application.
- **Sound Library Entry Name** – The **Sound Library Entry Name** section includes the options to make the sound's **Entry Name Same as File Name** and to **Show in Sound Drop-Downs**. If **Entry Name Same as File Name** is selected, the sound's library entry name will be the same as the name of the sound file. Deselect this option if you want your sound's library entry name to be different than its file name. The **Show in Sound Drop-Downs** option allows the sound's library entry to be accessed from the Sound dropdown menus.

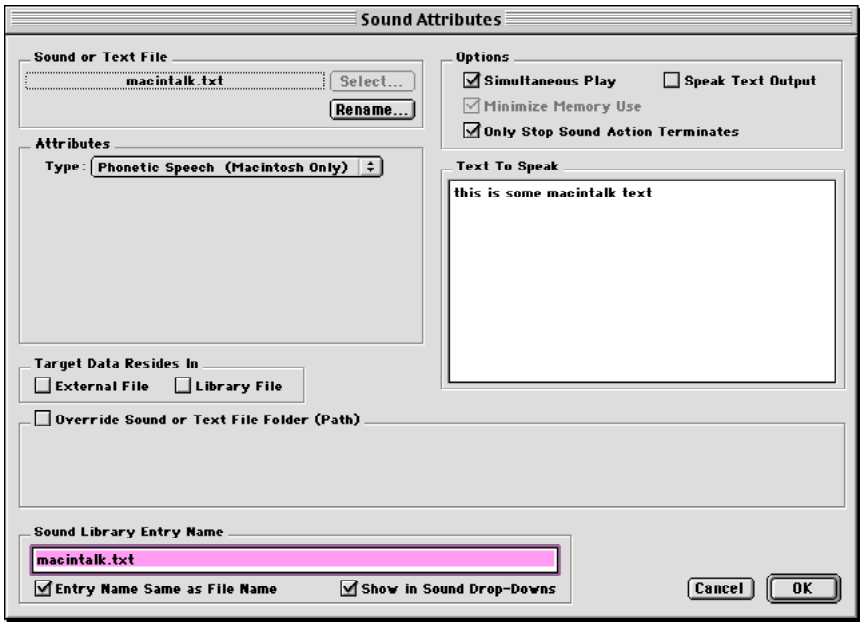
Tip

If you use a Multimedia state to play multiple buffered sounds simultaneously, the sound may stutter while getting the next sound byte. If playing multiple sounds, it is better to deselect this option for all sounds that will be playing simultaneously.

Phonetic Speech (Macintosh Only)

With the exception of the following options, the Sound Editor dialog settings are the same for Phonetic Speech as they are for Standard Sound.

- **Attributes** – Text is "spoken" from a text file. As with the Standard Sound, this file is disk based and stored in the Sound Support folder until the target application is built.



- **Text to Speak –** This edit field is displayed when the Phonetic Speech option is selected. If the default target reference is External, the text from this file will be displayed in this window. If the default target reference is Internal Library, text you type here will be saved into a file named the same as the Library Entry Name with a .txt extension when you close the library.
- **Options –** An option with Phonetic Speech is to 'Speak Text Output.' You can create a specific library entry that speaks the text typed in the Text Output for specific Output frames. Note that at runtime, it is not necessary to display the Text Window for the Phonetic Speech to read the text.
- When the Phonetic Speech option is selected, the **Characteristics** panel displays information for the voice that will be used to speak the text, including its speech rate and pitch. The **Speak** button allows you to preview the way the text is spoken.

Tip

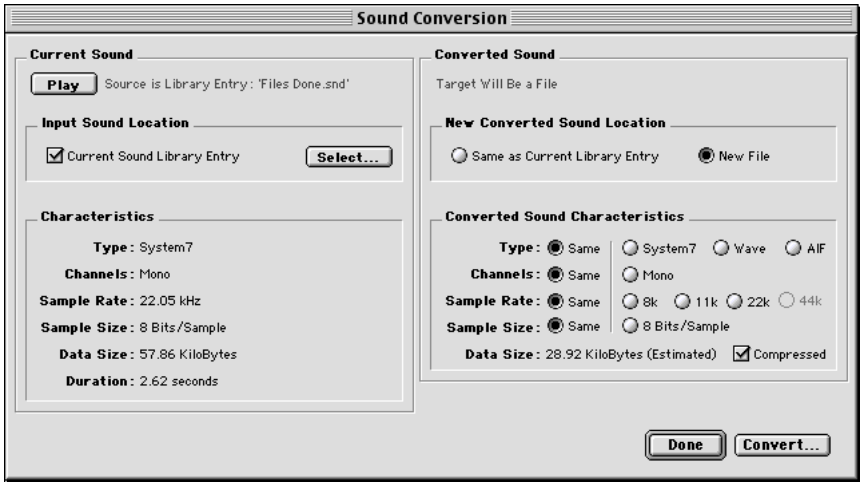
When speaking text, the Macintosh's Speech software often uses non-standard pronunciation. Experiment with different phonetic spellings of your text until the voice sounds more natural.

Tip

If you want to use Phonetic speech to read several text output windows, set up one Sound Library entry with the name "Read Text Window" and make it available in the dropdown menu. This way, the same library entry can be selected for those frames in which you want the Text Window read.

The Sound Conversion Dialog

The Sound Conversion dialog is divided into two sections, Current Sound and Converted Sound:



Best Practice

For cross-platform applications, convert all sounds to either Wave or AIF. onViz will play either of these sounds on both the Macintosh and Windows operating system eliminating the need for duplicate sound files.

- **Current Sound** – The Current Sound section displays the sound's current characteristics, and contains a **Play** button, an Input Sound Location panel, and a Characteristics panel.

  The **Play** button lets you preview the sound in its pre-converted state, while the **Characteristics** panel displays information about it, including its file format, sample rate, sample size, data size, and duration.

  The Input Sound Location panel includes the **Current Sound Library Entry** and a **Select** button. To convert the library entry currently selected in the Sound Editor, leave the **Current Sound Library Entry** check box selected. To convert a different file, click the **Select** button and use the Select a Sound File to Convert dialog to choose a new sound file. Clicking the **Select** button will automatically deselect the **Current Sound Library Entry** check box.

- **Converted Sound** – The Converted Sound section contains options for converting the sound, and contains a New Converted Sound Location panel and a Converted Sound Characteristics panel.

  The New Converted Sound Location panel contains the **Same as Current Library Entry** and **New File** radio buttons. If **Same as Current Library Entry** is selected, the converted file will overwrite the original library entry. **New File** displays a dialog so you can save the converted sound as a new file in the Sound Support folder, leaving the original unaltered.

  The Converted Sound Characteristics panel lets you choose the **options** for the file that will be created. Leave the **Same** radio button selected for any characteristic you don't want changed. If all the **Same** radio buttons are selected, the sound file will simply be duplicated.

  a) **Type** – Designates the new sound's file format:

  - System 7 – Native Macintosh sound format.

- Wave – Native Windows sound format. Good for using with cross-platform applications or delivery from the Internet.
- AIFF – Good for using with cross-platform applications.

b) **Channels** – Selecting **Mono** will convert a stereo sound to mono. Mono sounds cannot be converted to stereo, as this is a setting that must be set at the time the sound is recorded.

c) **Sample Rate** – This option allows you to downsample sound files. You may choose to do this to save disk space or to make smaller file sizes for streaming from the Internet. You may only downsample sounds, as recording sounds with higher sampling rates (thus better quality) is an option that must be set at the time the sound is recorded.

d) **Sample Size** – This is another option for downsampling sound files. You may downsample a 16 bit sound down to 8 bits/sample. The higher the sampling size, the better the quality of the sound. As with the channels and sample rates, a higher sample size is a setting at the time the sound is recorded.

e) **Data Size** – This number is updated based on conversion selections and represents the data (file) size of the sound after converting

Creating a New Sound Entry

1. Open the Sound Library dialog by choosing **Sound Library** (⌘7) from the **Libraries** menu.
2. Click the **New** button.

   The Sound Editor dialog is opened.
3. Select whether you want this sound entry to be Standard Sound or Phonetic Speech.
4. Select whether you want the default for this image to be referenced from an External file or from the Internal Library. Note that you can override this option when you build your application.
5. Click the **Select** button.

The Get File dialog is displayed, from which you can navigate to the sound you want to reference. Click **OK** to select it the sound file.

If the sound (or text file, if Phonetic Speech is selected) is not in your Sound Support folder, a Support file dialog is presented. From this dialog, choose whether you want to create a copy of the sound or text file or move the original file to the Sound Support folder. When you find the file, select it and click **Open**. The sound file will be added to the Sound Support folder with the appropriate extension.

Note: do not change this extension if you are planning on cross-platform or Internet delivery of your application.

6. The sound's Characteristics appear in the Sound Editor dialog, including Type, Sample Rate, Sample Size, Data Size, Duration, and Channels. Click **Play** to preview the sound.
7. Set the sound's **Options**. See the above section on the Sound Editor Dialog for an explanation of these fields.
8. In the **Sound Library Entry Name** field you can let onViz default the name based on the sound (or text) file name. Deselect this option if you want a different name for your movie entry.
9. To place the sound's library entry in all the Sound dropdown menus**,** leave the **Show in Sound Drop-Downs** check box selected.
10. Click **OK** to close the Edit Sound window and return to the Sound Library.

    If you had selected Phonetic Speech as the sound type, the text file on disk will be updated with any changes you have made in the **Text to Speak** field.
11. Click **Close** to close the Sound Library.

Editing an Existing Sound

To edit an existing sound entry, select the sound name from the list on the left and click **Attrb** (Attributes), or simply double-click the sound entry in the library list.

The Edit Sound dialog is opened from which you can modify the attributes or select a different sound for this entry.

Converting a Sound File

1. Open the Sound Library dialog by choosing **Sound Library** (⌘7) from the **Libraries** menu.
2. Select the sound entry from the list on the left, and click **Convert**.

   The Sound Conversion dialog is opened. Note that the Convert option is also available on the Sound Editor dialog. After selecting the sound entry from the library, it is the default for the **Input Sound Location**. If you want to change the sound to be converted, click the **Select** button to select a different sound.

3. Select a **New Converted Sound Location**.

   Same as Current Library Entry will overwrite the old file with the new one. **New File** will create a new file with a new name, leaving the original unaltered.

4. Choose the **Converted Sound Characteristics** for the file that will be created. See the above section on the Sound Conversion Dialog for an explanation of these fields.

5. Click **Convert** or **Save**.

   a) **Convert** – If **New File** is selected for the new sound file location, a Save As dialog is opened, allowing you to name and save the new file in the Sound Support folder.

   b) **Save** – The file is converted and saved to the designated location, and you are returned to the Sound Conversion dialog. You can choose more files for conversion by clicking the **Select** button.

   A warning dialog will be displayed informing you that this action cannot be undone. To convert without the warning dialog, hold down the option key while clicking **Convert**.

6. Click **Done** to close the Sound Conversion dialog and return to the Sound Library.

7. Click **Close** to close the Sound Library.

Chapter 5

Flow States

Covered in this Chapter:

- Using Flow States
- Start State
- Junction State
- Bridge State
- Return State
- Menu State
- Stop State

Using Flow States

Flow States work together with branching strategies, Paths, and Groups in determining an application's flow of activity. Flow States are OnViz' traffic cops. Not only do they designate where a project starts and stops, but they also give directions. Need to route your project's flow of activity to a State way across your Application Map? You could use a really long Path, or you could use a type of Flow State, called a Bridge, to simply tell the flow where to go next. Flow States are great tools for breaking away from purely linear flows and into more varied and interesting projects.

Because they are used primarily for routing purposes, Flow States have no Presentation windows. They do, however, have InfoCenter windows, which vary from State to State. To access the InfoCenter for any of the Flow States, double-click anywhere on its State icon, or choose **InfoCenter** (⌘I) from the **Map** menu.

For Starts, Stops, and Returns, the InfoCenter consists of only a **State Name** field. Since it is possible to change a State's name from the Application Map, without ever entering its InfoCenter, you never need to actually open the InfoCenter for these States.

To change a State's name from the Application Map window, click once on its name, wait for the Name field to become a black rectangle with a red outline, then type the new name. Pressing Enter or Return, or clicking anywhere outside the Name field, will make the new name take effect.

Bridges and Menus have more detailed InfoCenters that determine their use, and these features are described in the following section.

Types of Flow States

There are five different types of Flow States:

- Start
- Junction
- Bridge
- Return
- Menu
- Stop

Starts and Stops designate where an application or Group's flow of activity begins and ends (for more information on Groups, see chapter 12). Bridges and Returns are similar to Paths, in that they help route the project's flow of activity. Unlike Paths, however, they can route the flow to any named State at any level of the project, as well as to other OnViz documents or other applications on your user's computer. Menus function like Bridges, with the additional feature of displaying custom menu items from which the user can choose to alter the flow of activity. For instance, a custom menu item could be added that would allow your audience to view a glossary or help screen, control the sound volume, or go to different modules within the application.

The Start State

Used in both the top-level Application Map and in Groups, Starts designate where your project's flow of activity begins. While Starts are not necessary with many of the branching strategies, they must be used with the Freeform branching strategy (for more information on Branching Strategies, see chapter 3, the Application Map). If a freeform Application Map or Group Map contains more than one Start, the flow of activity will begin with the first one placed on the Map.

Starts have no Presentation window, and their InfoCenters are only used to change their **State Name**. Because it is possible to change a State's name from the Application Map, without ever entering its InfoCenter, you never need to actually open the Start InfoCenter.

The Junction State

The Junction State is a place holder on the application map that gives you a location to connect several paths and use Conditional Paths out of the state to determine application flow. This state type contains an InfoCenter, but no Presentation.

The Bridge State

Like Paths, Bridges route your project's flow of activity from one State to another. Unlike Paths, however, Bridges can be used to connect a State within a Group to one outside that Group. Bridges can also be

used to route flow to another OnViz document, or to an external application.

Bridges have no Presentation window, but they are one of the two Flow States that have an InfoCenter (Figure 5.1).

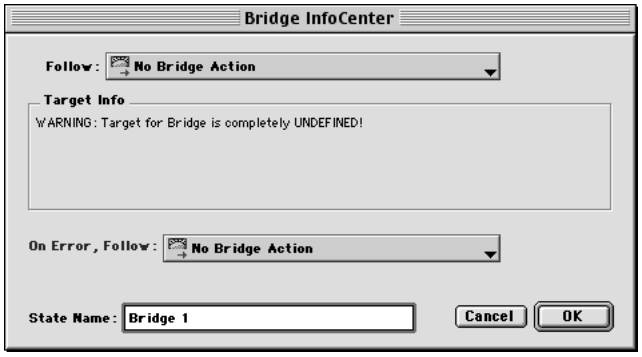


Figure 5.1: The Bridge InfoCenter.

Bridge InfoCenter

The Bridge InfoCenter is used to designate the Bridge's destination, or Target. Its options are as follows:

Follow: – allows you to create a new Bridge Target, edit an existing Bridge Target, or select an existing Bridge Target from the dropdown list. See below for instructions on creating a new Bridge or editing an existing one.

Target Info – describes the Bridge's Target, including its State type, State name, and location (in the current OnViz document or in a different one.)

On Error, Follow: – there may be instances in which a Bridge no longer works, possibly because a Target State has been renamed, a Target supporting document no longer exists, or a Target application cannot be found. To prevent your audience from hitting a dead end, you can designate a backup State to be used in the event of an error. This field operates exactly like the Follow: field. See below for instructions on creating a new Bridge or editing an existing one.

Best Practice

Always designate an alternative Bridge that can be used in case of error. One option is to create one or more error messages notifying your audience that an error has occurred, and letting them decide to either skip to the next section or to quit. You could even give your audience the option to send you email

Creating a New Bridge or Editing an Existing Bridge

1. From the **Follow:** or **On Error, Follow:** dropdown lists, select either **New Bridge** or **Edit Bridge**.

   The Setup Bridge Target screen appears, with a **Bridge Type:** dropdown list that defaults to the **No Target Defined** option (Figure 5.2). All options contain the **Bridge Library Entry Name** field, for naming your new Bridge Target, and the **Show Entry in Bridge Drop-Downs** checkbox, to make this Target available in the dropdown list for other Bridges.

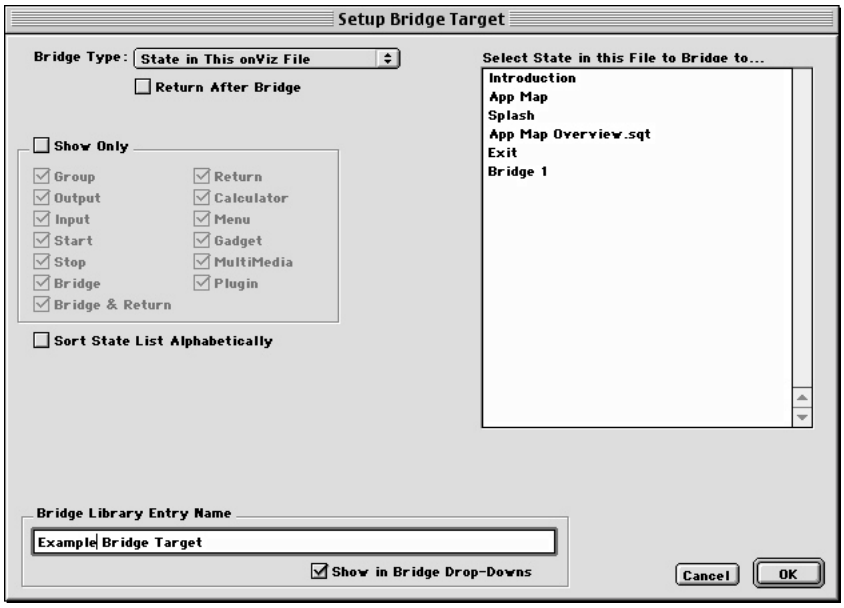


Figure 5.2: The Setup Bridge Target window.

2. Choose an option from the **Bridge Type:** dropdown menu (Figure 5.3).

   The Setup Bridge Target window's fields will change to reflect your choice. Listed below is each choice and its associated fields:

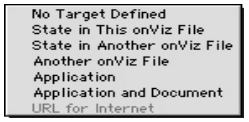


*Figure 5.3: The **Bridge Type** dropdown menu.*

   - **State in This OnViz File**
     - Return After Bridge – Routes flow back to the Bridge's Exit after the Target State finishes running. The Target must either have its **Automatic Return** box checked (in its InfoCenter) or have a Return attached to its Exit Point.
     - Select State in this File to Bridge to – Lists the States present in the current OnViz document from which you can choose a Bridge Target.
     - Show Only – Limits the States present in the adjacent list to only the types of States checked. This option is helpful if you have a large number of States in your project.
     - Sort State List Alphabetically – Presents the States in the adjacent list alphabetically according to their Names. If this box is not checked, States in the list are grouped alphabetically according to State type.
   - **State in Another OnViz File**
     - Return After Bridge – Routes flow back to the Bridge's Exit after the Target State finishes running. The Target must either have its **Automatic Return** box checked (in its InfoCenter) or have a Return attached to its Exit Point.
     - OnViz File Name – Clicking **Select** brings up a dialog that allows you to navigate to the desired OnViz file, whose name is then filled in to this field.
     - State Library Entry Name in File Bridging To – Type the name of the Target State from the external document to which this Bridge will connect.
     - Override Application File's Folder (Path) – Designating a location in this field overrides the Support Document location set up in the Project Attributes menu.

For cross platform development, this location may be different on Macintosh and Windows.

- **Another OnViz File**
  - Return After Bridge – Routes flow back to the Bridge's Exit after the Target State finishes running. The Target must either have its **Automatic Return** box checked (in its InfoCenter) or have a Return attached to its Exit Point.
  - OnViz File Name – Clicking **Select** brings up a dialog that allows you to navigate to the desired OnViz file, whose name is then filled in to this field. Flow will always begin at the start of the file, rather than in a specified State, as in the **State in Another OnViz File** option.
  - Override Application File's Folder (Path) – Designating a location in this field overrides the Support Document location set up in the Project Attributes menu. For cross platform development, this location may be different on Macintosh and Windows.
- **Application**
  - Return After Bridge – Select this option to keep your OnViz application running after opening the target application. If this option is not selected, your OnViz application will exit as if it had encountered a Stop.
  - Application File Name – Clicking **Select** brings up a dialog that allows you to navigate to the desired application, whose name is then filled in to this field.
  - Override Application File's Folder (Path) – Designating a location in this field overrides the Support Document location set up in the Project Attributes menu. For cross platform development, this location may be different on Macintosh and Windows.
- **Application and Document**
  - Return After Bridge – Select this option to keep your OnViz application running after opening the target application. If this option is not selected, your OnViz application will exit as if it had encountered a Stop.
  - Application File Name – Clicking **Select** brings up a dialog that allows you to navigate to the desired application, whose name is then filled in to this field.
  - Override Support Folder (Path) – Designating a location in this field overrides the Support Document location set up in the Project Attributes menu. For cross platform development, this location may be different on Macintosh and Windows.
  - Document File Name – Clicking **Select** brings up a dialog that allows you to navigate to the desired file, whose name is then filled in to this field.
  - Override Document Folder (Path) – Designating a location in this field overrides the Support Document location set up in the Project Attributes menu. For cross platform development, this location may be different on Macintosh and Windows.

3. Make sure that you have given the Bridge Target a meaningful **Bridge Library Entry Name**, so if you need to use it again you will know where it goes.

   Click **OK** to close the Setup Bridge Target window and return to the Bridge Info-Center.

4. Click **OK** to close the Bridge InfoCenter and return to the Application Map. Your Bridge icon now displays its target's Bridge Library Entry Name.

The Return State

Returns are attached to the Exit Point of a Bridge's Target State, and route your project's flow of activity from the Target back to the Bridge's Exit Point. Notice that the Return State icon has no Exit Point; this is because Returns always flows back to a Bridge. This same type of flow can be replicated, without the need of a Return, by checking the **Automatic Return** box in the Target State's InfoCenter.

Returns are only used when bridging to a State within the same OnViz document. When bridging to an external OnViz document, encountering a Stop in the external document will cause flow to return the Exit of the originating Bridge.

Returns have no Presentation window, and their InfoCenters are only used to change their **State Name**. Because it is possible to change a State's name from the Application Map, without ever entering its InfoCenter, you never need to actually open the Return InfoCenter.

The Menu State

(NOTE: Menu States are not operational in onViz Beta).

Menu States add a custom menu to your project's menu bar that will allow your audience to Bridge to another part of your project, such as a help screen, a glossary, or an index. Traditional **File** and **Edit** menu functions can also be replicated or replaced in your custom menus.

If you are using a custom menu, make sure that the Application Attribute's **Hide Menu Bar on Startup** option is not selected. Also, if you choose to display a custom menu in an Output, make sure that the **Hide Menu Bar** option, in the Output InfoCenter, is not selected.

For more information on Application Attributes, see chapter 2. For more information on Output States, see chapter 6.

Menu States have no Presentation window, but they are one of the two Flow States that have an InfoCenter (Figure 5.4).

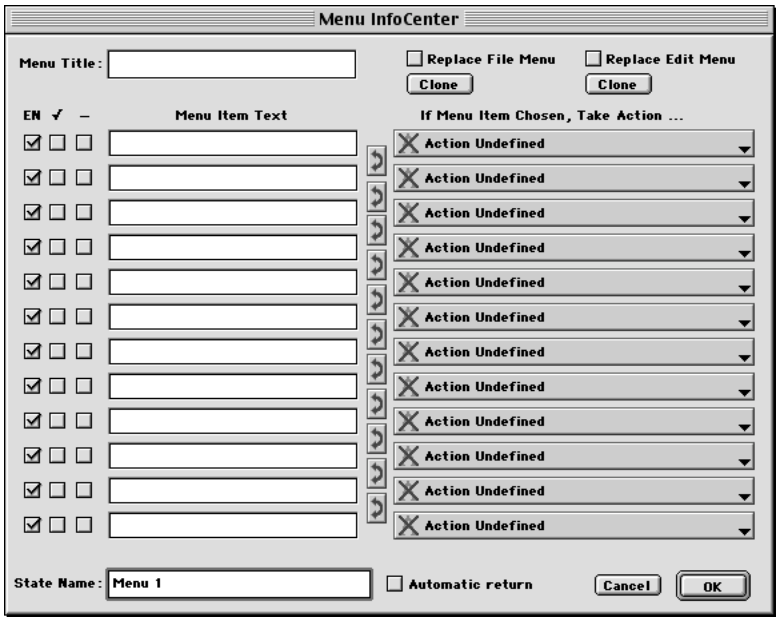


Figure 5.4: The Menu InfoCenter.

Menu InfoCenter

The Menu State's InfoCenter is used to choose the items that will be in your custom menu. Its options are as follows:

Menu Title – Your custom menu's name, which will appear in your application's menu bar.

Replace File Menu – This option creates a custom menu in place of the File menu. The Clone button fills in your custom menu with the functions normally found in the File menu. These functions can then edited, deleted, or replaced as desired. Particularly useful for multilingual applications, this feature provides the ability to maintain the functionality of File menu items, but with the names of your choosing.

Replace Edit Menu – This option will create a custom menu in place of the Edit menu. The Clone button fills in your custom menu with the functions normally found in the Edit menu. These functions can then edited, deleted, or replaced as desired. Particularly useful for multilingual applications, this feature provides the ability to maintain the functionality of Edit menu items, but with the names of your choosing.

EN – Enables the corresponding menu item. Removing the check mark disables the menu item, displaying it grayed out (and not selectable) in your custom menu.

✓– Places a check mark next to the corresponding menu item in your custom menu. Can be used as a navigation aid to show your audience the option currently selected in your custom menu. For example, if you have a custom menu to change the computer volume, the check mark might designate the current volume.

— – Places a separator at this point in your custom menu.

Menu Item Text – Text typed in this field will be displayed as items in your Custom Menu.

Automatic Return – If the Menu State is used as a Bridge's Target, checking this option will return flow back to the Bridge's Exit Point, without needing to go through a Return.

Figure 5.5: A Menu State without Automatic Return (left), and with Automatic Return (right). Note that the Menu State with Automatic Return, marked by a curved arrow, has lost its Exit Point.

Notice that checking this box causes the Menu State icon to lose its Exit Point (Figure 5.5); this is because the Automatic Return option always routes flow back to a Bridge.

The Stop State

Stops ends your project's flow of activity. While Stops must always be present in a freeform branching strategy, they are often helpful in the other branching strategies as well, particularly when used in Groups. If a Stop is contained within a Group, it ends the flow and transfers it to a corresponding Exit Point.

Typically, a Group has only one Stop and one Exit Point. However, the Group InfoCenter gives you the option for designating up to ten Exit Points. If a Group has more than one Exit Point, it should also contain corresponding Stops. Stops display a number designating the order in which they were created, and the Exit Point to which they are linked. If a Stop State icon is grayed out, or dim, it means there are more Stops than there are Exit Points. If, while running, the project flow encounters a Stop with no corresponding Exit Point, a "Dead End Route Reached" error message is displayed. You can use the Group InfoCenter to add another Exit Point. For more information on Group States, see chapter 12.

Stops have no Presentation window, and their InfoCenter is only used to change their **State Name**. Because it is possible to change a State's name from the Application Map, without ever entering its InfoCenter, you never need to actually open the Stop State InfoCenter.

Chapter 6

Output States

Covered in this Chapter:

- Types of Output States
- Output State InfoCenter
- Output State Presentation
- Notable Output Features
- Special Output Command Keys
- Output State Variables

Using Output States

The Output State is onViz' way of presenting information to your audience. While the other States perform primarily "behind the scene" functions, such as routing flow, capturing mouse inputs, etc., the Output is the face you put on your project for your audience to see. This is where you incorporate graphics, including animations and transitions, sound (speech, digitized sound, or music), text, and digital movies into your project. Not only do the Outputs allow you to present information, they also provides access to the tools you need to create it.

From the Output Presentation window, you can access libraries containing the tools you'll use to create vector-based and pixel-based graphics and formatted text, and the settings you'll need for the playback of recorded sound clips, and digital movies.

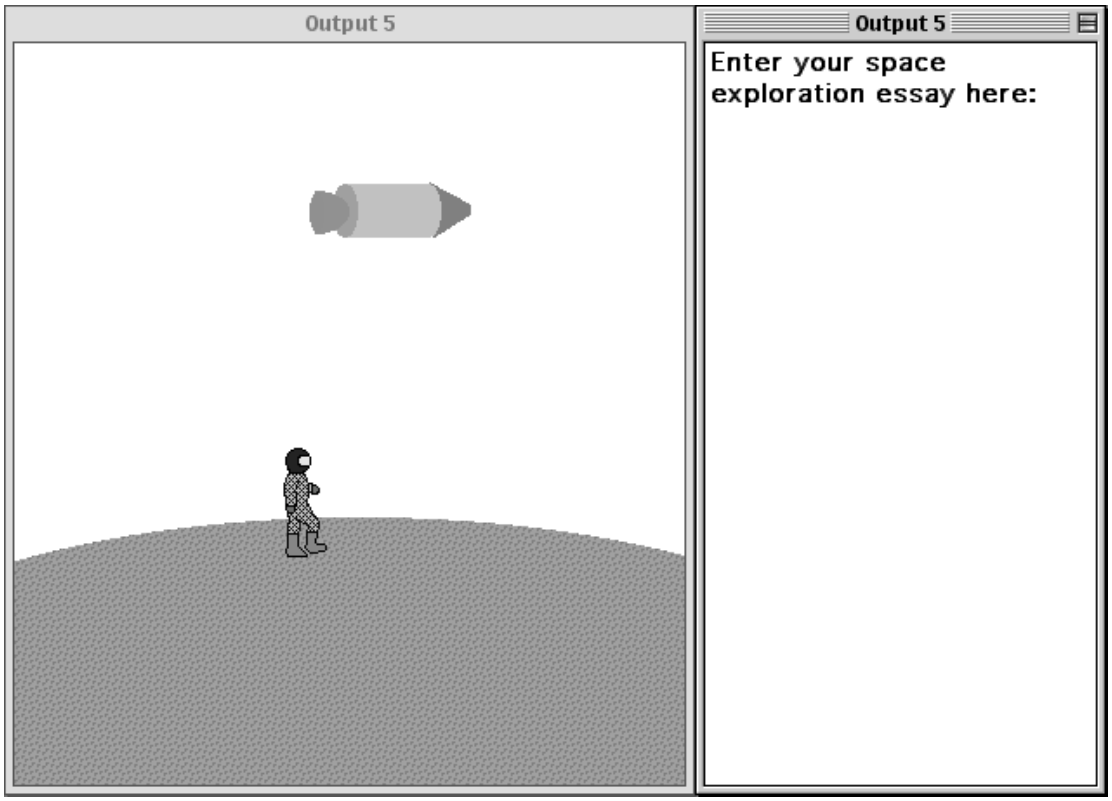


Figure 6.1: An Output viewed during runtime.

Types of Output States

There are four types of Output States, differentiated by the kinds of output they present:

- Design Window Visible
- Design and Text Visible
- Text Window Visible
- Hide Windows

All Outputs (with the exception of Hide Windows) support unique features that provide you with flexibility and variety when authoring your application. One such feature is variable substitution, with which your application can display items such as the user's name or score, the current date or time, or any of onViz' other built-in variables (Figure 6.2). For more information on variables, and how they can add personalization to your application, see chapter 13.

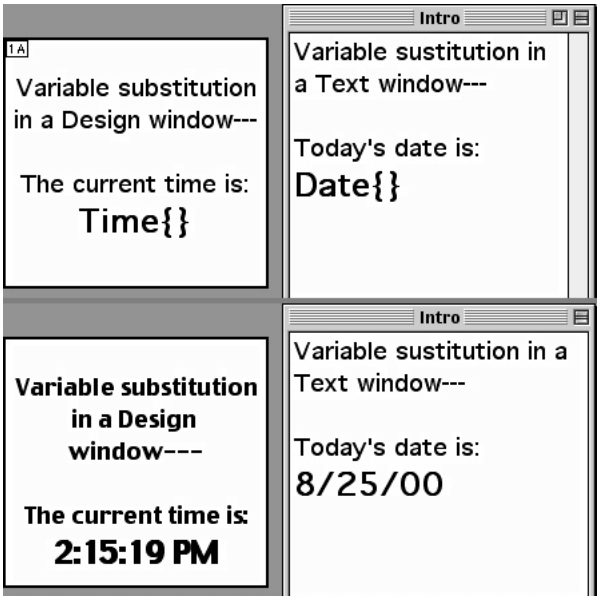


Figure 6.2: Variable substitution in Design and Text Windows. The time and date variables as they appear during authoring (top) and during runtime (bottom).

Another feature allows Outputs to have their contents automatically printed when the application flows through them. They can also have their contents displayed in several different types of windows, from standard dialog boxes to stylized 3-D windows. To add a more customized look and feel to your application, the Menu Bar can even be hidden.

Design Window Visible

The Design Window Visible Output, referred to in this documentation as simply the Design Output, uses a Design Window to display images stored in the Image Library or to play sounds and digital movies from the Sound and Movie Libraries. While static images can be created and displayed, onViz' built-in tools make it just as easy to develop sophisticated animations, simulations, and transitions. Features such as Tweening, Sprite Poses, and the Multi-frame Editor help automate the animation process, and are easily mastered.

By using the Sprite Attributes palette to set up Smart Sprites, Design Outputs can be used to create interactive buttons and controls for your application. Smart Sprites can respond to mouse actions, such as mouse-overs and mouse clicks, to send users to another frame in the animation or to another State in the application.

Design Outputs can be used in combination with Mouse Bays Input States and variables to create even higher levels of interactivity. Draggable sprites and mouse-click spots (mouse bays) allow the user to respond to questions graphically. Variables allow mouse movement to be tracked, and Smart Sprites to be automatically moved on the screen in response to user input.

Text Window Visible

The Text Window Visible Output, referred to in this documentation as simply the Text Output, is used for presenting text with no graphics. The Text Output displays text in a single window, and allows control over such formatting options as font face, style, size, color and alignment.

Text Windows can be set to be scrollable, which makes them ideal for lengthy text passages. Additionally, Text Windows can be set to be user editable, a feature that works well for essay type feedback from the person viewing your application.

You can use a Text Output to create a custom report for your application, or reports can be created using onViz' built-in report generator. Either way, you can choose to display the report to your user in the Text Output Window. You can also save the contents of the Text Window to a file (saved in the same location as the application report), or append its contents to the overall application report.

A special feature of text in a Text Output is that it can be configured to use MacinTalk to speak the text (available on Macintosh runtime only).

Design and Text Visible

The Design and Text Visible Output State, often referred to in this documentation as simply the Design and Text Output, combines the attributes of the Design State with those of the Text State. This output type consists of a two-window display: the Design Window and the Text Window. The Design Window contains graphics, and controls media components such as sound, movies, and text. The Text Window is a scrolling window that contains text only.

This type of Output State lets your application use a consistent interface and display of background graphics, while taking advantage of features like scrollable text, or a customized report which gets saved to the application report file.

When setting the contents of Design and Text Windows to print, two individual pages are printed, one with the contents of the Design Window and one with the contents of the Text Window.

Hide Windows

The Hide Windows Output State, or simply Hide Windows Output, removes from the screen all display windows until any designated synchronization has been met. If a Mat Color is specified in the Project Attributes Runtime dialog, the Hide Windows Output State displays a blank screen, filled in with the selected Mat Color. If no Mat Color is selected, the desktop of the user's computer is displayed. A Text Window is present during authoring, allowing you to add text to be spoken during runtime, if desired.

Output State InfoCenter

The Output State InfoCenter, accessed by double-clicking the rectangular area in the upper 1/3 of the State's icon, is used to set an Output's window attributes and runtime features (Figure 6.3). Features available in this dialog vary slightly depending on the type of Output selected. For example, if a Design State is chosen, some of the Text Window Attributes are grayed out.

Tip

The Output InfoCenter can also be accessed by selecting an Output and then choosing Output InfoCenter (⌘I) from the **Map** menu.

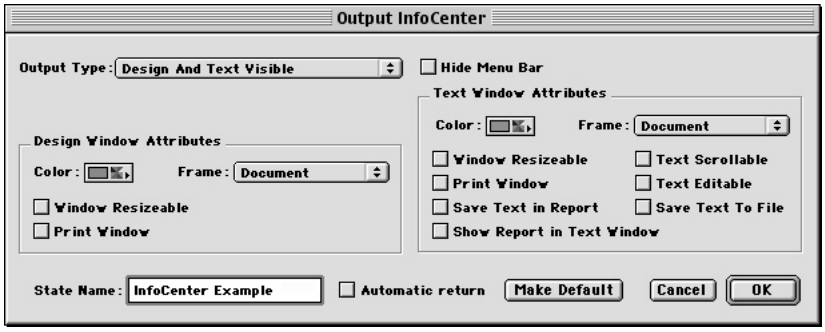


Figure 6.3: The Output State InfoCenter.

While a State's name can be changed from within its InfoCenter window, it can also be changed without leaving the Application Map. To change a State's name from the Application Map window, click once on the State's name, wait for the Name field to become a black rectangle with a red outline, then type the new name. Pressing Enter or Return, or clicking anywhere outside the Name field, makes the new name take effect.

Other features within the Output InfoCenter are as follows:

Output Type

Once placed, an Output type can be changed to any of the four types by choosing from this dropdown list. See the previous section for a description of the four Output types.

Hide Menu Bar

Hides the menu bar during runtime, so that your application takes up the entire screen. Even though the menu bar is hidden, command keys are still functional.

Note that if the Hide Menu Bar option is selected, any custom menus you've created for your application will also be hidden.

Windows note: when this option is selected for Windows playback, it does not automatically hide the task bar. And, even if the task bar is hidden, the display will still have a 4 pixel high horizontal strip where the task bar is.

Design Window Attributes:

- **Color**: Determines the Design Window color for this Output. Depending on options selected in the Project Attributes Runtime dialog, the Design Window will either be framed by a selected Mat Color, or will display the user's computer desktop behind the window. Clicking this button opens a palette from which to choose a color for the

Tip

If you choose a color not in the color palette and your application is run in 256 colors, onViz will use the closest color in the application color table for the window color.

Design Window. While in the color palette, you can choose Color Picker to pick a window color other than those in the 256-color display palette.

- **Frame**: Determines the border style for your Design Window. All frames, with the exception of Full Screen, can be sized and positioned from within the Output Presentation. If the user's monitor is larger than the target screen size, the display area outside the frame will be filled with the Mat Color designated in the Project Attributes. There are six types of frames (Figure 6.4):

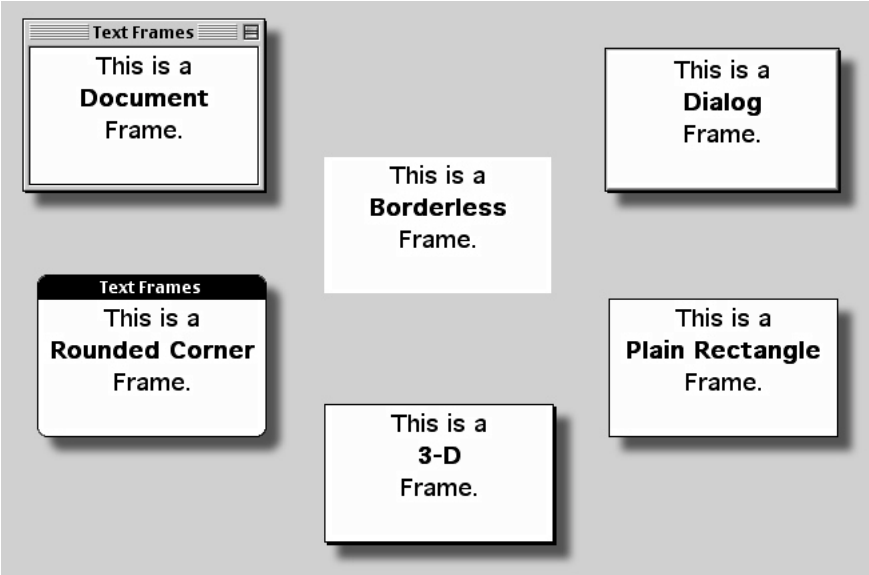


Figure 6.4: The six styles of Design Window frames.

  - Document - Rectangular border with a title bar and slight three-dimensional effect. In your development, keep in mind that this type of frame displays the State's name in the title bar. Therefore, you will want to give the state a name that is meaningful to your audience.
  - Dialogue - Rectangular border embellished with small inside border.
  - Plain Rectangle - Rectangular border with no additional embellishments.
  - 3-D - Rectangular border with a three-dimensional drop shadow.
  - Rounded Corner - Rectangular border with a black title bar and rounded corners. In your development, keep in mind that this type of frame displays the State's name in the title bar. Therefore, you will want to give the state a name that is meaningful to your audience.
  - Full Screen - A Full Screen frame fills the entire screen with the Design Window. If the user's monitor is larger than the target screen size, the frame is visible by the difference between the Design Window's color and the Application Attribute's Mat Color. If the Design Window's color is the same as the Application Attribute's Mat Color, the frame is invisible.

To reposition an Output Window set to Document or Rounded Corner, you can just drag it's title bar. To resize just drag the grow box in the lower right corner.

For Output Windows set to Dialog, Plain Rectangle, 3-D or Borderless, choose 'Window Move and Resize Mode from the Edit menu. This will put a locator handle in the upper left corner of the window with which you can reposition the window and a grow handle in the lower right corner to resize (Figure 6.5).

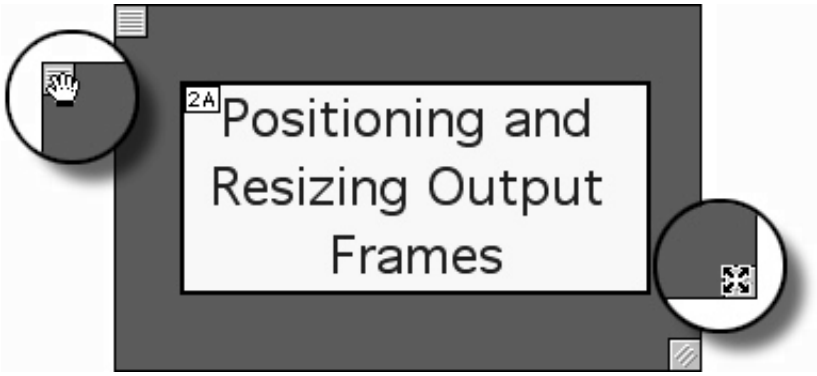


Figure 6.5: Positioning and resizing an Output frame. Note the cursor change: the cursor changes into a hand when it passes over the locator handle, and into a four-sided arrow when it passes over the grow handle.

Output Windows snap to horizontal boundaries for speed of execution. You can hold down option key for exact positioning, however your performance may suffer on slower machines and particularly with movies shown in 256 colors.

- **Window Resizable**: Allows your users to resize the frame around your Design Window. The three resizable frame types are: Document, Plain Rectangle, and 3-D.
- **Print Window**: Upon exiting the State, causes the standard page setup and print option dialogs to be automatically launched, giving the user the option to print the Design Window.

Note: if the Output consists of multiple frames of animation, the last frame before the Output is exited is the one that is printed.

The Gadget State offers automatic printing options that can be set to print pages automatically, without requiring the user to open the Page Setup and Print Option dialogs. See chapter 11 for more information about the Print Options Gadget.

Text Window Attributes:

- **Color**: Determines the Text Window color for this Output. Click this button to access a palette from which to choose a color. While in the palette, choose Color Picker to pick a window color other than those in the 256 color display palette.
- **Frame**: Determines the border style for your Text Window. All frames can be sized and positioned from within the Output Presentation. Text Window frame options are the same as for the Design Window described and illustrated in Design Window Attributes (Figure 6.4).
  - Document - Rectangular border with a title bar and slight three-dimensional effect. The State's name is displayed in the title bar.

Tip

When formatting the font color for the Text Window, ensure that it is not the same color as the Text Window itself, or the text will be invisible.

If you pick a color not in the application's color palette, and your application is run in 256 colors, onViz will use the closest color in the application color table for the window color.

  - Dialogue - Rectangular border embellished with small inside border.
  - Plain Rectangle - Rectangular border with no additional embellishments.
  - 3-D - Rectangular border with a three-dimensional drop shadow.
  - Rounded Corner - Rectangular border with black title bar and rounded corners. In your development, keep in mind that this type of frame displays the State's name in the title bar. Therefore, you will want to give the state a name that is meaningful to your audience.
  - Borderless - Text is enclosed within an invisible rectangular border. If the Text Window's color is the same as the Project Attribute's Mat Color, the text will be displayed seamlessly with the rest of the window.

- **Window Resizable**: Allows your users to resize the frame around the Text Window. Only Text Windows with a Document frame are resizable.
- **Print Window**: Upon the flow's exit from the State, automatically launches the standard page setup and print option dialogs, giving the user the option to print the Text Window.

  The Gadget State offers automatic printing options that can be set to print pages automatically, without requiring the user to open the Page Setup and Print Option dialogs. See chapter 11 for more information about the Print Options Gadget.

Tip

If the Output consists of multiple frames of text, the last frame before the Output is exited is the one that is printed.

- **Save Text in Report**: Once the application has flowed through a State with this option selected, the contents of the Text Window will be appended to the Report file. This option, when used with the Text Editable option, works well for essay questions or user feedback, as it allows the user to type in the Text Window in a free-form manner; the user-entered text is then captured and appended to the Report file.

  Note that it is not necessary to display the Text Window to save its contents to a report. A "behind the scenes" custom report can be created that will save the data without the user ever seeing it. Creating a custom report can be particularly valuable, as it can be used to capture information not found in the standard Report file. Custom reports can be created by using variable substitution in the Text Window. For instance, by placing the Number of Visits and Sprite Hit variables in the text window, the report will capture the number of times a user visited a specific Output and how which sprites they clicked while they were there.

  By using a Report Options Gadget, the custom report can be made to completely replace the standard Report file, which is helpful if you are collecting user statistics for a database, and need the information collected in a specific order. If you do not choose to replace the standard Report file, the statistics may not always be collected in the same order, especially if your users take different paths through your application. For more information on the Report Options Gadget, see chapter 11, Gadget States.
- **Show Report in Text Window**: Automatically displays the current Report statistics, including any custom reports that have been designed. This information is displayed in the Text Window for your audience to see. Note that the report will not contain

statistics for Inputs or custom report data until the application has flowed through the Inputs and State(s) containing the custom report.

- **Text Scrollable**: Adds a scroll bar to the Text Window. The scroll bar will only be present if the window is too small to display all of its contents. If this option is not checked, the scroll bar will not be present, even if the Text Window cannot display all of its text. All Text Window frames with the exception of Dialog and 3-D support scrollable text.
- **Text Editable**: Allows the user to type into the Text Window, including the ability to edit any text originally in the window. This option allows you to gather answers to essay type questions as well as get feedback from your user. When using this option, you typically will also want to use the Save Text in Report option.
- **Save Text to File**: Upon the flow's exit from the State, the contents of the Text Window, including any text added by the user, is captured in a file and saved to disk. Note that this file is saved to the location designated in the Build Target dialog. Note that it is not necessary to display the Text Window to save the contents to a file. You can create a "behind the scenes" custom report that saves the contents of the Text Window to disk without the user ever seeing it.

Automatic Return:

This option functions the same as having the Output followed by a Return State. If the Output State is used as a Bridge's Target State, checking this option will automatically return flow back to the Bridge's Exit Point, without needing to add a Return State to your Application Map. See chapter 5, Flow States, to learn how to use Return States in your application.

Notice that by checking this box the Output State icon no longer displays an Exit Point; this is because the Automatic Return option always routes flow back to the Bridge State that called it, eliminating the need for a path from the Exit.

Make Default:

After setting your Output State's attributes, press this button to cause all subsequent Output State placed on the Application Map to have the same attributes. This option can save a lot of time in setting up State attributes for each new Output placed.

You can create several template type Outputs and make them the default at different points in your application. For example, you might create one default for all Design Outputs that you use and another for all Design and Text Outputs. Depending on which state type you want to use, simply open the State, make it the new default, and drag an Output from the Map Tools palette. The new State will have the same attributes as your default. Of course, if you want multiple States with the same attributes, you can also simply copy and paste the state. If you use the copy and paste method, remember to rename each pasted State with a unique name.

Output State Presentation

The Output Presentation window is where you use media assets to compose frame-based animations and simulations that your audience will see while the application runs. If a Text Output is selected, a Text Window is available for entering long passages of text. This text can reside in a single scrollable window, or you can create display it within several frames of animation through which the user can navigate. Text Windows can also be used during runtime for users to answer essay questions, enter comments, etc.

If a Design Output is selected, a design area is available for displaying graphics and setting up frame-based animation sequences. A graphic, chosen from the Image Library, can be placed in the design area as a foreground image, background image, or a sprite. A sprite is a graphic that has been assigned a specific set of attributes, such as number, pose, or mouse action. While foreground and background images are essentially static, sprites can be included in animations or dragged around the screen by the user. Smart Sprites can be created that react to mouse actions and route application flow.

Animation in onViz is frame-based, and is controlled by the Frame Control palette. Frame-based animation works in much the same way as frames in a movie reel, or like those little flip-card animations you made as a child. An image is placed in the first frame, and then moved incrementally in each subsequent frame. All of the frames are then viewed in succession, giving the illusion of movement over a specified period of time. The Frame Control palette is used to add and remove frames, to add multimedia elements to frames, and to adjust the timing and transitions between frames.

Transitions let objects blend or wipe in and out of the frames in your animation, rather than just suddenly appearing or disappearing from one frame to the next. One type of transition, particularly useful for animations, is Tweening. While animations can be created that change position of an object incrementally from frame to frame, giving the illusion of movement, many animations can take advantage of onViz' tweening feature. Tweening looks at the difference in an object's size and location between two adjacent frames, and at the set transition time, and calculates how the object should move to make a smooth transition from the first location and size to the next. Tweening can accomplish in two frames what would have taken several frames if an object was moved incrementally from frame to frame.

The Output Presentation window is accessed by double-clicking the rectangular area in the lower 2/3 of the Output State's icon.

Tip

The Output State Presentation window can also be accessed by selecting an Output State and then choosing Output Presentation (⌘ E) from the Map menu.

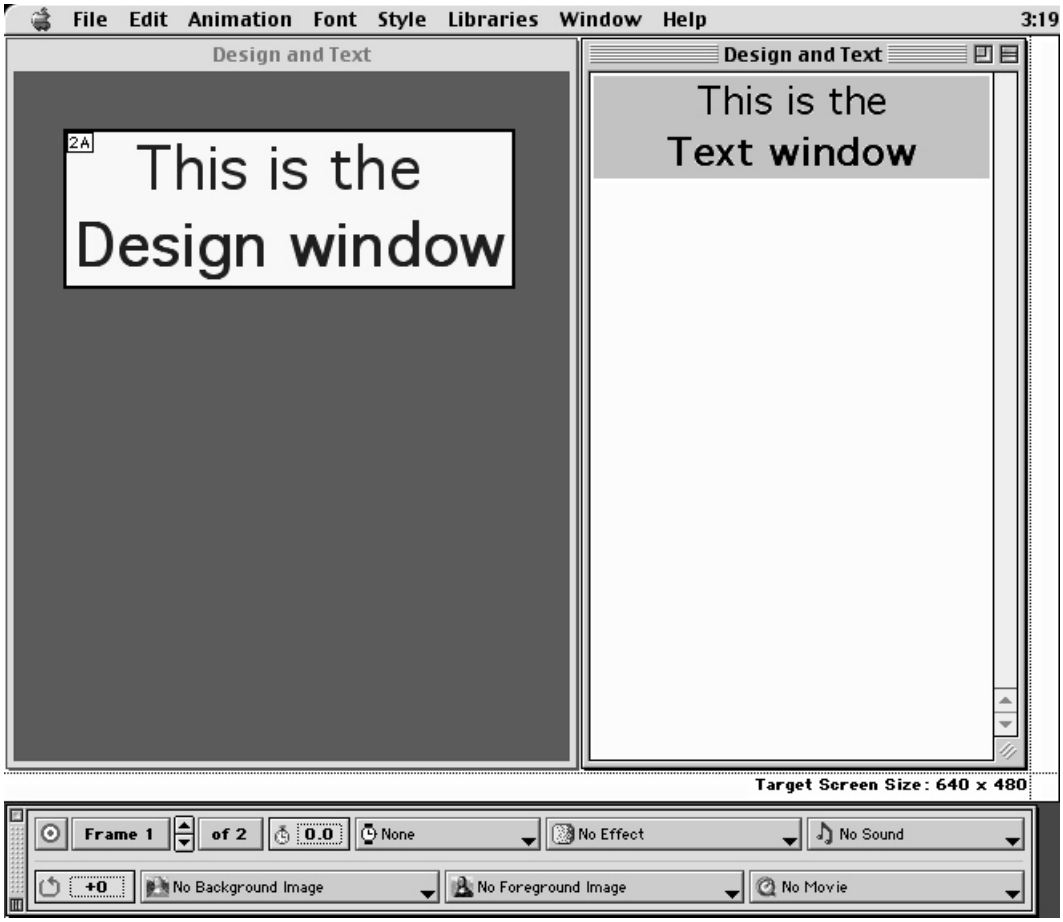


Figure 6.7: A Design and Text Output Presentation window. Note the Design Window on the left, the Text Window on the right, and the Frame Control palette at the bottom of the screen.

Window Elements

When you open an Output State's Presentation window, you will see a design area bounded by dotted lines (denoting target screen size) and the Frame Control palette at the bottom of the screen. Other elements of the Presentation window are determined by four settings:

1. **Type of Output State selected** - If a Design Output is selected, the Sprite Attributes palette will be displayed. If a Design and Text Output is selected, the Text Window will initially be displayed. Clicking in the Design Window brings the Sprite Attributes palette to the front. If a Text or No Window Output is selected, only a Text Window will be displayed.
2. **Window Attributes (Color and Frame)** - The color and frame type for the Design and Text Windows are determined by settings in the Output InfoCenter.
3. **Target Screen size** - When you open an Output State Presentation window, a dotted line denoting the Target Screen Size is visible. For Design Windows set to full screen, the design area will be the size of the target screen. Target screen size is set in the Project Setup dialogs or in the Project Attributes Runtime dialog.

Surrounding the target screen boundary is a white border that is not displayed at runtime. The bottom portion of this border provides useful information during authoring, such the target screen size, notification that the foreground and background images have been selected for movement, and the distances they have been moved.

If the Hide Menu Bar option is turned off, an additional dotted line (identified with "mb") will be displayed at the top of the screen designating the width of the menu bar. You can choose to show or hide the Target Screen bounds from the Edit menu.

4. **Full Screen Autocentering option** - This option is set in the Authoring Preferences window, with the Don't Autocenter Full Screen Designs while Authoring check box. If the Design Window Attributes' Frame setting is set to Full Screen, checking this option (the default) centers the design area on the screen (Figure 6.8). While displaying the Presentation using the Autocenter option is helpful, because the output is shown in a WYSIWYG fashion, this option uses more screen real estate. Deselecting this option during authoring will give you more room to work on the display without the Frame Control and Sprite Attributes palettes covering the design area. In the Full Screen frame style, the area that will hold the menu bar is defined by a dotted line with the letters "mb" at the top of the design area.

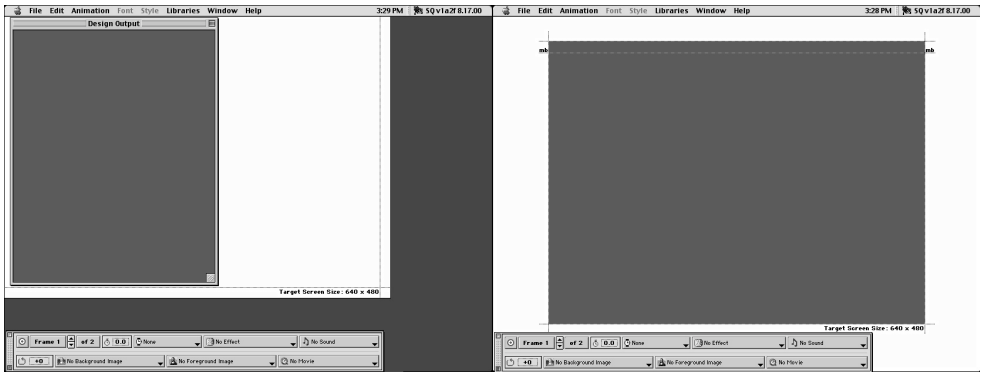


Figure 6.8: A Design Output Presentation window with Full Screen Autocentering off (left) and on (right).

When using Design and Text Outputs, clicking in the design area will bring it to the foreground, causing the Text Window to move into the background. If any portion of the Text Window is visible behind the design area, you can click on the Text Window to bring it back to the foreground. However, if the Design Window Attributes' Frame setting is set to Full Screen, or the Design area is hiding the Text Window, can bring the text window back to the foreground by selecting **Show Text Window** (⌘T) from the **Edit** menu.

Tip

To make the Text Window more accessible while authoring, move it outside the bounds of the screen size dotted lines, then move it to the desired location when you are finished adding content to the output.

Frame Control Palette

The Frame Control palette provides the tools to create animations and dynamic interactive interfaces for your project. With it, you can add and delete animation frames, set time delays, adjust synchronization, add target frames, cycle animation, designate multimedia elements, such as sound and QuickTime movies, for playback, and apply transitions, such as dissolves and fades to frame changes and/or the objects (sprites) on the frames.

The Frame Control palette works in conjunction with the Image Editor and the Sprite Attributes palette. Graphics are created in the Image Editor and are designated as either foreground/background images with the Frame Control palette or as sprites in the Sprite Attributes palette. Foregrounds and backgrounds are images that do not change within a specific Output. A sprite is a graphic that is added to specific frames of animation and can change size and/or position across a number of frames to display movement. The Frame Control Palette controls the timing and display transitions for the movement.

The Sprite Attributes palette supports mouse actions, such as mouse over and mouse down, allowing you to assign actions to sprites that make your project truly dynamic. For instance, you can use the Frame Control palette to create several frames of animation, designate one of those frames to be a target frame, and then use the Sprite Attributes palette to designate a sprite to, when clicked, take your user to the target frame. For more information on using the Frame Control palette to create animations, see chapter 8, Animation Support.

When you open any Output State's Presentation window, the Frame Control palette is visible at the bottom of the screen. It can be closed by clicking the Close Box in its upper-left corner. Once closed, it can be brought back to the foreground by choosing Frame Control Palette (⌘F) from the Window menu. The Frame Control palette has two collapse buttons: the horizontal collapse button hides the lower half of the palette, so the palette takes up less space on the display; the vertical collapse button causes the entire palette to retract into the drag bar. Double-clicking the drag bar also causes the palette to retract. While the palette is retracted, double-clicking the drag bar or clicking the vertical collapse button will cause the palette to open back up.

The Frame Control Palette contains the following tools:

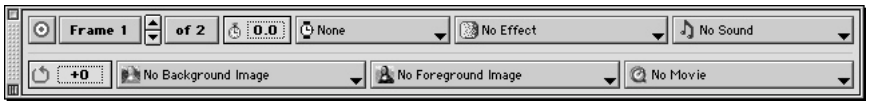


Figure 6.9: The Frame Control palette.

Target Frame - Designates the current frame as a Target Frame. Used with sprites that have been assigned mouse actions in the Sprite Attributes palette.

Goto Frame - Labeled with current frame number. Pressing this button opens the Goto Frame dialog, which allows you to go to the first frame, last frame, or a specific frame number (Figure 6.10).

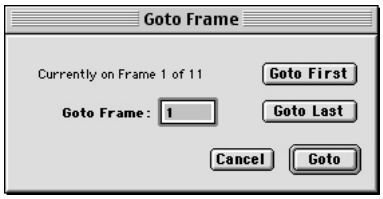


Figure 6.10: The Goto Frame dialog.

