SOFTIMAGE®

SOFTIMAGE®|XSI™

Version 1.0

Fundamentals

Avid

Fundamentals was written by Judy Bayne and Grahame Fuller; edited by Edna Kruger and John Woolfrey; and formatted by Luc Langevin.

Printed in Canada.

Document No. 0130-04612-01 0400

Contents

Roadmap

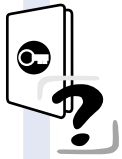
About This Guide

Fundamentals contains information and step-by-step procedures about the interface. It also provides tools and techniques you will use regularly while working in SOFTIMAGE®|XSI™.

- ***Chapter 1: Getting Started***—introduces you to SOFTIMAGE|XSI and explains how to start and exit the application.
- ***Chapter 2: The Interface***—the basic tools in the interface that you will use to build, animate, and render your scenes.
- ***Chapter 3: Managing Projects & Scenes***—covers all aspects of creating, building, and managing scenes in projects. Also covers how to merge models created in a 3D application into scenes and export scene files.
- ***Chapter 4: Viewing Your Work***—how to view elements in your scene, including the many options available to you in each of the viewports, such as viewing in wireframe, constant, or shaded mode; hiding and redisplaying objects; and zooming, panning, and dollying.
- ***Chapter 5: Working with Scene Elements***—how objects, shaders, and operators in a scene can be defined, selected, organized, duplicated, and deleted.
- ***Chapter 6: Working in 3D Space***—begins with an introduction to the fundamental concepts you need to know as you create scenes that simulate three-dimensional space. It then describes how to transform and deform objects within a 3D environment.
- ***Chapter 7: Commands & Scripts***—explains how to use the command box, script editor, and command log file. You can use scripts to automate repetitive tasks and create custom commmands. The chapter also includes some examples that you can refer to when writing your scripts.
- ***Chapter 8: Customizing SOFTIMAGE|XSI***—describes how to customize the versatile toolset to suit your own preferences and work habits.
- *Appendix: Standalones*—describes how to use the various standalone utilities that you can run from a command line.

Where to Find Information

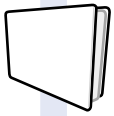
The SOFTIMAGE|XSI package includes a comprehensive set of learning materials. Use this Roadmap to find the information you need to get up and running quickly and effectively.



Start with the ***Setup Guide*** to install and license all components.
Setup Online Help is also available as you go through the process.
We recommend you choose Custom install so that you can perform the tutorials.

Refer to ***Release Notes***, an online listing of known problems and limitations for this version. Also includes workarounds and supplemental information. Access through the web at **www.softimage.com > support**.

Follow the ***Guided Tour*** (available from the Online Library CD).
This is a set of videoclips that provide overviews of features and tools.

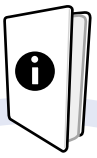


Work through ***Tutorials*** to learn the features in the context of basic productions.
This is a full-color set of lessons showing you step-by-step how to perform typical tasks.
You can install the scenes from the Software CD. (Choose Custom install when installing SOFTIMAGE|XSI). Then choose the **Content** option to install the Tutorials project.

The Softimage Discussion Group

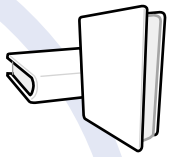
You can join the worldwide network of Softimage users exchanging ideas and techniques by e-mail. To find out more, e-mail **majordomo@softimage.com.** Leave the Subject line empty and type the word "help" in the body of your mail message.

The ***Global Index & Glossary*** is an index to all user guides and *Tutorials*; a glossary of terms; and a list of books, videos, and web sites related to the 3D animation industry.



The **user guides** contain conceptual information and procedures on how to use specific tools. These comprise**:**

- Fundamentals
- Animating
- Modeling & Deformations
- Shaders, Lights & Cameras
- Rendering



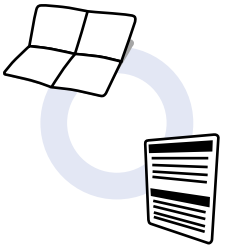
The Online Library CD
The Online Library contains the Guided Tour and all the SOFTIMAGE|XSI and some mental ray documentation in electronic form in both PDF and HTML formats. (See next page for how to use.)

Online Help
On-screen reference information on interface elements, commands, and parameters. There are two ways to access it:

- Click the **?** button in any property editor or tool view.
- Choose **Help > Contents and Index** from the main-menu bar.

Using SOFTIMAGE|3D with SOFTIMAGE|XSI provides tips and techniques about using the two software packages. Available from the Online Library CD and **softimage.com > support** only)

HTML Scripting Reference
An HTML-based reference help on the syntax for all scripting commands and arguments. It appears in your default HTML browser. Click on the icon (above) to open the script editor, then click **Help > Scripting Reference** or press **F1**.



Pin up the ***SOFTIMAGE|XSI Interface Layout*** and the ***Quick Reference Card*** to help you become familiar with the interface and keyboard shortcuts.

Using the Online Library

The Online Library contains the *Guided Tour* and all the SOFTIMAGE|XSI and some mental ray documentation in electronic form in both PDF and HTML formats.

For full-text searching and printing, we recommend PDF format. If you do not have Acrobat Reader installed, you can install it it free of charge from the Online Library CD: Follow the instructions in the readme file on the CD.

To access the Online Library

1. Insert the Online Library CD in your disk drive.
2. Open one of the following documents:
   - **mainmenu.pdf** (PDF format)
   - **mainmenu.htm** (HTML format)

Document Conventions

The following are ways that information is displayed in the SOFTIMAGE|XSI documentation.

Typography Conventions

Type style	Usage
Bold	Menu commands, dialog-box and property-editor options, and file and directory names.
Italics	Definitions and emphasized words.
`Courier`	Text that you must type exactly as it appears. For example, if you are asked to type `mkdir style`, you would type these characters and the spacing between words exactly as they are appear in this book.
>	The arrow (>) indicates menu commands (and subcommands) in the order that you choose them: *Menu name* > *Command name*. For example, when you see **File** > **Open**, it means to open the **File** menu and then choose the **Open** command.

Visual Identifiers

These icons help identify certain types of information:

Notes are used for information that is an aside to the text. Notes are reminders or contain important information.

Tips are useful tidbits of information, workarounds, and shortcuts that you might find helpful in a particular situation.

The 3D icon indicates information about differences in workflow or concepts between SOFTIMAGE|3D and SOFTIMAGE|XSI. You will find these very helpful when working with the two products.

Warnings are used when you can lose or damage information, such as deleting data or not being able to easily undo an action. Warnings always appear *before* you are about to do such a task!

Keyboard and Mouse Conventions

SOFTIMAGE|XSI uses a three-button mouse for most operations. These are referred to as the *left*, *middle*, and *right* mouse buttons. In many cases, you will use the different buttons to perform different operations; always use the left mouse button unless otherwise stated.

The two-button mouse is not supported in SOFTIMAGE|XSI.

This table shows the terms relating to the mouse and keyboard.

When this term is used...	**...it means this**
Click	Quickly press and release the left mouse button. Always use the left mouse button unless otherwise stated.
Middle-click	Quickly press and release the middle mouse button of a three-button mouse.
Right-click	Quickly press and release the right mouse button.
Double-click	Quickly click the left mouse button twice.
Shift+click, Ctrl+click, Alt+click	Hold down the Shift, Ctrl, or Alt key as you click a mouse button.
Drag	Hold down the left mouse button as you move the mouse.
Alt+*key,* Ctrl+*key,* Shift+*key*	Hold down the first key as you press the second key. For example, "Press Alt+Enter" means to hold down the Alt key as you press the Enter key.

Chapter 1 Getting Started

SOFTIMAGE®|XSI™ is the next-generation 3D system that integrates modeling, animation, and rendering into a single, seamless environment. It incorporates the tools and functions of SOFTIMAGE®|3D but goes far beyond in terms of tool sophistication and artistic control.

Seamless Modeling

The modeling features are specifically designed for creating and editing seamless animated characters. It offers tools for creating, editing, and deforming NURBS curves and surfaces. It also lets you create a new type of surface-mesh geometry that makes multiple NURBS surfaces behave as though they were one.

Non-linear Animation

SOFTIMAGE|XSI provides high-level non-linear editing and mixing functions in addition to the fundamental, detailed tuning tools that include keyframe and curve editing, custom parameters, linked parameters, constraints, expressions, shape animation, channels, and particle simulation.

The powerful animation mixer lets you mix, transition, and blend function curves, expressions, constraints, shapes, clusters, actions, audio, and images in a familiar non-linear environment. Animation is supported with 2D and 3D inverse kinematics, envelopes, and weight maps, storable actions, and powerful animation data-mapping templates.

Limitless Texture and Material Creation

By means of a graphical node-based connection tool called the *render tree*, you can create an unlimited range of materials by connecting any shader—based on a vast library of presets—to any object input, including surface, volume, environment, contour, displacement, shadow, photon, photon volume, or bump. You can also project 2D and 3D textures into texture spaces, which can then be manipulated like any 3D object.

Interactive Photorealistic Rendering

Drawing upon mental ray® version 2.1 rendering technology, the SOFTIMAGE|XSI renderer offers full-resolution interactive photorealistic rendering, caustics, global illumination, and motion blur. If compositing is required, you can break up render passes into smaller passes (such as specular, shadow, highlight, caustic, matte, etc.), which can in turn be edited non-destructively—all in a single scene.

Open Architecture

SOFTIMAGE|XSI provides full-import capability of SOFTIMAGE|3D scene data, including geometry, materials, textures, and animation.

Using any ActiveX-compliant scripting language (such as VBScript and JScript), you can easily automate repetitive tasks, build macros, customize the interface, communicate with other script-enabled applications (data transfer, mail notification), and record and replay creation steps.

Starting and Exiting

The method you use to start the software depends on which operating system you're using: Windows NT or IRIX systems. Methods for starting on both operating systems are described in this section. Exiting is the same on both operating systems (see *Exiting* on page 23).

SOFTIMAGE|XSI runs in its own window, which means that you can minimize, maximize, resize, and move it as you would any other window. You can also switch between it and any other open window (Alt+Tab for Windows NT users). For more information about these tasks, see your Windows NT or IRIX operating-system documentation.

Once You're Up and Running...

If you are starting SOFTIMAGE|XSI for the first time, the first dialog box to appear after the "welcome" screen is the Project Browser. Here, you can create a new project or choose from a list of projects from the browser's Project List window. For more information, refer to *Creating and Opening Projects* on page 58.

If you have already worked in SOFTIMAGE|XSI on previous occasions, the interface will open the last project created and display an empty scene, called "Untitled."

Starting SOFTIMAGE®|XSI™ on Windows NT Systems

There are several ways to start on a Windows NT system:

- Using the **Start** menu (see below).
- Using the command line (see *Starting from the Windows NT Command Line* on page 21).
- From a shortcut you create on your desktop (see your Windows NT operating-system documentation) or by double-clicking on the executable file called **xsi.exe**. By default, the executable file is in the **c:\Softimage\XSI_1.0\bin** folder.

Starting with the Windows NT Start Menu

If you installed in the default configuration, you can start it from the Windows NT **Start** menu in the lower-left corner of the Windows NT desktop:

Choose **Start** > **Programs** > **Softimage Products** > **SOFTIMAGE XSI 1.0**.

To modify the startup options on the **Start** menu, edit its shortcut in the following folder:

```
c:\winnt\profiles\all users\start menu\programs\softimage products\SOFTIMAGE XSI 1.0
```

Starting from the Windows NT Command Line

Starting from the Windows NT command line lets you specify various startup options, such as the project you want to load.

To start from the Windows NT command line

1. Open a command-prompt window. If your Start menu is in its default configuration, you can choose **Start** > **Programs** > **Softimage products** > **SOFTIMAGE XS1 1.0** > **Command Prompt**.

   Do not use a standard MS-DOS shell (unless you run the **xsi.bat** file first).

2. Enter the command to start. To start without any options, type:

   ```
   XSI
   ```

   To start and load a specific project file, type the startup option on the same line after the above command. For example, type:

   ```
   XSI [project file name]
   ```

Once SOFTIMAGE|XSI has started, you can exit the command-prompt window at any time.

Creating a Shortcut for Starting on Windows NT

To create a shortcut to SOFTIMAGE|XSI on the Windows NT desktop

1. From the Windows NT Explorer, go to the directory where the XSI batch file (xsi.bat) resides. By default, this should be in **XSI_1.0\bin**.
2. Right-click on the **xsi.bat** file and choose **Create Shortcut**.

   A shortcut to the batch file is created.

3. Drag the shortcut icon to your desktop.
4. Start by double-clicking the shortcut icon.
5. To change the name of the shortcut, click on the icon's name and then type a new name.

To modify a shortcut by specifying startup options

1. Right-click on the shortcut icon and choose **Properties** from the pop-up menu.
2. On the Shortcut page in the Shortcuts property editor, add any desired options to the command line specified in the **Target** text box. Make sure to add the options inside the quotation marks after the command that starts the program.

For more information about the desktop, shortcuts, the Windows NT Explorer, and folder windows, please see your Windows NT documentation.

Starting SOFTIMAGE|XSI on IRIX Systems

To start on an IRIX system, choose **Softimage Products** > **SOFTIMAGE XSI_1.0** from the Toolchest.

If you start SOFTIMAGE|XSI from the Toolchest, you may want to open a console window to monitor any error messages that may occur while you're working with SOFTIMAGE|XSI. To open a console window, choose **System** > **Start New Console** from the Toolchest.

The copyright screen appears after you start.

Starting from an IRIX Shell

You can also start from a shell if you want to start it using specific options.

1. From your user account, type `source .xsi_1.0`

You must now source the .xsi_1.0 file manually because the CSHRC file no longer sources the environments needed.

2. Type `xsi` at the prompt. The "welcome" screen appears.

If you will be starting from a shell on a regular basis, you can configure the CSHRC file so that you will not have to source the .xsi_1.0 file manually every time you start.

1. Make sure that the `.xsi_1.0` file is in your home directory.

   If the file does not exist, you have not yet updated your user account.

2. Use a text editor (such as vi or jot) to edit the CSHRC file.

   If you are upgrading from a previous version of SOFTIMAGE|3D, delete the line reading `source ~/.softimage`

3. Add a line reading `source .xsi_1.0`

   - If SOFTIMAGE®|Eddie version 3.5 SP1 is installed on your computer, this line must appear after the line that sources the SOFTIMAGE|Eddie 3.5 SP1 Variables.
   - If an earlier version of SOFTIMAGE|Eddie 3.5 SP1 is installed on your computer, this line must appear before the line that sources the SOFTIMAGE|Eddie 3.5 SP1 variables.

Exiting

After you have completed your work session, save your work and exit.

Depending on your preferences, any changes you made to your workspace layout are saved when you exit and automatically recalled the next time you start.

To exit

1. Do one of the following:
   - Choose **File** > **Exit** from the main-menu bar.

     or

   - Click the close icon (×) at the far right of the title bar.

     or

   - Press Alt+F4.

2. If you have made any changes to the scene or the layout, you are prompted to save them before exiting. Click one of the following:
   - **Yes** to save your changes. For more information about saving scenes in general, see *Saving and Renaming Scenes* on page 64.

     or

   - **No** to exit without saving.

     or

   - **Cancel** to continue working.

Chapter 2 The Interface

The SOFTIMAGE®|XSI™ Interface

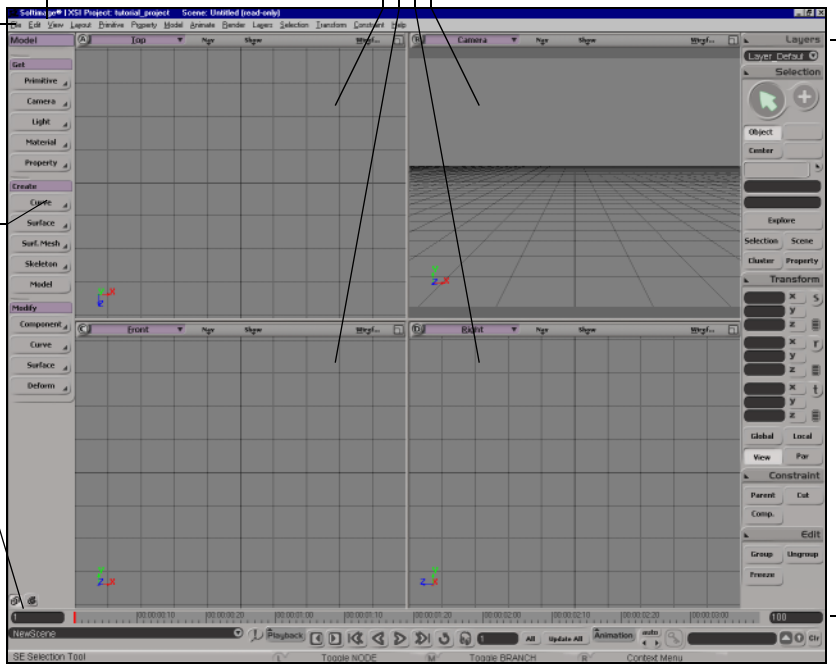
The illustrations below show the default layout of the interface. The interface is made up of controls panels and viewports, each of which can be resized, added, or deleted. You can save your layouts and reuse them as needed. For information on customizing layouts, refer to *Customizing the Layout* on page 217.

Viewports
Let you view the contents of your scene in different ways. You can resize, hide, and mute viewports in any combination.

Title bar
Displays the name of the project and open scene.

Main command area
(See page 30 for details.)

Main-menu bar
Provides access to all the primary commands.



Toolbar
Displays one of three toolbars used in modeling, animation, and rendering. Press 1, 2, or 3 on the keyboard to switch between toolbars.

Toolbar/Palette icons
Switch between toolbar view and palette view.

Lower interface controls
(See next page for details.)

Lower Interface Controls

Timeline
Displays the current position of your animation in time and lets you move manually between frames.

Script editor button
Displays an editor that lets you create, save, and modify scripts.

Animation panel
Provides access to a number of animation controls that let you edit function curves, set keyframes, and more.

Mouse/status line
Displays the current functions of your three mouse buttons as well as status and error messages when applicable.

Command box
Displays the last executed command and lets you type commands in manually. The arrow button displays a list of the most recently executed commands, which you can select and reuse.

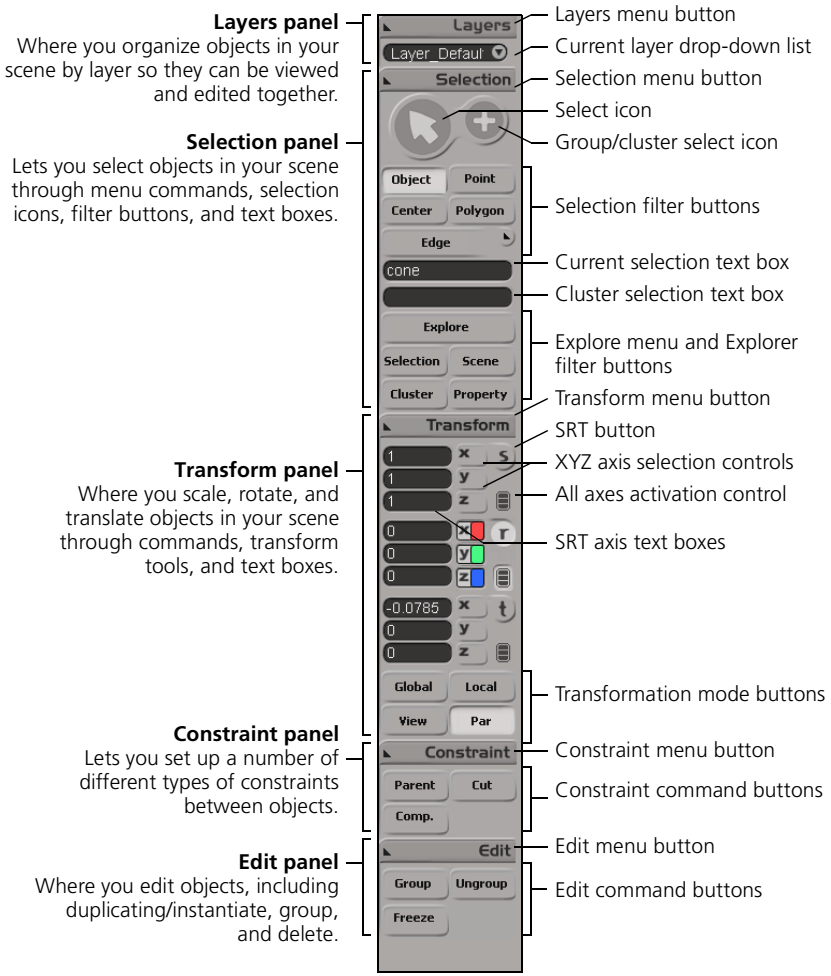
Playback panel
Lets you control animated scenes through a series of playback controls.

Refer to each item's Online Help topic (choose **Help > Contents and Index > Main Window**) for more information on the controls described above, as well as in the following areas of the documentation set.

- Command box—see *Using the Command Box* on page 174.
- Script editor button—see *Using the Script Editor* on page 175.
- Playback panel—see *Chapter 1: Basics of Animation* in the *Animating* guide.
- Animation panel—see *Chapter 1: Basics of Animation* in the *Animating* guide.
- *SOFTIMAGE®|XSI™ Interface Layout*—a foldout map of the main interface that you can post on your wall.

Main Command Area

The main command area on the right-hand side of the screen contains all the commands and tools you will most frequently use while working in your scene. Commands and tools that have a similar purpose are grouped into panels.



See the following areas in this guide for information on each group of commands found in the main command area. You can also search Online Help for a description of each individual command.

- Layers panel commands—see *Layers* on page 144.
- Selection panel commands and tools—see *Editing Parameter Values* on page 110.
- Transform panel commands and tools—see *Transforming Objects* on page 154.
- Constraint panel commands—see *Chapter 4: Animating with Constraints* in the *Animating* guide.
- Edit panel commands—see *Chapter 5: Working with Scene Elements*.

Choosing Commands

Most commands can be accessed from drop-down menus (main-menu bar) or from pop-up menus by clicking on a menu button like the ones found in the toolbars and the panels of the main command area.

Some pop-up menus are displayed by Alt+Ctrl-clicking on individual elements, such as geometric objects, cameras, and lights or from specific areas of the interface, such as the explorer, the viewports, etc. These type of pop-up menus are also called *context menus* because they provide commands that are applicable to the object you have chosen.

Windows NT users can also access context menus by pressing Alt while holding down the right mouse button.

A description of all commands is available from Online Help. Choose **Help > Contents and Index**.

A pop-up menu from a menu button in the Selection panel.

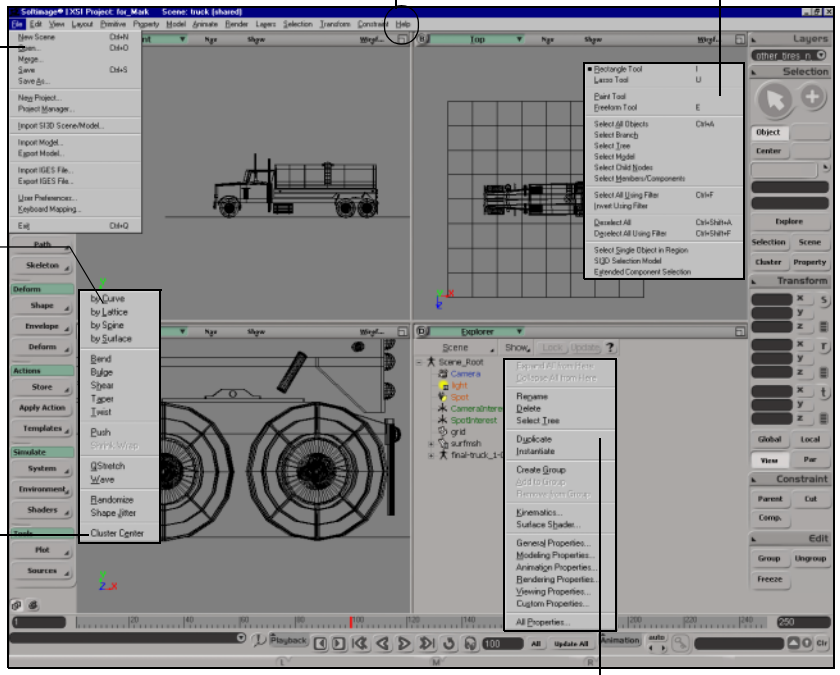
A drop-down menu from the main-menu bar.

A pop-up menu from a menu button on the toolbar.

The triangle at the edge of a button indicates that clicking the button opens a pop-up menu.

Select a command from the menu with your cursor *or* as a shortcut, press a key that corresponds to an underlined letter in the menu to instantly activate the command.

Middle-click the menu button to repeat the last command selected.



A pop-up menu from an element you right-click in the explorer.

This is also called a context menu.

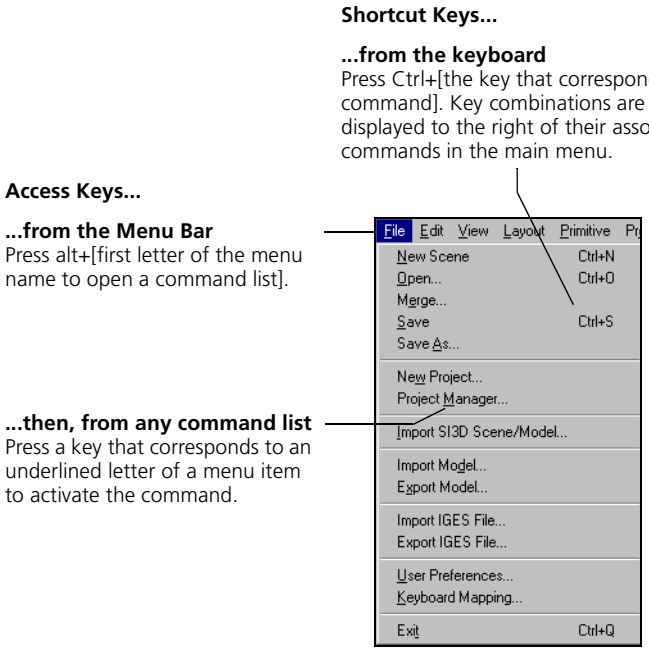
Using Keyboard Shortcuts

Keyboard shortcuts are specific key combinations that perform the equivalent of certain menu commands. There are several types of keyboard shortcuts:

- **Access keys**: The underlined letters you see in many menu names. Pressing **Alt**+[underlined letter] activates its associated command.
- **Shortcut keys**: Commands that have been mapped to your keyboard. For example, pressing **Crtl**+[keyboard key] activates an associated command.

  Shortcut keys are listed to the right of many menu command names. See Online Help or the Quick Reference card for a list of all available default keyboard shortcuts.

  Note that not every menu command has a shortcut or access key.You can create your own keyboard shortcuts as described in *Creating Keyboard Shortcuts* on page 215.



Switching to SOFTIMAGE|XSI Key Mapping from SOFTIMAGE|3D

When you start SOFTIMAGE|XSI for the first time, you are prompted to choose between the SOFTIMAGE|3D and SOFTIMAGE|XSI interaction model, which includes their own respective set of mapped shortcut keys.

If you have already chosen Softimage 3D interaction and now want to switch to SOFTIMAGE|XSI's shortcut keys, do the following:

1. Choose **File** > **User Preferences** > **Interaction**.
2. Toggle **Select Entire Contents of Text Boxes on First Click** on.
3. In the Tool Activation controls, toggle **Enable Sticky Keys** on.
4. In the Display Performance controls, toggle **Highlight Property Set Owners** on.
5. In XSI Collection Parser Settings, toggle the **Check All Models**... on.

Using the Mouse

You can use the mouse for performing basic operations such as manipulating objects in a scene, choosing commands, and selecting options in dialog boxes and property editors.

A three-button mouse is used on both the Windows NT and IRIX operating systems.

Supra Keys

The supra key is a type of shortcut key with its own special type of behavior. Supra keys are mapped to the most commonly used tools, and pressing a supra key activates a specific tool. Once the tool is active, any actions you perform with the mouse are related to that tool. For example, pressing the **z** key activates the pan and zoom tool so that holding down the right mouse button zooms out of the scene, and holding down the middle mouse button zooms in. The mouse pointer changes shape to indicate which tool is active.

In this guide, the supra keys mentioned are always the default ones. Changes you make to supra keys can be saved to a file containing keyboard setup information. You can always reset the supra-key settings to the default values.

Activating Tools in Sticky or Temporary Mode

Tools can be activated in one of two modes: "sticky," where you do not need to hold the key down to activate the tool; and "temporary," where the key must be held down for the tool to remain active.

Sticky-Key Mode

You can activate a tool in this mode by quickly pressing and releasing its supra key. The tool remains active until you choose another tool or press another supra key to perform a different operation. Pressing a supra key a second time deactivates the tool.

When you open SOFTIMAGE|XSI, you are placed in sticky-key mode by default. The property page in the User Preferences dialog box lets you toggle sticky keys on and off.

To toggle sticky-key mode on and off

1. Choose **File** > **User Preferences** to open the User Preferences dialog box, then click the **Interaction** tab.
2. In the Tool Activation box, click **Enable Sticky Keys** to toggle sticky-key mode on or off.
3. You can change the maximum amount of time (**Delay**) it takes to activate a tool in sticky-key mode. By default, pressing a key for no longer than a quarter a second will activate sticky keys.

Temporary Mode

You can activate a tool in this mode by holding down its supra key for longer than a quarter of a second. When you hold down a supra key and drag the mouse; the tool is switched off after you release the supra key. For example, when you release the **o** key after holding it down and dragging the mouse, you deactivate the "orbit" tool.

Using Supra Keys to Activate Tools

Each tool can be activated by a corresponding supra key. You can move back and forth between different tools quickly by using these supra keys. The current tool determines the way the mouse buttons behave. Only one tool can be active at a time.

The basic tools include:

- Selection (of either geometric objects, groups, components, or clusters).
- Transformations of geometric objects, groups, components, or clusters (such as scale, rotate, translate).
- Camera manipulation (such as pan and zoom, orbit, dolly).

Example

1. Start SOFTIMAGE|XSI and load a scene.
2. By default, the Select icon (the large arrow button on the Selection panel) is on and highlighted in green, indicating that you are now in selection mode. If the Select icon is not on, press the space bar to switch it on.

   Object-selection mode maps your left-, middle-, and right-mouse buttons to node, branch, and tree selection, respectively. If you select an object that is part of a hierarchy, left-clicking selects only the object, middle-clicking selects all objects in its branch, and right-clicking selects the entire tree (in the viewports only).

   You can select objects in the explorer at any time, regardless of the active tool.

3. Select an object in a User view by clicking on it.

   After you have selected the object, you can manipulate it, for example, by using a transformation tool such as scaling.

4. Click the **Local** transformation mode button in the Transform panel.
5. Press the **x** key and drag the mouse while holding down the left, middle, or right mouse buttons to scale along the X, Y, or Z axes, respectively. The **S** button in the Selection panel is highlighted and the pointer changes to indicate that you are using the scale tool. When you release the **x** key, the scale tool is deactivated and you return to the previous tool (selection tool).

   For more information on selection and transformation modes, see *Working with Scene Elements* on page 103.

To get a different view of your work, you can change the camera viewpoint.

6. In a User view, press the **o** key and drag while holding down the left mouse button to orbit your scene.
7. Quickly press and release the **z** key. This activates the zoom and pan tool in sticky mode.

8. Left-click and drag to pan your scene, hold down the middle mouse button to zoom in, and hold down the right mouse button to zoom out. Notice that when you release the mouse button you are still in zoom and pan mode. This is because the tool was activated as sticky: if you quickly press and release the key, the tool is activated and will remain that way until you select another tool.
9. To deactivate the zoom and pan tool, press the **z** key again or press Esc.

   For information on camera-manipulation tools, see *Navigating in 3D Views* on page 87.
10. Quickly press and release the **a** key. This activates the combination tool where:
    - The left mouse button lets you **pan**.
    - The middle mouse button lets you **dolly**.
    - The right mouse button lets you **orbit**.
11. To activate another tool, press any supra key or choose a tool from the main command panel. For example, you can switch to selection mode by pressing the space bar (the selection supra key) or by clicking the selection button on the main command panel.

Which Tool Is Active

You will often move back and forth between tools as you work. Here are a few ways to keep track of the tool you are working with:

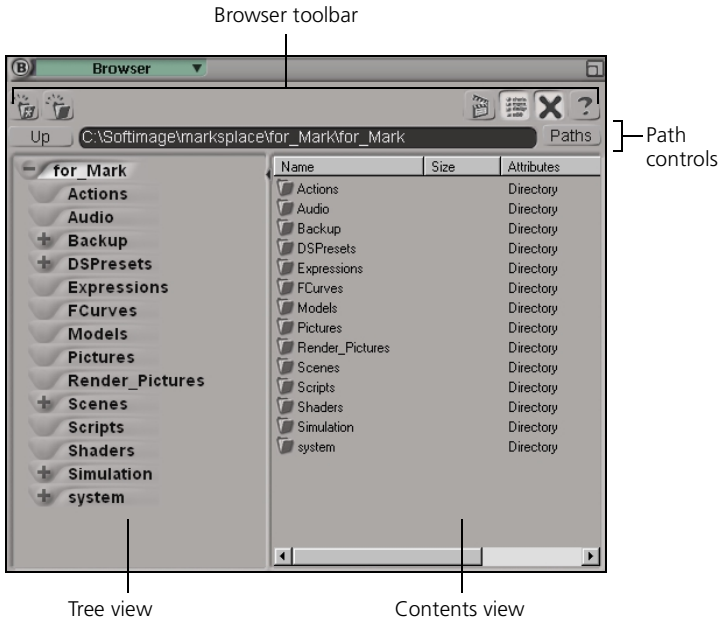
- You can activate a number of tools from the main command area. If the Select icon (large arrow) is highlighted, for example, you know that your mouse is mapped to the selection tool. If one of the SRT buttons is selected, you are in Transformation mode, and so on.
- In most cases, the mouse pointer changes to indicate the active tool. For example, when the orbit tool is active, the pointer turns into an orbit icon.
- The mouse line at the bottom of the interface displays which operation is mapped to each of the mouse buttons. Always check the mouse line for the current tool status.

The Browser

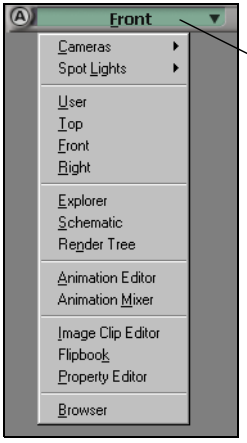
The browser is a type of window specifically used to:

- Search (browse) through scene databases, project directories, preset libraries, and other repositories whose files are required to build a project.
- Import scene files from databases as well as load scene and project files.
- Perform file-management tasks such as moving, copying, renaming, and deleting files.

The standard Windows NT browser is also used for certain operations, such as opening and saving a file.



To open the browser



Click on the arrow or label to the left of any viewport and choose **Browser** from the list. The browser opens in that viewport

or

Choose **View > Views > Browser** from the main-menu bar. This opens the browser in a floating window.

Active-browser icon

To view the active browser

If you have more than one browser open, the active-browser icon, when selected, makes its browser the active one.

The Tree View

The browser's tree view is a hierarchy of folders and subfolders that contain files belonging to the scenes, objects, properties, or presets that you use to build a project. When you select an item from the tree, its contents (if any) are displayed in the list view of the browser.

You can view items in the tree view by expanding or collapsing folders.

To move up a level in the browser tree view

- Click the **Up** button in the browser toolbar.

  or

- Press the back space key.

To create a new folder

1. Click on the New Folder icon in the browser toolbar.
2. Type a name in the text box that appears and click OK.

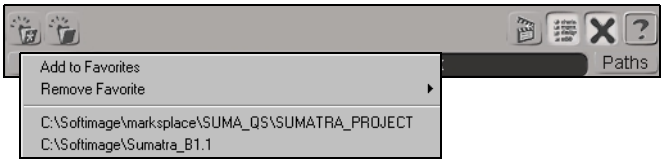
Setting Favorites

Favorites provide you with quick access to a folder without having to remember where the file is located. The Favorites icon lets you mark a direct path to a folder of interest on any local, network, or external disk to which your workstation is connected.

To add a favorite to the list

1. In the browser tree view, navigate to the folder that you want to add as a favorite.

2. Click the Favorites icon in the browser toolbar.
3. Choose the **Add to Favorites** command in the menu that appears.



The new favorite is added to the Favorites list at the bottom of the menu, and the browser updates to set the favorite folder at the top of the tree view.

To select a favorite

Click the Favorite icon and select a favorite from the list that appears at the bottom of the menu. Your selection is displayed in the browser.

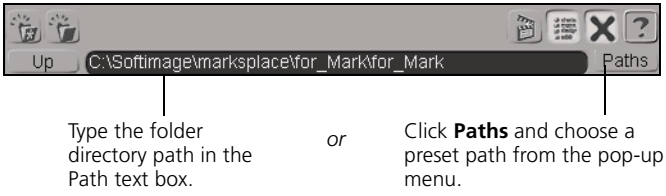
To delete a favorite

Click the Favorite icon, choose the **Remove Favorite** command, and select a favorite from the submenu that you want to delete. The favorite path is deleted from the list. The folders or files are not affected.

Accessing Files from the Browser Path Controls

The browser's Paths controls let you access files by inputting their folder directory path in a text box or by selecting a preset path from a pop-up menu.

To access files from the browser path controls



Once you have made your selection, the browser's tree view updates and displays the path's folders.

Using the Paths Option

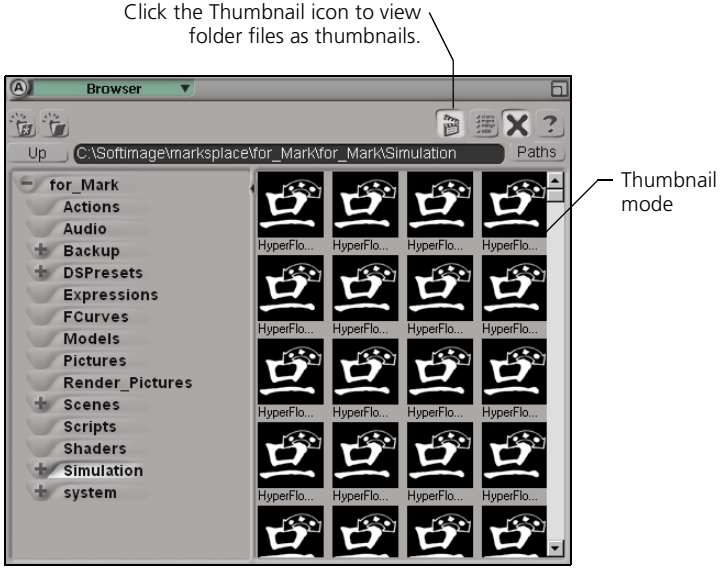
The Paths pop-up menu lets you choose from a list of directory paths, including paths that point to available projects.

Choose...	To
Factory	Access files found in the default path. This path is set during installation and can't be modified.
Workgroup	Access files that are shared by a group of people of your choosing. You can set this path in the Favorites section of the User Preferences dialog box.
Project	Access files for the project in which you are working. By default, this folder is located in your working project folder. This path is set during installation and can't be modified.
User	Access your own personal files. You can set this path in the Favorites section of the User Preferences dialog box.
(Projects)	Access the files associated with the selected project. The list of available projects is maintained in the Project Manager dialog box. (For information on how to add and delete projects from project lists, refer to *Maintaining Project Lists* on page 60.)

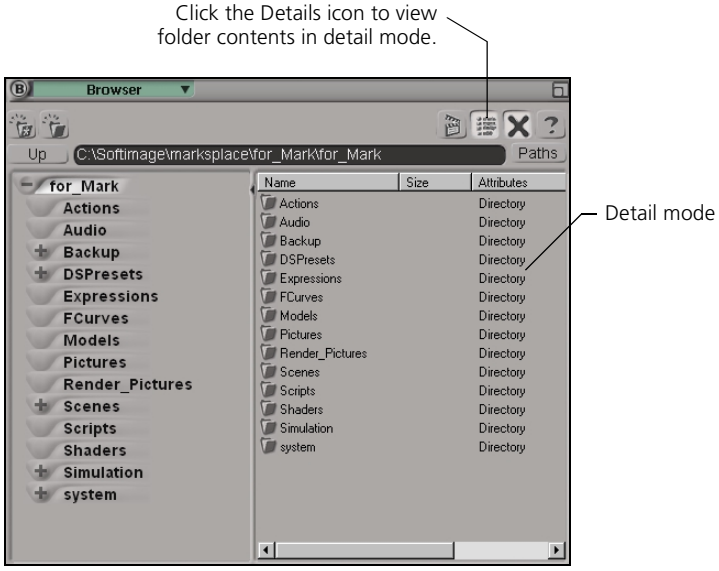
Viewing Folder Contents

The browser's contents view displays the files of a folder selected in the tree view. You can display these files either as thumbnails or in detail mode that includes file name, size, and comments.

To view folder files in thumbnail mode



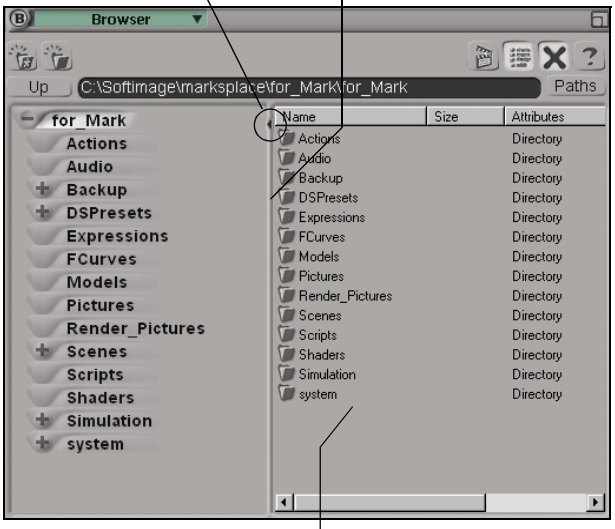
To view folder files in detail mode



To edit Contents view

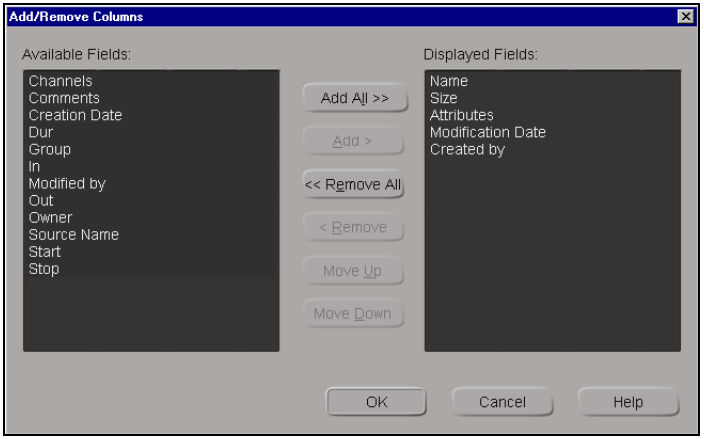
Click this arrow to show/ hide the Contents view.

Drag the split bar right or left to resize the browser's right and left panes.



Right-click anywhere in the Contents view...

...to add, move and rearrange columns in the Add/Remove Columns dialog box.

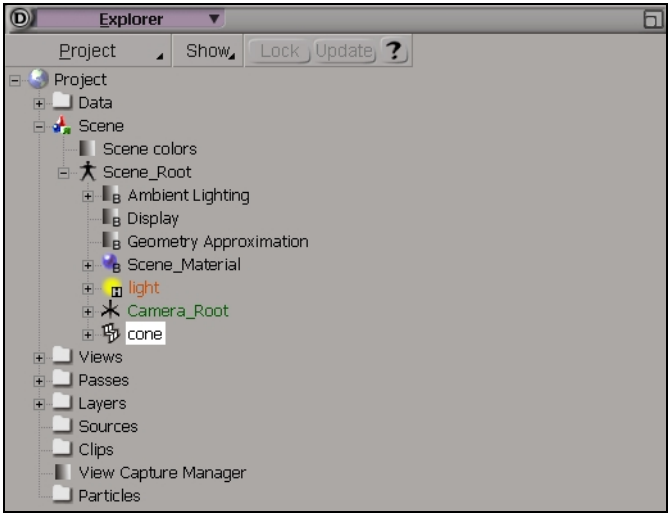


The Explorer

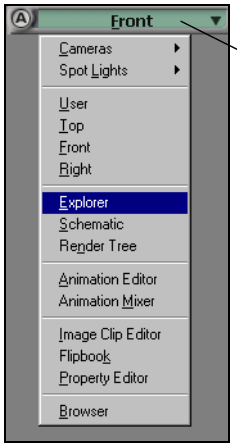
The explorer displays the contents of your scene in a hierarchical structure called a tree. This tree shows all objects in a scene as well as their properties, which are all displayed as a list of nodes that expand from the project root.

You use the explorer to perform a number of different tasks, such as:

- Displaying and navigating among the various scene elements in the current project. The types of elements displayed in the explorer are 3D objects, groups, clusters, shaders, texture nodes, properties, and so on.
- Renaming scene elements.
- Selecting scene elements and editing their properties.
- Creating parent/child relationships between objects.
- Assigning shaders, and textures to objects.
- Creating scene layers.
- Creating render passes.



Opening the Explorer



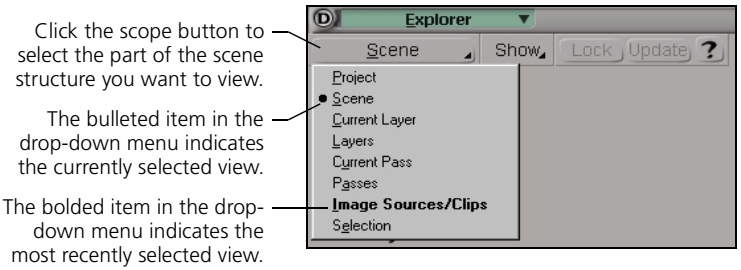
Click on the arrow or label to the left of any viewport and choose **Explorer** from the list. The explorer opens in that viewport.

or

Choose **View > Views > Explorer** from the main-menu bar. This opens the explorer in a floating window.

