



Using Tools

A Comprehensive User's Guide

The *SOFTIMAGE 3D User's Guide* was written by Susan Duciaume, Camille Lemberg, Suzanne Gerrior, Grahame Fuller, and Edna Kruger; edited by Edna Kruger; illustrations by Richard Z'Graggen; and layout by Paul Verrall.

Special thanks to Mario Fortin, Lafleche Dumais, and Jeanne Servat for their assistance in assuring the technical integrity of this guide.

© Copyright 1996 Softimage Inc., a wholly owned subsidiary of Microsoft Corporation. All rights reserved.

Microsoft®, MS®, MS-DOS®, Windows®, and Windows NT® are registered trademarks of Microsoft Corporation in the United States and/or other countries.

SOFTIMAGE® is a registered trademark of Softimage Inc., in the United States, Canada, and/or other countries.

3D Studio Max™ is a trademark of Autodesk, Inc.

Adobe Illustrator® is a registered trademark of Adobe Systems, Inc.

Abekas® is a registered trademark of Abekas Video Systems, Inc.

Alias® is a registered trademark of Alias Research Inc.

Apple® and Macintosh® are registered trademarks of Apple Computer, Inc.

Aurora® is a registered trademark of Chyron Corporation.

AutoCAD® is a registered trademark of AutoDesk, Inc.

Flock of Birds® is a registered trademark of Ascension Technology Corp.

Indigo® is a registered trademark and Indy™ is a trademark of Silicon Graphics, Inc.

Magellan™ is a trademark of Logitech, Inc.

Macromedia® is a registered trademark and Freehand™ is a trademark of Macromedia, Inc.

PostScript® is a registered trademark of Adobe Systems, Inc.

SEGA Saturn™ is a trademark of Kabushiki Kaisha Sega Enterprises.

Targa® is a registered trademark of Truevision Inc.

TIFF™ is a trademark of Adobe Systems, Inc.

UNIX® is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company, Ltd.

V-LAN® is a registered trademark of Videomedia, Inc.

Wacom® is a registered trademark of Wacom Co., Ltd.

WSD® is a registered trademark of Accom, Inc.

All other product names mentioned in this guide may be trademarks or registered trademarks of their respective companies and are hereby acknowledged.

This document is protected under copyright law. The contents of this document may not be copied or duplicated in any form, in whole or in part, without the express written permission of Softimage Inc. This document is supplied as a guide for SOFTIMAGE 3D products. Reasonable care has been taken in preparing the information it contains. However, this document may contain omissions, technical inaccuracies, or typographical errors. Softimage Inc. does not accept responsibility of any kind for customers' losses due to the use of this document.

Document No. 90444-0596

Printed in Canada.

Table of Contents

CHAPTER ONE

Introduction to Tools	1
Introduction	3
Using the Command Line	4

CHAPTER TWO

Viewing Images	7
Introduction	9
Viewing Any Image	10
Viewing Picture Images	11
Viewing Line Images.	14
Viewing Colour-mapped Images.	17
Viewing a Flipbook.	18

CHAPTER THREE

Getting File Information	21
Counting Lines and Vectors	23
Displaying Information	24
Comparing Images.	26
Evaluating a Scene	27

CHAPTER FOUR

Printing	29
Creating PostScript Files	31
Setting Up PostScript Files for Printing	33
Printing Files.	34

CHAPTER FIVE

Converting Files	37
Introduction	39
General File Conversion	40
Converting Line Files	44
Converting Files for Importing	46
Converting Files to Export.	54

C H A P T E R S I X

Combining Images	71
Compositing Images	73
Interleaving Fields	79

C H A P T E R S E V E N

Modifying Images	81
Introduction	83
Creating or Modifying an Alpha Channel	84
Extracting a Channel	85
Creating Gradations	86
Creating a High-Contrast Image	87
Changing the Colour Values	88
Resizing, Rotating, and Flipping Images	89
Modifying a Sequence Length	93
Inverting Positive and Negative Images	95
Adding Noise	96
Creating a Depth of Field Blur	97
Creating Special Effects	98

C H A P T E R E I G H T

Using Peripheral Devices	101
Introduction	103
Using an Abekas	104
Using an Accom	106
Recording to a Solitaire	108
Using a Cindy	110
Using a Centaur	111
Using a Galileo	112
Using a VideoFramer	114
Using a VideoLab	116
Using a Lyon-Lamb MiniVAS	118

I N D E X

C H A P T E R O N E

Introduction to Tools

Introduction

During the creative process of modelling, animating, defining the surface, and rendering your masterpieces in SOFTIMAGE 3D, there are other tasks which you will need to accomplish from time to time. The commands that are found in the Tools module help you do these tasks.

These tasks include displaying rendered images or image sequences, and compositing images. They also include exporting and importing images to and from other software formats, grabbing images, and outputting images to PostScript and film or video recorders.

The Tools commands work closely with other command-line utilities called *standalone* programs. These perform the same or similar tasks as the commands in the Tools module, allowing you a choice in the way you prefer to work – from within the SOFTIMAGE 3D interface or from a command line.

Using the Command Line

Opening a Shell

You can open a shell from within SOFTIMAGE 3D by choosing the **Shell** menu cell in the Tools module. You can then type commands as you would outside of SOFTIMAGE 3D.

To exit the shell, type `logout` in it or press Ctrl-d on the keyboard.

Using Standalone (Command Line) Utilities

Standalone utilities allow you to accomplish tasks outside SOFTIMAGE 3D. For example, different standalones may be used to render an image, create motion blur, or convert images to and from various formats.

Standalones can be run directly by typing specific commands in a shell, or they can be run from commands stored in a C-shell file.

For most standalones, the usage is displayed when you type the command name without parameters. For certain standalones that don't normally require parameters, you must type the standalone's name followed by `-h` option (help) to display the usage, such as `flipbook -h`.

For a complete list of the command line options for each standalone, see their description in the *Standalones* HTML file on the *On-line Documentation* CD.

Conventions

These are the typographical conventions used to describe the usage (syntax) of the standalone utilities.

- Chevrons (`< >`) indicate a parameter for which you must substitute a name or value. When entering a parameter, do not type the chevrons. For example, `<filename>` means that a file name must be typed. Unless otherwise specified, do not include the extension (such as `.pic`, `.lin`, `hrc`, etc.) when specifying file names.
- Commands and parameters that are not enclosed in chevrons must be typed exactly as they appear.
- Brackets (`[ ]`) indicate an option along with its input(s). Do not type the brackets. For example, `[-z <fact>]` means that the `-z` option (zoom) is available and that if you use it, a `<fact>` (zooming factor) must be specified.

- A dash (-) followed by a letter or letters (as in -z) indicates an option name. For example, -z could mean zoom, -s could mean sequence, -v could specify verbose mode, etc. The available options are listed and explained in the description of each standalone.
- A vertical bar (|) separates exclusive options. For example, the usage for **effect** includes [-r { -90 | 90 | 180 | 270 }]. This means that for the rotation option (-r), one of the four possibilities should be chosen. When specifying an option, do not type the vertical bar.
- Braces ({ }) enclose a group. In the previous example, the four possible rotation values are related to the rotation option. When entering parameters, do not type the braces.
- Three dots (. . .) indicate that the option or group of options can be repeated. For example, the usage for the **composite** standalone includes

```
{ <layer> [<start frame>] [ -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>] ... }.
```

This means that everything in the group contained between the brackets [] 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 in a C-shell file.

C H A P T E R T W O

Viewing Images

Introduction

There are a number of ways to view your image after it is rendered. You can use commands in the Tools module or standalone utilities:

- View any image with the **imgshow** standalone.
- View **.pic** image files with the **Picture** menu cell, or the **display**, **sequence**, and **showpic** standalone utilities.
- View **.lin** files with the **Line** menu cell commands or **showlin** standalone.
- View a colour-mapped image (**.icn**) with the **showicn** standalone.
- View an animation sequence with the **Flipbook** menu cell or the **flipbook** standalone.

Viewing Any Image

Using the `imgshow` Standalone

The `imgshow` standalone can display any image format for which there is a dynamic shared object (DSO) installed in the `SI3D_installdir/3D/dso/sil` directory.

To run `imgshow`, you must set the environment variable `SI_IMAGE_PATH` to point to `SI3D_installdir/3D/dso/sil`.

If `SI_IMAGE_PATH` is not set or empty, `imgshow` looks in `/usr/softimage/3D/dso/sil`.

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.1 200 250
```

`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 would need to specify it. For example, you would need to type:

```
imgshow toto.1.pic
```

to see the file on disk called `toto.1.pic`.

The supported image file formats depend on which DSOs are currently installed in `SI3D_installdir/3D/dso/sil`. The `imgconv` usage message lists the supported file formats.

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Viewing Picture Images

There are a number of ways to view a **.pic** (bitmapped image) file: using the **Picture** menu cell in the Tools module, or using the **display**, **sequence**, and **showpic** standalone utilities.

Using the Picture Menu Cell

The **Picture** menu cell allows you to view a rendered image or a sequence of rendered images, or allows you to get information about the image or image sequence. This shows your sequence once, unlike the flipbook which lets you play your scene back and forth as you like.

1. Choose the **Picture** menu cell in the Tools module. The View Pictures dialogue box appears.
2. In the scroll box, go to the **RENDER_PICTURES** directory and select the file you want to display. If you want to show a sequence, select one file from the sequence (such as **walkies.21**) and all images in that sequence are displayed.

Select the **Use Icons** option if you prefer to see pictures listed by icon rather than by file name in the scroll box.

3. Select one of the five options on the right side of the dialogue box, depending on what you want to view. For example, to see a sequence, select **Show sequence**, or select **Show picture** to show a single image.
4. Click **Display** to display the file that you selected. A window showing that image appears. Middle-click to close that window.
5. Click **EXIT** to close the View Pictures dialogue box.

Using the display Standalone

The **display** standalone displays SOFTIMAGE 3D 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.

For very large images, you can use the **showpic** standalone, which loads only the visible portion of the image. **showpic** is slower, but uses less memory.

This example displays the **flash.pic** file from frames 1 to 50 at a step of 1 (**-s 1 50 1**) and centres 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
```

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.

Displaying Stereo Images

To display stereo images, use the **-t** option. You will require StereoGraphics CrystalEyes Stereoscopic System to use this option.

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.

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Using the showpic Standalone

The **showpic** standalone displays SOFTIMAGE 3D picture files (.pic) and sequences on the screen, including field-rendered pictures. The **showpic** standalone 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.

To fit the entire image on to the screen, you may want to use the **reduce** standalone (see *Reducing an Image* on page 91) before running **showpic**.

You can view the red, green, blue, or alpha channels separately by pressing the r, g, b, or a keys on the keyboard.

This example shows the **bowl.pic** file from frames 20 to 50 at a step of 1 (**-s 20 50 1**).

```
showpic bowl -s 20 50 1
```

To view the odd-field dominance image, for example, in a field-rendered sequence, use this syntax where **output.1** is the image file name:

```
showpic output.1 -0
```

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, middle-click.

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Using the sequence Standalone

The **sequence** standalone displays a sequence of frames, as well as allowing you to execute scripts. If there are C-shell files with specific names in the directory from which the **sequence** standalone is issued, these scripts are executed at the appropriate time. These names are:

- **INIT_SEQUENCE** is run before the first frame is displayed.
- **PROCESS_FRAME** is run while each frame is being displayed.
- **RELEASE_SEQUENCE** is run once the whole sequence has been displayed.

This example shows only the alpha channel (**-a**) of the sequence of frames called **picture2** from frames 5 to 10 at a frame rate of 1.

```
sequence picture2 5 10 1 -a
```

To stop displaying a sequence, press all three mouse buttons at once.

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Viewing Line Images

There are a number of ways to view a **.lin** image (vector-based) file: using two of the **Line** menu cell commands in the Tools module, or the **showlin** standalone. A line image file is the result of a scene that has been rendered using the hidden line or wireframe renderer in SOFTIMAGE 3D.

Viewing a Single Image

The **Line → Show File** command displays a single **.lin** format image (wireframe or hidden-line rendered). It also lets you convert a file from **.lin** to **.pic** format.

1. Choose the **Line → Show File** command in the Tools module. The standard database browser appears.
2. Select a **.lin** file from the browser (usually the **RENDER_PICTURES** chapter). The Show Line dialogue box appears.
3. Set the image resolution in pixels by setting the **Width** option. The aspect ratio is defined by the width, so the height is automatically set accordingly.
4. To convert and save this image as a **.pic** file, select the **Output to a File** option and specify the name in the **Output File Name** text box.
5. If you like, select the **Antialiasing** option to reduce jagged edges that are often visible on oblique lines.
6. You can also define the **Background** and **Foreground** (lines) **Colours** by entering a value ranging from 0 to 1 for each Red, Green, and Blue component.
7. Click **Ok** to display the image.

Viewing a Sequence with Line → Show Sequence

The **Line → Show Sequence** command displays a sequence of .lin image files (wireframe or hidden-line rendered).

If you want to render some 3D elements to make them look like traditional 2D cel animation pieces, you can convert line files (.lin) into picture files (.pic) using this command. You can also use the **showlin** standalone, which does the same thing. The only difference with using **showlin** is that the line file is always displayed on the screen.

1. Choose the **Line → Show Sequence** command in the Tools module. The database browser is displayed.
2. Select the .lin file sequence by clicking any sequence file name and click Ok. The Show Line Sequence dialogue box is displayed.
3. Specify the **Start Frame**, **End Frame**, and **Step** (increment) values for the sequence you want to show.
4. Set the image resolution in pixels by setting the **Width** option. The aspect ratio is defined by the width, so the height is automatically set accordingly.
5. If you want to set the playback speed, select the **Buffered** option and the **Number of frames per second**. The **Buffered** option loads frames in RAM. If you don't select the **Buffered** option, the **Number of frames per second** option has no effect.
6. To display the sequence in a loop, select **Loop in Sequence**. When using this option, the left mouse button plays the sequence backward, the right mouse button plays the sequence forward, and the middle mouse button allows you to stop. If you hold down the middle mouse button while dragging left or right, you can play the sequence forward or backward at a speed relative to that at which you drag the mouse.

If the **Buffered** option is *not* selected, the animation plays from disk continuously, and the mouse buttons have no effect on the playback.

7. To convert and save this image as a .pic file, select the **Output to a File** option and specify the name in the **Output File Name** text box.
8. To save the alpha channel to a .pic file, select the **Alpha Channel** option and set the **Alpha Value**, which is the intensity of the lines in the alpha channel.

9. If you like, select the **Antialiasing** option to reduce jagged edges that are often visible on oblique lines.
10. You can also define the **Background** and **Foreground** (lines) **Colours** by entering a value ranging from 0 to 1 for each Red, Green, and Blue component.
11. Click Ok to display the image.

Viewing a Sequence with the **showlin** Standalone

The **showlin** standalone program displays line files (.lin) on the screen. You can also use it to convert line files into picture files (.pic).

To convert line files into picture files, you can also use the **lin2pic** standalone, which has the advantage of not displaying images on the screen while converting them. The **Line → Show Sequence** command also lets you do this.

This example shows the **giraffe.lin** file from frames 1 to 20 with a step of 2 (**-s 1 20 2**) and converts it to a .pic file called **graff** (**-o graff**) with a display window width of 600 pixels (**-w 600**).

```
showlin giraffe -o graff -s 1 20 2 -w 600
```

If only one frame is displayed, click the left mouse button to quit.