 **Up/Down Frame** - Takes you forward or backward through the animation a frame at a time. The Up arrow takes you forward one frame at a time. The Down arrow takes you backward one frame at a time.

 **Add, Insert, Delete Frames** - Labeled with the total number of frames in your animation (i.e. 1 of 1). Clicking this button opens the Add, Insert, Delete Frames dialog. By typing a number into the Enter a Number field and clicking the Add button, the specified number of frames are added to the end of the total number of frames (Figure 6.11).

By typing a number into the Enter a Number field and clicking the Insert button, the specified number of frames are inserted between the current frame and the next frames.

By typing a number into the Enter a Number field and clicking the Delete button, the specified number of frames will be deleted, starting with the frame you are currently on and counting forward. Because the frames are deleted forward, If you are on the last frame you can only delete one frame; if you are on the second-to-last frame, you can only delete two frames, etc. If you are on frame five of ten, and delete three frames, you will delete frames five, six, and seven.

Other options for adding/deleting frames of animation: choosing Add/Insert/Delete Frames (⌘I) from the Animation menu brings up the Add/Insert/Delete dialog, and Insert Single Frame (⇧⌘I) and Delete Current Frame (⌥⇧⌘I) can be chosen from the Animation menu.

Tip

Other options for navigating through frames of animation: Choose **Go To** from the **Animation** menu, then choose one of the options from the submenu: **Frame** (⌘ F) brings up the Goto Frame dialog, **First Frame** (⌘ ↑), **Last Frame** (⌘ ↓), **Next Frame** (⌘ →), **Previous Frame** (⌘ ←), **Next Target Frame** (⌘ →), and **Previous Target Frame** (⌘ ←).

 **Stopwatch** - Click in this edit field to enter a time delay for the current frame. If no other synchronization is set for this frame, the animation will pause for the designated time before continuing to the next frame. This field also determines the duration of a timed transition on any changing sprites or into the frame.

 **Synchronization** - Holds a frame until the manner of synchronization has been satisfied (Figure 6.12). Options include:

* Continue Button - Adds a Continue button in the lower left corner of the display. Position is relative to the Target Screen boundary. Animation will not proceed to the next frame until the user clicks the Continue button.
* Click Anywhere or Any Key - Animation will not proceed to the next frame until the user either clicks somewhere on the screen or presses a key on the keyboard.

Figure 6.12: The Synchronization dropdown menu.

* Wait for Sound - Animation will not proceed to the next frame until the sound associated with this frame has finished playing.
* Wait for Movie - Animation will not proceed to the next frame until the movie associated with this frame has finished playing.
* Wait for Smart Sprite - Animation is suspended until the user clicks a Smart Sprite. A Smart Sprite is a sprite that, when a mouse action is performed on it, will take the user to a specific frame in the animation, or to a different State. Mouse actions include mouse over, mouse down, and mouse up. Whenever a Smart Sprite is used, the Wait for Smart Sprite option is typically selected to prevent the animation from proceeding until the user clicks a sprite. Note: Click Anywhere synchronization can also be selected, but if the user clicks anywhere other than on a Smart Sprite, the animation will continue to the next frame.

Transitions - Determines the transition that will occur when this frame is entered. Transitions occur on elements that have changed between the previous frame and the current one. For example, if a single sprite is the only thing that has changed from one frame to the next, then the transition will only affect that sprite. If there is no change from one frame to the next, then no transition will be seen, even if one is chosen. A Transitions type is chosen from the lower section of the menu, and the duration of selected transition is chosen from the top section. If Timed Effect is chosen, a time delay must be entered into the Frame Control palette's Stopwatch field. Once a transition duration, such as Fast Effect or Timed Effect, is chosen, it will be remembered and applied by default to all subsequent transitions until a new duration is chosen.

For instance, if you select the Wipe Right transition with a Slow Affect, the next time you apply any transition to a frame it will default to Slow Affect. Simply choose a different duration, if so desired, and all subsequent transitions will default to the new duration.

S**ound** - Designates a sound that will be played when the current frame is displayed. From this menu, you can select a sound that already exists in the Sound Library, go to the Sound Library to edit an existing sound's attributes, or create a new sound reference. You can also choose to play any selected sound. The sounds listed at the bottom of this dropdown menu include two that have been designed specifically for this menu, System Alert and System Simple Beep, plus any sound references that you have created in the sound library and designated to be shown in the Sound dropdown menu. If you have created a sound reference but not included it in the dropdown menu, you can select Other to display a list of all sounds in the library.

Frame Cycling - Entering a negative number in this field causes the animation to jump back the specified number of frames (jump range), at which point it

starts moving forward again, creating a looping effect. This option works well for setting up an animation that will loop while the user performs some other activity, such as reading text. By entering a positive number, your animation can also jump forward the specified number of frames.

To allow the user to break out of the loop and continue on with the animation (or to exit the output), use a Smart Sprite set to go to a frame outside the jump range. This Smart Sprite will need to be placed on every frame within the jump range, as your users may decide at any point during the loop that they want to continue.

Any Synchronization placed on frames within the jump range will work normally, but if Synchronization has been applied to the frame designated to be a jump frame, it will be ignored so that the animation can jump to the designated frame.

Background Image/Foreground Image - Designates a background and foreground image for the design area. These images remain constant throughout all frames of animation. The image can be a new one, one that already exists in the Image Library, or one that is imported from an external file. You can also choose to edit an existing image's attributes. While the foreground image overlays the background image, any sprites in the design area will float above both.

Note that each Design Output can have different background and foreground images, which is useful if you want to create an overall look that remains constant throughout the application (i.e. a company logo banner as the background image), yet have each Output display a different foreground image (i.e. module headings).

Hold down the Command key and double-click in the design area to open the current background image in the Image Editor (use the Command and Shift keys for the foreground image). If there is no current background image, this action will open the Image Editor so you can create one. This new background is automatically named after the current Output State, followed by a .bg, as in OutputStateName.bg (.fg for the foreground image). After closing the Image Editor, the new background is in place, and is selected so it can be repositioned. Note the text in the information section of the design area stating that the "BG image selected" and giving its X and Y coordinates. If deselected, the background or foreground can be selected by choosing Select Background Image (⌘ ;) or Select Foreground Image (⌘ ') from the Select hierarchical menu item in the Edit menu.

Movie - Designates a movie that will be played when the current frame is displayed. From this menu, you can select a movie that already exists in the Movie Library, go to the Movie Library to edit an existing movie's attributes, or create a new movie reference. You can also choose to preview any selected movie. The movies listed at the bottom of this dropdown menu are movie references that you have created in the movie library and designated to be shown in the Movie dropdown menu. If you have created a movie reference but not included it in the dropdown menu, you can select Other to display a list of movies in the library.

While authoring, the Movie window will display the movie's Poster Frame, or, if no Poster Frame has been designated, it will display the first frame of the movie. The Movie window can be shown or hidden by toggling on/off Show Movie Window from the Edit menu.

Image Editor

onViz includes a full-featured Image Editor for creating both vector-based and pixel-based graphics. The Image Editor can be accessed in several different ways:

- Through the Image Library (see Creating a New Image Library Entry, below).
- By double-clicking a sprite in the Sprite Library or on a frame of animation.
- By command double-clicking the design area to create/edit a background image.
- By option command double clicking to create/edit a foreground image.
- By selecting New or Edit from the Sprite Selection dropdown list in the Sprite Library (see Sprite Attributes Palette, below, for more information on the Sprite Library).
- By selecting New Sprite or Edit Sprite from the Background Image and Foreground Image dropdown lists in the Frame Control palette.
- By creating a QuickSprite (see chapter 8, Animation Support, for more information on QuickSprites).

For additional information on creating, editing, and importing graphics, see chapter 7, Image Editor.

Creating a New Image Library Entry

1. Open the Image Library by choosing **Image Library** (⌘5) from the **Libraries** menu.
2. Click the **New** button, which opens the Image Attributes dialog.
3. Select whether you want the default for this image to be referenced from an External file or from the Internal Library. Note that you can override this option when you build your application.
4. To create a new image, give your new image a name in the **Image Library Entry Name** field. The name you type in this field will be the name of the image file in your image support folder.
5. To create or edit another image, reopen the Image Library or select from the Image Editor controls drop down menu.
   Otherwise, close the Image Editor by clicking the Close Box, or by choosing **Close Window** (⌘W) from the **File** menu.

   When you close the Image Editor, a file is in the Image Support folder with the same name as your Library Entry Name. This file name will be appended with either a .pct or .jpg extension, depending on whether this image is a PICT or jpeg image. Do not change this extension if you are planning cross platform or Internet delivery.

Sprite Attributes Palette

Note: this section is provided to give an overview of the Sprite Palette. Be sure to see Chapter 8 for an in-depth description of using this palette while creating animations and setting up Smart Sprites.

The Sprite Attributes palette is divided into two sections, the Sprite Library, on the left, and Mouse Actions, on the right (Figure 6.13).

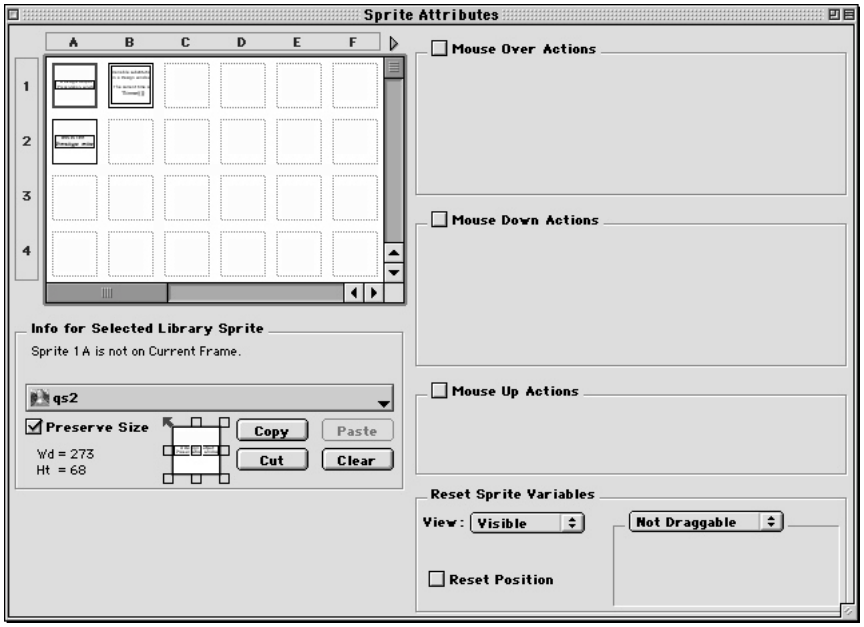


Figure 6.13: The Sprite Attributes palette. The left-hand side of the palette is the Sprite Library, and the right-hand side is the Mouse Actions section of the palette.

To create sprites, images are added to the Sprite Library and assigned attributes, such as sprite number and pose (Figure 6.14). With these attributes, they can be placed on individual frames of animation. Sprites are numbered from one to sixty-four, and each number can have up to sixteen poses, designated by the letters A-P. A pose is a variation of a given sprite. For instance, a sprite of a man walking might have a pose for each position his body goes through to take two complete steps - one with his left leg and one with his right. Each unique pose would be placed in a different sprite pose location. A sprite number with all of its poses is called a sprite family (Figure 6.15). Sprites are always referenced by their number and pose, attributes that are particularly important when using variables to manipulate sprites, or when branching the application flow based on the condition of a sprite.

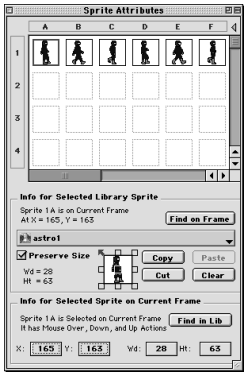


Figure 6.14: The Sprite Library. Sprite Poses 1A-1F make up a sprite family. Notice the Info section displaying information about sprite pose 1A.

Poses also work well to designate completed items on an on-screen menu. You might have one pose for an item not visited, another for an item which has been visited, but not complete, and a third for an item which is complete. In this case your sprite family will contain buttons in Pose A, Pose B, and Pose C. You will use a Calculator and the Sprite's Pose variable to determine which pose to display based on where your user has gone in the application. For more information on variables, see chapter 13, and for more information on Calculator States, see chapter 14.

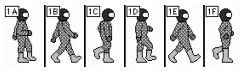


Figure 6.15: This sprite family is made up of all the poses one goes through when walking.

The Sprite Attributes palette also sets up on-screen controls, or Smart Sprites. A Smart Sprite is a sprite that, when a mouse action is performed on it, will show a different pose, play a sound, or take the user to a specific frame in the animation, or even to a different State. Mouse actions include mouse over, mouse down, and mouse up. Any of a sprite's poses can be made into a Smart Sprite. For more information on mouse actions, see chapter 8, Animation Support.

If you are not using mouse actions, or if you have already finished assigning them, you can conserve screen space by collapsing the Sprite Attributes palette so that only the Sprite Library section is showing. Click the triangle at the top-middle of the palette to hide/show the Mouse Actions section of the palette (Figure 6.16).

Figure 6.16: Collapsing the Sprite Attributes palette to show only the Sprite Library.

When a sprite is selected in the Sprite Library, details about that sprite are shown in the Info for Selected Library Sprite section (Figure 6.17). Information and options include:

- Sprite number and pose.
- If the sprite is on the current frame, and, if it is, its X and Y coordinates and the Find on Frame button.
- The sprite's width and height in pixels.
- The Sprite Selection dropdown menu.
- The Preserve Size check box.
- Copy, Cut, Paste, and Clear buttons.
- The Sprite Alignment Anchor.

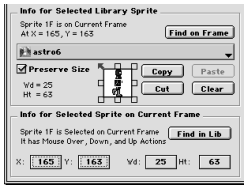


Figure 6.17: Selected Library Sprite information includes the sprite's current location and size, and controls for precisely positioning and resizing the sprite.

If the sprite is also selected on the current frame, the Info for Selected Sprite on Current Frame section appears (a sprite selected in the Sprite Library can be easily selected on the current frame by clicking the Find on Frame button). This Info section contains the following details and options:

- A list of the sprite's mouse actions.
- The Find in Lib button.
- X and Y coordinate buttons for precisely positioning the sprite.
- Width and Height buttons for resizing the sprite.

Resizing Sprites

After placing a sprite on a frame, there are two methods for resizing it, in the Image Editor or on the frame.

Resizing a sprite in the Image Editor - With the Preserve Size check box selected, a sprite can be resized in the Image Editor and its change will be reflected on the frame. First use the Sprite Alignment Anchor to designate an anchor point for the sprite, then double-click the sprite to open it inside the Image Editor. When the sprite is resized, its anchor point will remain at its original X and Y coordinates, even though the sprite's size on the frame has changed (Figure 6.18 and 6.19).

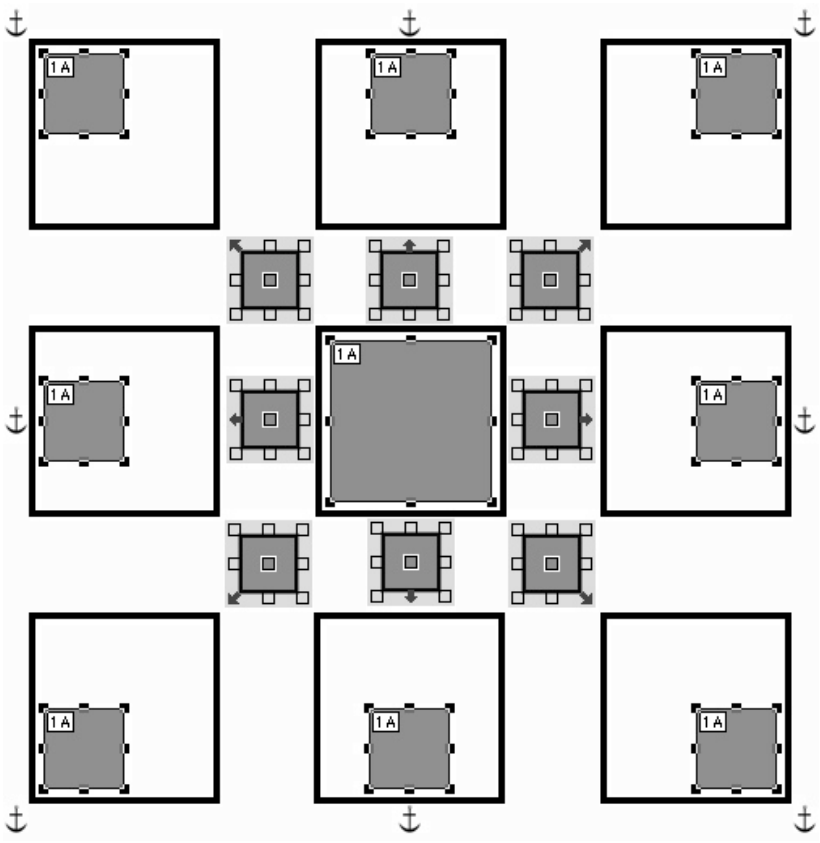


Figure 6.18: Resizing a sprite using the Image Editor and the Sprite Alignment Anchor. The sprite in the center is being reduced in size from 50 pixels to 25 pixels square. The surrounding images show how different anchor positions affect the sprite's location after reduction.

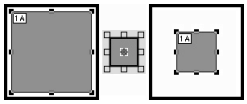


Figure 6.19: Using the center Sprite Alignment Anchor when resizing a sprite from 50 pixels (left) to 25 pixels (right).

Resizing a sprite on the frame - To change a sprite's size on the frame, first uncheck the Preserve Size check box. The sprite can then be resized with the Width and Height buttons in the Sprite Attributes palette, or by dragging the sprite's selection handles until it has been resized as desired (Figure 6.20).

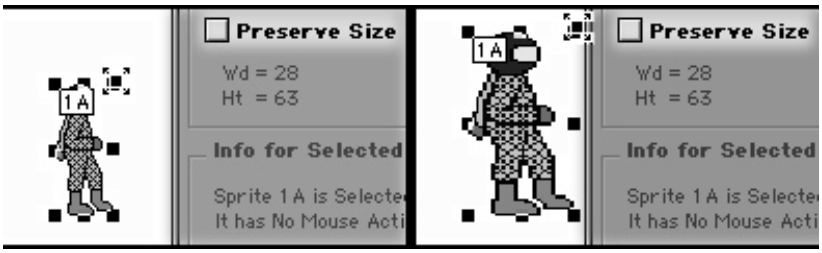


*Figure 6.20: Resizing a sprite on the frame. After deselecting the **Preserve Size** check box, a sprite can be resized by dragging its selection handles.*

Changing a sprite's size on the frame does not change the size of its base image. After changing a sprite's size on the frame, rechecking the Preserve Size check box will cause the sprite to revert to the size of its base image.

Menu Bar Commands

In addition to standard Macintosh functions, the menu bar for the Output State Presentation window provides support for the manipulation of animation and text, and varies slightly depending on the type of Output State selected. If Design Output is selected, no Text Window is available, so the menu bar commands that control font formatting, found in the Font and Style menus, are grayed out, or inactive. If the Text State is selected, no graphics can be added, so certain menu bar commands that control the manipulation of graphics and sprites, found in the Edit and Animation menus, are grayed out. If Design and Text Output is selected, the availability of these commands depends on which window is in the foreground, the Design Window or the Text Window.

Menus unique to the Output State Presentation are the Edit, Animation, Font, Style, and Window menus.

Edit Menu

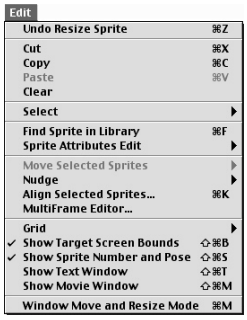


Figure 6.21: The Edit menu.

- **Undo** (⌘Z) - Reverses last command.
- **Cut** (⌘X) - Removes the selected information from the display and places it on the Clipboard.
- **Copy** (⌘C) - Copies the selected information from the display and places it on the Clipboard.
- **Paste** (⌘V) - Places information from the Clipboard onto the display
- **Clear** - Deletes the selected information.
- **Select** - Contains submenu options for selecting images in the design window. This feature can be particularly helpful if you have several sprites layered on top of each other and need to select the bottom sprite.
    * All Sprites (⌘A)
    * Next Sprite (⌘ `) - Sequentially selects sprites on the current frame of animation based on the sprite family reference number.
    * Previous Sprite (⇧⌘ `) - Sequentially selects sprites on the current frame of animation based on the sprite family reference number.
    * Foreground Image (⌘ ')
    * Background Image (⌘ ;)
    * De-Select All (⌘ .)

Sprites in the design area can also be selected with the mouse: simply click the desired sprite, or drag a marquee around it. Multiple sprites can be selected in the same way. Hold down the Shift key while clicking to add additional sprites to your selection, or drag a marquee around all the sprites you want to select. Holding down the Shift key while clicking an already selected Sprite will deselect it.

- **Find Sprite in Library** (⌘F) - Locates the selected Sprite's number and pose in the Sprite Library.

- **Move Selected Sprites** - Contains submenu options altering the layered order of the selected sprites:
  * To Front (⇧⌘ =)
  * Forward (⌘ =)
  * Backward (⌘ -)
  * To Back (⇧⌘ -)
- **Nudge** - Contains submenu options for moving the selected sprite:
  * Up (↑)
  * Down (↓)
  * Left (←)
  * Right (→)

If the selected sprite's Preserve Size attribute is not selected, clicking on one of the sprite's handles and using the nudge keyboard shortcuts results in the sprite being resized in the direction of the nudge.

- Align Selected Sprites (⌘K) - Opens a dialog for arranging the selected sprites, either relative to the grid or to each other.
- MultiFrame Editor - Opens a dialog for Adding/Moving, Removing, and Resizing sprites, and for changing a frame's timing and transition options. For more information on using the MultiFrame Editor, see chapter 8, Animation Support.
- Grid - Contains submenu options for displaying a grid:
  * Show Grid (⌘Y) - Toggles the grid on/off.
  * Snap Enabled - When dragging sprites or the background/foreground, they are snapped to the nearest gridlines.
  * Settings... - Opens a dialog for turning off and on the grid, selecting the visibility of the grid, and changing grid spacing (Figure 6.22).
- **Show Target Screen Bounds** (⇧⌘B) - Toggles on/off the dotted line marking the Target Screen Size as set in the Project Attributes runtime tab.
- **Show Sprite Number and Pose** (⇧⌘S) - Places a small square label on each sprite showing its number and pose.
- **Show Text Window** (⇧⌘T) - Brings the Text Window to the foreground. Not active for Text or Hide Window outputs, where the Text Window is always in the foreground The Text Window can be used with a Design Output to create a report that will not be seen by your user. By creating this report in the Text Window while displaying the Design Window, your report is being created and/or saved in the background. The Text Window can also be used with a Design Output to create phonetic text, while keeping the Text Window hidden. For more information, see Speaking hidden text, below. For more information about custom reports, see the Creating custom reports section at the end of this chapter.

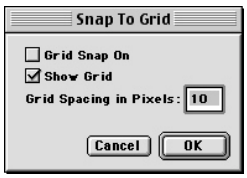


*Figure 6.22: After choosing **Grid Settings**, the Snap to Grid dialog appears.*

- **Show Movie Window** (⇧⌘M) - If a QuickTime movie is selected for this frame of animation, this option toggles off/on the QuickTime movie poster frame. Note that the movie poster frame only appears in the frame of animation on which the movie was selected, even though the movie may span several frames of animation at runtime.

Animation Menu

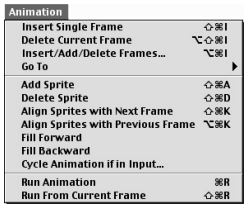


Figure 6.23: The Animation menu

- **Insert Single Frame** (⇧⌘I) - Inserts a frame between the current and next frame. If at the end of an animation sequence, inserts a frame after the current frame. Any unselected sprites on the frame are cloned with the frame. Any selected sprites on the frame will change to the next pose when inserting a frame.
- **Delete Current Frame** (⌥⇧⌘X) - Removes the current frame.
- Insert/Add/Delete Frames (⌥⌘I) - Opens the Add, Insert, Remove Frames dialog, which allows you to add, insert, or delete a specified number of frames.
- **Go To** - Contains submenu options for going to different frames in the animation:
  * Frame (⌘⇧F)
  * First Frame (⌘ ↑)
  * Last Frame (⌘ ↓)
  * Next Frame (⌘ →)
  * Previous Frame (⌘ ←)
  * Next Target Frame (⌥⌘ →)
  * Previous Target Frame (⌥⌘ ←)
- **Add Sprite** (⇧⌘A) – Places the sprite currently selected in the Sprite Library into the frame of animation. Note that only one pose from each sprite reference family can reside on a frame of animation at a time.
- **Delete Sprite** (⇧⌘D) – Removes the currently selected sprite from the design area.
- **Align Sprites with Next Frame** (⇧⌘K) – Moves the currently selected sprite(s) to the X and Y coordinates of the sprite(s) next frame in the animation.
- **Align Sprites with Previous Frame** (⌥⌘K) – Moves the currently selected sprite(s) to the X and Y coordinates of the sprite(s) the previous frame in the animation.
- **Fill Forward** - Examines all subsequent frames for instances of the selected sprite. If found, onViz performs a linear tween in size and position between current size and location and the subsequent size and location.
- **Fill Backward** - Examines all previous frames for instances of the selected sprite. If found, onViz performs a linear tween between current location and the previous location.
- **Cycle Animation if in Input** - If an Output State containing an animation is used as the background for an Input State, the animation will cycle from the last frame to the frame designated in this dialog until the Input State is exited.

- **Run Animation** (⌘R) - Closes all palettes and shows the animation from the first frame to the last in the format in which the user will view it.
- **Run From Current Frame** (⇧⌘R) - Closes all palettes and shows the animation from the current frame to the last as the user will view it.
- **Run Stop** (⌘ .) - Stops animation and returns focus to the frame which started the animation.
- **Run Stop and Show** (⌘ ,) - Stops animation with focus on the frame being played when the item was selected

Font Menu - (available while creating Text Windows)

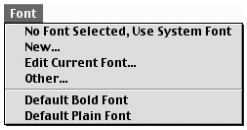


Figure 6.24: The Font menu.

- **No Font Selected, Use System Font** - onViz displays text with the computer's system font. Note that this is a bold font.
- **New...** - Opens the Font Attributes dialog, where you can add a new font library entry from the fonts installed on the computer. Also allows you to edit its attributes, such as its substitution font, library entry name, and whether or not it will appear in the font dropdown menus. For more information on adding fonts, see chapter 4.
- **Edit Current Font** - Opens the Font Attributes dialog, allowing you to edit the attributes of the currently selected font.
- **Other** - Includes onViz' application font, the computer's system font, and any fonts set up in the font library that have the attribute of Show Entry in Font Dropdowns.

  If you have an entry in the Font Library that is not being displayed in this list, the Show Entry in Font Dropdowns check box is probably deselected. To select this option, open the Font Library, select the font, and click the Attributes button. The Show Entry in Font Dropdowns check box is found at the bottom of the Font Attributes dialog.
- **Default Bold Font** - onViz displays text with the computer's system font, which is a bold font.
- **Default Plain Font** - onViz displays text with Geneva.

Style Menu - (available while creating Text Windows)

Contains font formatting options, including style, color, size, and alignment (Figure 6.25).

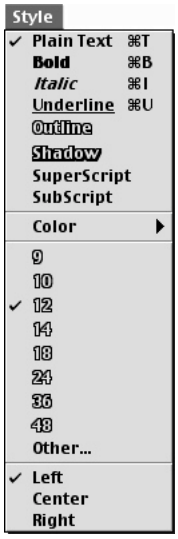


Figure 6.25: The Style menu.

Window Menu

- **Hide All Palettes & Menus** (⌘ Space) - Toggles on/off the Frame Control and Sprite Attributes palettes.
- **Frame Control Palette** (⌥⌘F) - Toggles the Frame Control palette on/off.
- **Sprite Attributes Palette** (⌥⌘S) - Toggles the Sprite Attributes palette on/off.
- **Drawing Tools Palette** (⌥⌘T) - Only available when the Image Editor is open. Toggles the Image Editor's Drawing Tools palette on/off.
- **Color, Pattern, Pen Palette** (⌥⌘P) - Only available when the Image Editor is open. Toggles the Image Editor's Color, Pattern, Pen palette on/off.

- **Edit Selected Sprite Image** (⌘ /) - Opens the currently selected sprite inside the Image Editor. Note: if the foreground or background image is selected, this menu item reads **Edit Foreground/Background Image**; the command key combination is the same.

Notable Output Features

Speaking Hidden Text

Even though the Design Output does not display text, it can still be configured to speak it. A Text Window can be opened, and text entered into it and configured to speak. When the application is run, the text will be spoken, while the Text Window will remain hidden.

1. Open a Design Output Presentation window, and choose **Show Text Window** (⇧⌘T) from the **Edit** menu.

   Type the text you want spoken into the text window.

2. Choose **New Sound** from the Frame Control palette's Sound dropdown menu.

   The Sound Attributes dialog is opened (Figure 6.26).

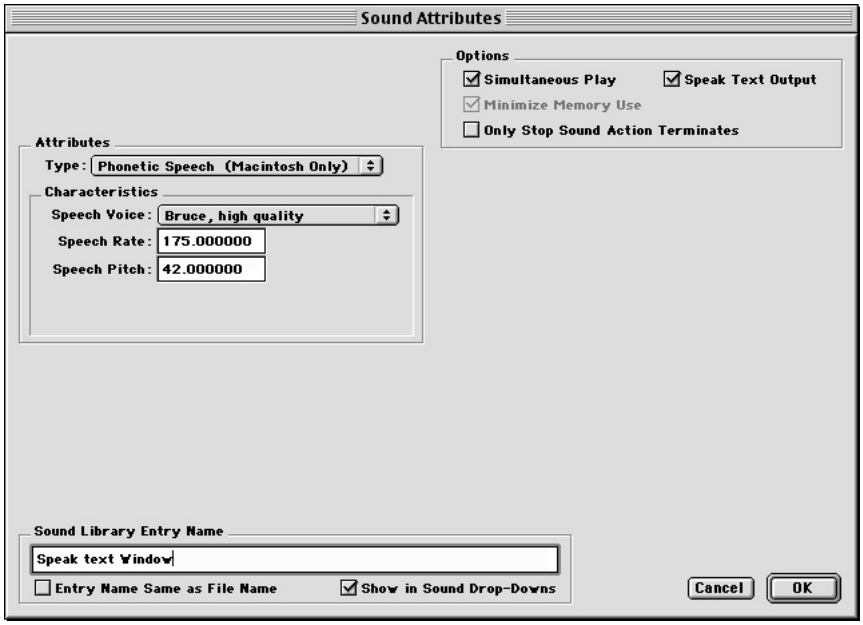


Figure 6.26: The Sound Attributes dialog.

3. From the Attribute's Type dropdown menu, choose **Phonetic Speech (Macintosh only)**.

4. From the Options section, choose **Speak Text Output**.

5. Deselect the **Entry Name Same as File Name** check box, and enter a new Sound Library Entry Name.

6. Click **OK**.

   The Sound Attributes dialog is closed, and your new sound name appears on the Sound dropdown menu. When the application is run, the text will be spoken, but the Text Window will not appear.

Creating Custom Reports

Special Command Key Operations

⌘ = **Command** ⌥ = **Option** ⇧ = **Shift**

Double-click sprite or Mouse Over/Down thumbnail	Bring up image in image editor
⌘ Double-click design area	Bring up background in image editor
⌘/⌥ Double-click design area	Bring up foreground in image editor
⌘/⌥ Double-click design area	Select background/foreground images
Escape/⌘ .	Deselect background/foreground images
⌥ Click on layered sprites	Select next sprite down in layer
⌥/⇧ click layered sprites	Select current and next sprite down in layer
Hold down option key (⌥) while changing frames with sprite(s) selected	Copies sprite to next/previous frame (if multiple pose - next/previous pose)
Hold down command and option keys (⌘⌥) while changing frames with sprite(s) selected	In the case of a multi-pose sprite, copies current sprite to next/previous frame rather than the next/previous pose
⌥ mouse down while moving sprite	Advances frame and copies sprite to next frame

Output State Variables

Read Only

- Total Frames (TotalFrames{State}) – Returns a value representing the total number of frames in the designated Output. This frame is read only because you cannot change the number of frames that were authored in the Output.

Write Only

- Reset Output (ResetOutput{State}) – Flowing the application through a Calculator that contains an equation setting the value of this variable equal to 1 resets all variables associated with designated output – including resetting sprite variables according to how the Output was authored.

Read/Write

- Design Window Visible (DesignWinVisible{State}) – Expected Values: 0 = invisible, 1 = visible. Reading this number provides information about the designated Design Window's visibility. Writing to this variable affects the visibility of the designated Design Window. (Note: this variable will show the designated Design Window even if originally authored as a Text or Hide Windows output)
- Text Window Visible (TextWinVisible{State}) – Expected Values: 0 = invisible, 1 = visible. Reading this number provides information about the designated Text Window's visibility. Writing to this variable affects the visibility of the designated Text Window. (Note: this variable will show the designated design window even if originally authored as a Design or No Window output)
- Number of Visits (OutputVisits{State}) – Reading this variable returns a value representing the total number of times the application has flowed through the designated Output. Writing to this variable sets the number of visits to the value assigned.
- Sprite Hit (SpriteHit{State}) – Reading this variable returns a value representing the sprite family number that was clicked by the user. Writing to this variable is typically used to reset it.

The following variables are replicated for each sprite family 1 – 64. *n* stands for the sprite family number.

- *n* Sprite's Pose (Sprite*n*Pose{State}) – Expected Values: -3 - 16. Reading this variable returns the pose currently being displayed by the designated sprite.
  Writing a value between 1 and 16 to this variable displays the specified pose of the designated sprite.
  Writing a value between 0 and –3 to a this variable causes the sprite to cycle through all of its poses – typically while waiting for user input. The more negative the number, the slower the cycling.
- *n* Sprite's Drag (Sprite*n*Drag{State}) – Expected Values: 0 – not draggable, 1 –drag any

direction, 2 – horizontal drag, 3 – vertical drag. Reading this variable gives information about the specified sprite's draggability. Writing to this variable makes specified sprite draggable or not and can constrain the drag to horizontal or vertical.

- *n* Sprite's Visible (Sprite*n*Drag{State}) – Expected Values: 0 = invisible; 1 = visible. Reading this variable gives information about the designated sprite's visibility. Writing to this variable affects the visibility of the specified sprite.
- *n* Sprite's Bay (Sprite*n*Bay{State}) – Expected Values: 0 – 10. This variable is typically used with draggable sprites. Reading its value returns a number representing the Mouse Bay region (if any) in which the specified sprite is positioned. Writing to this variable positions the center point of the designated sprite in the center of the designated Mouse Bay region.

Note: X/Y positions are based in pixels with 0,0 being the top left corner of the display. For example, the x/y position for lower left corner of a 640 x 480 display is 0,480 while the lower right corner is 640,480.

Chapter 7

The Image Editor

Covered in this Chapter:

- The Image Editor Window
- Using the Image Editor
- The Color, Pattern, Pen Palette
- The Drawing Tools Palette
- Menu Bar Items
- Creating, Editing, and Storing Images
- Masking Pixels
- Special Command Key Operations

The Image Editor Window

onViz' Image Editor works in conjunction with the Image Library to let you create and store new graphics, access and edit existing graphics, or to edit graphics imported from an external file. It is accessed through the Image Library by either creating a new image or editing an existing one, or through an Output Presentation window. For more information on the Image Library, see chapter 4, Libraries. For more information on creating and editing images through an Output Presentation window, see chapter 6, Output States.

The Image Editor window consists of a work area, the **Drawing Tools** palette, the **Color, Pattern, Pen** palette, and **Image Control** options. (Figure 7.1).

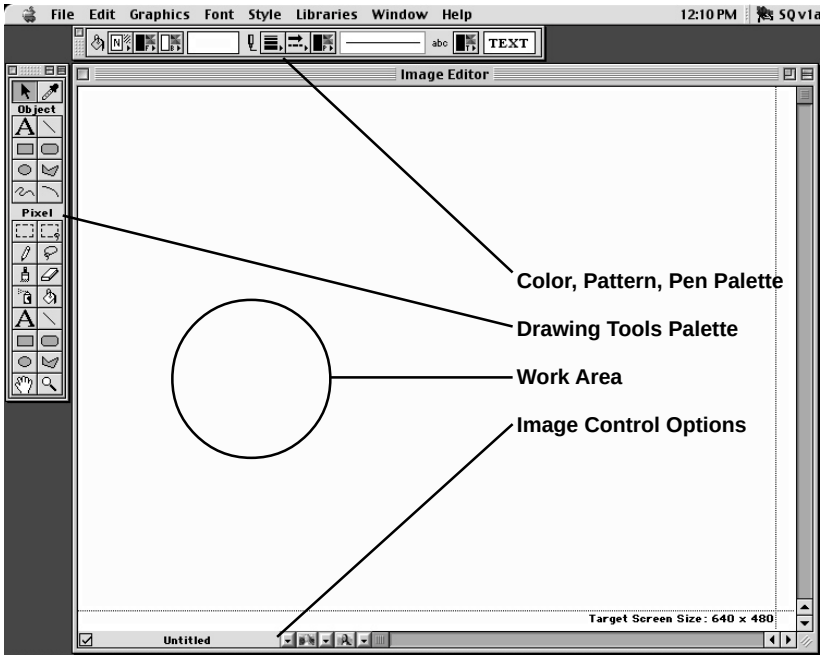


*Figure 7.1: The Image Editor window, showing the **Color, Pattern, Pen** palette at the top, the **Drawing Tools** palette at left, I**mage Control** options at the bottom, and the work area in the center.*

The Image Editor Work Area

The Image Editor work area is where images are created and edited. Bordering the work area is a dotted line marking your application's Target Screen Size. This dotted line can be toggled on/off by choosing **Show Target Screen Bounds** (⌘ ⇧ B) from the **Edit** menu. The work area can also display a grid to aid in sizing and placing your images. A submenu containing the **Grid** options is accessed from the **Edit** menu. For more information on Grid options, see the section on Menu Bar Items, below.

The bottom of the work area contains **Image Control** options to help you create and store your images without leaving the Image Editor window (Figure 7.2):

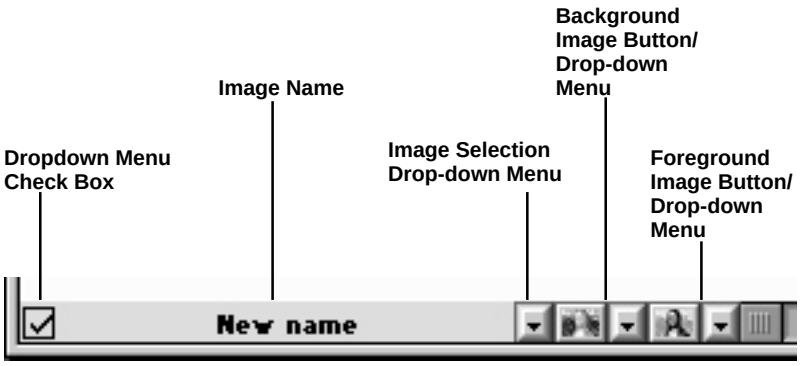


*Figure 7.2: **Image Control** options. The Background/Foreground buttons and drop-down menus allow you to place an image below the work area to serve as a reference while creating and editing images. The drop-down menus are for selecting the background/foreground images, while the buttons toggle the background/foreground image on and off.*

- **Dropdown Menu Check Box** – If this option is checked, the image name is conveniently placed in a list at the bottom of every Image dropdown menu (such as the **Image Selection Dropdown Menu**, below). These dropdown menus are a handy way to access your Image Library entries without having to actually open the library.

  When developing small projects, you may decide to use the **Drop-Down Menu** option for all of your images. In larger projects, however, where the list of entries could get very long, you may want to be more selective about which entries are included in the dropdown menus. Include the entries you'll use often, and not those that you'll only use once or twice. Your dropdown list will stay more manageable, and you'll still be able to access entries not in the list by opening the Image Library.
- **Image Name** – Click in this text field to give your new image a name, or to change the name of an existing image.
- **Image Selection Dropdown Menu** – Allows you to select an image to work on in the Image Editor, duplicate the current image, open the Image Attributes window for the current image, or open the Image Library to select an image not included in the Dropdown Menu. The images listed at the bottom of this menu, if any, are there because they were designated to be included in the Image Dropdown menus (see **Dropdown Menu Check box**, above).
- **Background Image Button/Dropdown Menu** – The dropdown menu allows you to designate a background image that will appear in the work area, while the button allows you to toggle the designated background image on/off. When on, the background image is not editable and will appear dim, so as not to interfere with the Image Editor's current image.

  Loading a background image in this manner is helpful if you are creating sprites that need to be precisely placed over a Design Output background, or if you want to compare your new image to an image already in the Image Library. The pre-existing image is loaded as a dimmed background in order to compare the two images while creating the new one. For more information on background images, see chapter 6, Output States.

- **Foreground Image Button/Dropdown Menu** – The dropdown menu allows you to designate a foreground image that will appear below the work area, while the button allows you to toggle the designated foreground image on/off. When on, the foreground image is not editable and will appear dim, so as not to interfere with the Image Editor's current image.

  As with the Background Image, loading a foreground image in this manner is helpful if you are creating sprites that need to be precisely placed over a Design Output foreground. Since an Output can have both a background and foreground image, this feature allows you to simulate the exact Output for which you are designing sprites. For more information on foreground images, see chapter 6, Output States.

Using the Image Editor

The Image Editor features two palettes: the **Drawing Tools** palette and the **Color, Pattern, Pen** palette. The **Color, Pattern, Pen** palette, as its name implies, lets you choose colors and line styles for your image. The Drawing Tools palette determines the shapes for your images, and is divided into two sets of tools: Object (or vector-based) tools and Pixel tools.

The Pixel tools palette creates images called bitmap images. Bitmap images are much like those you find on the Internet, scanned on a scanner, or captured with a digital camera, and are comprised of thousands of tiny square dots, or pixels. Each individual pixel is "painted" a certain color so that, collectively, they form a picture. Because the bitmap image stores this color information for each of its pixels, its file size is typically larger than Object graphics. You alter a bitmap image's color by "painting" over its pixels with another color. Scaling, or changing a bitmap image's size in onViz' Image Editor often results in a "stair-step effect," as its pixels are either stretched out over a larger area or compressed into a smaller area.

Tip

If you wish to change a bitmap image size, you will typically have better results by using a program such as GraphicConverter. A demonstration version of this shareware program is included on the onViz CD-ROM.

The images created by the Object tools palette are not "painted" images like a bitmap, but are instead shapes created by mathematical formulas, or vectors. Because these shapes, or objects, are created by formulas rather than colored pixels, they usually result in a smaller file size. In addition, it is easier to change an object graphic's appearance: simply select the object and change some attribute, such as fill pattern, color, or border (Figure 7.3). To change an object graphic's size, select it and drag one if its handles; its size will change with no visible distortion.

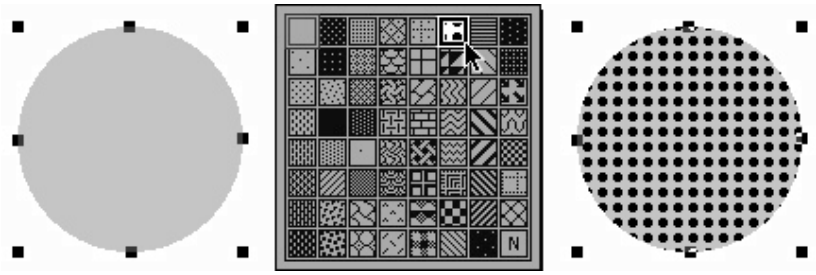


*Figure 7.3: Changing an object graphic's Fiill settings. The object is selected (left), a Fill pattern is chosen from the **Color, Pattern, Pen** palette (center), and and the graphic's appearance is immediately changed (right).*

The difference between object and pixel graphics is perhaps the most apparent in their treatment of text. Text created with the Object Text tool remains editable after it is has been created; you can change its font face, style, and color at any time. Text created with the Object Text tool also supports variable substitution, so that your text image can display, for example, the current time of day in the application. For more information on Variable Substitution, see chapter 13, Variables.

In contrast, once text has been created with the Pixel Text tool it cannot be edited. After its formatting is chosen and the text is typed, it becomes a part of the bitmap image, a collection of colored pixels in the shape of the text. One advantage to this type of text is that it can be used to display fonts that might not be installed on your target audience's computers.

For instance, you might create an object-based graphic with a particularly decorative font for your application. If the font does not exist on your user's computer, some other font will be

substituted, changing the graphic's appearance. If the graphic had been created with the Pixel Type tool instead, the font's appearance would have been preserved as a bitmap image, which does not require any fonts.

Graphics created in the Image Editor work area can overlay each other, and are "layered" in the order in which they are created. A graphic can be moved forward or backward relative to the other graphics by choosing a command from the **Move Selected Objects** submenu under the **Edit** menu (Figure 7.4).

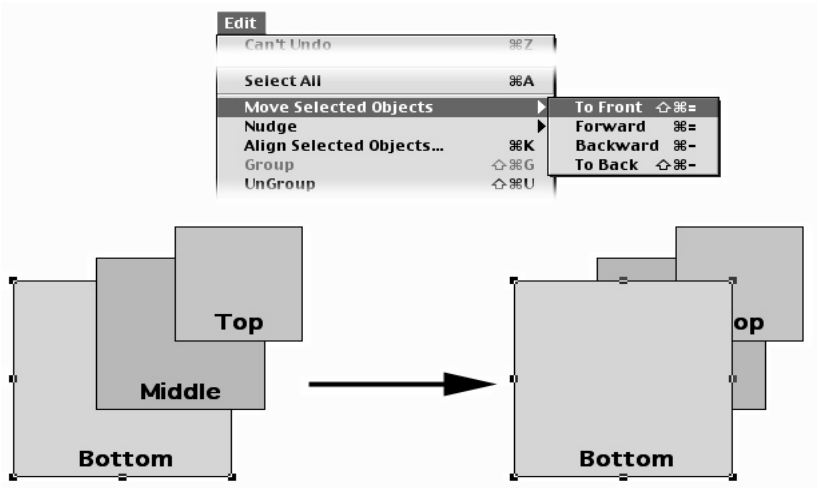


*Figure 7.4: Changing an object's layer order. In this example, the "Bottom" image is moved to the top, by selecting it (left) and choosing **Move Selected Objects To Front** from the **Edit** menu.*

The palettes, menus, and their individual settings and tools are covered in greater detail below.

The Color, Pattern, Pen Palette

The **Color, Pattern, Pen** palette contains Fill, Pen, and Text settings (Figure 7.5). These settings determine the image's attributes when the drawing tools are used.

*Figure 7.5: The **Color, Patter, Pen** palette.*

Settings found in the **Color, Pattern, Pen** palette include:

- **Fill Pattern** – Contains a variety of patterns, including solid colors and no fill, that can be used in your graphics. Choose the solid black square to display the solid color chosen to be your fill foreground. Choose the solid white square to display the solid color chosen to be your fill background. These patterns can be modified by choosing **Edit Patterns** from the **Graphics** menu.
- **Fill Foreground Color** – Opens a color palette from which you can choose a color for the foreground of your fill pattern. The color chosen here is used to color the areas of the selected pattern that are represented by black in the pattern palette.
- **Fill Background Color** – Opens a color palette from which you can choose a color for the background of your fill pattern. The color chosen here is used to color the areas of the selected pattern that are represented by white in the pattern palette.
- **Fill Display Window** – Displays a sample of what the object's fill will look like in terms of pattern, foreground color, background color, and border width and color. Border width and color are selected using the Pen Width and Color settings.
- **Pen Width** – Determines the thickness of the line drawn with the Line tool. Also determines the thickness of the border surrounding a graphic shape or a text object.
- **Pen Style** – Determines the style of line drawn by the Line tool. Note that only individual lines are given a style. The border surrounding a graphic shape or text object is always solid.
- **Pen Color** – Opens a color palette from which you choose a color for the line drawn with the Line tool. This color will also apply to the Pixel Tool palette's Pencil, Paint Brush, Spray Can, and Text tools. Also determines the border color, if any, displayed around a graphic shape or a text object.
- **Pen Display Window** – Displays a sample of the line the Pen tool will draw, in terms of width, style, and color.
- **Text Color** – Determines the color of text created by the Object Tools palette's Text tool.
- **Text Display Window** – Displays a sample of the text that will be created with the Object Tools palette's Text tool, including the text color and any fill patterns. The fill pattern/color for the text object is selected using the Fill Settings, while border width and color is selected using the Pen Width and Color settings.

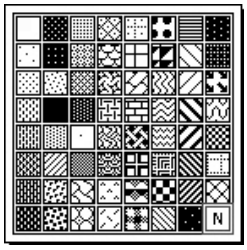


*Figure 7.6: The **Fill Pattern** menu.*

Note: Bitmap graphics are created with the current selection of Pattern, Fill, Color, and Pen. They cannot be altered once created. An object graphic is also created with the current palette selection, however, you can modify its attributes by clicking on the graphic and making selections from the **Color, Pattern, Pen** palette.

The Drawing Tools Palette

The **Drawing Tools** palette determines the shapes for your images, and is divided into two sets of tools: Object (or vector-based) tools and Pixel tools. The Pointer and Eye Dropper are used with both sets of tools (Figure 7.7).

Figure 7.7: The Drawing Tools palette, showing the Object Tools at the top and the Pixel Tools at the bottom. The Pointer and Eye Dropper work with both sets of tools.

Double-click a tool to maintain the use of that tool until another is selected.

Pointer – Selects, moves, and resizes objects and graphics in the design area. The Pointer tool works with both the Object tools and Pixel tools palettes.

Eye Dropper – *On object graphics:* picks up the color, pattern, and pen attributes of the object clicked and updates the palette accordingly. The next image you create will have these attributes. *On bitmap graphics:* fills in the foreground fill color with the color on which the end of the eyedropper tool is clicked.

The Object Tools Palette

The tools in the Object Tools palette create objects with attributes as set in the **Color, Pattern, Pen** palette. Want to change the way your object looks? No problem! Any attribute of an object graphic can be changed by selecting the graphic with the Pointer tool, then changing from the settings in the Color, Pattern, Pen palette. The changes are reflected immediately.

- **Text Tool** – Creates new text and edits existing text. To use the Text tool, select it from the Object Tools palette–the cursor takes on the shape of an I-beam. Click in the display area to automatically create a text box, and type the desired text. The text will have the attributes currently displayed in the **Color, Pattern, Pen** palette. You can change these settings before you begin typing, or while typing, to change the text color. If a text box is created but no text is typed inside it, the text box will disappear from the display when deselected.

  To create a text object of a specific width, select the Text tool, then click and drag the desired width on the display. When you type into this area, the text will automatically wrap when it gets to the end of the defined width. You can also set the width of the text object by ending a line with a return (pressing the Return key).

  Use the Pointer to resize and move the text box as necessary. To edit the text box's attributes, such as fill pattern and color or pen and color, select the text box with the Pointer, then choose the desired attributes from the **Color, Pattern, Pen** palette.

  To change the formatting of an entire block of text, select the text box with the Pointer tool, then choose the desired formatting from the **Font** and **Style** menus. The formatting changes are applied to all the text within the selected text box. To format only part of the text in the text box, use the Text tool to highlight the desired portion, then choose the formatting from the **Font** and **Style** menus (Figure 7.8).

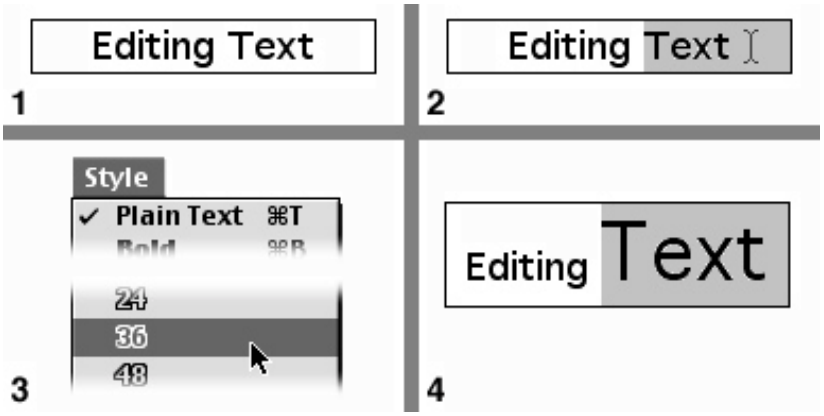


*Figure 7.8: Editing a selected segment of Object text. 1) The text before editing. 2) Selecting the text to be edited. 3) Choosing the desired formatting from the **Font** and **Style** menus. 4) The selected text is immediately updated.*

The formatting changes are applied to only the text that was selected.

- **Line Tool** – Creates straight lines using the selected Pen settings. Hold down the Shift key to constrain the Line tool to draw only horizontal, vertical, or 45˚ angles.

- **Rectangle Tool** – Creates a rectangle, of any height and width, with the selected fill and pen settings. Hold down the Shift key to constrain the tool to create squares.

- **Rounded Rectangle Tool** – Creates a rectangle, with rounded corners and of any height and width, using the selected fill and pen settings. Hold down the Shift key to constrain the tool to create rounded squares.

- **Oval Tool** – Creates an oval using the selected fill and pen settings. Hold down the Shift key to constrain the tool to create circles.

- **Polygon Tool** – Creates a polygon, of any shape or size, using the selected fill and pen settings. Vertices (corners) are added by clicking the mouse button while drawing the polygon. Hold down the Shift key to constrain the polygon lines to horizontal, vertical, or 45˚ angles. If a fill pattern is selected and the line's ends do not meet, a straight line is created joining the two ends, and the object is filled with the selected settings.

  Once created, a polygon can be reshaped by selecting it with the Pointer and choosing **Reshape Selected Polygons** from the **Graphics** menu. This command turns each of the polygon's vertices into a handle that can be dragged to different locations (Figure 7.9).

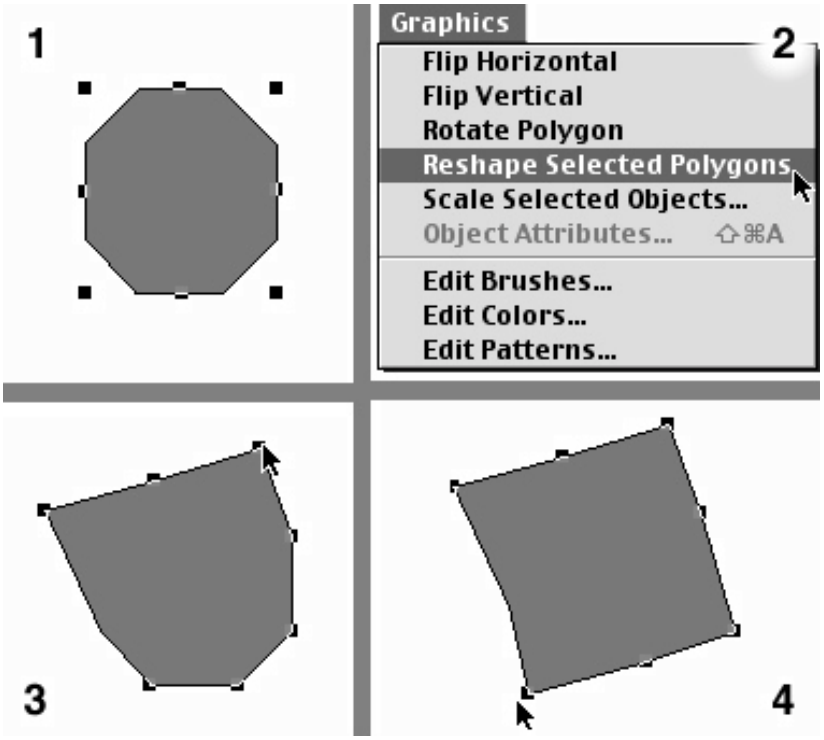


*Figure 7.9: Reshaping Polygons. 1) Select the desired polygon(s). 2) Choose **Reshape Selected Polygons** from the **Graphics** menu, which places handles at each of the polygon's vertices. 3) Drag the handles to shape the polygon as desired. 4) The newly reshaped polygon.*

A Polygon can also be rotated by selecting it and choosing **Rotate Polygon** from the **Graphics** menu. This command places a single handle on the polygon. Click and drag this handle to rotate as needed.

- **Freeform Tool** – Creates a line that corresponds to the mouse's movement from the moment the mouse button is pressed until it is released. If a fill pattern is selected and the line's ends do not meet, a straight line is created joining the two ends, and the object is filled with the selected settings. Once created, the freeform object can be reshaped and rotated in the same manner as a polygon.

- **Arc Tool** – Creates an arc that forms 1/4 of an oval. If a fill pattern is selected, the two end points are connected and the shape is filled in using the selected settings. Hold down the Shift key to constrain the tool to create 1/4 circles. Once created, the arc can be reshaped by selecting it with the Pointer and choosing **Reshape Selected Arcs** from the **Graphics** menu. This command places handles on the arc's end points. Dragging the handles adds to or takes away from the oval defined by the arc.

The Pixel Tools Palette

The tools in the Pixel Tools palette are for creating and editing bitmap images, and can be broken down into two basic types: selection tools and painting tools. Selection tools include the Lasso, Marquee, and the Lasso Marquee. Selection tools are used to select an area of a bitmap image for editing. Once selected, the area can be cut, copied, pasted and moved within your image.

Painting tools are used for both creating and editing bitmap images. The Pencil, Paint Brush, Spray Can, and Paint Bucket tools are used much like they are in real life, creating swatches

of color wherever they are applied. Deciding which one to use depends on how much paint you want to apply. The remaining tools, Text, Line, Rectangle, Rounded Rectangle, Oval, and Polygon, function similarly to their counterparts in the Object menu, with the exception that, once created, attributes of images created with these tools cannot be changed, as vector objects can.

Before any of the Pixel tools can be used, a rectangular "canvas" must be laid down. If used in a portion of the work area where there is no bitmap canvas, any of the Pixel tools will create this canvas before creating the bitmap image. Choose the desired tool, then click and drag the tool across the work area; you will see a rectangular dotted line that marks the border of your canvas. Make the canvas as large as needed, then release the mouse button (hold down the Shift key to constrain the canvas to form a square). You can now paint inside this canvas (Figure 7.10).

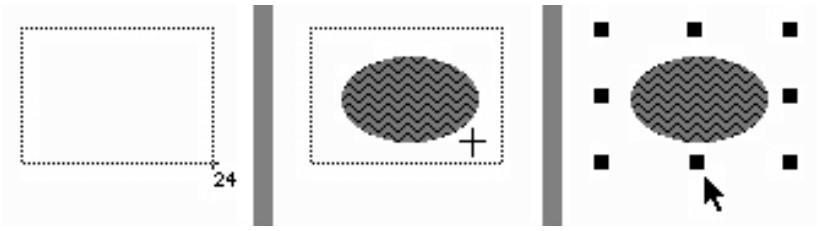


Figure 7.10: Creating a Pixel tool's "canvas." Drag any Pixel tool to create a rectangular "canvas" (left). Use any of the Pixel tools to "paint" on the canvas (center). A selected bitmap image (right).

As you are dragging the Pixel tool to create a canvas, notice the small number below the cursor; this number reflects the color depth for the image being created. When creating a canvas for a bitmap image, its color depth will be the same as your monitor's color depth. For instance, the number 24 below the cursor indicates that the image's color depth is 24-bit, which reflects a color depth of Millions of Colors on a Macintosh computer. Because creating images with a high color depth results in a larger file size and takes up more memory at runtime, you may want to consider converting your images to 8-bit, or 256 colors, with an adaptive palette, if your images do not require 16 or 24-bit color for display. For more information on color issues, including how to convert an image's color depth, see the section on Creating, Editing, and Storing Images, later in this chapter.