Displaying Basic Information

You can choose which part of the scene structure to display by selecting an item from a drop-down menu in the explorer window:



- **Project** displays all information associated with the project, from scene information to viewport properties, clips, sources, etc.
- **Scene** displays all scene element and property nodes from the scene root.
- **Current Layer** displays only those scene elements that are defined for the current layer. The current layer is the one whose name appears in the Current Layer list box on the main command panel.
- **Layers** displays all layers with their corresponding scene elements.
- **Current Pass** displays only those scene elements that are defined for the current pass.
- **Passes** displays the contents of the render passes you have defined for your scene.
- **Image Sources/Clips** displays only property nodes related to the image files and clips used in the scene.
- **Selection** displays only nodes related to the currently selected element(s).

Locking onto a Selected Object

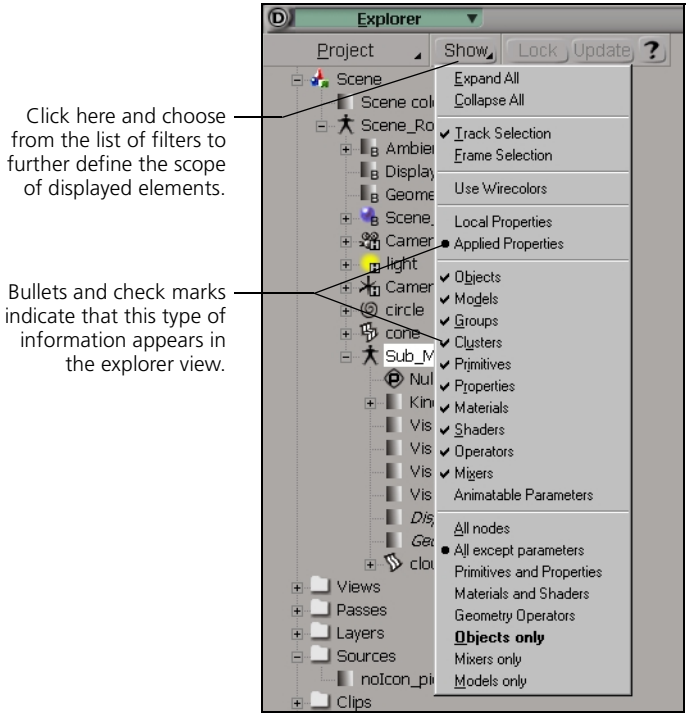
As mentioned above, the **Selection** option in the explorer's scope drop-down list displays information associated with the currently selected object.

If you click the **Lock** button with the **Selection** option active, the explorer continues to display the property nodes of the currently selected objects, even if you go on to select other objects in the viewports. Click **Lock** a second time to switch off the lock feature.

Click **Update** to reset the lock to the currently selected object.

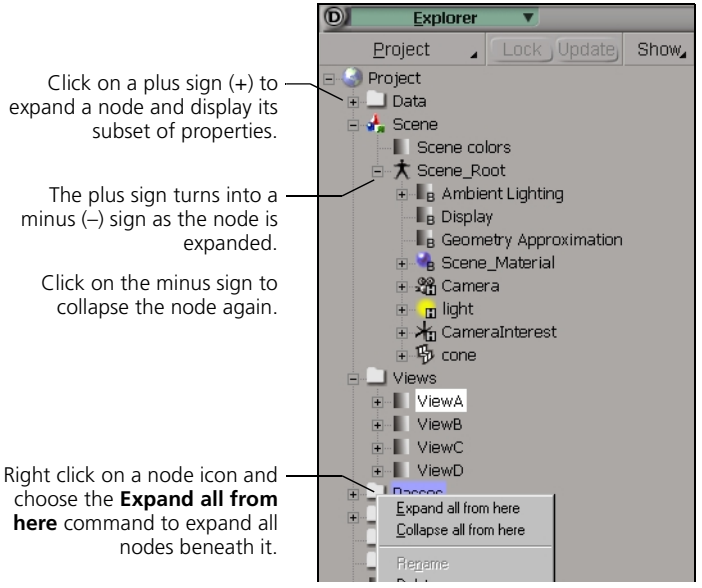
Filtering Information

The **Show** button at the top of the explorer window displays a set of filters that lets you further define how the scope of information is displayed. You can, for example, choose to display only objects and their materials in a scene, or all objects but no property nodes.



Viewing the Project Tree

The explorer displays the current project as a tree-like hierarchy of element nodes.



Show > **Use Wire Colors** lets you display the element nodes using their wireframe color.

- **Show** > **Expand All** and **Show** > **Collapse All** let you simultaneously expand or close all branches in the explorer tree.

If your mouse has a wheel, you can use it to scroll up and down and to the left and right of an explorer window. Simply click anywhere in the explorer before using the wheel.

You can also use keyboard shortcuts to navigate in your project tree. These arrow keys apply to the explorer only and cannot be used in the Schematic view.

Key	Does this
Up arrow	Moves the selection up one node.
Down arrow	Moves the selection down one node.
Right arrow	Expands the node or, if already expanded, selects the first child node.
Left arrow	Collapses the node or, if collapsed, selects the parent.
F	Frames the next selected node(s) in the centre of the explorer, starting with the first one.
Page Up	Scrolls the tree display upward.
Page Down	Scrolls the tree display downward.

Selecting Scene Elements from the Explorer

To select a scene element in the explorer, simply click on its node in the explorer window. Ctrl+click to multi-select individual nodes.

To select a range of elements of a similar type, click on the first element's node in the range, then Shift+click on the last element's node in the range. These two elements, plus all the elements of a similar type between them, are selected.

Only the visible nodes are selected; nodes that are hidden under a collapsed node remain unselected.

You can only select ranges of elements that are of the same type. If the second node picked is not the same as the first one, it is simply added to the selection and all nodes in between remain unselected.

Keeping Track of Selected Objects in the Explorer

If you have selected objects in any viewport view, their nodes appear as selected in the explorer. If their nodes are not visible in the explorer window, choose **Show** > **Find Next Selected Node**. The explorer will scroll up or down to display the first object node in the order of its selection. Each time you choose this option, the explorer will scroll up or down to display the next selected node. Once the end of the selection order is reached, the first object node is automatically selected.

Choose **Show** > **Track Selection** if you want to automatically scroll the explorer window so that the node of the first selected object is always visible.

Renaming Scene Elements

You can rename elements in your scene using the explorer. You can rename lights, cameras, geometric objects, render passes, and layers.

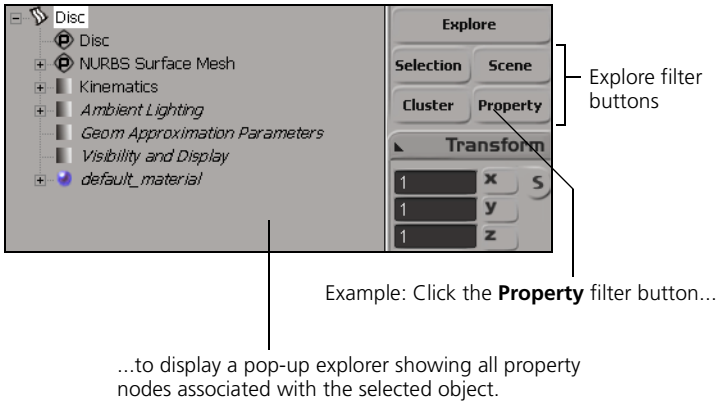
1. Right-click on any element node in the explorer.
2. Select **Rename** from the list and type a new name for the element.

Other Explorer Views

Other, smaller versions of the explorer—pop-up explorers—can be viewed elsewhere in the interface. They are used to view the properties of scene elements.

Selection Panel Explorer

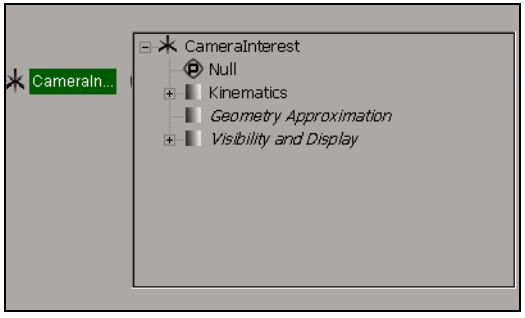
Explorer filter buttons in the Selection panel offer a shortcut by instantly displaying filtered information on specific aspects of currently selected objects.



The **Explore** button opens a pop-up menu of additional filters that allow you to specify the type of information you wish to obtain on the scene.

Schematic View Explorer

Explorer filter buttons in the Schematic view offer a shortcut by instantly displaying property information on a selected scene element.

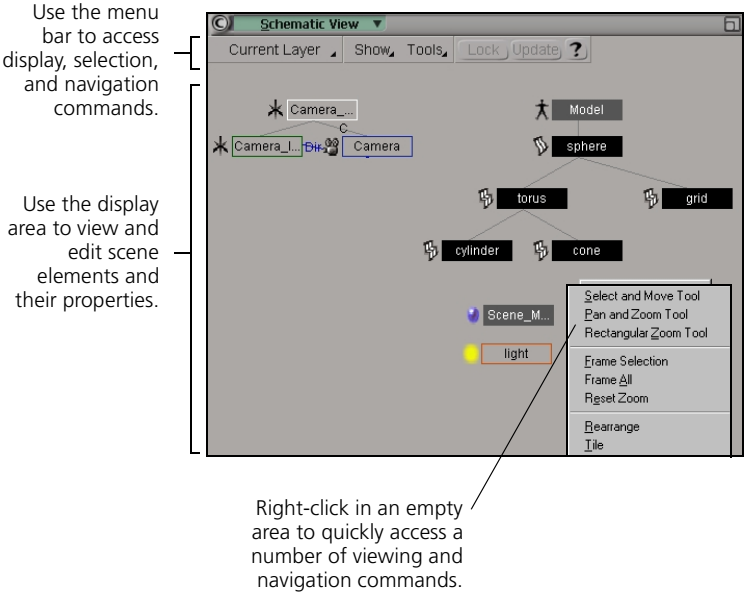


To display information on an element in the Schematic view, right-click on its node and choose **Explore** from the menu.

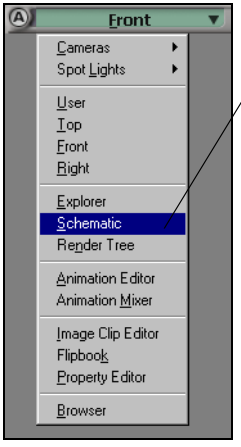
For more information on the Schematic view, see the following section.

The Schematic View

The Schematic view lets you analyze the way a scene is constructed by presenting the scene in a hierarchical structure. It includes graphical links that show the relationships between objects, as well as material and texture nodes to indicate how each object is defined.



Displaying a Schematic View



Click on the arrow or label to the left of any viewport and choose **Schematic** from the list. The explorer displays in that viewport.

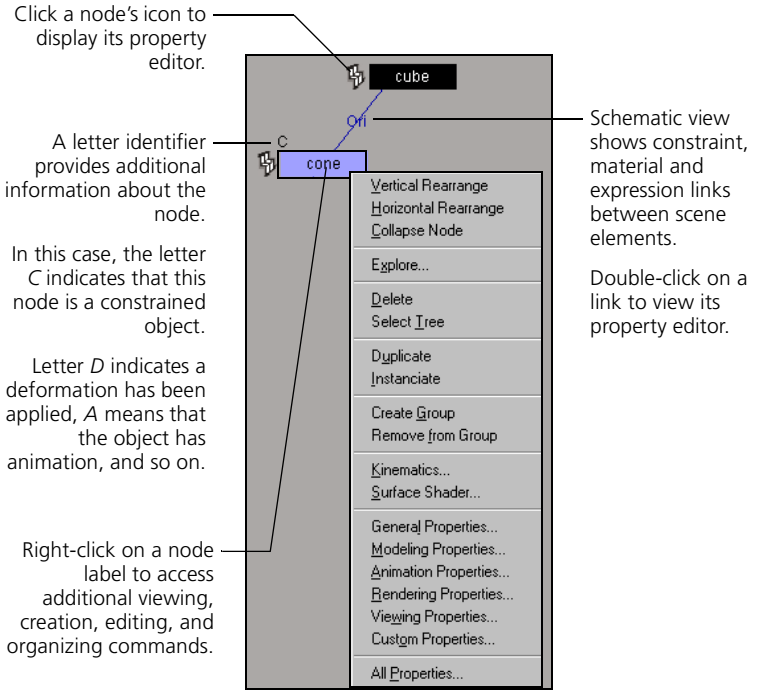
or

Choose **View > Views > Schematic View** from the main-menu bar. This displays a schematic view in a floating window.

Displaying Scene-Element Information

Scene elements in the Schematic view are displayed as graphical nodes. Relationships between elements are displayed as lines called *links*.

Nodes provide access to viewing, creating, editing, and organizing commands.



Displaying an Element's Property Nodes

You can display an element's set of property nodes in a pop-up explorer by right-clicking on the element and choosing **Explore**.

Displaying Links (Relationships) between Elements

Links between nodes appear by default. To toggle links on and off, choose **Show** > **Constraints**.

Double-clicking a link displays its property editor.

Constraint Links

Constraint links appear in blue and display an abbreviated label indicating the type of constraint or constraints in effect.

Material Links

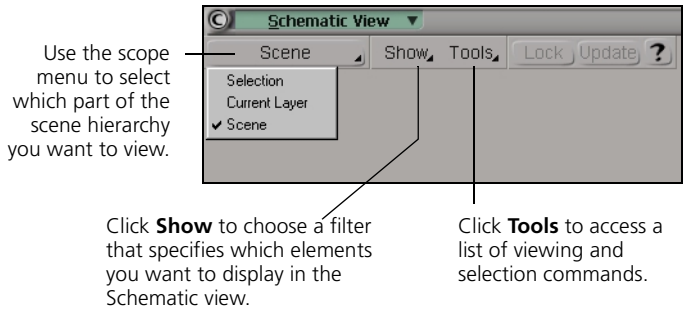
Material links appear in red and are identified by the label MAT.

Expression Links

Expression links appear in green and are identified by the label EXPR. Expression information is only displayed for local and global transformations.

Defining the View

You can choose which elements of the scene hierarchy to display by selecting an option from the Schematic view menus:



Defining the Scope of the Hierarchy View

You can define which elements to display in the scene hierarchy by choosing an option in the scope menu.

To define the scope of the hierarchy view

From the scope drop-down menu, choose:

- **Selection** to view only the nodes of selected elements.
- **Scene** to view the entire scene hierarchy.
- **Layer** to view only the nodes that belong to the currently displayed layer.

Using Filters Further Define the View

By default, the Schematic view displays all cameras, lights, and objects present in your scene. If needed, you can use the commands found in the Show menu to display additional information, such as material nodes associated with each scene element as well as links between elements.

To define the view using filters

In the Schematic view menu bar, choose **Show** and select the item to be shown or hidden from the Schematic view.

Locking the View onto Selected Objects

As mentioned above, the **Selection** option in the Schematic view's scope drop-down menu displays only the nodes of objects that are currently selected in the viewports.

If you click the **Lock** button with the **Selection** option active, the Schematic view continues to display the nodes of the currently selected objects, even if you go on to select other objects in the viewports. Click **Lock** a second time to switch off the lock feature.

Click **Update** to reset the lock to display the currently selected objects.

Navigating through the Hierarchy

The Schematic view's View menu provides access to a number of navigation commands:

- To pan across the schematic, choose **Tools** > **Pan and Zoom**, then click and drag the left mouse button.
- Zoom in by choosing **Tools** > **Pan and Zoom**, then hold down the middle mouse button.
- Zoom out by choosing **Tools** > **Pan and Zoom**, then hold down the right mouse button.
- To create a region in which to zoom, choose **Tools** > **Rectangular Zoom**, then drag diagonally to create the zoom region.

You can also scroll using the keyboard arrow shortcut keys. Use the left- and right-arrow keys to pan left and right, and the up- and down-arrow keys to up and down.

- To focus only on the selected objects in the hierarchy, choose **Tools** > **Frame Selection**.
- To view all objects in the hierarchy, choose **Tools** > **Frame All**.
- To revert back to a one-to-one viewing ratio, choose **Tools** > **Reset Zoom**.

You can also access the above commands by right-clicking in an empty area within the Schematic view.

Changing the Hierarchy Organization

You can change the organization of the hierarchy in the Schematic view to clear away unwanted clutter.

Fast Rearrange

Choose **Tools** > **Rearrange** to evenly redistribute hierarchies and their child nodes along a horizontal plane.

Rearranging Nodes Horizontally or Vertically

Right-click on a node in the schematic diagram and choose **Vertical rearrange** or **Horizontal rearrange** to redistribute parent nodes and their children horizontally or vertically.

Expanding and Collapsing Hierarchies

You can expand or collapse any hierarchy or sub-hierarchy in the schematic diagram to simplify the view.

Double click the parent node to alternate between expanding and collapsing the hierarchy beneath it. You can also click the "+" sign to the left of the node to expand the hierarchy.

How Elements Are Positioned

When you create a new scene, elements are added to the right of the previous element. The parent of a child in the hierarchy is always placed above its child.

Elements and hierarchies are organized by types, starting with cameras, geometric objects, and then all the others. All similar element types (such as lights, cameras, materials, etc.) are grouped together in the Schematic view to make your work easier. This, however, does not change their arrangement in the scene itself. For example, the camera and its interest are shown together in the Schematic view, but it has no relation to their position in any of the 3D views in your scene.

Selecting and Moving Elements in the Schematic View

Selecting an element in the Schematic view automatically selects the same element in all other views. Shift-click to select multiple nodes.

To move elements in the Schematic view, select and drag their nodes.

Chapter 3 Managing Projects & Scenes

What's a Project?

In SOFTIMAGE|XSI, you always work within the structure of a project. A project can contain an unlimited number of scenes.

Projects exist as folders and contain scene information in the form of scene-description files. Scene files are recognized by their **.scn** filename extension.

Scenes can be created from scratch in SOFTIMAGE|XSI, but you can also import existing SOFTIMAGE|3D scenes and use SOFTIMAGE|XSI's animation and rendering tools to edit them.

Creating and Opening Projects

When you start, the last project you worked in is opened and an empty scene is created. If no project exists or you are using SOFTIMAGE|XSI for the first time, you are prompted to specify the new project's name and file location. If you are already working in an existing project, you can create a new one at any time.

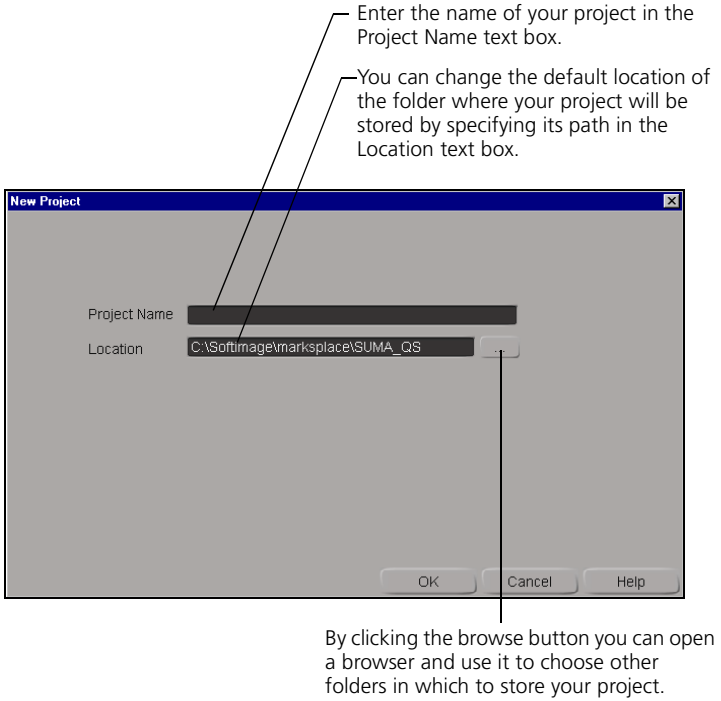
To create a new project

1. If you are already working in SOFTIMAGE|XSI and want to create a new project, choose **File** > **New Project** from the main-menu bar to display the New Project dialog box.

   or

   If you are starting SOFTIMAGE|XSI for the first time, the Project Browser dialog automatically appears. Click the **New Project** button to display the New Project dialog box.

2. In the **Project Name** text box, enter a unique name for your project.



3. In the Location text box, edit the path of the directory to which you would like the project to be saved.

Store your projects in any convenient location **outside** the Softimage directory structure. This way, when you remove the current version of the software to upgrade to a new version of SOFTIMAGE|XSI, you do not run the risk of deleting your work.

4. In the Project Browser, click **New Scene** to create a new scene for your project.

To opening an existing project

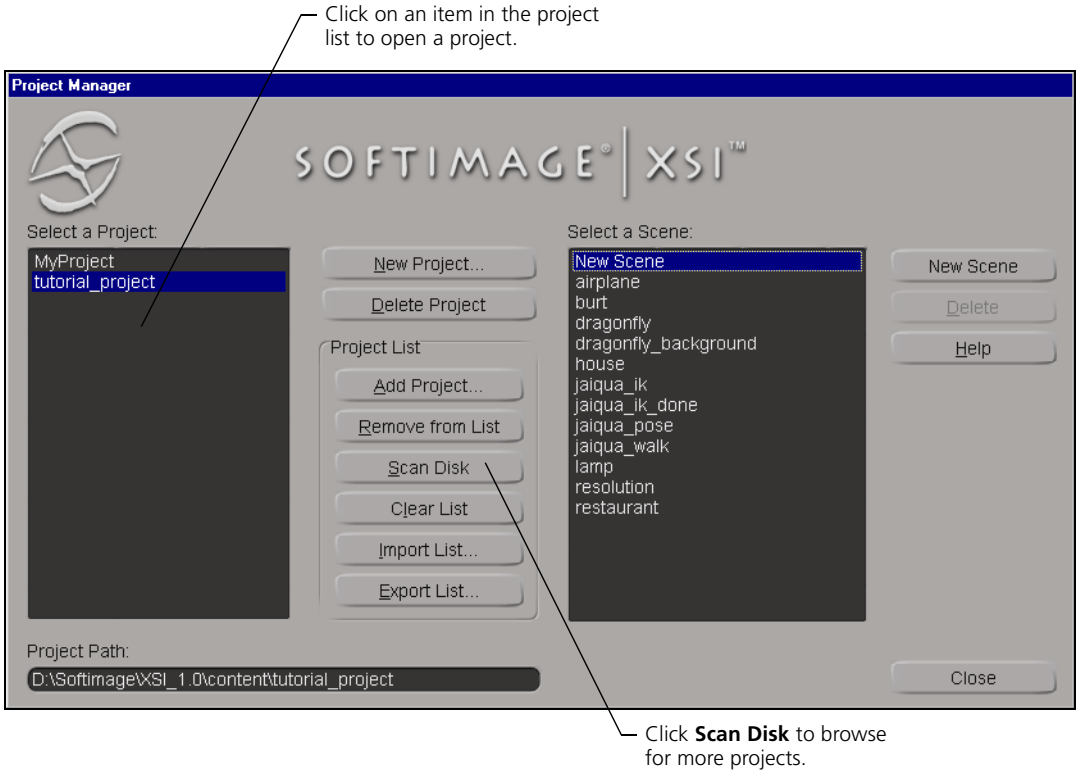
1. From the main-menu bar, choose **File** > **Project Manager.**

   A pre-defined set of directories is searched and all the projects stored in them are listed in the **Select a Project** pane of the Project Manager dialog box.

2. Click on a project name in the **Select a Project** list box.

   If you do not find the project you need in the displayed list, click the **Scan Disk** button to scan your files.

3. In the **Select a Scene** list box, double-click on a scene in the project you wish to work on.



Maintaining Project Lists

Complex productions can involve numerous individual projects. Some projects may be relevant to your sphere of work, and others not. Project lists in the Project Manager let you access the project you need with the click of a mouse.

To build and maintain project lists

1. Choose **File** > **Project Manager** from the main-menu bar to open the Project Manager.
2. In the **Project List** controls, click one of the following:
   - **Add Project** to add a project to the project list.
   - **Remove from List** to remove a project from the project list. This does not delete the project.
   - **Scan Disk** to open a browser that lets you search for more projects in selected directories.
   - **Clear List** removes all projects from the list. This is handy if you want to switch to another list.
   - **Import List** to open a browser that lets you search for a project list file containing a list of projects and their associated paths. These project names are added to any existing projects in the project list.
   - **Export List** to create a text file containing the path and name of each project currently displayed in the project list. You can then use **Import List** to access and display these projects.

Deleting Projects

To delete a project

1. From the main-menu bar, choose **File** > **Project Manager**.

   The Project Manager dialog box appears.
2. Click the name of the project to be deleted from the **Select a Project** pane.
3. Click the **Delete Project** button to delete the selected project.

You cannot delete a project you are currently working in. To delete your current project, you must close it, open another project, then delete the project in the Project Manager dialog box.

Creating and Opening Scenes

Creating Scenes

Each time you start SOFTIMAGE|XSI or create a new project, a new scene is automatically generated. However, you can create a new scene at any time. Because you cannot have more than one scene open at a time, you are prompted to save the current scene before you can create a new one.

To create a scene

Choose **File** > **New Scene** from the main-menu bar or press **Ctrl+n**. If you have modified the scene that is open, you are prompted to save the changes.

The new scene name appears as "Untitled" on the title bar.

Opening Scenes

You can only view one scene at a time. Before you open a different scene, you are prompted to save any changes made to the one open.

As the scene is loaded, SOFTIMAGE|XSI looks for all its files using the files' full path names.

If any file was not located using this method, SOFTIMAGE|XSI looks for the file name in the current project folder. If the file wasn't found here, the User Directory, then the Workgroup directory, and finally the Factory directory is searched. If the file was still not located, then you must display the explorer, open the Sources node, then manually input the file's correct path.

It is therefore important to make sure that all the files you use in a scene are stored in a folder that SOFTIMAGE|XSI can easily locate.

To open a scene from the browser

1. Choose the **File** > **Open** command from the main-menu bar.
2. In the browser, go to the location of the project file you want to load. You can use the **Up** button to locate folders at higher levels.
3. Select the scene file to be loaded—its name appears in the File Name text box—and click OK.

To open a scene from the Windows NT Explorer

You can take a scene displayed outside the application in the Windows NT Explorer and drag its icon to the SOFTIMAGE|XSI icon or a viewport to load it.

You cannot drag and drop scene icons from IRIX windows.

Locking Opened Scenes

When you have a scene open, it is "locked" so that anyone else who opens the file in the mean time is given a warning that the file is currently in use. That person can still open the scene to view its contents, but any attempt to save the scene under its current name may create problems.

Displaying Scene Information

The Scene Info dialog box provides you with a list of vital statistics, such as the number of poygon meshes, lights, camera, and objects used in the scene. This information can be helpful when evaluating a scene's complexity.

To get scene information

Choose **Edit** > **Info Scene** from the Edit panel (main command area).

The dialog's **Project** property page contains a list of external files used in the scene. In its **Resolved Path** text box, you can update a file's path if its location change since you first loaded the file into the scene. For more information, see *To update the path of a referenced file* on page 64.

Deleting Scenes

To delete a scene

1. From the main-menu bar, choose **File** > **Project Manager**.

   The Project Manager dialog box appears.

2. Click the name of the scene to be deleted from the **Select a Scene** pane.

3. Click the **Delete** button to delete the selected scene.

You cannot delete a scene you are currently working in. To delete your current scene, you must close it, open another scene, then delete the scene in the Project Manager dialog box.

Merging SOFTIMAGE|XSI Scenes

When you merge a SOFTIMAGE|XSI scene into the current scene, it is automatically made into a model. This ensures that the names of elements are preserved—because each model maintains its own namespace, there is no need to append element names with numbers to make unique names.

To merge a SOFTIMAGE|XSI scene into the current scene

1. From the main-menu bar, choose **File** > **Merge**. A browser opens.

2. Use the browser to locate and select a SOFTIMAGE|XSI scene, then click OK. The Model property editor opens.

3. Specify a **Name** for the model, as well as whether the model's storage is **Internal** or **External**.

If you don't want the elements of the merged scene to be a separate model, you can "unparent" the model's children using either the **Cut** button on the Edit panel or by dragging and dropping their nodes in the explorer, then deleting the model node.

For more information on models, refer to *Models* on page 137

Saving and Renaming Scenes

All scenes are saved as a single file with an **.scn** extension. The scene contains any models, 3D objects, lights, cameras, animation, rendering options, effects (including shaders, textures, and shader assignments), and render passes that you have applied.

To save a scene

1. Choose **File** > **Save** from the main-menu bar.

   If the scene has already been named, any changes are saved to the same file name. If the scene does not have a name, the Save Scene browser appears.

2. Enter a unique name for your scene in the File Name text box. You can use the Up icon to locate the folder where you want to save your scene.

3. Click OK.

Referenced Files in Saved Scenes

There may be times when some of the information on your scene originates, not from the current project folder, but from an external source. When you save a scene, the path information that lets SOFTIMAGE|XSI locate and refer to these external files is saved as well.

Saving Referenced Files to the Current Project

There are times, however, when you may want to take all the referenced external files and copy them to the current project. This is particularly useful when you send project files to other users who cannot access the referenced files themselves.

To save all referenced files to the current project

1. Choose **File** > **Save As** from the main-menu bar to display the Save As dialog box.

2. Choose **Copy Files Under Project** and click OK.

To update the path of a referenced file

1. Choose **Edit** > **Info Scene** from the Edit menu to Scene Info dialog box.

2. Click the **Project** tab.

3. In the **External Files** list, review the list of external files. In the **Status** column, files whose path are no longer valid display as "Invalid".

4. Select the file in need of updating and in the **Resolved Path** text box type the file's correct path.

Saving Scenes Under a Different Name

In some cases, you may want to create a copy of a scene and save it under a different name. This can be helpful when you want to test a more complex rendering effect before committing it to the actual scene.

To save a scene under a different name

1. Choose **File** > **Save As** from the main-menu bar to display the Save As dialog box.
2. Type another name for the scene in the File Name browser and click OK.

   The new scene name is displayed on the title bar. The original scene remains unchanged in its most recently saved form.

Importing Scenes

When you import a scene into SOFTIMAGE|XSI, it must be in DSC format (the format used in SOFTIMAGE|3D that contains information about a scene's hierarchies, camera, lights, animation, and geometry), HRC format (the format that contains information on models created in SOFTIMAGE|3D), or IGES format.

An imported scene contains timeline information that defines its length according to a start and end frame. By default, a scene's entire frame range is inserted at frame 1 of the timeline.

To import a scene created in SOFTIMAGE|3D

1. From the main-menu bar, choose **File** > **Import SI3D Scene/Model** to display a browser.
2. Navigate to the folder that contains the scene you want to import. You can use the Up icon to locate the folder where you have saved your scenes.
3. Select the scene file to be imported. Its name appears in the **File name** text box.
   - To select a SOFTIMAGE|3D scene file (**.dsc**), make sure that the **Files Types** text box is set to **Scene Files** (***.dsc**).
   - To select SOFTIMAGE|3D model file (**.hrc**), make sure that the **Files Types** text box is set to **Model Files** (***.hrc**).
4. Click **Open**. The imported scene's contents are added to the current scene.

If you have difficulty importing a scene, you can help pinpoint the problem by generating a file of commands logged during the loading process. To do so, choose **File** > **User Preferences**, click the **Scripting/Logging** tab, and toggle **Log Commands to File**.

Alternatively, you can view the import command history in the Script Editor. To display the Script Editor, simply click the "!" icon to the left of the **Playback** button at the bottom of the interface. To make sure all the information is displayed in the editor window, choose **File** >**User Preferences**, click the **Scripting/Logging** tab and increase the value in the **Command Log Size** box.

Importing and Exporting IGES Files

To import files created in IGES format

1. From the main-menu bar, choose **File** > **Import IGES File** to display a browser.
2. Navigate to the folder that contains the file you want to import. You can use the **Up** button to locate folders at higher levels.
3. Select the file to be imported. Its name appears in the **File name** text box.

   To select an IGES file (**.iges** or **.igs**), make sure that the **Files Types** text box is set to **Iges Files** (***.iges;.igs**).
4. Click **Open**. The imported scene's contents are added as a model in the current scene.

Exporting Files in IGES Format

You can take objects in your scene and export them in IGES format to a directory of your choosing.

Implicit objects cannot be exported in IGES format.

To export scene objects in IGES format

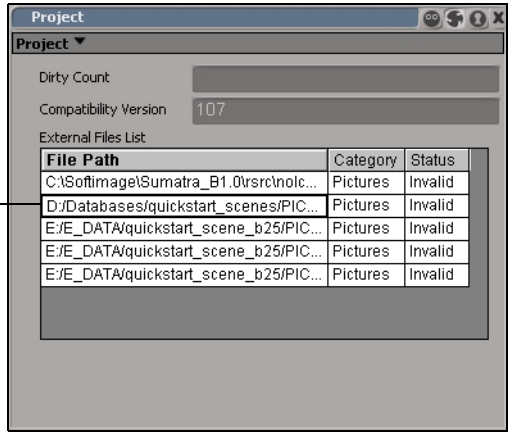
1. Select the object to be exported.
2. From the main-menu bar, choose **File** > **Export IGES File** to display a browser.
3. Navigate to the folder to which you want to export the file. You can use the **Up** button to locate folders at higher levels in the directory.
4. Select the object to be imported. Its name appears in the **File name** text box and click OK.

Correcting External File Paths for Scenes

Each time you open or import a scene, SOFTIMAGE|XSI attempts to locate the external files (e.g., picture files, audio files, etc.) associated with the scene.

To locate missing files in imported scenes

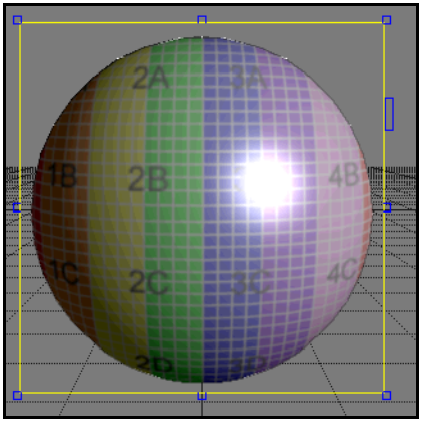
1. Open the project node's property editor to display a reference list that includes the paths of all external files referenced by the project.
2. If a file cannot be found, the word **Invalid** appears in the **Status** column.
3. Modify the external file's path to resolve the error.



Enter the external file's correct path to resolve the path error.

Setting Your Own Default Image

If no texture is associated with a shader, the application applies a default image to the rendered object, as shown below.



If you prefer, you can replace the default image with one of your own choosing.

To change the default image

1. In the **User Preferences** dialog box, click the **Rendering** tab.
2. In the **Default Picture** text box, enter the path and file name of your new default image file.

   or

   Click the (…) button next to the text box and select the default image file from its folder in the browser.

Locking Render Licenses

You can specify the number of mental ray rendering licenses to be used by your computer. Specifying a low number can free up licenses for other computers, if needed.

To lock one or more render licenses

1. In the User Preferences dialog box, click the **Rendering** tab to display the SOFTIMAGE|XSI property page.
2. In the **Lock Licenses** box, specify the number of rendering licenses you need to reserve for your machine's use.

Recovering Your Work

In most cases, you can easily recover your work if your system crashes. You also have the option of loading earlier saved versions of your scene if you want to backtrack and rework some of your content.

Automatically Recovering Scene Files

Usually after the crash occurs, a dialog box appears indicating that your scene file has been successfully saved to a backup folder and that you can continue to use the scene after re-starting the application.

If the crash save process cannot recover your file, SOFTIMAGE|XSI will look for an autosaved version of your scene file. You can activate the autosave feature in the General page of the User Preferences dialog box, under **Autosave**. By default, files are saved every 30 minutes.

To recover a scene file after a system crash

1. Re-start after the crash. A dialog box appear, asking you if you want to try to recover the scene. Click Yes.
2. If the scene is recoverable, another untitled scene that would have been retrieved from the crash save or autosave process will appear showing your latest work.
3. Save the scene under a new name and restart the application.

Setting Autosave Preferences

An option in the User Preferences dialog box lets you activate the autosave feature and determine how frequently autosaves are made.

Autosave does not overwrite current files, so be sure to save your work each time you exit.

To set the frequency of autosave

1. In the main-menu bar, choose **File** > **User Preferences** and click the **General** tab to display the general desktop settings.
2. Select the **Save Automatically** option to activate the autosave function. By default, autosave is set to save every 30 minutes, although you can change this time span in the **Minutes** text box.

Make sure you have enough disk space so that your backup files are created successfully.

Restoring Recently Saved Scenes

Each time you save your scene, its file is automatically saved in a backup folder. This means you can go back and load an earlier version of your scene if needed. This can also come in handy if your scene file could not be automatically recovered.

The default number of backed-up scenes is four, although you can change this number (to a maximum of 100) in the User Preferences dialog box.

To set backup preferences

Choose **File** > **User Preferences** > **Interaction** and set the number of backups in the **Number of Scene Backups** text box.

To reload a scene file from its backup directory

1. Choose **File** > **Open** and navigate to your backup folder. By default, this folder is called Backup and is located in your project directory.

   This folder contains all your saved scene files. These files are labeled **(Myscene)_1**, **(Myscene)_2**, **(Myscene)_3** etc., with **(Myscene)_1** being the most recently saved scene.

2. Select the required backup version and click OK to load the recovered scene.

Integrating 3D Models into SOFTIMAGE|XSI

Scene development is rarely a linear process. At times, you may need to re-import models into SOFTIMAGE|XSI that have since been modified in a 3D application.

Ordinarily, this would mean that, when you import the more recent version of the model, any textures, materials or other properties that you added to the current model would be lost. You would then have to re-apply all the lost properties to the newly imported model.

You can avoid this duplication of effort by gathering all the property sets you applied to your model, such as materials and textures, and bundling them into a single preset, called a *property override*. Once you delete the old model and re-import the new one, you can apply this preset to the model and effectively override all the properties and characteristics that existed on the original.

Maintaining the Model's Fundamental Structure

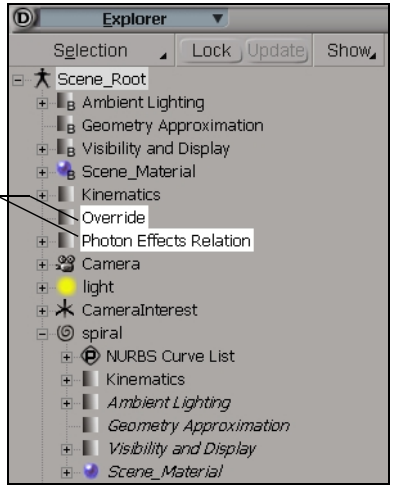
Re-imported models can receive any property that existed in the previous model, as long as the new model's hierarchy respects that of the original. For example, the texture on a robot's hand will not be reapplied if one or more joints connecting the hand to the body have been removed.

Creating Property Overrides

Property overrides are one or more property sets saved from the current model and applied to a replacement model bearing the same name as the original one.

To create an override

1. Open the explorer, choose **Scene_Root**.
2. Choose **Get** > **Property** > **Override**.
3. In the explorer, choose **Show** > **Local Properties** to view local branch properties.
4. Multiple-select the Override node as well as the property sets (property nodes) you want to apply to the replacement model.



5. Choose **Get** > **Property** > **Store In Override** > (**selected properties**).


Save icon

6. Open the Override property editor and click its Save icon (left) to save its properties.
7. Delete the original model and import its modified replacement. Remember, the replacement model MUST have the same name as the original model so that the override can recognize it.
8. Open the browser and locate the override.
9. Drag the override over the scene root label.

   The newly imported model takes on the characteristics of the deleted model. Property sets that have been applied to the new model are stored in the override node as a child of the Scene_Root node.

Exporting Scenes to MI2 Files

Instead of rendering a scene in SOFTIMAGE|XSI, you can export it to an MI2 file (mental images version 2.1) that can be read by mental ray rendering software. The final output of all of this is an MI2 file. You can manually edit the file for greater control over the final render, run scripts on it to perform extra processes before the file is rendered, or use it with mental ray on any supported platform, including non–Windows NT or non-IRIX systems.

You can reload the exported MI2 file into SOFTIMAGE|XSI to be rendered, but it is loaded in as read-only information and you cannot modify any of its rendering properties.

To export a project file to an MI2 file

1. From the Render toolbar, choose **Tools** > **Export to mi2**.
2. Enter a path and file name for the MI output file in the Export Filename text box.
3. Specify the export parameters in the dialog box controls and click **Export MI2** to create the output file. You may have to wait for the output file to be generated (go get a coffee!). You can then manually edit and render this file with mental ray from the command line.

You can start mental ray from a shell or command-prompt window.

Chapter 4 Viewing Your Work

As you work, you have a wide variety of ways in which to view your scene. Some viewing modes are more appropriate for certain tasks than others. Here are the most common ways to view the contents of your scene:

- View your scene's geometry in the viewports. The many ways in which you can view scene geometry in the viewports is the focus of this chapter.
- View the scene as a hierarchy of elements in the explorer (see *The Explorer* on page 43).
- View scene elements and the relationships between one another in the Schematic view (see *The Schematic View* on page 49).
- View scene contents in layers to focus on specific aspects of your scene or selectively render certain elements while excluding others (see *Layers* on page 144).

Viewing Animation

The *Animating* guide describes additional viewing techniques related specifically to animation. It shows you how to:

- View your scene's animation using playback controls (see *The Playback Panel* in Chapter 1 of the *Animating* guide).
- View your scene's animation as a flipbook in a viewport (see *Loading a Flipbook* in Chapter 1 of the *Animating* guide).

Previewing Rendered Scenes

As you reach the final stage of your project, you will likely want to preview your work prior to rendering. You can use the following preview techniques:

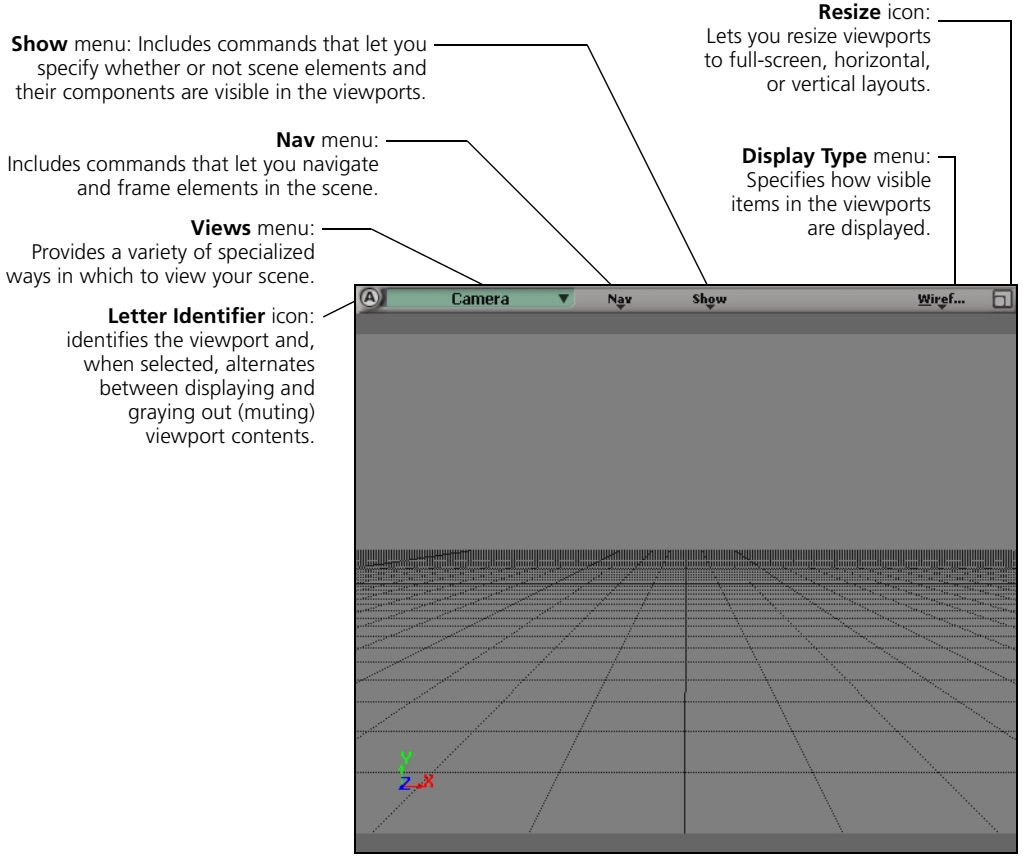
- Preview a single frame of animation in its final rendered format (refer to *Previewing Animation as a Flipbook* in Chapter 1 of the *Animating* guide).
- Preview your scene interactively using the **render region**, which lets you render a scene directly in a viewport without saving it to a file. It uses a subset of the global-rendering options so that your scene is a fairly good approximation of the final look (see *Previewing Rendered Scenes* on page 77).

Viewports

The viewing area of the interface contains windows, called *viewports*, in which you can view and work on your scene. Each viewport can display different types of views. You can display up to four viewports.

Viewports can have different geometry views (Top, Front, Right, and User) and display types (Shade, Wireframe, Texture, Hidden Line, etc.). They can also display other views such as the animation editor, browser, and the explorer. Any viewport can display any type of view.

A viewport's menu bar contains a set of controls that allow you to perform various functions such as changing views and display types, setting scene visibility, and resizing the viewport.