Here's another example where the **spherehidden_line** file has a background colour (**-b**) of 0.8 for the RGB values, the foreground colour (**-f**) has 0 for the RGB values (black), and the sequence of frames 1 to 100 loops (**-l**):

```
showlin spherehidden_line -b 0.8 0.8 0.8  
-f 0 0 0 -l -s 1 100 1
```

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Viewing Colour-mapped Images

The **showicn** standalone lets you view a colour-mapped image file (.icn). These .icn files are usually used as icons for SOFTIMAGE 3D files.

This example shows the **grail.icn** file from frames 1 to 10 with a step of 1 (**-s 1 10 1**) and uses the **grail.clm** colour map file (**-c grail**).

```
showicn grail -c grail -s 1 10 1
```



If you specify the colour map explicitly by using either the **-c** or **-C** option, the same colour map is used for the whole sequence.

When you run **showicn** with the **-s** option to show a sequence, you can use the mouse and keyboard to control the display.

Mouse Functions

- Left button starts flipping forward.
- Middle button stops flipping.
- Right button starts flipping backward.

Keyboard Functions

- F key moves forward one frame.
- B key moves backward one frame.
- L key toggles loop mode on and off.
- Esc quits **showicn**.

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Viewing a Flipbook

The **FlipBook** menu cell and the **flipbook** standalone let you display an animation sequence on the screen, similar to a flipbook used in traditional animation. Unlike simply viewing an image sequence, the flipbook lets you play the sequence back and forth as you like using the time line in the SOFTIMAGE 3D display.

Using the FlipBook Menu Cell

1. Choose the **FlipBook** menu cell in the Tools module. The FlipBook dialogue box appears.
2. Go to the directory in which the rendered image sequence is, which is usually the **RENDER_PICTURES** chapter. Click the desired sequence file name and frame number in the scroll box.
3. The sequence name appears in the **Sequence Name** text box. You can also select **Use icons** to display icons instead of file names in the scroll box for a better visual reference.
4. If you want to view a field-rendered image, select the field type from the menu:



Full frame
Field even-dominance
Field odd-dominance
EXIT POPUP

5. Set the starting frame, the last frame, and the frame step (**Do frames**).
6. Define the display speed with the **Number of frames per second**.
7. Define the **Horizontal** or **Vertical** **zooming factor** if needed.
8. Click **Ok**. SOFTIMAGE 3D reads the images as shown in the status bar and a window appears.
9. Use the playback box or the time line at the bottom of the display to play the sequence forward or backward.

Note

When the flipbook is in a paused state, you can use the hardware zoom on the flipbook by pressing the number keys (1 to 9) on the keyboard. For example, pressing 2 on the keyboard enlarges your flipbook image to twice the size of the default image size.

Using the flipbook Standalone

The **flipbook** standalone program 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 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, you would type:

```
flipbook output -E
```

where **output** is your rendered image.

Note

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.

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Once the first frame is displayed, right-click to begin flipping. The sequence loops when it reaches the last frame.

To flip backward, left-click. To pause, middle-click; to flip forward again, right-click.

To adjust the frame rate interactively, hold down the middle mouse button and drag the mouse. 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, right-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 key or right arrow move forward one frame.
- b key or left arrow move backward one frame.
- u key or up arrow increase the frame rate.

- d key or down arrow decrease the frame rate.
- l key toggles loop mode on and off.
- i key gets information about the sequence.
- 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):

$4 \times (\text{height of frame}) \times (\text{width of frame}) \times (\text{number of frames})$

For example, to calculate the memory required for a 256 x 256 image, the equation $256 \times 256 \times 4$ 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.

The C-shell “maxframes” is in **SI3D_install/3D/tools/scripts**. This calculates the number of frames that can be displayed based on the number of frames and the particular system’s memory. This C-shell accesses the program **free**, which is also found in **/scripts**.

C H A P T E R T H R E E

Getting File Information

Counting Lines and Vectors

Counting the Number of Lines in a Picture Image

The **howmany** standalone counts the actual number of lines in a SOFTIMAGE 3D picture file (.pic).

Normally, the number should match the height of the picture written in the file header. This indication makes a comparison between the real image height (number of lines) and the first number displayed when **howmany** is used. However, if the program creating the file was stopped during execution, the second number (the << ? remains to be done >> part) will be higher than zero.

The **howmany** standalone is useful for checking how many lines have already been written in a picture file that is currently being created.

For example, to see how many lines are in the **dog.pic** file, type:

```
howmany dog
```

or

```
howmany dog.pic
```

It is also useful when a render batch has been executed on many CPUs of machines and you want to know if all images are complete.

Counting the Number of Vectors in a Line Image

The **infolin** standalone program counts how many vectors and groups of connected vectors are in a line image file (.lin). A line file is the result of a scene that has been rendered using the hidden line or wireframe renderer in SOFTIMAGE 3D.

For example, to see how many vectors are in the **cat.lin** file, type:

```
infolin cat
```

or

```
infolin cat.lin
```

Displaying Information

Displaying Information about Any Image Format

The **imginfo** standalone can display information about any image format for which there is a dynamic shared object (DSO) installed in **SI3D_installdir/3D/dso/sil**. For example, to display information on the **portrait** image file, type:

```
imginfo portrait
```

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 would need to specify it.

The supported image file formats depend on which DSOs are currently installed in **SI3D_installdir/3D/dso/sil**. When you type **imgconv**, the usage message lists the supported file formats.

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Displaying Information about Picture Images

The **infopic** standalone displays information about SOFTIMAGE 3D picture files. For each subregion, the following information is displayed:

- Width
- Height
- Aspect ratio
- Field type (FULL_FRAME, EVEN_FIELDS, or ODD_FIELDS)
- The number of bits, data type, and compression method for the RGB channels
- The 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.

This example shows information on the **walkies1**, **walkies2**, and **walkies3** picture files.

```
infopic walkies1 walkies2 walkies3
```

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Displaying Information about Line Images

The **Line → Information** command displays information about a **.lin** image (wireframe or hidden-line rendered).

1. Choose the **Line → Information** command in the Tools module. The standard database browser appears.
2. Select the image file name, probably in the **RENDER_PICTURES** chapter, for which you need information by double-clicking it.
3. The Info Line message box appears containing information on the file name, comments that have been made, version number, picture ratio, number of vectors in the image, and data type (such as floating bits, monochrome, etc.).
4. Click Exit to close this box.

Comparing Images

The **diffpic** standalone displays an image whose pixels are highlighted when the corresponding pixels of the two input pictures are different.

If you use the **-i** option, the intensity of each displayed image pixel is equal to the difference of the intensities of the corresponding pixels of the two input pictures.

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
```

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Evaluating a Scene

The `evalScene` standalone evaluates a SOFTIMAGE 3D scene at the given frame or frames. The result of this evaluation is saved back as a complete scene in the desired database, or in an EVALSCENE database. The default database is the one defined in the `DatabaseDir.rsrc` file in your home directory. The resulting scene contains the state of all the objects (models, lights, camera) at the given time. Their file names have a substitution number corresponding to the requested frame.

This standalone is useful for resetting and freezing the animation elements of a SOFTIMAGE 3D scene before converting the scene to DXF or IGES format using the `soft2dxf` or `soft2iges` standalones.

This example evaluates the latest scene file (-l) called **planets** from frames 100 to 200 with a step of 2 (-s 100 200 2) and only triangulated polygon meshes are produced for output objects (-m). The output file is saved in the database called EVALSCENE (-o).

```
evalScene planets -l -o EVALSCENE -s 100 200 2 -m
```

The scene file can be in any format. The output scene is always in the new format.

Note

If you don't use the -f, -s, or -t options, the entire sequence is evaluated.

Object File Format

The object file contains a list of objects to dump, and is used with the -F option. Each line should start with a chapter name, followed by either the object name or an asterisk * (for all objects in the chapter). Names and version numbers may omit the prefix.

For example:

```
LIGHTS light
LIGHTS aaa-light2.5-2
MODELS *
CAMERAS camera
```

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

C H A P T E R F O U R

Printing

Creating PostScript Files

The **Output → PostScript** command creates a PostScript® file which can be printed using a laser printer, according to the parameters previously chosen with the **Output → Setup** command.

1. Select a window in SOFTIMAGE 3D by clicking the letter identifier box (A, B, C, or D) in the upper-left corner of the desired window. The window is highlighted in red.
2. Choose the **Output → PostScript** command in the Tools module. The Saving PostScript File dialogue box appears.

You can select the **Use icons** option to display your files as icons instead of file names in the scroll box.

3. Select an existing PostScript file from the list in the scroll box to overwrite it, or specify a new file name for the PostScript file in the **PostScript File** text box.
4. Click **Ok** to create a file or **Cancel** to exit.

Creating a PostScript File for Each Frame

The **Output → PostScript Playback** commands let you create a PostScript file for each frame in a sequence as you play it.

1. Select any window in SOFTIMAGE 3D by clicking the letter identifier box (A, B, C, or D) in the upper-left corner of the desired window. The window is highlighted in red.
2. Choose the **Output → PostScript Playback → On** command in the Tools module.
3. In the browser that appears, select the path and specify a file name for the PostScript files you want to be created. Close the browser.
4. Move the time line pointer at the appropriate frame, then click the forward arrow in the playback box to play your sequence (make sure that the **L** (loop) button in the playback box is not selected).

The whole sequence is played and stops when it reaches the end. If you interrupt the playback process, files are created only up to that frame.

5. Choose the **Output → PostScript Playback → Off** command.

You now have a PostScript file for each frame of your sequence. For example, if you named the files **outpost**, the files would be **outpost.1**, **outpost.2**, **outpost.3**, and on up to the last frame of the sequence.

Creating PostScript Files with Filled Faces

The **Output → PostScript Fill-in** command creates a PostScript file (.ps) in which faces are filled. This works on parallel projection windows only (Top, Front, Right, and Ortho). Make sure to use the **Output → Setup** command to control how a PostScript image is printed on paper.

1. Select a parallel projection window in SOFTIMAGE 3D by clicking the letter identifier box (A, B, C, or D) in the upper-left corner of the desired window. The window is highlighted in red.
2. Choose the **Output → PostScript Fill-in** command in the Tools module. The Saving PostScript File dialogue box appears.

You can select the **Use icons** option to display your files as icons instead of file names in the scroll box.

3. Select an existing PostScript file from the list in the scroll box to overwrite it, or specify a new file name for the PostScript file in the **Postscript File** text box.
4. Click **Ok**.

Setting Up PostScript Files for Printing

The **Output → Setup** command controls how a PostScript image is printed on paper.

1. Choose the **Output → Setup** command in the Tools module. The PostScript Setup dialogue box appears.
2. Select the **Viewport** options (Left, Right, Bottom, and Top) by clicking the appropriate boxes and typing in the desired values. This defines the size and location of the printed image on paper.
3. Specify the **# of copies** to be printed and if you want to print the name of the file on the PostScript image (**Print name**).
4. If you like, change the orientation of the paper from portrait to landscape by selecting the **Rotate 90** option.
5. Merge a second file with your original image by selecting the **Include** option and typing the file name in the text box. This is often used to include the file which defines the registration marks that are printed on the image.
6. Click Ok to confirm the settings and exit the dialogue box.

Printing Files

Printing Line Files on a PostScript Printer

The **Print** menu cell prints **.lin** format images (rendered in wireframe or hidden-line) on a PostScript printer.

1. Choose the **Print** menu cell in the Tools module. The Print line file on PostScript printer dialogue box is displayed.
2. Specify the name of the **.lin** image file to print in the **NewName** text box, or click the **Select** button to open a browser in which you can search for the file.
3. If the file is a sequence, specify the **Do Frames** values (from, **To**, **Step**, and **Rep**).
4. Select the **Line width** in pixels if you want to change it from the default.
5. Select the DPI (dots per inch) units for the **Output size**.
6. You can choose to **Offset** the printed image, add **Effects** (rotating, flipping, and rounded or pointed lines), or add information (**Print with**) to it such as a border, frame number, or sequence name.
7. Select the **Paper Format** as either Letter or Legal size.
8. Select the **Printer Name** from the list of supported printers for your machine. If your printer is on the network, select its name.
9. Specify the **Registration** file name to be used for registration marks, if you like.
10. Click **Print** to start printing.

Using the printlin Standalone

The **printlin** standalone prints line files (**.lin**) on a PostScript printer. The **SI_PRINTER** environment variable defines the default printer name. See the *SOFTIMAGE 3D Installation Guide* for more information about environment variables. For example, by adding this line to your **.softimage** file, you define the “ps100” printer as the default printer:

```
setenv SI_PRINTER ps100
```

This example prints the **jojo.lin** file sequence from frames 20 to 30

with a step of 1 (`-s 20 30 1`). It is rotated 90 degrees (`-R 90`) and printed at a width of 500 pixels (`-w 500`) on the ps1000 printer (`-d ps1000`).

```
printlin jojo -s 20 30 1 -R 90 -w 500 -d ps1000
```

Note The height of the image is computed automatically according to the width and the size of its bounding rectangle.

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Printing PostScript Files on a Laser Printer

The **printps** standalone prints a PostScript file on a laser printer.

This example shows the **flower** PostScript file printed from frames 1 to 100 with a step of 2 (`-s 1 100 2`) and the printer name of **laser1** (`-d`).

```
printps flower -s 1 100 2 -d laser1
```

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Printing to Network Printers

You can print **.lin** and **.ps** files to network printers. The **printlin** and **printps** standalones each have a `-d` option that specifies the name of the destination printer.

You can also use the **Print** menu cell in the Tools module to print a **.lin** file to a network printer. Specify the name of a network printer in the text box in the Print dialogue box.

For example, to print a file called **image3** to the **net1** printer, type:

```
printlin image3 -d net1
```

or

```
printps image3 -d net1
```

Choose a default printer by setting the **SI_PRINTER** environment variable. When you omit the `-d` option, **printlin** and **printps** print to **\$SI_PRINTER**. If **SI_PRINTER** is not defined, the standalones use the default system printer.

C H A P T E R F I V E

Converting Files

Introduction

There are a number of formats supported by SOFTIMAGE 3D to let you import and export files and objects, such as Alias®, Wavefront, SGI, Targa®, TIFF™, EPS, 3D Studio™, etc. You can also convert between SOFTIMAGE 3D images file types, such as **.pic** and **.lin**.

Although not described here, the translators for DXF (**dx2soft** and **soft2dxf**) and IGES (**iges2soft** and **soft2iges**) are described fully in the *DXF* and *IGES* HTML files on the *On-line Documentation* CD.

General File Conversion

Converting between Many File Formats

Using the **imgconv** Standalone

The **imgconv** standalone converts files between a wide variety of image file formats. It also provides a set of image processing tools such as the **blur**, **contrast**, and **zoom** standalones.

Each image converter and image processing tool is implemented as a dynamic shared object (DSO). To run **imgconv**, you must set the environment variable **SI_IMAGE_PATH** to point to the directory containing these DSOs (the DSOs are shipped in **SI3D_installdir/3D/dso/sil**).

If **SI_IMAGE_PATH** is not set or empty, **imgconv** looks in **/usr/softimage/3D/dso/sil**.

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 either use an extension (such as **.tif**) or the **-format** option.

The supported image file formats and image processors depend on which DSOs are currently installed in **SI3D_installdir/3D/dso/sil**. The **imgconv** usage message lists the supported file formats and image processors.

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Using the **Watch** Standalone

The **Watch** effect is a standalone used to display, convert, compress, and composite pictures or sequences of different formats: SOFTIMAGE 3D, Targa, SGI, and Composite YUV from different sources like the SOFTIMAGE 3D renderer or video tape.

When dealing with sequences of images, Watch can accept three methods of frame numbering:

Filename.12.ext

Filename.0012.ext

Filename0012.ext (DOS format)

1. Create a picture or sequence in any of the supported formats.

2. Open a shell and type:

```
Watch Image_file <options>
```

where *Image_file* is the name of the input picture or sequence and <options> are the parameters.

3. A display window appears. Use the r, g, b, and a keys on your keyboard to display the red, green, blue, and alpha channels respectively. Pressing the space bar restores the view to normal.

Watch automatically detects the file format of the picture (input and output) according to the file extension (.pic for SOFTIMAGE 3D, .sgi for SGI, .tga for Targa, .yuv for Composite YUV).

Watch can detect compressed files with the IRIX **compress** or **pack** commands. These commands must be available in your path definition. To check if they are available, type the following in a shell:

```
which compress
```

```
or
```

```
which pack
```

When dealing with compressed images, Watch decompresses (or unpacks) the picture before converting it.

Examples

Here are some examples of how to use the Watch effect:

- This example reads the picture file named **picture.1.pic** and writes the output picture in .yuv format.

