



Command Line Utilities

Command Line Utilities
Copyright and trademarks

StudioTools 13

Software copyright information is located in the application, and can be accessed from the menu by choosing Help > About StudioTools.

All documentation ("Documentation") is copyrighted © 2001-2005 Alias and contains proprietary and confidential information of Alias. The Documentation is protected by national and international intellectual property laws and treaties. All rights reserved. Use of the Documentation is subject to the terms of the license agreement that governs the use of the software product to which the Documentation pertains ("Software"). The authorized licensee of the Software is hereby authorized to print no more than one (1) hardcopy of any Documentation provided in digital format per valid license of the Software held by such licensee. Except for the foregoing, the Documentation may not be translated, copied or duplicated in any form (physically or electronically), in whole or in part, without the prior written consent of Alias.

Alias and the swirl logo, Maya and DesignStudio are registered trademarks and Alias Natural Phenomena, Alias OpenAlias, Alias OpenModel, Alias PowerCaster, Alias PowerTracer, Alias RayCasting, Alias RayTracing, Alias SDL, ImageStudio, Alias Spider, StudioPaint, StudioViewer, StudioTools and SurfaceStudio are trademarks of Alias Systems Corp. ("Alias") in the United States and/or other countries. Silicon Graphics, SGI and IRIX are registered trademarks and Inventor is a trademark of Silicon Graphic, Inc. in the United States and/or other countries worldwide. Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Renderman is a registered trademark of Pixar Corporation. Apple, Quicktime and Macintosh are trademarks of Apple Computer, Inc. registered in the United States and other countries. Adobe, Postscript and Illustrator are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. Unigraphics, NX, and I-deas are registered trademarks or trademarks of UGS Corp. or its subsidiaries in the United States and in other countries. Arius3D is a registered trademark of Arius3D Inc. Cyberware is a registered trademark of Cyberware Laboratory Inc.. Cyrax is a registered trademark of Leica Geosystems HDS Inc. Steinbichler is a registered trademark of Steinbichler Optotechnik GmbH. Autodesk and AutoCAD are either registered trademarks or trademarks of Autodesk, Inc./Autodesk Canada, Inc. in the USA and/or other countries. CATIA is a registered trademark of Dassault Systèmes S.A. PTC, Pro/ENGINEER and Granite are trademarks or registered trademarks of Parametric Technology Corporation or its subsidiaries in the U.S. and in other countries. All other trademarks mentioned herein are the property of their respective owners.

All PTC Technology logos are used under license from Parametric Technology Corporation, Needham, MA, USA.

Not all features described are available in all products.



Alias Systems Corp., 210 King Street East, Toronto, Canada M5A 1J7

Contents

Command line utilities 1

Platforms 2

Background utilities 5

Command reference 9

- AliasBatch 10
- AliasToRenderman 12
- AlToCa 20
- AlToC5 21
- AlToIv 24
- AlToObj 27
- AlToSt 28
- AlToUG, AlToUG19, AlToUG20, AlToUG21 31
- beep 32
- bsd1 33
- CaToAl 34
- C5ToAl 35
- dxf_info 38
- FieldAssembler 40
- findit 42
- flipbook 43
- fmovie 45
- from100to97 47
- fstats 50
- get_alias_variable 52
- getid 54
- hp_glplotf, hp_gl2plotf 57
- iges_info 59
- imgcvt 61
- IvToAl 66
- lmgrd 67
- lmutil 69
- makebot 70

nprocs 71
ObjToAI 72
pim 73
pixdiff 77
print_wire_header 79
psplotf 81
pst, pstget 82
rb_stereo 90
renderer/raytracer/powercaster/powertracer 92
renderit 98
setupacct 100
showstereo 102
StToAI 103
systemInfo 104
UGToAI, UGToAI19, UGToAI20, UGToAI21 105
vda_info 106
wrl 107

Index 111

Command line utilities

Describes various command line programs included with StudioTools.

Platforms

Lists which platforms each utility is available on.

The functions are available in the following operating environments.

Function	Platforms	Purpose
AliasBatch	IRIX, Windows	Batch File Translator does unattended translation of data files. This replaces Alias-B .
AliasToRenderman	IRIX	Translates Alias wire files to Pixar RIB files
AlToCa	IRIX, Windows	Translates Alias wire files to CAI file format.
AlToC5	IRIX, Windows	Translates Alias wire files to CATIA V5 file format.
AlToIv	IRIX, Windows	Translates Alias wire files to Inventor file format.
AlToObj	IRIX, Windows	Translate Alias wire files to OBJ file format.
AlToSt	IRIX, Windows	Translates Alias wire files to ISO10303 (STEP) format.
AlToUG , AlToUG19 , AlToUG20 , AltoUG21	IRIX, Windows	Translates Alias wire files to Unigraphics 19, 20 or 21 format.
beep	IRIX	Rings the console bell.
bsdl	IRIX, Windows	Converts a binary SDL file into ASCII text
CaToAl	IRIX, Windows	Translates CAI files to Alias wire files.
C5ToAl	IRIX, Windows	Translates CATIA V5 files to Alias wire files.
dxf_info	IRIX	Provides information on DXF entities.
EvalViewer	IRIX	Works as both a cloud data tool and a surface evaluation tool. See the Cloud Tool Overview in the EvalViewer online documentation.
FieldAssembler	Windows	Interlaces scan-lines of images rendered on fields into a single image; also pastes together incompletely rendered images. See the online documentation that comes with FieldAssembler.
findit	IRIX	Checks whether a specified picture file exists on the render engine. It is also used internally by StudioTools.

Function	Platforms	Purpose
flipbook	IRIX	Is an Animation preview utility with advanced playback and compression options.
fmovie	IRIX	Converts a sequence of Alias image files into an SGI or Apple QuickTime movie.
from100to97	IRIX, Windows	Converts V10 and later wire files to V9.7
fstats	IRIX	Provides information on the number of completed scan-lines in an Alias pix file.
get_alias_variable	IRIX	Determines the value of an Alias preferences variable.
getid	IRIX	Displays StudioTools system information.
hp_glplotf, hp_gl2plotf	IRIX, Windows	Translates Alias plot files into HP-GL (IBM-GL) and HP-GL2 formats.
iges_info	IRIX	Provides information on IGES entities.
imgcvt	IRIX, Windows	Converts between a wide variety of image file formats.
IvToAl	IRIX, Windows	Translates Inventor files to Alias wire files.
lmgrd	IRIX, Windows	Looks for the license file, which contains the information about vendors and features.
lmutil	IRIX, Windows	Is a licence management program from Globetrotter Software.
makebot	IRIX	Creates block ordered texture (BOT) file.
nprocs	IRIX	Displays information about the host machine's processor(s).
ObjToAl	IRIX, Windows	Converts OBJ files to Alias wire file format.
pim	IRIX	"plug-in manager" manages plug-in utilities.
pixdiff	IRIX	Compares two pix files.
powercaster	IRIX, Windows	Is a purchasable command line parallel renderer for StudioTools SDL files. See the section renderer/raytracer/powercaster/powertracer .
powertracer	IRIX, Windows	Is a purchasable command line parallel raytracer for StudioTools SDL files. See the section renderer/raytracer/powercaster/powertracer .
print_wire_header	IRIX, Windows	Views the headers of wire files.
psplotf	IRIX, Windows	Translates Alias plot files into PostScript.

Function	Platforms	Purpose
<code>pst</code> , <code>pstget</code>	IRIX	<code>pst</code> creates a simple editor windows of the type used by StudioTools, and <code>pstget</code> gets information for <code>pst</code> .
<code>raytracer</code>	IRIX, Windows	Is a command line raytracer for StudioTools SDL files. See the section <i>renderer/raytracer/powercaster/powertracer</i> .
<code>rb_stereo</code>	IRIX	Combines stereo pairs of images for viewing with red & blue eye-glasses.
<code>renderer</code>	IRIX, Windows	Is a command line renderer for StudioTools SDL files. See the section <i>renderer/raytracer/powercaster/powertracer</i> .
<code>renderit</code>	IRIX	Is a shell script to execute the rendering code. It copies files to the proper locations and, if necessary, starts the <code>renderit</code> script on a remote machine.
<code>setupacct</code>	IRIX	Customizes a user account for use with StudioTools.
<code>showstereo</code>	IRIX	Loads images onto Stereographics monitor.
<code>StToAl</code>	IRIX, Windows	Converts STEP format files to Alias wire files.
<code>systemInfo</code>	IRIX	Displays StudioTools system information in a window. For more information, see the description of <i>getid</i> .
<code>UGToAl</code> , <code>UGToAl19</code> , <code>UGToAl20</code> , <code>UGToAl21</code>	IRIX, Windows	Converts Unigraphics 19, 20, or 21 files to Alias wire file format.
<code>vda_info</code>	IRIX	Provides information about VDA entities.
<code>wrl</code>	IRIX	Displays Alias pix files.

Background utilities

The following utilities are called by other programs, are used in the background, or are documented elsewhere.