Identifying Viewports by Letter

A letter (**A**, **B**, **C**, and **D**) appears in the upper-left corner of each viewport to help simplify the way in which they are referenced. **A** refers to the upper-left viewport, **B** is the upper-right, **C** is the lower-left, and **D** is the lower-right viewport.

Muting and Soloing Viewports

When a viewport is muted, it is prevented from displaying its contents. Muting its neighbors helps a viewport speed up its refresh rate.

To mute a viewport, select its reference letter

- To view just one viewport (solo mode), left-click on its letter. Left-click a second time to redisplay all viewports.
- To mute a viewport, middle-click on its letter. Middle-click a second time to redisplay the viewport.
- Right click to display a menu displaying mute and solo commands.

Changing Viewport Color

By default, viewport grid, viewport background and perspective background appear in shades of gray. You can modify them to display any color you wish.

Viewport color is saved with your scene. Each time you create a new scene, the viewports revert to their default set of colors.

You cannot undo changes to your grid and background color once you modify them.

To change viewport color

1. In the explorer, choose **Project** in the scope menu.
2. Open the Scene colors node, located directly beneath the Scene node to display the Scene Colors property editor. Click the **View** tab.

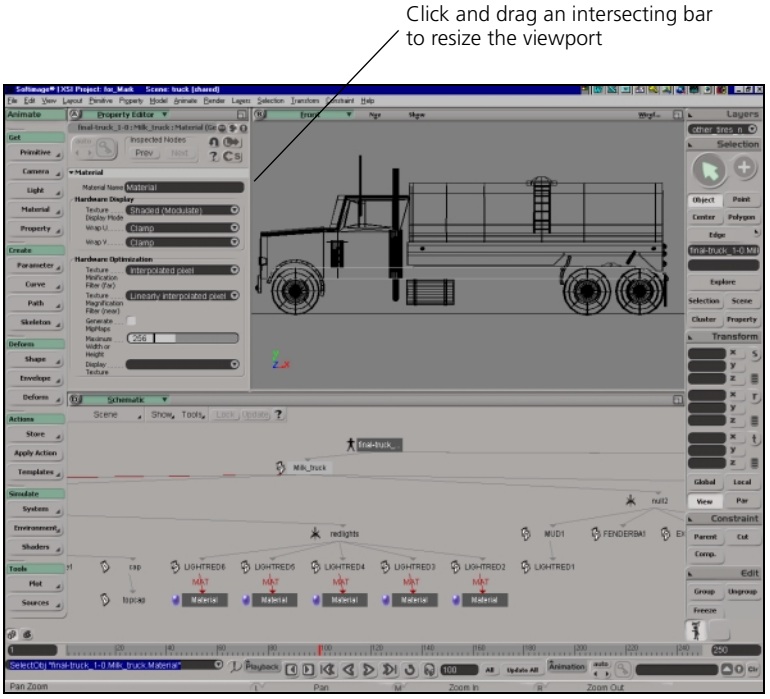
   To change viewport background color, click on the **Background** color chip; to change the color of the viewport border region, click on the **Viewport** color chip; to change the color of the viewport grid, click on the **Grid** color chip.Resizing the Viewports

There are two basic ways in which to resize viewports: interactively to any size, or using the sizing icon for setting specific sizes.

Resizing to Any Size

You can interactively resize the viewports by following these steps:

1. Position the mouse pointer at an intersection of the viewports. You can either place the pointer where the four viewports meet or where two viewports meet. The pointer becomes an arrow indicating in which direction you can resize the viewports.
2. Drag the viewports to whatever size you want. They change size in relation to each other.
3. To reset the viewports back to equal sizes, middle-click at the viewport intersection.



Resizing Viewports Using Set Sizes

The sizing icon in the upper-right corner of each viewport resizes it so that the viewport fills the entire screen (full view), a horizontal half-view (landscape), or vertical half-view (portrait).

You can easily toggle between these views and reduce the viewport to its original size (quarter-view) at any time.

- To toggle between a quarter-view (the four viewports) and full view, click the sizing icon.
- To toggle between a quarter-view and a landscape view (horizontal), middle-click the sizing icon.
- To toggle between a quarter-view and a portrait view (vertical), **Ctrl**+**middle-click** the sizing icon.

You cannot toggle directly between the landscape and portrait views.

You can also right-click the sizing icon to display a pop-up menu that includes the following commands:

- **Maximize** displays the viewport in full view.
- **Horizontal** displays the viewport in landscape view.
- **Vertical** displays the viewport in portrait view.
- **Reset Size** displays all viewports in quarter-view.
- **Reset All** resets all viewports to their default set of views (i.e., Camera, Right, Front, Top).

You can instantly maximize any viewport by placing the cursor in the viewport to be maximized and pressing the F12 key.

Customizing the Viewport Grid

When you translate objects, such as lights and cameras, you can do so relative to the displayed grid; that is, the translation is restricted to take place within the plane of the displayed grid. This can be done in any of the four viewports displaying any 3D or camera view.

You can modify grid characteristics using the Camera Visibility Settings property editor.

To modify a viewport grid

1. Click **Show** > **Visibility Options** from any viewport menu bar to display the Camera Visibility Settings property editor.
2. Click the **Grid** tab to display the Grid page, then click **Grid** to activate grid display in the viewport.

   The grid helps you position and orient the objects in your scene, but the grid itself is not rendered.
3. Set the **Grid Display** and **Field Guide** options as required. You can:
   - Choose a display grid with respect to the plane in which it is displayed.
   - Set the displayed grid size (Extent), which controls the actual cell size of the grid displayed in the viewport. This can only be set for finite grids.
   - Set the spacing between grid lines along the U and V axes. Measurements are in SOFTIMAGE units.
   - Activate or deactivate snapping by grid amounts.
   - Set the snapping grid size using the UV Step options.
   - Set safe areas for titles and actions.

   For a description of each of these options, click the **Help** button (left) in the Camera Visibility property editor.

Click the Help icon and refer to the Online Help for a description of each option on the property page.

You can set grid options for all viewports simultaneously in the Visibility Options of All Cameras property editor by choosing **View** > **Visibility Options... (all Cameras)** from the main-menu bar.

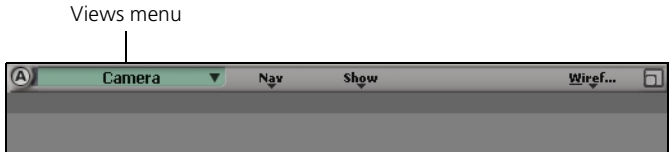
Viewport Views

There are many ways to view and edit your scene in the viewports.

The default set of views that appear when you start the application are, clockwise from the top-right corner, Camera, Right, Front, and Top. If you change any of these views during your work session and want to quickly return to the default view set, right-click in any top-right corner of a viewport and select **Reset All** from the drop-down menu. **Reset Size** in this same menu lets you reset all the viewports to their original size.

To display or change a view

Click the arrow or the name of the current view displayed on any viewport menu bar to open the views menu.



- You can toggle between the previous and current view by middle-clicking on the arrow or the name of the view on the viewport menu bar.

  or

Choose **View** > **Views** from the menu bar. This displays a menu listing view types that can be displayed independently in a floating window.

The Audio Output Monitor, which is accessed from **View** > **Views** in the menu bar, is not functional.

Camera Views

Camera views let you display your scene in a viewport from the point of view of a particular camera. You can also choose to display the viewpoint of the camera associated to the current render pass. The Render Pass view is also a camera view: it shows the viewpoint of the particular camera associated to the current render pass. Only a camera associated to a render pass is used in a final render.

Selecting a camera or the Render Pass item from a viewport's Cameras menu switches the viewpoint to that of a "real" camera in your scene. All other views such as User, Top, Front, and Right are not associated to an actual camera.

Viewpoints

Viewpoints show you the geometry of objects in a scene. They can be viewed in the render region, but they cannot be rendered like camera views.

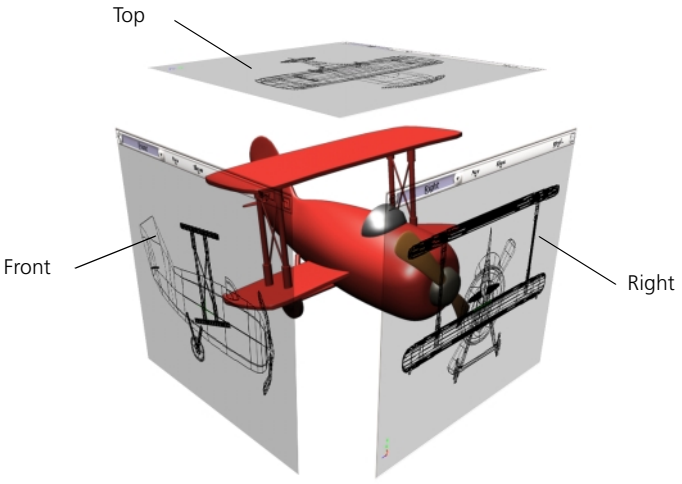
You can also choose different display types to change the visual appearance of the objects seen in viewpoints (see *Setting Object Display* on page 94).

Top, Front, and Right Views

The Top, Front, and Right views are parallel projection views, called such because an object's projection lines do not converge in these views. Because of this, the distance between an object and the camera has no influence on the scale of the object. If one object is close to the camera, and another is farther away, both appear to be the same size.

The Top, Front, and Right views are orthographic, which orient the camera so it is perpendicular (orthogonal) to specific planes:

- The Top view faces the XZ plane.
- The Front view faces the XY plane.
- The Right view faces the YZ plane.



User

The User view is a user-defined viewpoint that shows objects in a scene from a virtual camera's point of view. This view can be either perspective or orthographic.

- In the perspective view, objects appear to converge toward a central vanishing point, and objects closer to the camera appear larger than those farther away.
- In the orthographic view, objects remain in parallel projection with the view being perpendicular (orthogonal) to the XY plane in camera space.

The User point of view can be placed at any position and at any angle within the global 3D coordinate system. You can orbit, dolly, zoom, and pan in this view.

Other Views

Spotlight

Spotlight views let you select from a list of spotlights available in the scene. Selecting a spotlight from this list switches the point of view in the active viewport relative to the chosen spotlight. The point of view is set according to the direction of the light cone defined for the chosen spotlight.

To view the light cone of selected and unselected spotlights choose **Show** > **Visibility Options** > **Attributes**, then toggle **Cones** for selected and unselected objects.

Data

The following views can be displayed in a viewport:

- The Explorer view provides a hierarchical tree view of your project that allows you to manipulate all its elements. You can display the explorer in a floating window as well. For more information on viewing your project, see *The Explorer* on page 43.
- The Browser view lets you search for and load files. You can display the browser in a floating window as well. For information on using the browser, see *The Browser* on page 37.
- The Schematic view lets you view the elements in your scene in their hierarchical structure, as well as their relations with other elements. The Schematic view also lets you instantly display the property editors associated with each element. For more information, see *The Schematic View* on page 49.
- The Render Tree view lets you connect shaders, textures, or functions (called nodes) to one another to build a complex effect. Each node exposes a set of properties that can be dynamically assigned by connecting the output of one property to the input of another. For more information on how to use the render tree, refer to the *Rendering* guide.

Animation Editor

The Animation Editor option displays the animation editor in the viewport. You can use this editor to fine-tune existing animation and introduce new animation based on the scene's render properties. For more information on how to use the animation editor, see *The Animation Editor* on page 63 of the *Animating* guide.

Animation Mixer

The Animation Mixer option lets you work with actions as clips on tracks. Actions are animation segments (like walking or jumping) that you define once and apply as many times as you like. You can use the animation mixer to mix actions together, or create compound actions that contain other actions. For more information on how to use the animation mixer, see *Actions* in Chapter 11 of the *Animating* guide.

Image Clip Editor

The Image Clip Editor lets you edit a clip directly from a viewport. This view is especially useful for editing large textures or textures composed of several images.

Flipbook

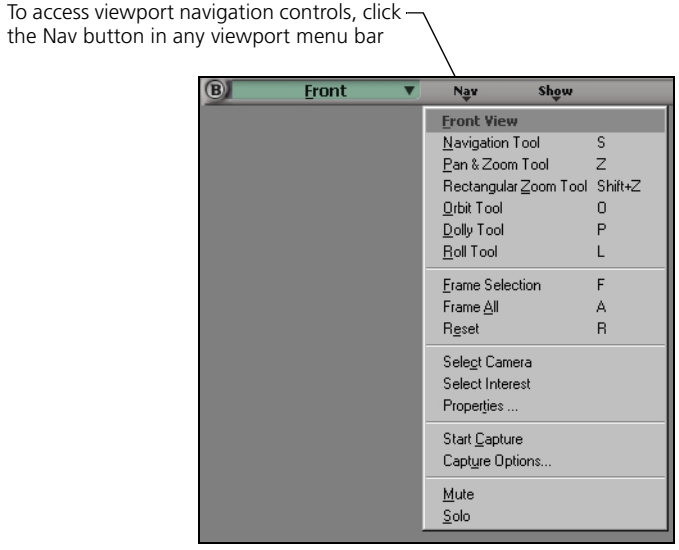
Flipbook lets you view a series of cached images that have been saved as PIC files. For more information on how to view flipbooks, see *Creating a Flipbook from Cached Images* in Chapter 1 of the *Animating* guide.

Property Editor

The Property Editor view anchors all displayed property editors in a viewport. Property editors are used to define the characteristics of elements in your scene. For more information on editing scene elements using property editors, refer to *Scene Elements and Their Properties* on page 105.

Navigating in 3D Views

A set of viewport-navigation controls and supra keys let you change the way in which you view your scene. You can use these controls and keys to zoom in and out of a scene, frame objects within a viewport, and orbit, track, and dolly.



A corresponding supra key exists for each interactive tool if you wish to access a particular type of navigation control from the keyboard. Supra keys are identified in the sections that follow.

Framing Objects in Viewports

Framing commands let you instantly zoom in to or out from selected objects in viewport geometry views. You can frame objects in two ways: framing selected objects or framing all objects in one or all viewports.

- To frame all visible objects in a viewport, position the mouse cursor over the viewport and press the **a** key.
- To frame all visible objects in all viewports, press **Shift+a** or choose **View** > **Frame All** (**All Views**) from the main-menu bar.
- To frame all selected objects in a single viewport, choose **Frame Selection** from the **Nav** drop-down menu.
- To frame all selected objects in all viewports, press **Shift+f** or choose **View** > **Frame All** (**All views**) from the main-menu bar

Zooming

You can zoom in and out of your scene or pan in all 3D views using the zoom controls.

Pan Zooming

Choose **Nav** > **Pan and Zoom Tool**, or press the **z** supra key to activate the zoom tool. Drag the left mouse button to pan, hold down the middle button to zoom in, and hold down the right button to zoom out. Press **z** again to deactivate zoom mode.

Zooming in or out changes the view angle setting of the camera—the same as when you zoom with a real camera.

Rectangular Zooming

You can also define a zoom area by choosing **Nav** > **Rectangular Zoom Tool** or by pressing the **Shift+z** supra key. This lets you define a rectangular area that becomes the new magnification factor.

To do a rectangular zoom, click in a window, hold down the left mouse button to zoom in, or the right mouse button to zoom out, and then drag the mouse diagonally to define the rectangle.

Orbiting

To pivot the camera, spotlight, or user viewpoint around its point of interest, choose **Nav** > **Orbit Tool**, or press the **o** supra key while in User view (you cannot orbit in the Front, Top and Right views). This lets you study your scene's overall "look" in any angle in any view. The left mouse button allows full movement, the middle mouse button allows movement in vertical, and the right mouse button allows movement in horizontal.

Dollying, and Rolling

Rolling is only possible in the User view; you can dolly in all camera views.

- To dolly toward the camera interest, choose **Nav** > **Dolly Tool** or press the **p** supra key. Follow the mouse instructions on the mouse line at the bottom of the window to dolly at different speeds.
- To roll the camera about its Z axis, choose **Nav** > **Roll Tool**, or press the **l** (L) supra key. Follow the mouse instructions on the mouse line at the bottom of the window to roll at different speeds.

Combination Mode

Choose **Nav** > **Navigation Tool**, or press the **s** supra key in a viewport perspective view to activate this combination of controls:

- Right-mouse button to pan
- Middle-mouse button to dolly
- Left-mouse button to orbit

With the Shift key held down:

- Right-mouse button to orbit horizontally/vertically
- Middle-mouse button to roll
- Left-mouse button to track (pan) horizontally/vertically

Resetting Coordinates

Choose **Nav** > **Reset** to reset the camera and other viewport views so that their global point of origin (X = 0, Y = 0, Z = 0) is in the center of the viewport.

Setting Object Visibility

Each camera in the scene has a set of visibility controls that determine whether objects, object attributes, or components are visible in the viewports and in the rendered image. An additional set of visibility controls exist for each object in its Visibility property editor.

Setting Visibility for All Objects in All 3D Views

You can set object visibility for all cameras, which in turn determine how objects are viewed and rendered in all viewports.

Besides showing and hiding objects and components, you can also choose to display or hide object attributes such as centers, points, and normals. Object attributes are useful visual cues when performing certain rendering tasks. For example, to verify in Wireframe whether a surface is correctly oriented for rendering purposes, try displaying the object's normals.

Showing or hiding object attributes affects the scene's refresh rate but does not affect the object information that is saved with a scene. Even if object attributes are displayed on objects in the viewport, they do not appear in the final render.

To set object visibility for all cameras in all viewports

Choose **View** > **Visibility Options...** (**All Cameras**) to display the Visibility Options of All Cameras property editor. From this editor you can choose the types of objects, object components, and object attributes to display or hide in all 3D views. In addition, you can set a number of grid display options.

See Online Help for a description of the options available in this property editor.

Setting Visibility for All Objects per Viewport

You can set object visibility on a per camera basis, which determines how objects are viewed and rendered in individual viewports.

To set object visibility per camera in a specific viewport

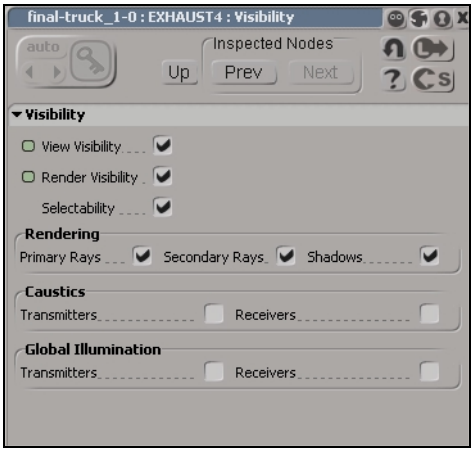
1. If required, change views in the viewport and select the camera to which to apply the visibility settings.
2. Click **Show** in the viewport's menu bar.
3. From the drop-down menu, choose:
   - **Name** to display the name of selected objects.
   - **Distance to Output Camera** to display the distance, in SOFTIMAGE units, between the camera used to render the scene and the selected object.
   - **Points**, **Knots**, **Lines**, etc. to display object components.
   - **Relations** to display expressions, constraints, and other links between objects.
   - **Property Maps** to display a graphical representation of weight-mapped properties on objects.
4. Choose **Show** > **Visibility Options** to display the Visibility Options property editor of the current view. From this editor you can choose additional types of objects, object components, and object attributes to display or hide in the viewport. In addition, you can set a number of grid display options.

Setting Visibility per Object

Each object in the scene has its own set of visibility controls that determine whether the objects appear in the viewports and rendered image.

In some cases, for example, you may wish to temporarily exclude objects from a render but retain them in the scene. This can come in handy when you are working with complex objects and want to reduce lengthy refresh times.

You can also choose to render only an object's shadow and reflection. Each geometric object's property editor has a variety of options that can be selected individually or combined with others to achieve the desired effect.



To set object visibility for a selected object

1. In the viewport, select the object whose visibility options you want to set.
2. Click the **Property** button in the Selection panel to display the selected object's property nodes.
3. Click the **Visibility** node to open the object's Visibility property page.
4. Select or deselect visibility, primary rays, secondary rays, and shadows for the object as required:

   View Visibility—for showing the selected object in the viewports. This is useful if you want to clean the area around the render region in a viewport. Turning off View Visibility will not affect the rendered image.

   Render Visibility—for showing the object in the rendered image. This overrides all other settings in the Rendering controls.

   Selectability—for choosing whether or not the object can be selected in the viewport.

   Rendering—controls which attributes of the selected object are visible when the scene is rendered:

- **Primary Rays**—Makes the object visible for viewing and rendering. A reflection of the object, however, may not be visible unless Secondary Rays are also selected.
- **Secondary Ray**—shows an object's reflection in a mirrored surface, also shows its refraction and transparency.
- **Shadows**—shows an object's shadow. A shadow is cast if the lights and rendering options are properly set. For detailed instructions on creating shadows, refer to *Creating Shadows* in Chapter 5 of the *Shaders, Lights & Cameras* guide.
- **Caustic** and **Global Illumination** options—for including photons for creating caustic effects or global illumination when rendering. For more information on caustics and global illumination, refer to *Chapter 6: Using Global Illumination and Caustics* of the *Shaders, Lights & Cameras* guide.

Example

Let's say you created two characters (a "real" mouse and a computer mouse). You can set the object's visibility parameters so that only the real mouse's shadow is visible to the camera while no shadow is visible for the computer mouse. To do this, select only the **Secondary rays** and **Shadows** options for the real mouse and only the **Primary rays** option for the computer mouse.

Visibility Shortcuts

You can quickly hide and unhide selected objects in the viewports using menu commands or the **h** shortcut key. This is a quick way to set the viewport visibility of a selected element.

Hiding objects eliminates their visibility both in the viewports and in the rendered output.

To hide and redisplay objects in the viewports

1. In any viewport, select the object you want to temporarily hide, and press the **h** key

   or

   from the main-menu bar, choose **View** > **Hide/Unhide Selection**.

   The selected object is hidden in all geometric views but remains visible in the explorer. Hidden elements are not rendered in the render region nor do they appear in the final output.

2. Pressing the **h** key or selecting **View** > **Hide/Unhide Selection** a second time will redisplay the hidden object. If you selected another element or modified the scene after you hid the object, pressing the **h** key or selecting **View** > **Hide/Unhide Selection** again will have no effect. In this situation, the hidden object can only be redisplayed by selecting its node in the explorer and pressing the **h** key or selecting **View** > **Hide/Unhide Selection** again.

To unhide all hidden objects in the viewports

From the main-menu bar, choose **View** > **Unhide All Objects**.

Setting Object Display

Once you have decided which objects, attributes, and components are visible in your viewports, you can choose how to display them. Some display types provide less detail but are less computationally intensive than others—and this speeds up your screen refresh.

You can define display types for:

- all cameras—how objects are displayed in all viewports
- individual cameras—how all objects in a *specific* viewport are displayed.
- individual objects.

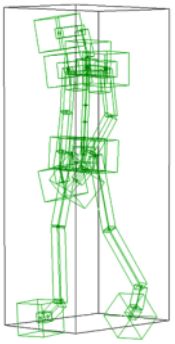
Display Types

The following paragraphs summarize the different ways you can display objects can be displayed. By default, all objects are displayed in Wireframe mode.

Wireframe

Shows the geometric object made up of its edges, drawn as lines resembling a model made of wire. This image displays all edges or contour lines without removing invisible or hidden parts or filling surfaces. This is the default display type in the viewport.

You can alter this view slightly by changing the color of the wireframe of a specified object. See *Setting Object Wireframe Color* on page 102.



Bounding Box

Reduces all scene objects to simple cubes. This speeds up the redrawing of the scene, since fewer details are calculated in the screen refresh.

Hidden Line Removal

Shows only the edges of objects that are facing the camera. Lines that are usually hidden from view by the surface in front of them are not displayed as they are in a "see through" wireframe.

Constant

This type ignores the orientation of surface normals and instead considers them to be pointing directly toward an infinite light source. All the object's surface triangles are considered to have the same orientation and be the same distance from the light.

This results in an object that appears to have no shading at all. This is particularly useful when you want to work in textures, because there are no attributes to interfere with the texture's definitions. This mode is also fast and is useful for previewing rotoscoped images.

Shaded

Provides an OGL hardware shaded view of your scene that closely approximates its realistic "look" but does not show shadows, reflections, or transparency. Wireframes of geometric objects are superimposed over their shaded surfaces showing you most display options, such as lines, points, tags, and centers. This makes it easy to manipulate points, lines, tagged points, etc. You can also view light (point and spot) and camera icons.

Textured

Displays textures and special material attributes such as non-raytraced reflection maps. This display type supports tiled textures, textures modified with the color palette, alpha-transparency, and the results of texture compositing. Geometric objects' wireframes are superimposed on their textured surfaces, showing you most display options such as lines, points, tags, and centers. This makes it easy to manipulate points, lines, tagged points, and so on.

This option slows down any changes of view in a viewport.

To quickly set display types for individual objects

1. In the bottom left of the interface, click the Palette icon.

   The palette icon displays a toolbar containing group of display type icons where:

   B = bounding box
   C = constant
   D = reset the viewport to its default display type
   W = wireframe
   S = shaded
   T = textured
   H = hidden line

2. Click an icon, then pick the objects to receive the associated display type. Right click to end the picking session.

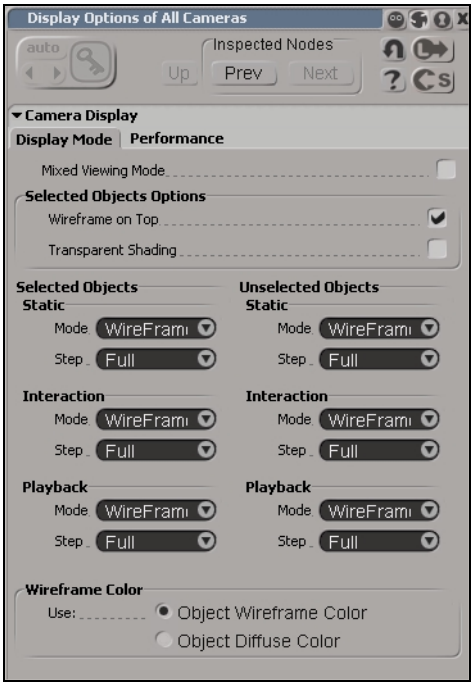
Setting Display Options for All Cameras

You can specify how cameras' view objects in the Display Options for all cameras' property editor. See Online Help for a full description of each option.

To specify how all cameras view objects in 3D views

1. In the main-menu bar, choose **View** > **Display Options...** (**All Cameras**).

   The Display Options for all Cameras property editor appears.



2. In the **Display Mode** controls, choose:

   - **Selected Objects Options** > **Wireframe on Top** to draw wireframes over shaded objects.
   - **Selected Objects Options** > **Transparent Shading** to draw a dotted wireframe outline on the back faces of objects.
   - **Wireframe Color** > **Use Object Wireframe Color** to cause all objects to display their custom wireframe color set in their Display property editors.
   - **Wireframe Color** > **Use Object Diffuse Color** to display all objects in diffuse color.

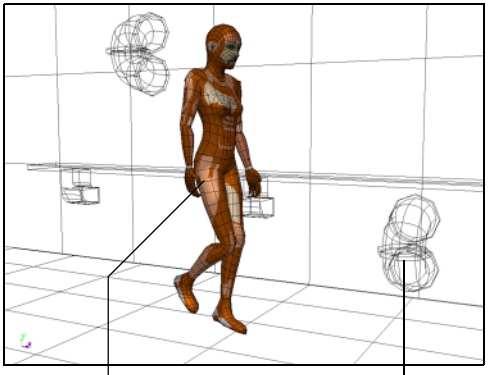
3. Choose a display type from the drop-down list next to each option in the Display Mode controls:

   • **Static - Selected Objects**—static (non-animated) selected objects within a scene.

- **Interaction - Selected Objects**—animated, selected objects within a scene, as well as objects that are being transformed or viewed from a different perspective as a result of camera manipulation.
- **Playback - Selected Objects**—all selected objects during playback of a scene.
- **Static - Unselected Objects**—static (non-animated) objects that are not selected within a scene.
- **Interaction - Unselected Objects**—interacted, unselected objects.
- **Playback - Unselected Objects**—all unselected objects during playback of a scene.

4. To take all object display types for into account, choose **Mixed Viewing Mode**. Settings for individual objects are set in their own Display property editors (see *Setting Display Options per Object* on page 101).

Example: Mixed-viewing mode



Interactive Selected: Textured

A selected static object displays in textured mode for full detail.

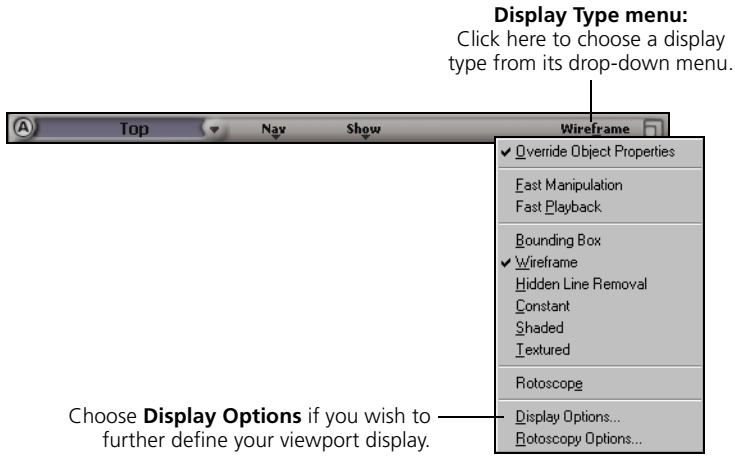
Static Unselected: Wireframe

Selected foreground objects display in wireframe to minimize image processing time.

Setting Display Options per Viewport

Display options selected from a viewport's Display Type drop-down menu determine how all objects in that particular viewport are displayed.

To set the display type for a specific viewport

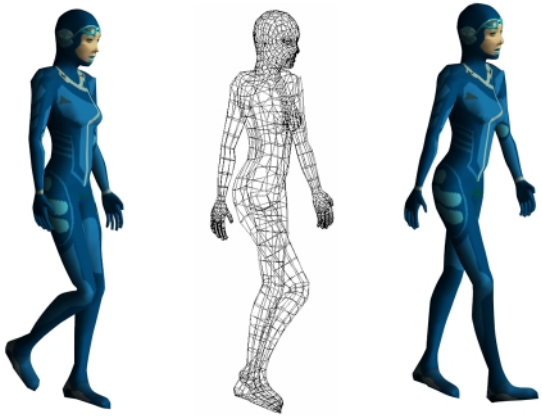


You can toggle between the previous and current display type by middle-clicking the Display Type name displayed in the viewport menu bar.

Improving Scene Display Performance

When you have Constant, Shaded, Textured, or Rotoscope display type selected, selecting **Fast Manipulation** from the Display Type drop-down menu speeds up the redrawing of your scene by switching to wireframe when you interact with an object.

Example: Fast manipulation



Static Select: Textured
Static object displays in textured mode for full detail.

Interactive Select: Wireframe
Object, when selected and interacted with, displays in wireframe to minimize image-processing time during interaction.

Static Select: Textured
Static object, after interaction, reverts back to a textured display type.

Similarly, selecting **Fast Playback** from the Display Type drop-down menu speeds up the redrawing of your scene by temporarily hiding the grid and drawing objects in wireframe during playback of an animated scene.

Overriding Individual Display Types

The **Override Object Properties** option in the Display Type drop-down menu, when selected, overrides any display type settings that may have been set for individual objects in the viewport.

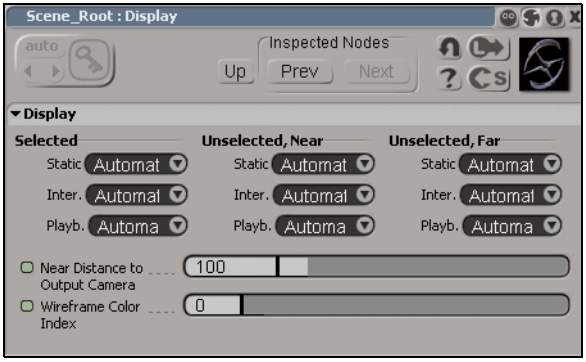
Setting Display Options per Object

You can control how individual objects are displayed in a viewport.

Giving an object or objects different display characteristics is particularly useful for heavy, animated scenes. For example, if you want to tweak a static object within a scene that has a complex, animated character, you could set the character in wireframe display mode while adjusting the lighting of your static object in shaded mode.

Setting display options for individual objects

1. Select an object in the viewport for which you want to set display options.
2. Click the **Property** button in the Selection panel and click the object's Display node to open its Display property editor.



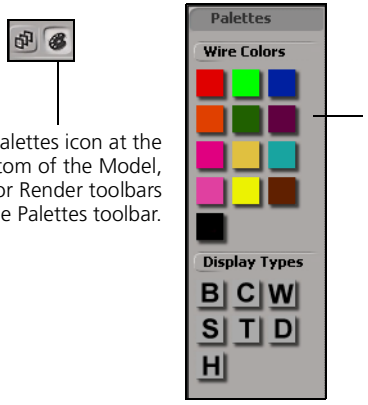
Here, not only can you select different display types, you can apply them, depending on the state of the object (selected, unselected near, unselected far). The distance at which an object is considered to be near is set in the **Near Distance to Output Camera** controls. For complete details on each control in this property page, refer to Online Help.

Setting Object Wireframe Color

You can change the color of objects displayed in wireframe in the viewports. Assigning wireframe colors to objects can be useful if your scene is complex and you want to make certain elements stand out for quick identification.

You can easily set object wireframe color using the Palettes toolbar. This toolbar contains a default set of 12 colors. The black chip on the bottom row is the object's default color.

To change an object's wireframe color using the Palettes toolbar



Click the Palettes icon at the bottom of the Model, Animate, or Render toolbars to display the Palettes toolbar.

Click on a color chip, then pick one or more wireframe objects. Click on the black chip to assign the object its default wireframe color. Right-click to end the picking session.

You can place objects in wireframe display type by clicking on the W icon, then picking the objects in a viewport.

You can also set wireframe colors using the **Wireframe Color** control in the object's Display property editor. Its slider can be set to any one of over a thousand colors. Each number represents one color.

Chapter 5 Working with Scene Elements

Scene Elements and Their Properties

Every element in your scene is distinguished by its own set of characteristics, which in turn are defined by a series of parameter values. When you work on elements in your scene, you will almost always use property editors to define these values.

Property editors are interactive. As you make changes to individual parameters, the scene is updated to reflect these changes. If you select a different scene element or change frames, the property editor updates to show the values for the modified parameter(s) at the current frame.

You can define how a property editor behaves as you work and select elements in SOFTIMAGE|XSI: it can follow your selection and display the properties of any selected element of the same type; it can recycle and display the properties of the selected element; or it can be locked so that it always displays the properties of the same element. For more information, see *Locking, Recycling, and Focusing Property Editors* on page 109.

Properties Viewed in Hierarchies

Each object, shader or operator in your scene is represented as a node in the scene hierarchy, which is viewed in an explorer. Each node is made up of default properties that are displayed and edited in a property editor. Depending on its position in the hierarchy, a node's property editor can also contain tabs that let you display property sets of its child nodes. For more information, see *Anatomy of a Property Editor* on page 108.

Displaying Property Editors

There is no one way to access a scene element's property editor. How you access an editor depends on your current work context.

- Select an object in a viewport and...
  - Press **Enter** to display a property editor of the object.
  - Press **Alt**+ **Enter** to display a property editor of the object, as well as those that belong to all its child nodes.
- In the explorer, choose the element node, then use the **Selection** filter button to display its properties. Double-click on the property set you wish to view.
- Use Shortcut keys to open a property editor. **Ctrl**+**k**, for example, will open the Local Transformation property editor of a selected object. You can also create your own shortcut keys to open a property editor of your choice. For more information on creating shortcut keys, refer to *Creating Keyboard Shortcuts* on page 215.

Certain property editors are displayed in other ways:

- Move back and forth through previously opened property editors by hitting the Page Up and Page Down keyboard keys or choosing the **Up**, **Prev** and **Next** buttons in a property editor's Inspected Nodes controls. (Right-click on the **Prev** or **Next** button to display a selectable list of most recently viewed nodes.)
- Right-click in a render region border and choose **Properties** from the pop-up menu. This displays the View Rendering Options property editor for the region.
- Choose **Show** > **Visibility Options** or **Display Options** from the display types menu bar in a viewport. The property editor that display let you specify how elements and the viewports themselves are viewed.

Automatic Display of Property Editors

In many cases after you choose a command, a property editor displays automatically. When you choose **Get** > **Material** and select a material shader preset from the drop-down list, for example, the material shader's property page automatically displays.

To prevent property editors from displaying automatically after choosing a command

Choose **File** > **User Preferences** to open the User Preferences dialog. In the Interaction property page, click **Auto inspect**.

This option is particularly useful if you do not want property pages to appear during script execution.

or

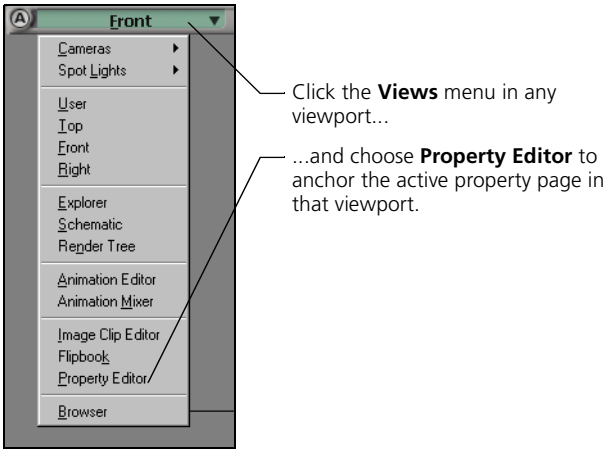
Press Ctrl when choosing a command.

Property editors that display as floating windows can quickly be closed (as can all other floating window views) by positioning the mouse over the window and pressing the left single quote (‘) key. For most keyboards, this key is found directly below the escape (Esc) key.

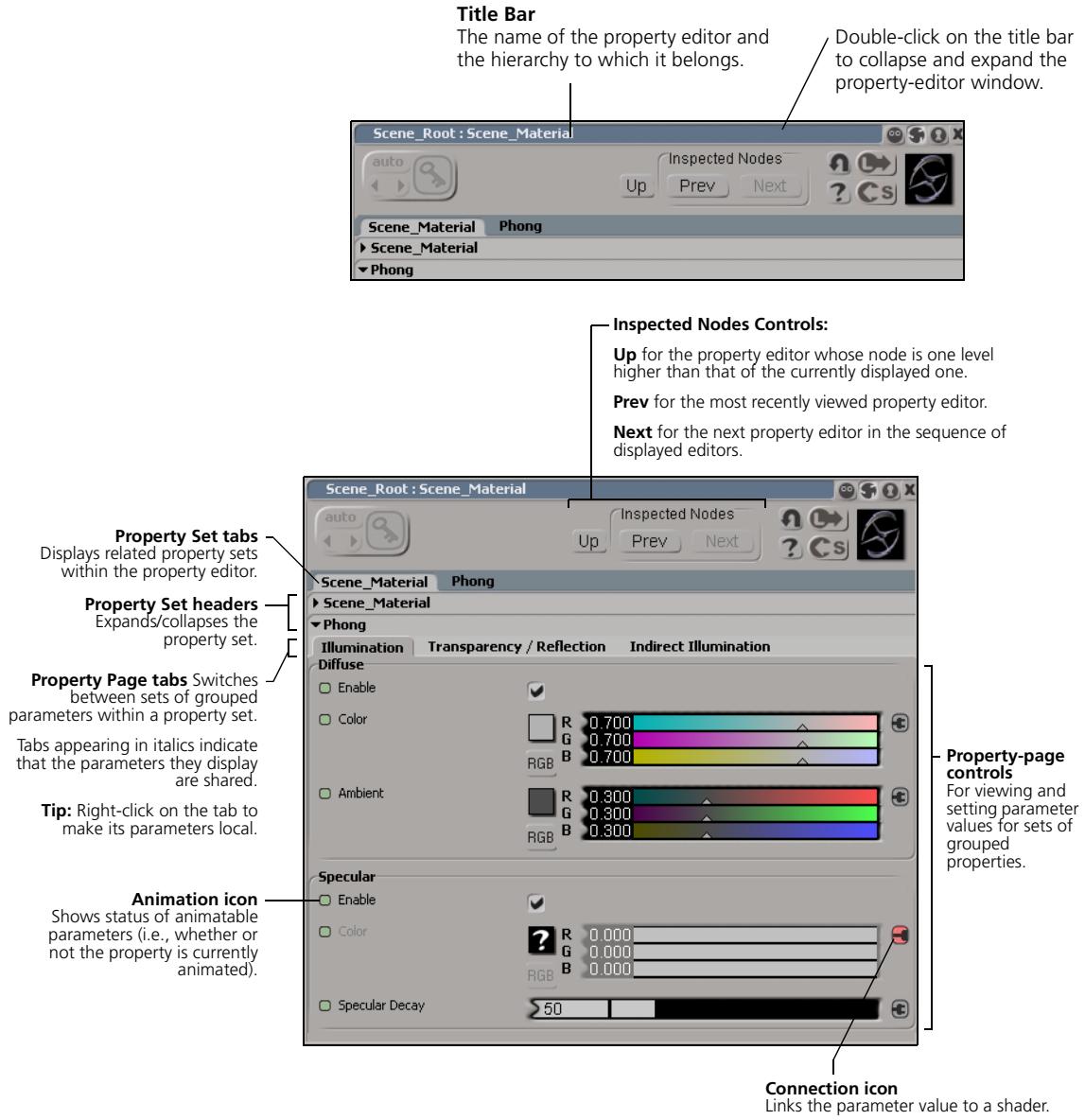
If you like, you can change this shortcut by choosing **File** > **Keyboard Mapping** and mapping the command to another key in the Keyboard Mapping dialog box.

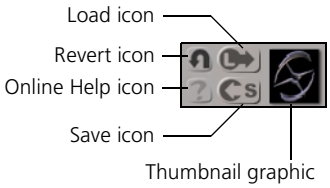
Docking the Property Editor in a Viewport

By default, property editors display as floating windows. Alternatively, you can anchor them in a viewport:



Anatomy of a Property Editor



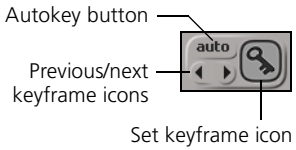


The area in the top right corner (below the title bar) contains:

- A **thumbnail** graphic of the property-editor settings. The information in this graphic can be processed from the render region. For example, the thumbnail can show the current texture on a selected object.

When multiple objects are selected, the thumbnail is blank. For objects such as lights and cameras, a default bitmap is displayed.

- **Load and Save** icons for opening and saving presets (see *Presets* on page 117).
- A **Revert** icon that lets you reset all controls to the values they had when you first opened the property editor.
- **Online Help** icon that gives you information on all properties on the property page.



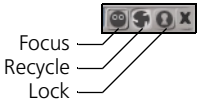
The top left corner of the property editor, just below the title bar, contains the following controls related to setting keyframes for properties:

- An **autokey** button that, when selected, automatically creates a keyframe each time you move to a new frame and set specific properties from the property editor.
- Two **arrow** icons that let you move to the previous keyframe and next keyframe.
- A **set keyframe** icon that lets you set keyframes for the current settings.

For more information on keyframes, see *Chapter 2: Animating with Keys* in the *Animating* guide.

Not all of the options described above are available in every property editor.

Locking, Recycling, and Focusing Property Editors



The three icons at the right end of the property-editor title bar control how the property editor is updated as you work and select elements:

- The **Focus** icon updates the current property editor to show the properties of the element you select, but only if the element is the same type as was previously selected (hence "focusing" on the type of element). For example, if you display the property editor of a spotlight and then select a point light, that property editor is updated to display the properties of the point light because spotlights and point lights are both the same type of element. However, if you then select a geometric object, a blank page is shown, because lights and geometric objects are different types of elements.

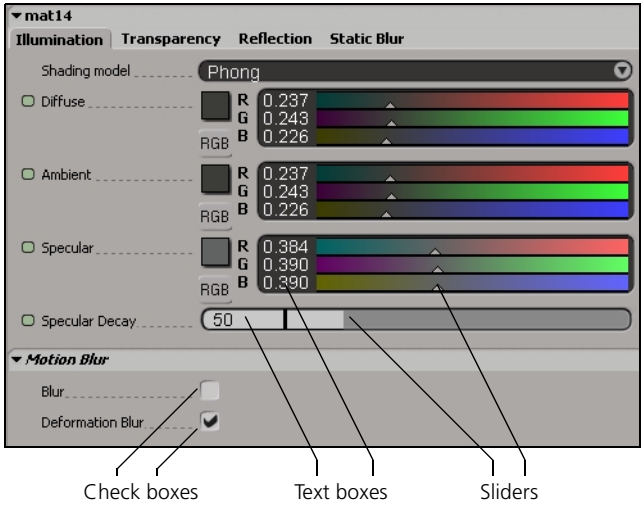
- The **Recycle** icon keeps that property editor open and refreshes it to display the properties of any element you select. This is a convenient way of keeping only one editor open at all times. For example, if you select a cube, the property editor shows the cube's properties; then, if you select a light, the same property editor is updated to show that light's properties.
- The **Lock** icon keeps the property editor open for the selected element. You can open other property editors for other elements, but a locked property editor remains open until you close or recycle it.

Editing Parameter Values

Object, shader, or operator parameter values can be edited directly by means of text boxes, check boxes, and sliders located throughout the interface.

You change the value of a parameter in a text box by typing in either text or numerical values. If a slider is associated with the text box, you can also drag the sliders to raise or lower the numerical values. Some sliders and text boxes, such as those that control color-channel values, are grouped together.

If the text box contains more characters than it can display, it automatically scrolls in response to the position of the text cursor.



Entering Parameter Values in a Text Box

There are many ways to enter information in a text box. You can enter information the conventional way by using your keyboard to enter values. For numeric input, you can also use gestural movements and key combinations to increase or decrease parameter values.