```
Watch picture.1.pic -o out.0010.yuv
```

- This example reads the sequence of pictures from **pict0020.tga** to **pict0030.tga** and converts them into a sequence of pictures in .sgi format, from **outpic.20.sgi** to **outpic.30.sgi**. In this example, Watch converts without showing the pictures because the -d option is used.

```
Watch pict0100.tga -o outpic.1.sgi -s 20 30 1 -d
```

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Converting from Various SOFTIMAGE 3D Versions

The **hrcConvert** standalone converts hierarchy model files (**.hrc**) from various versions of SOFTIMAGE 3D.

This example converts the **newver** file made with version 3.5 to **oldver**, which becomes a SOFTIMAGE 3D version 3.0 file.

```
hrcConvert newver oldver -v3.0
```

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Converting between 3D Studio and Picture Files

The **convpic** standalone is designed to convert between the different image file types supported by 3D Studio and the SOFTIMAGE 3D **.pic** files. Although the product is primarily designed to support the SOFTIMAGE 3D to 3D Studio Object Converter, the image file converter can also be used as a standalone.

This example converts the **dino** file to **dino2** and keeps the existing file (**-k**) if that file already exists (that is, it doesn't write a new **dino2** file if one already exists).

```
convpic dino dino2 -k
```

If the output file name is not given, the input file name and path is used, and the appropriate extension is automatically generated (see below). If no file extension is specified, none is generated.

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Converting between Binary and ASCII

The **bin2ascii** standalone reads a SOFTIMAGE 3D data file and converts it from binary to ASCII, or vice-versa.

The supported types are material, 2D or 3D texture, lattices, lights, cameras, animation, and structure keys. Data types are recognized from the input file header. To convert a hierarchy file (**.hrc**), use **hrcConvert**.

This example converts the binary file **data.cam** (camera) to the ASCII file called **plain** using the **-k** option, which doesn't create an output file if a file called **plain** already exists.

```
bin2ascii data.cam plain -k
```

Note

Use extensions when specifying the input file name: **.mtr**, **.t2d**, **.t3d**, **.lat**, **.cam**, **.lig**, **.ani**, or **.key**. Extensions are appended automatically to the binary output file name.

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Converting Colour to Grey Scale

The **gray_scale** standalone converts a colour image or sequence of images into a grey-scale image or sequence of images.

Two averaging methods are available to convert the colours into grey scale. The default method uses the NTSC's averaging formula:

$$\text{grey scale value} = (0.299 \times \text{red}) + (0.587 \times \text{green}) + (0.144 \times \text{blue})$$

The second method simply computes the normal average of the three colours (**-n** option).

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
```

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Converting Line Files

Converting a Line File to a Picture File

To convert line files into picture files, you can use either the **lin2pic** or **showlin** standalones. The **lin2pic** standalone has the advantage of not using the screen to convert the files.

For a complete list of the command line options for these standalones, see their descriptions in the *Standalones* HTML file on the *On-line Documentation* CD.

Using the lin2pic Standalone

The **lin2pic** standalone converts line files (**.lin**) to SOFTIMAGE 3D picture files (**.pic**).

This example converts the **squiggly.lin** file to the **full.pic** file from frames 1 to 85 with a step of 1 (**-s 1 85 1**) and with an image width of 250 pixels and has antialiasing mode on (**-a**).

```
lin2pic squiggly full 250 -a -s 1 85 1
```

The output picture height is computed automatically according to the width and the bounding rectangle of the input image.

Using the showlin Standalone

The **showlin** standalone is used to convert line files into picture files (**.pic**). You can also use it to simply display line files (**.lin**) on the screen.

This example converts the **straight.lin** file to the **full.pic** file (**-o full**) from frames 1 to 90 with a step of 2 (**-s 1 90 2**) as it is displayed on the screen, looping the sequence (**-l**).

```
showlin straight -l -o full -s 1 90 2
```

If only one frame is displayed, click the left mouse button to quit.

Converting a Line File to Encapsulated PostScript

The **lin2eps** standalone converts a line file (**.lin**) into an encapsulated PostScript file (**.eps**) that can be included in a PostScript document.

The following example creates an encapsulated PostScript file that can be sent to a Macintosh® computer. The file **test.1** is the original line file, and **test.3** is the **.eps** file that is created.

```
lin2eps test.1 test.3 -l 0.25 -m -W 10 -p
```

- The **-l** option sets the line width to 0.25 points, which is about the thinnest line that a 300 DPI laser printer can print.
- The **-m** option adds a MacBinary header to the file.
- The **-W** option sets the absolute width of the image to 10 centimeters.
- The **-p** option adds an embedded PICT preview of the image, which can be useful when using the picture in a word-processing or page layout program.

Notes

- When sending **.eps** files to a Macintosh, they should be transferred as binary files (not ASCII files) with the MacBinary header option.
- Make sure to use an upper-case **V** for the verbose option, since the lower-case **v** is used for the vertical flip option.
- The height of the image is computed automatically according to its width and the size of its bounding rectangle.

Converting Line Files to HPGL

The **lin2hppl** standalone converts SOFTIMAGE 3D line files (**.lin**) to HPGL format. The HPGL pictures are given the **.PLT** extension.

This example converts the **jar.lin** file to **jar2.plt** from frames 1 to 250 with a step of 2.

```
lin2hppl jar jar2 -s 1 250 2
```

Converting Files for Importing

You can convert files for importing into SOFTIMAGE 3D using two main ways: the **Import → Images** commands in the Tools module, or the conversion standalones for specific file types. To import objects, use the **Import → Object** commands.

For a complete list of the command line options for these standalones, see their descriptions in the *Standalones* HTML file on the *On-line Documentation* CD.

Importing Images

The **Import → Images** commands convert images created with another software into SOFTIMAGE 3D format. The submenu displays the formats from which you can choose.

The converted SOFTIMAGE 3D picture file keeps the same name as the imported file except with a **.pic** suffix.

1. Choose the **Import → Images** command in the Tools module. Then choose the subcommand (file format) of your choice. The standard database browser appears.
2. Select a file name by double-clicking it. When the file is converted, it keeps this file name except with the **.pic** file suffix to indicate that it is now a SOFTIMAGE 3D picture file.
3. When you choose a file to import, a message box appears asking you to enter information which will be stored in the SOFTIMAGE 3D picture file. Depending on the format, you will need to provide this information.
 - For RGB and RGBA images, supply the width and height of the pictures.
 - For TIFF images, choose a file suffix. The MacBinary header can be removed if there is one already attached to the file.
 - For Vertigo images, the Protect option enables or disables file name protection, and the Wind option lets you convert only the coded portion of the Vertigo image.
 - For Wavefront images, the **<keep border>** option is used to preserve the original width and height of the Wavefront picture when converting it to a SOFTIMAGE 3D file. Note that SOFTIMAGE 3D can read the format of **.rlb** files, but only recognizes the **.rla** suffix in the browser. As a workaround, you can

simply give your **.rlb** files the **.rla** suffix before you import them, or you can use the **wave2soft** standalone (see page 52), which recognizes both **.rla** and **.rlb** suffixes, to convert the Wavefront files.

4. After filling out the appropriate information, click **Ok** to proceed or **Cancel** to exit the procedure.

3DS to SOFTIMAGE 3D

The **Get → 3DS2Soft** command in the **Model** module lets you import 3D Studio model files into SOFTIMAGE 3D. The result is a hierarchy of SOFTIMAGE 3D polygon meshes with any local materials correctly assigned.

1. Choose the **Get → 3DS2Soft** command in the **Model** module.

The 3DS2Soft dialogue box is displayed.

2. Specify the **.3ds** file that you want to convert by typing its name in the **Input File** text box. The **.3ds** extension may be omitted, but the file name itself must end in **3ds**.

Click **Browser** to open the database browser to find the files that qualify as **.3ds** model files.

3. You can **Scale**, **Rotate**, or **Translate** the models as you import them into SOFTIMAGE 3D, or you can freeze the resulting transformation.
4. Click **Ok** and the model is imported into SOFTIMAGE 3D.

If the **.3ds** file contains multiple models, they are placed in a SOFTIMAGE 3D hierarchy. You may want to rearrange the hierarchy after the models are imported.

Alias to SOFTIMAGE 3D

The **alias2soft** standalone converts Alias picture files (**.als**) into SOFTIMAGE 3D picture files.

For example, to convert an Alias file called **alien** to a SOFTIMAGE 3D file called **alien1** from frames 1 to 100 with a step of 1, type:

```
alias2soft alien alien1 -s 1 100 1
```

Although there is no default extension for Alias picture files, **alias2soft** assumes an extension of **.als**.

Artisan to SOFTIMAGE 3D

The **art2soft** standalone converts Artisan picture files (**.art**) into SOFTIMAGE 3D picture files (**.pic**).

For example, to convert an Artisan file called **picasso** to a SOFTIMAGE 3D file called **picasso1** from frames 1 to 100 with a step of 1, type:

```
art2soft picasso picasso1 -s 1 100 1
```

Aurora to SOFTIMAGE 3D

The **aurora2soft** standalone converts Aurora® picture files (**.sim**) into SOFTIMAGE 3D picture files.

For example, to convert an Aurora file called **borealis** to a SOFTIMAGE 3D file called **borealis1** from frames 1 to 100 with a step of 1, type:

```
aurora2soft borealis borealis1 -s 1 100 1
```

EPS to SOFTIMAGE 3D

The **Get ➔ Eps2Soft** command in the Model module allows you to convert Illustrator® and Freehand™ files into curves in SOFTIMAGE 3D. This gives you access to a wide variety of fonts and design tools that can be easily imported and translated into a hierarchy of Bezier curves.



If the **.eps** file contains text, it should be saved in outline form from Illustrator or Freehand.

1. Choose the **Get ➔ Eps2Soft** command in the Model module. The Eps2Soft dialogue box is displayed.
2. Specify the software product name (Illustrator or Freehand) used to create the file you want to convert.
3. Enter the name of the file you want to convert.



This file should be placed in a directory whose full path name is easy to remember.

4. Set other parameters as desired (such as **Scale** factor and **Offset**).
5. Click **Ok** to accept the dialogue box settings.

The **.eps** file is converted to a hierarchy of Bezier curves.

Everest to SOFTIMAGE 3D

The **everest2soft** standalone converts an Everest picture file (.ima) into a SOFTIMAGE 3D picture file (.pic).

For example, to convert an Everest file called **mountain** to a SOFTIMAGE 3D file called **mountain1** from frames 1 to 100 with a step of 1, type:

```
everest2soft mountain mountain1 -s 1 100 1
```

Macintosh PICT to SOFTIMAGE 3D

The **pict2soft** standalone converts a Macintosh PICT version 2 file into a SOFTIMAGE 3D picture file (.pic).

For example, to convert a PICT file called **apple** to a SOFTIMAGE 3D file called **apple1** from frames 1 to 100 with a step of 1, type:

```
pict2soft apple apple1 -s 1 100 1
```

Matisse to SOFTIMAGE 3D

The **r2soft** standalone converts Matisse files (.r) into SOFTIMAGE 3D picture files (.pic).

Note

Only the 24-bit Matisse .r format is supported: the 8-bit format using colour lookup tables is not.

For example, to convert a Matisse file called **dance** to a SOFTIMAGE 3D file called **dance1** from frames 1 to 100 with a step of 1, type:

```
r2soft dance dance1 -s 1 100 1
```

PictureMaker to SOFTIMAGE 3D

The **pm2soft** standalone converts PictureMaker images (.r8, .g8, .b8, .a8) to SOFTIMAGE 3D format (.pic) with an alpha channel.

For example, to convert a PictureMaker file called **brush** to a SOFTIMAGE 3D file called **brush1** from frames 1 to 100 with a step of 1, type:

```
pm2soft brush brush1 -s 1 100 1
```

Raw to SOFTIMAGE 3D

The **raw** standalone converts a raw image into a SOFTIMAGE 3D picture file.

The raw image must be stored in three or four files with the same name, but different extensions. The extensions are **.r** for the red channel, **.g** for the green channel, **.b** for the blue channel, and **.a** for the alpha channel. The **.a** file containing the alpha channel is optional.

To convert the raw image into a SOFTIMAGE 3D picture file, you must know the image width and height in pixels and specify them on the command line.

For example, to convert a raw image file called **hide.r** to a SOFTIMAGE 3D file called **hide1** with a width of 250 and a height of 200 from frames 1 to 100 with a step of 1, type:

```
raw2soft hide hide1 250 200 -s 1 100 1
```

RGB/RGBA to SOFTIMAGE 3D

The **rgb2soft** standalone converts an RGB picture file (**.rgb** or **.rgba**) into a SOFTIMAGE 3D picture file (**.pic**).

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 the 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.

For example, to convert an RGB image file called **colour** to a SOFTIMAGE 3D file called **colour1** from frames 1 to 100 with a step of 1, and with a width of 250 pixels and a height of 200 pixels, type:

```
rgb2soft colour colour1 -d 250 200 -s 1 100 1
```

You do not need to specify the width and height when converting Macintosh **.rgb** files that contain a header.

SGI to SOFTIMAGE 3D

The **sgi2soft** standalone converts SGI picture files (.sgi) into SOFTIMAGE 3D picture files (.pic).

For example, to convert an SGI image file called **map** to a SOFTIMAGE 3D file called **map1** from frames 1 to 100 with a step of 1, type:

```
sgi2soft map map1 -s 1 100 1
```

Targa to SOFTIMAGE 3D

The **tga2soft** standalone converts Targa picture files (.tga) into SOFTIMAGE 3D picture files (.pic).

For example, to convert an SGI image file called **target** to a SOFTIMAGE 3D file called **target1** from frames 1 to 100 with a step of 1, type:

```
tga2soft target target1 -s 1 100 1
```

TIFF to SOFTIMAGE 3D

The **tiff2soft** standalone onverts TIFF picture files into SOFTIMAGE 3D picture files.

For example, to convert a TIFF picture file called **argue** to a SOFTIMAGE 3D file called **argue1** from frames 1 to 100 with a step of 1, type:

```
tiff2soft argue argue1 -s 1 100 1
```

Vertigo to SOFTIMAGE 3D

The **vertigo2soft** standalone converts Vertigo picture files (.pic) into SOFTIMAGE 3D picture files (.pic).

Because the extensions of Vertigo and SOFTIMAGE 3D picture file names are both .pic, an S is automatically added to the beginning of the output SOFTIMAGE 3D file name if it is the same as the Vertigo input file name. If the file names are different, no S is added.

If you have specified the *same* file names on the command line and you don't want the S to be added to the SOFTIMAGE 3D file, use the **-p** option.