Function	Platforms	Purpose
Alias	IRIX	Sets all required environment variables and runs the interactive package in the same directory as this file
.Alias	IRIX	Is the executable for the Studio interactive package
Alias.exe	Windows	Is the executable for the Studio interactive package
AliasWarn	Windows	a background utility to display error messages
AliasWarning	IRIX	a background utility to display error messages
appLaunch	IRIX, Windows	Enables the user to select which Studio product to run. It is launched on start-up.
lmutil	IRIX, Windows	component of the Licence Manager
lmgrd	IRIX, Windows	component of the Licence Manager. See the description of lmutil
lmtools	IRIX, Windows	component of the Licence Manager
onlineDocs	IRIX, Windows	a background utility that opens StudioTools context-sensitive documentation in an HTML browser
RunComposer	IRIX	starts the Record Only Composer program and sets the environment variables.
sgiawd	IRIX, Windows	component of the Licence Manager
splash	IRIX, Windows	displays the initial product-specific screen image when the product is started
toimg	Windows	works with imgcvt during conversion of image files

Using command line utilities on Windows

1 Choose **Programs > Accessories > Command Prompt** from the Start menu.

2 Click the cursor in the window.

You can see what directory you are in by typing

```
dir
```

A list of the directory contents will be provided; at the top of the list is your current location.

3 Navigate to the directory containing the files to be converted. If you have installed StudioTools in its default location, the path will be similar to

```
C:\Documents and Settings\[UserID]\My  
Documents\StudioTools\user_data\[Project]
```

where [UserID] is replaced by your login userid (and no square brackets) and [Project] is replaced by the name you have assigned to the project: the default project is demo (again, no square brackets). So to navigate to the directory, on a single line, type

```
cd C:\Documents and Settings\[UserID]\My  
Documents\StudioTools\user_data\[Project]\wire
```

4 To use a stand-alone utility to convert a file named “headlight.[fileextension]” to a StudioTools wirefile named “headlight.wire, choose the translator and type

```
[translator] headlight.[fileextension] [flags]  
headlight.wire
```

where [translator] is the name of the stand-alone utility being used, [fileextension] is the three-letter extension after the dot, and [flags] are the stand-alone utility’s options.

Using command line utilities on IRIX

1 Open a UNIX shell through the Toolchest or select UNIX shell from the Studio Utilities menu.

2 Click the cursor in the window.

You can see what directory you are in by typing

```
pwd
```

3 Navigate to the directory containing the files to be converted. If you have installed StudioTools in its default location, the path will be similar to

```
/usr/u/[UserID]/Alias/user_data/[Project]/wire
```

where [UserID] is replaced by your login userid (and no square brackets) and [Project] is replaced by the name you have assigned to the project: the default project is demo (again, no square brackets). So to navigate to the directory, type

```
cd /usr/u/[UserID]/Alias/user_data/[Project]/wire
```

- 4 To use a stand-alone utility to convert a file named “headlight.[fileextension]” to a StudioTools wirefile named “headlight.wire, choose the translator and type

```
[translator] headlight.[fileextension] [flags]  
headlight.wire
```

where [translator] is the name of the stand-alone utility being used, [fileextension] is the three-letter extension after the dot, and [flags] are the stand-alone utility’s options.

Command reference

AliasBatch

Platforms

IRIX, Windows

Description

The Alias Batch File Translator is a command-line invocation of StudioTools that allows unattended data file translation. You can use this interface to StudioTools in a UNIX shell script to run multiple file translations unattended, perhaps in an overnight batch job. All file formats supported by StudioTools retrieve/save features are supported, with the exception of Filter.

The standalone utility Alias -B has been replaced by the application AliasBatch, which uses the OpenModel API. You will find that AliasBatch is more robust. For help, enter **AliasBatch -h**.

Note: Alias -B is no longer supported.

Command Line Options

AliasBatch [-h] [-w -sdl -r1 -r2] [-o...] -s RetFile...

AliasBatch	Launches the Alias Batch Processor, reads in the RetFile, translates it according to settings in an options file, writes it out as SavFile, and exits.
-h	Displays help information.
-o[OptFile]	Loads the options file OptFile. (Its full path is required.)
-s[SavFile]	Writes output to SavFile. This option is compulsory.
-w	Overwrites the SavFile if it exists.
-sdl	Sets the format to SDL; overrides the save format in the options file.
-r1	Optimizes redundant pre-V5.0 trimmed surfaces by converting them to surfaces.

-r2	Optimizes redundant pre-V5.0 trimmed surfaces by converting to face geometry.
RetFile	Retrieves files at start up (wire, IGES, etc.). If more than one file is retrieved, the SavFile is a merged stage or stage set.

AliasToRenderman

Platforms

IRIX

Purpose

AliasToRenderman converts Alias wire files to Renderman Interchange Bytestream (RIB) format files.

Description

AliasToRenderman [-a #] [-A #] [-b #] [-B #] [-c #] [-C <channels>] [-e #] [-E #] [-f #] [-F <format>] [-h#] [-H] [-I <include_file>] [-j #] [-L <light_file>] [-m #] [-N <extract_name>] [-o <rib_file>] [-O <object_file>] [-p <image_file>] [-r #] [-s #] [-S <shader_file>] [-T #] [-v #] [-w #] [-x #] <alias_wire_file>

-a #	specifies # as the supersampling rate. The supersampling rate is also known as the AAlevel or PixelSamples. The value # is treated as an integer. There is no constraint on the value of this field nor is any checking performed. The default value is 2.
-A #	specifies # as the shutter angle. This option is ignored if there is no animation or if motion blur is turned off. When animation exists and motion blur is being used, this option specifies the duration of the shutter opening. # is specified in degrees by convention. Zero degrees means zero duration, 360 degrees means one full frame, 180 degrees means one half of a frame time, and so on. The shutter is considered to open at current_frame time and to close at current_frame+(shutter_angle/360). A value of zero forces motion blur to be off. The value # is treated as a floating point number. There are no limits on the value #; the default is 180.
-b #	specifies # as the increment in frame time between frames of an animation. This is ignored if the model does not contain any animation. The value # is treated as a floating point number, and must be greater than zero. A value less than or equal to zero is ignored and the default value is used. The default value is 1. See also option -o for a discussion of inter-related options.

-B #	<p>specifies # as the increment to the file name extensions when outputting an animation. This option is ignored if the model does not contain any animation. The value # is treated as an integer, and must be greater than zero. A value less than or equal to zero is ignored and the default is used. The default value is the smallest integer greater than or equal to by_frame.</p> <p>See also option -o for a discussion of inter-related options.</p>
-c #	<p>specifies the form of the output. If # is zero, all RIB output is in compacted (binary) form. If # is any other value, RIB output is in ASCII text form. The default is to output in ASCII format.</p>
-C <channels>	<p>controls the number and type of image channels to be output by the renderer. <channels> specifies the string to pass directly to the RIB output via the <i>mode</i> parameter to the <i>Display</i> command. There is no constraint on the value of this option. The default value is "rgba".</p>
-e #	<p>specifies # as the ending frame time of the animation. # does not need to relate to the actual animation range of the model at all. This option is ignored if the model does not contain animation. The value # is treated as a floating point number and must be greater than or equal to the start frame. If it is not, the translator automatically swaps them. The default value is 1.</p> <p>See also option -o for a discussion of inter-related options.</p>
-E #	<p>specifies # as the minimum field size for the file name extension when generating multiple files for an animation. This option is ignored if the model does not contain any animation. The actual extension size used is the larger of the value given and the actual size of start_extension. The actual size of start_extension may in turn derive from its default. The value # is treated as an integer. The default is the actual size of start_extension.</p> <p>See also option -o for a discussion of inter-related options.</p>
-f #	<p>specifies how faces are output. If # is zero, all Alias faces are output as trimmed NURBS. If # is any other value, faces are output as polygons. The default is to output faces as polygons.</p>
-F <format>	<p>controls the destination of the rendered output where <format> is a string which is passed directly to the RIB output via the mode parameter to the Display command. There is no constraint on the value of this field nor is any checking performed. The default <format> is file.</p>
-h #	<p>specifies # as the height of the rendered image. The value # is treated as an integer and must be greater than zero. Invalid values are ignored and the default value is used. The default value is 486 (NTSC).</p>
-H	<p>prints usage information on standard output and exits the translator. This is the only option which does not require an argument.</p>

-I <include_file>	copies the file to the RIB output file. <include_file> is a string that is interpreted as a UNIX filename. If you are animating the file, it is copied to each of the output files generated. Any I/O problems are reported as errors. The default is that no file is included.
-j #	controls jittering. If # is 0, jittering is not used for the rendering. If # is any other value, jittering is used for the rendering. The amount of jittering depends on the renderer implementation. The default is 0.
-L <light_file>	specifies a light extraction file. <light_file> is a string that is interpreted as a UNIX filename. Which lights are output is controlled by the -N option to be either all lights or only the light named by -N. This output is in addition to any other specified output files. The actual filename used has an extension appended in the same manner as other output files. Only light definitions and messages are output to this file. No other information is output to this file. The default is that no light file is created. See the -o option for further information.
-m #	specifies the state of motion blur. If # is 0, motion blur is turned off in the output. If # is any other value, motion blur is turned on. The default is that motion blur is on.
-N <extract_name>	specifies the name of an item in the model as the string <extract_name>. If <extract_name> is specified, only the item named is output to an extract file. Since items in a StudioTools model have unique names, only one item can possibly match no matter how many extract files are specified. If no extract files are specified, the extract name is ignored. If no extract name is specified, then each extract file contains all items of the appropriate type. The default behavior is that no extract name is specified. See the -L, -O, and -S options for further information.
-o <rib_file>	specifies that the entire StudioTools model is to be translated and output to <rib_file>, where <rib_file> is a string that is interpreted as a filename. The <rib_file> output is in addition to any output that may be generated by extract files. For further information see -L, -O and -S.
	If the model contains animation, the translator is animating. When animating, each frame to be output generates a separate file using a filename consisting of the given filename with an extension. The extension is a positive integer. The number of output frames is controlled by the values of start_frame (-s), by_frame (-b), and end_frame (-e). The sense of frame time is first set to the start_frame value and one file is generated. Frame time then increments by the by_frame value and tests against the end_frame value. If the current frame time is less than or equal to the end_frame value, another file is output. This continues until the test of frame time fails.