To enter information by typing

Move the mouse pointer inside the text box and left-click. The pointer changes shape to become a text cursor (flashing vertical bar). Any existing text is selected and will be overwritten as soon as you begin typing.

If you middle-click in the text box, no text is selected. You can now:

- Use the left- and right-arrow keys on the keyboard to move the cursor to the appropriate place in the text box.
- Drag the cursor over the part of the contents you want to replace. The selected characters are highlighted. Now type in the text you want.
- Press **Shift+left-** *or* **right-arrow** key to select characters that are to the left or right of the cursor. You can delete or replace these characters.
- Double-click to select a word so that you can replace it completely with the text that you type.
- **Right-click** to open a pop-up menu and choose the command to **Undo**, **Cut**, **Copy**, **Paste**, **Delete**, or **Select All** the contents of the text box.
- Delete characters to the left of the cursor using the backspace key, or delete characters to the right of the cursor using the Del key.

To cancel text box input, press the Esc key.

To enter information by gestural input (scrubbing)

- Click and drag the mouse pointer in a circular motion over a text box that supports numeric values. This is known as *scrubbing*. To increase the value, scrub in a clockwise direction; to decrease the value, scrub in a counterclockwise direction.
- Hold down the Ctrl key while scrubbing to increment or decrement values by a factor of 10.
- Hold down the Shift key while scrubbing to increment or decrement values by a factor of 0.1.

To enter information by increments

- Click in the numeric text field and press the square bracket keys ([) and (]) to increment and decrement values as follows:

  ] to increment by 1
  [to decrement by 1
  Ctrl+] to increment by 10
  Ctrl+ [to decrement by 10
  Shift+] to increment by 0.1
  Shift+ [to decrement by 0.1

- Click in the numeric text field and press the arrow keys to increment or decrement values as follows:
- Ctrl+right-arrow to increment by 10

  Ctrl+left-arrow to decrement by 10

  Shift+Ctrl+right-arrow to increment by 0.1

  Shift+Ctrl+left-arrow to decrement by 0.1
- If you have a mouse wheel, click in the numeric text field, then roll the wheel forward to increment the value or backward to decrement the value.

Entering Parameters Values Using Sliders

There are three ways:

- Drag a slider to the right to raise the value or to the left to lower the value in the text box. Use the left mouse button for a continuous update as you drag; use the right button for an update only when you release the mouse.
- Drag a slider to the right or left while holding down the Shift key to make finer adjustments.
- Drag a slider to the right or left while holding down the Ctrl key to change the value of all the sliders in a color-control group simultaneously.

Moving among Text Boxes

You can move from one text box to the next one in the same dialog box or property editor by pressing the Tab key. To move back to the previous text box, press **Shift+Tab**.

When a control in the dialog box or property editor has the input focus (active), press the Enter key to activate the selected command button.

Relative Input

All text boxes allow you to input relative values by means of four basic math operations (addition, subtraction, multiplications, and division). To do so:

1. Click in a text box to select the value it contains.
2. Enter the number to add, subtract, divide, or multiply.
3. Enter the symbol for the math operator (`+ – * /`). You can use only one math operator per text box.
4. Press Enter.

For example, if the selected value in the text box equals 10 and you replace this value by 5+ and then press Enter, the result will be 15.

Entering Information for Multiple Selections

You can enter values in text boxes that apply to all similar parameter values of multiple-selected scene elements:

- Entering a parameter value in a text box resets all selected element properties to that input value. For example, entering an X-axis rotation value of 5 resets all selected objects' rotational values to 5.
- Entering a value followed by a plus sign (+) increments all the selected elements by that value. For example, entering 2+ in an X-axis translation parameter shifts all selected objects by two SOFTIMAGE units.
- Entering a value followed by a minus sign (–) decrements all the selected elements by that value.
- Entering a value followed by an asterisk (*) multiplies all the selected elements by that value.
- Entering a value followed by a forward slash (/) divides all the selected elements by that value.
- Entering the letter **l** followed by (min,max) creates a linear range through the selection, according to the original selection order. For example, L(4,8) when applied to the X-axis translation parameter of a group of three cubes causes each object to move 4, 6, and 8 SOFTIMAGE units along the X axis, respectively.
- Entering the letter **r** creates a random value for each selected element, between 0 and 1.
- Entering the letter **r** followed by (x) creates a random value for each selected element between 0 and the value for (x). If r is followed by (–x), a random value between 0 and the negative value will be inserted.
- Entering the letter **r** followed by (min, max) creates a random value for each selected element, within the specified range.
- Entering the letter **g** followed by (mean, var) creates a random value following a normal distribution among the selected elements.

Multiple Element Editing through the Property Editors

Each time you multiple-select a number of elements and open their property editors, you can simultaneously edit their common parameters.

Here are a few techniques where property editors can be used to edit multiple objects:

Multiple editing from the viewports, explorer, or Schematic view

1. In the viewports, explorer, or schematic view, select the elements in your scene to be edited.
2. Press Enter.

   A property editor displays the common parameters of the selected elements.
3. Make your edits. All your selected elements are edited simultaneously.

Multiple Object Editing from the Command Box

Let's say you want to modify a number of elements that are of the same type in your scene but you don't want to select each element individually.

1. Open the explorer and select the property node containing the parameters you want to edit.
2. At the command line, add an asterisk (*) after the selected element's name while being sure to delete any version number that may follow.

   Selecting the Global Transformation property editor of an object called cone1, for example, would look like this:

   ```
   InspectObj"cone1.kine.gtransfo"
   ```

   You would edit cone1 as follows:

   ```
   InspectObj "cone*.kine.gtransfo"
   ```
3. Press Enter to display the Global Transformation property editor for all the objects.

   Now any edits you make in the property editor are applied to all the objects simultaneously.

How Properties Are Propagated

The properties you define in property editors can be applied directly to elements at their property-node level.

Properties, however, can also be applied to nodes at a higher level and passed down, or *propagated*, to lower-level nodes of other elements in a hierarchy. In other words, propagation is the method in which an object's properties are passed down to its children, or how children in a hierarchy inherit their parents' attributes. The main reason to organize elements into a hierarchy is to allow for this type of property propagation to occur.

Propagation works in a unique way. Every object has three possible levels of propagation. They are:

- Scene Root propagation
- Branch propagation
- Local propagation.

Although an object can have all three types at once, it only displays one of these types at a time. When SOFTIMAGE|XSI is looking for a propagated properties to display, it will always select local properties over a branch properties and branch properties over scene-root properties.

Properties are only propagated if the parent is branch-selected before the property is specified. Otherwise, only local properties are applied; the object's children will not inherit any of these properties.

You can use either the explorer or the schematic view to see how propagation affects different objects in different hierarchies. Similarly, you can visualize this propagation in 3D views when objects you select as a branch display in light gray.

When you are about to modify or delete shared properties, you will be warned and asked for confirmation,

Using the Explorer to View Propagation

The explorer view is not only useful for viewing your scene's elements and their properties, but also for how they are propagated through various hierarchies.

In the explorer, properties that are applied in branch-mode, and therefore propagated, are noted with a B symbol.

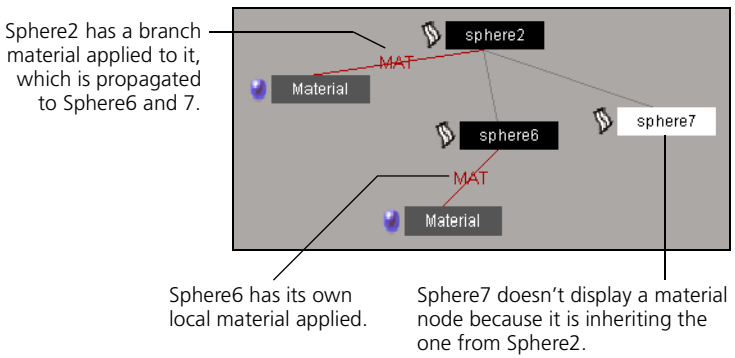
To simplify your explorer display, select **Scene** from the scope menu and **Show > Materials** from the explorer toolbar.

An object that has a shared material (such as Sphere5, above) displays its shared material in italics. The material's source (where it's propagated from) is shown in parentheses.

Properties are only propagated if the parent is branch-selected before the property is specified. Otherwise, only local properties are applied; the object's children will not inherit any of these properties.

Using the Schematic View to View Propagation

The schematic view offers a hierarchal representation of how an object's properties relate to one another.



Presets

Each time you edit properties in a property editor, you have the option of saving your settings as a *preset*. Presets are simply data files with a **.preset** file extension that contain property information. Presets let you work more efficiently because you can save the modified properties and reuse them as needed.

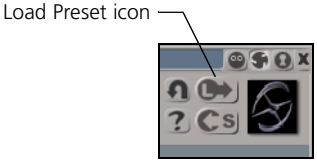
In the DSPresets folder, there are subfolders, each of which contain the existing default presets for objects, shaders, and operators. Any presets you create can be added to these folders or saved to another location of your choosing.

For quick access, you can also convert a preset into a command and place it in a toolbar. For example, you could have three buttons made from converted material presets that each apply a different surface material to an object. For more information on converting presets into commands, refer to *Chapter 8: Customizing SOFTIMAGE|XSI*.

Loading Presets

You load or open presets from property editors that define existing objects, shaders, or operators. If, for example, you wish to load a camera preset, you must first open an existing camera property page. Presets for operators are loaded from operator property pages, and so on.

To load a preset



1. Open an object, shader, or operator property editor from the toolbar, explorer, or render tree.
2. In the property editor, click the Load Preset icon.
3. In the Load Preset browser, select the preset you wish to load and click OK.

   The selected preset parameter values appears, replacing those of the currently displayed property page.

Saving Presets

Once you have modified properties in a property editor, you can save them as a preset for later use.

To save a preset

1. In the property editor, click the Save Preset icon.
2. In the Save Preset browser, select the folder in which you wish to save the preset.
3. In the **File Name** text box, enter the name you wish to give the preset, then click OK.

Selecting and Deselecting Objects

As you work, you constantly select and manipulate objects and their components. There are a number of tools you can use to select these elements, as well as a number of views in which your selections can be made.

SOFTIMAGE|XSI and SOFTIMAGE®|3D Interaction

When SOFTIMAGE|XSI was first loaded into your system, you had the option of using the SOFTIMAGE|XSI keyboard mapping and mouse-interaction method or that used by SOFTIMAGE|3D. All the selection tools and interaction discussed in the pages that follow assume that you are currently using the SOFTIMAGE|XSI mapping and interaction method.

If you are unsure of the method set in your system or know you are using the SOFTIMAGE|3D method and want to switch to that of SOFTIMAGE|XSI, do the following:

1. Choose **File** > **Keyboard Mapping**, and in the Key Map pop-up menu choose **XSI Key Map** or a similar keyboard mapping template that you know is based on XSI keyboard mapping.
2. Choose **File** > **User Preferences** to open the User Preferences dialog box.
3. Click the **Interaction** tab and in the Tool Activation controls, toggle **Selection** on and **Enable Sticky Keys** on.
4. In the Selection panel, click the **Selection** menu button and make sure **SI3D Selection Model** is not selected.

For more information on the differences between SOFTIMAGE|XSI and SOFTIMAGE|3D interaction, from the main-menu bar choose **Help** > **Contents and Index** > **Data Views** > **User Preferences** > **Interaction Model Dialog Box**.

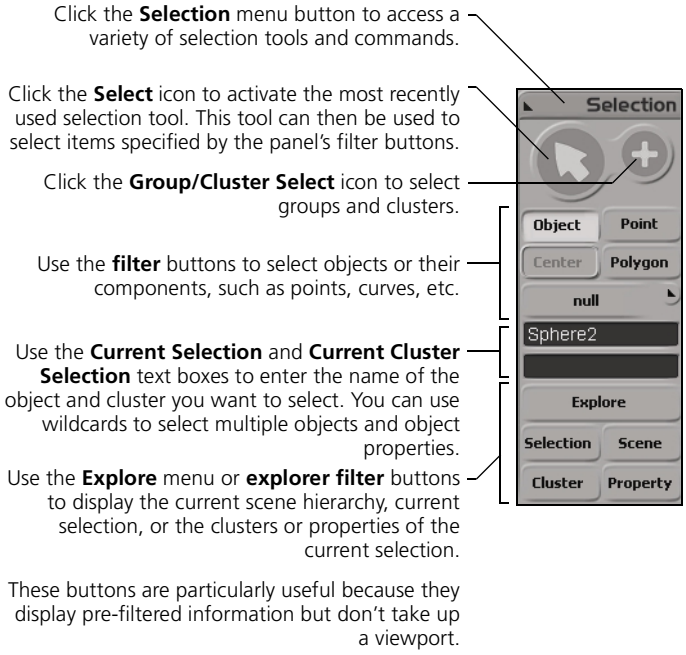
Using Supra Keys for Selecting Objects

There are a number of supra keys that assist you while selecting and manipulating objects.

- Use the space bar to activate object-selection mode.
- Use the Shift or Ctrl keys for selecting and deselecting multiple objects:
  - **Shift** extends a selection and adds to the current selection.
  - **Ctrl** extends a selection and toggles the selected state of picked objects.

The Selection Panel

The Selection panel provides a number of commands and tools for selecting objects in your scene.



Select Icon

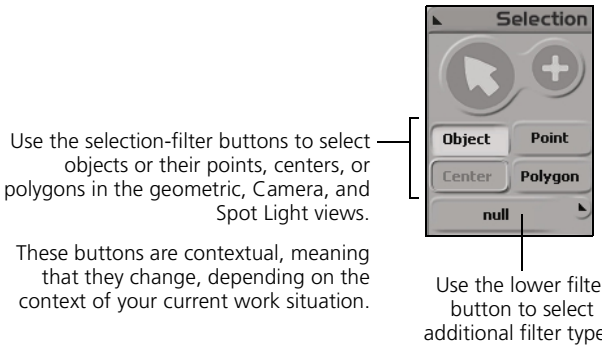
The Select icon in the Selection panel activates the last selection tool. Selection tools, which include Rectangle, Lasso, Free-Form, and Paint, let you select objects or components in the viewports.

The Select icon applies only to selecting objects in the viewports. You can, however, select any object at any time by clicking on its node in the explorer or schematic view.

Selecting an object highlights it, indicating that it is ready to receive the actions that you are about to perform.

Using Filters to Define Selectability

The selection filters (below the Select icon) let you specify what can be selected (point, polygon, edge, etc.) with your selection tool (e.g., Rectangle, Lasso, Free Form, etc.). in any of the viewport's 3D views.



Using filters can come in handy if your scene is complex and you wish isolate certain types of objects for selection.

Selecting a new filter deselects any currently selected components unless you Shift+click on the filter.

Selecting a component filter will restore the selection context, as it was the last time the filter was active. If, for example, you select the **Point** filter, the most recently selected points on the active objects when that button was last active will automatically be selected.

Selecting Interactively

Selecting objects with your mouse is as simple as clicking and dragging on them in a geometry view or by clicking on their nodes in the explorer or schematic views.

Selecting objects deselects any other selected object unless you use a modifier key such as Shift or Ctrl to extend the selection.

Selected objects are highlighted in white in the geometry, explorer, and schematic views.

To select a single object

1. Click the Select icon. You can now select any item in the scene that corresponds to one of the active filter buttons in the Selection panel. In other words: to select a single object, the **Object** filter button must be selected; to select a point, the **Point** filter button must be selected; and so on.
2. Click on any object in the viewport.

You can click on an object node in the explorer to select it even if the Select icon is not active.

3. The object is highlighted in white to show it is selected. If you selected a node in the explorer, its associated object also appears in blue-gray in the viewport.

Selecting Multiple Objects

There are many ways of selecting more than one scene element at a time. You can:

- Use modifier keys such as Ctrl+a to select all objects in the scene, or Alt+a to select all elements that correspond to the Selection panel's active selection filter.
- Use the Rectangular, Lasso, and Free Form selection tools found in the Selection menu.
- Enter object or component names together with wildcards in the Selection panel's selection text boxes. For more information on this method, see *Selecting Objects by Name* on page 125.

Extended Object Selection

1. Click the Select icon.

2. Do one of the following:
   - Press **Shift** while successively picking objects. The existing selection is preserved and any objects picked afterward are added to the current selection.
   - Press **Ctrl** while successively picking objects. Any existing selection is preserved and the selection state of the objects picked afterward is toggled (that is, selected objects become deselected).
   - Press **Ctrl+Shift** to extend your selection but deselect target objects.

   Any existing selection is preserved once the Shift or Ctrl key is depressed, and any subsequent objects that are clicked are added to the selection. Release Shift or Ctrl when you have finished selecting objects.

Rectangular Selection

Rectangular selection is a region selection that lets you select more than one item at time.

1. Make sure the Select icon is active.
2. Choose **Selection** > **Rectangle Tool**.
3. Click and drag the mouse pointer in the viewport to create a rectangle around any set of objects in the scene. Make sure that the rectangle touches at least a part of the object you want included in the selection.

Lasso Selection

A Lasso selection is a region selection that lets you select one or more objects at any level (node, branch, or tree) by drawing a free-from shape around it.

1. Choose the **Selection** > **Lasso Tool**.
2. Drag the mouse pointer around any objects in the scene. This draws a "lasso" around the objects.

Free-Form Selection

Free-Form selection lets you draw a free-form line across any number of objects to select them. Any object touched by the line you draw is selected.

1. Choose the **Selection** > **Free-Form Tool**.
2. Drag the mouse pointer over any object in the viewport; this draws a free-form line across the objects. All objects touched by this line are selected.

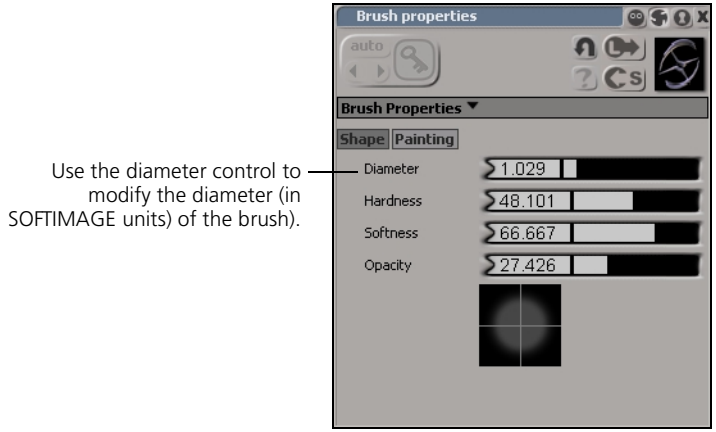
Paintbrush Selection

The Paint selection tool is a sophisticated way to select components of objects. By modifying the paint tool's brush size, you affect the way in which components are selected.

You can activate the tool by choosing **Selection** > **Paint Tool** in the Selection panel.

To modify paint selection tool brush size

Select the paint tool and enter **Ctrl+w** to display the brush property editor.



Selecting Single Objects in a Region

By default, when you use the Rectangle, Lasso, or Free Form tool, all the objects in the drawn area are selected. However, if you choose **Selection** > **Select Single Object in Region**, only one object is selected.

Extending a Region Selection

You can use the Ctrl or Shift keys for extending the selection (in conjunction with mouse dragging). You must press one of these keys before clicking the mouse button to extend the selection. The following example uses Lasso selection and the Shift key to select three objects at different locations in a scene.

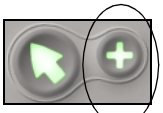
1. Choose the **Selection** > **Lasso Tool**.
2. Drag the mouse pointer around the first object.
3. Press Shift and drag the mouse pointer around the second object.
4. Press Shift again and Lasso-select object C. All three objects are now selected. You can add objects to any type of region selection in this manner.

Selecting All Objects

You can select all objects in a scene, including geometric objects, lights, and cameras. To do so, choose **Selection** > **Select All Objects**.

The objects are highlighted in white to show they are selected.

Selecting Groups and Clusters



Group/Cluster Select icon activated for group selection

Selecting Groups

To select groups of objects in the viewport, click the **Group/Cluster Select** (+) icon. In this mode, you can select only object groups or ungrouped objects.

When the Group/Cluster Select icon is active, you cannot select the individual objects in a group.

Selecting Clusters

A cluster is a set of components (polygons, points, edges) that have been selected and grouped together (see *Creating Groups and Clusters* on page 141).

To select a cluster in a viewport

Click the **Group/Cluster Select** icon.

In this mode, you can select only clusters of the same type as the currently active filter. You can only select a cluster of points, for example, if the **Points** filter is active.

When the **Group/Cluster Select** icon is active, you cannot select individual components in a cluster. If a selected component is included in more than one cluster, all clusters are selected.

To select a cluster by name

1. Select the object to which the cluster belongs.
2. Type the cluster's name (such as **cluster_a**) directly in the **Current Cluster Selection** text box and press Enter.

or

Use the asterisk (*) symbol (such as **cluster***) to select all clusters beginning with a specific string.

When more than one cluster is selected, the text box displays the word "**MULTI (n)**" with (**n**) representing the number of clusters selected.

To select a cluster from the pop-up explorer

1. Click the **Cluster** button in the Selection panel to display a pop-up explorer listing clusters of currently selected objects. This list displays all the clusters from which you can select in a manner similar to that of the explorer.
2. Choose the name of the cluster you want to select. The cluster name is displayed in the text box and the components appear as selected in the viewport.

Selecting Points

A point is an XYZ location in space. Objects are made up of points. You can manipulate points in polygon-mesh or surface-mesh objects to define its size and shape. For more information on transforming, adding and deleting points, refer to*Chapter 2: Polygons & Polygon Meshes* in the *Modeling & Deformations*.

To select (tag) and deselect (untag) points

1. Select a polygon- or surface-mesh object.
2. Choose the **Point** filter button on the Selection panel or press the **t** key.
3. With the Rectangle, Lasso, Free-Form, or Paint selection tool active, and **Selection** > **Extended Component Selection** set, then you can

   - Left-click to select points.
   - Middle-click to deselect points.
   - Right-click to switch between selecting and deselecting points.

   If Extended Component Select is not set, these selection tools do the following:

   - Left-click to select points.
   - Middle-click to select clusters.

Extended component selection mimics the SOFTIMAGE|3D method for tagging, untagging, and inverting selection of vertices (points). Choose **Selection** > **Extended Component Selection** to switch this option on or off.

Selecting Polygons

You can modify polygon-mesh objects by transforming, adding, or deleting its polygons. For more information on these procedures, refer to *Chapter 2: Polygons & Polygon Meshes* in the *Modeling & Deformations* guide.

To select polygons

1. Select a polygon-mesh object.
2. Choose the **Polygon** filter button on the Selection panel or press the **y** key.
3. Select one or more points in the polygon.
   - With the Rectangle, Lasso, and Paint selection tools, you must include all of a polygon's points to select it.
   - With the Free-Form Selection tool, you need only draw anywhere within the polygon to select it.

Selecting Objects by Name

You can select objects and parameters by entering their names in the Command box or in the Selection panel's text boxes. This method of selection lets you use wildcards, which are particularly helpful if you need to select multiple items.

Command Box Entry

To select objects and parameters by name in the Command box, type the SelectObj command followed by the object name, then press Enter. The name itself must be surrounded by double quotes. For example:

```
SelectObj "CarTire1"
```

Selection Panel Text Box Entry

The Current Selection and Current Cluster Selection text boxes do not require entry of the SelectObj command or double quotes. Simply type the object name in the upper box, then press Enter.

When more than one object is selected, the text box displays "**MULTI (n)**", where (**n**) indicates the number of objects that are selected.

Examples of Selection Syntax

The table below demonstrates the many ways you can enter names and wildcards for selection of one or more objects.

Syntax	Selects...
"Obj1,Obj2"	two 3D objects called Obj1 and Obj2.
".kine.ltransfo.posx"	the posx parameter of the local transformation of currently selected objects.
"Obj*"	all 3D objects whose names begin with "Obj"
"*.MyProp*"	all properties that begin with the name "MyProp" in all objects in the scene.

Syntax	Selects...
"*.#material*"	the material of all 3D objects in the scene. In this case, the word "#material" is recognized as the generic name for all material used in the scene.
"{Obj1,Obj2}.kine.ltransfo.posx"*"	the posx parameter of the local transformation of objects called "Obj1" and "Obj2".
"Cone.point[3,6,8]"	points 3, 6, and 8 on an object called "Cone".
"Cone.point[3-8,10]"	points 3 to 8 and 10 on an object called "Cone".
"Cone.point[*]"	all points in an object called "Cone".
"Cone.point[3-LAST]"	points three and higher in an object called "Cone".

Inverting Selection Status

There are a few ways to invert the selection of objects so that selected objects become deselected, and vice versa:

- Choosing **Selection** > **Invert Using Filter** inverts the selection status of all objects in the scene. (**Object** must be set as the selection filter.)
- Toggle the selection state of an object on and off simply by clicking on it.
  - **Ctrl**+**left-click** a selected object in a multiple selection to deselect just that object.
- Hold down the Ctrl key and do a region selection. The selection state of all objects included in the region selection is toggled.

Deselecting Objects and Components

There are several ways to deselect objects and their components:

- Select another object: this automatically deselects all previously selected items, unless you use a modifier key such as Shift and/or Ctrl for multiple selection.
- To deselect all items currently selected in the scene:
  - Drag across an empty region in a viewport to clear the current selection. Simply clicking on an empty region will not deselect anything. This makes it harder to inadvertently deselect your objects and components.
  - Choose **Selection** > **Deselect All** in the Selection panel, or use the keyboard shortcut **Shift**+**Ctrl**+**a**.
  - Press Ctrl and click on the selected object or component: This toggles the selection state, thus deselecting it.
- Hold down the **Shift**+**Ctrl** keys and do a region selection. All objects and components included in the region are removed from the selection.

Defining Object Selectability

You can define whether an object, group, or cluster in a viewport is selectable or not. This will come in handy and speed up your workflow if you are working in a very dense scene and there are one or more objects that you don't wish to select.

Setting an object's selectability

1. Select an object, group, or cluster in the viewport for which you want to toggle its selectability.
2. Click the **Property** button on the Selection panel to display the selected object's property nodes.
3. Click the Visibility node, then click the **Selectability** check box to remove the check mark. Your selected object, group, or cluster will now be unselectable until you click the check box again.

   Unselectable objects are displayed in dark gray.

Duplicating Objects

There are two ways to duplicate objects in your scene. You can make *copies* or *instances*. These types of duplications can also be applied to groups and hierarchies.

Creating Copies of Objects

When you create a copy, you create an independent duplicate of the original. The copy bears some or all the characteristics of the original at its moment of duplication, but from that moment on any changes to the original has no effect on the copy. Which characteristics of the original are copied depend on the settings you specify in the Duplicate Options property editor.

To copy one or more objects

1. In any viewport, select the object(s) to be copied.
2. If you want to specify only certain characteristics of the original to be copied, choose **Edit** > **Duplicate/Instantiate Options** from the Edit panel and make your selections.
3. *For a single copy:*

   Press **Ctrl**+d or choose **Edit** > **Duplicate Single** from the Edit panel.

   For multiple copies:

   Press **Ctrl**+**Shift**+**d** or choose **Edit** > **Duplicate Multiple** from the Edit panel and specify the number of copies required from the pop-up dialog box.

Including Transformations in Duplicated Objects

If you duplicate an object and transform it, the same transformation will be applied to the object created by the next Duplicate command.

To include transformations in duplicated objects

1. In any viewport, select the object(s) to be copied.
2. Choose **Edit** > **Duplicate/Instantiate Options** from the Edit panel and make your selections from the Duplicate Options property editor.

   See Online Help for a description of the duplicate controls available from this editor.
3. *For a single transformed copy:*

   Press **Ctrl**+**d** or choose **Edit** > **Duplicate Single** from the Edit panel.

   The duplicated object is scaled, translated, and/or rotated using the same factor/offset values with respect to the previous duplicated object.

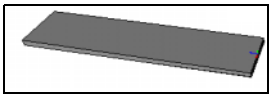
For multiple transformed copies:

Press Ctrl+**Shift**+**d** or choose **Edit** > **Duplicate Multiple** from the Edit panel and specify the number of copies required from the pop-up dialog box.

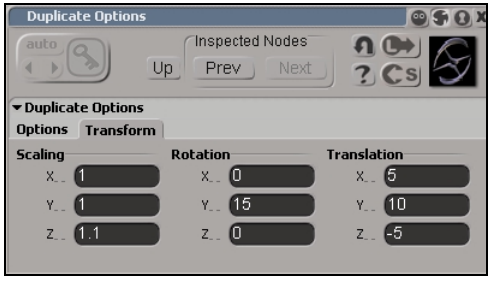
The specified number of duplicated objects appears, with each one being scaled, translated, and/or rotated using the same factor/offset values with respect to the previous object generated by the command.

Example: Applying multiple transformations to duplicated objects

1. The object to be duplicated and transformed—a step—is selected.



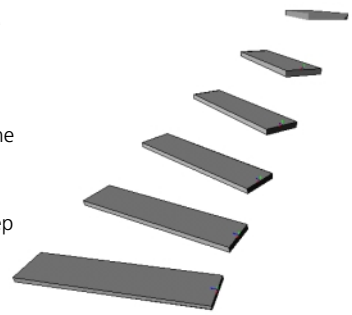
2. The translation, rotation, and scaling values shown to the right are specified in the Duplicate Options Transform controls.



3. With the object selected, **Edit > Duplicate Multiple** is chosen and five copies are specified.

   Result: Five copies of the original step are generated, with each duplicate translated, rotated and scaled to give the appearance of a flight of stairs.

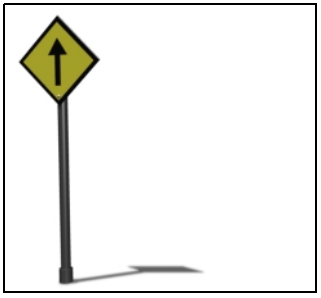
   (Note: the center of the step was repositioned to the right so that the step could be rotated along its right edge.)



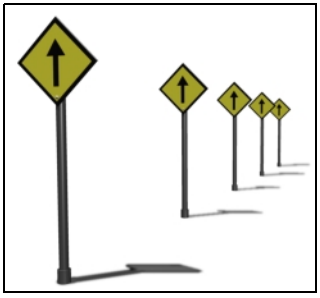
Creating Instances of Objects

When you create an instance, you create an entity whose characteristics reflect those of its original. Transformations (scaling, rotation, translation) are local to the instance. However, any change to the geometry of the original is reflected in all of its instances.

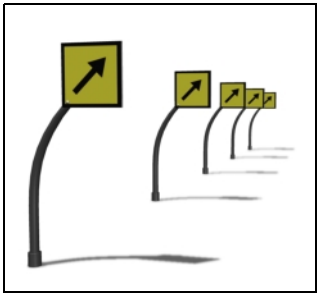
Copying Objects vs Instancing



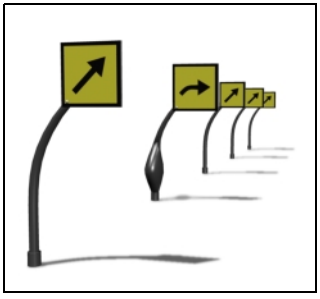
When an object is **copied**...



... the original and its duplicates can be modified separately with no effect on each other.



When an object is **instanced**, editing the master object affects all the slaves...



... but editing only one slave has no effect on the others.

Instantiation has the following advantages:

- Instances use much less disk space than duplicates.
- Editing multiple identical objects is very simple.
- Wireframe, shading, and memory operations are much faster.

You can further define how the original is instanced in the Duplicate/Instantiate Options property editor.

To create instances of one or more objects, groups, or hierarchies

1. In any viewport, select the item(s) to be instantiated.
2. If you want to specify only certain characteristics of the original to be instanced, choose **Edit** > **Duplicate/Instantiate Options** from the Edit panel and make your selections.
3. *For a single instance:*

   Press **Ctrl+i** or choose **Edit** > **Instantiate Single** from the Edit panel.

   For multiple instances:

   Press **Ctrl+Shift+i** or choose **Edit** > **Instantiate Multiple** from the Edit panel and specify the number of instances required from the pop-up dialog box.

Instanced items are identified in the schematic view by a capital "I" above their node label and in the explorer by an "I" superimposed over their node icon.

Deleting Objects

To delete objects in your scene, do one of the following:

1. Press the Backspace key to activate the Delete tool. The mouse pointer changes shape.
2. Select the objects you want deleted from the scene.
3. Backspace a second time to deactivate the tool.

or

1. Select one or more objects in the scene using the Selection panel tools.
2. Choose **Edit** > **Delete Selected** or press the Delete key.

   If you want to delete all the objects in the scene, simply choose **Edit** > **Delete All** or press **Shift**+**Delete**. This is the same as deleting the entire scene. When you choose this command, you are asked if you want to delete the scene.

Undoing and Redoing Edits

You can undo most edits made from the interface, such as scaling an object or deleting a camera.

To undo an edit, choose the **Edit** > **Undo** command or press **Ctrl+z**. You can continue to undo actions as far back as the system memory can recall. This is limited by the number of undo levels that you have specified in your user preferences.

To redo an edit, choose the **Edit** > **Redo** command or press **Ctrl+y**. You can invoke one redo command for each undo action.

Undo does not work on:

- changes made to shaders
- changes in viewport manipulations
- changes to light ray properties

Setting the Number of Undo Levels

There's enough space in the system memory to let you go back and undo a number of edits.

To set the number of undo levels

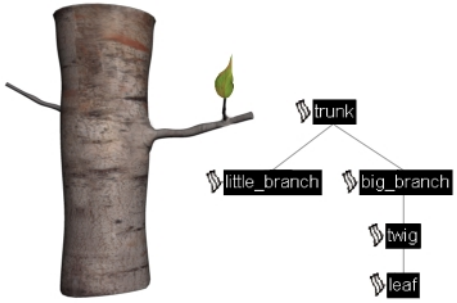
1. Choose **File** > **User Preferences**.
2. In the **User Preferences** dialog box, click the **General** tab.
3. In the **Number of undo levels** text box, enter the number of edits that can be undone (maximum 1000).

The higher number of levels you set, the more memory is required (which can slow down performance).

Hierarchies

A hierarchy describes the relationship between objects (usually using a parent-child analogy). For example, the parent-child relationship means that any properties applied to the parent (in branch mode) also affects the child.

- An *element* is anything that can be visualized in 3D space or displayed. Elements can be renderable or non-renderable. They include projects, scenes, passes, lights, cameras, geometric objects, shaders, and certain object properties.
- A *node* is a graphical representation of any element displayed in a tree structure in the explorer. Each node is identified by a separate icon and a label. A node can have children, and it is called the **parent** of its children. Nodes at the same hierarchical level are called *siblings*.
- A *root* is a node at the base of either a branch or the entire tree. The only nodes that have no parents are the scene nodes and the render passes.
- A *tree* is a hierarchy of nodes starting just below a *model* (see below). A subtree is a subhierarchy of nodes starting from any node in a tree. In the case of geometric object nodes, this is also called a *branch*.
- A group of hierarchies and parts of hierarchies can be saved as a single entity called a *model*. A model generally makes up a unified whole, in the same way a lamp is composed of component parts or a room is made up of furniture. Each time you create a new scene, Scene_Root is set up as the default model.



Selecting Objects in a Hierarchy

When objects are in a hierarchical relationship, you can use the different mouse buttons to select a single object in a hierarchy, a branch of the hierarchy, or, in the case of 3D views, an entire tree. The **Selection** menu in the Selection panel lets you select these and other objects in a hierarchy.

Selecting objects in a hierarchy is different than selecting multiple individual objects because it respects the hierarchical links you created between objects in your scene.

To select objects in a hierarchy

1. Make sure you are in selection mode by clicking the Select icon or hitting the space bar.
2. Click on any object in the scene hierarchy as follows:

Node selection

- Left-click to select an object at the node level. Only that object is selected, not the rest of the branch.

Branch selection

- Middle-click on any object to select the whole branch. (If you middle-click on a node with no children, just this node is selected.)

  or

- Choose **Selection** > **Select Branch** from the Selection panel.

Tree selection

- 3D views only: Right-click on any object in the hierarchy to select the tree. Schematic view and explorer: Right-click on any object in the hierarchy to display a context menu and choose **Select Tree**.

  The entire hierarchy, with the exception of the model root, is selected.

Model selection

- Choose **Selection** > **Select Model** from the Selection panel.

Other hierarchy selections

Select an object and use the **Selection** menu to access these other hierarchy selection commands: **Select Tree**, **Select Child Nodes**, and **Select Members/Components**.

When a hierarchy is selected, the parent is highlighted in white and the children are highlighted in gray.

Navigating Hierarchies

You can use your keyboard arrow keys to move up and down hierarchies in all views:

- Alt+up arrow = move to the parent
- Alt+down arrow = move to the first child
- Alt+left arrow = move to the previous sibling
- Alt+right arrow = move to the next sibling

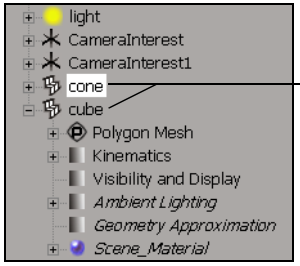
For more information on navigating hierarchies in the explorer, refer to *Viewing the Project Tree* on page 46.

Creating Hierarchies

Objects can be associated to each other in a hierarchy for a number of reasons, such as to facilitate manipulation or to propagate applied properties. This is usually referred to as *parenting*. In a hierarchy there is a root, its children, the children's children, and so on. The combination of links that can be created is almost infinite.

When creating a hierarchy, you should first analyze how you want the properties in the hierarchy arranged so that you can parent the objects appropriately.

You can pick a parent from different elements to create a hierarchy. In a hierarchical structure, properties are propagated vertically from top to bottom, as in parent to child. You can create a hierarchy by parenting objects with the **Parent** button or by dragging and dropping one node onto another in the explorer.



Hierarchy example: Make a cone the child of a cube:

Drag and drop the label of the cone node to that of the cube node.

There are no restrictions on whether you start creating your hierarchy by defining the parent or the children, but sometimes it's easier to create the parent first. Here's how:

1. Select the object you wish to be the parent.
2. Click the **Parent** button in the Constraint panel or press the / key.

   The cursor is accompanied by the text "pick" to indicate you are now ready to select the children of the parent object.
3. Pick each of the children with the left mouse button.
4. Right-click or press the Esc key to end the parenting mode.

Cutting Links in a Hierarchy

You will often need to cut the hierarchical links between objects. The **Cut** button in the Constraint panel allows you to cut the link between a parent and its child or children in a hierarchy of objects. You can also cut a hierarchical link in the explorer by dragging and dropping the child label to a node higher than its parent.

If the child is also a parent, the links to its own children are not affected.

1. Select the object you want to detach from its parent.

Click the **Cut** button in the Constraint panel or press **Crtl+/**.

Models

Models are a powerful way to organize objects in your scenes and projects. Models contain not just the geometry of hierarchies, but also the function curves, shaders, and other properties that have been applied to it. They can also contain internal expressions and constraints; that is, those expressions and constraints that refer only to elements within the model's hierarchy. Models are like "sub scenes" that can be easily reused in scenes and projects.

Every scene contains at least one model, the Scene_Root element, which is the parent of all other models. You can create as many other models as you like.

Models are nulls that act as the root of a hierarchy, and they also have other properties. In the explorer, models are distinguished by special icons. You can set the explorer to show only models and greatly simplify the representation of your scene.

Models contain mixer elements that can store and reuse actions, shapes, and so on. This lets you sequence and mix animation on models. Models are the only type of elements in which you can store actions. If you select an object that is not a model and store an action, it is stored in the parent model if you have created one, or in the Scene_Root otherwise.

Models can also be children of other models.

Namespaces

Each model maintains its own namespace. This means that each 3D object in a model's hierarchy must have a unique name, but 3D objects in different models can have the same name. For example, two characters in the same scene can both have chains named left_arm, right_arm, and so on.

If you are familiar with SOFTIMAGE|3D, note that the concepts of models and namespaces replace the method of using prefixes to organize scene elements.

All models exist in the namespace of the scene. This means that each model must have its own unique name, even if it is within the hierarchy of another model.

Namespaces let you reuse animations that have been stored as actions. If an action contains animation for one model's left_arm chain, you can apply the action to another model and it automatically connects to the second model's left_arm. If your models contain elements with different naming schemes, for example, LEFTARM or LeftArm, you can use connection mapping templates to specify the proper connections. Actions and templates are described in *Chapter 11: Actions* of the *Animating* guide.

Types of Models

There are three types of models: internal, external, and exported. You can specify whether a model is internal or external when you create it as well as when you merge a scene. Once your scene contains a model, you can export it.

Internal Models

Internal models are saved within the scene file. They provide a way to organize a scene, allowing you to take advantage of namespaces so that models can contain objects with the same name.

External Models

External models are saved outside of the scene file. However, they retain their links to other scene elements. For example, if a model is animated on a path and the path is not a child of the model, the model still retains the path constraint when the scene is reopened.

External models not only provide a way to organize your scene, they can also reduce time when saving complex scenes. When you save a scene that has many external models, only those models that are "dirty"—that is, that have been modified since the last save—are re-saved.

External models are saved in the Scenes folder of your project. They have an **.mdl** extension and a file name composed of the model name prefixed by the scene name, for example:

```
myScene-myModel.mdl
```

Exported Models

Exported models can be used in other scenes. When you export a model, a copy is saved as an independent file. The exported model contains only its internal relationships—the export process removes constraints, modeling relationships, expressions, linked parameters, and so on if they involve elements that are not children of the model.

Exported models are saved with an **.emdl** extension and a file name you choose.

Working with Models

Creating Models

2. Select the objects that will compose the model. The objects must be siblings; that is, they must be at the same level and share a parent node in the explorer hierarchy.
3. Click **Create** > **Model** on the Model toolbar. The Model property editor opens.
4. Specify a **Name** for the model, as well as whether the model's storage is **Internal** or **External**.

   A null is added to the scene as the parent of the selected objects. You can add other objects to the model at a later time by using the **Parent** button on the Constraint panel or by dragging and dropping them onto the model's node in the explorer.

Selecting Models

To select the model and the hierarchy beneath it, choose **Selection** > **Select Model** from the Selection panel. To select just the model itself, click on its node in the explorer or the schematic view.

Editing Model Properties

Model properties are edited in their respective property editors. You can open a model's property editor by clicking the model icon in an explorer. You can change the model name as well as specify whether the storage is internal or external.

Merging SOFTIMAGE|XSI Scenes

When you merge a SOFTIMAGE|XSI scene into the current scene, it is automatically made into a model. This ensures that the names of elements are preserved—because each model maintains its own namespace, there is no need to append element names with numbers to make unique names.

To merge a SOFTIMAGE|XSI scene into the current scene

1. From the main-menu bar, choose **File** > **Merge**. A browser opens.
2. Use the browser to locate and select a SOFTIMAGE|XSI scene, then click OK. The Model property editor opens.
3. Specify a **Name** for the model, as well as whether the model's storage is **Internal** or **External**.

If you don't want the elements of the merged scene to be a separate model, you can "unparent" the model's children using either the **Cut** button on the Edit panel or by dragging and dropping their nodes in the explorer, then delete the model node.

Importing Models

When you import a model, you have the option of placing the model directly below the Scene_Root model or elsewhere in the hierarchy.

To import a model

1. Choose **File** > **Import Model**.

   The Import Model browser displays.
2. Open the folder containing the model you wish to import. SOFTIMAGE|XSI models are identified by their **.mdl** and **.emdl** file extensions.
3. Select the model filename and click OK.

   The model is placed just below the Scene_Root in your scene hierarchy.

Exporting Models

You can export models created in SOFTIMAGE|XSI for use in other scenes. When you export a model, a copy is saved as an independent file.

To export a model

1. Select the model to be exported.
2. Choose **File** > **Export Model**.

   The Export Model browser displays.
3. Open the folder in which the model is to be saved and, under **File Name**, enter the name of the model to be exported then click OK.

Groups and Clusters

You can organize 3D objects, cameras, and lights into groups for the purpose of selection, applying operations, assigning properties and shaders, and attaching local materials and textures. Similarly, you can group object components such as polygons and points into clusters to apply operations, assign properties, etc.

Besides being able to organize objects into groups and object components into clusters, you can also create a group of groups or a group of clusters.

An object can be a member of more than one group.

Even if an object or component is in a group, it can still be individually selected.

Creating Groups and Clusters

Any parameter values or operations applied to a group of objects or a cluster of components are shared by each member of the group. Parameter values are stored at a group level and override any existing values previously assigned to any members within the group. Original values are kept for each object or component in a group.

Grouping avoids re-selecting the same set of objects or polygons each time you want to work with them.

To create a group or cluster

1. Select one or more objects or components; you can also select multiple groups.
2. Click the **Group** button in the Edit panel, or, if you are in component-selection mode, click the **Cluster** button.

   All selected objects are grouped together using the default names **Group1**, **Group2**, **Group3**, and so on.

   All selected components are grouped together using the default name **Cluster1**, **Cluster2**, **Cluster3** and so on.

You can also create, add to, and delete groups from context menus in the explorer.

Adding and Removing Elements from Groups or Clusters

You can add or remove individual items to and from groups and clusters.

To add items to a group or cluster

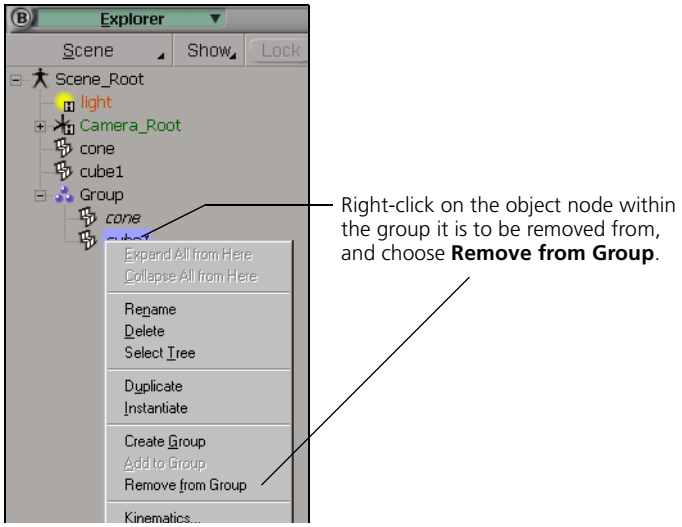
1. Select one or more objects or components and the group or cluster to which you want to add the object or component.
2. In the Edit panel, choose **Edit** > **Add to Group**. If you are in component selection mode, choose **Edit** > **Add to Cluster**.