For example, to convert a Vertigo picture file called **height** to a SOFTIMAGE 3D file also called **height** without adding an S, type:

```
vertigo2soft height height -p
```

Wavefront to SOFTIMAGE 3D

The **wave2soft** standalone converts Wavefront picture files (.rla or .rlb) into SOFTIMAGE 3D picture files (.pic).

For example, to convert a Wavefront picture file called **surf** to a SOFTIMAGE 3D file called **surf1** from frames 1 to 100 with a step of 1, type:

```
wave2soft surf surf1 -s 1 100 1
```

YUV to SOFTIMAGE 3D

The **yuv2soft** standalone converts YUV picture files (.yuv) into SOFTIMAGE 3D picture files (.pic).



The standard (NTSC or PAL) for YUV files is defined by environment variable **ABEKAS** (default is NTSC).

For example, to convert a YUV picture file called **wxy** to a SOFTIMAGE 3D file called **wxy1** from frames 1 to 100 with a step of 1, type:

```
yuv2soft wxy wxy1 -s 1 100 1
```

Importing Objects

Although not described here, the translator standalones for DXF (**dx2soft**) and IGES (**iges2soft**) are described fully in the *DXF* and *IGES* HTML files on the *On-line Documentation* CD.

The **Import → Objects** commands loads objects created with another software into SOFTIMAGE 3D format. The submenu displays the formats from which you can choose.

1. Choose the **Import → Objects** command in the Tools module.
Then choose the subcommand of for the appropriate file format.
The standard database browser appears.
2. Select a file name by clicking it.
3. Click Ok to proceed. The object is loaded.

Vertigo objects do not have normals. We recommend using the **Effect → Randomize** command in the Model module to process the normals using 0 (zero) for the y and z values.

Saturn to SOFTIMAGE 3D

The **Import** ➔ **Objects** ➔ **saturnImport** command allows you to import Saturn™ files (models, materials, textures, and animation) into SOFTIMAGE 3D.

1. Choose the **Import** ➔ **Objects** ➔ **saturnImport** command in the Tools module. The Saturn Import Setup dialogue box is displayed.
2. Type the prefix of the Saturn file you want to import in the **File Name** text box. If you enter a valid Saturn suffix (**.mdl**, **.txr**, or **.mtn**), it is ignored and only the prefix is used.

Alternatively, click the **Select** button to open a file browser and select a file. Only prefixes with corresponding **.mdl** files are shown in the browser. This is because **.mdl** files contain the geometry data required by texture and animation files.

3. Select the **Animation** option to import animation from the corresponding Saturn **.mtn** file, if any.
4. Select the **Textures** option to imports textures from the corresponding Saturn **.txr** file, if any. If you use this option, you must specify a location for the converted texture files in the **Output Directory** text box.
5. Click **Ok** to import the file.

Converting Files to Export

You can convert files for exporting from SOFTIMAGE 3D using two main ways: the **Export → Images** commands in the Tools module or the conversion standalones for specific file types. To export objects, use the **Export → Object** commands.

For a complete list of the command line options for these standalones, see their descriptions in the *Standalones* HTML file on the *On-line Documentation* CD.

Exporting Images

The **Export → Images** commands convert images created (.pic files) in SOFTIMAGE 3D into another format. The submenu displays these formats from which you can choose.

1. Choose the **Export → Images** command in the Tools module. Then choose the appropriate subcommand (file format).
 2. In the browser that appears, select an image file name to export or specify a name in the **File name** text box. When you choose a file to export, a message box may appear asking you to enter information.
- When exporting to Raw images, the width and height of the picture is shown at the bottom of the screen. Since these file formats don't store the picture's width and height, you should make note of them because you will need them if you ever want to import the image back into SOFTIMAGE 3D.
 - When exporting to RGB or RGBA images, the width and height of the picture is shown at the bottom of the screen. Since these file formats don't store the picture's width and height, you should make note of them because you will need them if you ever want to import the image back into SOFTIMAGE 3D.
 - When exporting to Targa, you can store the image in a compressed format (default) or as raw pixels.
 - When exporting to TIFF, you can choose the following options:
 - Put a MacBinary header in the file, which is useful for some applications.
 - Use PackBit compression to compress the file.
 - Use Planar configuration: 1 or 2.

- Specify a suffix (extension) for the file.
 - When exporting to Vertigo, the Protect option enables or disables the protection on the file. Since both Vertigo and SOFTIMAGE 3D file formats use **.pic** as a suffix, the Protect option adds a “v” before the name of the converted Vertigo file.
3. Click Ok to proceed or Cancel to exit the procedure.

SOFTIMAGE 3D to Alias

The **soft2alias** standalone converts SOFTIMAGE 3D picture files (**.pic**) into Alias picture files (**.als**).

For example, to convert a SOFTIMAGE 3D file called **alien** to an Alias file called **alien1** from frames 10 to 50 with a step of 1, type:

```
soft2alias alien alien1 -s 10 50 1
```

SOFTIMAGE 3D to Artisan

The **soft2art** standalone converts SOFTIMAGE 3D picture files (**.pic**) into Artisan picture files (**.art**).

For example, to convert a SOFTIMAGE 3D file called **picasso** to an Artisan file called **picasso1** from frames 10 to 50 with a step of 1, type:

```
soft2art picasso picasso1 -s 10 50 1
```

SOFTIMAGE 3D to Aurora

The **soft2aurora** standalone converts SOFTIMAGE 3D picture files (**.pic**) into Aurora picture files (**.sim**).

For example, to convert a SOFTIMAGE 3D file called **borealis** to an Aurora file called **borealis1** from frames 10 to 50 with a step of 1, type:

```
soft2aurora borealis borealis1 -s 10 50 1
```

SOFTIMAGE 3D to Colour-mapped

The **soft2map** standalone converts SOFTIMAGE 3D true colour image file (**.pic**) to a colour-mapped image file (**.icn**). You can specify an existing colour-map file, or let the standalone choose colours and build a colour map for you.

For example, to convert a SOFTIMAGE 3D file called **colour** to a colour-mapped file called **colour1** from frames 10 to 50 with a step of 1 and using an existing colour map (-c) called **map3**, type:

```
soft2map colour colour1 -c map3 -s 10 50 1
```

SOFTIMAGE 3D to Everest

The **soft2everest** standalone converts SOFTIMAGE 3D picture files (.pic) into Everest picture files (.ima).

For example, to convert a SOFTIMAGE 3D file called **mountain** to an Everest file called **mountain1** from frames 10 to 50 with a step of 1, type:

```
soft2everest mountain mountain1 -s 10 50 1
```

SOFTIMAGE 3D to Macintosh PICT

The **soft2pict** standalone converts SOFTIMAGE 3D picture files (.pic) into Macintosh PICT version 2 files (.pict).

For example, to convert a SOFTIMAGE 3D file called **spartan** to a Macintosh PICT file called **spartan1** from frames 10 to 50 with a step of 1, type:

```
soft2pict spartan spartan1 -s 10 50 1
```

The extension **.pict** is added automatically to the Macintosh file.

The **lin2pict** standalone converts a line file (.lin) into a Macintosh PICT version 2 file (.pict).

For example, to convert a SOFTIMAGE 3D line file called **apple** to a Macintosh PICT file called **apple1** with a pixel width of 250, from frames 10 to 50 with a step of 1, type:

```
lin2pict apple apple1 250 -s 10 50 1
```

The extension **.pict** is added automatically to the Macintosh file.

SOFTIMAGE 3D to Matisse

The **soft2r** standalone converts SOFTIMAGE 3D picture files (.pic) into Matisse files (.r).



Currently, only the 24-bit Matisse **.r** format is supported: the 8-bit format using colour look-up tables is not.

For example, to convert a SOFTIMAGE 3D file called **dance** to a Matisse file called **dance1** from frames 10 to 50 with a step of 1, type:

```
soft2r dance dance1 -s 10 50 1
```

SOFTIMAGE 3D to PictureMaker

The **soft2pm** standalone converts SOFTIMAGE 3D picture files (.pic) to PictureMaker format (.r8, .g8, .b8, and .a8).

For example, to convert a SOFTIMAGE 3D file called **brush** to a PictureMaker file called **brush1** from frames 10 to 50 with a step of 1, type:

```
soft2pm brush brush1 -s 10 50 1
```

SOFTIMAGE 3D to Raw

The **soft2raw** standalone converts SOFTIMAGE 3D picture files (.pic) into raw images.

The raw image will be stored in three or four files with the same name, but different extensions. The extensions are .r for the red channel, .g for the green channel, .b for the blue channel, and .a for the alpha channel. The .a file containing the alpha channel is optional.

For example, to convert a SOFTIMAGE 3D file called **hide** to a raw image file called **hide1** from frames 10 to 50 with a step of 1, type:

```
soft2raw hide hide1 -s 10 50 1
```

SOFTIMAGE 3D to RGB/RGBA

The **soft2rgb** standalone converts SOFTIMAGE 3D picture files (.pic) into RGB picture files (.rgb or .rgba).

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 the Macintosh place a header with width and height information in files with the .rgb extension.

soft2rgb creates only pure, plain RGB files. These files may not be understood by other applications that use the .rgb extension. In addition, the height and width of the image is not stored in the RGB file; you will need to remember the picture's resolution.

For example, to convert a SOFTIMAGE 3D file called **colour** to an RGB image file called **colour1** from frames 10 to 50 with a step of 1, type:

```
soft2rgb colour colour1 -s 10 50 1
```

SOFTIMAGE 3D to SGI

The **soft2sgi** standalone converts SOFTIMAGE 3D picture files (**.pic**) into SGI picture files (**.sgi**).

For example, to convert a SOFTIMAGE 3D file called **map** to an SGI picture file called **map1** from frames 10 to 50 with a step of 1, type:

```
soft2sgi map map1 -s 10 50 1
```

SOFTIMAGE 3D to Targa

The **soft2tga** standalone converts SOFTIMAGE 3D picture files (**.pic**) into Targa picture files (**.tga**).

For example, to convert a SOFTIMAGE 3D file called **target** to a Targa image file called **target1** from frames 10 to 50 with a step of 1, type:

```
soft2tga target target1 -s 10 50 1
```

SOFTIMAGE 3D to TIFF

The **soft2tiff** standalone converts SOFTIMAGE picture files (**.pic**) into TIFF (**.tif**) files.

For example, to convert a SOFTIMAGE 3D file called **argue** to a TIFF image file called **argue1** and create a MacBinary heading for the output file (to be transferred to a Macintosh), type:

```
soft2tiff argue argue1 -m
```

Don't forget the extension of the output file name if the software that is going to read the image needs one.

SOFTIMAGE 3D to Vertigo

The **soft2vertigo** standalone converts SOFTIMAGE 3D picture files (**.pic**) into Vertigo picture files (**.pic**).

Because the extensions of Vertigo and SOFTIMAGE 3D picture file names are both **.pic**, a **V** is automatically added to the beginning of the output Vertigo file name if it is the same as the SOFTIMAGE 3D input file name. If the file names are different, no **V** is added.

If you have specified the *same* file names on the command line and you don't want the **V** to be added to the Vertigo file, use the **-p** option.

For example, to convert a SOFTIMAGE 3D picture file called **height** to a Vertigo file also called **height** without adding a **V**, type:

```
soft2vertigo height height -p
```

SOFTIMAGE 3D to Wavefront

The **soft2wav** standalone converts SOFTIMAGE 3D picture files (**.pic**) into Wavefront picture files (**.rla** or **.rlb**).

For example, to convert a SOFTIMAGE 3D file called **surf** to a Wavefront image file called **surf1** from frames 10 to 50 with a step of 1, type:

```
soft2wave surf surf1 -s 10 50 1
```

SOFTIMAGE 3D to YUV

The **soft2yuv** standalone converts SOFTIMAGE 3D picture files (**.pic**) into YUV picture files (**.yuv**).

A YUV image must have a width of 720 pixels and a height of 486 pixels (NTSC standard) or 576 (PAL standard) pixels. The standard is defined by the environment variable **ABEKAS**. The default standard is NTSC.

If the SOFTIMAGE 3D picture does not correspond to these dimensions, it is cropped or filled with black when you convert it.

For example, to convert a SOFTIMAGE 3D file called **wxy** to a YUV image file called **wxy1** from frames 10 to 50 with a step of 1, type:

```
soft2yuv wxy wxy1 -s 10 50 1
```

SOFTIMAGE 3D to 3D Studio

The **soft3ds** standalone is the Animatix Object File Converter that converts files from 3D Studio (**.3DS** version 3.0) to SOFTIMAGE 3D (version 2.65) format. The standalone converts an extensive subset of geometry, lights, cameras, materials, 2D textures, and animation.

Note

The files written by 3D Studio version 2.0 contain information not present in later versions, and this information is ignored, resulting in defaults for certain parameters.

```
soft3ds <3DS filename> <dbase> <scene>  
[-r <rPath>] [-m <mPath>]
```

where:

<3DS filename> is the name of the 3D Studio file (including extension).

<dbase> is the SOFTIMAGE 3D database name.

<scene> is the SOFTIMAGE 3D scene name.

<rPath> is the full path to the SOFTIMAGE 3D resource directory (location of Relations).

<mPath> is the path to the 3D Studio material map files directory.

Name Mangling

When the object converter performs a conversion between an element (such as a light, camera, material, or polygon mesh object) in one format to an equivalent item in another format, it attempts to match the name of the converted item as closely as possible to that of the original item.

Since permitted name lengths and format differ between the SOFTIMAGE 3D and 3D Studio environments, and since a name may already be in use, the object converter may be forced to alter names during conversion. This process is called “name mangling” and uses the following algorithm:

As names (called “base names”) are encountered during the conversion processes, they are registered with the program. If a base name has previously been registered, a numeric suffix (starting at 01) is added to the base name to produce a different name. If this name is also in use, the suffix is incremented and the process repeated until the new name is unique. For names used in 3D Studio, characters are stripped from the base name as required so that the combination of base name and suffix does not exceed the maximum length permitted for the type of item being referenced (objects, lights, cameras: 10 characters; materials: 16 characters). Name mangling for DOS file names is also performed. In this case, the base name is formed by using the first eight characters of the input name. This is further truncated, if necessary, to remove any characters after and including the first period (if the first character is a period, it is replaced by a “-” character).

3D Studio File Read

This section describes the information that will be converted:

- **Ambient Light:** This is converted to **Atmosphere → Ambience** in SOFTIMAGE 3D. RGB colour maps directly to RGB colour.
- **Global Shadow Parameters:** These are applied to each light if local settings are not present for that light.
- **Fog/Depth Cueing:** This is converted to **Atmosphere → Depth Fading**. The 3D Studio initial % setting is ignored (no equivalent), the start and end distances and fog RGB colour map directly, and the final % setting maps to Transparency.
- **Background:** 3D Studio backgrounds have no direct equivalent in SOFTIMAGE 3D and are not converted.
- **Viewport Information:** This information is not compatible between 3D Studio and SOFTIMAGE 3D and is not converted.
- **Cameras:** The first camera encountered in the 3D Studio file is read and converted to the camera in SOFTIMAGE 3D. Camera position, interest, hidden status, near plane clipping, and roll are converted directly. Far plane clipping is not available. The remaining camera parameters are set to the defaults of SOFTIMAGE 3D.

Note

If no camera exists in the 3D Studio file, a camera is created with the SOFTIMAGE 3D default parameters.

Any hierarchical relationship defined in the 3D Studio file between cameras and other objects is ignored (SOFTIMAGE 3D's camera cannot be part of a hierarchy).