Tip

Any object image can be converted to a bitmap image by dragging the Pixel tool's Marquee tool over the image. This will automatically create a duplicate of the image as a bitmap.

Once a canvas is created, any of the Pixel tools can be used inside it without having to create an additional canvas. To paint inside a canvas, choose a Pixel tool and click the canvas once to activate it, then begin painting. Canvases can be created on top of other canvases. They are layered in the work area like any other graphic, and can be moved forward or backward in their layer order by choosing a command from the **Move Selected Objects** submenu under the **Edit** menu.

- **Marquee Tool** – Creates a rectangular selection area in a bitmap image. Hold down the Option Key while using the Rectangular Marquee tool to select only non-white pixels.

- **Lasso Marquee Tool** – A combination of the Lasso and Marquee selection tools, this tool lets you select a rectangular selection area in a bitmap image. After releasing the mouse button, the selection area collapses to the nearest border of non-white pixels (Figure 7.11).

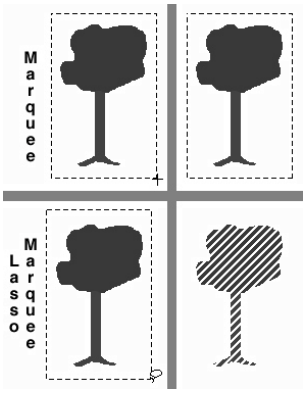


Figure 7.11: The Marquee creates a rectangular selection area (top right), while the Lasso Marquee's selection area collapses to the nearest border of non-white pixels.

- **Pencil Tool** – Allows for pixel by pixel editing of bitmapped graphics, as well as the creation of free form lines using the selected Pen Color. Hold down the Shift key to constrain the tool to draw only horizontal and vertical lines with a single pixel width.

  Because the Pencil tool is often used for editing pixels one at a time, it has the added feature of being able to erase a pixel's color. The first time the Pencil tool clicks on a pixel of a different color than the selected pen color, the pixel is painted. If the Pencil tool is clicked on the pixel for a second time, or is clicked on another pixel the same color as the selected pen color, the paint is erased, leaving white. This feature speeds the pixel by pixel editing of large areas. The Zoom tool can be used to magnify the image so that the individual pixels can be seen more easily.

  Hold down the Command key and click a bitmap image with the pencil tool to zoom in on the image for individual pixel editing. To return to normal size, again hold down the Command key and click in the bitmap image.

- **Lasso Tool** – Creates a freeform selection area in a bitmap image. After releasing the mouse button, the selection area collapses to the nearest border of non-white pixels. Hold down the Option key while using the Lasso tool to select the entire area enclosed by the lasso, including the areas with white pixels.

  The Lasso Tool can also distinguish between light and dark colored pixels. Figure 7.12 shows the Lasso used on a tree against a light colored background. By default, the Lasso Tool will select the entire non-white area enclosed within its lasso. If the Command key is held down while using the Lasso Tool, the selection area will collapse down to the border between light and dark colored pixels.

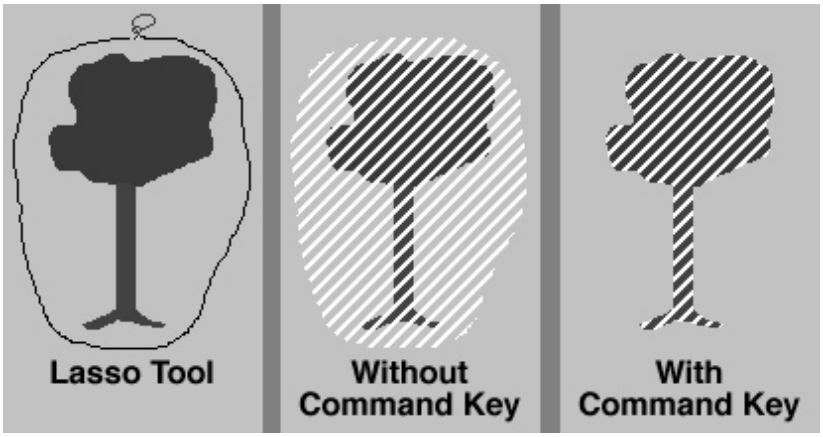


Figure 7.12: Using the Lasso while holding the Command key causes the selection area to collapse to the nearest border of light and dark pixels.

After making a selection, you can "paint" with the selected area by holding down the Command and Option keys while dragging the selection.

- **Paint Brush Tool** – Paints with the color selected in the Colors, Patterns, Pen palette's Pen Color setting. The amount of color applied is determined by the tool's brush size and shape, which are selected by double-clicking the Paint Brush tool, or by choosing **Edit Brushes** from the **Graphics** menu.

- **Eraser Tool** – Paints a white square over any bitmap image. By clicking with the Eraser tool inside a bitmap canvas, then double-clicking the Eraser tool icon in the tool palette, the selected canvas is painted white.

- **Spray Can Tool** – Paints with the color selected in the Colors, Patterns, Pen palette's Pen Color setting. This tool paints a diffuse pattern of dots, rather than a solid.

- **Paint Bucket Tool** – Paints over any solid colored area using the selected fill pattern and color until it reaches pixels filled with of another color than the original.

- **Text Tool** – Creates bitmap text. To choose how the text is formatted, click and drag the Text tool across the work area to create a canvas. After the canvas is created, choose the desired formatting from the **Font** and **Style** menus, then click in the canvas to place the cursor and begin typing. If you don't like the way the text looks, choose different formatting options before you deselect the text. Any changes are applied to all of the text just typed. Once the text canvas is deselected, the text cannot be edited.

  Unlike text created with the Object Text tool, once deselected, bitmap text loses its identification as a font and becomes a collection of pixels in the shape of text. As a result, bitmap text is a great way to incorporate unusual or elaborate fonts into your graphics without having to worry about whether or not your audience has the fonts loaded to display them correctly.

- **Line Tool** – Creates a straight line using the selected Pen settings. Hold down the Shift key to constrain the Line tool's angles to horizontal, vertical, and 45˚. Note: bitmap lines do not pick up the selected line styles. (e.g. dotted lines or arrowheads).

- **Rectangle Tool** – Creates a rectangle, of any height and width, using the selected fill and pen settings. Hold down the Shift key to constrain the tool to create only squares.

- **Rounded Rectangle Tool** – Creates a rectangle with rounded corners, of any height and width, using the selected fill and pen settings. Hold down the Shift key to constrain the tool to create only rounded squares.

- **Oval Tool** – Creates an oval using the selected fill and pen settings. Hold down the Shift key to constrain the tool to create only circles.

- **Polygon Tool** – Creates a polygon, of any shape or size, using the selected fill and pen settings. Vertices are added by clicking the mouse button while drawing the polygon. If a fill pattern is selected and the polygon's ends do not meet, a straight line is created joining the two ends, and the bitmap image is filled with the selected settings.

- **Grabber** – Used on zoomed bitmap images, the Grabber tool lets you move the image display area to see portions that don't fit on the display.

- **Magnifier –** Works on bitmap images; zooms in on the image for pixel by pixel editing.

Menu Bar Items

Edit Menu

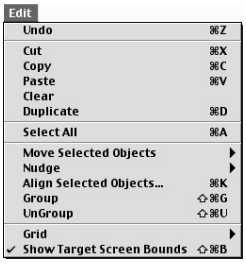


Figure 7.13: The Edit menu.

- **Undo** (⌘ Z) – Reverses last command.
- **Cut** (⌘ X) – Removes the selected information from the display and places it on the Clipboard.
- **Copy** (⌘ C) – Copies the selected information from the display and places it on the Clipboard.
- **Paste** (⌘ V) – Places information from the Clipboard onto the display.
- Clear – Removes the selected information from the display. Functions similar to Cut, except the selected information is not placed on the Clipboard.
- **Duplicate** – Creates an exact copy of the selected object and places it in the work area. More than one object can be selected and duplicated at a time.
- **Select All** (⌘ A) – Selects all objects in the work area.
- **Move Selected Objects** – Objects are layered in the work area in the order in which they were created. This menu item contains submenu options for moving the selected objects:
  - To Front (⌘ ⇧ =) – Moves the selected object(s) to the top layer.
  - Forward (⌘ =) – Moves the selected object(s) toward the top by one layer.
  - Backward (⌘ -) – Moves the selected object(s) toward the bottom by one layer.
  - To Back (⌘⇧ -) – Moves the selected object(s) to the bottom layer.
- **Nudge** – Contains submenu options for moving the selected object on the work area. Used alone, each key moves the selected object one pixel at a time. When used while holding down the Shift key, each key will move the selected object one grid unit at a time, based on the current grid settings.
  - Up (↑)
  - Down (↓)
  - Left (←)
  - Right (→)
- **Align Selected Objects** – Opens a dialog for arranging the selected objects relative to each other.
- **Group** – Combines all the selected objects so that they are treated as a single object (Figure 7.14). Useful when an image is composed of many parts and all need to be moved or copied together.

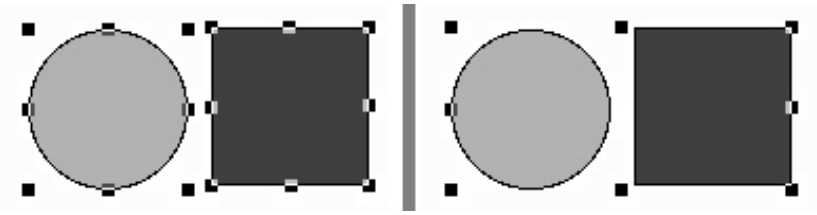


Figure 7.14: Two individual objects, selected (left) and grouped (right).

- **Ungroup** – Breaks a grouped object back down into its individual component objects. When an object is selected, this menu item is active if there are any grouped items in the object. You may have to select this option several times to ungroup all objects into individual components.
- **Grid** – Contains submenu options for setting and displaying a grid:
  * Show Grid (⌘ Y) – Toggles the grid on/off.
  * Snap Enabled – When creating or dragging objects, they are snapped to the nearest gridlines.
  * Settings… – Opens a dialog for turning off and on the grid, selecting the visibility of the grid, and changing grid spacing.
- **Show Target Screen Bounds** (⌘ ⇧ B) – Toggles on/off the dotted line marking the boundary of the Target Screen Size.

Graphics Menu

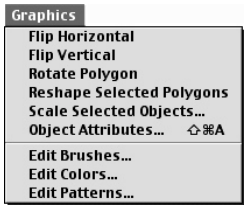


Figure 7.15: The Graphics menu.

- **Flip Horizontal** – Flips an object along its horizontal axis. Works with both Object-based and bitmap graphics. Note: this item does not affect text objects.
- **Flip Vertical** – Flips an object along its vertical axis. Works with both Object-based and bitmap graphics. Note: this item does not affect text objects.
- **Rotate Polygon** – Places a single handle on the selected polygon with which to rotate it as desired. Can be used on more than one polygon at a time. Works with images created with the Object-based Polygon tool.
- **Reshape Selected Polygons and Arcs** – If a polygon is selected, this tool turns each of the polygon's vertices into handles that can be dragged to different locations (see Figure 7.9). If an Arc is selected, it places handles on the arc's end points. Dragging the handles adds to or takes away from the oval defined by the arc. Can be used on more than one polygon and arc at a time. Only works with images created with the Object-based Polygon or Arc tools.
- **Scale Selected Objects** – Opens a dialog for scaling all selected graphics by the designated pixel values. Works with both Object-based and bitmap graphics.

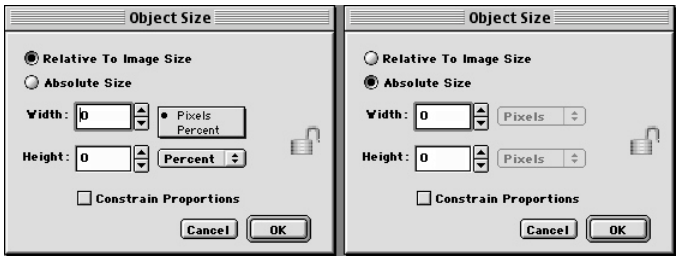


*Figure 7.16: The **Scale Selected Objects** dialog. Note that when **Relative to Image Size** is chosen (left), the image can be resized by pixels or percent values. When **Absolute Size** is chosen (right), the image can only be resized by pixel values.*

- Relative to Image Size – Increases or decreases an image's size by the designated pixel or percent values.
- Absolute Size – Changes an image's size to the designated pixel values.
- Constrain Proportions – Causes an image's width and height to be increased or decreased by the same value. If pixel values are chosen, the image's width and height is changed by the same number of pixels. If percent values are chosen, the image maintains its width to height ratio when changed.

- **Object Attributes** (⌘ ⇧ A) – Opens the Object Attributes dialog for the selected bitmap image, which gives you the option to change the color depth and mask its white pixels. See below for more information on masking pixels. Works with one ungrouped bitmap image at a time.
- **Edit Brushes** – Opens the Brushes Editor dialog, where you can edit the brushes available for use by the Paint Brush tool (Figure 7.17). Select a brush style to edit from the panel on the left, and use the pencil to edit its pattern in the panel on the right.

Tip

You may get unexpected results when scaling bitmap graphics because the individual pixels are either stretched or compressed. If you wish to change a bitmap image size, you will typically have better results by using a program such as GraphicConverter. A demonstration version of this shareware program is included on the onViz CD-ROM.

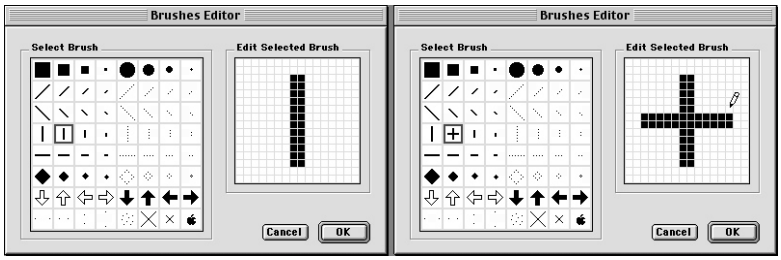


Figure 7.17: The Edit Brushes Dialog. The selected brush is shown before eidting (left), and after editing (right).

- **Edit Colors** – Opens the Colors Editor dialog, where you can edit the colors available in the pop-up palette for all of the color selection settings, such as those in a State's InfoCenter, or in the Color, Pattern, Pen palette.

  To edit individual colors, select the color, then click the **Edit** button to open the Apple Color Picker. Edit the color as desired, then click **OK** to return to the Colors Editor dialog.
- **Edit Patterns** – Opens the Patterns Editor dialog, where you can edit the patterns available in the Fill Pattern menu (Figure 7.18). To edit a pattern, select the desired pattern from the box on the left and use the pencil to edit the pattern in the middle box. Edit the pattern until the Example box on the right displays the pattern you want.

Creating, Editing, and Storing Images

All images created by or imported into the Image Editor are stored in the Image Library (Figure 7.19). The Image Library offers a convenient venue for creating new images, accessing and editing existing images, or importing images from an external file.

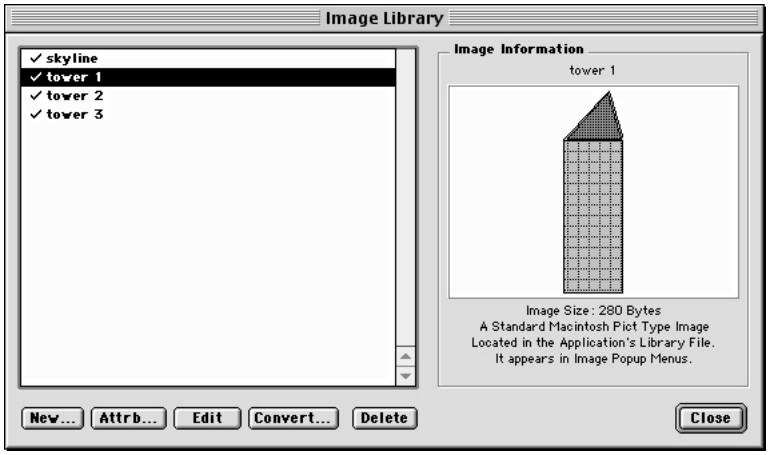


Figure 7.19: The Image LIbrary. Note the information displayed on the right for the image entry selected on the left.

During development, images are referenced by your application as files in the Image support folder in your project folder. When you build your target application, you have the option of integrating these image files into your application (internal library), or letting them exist as separate files saved to a disk (external).

Because all files are external during development, it is easy to update your application content. If, for example, you're working on a project containing a company's logo, and the company changes names, you can simply replace the old logo file with the new one and the change is automatically updated throughout your project. When you're ready to distribute your application, you have the option to import any or all media files into your application. Any assets not imported must be accessible to your application in a support folder or in a designated server location, if delivering over a network.

If you are delivering your application on CD-ROM, you may want to import all assets to make a single, self-contained file. In this way, individual image files cannot be opened or copied. For more information on the Image Library, see chapter 4, onViz Libraries.

Image Color and Format Considerations

When creating a canvas for a bitmap image, its color depth will be the same as your monitor's current color depth. For instance, the number 24 below the cursor indicates that the image's color depth is 24-bit, which reflects a color depth of Millions of Colors on a Macintosh computer. Because creating images with a high color depth results in a larger file size and takes up more memory at runtime, you may want to consider converting your images to 8-bit, or 256 colors, with an adaptive palette, if your images do not require 16 or 24-bit color for display.

After creating an image in the Image Editor, a file is in the Image Support folder with the same name as your Library Entry Name. This file name will be appended with either a .pct or .jpg extension, depending on whether this image is a PICT or jpeg image. Do not change this extension if you are planning cross platform or Internet delivery.

If you plan on delivering your application's images over the Internet, you will need to convert them to .jpg files. Images saved as .jpg files are compressed, which speeds the time they take to download. To convert your bitmap images to .jpg files, use a graphics editing program such as GraphicConverter. A demonstration version of this shareware program is included on the onViz CD-ROM.

Creating a New Image Library Entry

1. Open the Image Library by choosing **Image Library** (⌘ 5) from the **Libraries** menu.
2. Click the **New** button, which opens the Image Attributes window (Figure 7.20).

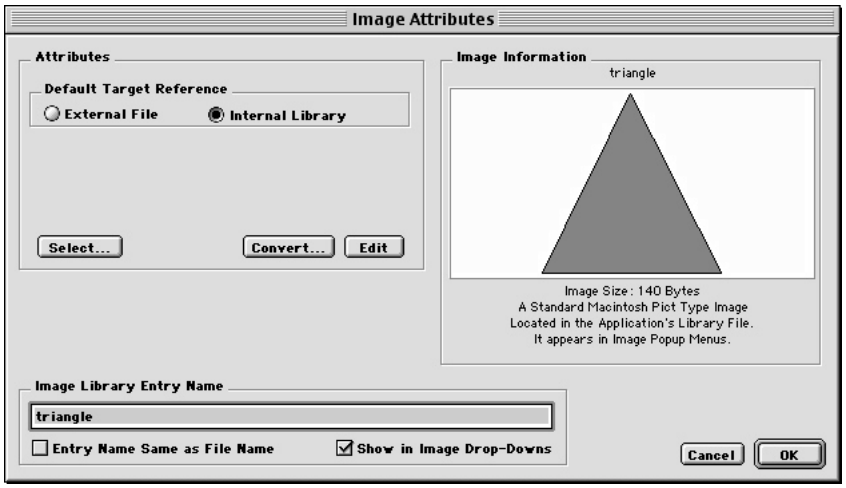


*Figure 7.20: The Image Attributes dialog. Note the **Image LIbrary Entry Name** field and the **Show in Image Drop-downs** check box at the bottom of the window.*

3. Select whether you want the default for this image to be referenced from an External file or from the Internal Library. Note that you can override this option when you build your application.

   From this point you can either select an image that you already have on disk, or create a new image in the Image Editor.

4. To select an image, click the **Select** button.

   A dialog opens allowing you to navigate to the image file (Figure 7.21).

5. When you find the file, select it and click **Open**.

   If the image is not already in your Image Support Folder, a dialog is presented that allows you to either move the file or create a copy of the file in your Image Support Folder. Select the option you prefer. After the image has been selected, you can click the **Edit** button to open the image in the Image Editor.

6. To create a new image, give your new image a name in the Image Library Entry **Name** field. The name you type in this field will be the name of the image file in your image support folder.
7. To create or edit another image, reopen the Image Library or select from the Image Editor controls drop down menu.
   Otherwise, close the Image Editor by clicking the Close Box, or by choosing **Close Window** (⌘ W) from the **File** menu.

   When you close the Image Editor, a file is in the Image Support folder with the same name as your Library Entry Name. This file name will be appended with either a .pct or .jpg extension, depending on whether this image is a PICT or jpeg image. Do not change this extension if you are planning cross platform or Internet delivery.

Tip

While creating your new image, you may want to compare it to one or two images already in the Image Library. You can conveniently view any two images in the library as a dimmed foreground/background while creating your new image by selecting the image from the dropdown list at the bottom of the Image Editor Window. For more information, see the above section on the Image Editor Window.

Editing an Existing or Imported Library Entry

1. Open the Image Library by choosing **Image Library** (⌘ 5) from the **Libraries** menu.
2. Choose the image you'd like to edit from the list on the left.
3. Click the **Edit** button.

   Your chosen image opens inside the Image Editor.
4. Edit the image as you want.
5. To edit another image, reopen the Image Library, or select from the Image Editor controls drop down menu. To close the Image Editor, click the Close Box, or choose **Close Window** (⌘ W) from the **File** menu. When you close the Image Editor, you are presented with a dialog that gives you the option to update the original image file or save the updated file with a new name. This dialog contains a **Do not show this dialog again** option. By selecting this option, onViz will always update the image file in the Support folder when closing the Image Editor window.

Converting an Image's File Format or Color Depth

1. Open the Image Library by choosing **Image Library** (⌘ 5) from the **Libraries** menu.
2. Select the image you'd like to convert from the list on the left.
3. TBA

Deleting an Entry from the Image Library

1. Open the Image Library by choosing **Image Library** (⌘ 5) from the **Libraries** menu.
2. Select the image you want to delete from the list on the left.
3. Click the **Delete** button.

   A warning message appears, asking you "Deleting this entry from the Image Library cannot be undone, delete it anyway?" Hold down the Option key when clicking delete to remove the item with no warning dialog.

4. Click **OK** to dismiss the warning message and complete deleting the image.

   The image is removed from the list.

5. Click **Close** to close the Image Library.

Deleting an image library entry does not remove it from the Image Support folder, but it does remove the reference for the current and all supporting documents.

Masking Pixels

While vector-based graphics take on the shape created, bitmap graphic canvases are rectangular. When you take a picture with a camera, whether it's of an apple or a person or the Grand Canyon, the printed picture is always rectangular. The same is true with bitmap images. The image may be a circular button or a car or a landscape scene, but the canvas on which the image resides is rectangular, and will be filled with white pixels where the border of the image ends. There may be times when you want your image to stand on its own, without the white-filled canvas. onViz' Mask Pixels feature allows you to break out of this box by making the unwanted white area around your image transparent.

For instance, if you are creating an animation of a doughnut floating over a background image of a red and white checkered tablecloth, you'll want to be able to see the tablecloth around the edges of the doughnut, as well as through the hole in its center. However, the unwanted white area in the doughnut's center and around its outer edges will obscure the tablecloth, ruining your illusion (Figure 7.22).

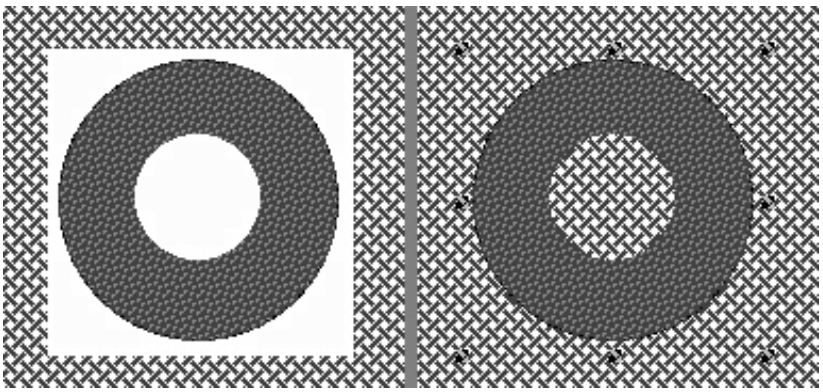


*Figure 7.22: A bitmap image before **Mask Pixels** (left) and after (right). Note the handles marking the edge of the masked image's selection outline.*

By using Mask Pixels, you can make these white areas transparent, and your image will appear to be a unique shape, such as a doughnut, instead of a rectangular picture of a doughnut. Here's how to use onViz' Mask Pixels feature:

Applying Mask Pixels

1. Open the desired image in the Image Editor.
2. Because Mask Pixels works by making the white pixels transparent, make sure those areas you want to be transparent are filled with white pixels. Use the Paint Brush tool to quickly cover large areas, then zoom in and use the Pencil tool for more precise pixel by pixel applications of white.
3. After you've applied white to the parts of your image that you don't want seen, choose **Object Attributes** (⌘ ⇧ A) from the **Graphics** menu to open the Object Attributes dialog.
4. Select the **Mask Pixels** check box, then click **OK**. The white areas of your image are now transparent.

5. To verify that the white areas have been masked, make a simple rectangle of some dark color. Select the rectangle, and choose **Move Selected Objects To Back** from the **Edit** menu. Now select the masked image and drag it over the rectangle. All the unwanted areas of your masked image should be transparent.
6. Don't forget to delete the rectangle you used for comparison before closing the Image Editor.

Tip

Because onViz' Mask Pixels feature works on every white pixel in the bitmap image, it may affect areas that you don't want masked. To avoid this effect, use the Pencil tool to color the pixels you do not want to be transparent with a very light shade of some color, such as gray.

Special Command Key Operations

⌘ = **Command** ⌥ = **Option** ⇧ = **Shift**

⌘⌥	Temporary Eyedropper
⌘ or double-click tool	Retain drawing tool
⌥ drag	Duplicate Image
⌥ nudge key	Move by grid setting
⌥⇧ nudge key	Snap to closest grid
⌥ click single handle	Highlights handle and allows drag/mudge on anchor point only
⌘ click	Shows current cursor x/y position
⌥ Pencil clicks	Auto draw selected line between click points
⇧ Resize	Constrains to circle/square
⌘ while creating oval or rectangle	Creates shape from center point
⇧ Lasso tool	Adds to selection
⌘ Lasso tool	Subtracts from selection
⌘ Lasso tool	Creates lasso by clicking points on screen (similar to polygon tool)
Double-click Lasso tool	Selects all paint objects as if lassoed
Double-click Marquee tool	Selects entire paint region
Drag with Magnifier tool	Magnifies selected area
⌥ click layered sprites	Select next sprite down in area

Chapter 8

Animation Support

Covered in this Chapter:

- Animation Terms
- Frame Control Palette
- Adding and Removing Frames
- Sprite Attributes Palette
- Adding and Removing Sprites
- Using the Multiframe Editor
- Previewing your Animation
- Using Mouse Actions (Smart Sprites)
- Animation Cycling
- Tweening
- Freeform Sprite Motion
- Animation using Variables

NOTE: This section of the documentation is largely incomplete due to ongoing changes in the program.

Animation Terms

Background

The static image underlying a frame of animation and all the sprites on the frame. The background is chosen with the Frame Control palette.

Foreground

The static image floating above the background, but underneath all the sprites, in a frame of animation. The foreground is chosen with the Frame Control palette.

Frame

One of possibly several sequentially numbered collections of graphics and/or text windows. Joined together, they form an effect similar to that of a motion picture reel. Frames are added and deleted with the Frame Control palette.

Transition

A dynamic transformation between the end of one frame of animation and the beginning of the next frame. Transitions are chosen using the Frame Control palette, and typically affect only the elements that have changed from one frame to the next (if nothing has changed, there is no transformation). However, a transition applied to the first frame of an animation affects the entire frame (a transformation from no frame to the first frame).

Sprite

An image stored in the Sprite Library, where it acquires an identity number (1-64) and letter (A-P). This identity allows it to be added to a frame and manipulated to form an animation.

Pose

One of possibly several variations of a given sprite. A sprite can have up to sixteen poses, which are collectively known as a sprite family. Poses are useful for creating sprites that seem to perform some activity, such as walking or running, as well as for automating the placement of sprites onto subsequent frames of animation.

Smart Sprite

A sprite with mouse actions assigned to it. Mouse actions are assigned using the Sprite Attributes palette, and control navigation by allowing the animation to jump to some other frame, or by causing application flow to exit the Output State or to bridge to some other location in the project.

Tween

A special type of transition available for use only between two adjacent frames of animation. Tweening looks at the difference in a sprite's size and location between the two frames and at the set transition time, and then calculates how the object should move to make a smooth transition from the first size and location to the next.

Frame Control Palette

The Frame Control palette provides the tools to create animations and dynamic interactive interfaces for your project. With it, you can add and delete animation frames, set time delays, adjust synchronization, add target frames, cycle animation, designate multimedia elements, such as sound and QuickTime movies, for playback, and apply transitions, such as dissolves and fades, to frame changes and/or the objects (sprites) on the frames.

The Frame Control palette works in conjunction with the Image Editor and the Sprite Attributes palette. Graphics are created in the Image Editor and are designated as sprites in the Sprite Attributes palette. The Sprite Attributes palette supports mouse actions, such as mouse over and mouse down, allowing you to assign actions to sprites that make your project truly dynamic. For instance, you can use the Frame Control palette to create several frames of animation, designate one of those frames to be a target frame, and then use the Sprite Attributes palette to designate a sprite to, when clicked, take your user to the target frame. For more information on using the Frame Control palette's tools, see chapter 6, Output State.

The Frame Control palette can be accessed by opening any Output State's Presentation window. It can be closed by clicking the Close Box in its upper-left corner. Once closed, it can be brought back to the foreground by choosing **Frame Control Palette** (⌥ ⌘ F) from the **Window** menu. The Frame Control palette has two collapse buttons: the horizontal collapse button hides the lower half of the palette, so the palette takes up less space on the display; the vertical collapse button causes the entire palette to retract into the drag bar. Double-clicking the drag bar also causes the palette to retract. While the palette is retracted, double-clicking the drag bar or clicking the vertical collapse button will cause the palette to open back up.

Adding and Deleting Frames

Adding Frames to your Animation

1. Click the **Add, Insert, Remove Frames** button. The **Add, Insert, Remove Frames** button is labeled with the total number of frames in your animation (i.e. **of 1**).

   The Add, Insert, Remove Frames dialog is opened.

2. Type the number you want to add into the **Enter a Number** field.

3. Click **Add**.

The specified number of frames are *added to the end* of any current frames.

You can also open the Add, Insert, Remove Frames dialog by choosing **Add/Insert/Delete Frames** (sc I) from the **Animation** menu.

Tip

You can easily insert a single frame by choosing **Insert Single Frame** (⇧⌘I) from the **Animation** menu.

Inserting Frames into your Current Frames

Using the previous technique adds frames to the end of your current set of frames, but there will be times when you want to insert frames *between* your current frames:

1. Go to the frame before which you want to insert the new frames.

   Use the **Goto Frame** button, which is labeled with the current frame number, or use the **Up/Down Frame** arrows.

2. Click the **Add, Insert, Remove Frames** button. The **Add, Insert, Remove Frames** button is labeled with the total number of frames in your animation (i.e. **of 1**).

   The Add, Insert, Remove Frames dialog is opened.

3. Type the number you want to insert into the **Enter a Number** field.
4. Click **Insert**.

   The specified number of frames are *inserted right after the current frame*.

Deleting Frames

Frames are deleted forwards, so you'll need to perform this action from the first frame of all those you want to delete. In other words, if you are on the last frame you can only delete one frame; if you are on the second-to-last frame, you can only delete two frames, etc. If you are on frame five of ten, and delete three frames, you will delete frames five, six, and seven.

Tip

You can easily delete the current frame by choosing **Delete Current Frame** (⌘X) from the **Animation** menu.

1. Go to the first frame of those you want to delete.
2. Use the **Goto Frame** button, which is labeled with the current frame number, or use the **Up/Down Frame** arrows.
3. Click the **Add, Insert, Remove Frames** button. The **Add, Insert, Remove Frames** button is labeled with the total number of frames in your animation (i.e. **of 1**).

   The Add, Insert, Remove Frames dialog is opened.

4. Type the number of frames you want to delete into the **Enter a Number** field.
5. Click **Delete**.

Sprite Attributes Palette

The Sprite Attributes palette creates sprites from images in the Image Library. These images are added to the Sprite Library and assigned attributes, such as sprite number and pose, so that they can be placed on individual frames of animation. Sprites are numbered from 1 to 64, and each number can have up to sixteen poses, designated by the letters **A**–**P**. A pose is a variation of a given sprite. For instance, a sprite of a man walking might have a pose for each position his body goes through to take two complete steps. A sprite number with all of its poses is called a sprite family. Sprites are always referenced by their number and pose, attributes that are particularly important when using variables to manipulate sprites, or when branching the application flow based on the condition of a sprite.

The Sprite Attributes palette also creates Smart Sprites. A Smart Sprite is a sprite that, when a mouse action is performed on it, will take the user to a specific frame in the animation, or to a different State. Mouse actions include mouse over, mouse down, and mouse up. Any of a sprite's poses can be made into a Smart Sprite.

The Sprite Attributes palette is divided into two sections, the Sprite Library, on the left, and Mouse Actions, on the right. If you are not using mouse actions, or if you have already finished assigning them, you can conserve screen space by collapsing the Sprite Attributes palette so that only the Sprite Library section is showing. Click the triangle at the top-middle of the palette to hide/show the Mouse Actions section of the palette.

The Sprite Attributes palette can be accessed by opening any Output State's Presentation window. If it is not visible, Open the **Window** menu and place a check mark next to Sprite Attributes palette (⌥⌘S) .

Options available on the Sprite Attributes palette:

Sprite Library

This grid displays all the Sprite Library entries for the current Output State. Double-click an empty sprite location to create a new sprite. Double-click an existing sprite entry to open the image in the Image Editor.

For more information, see Adding Images to your Animation, below.

- **Sprite Number** – Sprites are referenced by this number, which starts at one and goes through sixty-four. Each sprite number can have up to sixteen poses associated with it, collectively referred to as a sprite family. Only one member of any sprite family can be on a single frame of animation. You can have up to 64 individual sprites on any given frame of animation.
- **Sprite Pose** – Sprites are also referenced according to their position, or pose, in the sprite family (1-A, 1-B, 1-C, etc.). Incorporating sprites as poses is a great way to automate sprite placement. Once the first sprite pose is placed in the design area, the rest of the sprite family can be added by simply leaving it selected and adding more frames. For instance, if sprite 1-A is added to frame 1 of 1 and left selected, adding five more frames will automatically add sprite 1-B to frame 2, sprite 1-C to frame 3, sprite 1-D to frame 4, and sprite 1-E to frame 5.

Tip

Each Output State carries its own set of sprites. If you need to place the sprites in the current Output State into a different Output State, You can use the Sprite Attributes palette to copy and paste them. You can also copy and paste the entire Output State, and its sprites will be copied with it. For more information, see Adding Images to your Animation, below.

- **Info for Selected Library Sprite** – Information includes: sprite number and pose, if the sprite is on the current frame, and its X and Y coordinates in the design area. The **Find on Frame** button locates the sprite on the current frame, if there, and selects it, which is particularly helpful if you have many sprites on one frame, or are searching for a sprite that is layered underneath another sprite.
- **Sprite Selection** – Designates a graphic for the selected entry in the sprite library. The graphic can be taken from the Image Library, imported from an external file, or duplicated from another Sprite Library entry. The **Sprite Selection** dropdown menu can also be used to create a new graphic or edit an existing one.
- **Preserve Size** – Prevents a sprite's size from being altered in a frame of animation. However, if a sprite's size is altered in the Image Editor, the change in size is reflected in the sprite library and in the frame. Deselecting this option allows the sprite to be resized on a frame of animation without affecting its original size in the Image Library. The image's original size in pixels is displayed just below the **Preserve Size** check box.

  If **Preserve Size** is deselected, and changes are made to the image in the Image Library, the changes will be reflected in the sprite's thumbnail in the Sprite Library, but not on the frame of animation. If the **Preserve Size** check box is then selected, the changes will be passed on to the sprite on the frame.
- **Sprite Alignment Anchor** – Works in combination with **Preserve Size**. Designates an anchor point for the sprite, so that if its size is changed in the Image Editor, and **Preserve Size** is selected, the anchor point will remain at its original X and Y coordinates, even though the sprite's size on the frame has changed. For example, if a sprite's center alignment anchor is selected, and the sprite's size is scaled up in the Image Editor, the sprite's center point will remain at its original X and Y coordinates on the frame, leaving the sprite's size to enlarge outward
- **Copy/Cut/Paste/Clear** – Allows these actions to be taken on selected sprites in the Sprite Library. When copying or pasting a sprite, all of its attributes are copied or pasted with it.
- **Info for Selected Sprite on Current Frame** – Selecting a sprite on a frame of animation causes this information about the sprite to be displayed: sprite number and pose, if the sprite is on the current frame, if it has any mouse actions, its X and Y coordinates in the design area, and its height and width in pixels. The **Find in Lib** button locates the sprite in the Sprite Library and selects it, which is helpful if there are very many sprites in the library.

  As well as showing a sprite's current position, the **X** and **Y** edit fields are also used to change its position on the frame. If **Preserve Size** is deselected, the **Wd** (width) and **Ht** (height) edit fields can be used to change a sprite's size on the frame.

Mouse Over/Down Actions

Allow a sprite to do one or more of the following actions: change appearance, play a sound, display a different cursor, go to a specific frame of animation, or route application flow. Sprites with mouse actions assigned to them are referred to as Smart Sprites.

- The **Mouse Over/Down/Up Actions** check boxes activate several options that control the way the sprite responds when acted upon by the cursor.
  - **Mouse Over/Down Image** – Chooses an image that the selected sprite will change into when the cursor passes over/clicks on it. The dropdown menu allows you to choose an existing image from the Image Library, create a new image, import an image from an external file, or edit an existing image.
  - **Mouse Over/Down/Up Action** – Causes the animation to respond in a certain way when the cursor passes over the selected sprite. The dropdown menu contains responses such as moving to a specific frame in the animation, bridging to a different State in your project, or exiting the current Output State.
  - **Mouse Over/Down/Up Sound** – Causes a sound to be played when the cursor passes over the selected sprite. The sound can be a new sound or one that already exists in the Sound Library. You can also choose to edit an existing sound's attributes.

    If the sound's attributes have not been set to loop, the sound will be played only once each time the cursor enters the Mouse area or the Mouse button is depressed on the sprite. To set a sound to loop, edit its attributes in the Sound Library. See chapter 4 for more information about the Sound Library.
  - **Mouse Over/Down Inv (Invert)** – Causes the sprite's colors to be inverted when the cursor passes over it. When this option is checked, a small thumbnail image appears to the right of the check box showing what the sprite will look like inverted.
  - **Mouse Over/Down Thumbnail** – After a mouse over/down image is chosen, a thumbnail of the image is displayed. Double-clicking this thumbnail opens the image in the Image Editor.

    The alignment anchors around the edges and in the center of the mouse over/down thumbnail determine how the mouse over/down image will be aligned with the sprite selected in the Sprite Library. For instance, if the top-left alignment anchor is chosen, the two images will be aligned at their top-left corners. If the center alignment anchor is chosen, the two images will be aligned at their centers.
  - **Mouse Over/Down/Up Cursor** – Causes the cursor to change appearance when it passes over the selected sprite. The Cursor dropdown menu allows you to select a cursor from the Cursor Library for the cursor's new appearance, or to specify **No Change**. See chapter 4 for more information about the Cursor Library.

    When performing mouse events, onViz only recognizes the non-white areas of a sprite. To make mouse events recognize a sprite's white area, change the white to a very pale shade of color, such as pale gray or yellow. For more information, see chapter 7, the Image Editor.

Tip

Because mouse actions are associated with specific Output States, and not with specific images, mouse actions assigned to a sprite within one Output State will not automatically be assigned to that sprite in another Output State. However, a sprite that is copied and pasted between Output States will retain its mouse actions from the first State.

- **Reset Sprite Variables** – Contains options for changing a sprite's appearance and location based on the application of variables.
- **View –** Determines how the selected sprite responds to variables.
  - **No Change** – Selected by default. The selected sprite does not respond to variables.
  - **Hidden** – The selected sprite becomes invisible in response to some variable.
  - **Visible** – The selected sprite becomes visible in response to some variable.
- **Reset Position** – Causes the selected sprite to be reset to its original position when the Output State is exited.
- **Draggable** – Contains options for constraining the way a user can drag sprites across the design area. If one of the draggable options is chosen, X and Y coordinate fields appear that allow you to limit the distance the selected sprite can be dragged.
  - **No Drag Change –** Selected by default. If sprite draggability had been turned on or off with a Calculator, selecting this option will not affect the sprite draggability when the output is entered.
  - **Not Draggable** – The selected sprite remains static.
  - **Drag Anywhere** – Allows the user to drag the selected sprite.
  - **Drag Horizontal** – Allows the user to drag the selected sprite on its horizontal axis only.
  - **Drag Vertical** – Allows the user to drag the selected sprite on its vertical axis only.

Adding and Removing Sprites

Adding Images to the Sprite Library

Before an image can be added to an animation, it must first be converted to a sprite. Adding an image to the Sprite Library assigns it the necessary sprite number and pose so that it can be used as a sprite in animations. An image can be added to the sprite library using the following methods:

- The Sprite Selection dropdown menu lets you choose a sprite from images in the Image Library, create a brand new sprite, or edit existing sprites.
- Double-clicking an empty box in the Sprite Library creates a new image in the Image Library, and then places that image directly into the Sprite Library.
- A QuickSprite can be created by holding down the Control key while clicking in the design area. For more information, see the QuickSprites section, below.

Adding Sprites to your Animation

Once you have sprites in your sprite library, they can be added to your animation. To place a sprite onto the design area, select a sprite in the Sprite Library (the selected sprite will be highlighted with a red box), and choose **Add Sprite** (⌥ ⌘ 5) from the **Animation** menu.

Sprites can also be added to the Design Area by simply dragging them from the Sprite Library.

Sprite placement onto pre-existing frames can also be automated:

1. Add the sprite to the first frame.
2. Make sure that the sprite remains selected.
3. Click the **Frame Up** arrow to move forward through the frames of your animation.

   Each time the **Frame Up** arrow is clicked, all selected sprites are automatically added to the next frame.

Tip

Once you've added sprites to a frame, they will automatically be cloned onto every frame that is added after that. If a sprite is selected, however, it will clone to its next pose, if there is one. Unselected sprites will always clone as the exact same sprite. Hold down the Option key to clone the selected sprite, rather than its poses, to subsequent frames. For more information, see Adding Poses to your Animation, below.

Adding Poses Automatically

One of the best things about using poses is that you can automate their placement in your animation:

1. Add the first sprite pose to frame 1.
2. Make sure that the sprite remains selected.
3. Click the **Add, Insert, Delete Frames** button (labeled with **of 1**).

   The Add, Insert, Delete Frames dialog is opened.
4. Enter the **number of frames** you'd like to add, hold down the option key (⌥)then click **Add**.

   Add at least as many frames as you have poses. You'll be able to watch as each subsequent pose is automatically placed on the new frames.

Tip

If a selected sprite has additional poses, the next pose will be added when the **Frame Up** arrow is clicked while holding down the option key. Hold down the Cmd and Option keys to clone the selected sprite, rather than its next/previous poses, to subsequent frames.

Removing Sprites

1. Go to the frame containing the sprite you want to delete.

   Use the **Goto Frame** button, which is labeled with the current frame number, or use the **Up/Down Frame** arrows.

2. Select the sprite you want to delete.

   To select multiple sprites, hold down the Shift key while clicking each sprite, or simply drag a selection marquee around all of the sprites. If you select too many sprites, hold down the Shift key and click the extra sprites to deselect them.

3. Choose Delete Sprites (⌥ ⌘ D) from the **Animation** menu.

   The offending sprites are removed from the frame.

Tip

To delete sprites from multiple frames, use the Multiframe Editor. For more information, see Using the Multiframe Editor, below.

QuickSprites

There may be times when you want to create a sprite for an image that is not currently in the Image Library. QuickSprites streamline the process by allowing you to go directly from the design area to the Image Editor. Your image is then automatically inserted into the Sprite Control palette and onto the design area.

Creating a QuickSprite

1. Hold down the Control key and click anywhere over the design area to open a contextual menu. This menu also contains the Multiframe Editor.

2. Choose **QuickSprite** from the menu.

   The Image Editor appears.

3. Create your graphic, then close the Image Editor by clicking the Close Box, or choosing **Close Window** (⌘ W) from the **File** menu.

   Your image is automatically inserted onto the design area, as a sprite, in a location relative to where the image was located in the Image Editor. Your image is also automatically inserted into the Sprite Library, in the next available location.

4. Move your sprite to the desired location, if necessary.

Using the Multiframe Editor

The Frame Control palette makes it easy to perform operations like moving sprites or adding transitions, but performing those operations on a large number of frames can still be time consuming. The Multiframe Editor allows you to perform these operations and others on your entire animation, or just across a range of frames.

The Multiframe Editor can be accessed by holding down the Control key and clicking anywhere over the design area. This contextual menu also contains the QuickSprite command.

Adding Sprites using the Multiframe Editor

1. Find a frame that contains an instance of the sprite you want to add. If you want to add the sprite to every frame, it does not matter which frame you choose, as long as it has that sprite on it. If you want to add the sprite to a range of frames, choose the first frame in that range; that frame will be designated the **Start** frame in the **Frame Range** area of the window.
2. Select the sprite.
3. Hold down the Control key and click anywhere over the design area to open the Multiframe Editor.
4. Select **Add/Move Sprite(s)**.

   The Multiframe Editor is opened with the **Add/Move** tab in the foreground. The selected sprites are listed above the radio buttons.
5. Make sure that the **No Position Change** radio button is selected, and that the **Add Sprite(s) if not on Frame** check box is checked.
6. Designate the **Frame Range** to which you want the sprite added.

   If you want the sprite to be on every frame, place a check in the **All Frames** check box. If not, choose a range of frames. The frame number next to **Start** is the current frame, not necessarily the first frame in the animation. Use the up and down arrows to specify the **End** of the frame range.
7. Click **Add/Move** to add the sprite to the specified range of frames.
8. Click **Close** to close the Multiframe Editor and return to the Output State Presentation window.

Moving Sprites using the Multiframe Editor

When moving a sprite with the Multiframe Editor, the sprite is not only moved on the current frame, it is moved on every frame that contains it. If it does not reside on a frame, it can be added, and moved to the specified location, by placing a check in the **Add Sprite(s) if not on Frame** check box. If you want to move every instance of a sprite, including its various poses, place a check in the **Move all Poses of Selected Sprite(s)** check box.

1. Find a frame that contains an instance of the sprite you want to move. If you want to

move every instance of a specific sprite, it does not matter which frame you choose, as long as it has that sprite on it. If you want to move the sprite on a range of frames, choose the first frame in that range; that frame will be designated the **Start** frame in the **Frame Range** area of the window.

2. Select the sprite.
3. Hold down the Control key and click anywhere over the design area to open the Multiframe Editor.
4. Select **Add/Move Sprite(s)**.

   The Multiframe Editor is opened with the **Add/Move** tab in the foreground. The selected sprites are listed above the radio buttons.
5. Use the **Sprite Add/Move Options** to specify how you want the sprite moved:
   - **Relative to Position on Start Frame** – moves every instance of a sprite to a location a specified number of pixels from where the sprite sits on the Start frame. The Start frame is displayed in the Frame Range area, and is the current frame, not necessarily the first frame in the animation. After applying this option, every instance of the sprite will be at the same X and Y coordinates.
   - **Offset from Current Position** – moves every instance of a sprite a specified number of pixels from where it sits on its own frame. After applying this option, every instance of the sprite will have moved a specified number of pixels from where it used to sit.
   - **Move to Specified Screen Location** – moves every instance of a sprite a to a specific X and Y coordinate.
6. Use the up and down arrows to specify the distance in pixels you want the sprite to move. If **Move to Specified Screen Location** is selected, the X and Y values will represent the sprite's new coordinates on the grid.
7. If you want to move every instance of a sprite, including its various poses, place a check in the **Move all Poses of Selected Sprite(s)** check box.
3. If you want to add the selected sprite to frames it is not currently on, place a check in the **Add Sprite(s) if not on Frame** check box.
4. Designate the **Frame Range** in which you want the sprite moved.

   If you want the sprite to be moved on every frame, place a check in the **All Frames** check box. If not, choose a range of frames. The frame number next to **Start** is the current frame, not necessarily the first frame in the animation. Use the up and down arrows to specify the **End** of the frame range.
5. Click **Add/Move** to move the sprite on the specified range of frames.
6. Click **Close** to close the Multiframe Editor and return to the Output State Presentation window.

Removing Sprites using the Multiframe Editor

1. Find a frame that contains an instance of the sprite you want removed. If you want to remove every instance of a specific sprite, it does not matter which frame you choose, as long as it has that sprite on it. If you want to remove the sprite from a range of frames, choose the first frame in that range; that frame will be designated the **Start** frame in the **Frame Range** area of the window.
2. Select the sprite.
3. Hold down the Control key and click anywhere over the design area to open the Multiframe Editor.
4. Select **Remove Sprite(s)**.

   The Multiframe Editor is opened with the **Remove** tab in the foreground. The selected sprites are listed above the radio buttons.
5. Use the **Sprite Remove Options** to determine how many sprites you want removed:
   * **Selected Sprites** – Removes every instance of the selected sprite(s) from the designated frames. If you want to remove every pose of every selected sprite from the designated frames, be sure to place a check in the **Remove all Poses of Selected Sprite(s)** check box.
   * **All Sprites** – Removes every sprite from the designated frames.
6. Designate the **Frame Range** from which you want the sprite removed.

   If you want the sprite to be removed from every frame, place a check in the **All Frames** check box. If not, choose a range of frames. The frame number next to **Start** is the current frame, not necessarily the first frame in the animation. Use the up and down arrows to specify the **End** of the frame range.
7. Click **Remove** to remove the sprite from the specified range of frames.
8. Click **Close** to close the Multiframe Editor and return to the Output State Presentation window.

Resizing Sprites using the Multiframe Editor

1. Find a frame that contains an instance of the sprite you want resized. If you want to resize every instance of a specific sprite, it does not matter which frame you choose, as long as it has that sprite on it. If you want to resize the sprite from a range of frames, choose the first frame in that range; that frame will be designated the **Start** frame in the **Frame Range** area of the window.
2. Select the sprite.
3. Uncheck the **Preserve Size** check box.
4. Hold down the Control key and click anywhere over the design area to open the Multiframe Editor.

5. Select **Resize Sprite(s)**.

   The Multiframe Editor is opened with the **Remove** tab in the foreground. The selected sprites are listed above the radio buttons.

6. Use the **Sprite Resize Options** to specify how you want the sprite resized:
   * **Relative to Image Size** – Increases/decreases a sprite's size, relative to its Image Library size, by the either the specified number of pixels or the specified percent value. Results in every instance of the sprite being the same size.
   * **Relative to Current Size** – Increases/decreases a sprite's size, relative to its size on the frame, by the either the specified number of pixels or the specified percent value.
   * **Absolute Specified Size** – Resizes a sprite to the specified width and height, in pixels.

7. Use the up and down arrows to enter a **Width** and **Height** value, or simply type in a positive or negative number; these values can be either in pixels or by percent. The **Constrain Proportions** check box causes the sprite's width and height to be resized by the same amount.

8. If you want to resize every instance of a sprite, including its various poses, place a check in the **Resize all Poses of Selected Sprite(s)** check box.

9. If you want to add the selected sprite to frames it is not currently on, place a check in the **Add Sprite(s) if not on Frame** check box.

10. Designate the **Frame Range** in which you want the sprite resized.

    If you want the sprite to be resized on every frame, place a check in the **All Frames** check box. If not, choose a range of frames. The frame number next to **Start** is the current frame, not necessarily the first frame in the animation. Use the up and down arrows to specify the **End** of the frame range.

11. Click **Resize** to resize the sprite on the specified range of frames.

12. Click **Close** to close the Multiframe Editor and return to the Output State Presentation window.

Changing a Frame's Timing/Transition Options using the Multiframe Editor

1. Hold down the Control key and click anywhere over the design area to open the Multiframe Editor.

2. Select **Timing/Transition Options**.

   The Multiframe Editor is opened with the **Timing/Transition** tab in the foreground.

3. Place a check in the check box next to the options you want to change:
   * **Change Delay** – enter a new value, in seconds.

- **Change Sync** – choose a new synchronization from the dropdown list.
- **Change Transition** – choose a new transition from the dropdown list.

For more information on these options, see the Frame Control Palette section, above.

4. Designate the **Frame Range** in which you want the options changed.

   If you want the options to be changed in every frame, place a check in the **All Frames** check box. If not, choose a range of frames. The frame number next to **Start** is the current frame, not necessarily the first frame in the animation. Use the up and down arrows to specify the **End** of the frame range.

5. Click **Change** to change the timing and transition options on the specified range of frames.

6. Click **Close** to close the Multiframe Editor and return to the Output State Presentation window.

Previewing your Animation

After doing all that work on your animation, you'll probably want to take a look at it to see if it runs correctly. You can preview your animation from the beginning or from the current frame.

Previewing your Animation from the Beginning

1. From the **Animation** menu, choose **Run Animation** (⌘ R).
2. To stop the animation and return to the design window, choose **Stop Animation** (⌘ .) from the **Stop** menu. If the menu bar is hidden, use the keyboard shortcut (⌘ .) to stop the animation.

Previewing your Animation from the Current Frame

1. Go to the frame of animation from which you want to begin the preview.
2. From the **Animation** menu, choose **Run from Current Frame** (⇧⌘ R).
3. To stop the animation and return to the design window, choose **Stop Animation** (⌘ .) from the **Stop** menu. If the menu bar is hidden, use the keyboard shortcut (⌘ .) to stop the animation.

Using Mouse Actions (Smart Sprites)

Smart Sprites are sprites that have had a mouse action assigned to them. Mouse actions, assigned using the Sprite Attributes palette, allow you to route your users to another State or to another frame in the animation based on how they move their mouse or click on a sprite. The first example below uses Smart Sprites to bridge to another State. The second example uses Smart Sprites to jump to another frame in the animation.

Using Smart Sprites to Bridge to Another State

1. Create a new Application Map containing 1) a Start State, 2) a Design Output State, 3) two Radio Button Input States, and 4) a Stop State. Name the Radio Button Input States "target 1" and "target 2."
2. Open the InfoCenter for the Input State "target 1." On the Attributes page, set the **Number of Answers** to 1. Click **OK** to close the Input State InfoCenter.
3. Repeat step 2 for the Input State "target 2."
4. Use a Path to connect the Start State to the Design Output, then use another Path to connect the Input States to the Stop State.

   Do not connect the Design Output to the Input States, as a Bridge will be used instead.
5. Open the Design State Presentation window, and create two QuickSprites (see above for information on creating QuickSprites).

   The QuickSprites can be simple navigation buttons. The first QuickSprite should have the label "target 1," and the second "target 2."
6. Select sprite **1A** in the Sprite Attributes palette, and place a check in the **Mouse Up Actions** check box.
7. From the **Mouse Up Actions** dropdown menu, choose **Bridge to New Bridge**.

   The Setup Bridge Target window is opened.
8. From the **Bridge Type:** dropdown menu, choose **State in This onViz File**, then choose **target 1** from the list on the right. Click **OK** to close the Select a variable window and return to the Sprite Attributes palette.
9. For sprite **2A**, repeat steps 4-6 to create a Bridge to **target 2**.
10. From the Frame Control palette's **Synchronization** dropdown menu, choose **Wait for Smart Sprite**.
11. Close the Output Presentation window and return to the Application Map by choosing **Close Window** (⌘ W) from the **Edit** menu.
12. Open the InfoCenter for the Input State "target 1." Select the Answers 1-5 tab, and type "You chose target 1" into the **Enter Question** field. Click **OK** to close the Input State InfoCenter.

13. Repeat step 9 for the Input State "target 2," except type "You chose target 2" into the **Enter Question** field.
14. Test your application by choosing **Run** (⌘ R) from the **Map** menu.

    Each time you click one of the Sprites, you are taken to the corresponding Input State.

Using Smart Sprites to Change Animation Frames

1. Create a new Application Map with only a Design Output. No other States are necessary.
2. Open the Design Presentation window, and create three QuickSprites.

   The QuickSprites can be simple navigation buttons, and they should have the labels Go to Frame 1, Go to Frame 2, and Go to Frame 3, respectively.
3. Choose **Wait for Smart Sprite** from the **Synchronization** dropdown menu.
4. Add two frames to your animation.

   You should now have a total of three frames in your animation. Each frame should have all three sprites on it, and have the **Wait for Smart Sprite** option chosen from the **Synchronization** dropdown menu.
5. Go to frame 1, and create a QuickSprite that is simply labeled "Frame 1."

   We want this sprite to be on frame 1 only, so be sure to deselect it before going to any other frames. If you leave a sprite selected and go to another frame, the sprite will be copied to that frame.
6. Go to frames 2 and 3, and create two more QuickSprites labeled "Frame 2" and "Frame 3," respectively.
7. Select sprite **1A** in the Sprite Attributes palette, and place a check in the **Mouse Up Actions** check box.
8. From the **Mouse Up Actions** dropdown menu, choose **Go To Frame Number…**, and then type **1** when prompted.
9. Repeat Steps 5-6 for sprites **2A** and **3A**, except choose frames 2 and 3, respectively.

10. Test your animation by choosing **Run Animation** (⌘ R) from the Animation menu.

    Clicking one of the Go to Frame buttons should take you to that frame. To end your test and return to the Output Presentation window, click **Stop Animation** (⌘ .) from the **Stop** menu.

Limiting a Smart Sprite's Mouse Action Area

There may be times when you don't want a sprite's entire area to be clickable for mouse actions. For instance, if you have a sprite shaped like a doughnut, you might not want the center (the doughnut's hole) to be clickable.

To prevent part of a sprite from being clickable, fill it with the color white. When onViz examines a Smart Sprite for mouse actions, it ignores anything colored white. Therefore, if a doughnut's center has been filled in with white, it will not be clickable.

From time to time, however, you may need to use a Smart Sprite that has a great deal of white in it, and need the white areas of the sprite to be clickable also. For example, you may have a black background and want to use a clickable white text label. Although onViz will not allow the color white to be clickable, you can use in its place a very light shade of some color, such as gray or yellow. If you choose a color that is pale enough, it will appear as white in your graphic and still be clickable.

If your sprites are bitmap images, rather than vector images, you have even more options. Bitmap images can take advantage of ***pixel masking***, in which any of the image's white pixels are made transparent. ***Masking*** a bitmap's pixels can be particularly useful if your design area's background is a color other than white. For example, if you set your doughnut down on top of a red and white checkered tablecloth, you would expect to be able to see the tablecloth through the doughnut's hole. However, if you filled the hole with white to prevent it from being a clickable area, the tablecloth will be obscured.

Using onViz' Mask Pixels command allows you to make the white area in the doughnut's center transparent, thus serving two purposes: the doughnut's center is not clickable for mouse actions, and it is transparent, giving the doughnut the illusion of resting above the tablecloth. Pixel masking can also be used to hide a bitmap's rectangular canvas, allowing it to appear as a unique shape, rather than a rectangular picture of a unique shape. See chapter 7, the Image Editor, for instructions on masking pixels.

While vector images have the advantage of being able to assume any shape without being bounded by a rectangular canvas, they cannot take advantage of onViz' Mask Pixels feature. If you need to make a segment of a graphic transparent, use a bitmap, rather than a vector image.