To remove objects from a group

1. Select the object to be removed from the group.
2. In the Edit panel, choose **Edit** > **Remove from Group**.

   or

   If the object to be removed belongs to more than one group, open the node of the desired group in the explorer, right-click the node of the object to be removed, and choose **Remove from Group**.



To remove components from a cluster

1. Select an object with a cluster.
2. Select a cluster.
3. While holding the Shift key down, select components with the left mouse button.
4. Click the **Uncluster** button or choose **Edit** > **Remove from Cluster** from the Edit panel.

Deleting Groups and Clusters

When you delete groups and clusters, only the relationship between the objects or components is deleted, not the objects or components themselves.

To delete a group

1. Select the group you want ungrouped.
2. Click the **Ungroup** button or choose **Edit** > **Remove Group** from the Edit panel.

Alternatively, you can delete a group using the explorer view.

When you delete an entity that contains other groups, the "parent" group is removed and the "children" groups are preserved.

To delete a cluster

1. Select an object with a cluster.
2. Select a cluster.
3. Click the **Uncluster** button or choose **Edit** > **Remove Cluster** from the Edit panel.

Alternatively, you can delete a cluster using the explorer view.

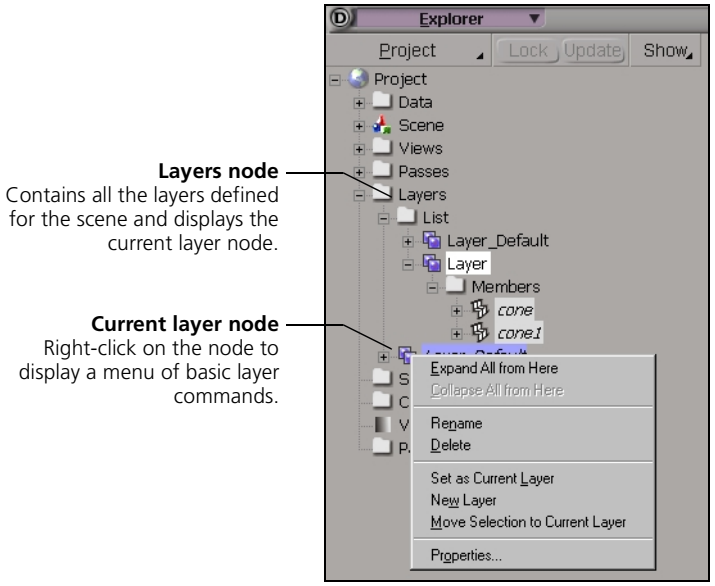
Layers

A large and complex scene can significantly slow down the refresh rate and clutter the display with many objects. Creating scene layers is useful when you wish to concentrate on only a few objects in a crowded scene or to save time refreshing the display when applying rendering options and effects.

Layers help you organize, view, and edit the contents in your scene but do not affect the final render and are not associated with render passes. You can put different objects into each layer and you can hide a particular layer if you don't want to see that part of your scene. You can also make a layer unselectable. Layer definitions are saved with the scene file.

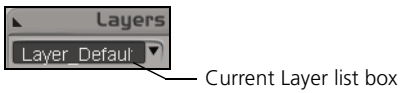
Controlling Layers Using the Explorer

Layers are displayed in the explorer as subnodes under the project. The Layers node is a container for all the layers defined for the scene.



Controlling Layers from the Layers Panel

You can choose additional layer options from the Layers menu in the Layers panel.



Creating a Scene Layer

By default, it is set as visible and selectable. To create a new layer, choose **Layers** > **New Layer**.

Setting the Current Layer

Before you add objects to a layer, you must specify that it's the current layer. The current layer is the layer to which all layer operations are applied. When you create new objects, they are automatically assigned to the current layer. Also, when you choose commands like **Move Selection to Current Layer**, the command moves the selected objects to the current layer. When you select a layer, its name appears in the Current Layer display box in the Layers panel.

When you import a scene into SOFTIMAGE|XSI, it contains a single layer named *Layer_Default*. This layer is set to display every object in your scene. It is also the current layer for the scene.

To set the current layer

1. Click the arrow button next to the **Current Layer** text box to display the drop-down list. The list contains all available layers defined for the scene.
2. Choose the layer you want to set as current. Any new objects you create are assigned to that layer.

To move objects to a layer

1. Click the arrow button next to the Current Layer text box and choose the layer you want to set as current. This is the layer to which you want to add objects.
2. Select the objects in the viewport that you want to move to the current layer.
3. Choose **Layers** > **Move Selection to Current Layer**. The selected objects are assigned to the current layer.

Each object in your scene can belong to only one layer at a time. It is possible for different objects within a given hierarchy or group to be assigned to different layers.

To delete a scene layer

1. Click the arrow button beside the **Current Layer** text box and choose the layer you want to delete.
2. Choose **Layers** > **Delete Current Layer**. The current layer is deleted and the objects it contained are automatically reassigned to the *Layer_Default*.

When the current layer is deleted, the preceding layer in the list becomes the new current layer. You cannot delete a layer when it's the only one in the scene.

Setting Layer Visibility Using "Layer Control"

You can choose which layers you want to view while working. By default, the objects in all of the defined layers are displayed in the viewports and rendered in the render region, but you can deactivate the visibility of the objects in a layer at any time. For example, you can set the objects in a layer to be visible in the viewports but not in the render region, and vice versa. You can also activate or deactivate the selection of objects in a layer. It is possible to make a layer invisible, but all the objects inside it can still be selected in the explorer.

Setting layer visibility in the viewports

1. Click the arrow button next to the Layer Control text box and choose the layer you want to set as current. This is the layer for which you want to set visibility.
2. Choose **Layers** > **Viewport Visibility**. When active (a check mark appears beside the command), all the objects in the current layer are visible in the viewport.

   To deactivate visibility so that the objects in the current layer are not displayed in the viewports, choose **Layers** > **Viewport Visibility** again.

Setting layer visibility in the render region and final render

1. Click the arrow button next to the **Current Layer** text box and choose the layer you want to set as current. This is the layer for which you want to set visibility.
2. Choose the **Layers** > **Rendering Visibility**. When active (a check mark appears beside the command), all the objects in the current layer are visible and rendered in the render region.

   To deactivate visibility so that the objects in the current layer are not displayed in the render region or rendered in a final render, choose **Layers** > **Rendering Visibility** again.

Setting Layer Selectibility

You can specify whether or not the objects in a layer can be selected. If you have a number of layers whose objects are visible in the viewport, you can choose to deactivate the selection of the objects in some of these layers. This way you can view all objects but control which objects you want to be able to select.

1. Click the arrow button next to the **Current Layer** text box and choose the layer you want to set as current. This is the layer for which you want to set selectability.
2. Choose **Layer** > **Selectability**. When active (a check mark appears beside the command), all the objects in the current layer can be selected in the viewport.

   To deactivate selectability so that the objects in the current layer are not selectable in the viewport, choose **Layer** > **Selectability** again.

Setting Layer Visibility and Selection from the Layers Container Property Page

The Layer Container property page provides a single, convenient location to set layer visibility and object selectability for one or more layers.

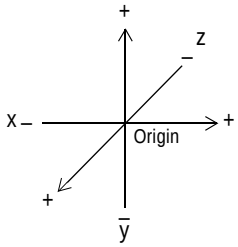
1. Choose **Layer** > **Layer Control**.

   The Layer Control property page appears, showing a grid. The first column lists all the layers in your scene; the second column lets you set layer visibility in the viewports; the third column lets you set layer visibility in the render region; the fourth column lets you specify whether or not objects in the layer can be selected.

2. To set a layer's visibility or selectability, click on a column cell. The selected cell is indicated by a checkmark. To deselect a cell, click it a second time.

Chapter 6 Working in 3D Space

Coordinate Systems



Cartesian Coordinates

One essential concept that a first-time user of 3D computer graphics should understand is the notion of working within a virtual three-dimensional space using a two-dimensional user interface.

To represent pictorial reality, 3D computer software uses the classical Euclidean/Cartesian mathematical representation of space. To represent the geometry of an object, the software uses the Cartesian coordinate system based on three perpendicular axes, X, Y, and Z, intersecting at one point. This reference point is called the *origin*. You can find it by looking at the center of the grid in any of the display windows.

XYZ Axes

To remember the direction of the X, Y, Z axes, use the "right-hand" rule: hold up your right hand so that your palm is facing you, then extend your thumb to the right, hold your index finger up, and point your middle finger towards you. Your thumb is pointing in positive X, your index finger in positive Y, and your middle finger in positive Z. The point of origin is 0, 0, 0. The opposite directions represent negative X, Y, and Z.

A small icon representing the three axes and their directions is shown in the corner of each viewport in the camera, geometry, and spotlight views.

XYZ Coordinates

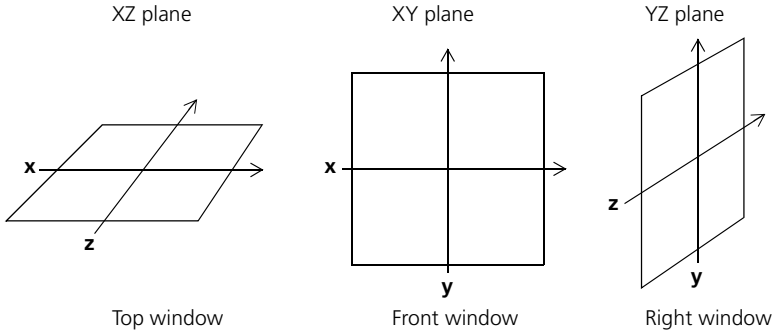
With the Cartesian coordinate system, you can locate any point in space using three coordinates. For example, if X = +6, Y = –6, Z = +6, a point would be located to the right of, below, and in front of the origin.

XZ, XY, YZ Planes

Since you are working with a two-dimensional interface, spatial planes are used to locate points in three-dimensional space.

The perpendicular axes extend as spatial planes: XZ, XY, and YZ. In the viewports, these planes correspond to three of the parallel projection windows: Top, Front, and Right.

Imagine that the XZ, XY, and YZ planes are folded together like the top, front, and right side of a box.



This helps give you to keep a sense of orientation when you are working within the parallel projection windows.

Global and Local Coordinate Systems

The XYZ coordinate system can be global or local.

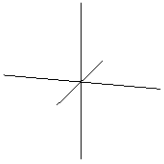
When you place an object in 3D space, it is inside a world, with origin at (0, 0, 0) of the reference grid in the windows. Accordingly, the XYZ coordinates locating the object in relation to the origin are called *global coordinates*.

A local coordinate system is thought of in terms of an object's own point of reference, which is its own center. This center also has three axes: X, Y, and Z. They are represented by color-coded vectors: red for X, green for Y, and blue for Z.

An easy way to remember the color coding is RGB = XYZ.

The center of an object is only a reference—it is not necessarily in the middle of the object because it can be relocated (as well as rotated and scaled).

Null Objects



Null object

A *null object* is an object that has no geometry. Because of this, it is not visible in a rendered scene and it cannot respond to any geometric modification, such as deformation.

Typically, nulls are used as assembly tools, especially useful in determining how objects will move in relation to each other or how material and texture is propagated through all or part of a hierarchy.

You could use any type of object to impose constraints or as nodes in a hierarchy. However, nulls are the best tools for the job because they have no overhead in modeling, rendering, or disk space.

Nulls are especially useful for occasions when you do not want any graphics associated with a function.

A null consists of a center represented by an icon. By default, it is displayed in the windows as three intersecting lines. The center cannot be dissociated from the null, unlike other object types that allow you to move their center inside or outside of them.

Other objects have control points added around their centers—points that define specific object types as polygon mesh, surface, or curve. No such points can be added to a null.

The null's cross icon can be easily mistaken for other objects of similar appearance. For example, if you use a null as a constraint on part of a skeleton, it can be difficult to distinguish it from the (similar-looking) chain roots and effectors, especially against the background of a grid. In this case, try using curved objects for the constraint, such as a linear circle (radius 1 or 2) or a short curve. While this takes up a little more disk space when saved, it makes selection easier and gives you all the abilities of a null.

Transforming Objects

An object in 3D space has three main characteristics, which fall under the term *transformation*. These are:

- Size (known as *scale*)
- Orientation (known as *rotation*)
- Position (known as *translation*)

When you create an object, SOFTIMAGE|XSI automatically initializes its parameters of transformation, locating the origin at the center of its world. During your work session, you will need to change these parameters to assemble objects and animate them. The conventions for using transformation are described as follows.

Scaling

The size of an object is thought of as its *scale*. When an object is created, its scale is automatically set to 1.0. An object is scaled from its center. The values related to scaling are called *factors*. A cube created at a length of 5 and scaled with a factor of 2 has a length of 10. A negative scaling yields an inverted object.

Rotation

The orientation of an object is set by the rotation of its center. All possible orientations can be set with a combination of three angles of rotation: X axis, Y axis, and Z axis. The reference axis for the rotation can be the global axis or its local axis.

Translation

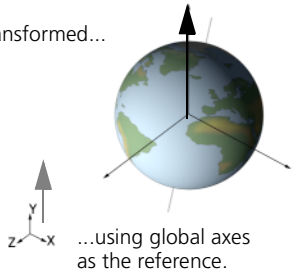
The location of an object is defined by its points in space. As mentioned earlier, a *point* is defined by three coordinates on the Cartesian axes X, Y, and Z. When you move an object, you are changing these coordinate values.

Transformation Modes

There are numerous ways, or modes, in which objects can be transformed.

Global

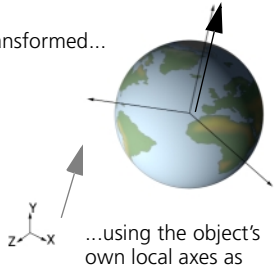
Object is transformed...



...using global axes as the reference.

Local

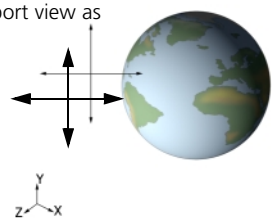
Object is transformed...



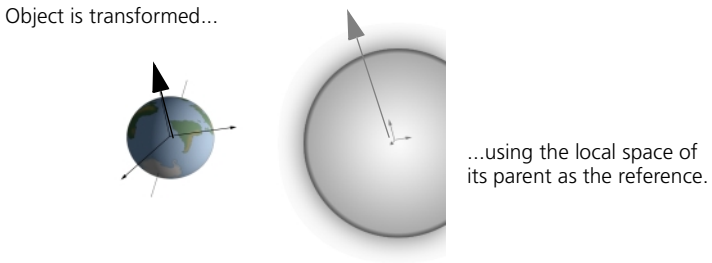
...using the object's own local axes as the reference.

View

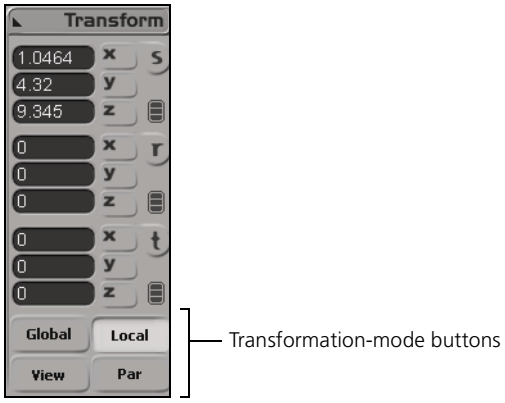
Object is transformed using the axes of the viewport view as the reference.



Parent



You choose which transformation mode to work in by clicking the transformation buttons in the Transform panel, shown below.

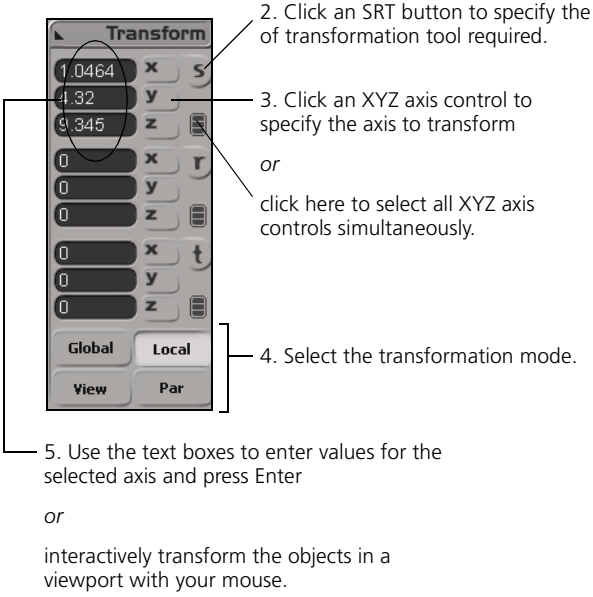


Note that, if you are rotating objects, the **Par** button changes to **Add**, allowing you to select additive mode for your rotations. In additive mode, numeric or interactive input does not result in values looping from –180 to 180 degrees; the rotation values continue to increase or decrease in one direction (positive or negative).

Transformation Methods

You can translate, rotate, and scale selected objects in a number of different ways. The methods described below let you transform object interactively or with greater precision through text-box input.

Transforming Objects in the Transform panel



SRT Text Box Input

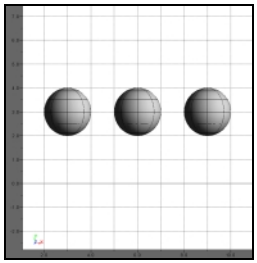
The SRT text boxes let you specify a transformation value for any selected object's scale, rotation, or translation axis.

Like other text boxes, SRT text boxes let you input absolute and relative values for one or more selected objects. For multiple objects, you can also input random values and specify a range of linear values.

The illustration below describes the different ways in which objects can be transformed using SRT text box input:

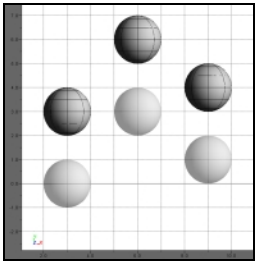
Absolute Input
Entering a numeric value in an XYZ text box repositions all selected objects to the specified position on their X,Y, or Z axes.

This example: **Translation: Y = 3**
(spheres all repositioned to 3 on the Y axis)



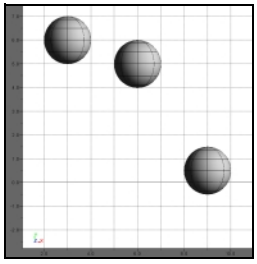
Relative Input
Entering (value)+ or (value)– in an XYZ text box increases/decreases the relative value of each selected object. You can also enter (value)* and (value)/ to multiply and divide the relative value.

This example: **Translation: Y = 3+**
(Y value of each selected sphere increased by three SOFTIMAGE units.)



Random Input
Entering a value of R(min,max) in an XYZ text box randomly repositions each selected object anywhere between the specified minimum and maximum values.

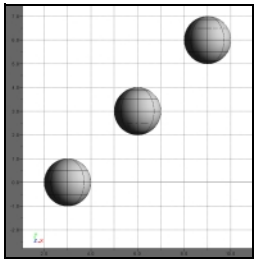
This example: **Translation: Y = R(0,6)**
(Spheres randomly repositioned between 0 and 6 on the Y axis)



Linear Range
Entering a value of L(min,max) in an XYZ text box repositions each selected object on a linear path between the specified minimum and maximum values.

The first selected object receives the minimum value and last selected object receives the maximum value.

This example: **Translation: Y = L(0,6)**
(Spheres repositioned equally on a linear range between 0 and 6 on the Y axis.)



Scaling

To scale objects interactively

1. Select one or more objects.
2. Click the scale button (**s**) on the Transform panel or press the **x** key on your keyboard.
3. Click and drag the mouse in a viewport to scale the objects about their individual centers. Drag to the right or up to increase the scale, and drag to the left or down to decrease the scale.
   - **Left-click and drag** to scale along the X axis (horizontally).
   - **Middle-click and drag** to scale along the Y axis (vertically).
   - **Right-click and drag** to scale along the Z axis (in depth).

To scale evenly along all axes, hold the Shift key and drag the mouse.

When you scale lights or cameras using the scale tool, you are only scaling the object's icon. To change a camera's properties, access its property page. To change a light's properties, access its property page or use the 3D manipulators, which are described in *Transforming with 3D Manipulators* on page 162.

To scale by numerical precision

- Type a value directly in the text box of any XYZ axis control and press Enter.

To simultaneously scale on two or three axes

- **Left-click** on the XYZ axis control to activate it prior to scaling.
- Press down any two (or three) mouse buttons (as described in Step 3, above) and drag the mouse.

Rotating

To rotate objects interactively

Rotation has no effect on infinite and point lights.

1. Select one or more objects.
2. Select either the **Global**, **Local**, or **View** button, depending about which center you want to rotate the object.
3. Click the rotate button (**r**) on the Transform panel, or press the **c** key.

4. Click and drag the mouse in a viewport to rotate the objects about their individual centers. Drag to the right or up to rotate clockwise with respect to the corresponding axis, and drag to the left or down to rotate counterclockwise.

   - **Left-click and drag** to rotate about the X axis.
   - **Middle-click and drag** to rotate about the Y axis.
   - **Right-click and drag** to rotate about the Z axis.

To rotate by numerical precision

- Type a value directly in the text box of any XYZ axis control and press Enter.

To simultaneously rotate on two or three axes

- **Left-click** on the XYZ axis control to activate it prior to rotation.
- Press down any two (or three) mouse buttons (as described in Step 4 above) and drag the mouse.

Translating

To translate objects interactively

1. Select one or more objects.
2. Click the translate button (**t**) on the Transform panel, or press the **v** key.
3. Select the **Global**, **Local**, or **View** button.
4. Click and drag the mouse in a viewport to translate the objects.

- If the selected referential is View, left-click and drag in any direction to translate the objects freely along the viewing plane (parallel to the viewport). If the selected referential is Global or Local, left-click to translate along the X axis, middle-click to translate along the Y axis, and right-click to translate along the Z axis.

  - **Shift+left-click and drag** left or right to translate the objects horizontally with respect to the viewport.
  - **Shift+middle-click and drag** up or down to translate the objects vertically with respect to the viewport.
  - **Right-click** to translate perpendicularly.

To translate by numerical precision

- Type a value directly in the text box of any XYZ axis control and press Enter.

To simultaneously translate on two or three axes

- **Left-click** on the XYZ axis control to activate it prior to translation.
- Press down any two (or three) mouse buttons (as described in Step 4 above) and drag the mouse.

Transforming Techniques: Transformation vs. Sticky Mode

There are two approaches to transforming objects: You can modify your objects temporarily in transformation mode while based in sticky selection mode, or vice versa. Each approach has its own benefits.

- Transforming objects in selection sticky mode allows you to quickly change selection in supra mode with the touch of the space bar. This is particularly useful when you have many objects in a scene whose scaling, rotation, or translation you want to individually tweak with an SRT tool.
- Transforming objects in transformation mode provides you with more flexibility when it comes to making extended selections (i.e., you can easily Shift or Ctrl+click to select multiple objects, then use the SRT tools to transform them as a group).

Method one: To transform objects based in selection sticky mode

1. Activate the Select tool in sticky mode (press and release the space bar).
2. Select an object in a viewport.
3. Press and hold the Scale, Rotate, or Translate supra key (**x**, **c**, and **v**, respectively). (A reminder: You activate supra keys by holding down the key and dragging the mouse without releasing the key.)
4. Transform the object interactively in the viewport with your mouse.
5. Release the supra key to revert back to selection mode.
6. In a viewport, select the next object to transform.

Method two: To transforming objects based in transformation mode

1. Select an object in a viewport, then activate a Scale, Rotate, or Translate tool (by clicking an SRT button or pressing/releasing the **x**, **x**, or **v** supra key).
2. Transform the object interactively in the viewport with your mouse.
3. Hold down the space bar to temporarily switch to selection mode.
4. In a viewport, select the next object to transform.
5. Release the space bar.

   SOFTIMAGE|XSI automatically reverts back to transformation mode.
6. Transform the object as you did in Step 2.

You can switch back to Select mode at any time by hitting the space bar.

Setting Transformation Tool Memory

If you are working primarily in transformation sticky modes, you can command SOFTIMAGE|XSI to remember the last transformation tool activated on each object.

Let's say, for example, you used the rotate tool on Object A, then went and used the scale tool on Object B. If you set transformation tool memory, then, the next time you select Object A in supra mode, the SRT controls will automatically switch you back to the rotate tool.

To set transformation tool memory

1. Choose **File** > **User Preferences**.
2. In the User Preferences dialog box, click the **Interaction** tab to display the Interaction property page.
3. Click **Activate last transformation tool used on each object** and click OK.

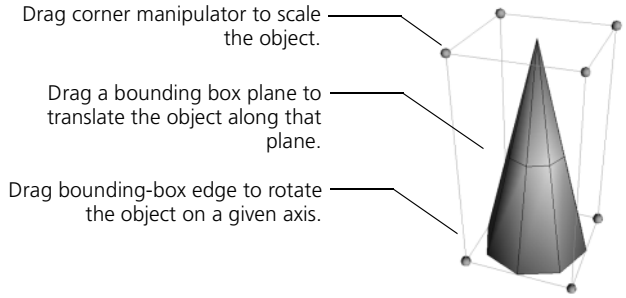
Transforming with 3D Manipulators

A 3D manipulator provides a set of handles that let you interactively manipulate objects in the viewports. Each type of object has its own 3D manipulators that let you control it in different ways.

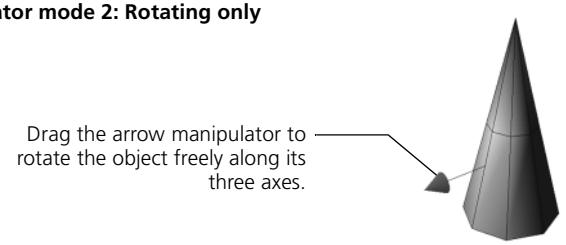
To activate a 3D manipulator

1. Select the object.
2. Press **b** on the keyboard or choose **Show** > **Manipulator Tool** from a viewport. Usually, the 3D manipulator is displayed as a box around the selected object.
3. Press the Tab key to alternate between the two types of manipulator handles, each of which let you perform different types of operations (see the illustration below).

Manipulator mode 1: Scaling, rotating & translating



Manipulator mode 2: Rotating only



Mode 1: Scaling

Position the mouse pointer over one of the 3D manipulator's corners:

- To scale uniformly about the object's center, click a corner and drag.
- To scale non-uniformly relative to the opposite corner, Shift-click and drag.
- To scale uniformly relative to the opposite corner, Ctrl-click and drag.

Mode 1: Rotating

Position the mouse pointer over one of the 3D manipulator's edges:

- To rotate about an axis parallel to the picked edge passing through the object's center, click an edge and drag.
- Hold down the Shift key to rotate relative to the opposite edge.
- Hold down the Crtl key to rotate the edge freely.

Mode 1: Translating

Position the mouse pointer over one of the 3D manipulator's planes:

- To translate in any direction along the plane, click the plane and drag.
- To translate along the picked plane's normals, Shift-click and drag.
- To translate freely in any direction, hold down the Crtl key and drag.

Mode 2: Rotating

Drag the arrow manipulator to rotate the object freely along its three axes.

Manipulating Cameras and Lights

You can translate, rotate, and scale cameras and lights (including area lights) using the 3D manipulators described above.

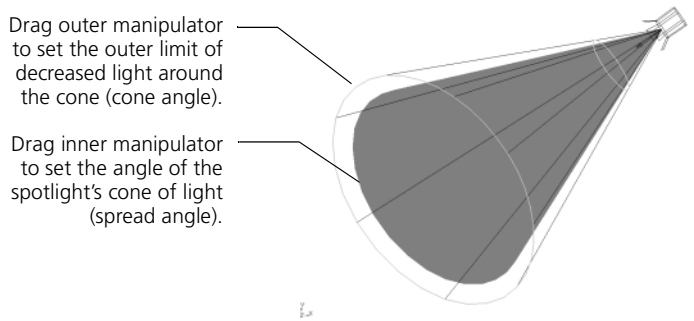
Scaling a camera or light only affects the size of the icon and does not change any of the camera or light properties. Scaling only has an effect when manipulating an area light's geometry. You can increase or decrease the size of the surface from which the area light rays emanate. For more information on area lights, see *Creating Soft Shadows with Area Lights* in Chapter 5 of the *Shaders, Lights & Cameras* guide.

Manipulating Spotlights

Spotlights have a third set of manipulators that let you control their start and end falloff, as well as their spread and cone angles.

You access these manipulators by clicking the **b** key and pressing Tab until these manipulators are displayed.

Spotlight manipulators



Setting Spotlight Falloff Values

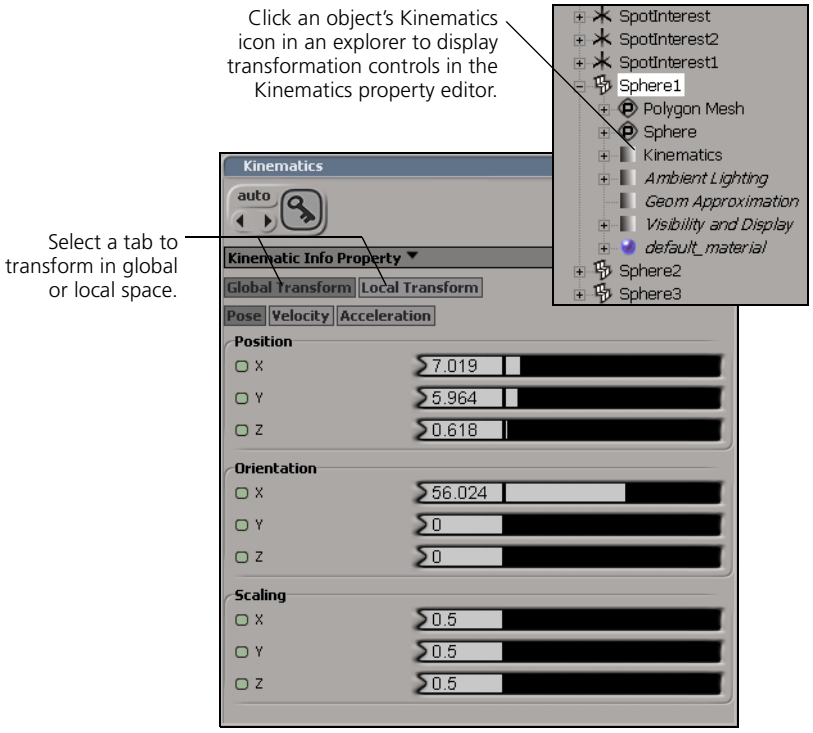
A spotlight's start and end falloff values specifies the point at which light intensity starts to decrease in all directions.

Crtl+click on the base of the cone to toggle falloff values.

- When falloff is on, drag the purple-red ring to modify the end falloff values and drag the light-blue ring to modify the start falloff values.
- **Ctrl+click** on the cone to reposition the start falloff to the current mouse position.
- **Shift+Ctrl-click** on the cone to reposition the end falloff to the current mouse position.

Transforming Objects from the Kinematics Property Editor

Each object has a set of transformation controls, found in its Kinematics property editor. The Kinematics property editor's transformation controls can be used to modify the selected object's scaling, rotation, and translation in X, Y, and Z in either local and global space.



Imposing Limits to Transformations

You can set maximum and minimum limits to both an object's position and rotation in local space. This is particularly useful for constrained objects. You can, for example, specify that a sphere used in a model of a head can rotate no more than 45 degrees to either side of its current position. By setting limits to its rotation, the sphere is prevented from exhibiting awkward or unnatural movement if it is constrained to another object.

To limit an object's position

1. From an explorer, open the object's Kinematic property editor.
2. Click the **Local Transform** tab, followed by the **Position Limits** tab.
3. Toggle the minimum and maximum position box for the object's X, Y, or Z axis.
4. For each check box you toggle, a corresponding control displays, allowing you to set the minimum position (–100 to 0) and maximum position (0 to 100) for that axis in SOFTIMAGE units.

To limit an object's rotation

1. From an explorer, open the object's Kinematic property editor.
2. Choose the **Rotation Limits** tab and toggle the minimum and maximum rotation box for the object's X, Y, or Z axis.
3. For each check box you toggle, a corresponding control appears, allowing you to set the minimum rotational angle (–360 to 0 degrees) and maximum rotational angle (0 to 360 degrees) for that axis.

Transforming an Object's Center

By default, an object's center is located at the center of its geometry. You can move an object's center if required.

To move an object's center to the center

1. Select an object.
2. Select the **Center** selection filter from the Selection panel.
3. Activate the translate tool (**t**) in the Transform panel.
4. Move the object's center interactively.

To move an object's center to the center of selected points or cluster

1. Select a series of points or a cluster.
2. Choose **Transform** > **Move Center to Vertices**.

   The object's center moves to the geometric center of the selected points or cluster.

You can move the center of multiple objects at once.

Example: To move the centers of ten cubes simultaneously, select all the bottom points of the cubes, then choose **Transform** > **Move Center to Vertices**.

Aligning Objects

You can reposition a group of objects so that the bottom, top, right, left, or center of their bounding boxes are aligned.

Aligning the bottoms of objects' bounding boxes along the X axis, for example, is a quick way to position a group of objects onto a flat surface, like a table.

To align objects

1. Select the objects to be aligned.
2. Choose **Transform** > **Align Objects**.

   The Align dialog box appears.
3. Toggle **Active** on to select each axis you wish to have the objects aligned to.
4. In the Position controls, choose how the alignment is to take place, where.

- **Global Minimum** aligns the bottom of all object bounding boxes with the object whose bounding-box bottom has the lowest axis value.
- **Global Middle** aligns objects according to the average mid point of all their bounding boxes.
- **Global Maximum** aligns the top of all object bounding boxes with the object whose bounding box top has the highest axis value.
- **First Minimum** aligns all object bounding-box bottoms with that of the first object selected.
- **First Middle** aligns the midpoint of all object bounding boxes with that of the first object selected.
- **First Maximum** aligns all object bounding-box tops with that of the first object selected.

Undoing a Transformation

If you make a mistake, you can remove any changes you made in scale, rotation, or translation values and return the object to its initial size, orientation, or position. You can undo transformations by doing any of the following:

- Choose **Edit** > **Undo Set** from the main-menu bar. This undoes the last operation you performed.
- Choose **Transform** > **Reset all Transforms** in the Transform panel. This returns the object selected back to the size, orientation, and location it had at the beginning of your work session.
- If you have done a series of transformations to an object, you can reset only the active transformation. Choose **Transform** > **Reset Active Transform** from the Transform panel. This resets the transformation back to its original values, corresponding to the current SRT tool.

  Let's say you rotated a light about the X axis and then scaled it by 2. If you want to undo only the rotation, click the rotate (**r**) button in the Transform panel or press the **c** key. This activates rotation mode. Choosing **Transform** > **Reset Active Transform** resets only the rotation you applied to your object and leaves the scaling as is.
- You can also retrace the same steps you followed when you edited your transformations. For example, if you have scaled the object, you can select it and use the mouse or the Scaling text fields in the Transform panel to reset any or all of scaling values to the default of 1. For rotation and translation, you can reset the values to the default of 0.

Spatial Deformations

One common way to create deformations is to alter the topology of an object by selecting points and dragging them to another location (e.g., dragging points inward around a model's mouth to create the mouth cavity).

Spatial deformation is another way to deform objects—rather than modify the object itself, spatial deformations change the very coordinates of space the object occupies.

Ordinarily, the spatial coordinate system can be visualized as a uniform grid where the X, Y, and Z axes are equally proportional to one another. Spatial deformations disrupt this uniformity, so that one or more axis is "stretched" out of proportion to the others. Any object located within that space is "warped" according to the modified coordinate system.

For more information about spatial deformations, see *Chapter 9: Spatial Deformations* in the *Modeling & Deformations* guide.

Chapter 7 Commands & Scripts

SOFTIMAGE|XSI is command based—anything you can do interactively in the interface, you can also accomplish with commands. As you work, the corresponding commands are logged to the command history. You can then repeat these commands or use them as the basis for a script.

A script is a set of commands that can be executed in sequence as if they were a single command. Simple scripts are a sequence of native SOFTIMAGE|XSI commands. More advanced scripts use a third-party host scripting language as the glue that holds the SOFTIMAGE|XSI commands together—with a scripting language, your scripts can use variables, constants, conditional statements, loops, and procedures.

The possible uses of scripts are endless. For example, you can:

- Create scripts to automate repetitive tasks.
- Create custom commands. Custom commands are scripts that have been packaged and added to a toolbar for quick access to a command or sequence of commands.
- Interact with other programs. For example, you can send e-mail when a render is finished, or import and export data from a spreadsheet.
- Run scripts in batch mode, without invoking the interface.
- Replay all the commands issued during a session.

There are several features for working with commands and scripts:

- The command box is described on page 174.
- The script editor is described on page 175.
- The command log file is described on page 187.

What Is a Command?

Commands are actions that have an immediate effect. Whenever you do something that changes the scene data—for example, select a scene element, move an element, change a parameter value, or set a key—a command is logged to its command history. When this happens, the corresponding command appears in the Command Box in the lower left corner of the interface:

In this respect, commands are different from tools. For example, when you activate the Translate tool by clicking the Translate (**T**) button on the Transform panel, no command is logged to the history: it is only when you actually translate something that a command is logged. Similarly, no command is logged when you switch from Local to Global transformation mode.

Actions that affect only the interface are not logged, either. For example, nothing is logged when you open or close a view, or when you maximize a viewport.

Usually, any command that is logged can be undone with the **Edit > Undo** command available on the main-menu bar. There are some exceptions to this, including the **DeleteAll** command.

Command Levels

Commands can be divided into high-level and low-level ones:

- High-level commands correspond to the commands that are available in the interface: these are the commands that get logged as you work in.
- Low-level commands correspond to what is happening behind the scenes. Often, a high-level command in the history is actually several low-level commands executed in sequence. Low-level commands are especially useful when you are writing your own scripts from scratch rather than simply repeating commands from the history.

Anatomy of a Command

SOFTIMAGE|XSI logs commands using the current scripting language as set in your user preferences. For example, select an object, press the **v** key to activate the Translate tool, then move the object around in View mode. If you are using the default scripting language of VBScript, a line similar to the following is logged:

```
Translate , 2.996, 1.887, -2.635, siRelative, siView, siObj, siXYZ
```

The first word is the command name, in this case **Translate**. After the command name come the arguments, in order and separated by commas. You can get help on the syntax for all the commands and arguments from the script editor—see page 177.

In the Translate example, notice that the first comma is preceded by a space but no argument: this means that the first argument has been omitted. If an optional argument is omitted, its default value is used. In the case of the Translate command, the first argument specifies which object to translate and the default value is the current selection. The next three arguments specify the translation values in the X, Y, and Z directions. The remaining arguments specify options like the current transformation mode.

All measurements are in SOFTIMAGE units, not the display units set in your user preferences.

When constructing or deconstructing a SOFTIMAGE|XSI command, be aware of the following VBScript syntax conventions:

- Commands do not use parentheses for their arguments. The exception is when you want to use a return value; for example, when using **GetValue**().
- If an optional argument is omitted, the default is used. If you omit an optional argument in the middle of a line, you must use a comma as a place-holder. If you omit one or more optional arguments at the end of a line, you do not need commas.
- Literal strings must always appear inside double quotes.
- There is no special character needed at the end of a command.
- You can include multiple commands on a single line by separating them with colons.
- Any text after a single quote (') is a comment and ignored by the interpreter.
- You can continue a long line on the next line by adding an underscore character (_) to the end of the first line.

Using the Command Box

The command box is displayed in the lower-left corner of the SOFTIMAGE|XSI interface. As you work, the Command Box shows the scripting equivalent of the last command you issued.

You can use the Command Box to type commands or repeat a recent command.

Typing Commands

You can enter a command by typing it in the Command Box. For example, you can modify the parameters of the last command before running it again or type an entirely different command.

To enter a command, simply click in the Command Box and type it. The command is executed when you press Enter. To cancel, press the Esc key or click outside of the Command Box.

When typing a command, use the syntax of the scripting language set in your preferences, which is VBScript by default. For more information about scripting languages, see page 188.

Repeating Recent Commands

The pop-up list next to the Command Box stores the last 25 commands. You can use this list to quickly repeat a recent command:

1. Click on the triangle arrow next to the Command Box.
2. Select a command from the list of recent commands that is displayed. The command is placed in the Command Box but not executed yet.
3. Change the command as desired by typing, then press Enter to execute it.

   If you change your mind and don't want to execute the command, simply move the mouse pointer away from the Command Box. Its contents will be replaced by the next command you issue through the interface.

Using the Script Editor

The script editor is a window that lets you create, modify, run, and manage scripts.

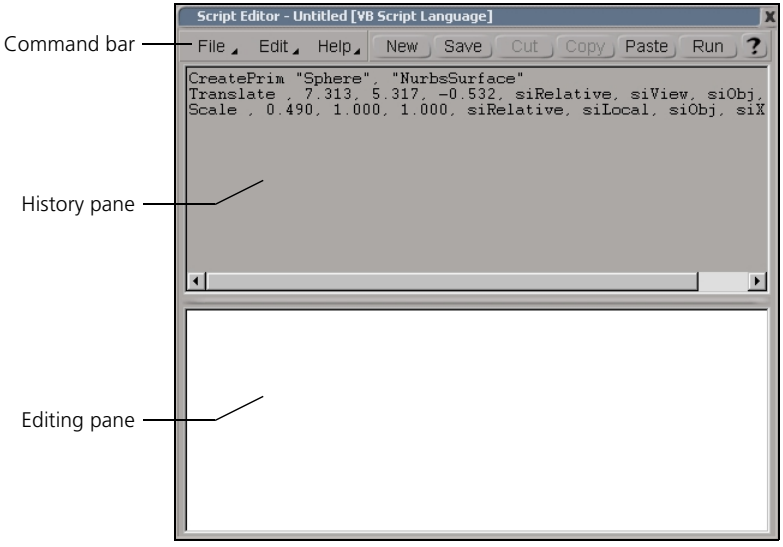
Opening the Script Editor

To open the script editor, click the Script Editor icon (!) on the Scripting panel at the bottom of the main window.

Another way to display the script editor is to choose **View** > **Views** > **Script Editor** from the main-menu bar.

Script-Editor Interface

The script editor includes a command bar, history pane, and editing pane.



- The **command bar** contains a variety of menus and commands.
- The **history pane** contains the most recently used commands in your current session. You cannot edit the text in the history pane, but you can use it as a source for pasting into the editing pane. By default, the history pane keeps the last 200 commands, but you can change this number—see *Setting Scripting-Editor Preferences* on page 181.
- The **editing pane** is a text editor in which you can create scripts by typing, cutting, copying, and pasting. To clear it, click **New** on the command bar.

You can resize the history and editing panes relative to each other by clicking and dragging on the bar between them.

Editing Scripts

To create and edit a script, use the editing pane as you would any other text editor. You can cut, copy, paste, move, and type text. Use the history pane as a source for copying text or as a reference for command names and syntax.

As you work in the script editor, you can use the commands and options in the **Edit** and **View** menus on the command bar as well as in the pop-up menu that appears when you right-click. You can also use standard mouse and keyboard commands:

To do this...	Do this...
Select the entire contents of the editing pane.	Ctrl+a
Select a word in the history or editing pane.	Double-click on the word.
Select a line in the history or editing pane.	Triple-click on the line.
Move selected text.	Drag to new location.
Copy selected text.	Ctrl+drag to new location.
Cut selected text to the clipboard.	Ctrl+x
Copy selected text to the clipboard.	Ctrl+c
Paste text from the clipboard.	Ctrl+v
Undo the last edit.	Ctrl+z, or Alt+Backspace
Go to the next/previous word.	Ctrl+right/left arrow
Go to the beginning of the script.	Ctrl+Home
Go to the end of the script.	Ctrl+End
Extend the selection.	Shift+arrow keys, Ctrl+Shift+arrow keys
Select from the cursor to the beginning of the line.	Shift+Home
Select from the cursor to the end of the line.	Shift+End
Select from the cursor to the beginning of the script.	Ctrl+Shift+Home
Select from the cursor to the end of the script.	Ctrl+Shift+End
Toggle overtype mode on or off.	Insert

If the text you type overwrites existing text, it's probably because you have accidentally pressed the Insert key. Press the Insert key again to turn overtype mode off.

Finding and Replacing Text

You can find and replace text in the editing pane:

- To open the Find in Editing Pane dialog box, choose **Edit** > **Find** or press Ctrl+f.
- To open the Replace dialog box, choose **Edit** > **Replace** or press Ctrl+h.

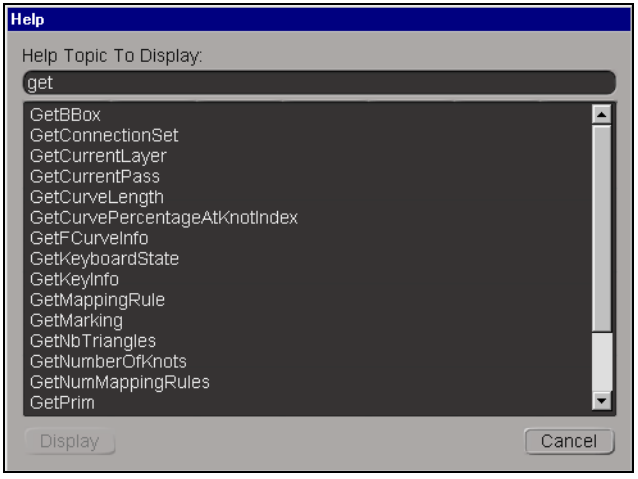
Getting Help on Commands

To get help on the commands, click the Help (**?**) icon on the command bar of the script editor. A list of available commands is displayed in your default HTML browser.