Each output file generated has an extension. The first file has an extension specified as `start_extension` (-T). Subsequent files have the extension number incremented by the `by_extension` value (-B). All extension numbers are increased by high order zeros to a length specified by `extension_size` characters (-E). The values of the extensions do not need to be related to the sense of frame time in any way except that they are constrained to increase over the course of the animation. All extracted files, lights (-L), objects (-O), and shaders (-S), also have extensions appended in the same manner.

`Start_frame`, `by_frame`, and `end_frame` all default to 1. `Start_extension` and `by_extension` default to the actual values for `start_frame` and `by_frame` respectively. None of these values reflect the actual animation range of the model unless you specify it. Information messages are generated stating both the range occurring in the model and the range you specified.

If the model contains no animation, the translator does not animate. When not animating, only one file of each type discussed above (-L, -O, -o, and -S) is created. The file generated does not have an extension. The values of `start_frame`, `by_frame`, `end_frame`, `start_extension`, `by_extension`, and `motion_blur` are all ignored.

The value of `<rib_file>` is not verified. The value of `<rib_file>` is used to construct the default value for the rendered image. For more information see -p.
`<rib_file>` has no default value of its own. The default behavior is to not translate the model at all.

-O `<object_file>` specifies the file where objects in the model are to be stored. `<object_file>` is a string that is interpreted as a UNIX filename. Which objects are output is controlled by the -N option to be either all objects or only the object named by -N. This output is in addition to any going to other output files. The actual filename used has an extension in the same manner as other output files. See the -o option for further information. Only object definitions and messages are output to this file. No other information is output to this file. The default is that no object file is created.

-p `<image_file>` passes the name of the rendered output to the RIB file using the *Display* command. `<image_file>` is the name of the rendered output. If the rendered output is an animation, the name specified has an extension as if it were a filename. The name given is not checked for validity in any way. The default name is `<rib_file>.tiff`. For further information see the -o option.

-r #	specifies # as the pixel aspect ratio of the rendering. The pixel aspect ratio of the rendering is passed on to the output via the <i>Format</i> command. The value # is treated as a floating point number. The default value is 1.
-s #	specifies # as the starting frame time of the animation. The starting frame time does not need to relate to the actual animation range of the model, except that it is ignored if the model does not contain any animation. The value must be less than or equal to the end frame. If it is not, the translator automatically swaps the start and end frame times. See also option -o for a discussion of inter-related options. The value # is treated as a floating point number. The default is 1.
-S <shader_file>	outputs the shaders in the model to the specified file where <shader_file> is a string that is interpreted as a UNIX filename. Which shaders are output is controlled by the -N option to be either all shaders or only the shader named by -N. This output is in addition to any other output files. The actual filename used has an extension in the same manner as other output files. See the -o option for further information. Only shader definitions and messages are output to this file. No other information is output to this file. The default is that no shader file is created.
-T #	specifies # as the starting number for the file name extensions when outputting an animation. This option is ignored if the model does not contain animation. The value # is treated as an integer, and must be greater than or equal to zero. A value less than zero is ignored and the default is used. See also option -o for a discussion of inter-related options. The default value is the smallest integer greater than or equal to start_frame.
-v #	specifies the type of messages to output. If # is zero, only error messages are output. If # is any other value, information messages are also output. Error messages are output to all output files as comments as well as to <i>stderr</i> at all times. Information messages are output to all output files and to <i>stderr</i> , only if this option is on. The default is to output all messages.
-w #	specifies # as the width of the rendered image. The value # is treated as an integer and must be greater than zero. An invalid value is ignored and the default is used. The default value is 645 (NTSC).
-x #	specifies whether or not header information will be displayed. If # is zero, all RIB header information (Format, Display, Hider, and PixelSamples) are output. If # is any other value, the header information is suppressed. This is useful if the header is supplied by an <i>include</i> file. The default is to output header information.

`-o <rib_file>` specifies that the entire StudioTools model is to be translated and output to `<rib_file>`, where `<rib_file>` is a string that is interpreted as a filename. The `<rib_file>` output is in addition to any output that may be generated by extract files. For further information see `-L`, `-O` and `-S`.

If the model contains animation, the translator is animating. When animating, each frame to be output generates a separate file using a filename consisting of the given filename with an extension. The extension is a positive integer. The number of output frames is controlled by the values of `start_frame` (`-s`), `by_frame` (`-b`), and `end_frame` (`-e`). The sense of frame time is first set to the `start_frame` value and one file is generated. Frame time then increments by the `by_frame` value and tests against the `end_frame` value. If the current frame time is less than or equal to the `end_frame` value, another file is output. This continues until the test of frame time fails.

Each output file generated has an extension. The first file has an extension specified as `start_extension` (`-T`). Subsequent files have the extension number incremented by the `by_extension` value (`-B`). All extension numbers are increased by high order zeros to a length specified by `extension_size` characters (`-E`). The values of the extensions do not need to be related to the sense of frame time in any way except that they are constrained to increase over the course of the animation. All extracted files, lights (`-L`), objects (`-O`), and shaders (`-S`), also have extensions appended in the same manner.

`Start_frame`, `by_frame`, and `end_frame` all default to 1. `Start_extension` and `by_extension` default to the actual values for `start_frame` and `by_frame` respectively. None of these values reflect the actual animation range of the model unless you specify it. Information messages are generated stating both the range occurring in the model and the range you specified. If the model contains no animation, the translator does not animate. When not animating, only one file of each type discussed above (`-L`, `-O`, `-o`, and `-S`) is created. The file generated does not have an extension. The values of `start_frame`, `by_frame`, `end_frame`, `start_extension`, `by_extension`, and `motion blur` are all ignored. The value of `<rib_file>` is not verified. The value of `<rib_file>` is used to construct the default value for the rendered image. For further information see `-p`. `<rib_file>` has no default value of its own. The default behavior is to not translate the model at all.

Using options

All options except `-H` require an argument.

You do not need to specify any options, but omitted options assume their default value or behavior.

You can specify any combination of options in any order.

Limitations

- Procedural textures in StudioTools are not supported in our RIB output. This is due to the large differences between the shading languages of StudioTools and PIXAR. We have no plans to write StudioTools's textures in RenderMan format.
- There is no one-to-one mapping from StudioTools's shaders to PIXAR's shaders. We approximate the look, but you should always do a test render before committing an entire animation.
- Other significant features of StudioTools which are unsupported include linear lights, area lights, backgrounds (all types), ambient shade, bump mapping, displacement mapping, and LAYERED fog.
- AliasToRenderman does not support orthographic cameras. If more than one perspective camera is present, only the first is output.
- RenderMan does not automatically generate shadow maps for spotlights; therefore StudioTools's RIB output does not either. For information on creating shadows, please refer to *The RenderMan Companion* by Steve Upstill, Addison Wesley, 1990, ISBN 0-201-50868-0.
- AliasToRenderman generates RIB output which references two custom shaders, **ari_pointlight** and **ari_spotlight**.

The Shading Language source for these shaders is provided in the **gifts** shaders. These shaders are identical to the corresponding published standard shaders in all respects except for one extra parameter — **light decay**.

These versions allow light decay in the same form as do the StudioTools lights, and are provided as a convenience. However, to use them for rendering, you must compile

them (using a RenderMan Shading Language compiler that goes with your RenderMan Renderer) and place the result in your file system where the renderer can find it.

That setup may be inconvenient, so you can override it on the command line for AliasToRenderman standalone. The option is `-d`, and it takes an argument:

- ◆ If the argument is 0 (zero, the default), the StudioTools shader names are generated.
- ◆ If the argument is any other value, the “pointlight” or “spotlight” names are generated.



The `-H` help message includes `-d <d>`.

If `<d>` is zero, “StudioTools” shaders are used where possible, or only RenderMan standard shaders are used.