Animation Cycling

Using the Frame Control palette's **Cycling** option, your animation can simulate a looping effect. You may want to use this option if you need your animation to repeat itself while your user looks over some information on the screen.

Entering a negative number in this field causes the animation to jump back the specified number of frames (jump range), at which point it starts moving forward again. By entering a positive number, your animation can also jump forward the specified number of frames.

To allow the user to break out of the loop and continue on with the animation (or to exit the output), use a Smart Sprite set to go to a frame outside the jump range. This Smart Sprite will need to be placed on every frame within the jump range, as your users may decide at any point during the loop that they want to continue.

Any Synchronization placed on frames within the jump range will work normally, but if Synchronization has been applied to the frame designated to be a jump frame, it will be ignored so that the animation can jump to the designated frame.

Tweening

Tweening looks at the difference in a sprite's size and location between two adjacent frames and at the set transition time, and then calculates how the object should move to make a smooth transition from the first size and location to the next. To demonstrate how tweening works, this example places a sprite in the first frame, changes its location in the second frame, and applies a tween transition to the movement.

1. Create a new Application Map with only a Design Output. No other States are necessary.
2. Open the Design Output's Presentation window.
3. Create a QuickSprite (see above for information on creating QuickSprites).

   Your QuickSprite can be any shape or size, but for simplicity's sake, choose a simple shape or design.
4. Move your sprite to the upper left corner of the design area.
5. Add a frame to your animation.
6. In the second frame, move your sprite down to the lower right corner of the design area.
7. Choose **Tween** from the **Transition** dropdown menu.
8. Click in the **Duration** field and type a "3," then press the Return key.

   If a transition type is chosen, setting a time here determines the duration of the transition.
9. Choose **Run Animation** (⌘ R) from the **Animation** menu.

   The sprite should flow smoothly from the top-left to the bottom-right of the window. Experiment with different durations to see how they affect sprite movement.

Freeform Sprite Motion

While Tweening is the best way to move a sprite in a straight line from one frame to the next, you'll need to use a different method for moving a sprite in a non-linear fashion. Non-linear, or freeform, sprite movement can be accomplished by creating the desired number of frames, then holding down the Option key and dragging a sprite across the design area. As the Sprite is dragged, the frames tick off at a rate of about 1 every half-second, and record the sprite's position at that time.

1. Add 19 frames to your animation.

   Any number of frames can be added, but the animation will be smoother if you use a lot of frames and move the sprite just a short distance between each frame.

2. Go to frame 1, and create a QuickSprite (see above for information on creating QuickSprites).

   Your QuickSprite can be any shape or size, but for simplicity's sake, choose a simple shape or design.

3. Hold down the Option key and begin slowly dragging the sprite.

   As soon as you grab the sprite, the frames begin progressing. If you move the sprite too slowly, its movement may not be registered from one frame to the next. If you move the sprite too quickly, it may appear to move too far between frames. Keep moving the sprite until you run out of frames. If it turns out that you didn't add enough frames, more can be added when the current ones have run out.

4. Choose **Run Animation** (⌘ R) from the **Animation** menu.

   Your animation will show your sprite moving incrementally from frame to frame.

Animations using Variables

Variables can be used to update many aspects of your animation, such as sprite number, sprite pose, sprite placement, and frame number. These changes can be based on any number of variables, including **Answer Entered**, **Total Score**, **Number of Visits**, or **Exit Path**, among others. For instance, you can create an animation of a thermometer in which the poses change to simulate mercury rising each time the user gets a correct answer, or an animation in which a sprite moves to a different location on the screen based on the user's answer.

Chapter 9

Input States

Covered in this Chapter:

- Input State Overview
- Types of Input States
- Input State InfoCenter
- Input State Presentation
- Positioning Mouse Bays for the Mouse Bays Input State
- Parsing Text
- Parsing Numbers
- Parsing Check Boxes Input
- Input Variables
- Using Input Features

Input State Overview

If the Output State is onViz' way of speaking to your audience, then the Input State is onViz' way of listening to their replies. The Input State provides a special dialog, called an input window, as a means to ask your users questions and capture their replies (Figure 9.1). The Input State then routes them through your application based on those replies. Not only can you use the Input State to test your users, but you can also evaluate their responses, record their scores, track how many times they have gone through a particular input and, track how long they have taken to reply to the question posed.

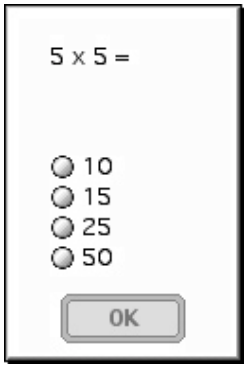


Figure 9.1: An Input State as it appears during runtime. Shown is a Radio Buttons Input.

The Input State lets you solicit information from your user, such as name, age, job title, etc., and then incorporate that information into your application using variable substitution (Figure 9.2). You could, for instance, ask your user's name, and then incorporate it when asking another question, as in "Thank you, Pat! And what is your age?"

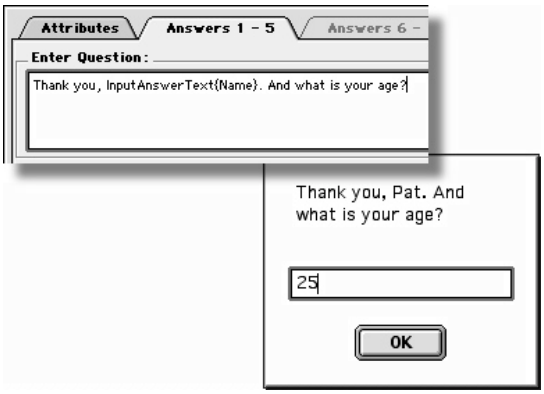


Figure 9.2: Variable substitution in an Input State. The user was asked her name in a previous Input. This information was captured using a variable and displayed in another Input during runtime.

In this example, the name is a variable that has been substituted into the feedback text. The Input State also collects statistics, like scores, and then makes them available for use as variables through the balance of your application. These variables can be used to add personalization to your application and, with a Conditional Path, to make routing decisions.

You can use a Calculator State to update the values of ***read/write*** Input Variables, such as **Input Score** and **Input Correct**; this is particularly beneficial if you are using Smart Sprites to develop questions for your application. You can place an Input State on the Application Map and use a calculator to write to the total score. In this way this statistic is included in the application report without having the application actually flow through the Input. For more information on Calculator States, see chapter 14.

Types of Input States

There are five types of Input States, differentiated by the kinds of input they allow:

- Mouse Bays
- Radio Buttons
- Check Boxes
- Enter Text
- Enter Numbers

The first three types of Input States, Mouse Bays, Radio Buttons, and Check Boxes, are mouse-based Inputs. Similar to multiple choice questions, mouse-based inputs present the user with a series of choices, at least one of which must be selected.

The last two types of Input States, Enter Text and Enter Numbers, are keyboard-based Inputs, which allow free-form responses. In the case of the Enter Text Input, the free-form response is similar to a short answer essay question, as the user can type in up to 63 characters (use a Text Output set to be editable for longer essay type questions). In the case of the Enter Numbers Input, the free-form response is similar to a fill-in-the-blank question. These Inputs can be set up to evaluate the user's response, checking for anything from an exact answer, to keywords, and even to specific letters or numbers.

When using the Map Tools palette to add States to your Application Map, you can access a contextual menu listing the different types of Inputs by holding down the Control key while clicking on the Input State icon. Choose the desired type of State from the menu; your mouse pointer will turn into a star. Then click in an empty location in the work area where you want the State to be positioned.

Once placed, an Input can be changed to any of the five types by opening its InfoCenter and choosing one of the selections from the **Input Type:** dropdown list.

With the exception of the Check Boxes Input, Inputs can have up to 10 exits allowing you easily route the application based on the user's input.

Mouse Bays

The Mouse Bays Input lets you create up to ten click-sensitive areas, called Mouse Bays, over a graphic (Figure 9.3). onViz will route your users through the application based on which Mouse Bay they click.

You also use the Mouse Bays Input when creating draggable objects. onViz will track and report on the bays into which a user has dragged a sprite. You can use conditional paths to check if a sprite is in a specific bay and route the application flow accordingly.

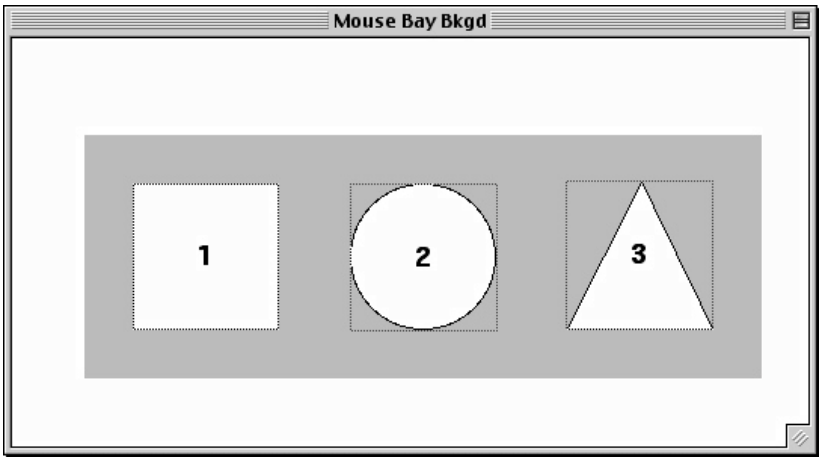


Figure 9.3: An example of three Mouse Bays in an Input Presentation. Notice the dotted lines marking the boundaries of the Mouse Bays.

You can use Smart Sprites to create up to 64 Mouse Bays at a time on the display, but they are not automatically tracked and scored as is the Mouse Bays Input. Therefore, use the Mouse Bays Input when you want to track and score your users' response.

Radio Buttons

The Radio Buttons Input most closely resembles a multiple-choice question. Your user clicks one of up to ten radio buttons that correspond to the chosen response (Figure 9.4). The Radio Buttons Input allows only one answer at a time to be chosen from the list of possible answers.

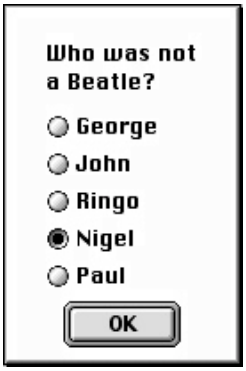


Figure 9.4: A Radio Buttons Input as it appears during runtime.

The Radio Buttons Input also offers a quick way to give a user information or feedback. Simply put the information or feedback in the Question field, and do not enter anything in the answer field (Figure 9.5). When onViz passes through a Radio Buttons Input set up in this way, the information is presented with an **OK** button. Once your users have read the information, they can dismiss the dialog and continue the application by pressing the Return key or clicking the **OK** button.

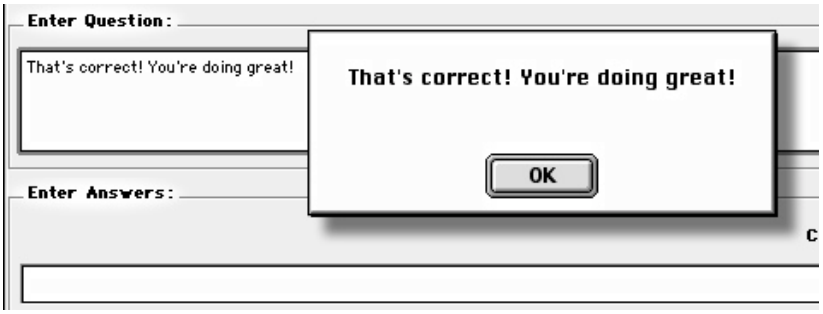


*Figure 9.5: A Radio Buttons Input being used to provide feedback. Notice that the feedback is placed in the **Enter Question** field, and that the **Enter Answer** field has been left blank.*

Check Boxes

The Check Boxes Input is similar to the Radio Buttons Input, except that it allows more than one answer at a time to be chosen from the list of up to ten possible answers. With this flexibility, you can require your user to choose *all* designated correct responses in order to answer the question correctly, or just one correct response out of several. This type of Input always has four Exit Points, regardless of the number of answers it contains. The four Exits correspond to a Correct Selection, an Incorrect Selection, a "Time Out" path, and a limit on the number of tries. For more information, see the section below on Parsing Check Boxes Input.

onViz has a number of variables designed specifically for the Check Boxes Inputs. These variables will let you test to see exactly which checkboxes the user selected. See the Working with Checkbox Inputs section later in this chapter for details on using these special variables.

Figure 9.6: A Check Boxes Input as it appears during runtime.

Enter Text

The Enter Text Input allows your audience to type their own response, rather than choosing one that you have presented. Their response is then judged against a series of conditions you set in the Input Info-Center (Figure 9.7). You can use up to ten sets of conditions to evaluate, or parse, the response for its accuracy. The response is checked for accuracy against the first condition, then the second, then the third, and continues until it either meets a condition or runs out of conditions and is judged as incorrect. The conditions include criteria such as Exact Match, Has String, and Has Word, with the added ability to combine criteria using "and" and "or." The Enter Text Input accepts only text, and cannot parse numbers. For more information, see the Parsing Text section, later in this chapter.

Best Practice

Best Practice: If none of the user's responses match the set conditions, onViz will not let the user continue. Therefore, It is a good idea to leave the last condition blank. That way, if the user enters an unanticipated response, the application flow can continue

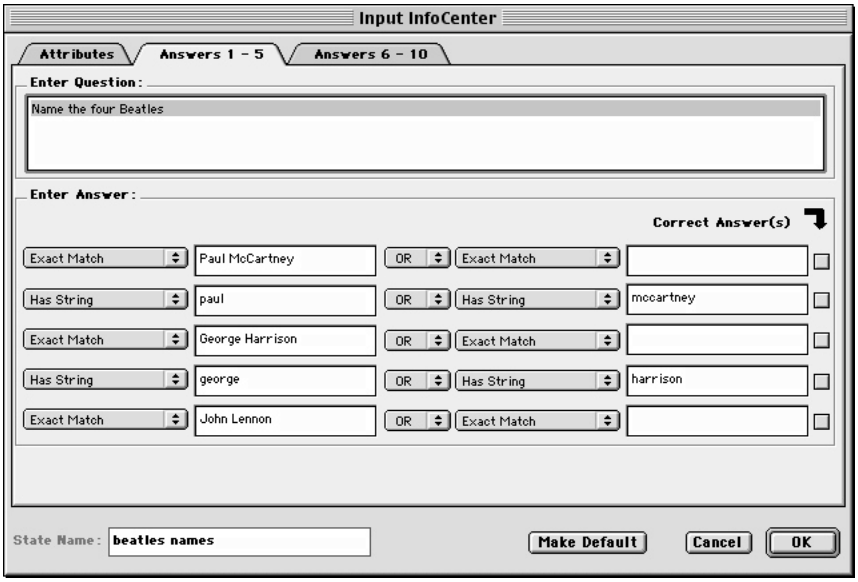
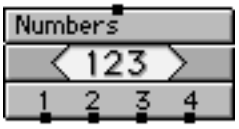


Figure 9.7: An Enter Text InfoCenter, with five conditions for evaluating a user's response.

Enter Numbers



The Enter Numbers Input is similar to the Enter Text Input, in that allows your audience to type their own response, and allows you to create up to ten conditions in order to evaluate, or parse, the response for accuracy (Figure 9.8). The Enter Numbers Input accepts only numbers, decimal points, commas, and currency units. For more information, see the Parsing Numbers section, later in this chapter.

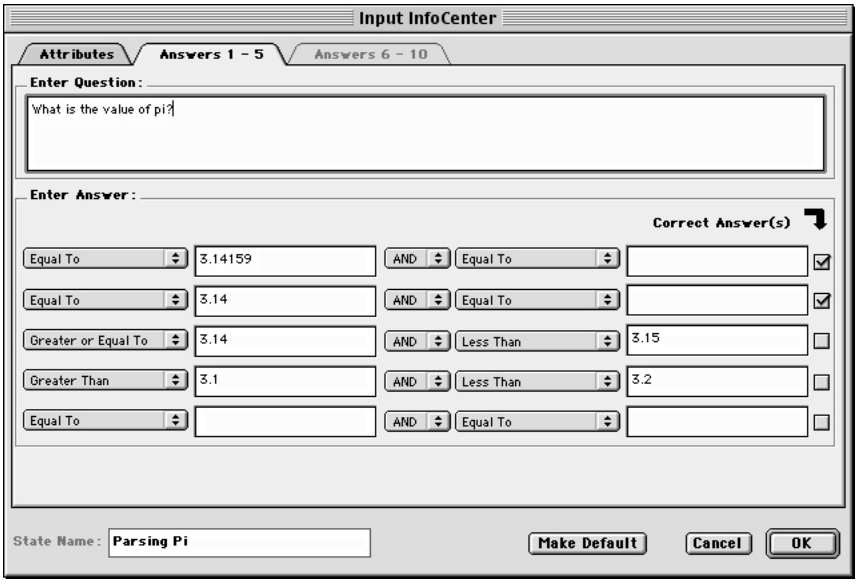


Figure 9.8: An Enter Numbers InfoCenter, with numerical condition fields for evaluating a user's response.

Input InfoCenter

The Input InfoCenter is used to set up an Input's presentation attributes and runtime features, and is divided into three tabbed panels: the Attributes tab, the Answers 1–5 tab, and the Answers 6–10 tab (active only if more than five possible answers are selected) (Figure 9.9). The Input InfoCenter varies depending on the type of Input selected. For instance, if the Mouse Bays Input type is chosen, the Answer tabs will be grayed out, or inactive, and the Runtime Features section will include options concerned with mouse clicks. If the Enter Text Input type is chosen, the Answer tabs will be available and the Runtime Features section will include options concerning capitalization and punctuation. The InfoCenter can be accessed by double-clicking the rectangular area in the upper 1/3 of the State's icon.

Tip

The Input State InfoCenter can also be accessed by selecting an Input State and then choosing Input InfoCenter (⌘I) from the **Map** menu.

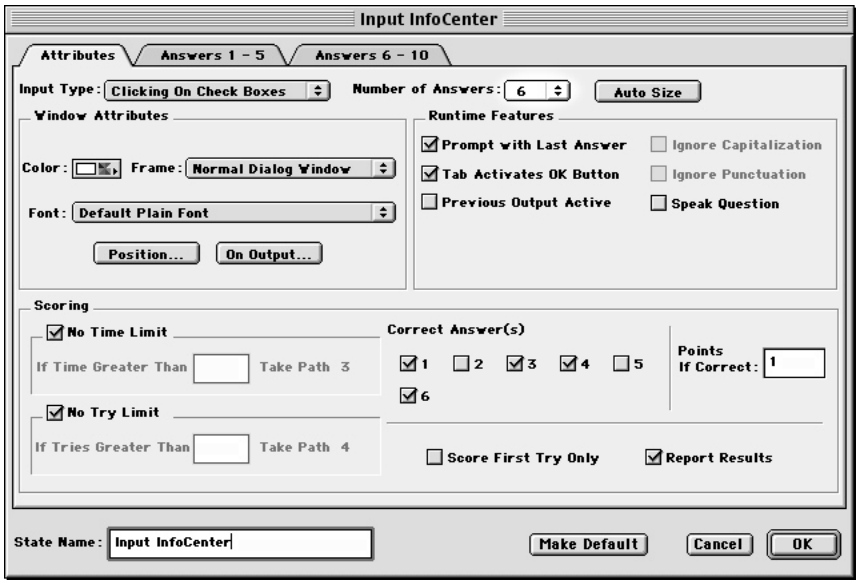


Figure 9.9: The Input InfoCenter. Notice that the Number of Answers dropdown menu is set to 6, causing both of the InfoCenter's Answer tabs to be active.

While a State's name can be changed from within its InfoCenter window, it can also be changed without leaving the Application Map. To change a State's name from the Application Map window, click once on the State's name, wait for the Name field to become a black rectangle with a red outline, then type the new name. Pressing Enter or Return, or clicking anywhere outside the Name field, makes the new name take effect.

Attributes Tab

Input Type

Once placed, an Input can be changed to any of the five types by choosing one of the selections from this dropdown list. See above for a description of the five Input types.

Number of Answers

Designates the number of answers, up to ten, from which your audience can choose when this Input is run. The Number of Answers dropdown affects the following Input elements:

- Adds a corresponding number of boxes in the **Scoring** section's **Correct Answer(s)** field.
- Adds a corresponding number of Exit Points to the State's icon (except in the case of the Check Boxes. For more information, see the section below on Parsing Check Boxes Input.).
- Adds a corresponding number of **Enter Answer** fields to the Answers tabs.
- If five or less answers are chosen from the dropdown list, only the Answers 1–5 tab will be available. If six or more answers are chosen from the dropdown list, both Answers tabs will be available. See below for more information on the Answers 1–5 and Answers 6–10 tabs (Figure 9.10).

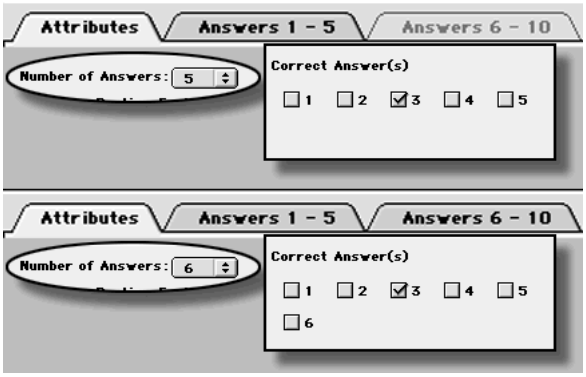


*Figure 9.10: If the **Number of Answers** dropdown menu is set to five or less, the Answers 6-10 tab is inactive (top). If the **Number of Answers** dropdown menu is set to six or more, the Answers 6-10 tab is active (bottom). The **Correct Answer(s)** section always contains a corresponding number of check boxes.*

- If the Mouse Bays Input type is chosen, adds a corresponding number of Mouse Bays to the State's Presentation window.

Window Attributes

Color: Determines the background color for this State's Input window. Each Input can have a different background color for its Input window. Click the button to access a palette from which to choose a color. While in the palette, click **Color Picker** to create a custom color.

Frame: Determines the style of border in which your Input window will be presented (Figure 9.11). There are four types of frames:

Best Practice

Note: If you select a custom color and your user is viewing the application in 256 color mode, your presentation window may not look as you designed it. If the minimum color depth for your application is 256, you would be better to select a color in the current application palette.

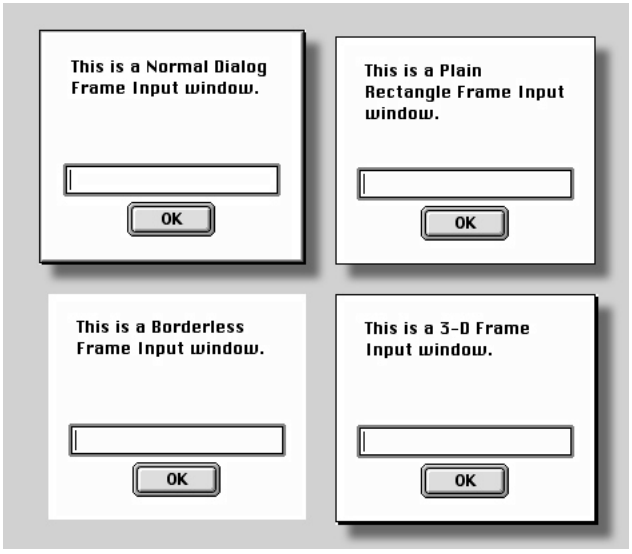


Figure 9.11: Input frame types.

- **Normal Dialog Window** – slightly beveled window with a small drop shadow.
- **Plain Rectangle Window** – Flat window surrounded only by a thin black line.
- **3-D Window** – Rectangular border with a three-dimensional drop shadow.
- **Borderless Window** – Input information is enclosed within an invisible rectangular border.

Font: Determines the display font for this State's Input window. The dropdown list lets you choose an existing font from the Font Library, add a new font, or edit an existing font.

Position: Opens a positioning window that lets you set up the Input window's screen position and size. The first time the **Position** button is used within each Input, the Background Output dialog opens, prompting you to choose an Output that will serve as the background for positioning the input window. After choosing the background and clicking **OK**, you can relocate and resize the positioning window as desired. Use the locator handle in the upper left-hand corner of the positioning window to move it around. Use the size handle in the lower right-hand corner to change the positioning window's size (Figure 9.12). When resizing the positioning window, be sure to that you do not make it too small to display its answers.

Tip

There may be times when you resize the positioning window to purposely hide the **OK** button. With the **OK** button hidden, you can still press the Return, or Enter, keys to close the positioning window and return to the InfoCenter.

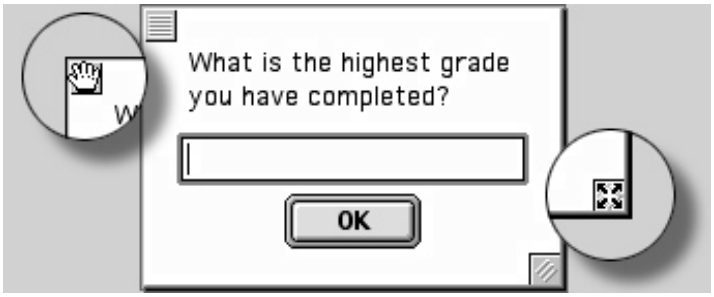


Figure 9.12: Moving and Resizing Input windows. Note the locator handle in the upper-left corner, and the positioning handle in the lower-right corner.

After moving and resizing the positioning window, choose **Close Window** (⌘ W) from the **File** menu, click **OK**, or press the Enter or Return keys, to close the window with the new size and position saved and return to the InfoCenter. Hold down the Command key and press the Period key to close the window without saving your changes.

To change the Output chosen to be the input window's positioning background, click the InfoCenter's **On Output** button to once again access the Background Output dialog.

On Output: Opens the Background Output dialog, prompting you to choose an Output that will serve as the positioning background for this input window. Functions similarly to the Position button, except that the Background Output dialog is opened every time, rather than just the first time. After choosing the background and clicking **OK**, you can relocate and resize the positioning window as desired.

No Output allows you to position the Input Presentation Window with no Output background.

After moving and resizing the positioning window, choose **Close Window** (⌘ W) from the **File** menu, click **OK**, or press the Enter or Return keys, to close the window with the new position saved and return to the InfoCenter. Hold down the Command key and press the Period key to close the window without saving your changes.

When the Mouse Bays Input type is chosen, the **Position** and **On Output** buttons can be used to position the State's Mouse Bays. You can also position the Mouse Bays by double-clicking the Input State's Presentation field. However, the **Position** and **On Output** buttons may be the more convenient methods, as they can be accessed from the InfoCenter along with the **Number of Answer** and **Runtime Features** settings, allowing all these options to be set from the same window. See below for instructions on positioning a Mouse Bays Input's Mouse Bays.

Best Practice

Best Practice Tip: Because Mouse Bays and their borders are invisible to your audience, it's a good idea to designate a cursor change whenever the mouse passes over one. That way your audience's challenge will be choosing the right Mouse Bay, instead of trying to find where they are hidden on the screen.

Runtime Features:

Different Input types have different runtime features. These features are described according to Input type.

Mouse Bays

The following Runtime Features are specific to Mouse Bays only (Figure 9.13). Features for all other Input types are described after this section.

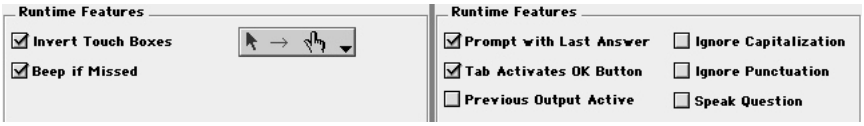


Figure 9.13: Runtime Features for the Mouse Bays Input (left) and all other Inputs (right).

Invert Mouse Bays: Causes a Mouse Bay's color to be inverted when it is clicked.

Beep if Missed: Causes the computer to emit an error sound (system beep) when the mouse clicks anywhere that is not a designated Mouse Bay.

Cursor Change: Contains a pop-up palette that lets you select a different cursor that will be displayed whenever it passes over a Mouse Bay. You can create custom cursors for your application in the Cursor Library. For more information about the Cursor Library and creating custom cursors, see chapter 4.

All other Input Types

Prompt with Last Answer: Used primarily when a question is repeated. Causes the answer field to be filled in with the last answer entered for that question. In the case of Radio Buttons and Check Boxes Inputs, shows the last button or boxes the user selected.

Tab Activates OK Button: Allows your audience to use the Tab key, in addition to the Enter and Return keys, to dismiss the input window. This is particularly helpful for "on-screen" forms, where your audience can simply tab between fields.

Previous Output Active: Selecting this option will maintain any mouse actions set up with smart sprites on the underlying Output. If this option is not selected, only the Input Presentation window is active.

Default First Radio Buttons (Radio Buttons Input)**:** Causes the first radio button to be selected when the input window opens. Without this option selected, the Radio Buttons Presentation window will be presented with no radio buttons selected, requiring the user to select at least one option.

Ignore Capitalization (Enter Text Input)**:** Causes capitalization to be disregarded when determining if an answer is correct. Unless capitalization is critical for a correct response, it is usually best to select this option,

Ignore Punctuation (EnterText Input)**:** Causes punctuation to be disregarded when determining if an answer is correct. Unless punctuation is critical for a correct response, it is usually best to select this option,

Restrict to Currency (Enter Numbers Input)**:** Requires the answer given include a dollar sign and decimal in order to be judged correct – **if that's the format used in the Answer Field**. Unless the currency format is critical for a correct response, it is usually best to select this option.

Ignore Number Format (Enter Numbers Input)**:** Requires the answer given include commas every three digits – **if that's the format used in the Answer Field.** Unless number format is critical for a correct response, it is usually best to select this option.

Speak Question: Causes the question text to be spoken out loud using MacinTalk. Selecting this option activates the **Speaker** button, which lets you select from the installed MacinTalk voices. Note that in order to use this option, your audience must have both MacinTalk and the selected voice installed on their computer. If the selected voice is not installed, onViz will read the text using the default voice, Norman.

Tip

If speaking text is critical to your application, make sure you have selected this as a requirement in the Project Attributes Runtime dialog. This feature is not currently supported on the Windows operating system.

Scoring

No Time Limit: Allows your audience to take as much time as needed to respond with their input. If unchecked, activates a timer option that lets you designate a time limit for user response, and to specify a default Route if the user goes over the time limit. The timer starts as soon as the input is entered, and is reset with every entry to the input. If this option is used with the Check Boxes Input, the default Timeout Route is always Exit 3.

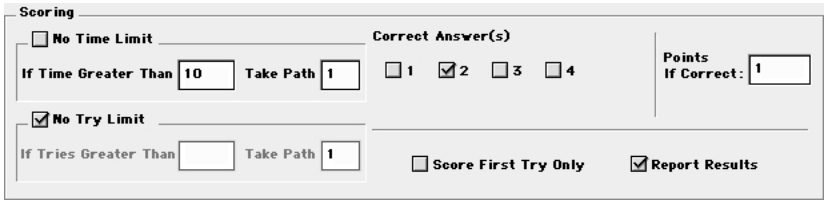


Figure 9.14: The Input InfoCenter's Scoring Panel.

No Try Limit: Allows your audience to respond as many times as necessary until they enter the correct answer. If unchecked, activates an option that lets you specify a maximum number of tries for the user, and to specify a default Exit Path if the user exceeds that number. The number of tries is cumulative according to how many times the input has been entered. You can use a calculator to change this value by setting the **Input Number of Visits** variable equal to the desired number. Also, this value can be reset by first flowing the application through a Calculator with its **Reset Input** value equal to 1 before entering the Input.

If the No Try Limit option is used with the Check Boxes Input, the default try limit Path is always Exit 4.

Correct Answer(s): Lets you designate which of the possible answers are correct. Any number of answers may be designated as correct. The number of check boxes in this field is determined by the **Number of Answers** dropdown list. A correct answer can also be designated in the Answers tabs, at the right-hand side of the **Enter Answer** field. The answers designated as **Correct** are available as Input Variables – **Answer Expected**. For more information on variables, see chapter 13.

Points If Correct: Specifies the number of points to be awarded if the correct answer is chosen. You must designate at least one **Correct Answer** in order for question points to be awarded. Points are accumulated each time a correct answer is entered, unless the **Score First Try Only** option is selected.

Score First Try Only: If selected, points for a correct answer are only rewarded if the question is answered correctly on the first try.

Report Results: Causes statistics such as user response, time to answer, and number of tries for this Input to be recorded in the course report.

Make Default:

After setting all of your Input's attributes, press this button to cause all subsequent Input placements to have the same attributes. This option can save a lot of time for repetitive State placements. To change the default setting, simply set up another Input and make it the default.

Tip

Special considerations using these limit options: Whenever possible, create one more exit than the number of answers for your question. This way the exit can be used exclusively for a limitation exit. If your question requires all 10 answer fields, you can still use limitation exits, but you will also need to use a conditional route to determine whether the route is based on user input, or limitation.

Answers 1–5, 6–10 Tabs

The number of answer tabs active depends on the number of answers selected from the Attributes tab's **Number of Answers** dropdown list. If five or less answers are chosen, only the Answers 1–5 tab (Figure 9.15) will be active. If six or more answers are chosen from the **Number of Answers** dropdown list, both Answers tabs will be active.

Enter Question

This field contains the text that will be displayed in the Input's Presentation window. This field supports variable substitution, so the question can be customized to contain information and statistics generated by the user.

Because this field can hold only 127 characters, make sure the variable being substituted does not contain more than this limit or it will not be displayed in its entirety.

Tip

To add a hard return to the Enter Question field, hold down the Command key while pressing the Return key. If the Command key is not held down, pressing the Return key selects the **OK** button, closing the InfoCenter window.

Enter Answers

Contains all the possible responses to the question entered in the **Enter Question** field. The **Enter Answers** field varies slightly depending on options selected in the Attributes tab, and the number of fields depends on the number of answers selected from the **Number of Answers** dropdown list. Each of the **Enter Answer** fields can hold up to 63 characters.

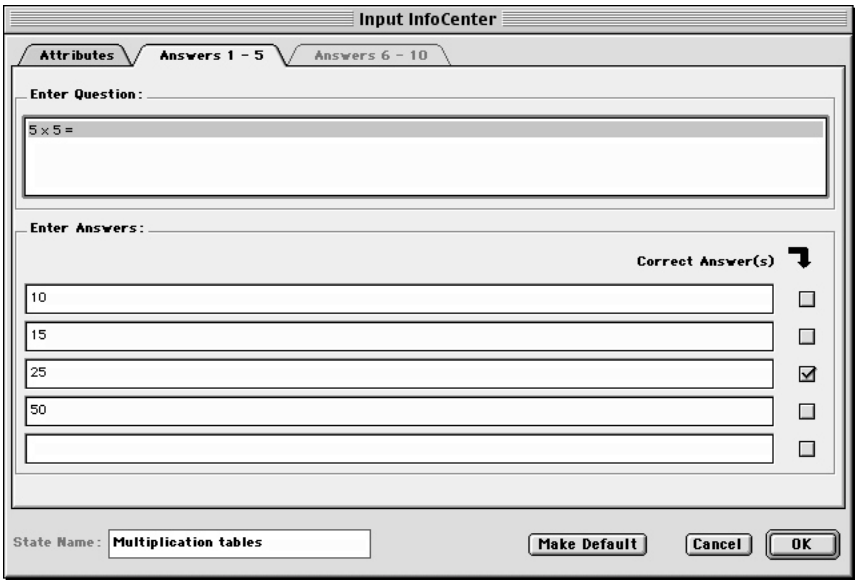


Figure 9.15: The Answers 1-5 tab of an Input InfoCenter.

While in the Application Map work area, you can conveniently view an Input's Answer field text by clicking its Exit Points (Figure 9.16). The Mouse Bays Input Type is an exception to this rule, however, as it has no Answer text.

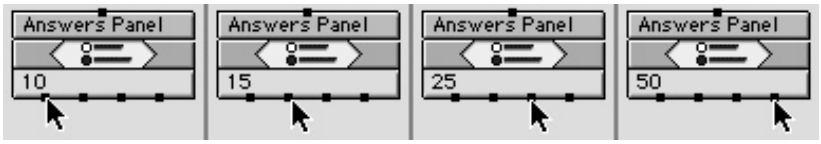


Figure 9.16: Clicking an Exit point will display the corresponding Answer text. In this example, Exit point 1 = 10, 2 = 15, 3 = 25, and 4 = 50.

The **Enter Answers** fields are also affected by the choice of **Input Type**. If either the Radio Buttons or Check Boxes Input types are selected, the **Enter Answer** fields appear as simple edit fields where the text for the answers is typed. If either of the keyboard-based input types is selected, the **Enter Answers** fields change to contain three components:

1. **Comparison Type**
2. **Matching String fields**
3. **Combiner**

The function of these components is to evaluate, or parse, a user's answer for correctness. Because the keyboard-based input types, Enter Text and Enter Numbers, allow your user to enter free-form responses, onViz uses these fields to let you set up judging criteria to determine if the answer is correct. You can craft a series of criteria, called Matching Strings, to take into account a wide range of possible user replies.

The user's reply is parsed against the Matching Strings using Comparison Types such as **Exact Match**, **Doesn't Match**, **Has Word**, **Doesn't Have Word**, etc, which let you determine

what the user typed in and how to route the application to give feedback. If the **Has Word** Comparison Type is chosen, and the user's reply contains the same word as that found in the Matching String, then the reply will be parsed and determined to be correct. If a Combiner (**And**/**Or**) is used, the reply must be parsed against both Matching Strings in order to be found correct.

The comparison operators are also affected by the type of Input chosen. If the Enter Text Input type is chosen, the comparison type operators will be those that affect words and letters. If the Enter Numbers Input type is chosen, the comparison type operators will be those that affect numerical operations.

See the following sections on Parsing Text and Parsing Numbers for more information on using comparison types, matching strings, and combiners.

Correct Answer(s)

The **Correct Answer(s)** check boxes reflect settings from the Attributes tab's **Correct Answer(s)** check boxes, found in the **Scoring** section. All or none of the check boxes may be selected. Selecting the correct answer from this checkbox will update the Attributes tab's **Correct Answer** field.

Input Presentation

Opening an Input Presentation window is just like clicking the InfoCenter's **Position** button. They both open a positioning window that is used to determine the input window's screen position and size. The first time the Input Presentation window is opened, the Background Output dialog appears, prompting you to choose an Output that will serve as the background for positioning this input window. After choosing the background and clicking **OK**, you can relocate and resize the positioning window as desired. Use the locator handle in the upper left-hand corner of the positioning window to move it around. Use the size handle in the lower right-hand corner to change the positioning window's size. When resizing the positioning window, be sure to that you do not make it too small to display its answers or its **OK** button.

After moving and resizing the positioning window, close the window and return to the Application Map by either clicking **OK**, pressing the Enter or Return keys, or choosing Close Window from the Edit menu. Note: holding down the command key while hitting the period closes the positioning window without saving changes. To change the Output chosen to be the positioning background, open the State's InfoCenter and click the **On Output** button to access the Background Output dialog.

When the Mouse Bays Input type is chosen, the Presentation window is used to position the State's Mouse Bays. However, you can also position the bays using the **Position** and **On Output** buttons, accessed in the State's InfoCenter. See below for instructions on positioning a Mouse Bays Input's Mouse Bays.

Tip

There may be times when you resize the positioning window to purposely hide the **OK** button. With the **OK** button hidden, your user can still press the Enter or Return keys to close the positioning window and return to the InfoCenter.

Parsing Text

onViz' Enter Text Input allows users to respond to questions with free-form, or essay type, answers. When a user responds to questions in this type of Input, a text parser searches the response for certain key words and phrases in order to judge whether or not it is correct. The words and phrases used to evaluate a user's response are called matching strings, and are set up in the **Enter Answer** fields of the Enter Text InfoCenter.

Each Answer field is made up of two comparison type dropdown menus, two matching string fields, and a combiner dropdown menu. The comparison types determine how precisely the user's response is compared to the matching string, and include:

- Exact Match – User's response must exactly match the matching string.
- Doesn't Match
- Has Word – User's response must contain this word.
- Doesn't Have Word
- Has String – User's response must contain this string of words.
- Doesn't Have String

The combiner dropdown menu lets you join two matching strings with **AND** or **OR**. If a combiner is used, the user's response is compared to both of the Answer field's matching strings. If **AND** is used, both comparisons must be true for the answer to be correct. If **OR** is used, either of the comparisons, or both, can be true for the answer to be correct.

Wildcard characters are supported in the matching strings. Substitute a question mark (?) for any single character in the matching string, or an asterisk (*) for multiple characters. For instance, if the matching string reads **Lin???n**, the responses **Lincoln**, **Linkoln**, or **Linconn** will all match. **Lincon** will not be a match, as it does not account for all the question marks. If the matching string reads **Lin***, the responses **Lincoln**, **Linkoln**, **Lincon**, or **Linkletter** will all match, as the asterisk stands in place of any number of characters.

onViz can be set to ignore capitalization and/or punctuation. Both of these setting are found in the Input InfoCenter.

Because the user is responding in a free-form fashion, it is best to use several Answer fields in order to anticipate a wide range of responses. When evaluating the user's response, onViz goes through the Answer fields sequentially; as soon as a match is found, onViz takes that answer's exit path. Because of this sequential evaluation, you should place your most precise matching string first, then follow it with those that take into account the variety of possible responses. Always include one blank Answer field that can be followed if the user's response matches none of the answers.

This functionality can be used to create multiple feedbacks based on the user's response. If the user's response is an exact match, you might give feedback proclaiming "Your answer is perfect." If the response matched an Answer field containing a wild card, you might give feedback saying "Check your spelling." If the user's response matched none of the possible answers, you might provide feedback suggesting the material be read again.

Figure 9.17 shows an example question that is seeking a short, two-part answer. The question

is "What two pieces of safety gear must you wear at all times?" The answer must take into account a variety of possible responses, such as different wording, incorrect spelling, and not including both parts of the answer.

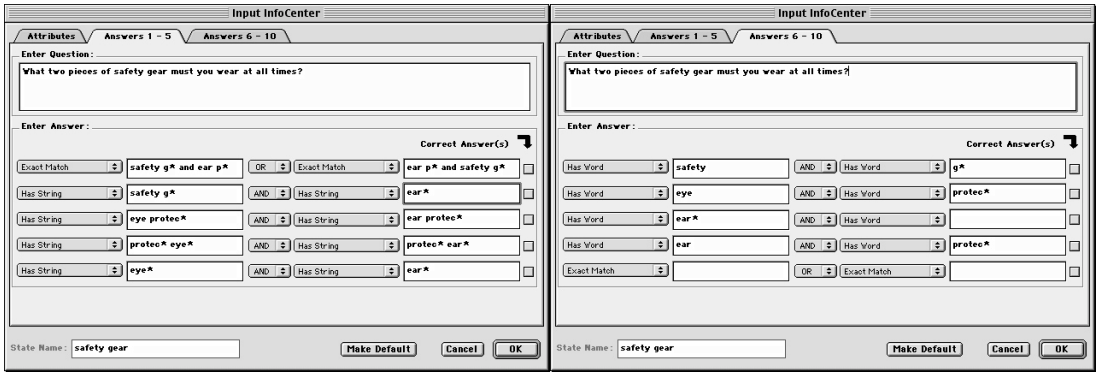


Figure 9.17: The two Answer tabs for an Enter Text Input. Notice that the last Answer field is left blank; this Exit will be taken if the user's response does not match any of the other Answer fields.

The first Answer field is an attempt to anticipate the most common response, "safety glasses and ear plugs." The asterisk after the letter "g" allows the user to answer with either "goggles" or "glasses," while the asterisk after the letter "p" allows the user to answer with either "plugs" or "protection." The **OR** combiner lets the user respond with either piece of equipment first. Note that when **Exact Match** is used as the comparison type, the AND combiner cannot be used, as the answer can only be one exact thing, it cannot be exactly one thing and exactly something else.

The second Answer field allows the user to structure the response in some other way, such as a complete sentence. As long as the response contains the matching strings, the comparison will be true. The asterisk immediately following "ear" allows the user to respond with "earplugs," "ear plugs," or "ear protection."

Answers three and four simply allow the user to respond with different terminology. The word order is reversed in the two answers so that the user can respond with "eye protection" or "protective eye gear." The asterisks allow different forms of the words to be used, such as "protective," "protection," "eye," or "eyewear."

Answer five anticipates any other response; as long as it includes some variation of the words "eye" and "ear," the comparison will be true. Answers six through nine cover many of the same variations in spelling and terminology, but anticipate the user only responding with half of the answer. Answer field ten is an escape exit, used in the event that none of the comparisons are found to be true.

Although you can provide separate feedback for every answer, you can also narrow it down to only three: if exits one through five are taken, the feedback could read "Correct." If exits six through nine are taken, the feedback could read "Please list two pieces of safety equipment." If exit ten is taken, the feedback might read "Your answer is incorrect, please try again."

Parsing Numbers

The Enter Numbers Input allows users to respond to questions in a free-form manner, much like a fill-in-the-blank type question. onViz uses a numeric parser to compare the response against Matching Values set up in the **Enter Answer** fields of the Enter Numbers InfoCenter (Figure 9.18).

Each Answer field is made up of two comparison type dropdown menus, two matching value fields, and a combiner dropdown menu. The comparison types analyze the response's value relative to the matching values, and include the numerical operators:

- Equal To
- Not Equal To
- Less or Equal To
- Greater or Equal To
- Less Than
- Greater Than

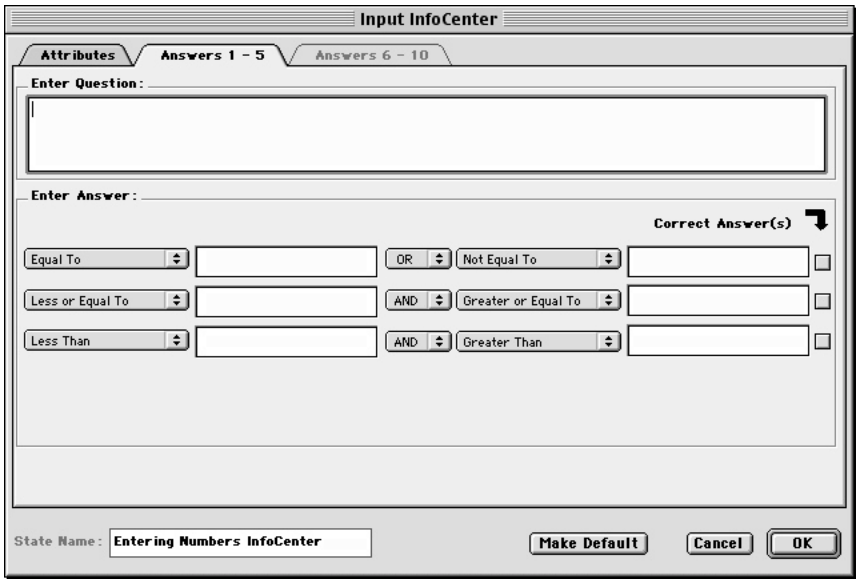


Figure 9.18: The Enter Numbers Input Answers 1-5 tab.

The combiner dropdown menu lets you join two matching values with **AND** or **OR**. If a combiner is used, the user's response is compared to both of the Answer field's matching values. If **AND** is used, both comparisons must be true for the answer to be correct. If **OR** is used, either of the comparisons, or both, can be true for the answer to be correct.

If the Equal To comparison type is used, the AND combiner cannot be used, as the response cannot be equal to two different numbers. If you are asking a two-part question and need two different numbers, such as dates, to be the answers, use an Enter Text Input with the **Has String** comparison type and the **AND** combiner (Figure 9.19).

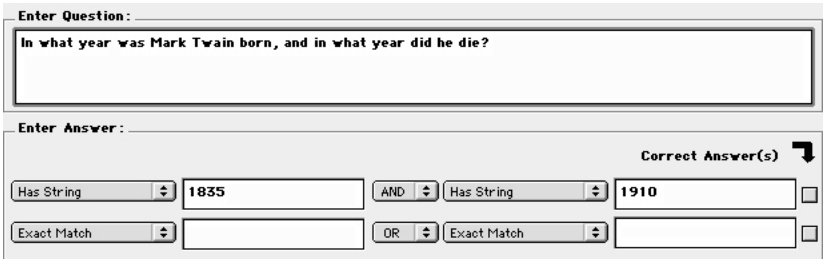


*Figure 9.19: Using an Enter Text to parse a two-part question involving numbers. Using **Has String** as a comparison type allows **AND** to be used as a combiner.*

When evaluating the user's response, onViz goes through the Answer fields sequentially; as soon as a match is found, onViz takes that answer's exit path. Because of this sequential evaluation, you should place your most precise matching string first, then follow it with those that take into account the variety of possible responses. Always include one blank Answer field that can be followed if the user's response matches none of the answers.

Figure ? contains an example question that asks for the value of pi carried to at least two decimal points. The first four answer fields demonstrate how a question may have more than one correct response; each of these fields contain a correct value of pi when it is carried to two to five decimal points. A true comparison for any of these fields results in the user being routed to “Correct” feedback.

The fifth answer field, using the Greater or Equal To and Less or Equal To Comparison Types and the combiner AND, matches any attempts by the user to round pi off to the nearest decimal point. If a true comparison is made for this answer, the user is routed to feedback that states “Please do not round off your answer.”

The sixth and seventh answer fields use the Less Than and Greater Than Comparison Types, which route the user to feedback stating “Too low” and “Too high,” respectively.

The seventh answer field is left blank to account for any unanticipated responses, and will route the user to “Incorrect” feedback.

Parsing Check Boxes Input

The Check Boxes Input differs from the other Inputs, in that the user can be required to select more than one response for a correct answer. This feature requires the user's answer to be parsed and routed in a different manner than the Radio Buttons Input. In order to be judged correct, the user must choose all of the answers that have been designated as correct in the **Scoring** section's **Correct Answers** field.

The Check Boxes Input will always have four Exits, regardless of the number of answers options it contains.

- Exit 1 will be taken if the user chooses all the correct answers and no others.
- Exit 2 will be taken if the user either did not choose all the correct answers, chose one or more wrong answers, or both.
- Exit 3 will be taken if the user does not answer the question within the time period specified in the **Scoring** section's **Time Limit** field.
- Exit 4 will be taken if the user attempts the question more times than are designated in the **Scoring** section's **Try Limit** field.

For the example in Figure ?, check boxes one, two, four, and five must be chosen, and no others, for the answer to be judged correct; if so, Exit 1 is taken. If any other combination of answers is chosen, the answer is judged incorrect, and Exit 2 is taken. Exit 3, the "time out" exit, is taken if the user does not respond within the time set in the InfoCenter's Scoring section (thirty seconds in this example). Exit 4 is the "try limit" exit, and it is taken if the user answers the question incorrectly a certain number of times, as set in the InfoCenter's Scoring section (this example allows three tries).

Input Variables

Read Only

- Answer Expected (InputAnswerExpected{State}) – This text variable represents the text from the answer field(s) designated as **Correct Answer(s)** in an Input's InfoCenter.

  Since this information can only be changed by updating the text in the InfoCenter, it is read only.

- **Exit Path Expected InputExit Expected{State}) – This text variable represents the Exit Path(s) for the answer field(s) designated as Correct Answer(s) in an Input's InfoCenter.**

  Since this number can only be changed by updating the Correct Answer designation in the InfoCenter, it is read only.

- Percent Score (InputPercentScore{State}) – This numeric variable represents the calculation of a user's actual score for an Input divided by the total possible score for the Input, and is updated each time the Input is exited. The Input must have a designated **Correct Answer** and a value in the **Points if Correct** field in order for this statistic to be tracked.

  Since this information is a calculation based on Total Score statistics, it is read only.

- Percent Correct (InputPercentCorrect{State}) – This numeric variable represents the calculation of the number of times a user selected an Input's correct response divided by the number of times the Input was entered, and is updated each time the Input is exited. The Input must have a designated **Correct Answer** in order for this statistic to be tracked.

  Since this information is a calculation based on Total Correct statistics, it is read only.

- *n* Answer Selected (Answer*n*Selected{State}) – *n* = answer number (1-10). These numeric variables are used with Check Boxes Inputs. There is a unique "Answer Selected" variable associated with each of the 10 possible answer fields. If the user selects a specific check box answer, its "Answer Selected" variable is equal to 1; if the check box is not selected, its "Answer Selected" variable is equal to 0.

Write Only

- Reset Input (ResetInput{State}) – Flowing the application through a Calculator that contains an equation setting the value of this variable equal to 1 resets all variables associated with designated Input.

Read/Write

- Answer Entered (InputAnswer{State}) – This text variable represents the answer entered by the user.

  In the case of Radio Buttons Inputs, this variable represents the text of the selected answer.

In the case of Check Boxes Inputs, this variable represents the text from each selected answer. If more than one answer is chosen, each answer's text is separated by a comma and space.

In the case of Mouse Inputs, this variable represents the Mouse Bay number the user clicked.

This variable's value can be updated using a Text Calculator. An example of its use in an application would be to preset an Input's **Prompt With Last Answer** option, which will fill in the answer field with the value set in the Calculator.

- Time to Answer (InputTime{State}) – This numeric variable represents the amount of time the user has spent in the designated Input, and is reset each time the Input is entered.

  This variable's value can be updated using a Numeric Calculator. An example of its use in an application would be to clear the time to answer without resetting other Input statistics.

- Total Score (InputTotalScore{State}) - This numeric variable represents the user's score for the Input. Unless the **Score First Try Only** option is selected, this number is cumulative with each time the user enters the Input and answers the question correctly. The Input must have a designated **Correct Answer** and a value in the **Points if Correct** field in order for this statistic to be tracked.

  This variable's value can be updated using a Numeric Calculator. An example of its use in an application would be to set this value based on Smart Sprite selections.

- Total Correct (InputTotalCorrect{State}) – This numeric variable represents the number of times the user has provided a correct answer for the designated Input. Unless the **Score First Try Only** option is selected, this number is cumulative with each time the user enters the Input and answers the question correctly. The Input must have a designated **Correct Answer** in order for this statistic to be tracked.

  This variable's value can be updated using a Numeric Calculator. An example of its use in an application would be to set this value based on Smart Sprite selections.

- Number of Visits (InputVisits{State}) – This numeric variable represents the number of times the user has visited the designated Input.

  This variable's value can be updated using a Numeric Calculator. An example of its use in an application would be to clear the number of visits without resetting other Input statistics.

- Exit Path (InputExit{State}) – This numeric variable represents the Exit number taken from the designated Input.

  This variable's value can be updated using a Numeric Calculator.

- Last Sprite Dragged (LastSpiteDragged{State}) – This numeric variable represents the family number of the most recent sprite dragged by the user.

  This variable's value can be updated using a Numeric Calculator. An example of its use in an application would be to clear this value without resetting other Input Statistics.

- *n* Bay's Sprite (Bay*n*Sprite{State}) – *n* = Mouse Bay number (1-10). This numeric variable represents the sprite family number(s) of sprites that are within the boundary of the designated Mouse Bay. There is a unique "Bay's Sprite" variable associated with each of the 10 possible Mouse Bays.

  This variable's value can be updated using a Numeric Calculator.

Using Input Features

Positioning Mouse Bays for the Mouse Bays Input

When using a Mouse Bays Input, Mouse Bays are positioned on the screen where you want to detect your user's mouse clicks. Create your graphics and position them in an Output first, then follow the instructions below to use the Output as a background for your Mouse Bays.

1. Open the Input's InfoCenter, and select the Mouse Bays **Input Type**.
2. Select the desired **Number of Answers** from the dropdown list.
3. Set the desired **Runtime Features**. See above for an explanation of the features found in the Input InfoCenter.
4. Click the On Output button to open the Background Output dialog.
5. Select an Output for positioning the Mouse Bays, then click **OK**.

   The Background Output dialog closes and the Presentation window opens, with the chosen number of Mouse Bays in the upper-left corner.
6. Click in the middle of a bay and drag it over the desired areas. Resize these bays as necessary using the handles at the corners of the box (Figure 9.20).

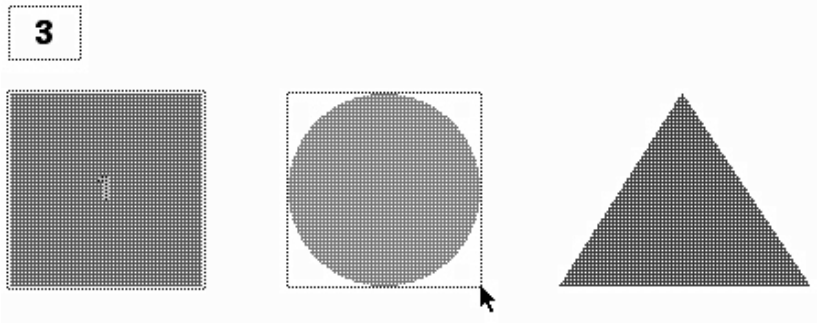


Figure 9.20: Positioning Mouse Bays over a graphic. The Mouse Bay is already positioned over the square (note the dotted line around the square), and is being resized around the circle. The number three Mouse Bay, in the upper-left corner, has yet to be positioned over the triangle.

7. When the Mouse Bays are satisfactorily placed, close the Presentation window by choosing **Close Window** (⌘ W) from the **Edit** menu. Holding down the Command key and pressing the Period key will close this window without saving any repositioning.

Creating On-screen Forms with Inputs

1. Create a new onViz project, add the following States to the Application Map, and use Paths to connect them together in the order listed: Start, Design Output, Radio Buttons Input, and Stop. For more information on adding States, see chapter 3, the Application Map.
2. Name the Design Output "Form Graphics," and name the Radio Buttons Input "Feedback."

   Your Application Map should resemble Figure 9.21.

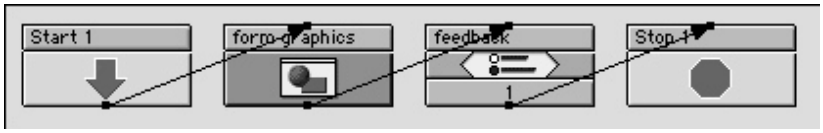


Figure 9.21: The first four States of the Application Map, connected with Paths, and named according to step two.

3. Add four more States to your Application Map, and use Paths to connect them together in the order listed (do not connect them to the previous set of States): Enter Text, Enter Numbers Input, Enter Text, and Return.
4. Name the new Inputs in the following order: Name, Age, and Email.

   Your Application Map should now resemble Figure 9.22.

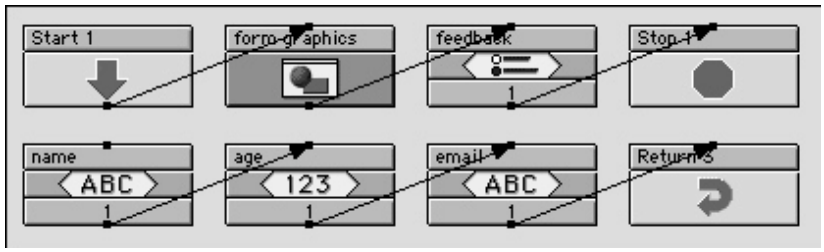


Figure 9.22: The Application Map, now with four more States. Note that the Inputs are named according to step four, and that the bottom set of States are not connected to the top set.

5. Open the Design Output Presentation (⌘ E), and create a background for your form (hold down the command key and double-click in the Design area to open the Image Editor and start working on the form background).

   The background will consist of six object text boxes: three for the field titles and three for the actual fields. Your form will look nicer if all the fields are precisely aligned, so turn on the grid while creating and positioning the text boxes. For more information on Output States, see chapter 6. For more information on using the Image Editor, see chapter 7.
6. Create the first three object text boxes; they should include the following text: Name, Age, and Email.

   The Image Editor work area should resemble Figure 9.23.