To get help on a specific command, you can also place the cursor in that command in the editing pane and press F1. Your default HTML browser should display help on that command—if it doesn't, it may be because of one of the following reasons:

- The command is misspelled.
- The cursor is in a keyword of the scripting language and not in a command.
- The command is not licensed for this installation of SOFTIMAGE|XSI.
- The command is a custom command and not a native SOFTIMAGE|XSI command. You can provide your own help for your custom commands as described on page 186.

If you know the beginning of a command name, you can type it and then press F1. A list of commands is displayed from which you can choose the command for you want information about. For example, if you type **get** and press F1, you'll see that the following commands are available:



Managing Script Files

The first time you open the script editor, the editing pane is empty. Once you have added commands to create a script, you can save the contents of the editing pane as a script file. The script editor also lets you open existing script files and create new ones. By default, script files are stored in the **Softimage\Scripts** subdirectory of your user directory.

You can share your scripts with other users on your network by saving them in a common, shared workgroup directory.

To save the contents of the editing pane to disk as a script file

1. Do one of the following:
   - Click the **Save** button on the command bar.

     or

   - Choose **File** > **Save** from the command bar.

     or

   - Press Ctrl+s.
2. If the script has not been saved before, the Save As dialog box prompts you to supply a file name. You should also make sure that the extension is the correct one for the language of the script. Once you supply a file name for the script, the name is displayed in the title bar of the script editor window.

To save the contents of the editing pane with a different name than the one displayed in the title bar, choose **File** > **Save As** from the command bar.

By default, scripts are saved as VBScript files with a **.vbs** extension. You can change this by choosing a different default scripting language as described on page 188.

To open a script that has been saved as a file on disk

1. Display the Open dialog box by doing one of the following:
   - Choose **File** > **Open** from the command bar.

     or

   - Press Ctrl+o.

   If the current contents of the editing pane have not been saved, you are prompted to save them. Click Yes or No as desired.
2. Select a script file, then click Open. Alternatively, you can double-click on a file name.

   The contents of the script file are displayed in the editing pane, and the name of the file is displayed in the script editor's title bar.

Another way to open a script file is to drag it from the browser or a folder window into the editing pane.

If you open a script file in a different language, it automatically changes the current scripting language that is set in your user preferences to the corresponding language.

Creating New Script Files

To clear the contents of the editing pane and begin working on a new script, do one of the following:

- Click the **New** button on the command bar.
- Choose **File** > **New** from the command bar.
- Press Ctrl+n.

When you create a new script file, you are prompted to save the contents of the editing pane. After you have clicked Yes or No, the editing pane is cleared and the name of the script is displayed as Untitled in the script editor's title bar.

In the script editor, you can also clear the contents of the editing pane by choosing **Clear Script Editor** from the **Edit** menu. However, this does not change the name of the script as displayed in the title bar of the script editor—accidentally choosing **Save** instead of **Save As** will overwrite the named file on disk.

Running Scripts

You can run a script directly from the script editor. In this case, only global code is executed; that is, code that is not within a defined procedure. Note that procedures are still executed if they are called from global code.

If one or more lines are selected when you run a script from the script editor, only those lines are executed. If nothing is selected, the entire contents of the editing pane are executed.

To run a script from the script editor, do one of the following:

- Click the **Run** button on the command bar.
- Press **F5**.
- Choose **Edit** > **Run** from the command bar.
- Right-click in the editing pane and choose **Run** from the pop-up menu.

Alternatively, see page 182 for information about creating a custom command to run a script from a button on a custom toolbar, or see *Batch Scripts* on page 193 for information about packaging and running scripts from a command prompt.

Terminating Scripts

To terminate a script while it's running, press Ctrl+Break (Ctrl+Pause on some keyboards). You may need to press this key combination a few times because the keyboard is only checked before each SOFTIMAGE|XSI command is executed.

Running Scripts Automatically

You can run scripts or any native SOFTIMAGE|XSI command automatically in two ways:

- Whenever an object is selected.
- Whenever you change frames.

In either case, you must first create a custom command for your script as described on page 182.

Running scripts automatically when an object is selected

To run a script or native SOFTIMAGE|XSI command automatically whenever an object is selected in a geometry view:

1. Choose **File** > **User Preferences** from the SOFTIMAGE|XSI main-menu bar. The User Preferences dialog box opens.
2. On the Interaction page, enter the command name and any parameters in the **OnSelectionChange Command** box.

   This setting is stored with your user preferences and is used in all scenes. For more information about user preferences in general, see *Setting User Preferences* on page 214.

Running scripts automatically when the current frame changes

You can select a script or native SOFTIMAGE|XSI command to run automatically whenever you change the current frame. This lets you use scripts to simulate "persistent" effects.

1. Choose **Playback** > **Playback Options** from the Playback panel below the timeline. The Play Control property editor opens.
2. Click the **Update** tab. In the **On-Frame-Change Script Command** box, enter the command name and any parameters. This setting is stored with the scene.

The **On-Frame-Change Script Command** setting is saved with the scene. If you set it to a custom command, make sure the custom command is available on all systems used to open the scene.

Setting Scripting-Editor Preferences

Your user preferences include the maximum number of recent commands listed in the history pane of the script editor.

To set the number of commands in the list

1. Choose **File** > **User Preferences** from the main-menu bar. The User Preferences dialog box opens.
2. Click the **Scripting/Logging** tab.
3. Specify the number of lines under **Command Log Size**, or select the **Unlimited** checkbox.

The other options on the **Scripting/Logging** page apply to the default scripting language (see page 188), the log file (see page 187), and real-time message logging (see page 196). For more information about preferences in general, see *Setting User Preferences* on page 214.

Custom Commands

A custom command is a script that has been packaged and added as a button on a custom toolbar. This lets you run the script quickly and conveniently by clicking the button.

While a script is simply a plain text file, custom commands contain the following information:

- A reference to the file containing the script to run. If you have not already saved your script as a file, it is automatically saved for you when you create a custom command. If the saved script file is modified later, clicking the button runs the modified script, not the original one.
- The command name to log in the history whenever the button is clicked, and to use when calling the custom command from other scripts.
- A description of the command. This description is displayed when you select the command in the Customize Toolbar dialog box.
- The language (scripting engine) to use to interpret the custom command.
- The procedure to call when running the custom command, if your script contains procedures.
- The values of any arguments, if your script requires them. You can also prompt users to supply values whenever the script is run.
- The text label to display on the button that runs the custom command.

Creating a Custom Command

To create a custom command and add it to a custom toolbar for the first time

1. Make sure that a custom toolbar is open.

   - To display an existing custom toolbar, choose it from the **View** > **Toolbars** menu available from the main-menu bar.

   - To create a new custom toolbar, choose **View** > **Toolbars** > **New Toolbar**.

   For more information about custom toolbars in general, see *Creating Custom Toolbars* on page 221.

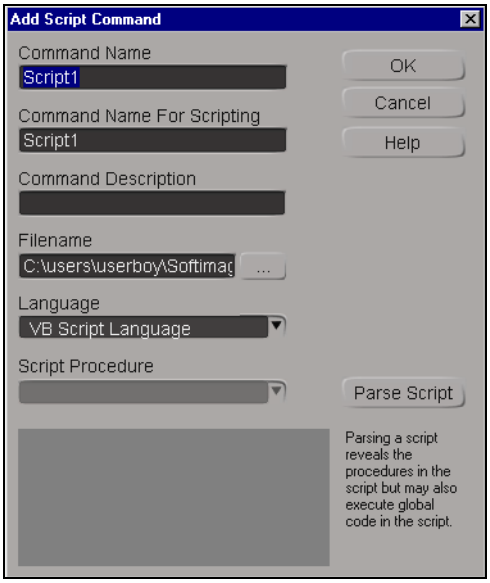
2. Do one of the following:

   - Select one or more lines in the history or editing pane of the script editor and drag them onto your custom toolbar. Note that text dragged from the editing pane is removed. To leave it in place, Ctrl+drag it onto a toolbar. You can also drag text from any program that supports drag-and-drop actions.

     or

   - Drag a saved script file from the browser onto your toolbar. On Windows NT, you can also drag from a folder window.

The Add Script Command dialog box appears.



3. Specify a **Command Name** for the script. This is the text that will appear as a label on the button.

   If you dragged lines from the script editor, the default command name is of the form Script*number*; if you dragged a saved script file, the default is the file name.

4. Specify a **Command Name for Scripting**. This is the name that gets logged to the command history when you click on the button. You can also use this name to invoke your custom command from within another script. The command name for scripting cannot use spaces or punctuation.

5. You can also provide a brief **Command Description**. This description is displayed in the Customize Toolbar dialog box if you later add the command to yet other toolbars as described in *Adding a Single Custom Command to Multiple Toolbars* on page 185. It is also displayed in the Keyboard Mapping dialog box—see *Mapping Custom Commands to Keys* on page 185.

6. Specify the file name for the script. If you dragged lines from the script editor, a default file name is supplied. If you dragged a saved script file, it is copied to the **Softimage\Scripts** subdirectory of your user directory. In either case, you can change the name and directory.

7. If necessary, select the scripting language from the pop-up list in the **Language** box. The box lists the supported languages that are installed on your computer.

If you dragged lines from the script editor onto the toolbar, the default language is the one specified in your preferences—see *Scripting Languages* on page 188 for more information.

If you dragged a saved file, the default language is based on the file name extension: **.vbs** for VBScript, **.js** for JScript, **.pls** for PerlScript, and **.pys** for Python ActiveX Scripting Engine.

8. If your script does not contain any procedures, there is nothing more to do—click Ok to close the Add Script Command dialog box and add the button to your toolbar.

   If your script does contain procedures and you want to call them instead of global code, continue with the rest of the procedure.

9. Click the **Parse** button. This parses the script and "finds" the procedures and arguments.

Parsing a script may execute global code; that is, any code that is not within a defined procedure. In such cases, it will also run procedures that are called from global code.

10. If your script contains procedures, you can specify which procedure to execute when the button is clicked. Select the desired procedure from the pop-up list in the **Script Procedure** box. If you do not specify a procedure, only global code is executed, as well as any procedures that are called from global code.

Even when a procedure is specified, global code may be executed before the procedure is called. This is a side-effect of parsing the script with some scripting engines. To be certain that your script behaves predictably in all situations, do not mix global code and procedures.

SOFTIMAGE|XSI cannot directly call procedures in PerlScript or Python. However, procedures called from global code or other procedures work properly.

11. If the selected procedure contains arguments, they are listed under **Parameters**. For each argument, specify the value to use when the script is run by doing one of the following:

    - Enter a value in the corresponding **Value** box. This value is used when you click the button to run the script.

      or

    - Click twice in the **Value** box and select **Prompt On Run** from the pop-up list. This option prompts you to enter a value in a dialog box when you click on the button to run the script.

12. Click OK. The Add Script Command dialog box closes and the new button is added to your custom toolbar.

Running and Undoing Custom Commands

To run a custom command, simply click the corresponding button on the toolbar. The command name for scripting gets logged to the history, together with any parameter values.

If the script contains only commands that can be undone (for example, if it does not contain commands like **DeleteAll**), then you can undo the result of your custom command by choosing **Edit** > **Undo** from the main-menu bar.

Adding a Single Custom Command to Multiple Toolbars

Once you have created a button for a custom command by dragging a script onto a toolbar, you can add the same custom command to other toolbars.

1. Display the toolbar to which you want to add the custom command.
2. Right-click on the toolbar, then choose **Customize** from the pop-up menu. The Customize Toolbar dialog box is displayed.
3. From the **Category** list, select **Custom**. The **Available Commands** box lists all your available custom commands. Click on a command to see its description.
4. Select a custom command from the **Available Commands** list and drag it onto your toolbar.

If you drag a script directly onto another toolbar instead of following this procedure, you must provide a unique command name for scripting. This means that different commands will be logged when you click on the different buttons. The exception is when the script contains procedures and you specify the same procedure for both custom commands.

Mapping Custom Commands to Keys

You can also map a custom command to a key on your keyboard. This lets you run the custom command simply by pressing that key. For more information about mapping keys, see *Creating Keyboard Shortcuts* on page 215. Your custom commands are listed in the Custom Script Commands group.

Editing Custom Commands

Once you have created a custom command button by dragging a script onto a toolbar, you can easily open and modify the script in the script editor.

1. Do one of the following:
   - Right-click on the custom command button and choose **Edit Script** from the pop-up menu.

     or

   - Open the script editor and drag the button into the editing pane.
2. A message box asks whether you want to save the current contents of the script editor's editing pane. Click Yes or No as desired.
3. The script file opens in the editing pane and its name is displayed on the title bar. You can now modify the script.
4. When you have finished, do not forget to save your changes.

If you need to parse the script again or change any of the values you set in the Add Script Command dialog box, you must first remove the button and then drag the script back onto the toolbar.

Providing Help for Your Custom Commands

You can provide help for your custom commands by creating an HTML file and adding it to the **htmldoc\commands** subdirectory where SOFTIMAGE|XSI is installed. The HTML file must have the same name as the command name for scripting, as well as the **.html** extension. You can copy one of the existing HTML files as a template.

To display your help, place the cursor inside your custom command's command name in the editing pane of the script editor, then press the F1 key. Note that the custom command is not listed in the left frame.

The Command Log

In addition to maintaining a history of commands used in the current session, you can save a log of commands to disk. This feature is sometimes called "journaling."

You can use the command log for a variety of purposes. For example:

- If you imported a scene from SOFTIMAGE|3D or another program to apply textures and lighting before you animated the scene, you can copy the contents of the log into the script editor to help redo your SOFTIMAGE|XSI commands after the scene is animated.
- You can use the log to help recover your work after a system crash.

If you want to keep a log file, make sure you rename it before you restart SOFTIMAGE|XSI. You cannot access the file while SOFTIMAGE|XSI is running, and the file is overwritten once you begin working in a new SOFTIMAGE|XSI session.

Activating the Command Log

The command log is controlled by your preferences. To activate logging:

1. Choose **File** > **User Preferences** from the main-menu bar. The User Preferences dialog box opens.
2. On the **Scripting/Logging** page, select the **Activate** option under **Log File**.
3. If desired, you can also specify a path and file name in the **Log File Path** box.

Intermediate and Advanced Scripting

Simple scripts involve the sequential repetition of SOFTIMAGE|XSI commands. More advanced scripts, however, require the use of variables, conditional statements, loops, and subprocedures. None of these are provided: instead, you can use existing scripting languages as the glue that holds the native SOFTIMAGE|XSI commands together. If you already know a supported scripting language, you do not need to learn a new one to create advanced scripts in SOFTIMAGE|XSI.

Using a scripting language, you can create arguments for your scripts and interact with other applications. You can also use third-party tools to test and debug your scripts.

Scripting Languages

Several popular scripting languages are supported. To use a scripting language with SOFTIMAGE|XSI, you must first install the scripting engine for that language.

Supported Scripting Languages

Although commands are logged in its history using VBScript syntax, you can write and run scripts using any language that is ActiveX-compliant. ActiveX is a technology for sharing data between programs. Examples of ActiveX-compliant scripting languages include:

- VBScript (the default)
- JScript
- PerlScript
- Python ActiveX Scripting

VBScript

VBScript is the default scripting language used by SOFTIMAGE|XSI. The SOFTIMAGE|XSI installation automatically includes VBScript on both IRIX and Windows NT, if it is not already installed. You can also download the Windows NT version and obtain further documentation from **msdn.microsoft.com/scripting/**

JScript

JScript is an ActiveX-compliant version of the JavaScript scripting language. The SOFTIMAGE|XSI installation automatically includes JScript on both IRIX and Windows NT, if it is not already installed. You can also download the Windows NT version and obtain further documentation from **msdn.microsoft.com/scripting/**

PerlScript

PerlScript is a version of the Perl scripting language that supports ActiveX. It is included in ActivePerl, which is available for Windows NT and can be downloaded from **www.activestate.com/activeperl**

SOFTIMAGE|XSI cannot directly call procedures in PerlScript. However, procedures called from global code or other procedures work properly.

Python ActiveX Scripting

Python ActiveX Scripting is a version of the Python scripting language that supports ActiveX. It is available for Windows NT and can be downloaded from **www.python.org/windows/**

SOFTIMAGE|XSI cannot directly call procedures in Python. However, procedures called from global code or other procedures work properly.

Setting Your Preferred Scripting Language

Once you have installed an ActiveX scripting engine, you can set it as your preferred scripting language. The preferred scripting language is used to interpret commands when running them directly from the script editor or Command Box. It also determines the default extension when saving script files.

In addition, the preferred scripting language is used to log commands to the history if a deparser is available. If a deparser cannot be found when SOFTIMAGE|XSI starts, commands are logged in VBScript. If you change scripting languages during a session and a deparser cannot be found, the previous language is used for logging.

To set your preferred scripting language:

1. Choose **File** > **User Preferences** from the main-menu bar. The User Preferences dialog box opens.
2. On the **Scripting/Logging** page, select an option from the **Scripting Language** pop-up list.

If you open a script file in a different language, it automatically changes the current scripting language that is set in your user preferences to the corresponding language.

The Scripting Language box lists the ActiveX scripting engines that are installed on your computer. If you just installed an engine and it is not listed, restart SOFTIMAGE|XSI. If it still isn't listed, restart your operating system.

When adding a script to a toolbar, you can specify any installed language even if it is different from your preferences—see *Creating a Custom Command* on page 182.

Scripting-Language Features

You can use the commands, functions, operators, and statements of a scripting language to create complex scripts. In this way, scripts can be more than a simple, sequential repetition of SOFTIMAGE|XSI commands.

All scripting languages include the certain basic features, although the specific implementations are different. These basic features include:

- The ability to declare and use variables and constants.

  Note that variables are not global to the scene and cannot be shared between scripts—however, you can use custom parameters to share data between scripts. For more information, see *Chapter 7: Custom Parameters* in the *Animating* guide.

- Statements that control the flow of execution, including conditionals (**If… Then… Else…** or **Select Case**) and loops (**For… Next** or **While… Wend**).

- Functions that convert between basic data types, such as numbers and strings. Many languages convert data types automatically, based on the context.

- The ability to define procedures and functions.

- The ability to provide simple interaction with users, including message and input boxes. Note that if you want to pass data to your scripts, you should use arguments instead of input boxes—see the next section.

For more information about any of these features, refer to the documentation for your scripting language.

Arguments

A mechanism for passing arguments to scripts is provided. When arguments are passed using this mechanism, they are logged to the command history, just like with any native SOFTIMAGE|XSI command.

To use this mechanism to pass arguments to your script, put your script inside a procedure and declare the arguments. For example, in VBScript:

```
Sub myproc(myargname1, myargname2)
   ...
End Sub
```

When you create a custom command for the script as described on page 182, make sure to press the **Parse** button and then select the procedure in the **Script Procedure** box. You can then set the arguments to specific values or to **Prompt on Run**.

Make sure to give meaningful names to your arguments. When your custom command runs and prompts for values, the names are the only clues to the arguments' functions.

SOFTIMAGE|XSI cannot directly call procedures in PerlScript or Python. However, procedures called from global code or other procedures work properly.

Interacting with Other Applications

SOFTIMAGE|XSI scripts can interact with any other program that supports ActiveX. This means that the possibilities are endless. For example, you can send e-mail, import and export data from spreadsheets, or do anything that you can do with another ActiveX-compliant program. For specific details on the objects and methods used by other programs, refer to their documentation.

Example: Sending E-mail

The VBScript code that follows can be used to send an e-mail message using Microsoft Outlook. It could be included in another script; for example, to send yourself a message when a rendering job is finished.

```
sub msg( dst, subj, body )

dim a, b

set a = createobject( "outlook.application" )
set b = a.createitem( olMailitem )

b.to = dst
b.subject = subj
b.body = body

b.send()

end sub


msg "jdoe@acme_inc.com","Your render","has finished."
```

Some pager services will forward e-mail messages to your pager. If you subscribe to such a service, you can adapt this script to send a message to your pager.

Debugging Scripts

There are a few simple features to help you debug your scripts. In addition, you can use the Microsoft Script Debugger to interactively debug VBScript and JScript. Debugging utilities are also available for Perl and Python.

SOFTIMAGE|XSI Debugging Features

When the scripting engine cannot interpret a command in a script, the execution halts. If you are running the script from the script editor, the line with the problem is outlined in red. If you are running a custom command from a toolbar, the custom command's scripting name is displayed in the history using red text.

In addition, any error messages from the script engine are echoed in the history pane. For example, the VBScript engine provides error messages such as the following:

```
'ERROR : "Unterminated string constant - [line 1]"
```

Debugging VBScript and JScript

You can debug VBScript and JScript interactively using the Microsoft Script Debugger. You can download and install the Microsoft Script Debugger using the instructions available at **msdn.microsoft.com/scripting/**

If you installed VBScript from this site, you may already have the Microsoft Script Debugger installed.

Once you have installed the Microsoft Script Debugger, it is automatically invoked whenever there is an error running VBScript or JScript. You can also invoke it manually using the command **stop** in VBScript, or the command **debugger** in JScript.

In the Microsoft Script Debugger, you can step through your script and set breakpoints. You can also use the Command window to view and set variable values on the fly. In addition, the Call Stack is useful when you are debugging scripts called from within scripts.

For more information about using the Microsoft Script Debugger, refer to the documentation included with it.

Debugging Perl

Perl includes its own debugger. For more information, see the documentation included with Perl. In addition, front ends and alternative debuggers are available at **reference.perl.com/query.cgi?section=debug**

Debugging Python

Python includes its own debugger, **pdb.py**. For more information, see the documentation included with Python.

Batch Scripts

Scripts can be run in batch mode from a shell or command prompt window, without invoking the interface. This is also known as "command line scripting." You can specify the procedure to run, as well as supply any necessary arguments. Before running a script in batch mode, you may need to prepare it first.

Preparing Scripts for Batch Mode

In batch mode, scripts cannot require any interactive input such as prompting for a file or path—this does not include property editors that open automatically as you work in the interface. For example, the following code would generate an error if it is run in batch mode. This is because the **OpenScene** command, when invoked without any arguments, displays a dialog box that prompts for a scene to import.

```
sub MyBatchScript()
   OpenScene                        ' error
end sub

' Main
MyBatchScript
```

There are two ways to deal with this. The first way is to explicitly provide all arguments in the script. For example:

```
OpenScene "C:\MyProject\MyScene.scn", false
```

In this case, the **false** argument is a flag that indicates that there should be no prompt to save changes to the current scene. There is a similar flag for **DeleteAll**.

The other way is to declare any necessary parameters as arguments in a procedure declaration and then pass them on the command line when the script is run in batch mode. For example, in VBScript you would declare the arguments for a procedure as follows:

```
Sub myproc(myargname1, myargname2)

   ...

End Sub
```

At a command prompt, you would then run the script as shown below (it should all be on one line):

```
xsi -script myscriptfile.vbs -main myproc
                          -args -myargname1 myargvalue1
                                -myargname2 myargvalue2
```

Running Scripts in Batch Mode

To run a script in batch mode, you must set the environment and then launch SOFTIMAGE|XSI and the script.

Setting the Environment

Before running a script in batch mode, you should first set the environment properly:

- On Windows NT, open a command prompt. To do this, choose **Programs** > **SOFTIMAGE Products** > **SOFTIMAGE XSI 1.0** > **Command Prompt** from the Windows NT **Start** menu.
- On IRIX, open a shell and source the **.xsi_1.0** XSI resource file.

Launching SOFTIMAGE|XSI and a Script

To launch the program and a script in batch mode, start SOFTIMAGE|XSI with the **-script** switch, specifying your script file name. There are several ways to do this, depending on whether you need to specify the scripting language, specify the procedure to run, or supply values for arguments.

Running simple scripts in batch mode

To run a simple VBScript file named **myscript.vbs**, use the following syntax at the command prompt:

```
xsi -script myscript.vbs
```

Specifying the scripting language in batch mode

The scripting language is determined by the file name extension according to information in the registry: by default, this is **.vbs** for VBScript, **.js** for JScript, **.pls** for Perl, and **.pys** for Python. If your script file uses a different extension, you can specify the language explicitly with the **-lang** switch. For example, to run a script file named **myscript.xxx**, use one of the commands below:

```
xsi -script myscript.xxx -lang VBScript
xsi -script myscript.xxx -lang JScript
xsi -script myscript.xxx -lang PerlScript
xsi -script myscript.xxx -lang Python
```

Specifying a procedure in batch mode

By default, when you run a script in batch mode, only global code is executed. If your script contains procedures, you can use the **-main** switch to specify which procedure to run. For example, to run a procedure named **myproc** in a VBScript file named **myscript.vbs**, use the following syntax:

```
xsi -script myscript.vbs -main myproc
```

Even when a procedure is specified, global code may be executed before the procedure is called. This is a side-effect of parsing the script with some scripting engines. To be certain that your script behaves predictably in all situations, do not mix global code and procedures.

SOFTIMAGE|XSI cannot directly call procedures in PerlScript or Python. However, procedures called from global code or other procedures work properly.

Supplying arguments in batch mode

If your procedure requires arguments, you can specify them after the **-args** switch. For example, if the procedure **myproc** in **myscript.vbs** requires two arguments named **myargname1** and **myargname2**, you can run the procedure and set the argument values with the following syntax (it should all be on one line):

```
xsi -script myscript.vbs -main myproc
-args -myargname1 myargvalue1 -myargname2 myargvalue2
```

If you specify arguments in batch mode, you must also specify a procedure.

Real-Time Message Logging

When running scripts, you can set your preferences to allow for real-time message logging. This is particularly useful when rendering in batch mode. To activate real-time message logging:

1. Choose **File** > **User Preferences** from the main-menu bar. The User Preferences dialog box opens.
2. On the **Scripting/Logging** page, set the **Real-Time Message Logging** option as you want it:
   - When this option is off (default), commands that involve picking sessions are properly logged after all the inputs have been picked.
   - When this option is on, messages are logged as they occur. This lets you see, for example, messages about each frame as it is rendered rather than after all frames are rendered.

You can also get and set this preference using the **"RealTimeMessageLogging"** string and the **GetUserPref** and **SetUserPref** commands. For example, the following VBScript script stores the current preference, then activates real-time message logging, renders the current pass, and finally restores the original preference:

```
Dim flag

flag = GetUserPref("RealTimeMessageLogging")

SetUserPref "RealTimeMessageLogging", True

RenderPass

SetUserPref "RealTimeMessageLogging", flag
```

Scripting Tips and Tricks

This section contains a variety of techniques and recipes for using SOFTIMAGE|XSI commands in scripts.

All examples are written in the VBScript language.

General Tips

Logging Messages

As you build scripts, you can use **LogMessage** to uncover information instead of using the HTML pages on commands and objects. For example, if you need to know the types of the current selection:

```
set MySel = GetValue("SelectionList")
for each myItem in MySel
   LogMessage myItem & " is a " & myItem.Type
next
```

For more information about the SelectionList, see *Selection* on page 204.

Running Parts of Scripts

As you build, test, and refine scripts, you can choose to run only part of them. This lets you concentrate on a particular portion of your script. There are a couple of ways to do this:

- "Comment out" lines that you do not want to run. To do this in VBScript, add an apostrophe (`'`) at the beginning of the line. When you want to run the line again, remove the apostrophe.
- To run a contiguous block of lines, select them in the editing pane before running the script. If text is selected, only the selection is executed. Be careful to select complete lines.

Declaring Variable

It's a good idea to declare variables before using them. Not only does it make your scripts a bit faster, it also makes it easier to find and fix problems in your code. For example, to declare variables in VBScript:

```
dim myVar1, myVar2, myVar3
```

Returning Values

Any SOFTIMAGE|XSI command that creates something also returns it. You can use the return value to make your scripts reusable. For example, the following line creates a primitive surface sphere, and tries to name it **sphere**:

```
CreatePrim "Sphere", "NurbsSurface", "sphere"
```

The new sphere could be named **sphere**, **sphere1**, or **sphere127**, depending on how many other spheres were already in the scene. Because of this, you cannot rely on an object's name when scripting.

The solution is to use the return value of the command. Some commands return numbers or strings. You can get these return values using a variable and the assignment operator. In these cases you must use parentheses around the command's arguments, for example:

```
myVar = GetValue("sphere.kine.posx")
```

Other commands return objects. To work with these, use the **set** command in VBScript or the equivalent command in another language. Again, you must use parentheses around the arguments of commands.

```
set myVar = CreatePrim("Sphere", "NurbsSurface")
```

Still other commands use output parameters. To work with these, you pass a variable as an argument to the command. Once the command has been executed, the variable contains the returned value. For example:

```
ApplyOp "Extrusion",,myVar
```

Once you have the return value stored in a variable, you can use it in the rest of your script:

```
msgbox myVar

SelectObj myVar

Translate myVar, 6.085, 3.164, -0.316
```

Testing Returned Variables

When writing and testing scripts, it is often useful to check that a returned object variable refers to an object. There are commands in host scripting languages that help you do this. For example, in VBScript you can use **TypeName**()—if the type name is not **Nothing**, then you know that the variable refers to something. For example:

```
sub AddMeatToBone ( Radius )
   'Filter The Current Selection for Chains
   SelectChildNodes
   set selList = SIFilter(, "Chain_Element")
   if TypeName(selList) = "Nothing" then
      LogMessage "You might want to pick a Chain..."
      Exit Sub
   end if
```

Reserved Words

When scripting, be careful not to use words that are reserved by the host scripting language as variables names, object or parameter names in the scene, and so on. For example, "type" is a reserved word in VBScript. At first it might not be obvious why the following fails:

```
set return = SomeCommand(arg1, arg2, type)
```

If you declared the variable, this would be the line that fails:

```
dim type
```

Limitations on IRIX

There are currently certain limitations with scripting on IRIX:

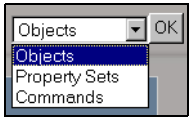
- You cannot create a Scripting.Dictionary object.
- You cannot access drives, folders, or files.
- You cannot perform text input/output.
- When using the VBScript **split**() command, use a large number for the **count** argument. For example:

  ```
  parsed = split(textstring, ".", 2000)
  ```

Additional Information

HTML documentation

To obtain information in HTML format, click the Help (**?**) icon on the command bar of the script editor. To display help on different subjects, select an option from the drop-down list in the upper-right corner of the HTML page, then click OK:



- **Commands** describes the SOFTIMAGE|XSI commands available for scripting.
- **Property Sets** describes the parameters supported by the various property sets and operators. It also describes a variety of math-helper functions for working with vectors and matrices.
- **Objects** describes the properties, methods, and constants supported by the various object and component types.

Other resources and their URLs

There are many additional sources of information on various scripting languages. While many of them are focused on web-page development, there are some that are more general:

- **Win32 Scripting:** cwashington.netreach.net/
- **Winscripter:** www.winscripter.com/
- **Windows Script Host FAQ:** wsh.glazier.co.nz/

- **Born's Bookp@ge:** wsh.glazier.co.nz/
- **Power Scripting:** homepages.go.com/~sbondi/powerscripting/index.html
- **The (unofficial) JScript FAQ:** www.netspace.net.au/~torrboy/code/jscriptfaq/

Names

For most scripting commands, you need to know the names of objects and parameters. If the object was created by the script, you can use the return value instead of the object name; see *Returning Values* on page 197.

Object Names

The object name is simply the name of an object as it appears in the explorer as well as in the Object Name text box on the Selection panel. For example,

```
SelectObj "cone"
```

selects the object named **cone**.

The explorer also shows the element's path, but in most cases you can ignore the beginning of the path. For example, the following two commands are equivalent:

```
SelectObj "Camera"
```

```
SelectObj "Project.Scene.Scene_Root.Children.Camera"
```

If an object is part of a model other than the scene root, you must also specify the model name. This is because each model has its own namespace. For example,

```
SelectObj "Model1.cone"
```

selects the object named **cone** in the model named **Model1**.

You can use the wildcard character (*) to represent any character string, including the empty string. You can place the wildcard character at either the beginning or the end of model and object names. For example,

```
SelectObj "cone*"
```

selects **cone**, **cone1**, **cone2**, and so on. Furthermore,

```
SelectObj "*.cone"
```

selects all objects named **cone** in any model including the scene root.

When writing scripts, you can't always be certain beforehand what an object is called in a scene when the script is run. Even if you create an object and specify a name within a script, the name may be appended with a number if there is already an object with that same name in the scene. For information about dealing with these situations, see *Enumerating Elements* on page 207 and *Returning Values* on page 197.

Parameter Names

The parameter name for scripting is the same name as that which appears in the marked-parameter box on the Animation panel. It corresponds to the branch structure of the property sets as shown in the explorer, and it works in conjunction with model and object names. For example:

```
SetValue "Model11.cone.kine.local.ori.euler.rotx", 90
```

where:

- **Model1** is the model. This can be omitted in the case of **Scene_Root**.
- **cone** is the object. For the current selection, you can omit the object name and put a period before the rest of the parameter name. For example, **.kine.local.ori.euler.rotx** specifies the local X rotation of the currently selected object.
- **kine.local.ori.euler** is the property set. In certain cases, parts of this may be omitted—see *Omitting portions of property-set names* on page 202.
- **rotx** is the parameter.

The parameter scripting names are also documented in HTML files: see *HTML documentation* on page 199.

For parameters, the scripting name is not always the same as the path that is shown in the explorer. Scripting names are internal names that cannot contain spaces, while the explorer displays UI labels.

Determining parameter names

The easiest way to determine a parameter name is to change its value in a property editor and see what gets logged in the command history:

1. Select an object (or any node in the explorer—use the scope and filter options to find the object you want).
2. Press Enter to open a property editor with the object's properties, or press Alt+Enter to open a property editor with all the object's properties.
3. On the appropriate property page, modify a parameter.
4. View the resulting command in the history pane of the script editor. It should resemble the following:

   ```
   SetValue "[Model.]Object.Property.Parameter", Value
   ```

You can also determine a parameter name by marking a parameter in the explorer, then checking the argument in the resulting **SetMarking** command. However, the name that is logged is relative to the currently selected node and may not show the whole property set.

Omitting portions of property-set names

Although the full property-set and parameter names will always work in scripts, there are certain cases where you can omit the names of intermediate nodes. This may be convenient if you are typing commands manually rather than copying and pasting from the history pane.

The following node names can be omitted in property-set names:

- **surfmesh**, **polymsh**, **crvlist**, and other geometry types.
- **local** or **global** coordinate systems for transformations. If this is omitted, local is used by default.
- for transformations, any other nodes (such as **euler**) between **kine** and the parameter itself.

For example, if cone is selected then the following two commands are equivalent:

```
SetValue "Model11.cone.kine.local.ori.euler.rotx", 90
SetValue ".kine.rotx", 90
```

Element Types

You can use the # character in strings to work with a specific type of element. For example:

- `SelectObj "*.#override"` selects all overrides in the scene.
- `SelectObj "#model"` selects all models in the scene.
- `SelectObj "*.#3dobject"` selects all 3D objects in the scene.
- `SelectObj "*.#pass"` selects all passes in the scene.
- `SelectObj "*.#group"` selects all groups in the scene.

Parameters

To work with parameter values, use **SetValue** and **GetValue**(). To use these commands, you need to know a parameter's scripting name—see *Names* on page 200.

Setting and Getting Parameter Values

To get the value of a parameter, first use the **GetValue**() function to return an object, then use the returned object to get the value. For example, to get the Camera's Field of View Angle:

```
'Get an object containing the camera's fov
myObj = GetValue("Camera.camera.fov")


'Display the value
MsgBox "FOV = " & myObj
```

You can also use **GetValue**() to return a parameter's value at a particular frame. For example, the following code gets the camera's local X-axis position at a frame number specified by the **fr** variable. Note that for local transformations, you can omit **local** from the parameter's name.

```
fr = 24
myObj = GetValue("Scene_Root.Camera.kine.local.pos.posx", fr)
MsgBox "Camera's X position at frame " & CStr(fr) & " is " & CStr(myObj)
```

When setting values, you can use the same characters you would type in a numeric field in the interface to change values relatively, use ranges of values, use random values, and so on. For example, to add 10 to the local Y position of all objects whose name starts with "sphere":

```
SetValue "sphere*.kine.local.posy", "10+"
```

Working with Marked Parameters

There are several commands that help you work with marked parameters: **SetMarking**, **GetMarking**, **AddToMarking**, **RemoveFromMarking**, and **ClearMarking**. See the HTML scripting documentation for details.

Working with Unmarked Parameters

When a command uses marked parameters by default, you can specify different parameters by using a slash / character:

```
SaveKey "/kine.local.pos"
```

Custom Parameters

You can work with custom parameters using the **AddProp**, **SIAddProp**, **SIAddCustomParam**, and **SIAddCustomParameter** commands. In addition to their other uses, you can use custom parameters to store information that is shared between scripts, as if they were global scene variables.

For example, you can store a text string to store and retrieve a list of parameters:

```
SaveMarks = "kine.global.posz,kine.global.roty,kine.global.sclz"
AddProp "Custom_parameter_list", , , "uData"
SIAddCustomParameter "sphere", "SaveMarks", siString, 0, 0, 1, , 4, 0, 1
setValue "Sphere.uData.SaveMarks", SaveMarks
ReadMarks = GetValue("Sphere.uData.SaveMarks")
myList = split(ReadMarks, ",", 2000)
```

You can also use custom parameters to display property editors that prompt you for data. In the **InspectObject** command, the **siModal** token adds OK and Cancel buttons and waits for a response before continuing:

```
set cursel = GetValue("SelectionList")
InspectObj cursel & ".uData",, "User Data: " & myItem, siModal
```

You can then delete the custom parameters before exiting the script if they are no longer needed.

Selection

SelectionList

The currently selected objects are available in the SelectionList. To access it, use the **GetValue**() command:

```
dim return
set return = GetValue("SelectionList")
for each obj in return
   MsgBox obj & " is an " & obj.Type
next
```

SelectionList is a type of object called a collection. For more information about collections, see page 205.

Selecting and Deselecting

To select an element or parameter, use the **SelectObj** command with the corresponding name. For example:

```
SelectObj "CameraInterest"
```

To add something to the current selection, use the **AddToSelection** command:

```
AddToSelection "SpotInterest"
```

To remove something from the current selection, use the **RemoveFromSelection** command:

```
RemoveFromSelection "Scene.Scenecolors.litcol"
```

To select all cameras, lights, and 3D objects in a scene, use:

```
SelectAll
```

Filtering the Selection List

You can use the **SIFilter** command to filter the selection list. The names of the filters are exactly the same as appear in the selection filter list in the interface. For example, to filter the current selection and return only chain elements:

```
set selList = SIFilter(, "Chain_Element")
```

Picking

You can use the **PickElement** command to start an interactive picking session in a script:

```
Dim ButtonPressed, PickedElement
call PickElement( "property", "Select a property", "Select a property",_
                              PickedElement, ButtonPressed  )
if ButtonPressed <> 0 Then
      SelectObj PickedElement
End if
```

Collections

A collection is a group of objects. For example, the selection list is stored as a collection.

```
dim list
set list = GetValue("SelectionList")
```

- `list` is a collection.
- `list.Count` is the number of objects in the collection.

You can access objects in a collection using the Items method:

```
for i = 0 to list.Count - 1
    msgbox list.Items(i)
next
```

Items is the default method, so you can also write:

```
msgbox list(i)
```

The Items property returns a string representation of the collection.

```
msgbox list.Items
```

Items is the default property, so you can also write:

```
msgbox list
```

You can also index into a collection:

```
set list = GetValue("SelectionList")
msgbox list.Items(0)
```

Creating a Collection

```
' Create XSI.Collection object and add items
'
set col = CreateObject( "XSI.Collection" )
```

Adding Items to a Collection

To add an item to a collection, use the collection's **Add** method. For example, if **col** is a collection:

```
col.Add "CameraInterest"
```

You can also create a **CollectionItem** and set its value:

```
set item = CreateObject( "XSI.CollectionItem" )
item.value = "Camera"
col.Add item
```

In addition, you can populate a collection using wildcards:

```
set l_Layers = CreateObject("XSI.Collection")
l_Layers.Items = "Layers.List.*"
for each l_Layer in l_Layers
   logmessage l_Layer.name
next
```

Removing Items from a Collection

To remove an item from a collection, use the collection's **Remove**() method.

```
col.Remove "CameraInterest"
col.Remove item
logmessage , "Collection= " & col
```

To remove all items from a collection, use the collection's **Clear**() method.

```
col.Clear
logmessage , "Object Count = " & col.Count
```

Components

Object components, such as points, polygons, isolines, and so on, can be referenced like array elements using square-bracket notation. Each component is referred to by its index number, which is part of the internal representation of the geometry and cannot be changed. Index numbers run from zero to the number of components minus one (you can also use the token **LAST**). For example:

```
AddToSelection "cone.pnt[0-14,16-LAST]"
```

Enumerating Elements

You can use the **EnumElements** command to return a list of an element's children or parents. This is useful when you do not know what elements are in a scene, or what they are called.

You can also use an element's **PathItems** property to go up the hierarchy. For example:

```
dim sellist, item
set sellist = GetValue("SelectionList")
set item = sellist(0)
for each item in item.PathItems
   logmessage item
next
```

If **cube.polymsh.geom.cube** was selected, the example above logs something like this:

```
'INFORMATION : "cube.polymsh.geom"
'INFORMATION : "cube.polymsh"
'INFORMATION : "cube"
```

You can use **PathItems** to get the owner of a cluster:

```
set list = SIFilter( , "cluster" )
if ( TypeName( list ) <> "Nothing" ) then
      ' Get the object that owns the cluster
      n = list(0).pathitems.Count - 1
      set object = list(0).pathitems(n)
end if
```

Scenes

To work with scenes, use the **NewScene**, **OpenScene**, and **SaveScene** commands. Note that if you use **OpenScene** in a batch script, you must specify a path and scene name explicitly or use a command-line parameter—see *Preparing Scripts for Batch Mode* on page 193.

Scene Root

Remember that you can rename the scene root. If you want your script to work reliably in such cases, you have some extra work to do:

```
set mySceneRoot = EnumElements("Project", TRUE)
set mySceneRoot = SIFilter(mySceneRoot, "Scene", TRUE )(0)
set mySceneRoot = EnumElements( mySceneRoot , TRUE)
set mySceneRoot = SIFilter(mySceneRoot, "SceneObject", TRUE)(0)
```

Groups

If you added an object in branch mode to a group, to remove it you must select it properly, then use the **RemoveFromGroup** command without any parameters. For example:

```
SelectObj "null", "BRANCH", True
AddToGroup "Group"
...
SelectObj "Group"
AddToSelection "null", "BRANCH"
RemoveFromGroup
```

Installing Script Commands

You can use the **CreateToolbar**, **CreateScriptCommand**, and **CreateToolbarButton** commands to automatically install script commands and make them available on toolbars.

```
sub AutoInstall
   l_UserPath = GetUserPref("DS_PRESET_USER_PATH")
   l_FilePath = l_UserPath & "\Scripts\myscript.vbs"

   'Create Toolbar
   l_Toolbar = CreateToolbar("Clip Tools")

   'Create Commands
   l_Command = CreateScriptCommand( "My Command" ,_
                                    "MyCommand" ,_
                                    l_FilePath ,_
                                    "myprocedure" ,_
                                    "This is a test." )
      LogMessage l_Toolbar

      'Add button to toolbar.
      CreateToolbarButton l_Toolbar, l_Command
end sub
```

Optimizing Scripts

One way to speed up your scripts is to declare your variables. Although most scripting engines do not require you to declare variables, declaring them anyway means that the scripting engine has less checking to do. For example, to declare variables in VBScript:

```
dim myVar1, myVar2, myVar3
```

Another way to speed up your scripts is to check your arguments within the scripting language before making a potentially unnecessary call to a SOFTIMAGE|XSI command. For example:

```
...
If Not IsEmpty (myList) Then
   AddToSelection myList
End If
```

Chapter 8 Customizing SOFTIMAGE®|XSI™

Customization Levels

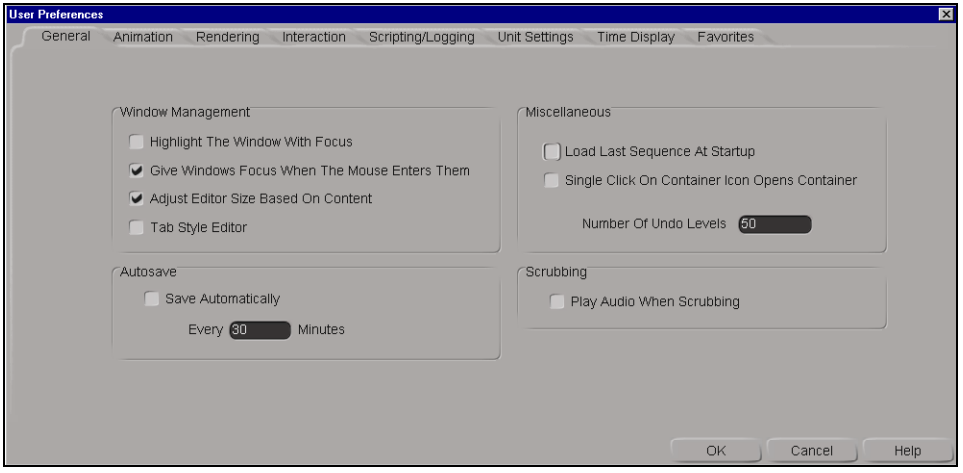
There are several ways to customize your workspace. The table below summarizes the levels of customization available.