AlToCa

Platforms

IRIX, Windows

Purpose

AlToCa translates Alias wire file to CAI format files. CAI is CATIA/Alias Interoperability format.

Description

The usage statement for AlToCa is as follows:

:

AlToCa [options] [-i<input Wire file>] -o<output CAI file>

-h outputs online help, then exits.

-b displays a log file on screen during execution

-x produces an extended log file

-i name of input wire file; if not specified, stdin
(generally the keyboard) is used

-o name of output CAI file; must be specified

AIToC5

AIToC5 - Convert Alias Wire files to CATIA Part (.CATPart) files.

Platforms:

IRIX, Windows

Usage:

```
AIToC5 [<options>] -i <input Wire file> -o <output CATIA file>
```

where

Argument	Value
-i	input Wire filename (.wire)
-o	output CATIA filename (.CATPart)
-l	logfile option: 0=No logfile output (default) 1=Output a logfile (-l1) 2=Output an extended logfile (-l2)
-a	Converts each Alias layer to a CATIA GSMTool (Open Body or Geometrical Set) feature, and objects belonging to each layer are converted and belong to the corresponding GSMTool feature (surfacic boby).
-b	Processes symmetry information from Alias layers; the geometric objects are converted.
-v	do not convert invisible geometric entities
-h	display this help information, then exit
Surface (divide/split) options:	
-c	divide closed/periodic surface in multiple surfaces
-d	divide surface with multi-knots in multiple surfaces
Shell stitch (join/close) options:	
-t <tol>	merging tolerance for stitching shell faces (default 0.001 cm)

Argument	Value
-k	keep shell faces after stitching
-g	stitch open shell faces
-f	confirmation for overwriting an existing CATIA Part (.CATPart) file
Join operation for shells	
-s	Do not stitch open/closed shell faces
-j f	Close the resulting skin body (by creating a solid body)
-m	Merging tolerance for joining shell surfaces; should be a value between 0 and 0.1mm. Default value is 0.001mm.
-j a	Turn off angular threshold and angle tolerance option.
-n	Angular tolerance below which the surfaces are to be joined (when angular threshold option (-ja) is on). If the angle value on the edge between two elements is greater than the Angle tolerance value, the surfaces are not joined. This is particularly useful to avoid joining overlapping elements. Its value should be between 0 and 180 degrees. Default value is 0.5 degrees.
-j 0	Turn off check to find out whether faces to be joined are connected.
-j 1	Turn off check to find out whether faces to be joined are tangent.
-j s	Do not automatically reduce the number of faces or edges in the resulting join.
-j r	Do not ignore surfaces and edges that would cause a join to fail.
-j k	Do not keep surfaces of Alias shells after join operation.
Heal operation for shells	
-r	Fill gaps (heal) between shell faces.

Argument	Value
-ff	Do not close the resulting skin body (by creating a solid body)
-fc	Set tangent continuity for healing.
-p	Specify merging tolerance below which the shell's surfaces are to be healed so there is no gap between them. Its value should be greater than or equal to 0.001mm. Default value is 0.001mm.
-q	Specify maximum gap allowed between two healed elements. Default value of 0.001mm; can be increased to 0.1mm.
-t	Specify the tangency angle below which the tangency deviation should be corrected. Its minimum value is 0.5 degrees and its maximum is 10 degrees. Default value is 0.5 degrees.
-u	Specify maximum allowed tangency deviation between healed elements. Value between 0.1 degrees and 2 degrees. Default value is 0.5 degrees.
-fk	Keep surfaces of Alias shells after heal operation.

Platforms

IRIX, Windows

Purpose

AITolv translates Alias wire file to SGI Inventor format files.

Description

:

AITolv [options] [-i <infile> [-o <outfile>]]

-h	outputs online help
-binary	switches the output to a more compact Inventor binary format. The default is Inventor ASCII format.
-ascii	outputs the ASCII inventor
-verbose	displays all messages.
-quiet	operates with no feedback
-notransforms	outputs world space objects
-alltransforms	outputs a full hierachy with transforms at each DAG node and each object space object
-transforms	outputs required transforms at each DAG node and at each object space object
-tri	tesselates all NURBS surfaces to triangles using settings from the Render Stats window
-quad	tesselates all NURBS surfaces to quadrangles wherever possible, using settings from thje Render Stats window
-nurb	outputs NURBS surfaces
-cameras	outputs cameras
-nocameras	does not output cameras

-instances	converts StudioTools instances to Inventor instances
-noinstances	converts StudioTools instances to copies
-inline	creates inline texture data instead of references
-noinline	creates referenced textures where possible (for file textures only)
-inventory	creates a single Inventor material or texture for each StudioTools material or texture, and an instance for each object
-noinventory	creates an Inventor material or texture for each object
-units <i>name</i>	outputs units with a specific name to indicate the type of the units. <i>Name</i> can be: MICRONS MILLIMETERS CENTIMETERS METERS KILOMETERS INCHES FEET MILES
-nounits	don't output units
-scale <i>scale</i>	scale the Inventor file by a specified amount
-xres <i>resolution</i>	Non-file textures sample to this X resolution
-yres <i>resolution</i>	Non-file textures sample to this Y resolution
-noinstances	does not convert StudioTools instances to Inventor instances
Where:	
-i <infile>	specifies an Alias wire file to use as input. If not specified, input comes from stdin.
-o <outfile>	specifies an Inventor file to write output to. If it is not specified, the output is written to stdout.

Limitations

- The Inventor/VRML translator does not support annotation information.
- If you have extended **SoNodeKit** classes in your Inventor files, StudioTools can only read these nodes if there is a Dynamically Shared Object (DSO), containing the extensions, in the path of `LD_LIBRARY_PATH`.

For more information, see *The Inventor Mentor: Programming Object Oriented 3D Graphics with Open Inventor: Release 2*, Chapter 11, “File Format,” Page 17: *Reading in Extender Nodes and Engines*.

The rest of this section lists the Inventor nodes that are partially supported or not supported by the Inventor import and export translators (IvToAl and AlToIv).

- The analytical geometry nodes `SoCube` and `SoCubeDetail` are not supported directly. However, an Inventor file containing these nodes can be imported if optimization is on.
- These text and camera nodes are not supported: `SoAsciiText`, `SoText2`, `SoText3`, `SoTextDetail`, `SoLabel`, `SoFont`, `SoFontStyle`, `SoDrawStyle`, `SoOrthographicCamera`, `SoTranslation` and all their subclasses.
- These animation nodes are not supported: `SoSensor` and `SoEngine` and their subclasses.
- These nodes are implemented as `SoGroup` and do not fully support the concept of level of detail: `SoLOD` and `SoLevelOfDetail`.

AlToObj

Platforms

Windows

Purpose

Converts Alias wire files to OBJ files.

Description

AlToObj [-h] -i <infile> [-o <outfile>] [-h]

-h	displays help information, then exits.
infile	name of input Alias wire file; If this is not specified, standard input (generally the keyboard) is used.
outfile	name of output OBJ format file; this must be specified.
-c	convert faces to curves; default is trimmed surfaces
-g	do not create a group for each geometric object
-u	<units> where 1 is inches 2 is millimeters 3 is feet 4 is miles 5 is meters 6 is kilometers 7 is millimeters 8 is microns 9 is centimeters 10 is microinches
-s	<scale> geometry scale factor
-t	<tol> tolerance value for tessellation

Platforms

IRIX, Windows

Purpose

Converts Alias wire files to ISO10303 format, specifically these:

- application protocols ISO10303-203, or Configuration Controlled Design, conformance classes 1 to 4, and
- ISO10303-214, or Core Data for Automotive Mechanical Design Process, conformance classes 1 and 2.

The import and export of this data is supported via ISO10303-21 Physical File exchange. You might need a separate licence to run this utility.

Application protocol support

The geometric data formats in ISO10303-203 and ISO10303-214 are identical. This comprises the core of the implementation of the translator. The following table shows the mappings made.

Step Entity	StudioTools Entity
Cartesian Point	Point
Line	B-Spline Curve
Circle	B-Spline Curve
Ellipse	B-Spline Curve
Parabola	B-Spline Curve
Hyperbola	B-Spline Curve
PolyLine	B-Spline Curve
Composite Curve	B-Spline Curve (Grouped)
Trimmed Curve	B-Spline Curve
B-Spline Curve	B-Spline Curve

Step Entity	StudioTools Entity
Plane	B-Spline Surface
Cylindrical Surface	B-Spline Surface
Conical Surface	B-Spline Surface
Spherical Surface	B-Spline Surface
Toroidal Surface	B-Spline Surface
Surface of Linear Extrusion	B-Spline Surface
Surface of Revolution	B-Spline Surface
B-Spline Surface	B-Spline Surface
Rectangular Trimmed Surface	Trimmed Surface
Curve Bounded Surface	Trimmed Surface
Offset Surface	B-Spline Surface
Manifold Solid Brep	Shell (Closed)
Shell Based Surface Model	Shell (Open/Closed)

STEP logfile

When retrieval finishes without errors, this message is displayed:

```
STEP files retrieved successfully.
```

If there were errors in the retrieval, you see:

```
Problem Reading Step File, refer to log file for details.
```

The STEP logfile contain an error message for each problem entity that is encountered. Each error contains the STEP entity ID and entity type.