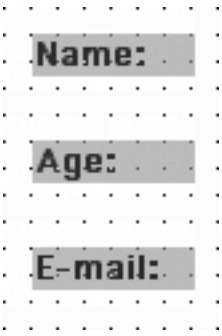


Figure 9.23: The background for your form, as it appears so far in the Image Editor. These three text boxes will be the labels for the form fields.

7. Begin creating the second three object text boxes; each one will contain a variable as its text. Create the first text box; use the color black for the font and a white background. Hold down the Option key and click in the text box to open the Select a Variable dialog. Choose the Input variables tab, select the Input State called "name," select the

Input variable **Answer Entered (Text)**, then click **Select**. Line this text box up next to the "Name" text box. For more information on using variable substitution, see chapter 13, Variables.

The two "Name" text fields should resemble Figure 9.24.

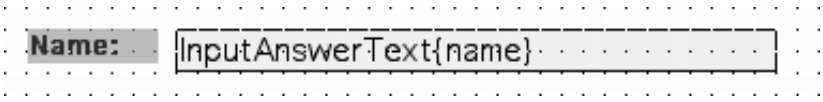


Figure 9.24: The two text boxes that comprise the "Name" section of the form's background. The text box on the left is the label for the form field, while the text box on the left is the blank form field.

8. Create the final two object text boxes as described in step seven. This time, when selecting the Input State for the **Answer Entered (Text)** variable, choose the Input named "age" for the second text box and the Input named "Email" for the final text box.

   The work area should now resemble Figure 9.25.

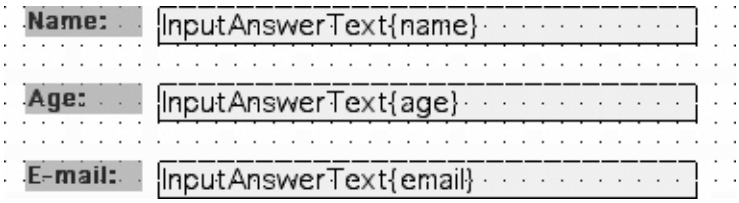


Figure 9.25: The completed form background, composed of three form fields. Note that each field has a label and a field with a variable. During runtime, when the user completes the form, the variables will be replaced with information provided by the user.

9. You now need to create three sprites, which will be placed over the background you just finished. To the user, these sprites will appear to be the form fields. What actually happens, however, is that these sprites, when clicked, will bridge to their respective Inputs, which is where the user will fill in the "form."

   The sprites need to be exact replicas of the variable fields you created in steps seven and eight, so the easiest way to make them is to copy each of the variable fields and paste them into three new image files.

   First copy the field containing the name variable, then use the Image Control options at the bottom of the screen to create a new image. In the Image Attributes dialog, give the image a meaningful name then click **OK**. Paste the copied field into the Image Editor work area (Figure 9.26). Repeat this procedure to create image files for the Age and Email fields.

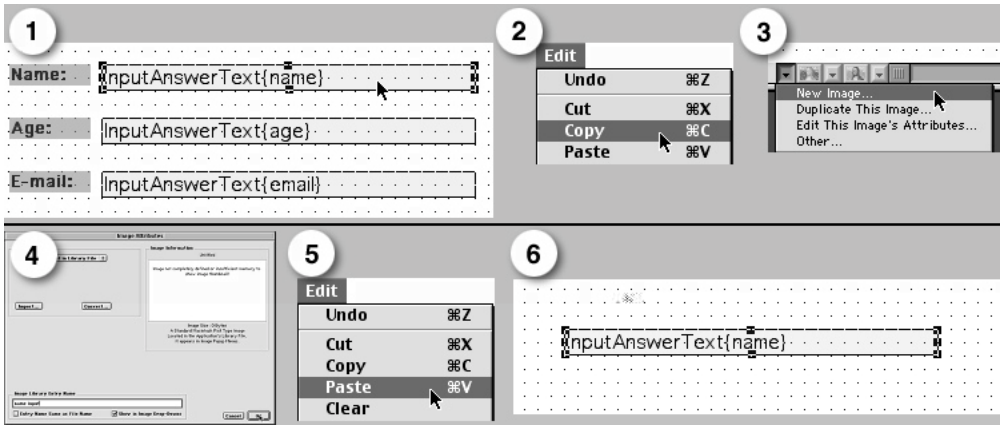


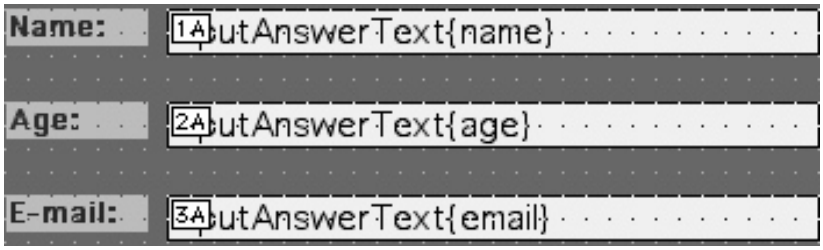
*Figure 9.26: Creating three sprites by copying the background's form fields. 1) From the form's background image, select the field containing the name variable. 2) From the **Edit** menu, choose **Copy** (⌘C). 3) From the Image Control options at the bottom of the screen, select **New Image**. 4) In the Image Attributes dialog, name the image then click **OK**. 5) From the **Edit** menu, choose **Paste** (⌘V). 6) The new image in the Image Editor.*

10. Reopen the Design Output Presentation (⌘ E). The background you finished in step eight is visible in the Design Window. To use the three images you created in step nine, you first need to add them to the Sprite Library. Once they are in the Sprite Library, they can be added to the Design window as Sprites.

    Select box 1A in the Sprite Library (the box will become outlined in red), then use the Sprite Selection dropdown menu to select the image with the name variable (Figure 9.27). Repeat for boxes 2A and 3A.

11. Choose Add Sprite from the Animation menu, or simply drag the sprite from the Sprite Library to the Design window. Reposition the sprite until it covers the name variable field in the background. Repeat this procedure for the other two images. For more information on the Sprite Library and adding and configuring sprites, see chapter 8, Animation Support.

    Your Design window should resemble Figure 9.28.



12. To make the three new sprites function as Smart Sprites, you need to assign mouse actions to them. Select box 1A in the Sprite Library, and, in the Mouse Actions section of the Sprite Attributes palette, select **Mouse Up Actions**. From the **Mouse Up Actions** dropdown menu, select **Bridge To New Bridge**. In the Setup Bridge Target dialog, select **State in this onViz File** from the **Bridge Type:** dropdown menu, then select the "Name" Input State from the list of States. Be sure to select the **Return After Bridge** check box before clicking **OK**. Repeat this procedure for sprites 2A and 3A.

    For more information on setting up Bridge Targets, see chapter 5, Flow States, and the Bridge Target section of chapter 4, onViz Libraries.

    Note that when you select **Mouse Up Actions**, the **Mouse Down Actions** section appears also; be sure to deselect the **Inv** (invert) check box in this section.

    After setting up your **Mouse Up Actions**, the Sprite Attributes palette should resemble Figure 9.29.

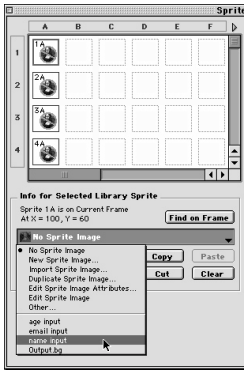


Figure 9.27: Adding the images created in step nine to the Sprite Library. Select a box in the Sprite Library, then choose the desired image from the Sprite Selection dropdown menu.

Figure 9.28: The three new sprites in the Design window. The sprites are visible by their labels-1A, 2A, and 3A. Note the field labels in the background; the variable fields should be completely obscured by the sprites.

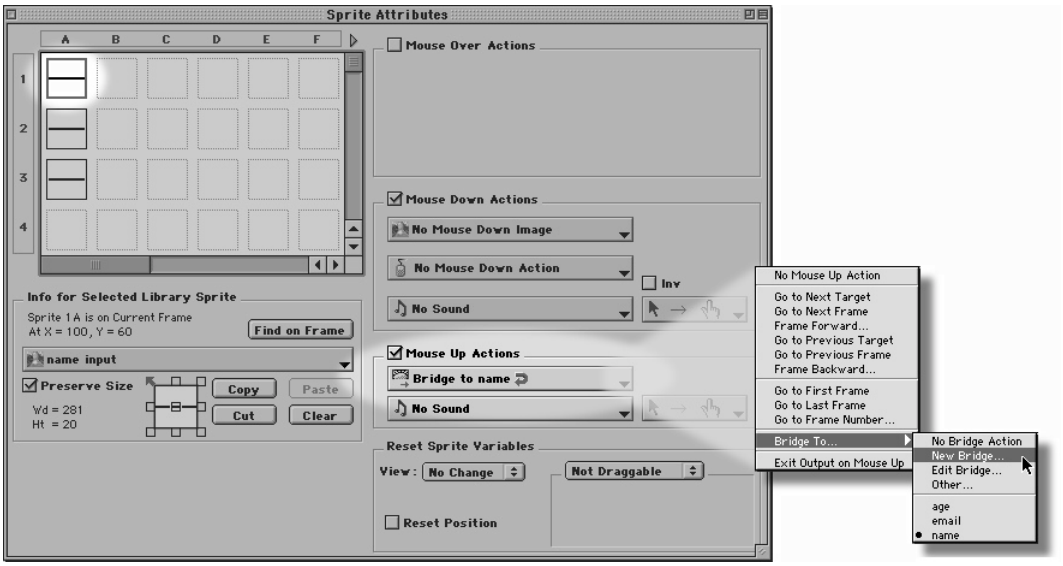


*Figure 9.29: Assigning mouse actions to your sprites. Select box 1A in the Sprite Library, then select **Mouse Up Actions**. From the **Mouse Up Actions** dropdown menu, select **Bridge To New Bridge**. In the Bridge Target Editor, select **State in This onViz File**, then choose the "Name" State from the list.*

13. Use the Frame Control palette to set this frame's synchronization, which allows the frame to remain on the screen while the user fills out the form. From the Synchronization dropdown menu, select **Wait for Smart Sprite** (Figure 9.30).

*Figure 9.30: Selecting **Wait for Smart Sprite** from the Frame Control palette's **Synchronization** dropdown menu.*

14. The Inputs now need to be positioned over the Output background. With the Inputs positioned properly, your users will not realize that they are bridging between three different Inputs as they fill out the form.

    Select the "Name" Input State, and open its InfoCenter window (⌘ I). Click the **On Output** button, and when asked to **Select Output for Background**, choose "Form Graphics" and click **OK**. Use the locator and size handles to precisely resize and position the Input over the form field in the background, and press the Return or Enter key to save the new size and position (Figure 9.31).

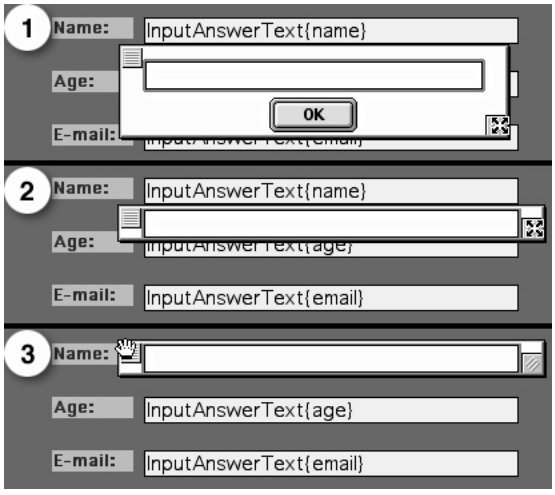


Figure 9.31: Positioning the Inputs over the Output named "Form Graphics." 1) Use the size handle to hide the OK button and make the Input as short as possible. 2) Use the size handle to make the Input the same width as the sprite. 3) Use the locator handle to position the Input over the Output's form fields. Press Enter or Return to save the Input's new size and position.

15. Repeat step fourteen for the other two Inputs, "Age" and "Email."

16. Before closing the Input InfoCenter, select the **Tab Activates OK Button** option, found in the InfoCenter's Runtime Features (Figure 9.32). Selecting this option allows your user to tab between fields in the form. Click **OK** to close the InfoCenter window.

When your users run the application, they'll click in the Name field to start filling in the form, which will bridge them to the "Name" Input. When they press Tab to go to the next field, they're actually closing the first Input, which routes them to the second Input. When they have finished filling in all the fields, their responses are displayed through variable substitution by the sprites.

Figure 9.32: Found in the Runtime Features section on the Input InfoCenter, the Tab Activates OK Button option allows your users to Tab between fields on the form.

Working with Check Boxes Inputs

Section not yet complete.

Creating and Checking Draggable Sprites

Section not yet complete.

Chapter 10

Multimedia States

Covered in this Chapter:

- Using Multimedia States
- Play Sound InfoCenter
- Play Movie InfoCenter
- Overlay Image InfoCenter

Using Multimedia States

Generally, an Output is used to play sounds, movies, and display images. However, by sending application flow through a Multimedia State, you can play sounds, movies, and/or display images at any point in your application. Use the Multimedia State to play looping background music, add to the media being presented in an Output, and display or set up items that can be quickly accessed with a Bridge and Return from any point in the application.

For instance, you could set up Random Groups for feedback and, based on user response, bridge from an Output to the appropriate group to simultaneously display a message and play a sound, then return to the same Output display. Or, perhaps a presentation on the rain forests could include insect noises that looped continuously in the background. Multimedia States, in combination with a Bridge and Return, are a good way to set up glossary items that can quickly be accessed from any point in the application (Figure 10.1). The image will appear on screen without replacing the current display. When the user dismisses the glossary dialog, the application returns with the active Output presentation displayed so the user can continue.

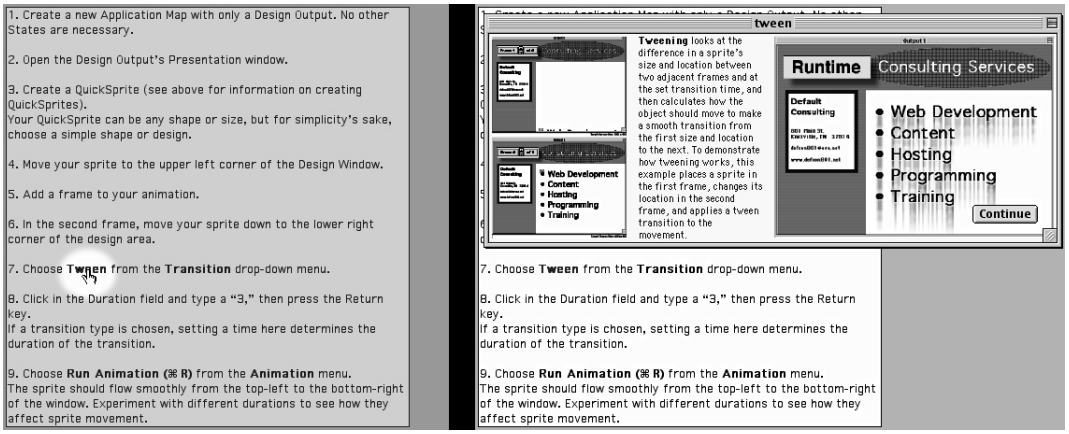


Figure : Changing the location of an overlay image inside its positioning window. The dotted line shows an outline of the image as it is being dragged, while the information bar at the bottom of the window displays the image's X and Y coordinates, as well as how far is has been moved.

There are three actions that Multimedia States can initiate:

- Play/Stop Sound
- Play/Stop Movie
- Overlay Image

The InfoCenter for each of these types of Multimedia States is structured in essentially the same manner, divided into two segments (Figure 10.2). The top segment is the **Clip List**, and contains eight slots. When the Multimedia State is run, the clips are presented in the order in which they appear in the list. Note that you can launch all three media types simultaneously, and you can have more than one sound or movie file playing. However, only one overlay image can be displayed at any given time.

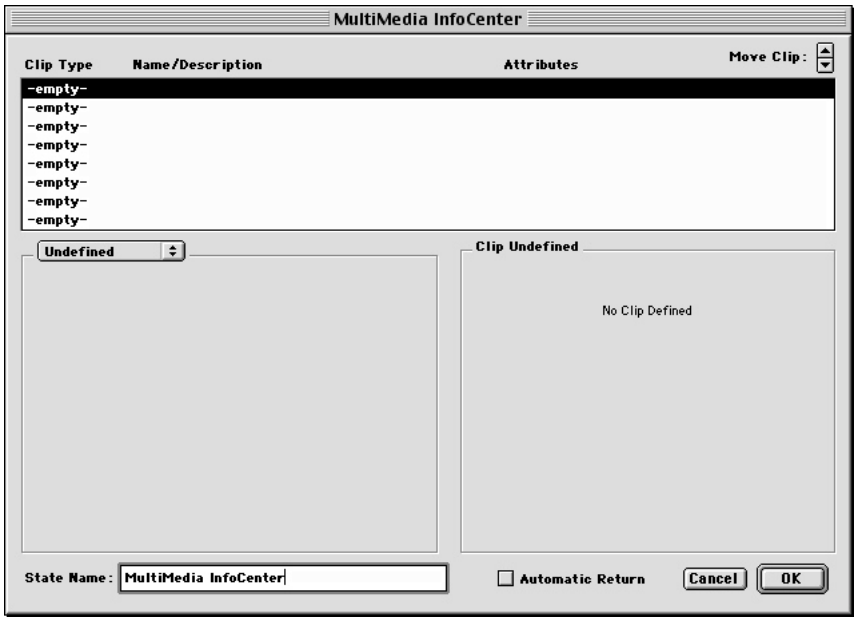


Figure 10.2: The Multimedia State InfoCenter. At the top of the dialog is the Clip List, while the bottom section of the dialog is for choosing the clips.

The **Move Clip:** Up/Down arrows allow you to change the list order of the clips and the order in which they will be launched.

The bottom segment of the dialog is used to choose the clips that will appear in the list. Its left-hand side is used for selecting the clip and adjusting its attributes, while the right-hand side displays information about the chosen clip.

The first time the InfoCenter is opened, the top line in the **Clip List** is selected, and the **Clip Type** dropdown menu reads **Undefined**. Choosing a clip type from this dropdown menu reveals the **Clip Selection** dropdown menu; this menu is for selecting the onViz library entry (from the Image, Movie, or Sound Libraries) that will appear in the **Clip List** and be launched when the application flows through the Multimedia State (Figure 10.3).

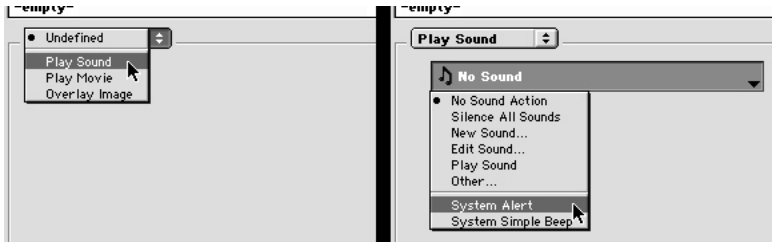


Figure 10.3: After a Clip Type is chosen from the dropdown menu (left), the Clip Selection dropdown menu appears, which is used for selecting the library entry that will appear in the Clip List; note that this menu can also be used to silence all sounds currently playing.

Once a clip is chosen, the right side of the dialog displays the media file attribute settings as set in the library. An attributes panel appears, allowing you to **Override Library Attributes** for use in this Multimedia State. Note that altering a clip's attributes in this panel has no has no effect on its Library entry attributes. The settings in the attributes panel vary slightly depending on the type of clip chosen: if a sound or movie is chosen, this panel includes a **Play** button. The **Play** button is not present if an overlay image is chosen, but the panel changes to include attributes similar to those found in the Output InfoCenter and in the

Frame Control palette. The attributes panel for each type of Multimedia State is discussed below.

The Multimedia State InfoCenter contains a **State Name** field for giving the State a unique, meaningful name. While a State's name can be changed from within its InfoCenter window, it can also be changed without leaving the Application Map. To change a State's name from the Application Map window, click once on the State's name, wait for the Name field to become a black rectangle with a red outline, then type the new name. Pressing Enter or Return, or clicking anywhere outside the Name field, makes the new name take effect.

The InfoCenter also contains an **Automatic Return** option, which functions the same as having it followed by a Return State. If the Multimedia State is used as a Bridge's Target State, checking this option will automatically return flow back to the Bridge's Exit Point, without needing to add a Return State to your Application Map. See chapter 5, Flow States, to learn how to use Return States in your application.

Notice that by checking the **Automatic Return** option the Multimedia State icon no longer displays an Exit Point (Figure 10.4); this is because the **Automatic Return** option always routes flow back to the Bridge State that called it, eliminating the need for a path from the Exit.

Figure 10.4: A Multimedia State without the Automatic Return option selected has an Exit Point, while one with the Automatic Return option selected no longer displays an Exit Point.

The InfoCenter can be accessed by double-clicking the rectangular area in the upper 1/3 of the State's icon.

Unique options for each type of Multimedia State are as follows:

Tip

The Multimedia State InfoCenter can also be accessed by selecting a Multimedia State and then choosing Multimedia InfoCenter (⌘ I) from the **Map** menu.

Play Sound InfoCenter

Options found within the Play Sound InfoCenter (Figure 10.5):

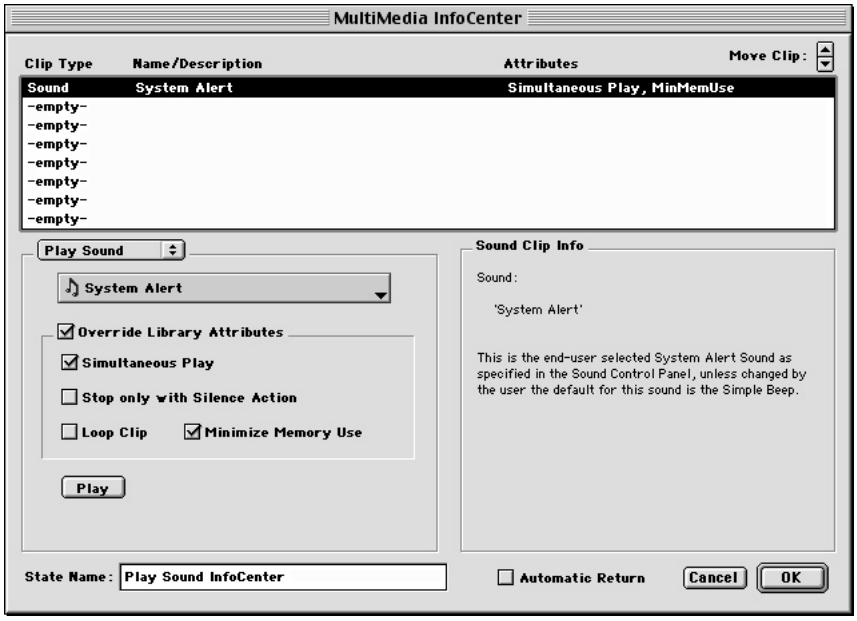


Figure 10.5: The Play Sound InfoCenter.

- **Simultaneous Play** – Allows the sound to play while the application continues to run. Deselecting this option will pause the application until the sound is complete.
- **Stop only with Silence Action** – Once the sound begins playing, it will continue to play until the application encounters a Multimedia State (see Figure 10.3), Output frame (Figure 10.6), or a Smart Sprite with a mouse action from the Sound dropdown list set to **Silence All Sounds**.
- **Loop Clip** – Causes the sound to repeat until the application encounters a Stop Sound Action as described above.
- **Minimize Memory Use** – This option is particularly beneficial when playing large sound files. onViz will store large sound files in small segments. Rather than having to of load the entire sound in memory, onViz can quickly load these segments instead. As the first sound segment nears completion, the second segment is being loaded, resulting in the sound playing with no interruption. This not only minimizes the amount of memory required, but also allows the sound to begin as soon as it is called by a Multimedia State or Output frame.

*Figure 10.6: The **Silence All Sounds** command can be found in the Frame Control palette's Sound dropdown menu.*

Tip

If you use a Multimedia state to play multiple buffered sounds simultaneously, the sound may stutter while getting the next sound byte. If playing multiple sounds, it is better to deselect this option for all sounds that will be playing simultaneously.

Play Movie InfoCenter

Options found within the Play Movie InfoCenter (Figure 10.7):

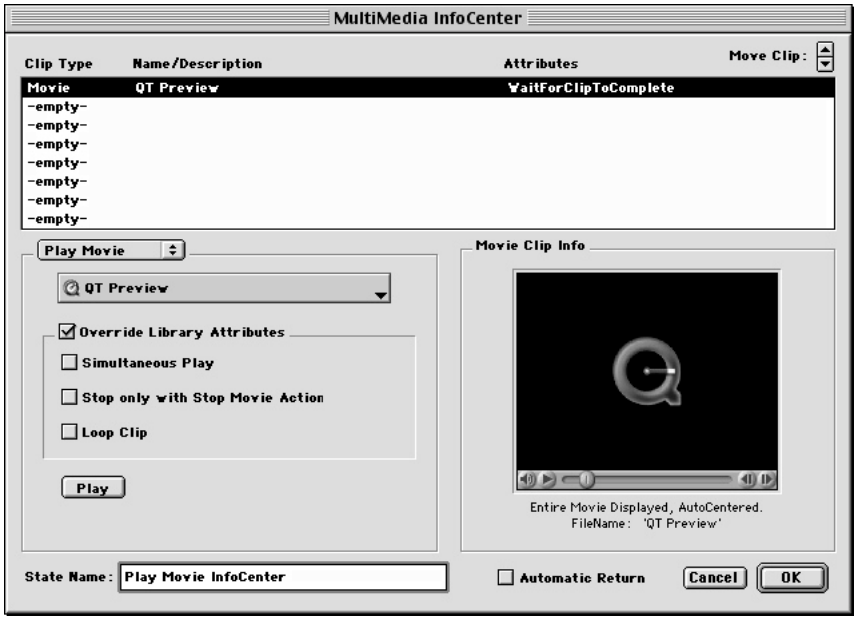


Figure 10.7: The Play Movie InfoCenter.

- **Simultaneous Play** – Allows the movie to play while the application continues to run. This option works well in conjunction with Smart Sprites that act as button controls for the movie. Deselecting this option will show the entire movie before the application continues. Note: when selecting the **Simultaneous Play** option, the majority of the computer's processing power will be dedicated to the QuickTime movie. Animation sequences are not as smooth as they are with no QuickTime movie playing.
- **Stop only with Stop Movie Action** – Once the movie begins playing, it will continue to play until the application encounters a Multimedia State, Output frame, or a Smart Sprite with a mouse action from the Sound dropdown list set to **Stop any Movie Action**.
- **Loop Clip** – Causes the movie to keep repeating until the application encounters an Output frame or Multimedia State set to **Stop Any Movie Action** from the Movie drop down list.

*Figure 10.8: The **Silence All Sounds** command can be found in the Sprite Attributes palette's Sound dropdown menu.*

Overlay Image InfoCenter

Options found within the Overlay Image InfoCenter (Figure 10.9):

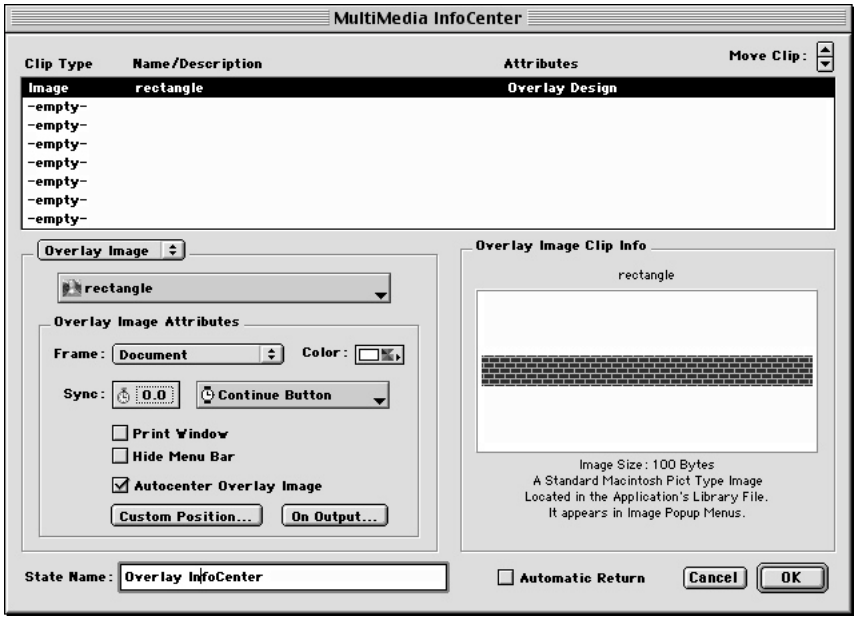


Figure 10.9: The Overlay Image InfoCenter.

Tip

Note: if you choose a color not in the color palette and your application is run in 256 colors, onViz will use the closest color in the application color table for the window color.

- **Frame** – Determines the border style for the Overlay Image window. See chapter 6, Output States, for a description of the various frame types.
- **Color** – Determines the window color for the Overlay Window. Clicking this button opens a palette from which to choose a color for the Overlay window. While in the color palette, you can choose **Color Picker** to pick a window color other than those in the 256-color display palette.
- **Synch** – Application flow will not exit the State until the selected synchronization has been satisfied. Options include typing a specific amount of time for the overlay to display in the **Stopwatch** edit field (Figure 10.10) or making a selection from the **Synchronization** dropdown menu (Figure 10.11). If you have specified an amount of time and no other synchronization is set, the Overlay will display for the designated time before exiting. The **Synchronization** dropdown menu causes flow to be suspended until some other action has taken place. Choices include:
  - **Continue Button** – Adds a **Continue** button in the lower right corner of the display in a position that is relative to the Overlay Frame. The Overlay will not be dismissed and flow will not continue until the user clicks the Continue button.
  - **Click Anywhere or Any Key** – The Overlay will not be dismissed and flow will not continue until the user either clicks somewhere on the screen or presses a key on the keyboard.
  - **Wait for Sound** – The Overlay will not be dismissed and flow will not continue until any sounds currently playing have stopped.
  - **Wait for Movie** – The Overlay will not be dismissed and flow will not continue until any movies currently playing have stopped.

Figure 10.10: Designating a synchronization time: 1) the Stopwatch field before editing. 2) Click in the Stopwatch field to open an edit window. 3) Enter a time in seconds. 4) Press return, or click anywhere outside the edit field, to set the time.

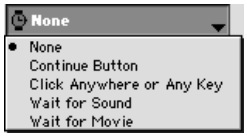


Figure 10.11: The Synchronization dropdown menu.

- **Print Window** –This option causes the contents of the Overlay window to be printed. You can use a Print Options Gadget to suppress the standard page setup and print option dialogs. See Chapter 11, Gadget States, for more information about using the Gadget Print options.
- **Hide Menu Bar** – Hides the menu bar during runtime, while displaying the Overlay Image. Even though the menu bar is hidden, command keys are still functional.

  Note that if the **Hide Menu Bar** option is selected, any custom menus you've created for your application will also be hidden while the Overlay Image is displayed.
- **Autocenter Overlay Image** – Automatically positions the Overlay Window in the center of the target screen.
- **Custom Position** – Opens a positioning window that lets you set up the Overlay Window's position and size. The first time the **Position** button is used for an Overlay, the Background Output dialog is displayed, prompting you to choose an Output State that will serve as the background for positioning the Overlay Window. After choosing the background and clicking **OK**, you can relocate and resize the positioning window as desired. Use the locator handle in the upper left-hand corner of the positioning window to move it around. Use the size handle in the lower right-hand corner to change the positioning window's size. After moving and resizing the positioning window, select **Close Window** (⌘ W) from the **File** menu or press the Enter or Return keys to close the window with the new position saved and return to the InfoCenter. Press Command period or escape to close the positioning window without saving changes.

  While in the position mode, you can change the location of the overlay graphic image inside its positioning window by clicking on the image and moving it (Figure 10.12). An information bar is displayed at the bottom of the positioning window showing the distance in pixels that the image has been moved.
- **On Output** – Opens the Background Output dialog, prompting you to choose an Output State that will serve as the positioning background for this Overlay image. Functions similarly to the Position button, except that the Background Output dialog is opened every time, rather than just the first time. After choosing the background and clicking **OK**, you can relocate and resize the positioning window as desired.

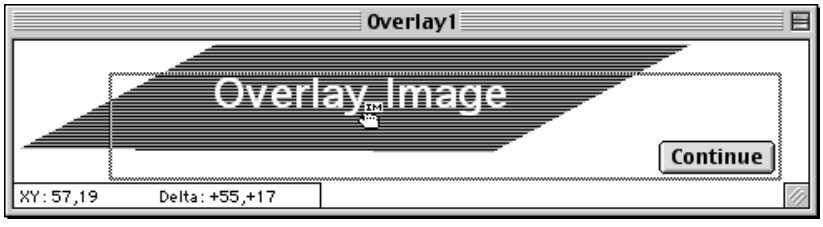


Figure : Changing the location of an overlay image inside its positioning window. The dotted line shows an outline of the image as it is being dragged, while the information bar at the bottom of the window displays the image's X and Y coordinates, as well as how far is has been moved.

Note that an Overlay's window's size and/or position is not based on the Output chosen for positioning. If you want an Overlay window with a different size and position, create a new Multimedia State with the image selected.

No Output allows you to position the Overlay window with no Output background.

Chapter 11

Gadget States

Covered in this Chapter:

- Using Gadget States
- Bookmark Gadget
- Cursor Display Gadget
- File Maintenance Gadget
- Print Options Gadget
- Report Options Gadget
- Restart Application Gadget
- Send Email Gadget
- Show Web Page Gadget
- Monitor and Sound Gadget
- Timer Gadget

Using Gadget States

Gadget States are special function States. Route your application through them to achieve a number of different effects, such as changing the appearance of the cursor, opening a Web site, setting a time limit for a State, or changing the way report data is saved.

There are ten different functions that a Gadget State can perform in your application:

- Create a Bookmark File
- Change the Cursor Display
- Download Files from a server location
- Preset Print Options
- Report Saving Options
- Restart Applications
- Send Email
- Show a Web Page
- Adjust Monitor and Sound Volume
- Add a Timer to your application.

You might incorporate the same Gadget function in several locations in your application - each set with different options. As your application flows though the Gadget, the behavior for the function is changed according to the set options. Remember that Gadget functions inherit the behavior set in the most recent Gadget the application flows through.

For example, you may want to change the information onViz saves in the report file based where your user is in the application. To prevent reports from being saved if the user quits in the middle of the application, start your application with a Report Gadget set with the **Do Not Save Report** option. Add another Report Gadget set with the **Save Report on Exit** option before the final exit point. A complete report will be saved only if the user completes the application.

The Gadget State has an InfoCenter containing the settings that will determine which function is used during runtime. The InfoCenter contains a **State Name** field for giving the State a unique, meaningful name. While a State's name can be changed from within its InfoCenter window, it can also be changed without leaving the Application Map. To change a State's name from the Application Map window, click once on the State's name, wait for the Name field to become a black rectangle with a red outline, then type the new name. Pressing Enter or Return, or clicking anywhere outside the Name field, makes the new name take effect.

The InfoCenter also contains an **Automatic Return** option, which functions the same as having the Gadget State followed by a Return State. If the Gadget State is used as a Bridge's Target State, checking this option will automatically return flow back to the Bridge's Exit Point, without needing to add a Return State to your Application Map. See chapter 5, Flow States, to learn how to use Return States in your application.

Tip

The Gadget State InfoCenter can also be accessed by selecting a Gadget State and then choosing Gadget InfoCenter (⌘ I) from the **Map** menu.

Notice that by checking this box the Gadget State icon no longer displays an Exit Point; this is because the **Automatic Return** option always routes flow back to the Bridge State that called it, eliminating the need for a path from the Exit.

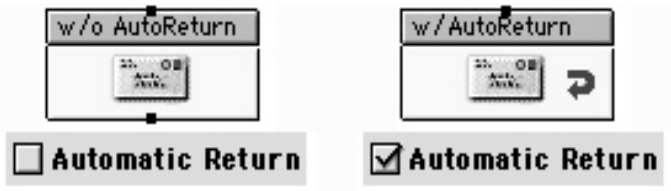


Figure 11.1: A Gadget State without its Automatic Return option selected (left), and with it selected (right). Note that selecting the Automatic Return option causes the State icon to lose its Exit Point, while labeling the icon with an arrow.

Gadget Error Paths

Most Gadgets include an option to designate a Bridge target for your application in case the gadget function fails.

It's good practice to designate a Bridge that can be used in case of error. One option is to display an error message alerting your audience of the error, and giving the option to either skip to the next section or quit. You could also give your audience the option to send you an email notifying you of the problem.

Bookmark Gadget

A Bookmark is a file that can be saved to the user's disk when the application exits before it is complete. Typically, this file is saved in a location designated by the application author. Double-clicking a bookmark file restarts the application at the point from which the bookmark was selected. All report information, statistics and variable values are preserved in the bookmark file.

During runtime of your application, choosing Quit from the File menu presents users with the option to 'bookmark' their place in the application to return at a later time. If, however, the menu bar is hidden or the File menu is disabled, your users cannot select the 'Quit' menu item. The Bookmark Gadget gives you the freedom to add an exit with option to save a bookmark file at any point in your application.

For example, you may want a user to complete a specific section before exiting the application. You can disable the File menu and at the end of the section of interest, either use an Input State to give users the option to bookmark their place or continue, or simply route the application through a Bookmark Gadget to automatically exit the application and save a bookmark.

Or, you might create a perpetual on-screen button labeled 'Bookmark' and set to Bridge to a Bookmark Gadget when clicked. When users select this option, the application bridges to the Bookmark Gadget and gives the option to save their place.

Note: The 'default' file storage for Bookmark files is designated in the 'Build Target' dialog (or can be end-user configurable). Options in the Bookmark Gadget let users save their bookmark files in a location different than set as the default. See Chapter 2 for more information about saving your project source for delivery.

A Bookmark Gadget can also be used at the beginning of an application to give users an opportunity to open previously saved bookmarks. In this instance, you might use an Input State to ask users if they wish to open their bookmark file. A Bookmark Gadget with the **Give Option to Open Saved Bookmark** option selected will display a 'Get File' dialog listing the contents of the 'default file storage.' Users can select their saved bookmark from this location, or navigate to a different location. Note that if you are using passwords to restrict access to the application, the bookmark file will be restricted with the same user password.

The application exits when a bookmark is saved. If users want to continue where they left off, they will either need to double-click the bookmark file to launch it, or if you have used a Bookmark Gadget to allow opening of a previously saved bookmark, launch the top level application again.

Note that when your user selects Quit (⌘ Q) from the file menu, the Bookmark action follows the settings of the last Bookmark gadget the application encountered. To change options during runtime, simply route the application through another Bookmark Gadget with the desired options selected.

Best Practice

If you're using bookmarks in your application, a Bookmark Gadget with the **Give Option to Open Saved Bookmark** setting selected should be included at the beginning of your application. By doing so, you give your users the chance to pick up where they left off without having to search for and launch the bookmark file.

Bookmark InfoCenter

The Bookmark InfoCenter contains a **Give Option to Open Saved Bookmark** setting and a **Save Bookmark** section. When your application flows through a Bookmark gadget with the **Give Option to Open Saved Bookmark** selected, a dialog is presented for your user to open a previously saved bookmark. This dialog will present a list of the files saved in the default bookmark location. However, the user can navigate to any other location to find and open a bookmark.

If **Save Bookmark** is selected, additional settings become available that help you manage your bookmarks:

- **Allow User to Cancel** – This adds a 'Cancel' button to the Save dialog. When users click **Cancel**, they are returned to the application at the exit point of the Bookmark Gadget.

  If this option is not selected, the user will not have the option to cancel the bookmark process and the application will exit once the bookmark file has been saved.

- **Skip Save Dialog if File Already Exists** – If the application was begun from a bookmark file, or the user opened a bookmark file from an 'Open Saved Bookmark' Gadget, onViz will not present the save dialog when the user exits the application through another Bookmark Gadget. The new bookmark will automatically overwrite the user's previous bookmark file.

  Selecting this option ensures that each user creates only one bookmark file and it contains the most current status of progression in the application.

- **Use Default Storage Location** – onViz will place the user's bookmark file in the location specified in the Build Target dialog. The user will not have the option to navigate to another location for saving the bookmark file.

- **Delete Bookmark File if Application is Complete** – When your application goes through a Bookmark Gadget with this option selected, onViz will automatically delete the user's bookmark file when the application is complete.

Note: if the user has created several bookmark files, only the bookmark that was used to re-enter the application will be deleted when the application completes.

You limit each user to one bookmark by selecting the **Skip Save Dialog if File Already Exists;** with this option selected, onViz will overwrite the older bookmark file with the newer one every time it is saved.

- **Default with User Name** – When a bookmark file is created, the Save As dialog will, by default, be filled in with the 'User Name' General Variable. It the **Allow User to Edit Name** option is selected, the user may choose to change the name of the file.

- **Default with Specified File Name** – Selecting this option presents an Edit field in which you can type a Default File Name for the bookmark file. When a bookmark file is created, the Save As dialog will, by default, be filled in with the name you typed in the edit field. It the **Allow User to Edit Name** option is selected, the user may choose to change the name of the file.

Tip

The User Name variable is set by either selecting the 'Ask for User Name' option in the Project Attributes Report tab, or you can use an input state to request the user enter unique identification and use a Text Calculator to assign this entry to the User Name Variable. Now, your bookmark file names will include the user's name or ID.

- **Allow User to Edit Name** – When the bookmark Save dialog is presented, lets the user change the name for the bookmark file. If this option is not selected, the bookmark file will be saved with either the User Name or the name specified.

On Error, Follow: – This allows you to select a bridge to follow if an error occurs and onViz was not able to write a bookmark file.

Cursor Display- Gadget

You can change your application's cursor appearance or show and hide the cursor by routing the application through a Cursor Gadget. This Gadget type accesses the Cursor Library, a collection of default and custom cursors.

Cursor Display InfoCenter

The Cursor Display InfoCenter contains a selection of default and custom cursors that you can use in your application. The top section of **Platform Default Cursors** lets you choose between the cursors that are standard in the Macintosh and Windows operating systems. These cursors cannot be edited. The Custom Cursors section lets you choose between cursors supplied in onViz or one that you have designed yourself. You can modify one of onViz' cursors create your own by selecting a cursor and clicking the Edit Cursor button, which opens the Cursor Library. For more information on using the Cursor Library to create and edit custom cursors, see chapter 4, Libraries.

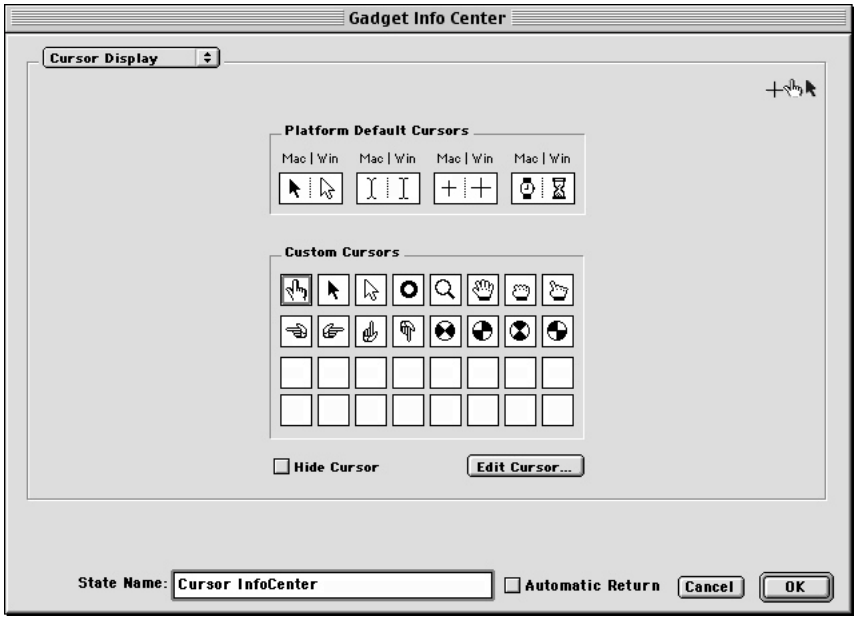


Figure 11.4: The Cursor Display InfoCenter. Clicking the Edit Cursor button opens the Cursor Library.

By selecting the **Hide Cursor** option, you can remove the cursor from the display. This option works well for kiosk applications, where you **may only want the cursor visible to make menu choices, and then hidden the rest of the time. Likewise, you might use this option during animations, if there are no items on the screen for your users to click.**

File Maintenance

File Maintenance is a new onViz feature and is not yet complete when running on Windows. This section of the documentation will be updated when this feature is complete.

Print Options Gadget

By routing your application through a Print Options Gadget, you can preset an Output State's page setup and printing options. Use this setting in conjunction with the Output State's Print Window option to print an Output's contents automatically and seamlessly, without requiring input from the user. You can use this Gadget at the very beginning of your application to preset the printing options for all of your Outputs, or place it just before individual Outputs to preset their print options separately. You will often want to print Design Windows with landscape option and Text Windows with portrait option. The Print Options Gadget allows you to change print orientation at any point in your application.

Default action for Outputs set to **Print** the Design and/or Text Window is to present the user with a standard print dialog for each window to be printed. You can suppress this dialog by running the application through a Print Options Gadget with **Show Print Dialog** deselected.

Print Options InfoCenter

The Print Options InfoCenter lets you set the printout's paper orientation, scaling, and number of copies. Click the left-hand button to set your printout's **Orientation** to portrait, or click the right-hand button to set it to landscape. Enter a percent value in the **Scale** field to increase or decrease the size of the content being printed. Entering a value less than 100 reduces the size of the content being printed, helpful if you want to ensure that it all prints on one page. A value greater than 100 increases the size of the content being printed.

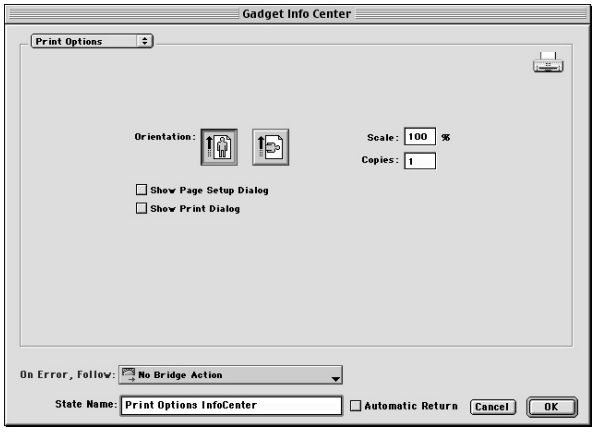


Figure 11.5: The Print Options InfoCenter.

Select the **Show Page Setup Dialog** and/or the **Show Print Dialog** if you want your users to be able to adjust these settings before printing. If these options are not selected, onViz will print using the settings selected in the Print Options InfoCenter.

On Error, Follow: – This allows you to select a bridge to follow if an error occurs and onViz was not able to write print.

Report Options Gadget

Typically, report saving options are set up in the Project Attributes Report tab. However, you can change these options throughout the application with the Report Options Gadget. You can use its settings to change and/override the content that goes into the application's report, as well as saving incremental (milestone) reports. For more information on report options, see chapter 2, Customizing the Development Environment.

Report Options InfoCenter

Save Report on Exit – onViz' default for saving reports; the report is created upon exiting the application. Routing the application through a Gadget with this option turned off, overrides the options set in the Project Attributes.

Figure 11.6: The Report Options InfoCenter.

- **Standard Application Data** – onViz will generate an automatic report based on options selected:
  - **Application Summary** – Statistics on the overall application including name of the application, time spent in the application and scoring statistics.
  - **Groups** – Statistics on all groups executed including time spent in the group, and scoring statistics.
  - **Inputs** – Statistics on all inputs executed including time spent in the input, number of tries, and scoring statistics.
- **Sort by** – Report content can either be sorted **Sequentially** (the order in which the states were executed, or **Alphabetically** (sorted by group name and inputs in the group).
- **Collect Text Output Data Only** – Used in conjunction with the Text Output's **Save Text in Report** option (found in the Output InfoCenter). onViz will only gather for its

report the contents of the application's Text windows. Use this option to create custom reports for your application.

- **File Name Format** – Use this dropdown menu to select a naming convention for your report files. Options include:
  - **User Name** – You may use this option if you have selected the **Ask for user name** option in the Project Attributes Report tab.
  - **Sequential** – Report file names will be sequentially numbered (i.e., Report #1, Report #2, etc.)
  - **Date and Time** – Report file name will contain the date and time that the report was saved.

Save Milestone Report Now – With this option selected, a Milestone report is saved in the report storage location as set in the Build Target dialog. A Milestone report is an incremental report containing statistics for the application up to the point that the Gadget was encountered. Each Milestone Report is saved in a separate file. If this option is selected, you also have the option to create a Report Heading. This heading field supports variable substitution, so it can be used to capture information such as time, date, user name, etc.

Clear Report Contents – with this option selected, any report data gathered before this Gadget State is run will be deleted when the application flows through this gadget. Exercise caution when using this option, as the report data cannot be recovered after it is deleted.

On Error, Follow: – In case of instances where an error occurs in saving the application report, this option allows you to create a Bridge for your user to follow.

Tip

The User Name variable is set by either selecting the 'Ask for User Name' option in the Project Attributes Report tab, or you can use an input state to request the user enter unique identification and use a Text Calculator to assign this entry to the User Name Variable. Now, your report file names will include the user's name or ID.

Restart Application Gadget

Routing application flow through a Restart Application Gadget causes the application to restart either immediately or if idle for a specified time. This Gadget State works well for kiosk applications, where users are likely to view a segment of the application, potentially leaving it at a point where the next user would be lost. By restarting the application, you ensure that each user accesses the presentation at its starting point.

Restart Applications InfoCenter

The Restart Applications InfoCenter lets you determine if and when your application will restart itself. Note a report is generated each time the application restarts.

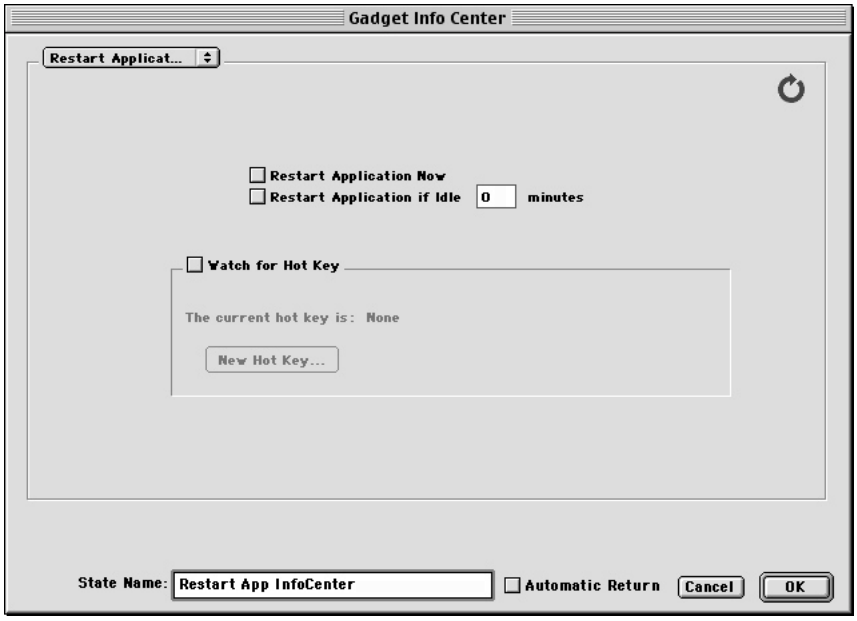


Figure 11.7: The Restart Application InfoCenter.

Restart Application Now – When the application flows through a Gadget State with this option selected, it will immediately restart.

Restart Application if Idle – When the application flows through a Gadget State with this option selected, it sets an 'idle' flag, If the application is idle for the amount of time specified in this edit field, it will automatically restart. Note that any mouse or keyboard activity resets the idle flag.

Watch for Hot Key – Lets you designate a keyboard shortcut that will restart the application whenever it is used. Use the **New Hot Key** button to choose a keyboard shortcut. During runtime of the application, anytime the keyboard shortcut is executed, the application will restart.

Send Email Gadget

When application flow is routed through an Email Gadget, onViz sends an email in the background to a designated recipient. By bridging to the Email Gadget from an **On Error, Follow:** dropdown, it can be used to notify an administrator of problems with the application. Also, by placing Email Gadgets at key points in an application, they can be used to keep an administrator apprised of a user's progress.

Send Email InfoCenter

The Email InfoCenter is used to create the email message that will be sent.

Figure 11.8: The Send Email InfoCenter.

Select the **Append Application Report** check box if you want the email to include a report of statistics collected to the point that the gadget was encountered. Use the **Message To** section to specify the **Name** of the email recipient, the recipient's **Email Address**, the email **Subject**. Text in the **Message** field will precede any appended application report.

A special feature of the Message field is that you can use the Output Variable 'Text Window Content' to automatically include any text from the currently displayed Output in the email message. You *must* use a smart sprite with a 'Bridge and Return' to the Send Email Gadget in the output – once the output has been dismissed, the text window is reset and the information is no longer available unless it has been set to be saved in the report file. This feature enables you to set up options for the user to send you a message.

Send Via SMTP Server – if you have this information for the user's machine you can enter this here. Otherwise, onViz will attempt to use the default email configurations on the machine.

Tip

All fields in the Send Email InfoCenter support Variable Substitution. This allows the setup and message to be dynamic based on the user's input and other application statistics.

Tip

Create a Custom Variable for the SMTP Server field. Preset this variable with the SMTP address if known or leave it blank. In case of error sending the email, use an Input to prompt the user for a correct server address and use a Text Calculator to set the Custom Variable to the user's input. The gadget can then be tried again with this new user entered address.

Optional From Name – the default for this field is the User Name variable. If this variable is not filled in, this field will be blank. You can use variable substitution to fill this field with anything of your choosing.

Optional From Email Address – You can either complete this field or, use an Input State to ask the user this information and use Variable Substitution to fill in the field.

On Error, Follow: – In case of instances where an error occurs and the email is not sent, this option allows you to create a Bridge for your user to follow. One option is to ask for the user's SMTP server information if the email fails to be sent.

Show Web Page Gadget

When application flow is routed through a Show Web Page Gadget, the user's computer launches its default Web Browser and opens the specified Web site.

Show Web Page InfoCenter

The Show Web Page InfoCenter is used to specify the address for the Web page you want to open, and if desired, automatically return to your application. Enter the web page address into the **URL:** (Uniform Resource Locator) field.

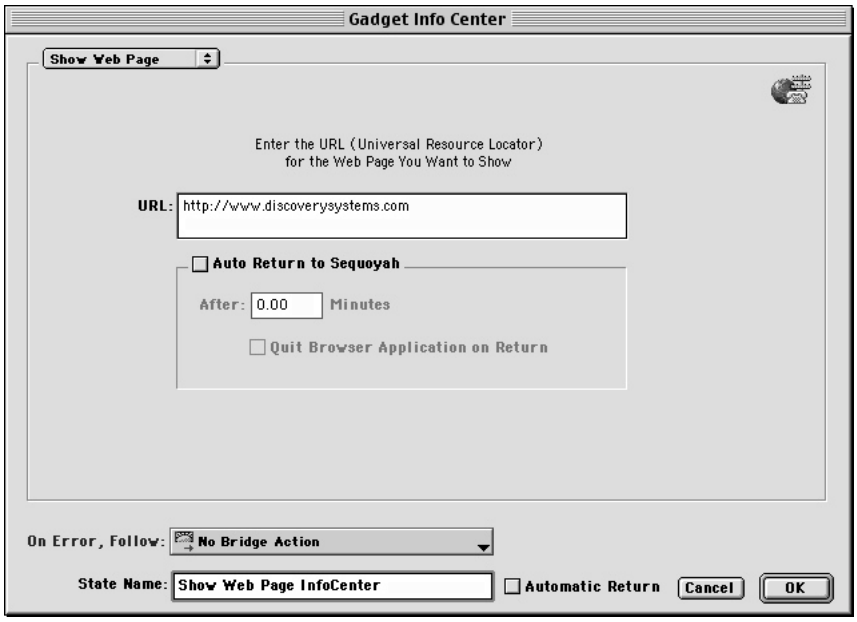


Figure 11.8: The Show Web Page InfoCenter.

If the **Auto Return to onViz** check box is selected, you can designate the number of minutes your users will have to view the Web page you've specified before they are automatically returned to your application. This option also allows onViz to automatically quit the Web Browser on return.

On Error, Follow: – In case of instances where browser fails to open or connection to the Internet fails, this option allows you to create a Bridge for your user to follow.

Sound and Display Gadget

The Sound and Display Gadget lets you automatically change the color depth and volume level on your user's computer. When used with a Conditional Path at the beginning of an application, you can check to see if the user's Monitor and Sound levels are at the appropriate settings, and route flow through this Gadget if they are not. The Sound and Display Gadget can also be used to change the application's mat color - the solid background around your application display on displays that are set to a higher resolution than your target screen. Mat color for an application is set in the Project Attributes Runtime tab. Once changed with a Sound and Display Gadget, the mat color remains until the application flows through another Sound and Display gadget with a new mat color.

Sound and Display InfoCenter

The Sound and Display InfoCenter is used to specify which settings on the user's computer will be changed. You can choose to change the computer's color depth, and/or sound volume and the Mat color for your application.

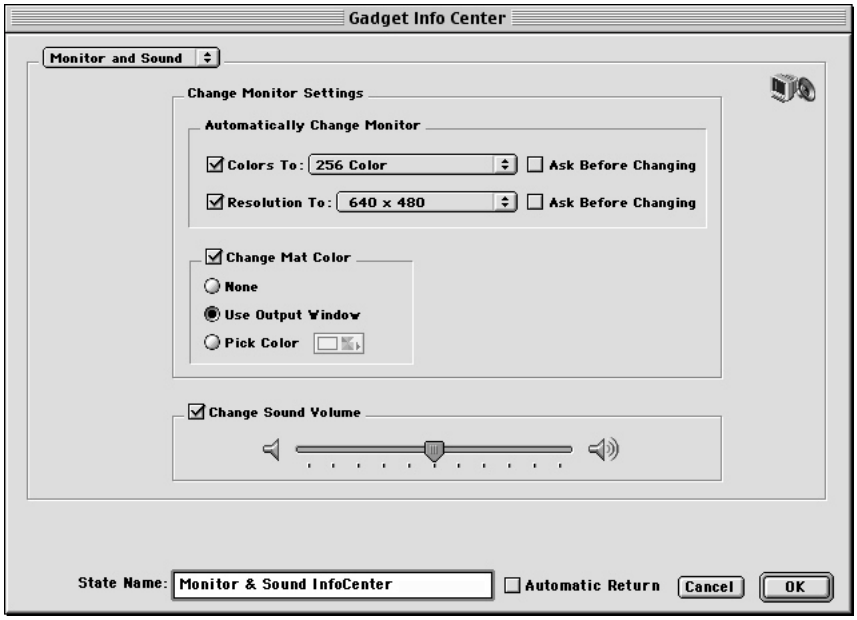


Figure 11.10: The Monitor and Sound InfoCenter.

Change Monitor Settings

- **Automatically Change Monitor** – Use the dropdown menu to choose the color depth to which you want your user's monitor changed. onViz will revert to the original monitor settings when your application exits.
- **Ask Before Changing** – This option, lets your users either approve or reject the changing of their monitor settings.
- **Change Mat Color** –By selecting this option, you change the mat color (any area visible around the application's 'target' screen) for your application. This could be a handy way to designate different sections or chapters of an application.

Best Practice

Before changing settings on your users' computers first inform them, and give them the option to cancel.