Lights

- 3D Studio omni lights are converted to SOFTIMAGE 3D point lights of the same colour and position with no shadow set. If attenuation is enabled for a light, the SOFTIMAGE 3D falloff parameters are set according to the 3D Studio file range values specified for that light. If objects are included with the light, the light is set as a selective and exclusive light. The included objects are associated with the light.
- The umbra, penumbra, and shadow parameters are set to SOFTIMAGE 3D defaults. 3D Studio spot lights are converted to SOFTIMAGE 3D spot lights of the same colour, position, interest, and cone/spread angles.

- If shadows are enabled for the light, they are set to either raytraced or depth-mapped accordingly. Depth-mapped shadow parameters are set for map resolution and filter size, when applicable. If attenuation is enabled for a light, the SOFTIMAGE 3D falloff parameters are set according to the 3D Studio file range values specified for that light.
- If objects are excluded from the light, the light is set as a selective and exclusive light. The excluded objects are associated with the light. The umbra and penumbra parameters are set to SOFTIMAGE 3D defaults.
- 3D Studio rectangular and projector spot lights are converted to normal spot lights in SOFTIMAGE 3D, as previously described.

Note

Any hierarchical relationship defined in the 3D Studio file between lights and other objects is ignored.

Material Definitions

Each valid material definition in the 3D Studio file is converted to one material definition and zero or more associated 2D textures. The name of the material definition is the same name as that in the 3D Studio file. 2D textures associated with a material use names derived from the material name. Note that since SOFTIMAGE 3D does not support independent mark maps, these will not be converted from 3D Studio materials.

Materials definitions are converted as follows:

- Wireframe and flat-shaded material types are converted to Constant-shaded materials.
- Gouraud, Phong, and Metal materials are converted to Phong-shaded materials.
- For all material types other than Phong and Metal, the RGB values for ambient, diffuse, and specular components are converted directly to their equivalents in SOFTIMAGE 3D.
- For Phong and Metal materials, the ambient and diffuse components are converted directly to their equivalents in SOFTIMAGE 3D, while the specular component colour intensity is attenuated by the 3D Studio Shininess Strength slider values. The 3D Studio Shininess value is scaled and converted to the Specular Decay setting. The 3D Studio Transparency slider value is scaled and converted to the Transparency setting.

- In 3D Studio, the Phong Soften toggle control, the Transparency Falloff controls, the Reflection Blur slider, the Self-Illumination slider, and the Face-Map control have no equivalent in SOFTIMAGE 3D and are not converted.

2D Texture Definitions

Due to the differences between 3D Studio and SOFTIMAGE 3D in the way that materials are defined and assigned to objects, the 3D Studio file read process creates one 2D texture definition for each distinct mapping file/mapping assignment combination in 3D Studio.

- 3D Studio material map definitions do not exactly correspond with those in SOFTIMAGE 3D. The default settings for 2D textures may be modified as described in the sections for each map type.
- The Picture Name field is set to the base name of the corresponding 3D Studio map file.
- The Animation setting is determined by the type of file used in the corresponding 3D Studio map. If the map file type is .FLI or .FLC, the animation type is set to Sequence; if the map file type is .IFL, the animation type is set to Script; otherwise the animation type is set to Static.
- The UV Scale, Offset, and Repeats settings are assigned the corresponding values from the 3D Studio map and object. Note that if the correct parametric information for mapping is not present in the 3D Studio file (for example, if the file was created by an earlier version of 3D Studio, or if objects are deformed after mapping is assigned), the mapping conversion cannot be performed accurately.
- The Tiling mode is set to “non-alternating” for both u and v, unless the Mirror option is set for the 3D Studio bitmap, in which case the tiling is set to “alternating” for both u and v.
- The Swap UV option is deselected.
- The Pixel Interpolation option is selected.
- The Reflection Map Rotation field and Intensity slider are set to 0.0.
- The Mapping Method is set to XY, Cylindrical, or Spherical, depending on the mapping method used for this material’s application in 3D Studio.

- The Map Component selection is set to RGB Intensity and the Transparency, Reflectivity, and Roughness sliders are set to 0.0.
- All Cropping values are set to 0.0.
- The Blending option is set to Without Mask, and the Overall Blending, Ambient, Diffuse, and Specular sliders are set to 0.0.
- For Texture maps (Texture1, Texture2), a 2D texture is generated with the Blending control set to Alpha Channel Mask, unless the 3D Studio texture has the Ignore Alpha button set, in which case the Blending control is set to Without Mask.
- The Overall Blending slider is set to the scaled value from the 3D Studio Amount slider for the appropriate map and the Ambient, Diffuse, and Specular sliders are all set to 1.0.
- For the Specular map, a 2D texture is generated with the Blending control set to Alpha Channel Mask, unless the 3D Studio texture has the Ignore Alpha button set, in which case the Blending control is set to Without Mask. The Overall Blending slider is set to the scaled value from the 3D Studio Amount slider for that map and the Specular sliders is set to 1.0.
- For the Reflection map, a 2D texture is generated with the Reflection Map Intensity slider set to the scaled value from the 3D Studio Amount slider for the Reflection map.
- For the Opacity Map, the Roughness slider is set to the Amount slider for the Opacity map.
- For the Bump Map, the Roughness slider is set to the scaled value from the 3D Studio Amount slider for the Bump map.
- The 3D Studio Shininess Map, Self-Illumination maps, and SXP (procedural) textures cannot be converted to SOFTIMAGE 3D.

See *Image File Conversion Process* on page 65 for more information on how image file conversion is managed.

Object Geometry

- Each mesh object from the 3D Studio file is converted to a polygon mesh object in SOFTIMAGE 3D and assigned the same name as the 3D Studio object. Automatic Discontinuity is set for each mesh using SOFTIMAGE 3D's default angle of 60 degrees. This setting is overridden using the discontinuity edge flag to simulate the effect of 3D Studio's smoothing groups.

- Materials created are associated with the appropriate mesh items.
- Any hierarchical relationship defined in the 3D Studio file for objects is recreated in SOFTIMAGE 3D. 3D Studio instance objects are converted to separate objects in SOFTIMAGE 3D.

Animation

- 3D Studio animations for mesh objects, lights, and cameras using scale, rotation, and translation (SRT) are converted directly into their equivalent in SOFTIMAGE 3D – keyframes are generated on a one-to-one basis, with function curve shapes being determined by keyframe settings. Additional animated parameters for lights (colour, hotspot, and falloff) and cameras (roll and field of view) are similarly converted.
- 3D Studio morphing of objects is converted by using SOFTIMAGE 3D shape animation techniques (such as creating a series of different shapes with the same number of vertices and assigning them to the appropriate keyframes).
- 3D Studio material morphing is converted by applying the converted material definition to the appropriate object/keyframe combination.
- Pivot points for the objects in 3D Studio are converted to object centres in SOFTIMAGE 3D.
- 3D Studio hide keys are converted to an equivalent activation function curve in SOFTIMAGE 3D
- Dummy objects in 3D Studio are converted to null objects in SOFTIMAGE 3D.

Image File Conversion Process

When 3D Studio materials containing maps are converted to their equivalent in SOFTIMAGE 3D, the image files representing those maps are also converted. This process is accomplished by invoking the Animatix Image File Converter for each of the image files. The object converter is responsible for determining the names of the input file passed to the image file converter and the output file created by the image file converter. The output file type is determined by the input file type passed to the image file converter.

In converting 3D Studio to SOFTIMAGE 3D image files, the object converter first retrieves the file name of the 3D Studio material map

component to be converted. The object converter then determines the output file name, which is achieved by taking the input file name and extension and appending the extension **.pic** (for example, the input file **marble.tga** becomes **marble.tga.pic**).

The object converter then invokes the image file converter with the input and output file names, specifying the **-k** option to prevent existing files with the same name as the output file name from being overwritten. The output file name is then referenced in the SOFTIMAGE 3D 2D texture that is created. If the input file type is **.IFL**, the object converter traverses the list of files specified and converts each to the SOFTIMAGE 3D format, as described previously. The object translator also writes all the converted files' names to a script file (this has the same name as the source **.IFL** file and has the extension **.scr** appended).

Exporting Objects

Although not described here, the translators for DXF (**soft2dxf**) and IGES (**soft2iges**) are described fully in the *DXF* and *IGES* HTML files on the *On-line Documentation* CD.

The **Export → Objects** commands let you convert an object created in SOFTIMAGE 3D into another format. The submenu displays the Vertigo, Wavefront, and Saturn export formats from which you can choose.

1. Select one or more objects that you want to export.



If you export a NURBS object to Vertigo or Wavefront, it is converted into a polygon mesh object.

2. Choose the **Export → Objects** command in the Tools module. Then choose the appropriate subcommand for the file format.
 - If you choose the **Vertigo** command, the database browser appears. Select the object file name to export or specify a name in the **File name** text box and click Ok.
 - If you choose the **Wavefront** command, a dialogue box appears in which you can convert patch objects to polygon mesh and/or splines (curves) to lines in Wavefront.

When you click Ok, the database browser appears in which you can select the object file name to export or specify a name in the **File name** text box and click Ok.

- For a description of the options available with the **saturnExport** command, see the next section.

SOFTIMAGE 3D to Saturn

The **Export → Objects → saturnExport** command in the Tools module allows Saturn™ game designers to export a hierarchy of models, textures, and animation as Saturn data in Saturn's SGL file format.

For more details on the parameters for using this command, see *saturnExport* on page 639 in the *Reference Guide*.

The hierarchy must contain at least one polygon mesh model. Any other type of model is automatically removed from the exported hierarchy, and transformations are automatically updated to compensate for the removal.

Any periods (.) or hyphens (-) in model or texture names are converted to underscores (_) in order to avoid problems when compiling the resulting files.

Note

All selected models must be in the same hierarchy.

In addition, this command does not automatically tessellate invalid polygons (that is, polygons other than triangles and quadrilaterals) or non-planar polygons. If the model contains such polygons, a message box warns you and gives you the option to cancel or continue.

1. Choose **Export → Objects → saturnExport** command in the Tools module. The Saturn Export Setup dialogue box is displayed.
2. Type a prefix for the exported Saturn files in the **File Name** text box. Alternatively, click the **Select** button to open a file browser and select an existing file name.

If you enter a valid Saturn suffix (**.mdl**, **.txr**, or **.mtn**), it is ignored and only the prefix is used.

3. Select the **Generate Geometry (.mdl)** option if you want to generate an **.mdl** file containing the geometry information. You can also select the **Model Shading** option to include the Saturn SGL shading attribute for the exported models.
4. Select the **Generate Textures (.txr)** option to generate a **.txr** file containing the model's texture data. SOFTIMAGE 3D generates one rectangular texture "slice" for each texture-mapped polygon.

5. If you selected the **Generate Textures (.txr)** option, you can select **Smoothing** to control the smoothing of colours when textures are resampled. When this option is selected, a bilinear interpolation method is used to enhance the quality of the generated texture slices. When this option is not selected, each pixel in the resulting texture map uses the colour of the closest pixel in the original map.

Note When this option is selected, the smoothing process might generate colours that were not used in the original texture map.

6. You can also select the **Use Projected Polygon Area** option to control how textures are resampled. If this option is selected, the texture slice size is computed from the actual polygon surface area in SOFTIMAGE units. The resolution of the generated texture slice is obtained by multiplying this surface area with the **Sampling Density** value.

When this option is not selected, the resolution of the texture slice is obtained from the resolution of an average rectangular area corresponding to the region of the texture used by the polygon. This value is then multiplied by the **Sampling Density** value. In this case, the size of the texture slice depends on the resolution of the original texture map.

7. The **Sampling Density** option specifies the degree of resampling in texture elements (*texels*) per SOFTIMAGE unit. The default is 40. Use higher values to keep more detail, and lower values to eliminate detail.
8. Select the **Generate Animation (.mtn)** option to generate an **.mtn** file containing the model's animation data in time (that is, local transformations of the models from frame to frame). If the values for one frame are identical to the previous frame, the animation data is not exported.

SOFTIMAGE 3D to VRML

The **Save → vrml** command in the Model module lets you convert polygon mesh objects in a scene to VRML format (**.wrl** for world).

This version supports Point and Spot lights, as well as multiple lights. It also supports local and global materials, and 2D textures.

Note

The texture needs to be UV mapped on the object using the **Txt_Oper → Info UV Coord.** command in the Matter module with vertices generation. Each texture that is attached to a model being converted is automatically converted in a JPEG file format during the object or scene conversion. The conversion is made using the **Imgconvert** tool.

1. When you choose the **Save → vrml** command, the VRML Plug-In dialogue box is displayed.
2. Specify a **File Name** in which to save the VRML file. The **.wrl** extension is added automatically. Click the **Browser** button to have access to existing file names in databases.
3. Select the **Options** you want for the conversion:
 - If you select the **Save as multiple inline files** option, every polygon mesh object in the scene file is converted and saved in a separate file. A file is also saved for the entire scene which references each polygon mesh object as an inline file. Inline files disregard hierarchical structures.
 - Select the **Save selected model** option to save only the selected objects in the scene. If you don't select this option, the entire scene is converted, including all objects and lights.
 - Select the **VRML preview** option to actually do the scene conversion and then start your favourite VRML viewer directly in SOFTIMAGE 3D. You can then view the resulting conversion right away. Before you can use this option, add the **SI_VRML** environment variable to your **.softimage** file. It should point to where the viewer is stored on your disk, such as **/net/bin/favourite viewer**.
4. Click **Ok** to convert the scene or **Cancel** to cancel the operation.

C H A P T E R S I X

Combining Images

Compositing Images

To composite layers of images, you can use either the **Composite** menu cell in the Tools module or the **composite** standalone. The standalone offers a few more options than the menu cell, such as superimposed slides and negatives, blends, halo effects, etc.

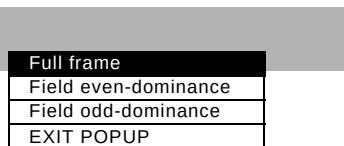
Using the Composite Menu Cell

The **Composite** menu cell allows you to create new images by adding layers of images on top of each other.

1. Choose the **Composite** menu cell in the Tools module. The Compositor dialogue box appears.
2. Type the **Output File Name** in its text box. To select an existing file from a browser, click the **Select** button.
3. If you intend to composite a sequence, select the **Sequence** option and type the **Start Frame**, **End Frame**, and **Step** in their respective text boxes. Type the number of times each output frame is to be repeated in the **Repeat each frame** text box.

If the layer is intended to be composited as a static frame, deselect the **Sequence** option.

4. If you want to composite a field-rendered image, select the **Field** type from the menu:



5. If the layers do not cover all the pixels on the final image, you can set a **Background Colour** by specifying its value from 0 to 1 for the Red, Green, and Blue components.
6. On the right side of the dialogue box, there are several options that allow you to define the output resolution. Do one of the following:
 - If you want the output image to have the same resolution as the first layer, select **Info of first layer**. The **Width**, **Height** and **Pixel Ratio** options will be set automatically. The pixel ratio refers to the width and height of each pixel rather than the aspect ratio, which is the number of pixels horizontally and vertically in the whole picture.

- Define the output resolution with the **User Defined** option and typing in values for **Width**, **Height** and **Pixel Ratio**. Again, the pixel ratio refers to the width and height of each pixel.
- Define the resolution according to the picture aspect by selecting **Y is relative to X** or **X is relative to Y**. Make sure the picture aspect is correct. If not, click **Picture Aspect** in the upper-right corner of the dialogue box to access the Picture Format dialogue box.

In this dialogue box, you can define the **Custom** aspect ratio and size, and the appropriate **Cine**, **Slide**, or **Video** options. Select **Accept** or **Cancel** to return to the Compositor dialogue box.

Define the **Width** and the **Pixel Ratio**; the **Height** is set accordingly. (With the option **X is relative to Y**, the same procedure applies except that you define the height, and the width is calculated.)