Description

AlToSt [<options>] -i<infile> -o<outfile>

-p	specifies which AP to output: either 203 or 214. The default is AP214
-m <i>value</i>	specifies which model representation to output. <i>Value</i> can be one of: <ul style="list-style-type: none">• 1 = wireframes• 2 = surface models• 3 = manifold surface models (shells)• 5 = brep solids• 6 = hybrid models (the default)
-t <i>value</i>	specifies the type of trimming that is done. This option is valid for surface models only. <i>Value</i> can be one of: <ul style="list-style-type: none">• 1 = parameter space trimming (the default)• 2 = world space trimming
-g <i>value</i>	specifies the type of geometry to output. This option is valid only for shells. <i>Value</i> can be one of: <ul style="list-style-type: none">• 1 = shells only (the default)• 2 = all geometry
-c	outputs presentation data. This option is valid only for AP214 files.
-l	outputs layer data. This option is valid only for AP214 files.
-h	Displays help on usage
Where:	
<infile>	specifies an StudioTools wire file to use as input.
<outfile>	specifies the STEP file to write output to.

AIToUG, AIToUG19, AIToUG20, AIToUG21

Platforms

- IRIX, Windows

AIToUG

Purpose

Convert StudioTools wire files to Unigraphics files.

- AIToUG detects the required output format.
- AIToUG19, AIToUG20, and AIToUG21 convert StudioTools wire files to the respective Unigraphics format (19, 20, or 21).

Description

AIToUG [options] <infile> <outfile>	
infile	The name of the input StudioTools wire file.
outfile	The name of the output Unigraphics file.
Options:	
-i	Input Wire file.
-o	Output Unigraphics file.
-u	Units to be output 1 = millimeters 2 = inches.
-b	Ignore surface continuity breaks (multiknots). Default is to split.
-g	Convert categories. Default is not to convert.
-c	Support collaboration mode.
-d<tol>	Unigraphics distance value to use in a Unigraphics part.
-f	Force to overwrite an existing outfile.
-e	Echo logfile to console.
-l	Create extended logfile.

beep

Platforms

IRIX

Purpose

beep causes the console to make a beep sound.

Overview

beep has several uses. One common use of beep is to have the console beep at you when a render is complete.

Description

The usage statement for beep is as follows:

```
beep
```

Platforms

Windows, IRIX.

Overview

This command-line utility extracts ASCII SDL from a binary SDL file, and inserts ASCII SDL back into a binary SDL file. It is useful when you need to hand-edit scene description language files.

Binary files must be used in StudioTools 12 for rendering, so any file that is converted to ASCII must be converted back to binary format before rendering.

Description

```
bsdl extract [-f] <text> <binary>
```

Extract ASCII SDL to <text> from a binary SDL file called <binary>. The file <text> will not be overwritten unless the option -f is specified.

```
bsdl replace <text> <binary>
```

Insert the ASCII SDL file <text> into the binary SDL file <binary>, replacing the ASCII SDL data.

To see the help statement for the utility, type

```
bsdl -h
```



On Windows operating systems, this stand-alone utility is called bsdl.exe.

CaToAI

Platforms

IRIX, Windows

Purpose

Translates CAI files to Alias wire file format. CAI is CATIA/
Alias Interoperability format.

Description

The usage statement for CaToAI is as follows:

:

CaToAI [options] [-i<input Wire file>] -o<output CAI file>

-h displays help information, then exits

-b displays a log file on screen during execution

-x produces an extended log file

-i name of input CAI file; must be specified

-o output CAI filename; if not specified. stdout is
used.

C5ToAI

Alias provides a stand-alone utility, C5ToAI, to enable you to convert CATIA Part (.CATPart) and CATIA product (.CATProduct) document files to Alias Wire files without running the StudioTools application. You may find this useful for creating a batch script if you have many files that need to be translated from the CATIA format.

Usage:

```
C5ToAI [<options>] -i <input CATIA file> -o <output Wire file>
```

where

Argument	Value
-i	input CATIA filename (either .CATPart or .CATProduct)
-o	output Wire filename (.wire)
-l	logfile option: 0=No logfile output (default) 1=Output a logfile (-l1) 2=Output an extended logfile (-l2)
-a	Converts each CATIA surfacic body or GSMTool (Open Body or Geometrical Set) feature to an Alias layer, and geometrical objects belonging to each GSMTool are assigned to the corresponding layer.
-c	Converts CATIA entities such as points, lines and planes to Alias geometric entities (curves, lines, and bi-linear surfaces) instead of importing them as construction entities.
-d	Import invisible datum features (autonomous geometrical elements with no parents, which do not depend on other specifications) into StudioTools as invisible Alias entities.
-p	Ignore parametric curves and do not convert them to Alias curves.
-s	do not stitch surfaces of solid bodies.
-v	do not convert invisible (hidden) geometric entities

Argument	Value
-x	Ignore CATIA infinite entities (such as lines and planes) and do not convert them to StudioTools.
-h	display this help information, then exit

Using stand-alone utilities on Windows

1 Choose **Programs > Accessories > Command Prompt** from the Start menu.

2 Click the cursor in the window.

You can see what directory you are in by typing
dir

A list of the directory contents will be provided; at the top of the list is your current location.

3 Navigate to the directory containing the files to be converted. If you have installed StudioTools in its default location, the path will be similar to
C:\aw\[UserID]\Alias\user_data\[Project]\wire
where [UserID] is replaced by your login userid (and no square brackets) and [Project] is replaced by the name you have assigned to the project: the default project is demo (again, no square brackets). So to navigate to the directory, type

```
cd C:\aw\[UserID]\Alias\user_data\[Project]\wire
```

4 To use the stand-alone utility to convert a CATIA file named "headlight.CATPart" to a StudioTools wirefile named "headlight.wire" without autostitching, and with a logfile created, type

```
C5ToA1 -l1 -s -i headlight.CATPart -o headlight.wire
```

Using stand-alone utilities on IRIX

1 Open a UNIX shell through the Toolchest or select UNIX shell from the Studio Utilities menu.

2 Click the cursor in the window.

You can see what directory you are in by typing
pwd

- 3 Navigate to the directory containing the files to be converted. If you have installed StudioTools in its default location, the path will be similar to

```
/usr/u/[UserID]/Alias/user_data/[Project]/wire
```

where [UserID] is replaced by your login userid (and no square brackets) and [Project] is replaced by the name you have assigned to the project: the default project is demo (again, no square brackets). So to navigate to the directory, type

```
cd /usr/u/[UserID]/Alias/user_data/[Project]/wire
```

- 4 To use the stand-alone utility to convert a CATIA CATPart file named “headlight.CATPart” to a StudioTools wirefile named “headlight.wire” without autostitching, and with a logfile created, type

```
C5ToA1 -l1 -s -i headlight.CATPart -o headlight.wire
```

dxg_info

Platforms

IRIX

Purpose

dxg_info prints on the screen a summary of the dxg entities used in the named dxg files.

Description

The usage statement for dxg_info is as follows:

```
dxg_info file [file ...]
```

Example

The following is an example of how dxg_info is used.

```
dxg_info SAMPLE.DXF
```

Information about the file SAMPLE.DXF is displayed as follows:

```
Summary of file SAMPLE.DXF
```

```
-----
```

```
entity unsupported if marked ***
```

```
BLOCKS SECTION:
```

entity	BLOCK	occurs	14 times
entity	CIRCLE	occurs	91 times
entity	POLYLINE	occurs	38 times
entity	VERTEX	occurs	1661 times
entity	LINE	occurs	61536 times
entity	3DFACE	occurs	597 times
entity	POINT	occurs	1 times
entity	ARC	occurs	2 times

ENTITIES SECTION:

entity	LINE	occurs	672 times
entity	ARC	occurs	44 times
entity	3DFACE	occurs	44 times
entity	POLYLINE	occurs	164 times
entity	VERTEX	occurs	1705 times
entity	CIRCLE	occurs	34 times
entity	INSERT	occurs	129 times
entity	TEXT***	occurs	4 times

FieldAssembler

Platforms

Windows

Purpose

Interlaces scan-lines of images rendered on fields into a single interlaced image; also pastes together incompletely rendered images.

Description

The FieldAssembler utility interlaces field-rendered images. Field-rendered images are created in a sequence for video playback. Television draws every other line on the screen and then fills in the alternate lines. Each field is one-half of a frame (the even lines or the odd lines).

NTSC and PAL video systems both use interlaced fields.

- NTSC video systems, used in North America, display 30 frames or 60 fields per second.
- PAL video systems, used in Europe, display 25 frames or 50 fields per second.

Each field is stored in a separate file. For animations that will be played back on television, you can render images into fields. However, to view them on a computer screen or to transfer them to certain other computer programs, you need to reassemble, or interlace, them into full images. The result is half as many files that each have twice as much data.

For information on FieldAssembler, from the FieldAssembler window select **Help > Help**.