Change Sound Volume – When application flows through a gadget with this option set, the computer's speaker volume will be adjusted. Use the volume slider to set the desired volume. onViz will revert to the original sound volume when your application exits.

On Error, Follow: – In case of instances where an error occurs such as if the user 'cancels' change in color depth, this option allows you to create a Bridge for your user to follow.

Timer Gadget

Run your application through a Timer Gadget in to display a count down or count up digital timer. The Timer Gadget can be used to start, stop, resume, or pause a timer's count. A Bridge action can be added to the timer to signal expiration or a warning.

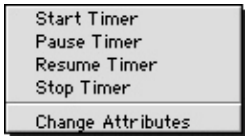


Figure 11.13: The Timer Function dropdown menu.

Timer InfoCenter

Timer Function Dropdown – Determines if the Timer Gadget will serve to **Start**, **Pause**, **Resume**, or **Stop** the Timer. If **Change Attributes** is selected from the dropdown menu, the InfoCenter is used to change the timer's appearance only – you cannot change the timer time. If **Pause**, **Resume**, or **Stop** timer are selected, the **Don't Reset** check boxes become available.

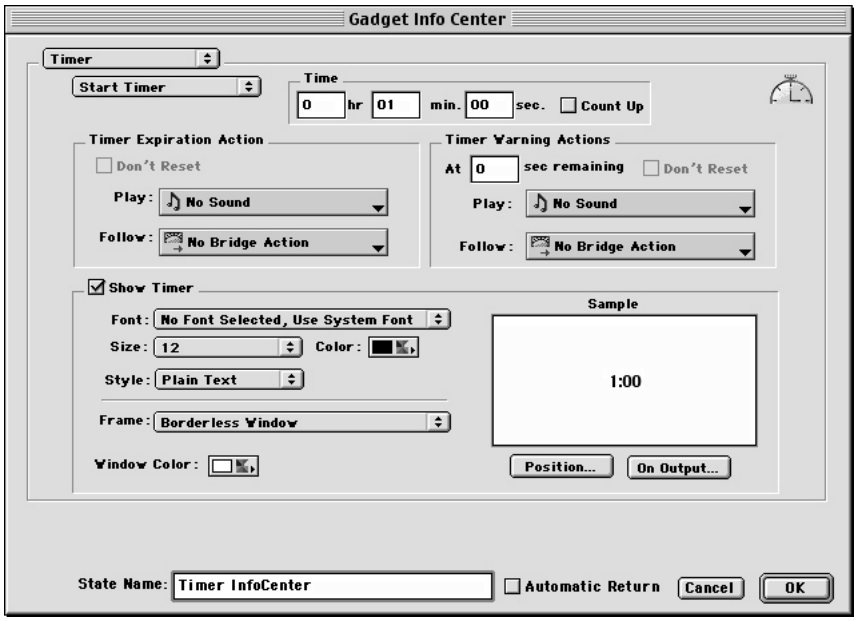


Figure 11.12: The Timer InfoCenter.

Time – Enter a time in hours, minutes, and seconds.

Count Up – The timer will count up to the specified time from zero.

Timer Expiration Action – Controls onViz' actions when the timer runs out:

- **Play** – Designates a sound that will be played when the timer expires. From this menu, you can select a sound that already exists in the Sound Library, go to the Sound Library to edit an existing sound's attributes, or create a new sound reference. The sounds listed at the bottom of this dropdown menu include two default sounds, System Alert and System Simple Beep, plus any sounds that have been designated to be shown in the Sound dropdown menu. If you have created a sound reference but not included it in the dropdown menu, you can select Other to display a list of all sounds in the library.
- **Follow** – Similar to a Bridge Target dropdown menu, this setting allows you to designate where the application goes when the timer expires.

Timer Warning Action – Enter a value in the Time Remaining field, and onViz will perform the designated actions when the specified time is reached:

- **Play** – Same as Expiration Actions above.
- **Follow** – Same as Expiration Actions above.

Show Timer – If this option is not selected, the timer will not be visible to the user, but all other actions are still active. With this option selected, the timer will be displayed so that the user can track the amount of time remaining. You can also choose the timer's **Font**, **Size**, **Color**, **Frame**, **Window Color**, **Position**, and **On Output...** position. Pressing **Position…** displays a list of named Output States in the application. Select one of these states to position the timer where you want it displayed. Once an output has been selected, it remains the default for positioning the timer. To change outputs, press **On Output…** to again display a list of named Outputs.

Chapter 12

Group States

Covered in this Chapter:

- Using Group States
- Types of Groups
- The Group InfoCenter
- The Group Presentation Window
- Group Variables

Using Group States

Group States are subsections of the larger Application Map. Viewing a Group on the Application Map, it appears as any other State with an entrance and up to 10 exits. When you open the presentation, however, you see that it's very much like the Application Map: it has a work area, shares the Map Tools palette, and has the same menu bar. Because it is so similar to the Application Map, the Group State Presentation is referred to as the Group Map.

Imagine that you're working on a very large project that takes up a great deal of the Application Map work area (Figure 12.1). Chances are that some elements of your project are pretty straightforward, with perhaps several sections of States connected in a linear fashion, while others may demonstrate freeform routing from one side of the work area to the other. In situations like this, it is easy to lose sight of your States, as well as being difficult to trace your project's flow of activity. Groups provide a means to control this type of sprawl.

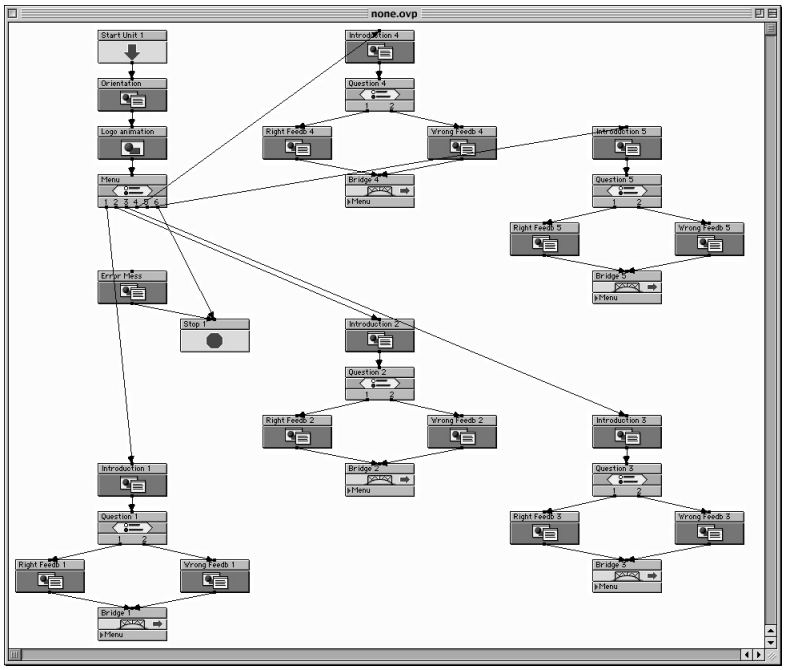


Figure 12.1: A large Application Map, and an ideal candidate for the use of Groups.

There are several reasons you might want to use a Group. You may want to organize the States in your project into functional sections, such as topics or chapters (Figure 12.2). You can also use them to group branching strategies. For instance, your top-level Application Map may use a freeform branching strategy, but have long sections of States connected linearly. By using Groups, you can organize your long sections of linear States into a single Group that has been designated to be linear. Not only does this type of organization eliminate the need to use Paths within the Group, it also allows you to organize your top-level Application Map.

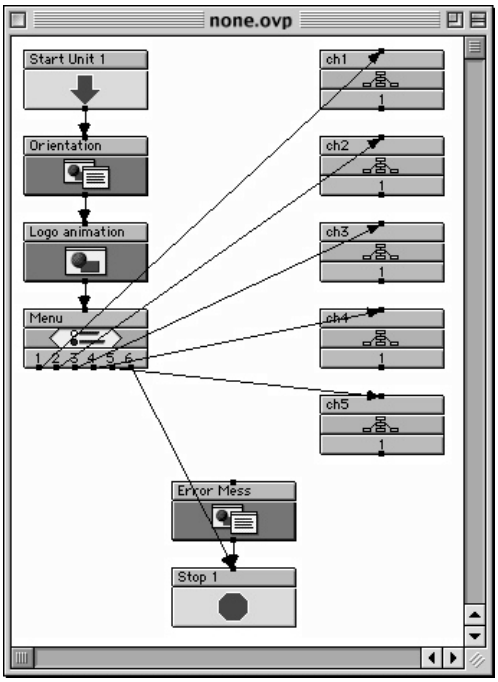


Figure 12.2: The Application Map from Fig. 12.1, this time with its chapters organized using Groups.

Groups can also be used to automate certain types of flow that would be difficult to set up using only the top-level Application Map. For example, setting up States to randomly draw from a set of questions would be a challenge to lay out. However, using a Group that has been designated to choose at random from the States inside it is as simple as moving the States into the Group, setting its branching strategy to be Random Pool, and designating the number of States to play. If necessary, Groups may even be nested inside other Groups, increasing your options for varied branching strategies.

Typically, a Group has only one Exit Point. However, the Group InfoCenter gives you the option for designating up to ten Exit Points. If a Group has more than one Exit Point, it should also contain a corresponding number of Stop States. The Stops display a number designating the order in which they were created, and the Exit Point to which they are linked. If a Stop is grayed out, or dim, it means there are more Stops than there are Exit Points (Figure 12.3).

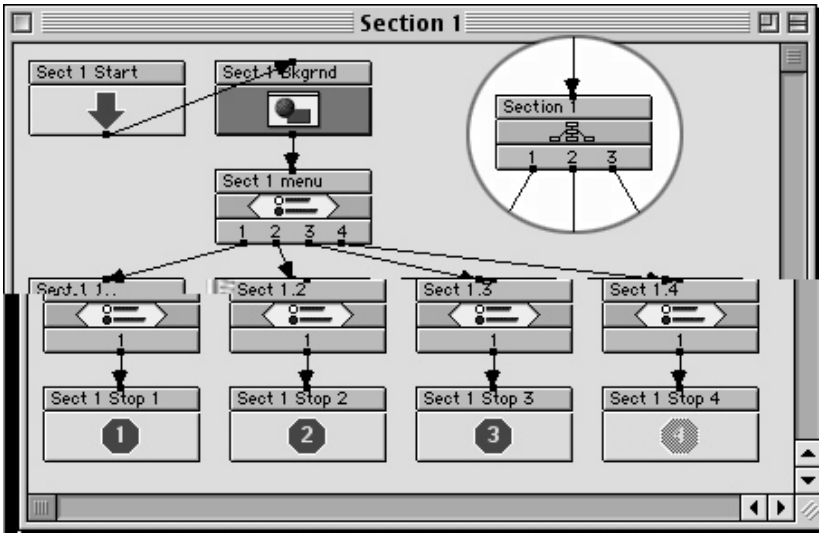


Figure 12.3: A Group with multiple Exit Points must contain a corresponding number of Stops. The inset shows a Group with three Exit Points. However, this Group contains four Stops. The fourth Stop is dimmed out because it exceeds the number of Exit Points.

Types of Groups

There are five types of Groups, differentiated by the way they branch, or route your application's flow of activity:

- Freeform
- Linear
- One per Row
- Random Pool
- List Access

When using the Map Tools palette to add States to your Application Map, you can access a contextual menu listing the different types of Groups by holding down the Control key while clicking on the Group icon. Choose the type of Group from the menu; your mouse pointer will turn into a star. Then click in the Application Map work area where you want the State to be positioned.

Freeform Groups

In the Freeform Group, routing of your project's flow of activity is determined by the way in which its States are connected by Paths, which can branch in any way desired. Freeform Group Maps *must* begin with a Start, have all of their Paths properly connected, and end with a Stop (Figure 12.4). Flow begins with the Start, and continues until it reaches a Stop, at which point the flow exits the Group and moves to the next State in the Application Map. A "Dead End Route Reached" error message is displayed if an Exit is encountered that does not have any properly connected paths, or if an Exit is encountered whose only connections are unsatisfied Conditional Paths.

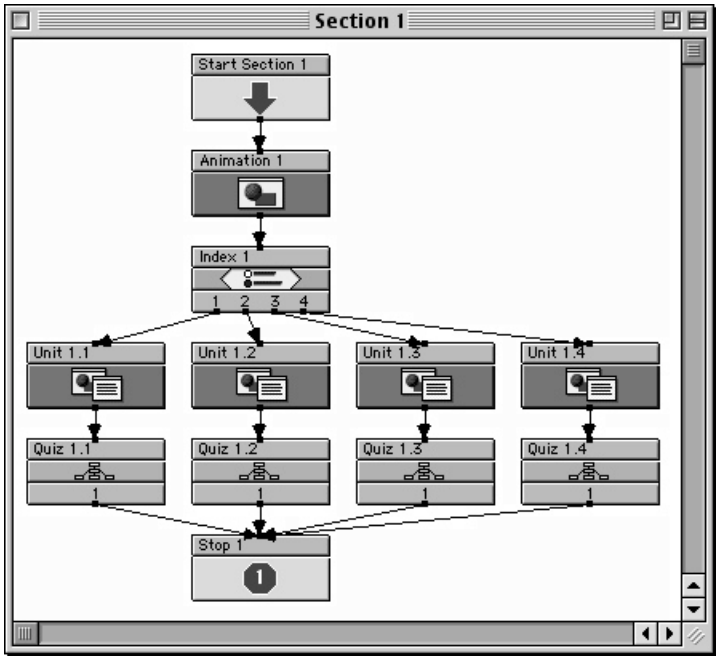
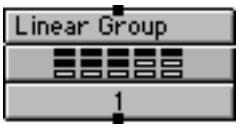


Figure 12.4: A Freeform Group map. Notice that this Group begins with a Start, ends with a Stop, and in between has several diverging branches in which the application flow is routed by Paths.

Freeform Groups are most commonly used for interactive projects using various forms of feedback, as the Application Map is an excellent visual method for viewing this type of flow of activity. If you are testing your project and encounter a "Dead End Route Reached" error message, you can click the **Show Me** button to see at what point your project's flow of activity stopped, making it easier to connect the dead end.

Linear Groups



The Linear Group routes your project's flow of activity through the Group Map sequentially from left to right and top to bottom. When determining flow sequence, the Linear Group starts with the upper-most left-hand State, then flows to the right (Figure 12.5). If no States are encountered to the right of the first one, it moves down to the far left of the next row and scans right until it finds another State. The Linear Group keeps scanning from left to right, top to bottom, until it has either scanned all of the Group's rows or encounters a Stop, at which point the flow exits the Group and moves to the next State in the Application Map.

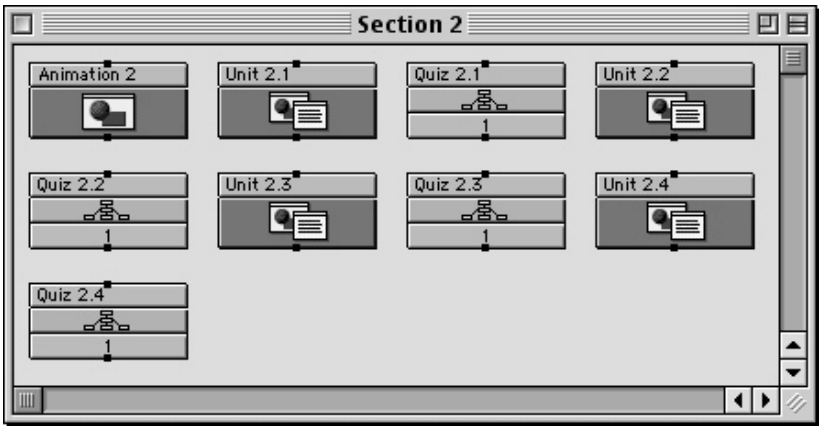
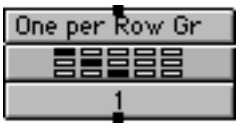


Figure 12.5: A Linear Group map. Notice that there is no Start or Stop. Instead, application flow begins with the top-left State, moves right to the end of the row, then drops down to the far-left State in the next row. This flow continues until it reaches the far-right State in the bottom row.

Although the Linear Group requires no Stops, or Paths, they may still be used. Paths can be used at any point in the flow of activity to override the linear flow. For instance, if a State inside the Group has a Path attached to its Exit Point, the flow of activity will follow that Path rather than flowing to the right. In this way, a State may use a Path to branch off to give some feedback before continuing its flow, or the Path may branch off to a Stop that marks the end of the Group.

One per Row Groups



The One per Row Group routes your project's flow of activity by randomly picking one State from each of the Group's rows. This flow continues until the last row is reached, at which point the flow exits the Group and moves to the next State in the Application Map.

Figure 12.6 shows a One per Row Group Map that contains five rows, each with a different types of joke. onViz will choose one joke from each row until, and will exit the Group after a joke is chosen from the last row.

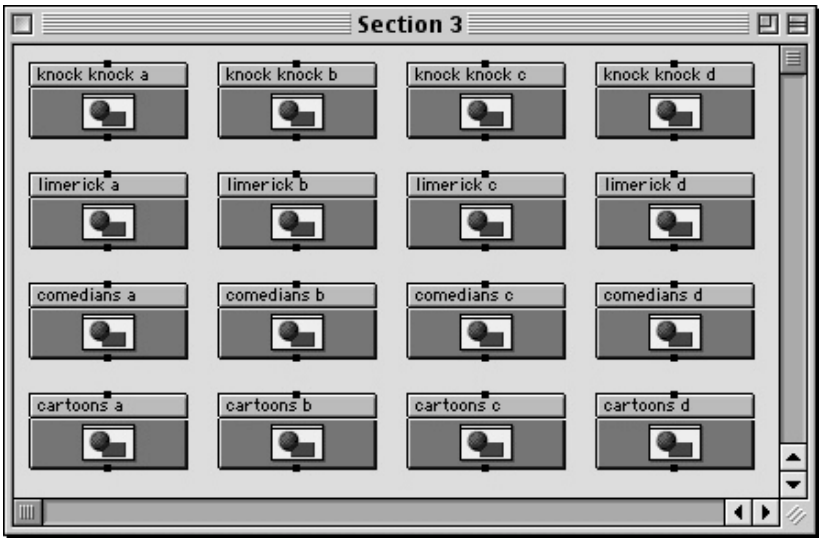
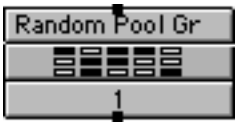


Figure 12.6: A One per Row Group map. Notice that there are no Starts or Stops. Instead, application flow is routed by choosing one State from each row, from top to bottom. In this example, onViz will choose one joke from each row, and will exit the Group after a joke is chosen from the last row.

The One per Row Group randomly chooses one State from each row without regard to its function. As a result, it is advisable to use States that can stand alone in their content. For instance, if you are using a series of Inputs and Outputs for questions and feedback, it is possible that one of the feedback States could be chosen, rather than one of the question States, resulting in the display of feedback to a question that has never been asked. To avoid having the wrong State selected in this way, incorporate any associated States their own Group that can then be chosen from the row safely. Also, this Group does not require Starts, Stops, or Paths, although Paths and Stops can be incorporated to override the One per Row flow.

The One per Row Group works well for tests, as it can vary the questions enough so that no one gets the same test. It is also good for practice drills, as the same questions are not continually asked.

Random Pool Groups



As its name suggests, the Random Pool Group determines your project's flow of activity by choosing States from the Group at random. The number of States to be chosen is designated in the Group InfoCenter, and random execution continues until that number is reached, at which point application flow exits the Group and moves to the next State in the Application Map. If the number of States designated in the InfoCenter exceeds the number of States in the Group, flow will exit the Group after all of its States have been chosen once.

Figure 12.7 shows a Random Pool Group Map that contains twenty old sayings. onViz will choose from these old sayings at random until the specified number is reached, or until all of the sayings have been chosen.

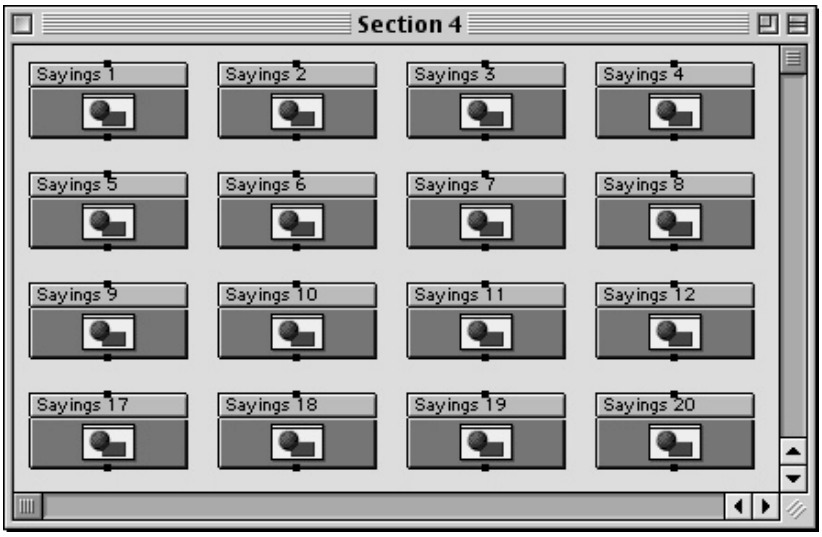


Figure 12.7: A Random Pool Group map. Notice that there are no Starts or Stops. Instead, application flow is routed by choosing States from the Group at random, until a specified number (set in the Group InfoCenter) is reached. In this example, onViz will choose from these old sayings at random until the specified number is reached, or until all of the sayings have been chosen.

As in the One per Row Group, the Random Pool Group randomly chooses States without regard to their function. As such, include to use States that can stand alone in their content. Also, this Group does not require Starts, Stops, or Paths, although Paths and Stops can be incorporated to interrupt the random flow. Note that if you use a Stop, this State may be randomly selected and will exit the Random Pool.

The Random Pool Group works well for varying the questions in tests and practice drills.

List Access Groups

The List Access Group presents the user with a list of names of all the States contained within it (Figure 12.8). The user can browse this list and from it select a choice of States to run. When application flow reaches the Exit Point of the selected State, the list is once again presented. This flow continues indefinitely until the user selects **Exit**, at which point the flow exits the Group Map and moves to the next State in the Application Map. The list dialog window can have a title, which is typed into the **List Access Title** field in the Group InfoCenter.

Tip

You can assign your own unique names to the 'Exit' and 'OK' buttons from within the List Access Group InfoCenter.

Since the List Access Group lists all of the named States within the Group, regardless of function, it is advisable to use States that can stand alone in their content.

Since the user can opt to cancel and leave the Group at any time, no Starts, Stops, or Paths are necessary. However, if a Stop is included, its name appears on the list and can be selected by the user to exit the Group.

List Access Groups are an excellent means of conveniently accessing a large database of information. By organizing your data into a hierarchy of List Access Groups, each level is displayed as a list, and can be accessed either by selecting a State name and then selecting **OK** to run it, or by double-clicking the State name.

The Group InfoCenter

Tip

The Group InfoCenter can also be accessed by selecting a Group and then choosing **Group InfoCenter** (⌘ I) from the **Map** menu.

The Group InfoCenter is used to select a Group's branching strategy and access runtime features, and varies slightly depending on the type of Group selected. The InfoCenter can be accessed by double-clicking the rectangular area in the upper 1/3 of the State's icon.

While a State's name can be changed from within its InfoCenter window, it can also be changed without leaving the Application Map. To change a State's name from the Application Map window, click once on the its name, wait for the Name field to become a black rectangle with a red outline, then type the new name. Pressing Enter or Return, or clicking anywhere outside the Name field, makes the new name take effect.

Other features within the Group InfoCenter are as follows:

Select Branching Strategy

This dropdown list determines the type of branching followed by your project's flow of activity (Figure 12.10). A Group State's branching strategy can be designated when it is chosen from the Map Tools palette and it can be changed from this list at any time during development.

*Figure 12.10: The **Select Branching Strategy** dropdown menu.*

Figure 12.9: The Group InfoCenter.

Number of Exits

This dropdown list determines the number of Exit Points your Group has. If a Group has more than one Exit Point, it should also contain corresponding Stops. The Stops displays a number designating the Exit Point to which they are linked.

Save Group Statistics in Report

If checked, this option automatically saves the Group's statistics in the application report. These statistics include Time spent answering questions within the group, and scoring information for the Inputs in the Group.

On Every Entry, Reinitialize Scoring for Inputs in this Group

If checked, this option causes the Group's statistics, variables, and input statistics to be reset each time the Group is entered.

Make Default

After setting all of your Group's attributes, press this button to cause all subsequent Group placements to have the same attributes. This option can save a lot of time for repetitive State placements.

States in Pool

Used for Random Pool Groups only. The number typed here determines how many States are chosen at random from your Random Pool Group.

List Access Title

Used for List Access Groups only. Typing a title here causes it to be displayed as the list title when the List Access Group is run.

List Access Button Names

Provides an option for renaming the default 'Exit' and 'OK' buttons that are presented when the List Access dialog is presented.

The Group Presentation Window

A Group's Presentation window is used to design and organize the flow of activity for the States within the Group, and is accessed by double-clicking in the rectangular area in the lower 2/3 of the Group State's icon.

Since the purpose of the Group is to organize the Application Map by grouping other States inside it, the Group Presentation window looks and functions exactly like the Application Map work area. Click the desired State on the Map Tools palette, and drag and drop it onto the Group Map. Like the Application Map work area, the Group Map can hold 64 States horizontally and 64 States vertically. To close the Group Map and return to the Application Map, click the Close Box in the upper-left corner of its window.

Tip

The Group Presentation window can also be accessed by selecting a Group and then choosing **Group Presentation** (⌘ E) from the **Map** menu.

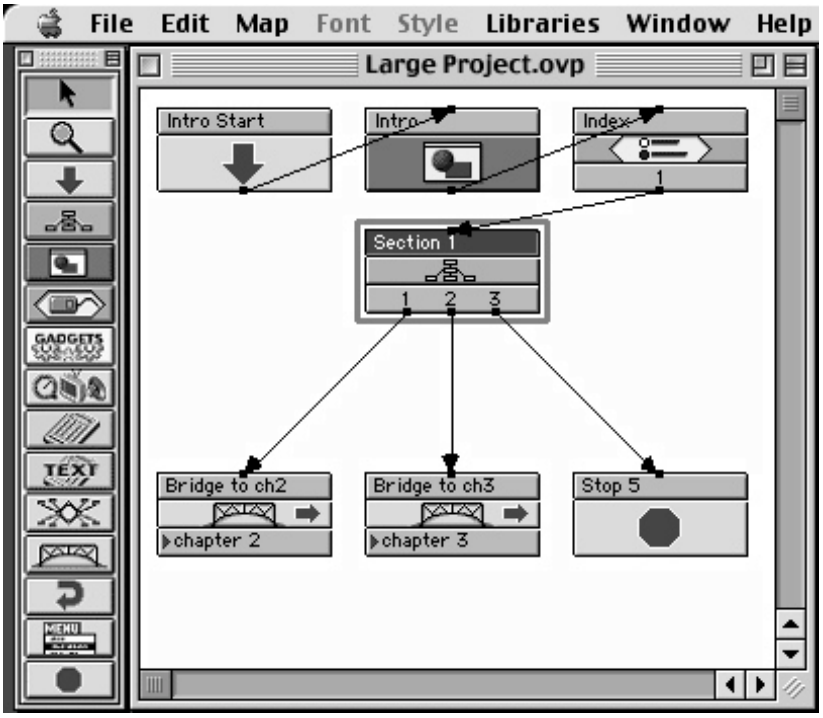


Figure 12.11: An onViz Application Map. Note the application's file name (Large Project.ovt) in the Title Bar. Also notice the Group, which is currently selected, in the middle of the map. To compare this Group's Map with this Application Map, see Figure 12.12.

Exercise caution when closing the Group Map, as it looks very similar to the Application Map work area. Closing the Group Map returns you to the Application Map or, if enclosed in another Group, the enclosing Group's Group Map. Closing the Application Map work area closes your project, make sure you select **Save** from the dialog presented to save your work. You can differentiate between the two by comparing their window Title Bars: the Group Map displays your Group's title, while the Application Map work area displays your interactive project's title.

Tip

The Group State Map can also be closed by selecting **Close Window** (⌘ W) from the **File** menu.

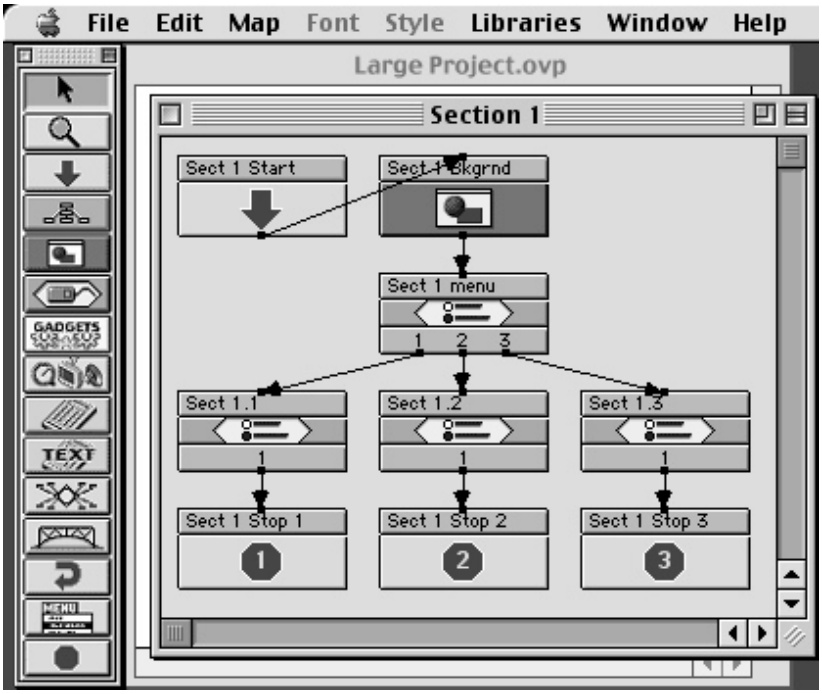


Figure 12.12: A Group Map in the foreground, with the top-level Application Map in the background. Though the Group Map functions similarly to the Application Map, the two can be distinguished by their Title Bars: the Application Map's Title Bar contains the application's file name, while the Group Map's Title Bar contains the Group's State name.

Group Variables

Most Group Variables track Input statistics as a collection within the designated Group. These variables are cumulative, and update as each Input exits.

Read Only

- Percent Score (GroupPercentScore{State}) - This numeric variable represents the calculation of a user's actual score, divided by the total possible score for all visited Inputs in the designated Group, and is updated each time the Group is exited. The Inputs must each have a designated **Correct Answer** and a value in the **Points if Correct** field in order for this statistic to be tracked.
- Percent Correct (GroupPercentCorrect{State}) - This numeric variable represents a calculation of a user's number of correct responses, divided by the number of visited Inputs, in the designated Group, and is updated each time the Group is exited. The Inputs must each have a designated **Correct Answer** and a value in the **Points if Correct** field in order for this statistic to be tracked.

Write Only

- Reset Group (GroupReset{State}) - Flowing the application through a Calculator that contains an equation setting the value of this variable equal to 1 resets all variables associated with the designated Group. This includes resetting any Input or Output States within the Group.

Read/Write

- Time Spent in Inputs (GroupTimeInInputs{State}) - This numeric variable represents the time a user has spent in the designated Group's Input States. You can write to this variable and modify its value using a Calculator.
- Total Score (GroupTotalScore{State}) - This numeric variable represents a user’s score for all Inputs within the designated Group. You can write to this variable and modify its value using a Calculator.
- Total Correct (GroupTotalCorrect{State}) - This numeric variable represents a user's number of correct responses for all Inputs within the designated Group. You can write to this variable to clear or modify its value using a Calculator.
- Exit Path (GroupExit{State}) - This numeric variable represents the Exit path taken from the Group. You can write to this variable to clear or modify its value using a Calculator.

Chapter 13

Variables

Covered in this Chapter:

- Using Variables
- Using Variable Substitution
- Creating Custom Variables
- Input Variables
- Output Variables
- Group Variables
- General Variables
- Creating Custom Reports

Using Variables

One of onViz' unique and valuable feature is its ability to capture a great deal of information about the application that is running and the user who is running it, and to manipulate this information in a variety of ways using variables. Variables can be thought of as containers that hold information.

For instance, the Document's Total Score variable is meant to hold only one type of information: the user's total score. The variable can display this information, or use it to intelligently route the application's flow (Figure 13.1). Note that information such as a user's total score does not always remain the same; as a user progresses through an application, the score will undoubtedly change. As the score changes, the information held by the variable changes also.

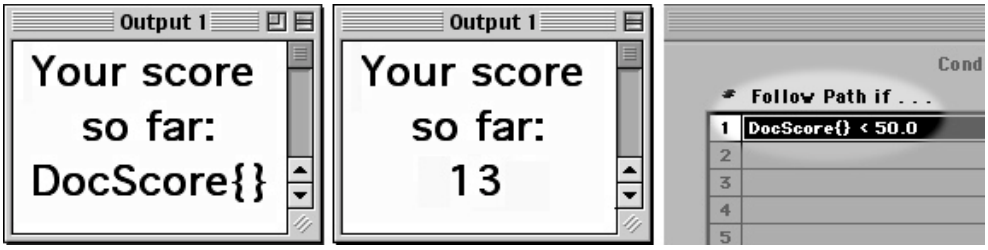


*Figure 13.1: The **Document's Total Score** variable **(DocScore{})**, viewed in a Text Window during authoring (left), and during runtime (center). Note that during runtime the variable is replaced by the user's actual score. This variable can also be used in a Conditional Path to route application flow (right).*

The information that fills these variables may come from actions taking place within the application, as in the total score example above, and/or the information may come from you, the author. You might use a Sprite's X position and Y position variables to place a sprite at a specific X and Y coordinate, in which case you, rather than the application, are providing the variable with the information that it will hold.

Variables can be ***read/write***, ***write only***, or ***read only***. ***Read/write*** variables can either be ***read***, as in the total score example above, or ***written to***, as in the X/Y sprite position example. Using a variable to ***read*** information is like using a newspaper reporter to cover a story. For instance, you can use the Answer Entered variable to report back on the way a user responded to a question in an Input State. When the application is run, the variable will capture the information and display it for you to read (Figure 13.2).

*Figure 13.2: Variable substitution with the **Answer Entered** variable. The variable viewed in an Input window during authoring (left).*

onViz can also ***read*** a variable and then use a Conditional Path to make a branching decision based on the value of the variable. For instance, if you decide that users with a score of 80 or above can skip the rest of your quizzes, you can use the Total Score variable with a Conditional Path. The Path will be followed if this condition is met: "the total score greater than or equal to 80." If the total score is less than 80, the Conditional Path will not be followed. For more information on Conditional Paths, see chapter 15.

When you ***write*** information to a variable, the variable ceases to be just a passive reporter and takes on an active role in the application. If you select the **Text Window Visible** variable, and set it to hide a Design and Text Output's Text Window, the variable is not merely reporting that the window is hidden, it is actively causing it to be hidden. However, the **Text Window Visible** variable can work both ways: it can be used to report back on the window status and it can cause the Text Window to be hidden, making it a ***read/write*** variable.

You use a Calculator State to ***write*** to a variable. If you decide to hide a Design and Text Output's Text Window, you send the application flow through a Calculator. In the Calculator, you choose the variable **Text Window Visible** on the left side of the calculation field, and set that variable equal to zero (zero equals invisible, one equals visible) (Figure 13.3). Then, when your application flows through the Calculator State and enters the designated Output State, the Text Window will be hidden. To make it visible again route your flow through another Calculator, one that has the same variable set equal to one (visible).

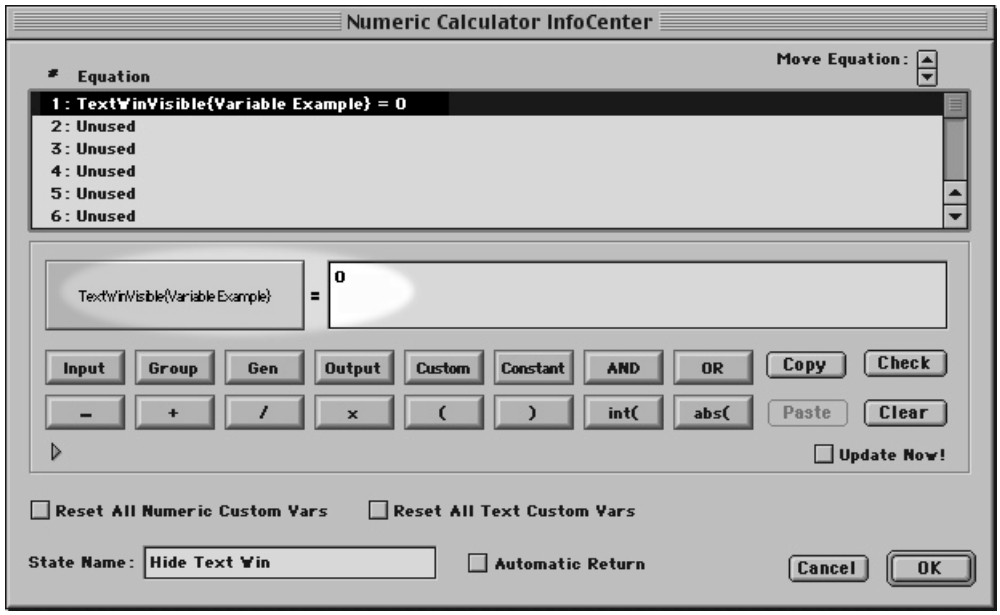


Figure 13.3: Using a Calculator to hide a Text Window. The top portion of this Calculator InfoCenter contains a list of equations, and the lower portion is used to create the equations.

You can use a Calculator State to update the value of any ***read*** or ***read/write*** Variables. For more information on Calculator States, see chapter 14.

Read only variables cannot be written to using a Calculator. An example of a read only variable is the **Total Frames variable**, which "reads" the number of animation frames in a particular Output (Figure 13.4). An Output's number of frames is data that can only be reported; no variable can change it.

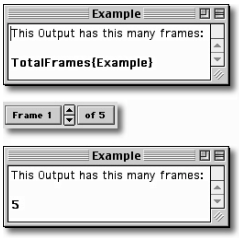


Figure 13.4: Using the Total Frames variable to read the number of frames in an animation. The top window shows the variable during development, and the controls in the middle, taken from the Frame Control palette, show an animations number of frames. During runtime (bottom), the variable is replaced with the number of frames in the animation.

Returning to our 'reporter' scenario, say for example, you send your reporter down to the movie theater to count the number of frames in a movie reel, the reporter can't change the number of frames, only count them and report the findings. Of course, it is possible to change an output's number of animation frames by going into the Output State Presentation and using the Frame Control palette to add or delete them.

The act of ***reading*** a variable and ***displaying*** it is referred to as ***variable substitution***. You might use an Input State in your application to ask your user's name, and use an Output

State to display that name to personalize your application. Since, at the beginning of your application, you don't yet know what your user's name will be, you insert the **Input Answer Entered** variable in the Output State text where you want the user's name to appear. When the application is run, your variable acts like a reporter again; this time the breaking story is "How did the user answer the question, 'What is your name?'" When the Output State with the variable is run, the user's name appears in place of the variable name. You can use variable substitution to display a variety of information, from the current date and time to your user's total score.

onViz supports a large number of variables, which are grouped into five categories:

- **Input** – Input variables are associated with Input States, and deal primarily with question, answer, and scoring issues.
- **Output** – Output variables are associated with Output States, and revolve around animation frames and sprites.
- **Group** – Group variables deal with application statistics for a specific group, such as time and scoring.
- **General** – General variables are not associated with any specific state type, but instead focus on issues that involve the entire application, such as total document time and score, date and time, and attributes of the end-user's computer.
- **Custom** – Custom variables are created by you, the developer, and allow you to maintain your own values, which can be passed between the top-level application and any supporting documents. For more information on custom variables and the Custom Variables Library, see chapter 4, Libraries.

onViz' variables are accessed through the Variable Selection dialog, which contains a tabbed page for each category of variable. Each tabbed page lists the available variables in that category (Figure 13.5). When browsing the variable lists that are too long to fit in the window, you can speed selection by clicking in the window once to activate it, then typing a letter or number to jump automatically to that part of the list.

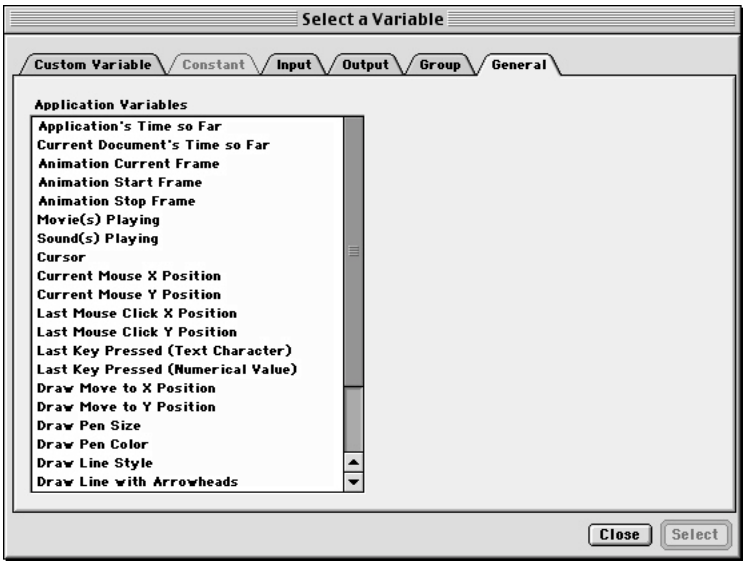


Figure 13.5: The Variable Selection dialog. Note that the General tab is in the foreground, listing all the General variables.

The Variable Selection dialog can be accessed in the following ways:

- By holding down the Option key and clicking in Input InfoCenter question and answer fields, in the "edit fields" in the Report Options and Email Gadgets, in Custom Menu States, or in object text in the Image Editor.
- From a Calculator State.
- From the Conditional Path InfoCenter.

Using Variable Substitution

Variable substitution adds a personal and dynamic quality to your application. You can add variables to your application that will, during runtime, display information personalized to each user, such as name and application statistics, or information pertaining to the date and time the user was in the application.

When using variable substitution, be careful to leave the variable exactly as it is when inserted. onViz recognizes that the text contains a variable by the exact formatting of the text string. Altering the text string by adding or removing words, spaces, etc., will cause onViz to not recognize the variable.

Displaying a Variable in a Text Window

For this example, we'll be inserting the **Current Time (Text)** variable into the Text Window of a Design and Text Output. Any variable, however, can be inserted using the same procedure. Because we'll only be creating and viewing a one-frame animation, all you'll need on your Application Map for this example is one Design and Text Output State.

1. Create a Design and Text Output State, and open its Presentation window.

   The Output State Presentation window can be opened by double-clicking the bottom two-thirds of the Output State icon, or by selecting the Output State and choosing **Output Presentation** (⌘ E) from the **Map** menu.

2. In the Text Window, type "The current time is:" and then press the Return key.

3. On the next line, hold down the Option key while clicking.

   The Variable Selection dialog is opened, which has a tabbed page for every category of variable.

4. Select the General tab to display the list of General variables. Scroll down the list until you see the **Current Time (Text)** variable. Click on the variable name, then click the **Select** button. Or, you can select the variable by double-clicking the variable name in the list.

   You could make a selection from any of the variable categories, but for this example we'll be using a General variable, **Current Time (Text)**. Clicking **Select** closes the Variable Selection dialog and returns you to the Output Presentation window, where the variable appears as **Time{}** (Figure 13.6). This variable holds a place in the window's text, where the current time will be displayed when the animation is run.

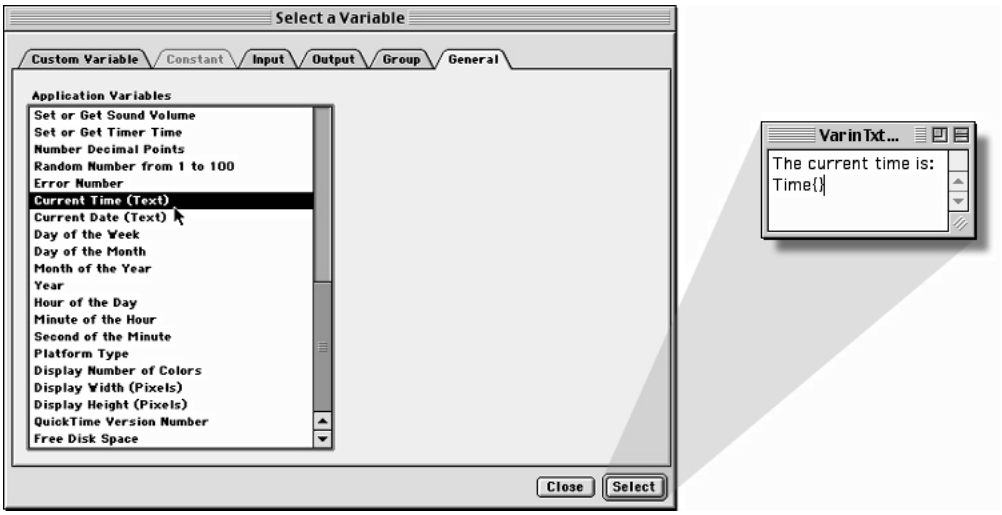


*Figure 13.6: Choosing the General variable **Current Time (Text)**. Clicking **Select** closes the Variable Selection dialog and returns you to the Output Presentation window, where the variable appears in the sprite as **Time{}***

5. Set the Frame Control palette's **Synchronization** tool to **Click Anywhere or Any Key (Figure 13.7)**.

   In this example, the **Click Anywhere or Any Key** option keeps your text output visible so that you can verify that the current time is being displayed. During actual development, you may decide to use any Synchronization option, or none at all. For more information on using the Frame Control palette, see chapter 6, Output States.

*Figure 13.7: Choosing **Click Anywhere or Any Key** from the Frame Control palette's **Synchronization** dropdown menu.*

6. Choose **Run Animation** (⌘ R) from the **Animation** menu.

   When the animation runs, the Text Window displays the current time (Figure 13.8). Click anywhere to end the animation and return to the Presentation window.

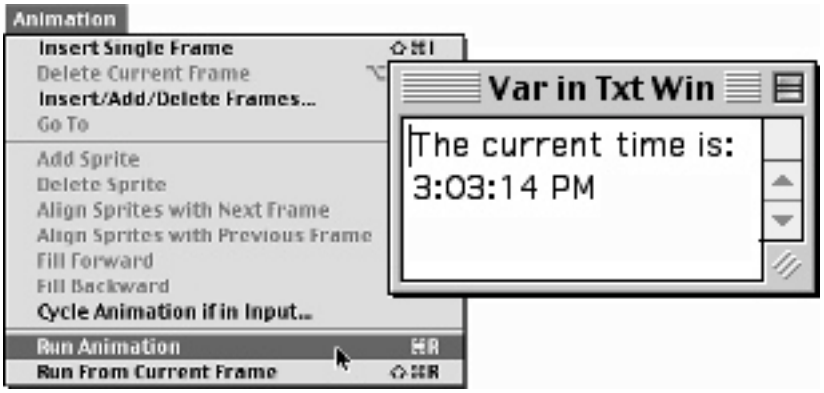


*Figure 13.8: After choosing **Run Animation** from the **Animation** menu, the current time appears in the sprite in place of the variable.*

Displaying a Variable in a Design Output

For this example, we'll use an Input State to ask our user's name, and then display that name in a Design Output. To perform this trick, you'll need to create an Application Map with four States: a Start State, an Enter Text Input, a Design Window Visible Output, and a Stop State. The Answer Entered (Text) variable will be inserted into a sprite in the Design Output.

1. Create a new Application Map with 1) a Start State, 2) an Enter Text Input, 3) a Design Window Visible Output, and 4) a Stop State, and use Paths to connect them in the

order listed. Be sure to give the States meaningful names. For instance, name the Input State "name," as it will ask your users their name (Figure 13.9).

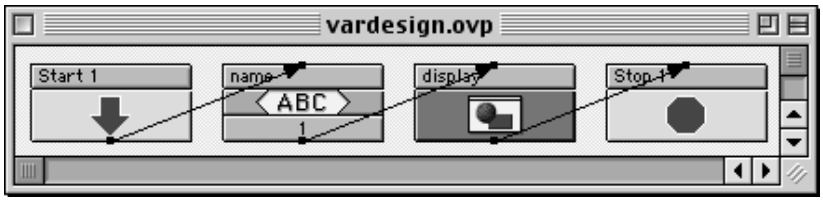


Figure 13.9: The Aplication Map from step 1, complete with named States and Paths.

2. Open the Input State InfoCenter. Under the Attributes tab, choose 1 from the Number of Answers: dropdown menu. Under the Answers 1-5 tab, type "What is your name?" into the **Enter Question:** field, then click **OK** (Figure 13.10).

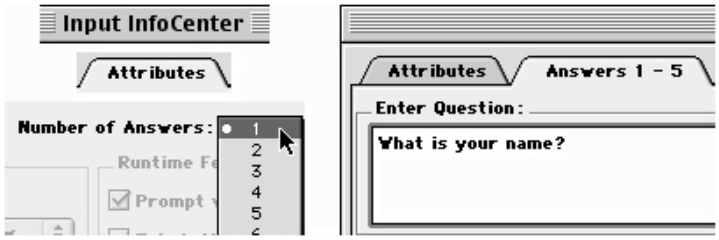


*Figure 13.10: While in the Input InfoCenter, set the **Number of Answers:** dropdown menu to **1** (left), and type "What is your name?" into the **Enter Question:** field (right).*

The Input State InfoCenter can be opened by double-clicking the upper one-third of the Input State icon, or by selecting the Input State and choosing **Input InfoCenter** (⌘ I) from the **Map** menu. The Attributes tab and the Answers 1–5 tab are visible at the top of the InfoCenter dialog. Clicking OK closes the Input State InfoCenter and returns you to the Application Map.

3. Open the Output State Presentation window.

   The Output State Presentation window can be opened by double-clicking the lower two-thirds of the Output State icon, or by selecting the Output State and choosing **Output Presentation** (⌘ E) from the **Map** menu. When the Presentation window opens, you may need to drag the Sprite Attributes palette off to the side in order to access the Design Window.

4. Hold down the Control key and click in the Design Window, then choose QuickSprite from the contextual menu (Figure 13.11).

   Choosing QuickSprite opens the Image Editor, which is where you'll create a sprite that contains the **Answer Entered (Text)** variable.

Figure 13.11: Control-clicking in the Design Window opens a contextual menu from which you can create a QuickSprite.

5. From the Object Tools palette, choose the Text tool, and click and drag in the work area to create a new text box. In the text box, type "Your name is:" and press the Return key.

   You must use the Text tool from the Object tools palette, as it creates text that remains editable and thus supports variable substitution. The Text tool from the Pixel tools palette is not editable once it has been created, and does not support variable substitution.

6. On the next line, hold down the Option key while clicking (Figure 13.12).

   Holding down the Option key while clicking opens the Variable Selection dialog, which has a tabbed page for every category of variable.

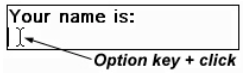


Figure 13.12: After typing "Your name is:" press the Return key to move down to the next line, then hold down the Option key while clicking to open the Variable Selection dialog.

7. Select the Input tab, to display the list of Input variables. Choose your Input State's name from the list of Input States on the left, choose **Answer Entered (Text)** from the list of Input variables on the right, and then click the **Select** button.

   You could make a selection from any of the variable categories, but for this example we're using an Input variable. Notice that the **Select** button is grayed out, or unavailable, until you select both an Input State and an Input variable. After selecting an Input State and an Input variable, the **Select** button becomes active. Clicking **Select** closes the Variable Selection dialog and returns you to your QuickSprite in the Image Editor, where the variable appears as **InputAnswerText{Name} (Figure 13.13)**.

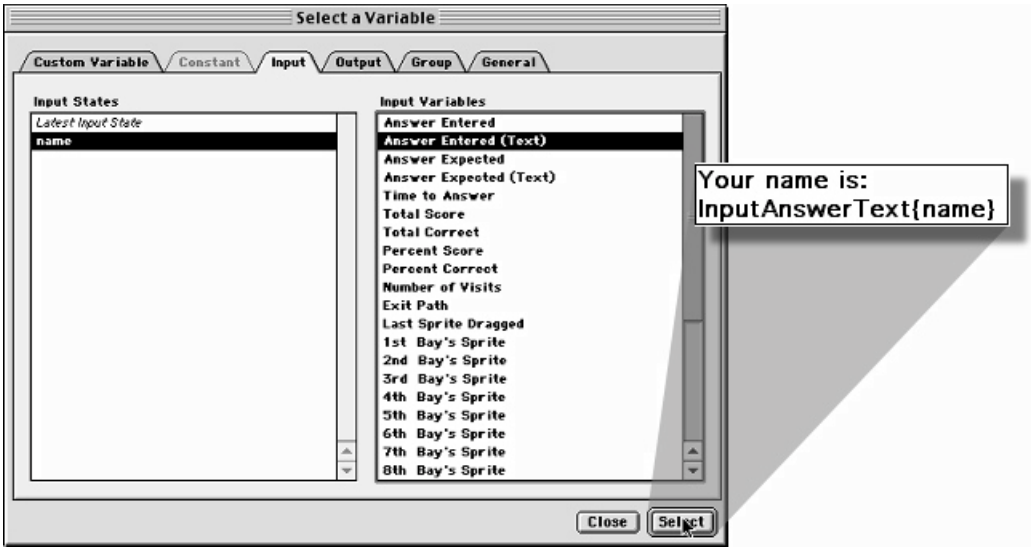


Figure 13.13: The Variable Selection dialog, with the list of Input variables in the foreground.

8. Format the text as desired, making sure to resize the text box until it is wide enough to display your text correctly, then close the Image Editor. For more information on using the Image Editor, see chapter 7.

   The Image Editor can be closed by clicking the Close Box to, or by choosing **Close Window** (⌘ W) from the **File** menu. Closing the Image Editor returns you to the Presentation window's design area, where your new sprite is now displayed and the QuickSprite automatically added to the Sprite Palette. For more information on using the Image Editor, see chapter 7.

9. Set the Frame Control palette's **Synchronization** tool to **Click Anywhere or Any Key**.

   In this example, the **Click Anywhere or Any Key** option will keep your design output visible so that you can verify that the user's name is being displayed. During actual development, you may decide to use any Synchronization option, or none at all. For more information on using the Frame Control palette, see chapter 6, Output States.

10. Close the Output Presentation window to return to the Application Map by choosing **Close Window** (⌘ W) from the **File** menu.

11. Back in the Application Map, choose **Run** (⌘ R) from the **Map** menu.

    When your application begins running, enter your name when the input is displayed, then click **OK**. You will then see your name displayed in the Design Output (Figure 13.14). Click anywhere to stop the test run and return to the Application Map.

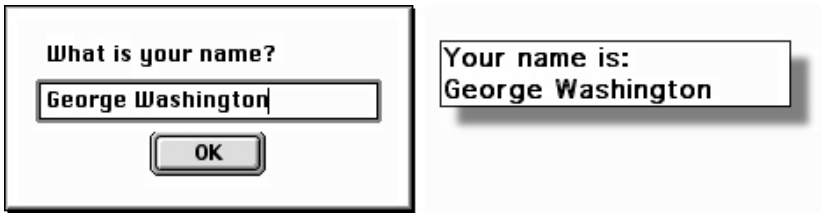


Figure 13.14: During runtime, enter your name into the Input window, and it is displayed inside the Design Output.

Displaying a Variable in an Input State

Variable substitution can also be performed in Input States. In this example, we'll be using the Number of Visits Input variable to track the number of times the user goes through your Input State.

1. Create a new Application Map with 1) a Start State, 2) a Radio Button Input State, and 3) a Stop State, and then connect the Start State to the Input State. The rest of the Paths will be added later. Name the Input State "times through."

2. Open the Input State InfoCenter. Under the Attributes tab, choose 2 from the Number of Answers: dropdown menu. Under the Answers 1-5 tab, type "You have been here times. Do you want to continue?" into the Enter Question: field. Be sure to leave an extra space between the words "here" and "times," which is where the variable will be placed.

3. Under **Enter Answers:**, type "Yes" into the first answer field, and "No" into the second answer field (Figure 13.15).

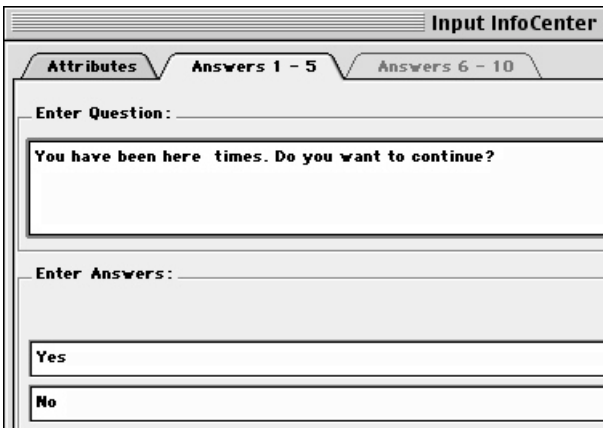


*Figure 13.15: The Input InfoCenter. Notice the extra space between the words "here" and "times," and the **Yes** and **No** answer fields.*

Because you chose 2 in the **Number of Answers:** field, your Input State has two Exit Points.

4. Position the cursor between the two spaces (between the words “here” and “times”), and click while holding down the Option key (Figure 13.16).

   Holding down the Option key while clicking opens the Variable Selection dialog, which has a tabbed page for every category of variable.

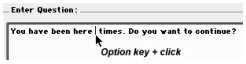


Figure 13.16: To open the Variable Selection dialog, position the cursor between the words "here" and "times," then click while pressing the Option key.

5. Select the Input tab to display the list of Input variables. Choose your Input State’s name ("times through" in this example) from the list of Inputs on the left, choose **Number of Visits** from the list of Input variables on the right, then click Select (Figure 13.17).

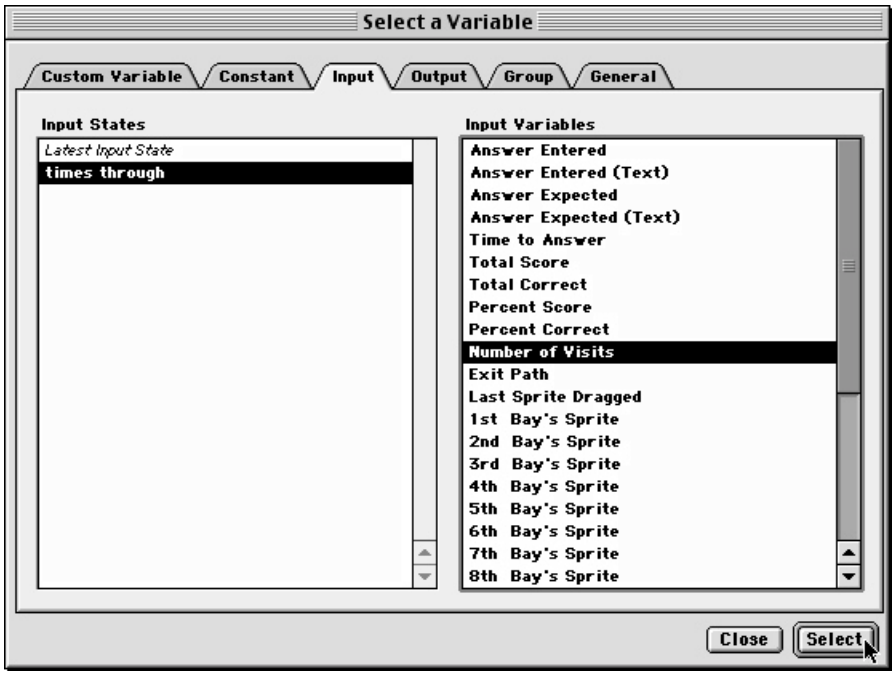


*Figure 13.17: The Variable Selection dialog, with the list of Input variables in the foreground. Choose "times through" from the list of States on the left, and then choose **Number of Visits** from the list of variables on the right.*

Clicking Select closes the Variable Selection dialog and returns you to the Input State InfoCenter. The text in the **Enter Question:** field, including the variable, will read “You have been here InputVisits{times through} times. Do you want to continue?" (Figure 13.18).