7. If you need the alpha channel in the final image, select **Alpha Channel in Output**.

If you want the image to be displayed on screen, select **Display Output Image**.

8. Define the layers to be composited by clicking **New Layer**. The Layer dialogue box appears.

Type the layer file name in the **Layer Name** text box or use the **Browser Name** button to select it.

If the file is to be used as a sequence, select **Sequence** and specify the file number at which the layer should start. If the layer is intended to be composited as a static frame, deselect the **Sequence** option.

Three options control how the layer is added into the final picture:

- **Use this layer's alpha channel to composite** it uses the alpha channel so that the black and white tones more or less mask the underlayers. At extremes, black is transparent and white is opaque.
- **This layer is opaque** means that the layer will have no transparency values (opaque).
- **Use fading value** fades the layer using the values specified. The fading value can be animated by setting different values in the **Start Fading Value** and **End Fading Value** text boxes. The values are then interpolated between the starting and ending frames set in the Compositor dialogue box.



This option uses the absolute pixel values of the image, and does not take into consideration the alpha channel of the image. This is useful for doing camera fades between shots. 0 = no fade, and 1 = full fade.

If you like, offset the layer horizontally and vertically. By default, the layer position is defined by the lower left corner (0,0). The offset is measured in pixels. In **Image Horizontal Offset**, positive numbers move the layer to the right, and negative numbers move the layer to the left. In **Image Vertical Offset**, positive numbers move the layer to the top and negative numbers move the layer to the bottom. Offsets can be animated by setting different values in the **Start Offset** and **End Offset** text boxes.

9. When the setup for a layer is completed, click Ok to close the Layer dialogue box and return to the Compositor dialogue box.
 - You can continue to add new layers to the end of the list by clicking **New Layer** for each one and defining the layers as you did in the previous step.
 - You can add a layer between two others by selecting the second layer in the **Layer** scroll box and clicking **New Layer**.
 - You can edit a layer by double-clicking its name in the **Layers** scroll box.
 - You can delete a layer by selecting its name and clicking **Delete Layer**. The **Delete All** button deletes all of the layers in the list.
10. When all the layers have been set up, start the compositing process by clicking Ok.

Using the composite Standalone

The **composite** standalone composites many images into one.

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.

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 centre 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** colours the background white. The layer **logo.shadow** is composited using the **-c** option to force use of the alpha channel over the coloured 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
```

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Using the Watch Standalone

The **Watch** effect is a standalone used to display, convert, compress, and composite pictures or sequences of different formats: SOFTIMAGE 3D, Targa, Silicon Graphics, and Composite YUV from different sources like the SOFTIMAGE 3D renderer or video tape.

When dealing with sequences of images, Watch can accept three methods of frame numbering:

```
Filename.12.ext
```

`Filename.0012.ext`

`Filename0012.ext` (DOS format)

1. Create a picture or sequence in any of the supported formats.
2. Open a shell and type:

`Watch Image_file <options>`

where *Image_file* is the name of the input picture or sequence and *<options>* are the parameters listed below.

3. A display window appears. Use the r, g, b, and a keys on your keyboard to display the red, green, blue, and alpha channels respectively. Pressing the space bar restores the view to normal.

With the **-b** option, Watch can composite two pictures together. These two pictures can be of two different formats and have two different resolutions. The output picture will have the background picture resolution, and the foreground picture will be placed starting from the bottom left.

This example composites the first picture called **pic.1001.yuv** with the picture called **back0101.pic** as the background image.

`Watch pic.1001.yuv -b back0101.pic`

Watch automatically detects the file format of the picture (input and output) according to the file extension (**.pic** for SOFTIMAGE 3D, **.sgi** for Silicon Graphics, **.tga** for Targa, **.yuv** for Composite YUV).

Watch can detect compressed files with the IRIX **compress** or **pack** commands. These commands must be available in your path definition. To check if they are available, type the following in a shell:

```
which compress
```

or

```
which pack
```

When dealing with compressed images, Watch decompresses (or unpacks) the picture before converting it.

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Interleaving Fields

The **interleave** standalone merges pictures together by interleaving their fields (scan lines).

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]`.

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
```

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

C H A P T E R S E V E N

Modifying Images

Introduction

Once an image is rendered or imported, you can modify it in many ways. Some examples are changing its size, changing the colour values and gradations, rotating, and flipping the images. You can also create an alpha channel or modify an existing one for an image after it has been rendered, or extract any channel (red, green, blue, or alpha) from an image.

Creating or Modifying an Alpha Channel

You can use the **alpha** standalone to create an alpha channel for a picture file, or to modify the existing alpha channel.

An intensity range must be given for each channel. If the intensity of a pixel is within the range, the alpha value for this pixel is set to zero; otherwise, it is set to the alpha value specified on the command line.

To use **alpha** effectively, you should have a good idea of the RGB content of the picture file. Intensity ranges that work with one picture might not work well with another.

This example uses the **bets** file from frames 4 to 100 (-s) with a step of 2 and creates a **bets2** file with an alpha channel.

```
alpha bets bets2 1 2 255 3 80 200 255  
-s 4 100 2
```

The alpha value is 1, and the minimum and maximum values for red, green, and blue are as follows:

- Red minimum of 2 and maximum of 255
- Green minimum of 3 and maximum of 80
- Blue minimum of 200 and maximum of 255

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Extracting a Channel

The **extract** standalone extracts a channel from a picture file and creates another picture whose channels are all copies of the extracted channel. This creates a grey-scale image that corresponds to the extracted channel.

In this example, the blue channel (**b**) is extracted from the **colour** file and saved as the **blue** file, starting at frame 1 and ending at frame 100 with a step of 2:

```
extract colour blue b -s 1 100 2
```

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Creating Gradations

The **gradation** standalone creates a gradation between colours and saves it as a picture. You can vary the colours horizontally, vertically, or between all four corners.

The colour values must be in the range [0.0, 1.0].

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 RGB values defined for the top (0, 0.15, 0.15) as a greenish colour and then the bottom (0.6, 0.45, 0) for a yellowish colour.

```
gradation finalgrad 150 150 -H 0 0.15 0.15  
0.6 0.45 0
```

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Creating a High-Contrast Image

The **contrast** standalone creates a high-contrast version of an image.

Pixels with an intensity smaller than the given threshold are white, and pixels with an intensity greater than the threshold are black. The result can be inverted with the **-i** option.

For example, the **low** image is saved as **high** with the starting threshold as 0 and the ending threshold (**-e**) as 1 with the black and white pixels inverted (**-i**).

```
contrast low high 0 -e 1 -i
```

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Changing the Colour Values

Changing the Hue, Lightness, and Saturation

The **hls** standalone modifies the hue, lightness, and saturation of an image.

To modify the hue, specify an angle in degrees. To understand how this works, imagine a chromatic circle. The hue of each pixel in the image is determined by its position on the chromatic circle. If the underlying circle is rotated by the specified angle, the coordinates of each pixel determine a new hue.

To modify the lightness, specify a number in the range of -1.0 to 1.0 to be added to the current lightness value. You can change the saturation in the same way.

This example changes the hue to 300 degrees, adds a value of 0.5 to the lightness, and adds -1.0 to the saturation for the **chroma** file (a single frame) and saves it as **chroma2**.

```
hls chroma chroma2 300 0.5 -1
```

Changing the Colour Intensity

The **intensities** standalone scales colour values up or down relative to an origin called the pivot. This is useful if you want to adjust the intensity of a rendered image without rendering the image again.

The following example fades in from black over 10 frames with a step of 1 (**-s 1 10 1**) and the messages shown in the console (**-v**). The start factors for each channel are 0 (black) and the end factors (**-e**) are 1 (full colour).

```
intensities myInput myOutput 0 0 0 0 -e 1 1 1 1  
-s 1 10 1 -v
```

The factors are the degree by which you want to adjust each channel's intensity. A factor in the range [0, 1] scales down the channel's intensity, and a factor above 1.0 scales it up.

For a complete list of the command line options for the **hls** and **intensities** standalones, see their descriptions in the *Standalones* HTML file on the *On-line Documentation* CD.

Resizing, Rotating, and Flipping Images

There are a number of ways to resize, rotate, and flip an image after it has been rendered. You can do all three of these effects using the **Effects** menu cell in the Tools module.

To do specific effects, you can use these standalones:

- To enlarge an image, use the **enlarge**, **scale**, or **zoom** standalones.
- To reduce an image, use the **reduce** or **scale** standalones.
- To rotate and flip an image, use the **effects** or **rotate** standalones.

For a complete list of the command line options for these standalones, see their descriptions in the *Standalones* HTML file on the *On-line Documentation* CD.

Using the Effects Menu Cell

The **Effects** menu cell allows you to perform effects on pictures. These effects include scaling, rotation, and flipping. The processed picture is saved in a new **.pic** file.

1. Choose the **Effects** menu cell in the Tools module. The Picture Effects dialogue box appears.
2. Select the frame to be processed from the scroll box by clicking its file name. The file name is then displayed beside **Input file**.
3. Enter a name for the file to be created by typing it in the **Output file** text box.
4. Select the **SCALING**, **ROTATION**, or **FLIPPING** options as you like.
 - For **SCALING**, you can select the final **Size** and specify its number of pixels in the X and Y text boxes. Select the Mitchell filter option to blur the image by averaging the associated pixels. You can also select the **Factor** option, choose to either **Reduce** or **Enlarge**, and set the X and Y values.
 - For **ROTATION**, select the degrees that you want to rotate the image (90, 180, or 270 degrees).
 - For **FLIPPING**, select **Horizontally** or **Vertically**, or both.
5. Select the **Display on screen** option to see the image being processed.

6. Click **Process** to do just that. A message appears in the status line when finished.
7. Click **Exit** when you're done.

Scaling an Image

The **scale** standalone scales the width and height of an image up or down. The scaling factors do not need to be integers.

To scale by integer factors, you might want to use the **reduce** or **enlarge** standalones. These two programs are faster, but run only with integers.

This program is optimal with scaling factors in the range of 0 to 2. For scaling factors larger than 2, it is preferable to use the **zoom** standalone.

This example scales down the **fish** sequence to 100 pixels wide by 85 pixels high (-S) from frames 1 to 200 with a step of 2 (-s 1 200 2).

```
scale fish fish2 -S 100 85 -s 1 200 2
```

The -S option lets you specify the exact resulting size. If you don't use the -S option, the width value is the horizontal scaling factor and the height value is the vertical scaling factor.

Here's another example with the **torch2** file being scaled on x to 500 and on y to 424 (-S) for the sequence from frames 61 to 210 with a step of 1 and the verbose option on (-v). The resulting file is called **flambeau**.

```
scale torch2 flambeau -S 500 424 -s 61 210 1 -v
```

Enlarging an Image

To enlarge a picture by a non-integer value, use **scale** or **zoom**. The **zoom** standalone also allows you to add blur to obtain better results.

The **enlarge** standalone duplicates pixels to enlarge pictures horizontally or vertically by an integer value.

This example enlarges the **bgsky** image to **Bgsky** from frames 1 to 72 with a step of 1 (-s 1 72 1), x and y factors of 4, and the messages displayed (-v).

```
enlarge bgsky Bgsky 4 4 -s 1 72 1 -v
```

The **zoom** standalone enlarges SOFTIMAGE 3D pictures by any factor greater than 1.0.

The following example zooms **test.1** and saves the result as **test.2**. Since only the width of the final image is specified (**-x**), the height is calculated automatically.

```
zoom test.1 test.2 -x 506 -v
```

The **-x** and **-y** options let you specify the exact resulting size in pixels.

If you do *not* use the **-x** and **-y** options, **<x factor>** is the horizontal scaling factor and **<y factor>** is the vertical scaling factor instead of the absolute value in pixels. These factors can be any value greater than 1.

Note

To maintain the aspect ratio, don't specify a value of zero (0) as you would in the **Effects** menu cell. This actually creates the image with zero pixels in that direction.

Reducing an Image

The **reduce** standalone reduces the size of an image. The reduction factors must be integers. To reduce the image, groups of pixels are averaged into single pixels.

If you want to reduce an image by factors that are not integers, use the **scale** standalone — it's slower, but provides better results.

This example reduces the **fat** image and saves it as **Fat** from frames 1 to 40 with a step of 1 (**-s 1 40 1**), an initial x and y factors of 2 and 3.

```
reduce fat Fat 2 3 -s 1 40 1
```

A scaling factor of 2 produces an image that is half the width of the original, and a scaling factor of 3 produces an image that is one-third the height of the original.

Flipping and Rotating Images

The **effect** standalone flips and rotates images. The image can be flipped horizontally, flipped vertically, or rotated by 90, 180, or 270 degrees counterclockwise.

If multiple operations are specified, they are performed in the order that they appear on the command line.

To rotate very large pictures, you should use the **rotate** standalone (see the next section) which doesn't keep the whole image in main memory.

This example flips the **monkey** image horizontally (**-H**) and rotates it by 180 (**-R 180**) degrees over 20 frames with a step of 1 (**-s 1 20 1**). The output file is called **flipmonkey**.

```
effect monkey flipmonkey -H -R 180 -s 1 20 1
```

Rotating an Image

The **rotate** standalone rotates an image by 90, 180, or 270 degrees counterclockwise.

This example rotates the **clock** image 90 degrees over 50 frames with a step of 2 (**-s 1 50 2**). The image is saved to the file called **counterclock**.

```
rotate clock counterclock 90 -s 1 50 2
```

The mutually exclusive options **90**, **180**, **270**, and **-90** specify the angle of rotation counterclockwise. Note that **270** and **-90** produce the same result.

Modifying a Sequence Length

The **Timewarp** effect is a standalone tool that modifies the length of a sequence of images (rendered with SOFTIMAGE 3D or from video tape). You can quickly arrange sequences of rendered pictures to be just a few frames longer or shorter. You can also specify the amount of trail effect you want.

The command line syntax is:

```
Timewarp OrigSeq OrigLength NewSeq NewLength  
Interpolation
```

- *OrigSeq* is the name of the sequence to process.
- *OrigLength* is the original length of the sequence in frames.
- *NewSeq* is the new sequence name.
- *NewLength* is the new length of the resulting sequence in frames.
- *Interpolation* determines how many frames are taken into account (must be 2 or 4).

Note

Step frame value for input and output sequences must be 1. If you specify the same value for *OrigLength* and *NewLength*, Timewarp performs an “average” of the sequence (something like motion blur) mixing several frames into one.

Timewarp Tutorial

1. Open a shell.
2. Go to the directory where you installed the **Rebels** custom effects.
3. Open the **RENDER_PICTURES** directory. You’ll find a sequence called **FourthD_Beethoven** with 200 frames of a Beethoven model rotating around the y-axis in 50 frames over a background.
4. Type:

```
Timewarp FourthD_Beethoven 200 New 75 4
```

FourthD_Beethoven is the name of the sequence to process, **200** is the sequence length in frames, **New** is the new sequence name, **75** is the new sequence length, and **4** is the interpolation.

5. Wait until Timewarp renders the complete new sequence called **New**.
6. Play a SOFTIMAGE 3D flipbook to see the result. The **New** sequence is 75 frames long.

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Inverting Positive and Negative Images

The **negate** standalone creates the negative of a positive image, or a positive of a negative image. For each channel:

$$(\text{new pixel value}) = 255 - (\text{old pixel value})$$

This standalone also reverses the alpha channel, which can be useful for creating a matte.

This example makes a negative image (**blackphoto**) out of the **photo** image over 200 frames with a step of 2 (-s 1 200 2).

```
negate photo blackphoto -s 1 200 2
```

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Adding Noise

The **noise** standalone adds noise (perturbation) to a picture.