If you will be using Composer to composite images rendered as fields, you must either interlace the fields together before importing them into Composer, or render the images in Alias PIX or RLA format. You can only import fields into Composer if they are in Alias PIX or RLA format. (In addition, RLA format fields

must be named name.1, name.2, name.3, name.4, and so on, not name.1o, name.1e, name.2o, name.2e, and so on.)

findit

Platforms

IRIX

Purpose

findit determines whether a specified picture file exists on the render engine.

Description

:

```
findit <pix_path> <machine_name>
```

<pix_path>	specifies the picture file to locate.
-------------------------	--

<machine_name>	specifies the name of the render engine on which to search for the picture file.
-----------------------------	--

findit completes the search for the picture file and then returns one of the following values:

\$1	is the render code location, the location of picturefile.1.
-----	---

\$2	is the host machine directory location.
-----	---

StudioTools uses **findit** for remote rendering started via the Render Control window.

If the picture file exists, **findit** writes information about it to the file `_lout` on the host machine.

flipbook

Platforms

IRIX

Purpose

flipbook is an animation viewer which allows you to:

- view sequences of PIX, SGI, TIFF, or TIFF16 files simultaneously
- view sequences of PIX, SGI, TIFF, or TIFF16 files in sequence
- maintain a specified framerate
- compress animations in a variety of ways.

Overview

FlipBook is available inside the StudioTools package, and as the stand-alone utility, `flipbook`, suitable for presentation use. Compressed FlipBook animations can be saved as single files to quickly load later on.

Description

```
flipbook [-c# #] [-d] [-e #] [-h] [-i] [-m] [-o] [-p <path>] [-P<root>] [-r #] [-R] [-s] [-t <text>] [-z #]
<filename> [,start[,end[,by]]]
```

<code>-c# #</code>	sets the space and time compression.
<code>-d</code>	turns OFF double-buffering.
<code>-e #</code>	sets the extension width which is the pad extension with leading zeros.
<code>-h</code>	prints the on-line help.
<code>-i</code>	displays information about which version of <code>flipbook</code> is running.
<code>-m</code>	opens the FlipBook control window.
<code>-o</code>	sets oscillation of animation ON.
<code>-p <path></code>	sets the default browser directory path to a specific path.
<code>-P <root></code>	reads the book file and creates corresponding pix files with a specified root name.

-r #	sets the pixel sample rate, with or without filtering.
-R	forces flipbook to avoid memory mapping and use real memory. This option is handy when you have lots of main memory but little disk space.
-s	turns OFF framerate synchronization.
-t <text>	specifies a file extension to be found. The filename is in the format XXXX.##.ext. For example, -t Z finds the file testpix.1.Z.
-z #	zooms into the image by an integer factor.
<filename> [,start[,end[,by]]]	specifies the animation sequence to be loaded, and can specify the start, end, and by values to use while loading. One or more filenames can be specified at once.

When the command is entered, the FlipBook window appears and starts cycling through the animation.

If you have specified the [-m] option, the FlipBook control window is also displayed.

For information about using Flipbook within StudioTools, see the *Animating* book.

Notes

- Flipbook does not display mask files. On IRIX, use the **fcheck** utility supplied in directory `/usr/aw/com2.0/bin` to display sequences of mask files.
- Flipbook does not support negative file extensions. Always use positive integers for file extensions.

fmovie

Platforms

IRIX

Purpose

fmovie converts a sequence of Alias image files into an SGI or Apple QuickTime movie.

Overview

fmovie is used to take a sequence of image files and create an industry standard movie file from them, such as a SGI movie file or an Apple QuickTime movie.

The QuickTime option can also be used to ease the transfer of animation output to an Apple Macintosh.

Description

fmovie [-a <audiofile>] [-b #] [-c <method>] [-e #] [-f #] [-h] [-i <infile>] [-o <outfile>] [-r #] [-s #] [-S #] [-v] [-w #]

-a <audiofile>	specifies the name of the audio track to go with the movie file. The audio track may be in AIFF or AIFC format.
-b #	specifies how many frames to count by.
-c <method>	specifies the compression method to use in creating the movie file. The choices are: NONE = No compression RLE = 8 bit run length encoding RLE24 = 24 bit run length encoding JPEG = JPEG encoding MVC1 = SGI format 1 encoding MVC2 = SGI format 2 encoding "Apple Video" = Apple video encoding "Apple Animation" = Apple animation encoding.
-e #	specifies the frame number of the last image file to use.

-f #	specifies the format of the movie file to be created. 4 formats are currently available: 0 = SGI format 1 1 = SGI format 2 2 = SGI format 3 (default) 3 = QuickTime.
-h	displays the help messages.
-i <infile>	specifies the input sequence of image files. Files may be in PIX, SGI, TIFF, TIFF16, RLA, or HARRY format.
-o <outfile>	specifies the name of the output movie file.
-r #	specifies the playback rate of the movie.
-s #	specifies the frame number of the first image file to use.
-S #	specifies a scale factor to apply to the input images while reading them. The choices are: 1 = full scale 2 = 1/2 scale 4 = 1/4 scale 8 = 1/8 scale 16 = 1/16 scale.
-v	displays all messages while creating the movie.
-w #	specifies the extension width of the input frames. For example, for foo.001,width would be 3.

Example 1

To JPEG compress all frames of foo into foo.mv, use the following:

```
fmovie -i foo -o foo.mv -c JPEG -v
```

Example 2

To create a quick time movie of the above, use the following:

```
fmovie -i foo -o foo.qt -f 3 -c "Apple Video"
```

Platforms

IRIX, Windows

Purpose

The standalone utility **from100to97** lets you convert a wire file from StudioTools 10 or later to a StudioTools 9.7 wire file.

Description

The following table outlines the details of this function:

from11to97 -s [saved_file] [retrieved_file]	
-s	saves the result
saved_file	is the name of the new, V9.7 file
retrieved_f ile	is the name of the old, V10 file
-w	overwrite savFile if it exists



The **from100to97** utility will convert fillets created with the new **Fillet surface** tool and save only the surface information. It will *not* save construction history, so you will not be able to modify the fillet after conversion. See New Surface Fillet tool section in the What's New manual.

The **from100to97** utility will convert round surfaces created with the **Round** tool and save only the surface information. It will *not* save construction history, so you will not be able to modify the round after conversion.

Limitations

- Object editor commands are discarded (because they were not available in Version 9.7)

- Offset commands are discarded (because they were not available in Version 9.7)
- Fillet surface commands are discarded. Because of the great number of improvements to fillets that were made in Version 10, these commands aren't readable in previous versions.
- Because sketching is only available on the Windows platforms, any sketch data that is contained in a StudioTools 10 file is only converted to Version 9.7 if the from100to97 utility is run on a Windows platform. If run on any other platform, this data is removed.

To use the from100to95 utility on windows

- 1 Choose Start Menu -> Programs -> Accessories and open the Command Prompt window



In this example, we are assuming that StudioTools is installed in the default location, and that the wire file directory is also in the default location. Let's say your wirefile is called "bunchaspheres.wire"

- 2 Type in the Command Prompt window:

```
cd C:\Program Files\Alias\StudioTools10.0\bin
and press Enter.
```

This puts you in the directory where from100to97.exe is located.

- 3 If your original wire file is called "bunchaspheres.wire", you need to choose a new file name for the 9.7 file. Let's call it "bunchaOLDSpheres.wire"

You'll also need to use your full path to where the wire file is located; in this case, let's assume it's

```
C:\aw\username\Alias\user_data\wire\
```

- 4 To convert the file to 9.7, type:

```
from100to97 -s
C:\aw\username\Alias\user_data\wire\buncha
OLDSpheres.wire
C:\aw\username\Alias\user_data\wire\buncha
spheres.wire
```

To use the from100to95 utility on UNIX

In this example, we are assuming that StudioTools is installed in the default location, and that the wire file directory is also in the default location. Let's say your wirefile is called "bunchaspheres.wire"

- 1 Open a shell and type:

```
cd /usr/aw/alias10.0/bin
```

and press **Enter**.

This puts you in the directory where from100to97.exe is located.

- 2 If your original wire file is called "bunchaspheres.wire", you need to choose a new file name for the 9.7 file. Let's call it "bunchaOLDSpheres.wire"

You'll also need to use your full path to where the wire file is located; in this case, let's assume it's

```
/aw/username/alias/user_data/wire/
```

- 3 To convert the file to 9.7, type:

```
from100to97 -s
```

```
C:\aw\username\alias\user_data\wire\buncha  
OLDSpheres.wire
```

```
C:\aw\username\alias\user_data\wire\buncha  
spheres.wire
```

fstats

Platforms

IRIX

Purpose

fstats reports pixel and scan-line statistics for images.

Description

fstats supports the following file formats:

- Alias
- SGI
- TIFF
- TIFF16
- RLA
- TIM
- HARRY

fstats [-f] [-v] image [...]

-f	attempts to read more pixels every second and update the scan-line/pixel count when the end of an incomplete image is reached.
-v	displays the total number of pixels. If the image is an Alias format file, the number of run-length packets and the average run-length of each packet are also displayed.
image	is a picture or matte file in one of the formats specified above.