Customization Level	Description
User preferences	Sets a range of operating conditions (e.g.: number of undo levels, keyframe settings, save preferences). For more information, see *Setting User Preferences* on page 214.
Keyboard shortcuts	Associates commands to keyboard keys. For more information, see *Creating Keyboard Shortcuts* on page 215.
Custom layouts	Modifies the default interface layout to suit your own tastes or create your own layout from scratch. For more information, see *Customizing the Layout* on page 217.
Presets	Lets you saves sets of modified properties belonging to objects, operators and shaders and reuse them as needed. For more information, see *Presets* on page 117.
Scripts	Lets you program a series of commands that can be used to execute repetitive operations. For more information, see *Commands & Scripts* on page 169.

Setting User Preferences

The User Preferences dialog box lets you set a number of commonly used operating conditions. Display the dialog box by choosing the **File** > **User Preferences** command from the main-menu bar.

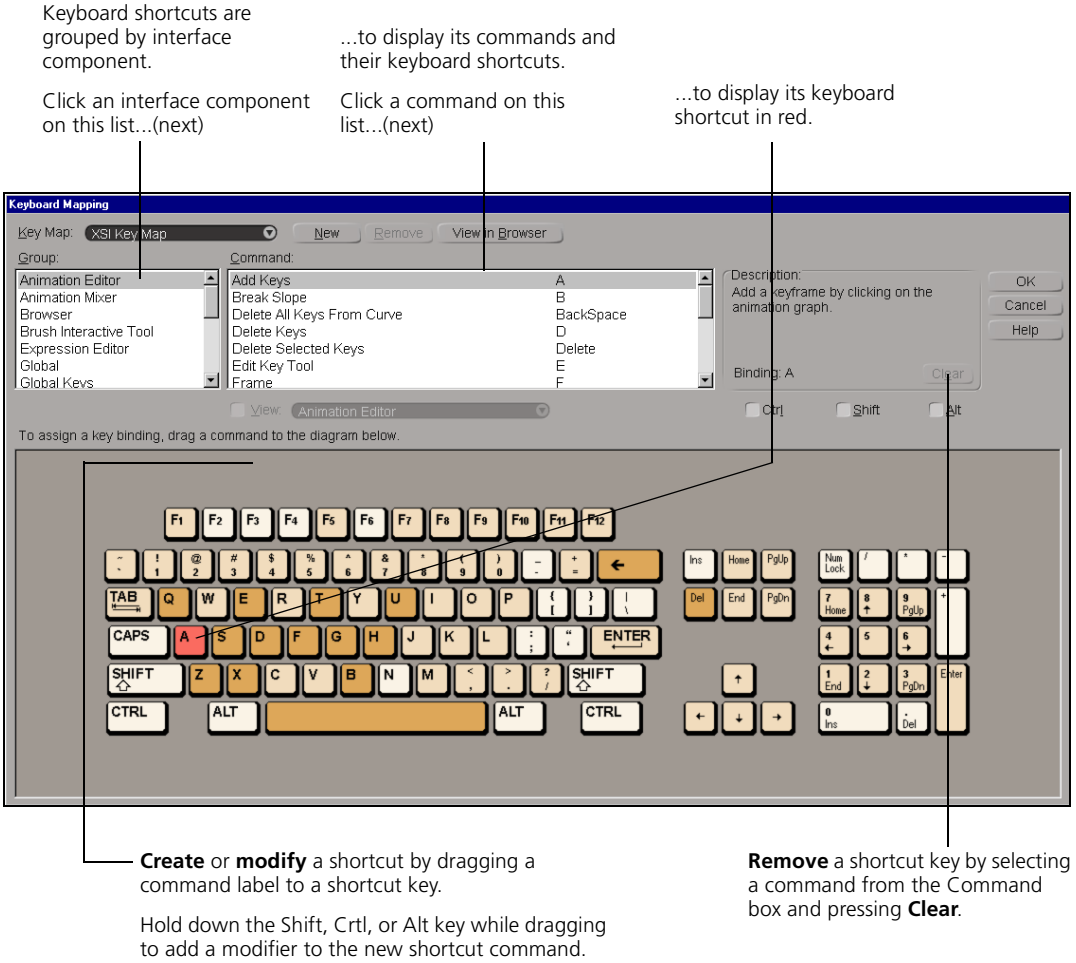
You will find a description of the various user preferences found in the User Preferences dialog box throughout this book. For a comprehensive description of each control, refer to Online Help.



Creating Keyboard Shortcuts

Supra keys and other keyboard shortcuts let you enter commands at a touch of a key, rather than from point-and-click menus on screen. The Keyboard Mapping dialog box lets you view a default set of shortcuts as well as create shortcuts of your own.

You can create sets of keyboard shortcuts and store them for later use. The mapping of keyboard shortcuts to commands are stored in a separate file (under Preferences/Keymaps).



To view keyboard shortcuts

1. Choose **File** > **Keyboard Mapping** from the main-menu bar. The Keyboard Mapping dialog box appears.
2. In the **Group** box, choose the interface component whose commands you wish to view.

   The Command box displays a two-column list of commands and their keyboard shortcuts. (Not all commands have shortcuts assigned to them.)

   The keyboard graphic below the Command box depicts the keyboard shortcuts that correspond to the interface component you selected.

   The keyboard keys are color coded to indicate the following:

   White—No keyboard shortcut has been assigned to this key.

   Beige—A keyboard shortcut from another interface component has been assigned to this key.

   Brown—A keyboard shortcut from the currently selected interface component has been assigned to this key.

   Red—This keyboard shortcut corresponds to the currently selected item in the Command box.

To create or modify keyboard shortcuts

1. Choose **File** > **Keyboard Mapping** from the main-menu bar. The Keyboard Mapping dialog box appears.
2. You cannot change the default shortcut key templates; you can only modify keyboard shortcuts from your own template.

   Click the **New** button to create a shortcut key template and enter its name.

   or

   Select an existing editable shortcut key template from the Template combo box.
3. In the **Group** box, choose the interface component whose commands you wish to work on.
4. In the **Command** box, click and drag the command whose shortcut you want to create or modify down to the keyboard graphic below.

   Hold down the Shift, Ctrl, or Alt key as you drag to add a modifier to the new shortcut.
5. Drop the command icon onto the key you want to assign the command's shortcut to.

   The new shortcut key appears in the column to the right of the command name in the Command box. Any previous key association is removed.
6. To remove a keyboard shortcut, select the shortcut's command from the Command box and click the **Clear** button in the Description box.

Customizing the Layout

After you have worked with SOFTIMAGE|XSI for awhile, you will probably want to adjust the layout of its interface in a way that is the most comfortable and efficient for you. You may also find that you require different layouts for different types of projects. You can create as many layouts as you need.

Each layout is composed of an arrangement of panels containing logically grouped sets of controls that occupy fixed areas of the screen. You can customize a layout by rearranging the location and size of these panels. You can also assign more than one view to a panel by creating a layout stack.

All these changes can be saved as a separate layout and displayed again in subsequent sessions. If you do not save the changes you make to your layout, they can only be viewed in the current session.

Editing the Layout

Creating Layouts

Before you begin editing, it is a good idea to first create a copy of the current or default layout, then edit the copy. This way, you can revert to the original layout if you no longer want to use the one you modified.

You can create new layouts from either a blank template or an existing layout. Each time you create a layout, its icon is added to the taskbar. (You display the taskbar by choosing **View** > **Taskbar** from the main-menu bar.)

Taskbar showing one default layout and one custom layout.

Tip: Right-click on a layout icon to quickly copy the layout or create a new one.

To create a layout

1. Choose **Layout** > **Edit Layout**.
2. Choose **Layout** > **New Layout**.
3. Choose a command from this menu:
   - **Blank** creates a blank layout to which you add the various panels that you want to work with. Right-click on the blank layout and use the **Split** options and the components in the **Set View** list to build your layout.
   - **Copy Current Layout** lets you create a custom layout based on the currently active layout.

- **Copy Layout** opens a menu that lets you create a custom layout based on a copy of a particular layout.

4. Enter a name for the new layout you are creating and click OK. An icon is created for the new layout and is added to the taskbar.

To edit a layout

1. Choose the **Layout** > (**layout name**) to select the layout you want to edit.
2. Choose **Layout** > **Edit Layout** to place your layout into edit mode.
3. A red border surrounds each panel as you pass over it with your cursor, indicating that it is in edit mode. You can resize the panels, split them into smaller sections, or move them to a new location in the layout.
   - To resize a panel, place the mouse pointer over this red border and drag it to the required size.
   - To move a panel, right-click over the location on the desktop you want to move the panel to, choose **Set View**, and select the panel's name. If the panel was already displayed elsewhere in the layout, its former location is vacated. You can choose to fill this gap with another panel or resize the adjoining panels.
   - To split a panel, right-click on it and select one of the three split options.
4. When you are satisfied with the adjustments and want to save your changes for future sessions, choose **Layout** > **Save Layout**, then choose the **Edit Layout** command again to end the edit layout mode. For details on saving layouts automatically, see *Saving Layouts* on page 220.

   If you want to save your layout only for the current session, no not choose **Layout** > **Save Layout**; simply choose the **Edit Layout** command.

To restore the default layout

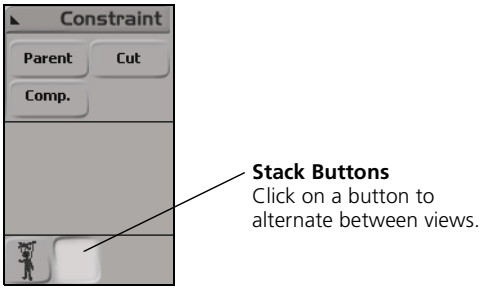
If you went ahead and edited the default layout without making a copy and now want to revert to the original, do the following:

1. Exit SOFTIMAGE|XSI.
2. Navigate to **C:\users\(username)\Softimage\Preferences\Layouts\(current screen resolution)**.
3. Delete the files in the Layouts folder.

   The next time you start up the application, the factory-supplied default layout will be used.

Assigning Additional Views to a Panel

Using the Layout Stack controls, you can assign additional views to a panel in the interface. Once you do so, you can display an alternate view in the panel by clicking a layout stack button.



To add views to a panel

1. Choose **Layout** > **Edit Layout** to place your layout into edit mode.
2. Right-click on the panel you want to assign the additional view to.
3. Choose **Add View Switcher.** The panel goes blank and two layout stack buttons appear.
4. Right-click on the panel a second time and choose **Set View** to display the pop-up list of panel views.
5. Select the required panel view. The blank panel is now replaced with the selected panel view.
6. To switch between panel views, click on a layout-stack button.

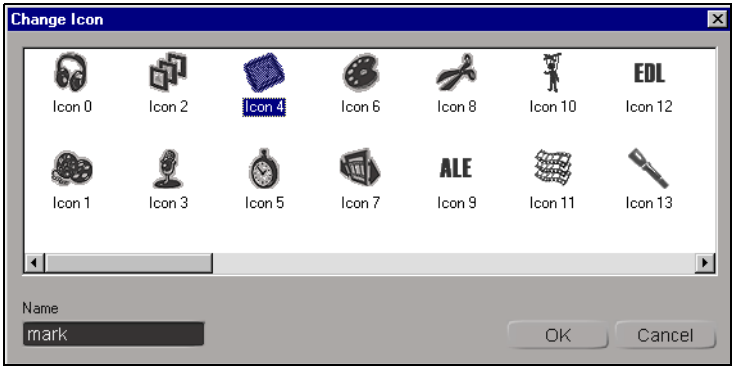
You can continue to add views to the panel by right-clicking on any layout stack button and selecting **Add New Set of Views.**

To remove a panel view

Right-click on any layout-stack button and choose **Remove This Set of Views.**

To customize the layout-stack buttons

Right-click on either layout-stack button and choose from the displayed set of options. Selecting **Icon...** opens the Change Icon dialog box (shown below), which contains an assortment of icons that you can apply to the layout-stack button.



Saving Layouts

Any changes that you make to your layouts are saved to your user preference file. These settings are made available each time you start SOFTIMAGE|XSI.

To save a layout

1. While still in edit mode, choose **Layout** > **Save Layout**.
2. Choose **Layout** > **Edit Layout** a second time to exit edit mode.

To save a layout automatically

1. Choose **File** > **User Preferences**.
2. Click the **General** tab.
3. Select the **Save Layouts on Exit** option.

   As long as this option is selected, any changes you make in your layout are automatically saved.

Deleting Layouts

Display the desktop taskbar (choose **View** > **Taskbar** from the main-menu bar) and right-click on the icon that corresponds to the layout you want deleted. From the drop-down list, choose **Remove Layout**.

Creating Custom Toolbars

You can create your own toolbar and use it to hold commonly used tools and presets (specially customized effects).

You add to your toolbar by dragging tools from the Customize Toolbar dialog box or from the browser. Tools and presets are represented as buttons on the toolbar.

To create a new toolbar

1. Choose the **View** > **Custom Toolbars** > **New Toolbar** command from the menu bar.
2. In the Name of New Toolbar dialog box, enter a name for the toolbar.

   A new toolbar is created and added to the desktop.

To add a button to a toolbar

1. Right-click on an empty area of a toolbar and choose **Customize**.

   The Customize Toolbar box appears.
2. From the Category list, select an item.
3. From the Tools list, select a tool and drag it to a toolbar.
   A button representing the tool you selected is added to the toolbar.

To add a preset to a toolbar

In the browser, locate the preset you wish to include on your toolbar.

To remove a button from a custom toolbar

- Right-click on a toolbar button and choose **Remove Button**.

To change the display of toolbar buttons

- Right-click on an empty area of a toolbar and choose **Display as Text** or **Display as Icon**.

To delete a custom toolbar

- Right-click on an empty area of a toolbar and choose **Delete Toolbar**.

To display a closed toolbar

- Choose **Views** > **Custom Toolbars** > (**toolbar name**).

Importing and Exporting Your Customized Preferences

You can convert customized layouts, toolbars, keyboard shortcuts, and scripts into files and import and export them between computers.

Preferences can be imported and exported, according to a set hierarchy. If you export a customized layout, for example, you automatically export its keyboard shortcuts, toolbars, and scripts as well. Similarly, if you import or export a toolbar, you also import or export any associated keyboard shortcuts and scripts that may accompany it. Scripts can be imported and exported on their own.

To export your customized preferences

1. Choose **Layout** > **Import/Export Preferences**.
2. Click the **Export Preferences** tab.
3. In the **Type** drop-down menu, choose the type of preference you want to export, then choose from the list of preference types that display in the box below.
4. Click **Export** and, in the browser that displays, navigate to the folder you want to place the exported preference file in.
5. Give the preference file a name in the **File Name** box and click OK.

To import your customized preferences

1. Choose **Layout** > **Import/Export Preferences**.
2. Click the **Import Preferences** tab.
3. In the **File Name** box enter the name of the preference file you want to import, or click **Browse** to search for the file in a browser.
4. In the **Type** drop-down menu, choose the type of preference you want to import, then choose from the list of preference types that display in the box below.
5. Click **Import** and click OK.

Appendix **Standalones**

Using Standalones (Command Line Utilities)

During the creative process of defining the surface, animating, and rendering your masterpieces, there are other tasks that you need to accomplish from time to time. These tasks include displaying rendered images or image sequences, compositing images, converting images formats, grabbing images, and outputting images to film or video recorders. SOFTIMAGE|XSI provides a number of command line utilities called *standalones* that help you do these tasks.

Standalones allow you to accomplish tasks outside the SOFTIMAGE|XSI interface. In some cases, you can perform these tasks from within the interface, but may choose to use a standalone for more control or speed—the choice is yours. There are also tasks accomplished by standalones that cannot be done in the interface, such as recording to peripheral devices.

You can run standalones by typing specific commands in a command prompt window or a shell, or you can run them from commands saved in a C-shell file or MS-DOS batch file.

Getting Help on the Options

For most standalones, the usage is displayed when you type the command name without options. For certain standalones, you must type the standalone's name followed by the `-h` or `-help` option to display the usage, such as `mb2d -help`

Syntax Conventions

These are the typographical conventions used to describe the usage (syntax) of the standalones.

- A dash (-) followed by one or more letters (as in -z) indicates an option name. For example, -z usually means zoom, -s usually means sequence, -v usually means verbose mode, etc. The available options are listed and explained in the description of each standalone.
- Angle brackets(< >) indicate a parameter for which you must substitute a name or value. For example, <filename> means that you must type the name of the file you want to use. When entering a parameter, do not type the angle brackets. Unless otherwise specified, do not include the extension (such as .pic, .lin, .hrc, .mi, etc.) when specifying file names.
- Commands and parameters that are not enclosed in angle brackets must be typed exactly as they appear.
- Strings are quoted within double quotes; this includes all names. The double quotes protect names from interpretation by the shell or command prompt window.
- Brackets ([ ]) indicate an option along with its input. Do not type the brackets. For example, [-z <fact>] means that if you use the -z option (zoom), you must also specify its <fact> (zooming factor).

- A vertical bar (|) separates exclusive options. For example, the usage for -face includes front|back|both. This means that you must specify only one of these three options for defining which face of the object is to be rendered. When specifying an option, do not type the vertical bar.
- Braces ({ }) enclose a group of possible values related an option. When entering parameters, do not type the braces.
- The ellipsis (...) indicates that the option or group of options can be repeated.
- Options and inputs appearing on the same command line must be separated by spaces.
- Commands should be entered as a single line in a shell or command prompt window, or in a C-shell file.

Converting Images

You can choose from six standalones that let you convert image file formats:

- Convert files between a variety of image file formats with **imgconv** standalone.
- Convert image files to other formats including the format for creating memory-map textures with **imf_copy** standalone.
- Convert AVI files to SOFTIMAGE picture files and back with **AVI2soft** and **soft2AVI** standalones.
- Convert color images or sequences to gray-scale images or sequences with **gray_scale** standalone.
- Convert RGB or RGBA picture files to SOFTIMAGE picture files with **rgb2soft** standalone.

Converting between Many File Formats

The **imgconv** standalone converts files between a wide variety of image file formats. It also provides a set of image processing tools such as **blur**, **contrast**, and **zoom**.

This standalone runs on both the Windows NT and IRIX operating systems.

imgconv automatically detects supported image file formats, so input files do not require an extension (such as **.pic**) to identify their format. To specify the format of the output file, you can use either an extension (such as **.tif**) or the `-format` option.

The **imgconv** standalone can convert to any file format for which there is a dynamic shared object (DSO on IRIX systems) or a dynamic link library (DLL on Windows NT systems). The setup program installs these DSOs or DLLs to <**install directory**>**/bin/sil.** When you type `imgconv`, the usage message lists the supported file formats.

To run **imgconv**, you must set the environment variable **SI_IMAGE_PATH** to point to the <**install directory**>**/bin/sil** directory. If **SI_IMAGE_PATH** is not set or empty, **imgconv** checks the environment variable **SI_HOME**. If it is set, **imgconv** looks in **$SI_HOME/bin/sil**.

Usage

```
imgconv <input file> <output file> [-height <height>]
[-width <width>][-reduction <colormap size>]
[-format <name>][-compression <none|rle|lzw|packbits>]
[-description <string>]
[-processor <name> ["<arguments>"]][-verbose]
```

`<input file>` is the name of the file you want to convert or process. Use a dash (-) to read from the standard input. **imgconv** automatically detects supported image file formats, so input files do not require an extension (such as **.pic**) to identify their format.

`<output file>` is the name of the output file. Use a dash (-) to write to the standard output. To specify the format of the output file, you can either use an extension (such as **.tif**) or the `-format` option.

Options

`-width`

Specifies the width in pixels of output images. Used only for raw image formats with no image header.

`-height`

Specifies the height in pixels of output images. Used only for raw image formats with no image header.

`-reduction`

Specifies that output images must be color-reduced using the given color map size.

`-format`

Specifies the format to use for output images, or for input images having no signature.

The usage message for **imgconv** lists the supported formats.

`-compression`

Specifies the compression technique used for output images.

`-description`

Specifies the description string to use for output images.

`-processor`

Specifies a processor name to apply to output images. Processor arguments must be enclosed in quotes.

The usage message for **imgconv** lists the supported processors.

`-verbose`

Activates verbose output.

There is no `-script` option available for **imgconv** despite what the usage message lists.

Converting Image Files to Memory-Map Textures

The **imf_copy** standalone is used to convert image files to other formats including the format required to memory-map texture images into SOFTIMAGE|XSI and the mental ray command line renderer.

For more information on memory mapping, see *Creating Memory-Mapped Textures* on page 126 of the *Shaders, Lights & Cameras* guide.

This standalone runs on both the Windows NT and IRIX operating systems.

Usage

```
imf_copy [-v] [-p] inimage outimage [outformat
[outtype]]
```

imf_copy supports the following file formats:

File format	Symbol	Supported data types
Wavefront image	rla	**rgba** (8-bit RGBA color and alpha)
SOFTIMAGE picture image (compressed)	pic	**rgba** (8-bit RGBA color and alpha)
Alias Research image	alias	**rgb** (8-bit RGB color)
SGI color (compressed)	rgb	**rgba** (8-bit RGBA color and alpha)
JFIF image	jpg	**rgb** (8-bit RGB color)
Portable pixmap	ppm	**rgb** (8-bit RGB color)
Targa image	targa	**rgba** (8-bit RGBA color and alpha)
MS Windows bitmap	bmp	**rgba** (8-bit RGBA color and alpha)
mi color texture	ct	**rgba** (8-bit RGBA color and alpha), **rgba_16** (16-bit RGBA color and alpha), **rgba_fp** (floating-point RGBA color and alpha)
mi scalar texture	st	**s**, **s_16** (16 bit mi scalar texture)
mi vector texture (from alpha)	vta	**vta**
mi vector texture (from intensity)	vts	**vts**
mi z channel (from intersect depth)	zt	**z** (depth information)
mi normal vectors	nt	**n**
mi tag channel (from material tags)	tt	**tag**
mi bit mask	bit	**bit**
4-component TIFF (compressed)	tif	**rgba** (8-bit RGBA color and alpha), **rgba_16** (16-bit RGBA color and alpha), **rgb** (8-bit RGB color), **rgb_16** (16-bit RGB color)

File format	Symbol	Supported data types
uncompressed TIFF image	tifu	**rgba** (8-bit RGBA color and alpha), **rgba_16** (16-bit RGBA color and alpha), **rgb** (8-bit RGB color), **rgb_16** (16-bit RGB color)
Dassault Systemes Catia format	picture	**rgb** (8-bit RGB color)
memory mapped raw texture file	map	any
Quantel/Abekas YUV image, 576x720	qntpal	**rgb** (8-bit RGB color)
Quantel/Abekas YUV image, 486x720	qntntsc	**rgb** (8-bit RGB color)

Options

`-v`

Activates verbose output.

`-p`

Creates a memory-mappable, filtered image pyramid.

Converting AVI Picture Files to PIC

The **AVI2soft** standalone converts AVI picture files (.avi) into SOFTIMAGE picture (.pic) files.

This standalone runs only on the Windows NT operating system.

Usage

```
AVI2soft <input file> <output file>
[-s <start frame> <end frame> <step>]
```

`<input file>` is the name of the AVI picture file to be converted.

`<output file>` is the name of the SOFTIMAGE picture file to be created.

Options

`-s`

Converts a sequence of frames.

Picture files in the AVI file format begin at 0 and end at <number of frames –1>.

Converting PIC Files to AVI

The **soft2AVI** standalone converts SOFTIMAGE picture (**.pic**) files into AVI picture files (**.avi**).

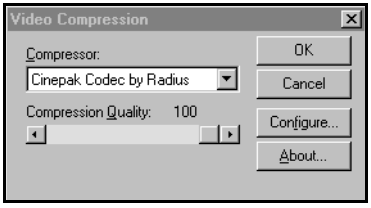
This standalone runs only on the Windows NT operating system.

Usage

```
soft2AVI <input file> <output file> [-r <frame rate>]
[-s <start frame> <end frame> <step>]
[-f <sequence width>]
```

Where `<input file>` is the name of the SOFTIMAGE picture file (**.pic**) and `<output file>` is the name you want to give to the AVI picture file (**.avi**).

When you enter the usage for the **soft2avi** standalone on the command line, a **Video Compression** dialog box opens in which you can select the codec to be used for compression.



Options

`-r`

Specifies the number of frames per second. The default number of frames per second is 30.

`-s`

Converts a sequence of frames.

`-f`

Defines the number of digits contained in each frame number. The default number of digits is 1.

Example

The following example shows the conversion of the SOFTIMAGE picture file foo.0001.pic to AVI format.

```
soft2AVI foo -s 1 10 1 -f 4
```

Converting Color to Gray Scale

The **gray_scale** standalone converts a color image or sequence of images into a gray-scale image or sequence of images.

This standalone runs on both the Windows NT and IRIX operating systems.

Two averaging methods are available to convert the colors into gray scale. The default method uses the NTSC averaging formula:

```
gray scale value = (0.299 x red) + (0.587 x green) +
(0.144 x blue)
```

The second method simply computes the normal average of the three colors (`-n` option).

Usage

```
gray_scale <input file> <output file>
[-n] [-r <repeat count>]
[-s <start frame> <end frame> <step>] [-v]
```

where `<input file>` is the name of the file you want to convert and `<output file>` is the name you want to give to the result.

Options

`-n`

Uses the normal averaging method.

`-r`

Specifies how many times each frame should be repeated. The default is 1.

`-s`

Performs the transformation on a sequence of frames.

`-v`

Activates verbose output.

Example

This example converts the **drab** image to the **vibrant** image file from frames 10 to 20 with a step of 1 (**-s 10 20 1**) using the normal averaging method (**-n**).

```
gray_scale drab vibrant -n -s 10 20 1
```

Converting RGB Picture Files to PIC

The **rgb2soft** standalone converts an RGB picture file (**.rgb** or **.rgba**) to a SOFTIMAGE picture file (**.pic**).

This standalone runs on both the Windows NT and IRIX operating systems.

Not all files with the **.rgb** extension are pure, plain RGB files. Some applications on the Silicon Graphics® platform, such as **snapshot**, save SGI files with an **.rgb** extension. Other applications on Macintosh place a header with width and height information in files with the **.rgb** extension.

rgb2soft converts these two types of files correctly but displays a message indicating that the input file did not correspond to the RGB format.

Usage

```
rgb2soft <input file> <output file>
[-d <width><height>] [-v] [-r <repeat count>]
[-s <start frame> <end frame> <step>]
```

Where `<input file>` is the name of the RGB picture to be converted (**.rgb** or **.rgba**) and `<output file>` is the name you want to give to the SOFTIMAGE picture file. If no extension is specified, the output file is given a **.pic** extension.

`<width>` is the width of the picture in pixels.

`<height>` is the height of the picture in pixels.

You do not need to specify `<width>` and `<height>` when converting Macintosh **.rgb** files that contain a header.

Options

`-d`

Specifies the width and height of the input image when converting old RGB and RGBA files.

`-v`

Activates verbose output.

`-r`

Specifies how many times each frame should be repeated. The default is 1.

`-s`

Converts a sequence of frames.

Displaying Images

There are a number of ways to view your image after it is rendered. You can use the following standalone utilities:

- View any image with the **imgshow** standalone.
- View **.pic** image files with the **display** or **showpic** standalones.
- View an animation sequence with the **flipbook** standalone.
- View images with the **imf_disp** standalone.

Displaying Many Image Formats

The **imgshow** standalone displays an image on the screen.

This standalone runs on both the Windows NT and IRIX operating systems.

The **imgshow** standalone can display any file format for which there is a dynamic shared object (DSO on IRIX systems) or a dynamic link library (DLL on Windows NT systems). The Setup program installs these DSOs or DLLs to <**install directory**>**/bin/sil.** When you type `imgshow`, the usage message lists the supported file formats.

To run **imgshow**, you must set the environment variable **SI_IMAGE_PATH** to point to the <**install directory**>**/bin/sil** directory. If **SI_IMAGE_PATH** is not set or empty, **imgshow** checks the environment variable **SI_HOME.** If it is set, **imgconv** looks in **$SI_HOME/bin/sil**.

Usage

```
imgshow <image file> [-height <height>]
[-width <width>]
```

`<image file>` is the name of the file you want to show. For example, to see the image file called **toto.1.pic** at the size of 200 pixels high and 250 pixels wide, you would type:

```
imgshow toto.1 -height 200 -width 250
```

imgshow automatically detects supported image file formats, so input files do not require an extension (such as **.pic**) to identify their format. However, if the file on disk has an extension, you should specify it. For example, type:

```
imgshow toto.1.pic
```

to see the file on disk called **toto.1.pic**.

Options

`-width`

Specifies the width in pixels of output images. Used only for raw image formats with no image header.

`-height`

Specifies the height in pixels of output images. Used only for raw image formats with no image header.

Displaying Images or Sequences

The **display** standalone displays SOFTIMAGE picture files (**.pic**) and sequences on the screen, including field-rendered pictures. The **display** standalone loads the entire image, which means that it scrolls quickly but uses a large amount of main memory.

This standalone runs on both the Windows NT and IRIX operating systems.

If the picture is larger than the screen, you can drag the mouse to scroll the display. To return the display to its original state, middle-click.

To stop or quit displaying a sequence, do one of the following:

- Press the Esc key.
- Press all three mouse buttons at once.
- Double-click in the upper-left corner of the window.

Usage

```
display <filename> [-a] [-z <fact>]
[-s <start> <end> <step>] [-F|-E|-O] [-v] [-c]
[-x <xoff>] [-y <yoff>] [-b <red> <green> <blue>]
[-S <script>] [-f] [-t|-T]
```

Where `<file>` is the name of the picture or sequence.

Options

`-a`

Displays only the alpha channel.

`-z`

Zooms the image by `<fact>`. A negative value reduces the image and a positive value enlarges it. `<fact>` must be an integer.

`-s`

Displays a sequence of frames.

`-F`

Specifies full frame images. This is the default.

`-E`

Specifies field images with even field dominance.

`-O`

Specifies field images with odd field dominance.

`-v`

Turns on the verbose mode.

`-c`

Centers the picture.

`-x`

Image offset in pixels from the left of the screen (default 0).

`-y`

Image offset in pixels from the bottom of the screen (default 0).

`-b`

Followed by three values in the range [0.0, 1.0] sets the background color (default: 0.0, 0.0, 0.0).

`-S`

Runs a shell script after displaying the image completely. If the script name begins with an "at" sign (@), it requires three arguments:

- Image file name (without the **.pic** extension)
- Current frame number
- Total number of frames

You can specify arguments as follows:

```
-S "<script> <arg1> <arg2> <arg3>"
```

`-f`

Runs display in the foreground.

`-t`

Displays stereo images (*_L.pic and *_R.pic).

In stereo mode, you can use the left and right arrow keys to adjust the eye separation offset.

`-T`

Displays merged stereo pictures.

Example

This example displays the **flash.pic** file from frames 1 to 50 at a step of 1 (**-s 1 50 1**) and centers the picture (**-c**) in the frame.

```
display flash -s 1 50 1 -c
```

To view the even-field dominance image, for example, in a field-rendered sequence, use this syntax where **output.1** is the image file name:

```
display output.1 -E
```

Displaying Stereo Images

To display stereo images, use the **-t** option. You require StereoGraphics CrystalEyes Stereoscopic System to use this option (IRIX only).

The naming convention for stereo picture files is:

myPicture_L.1.pic for the left eye.

myPicture_R.1.pic for the right eye.

For example, to display **myPicture**, enter the following command line:

```
display -t myPicture
```

The stereo buffer is 491 x 532 pixels. To render pictures in stereo, you should use a pixel ratio of 0.5.

In stereo mode, you can use the left and right arrow keys to adjust the eye separation offset.

Displaying Images with showpic

The **showpic** standalone displays a SOFTIMAGE picture file (**.pic**) on the screen.

This standalone runs on both the Windows NT and IRIX operating systems.

If the picture is larger than the screen, you can drag the mouse to scroll the display. To reset the picture to its initial position, click the middle mouse button.

You can view the red, green, blue, or alpha channels separately by pressing the R, G, B, or A keys on the keyboard. To return to normal view, press the same key again.

showpic loads only the visible portion of the picture into main memory. This means that it is slower at scrolling than the **display** standalone, but it uses less memory.

When rendering field images, you can use **showpic** to view field-rendered pictures and sequences. For example, you can use **showpic** to view the first image in the sequence:

```
showpic output.1 -E
```

Usage

```
showpic <file> [-a] [-F|-E|-O]
[-s <start frame> <end frame> <step>] [-v]
```

`Where <file>` is the name of the picture or sequence to show.

Options

`-a`

Displays only the alpha channel.

`-F`

Specifies full frame images. This is the default.

`-E`

Specifies field images with even field dominance.

`-O`

Specifies field images with odd field dominance.

`-s`

Displays a sequence of frames.

`-v`

Turns on the verbose mode.

Displaying Image Sequences as a Flipbook

The **flipbook** standalone displays image sequences and field-rendered image sequences at the rate you specify. If the program cannot display the frames at that rate, it skips frames. If the image is bigger than the screen, it automatically reduces the image.

This standalone runs on both the Windows NT and IRIX operating systems.

Usage

```
flipbook <sequence> <first>
[ <last> [ <step> [<rate> ]]] [-R <factor>]
[-m] [-F|-E|-O] [-i] [v]
```

`<sequence>` is the name of the sequence.

`<first>` is the number of the first frame.

`<last>` is the number of the last frame.

`<step>` is the frame number increment (default is 1).

`<rate>` is the number of frames to be displayed per second (default is 30 fps).

Options

`-R`

Reduces image size by an integer factor. The reduction in memory requirements is the square of this integer factor. For example, reducing the image size by a factor of 2 reduces memory requirements by a factor of 4.

Use the -R option with the -m option to load more frames into main memory.

`-m`

Loads all frames into main memory before displaying the sequence. By default, frames are loaded from disk only when needed.

When using the **-m** option, the computer starts swapping if the free RAM memory is insufficient to load all frames. To load more frames, use the **-R** option to reduce image size. reduce image size. You can display one static picture by typing:

```
flipbook <picture_name>
```

`-F`

Specifies full frame images. This is the default.

`-E`

Specifies field images with even field dominance.

`-O`

Specifies field images with odd field dominance.

`-i`

Gets information about the sequence without display.

`-v`

Turns on the verbose mode.

Example

This example shows the **jumbo.pic** image sequence from frames 1 to 150 with a step of 1 (**1 150 1**) at 24 frames per second.

```
flipbook jumbo 1 150 1 24
```

For field-rendered image sequences, you can view the odd-field or even-field dominated image, or both. For example, to view the even-field image, type:

```
flipbook output -E
```

where <`output`> is your rendered image.

Mouse Functions

Once the first frame is displayed:

- Right-click to flip forward.The sequence loops when it reaches the last frame.

- Left-click to flip backward.
- Middle-click to stop flipping or loading frames (pause).
- To adjust the frame rate interactively, hold down the middle mouse button and drag the mouse to the right to flip forward, and to the left to flip backward. The display rate corresponds to the speed at which you drag the mouse.
- To set the zoom factor interactively, middle-click to pause, then press keys 1 to 9.
- To exit, click two or more mouse buttons at once or press Esc.

Keyboard Functions

- f or right arrow key move forward one frame.
- b or left arrow key move backward one frame.
- u or up arrow key increase the frame rate.
- d or down arrow key decrease the frame rate.
- l (L) key toggles loop mode on and off.
- i key gets information about the sequence.
- h key displays a usage message.
- 1 to 9 keys set zoom factor.
- Esc key exits **flipbook**.

Memory Guidelines for Using a Flipbook

Images loaded in a flipbook are not compressed, so they take up the full amount of memory for each pixel regardless of their size on the disk. The physical size of the image in pixels and the amount of RAM available determines how many images you can play back (at the specified frame rate) without skipping frames. For the flipbook to work efficiently, the images must fit into RAM (virtual memory is too slow for acceptable playback speed).

The space taken by the images should not exceed the amount of free main memory. If it does, the computer starts swapping memory to disk and the display is slowed down considerably. Use the following formula to calculate the number of bytes needed to load the images (4 refers to the four channels – RGBA):

```
4 x (height of frame) x (width of frame) x (number of frames)
```

For example, to calculate the memory required for a 256 x 256 image, the equation 256 x 256 x 4 gives you 262 144 bytes, or 256 Kbytes, so 100 images would require at least 25.6 Mbytes of RAM to display without disk swapping.

More memory would actually be required because the operating system takes up about 5 Mbytes of memory. A system with 16 Mbytes of main memory can flip approximately 100 frames of 165 x 165 resolution images.

The byte value can be converted to Kbytes by dividing it by 1024.

For displays of longer animations, flipbook can use a step frame. If every other frame was rendered using a step value of 2 and a display rate of 15 frames per second, you would have an accurate representation of the timing of the motion.

Building a Better Flipbook

This script provides a more automated way of using the **flipbook** standalone.

This is an IRIX-only example.

Enter the following lines in a new file named `myflip`, for example. Copy the file to <**install directory**>**/bin** (or wherever your executable files are located). Make sure that `myflip` is executable.

Next time, instead of using **flipbook**, use `myflip`.

```
#!/bin/csh -f
# Automatically finds start, end and increment
# of existing .pic files.
if ($#argv == 0 ) then
        echo "Usage: $0 <picfile>"
        exit
endif
set scene = $1:r; set scene = $scene:r
set files = $scene.*.pic; set file1 = ${files[1]:r}
@ min = $file1:e; @ max = $min;
                                @ second_max = $max
foreach file ($files)
        set file = $file:r
        @ frame = $file:e
        if ( $frame > $max) then
                @ second_max = $max
                @ max = $frame
        else if ($frame > $second_max) then
                @ second_max = $frame
        endif
end
@ step = $max - $second_max
set echo
flipbook $scene $min $max $step 30
#end of script
```

Displaying Images While Rendering

The **imf_disp** standalone displays image files, including memory-map texture images created by **imf_copy**. The **imf_disp** standalone displays all formats supported by the mental ray rendering software (except the SOFTIMAGE Zpic format).

This standalone runs on both the Windows NT and IRIX operating systems.

Examples

This example displays the **toto.pic** image on screen:

```
imf_disp toto
```

imf_disp automatically detects supported image file formats, so files do not require an extension (such as **.pic**) to identify their format.

imf_disp also supports image piping. Normally, mental ray prints connection information into the output image file that lets programs like **imf_disp** connect and display pictures while being rendered. If the `-imgpipe` option is used when rendering from the command line, the relevant information is printed to the given file descriptor instead:

```
ray2 -imgpipe 1 scene.mi | imf_disp-
```

Creating Thumbnail Images for Presets

The **setthumb** standalone lets you associate thumbnail icons to your user-defined presets.

This standalone runs on both the Windows NT and IRIX operating systems.

setthumb takes a bitmap file and adds it to the preset file so that when the preset is viewed in the browser (with icon mode active), its thumbnail image is displayed.

Usage

```
setthumb <preset filename> <bitmap filename>
```

Where `<preset filename>` is the name of the preset file, including the **.preset** extension, and `<bitmap filename>` is the name of the bitmap file, including the **.bmp** extension, that you want to use as the thumbnail for the preset file.

If the **setthumb** standalone cannot find the preset file in the path you have specified at the command line, it tries to locate the preset file in the default installation path for all presets (<**install path**>**\DSPresets**).

The thumbnail image must be a standard bitmap file with a width and height of 128 x 128 pixels. If the image is larger than 128 x 128 pixels, it is cropped to these dimensions.

Example

To associate a bitmap image called **myThumbnail** to a preset file called **myPreset**, type the following at the command line:

```
setthumb myPreset.preset myThumbnail.bmp
```

Getting File Information

There are a few standalones that let you get information on image files:

- Get information about a variety of image file formats with **imginfo** standalone.
- Get information about SOFTIMAGE picture files with **infopic** standalone.
- Highlight on a pixel-by-pixel basis those pixels that differ between two input images with the **diffpic** standalone.

Getting Format Information

The **imginfo** standalone can display information about a number of image formats.

This standalone runs on both the Windows NT and IRIX operating systems.

The **imginfo** standalone automatically detects supported image file formats, so input files do not require an extension (such as **.pic**) to identify their format. However, if the file on disk has an extension, you should specify it.

The **imginfo** standalone can display information for any file format for which their is a dynamic shared object (DSO on IRIX systems) or a dynamic link library (DLL on Windows NT systems). The Setup program installs these DSOs or DLLs to **<install directory>/bin/sil.** When you type `imgconv`, the usage message lists the supported file formats.

To run **imginfo**, you must set the environment variable **SI_IMAGE_PATH** to point to the **<install directory>/bin/sil** directory. If **SI_IMAGE_PATH** is not set or empty, **imginfo** checks the environment variable **SI_HOME**. If it is set, **imgconv** looks in **$SI_HOME/bin/sil**.

Usage

```
imginfo <image file>
```

Where `<image file>` is the name of an image file.

Options

`-format <name>`

Specifies the format of the input image file. This is required for images that do not have a signature.

Example

To display information on the **portrait** image file, type:

```
imginfo portrait
```

Getting Picture File Information

The **infopic** standalone displays information about SOFTIMAGE picture files.

This standalone runs on both the Windows NT and IRIX operating systems.

For each subregion, the following information is displayed:

- Width
- Height
- Aspect ratio
- Field type (FULL_FRAME, EVEN_FIELDS, or ODD_FIELDS)
- Number of bits, data type, and compression method for the RGB channels
- Number of bits, data type, and compression method for the alpha channel

The **infopic** standalone is useful for finding the pixel ratio, and to determine whether a picture is too big to display on the monitor. It is also useful for determining whether a picture corresponds to the correct format for video transfers.

Usage

```
infopic [ <file> [<file>... ]
```

Where `<file>` is the name of a SOFTIMAGE picture file. If no extension is specified, **.pic** is assumed.

Example

This example shows information on the **walkies1**, **walkies2**, and **walkies3** picture files.

```
infopic walkies1 walkies2 walkies3
```

Comparing Images

The **diffpic** standalone displays an image whose pixels are highlighted when the corresponding pixels of the two input pictures are different.

This standalone runs on both the Windows NT and IRIX operating systems.

If you use the `-i` option, the intensity of each displayed image pixel is equal to the difference of the intensity of the corresponding pixels of the two input pictures.

Usage

```
diffpic <picture 1> <picture 2> [-o <output filename>]
[-i] [-a] [-d] [-r <repeat count>]
[-s <start frame> <end frame> <step>] [-v]
```

Where <picture 1> is the name of the first picture and <picture 2> is the name of the second picture.

Options

`-o`

Saves the result in <output filename>.

`-i`

Displays intensity differences.

`-a`

Compares only the alpha channels. Both images must have alpha channel information.

`-d`

Does not display the images.

`-r`

Specifies how many times each frame should be repeated. The default is 1.

`-s`

Compares a sequence of frames.

`-v`

Turns on the verbose mode.

Example

This example compares the **tweedledee** and **tweedledum** images from frames 1 to 50 with a step of 1 (**-s 1 50 1**) and saves the results in the output file called **tweedles** (**-o**).

```
diffpic tweedledee tweedledum -o tweedles -s 1 50 1
```

Compositing Images

The **composite** standalone composites many images into one.

This standalone runs on both the Windows NT and IRIX operating systems.

When compositing sequences, you can add a single still frame as a layer by appending the extension **.0** (zero) to the layer name.

For field-rendered image sequences, you can composite the odd-field or even-field dominated image, or both. For example, to composite the even-field image, you would type:

```
composite output -E
```

where **output** is your rendered image.

Usage

```
composite [-a] [-b <red> <green> <blue>]
[-S <width> <height>] [-p <pixel ratio>] <output file>

[-r <repeat count>]
[-s <start frame> <end frame> <step>] [-d] [-v] [-Z]

[-F|-E|-O ] { <layer> [<start frame>] [-F|-E|-O ]
[ -c | -o | -f <fact> [-e <end fact>] | -h <radius>
<fact> | -B | -C | -D | -N | -P ] [-x <xoff>]
[-y <yoff>] | [-X <end xoff>] [-Y <end yoff>] ...}
```

Where `<output file>` is the name you want to give to the final picture and `<layer>` is the name of a layer to add to the final image. Use the corresponding `<start frame>` to specify the starting frame of the `<layer>` sequence.

When compositing sequences, you can add a single still frame as a layer by appending the extension .0 (zero) to the layer name.

Options

`-a`

Omits the alpha channel in the output. By default, the alpha channel is included in the output.

`-b`

Sets the background color (default values: 0.0, 0.0, 0.0). Each color value must be in the range [0,1].

`-S`

Specifies the size of the output file. The default is the size of the first layer. If the first layer is a subregion rendering, the default is the full image resolution.

`-p`

Specifies the output picture pixel ratio.

`-r`

Specifies how many times each frame should be repeated. The default is 1.

`-s`

Composites a sequence of frames.

`-d`

Displays the result on screen.

`-v`

Turns on the verbose mode.

`-Z`

Uses the Z channel for compositing. To use this option, you must render each sequence using the Z channel and have a corresponding **.Zpic** file for each layer. Do not type the extension at the end of the layer name. In addition, you must specify the `-c` option for all layers that you want to composite.

`-F`

Specifies full frame images. This is the default.

`-E`

Specifies field images with even field dominance.

`-O`

Specifies field images with odd field dominance.

`-c`

Composites this layer. This is the default for all layers after the first, unless the `-Z` option is specified.

`-o`

Overlaps this layer. This is the default for the first layer.

`-f`

Fades this layer, and specifies an initial fading factor.

`-e`

Varies the fading factor continuously from frame to frame. You must specify the fading factor for the final frame.

`-h`

Creates a halo on the foreground where its contour pixels are more intense than their adjacent background pixels. `<radius>` is the width of the halo, and `<fact>` is the blending factor which determines the strength of the halo effect.

`-B`

Blends the layer's intensity with the background's intensity, according to the layer's alpha channel.

`-C`

Composites the layer with the background by taking the brightest of the corresponding pixels.

`-D`

Generates the difference between the layer and the background.