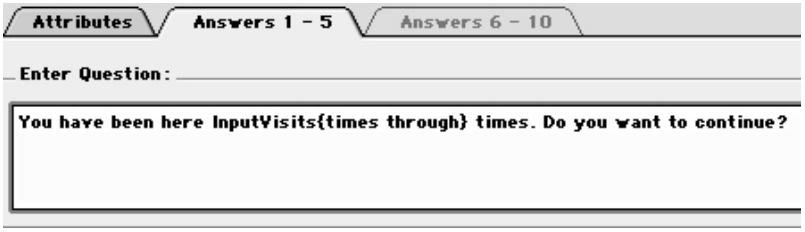


Figure 13.18: After closing the Variable Selection dialog, the variable is inserted into the question.

6. Create a Path from the first Exit Point (which corresponds to "Yes") up to the Input State's Entrance Point. Then create a Path from the second Exit Point (which corresponds to "No") to the Stop State (Figure 13.19).

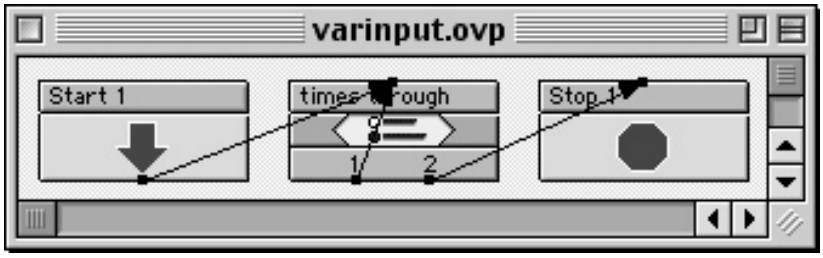


Figure 13.19: The completed Application Map. Notice the Path connecting Exit 1 of "times through" to the Entrance.

Connecting a State's Exit Point to its own Entrance Point will cause the application flow to loop through this State until the user chooses to exit.

7. Choose **Run** (⌘ R) from the **Map** menu.

When the application runs, you will be informed that "You have been here 0.00 times. Do you want to continue?" Each time you choose the **Yes** radio button and click **OK**, the number of visits will increase by one. Choose **No** and click **OK** to stop the test run and return to the Application Map (Figure 13.20).

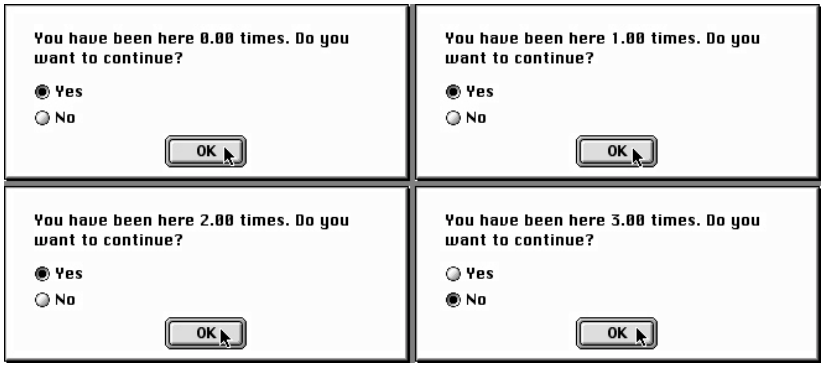


*Figure 13.20: Each time the user goes through the Input, the **Number of Visits** variable is increased by one. Clicking the **No** button stops the test run.*

Creating Custom Variables

onViz offers a wide variety of built-in variables for use in your application, but there are times you may want to create your own variables for tracking statistics. For example, you may want to track how many items on a display the user has clicked. onViz does not have a built-in variable to track this information. However, you can create a custom variable whose value is incremented each time the user clicks an item. You can display this variable to the user with variable substitution or use it in a conditional path to make branching decisions.

In addition, there may be information you want to carry between different onViz documents. Remember that variables associated with specific Outputs, Inputs, and Groups, such as **Input Last Answer**, **Sprite Hit**, and **Group Exit**, are available only in the document in which the specified state is located. Because Custom Variables are stored in a library, if you want to make one of these variables available in another document, you create a custom variable. Use

a calculator in the document containing the state with the information you want to pass to write the value to the custom variable. You can then access the custom variable in other documents for variable substitution or application flow decisions.

An initial value can be written to a custom variable when it is created in the Custom Variables Library, or the value can be modified during the running of the application by using a Calculator State. Once a custom variable has a value written to it, it will retain that value until it is either reset, or has another value written to it. For more information on custom variables and the Custom Variables Library, see chapter 4, Libraries.

The following three-step example demonstrates how to create a custom variable, write a value to it, and to read that value. The steps to creating this example are: creating the application map, creating the custom variable, and using a Text Calculator to equate the custom variable with an Input variable. The user's name will be entered by the user in an Enter Text Input in the top-level application, and written to a custom variable with a Text Calculator. The user's name will then be read from the custom variable and displayed.

1) Creating the top-level application

1. Create a new top-level application, and place 1) a Start State, 2) an Enter Text Input, 3) a Text Calculator, 4) a Radio Button Input, and
   5) a Stop State on the Application Map. Connect the states in the order listed. Name the Enter Text Input "name" and the Radio Button Input "hello." It is important to give your new States meaningful names, especially the Enter Text Input State, as you will be accessing an Input Variable associated with this State..

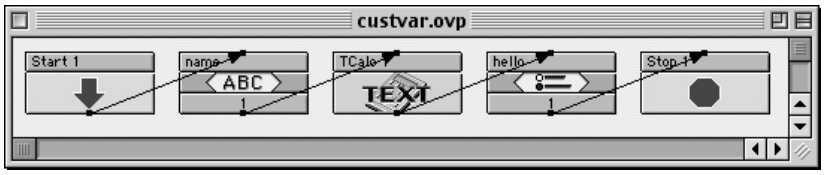


Figure 13.22: The Application Map from steps 1 and 2.

2. Open the Input State InfoCenter. Under the Attributes tab, choose **1** from the **Number of Answers:** dropdown menu. Under the Answers 1-5 tab, type "What is your name?" into the **Enter Question:** field, then click **OK (Figure 13.21)**.

*Figure 13.21: Type "What is your name?" into the **Enter Question** field.*

2) Creating the custom variable

1. Choose **Custom Variables Library** (⌘ 3) from the **Libraries** menu to open the Custom Variables Library.
2. Click the **New Text** button. Note: if this button is dimmed, select the **Show All** option from the drop down menu at the top of the dialog.

   We use a text variable because our new custom variable will hold a text value. The **New Text** button creates a new untitled variable; the "T" preceding "Untitled" indicates that it is a text variable. Notice that the name "Untitled" also appears in the **Selected Variable Name** field on the right side of the window, where it is highlighted.
3. Type a name for the variable in the **Selected Variable Name** field, for example

"NameInput," and type a short description of the variable in the **Description** field, for example "User's name input in Part 1," then click **Close (Figure 13.23)**.

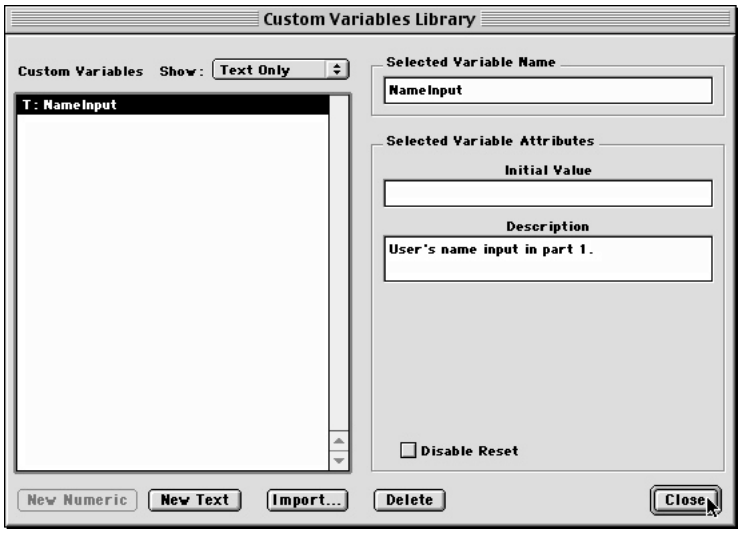


Figure 13.23: The Custom Variables Library with the new Text variable "NameInput."

The **Close** button closes the Custom Variables Library and returns you to the Application Map.

3) Tying it all Together

1. Open the Text Calculator's InfoCenter by double-clicking anywhere on its icon, or by selecting it and choosing **Calculator InfoCenter** (⌘ I) from the **Map** menu.

   When the Calculator InfoCenter opens, the first equation field is highlighted.

   The Text Calculator will be used to set the new custom variable equal to the **Answer Entered (Text)** input variable. When the application is run, the **Answer Entered (Text)** variable captures the user's name as entered in the Input State. By equating it with the custom variable, you make it available to any supporting documents.

2. To begin the equation, click the box labeled **?**.

   The Variable Selection dialog is opened, with all of the categories of variables represented by tabs.

3. Click the Custom Variable tab to display the list of custom variables, click on the custom variable you created to choose it, and then click the **Select** button.

   Clicking the **Select** button adds the custom variable to the Text Calculator InfoCenter on the left side of the equal sign (Figure 13.24). You now need to fill in the right side of the equation.

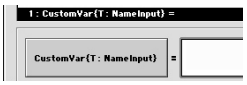


Figure 13.24: The left-hand expression of a Text Calculator equation.

4. Click the **Input** button to fill in the rest of the equation with an Input variable.

   The Variable Selection dialog is opened again, this time with the list of Input variables displayed. An Input variable is being used here because the user's name was entered in an Input State.

5. Choose the name of your Input State from the list on the left, choose **Answer Entered (Text)** from the Input Variables list on the right, then click **Select**.

   Clicking the **Select** button closes the Variable Selection dialog and adds the **Answer Entered (Text)** variable to the right side of the equation, where it completes the calculation (Figure 13.25).

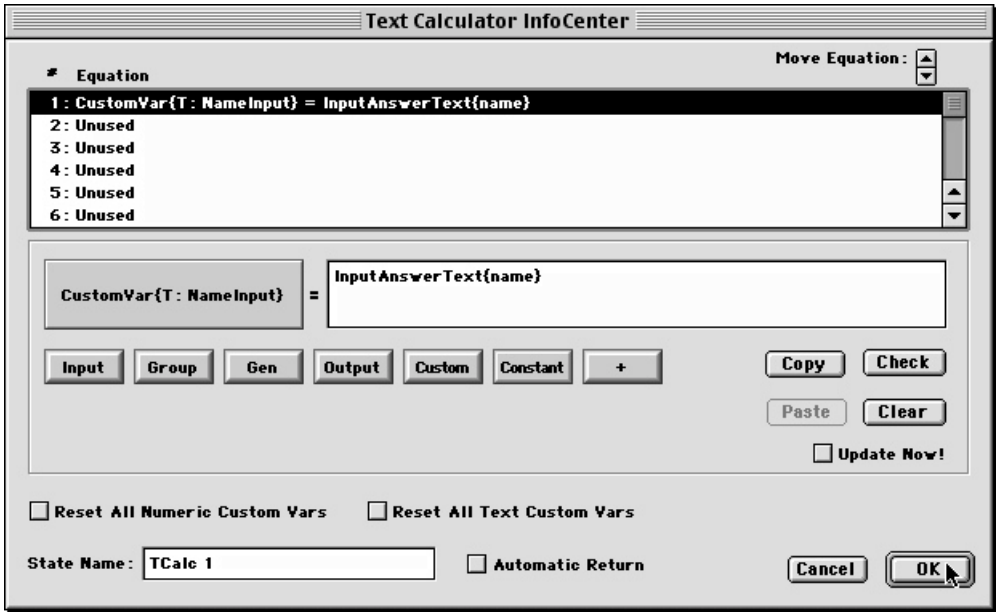


Figure 13.25: The Text Calculator with a completed equation. Click OK to close the InfoCenter and return to the Application Map.

6. Click **OK** to close the Text Calculator InfoCenter and return to the Application Map.
7. Open the Radio Button Input InfoCenter. Under the Attributes tab, choose 1 from the **Number of Answers:** dropdown menu. Under the Answers 1-5 tab, type "Hello," into the **Enter Question:** field.
8. Position the cursor after the colon, and click while holding down the Option key.

   Holding down the Option key while clicking opens the Variable Selection dialog.
9. Select the Custom Variable tab, click your custom variable's name ("NameInput" in this example) from the list of variables on the left, and then click **Select**.

   Clicking the **Select** button returns you to the Input InfoCenter. The **Enter Question:** field should read "Hello, CustomVar{T: NameInput}" (Figure 13.26).

Figure 13.26: Pressing the Option key while clicking opens the Variable Selection dialog (left). The Input InfoCenter with the chosen variable (right).

10. Click **OK** to close the Input InfoCenter and return to the Application Map.

    You are now ready to give your application a test run.
11. Choose **Run** (⌘ R) from the **Map** menu, and type your name when prompted.

Your name is captured by the **Answer Entered (Text)** variable, and written to your custom variable by the Text Calculator State. Your name is then displayed in a personal greeting.

Using Variables to create a Custom Application Report

Input Variables

Read Only

- Answer Expected (InputAnswerExpected{State}) – This text variable represents the text from the answer field(s) designated as **Correct Answer(s)** in an Input's InfoCenter.

  Since this information can only be changed by updating the text in the InfoCenter, it is read only.

- Exit Path Expected InputExit Expected{State}) – This text variable represents the Exit Path(s) for the answer field(s) designated as **Correct Answer(s)** in an Input's InfoCenter.

  Since this number can only be changed by updating the Correct Answer designation in the InfoCenter, it is read only.

- Percent Score (InputPercentScore{State}) – This numeric variable represents the calculation of a user's actual score for an Input divided by the total possible score for the Input, and is updated each time the Input is exited. The Input must have a designated **Correct Answer** and a value in the **Points if Correct** field in order for this statistic to be tracked.

  Since this information is a calculation based on Total Score statistics, it is read only.

- Percent Correct (InputPercentCorrect{State}) – This numeric variable represents the calculation of the number of times a user selected an Input's correct response divided by the number of times the Input was entered, and is updated each time the Input is exited. The Input must have a designated **Correct Answer** in order for this statistic to be tracked.

  Since this information is a calculation based on Total Correct statistics, it is read only.

- *n* Answer Selected (Answer*n*Selected{State}) – *n* = answer number (1-10). These numeric variables are used with Check Boxes Inputs. There is a unique "Answer Selected" variable associated with each of the 10 possible answer fields. If the user selects a specific check box answer, its "Answer Selected" variable is equal to 1; if the check box is not selected, its "Answer Selected" variable is equal to 0.

Write Only

- Reset Input (ResetInput{State}) – Flowing the application through a Calculator that contains an equation setting the value of this variable equal to 1 resets all variables associated with designated Input.

Read/Write

- Answer Entered (InputAnswer{State}) – This text variable represents the answer entered by the user.

  In the case of Radio Button Inputs, this variable represents the text of the selected answer.

  In the case of Check Boxes Inputs, this variable represents the text from each selected

answer. If more than one answer is chosen, each answer's text is separated by a comma and space.

In the case of Mouse Inputs, this variable represents the Mouse Bay number the user clicked.

This variable's value can be updated using a Text Calculator. An example of its use in an application would be to preset an Input's **Prompt With Last Answer** option, which will fill in the answer field with the value set in the Calculator.

- Time to Answer (InputTime{State}) – This numeric variable represents the amount of time the user has spent in the designated Input, and is reset each time the Input is entered.

  This variable's value can be updated using a Numeric Calculator. An example of its use in an application would be to clear the time to answer without resetting other Input statistics.

- Total Score (InputTotalScore{State}) - This numeric variable represents the user's score for the Input. Unless the **Score First Try Only** option is selected, this number is cumulative with each time the user enters the Input and answers the question correctly. The Input must have a designated **Correct Answer** and a value in the **Points if Correct** field in order for this statistic to be tracked.

  This variable's value can be updated using a Numeric Calculator. An example of its use in an application would be to set this value based on Smart Sprite selections.

- Total Correct (InputTotalCorrect{State}) – This numeric variable represents the number of times the user has provided a correct answer for the designated Input. Unless the **Score First Try Only** option is selected, this number is cumulative with each time the user enters the Input and answers the question correctly. The Input must have a designated **Correct Answer** in order for this statistic to be tracked.

  This variable's value can be updated using a Numeric Calculator. An example of its use in an application would be to set this value based on Smart Sprite selections.

- Number of Visits (InputVisits{State}) – This numeric variable represents the number of times the user has visited the designated Input.

This variable's value can be updated using a Numeric Calculator. An example of its use in an application would be to clear the number of visits without resetting other Input statistics.

- Exit Path (InputExit{State}) – This numeric variable represents the Exit number taken from the designated Input.

  This variable's value can be updated using a Numeric Calculator.

- Last Sprite Dragged (LastSpiteDragged{State}) – This numeric variable represents the family number of the most recent sprite dragged by the user.

  This variable's value can be updated using a Numeric Calculator. An example of its use in an application would be to clear this value without resetting other Input Statistics.

- *n* Bay's Sprite (Bay*n*Sprite{State}) – *n* = Mouse Bay number (1-10). This numeric variable represents the sprite family number(s) of sprites that are within the boundary of the designated Mouse Bay. There is a unique "Bay's Sprite" variable associated with each of the 10 possible Mouse Bays.

  This variable's value can be updated using a Numeric Calculator.

Output Variables

Read Only

- Total Frames (TotalFrames{State}) – Returns a value representing the total number of frames in the designated Output. This frame is read only because you cannot change the number of frames that were authored in the Output.

Write Only

- Reset Output (ResetOutput{State}) – Flowing the application through a Calculator that contains an equation setting the value of this variable equal to 1 resets all variables associated with designated output – including resetting sprite variables according to how the Output was authored.

Read/Write

- Design Window Visible (DesignWinVisible{State}) – Expected Values: 0 = invisible, 1 = visible. Reading this number provides information about the designated Design Window's visibility. Writing to this variable affects the visibility of the designated Design Window. (Note: this variable will show the designated Design Window even if originally authored as a Text or No Window output)
- Text Window Visible (TextWinVisible{State}) – Expected Values: 0 = invisible, 1 = visible. Reading this number provides information about the designated Text Window's visibility. Writing to this variable affects the visibility of the designated Text Window. (Note: this variable will show the designated Design Window even if originally authored as a Design or No Window output)
- Number of Visits (OutputVisits{State}) – Reading this variable returns a value representing the total number of times the application has flowed through the designated Output. Writing to this variable sets the number of visits to the value assigned.
- Sprite Hit (SpriteHit{State}) – Reading this variable returns a value representing the sprite family number that was clicked by the user. Writing to this variable is typically used to reset it.

The following variables are replicated for each sprite family 1 – 64. *n* stands for the sprite family number.

- *n* Sprite's Pose (Sprite*n*Pose{State}) – Expected Values: -3 - 16. Reading this variable returns the pose currently being displayed by the designated sprite.
  Writing a value between 1 and 16 to this variable displays the specified pose of the designated sprite.
  Writing a value between 0 and –3 to a this variable causes the sprite to cycle through all of its poses – typically while waiting for user input. The more negative the number, the slower the cycling.
- *n* Sprite's Drag (Sprite*n*Drag{State}) – Expected Values: 0 – not draggable, 1 –drag any

direction, 2 – horizontal drag, 3 – vertical drag. Reading this variable gives information about the specified sprite's draggability. Writing to this variable makes specified sprite draggable or not and can constrain the drag to horizontal or vertical.

- *n* Sprite's Visible (Sprite*n*Drag{State}) – Expected Values: 0 = invisible; 1 = visible. Reading this variable gives information about the designated sprite's visibility. Writing to this variable affects the visibility of the specified sprite.
- *n* Sprite's Bay (Sprite*n*Bay{State}) – Expected Values: 0 – 10. This variable is typically used with draggable sprites. Reading its value returns a number representing the Mouse Bay region (if any) in which the specified sprite is positioned. Writing to this variable positions the center point of the designated sprite in the center of the designated Mouse Bay region.
- *n* Sprite's X Position (Sprite*n*X{State}) - Values: determined by screen size. Reading this variable returns a number equal to the designated sprite's horizontal (x) position on the display, as measured from the sprite's center point. Writing to this variable moves the center point of the designated sprite to the specified horizontal (x) position on the display.
- *n* Sprite's X Minimum (Sprite*n*XMin{State}) - Values: determined by screen size. Used with draggable sprites. Reading this variable returns a number, representing the distance from the left side of the display, that the user can drag a specified sprite. Writing to this variable designates how far to the left the user can drag the designated sprite. The lower the number, the farther to the left the sprite can be dragged.
- *n* Sprite's X Maximum (Sprite*n*XMax{State}) - Values: determined by screen size. Used with draggable sprites. Reading this variable returns a number, representing the distance from the right side of the display, that the user can drag a specified sprite. Writing to this variable designates how far to the right the user can drag the designated sprite. The higher the number, the farther to the right the sprite can be dragged.
- *n* Sprite's Y Position (Sprite*n*Y{State}) - Values: determined by screen size. Reading this variable returns a number equal to the designated sprite's vertical (y) position on the display, as measured from the sprite's center point. Writing to this variable moves the center point of the designated sprite to the specified vertical (y) position on the display.
- *n* Sprite's Y Minimum (Sprite*n*YMin{State}) - Values: determined by screen size. Used with draggable sprites. Reading this variable returns a number, representing the distance from the top of the display, that the user can drag a specified sprite. Writing to this variable designates how far toward the top of the display the user can drag the designated sprite. The lower this number, the higher the sprite can be dragged.
- *n* Sprite's Y Maximum (Sprite*n*YMax{State}) - Values: determined by screen size. Used with draggable sprites. Reading this variable returns a number, representing the distance from the bottom of the display, that the user can drag a specified sprite. Writing to this variable designates how far toward the bottom of the display the user can drag the designated sprite. The higher this number, the lower the sprite can be dragged.

Group Variables

Most Group Variables track Input statistics as a collection within the designated Group. These variables are cumulative, and update as each Input exits.

Read Only

- Percent Score (GroupPercentScore{State}) - This numeric variable represents the calculation of a user's actual score, divided by the total possible score for all visited Inputs in the designated Group, and is updated each time the Group is exited. You must have correct answers and scores designated for the Inputs in order for this statistic to be tracked.
- Percent Correct (GroupPercentCorrect{State}) - This numeric variable represents a calculation of a user's number of correct responses, divided by the number of visited Inputs, in the designated Group, and is updated each time the Group is exited. The Inputs must each have a designated **Correct Answer** and a value in the **Points if Correct** field in order for this statistic to be tracked.

Write Only

- Reset Group (GroupReset{State}) - Flowing the application through a Calculator that contains an equation setting the value of this variable equal to 1 resets all variables associated with the designated Group. This includes resetting any Input or Output States within the Group.

Read/Write

- Time Spent in Inputs (GroupTimeInInputs{State}) - This numeric variable represents the time a user has spent in the designated Group's Input States. You can write to this variable and modify its value using a Calculator.
- Total Score (GroupTotalScore{State}) - This numeric variable represents a user’s score for all Inputs within the designated Group. You can write to this variable and modify its value using a Calculator.
- Total Correct (GroupTotalCorrect{State}) - This numeric variable represents a user's number of correct responses for all Inputs within the designated Group.
- Exit Path (GroupExit{State}) - This numeric variable represents the Exit path taken from the Group.

General Variables

Read Only

- Application's Total Score (AppScore{}) - This numeric variable represents a user's score for all visited Inputs in the application. This value is cumulative, and updates as each Input is Exited.
- Application's Total Correct (AppCorrect{}) - This numeric variable represents a user's total number of correct answers for all visited Inputs in the application. This value is cumulative, and updates as each Input is exited.
- Application's Percent Score (App%Score{}) - This numeric variable represents a user's actual score divided by the total possible score for all visited Inputs in the application. This value is cumulative, and updates as each Input is exited.
- Application's Percent Correct (App%Correct{}) - This numeric variable represents a user's number of correct answers divided by the number of all visited Inputs in the application. This value is cumulative, and updates as each Input is exited.
- Document's Total Score (DocScore{}) - This numeric variable represents a user's score for all visited Inputs in the current document. This value is cumulative, and updates as each Input is exited.
- Document's Total Correct (DocCorrect{}) - This numeric variable represents a user's total number of correct answers for all visited Inputs in the current document. This value is cumulative, and updates as each Input is exited.
- Document's Percent Score (Doc%Score{}) - This numeric variable represents a user's actual score divided by the total possible score for all visited Inputs in the current document. This value is cumulative, and updates as each Input is exited.
- Document's Percent Correct (Doc%Correct{}) - This numeric variable represents a user's number of correct answers divided by the number of all visited Inputs in the current document. This value is cumulative, and updates as each Input is exited.
- Last Animation Frame (LastAnimFrame{}) - This numeric variable represents the frame number of the last animation frame displayed. Because it is based on frame number, it is not specifically associated with any one Output. You can use this variable in a Conditional Path to make decisions about application flow based on the last displayed frame.
- Mouse Button Down (MouseDn{}) - This numeric variable returns a value to indicates the state of the mouse button. Typically used in a Conditional Path, you can determine if the mouse button is down (equal 1) or up (equal 0). For Windows runtime, this value reads the left mouse button only.
- Random Number from 1 to 100 (RandomNum{}) - This numeric variable automatically generates a random number from 1 to 100. You can create a custom variable and set it equal to this Random Number variable, which will generate a number from 1 to 100 to fill in the custom variable. Note that if your range will be any

other than 1 to 100, use the Random display format for your custom variable, which allows you to select different ranges.

- Current Time (Text) (Time{}) - This numeric variable is filled in with the current time, as read from the computer's system clock setting. The display format is hours:minutes:seconds am/pm. You can create a custom numeric variable to display this value in a different format.
- Current Date (Text) (Date{}) - This numeric variable is filled in with the current date, as read from the computer's system date setting. The display format is m/d/y. You can create a custom numeric variable to display this value in a different format.
- Day of the Week (DayOfWeek{}) - This numeric variable is filled in with the current day of the week, as read from the computer's system setting. The days are numbered starting with Sunday being day 1 and Saturday being day 7. You can create a custom numeric variable to display this value in a different format.
- Day of the Month (Day{}) - This numeric variable is filled in with the current day of the month, as read from the computer's system date setting.
- Month of the Year (Month{}) - This numeric variable is filled in with the current month, as read from the computer's system date setting. You can create a custom numeric variable to display this value in a different format.
- Year (Year{}) - This numeric variable is filled in with the current year, as read from the computer's system setting. You can create a custom numeric variable to display this value in a different format.
- Hour of the Day (Hour{}) - This numeric variable is filled in with the current hour, as read from the computer's system clock setting.
- Minute of the Hour (Minute{}) - This numeric variable is filled in with the current minute, as read from the computer's system clock setting.
- Second of the Minute (Second{}) - This numeric variable is filled in with the current second, as read from the computer's system clock setting.
- Platform Type (PlatformType{}) - This numeric variable checks the computer type on which the application is running and returns X if the computer is a Macintosh and Y if the computer is Windows based.
- Display Number of Colors (DispColors{}) - This numeric variable returns a number representing the color depth of the display on which the application is running.
- Display Width (Pixels) (DispWidth{}) - This numeric variable returns a number representing the width (in pixels) of the display on which the application is running.
- Display Height (Pixels) (DispHeight{}) - This numeric variable returns a number representing the height (in pixels) of the display on which the application is running.
- QuickTime Version Number (QkTimeVer{}) - This numeric variable returns a number representing the version of QuickTime installed on the computer on which the application is running.
- Free Disk Space (FreeDiskSpace{}) - This numeric variable returns the amount of free disk space (in kilobytes) of the volume on which the application is running.

Write Only

Note: onViz contains six variables that give control of a 'System Pen.' This System Pen draws in an Output Window based on values assigned with a Calculator. For example, you can use the System Pen variables to draw a sine wave based on a user's input.

- Draw Move to X Position (MoveX{}) - This variable describes the X position from which the System Pen will begin drawing.
- Draw Move to Y Position (MoveY{}) - This variable describes the Y position from which the System Pen will begin drawing.
- Draw Pen Size (PenSize{}) - This variable describes the width of the System Pen.
- Draw Pen Color (PenColor{}) - This variable describes the color of the System Pen. Note that color the color value is defined as the percentage of Red, Blue, and Green it contains, with 100 being full value and 000 being no value. For example, 000000000 is pure white, 100000000 is pure red, 000100000 is pure blue, 000000100 is pure green, and 100100100 is pure black.
- Draw Line Style (LineStyle{}) - This variable describes the style of the System Pen. Expected values: 1 = solid line, 2 = dotted line, 3, 4, & 6 = combination of dots and dashes, and 5 = dashed line (Note that these values correspond to the line style menu in the Image Editor.)
- Draw Line with Arrowheads (LineArrow{}) - This variable determines whether the line drawn by the System Pen will contain arrowheads. Expected values: 0 = no arrowhead, 1 = arrowhead at the origin of the line, 2 = arrowhead at the termination of the line, and 3 = arrowheads at both ends of the line (Note that these values correspond to the line style menu in the Image Editor.)
- Delay in Seconds (DelayInSecs{}) - Writing a value to this variable will pause the application for the designated number of seconds.

Read/Write

- Application's Name (Text) (AppName{}) - This text variable represents the file name of the top level application. The value of this variable can be changed with a Text Calculator.
- Document's Name (Text) (DocName{}) - This text variable represents the file name of the current document. The value of this variable can be changed with a Text Calculator.
- User's Name (UserName{}) - If the **Ask for Name** option (from the Project Attribute's Reports dialog) was selected during authoring, this text variable holds the name the user entered when beginning the application. You can write a new value to this variable using a Text Calculator. For example, you can use a Text Input State to ask the user's name, and follow this Input with a Text calculator that sets the "User's Name" General Variable equal to the "Input Last Answer" variable.
- User's Password (UserPswd{}) - If the **Use Passwords** option (from the Project Attribute's Reports dialog) was selected during authoring, this text variable holds the password the user entered when beginning the application.

- Application's Time so Far (AppTime{}) - Reading this numeric variable returns the time in seconds since the user first started the application. You can create a custom numeric variable to display this value in a different format, and you can write to it to set it to the desired value. A typical use of writing to this variable is to reset the application's time without resetting all application variables.
- Document's Time so Far (DocTime{}) - Reading this numeric variable returns the time in seconds since the user first entered the current document. You can create a custom numeric variable to display this value in a different format, and you can write to it to set it to the desired value. A typical use of writing to this variable is to reset the document's time without resetting all document variables.
- Animation Current Frame (AnimCurFrame{}) - Reading this numeric variable returns the animation frame number displayed in the current Output. You can use this variable in a Conditional Path to make decisions about application flow based on the current frame. Writing to this variable updates the frame currently being displayed to the frame that corresponds to the value set with the Calculator.
- Animation Start Frame (AnimStartFrame{}) - Reading this numeric variable returns a value that will be the starting frame number for the next Output encountered in the application. You can use this variable in a Conditional Path to make decisions about application flow based on the start frame. Writing to this variable tells onViz the starting frame number for the next Output to be encountered.
- Animation Stop Frame (AnimStopFrame{}) - Reading this numeric variable returns a value which is the frame number at which the animation will stop for the next Output encountered in the application. You can use this variable in a Conditional Path to make decisions about application flow based on the stop frame. Writing to this variable tells onViz the stop frame number for the next Output to be encountered.
- Movie(s) Playing (MovPlaying{}) - Expected values: 0 = no movie playing, 1 = movie playing. Reading this numeric variable returns the value corresponding to the status of any and all movies. You can use this variable in a Conditional Path to make decisions about application flow based on whether a movie is playing or not. Writing a 0 to this variable stops all currently playing movies, writing a 1 to this variable has no effect, as movies can only begin in an Output or a Multimedia State.
- Sound(s) Playing SndPlaying{}) - Expected values: 0 = no sound playing, 1 = sound playing. Reading this numeric variable returns the value corresponding to the status of any and all sounds. You can use this variable in a Conditional Path to make decisions about application flow based on whether a sound is playing or not. Writing a 0 to this variable stops all currently playing sounds, writing a 1 to this variable has no effect, as sounds can only begin in an Output or a Multimedia State.
- Cursor (Cursor{}) - Expected values: 0 = cursor hidden, 1 = cursor visible. Reading this numeric variable gives the current status of the Cursor. Writing to this variable makes the cursor visible or invisible. Note that a Cursor Gadget can also be used to affect the visibility of the cursor.

- Current Mouse X Position (CurMouseX{}) - Reading this numeric variable returns the current X position of the cursor. Writing to this variable moves the cursor to the designated X position.
- Current Mouse Y Position (CurMouseY{}) - Reading this numeric variable returns the current Y position of the cursor. Writing to this variable moves the cursor to the designated Y position.
- Last Mouse Click X Position (LastClkX{}) - Reading this numeric variable returns the X position of the location the user last clicked. Writing to this variable is typically used to "clear" the mouse click.
- Last Mouse Click Y Position (LastClkY{}) - Reading this numeric variable returns the Y position of the location the user last clicked. Writing to this variable is typically used to "clear" the mouse click.
- Last Key Pressed (Text Character) (KeyChar{}) - Reading this text variable returns the keyboard character last pressed. Writing to this variable is typically used to "clear" the key press.
- Last Key Pressed (Numerical Value) (KeyVal{}) - Reading this numeric variable returns the ASCII equivalent of the keyboard character last pressed. Writing to this variable is typically used to "clear" the key press.
- Set or Get Sound Volume (SndVolume{}) - Reading this numeric variable returns the current sound volume, as set in the user's current volume setting. Writing to this variable changes the volume setting on the user's computer. Note that when your onViz application exits, the volume is returned to its original settings. The volume can also be modified with a Monitor and Sound Gadget.
- Set or Get Timer Time (TimerTime{}) - Reading this numeric variable returns the remaining time on the Gadget Timer. Writing to this variable sets the remaining time to the value entered.
- Number Decimal Points (NumDecPts{}) - Reading this numeric variable returns the number of decimal points set to be shown when displaying any variable (not set to an integer). Writing to this variable sets the number of decimal points that will be shown when displaying a variable.
- Error Number (Error{}) - Reading this numeric variable identifies a specific error that onViz encountered during playback of the applications. Writing to this variable is typically used to "clear" the error number.

Chapter 14

Calculator States

Covered in this Chapter:

- Using Calculator States
- Numeric Calculator
- Text Calculator

Using Calculator States

Calculator States are used to ***write*** values to variables. Variables can be thought of as containers that hold information, and can be used in different ways. The information stored in a variable can be ***read***, in which case the variable is either displaying the information it contains, or being compared with another value in a Conditional Path to make a branching decision. Or a variable can be ***written to***, in which case the information it contains is modified or updated with a Calculator.

If, for example, you want to hide the Text Window of a Design and Text Output, send the application flow through a Calculator State in which the **Text Window Visible** variable is set equal to zero (zero equals invisible, one equals visible). When your application flows through the Calculator State, the Text Window of the designated Output State will be hidden. To make it visible again, simply route your flow through another Calculator State that has the same variable set equal to one (visible).

Or, perhaps you want to send the user to a specific location when they have clicked a designated number of items on a Design Window. You can create a custom variable and use a Calculator to increment the variable each time the user clicks an item. A Conditional Path will check the value of the variable at the exit of the Calculator and either route the user back to the Design Window for additional selections or, if the designated number has been reached, on to a different section of your application.

As your applications increase in their level of interactivity, you will find Calculators an invaluable tool in your development – from changing sprite poses, to creating randomized mathematical equations, to updating on-screen text. As with any new tool, start simply and soon you will have an understanding of how Calculators can dynamically adapt your application and personalize it to your user. Refer to Chapter 13 for a complete listing of onViz' built-in variables and a description how their values can be modified with Calculators.

Because Calculators are not actually presenting information to your user, they have no Presentation windows. Double-clicking anywhere on the state icon opens directly to the InfoCenter, which is used for creating the calculation that writes to a designated variable. Though the InfoCenter differs for Numeric Calculators and Text Calculators, many of the fields are the same, and they both function in essentially the same way. The top portion of both InfoCenters contains a list of up to 32 different equations that assign values to variables. The lower portion of the both InfoCenters is used for creating the equations (Figure 14.1).

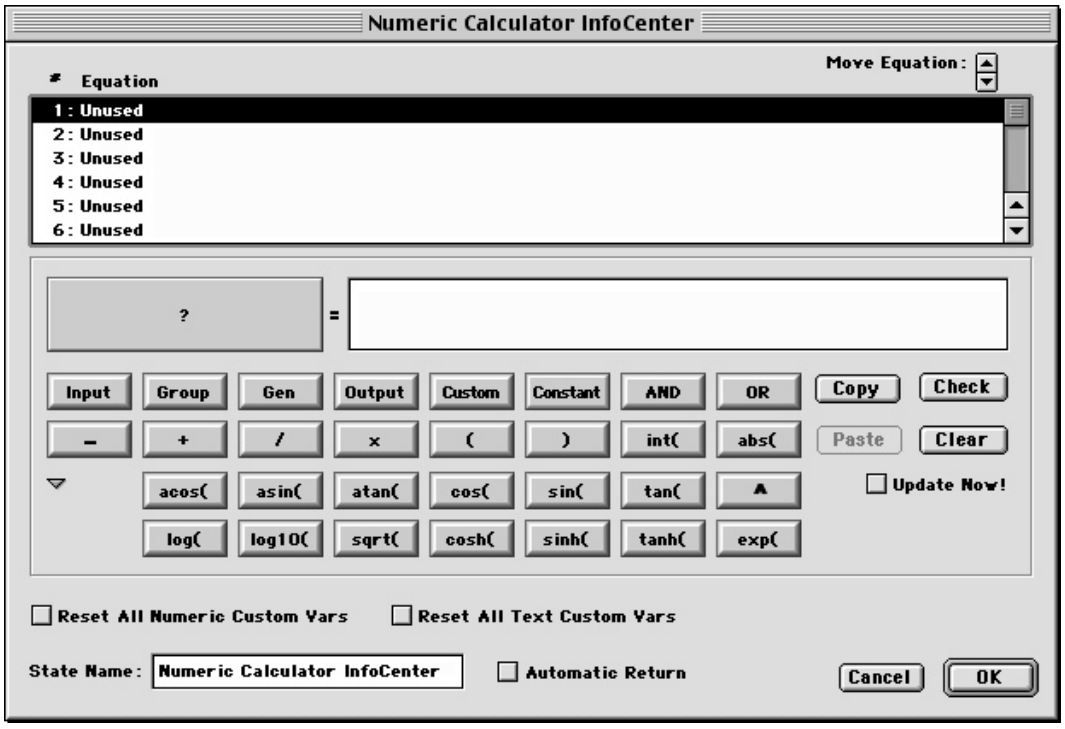


Figure 14.1: A Numeric Calculator InfoCenter. The Text Calculator InfoCenter is essentially the same, but does not have as many mathematical functions.

Calculator equations are made up of two expressions separated by an equal sign. The left-hand expression is the variable you want to change and is selected by clicking the button marked with a **? (question mark)**, which displays the Variable Selection dialog. The right-hand side of the equation can be a single value or an expression and is created by clicking one of the variable buttons or by typing a constant. Numeric Calculators have the option of also choosing from a series of math functions, such as **+**, **-**, **x**, **/**, **cos**, **sin**, etc., to fill in the right-hand expression.

For example, in the equation described above, in which a Text Window is hidden, a Numerical Calculator is used. Since the variable we want to modify is the Output variable **Text Window Visible**, this variable is selected for the left-hand side of the calculation. This variable is then set equal to the constant 0 in the right-hand expression field to hide the Text Window. The resulting equation appears as **TextWinVisible{OutputName}=0 (Figure 14.2)**.

Figure 14.2: An equation in a Numeric Calculator InfoCenter. At top is the equation itself; at bottom are two of the fields used to create the equation.

Routing the application through this Numeric Calculator before it enters the chosen Output will display the Design Window only with the Text Window hidden.

The InfoCenter is accessed by double-clicking anywhere on the State's icon.

While a State's name can be changed from within its InfoCenter window, it can also be changed without leaving the Application Map.

Numeric Calculator

The Numeric Calculator is used to modify and/or assign values to numeric variables. These variables can be either onViz' built-in variables, or a Custom Numeric variable. The Numeric Calculator lets you set up a variety of operations. Some of its applications include:

- Updating sprite position/pose/visibility.
- Manually updating input scores.
- Setting a variable (typically a Custom Numeric variable) equal to mathematical equations, such as analyzing selected Input scores.
- Equating a custom variable with a selected numerical format in order to modify the display of a built-in date and time variable.
- Creating math questions based on random numbers.

The Numeric Calculator InfoCenter

The Numeric Calculator InfoCenter is divided into three sections: an equation list, a section for creating the equation, and a section for naming and managing the Calculator State. The equation list holds up to 32 equations; if a Calculator contains more than one equation, calculations are performed according to their order in the list. An equation's order in the list can be changed by selecting it and using the **Move Equation** up/down arrows.

When an equation writes a value to a variable, the variable is updated when the application flow leaves the Calculator State. As a result, if the Calculator contains more than one equation operating on the same variable, the variable will be simultaneously updated with all the calculations when the Calculator is exited. To update a variable's value as soon as it is calculated, select the equation in the list and place a check mark in the **Immediate Update** check box.

For example, if your Calculator contains multiple equations affecting sprite placement in an Output, the sprite will be almost immediately moved to all of the positions, and left in the location designated by the last equation, as soon as the Calculator is exited. However, if the **Immediate Update** option is selected for these equations, the sprite will be moved one location at a time as soon as the calculation is encountered in the list. An equation with a delay variable can be inserted between the position move to slow the Calculator as it works through the equations to move the sprite.

Equations are composed of a variable on their left-hand side and a value that is being assigned to that variable on their right. An equation is created by selecting one of the numbered lines in the equation list and clicking the left-hand variable button (labeled with a **?**), which opens the Variable Selection dialog. When choosing a variable for the left-hand side of the equation in a Numeric Calculator, the Variable Selection dialog displays only numeric variables. When choosing a variable for the left-hand side of the equation in a Text Calculator, the Variable Selection dialog displays only text variables. After a variable is selected, its name is displayed in the box formerly labeled with a question mark (Figure 14.3).

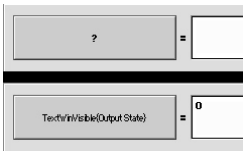


*Figure 14.3: The left-hand variable button is labeled with a **?** before a variable is chosen (top). After a variable is chosen, the button displays the variable (bottom).*

The right-hand side of the equation is completed by first clicking in the edit field and then clicking:

- One of the variable buttons.
- The **Constant** button.
- One of the mathematical or trigonometric function buttons.
- The **AND** or **OR** buttons.

The last two rows of buttons are mathematical and trigonometric functions. Clicking the disclosing triangle below the "Minus" button will toggle these buttons off and on. While, unless you are creating rather complex mathematical equations, you will probably rarely use most of these functions, you may occasionally find yourself using **abs(** or **int(**.

abs(gives a number's absolute value; for instance, abs(2) = 2; abs(-9) = 9.

int(displays fractions as integers; for instance, int(1/2) = .5; int(7/4) = 1.75.

The **Copy** and **Paste** buttons allow you to quickly create equations that are similar to each other; differing, for example, only in the sprite family number to be affected. To copy an equation, select the equation and press the **Copy** button or hold down the command and shift key while pressing C (⌘ ⇧ C). This places the equation on the clipboard. Click an empty equation line and click **Paste** or hold down the command and shift key while pressing P (⌘ ⇧ P) to paste the equation on the clipboard, or click an existing equation and then click **Paste** (⌘ ⇧ P) to replace the equation. You can then modify the equation as needed. Equations including multiple operations are calculated using standard mathematical convention that gives precedence to parenthetic expressions. For example to set the variable to 1/2 of the sum of 10 and 5, the equation would read **variable = (10+5)/2**.

The **Check** button analyzes the syntax of the selected equation and reports any errors, such as if the parentheses are not matched on the right-hand side expression.

If your application contains custom variables, there may be times when you need to reset these variables to their initial values (a custom variable's initial value is set in its Custom Variables Library entry). Route your application's flow through a Calculator with the R**eset all Numeric Custom Vars** and/or **Reset all Text Custom Vars** options selected, and, when flow exits the Calculator, these custom variables will be reset to their initial values

Note that these options reset the values for every text or numeric variable in your application. If there are custom variables in your application that you do not want reset, open their Library entries and select the **Disable Reset** option. A custom variable with this option selected is not affected by the Calculator's "Reset" options.

Selecting the **Automatic Return** option functions the same as following the Calculator with a Return on the Application Map. If the Calculator is called with a Bridge Target, checking this option automatically returns flow back to the Bridge's Exit Point, without needing to add a Return State to your Application Map. See chapter 5, Flow States, to learn how to use Returns in your application.

Notice that by selecting **Automatic Return**, the Calculator State icon no longer displays an Exit Point; this is because the **Automatic Return** option always routes flow back to the Bridge

State that called it, eliminating the need for a path from the Exit.

Writing to a Variable using a Numeric Calculator State

The first part of this example uses a Calculator to change a sprite's X and Y coordinates. The second part demonstrates the effect that the Calculator's **Immediate Update** option has on a variable.

Part 1: Using a Calculator to Change a Sprite's Position

1. Create a new Application Map with a Start, a Design Output, and a Numeric Calculator, in that order. Name the Design Output "sprite," and the Numeric Calculator "move sprite."
2. Use a Path to connect the Start to the Design Output, and to connect the Design Output to the Calculator.
3. Use a Path to connect the Exit of the Calculator to the Entrance of the Output; doing this will create a loop that will allow you to see how the Calculator changes the sprite's location (Figure 14.4).

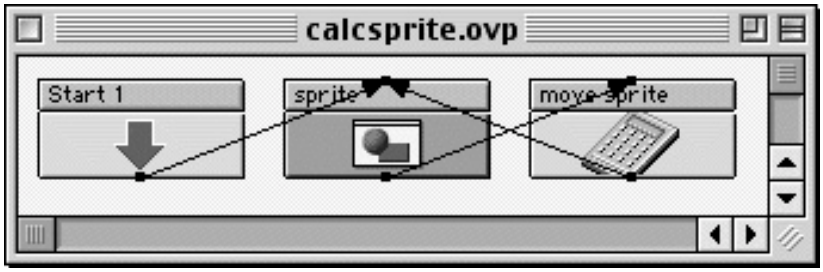


Figure 14.4: The Application Map created in steps 1-3. The States are placed in the correct order, named, and connected with Paths.

4. Open the Design Output's Presentation window and create a sprite. See chapter 6, Output States, for more information on creating sprites.
5. Open the Numeric Calculator's InfoCenter by double-clicking anywhere on its icon, or by selecting it and choosing **Calculator InfoCenter** (⌘ I) from the **Map** menu.

   When the Calculator InfoCenter opens, the first equation field is highlighted. For this example, we need to create two equations: one that will move the sprite along its X axis, and one that will move it along its Y axis.
6. To begin the first equation, fill in the left-hand expression by clicking the box labeled **?** (question mark).

   The Variable Selection dialog is opened, with all of the categories of variables represented by tabs.
7. Click the Output variable category tab to bring it to the foreground. Choose the Output State named "sprite" from the list on the left. Choose the variable **1st Sprite's X Position** from the list on the right, and then click the **Select** button.

   Clicking the **Select** button returns the chosen variable to the Numeric Calculator InfoCenter on the left-hand side of the equal sign (Figure 14.5). You now need to fill in the right-hand expression of the equation.

Figure 14.5: After choosing a variable, the equation's left-hand expression is filled in.

8. Click the **Output** variable category button and, when the Select a Variable dialog opens, once again choose the Output State named "sprite" from the list on the left, the variable **1st Sprite's X Position** from the list on the right, and then click the **Select** button. To complete this expression, click the + (plus) button, then click the **Constant** button, type twenty, and press the Return key.

   The completed equation reads **Sprite1X{sprite} = Sprite1X{sprite}+20 (Figure 14.6)**.

Figure 14.6: Both expressions of a completed equation.

   This equation will look at the sprite's current X position, and then move it twenty pixels along its X axis.

9. Repeat steps 6–8 for the **1st Sprite's Y Position** variable, to move the sprite twenty pixels along its Y axis.

10. Click **OK** to close the Numeric Calculator InfoCenter and return to the Application Map.

Test your application by choosing **Run** (⌘ R) from the **Edit** menu. Your sprite will move twenty pixels down and to the right each time your application cycles through the Calculator. You can stop the test run by choosing **Stop Run** (⌘ .) from the **Stop** menu.

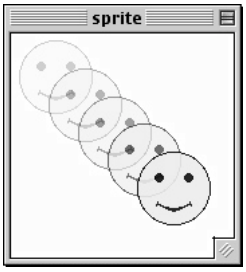


Figure 14.8: During runtime, the sprite moves twenty pixels to the right (its X axis), and twenty pixels down (its Y axis).

Part 2: Using the Calculator's Update Now Feature

The best way to observe how the Calculator's **Update Now** feature works is to observe some equations with this option turned on, and then to watch the same equations with it turned off. For this example, we'll use the application map we created in part 1, but first we'll need to add three more equations to it. The third equation will simply be a delay, which will allow us to see how the **Update Now** feature affects the variables. The last two equations will move the sprite back into its original position.

1. Open the InfoCenter for the Numeric Calculator you created in part 1.
2. Click in the third equation field, then fill in the left-hand expression by clicking the box labeled **?** (question mark).

   The Select a Variable dialog is opened, with all of the categories of variables represented by tabs.
3. Click the General variable category tab to bring it to the foreground. Choose the variable **Delay in Seconds** from the list, and then click the **Select** button.

   Clicking the **Select** button returns the chosen variable to the Numeric Calculator InfoCenter on the left-hand side of the equal sign. You now need to fill in the right-hand expression of the equation with the number of seconds you want the variables to be delayed.
4. Click in the right-hand expression field, type "3," and press the Return key. The full equation now reads **DelayInSec{} = 3.0** (Figure 14.9). This equation makes the Calculator pause for three seconds before moving to the next equation.

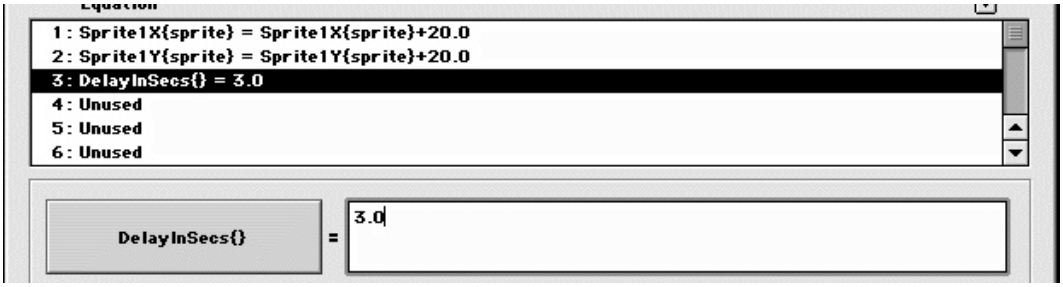


Figure 14.9: The completed equation from step 4.

Since the last two equations will simply move the sprite back to its original position, we can copy the first two equations, paste them, and change the + (plus) to a – (minus).

5. Click on equation number one to highlight it, and click the **Copy** (⌘ ⇧ C) button. Then click in equation field number four and click the **Paste** (⌘ ⇧ P) button. Change the equation's + (plus) to a – (minus) by highlighting it and clicking the – (minus) button.

Tip

A numerical constant can also be entered by first clicking in the right-hand equation field and then simply typing numbers. When you type a number, the **Constant** field automatically appears with the number already entered into it. After typing the full number, press the Return key or click anywhere outside the **Constant** field to return the value to the equation in the InfoCenter window.

6. Repeat step 5 for the sprite's Y variable (Figure 14.10).

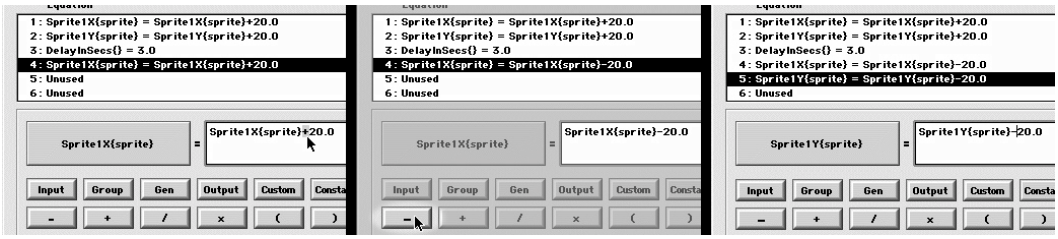


Figure 14.10: To change the operator in an equation's right-hand expression, highlight the plus sign (left) and click the minus sign button (center); repeat for equation #5 (right).

7. The **Update Now** option needs to be selected for all of the equations; to do this, select each equation (including #3), and then select the **Update Now** check box (Figure 14.11).

   With this option selected, the sprite will move as soon as each equation is calculated.

*Figure 14.11: For step 7, the **Immediate Update** option needs to be selected for all five equations.*

8. Click **OK** to close the Calculator InfoCenter and return to the Application Map.

   Test your application by choosing **Run** (⌘ R) from the **Edit** menu. Your sprite will move twenty pixels down and to the right, pause for three seconds, and then move back to its original location each time your application cycles through the Calculator (Figure 14.12). You can stop the test run by choosing **Stop Run** (⌘ .) from the **Stop** menu.

9. Reopen the Calculator InfoCenter, and this time deselect the **Immediate Update** option for each equation.

10. Click **OK** to close the Calculator InfoCenter and return to the Application Map.

    Test your application by choosing **Run** (⌘ R) from the **Edit** menu. This time, you will not see the sprite moving at all; this is because the Calculator, without the **Immediate Update** option selected, moves through all the equations and then applies them at the same time, when the application flow exits the Calculator. As a result, you can only see the sprite's position as it is left in the last equation, which, in this example, is right where it started.

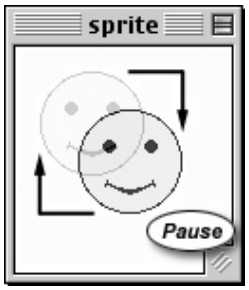
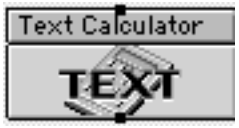


Figure 14.12: A sprite during runtime, with the Immediate Update option applied to all of the Calculator's equations; the sprite will move twenty pixels to the right, twenty pixels down, pause for three seconds, then return to its original position. Without the Immediate Update option selected, this movement would not be seen.

Text Calculator



The Text Calculator is used to modify and/or assign text strings to text variables. These variables can be either onViz' built-in variables or a Custom Text variable. The Text Calculator is more specialized than the Numeric Calculator, and therefore only supports writing to Input, General, and Custom variables. Some applications include:

- Assigning a user's answer to a Text Input to a Custom Text variable to make this information available in supporting documents.
- Using the **AND** function to combine multiple text strings.
- Dynamically changing on-screen text display and/or radio button/checkbox answers during runtime of the application.

The Text Calculator InfoCenter

As with the Numeric Calculator InfoCenter described above, the Text Calculator InfoCenter is divided into three sections: an equation list, a section for creating the equation, and a section for naming and managing the Calculator State (Figure 14.13).

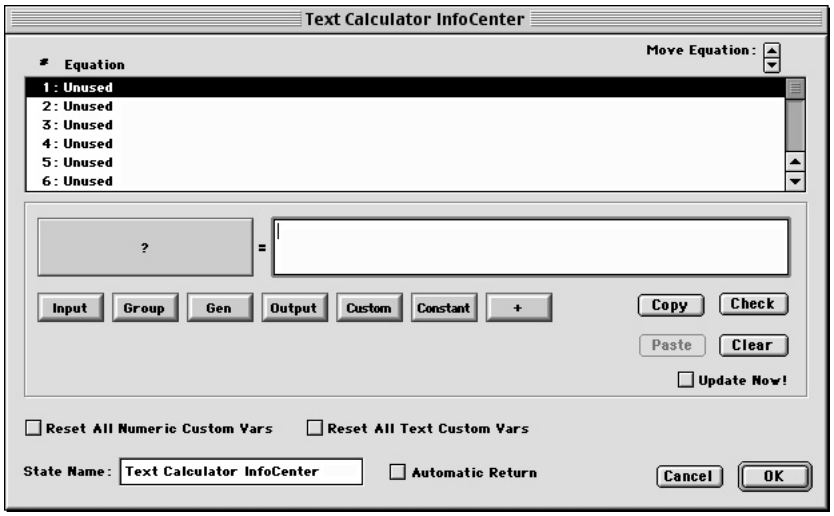


Figure 14.13: The Text Calculator InfoCenter.

Since many of the Text Calculator Info Center features are similar to the Numeric Calculator, this section will cover only the differences. If you haven't already, we suggest you read the Numeric Calculator InfoCenter section before continuing.

When choosing a variable for the left-hand side of the equation in a Text Calculator, the Variable Selection dialog displays only those variables to which text strings can be assigned: Input Answer Entered, several General variables, and any Custom Text Variables. After a variable is selected, its name is displayed in the box formerly labeled with a question mark.

The most noticeable difference in the Text Calculator InfoCenter is that there are no numeric function buttons for filling in the right side of the calculation. The right-hand side of the equation is completed by clicking:

- One of the variable buttons.
- The **Constant** button.
- The + (Plus) button.

The plus button allows you to combine text strings. For example, if you used two Text Input States to get the user's first and last name, you can use this button to equate the user's full name to a Custom Text variable.

Writing to a Variable using a Text Calculator

This example uses two Inputs to gather a user's first and last name, then uses a Text Calculator to equate these Inputs with a custom variable. The custom variable is then used to display the full name in an Output.

1. Create a new Application Map with a Start, two Enter Text Inputs, a Text Calculator, a Design Output, and a Stop. Name the Inputs "first name" and "last name," respectively. Name the Text Calculator "calc name," and name the Output "display name." Connect the States in order, as shown in Figure 14.14.

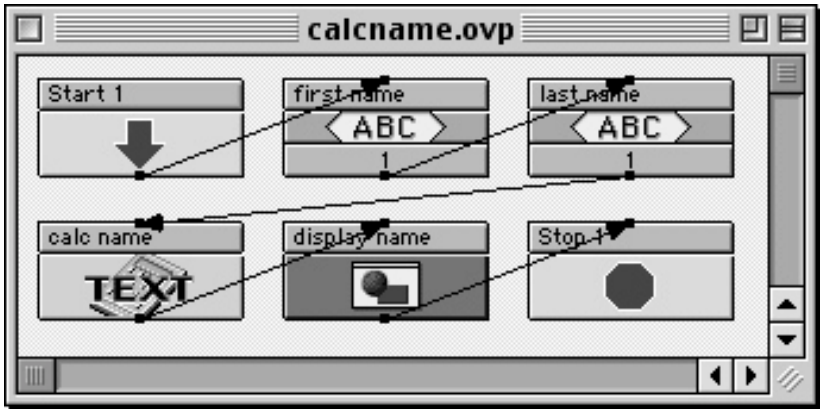


Figure 14.14: The Application Map created in step 1. The States are placed in the correct order, named, and connected with Paths.