The X and Y resolution are reported to standard output.

If the image is an Alias format file, the Y resolution is printed as `yres.fraction` if the number of complete scan-lines does not match the Y resolution specified in the header. If this is the case, the image is incomplete. `yres` always represents the actual

number of complete scan-lines.= and.fraction represents the number of pixels calculated so far on the incomplete scan-line.

Example

yres=483.639 is one pixel short of a complete 640x484 image in Alias file format.

get_alias_variable

Platforms

IRIX

Purpose

get_alias_variable queries the value of Alias preference variables from the command line.

Overview

`get_alias_variable` is called from your file.cshrc at login with a variable name as an argument. It attempts to determine the value of the variable by scanning through `$HOME/.AliasPrefs` and `/usr/alias/.AliasPrefs`, in that order. This function does not find environment variables, such as `ALIAS_LOCATION`, that are set at the shell level (in a command window).

If a value is found, it is written to stdout. If no value is found, an error message is written to stderr.

Description

The usage statement is

```
get_alias_variable <var_name>
```

where `<var_name>` is the name of an Alias preference variable.

`get_alias_variable` completes the search for the variables and then returns one of the following:

- | | |
|---|--|
| 0 | indicates that a variable was defined and a value was found. |
| 1 | indicates that the variable could not be found in a <code>.AliasVariables</code> file. |
| 2 | indicates unauthorized use of the stand-alone. |

Example

The following example illustrates a use of `get_alias_variable`.

```
get_alias_variable ALIAS_EDITOR
```

returns the value 0 and the pathname to the editor:

```
/usr/sbim/jot_f
```

getid

Platforms

IRIX

Purpose

getid provides helpful information about your machine's set-up and about the version of StudioTools that you are running.

The information provided by `getid -i -v` is available from within StudioTools by your selecting `Help > About StudioTools`.

Overview

You need some of the information provided by `getid` to get an encrypted text string for a licence. If you are having problems with your system, it is useful to write this information down before calling for assistance.

Description

```
getid [-g] [-h] [-i] [-v] [-w]
```

no options	displays the getid number and system information.
-g	displays only the getid number.
-h	displays the on-line help.
-i	displays more system information.
-v	verifies the encrypted string for your machine.
-w	displays additional information about Wavefront products after the getid number.

Example

The following is an example of the information displayed if `getid -i -v` is typed at the command line:

Getid number:	87978551325080 (tempest)
Host type:	S,1,150,12,11,163 (Indigo2 Extreme)
Operating System:	SGI Irix 6.5 (04151556)
Main memory size:	128 Mbytes
Swap space size:	100 Mbytes
Running NFS:	yes
Media drives:	CD-ROM, DAT
Software cut:	1999-10-17 01:53
Software location:	/usr/aw/alias
Runtime checksum:	906.142.2
Current date/time:	Tue Jul 18 12:33:33 EDT 2000
Free disk space:	187 Mbytes in / 637 Mbytes in /usr

Features (versions)	ST95 (1.000) AliasUtils95 (1.000)
Expires on:	none (permanent)
Product:	Studio V11
Enabled options:	Studio base
	Advanced Evaluation
	Advanced Modeling
	Advanced Design Animation
	Direct Model
	Alias Utilities

Explanation of the system information fields:

Getid number:	[unique number for software licensing] (hostname)
Host type:	[numerical description] (official SGI name, if known)
Operating System:	SGI Irix [Irix release] (Irix version)
Main memory size:	rounded up to nearest Mbyte. 1 Mbyte is 1024 * 1024 bytes.
Swap space size:	rounded up to nearest Mbyte.
Running NFS:	yes or no.
Media drives:	CD-ROM, QIC 24, QIC 150, DAT, 8 mm (2.3 Gbyte), 8 mm (5 Gbyte), etc.
Software cut:	the date and time when the software was built.
Software location:	where software is installed.
Runtime checksum:	used on the SPAR form.
Current date/time:	same format as the result of the UNIX date command.
Free disk space:	rounded down to nearest Mbyte. Only writable local EFS and XFS filesystems are reported. NFS filesystems are not reported.

Information about the Host type

The Host type field reports the official SGI name (or our best guess at the official name in the case of new systems) for almost all SGI systems produced in the last three years. The Host type is reported even if the system is not qualified to run StudioTools software. Check the StudioTools CD-ROM booklet or the *Installation Notes* to determine whether the system is capable of running StudioTools. The numerical Host type was devised by Alias Systems to identify the system when the official SGI name is not known, such as for new systems or unusual configurations. The SPAR form requires the numerical host type so that there is no ambiguity.

hp_glplotf, hp_gl2plotf

Platforms

IRIX, Windows

Purpose

hp_glplotf translates Alias plot files into HP-GL (IBM GL) format files that are suitable for plotting on any supported plotter.

hp_gl2plotf creates similar output, but includes initialization commands required for HP-GL2 plotters.

Overview

hp_glplotf and hp_gl2plotf are filters which take input from stdin or <in_file> and send output to stdout or <out_file>, which would normally be sent to lp.

Description

hp_glplotf [-a] [-c<x>[<y>]] [-h] [-i] [-m<model>] [-n] [-p<paper>] [-r<angle>] [-s] [t # #]
[<in_file> [<out_file>]]

-a	specifies automatic feed is to be used.
-c<x>[<y>]	specifies scale correction factors.
-h	displays the on-line help.
-i	displays version information.
-m<model>	specifies the plotter model type.
-n	separates HPGL plot commands on new lines.
-p<paper>	specifies the paper type.
-r<angle>	specifies the force plot rotation angle.
-s	sizes the plot produced automatically. The plotter's default size is used.
-t # #	specifies the text scaling where # and # are the width and height.
<in_file>	reads input from the file <in_file>. The default is to read from stdin.
<out_file>	writes output to the file <out_file>. The default is to write to stdout.



Any errors encountered by the HP-GL and IBM-GL device drivers are written to a file `/tmp/plot_err.<number>` as well as `/usr/adm/syslog`. Look there for any clues as to why the plotter is not working. The `<number>` is determined by the operating system.

Automatic page advance

Some plotters can be fitted with a spool feed or have an automatic page advance option. This also applies to printers like the IBM-4019 and IBM-4029, which can automatically advance the page. To tell the plotter driver that you are using the spool feed option the following option should be specified:

```
-fa
```

Correction factor

When large plots are plotted, some inaccuracies can creep in over large distances. As an interim fix, it is possible to compensate for this error by specifying x and y correction factors. The default x and y correction factor is 1.0, which is no correction at all. The correction factors are used internally as multipliers.

Example

If you plotted something that should have come out 30" and it came out to 30.125", the plot was 0.125" too large. Assuming that the same error occurs in both the x and y directions, you would need a correction factor of $30/30.125 = 0.9959$. The following should compensate for the error:

```
-fc0.9959
```



Setting the correction values in the **Plotting Interface** window opened by **File > Plot**, overrides the correction values in the **LP queue options** for plot output section of the **Alias Preferences** window.

iges_info

Platforms

IRIX

Purpose

iges_info provides a summary of the iges entities used in the named files.

Description

The usage statement for **iges_info** is as follows:

```
iges_info file [file...]
```

Example

The following is an example of the use of **iges_info**.

```
iges_info sample.iges
```

The following is an example of the type of information provided by the command stated above.

```
Summary of IGES file sample.iges
```

```
-----
```

```
entity unsupported if marked ***
```

entity	124 form	0 occurs	9 times.	-- Transformation Matrix
entity	108 form	0 occurs	9 times.	-- Plane
entity	406 form	15 occurs	2 times.	-- Name
entity	110 form	0 occurs	64 times.	-- Line
entity	104 form	1 occurs	1 times.	-- Conic Arc
entity	212 form	0 occurs	109 times.	-- General Note ***
entity	100 form	0 occurs	11 times.	-- Circular Arc
entity	102 form	0 occurs	2 times.	-- Composite Curve
entity	106 form	3 occurs	1 times.	-- Copious Data

entity	106 form	63 occurs	2 times. -- Closed Planar Curve
entity	128 form	0 occurs	2 times. -- Rational B-Spline Surface
entity	126 form	0 occurs	14 times. -- Rational B-Spline Curve
entity	116 form	0 occurs	3 times. -- Point
entity	120 form	0 occurs	1 times. -- Surface of Revolution
entity	118 form	1 occurs	1 times. -- Ruled Surface
entity	140 form	0 occurs	1 times. -- Offset Surface
entity	118 form	0 occurs	2 times. -- Ruled Surface
entity	106 form	12 occurs	1 times. -- 3D Path
entity	142 form	0 occurs	1 times. -- Curve on a Parametric Surface
entity	144 form	0 occurs	1 times. -- Trimmed Parametric Surface
entity	214 form	2 occurs	18 times. -- Leader (Arrow) ***
entity	206 form	0 occurs	1 times. -- Diameter Dimension ***
entity	106 form	40 occurs	9 times. -- Copious Data
entity	402 form	7 occurs	1 times. -- Group Without Back Pointers
entity	114 form	0 occurs	2 times. -- Parametric Spline Surface

Done for the named file(s).

imgcvt

Platforms

IRIX, Windows

Purpose

The `Imgcvt` utility converts images or a sequence of images from one image format to another.