`-N`

Composites the background and the layer as if two superimposed negatives were printed.

`-P`

Composites the background and the layer as if they were two superimposed slides.

`-x`

Offsets the picture from the left of the screen by the amount specified in pixels.

`-y`

Offsets the picture from the bottom of the screen by the amount specified in pixels.

`-X`

Picture translation in pixels from the left of the screen (the default is zero).

`-Y`

Picture translation in pixels from the bottom of the screen (the default is zero).

Examples

The following example simply merges two layers to give a straightforward composite. The result is called **imageA**. The background image is **imageB.1** and the foreground image is **imageC.1**. The **-v** option displays information while the standalone is running.

```
composite imageA imageB.1 imageC.1 -v
```

The following command composites two sequences of field images with even field dominance:

```
composite output -s 1 10 1 -E layer1 -E layer2 -E
```

The following example composites **background** and **foreground** as if they were two superimposed slides, and saves the result as **finalimage**.

```
composite finalimage background foreground -P
```

The following example shows how to crop 100 pixels from the edges of a 1000 x 1000 pixel image, leaving the center 800 x 800 pixels.

```
composite -S 800 800 output input -x 100 -y 100
```

The following example fades from **scene_1** to **scene_2** for 100 frames. The mix begins with frame 500 of **scene_1** and frame 1 of **scene_2**.

```
composite output -s 1 100 1 scene_1 500 scene_2 1
-f 0 -e 1
```

The following example composites two image files (**tore.pic** and **cone.pic**) using information in the Z channels (**-Z**). It offsets the first layer by 100 pixels from the left and 100 pixels from the bottom, and offsets the second layer by 50 pixels from the left and 50 pixels from the bottom. It also displays verbose messages (**-v**) while running.

```
composite result -Z tore -c -x 100 -y 100 cone -c
-x 50 -y 50 -v
```

The following example composites two sequences of images (**spin1** and **spin2**) with a step of 2; this means that images **spin1.1.pic**, **spin1.3.pic**, etc., is used. The result is a sequence of images named **res.1.pic**, **res.3.pic**, etc. The other options specify to use Z channels
(**-Z**), to run in verbose mode (**-v**), and to display the final image.

```
composite res -d -Z -s 1 25 2 spin1 -c spin2 -c -v
```

The following example composites **imageB.1** onto a white background in video size. The new image is called **imageA**. The **-c** option is needed because the first layer is not composited by default.

```
composite -S 720 486 -b 1 1 1 imageA imageB.1 -c -v
```

The following example crops a full 720 x 486 x 0.9 image (**logo.1**) to a 720 x 60 x 0.9 image, and writes this to a file called **logo.final**.

```
composite -S 720 60 logo.final logo.1
```

The following example composites **logo.shadow** and **logo.original**, and creates **logo.final**. The option **-b 1 1 1** colors the background white. The layer **logo.shadow** is composited using the **-c** option to force use of the alpha channel over the colored background. The **-x** and **-y** parameters specify offsets for **logo.shadow**. The layer **logo.original** does not require the **-c** option because it is the second layer specified and is therefore composited by default.

```
composite -b 1 1 1 logo.final logo.shadow -c -x 2
-y -2 logo.original
```

The following examples show the appropriate parameters for cropping **imageA.1** to the appropriate size for the safe title area at different resolutions.

```
composite -S 1024 70 logo1k imageA.1
composite -S 2048 140 logo2k imageA.1
composite -S 4096 280 logo4k imageA.1
composite -S 6144 420 logo6k imageA.1
composite -S 8192 560 logo8k imageA.1
```

Viewing and Compositing Field Images

You can use **composite** to view and composite field-rendered pictures and sequences. For example, the following command composites two sequences of field images:

```
composite output -s 1 10 1 -E layer1 -E layer2 -E
```

The output of this command is also a sequence of field images—you can view the output sequence using the **flipbook** standalone:

```
flipbook output 1 -E
```

To view the first image in the sequence, use the **showpic** or **display** standalone:

```
showpic output.1 -E
display output.1 -E
```

Interleaving Fields

The **interleave** standalone merges pictures together by interleaving their fields (scan lines).

This standalone runs on both the Windows NT and IRIX operating systems.

The **interleave** standalone allows you to accomplish the task of interleaving picture files outside the interface. You can also perform this task from within the SOFTIMAGE|XSI interface using the built-in interleave function, but you may choose to use the standalone for more control or speed—the choice is yours.

For more information on interleaving fields within SOFTIMAGE|XSI, see *Field Rendering* on page 70 of the *Rendering* guide.

Usage

```
interleave <input file 1> [<input file 2>]
<output file> [-e | -o | -i] [-f] [-r <repeat count>]
[-s <start frame> <end frame> <step>] [-v]
```

Where `<input file 1>` is the name of the first picture and `<input file 2>` is the name of the second picture. This parameter is optional when using the `-s` option. `<output file>` is the name you want to give to the result.

By default, the first scan line of the output file is the first scan line of the first input file. You can change this by specifying the options [`-e`| `-o`|`-i`].

Options

`-e`

Uses the first scanline of the input picture containing the even fields for the first scanline of the output.

`-o`

Uses the first scanline of the input picture containing the odd fields for the first scanline of the output.

`-i`

Can be used only when both a single input file is given and the -s option is specified.

If this option is used, the first scanline of the even-numbered frames becomes the first scanline of the output.

If this option is not used, the first scanline of the odd-numbered frames becomes the first scanline of the output by default.

`-f`

Reads field frames (such as default.1.1.pic, default.1.2.pic).

`-r`

Specifies how many times each frame should be repeated. The default is 1.

`-s`

Interleaves a sequence of frames. If only one input sequence name is given, the odd-numbered scanlines of the odd-numbered frames interleave with the even- numbered scanlines of the even-numbered frames (assuming you start with scanline 1, not 0, and the step is 2). For example, the odd scanlines of frame 1 interleave with the even scanlines of frame 2, similarly for frames 3 and 4, and so on. In this way, a sequence of 100 frames becomes a sequence of 50 frames.

`-v`

Activates verbose output.

Example

For example, to interleave the image **house1** and **house2** from frames 1 to 300 with a step of 2 (`-s 1 300 2`) with the resulting image called **houses**, you would type:

```
interleave house1 house2 houses -s 1 300 2
```

The name of the second picture is optional when using the `-s` option.

Here is another example which interleaves the files **spherelayer1** and **cylinderlayer2** with the resulting output as **movecomposite**. The operation is done from frames 1 and 100 with a step of 1, the field frame (`-f`) is read, and verbose mode is on (`-v`).

```
interleave spherelayer1 cylinderlayer2 movecomposite -
s 1 100 1 -f -v
```

Creating Gradations

The **gradation** standalone creates a gradation between colors and saves it as a picture. You can vary the colors horizontally, vertically, or between all four corners.

The color values must be in the range [0..1].

This standalone runs on both the Windows NT and IRIX operating systems.

Usage

```
gradation <output> <width> <height>

[-V <start r0> <start g0> <start b0> <start r1> <start
g1> <start b1> |

-H <start r0> <start g0> <start b0> <start r1> <start
g1> <start b1> |

-A <start r0> <start g0> <start b0> <start r1> <start
g1> <start b1> <start r2> <start g2>

<start b2>  <start r3> <start g3> <start b3>]

[-e <end r0> <end g0> <end b0>  <end r1> <end g1> <end
b1> [<end r2> <end g2> <end b2>  <end r3>

<end g3> <end b3>]]

[-r <repeat count>] [-s <start frame> <end frame>
<step>] [-V]
```

Where `<output>` is the name you want to give to the final picture and `<width>` is the width of the picture in pixels. `<height>` is the height of the picture in pixels.

Options

`-V`

Produces a gradation composed of vertical lines of uniform color. The first set of RGB values corresponds to the left, and the second set to the right.

`-H`

Produces a gradation composed of horizontal lines of uniform color. The first set of RGB values corresponds to the top, and the second set to the bottom.

`-A`

Produces a gradation among all four corners. The first set of RGB values corresponds to the upper-left corner, the second set to the upper-right corner, the third set to the lower-left corner, and the fourth set to the lower-right corner.

`-e`

Modifies the colors continuously over a sequence. Specify the RGB values for the last frame. If you are using the -V or -H options, you must specify two sets of RGB values; if you are using the -A option, you must specify four sets of RGB values.

`-r`

Specifies how many times each frame should be repeated. The default is 1.

`-s`

Creates a sequence of gradations.

`-v`

Activates verbose output. Make sure to use a lower-case v for the verbose option, since the upper-case V is used for the vertical option.

Example

In this example, **finalgrad** is the output image with a width of 150 pixels and a height of 150 pixels. The gradation is made of horizontal lines (-**H**) with the values defined for the top (0, 0.15, 0.15) as a greenish color and then the bottom (0.6, 0.45, 0) for a yellowish color.

```
gradation finalgrad 150 150 -H 0 0.15 0.15
0.6 0.45 0
```

Creating 2D Motion Blur

The **mb2d** standalone receives frame buffers containing beauty (color), mask, depth and motion information. From this information, it selectively blurs the image, producing a motion blur effect.

The **mb2d** standalone provides several parameters which control the duration and the quality of the motion blur:

- Shutter length specifies the amount of time the shutter is open to control the length of the motion blur.
- The type parameter specifies the blur algorithm (0=uniform, 1=linear decay, 2=Gaussian decay) to be used.
- Samples indicates the number of lines drawn per motion blurred pixel.

This standalone runs on both the Windows NT and IRIX operating systems.

The **mb2d** standalone allows you to accomplish the task of creating a post-process motion blur outside the SOFTIMAGE|XSI interface. You can also perform this task from within the interface using the 2D motion blur output shader available from the shader library, but you may choose to use the standalone for more control or speed—the choice is yours.

For information on full 3D motion blur within SOFTIMAGE|XSI, see *Creating Motion Blur* on page 187 of the *Shaders, Lights & Cameras* guide.

Usage

There are several command-line options that can be given before the name of the beauty input and output files. **mb2d** is typically invoked as follows:

```
mb2d [options] beauty depth motion beautyOut
[depthOut]
```

`<beauty>` is the input image beauty file.

`<depth>` is the input image depth file.

`<motion>` is the input image motion file.

`<beautyOut>` is the output image beauty file.

`<depthOut>` is the output image depth file.

Alternatively, you may specify an options file. Any options specified on the command line override the corresponding options read from the options file. Below is an example of how to specify the options file:

```
mb2d -o mb2d-options.1 beauty depth motion beautyOut
```

Note that the file name is `mb2d-options` followed by a dot and the file ID number. See *Options File Grammar* on page 258 for a description of the options file syntax.

Options

`- aperture <scalar>`

Specifies the camera aperture. Default setting is 1.

`- aspect <scalar>`

specifies the camera aspect ratio. Default setting is 1.33.

`- focal <scalar>`

Sets the camera focal length. Default setting is 1.

`- help`

Prints help for all options to screen.

`-o <name>`

Uses the options file specified by name.

`- samples <integer>`

Sets the number of lines drawn per motion blurred pixel. Values greater than 1 will antialias the blurred pixels. Default setting is 1.

`- shutter <scalar>`

Sets the length of time the shutter is open. This controls the length of the motion blur. Valid range is 0 to 1. Default max setting is 1.

`- t <scalar> <scalar> <scalar> <scalar> <scalar> <scalar> <scalar> <scalar> <scalar>`

Specifies motion vector or 3 x 3 transform.

`-trail <scalar>`

Specifies the motion trail proportional length. Valid range is 0 to 1. The other portion (1-trail) will be head motion.

`- type <integer>`

Sets the blur algorithm to be used (0=uniform, 1=linear decay, 2=Gaussian decay).

`- v`

Activates verbose output.

Options File Grammar

This section contains the formal syntax of the mental images® motion blur options file format. The grammar is in **yacc** format.

```
%start options_file
%token MB_APERTURE
%token MB_ASPECT
%token MB_FOCAL
%token MB_SAMPLES
%token MB_SHUTTER
%token MB_TYPE
%token MB_TRANSFORM
%token MB_SCALAR
%%
options_file : commands
;
commands : command
| commands command
;
command : aperture
| aspect
| focal
| samples
| shutter
| type
| transform
;
aperture : MB_APERTURE scalar
;
aspect : MB_ASPECT scalar
;
focal : MB_FOCAL scalar
;
samples : MB_SAMPLES scalar
;
shutter : MB_SHUTTER scalar
;
type : MB_TYPE scalar
;
transform : MB_TRANSFORM scalar scalar scalar scalar
scalar scalar scalar scalar scalar
;
scalar : MB_SCALAR
;
%%
```

Recording Images

SOFTIMAGE|XSI supplies a set of standalones for hardware devices to which you can record images:

- Record image sequences to an Accom WSD with the **wsd_record** standalone.
- Record picture files to a Solitaire with the **FilmSolitaire** standalone.
- Record images to video with devices using Cindy, Centaur, or Galileo boards with the **civa_record**, **cva_record**, or **gv_record** standalones.
- Specify the digital disk recorder to which you are recording with the **ddr_rec** standalone.

These standalones are described in this section.

Recording Images to an Accom WSD

The **wsd_record** standalone records a sequence of images to the Accom WSD.

This standalone runs on both the Windows NT and IRIX operating systems.

Usage

```
wsd_record <file> <start> <end> <step> <repetition>
<location> [-b] [-m] [-d <devname>] [-p <path>]
```

`<file>` is the name of the file to record.

`<start>` is the first frame of the sequence.

`<end>` is the last frame of the sequence.

`<step>` is the step increment for the sequence.

`<repetition>` is the number of frames to record per image.

`<location>` is the starting location.

Options

`-b`

Uses true black where the image is completely transparent (alpha component = 0).

`-m`

Records the image mask (alpha channel).

`-d`

Sets the SCSI device name.

`-p`

Specifies the path for the Accom.rsrc file.

Example

This example records the **fish.pic** file from frame 1 to 120 with a step of 2 and frames repeated only once starting recording at time code 00:00 on the Accom. It also records the alpha channel values (**-m**).

```
wsd_record fish 1 120 2 1 00:00 -m
```

Recording to Film on a Solitaire

The **filmSolitaire** standalone sends a picture file to the Solitaire 8 and 8XP film recorders. Only the SCSI interface is supported.

This standalone runs on both the Windows NT and IRIX operating systems.

Usage

```
filmSolitaire <file> [-q <start> <end> <step>
<repeat>][-w [<path>]] [-x <xoff>] [-y <yoff>] [-c] [-
r <res>] [-p] [-g <gamma>] [-l {1|2|3|9 <lut
filename>}] [-s] [-f][-z <factor> [<blur>]] [-C] [-e
[<format>]] [-n <copies>][-a] [-R] [-i <red > <green>
<blue> <alpha>] [-S] [-d] [-sc <shell script name>] [-
A <nbfr>] [-B <nbfr>] [-b]
```

Where `<file>` is the name of the picture and `<path>` is the path name of the resource directory.

Options

`-q`

Sends a sequence of frames.

`-w`

Specifies dialogue mode.

`-x`

Sets the horizontal offset.

`-y`

Sets the vertical offset.

`-c`

Centers the picture (default = off). If you omit this option, the image may appear at the left or right edge of the screen, depending on the image format.

This option, followed by an offset x or y, moves the origin to the lower-left corner of the centered picture. Without the option `-c`, the origin is set at the lower-left corner of the crt.

`-r`

Sets the resolution. Possible values for `<res>` are 2, 4 and 8 (default = 2).

`-p`

Turns pixel replication on (default = off).

`-g`

Sets the gamma correction (default = 1).

On the camera, an increased gamma value gives a darker picture, while a decreased gamma gives a lighter picture. This is the opposite of what happens on the screen.

`-l`

Sets the LUT (film format look-up table) type. If you do not set a LUT type, the default LUT is used. Use one of the following numbers to specify the LUT type:

1 EK100 (4x5)
2 EK100+
3 LINEAR
9 Custom. You must include the name of a custom LUT file.

`-s`

Displays the output on screen. Default is off.

`-f`

Uses a full handshake. Defaultis off. For more information, refer to the *Solitaire Programmer's Guide*.

`-z`

Zooms the image by the specified zoom factor. `<fact>` can be any positive value. If desired, you can also blur the image by the specified amount. The default `<blur>` is 1; the minimum `<blur>` is 0.2.

`-C`

Does not calibrate the Solitaire. If this parameter is omitted, the Solitaire is calibrated only once.

`-e`

Specifies the format. The image is automatically zoomed to the correct resolution. Use one of the following numbers to specify the corresponding resolution:

1 (35mm slide, ratio 1.5)
2 (35mm cine acad, ratio 1.37)
3 (35mm cine, ratio 1.66)
4 (4" x 5", ratio 1.25)

If you use the `-e` option without specifying a number, the default is 4.

`-n`

Sets the number of copies (default = 1).

`-a`

Reads one color at a time.

`-R`

Removes the frame after it has been exposed.

`-i`

Sends the specified Calset (color balance setup). The values depend on your film stock.

`-S`

Returns the Solitaire setup.

`-d`

Turns verbose mode on.

`-sc`

Executes the specified shell script before each exposure.

`-A`

Advances the film recorder by the specified number of frames.

`-B`

Rewinds the film recorder by the specified number of frames.

`-b`

Enables double-buffering mode (ring buffer).

Example

The following example sends frames of **myFile** to the Solitaire 8.

```
filmSolitaire myFile -q 1 100 1 1 -c -g .75 -i 184 164
159 159 -d
```

- The four numbers after **-q** specify to record the first 100 frames without repetition.
- The **-c** option centers the image on the screen.
- The number after the **-g** option sets the gamma value for the light exposure.
- The numbers after **-i** calibrate the colors for the particular film being used.
- The **-d** option provides messages while the process is running.

Recording Images to Video Using a Cindy

The **civa_record** standalone records a series of images to a video device (Betacam, Laser, or Hi8) in any of four output formats. It is designed for use with the Cindy board.

This standalone runs only on the IRIX operating system.

Usage

```
civa_record <file> <start> <end> <step> <repetition>
<frame> <format>
[-b <red> <green> <blue>] [-x <xoff>] [-y <yoff>]
[-n <node>] [-m] [-T] [-z] [-v <videotype>] [-d]
[-p <preroll>] [-f] [-S <script>]
```

`<file>` is the name of the sequence.

`<start>` is the first frame of the sequence.

`<end>` is the last frame of the sequence. If `<end>` is less than `<start>`, the frames are recorded in reverse order.

`<step>` is the step increment for the sequence.

`<repetition>` is the number of frames to record per image.

`<frame>` is the starting time code (without the colon ":"). If you use the `-v` laser option without the `-T` option, `<frame>` is the linear frame number instead.

`<format>` specifies the format. Use NTSC or PAL for composite analog, and ntsc_d1 or pal_d1 for component digital.

Options

`-b`

Sets background color. This option does not use the alpha channel. The background color appears a border or fill if the image is smaller than the nominal resolution.

`-x`

Offsets the image horizontally by `<xoff>` pixels.

`-y`

Offsets the image vertically by `<yoff>` pixels.

`-n`

Sets the V-LAN node number (default = 1).

`-m`

Displays the image mask.

`-T`

Specifies time code mode instead of frame number mode for the Laser VideoDisc. This option can only be used with the `-v` laser option.

`-z`

Disables initialization of the Cindy at startup. Use this option to ensure that the color framing does not change during a transfer session.

`-v`

Specifies the video type. Use one of the following parameters (default = betacam).

Video type	Description
betacam	Videocassette BVW-XX
laser	Laser VideoDisc LVR-XXXX
Hi8	Videocassette Hi8 EVO-XXXX

`-d`

Uses double-buffering (with V-LAN only).

`-p`

Sets the preroll. Use the format XX:XX to specify the preroll amount (default = 5:00).

`-f`

Records fields from separate files.

`-S`

Uses the specified shell script.

Example

This example records the **cat.pic** file from frames 1 to 500 at a step of 2 with each frame repeated once. The recording starts at time code 00:00 (no colon used in the syntax) on the device and the format is NTSC.

```
civa_record cat 1 500 2 1 0000 ntsc
```

Recording Images to Video Using a Centaur

The **cva_record** standalone records a series of images to a video device (Betacam, Laser, or Hi8) in any of four output formats. It is designed for use with the Centaur board.

This standalone runs only on the IRIX operating system.

Usage

```
cva_record <file> <start> <end> <step> <repetition>
```

```
<frame> <format> [-b <red> <green> <blue>] [-x <xoff>]
[-y <yoff>] [-n <node>] [-m] [-T] [-z] [-v
<videotype>] [-d] [-p <preroll>] [-f] [-S <script>]
```

`<file>` is the name of the sequence.

`<start>` is the first frame of the sequence.

`<end>` is the last frame of the sequence. If `<end>` is less than `<start>`, the frames are recorded in reverse order.

`<step>` is the step increment for the sequence.

`<repetition>` is the number of frames to record per image.

`<frame>` is the starting time code (without the colon ":"). If you use the `-v` laser option without the `-T` option, `<frame>` is the frame number instead.

`<format>` specifies the format. Use NTSC or PAL for composite analog, and ntsc_d1 or pal_d1 for component digital.

Options

`-b`

Sets background color. This option does not use the alpha channel. The background color appears as a border or fill if the image is smaller than the nominal resolution.

`-x`

Offsets the image horizontally by `<xoff>` pixels.

`-y`

Offsets the image vertically by `<yoff>` pixels.

`-n`

Sets the V-LAN node number (default = 1).

`-m`

Records the alpha channel as a gray-scale representation. Opaque areas are white and transparent areas are black.

`-T`

Specifies time code mode instead of frame number mode for the Laser VideoDisc. This option can only be used with the `-v` laser option.

`-z`

Disables initialization of the Centaur at startup.

`-v`

Specifies the video type. Use one of the following parameters (default = betacam).

Video type	Description
betacam	Videocassette BVW-XX
laser	Laser VideoDisc LVR-XXXX
Hi8	Videocassette Hi8 EVO-XXXX

`-d`

Uses double-buffering (with V-LAN only).

`-p`

Sets the preroll. Use the format XX:XX to specify the preroll amount (default = 5:00).

`-f`

Records fields from separate files.

`-S`

Uses the specified shell script.

Example

This example records the **cat.pic** file from frames 1 to 500 at a step of 2 with each frame repeated once. The recording starts at time code 00:00 (no colon used in the syntax) on the device and the format is NTSC.

```
cva_record cat 1 500 2 1 0000 ntsc
```

Recording Images to Video Using a Galileo

The **gv_record** standalone records a series of images to a video device (Betacam, Laser, or Hi8) in any of 12 output formats. It is designed for use with the Galileo board.

This standalone runs only on the IRIX operating system.

Usage

```
gv_record <file> <start> <end> <step> <repetition>
<frame> <format> [-d] [-n <node>] [-p <preroll>]
[-v <videotype>]
```

`<file>` is the name of the sequence.

`<start>` is the first frame of the sequence.

`<end>` is the last frame of the sequence. If `<end>` is less than `<start>`, the frames will be recorded in reverse order.

`<step>` is the step increment for the sequence.

`<repetition>` is the number of frames recorded per image in the sequence.

`<frame>` is the starting time code (without the colon ":").

`<format>` specifies the format. Use one of the following format identifiers.

NTSC	PAL	
rgb_525	rgb_625	analog RGB
beta_525	beta_625	betacam YUV
m2_525	m2_625	MII YUV
smpte_525	smpte_625	S MPTE YUV
svhs_525	svhs_625	SVIDEO
comp_525	comp_625	Composite
d2_525	d2_625	Composite Digital
d1_par_525	d1_par_625	d1 Parallel
d1_ser_525	d1_ser_625	d1 Serial

Options

`-d`

Turns debug mode on.

`-n`

Sets the V-LAN node (default = 1).

`-p`

Sets the preroll. Use the format XX:XX to specify the amount (default = 5:00).

`-v`

Specifies the video type. Use one of the following video type identifiers (default = betacam).

Video type	Description
betacam	Videocassette BVW-XX
laser	Laser VideoDisc LVR-XXXX
Hi8	Videocassette Hi8 EVO-XXXX

Example

This example records the **cat.pic** file from frames 1 to 500 at a step of 2 with each frame repeated once. The recording starts at time code 00:00 (no colon used in the syntax) on the device and the format is RGB for NTSC.

```
gv_record cat 1 500 2 1 0000 rgb_525
```

In some cases, you may want to use external encoding, particularly if a high quality external hardware encoding device is available. For external encoding, use the **rgb_*** format identifier.

Recording Image to Video Using any DDR

The **ddr_rec** standalone records SOFTIMAGE picture files (**.pic**) as frames on a Digital Disk Recorder (DDR).

This standalone runs on both the Windows NT and IRIX operating systems.

The DDR name must be defined in the winnt\system32\drivers\etc\hosts file on Windows NT or in the /etc/hosts file on IRIX.

At the command line, type one of the following:

- `ddr_rec -i` to display a series of interactive prompts.

  or

- `ddr_rec` or `ddr_rec -h` to display a list and definitions of all required parameters and options.

Before You Begin

To use the **ddr_rec** standalone, the following conditions must apply:

- The TCP/IP protocol must be installed.
- The names and IP addresses of your DDR device must be defined in the winnt\system32\drivers\etc\hosts file on Windows NT or in any valid DNS server. On IRIX, the DDR names must be defined in /etc/hosts.

To check that your DDR is properly connected, open a shell and type:

```
ping <DDR name>
```

where `<DDR name>` has been defined in the hosts file.

Setting the Default DDR Name

The **ddr_rec** standalone assumes that you are using an Accom DDR and looks for Accom among the IP addresses. To use another type of DDR, you can either:

- Use the `-a <DDR name>` option (such as `-a abekas`) when invoking the standalone.

  or

- Add the desired DDR name to the IP address line in the hosts file for a more permanent solution. For example:

  ```
  <IP address> accom abekas
  ```

This means that the standalone accepts both Accom and Abekas as the device name.

Usage

```
ddr_rec <input> <start> <end> <step> <frame>
[-a <DDRname>] [-b] [-m] [-v]
```

where `<input>` is the name of the SOFTIMAGE picture file (**.pic**) without the .pic extension.

`<start>` is the first frame of the sequence.

`<end>` is the last frame of the sequence.

`<step>` is the frame step of the sequence.

`<frame>` is the frame location on the DDR to start recording.

Options

`- a <DDR name>`

Specifies the name of a DDR device if this is other than Accom.

`-b`

Specifies True Black.

`-m`

Outputs only the alpha channel of the specified frames.

`-v`

Activates verbose output.

Example

```
ddr_rec image 1 50 2 100 -a abekas
```

Records SOFTIMAGE picture file **image.1.pic** to **image.50.pic** with a step of 2 to frame 100 on a DDR called abekas.

Grabbing Images

SOFTIMAGE|XSI supplies a set of standalones for hardware devices from which you can grab images:

- Grab image sequences from an Accom WSD with the **wsd_grab** standalone.
- Grab images from video devices using Cindy, Centaur, or Galileo boards with the **civa_grab**, **cva_grab**, or **gv_grab** standalones.
- Specify the digital disk recorder from which you are grabbing images with the **ddr_grab** standalone.

These standalones are all described in this section.

Grabbing from an Accom WSD

The **wsd_grab** standalone grabs a sequence of images from the Accom WSD and saves it to disk.

This standalone runs on both the Windows NT and IRIX operating systems.

Usage

```
wsd_grab <file> <start> <end> <step> <location>
[-d <devname>] [-p <path>]
```

`<file>` is the name you want to give to the file.

`<start>` is the first frame of the sequence.

`<end>` is the last frame of the sequence.

`<step>` is the step interval for the sequence.

`<location>` is the starting location.

Options

`-d`

Specifies the SCSI device name.

`-p`

Specifies the path for the Accom.rsrc file.

Example

This example grabs a sequence from the Accom WSD and saves it as **fish.pic**. The sequence is from frame 1 to 120 with a step of 2, and starting at time code 00:01 on the Accom.

```
wsd_grab fish 1 120 2 00:01
```

Grabbing Images from Video Using a Cindy

The **civa_grab** standalone grabs a series of images from a video device (Betacam, Laser, or Hi8) in any of four input formats. It is designed for use with the Cindy board.

This standalone runs only on the IRIX operating system.

Usage

```
civa_grab <file> <start> <end> <step> <frame> <format>
[-s <pixels>] [-t <pixels>] [-n <node>] [-z]
[-p <preroll>] [-c <phase>] [-r <RGB_max> <RGB_min>]
[-C <contrast>] [-v videotype] [-f <field>]
```

`<file>` is the name you want to give to the sequence.

`<start>` is the first frame of the sequence.

`<end>` is the last frame of the sequence.

`<step>` is the step increment for the sequence.

`<frame>` is the starting time code (without the colon ":").

`<format>` specifies the format. Use NTSC or PAL for composite analog, and ntsc_d1 or pal_d1 for component digital.

Options

`-s`

Horizontally scales the grabbed image to the size specified in pixels.

`-t`

Trims the right edge of the image by the amount specified in pixels.

`-n`

Sets the V-LAN node (default = 1).

`-z`

Disables initialization of the Cindy at startup. Use this option to ensure that the color framing does not change during a transfer session.

`-p`

Sets the preroll (default = 5:00).

`-c`

Adjusts the picture's horizontal phase (default = 35).

`-r`

Specifies the RGB references for contrast and brightness. Lower values for <RGB_max> increase brightness, and higher values for <RGB_min> increase contrast (defaults: <RGB_max> = 63, <RGB_min> = 0).

`-C`

Adjusts the picture's contrast (default = 100).

`-v`

Specifies the video type. Use one of the following video type identifiers (default = betacam).

Video type	Description
betacam	Videocassette BVW-XX
laser	Laser VideoDisc LVR-XXXX
Hi8	Videocassette Hi8 EVO-XXXX

`-f`

Specifies field capturing. Use one of the following identifiers (default = 0).

0 (both fields)
1 (first field)
2 (second field)

Example

This example grabs frames 5 to 100 at a step of 1 and saves it to a file called **dog.pic**. The grabbing starts at time code 00:20 (no colon used in the syntax) on the device and the format is PAL.

```
civa_grab dog 5 100 1 0020 pal
```

Grabbing Images from Video Using a Centaur

The **cva_grab** standalone grabs a series of images from a video device (Betacam, Laser, or Hi8) in any of four input formats. It is designed for use with the Centaur board.

This standalone runs only on IRIX.

Usage

```
cva_grab <file> <start> <end> <step> <frame> <format>

[-s <pixels>] [-t <pixels>] [-n <node>] [-z]
[-p <preroll>] [-c <phase>] [-r <RGB_max> <RGB_min>]

[-C <contrast>] [-v videotype] [-f <field>]
```

<file> is the name you want to give to the sequence.

<start> is the first frame of the sequence.

<end> is the last frame of the sequence.

<step> is the step increment for the sequence.

<frame> is the starting time code (without the colon ":").

<format> specifies the format. Use NTSC or PAL for composite analog, and ntsc_d1 or pal_d1 for component digital.

Options

-s

Horizontally scales the grabbed image to the size specified in pixels.

-t

Trims the right edge of the image by the amount specified in pixels.

-n

Sets the V-LAN node (default = 1).

-z

Disables initialization of the Centaur at startup.

-p

Sets the preroll (default = 5:00).

-c

Adjusts the picture's horizontal phase (default = 35).

-r

Specifies the RGB references for contrast and brightness. Lower values for <RGB_max> increase brightness, and higher values for <RGB_min> increase contrast (defaults: <RGB_max> = 63, <RGB_min> = 0).

-C

Adjusts the picture's contrast (default = 100).

-v

Specifies the video type. Use one of the following video type identifiers (default = betacam).

Video type	Description
betacam	Videocassette BVW-XX
laser	Laser VideoDisc LVR-XXXX
Hi8	Videocassette Hi8 EVO-XXXX

`-f`

Specifies field capturing. Use one of the following identifiers (default = 0).

0 (both fields)
1 (first field)
2 (second field)

Example

This example grabs frames 5 to 100 at a step of 1 and saves it to a file called **dog.pic**. The grabbing starts at time code 00:20 (no colon used in the syntax) on the device and the format is PAL.

```
cva_grab dog 5 100 1 0020 pal
```

Grabbing Images from Video Using a Galileo

The **gv_grab** standalone grabs a series of images from a video device (Betacam, Laser, or Hi8) in any of 12 input formats. It is designed for use with the Galileo board.

Usage

```
gv_grab <file> <start> <end> <step> <frame> <format>
[-n <node>] [-p <preroll>] [-v <videotype>] [-T] [-d]
[-t]
```

`<file>` is the name you want to give to the sequence.

`<start>` is the first frame of the sequence.

`<end>` is the last frame of the sequence.

`<step>` is the step increment for the sequence.

`<frame>` is the starting time code (without the colon ":"). If you use the `-v` laser option without the `-T` option, `<frame>` is the frame number instead.

`<format>` specifies the format. Use one of the following format identifiers:

NTSC	PAL	
rgb_525	rgb_625	analog RGB
beta_525	beta_625	betacam YUV
m2_525	m2_625	MII YUV

NTSC	PAL	
smpte_525	smpte_625	S MPTE YUV
svhs_525	svhs_625	SVIDEO
comp_525	comp_625	Composite
d2_525	d2_625	Composite Digital
d1_par_525	d1_par_625	d1 Parallel
d1_ser_525	d1_ser_625	d1 Serial

To use the `-t` option, you require a special cable from SGI unless you are grabbing an image in one of the **d1** formats.

Options

`-n`

Sets the V-LAN node (default = 1).

`-p`

Sets the preroll. Use the format XX:XX to specify the amount (default = 5:00).

`-v`

Specifies the video type. Use one of the following video type identifiers (default = betacam).

Video type	Description
betacam	Videocassette BVW-XX
laser	Laser VideoDisc LVR-XXXX
Hi8	Videocassette Hi8 EVO-XXXX

`-T`

Specifies time code mode instead of frame number mode for the Laser VideoDisc. This option can only be used with the `-v` laser option.

`-d`

Turns debug mode on.

`-t`

Waits for an external trigger signal. To use the `-t` option, you require a special cable from SGI unless you are grabbing an image in one of the d1 formats.

Example

This example grabs frames 5 to 100 at a step of 1 and saves it to a file called **dog.pic**. The grabbing starts at time code 00:20 (no colon used in the syntax) on the device and the format is RGB for PAL.

```
gv_grab dog 5 100 1 0020 rgb_625
```

Grabbing from Any DDR

The **ddr_grab** standalone grabs frames from a Digital Disk Recorder (DDR) and converts them to SOFTIMAGE picture (.pic) files.

This standalone runs on both the Windows NT and IRIX operating systems.

The DDR name must be defined in the winnt\system32\drivers\etc\hosts file on Windows NT or in the /etc/hosts file on IRIX.

At the command line, type one of the following:

- `ddr_grab -i` to display a series of interactive prompts.

  or

- `ddr_grab` or `ddr_grab -h` to display a list and definitions of all required parameters and options.

Before You Begin

To use the **ddr_grab** standalone, the following conditions must apply:

- The TCP/IP protocol must be installed.
- The names and IP addresses of your DDR device must be defined in the winnt\system32\drivers\etc\hosts file on Windows NT or in any valid DNS server. On IRIX, the DDR names must be defined in /etc/hosts.

To check that your DDR is properly connected, open a shell and type:

```
ping <DDR name>
```

where `<DDR name>` has been defined in the hosts file.

Setting the Default DDR Name

The **ddr_grab** standalone assume that you are using an Accom DDR and looks for Accom among the IP addresses. To use another type of DDR, you can either:

- Use the `-a <DDR name>` option (such as `-a abekas`) when invoking the standalone.

  or

- Add the desired DDR name to the IP address line in the hosts file for a more permanent solution. For example:

  ```
  <IP address> accom abekas
  ```

This means that the standalone accepts both Accom and Abekas as the device name.

Usage

```
ddr_grab <output> <start> <end> <step> <frame>
<framestep> [-a <DDRname>] [-v]
```

Where `<output>` is the name of the output SOFTIMAGE picture file, without the .pic extension.

`<start>` is the first frame of the output sequence.

`<end>` is the last frame of the output sequence.

`<step>` is the frame step of the output sequence.

`<frame>` is the frame location on the DDR to start grabbing.

`<framestep>` is the step of the frames grabbed from the DDR.

Options

`-a <DDRname>`

Specifies the name of your DDR, if this is other than Accom.

`-v`

Activates verbose mode.

Example

```
ddr_grab d:\grab\image 100 250 3 200 2 -a diskus
```

grabs a sequence of 50 frames with a step of 2 starting at frame 200 on a DDR named diskus. The output is a sequence of SOFTIMAGE picture (.pic) files with a step of 3, saved under the d:\grab directory as:

image.100.pic

image. 103.pic

... to

image.250.pic

Capturing Frames

SOFTIMAGE|XSI supplies a set of standalones for capturing frames from devices using Cindy, Centaur, or Galileo boards with the **civa_capture**, **cva_capture**, or **gv_capture** standalones.

Capturing a Single Frame with a Cindy

The **civa_capture** standalone captures one frame from the device connected to the Cindy and saves it to disk. It is designed for use with the Cindy board.

This standalone runs only on the IRIX operating system.

Usage

```
civa_capture <file> <format>
[-s <pixels>] [-c <phase>] [-t <pixels>] [-z]
[-r <RGB_max> <RGB_min>] [-v <videotype>]
[-C <contrast>]
```

`<file>` is the name you want to give to the file.

`<format>` specifies the format. Use NTSC or PAL for composite analog, and ntsc_d1 or pal_d1 for component digital.

Options

`-s`

Horizontally scales the grabbed image to the size specified in pixels.

`-c`

Adjusts the picture's horizontal phase (default = 35).

`-t`

Trims the right edge of the image by the amount specified in pixels.

`-z`

Disables initialization of the Cindy at startup. Use this option to ensure that the color framing does not change during a transfer session.

`-r`

Specifies the RGB references for contrast and brightness. Lower values for <RGB_max> increase brightness, and higher values for <RGB_min> increase contrast (defaults: <RGB_max> = 63, <RGB_min> = 0).

`-v`

Specifies the video type. Use one of the following video type identifiers (default = betacam).

Video type	Description
betacam	Videocassette BVW-XX
laser	Laser VideoDisc LVR-XXXX
Hi8	Videocassette Hi8 EVO-XXXX
camera	Camera with RGB output

`-C`

Adjusts the picture's contrast (default = 100).

Example

This example grabs the current frame on the device and saves it to a file called **frog.pic.** The format is NTSC and the grabbed image is scaled (**-s**) to be 200 pixels wide.

```
civa_capture frog ntsc -s 200
```

Capturing a Single Frame Using a Centaur

The **cva_capture** standalone captures one frame from the device connected to the Centaur and saves it to disk.

This standalone runs only on the IRIX operating system.

Usage

```
cva_capture <file> <format> [-s <pixels>] [-c <phase>]
[-t <pixels>] [-z] [-r <RGB_max> <RGB_min>] [-v
<videotype>] [-C <contrast>]
```

Where `<file>` is the name for the saved file and `<format>` specifies the format. Use NTSC or PAL for composite analog, and ntsc_d1 or pal_d1 for component digital.

Options

`-s`

Horizontally scales the grabbed image to the size specified in pixels.

`-c`

Adjusts the picture's horizontal phase (default = 35).

`-t`

Trims the right edge of the image by the amount specified in pixels.

`-z`

Disables initialization of the Centaur at startup.

`-r`

Specifies the RGB references for contrast and brightness. Lower values for <RGB_max> increase brightness, and higher values for <RGB_min> increase contrast (defaults: <RGB_max> = 63, <RGB_min> = 0).

`-v`

Specifies the video type. Use one of the following video type identifiers (default = betacam).

Video type	Description
betacam	Videocassette BVW-XX
laser	Laser VideoDisc LVR-XXXX
Hi8	Videocassette Hi8 EVO-XXXX
camera	Camera with RGB output

`-C`

Adjusts the picture's contrast (default = 100).

Example

This example grabs the current frame on the device and saves it to a file called **frog.pic.** The format is NTSC and the grabbed image is scaled (**-s**) to be 200 pixels wide.

```
cva_capture frog ntsc -s 200
```

Capturing a Single Frame Using a Galileo

The **gv_capture** standalone captures one frame from the device connected to the Galileo and saves it to disk.

Usage

```
gv_capture <output file> <format> [-T] [-v
<videotype>] [-d]
```

Where `<output file>` is the name you want to give to the saved picture and `<format>` specifies the format. Use one of the following parameters:

NTSC	PAL	
rgb_525	rgb_625	analog RGB
beta_525	beta_625	betacam YUV
m2_525	m2_625	MII YUV
smpte_525	smpte_625	S MPTE YUV
svhs_525	svhs_625	SVIDEO

NTSC	PAL	
comp_525	comp_625	Composite
d2_525	d2_625	Composite Digital
d1_par_525	d1_par_625	d1 Parallel
d1_ser_525	d1_ser_625	d1 Serial

Options

`-T`

Specifies time code mode instead of frame number mode for the Laser VideoDisc. This option can only be used with the `-v` laser option.

`-v`

Specifies the video type. Use one of the following parameters (default = betacam).

Video type	Description
betacam	Videocassette BVW-XX
laser	Laser VideoDisc LVR-XXXX
Hi8	Videocassette Hi8 EVO-XXXX

`-d`

Activates debug mode.

Example

This example grabs the current frame on the device and saves it to a file called **frog.pic.** The format is SMPTE for NTSC.

```
gv_capture frog smpte_525
```

Moving Frames

SOFTIMAGE|XSI supplies a set of standalones for moving frames to Cindy, Centaur, or Galileo device buffers with the **civa_load**, **cva_load**, or **gv_load** standalones.

Moving a Single Frame Using a Cindy

The **civa_load** standalone moves a frame from disk to the Cindy buffer. It is designed for use with the Cindy board.

This standalone runs only on the IRIX operating system.

Usage

```
civa_load <file> <format>
[-b <red> <green> <blue>] [-x <xoff>] [-y <yoff>]
[-m] [-z] [-f] [-v <videotype>]
```

`<file>` is the name of the file to load.

`<format>` specifies the format. Use NTSC or PAL for composite analog, and ntsc_d1 or pal_d1 for component digital.

Options

`-b`

Sets background color. This option does not use the alpha channel. The background color appears as a border or fill if the image is smaller than the nominal resolution.

`-x`

Offsets the image horizontally by `<xoff>` pixels.

`-y`

Offsets the image vertically by `<yoff>` pixels.

`-m`

Displays image mask.

`-z`

Disables initialization of the Cindy at startup. Use this option to ensure that the color framing does not change during a transfer session.

`-f`

Loads fields from separate files.

`-v`

Specifies the video type. Use one of the following video type identifiers (default = betacam).

Video type	Description
betacam	Videocassette BVW-XX
laser	Laser VideoDisc LVR-XXXX
Hi8	Videocassette Hi8 EVO-XXXX

Example

This example moves the current frame from disk and saves it to the buffer in a file called **snake.pic.** The format is PAL and the image mask (alpha channel **-m**) is displayed.

```
civa_load snake pal -m
```

Moving a Single Frame Using a Centaur

The **cva_load** standalone moves a frame from disk to the Centaur buffer.

This standalone runs only on the IRIX operating system.

Usage

```
cva_load <file> <format> [-b <red> <green> <blue>]
[-x <xoff>] [-y <yoff>] [-m] [-z] [-f] [-v
<videotype>]
```

Where `<file>` is the name of the file to load and `<format>` specifies the format. Use NTSC or PAL for composite analog, and ntsc_d1 or pal_d1 for component digital.

Options

`-b`

Sets background color. This option does not use the alpha channel. The background color appears as a border or fill if the image is smaller than the nominal resolution.

`-x`

Offsets image horizontally by `<xoff>` pixels.

`-y`

Offsets image vertically by `<yoff>` pixels.

`-m`

Displays image mask.

`-z`

Disables initialization of the Centaur at startup.

`-f`

Loads fields from separate files.

`-v`

Specifies the video type. Use one of the following video type identifiers (default = betacam).

Video type	Description
betacam	Videocassette BVW-XX
laser	Laser VideoDisc LVR-XXXX
Hi8	Videocassette Hi8 EVO-XXXX

Example

This example moves the current frame from disk and saves it to the buffer in a file called **snake.pic.** The format is PAL and the image mask (alpha channel **-m**) is displayed.

```
cva_load snake pal -m
```

Moving a Single Frame Using a Galileo

The **gv_load** standalone moves a frame from disk to the Galileo frame buffer in any output format.

Usage

```
gv_load <file> <format> [-d] [-v <videotype>] [-D
<start> <end> <step> [-p]]
```

Where `<file>` is the name of the file to load and `<format>` specifies the format. Use one of the following format identifiers:

NTSC	PAL	
rgb_525	rgb_625	analog RGB
beta_525	beta_625	betacam YUV
m2_525	m2_625	MII YUV
smpte_525	smpte_625	S MPTE YUV
svhs_525	svhs_625	SVIDEO
comp_525	comp_625	Composite
d2_525	d2_625	Composite Digital
d1_par_525	d1_par_625	d1 Parallel
d1_ser_525	d1_ser_625	d1 Serial

In some cases, you may want to use external encoding, particularly if a high quality external hardware encoding device is available. For external encoding, use the **rgb_*** format identifier.

Options

`-d`

Turns debug mode on.

`-v`

Specifies the video type. Use one of the following video type identifiers (default = betacam).

Video type	Description
betacam	Videocassette BVW-XX
laser	Laser VideoDisc LVR-XXXX
Hi8	Videocassette Hi8 EVO-XXXX

`-D`

Displays a sequence of images.

`-p`

Turns pause mode on. Used with the -D option.

Example

This example moves the current frame from disk and saves it to the buffer in a file called **snake.pic.** The format is SMPTE for PAL.

```
gv_load snake smpte_625
```

Index

Numerics

A

B

C

D

E

F

P

R

S

T

U

V

W

X

Y

Z