2. Create a new custom text variable: choose **Custom Variables Library** (⌘ 3) from the **Library** menu. When the Custom Variables Library is opened, choose **Text Only** from the **Show:** dropdown menu, click the **New Text** button, and name the new variable "full name." Leave the **Initial Value** field blank. Type in a **Description** for your Custom Variable, such as, "Joins first and last names as entered into Inputs" (Figure 14.15). Click **Close** to return to the Application Map. For more information on creating custom variables, see chapter 13, Variables.

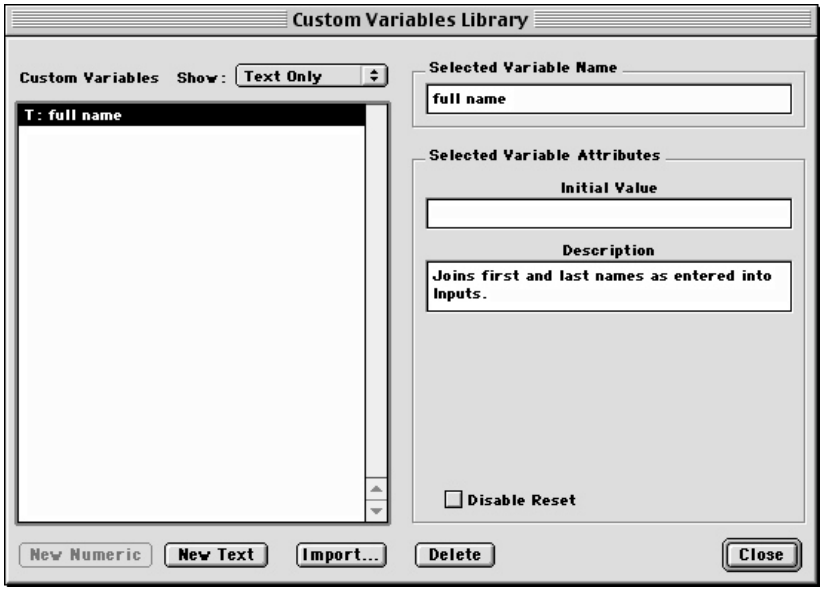


*Figure 14.15: The Custom Variables Library, displaying the new custom text variable ("full name"). Note that, for this variable, the **Initial Value** field can be left blank, but the **Description** field details what the variable does.*

3. Open the Text Calculator's InfoCenter by double-clicking anywhere on its icon, or by selecting it and choosing **Calculator InfoCenter** (⌘ I) from the **Map** menu.

   When the Calculator InfoCenter opens, the first equation field is highlighted.

4. To begin the equation, fill in the left-hand expression by clicking the box labeled **?** (question mark).

   The Select a Variable dialog is opened, with all of the categories of variables represented by tabs.

5. Click the Custom Variable tab to bring it to the foreground. Choose the "full name" custom variable from the list on the left, then click the **Select** button

   Clicking the **Select** button returns the chosen variable to the Text Calculator InfoCenter on the left-hand side of the equal sign. You now need to fill in the right-hand expression of the equation.

6. Click the **Input** button to fill in the right-hand side of the equation with an Input variable.

   Clicking any of the Variable category buttons once again opens the Select a Variable window, with the chosen category in the foreground.

7. Choose the "first name" Input from the list on the left, choose **Answer Entered (Text)** variable from the list on the right, and then click the **Select** button

   Clicking the **Select** button returns the chosen variable to the Text Calculator InfoCenter's right-hand expression, where it completes the equation.

8. Click the + (plus) button.

9. Repeat steps 6 and 7; this time, however, select the "last name" Input.

   Your Text Calculator should resemble Figure 14.16. Click **OK** to close the Text Calculator InfoCenter and return to the Application Map.

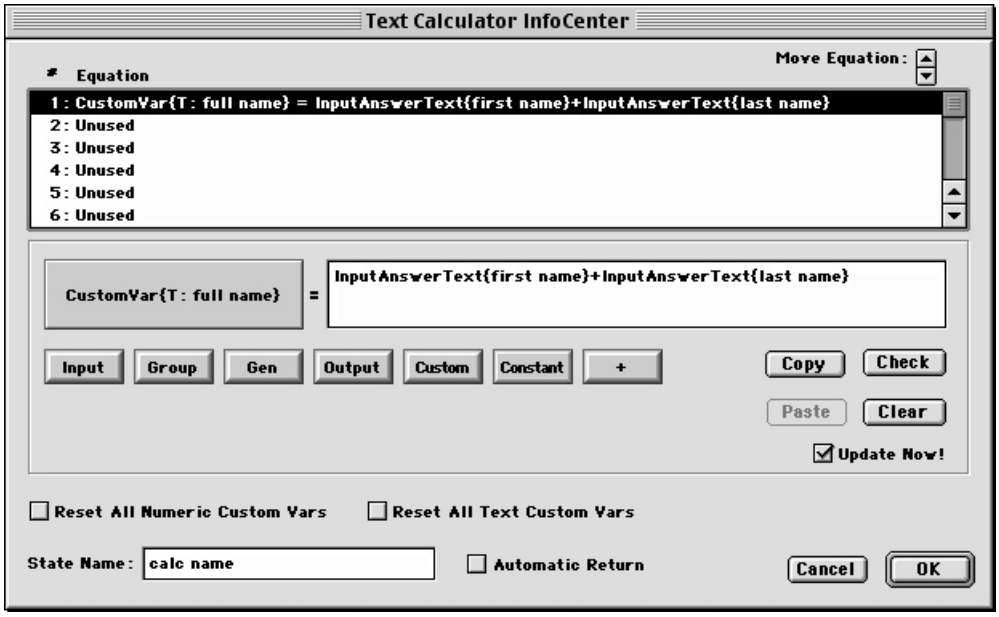


Figure 14.16: A Text Calculator equating the new custom text variable with two Input variables.

10. Open the Output presentation window, and create a sprite that will contain the custom variable. The sprite will use variable substitution to display the user's full name in place of the custom variable. The sprite should read something like "Your name is:" and then have the custom variable below it. For more information on using variable substitution in a sprite, see chapter 13, Variables.

11. From the Frame Control palette's **Synchronization** dropdown menu, choose **Continue button**.

    Adding a Continue button will allow the Output to remain on screen long enough for you to verify that the variable substitution is working. Your design window should resemble Figure 14.17.

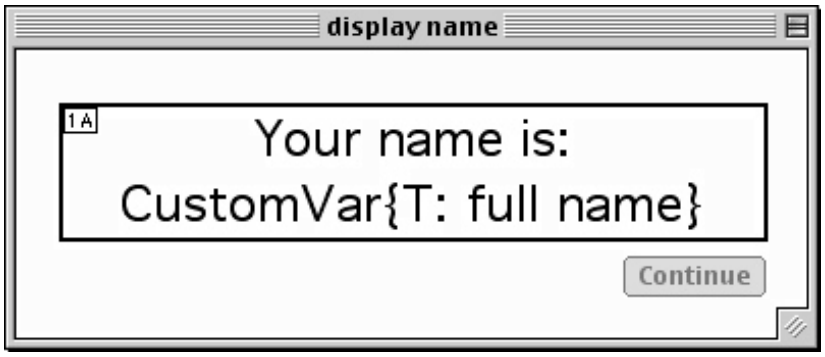


Figure 14.17: The Design Window after adding the sprite and the Continue button.

12. Close the Output Presentation window and return to the Application Map by choosing **Close Window** (⌘ W) from the **File** menu.

    Test your application by choosing **Run** (⌘ R) from the **Edit** menu. The Inputs will ask you for your first and last names, which the Calculator will combine into your custom variable. When the Output is run, the custom variable will display your first and last

names together (Figure 14.18). You can stop the test run by choosing **Stop Run** (⌘ .) from the **Stop** menu.

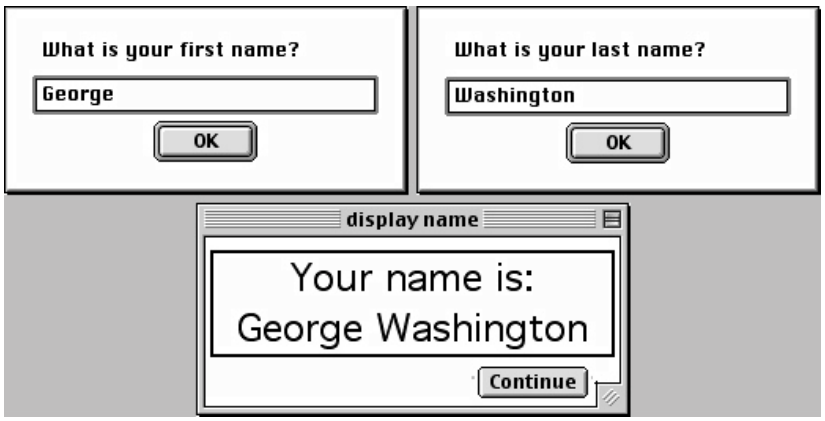


Figure 14.18: During runtime, the application will ask for your first name (top-left), and your last name (top-right). The Calculator equates your responses to these questions with your new custom variable, which is then displayed in an Output (bottom).

Chapter 15

Paths

Covered in this Chapter:

- Using Paths
- Types of Paths
- Conditional Paths

Using Paths

Paths, represented by arrows, determine the sequence in which an application's activities are performed. If your Map is set up with Freeform Branching strategy, your application always begins with a Start State. After exiting the Start State, onViz finds the Path connected to its Exit Point and follows this Path to determine which state to run next. If there is no path from the Start State, onViz has no way to determine where to go next and will display a "Dead End Route Reached" error message when the application is run. Therefore, Paths play a critical part in your application development. For more information on branching strategies, see chapter 2, the Customizing the Development Environment.

Paths are created by clicking and dragging from a State's Exit Point to the Entrance Point of another State. While clicking and dragging from an Exit Point, you see a line being created with an arrowhead on its lead end, indicating that a Path is being created. To connect the Path to another State, extend it until it is anywhere over the State you want to run next, and then release the mouse button. The Path will be automatically connected to the State's Entrance Point (Figure 15.1).

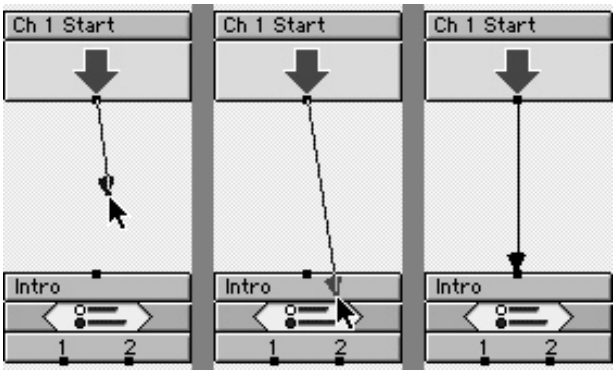


Figure 15.1: To connect a Path to a State, extend it until it is anywhere over the State you want to run next, and then release the mouse button. The Path will be automatically connected to the State's Entrance Point

Paths are only functional if they are properly connected at both ends (beginning at a State Exit and ending at a State Entrance). Improperly connected Paths are shown as blue arrows. If the project's flow of activity tries to follow an improperly connected Path, onViz displays a "Dead End Route Reached" error message when the application is run. You can choose **Show Me** to display the point the application reached a dead end.

Paths can be used to connect the Exit of any State with the Entrance of any other State at the same map level; for instance, two states within a Group can be connected with Paths. If you need to connect to a State outside the current Group, use a Bridge State. For more information on Bridge States, see chapter 5, Flow States.

A path can be selected by clicking on its line with a single mouse click; the end points of selected Paths are designated with drag handles (Figure 15.2). Additional Paths may be selected by holding down the shift key while clicking on the Paths' line. Multiple Paths can be selected by using the Pointer tool to drag a selection marquee around the desired Paths.

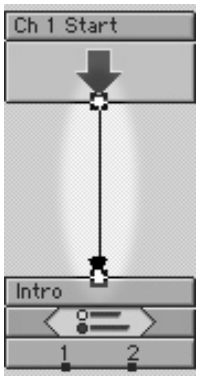


Figure 15.2: A selected Path. Note the drag handles at each of the Path's end points.

Note that when using the Pointer to select Paths with a selection marquee, the entire Path must be enclosed within the marquee. To select each Path element the marquee touches, hold down the Option key while dragging the marquee.

Types of Paths

There are two types of Paths, Unconditional and Conditional. Unconditional Paths simply direct the flow of your project from one State to the next. Conditional Paths allow you to set up situations to determine the flow of your application between States. Among the many situations that can be checked in a Condition Path are: checking the runtime machine's operating system, checking the current score or other user progress statistics, verifying the user has connection to the World Wide Web, or checking the position of a sprite.

It is possible for several Paths to leave a State by the same Exit Point. If all of these Paths are Unconditional, onViz selects one at random for its exit. This type of routing is fine if you want the project to flow in a random fashion. However, if you want to direct the project's flow of activity based on a series of events or variables, use Conditional Paths. Conditional Paths give your application intelligence and can make your application a truly interactive experience based on your users' actions, rather than a slide show presentation of information.

Conditional Paths

Typically, when using Conditional Paths, you have two or more Paths leaving the same Exit Point: one non-conditional and one or more Conditional Paths. Application flow will not proceed along a Conditional Path from one State to the next unless that Path's conditions are satisfied. For more control, you might use a single Unconditional Path and several Conditional Paths leaving from the same Exit Point. If connected in this manner, onViz begins evaluating the Conditional Paths and takes the first one whose conditions have been satisfied. If no conditions are satisfied, the application flow follows the Unconditional Path (Figure 15.3).

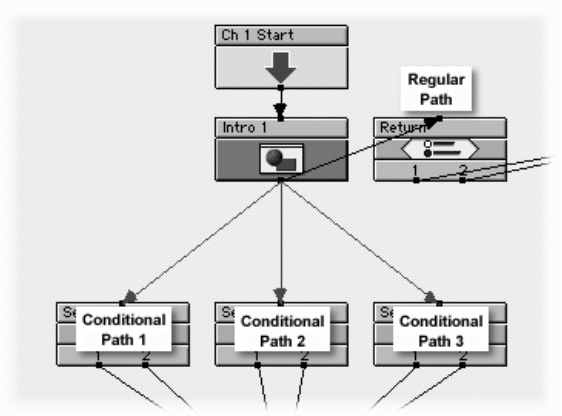


Figure 15.3: Multiple Paths leaving the same Exit Point. onViz evaluates the first Conditional Path, and if its conditions are not satisfied, moves on to the next until it finds one whose conditions are satisfied. If no conditions are satisfied, onViz takes the non-conditional (regular) Path.

Conditional Paths function by ***reading*** a variable's value. If its value satisfies the conditions set for the Path, then the Path will be followed. A condition is essentially an equation comparing one variable to either another variable or to a constant. If the comparison is true, then the condition is met and the Path is followed. If the comparison is false, then the Path's conditions are not met, and the Path will not be followed. For more information on variables, see chapter 13.

Path InfoCenter

The Path InfoCenter is used to create criteria for Conditional Paths. When initially created, all Paths are Unconditional, and are shown as a solid black arrows. To create a Conditional Path, double-click any point of a Path to bring up its Conditional Path Info dialog (Figure 15.4), fill in a Condition line, and click **OK** to close the window and return to the Application Map. The new Conditional Path is shown as a solid red arrow, a visual cue that the Path contains criteria that must be met.

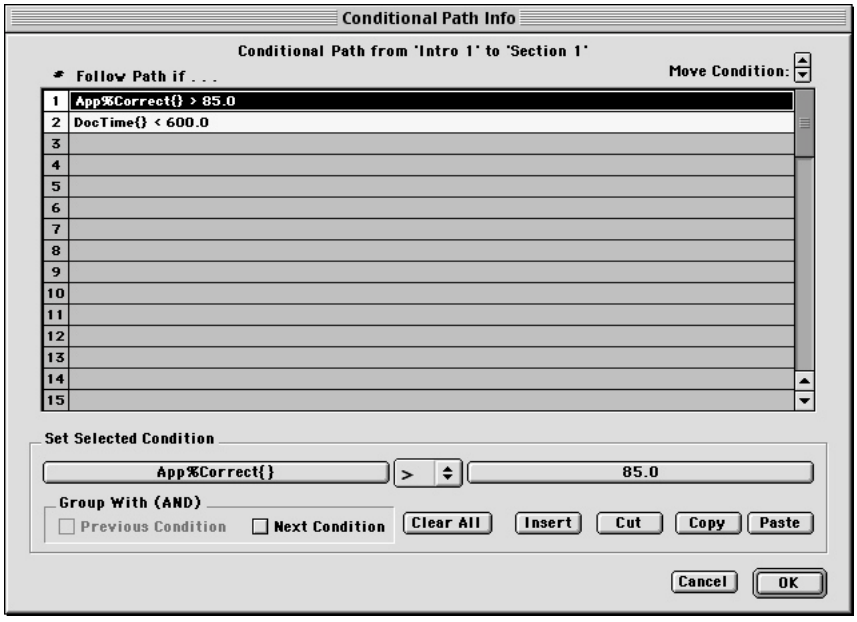


Figure 15.4: The Conditional Path InfoCenter

Each Conditional Path can be assigned up to 64 conditions, which are created using the **Set Selected Condition** fields. The **Set Selected Condition** fields consist of two variable buttons, initially labeled with a **?**, and a Comparison Type dropdown menu that contains operators such as =, <, and >.

Creating a Condition

1. Double-click the Path to open its InfoCenter dialog.
2. Click the left-hand variable button.

   The Variable Selection dialog is opened.
3. Choose the variable (or constant) on whose value you want to base the condition (for this example, choose the General variable **Application's Percent Score**), and the button label is updated with the selected variable (**App%Correct{}**). Note also that the selected condition line contains the variable to the left of the equal sign.
4. Choose an operator from the Comparison Type dropdown menu. For this example, choose > (greater than).

   The operator will be used to compare the two variables (or the variable and constant). Both the Comparison Type field and the condition line are updated with the selected operator.

5. Click the right-hand variable button and select the variable or constant that will be compared to the left side of the equation. To complete this example, select the Constant tab and type "85" for the value of this constant.

   Again, note that both the right-hand button label and the condition line are updated with the constant you entered. The condition line now reads **App%Correct{} > 85.0**, or, in other words, "If the Application's Percent Score is greater than 85," then follow this path (Figure 15.5).

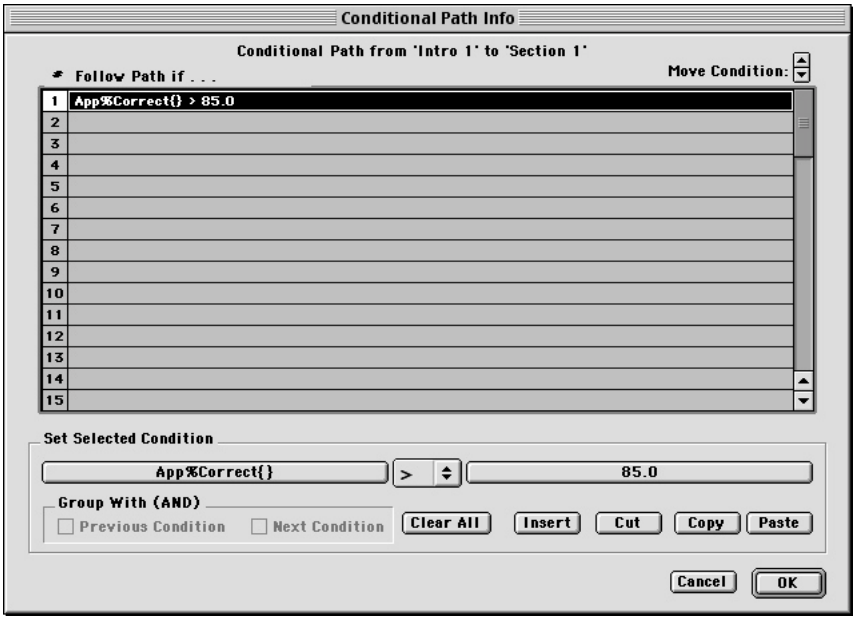


Figure 15.5: The Conditional Path InfoCenter with a completed condition. Note that the condition's components are reflected in the condition line and in the Set Selected Condition field.

6. Repeat this procedure for every condition you want to create.

To quickly create conditions using similar variables, use the **Cut**, **Copy**, and **Paste** buttons. Select a condition and click **Cut, or hold down the Shift and Command key and press X** (⌘⇧V) to remove it and place it on the clipboard, or click **Copy, or hold down the Shift and Command key and press C** (⌘⇧C) to copy the condition to the clipboard without deleting it, then select an empty Condition line and click **Paste**. After the new condition is pasted, use the **Set Selected Condition** fields to edit it as needed.

Creating Multiple Conditions in the Same Path

Each Path can contain up to 64 unique conditions. When a Path contains multiple conditions, they can be evaluated separately (logical OR function) or in groups (logical AND function).

If evaluated separately, conditions are examined in the order they appear on the list. As soon as a condition is found to be true, the Conditional Path will be taken. If no conditions are true, the Conditional Path will not be taken.

If you add a condition to the example above, onViz will examine them in order and take the path if *either* condition is true. For instance, you can set up a Conditional Path that will be followed if the user has achieved a base percent correct (85%), OR completed a segment within a specified period of time; to do this, simply set up two conditions on separate lines. To set up this second condition, open the Conditional Path just created and click on condition line 2 to select it.

1. Double-click the Path to open its InfoCenter dialog.
2. Click the left-hand variable button and select the General variable **Document's Time So Far**.

   The button label and the condition line are updated to read **DocTime{}**.
3. Choose the < (less than) operator from the dropdown menu.
4. Click the right-hand variable button, select the Constant tab, and enter "600." Remember that time in onViz is designated in seconds.

   Your condition line now reads Doc**Time{} < 600.0**. Because these two conditions are being evaluated separately, the Conditional Path will be followed if "the Application's Percent Score is greater than 85 *OR* the time spent in the application is less than ten minutes (600 seconds)" (Figure 15.6).

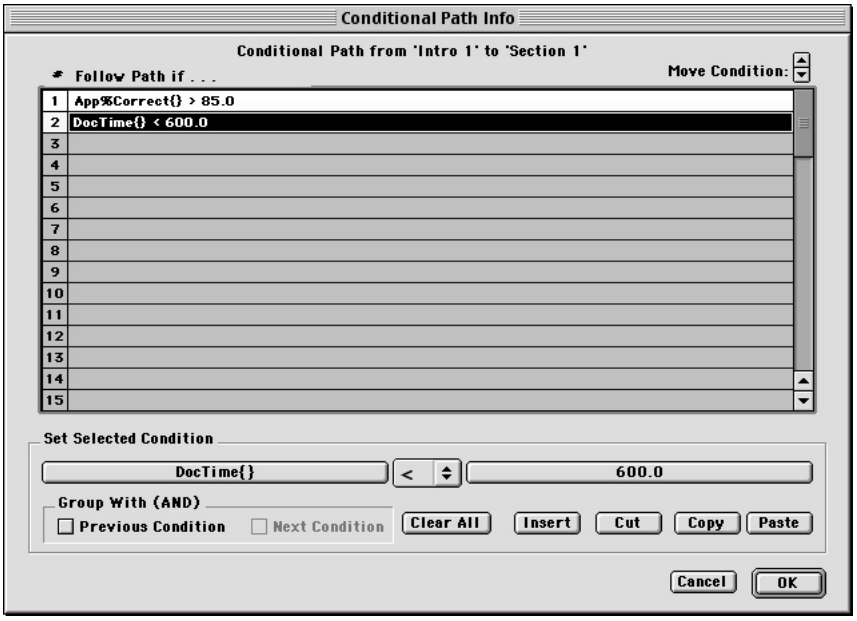


*Figure 15.6: The Conditional Path InfoCenter with two separate conditions. The Conditional Path will be followed if **either** of the two conditions are satisfied.*

If multiple conditions are evaluated as a group, they must all be true in order for the Conditional Path to be taken. Conditions are grouped by selecting a condition and choosing **Group with Previous Condition** or **Group with Next Condition**.

To update your condition so the Percent score must be greater than 85 percent *AND* the time is less than 10 minutes:

1. Click on condition line 1 and, from the **Group With field**, select **Next Condition**.

   Note that the separator line between the two conditions has disappeared; this is a visual cue that these two conditions are linked and that both must be true in order for the condition to be met (Figure 15.8).

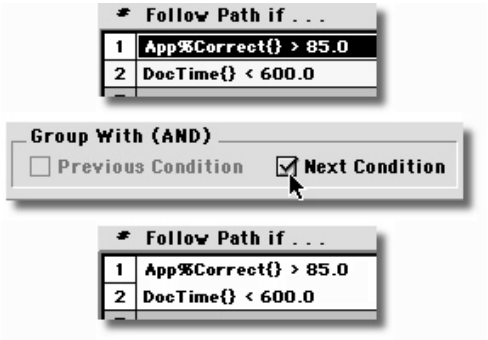


Figure 15.7: Grouping two conditions. Selecting the condition in line one (top), and choosing Group With Next Condition (middle). Note that the separator line between the two grouped conditions has disappeared (bottom).

You could have also clicked on condition line 2 and selected **Group With Previous Condition** to group these conditions.

Your Conditional Path now reads "If the Application's Percent Score is greater than 85 *AND* the time spent in the application is less than ten minutes (600 seconds)," then follow this Path.

To ungroup conditions, click the condition line that you do not want in the group and either deselect the **Group With** option or click **Cut** (⌘⇧X) to remove the condition from the list.

The order in which conditions are evaluated can be changed by selecting a condition and clicking the **Move Condition** arrows to move it up or down in the list. Grouped conditions will be moved up and down in the list together. There are several reasons you might want to change a condition's order in the list. Because onViz evaluates the first condition and then continues down the list until one is found to be true, the conditions most likely to be true should be placed at the top of the list. This will give better runtime performance, as onViz will not have to evaluate every condition before making a branching decision.

You may also want to change a condition's order in the list in order to group it with another condition. Because you can only group a condition with the previous or next conditions, conditions that are not adjacent to each other cannot be grouped. The **Move Condition** arrows can be used to move one condition adjacent to another so that the **Group with Previous Condition** or **Group with Next Condition** options can be used.

Inserting Conditions

You can "Insert" a condition line anywhere in the list. To insert a condition line between two ungrouped conditions, click on the condition line below the point you want to insert a condition and select **Insert**, which creates a blank line between the two conditions (Figure 15.8).

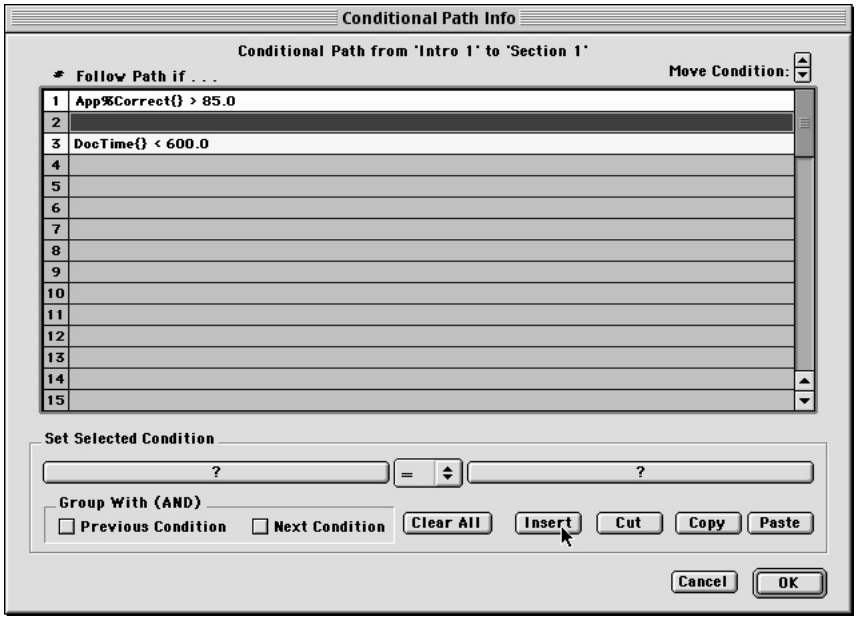


Figure 15.8: A condition line inserted between two ungrouped conditions.

To insert a condition line in a "group," click on any condition in the group, other than its first condition, and click **Insert**, which creates a blank line within the condition group. Inserting a line while a group's first condition is selected will place the new condition line above and outside the group.

Chapter 16

Glossary

Application Flow – The order in which **States** in an **Application Map** are run. By default, the order in which States are run is controlled by **Paths** and **branching strategies**, although it can also be determined by **Flow States**. Application flow is occasionally referred to as flow of activity.

Authoring – The window in which you create your project. This window includes the **Map Tools palette** and a **work area**, where your **States** and **Paths** are placed.

Background Image – In an **Output Presentation**, the background image is a static image underlying a **frame** of animation and all the **sprites** on the frame, and is chosen with the **Frame Control palette**. A background image can also be designated for the **Image Editor work area**, using the **Image Control options** at the bottom of the window. Placing a background image in the Image Editor is helpful if you are creating sprites that need to be precisely placed over a **Design Output** background, or if you want to compare your new image to an image already in the Image **Library**. The pre-existing image is loaded as a dimmed background in order for you to compare the two images while creating the new one.

Background Output – The **Output State** chosen to underlie an **Input** window during **runtime**; designated by clicking the Position or On Output buttons in an Input **InfoCenter**, or by opening an Input's **Presentation** window.

Bitmap Image – Images that are comprised of tiny square dots, or pixels. Each individual pixel is "painted" a certain color so that, collectively, they form a picture. Because the bitmap image stores this color information for each of its pixels, its file size is typically larger than object images. Occasionally referred to as pixel-based images, they are created with the **Drawing Tools palette**'s Pixel tools.

You alter a bitmap image's color by "painting" over its pixels with another color. Scaling, or changing a bitmap image's size in onViz' Image Editor often results in a "stair-step effect," as the image's colors are either stretched out over a larger number of pixels, or compressed into a smaller number of pixels.

Bookmark File – Created by the user when **application flow** passes through a **Bookmark Gadget**, bookmark files allow the user to open an onViz application at a previously saved location. When a user saves a bookmark file, it is, by default, saved to the same directory level as the user's onViz application. This location can be changed, during **development**, through the **Build Target** dialog, found under the File menu. By checking the "Allow End-User to Reconfigure" option, the developer can give an end-user the option of altering a text file that determines the location in which the bookmark file is saved. Bookmark files can be opened by double-clicking them, or by dragging them over an onViz application.

Bookmark Gadget – This **Gadget State**, when **application flow** is routed through it, allows the user to save a **bookmark file**, or to open a previously saved bookmark file. Bookmark files allow the user to save their place in an application.

Branching Strategy – The default route that **application flow** will take, in the absence of **Paths** or **Flow States**, through an **Application Map** or **Group Map**. Types of branching strategies include **freeform**, **linear**, **one per row**, **random pool**, and **list access**. For an Appli-

cation Map, branching strategy is designated through the Project Attributes dialog, found under the Edit menu. For a Group Map, branching strategy is determined by choosing a specific type of **Group State**.

Bridge – The act of jumping from one location in your onViz document to either another location, to a **supporting document**, or to another program. Bridges are created by using a **Bridge State** or a **Smart Sprite**.

Bridge State – When application flow enters this **Flow State**, it either jumps the flow to a different location in your project, to a **supporting document**, or to another program. The location to which the **bridge** jumps is called a **Bridge Target**.

Bridge Target – The **State**, onViz **supporting document**, application (e.g. Word or Netscape), or application document (e.g. Word document or HTML file) to which a **bridge** is jumping. A bridge target is set in the Bridge **Library**'s Attributes dialog.

Build – The act of compiling your project's source files into a stand-alone application, or **build target**, that can be distributed to your users. When building a target, onViz operates on a copy of your source files, leaving the original files untouched. In this manner, the same set of source files can be used to create multiple build targets. Also, if some part of your source files' content needs to be revised, you can change the original files and then build all new targets.

Build Target – The configuration of a stand-alone document, created from your project's source files using the Build Target dialog. Build Targets can be single or multiple-file and for Macintosh or Windows.

Calculator State – Typically refers to the Numeric Calculator State, which assigns values to numeric **variables**. Occasionally used to refer collectively to the Numeric Calculator State and the **Text Calculator State**.

Check Boxes Input – Resembling a multiple-choice question, this **Input State** lets your user click one or more answers at a time from the list of up to ten possible answers. With this flexibility, you can require your user to choose all designated correct responses in order to answer the question correctly, or just one correct response out of several. This type of Input State always has four **Exit Points**, regardless of the number of answers it contains. The four Exits correspond to a Correct Selection, an Incorrect Selection, a "Time Out" path, and a limit on the number of tries.

Color, Pattern, Pen palette – Determines an image's attributes when the **drawing tools** are used. Settings include: Fill Pattern, Fill Foreground Color, Fill Background Color, Pen Width, Pen Style, Pen Color, and Text Color.

Conditional Path – A type of **Path** that uses formulas and **variables** to create conditions that determine your project's **application flow**. Conditions are set up in the Path **InfoCenter**. Conditional Paths are colored red on an **Application Map** or **Group Map**.

Cursor Display Gadget – This **Gadget State**, when **application flow** is routed through it, allows you to specify an alternative cursor for your application. You can choose from sixteen custom cursors, or create one of your own.

Custom Variable – A variable created by developers and used for maintaining their own values. These values can be stored for the duration of the application, or passed between the top-level application and any supporting documents. There are two types of custom variables, numeric and text. Custom numeric variables hold numeric values, and can display them in a variety of formats. Custom text variables hold text strings.

Design and Text Visible Output – Combines the attributes of the **Design Output** with those of the **Text Output**. This **Output** type consists of a two-window display: the **Design Window** and the **Text Window**. The Design Window contains graphics, and controls media components such as sound, movies, and text. The Text Window is a scrolling window that contains text only. This type of Output lets your application use a consistent interface and display of **background** graphics, while taking advantage of features like scrollable text or customized reports, which get saved to the application **report file**. Often referred to as simply a Design and Text Output.

Design Window – The **Output** window in which static and animated images are displayed. To access a Design Window during **development**, you must be working with a **Design Window Visible Output** or a **Design and Text Visible Output**.

Design Window Visible Output – An **Output** that uses the **Design Window** to display images stored in the Image **Library**, and to play sounds and digital movies from the Sound and Movie Libraries. Often referred to as simply a Design Output.

Development – The process of creating your onViz application, including its **Application Map** and media assets. Also referred to as authoring.

Draggable Sprites – A **sprite** that has been set to be draggable in the **Mouse Actions** section of the **Sprite Attributes palette**. Draggable sprites are used with **Clicking On Objects Inputs** to allow users to move sprites into **Mouse Bays**, thereby affecting **application flow**.

Drawing Tools Palette – Determines the shapes for your images, and is divided into two sets of tools: **Object** (or **vector-based**) **tools** and **Pixel** (or **bitmap**) **tools**. The **Pointer**, Eye Dropper, and **Zoom** tools are used with both sets of tools.

Entering Numbers Input – An **Input State** that allows your audience to type their own numerical response, and allows you to create up to ten conditions in order to evaluate, or **parse**, the response for accuracy. The Entering Numbers Input accepts only numbers, decimal points, commas, and currency units.

Entering Text Strings Input – An **Input State** that allows your audience to type their own response, rather than choosing one that you have presented. Their response is then evaluated, or **parsed**, against a series of conditions you set in the Input **InfoCenter**. You can use up to ten sets of conditions to parse the response for its accuracy. The response is checked for accuracy against the first condition, then the second, then the third, and continues until it

either meets a condition or runs out of conditions and is judged as incorrect. The conditions include criteria such as Exact Match, Has String, and Has Word, with the added ability to combine criteria using "and" and "or." The Entering Text Strings Input accepts only text, and cannot parse numbers.

Entrance Point – The destination point for incoming **Paths**, the Entrance is where the project's **application flow** enters a **State**.

Exit Point – The point of departure for outgoing **Paths**. The project's **application flow** leaves a **State**'s Exit and continues to the **Entrance** of the next State, as determined by either their Path connections or **branching strategy**.

File Maintenance Gadget State – Provides settings to download individual files or folders of files from a different volume, CD-ROM, or remote server. Files that are downloaded are put in the applications 'Downloads' folder and will be used during application runtime.

Flow State – Works together with **branching strategies**, **Paths**, and **Group States** in determining **application flow**. Types of Flow States include **Start States**, **Bridge States**, **Return States**, **Menu States**, **Junction States**, and **Stop States**.

Foreground Image – In an **Output Presentation**, the foreground image is the static image floating above the **background**, but underneath all the **sprites**, in a **frame** of animation, and is chosen with the **Frame Control palette**. A foreground image can also be designated for the **Image Editor work area**, using the **Image Control options** at the bottom of the window. Placing a foreground image in the Image Editor is helpful if you are creating sprites that need to be precisely placed over a **Design Output** foreground. Since an Output can have both a background and foreground image, this feature allows you to simulate the exact Output for which you are designing sprites.

Frame (Animation) – One of possibly several sequentially numbered collections of graphics and/or **Text Windows**. Joined together, they form an effect similar to that of a motion picture reel. Frames are added, deleted, and manipulated with the **Frame Control palette**.

Frame (Window) – The type of border that encloses a window. Different types of frames can be chosen for several different windows in onViz, including: an **Output**'s **Design Window**, **Text Window**, and movie window, and an **Input**'s **Overlay Image** window and movie window. Types of frames include: document, dialog, plain rectangle, 3-D, rounded corner, full screen, and borderless.

Frame Control Palette – Found within an **Output**'s **Presentation**, the Frame Control palette contains tools for adding, deleting, and manipulating **frames** and multimedia elements within an animation.

Freeform Group – A type of **Group State** in which the routing of your project's **application flow** is determined by the way in which its **States** are connected by **Paths**, which can branch in any way desired. Freeform groups must begin with a **Start State**, have all of their Paths properly connected, and end with a **Stop State**. Flow begins with the Start State, and continues until it reaches a Stop State, at which point the flow exits the Group State and moves to the next State in the **Application Map**.

Gadget State – Contains a variety of functions that, when **application flow** is routed through them, enhance your application; includes **Bookmark**, **Cursor Display**, **Print Options**, **Report Options**, **Send Email**, **Show Web Page**, **Monitor and Sound**, and **Timer Gadgets**.

Group Map – Accessed by opening the **Group State Presentation**, the Group Map holds all the **States** and **Paths** for any given **Group State**. The Group Map is similar to the **Application Map**, in that it includes the **Map Tools palette** and a **work area**, where your States and Paths are placed.

Group State – Provides a means to organize the **States** used in your Project, and for automating your project's **application flow**. By holding down the Control key while choosing from the **Map Tools palette**, you can access a contextual menu containing the five variations of Group States: **Freeform**, **Linear**, **One Per Row**, **Random Pool**, and **List Access**.

Hide Windows Output – An **Output** that removes from the screen all display windows until any designated synchronization has been met. If a Mat Color is specified in the Project Attributes Runtime dialog, the Hide Windows Output displays a blank screen, filled in with the selected Mat Color. If no Mat Color is selected, the desktop of the user's computer is displayed. A **Text Window** is present during **authoring**, allowing you to add text to be spoken during **runtime**, if desired.

Image Control Options – The buttons and drop-down menus found at the bottom of the **Image Editor work area**. Includes an Image **Library** drop-down menu and **foreground/background image** controls, which let you place an image below the Image Editor's work area, against which you can compare the image currently being created.

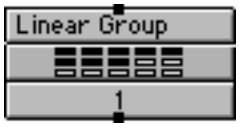
Image Editor – Used for creating **object**-based and **pixel-based** graphics. The Image Editor features a **Drawing Tools palette** and a **Color, Pattern, Pen palette**.

InfoCenter – A **State**'s InfoCenter is typically used to set window attributes and **runtime** options for that particular type of State. Because every type of State serves a different function and has different capabilities, they all have different InfoCenters. A State's InfoCenter is accessed by double-clicking the rectangular area in the upper 1/3 of the State's icon, or by selecting the State and then choosing "InfoCenter (Command I)" from the Map menu.

Input State – Provides templates for seeking input from your user. These templates include requesting input in the form of text or number answers, presenting a selection from multiple choice responses, or clicking on-screen buttons. The Input State lets you score your users' responses, as well as letting you track and capture their scores, number of attempts, and response time for use as **variables**. By holding down the Control key while choosing from the **Map Tools palette**, you can access a contextual menu containing the five variations of Input States: **Clicking On Objects**, **Entering Text Strings**, **Entering Numbers**, **Clicking on Radio Buttons**, and **Clicking On Check Boxes**.

Junction State – A type of **Flow State** used as a gathering point for multiple **Paths** in order for branching decisions to be made.

Library – One of seven collections of tools and multimedia assets that includes **Bridge Targets**, Cursors, **Custom Variables**, Fonts, Images, Movies, and Sounds. onViz' Libraries allow these tools and media assets to be shared between all of an application's **States**, and between a Project's **top-level application** and **supporting documents**. If your project contains several documents that all use the same sounds, digital movies, and images, its overall file size could be quite large if every document contained its own copy of the sound, digital movie and image files. Libraries let all of your project's documents draw their media assets from a common source, thereby reducing file size.Libraries also allow you to set up the way media assets play within your application, and to create custom cursors, define custom variables, designate display fonts, and set up **bridge** actions.



Linear Group – A type of **Group State** in which your project's **application flow** is routed through the **Group Map** sequentially from left to right and top to bottom. When determining flow sequence, the Linear Group starts with the uppermost left-hand **State**, then flows to the right. If no States are encountered to the right of the first one, it moves down to the far left of the next row and scans right until it finds another State. The Linear Group keeps scanning from left to right, top to bottom, until it has either scanned all of the Group's rows or encountered a **Stop State**, at which point the application flow exits the Group State and moves to the next State in the Application Map.

Although the Linear Group State requires no **Start States**, Stop States, or **Paths**, they may still be used. Paths can be used at any point in the application flow to override the linear flow. For instance, if a State's **Exit Point** has a Path attached, the flow of activity follows that Path rather than flowing to the right. In this way, a State may use a Path to branch off to give some feedback before continuing its flow, or the Path may branch off to a Stop State that marks the end of the Group.

List Access Group – A type of **Group State** in which the project's **application flow** is determined by presenting the user with a list of names of all the **States** contained within the Group. The user can browse this list and from it select a choice of States to run. When application flow reaches the **Exit Point** of the selected State the list is once again presented. This flow continues indefinitely until the user selects Exit, at which point the flow exits the Group State and moves to the next State in the **Application Map**. The list can have a title, which is typed into the List Access Title field in the Group State **InfoCenter**. Since the user can opt to cancel and leave the Group at any time, no **Start State**, **Stop State**, or **Paths** are necessary. However, if a Stop State is included, its name appears on the list and can be selected by the user to exit the Group.

Map Tools Palette – Contains the **States** used to create your project. Drag and drop the desired States from the **Map Tools palette** onto the **work area**. Also contains the **Pointer** and **Zoom tool**.

Mask Pixels – Used in **pixel-based**, or **bitmap**, images to make any instance of the color white transparent. The Mask Pixels command is only accessed from within the **Image Editor**, where it is found in the Image Attributes dialog, under the Graphics menu.

Media Assets – All of the multimedia files used by your project. During development, these files are stored in the project's **support folder**.

Menu State – When **application flow** enters this **Flow State**, it creates a custom menu that appears in your project's menu bar.

Monitor and Sound Gadget – This **Gadget State**, when **application flow** is routed through it, lets you automatically change the color depth, resolution, and volume level on your user's computer. When used with a **Conditional Path** at the beginning of an application, you can check to see if the user's Monitor and Sound levels are at the appropriate settings, and route flow through this Gadget if they are not. The Monitor and Sound Gadget can also be used to, at any point, change an application's mat color.

Mouse Actions – Behaviors assigned to a **sprite** using settings found on the right-hand side of the **Sprite Attributes palette**, and that are affected when the cursor passes over or clicks on them. Mouse actions can cause a sprite to: change appearance, play a sound, display a different cursor, go to a specific **frame** of animation, or route **application flow**. Sprites with mouse actions assigned to them are referred to as **Smart Sprites**.

Mouse Bay – An author-defined area within the **Clicking On Objects Input** that users can click, or into which they can move a **draggable sprite**. Up to ten Mouse Bays can be set up within the **Input**'s **Presentation** window.

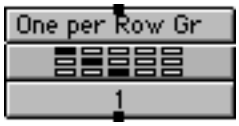
Mouse Bays Input – This **Input State** lets you create up to ten click-sensitive areas, called **Mouse Bays**, over a graphic. onViz will route your users through the application based on which Mouse Bay they click. You also use the Clicking on Objects Input when creating **draggable sprites**. onViz will track and report on the Mouse Bays into which a user has dragged a sprite. You can use **conditional paths** to check if a **sprite** is in a specific bay and route the **application flow** accordingly.

Multimedia State – Used for playback of multimedia components, and includes **Play Sound**, **Play Movie**, and **Overlay Image** Multimedia States.

Object Images – Shapes, or objects, created by mathematical formulas, or vectors. Because these objects are created by formulas rather than colored pixels, they usually result in a smaller file size than that of bitmap images. Object images are created with the **Drawing Tools palette**'s **Object tools**.

To change an object image's appearance, simply select the object and change some attribute, such as fill pattern, color, or border. To change an object image's size, select it and drag one if its handles; its size will change with no visible distortion. Text created with the Object Text tool remains editable after it is has been created; you can change its content, font face, style, and color at any time. Text created with the Object Text tool also supports **variable substitution**.

Object Tools – Part of the **Image Editor**'s **Drawing Tools palette**, these tools create **object**, or **vector-based**, **images**.



One Per Row Group – A type of **Group State** in which your project's **application flow** is routed by randomly picking one **State** from each of the Group's rows. This flow continues until the last row is reached, at which point the flow exits the Group State and moves to the next State in the **Application Map**. The One per Row Group does not require **Start States**, **Stop States**, or **Paths**, although Paths and Stop States can be incorporated to override the One per Row flow.

Output State – Presents information in the form of multimedia elements: graphics, text, animation, sound, and movies. By holding down the Control key while choosing from the **Map Tools palette**, you can access a contextual menu containing the four variations of Output States: **Design Window Visible**, **Design and Text Visible**, **Text Window Visible**, and **Hide Windows**.

Overlay Image – The image contained within an **Overlay Image Multimedia State**.

Overlay Image Multimedia State – Opens a window, containing an image, over an existing **Output**.

Parsing – Evaluating a user's response to a question in an **Input** against matching strings set up in the Enter Answer: fields of the Input's **InfoCenter**. When onViz parses a response, it compares it to the the first answer field to see if it matches. If the match is "true," the **Exit Point** corresponding to that answer field is taken. If the match is "false," onViz compares the response to the next answer field, and continues in this manner until a match is found to be "true," or until the response has been compared to all the answer fields.

Path – Represented by arrows, Paths determine the sequence in which each **State**'s activities are performed. You create a Path by clicking on a State's **Exit Point** and holding the mouse button down while dragging towards the desired State. Release the mouse button anywhere over the State's icon and the Path automatically connects to its **Entrance Point**. There are two types of Paths, **Unconditional** and **Conditional**. Unconditional Paths simply allow the flow of your project from one State to the next. Conditional Paths allow you to use formulas and **variables** in determining your project's **application flow**.

Pixel Tools – Part of the **Image Editor**'s **Drawing Tools palette**, these tools create **pixel-based**, or **bitmap**, **images**.

Pixel-based Image – Images that are comprised of thousands of tiny square dots, or pixels. Each individual pixel is "painted" a certain color so that, collectively, they form a picture. Because the pixel-based image stores this color information for each of its pixels, its file size is typically larger than **object images**. Occasionally referred to as **bitmap images**, these images are created with the **Drawing Tools palette**'s **Pixel tools**.

You alter a pixel-based image's color by "painting" over its pixels with another color. Scaling, or changing a pixel-based image's size in onViz' Image Editor often results in a "stair-step effect," as the image's colors are either stretched out over a larger number of pixels, or compressed into a smaller number of pixels.

Play Movie Multimedia State – Opens a movie window that will play over an existing **Output**.

Play Sound Multimedia State – Plays a sound over an existing **Output**.

Pointer – In **Application Maps** and **Group Maps**, the Pointer is found in the **Map Tools palette**, and is used to create, select, and move States and Paths. In the **Image Editor**, the Pointer is found in the **Drawing Tools palette**, and is used to select, move, and resize objects and graphics in the work area.

Pose – One of possibly several variations of a given **sprite**. A sprite can have up to sixteen poses, which are collectively known as a **sprite family**. Each of a sprite's poses share the same **Sprite Library** number, but have different Sprite Library letters (e.g. 2A, 2B, 2C, etc.). Poses are useful for creating sprites that seem to perform some activity, such as walking or running, as well as for automating the placement of sprites onto subsequent **frames** of animation.

Presentation – Used to create and modify the content displayed by a **State**, or functions that are carried out when **application flow** passes through a State. Only three States include a Presentation window: the **Group State**, the **Output State**, and the **Input State**. The Presentation window is accessed by double-clicking the rectangular area in the lower 2/3 of the State's icon, or by selecting the State and then choosing "Presentation (Command E)" from the Map menu.

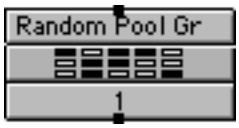
Print Options Gadget – This **Gadget State**, when **application flow** is routed through it, allows you to preset an **Output**'s page setup and printing options. Use this setting in conjunction with the Output's Print Window option to allow your application to print its contents automatically and seamlessly, without requiring any input from the user. Use this Gadget at the very beginning of your application to preset the printing options for all of your Outputs, or place it just before individual Outputs to preset their print options separately.

QuickSprite – A **sprite** created by Control-clicking in the **Design Window** and then choosing Quicksprite from the contextual menu, which opens the **Image Editor** to create a sprite. Quicksprites are "quick" because, after creation, they are automatically placed in the **Sprite Library** and in the **Design Window**.

Radio Button Input – Resembling a multiple-choice question, this **Input State** lets your user click one of up to ten radio buttons that correspond to a chosen response. The Radio Button Input allows only one answer at a time to be chosen from the list of possible answers.

The Radio Button Input also offers a quick way to give a user information or feedback. Simply put the information or feedback in the Question field, and do not enter anything in the Answer field. When onViz passes through a Radio Button Input set up in this way, the information is presented with an OK button. Once your users have read the information, they can dismiss the dialog and continue the application by pressing the Return key or clicking the OK button.

Random Pool Group – A type of **Group State** in which your project's **application flow** is determined by choosing **States** from the Group at random. The number of States to be chosen is designated in the Group **InfoCenter**, and random execution continues until that number is reached, at which point the flow exits the Group State and moves to the next State in the **Application Map**. The Random Pool Group State does not require **Start States**, **Stop States**, or **Paths**, although Paths and Stop States can be incorporated to interrupt the random flow.



Report File – The text file that is typically saved when an application is exited. The standard report file contains statistics about a user's performance in the application, such as score, percent correct, and how long the user spent in **Inputs** and **Groups**. However, by using a **Report Options Gadget**, you can modify a report's content and how often it is saved.

Report Options Gadget – This **Gadget State**, when **application flow** is routed through it, allows you to determine the content that goes into the application's report file, as well as how often the report is saved. If a Report Options Gadget is not used, the application will save its standard report upon exiting.

Restart Application Gadget – This **Gadget State**, when **application flow** is routed through it, restarts the application, either immediately or after a specified number of minutes.

Return State – Used in combination with a **Bridge State**, returns **application flow** to the Bridge's **Exit Point**. The Return State is one of five in the larger category of **States** known as **Flow States**.

Runtime – Any instance in which the application is operating as the end-user will experience it. Runtime can refer to an end-user operating the final product, but can also refer to test runs of the application. An application can be tested, while in the **Application Map**, by choosing "Run (Command R)" or "Run From Selected State (Shift Command R)" from the Map menu, or, while in an **Output Presentation**, by choosing "Run Animation (Command R)" or "Run From Current Frame (Shift Command R)" from the Animation menu.

Send Email Gadget – This **Gadget State**, when **application flow** is routed through it, sends an email in the background to a designated recipient; the user will be unaware that the email is being sent.

Show Web Page Gadget – This **Gadget State**, when **application flow** is routed through it, launches a Web Browser set to go to a specific Web site.

Smart Sprite – A **sprite** with **mouse actions** assigned to it. Mouse actions are assigned using settings found on the right-hand side of the **Sprite Attributes palette**, and control navigation by allowing the animation to jump to some other **frame**, or by causing **application flow** to exit the **Output** or to **bridge** to some other location in the project.

Sprite – An image stored in the **Sprite Library**, where it acquires an identity number (1-64) and letter (A-P). This identity allows it to be added to a **frame** and manipulated to form an animation. All sprites sharing the same Sprite Library number are considered poses within the same **sprite family** (e.g. 2A, 2B, 2C, etc.).

Sprite Attributes Palette – Divided into two sections, the **Sprite Library** on the left and **Mouse Actions** on the right, the Sprite Attributes palette creates **sprites** from images in the Image **Library**. These images are added to the Sprite Library where they are assigned attributes, such as sprite number and **pose**, so that they can be placed on individual **frames** of animation. Sprites are numbered from 1 to 64, and each number can have up to sixteen poses, designated by the letters A–P. In its Mouse Actions section, the Sprite Attributes palette creates **Smart Sprites** by assigning behaviors to them that are affected when the cursor passes over or clicks on them. The Mouse Actions section also sets sprites to be **draggable** and resets sprite variables.

Sprite Family – A **sprite** and all of its **poses**. All sprites within the same family have the same **Sprite Library** number and are lettered sequentially (e.g. 2A, 2B, 2C, etc.).

Sprite Library – Located on the left side of the **Sprite Attributes palette**, the Sprite Library creates **sprites** by assigning identity number (1-64) and letter (A-P), or **pose**, to images stored in the Image **Library**.

Start State – Begins **application flow** in all **freeform Application Maps** and **Group Maps**. The **Start State** is one of six in the larger category of **States** known as **Flow States**.

State – Accessed from the **Map Tools palette**, States are the basic building blocks of onViz, and contain your project's content and perform its activities. Represented by rectangular icons, States have an **Entrance Point** on their upper edge where **Paths** enter, and one or more **Exit Points** on their lower edge where Paths leave. The top third of the State icon is the **InfoCenter**, where you enter the State's Name and select options to determine its specific functions or runtime behavior. The lower two-thirds of the State icon is the **Presentation**, which provides access to tools for adding content and modifying the display viewed by your audience. Types of States include **Flow States**, **Group States**, **Output States**, **Input States**, **Gadget States**, **Multimedia States**, and **Calculator States**.

Stop State – Ends the application when reached in the top level **Application Map**, and exits a **Group** when used within a Group State. The Stop State is one of six in the larger category of **States** known as **Flow States**.

Support Folder – If, during the Project Setup process, a new **top-level application** is created, a corresponding set of support folders will be created also. During **development**, the support folder contains all of the media assets used by your application. The support folder contains the following directories: Fonts, Images, Movies, Sounds, and MiscDocs. The MiscDocs folder contains the media asset library, any onViz **supporting documents** you have created, and any applications or documents (or shortcuts to them) to which your project is **bridging**. These directories can be renamed in the Project Attribute's Runtime dialog, found under the File menu. If the top-level application is created at the same directory level as a pre-existing top-level application and support folder, onViz will not create a new support folder, but will display a notification stating that the applications will share the existing support folder.

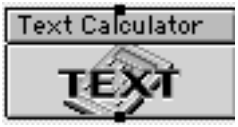
After development, when you **build** your **target application**, onViz will either integrate the support folder into your application (single file application), or leave them separated (multiple file application), depending on the options selected in the **Build Target** dialog. If the multiple file option is selected, you will need to specify a file path from your top-level application the support folder, so that onViz can have access to it.

Supporting Document – Secondary files in your project, **bridged** to from the **top-level application**. During the Project Setup process, you have the choice of creating a top-level application or a supporting document. When a supporting document is created, it is placed in the **support folder**'s MiscDocs directory.

Target Machine – The ideal computer system configuration for your target audience. This ideal system will influence the decisions you make regarding the technologies and minimum

requirements used by your application. When your users run your application, it will check their systems to ensure that they have these minimum system requirements.

Text Calculator State – Assigns values to text **variables**.



Text Window – The **Output** window in which text is displayed. You can access a Text Window during development by using a **Text Window Visible Output**, a **Design and Text Visible Output**, or a **Hide Windows Output**. If using a **Design Window Visible Output**, the Text Window can be toggled on and off by selecting "Show Text Window (Shift Command T)" from the Edit menu.

Text Window Visible Output – An **Output** used for presenting text with no graphics. The Text Window Visible Output displays text in a single **Text Window**, and allows control over formatting options such as font face, style, size, color and alignment. Often referred to as simply a Text Output.

Timer Gadget – This **Gadget State**, when **application flow** is routed through it, displays a timer with an **Output** or **Input State**. The Timer Gadget can be used to start, stop, resume, or pause a timer's count. A **Bridge** action can be added to the timer to signal an expiration or a warning.

Top-level Application – The primary file making up your onViz project. During the Project Setup process, you have the choice of creating a top-level application or a **supporting document**. When a top-level application is created, it may contain the bulk of your project's content, or it may just provide routing decisions and rely on the supporting documents for the detailed content, depending on factors such as how your project will be delivered and your personal development style. Every onViz project must contain one top-level application.

Transition – A dynamic transformation between the end of one **frame** of animation and the beginning of the next frame. Transitions are chosen using the **Frame Control palette**, and typically affect only the elements that have changed from one frame to the next (if nothing has changed, there is no transformation). However, a transition applied to the first frame of an animation affects the entire frame (a transformation from no frame to the first frame).

Tween – A special type of **transition** available for use only between two adjacent **frames** of animation. Tweening looks at the difference in a **sprite**'s size and location between the two frames, and at the set transition time, and then calculates how the sprite should move to make a smooth transition from the first size and location to the next.

Unconditional Path – A **Path** in which no conditions have been set, which simply allows your project's **application flow** to move from one **State** to the next. Appears as a black arrow in the **Application** and **Group Maps**.

Variable – A "container" used for storing and displaying information. The variables used within onViz can hold information gathered from user input, or information generated by an application during **runtime**. There are five categories of variables: Input, Output, Group, General, and **Custom**.

Variable Substitution – The act of using a **variable** during **development** to stand in place of information that will be collected during **runtime**. During runtime, the information represented by the variable may be displayed, stored as a **custom variable**, or used to make branching decisions.

Vector-based Images – Shapes, or objects, created by mathematical formulas, or vectors. Because these objects are created by formulas rather than colored pixels, they usually result in a smaller file size than that of **bitmap images**. Vector-based images are created with the **Drawing Tools palette**'s **Object tools**.

To change a vector-based image's appearance, simply select the object and change some attribute, such as fill pattern, color, or border. To change an vector-based image's size, select it and drag one if its handles; its size will change with no visible distortion. Text created with the Object Text tool remains editable after it is has been created; you can change its content, font face, style, and color at any time. Text created with the Object Text tool also supports **variable substitution**.

Work Area – In the **Application Map** and **Group Map**, refers to the window in which **States** and **Paths** are created. In the **Image Editor**, refers to the window in which images are created.

Zoom Tool – When used in an **Application Map**, the Zoom tool is found in the **Map Tools palette**, and zooms the map view out in order to see more of the Application Map **work area**. The Zoom tool can also be used to get back to normal map view. Note that you must be in normal map view to work on your Application Map. A Zoom tool is also found in the **Image Editor**'s **Drawing Tools palette**, where it can be used to zoom in on your images to work in greater detail.

Index

A

B

C

D

E

F

G

M

O

P

Q

R

S

T

V

W