The noise can affect only the intensity, or it can affect each colour channel independently.

When adding noise to sequences, you can keep the noise constant from frame to frame (as if the noise was a filter) by using the **-S** option.

This example uses the **maker** file and saves it as **maker2** from frames 1 to 80 with a step of 1 (**-s 1 80 1**). The start range is 0.5 and the **-S** option makes sure that the noise stays constant over all frames in the sequence.

```
noise maker maker2 0.5 -S -s 1 80 1
```

The start range is the largest deviation a pixel can have from its old value. This value can be any number in the range [0, 1].

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Creating a Depth of Field Blur

The Dof effect is a post-processing standalone that simulates Depth of Field (DOF) blurring.

It reads the following parameters from a SOFTIMAGE 3D scene, which are set using the **Camera → Settings** command:

- **Near Focus:** Nearest distance from the camera where objects are in focus measured in SOFTIMAGE 3D units.
- **Far Focus:** Farthest distance from the camera where objects are in focus measured in SOFTIMAGE 3D units.
- **Max COC:** Maximum circle of Confusion (COC). Controls the out-of-focus effect in terms of pixels. The higher the value, the more out of focus the object is. The default value is 20.
- **Max occurs at:** Distance from the camera at which Max COC occurs. This is measured in SOFTIMAGE 3D units.

All the depth of field parameters may be animated.

The Dof effect blurs the points of the image between the near and far planes in proportion to their distance from these planes, according to the **Max COC** parameter, which determines the intensity of the blurring effect. The Dof effect also blurs the alpha channel.

Note

You must deselect the **Depth of Field** option in the Camera Settings dialogue box for the Dof effect to work.

The Dof effect needs to know the depth information for each pixel, so all images must be rendered with the **Render Z Channel** option activated. Dof uses the depth information from the **.Zpic** file to calculate the depth of field blurring.

This example uses the latest version (**-L**) of the **depth** file from frames 1 to 50 with a step of 2 (**-s 1 50 2**):

```
Dof depth -L -s 1 50 2
```

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Creating Special Effects

The **FourthD** effect is a standalone used to modify a sequence of pictures to create the same effects as seen in “Fourth Dimension” by Zbig (pictures twirling around themselves).

The sequence you want to modify must have more frames than the vertical resolution, and the animation must be complete before the end of the sequence. In other words, the **FourthD** effect does not work unless the number of input frames (**OriginalSeq**) equal the number of output frames (**NewSeq**) plus the height in pixels of the input pictures. Also, the movement of the models must stop before the end of the sequence to obtain a correct result in the new sequence.

1. Create a sequence of pictures and render it using the SOFTIMAGE 3D renderer or by converting video sequences to the SOFTIMAGE 3D file format.
2. Open a shell and type:

```
FourthD OriginalSeq NewSeq StartFrm EndFrm  
StepFrm
```

where

OriginalSeq is the name of the sequence to process.

NewSeq is the name of the sequence to create.

StartFrm is the starting frame number (must be frame 1 in this version).

EndFrm is the final frame number of the original sequence.

StepFrm is the step value of the original sequence.

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Beethoven Tutorial

1. Open a shell.
2. Go to the **Rebels** directory and open the **RENDER_PICTURES** directory.

You'll find a 200-frame sequence called **FourthD_Beethoven** consisting of a Beethoven model rotating around the y-axis in 50 frames over a background.

3. Type:

```
FourthD FourthD_Beethoven New 1 200 1
```

FourthD renders the complete new sequence called **New**.

4. Play a SOFTIMAGE 3D flipbook to view the result (choose the **FlipBook** menu cell command in the Tools module or use the **flipbook** standalone).

C H A P T E R E I G H T

Using Peripheral Devices

Introduction

There are a number of peripheral hardware devices to which you can record and move SOFTIMAGE 3D files, and from which you can capture images. These peripheral devices are grouped by type, assuming that you have a main device that you use for all operations.

The peripheral devices discussed in this chapter are the Abekas®, Accom, Accom WSD®, Solitaire, Cindy, Centaur, Galileo, VideoFramer, VideoLab, and Lyon-Lamb MiniVAS.

Using an Abekas

You can record SOFTIMAGE 3D image files to an Abekas recorder, or you can grab images from it to be brought into SOFTIMAGE 3D.

Recording Images

The **Video → Abekas → Record** command allows you to output frames to an Abekas recorder.

1. Choose the **Video → Abekas → Record** command in the Tools module.

The Record to Abekas dialogue box is displayed.

2. Click **Select** to display the browser and locate the desired file to record on the Abekas.
3. Select the **Static** option if you want to only output a single image. To output a sequence of images, click **Sequence** and specify the **Start**, **End**, **Step**, and **Repetition** (number of frames per image).
4. Select the **True Black** option to ensure a true black colour.
5. Select the **Mask** option to record alpha channel values.
6. Enter the name of the Abekas host on which you want to record.
7. Enter the **Starting Location**, which is the recording start frame number on the Abekas. This can use either the **Linear** or **Time Code** option.

Linear tells the Abekas to interpret the recording from its starting location as a sequence of linear frame numbers (typical ranges are from 0 to 1615). **Time Code** tells the Abekas to record, but as a sequence of time code numbers (ranging from 00:00 to 50:00).

8. Click **Ok** to confirm the settings and exit the dialogue box or click **Cancel** to cancel the procedure.

Grabbing Frames

The **Video → Abekas → Grab** command allows you to input frames from an Abekas video recorder.

1. Choose the **Video → Abekas → Grab** command in the Tools module.

The Grab From Abekas dialogue box is displayed.

2. Select the path and file name in which you want to save the grabbed image. If you want to overwrite a file, click **Select** to display the browser and locate the desired file name.
3. Select the **Static** option if you want to only grab a single image. To grab a sequence of images, click **Sequence** and specify the **Start**, **End**, and **Step**.
4. Enter the name of the Abekas host from which you want to grab.
5. Enter the **Starting Location**, which is the start frame number of the sequence on the Abekas. This can use either the **Linear** or **Time Code** option.

Linear interprets the grabbing from its starting location as a sequence of linear frame numbers (typical ranges are from 0 to 1615). **Time Code** interprets the grabbing, but as a sequence of time code numbers (ranging from 00:00 to 50:00).

6. Click **Ok** to confirm the settings and exit the dialogue box.

Using an Accom

You can record SOFTIMAGE 3D image files to an Accom or Accom WSD recorder, or you can grab images from them to be brought into SOFTIMAGE 3D.

Recording Images

The **Video → Accom → Record** command allows you to output frames to an Accom recorder.

1. Choose the **Video → Accom → Record** command in the Tools module.

The Record To Accom dialogue box is displayed.

2. Click **Select** to display the browser and locate the desired file name to record to the Accom.
3. Select the **Static** option if you want to only output a single image. To output a sequence of images, click **Sequence** and specify the **Start**, **End**, **Step**, and **Repetition** (number of frames per image).
4. Select the **True Black** option to ensure a true black colour.
5. Select the **Mask** option to record alpha channel values.
6. Enter the name of the Accom host on which you want to record.
7. Enter the **Starting Location**, which is the recording start frame number on the Accom.
8. Click **Ok** to confirm the settings and exit the dialogue box or click **Cancel** to cancel the procedure.

Recording Images to an Accom WSD

The `wsd_record` standalone records a sequence of images to the Accom WSD.

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
```

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Grabbing

The **Video → Accom → Grab** command allows you to input frames from an Accom video recorder.

1. Choose the **Video → Accom → Grab** command in the Tools module.

The Grab From Accom dialogue box is displayed.

2. Select the path and file name in which you want to save the grabbed image. If you want to overwrite a file, click **Select** to display the browser and locate the desired file name.
3. Select the **Static** option if you want to only grab a single image. To grab a sequence of images, click **Sequence** and specify the **Start**, **End**, and **Step**.
4. Enter the name of the Accom host from which you want to grab.
5. Enter the **Starting Location**, which is the start frame number of the sequence on the Accom.
6. Click **Ok** to confirm the settings and exit the dialogue box.

Grabbing from an Accom WSD

The **wsd_grab** standalone grabs a sequence of images from the Accom WSD and saves it to disk.

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
```

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Recording to a Solitaire

You can record frames of images to the Solitaire film recorder using either the **Film_Out → Solitaire** command in the Tools module or the **filmSolitaire** standalone.

If you have an Indigo2® and your Solitaire is connected to SCSI connector bus 1 and address 4, you should add the **SI_SOLSCSI** variable to your **.softimage** file as follows:

```
setenv SI_SOLSCSI /dev/scsi/sc1d4l0
```

Note that the number one character (1) is used in **sc1d**, but the lower-case “L” character is used in **4l0**.

This will overwrite the default SCSI address which is

```
/dev/scsi/sc0d4l0
```

Using the Film_Out → Solitaire Command

1. Choose the **Film_Out → Solitaire** command in the Tools module.

The Solitaire dialogue box appears.

2. Type the name of the **.pic** file you want to record in the **Name** text box. You can also click on the **Name** button to open a browser from which you can select a file.
3. Select the number of frames to record using the **Do Frames** options (from, To, Step, Rep).
4. Set the **Offset** values if you want the recorded images to be offset.
5. **Advance** or **Rewind** the film a certain number of frames, if you want to start recording at a certain spot on the film.
6. Set the **Resolution** to 2k, 4k, or 8k. The higher the resolution, the smaller the image. You can double the image size by selecting **Pixel replication**, or select **Fit at selected resolution** to scale the image to fit the film output resolution.
7. Select the **SCSI Device** to specify the board you have, and select its mode. GPIB is no longer supported, so it is dimmed.
8. Set other options to zoom, blur, use colour balance, and film format Lookup Tables (LUT), if you need to.
9. Click **Print** to start recording or **Cancel** to exit without saving your changes.

Recording Using the filmSolitaire Standalone

The **filmSolitaire** standalone sends a picture file to the Solitaire 8 and 8XP film recorders.

The following example sends frames of **myFile** to the Solitaire 8.

```
filmSolitaire myFile -q 1 100 1 1 -c -g .75 -t 2  
-i 184 164 159 159 -d
```

The four numbers after **-q** specify to record the first 100 frames without repetition.

The **-c** option centres the image on the screen.

The number after the **-g** option sets the gamma value for the light exposure.

The **-t 2** option specifies that the Solitaire 8 is being controlled through a SCSI driver.

The numbers after **-i** calibrate the colours for the particular film being used.

The **-d** option provides messages while the process is running.

For a complete list of the command line options for this standalone, see its description in the *Standalones* HTML file on the *On-line Documentation* CD.

Using a Cindy

All “civa” standalones are designed to be run with the Cindy board. For a complete list of the command line options for each of these standalones, see their descriptions in the *Standalones* HTML file on the *On-line Documentation* CD.

Recording Images to Video

The **civa_record** standalone records a series of images to a video device (Betacam®, Laser, or Hi8) in any of four output formats.

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
```

Grabbing Images from Video

The **civa_grab** standalone grabs a series of images from a video device (Betacam, Laser, or Hi8) in any of four input formats.

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
```

Capturing a Single Frame

The **civa_capture** standalone captures one frame from the device connected to the Cindy and saves it to disk.

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
```

Moving a Single Frame

The **civa_load** standalone moves a frame from disk to the Cindy buffer. 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
```

Using a Centaur

All “cva” standalones are designed to be run with the Centaur board. For a complete list of the command line options for each of these standalones, see their descriptions in the *Standalones* HTML file on the *On-line Documentation* CD.

Recording Images to Video

The **cva_record** standalone records a series of images to a video device (Betacam, Laser, or Hi8) in any of four output formats.

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
```

Grabbing Images from Video

The **cva_grab** standalone grabs a series of images from a video device (Betacam, Laser, or Hi8) in any of four input formats.

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
```

Capturing a Single Frame

The **cva_capture** standalone captures one frame from the device connected to the Centaur and saves it to disk.

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
```

Moving a Single Frame

The **cva_load** standalone moves a frame from disk to the Centaur buffer. 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
```

Using a Galileo

All “**gv**” standalones are designed to be run with the Galileo board. For a complete list of the command line options for each of these standalones, see their descriptions in the *Standalones* HTML file on the *On-line Documentation* CD.

Recording Images to Video

The **gv_record** standalone records a series of images to a video device (Betacam, Laser, or Hi8) in any of 12 output formats.

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. If external encoding is desired, use the **rgb_*** format identifier.

Grabbing Images from Video

The **gv_grab** standalone grabs a series of images from a video device (Betacam, Laser, or Hi8) in any of 12 input formats.

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
```

To use the **-t** option, you require a special cable from SGI unless you are grabbing an image in one of the **d1** formats.

Capturing a Single Frame

The **gv_capture** standalone captures one frame from the device connected to the Galileo and saves it to disk.

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 a Single Frame

The **gv_load** standalone moves a frame from disk to the Galileo frame buffer in any output format.

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
```

In some cases, you may want to use external encoding, particularly if a high quality external hardware encoding device is available. If external encoding is desired, use the **rgb_*** format identifier.

Using a VideoFramer

All “vfr” standalones are designed to be run with the VideoFramer. For a complete list of the command line options for each of these standalones, see their descriptions in the *Standalones* HTML file on the *On-line Documentation* CD.

Recording Images to Video

The **vfr_record** standalone program records a series of images to a video device (Betacam, Laser, or Hi8) in any of 11 output formats.

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.

```
vfr_record cat 1 500 2 1 0000 rgb_525
```

Grabbing Images from Video

The **vfr_grab** standalone grabs a series of images from a video device (Betacam, Laser, or Hi8) in any of 11 input formats.

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.

```
vfr_grab dog 5 100 1 0020 rgb_625
```

Capturing a Single Frame

The **vfr_capture** standalone captures one frame from the device connected to the VideoFramer and saves it to disk.

This example grabs the current frame on the device and saves it to a file called **frog.pic**. The format is SMPTE for NTSC and the grabbed image is scaled (-s) to be 200 pixels wide.

```
vfr_capture frog -s 200 smpte_525
```

In some cases, you may want to use external encoding, particularly if a high quality external hardware encoding device is available. If external encoding is desired, use the **rgb_*** format identifier.

Moving a Single Frame

The **vfr_load** standalone program moves a frame from disk to the VideoFramer buffer in any of 11 output formats.

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.

```
vfr_load snake smpte_625
```

In some cases, you may want to use external encoding, particularly if a high quality external hardware encoding device is available. If external encoding is desired, use the **rgb_*** format identifier.

Using a VideoLab

All “vli” standalones are designed to be run with the VideoLab board. For a complete list of the command line options for each of these standalones, see their descriptions in the *Standalones* HTML file on the *On-line Documentation* CD.

Recording Images to Video

The **vli_record** standalone records a series of images to a video device (Betacam, Laser, or Hi8) in any of 12 output formats.

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.

```
vli_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. If external encoding is desired, use the **rgb_*** format identifier.

Grabbing from Video

The **vli_grab** standalone grabs a series of images from video (Betacam, Laser, or Hi8) in any of 12 input formats.

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.

```
vli_grab dog 5 100 1 0020 rgb_625
```

Capturing a Single Frame

The **vli_capture** standalone captures one frame from the device connected to the VideoLab and saves it to disk.

This example grabs the current frame on the device and saves it to a file called **frog.pic**. The format is SMPTE for NTSC.

```
vli_capture frog smpte_525
```

Moving a Single Frame

The `vli_load` standalone moves a frame from disk to the VideoLab buffer in any output format.

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 and the image mask (alpha channel **-m**) is displayed.

```
vli_load snake smpte_625 -m
```

In some cases, you may want to use external encoding, particularly if a high quality external hardware encoding device is available. If external encoding is desired, use the **rgb_*** format identifier.

Using a Lyon-Lamb MiniVAS

You can record picture (.pic) or line (.lin) files to a recording device controlled by a Lyon-Lamb MiniVAS using the **record** and **lrecord** standalones, respectively.

For a complete list of the command line options for each of these standalones, see their descriptions in the *Standalones* HTML file on the *On-line Documentation* CD.

Blacking a Video Tape

The **black** standalone initializes the MiniVAS and blacks the tape (that is, writes the time code).

This example displays the help screen (-h) and defines a pre-blackening command (-c).

```
black -h -c s:
```

The default command for the pre-blackening option is “s:”. For a list of available commands, consult the Lyon-Lamb documentation for the MiniVAS.

Recording Picture Image Files to Video

The **record** standalone sends SOFTIMAGE 3D picture files (.pic) to a recording device controlled by a Lyon-Lamb MiniVAS. This program displays a sequence of frames and controls the MiniVAS to record them.

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 frame 20 on the tape on the device.

```
record cat 1 500 2 1 20
```

Recording Line Image Files to Video

The **lrecord** standalone program sends a sequence of line files (.lin) to a recording device controlled by a Lyon-Lamb MiniVAS.

This example records the **cat.lin** file from frames 1 to 500 at a step of 2 with each frame repeated once. The recording starts at frame 20 on the tape on the device.

```
lrecord cat.lin 1 500 2 1 20
```

Index

Symbols

.3ds, see *3D Studio files*
 .a, see *Raw files*
 .a8, see *PictureMaker files*
 .als, see *Alias files*
 .art, see *Artisan files*
 .b, see *Raw files*
 .b8, see *PictureMaker files*
 .eps, see *Encapsulated PostScript files*
 .g, see *Raw files*
 .g8, see *PictureMaker files*
 .hrc, see *Hierarchy files*
 .icn files 17
 .ima, see *Everest files*
 .pic, see *Vertigo files*
 .pict, see *PICT files*
 .plt, see *HPGL files*
 .r, see *Matisse files, Raw files*
 .r8, see *PictureMaker files*
 .rgb, .rgba, see *RGB files*
 .rla, .rlb, see *Wavefront files*
 .sgi, see *SGI files*
 .sim, see *Aurora files*
 .tga, see *Targa files*
 .tif, see *TIFF files*
 .yuv, see *Wavefront files*
 .yuv, see *YUV files*

Numerics

3D Studio files, converting to SOFTIMAGE 3D 42, 47

A

Abekas recorder
 grabbing frames from 104
 recording images to 104
 Accom recorder
 grabbing frames from 107
 recording images to 106
 Accom WSD

 grabbing frames from 107
 recording files 106
 Alias files
 converting
 from SOFTIMAGE 3D 55
 to SOFTIMAGE 3D 47
 alias2soft standalone 47
 Alpha channel
 creating or modifying 84
 extracting 85
 alpha standalone 84
 Animation sequences, viewing 18, 19
 art2soft standalone 48
 Artisan files
 converting
 from SOFTIMAGE 3D 55
 to SOFTIMAGE 3D 48
 ASCII data files, converting to binary 42
 Aurora files
 converting
 from SOFTIMAGE 3D 55
 to SOFTIMAGE 3D 48
 aurora2soft standalone 48

B

bin2ascii standalone 42
 Binary data files, converting to ASCII 42
 black standalone 118
 Blacking video tape 118
 Blurring picture files 97

C

Camera, depth of field blur 97
 Capturing
 frames with Centaur 111
 frames with Cindy 110
 frames with Galileo 112
 frames with VideoFramer 114
 frames with VideoLab 116
 Centaur
 capturing frames with 111
 grabbing images from video 111
 moving frames to 111

- recording images to video 111
- Channels
 - creating or modifying alpha 84
 - extracting 85
- Cindy
 - capturing frames with 110
 - grabbing images from video 110
 - moving frames to 110
 - recording images to video 110
- civa_capture standalone 110
- civa_grab standalone 110
- civa_load standalone 110
- civa_record standalone 110
- Colour images, converting to grey-scale 43
- Colour-mapped files, converting from SOFTIMAGE 3D 55
- Colours
 - adding noise 96
 - changing hue, lightness, saturation 88
 - creating gradation 86
 - extracting channels 85
 - high contrast 87
 - intensity values 88
 - inverting positive and negative 95
- Combining
 - .lin and .pic files 15
 - images 73
 - picture files 79
- Command line utilities, see Standalones 4
- Comparing picture files 26
- composite standalone 75
- Compositing
 - any file 40, 77
 - picture files 73, 75
- Compressing a file 40
- contrast standalone 87
- Converting
 - .lin to .pic files 14, 16
 - 3D Studio files to SOFTIMAGE 3D 47
 - Alias files to SOFTIMAGE 3D 47
 - any file format 40
 - Artisan files to SOFTIMAGE 3D 48
 - Aurora files to SOFTIMAGE 3D 48
 - between binary and ASCII 42
 - between SOFTIMAGE 3D versions 42
 - colour images to grey-scale 43
 - Everest files to SOFTIMAGE 3D 49
 - hierarchy files 42
 - imported objects 52
 - line files
 - to .eps 44
 - to .pic 44
 - to .pict 56
 - to HPGL 45
 - Matisse files to SOFTIMAGE 3D 49
 - PICT files to SOFTIMAGE 3D 49
 - PictureMaker files to SOFTIMAGE 3D 49
 - PostScript files to SOFTIMAGE 3D 48
 - raw files to SOFTIMAGE 3D 50
 - RGB files to SOFTIMAGE 3D 50
 - Saturn files to SOFTIMAGE 3D 53
 - SGI files to SOFTIMAGE 3D 51
 - SOFTIMAGE 3D
 - to 3D Studio 42
 - to Alias files 55
 - to Artisan files 55
 - to Aurora files 55
 - to colour-mapped files 55
 - to Everest files 56
 - to Matisse files 56
 - to PICT files 56
 - to PictureMaker files 57
 - to raw files 57
 - to RGB files 57
 - to Saturn 67
 - to SGI files 58
 - to Targa files 58
 - to TIFF files 58
 - to Vertigo files 58
 - to VRML 68
 - to Wavefront files 59
 - to YUV files 59
 - Targa files to SOFTIMAGE 3D 51
 - TIFF files to SOFTIMAGE 3D 51
 - Vertigo files to SOFTIMAGE 3D 51

Wavefront files to SOFTIMAGE 3D 52
YUV files to SOFTIMAGE 3D 52
convpic standalone 42
Creating
 alpha channel 84
 colour gradations 86
 high-contrast images 87
 PostScript files 31
 during playback 31
 with filled faces 32
 special effects 98
cva_capture standalone 111
cva_grab standalone 111
cva_load standalone 111
cva_record standalone 111

D

Data files, converting binary and ASCII 42
Depth of field blur 97
Devices, see *Peripheral devices*
diffpic standalone 26
display standalone 11
Displaying, see *Viewing*
Dof standalone 97
dxf2soft standalone 52

E

effect standalone 91
enlarge standalone 90
Enlarging picture files 90
evalScene standalone 27
Evaluating a scene 27
Everest files
 converting
 from SOFTIMAGE 3D 56
 to SOFTIMAGE 3D 49
everest2soft standalone 49
Exporting
 file formats 54
 frames to film 108, 109
 image files 54
 objects 66
extract standalone 85
Extracting channels 85

F

Fields, interleaving 79
Files
 combining .lin and .pic 15
 compositing 40, 77
 compressing 40
 converting 40
 export formats 54
 export object formats 66
 import formats 46
 import object formats 52
 PostScript 31, 32, 33
 printing 33
 viewing 40
 viewing information 24
Filled faces, PostScript files 32
Film, output frames 108, 109
filmSolitaire standalone 109
flipbook standalone 19
Flipbooks 18, 19
 memory guidelines 20
Flipping picture files 91
FourthD standalone 98
Frames
 capturing
 with Centaur 111
 with Cindy 110
 with Galileo 112
 with VideoFramer 114
 with VideoLab 116
 grabbing
 from Abekas 104
 from Accom 107
 moving
 to Centaur 111
 to Cindy 110
 to Galileo 113
 to VideoFramer 115
 to VideoLab 117
 output
 to Abekas 104
 to Accom 106
 to film 108, 109

G

Galileo

- capturing frames with 112
- grabbing images from video 112
- moving frames to 113
- recording images to video 112

Grabbing

frames

- from Abekas 104
- from Accom 107
- from Accom WSD 107

images from video 110, 111, 112, 114, 116

gradation standalone 86

gray_scale standalone 43

gv_capture standalone 112

gv_grab standalone 112

gv_load standalone 113

gv_record standalone 112

H

Hierarchy files, converting versions 42

High-contrast picture files 87

hls standalone 88

howmany standalone 23

HPGL files, creating from line files 45

hrcConvert standalone 42

Hue, changing 88

I

Icon files 17

iges2soft 52

Image files

- exporting 54
- importing 46

imgconv standalone 40

imginfo standalone 24

imgshow standalone 10

Importing

- frames from Abekas 104
- frames from Accom 107
- image files 46

objects 52

infolin standalone 23

infopic standalone 24

Information

- any file format 24
- line files 25
- picture files 24

intensities standalone 88

Intensity

- adding noise 96
- colours 88

interleave standalone 79

L

Laser printer, printing files 34

Layers, compositing 73

Lightness, changing 88

lin2eps standalone 44

lin2hpgl standalone 45

lin2pic standalone 44

lin2pict standalone 56

Line files

- combining with .pic 15
- converting
 - to .eps 44
 - to .pic 14, 44
 - to .pict 56
 - to HPGL 45

number of vectors 23

printing 34, 35

recording to video 118

viewing 10, 14, 16

viewing information 25

Lines in a picture file 23

lrecord standalone 118

Lyon-Lamb MiniVAS

- blacking video tape 118
- recording images to video 118

M

Matisse files

- converting
 - from SOFTIMAGE 3D 56

to SOFTIMAGE 3D 49

Merging picture files 79

Modifying

alpha channel 84

colour intensity values 88

colour values 88

sequence length 93

Moving frames

to Centaur 111

to Cindy 110

to Galileo 113

to VideoFramer 115

to VideoLab 117

N

negate standalone 95

Network printers 35

noise standalone 96

Number of lines in picture files 23

Number of vectors in line files 23

O

Objects

exporting 66

importing 52

Opening a shell 4

P

Peripheral devices

Abekas recorder 104

Accom recorder 106

Accom WSD recorder 106

Centaur board 111

Cindy board 110

Galileo board 112

Lyon-Lamb MiniVAS 118

Solitaire film recorder 108

VideoFramer 114

VideoLab board 116

PICT files

converting

from line files 56

from SOFTIMAGE 3D 56

to SOFTIMAGE 3D 49

pict2soft standalone 49

Picture files

adding noise 96

colour intensity 88

comparing 26

compositing 73, 75

converting, see *Converting SOFTIMAGE 3D*

creating PostScript files 31

depth of field blur 97

enlarging 90

exporting 54

flipping and rotating 91

grabbing from video 110, 111, 112, 114, 116

high-contrast 87

icons 17

importing 46

inverting negative and positive 95

merging fields 79

modifying colour values 88

modifying sequence length 93

number of lines 23

printing 34

recording

to Abekas 104

to Accom 106

to film 108, 109

to video 110, 111, 112, 114, 116, 118

reducing 91

rotating 89, 92

scaling 89, 90

special effects 98

viewing 10, 11

a flipbook 18, 19

information 24

PictureMaker files

converting

from SOFTIMAGE 3D 57

to SOFTIMAGE 3D 49

pm2soft standalone 49

PostScript files

- converting to SOFTIMAGE 3D 48
- creating 31
 - during playback 31
 - with filled faces 32
- setting up for printing 33
- PostScript printer 34, 35
- Printing
 - laser printer 34
 - line files 34, 35
 - network printers 35
 - PostScript files 34
 - PostScript printer 34, 35
 - setting up 33
- printlin standalone 35
- printps standalone 34, 35

R

- r2soft standalone 49
- Raw files
 - converting
 - from SOFTIMAGE 3D 57
 - to SOFTIMAGE 3D 50
- raw2soft standalone 50
- record standalone 118
- Recording
 - images
 - to Abekas 104
 - to Accom 106
 - to video 110, 111, 112, 114, 116, 118
 - picture files
 - to Accom WSD 106
 - to film 108, 109
- reduce standalone 91
- Reducing picture files 90, 91
- RGB files
 - converting
 - from SOFTIMAGE 3D 57
 - to SOFTIMAGE 3D 50
- rgb2soft standalone 50
- RGBA files, *see* RGB files
- rotate standalone 92
- Rotating picture files 89, 91, 92

S

- Saturation, changing 88
- Saturn files
 - converting from SOFTIMAGE 3D 67
 - converting to SOFTIMAGE 3D 53
- scale standalone 90
- Scaling picture files 89, 90, 91
- Scan lines, interleaving 79
- Scenes, evaluating 27
- sequence standalone 13
- Setting up for printing 33
- SGI files
 - converting
 - from SOFTIMAGE 3D 58
 - to SOFTIMAGE 3D 51
- sgi2soft standalone 51
- Shell, opening 4
- showicn standalone 17
- Showing, *see* Viewing
- showlin standalone 16, 44
- showpic standalone 12
- soft2alias standalone 55
- soft2art standalone 55
- soft2aurora standalone 55
- soft2dxf standalone 66
- soft2everest standalone 56
- soft2iges standalone 66
- soft2map standalone 55
- soft2pict standalone 56
- soft2pm standalone 57
- soft2r standalone 56
- soft2raw standalone 57
- soft2rgb standalone 57
- soft2sgi standalone 58
- soft2tga standalone 58
- soft2tiff standalone 58
- soft2vertigo standalone 58
- soft2wave standalone 59
- soft2yuv standalone 59
- Solitaire film recorder 108, 109
- Special effects 98
- Standalones

syntax conventions 4
 using 4
Stereo images, viewing 12
Syntax conventions, standalones 4

T

Targa files
 converting
 from SOFTIMAGE 3D 58
 to SOFTIMAGE 3D 51
tga2soft standalone 51
TIFF files
 converting
 from SOFTIMAGE 3D 58
 to SOFTIMAGE 3D 51
tiff2soft standalone 51
Timewarp standalone 93
Twirling images 98

V

Vectors in a line file 23
Vertigo files
 converting
 from SOFTIMAGE 3D 58
 to SOFTIMAGE 3D 51
vertigo2soft standalone 51
vfr_capture standalone 114
vfr_grab standalone 114
vfr_load standalone 115
vfr_record standalone 114
Video tape, blacking 118
VideoFramer
 capturing frames with 114
 grabbing images from video 114
 moving frames to 115
 recording images to video 114
VideoLab
 capturing frames with 116
 grabbing images from video 116
 moving frames to 117
 recording images to video 116
Viewing
 an animation sequence 18, 19

any file 40
any file format 10, 40
colour-mapped image files 17
information
 any file format 24
 line files 25
 picture files 24
line files 14, 16, 44
picture files 11, 12, 13
stereo images 12
vli_capture standalone 116
vli_grab standalone 116
vli_load standalone 117
vli_record standalone 116
VRML format 68

W

Watch standalone 40, 77
wave2soft standalone 52
Wavefront files
 converting
 from SOFTIMAGE 3D 59
 to SOFTIMAGE 3D 52
WSD, see *Accom WSD*
wsd_grab standalone 107
wsd_record standalone 106

Y

YUV files
 converting
 from SOFTIMAGE 3D 59
 to SOFTIMAGE 3D 52
yuv2soft standalone 52

Z

zoom standalone 90