Description

In any shell window, enter `imgcvt` followed by the name of the image that you want to convert. The input and output image formats are usually determined by the filename extension or image content. However, when the input or output image has no extension, or cannot be identified, the `-f` and `-t` options can be used to indicate the desired formats. You can use the following options.

```
imgcvt <options> input_image output_image
```

Example 1

```
imgcvt -f sgi -t tiff input_image output_image
```

This line converts the image named `input_image` that is in the Silicon Graphics format to the TIFF format and saves the image out to a different name, `output_image`.

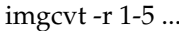
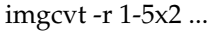
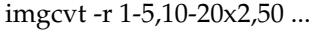
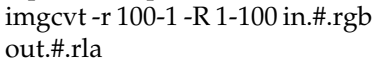
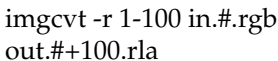
Example 2

```
imgcvt -f tiff -t iff input_image output_image
```

This line converts a TIFF image to the IFF file format.

The following table lists the `imgcvt` options.

Command line options	Description
-f input_image_extension	Identifies the specific image format that you are converting from. For example, -f sgi means that only files matching the Silicon Graphics image format are accepted as input.
-t output_image_extension	Identifies the specific image format that you are converting to. For example -t tiff means that files will be converted to the TIFF image format only. For the list of possible extensions, see the section on <i>Using image filename extensions</i> .
-n start end step	Is the start, end and step of an input image sequence (only whole numbers are accepted).
-N start end step	Is the start, end and step of an output image sequence (only whole numbers are accepted).

Command line options	Description
-r range	In an input image sequence, selects the range of images to be converted from. For example:    A sequence specifier is a list of single frames or ranges separated by a comma or a slash. In this example, they are individual, numbered files. For each range, an optional step can be specified after an <i>x</i> . Sequences can be reversed and offsets can be added to the input or output frame number:  
-R range	In an output image sequence, selects the range of images to be converted to.
-h	Provides a brief description of each option.
-s	Silent mode.
-v	Verbose mode.
-V pal/ntsc	Video mode for yuv files.
-C compress	TIFF compression types: LZW (the default) or NONE.
-q N	JPEG quality (0...100; default is 75).
-m	Explore map format.

When you are working on a sequence of images, the # and @ symbols refer to the current frame number:

- The hash (#) indicates a four-digit padded number,

- whereas the @ symbol indicates a non-padded number.

The following converts a sequence of 100 frames:

```
imgcvt -n 1 100 1 image_in.iff.@ image_out.#.rla
```

For more precise control of the frame number format, use either multiple @ symbols to set a specific padding or the standard printf(3S) notation:

```
imgcvt -n 1 100 1 image_in.@@@.rgb image_out.#.rla
imgcvt -n 1 100 1 image_in.%03d.rgb image_out.#.rla
imgcvt -n 1 100 1 image_in.%.2f.rgb image_out.#.rla
```

Using image filename extensions

Use the following filename extensions to convert images from one image format to another.

- In the -f option, you specify the extension of the image format that you are converting from.
- In the -t option, you specify the extension of the image format that you are converting to.

Image Format	Extension
Abekas NTSC or PAL	.yuv
Alias	.als
Explore	
GIF	.gif
JPEG	.jpg
Kodak Cineon	.cin
Lucas Film	.lff
Maya IFF	.iff
Pixibox PXB	.pxb
PXN	
PPM raw/ascii	.ppm
Prisms	.pri
Quantel	.qtl
SGI	.rgb, .sgi or .bw

Image Format	Extension
Softimage	.pic
Targa RGB/BW	.tga
TIFF 6.0	.tif or .tiff
Vista	.vst
Wavefront RLA	.rla
Windows bitmap	.bmp

IvToAl

Platforms

IRIX, Windows

Purpose

IvToAl translates SGI Inventor files into Alias wire files.

Description

IvToAl [<options>] [-i <infile> [-o <outfile>]]

-o	Do not optimize the Inventor file
-s <i>scale</i>	Use an input scale factor of <i>scale</i> (for example, 2.0)
-g	Do not group the geometry
<infile>	specifies an Inventor file to use as input. If infile is not specified, input comes from stdin.
<outfile>	specifies the Alias wire file to write output to. If outfile is not specified, output is written to stdout.

For information on this utility's limitations, see *AlToIv* (page 24).

lmgrd

When you invoke `lmgrd`, it looks for the license file which contains the information about vendors and features. Usage is:

```
lmgrd [ -app ] [ -c license_file ] [ -t  
timeout_interval ] [ -l logfile ]  
[ -s timestamp_interval ] [ -2 -p ] [ -v ] [ -x  
lmdown ] [ -x lmremove ]
```

Option	Purpose
-app	Required for Windows when run as a command, but not used when run as a service.
-c license_file	Use the license file specified by <code>license_file</code> .
-t timeout_interval	Sets a timeout interval, in seconds, during which redundant daemons must complete their connections to each other. The default value is 10 seconds. A larger value may be desirable if the daemons are run on busy systems or a very heavily loaded network.
-l logfile	Write the debug log to the specified logfile.
-s timestamp_interval	Specifies the logfile timestamp interval, in minutes. The default is 360 minutes.
-2 -p	Restricts usage of <code>lmdown</code> , <code>lmreread</code> , and <code>lmremove</code> to a FLEXlm administrator who is by default the root account. If there is a Unix group called <code>'lmadmin'</code> , then use is restricted to members of that group. If root is not a member of that group, then root can not use any of the above utilities. The <code>'-p'</code> option is available in FLEXlm v2.4 and later.

Option	Purpose
-v	Prints lmgrd's version number and copyright, and then exits.
-x lmdown	<p>Disallows the lmdown command (no user can run lmdown). If lmdown is disabled, you must stop lmgrd via 'kill pid' (Unix) or CTRL-ALT-DEL and stop the lmgrd and vendor daemon processes (Windows 95). On Unix, be sure that the kill command does not have a -9 argument.</p> <p>The -x lmdown option is available in FLEXlm v4.0 and later.</p>
-x lmremove	<p>Disallows the lmremove command (no user can run lmremove).</p> <p>The -x lmremove option is available in FLEXlm v4.0 and later.</p>

lmutil

Platforms

IRIX, Windows

Purpose

lmutil is the first among FLEXlm license manager utilities. FLEXlm is the network license manager used by Alias Systems to control the use of their software products.

If you are a UNIX systems administrator or user, it is likely that one or more of the products on your network is licensed by FLEXlm. Details about installing with FLEXlm are in your Installation Notes.

The StudioTools installation program creates a FLEXlm directory, which contains a FLEXlm control panel and its required DLL, plus a number of related utilities

- lmutil
- lmtools
- sgiawd (if you install the licensing component of the software)
- lmgrd.exe (if you install the licensing component of the software)

While you do not call all the utilities directly, they are needed and must not be deleted.

makebot

Platforms

IRIX

Purpose:

To create block ordered texture (BOT) files.

Description

makebot [-m | -d] input output

If no option is given, the output file created will be a compressed, block ordered texture file with no mipmap built in.

-m the output file created will be a compressed, block ordered texture file with a mipmap built in.

-d the input file must be a compressed, block ordered texture file. The resulting output file will be in Alias pix format, and will be equivalent to the image contained in the input file. (Note: If the input block ordered texture file contains a mipmap, only the highest resolution level will be contained in the resulting output image).

A compressed block ordered texture containing a mipmap (i.e., created with the -m option) can be used as a texture with either filter OFF or ON. A compressed block ordered texture created without a mipmap (i.e., created without the -m option) can only be used as a texture with filter OFF. Unless you do not plan on using filtering on a texture, the -m option should be specified when creating the compressed block ordered texture files.



If you use BOT files as textures, you must turn on **Texture Caching** in the **Render Globals** window or the renderer will fail.

nprocs

Platforms

IRIX

Purpose

nprocs displays information about the host machine's processor(s).

Description

nprocs is mainly of use on multi-processor machines. It displays information about the number and type of processors on a machine, and queries each processor to determine if any special flags have been set or if any of the processors have been locked by using IRIX system calls.

nprocs displays the following information:

```
HOST 'whistler'
```

```
IRIX 6.2 03131015 IP22 mips 1762190057 /etc/inittab  
Silicon Graphics, Inc.  
Silicon Graphics, Inc. IRIX IP22 1 c04b146e 6 2 0  
Processors      : R4600 2.0  
Available Procs: 1
```

```
Allowed processes  : 150  
Parallel processes : 1  
Page size         : 4096  
Number of processors: 1  
Unlocked processors : 1  
    Processor 0 --> Id: 0  Flags: Enabled Master  
Clock FastClock
```

ObjToAl

Platforms

IRIX, Windows

Purpose

Converts OBJ files to Alias wire files.

Description

ObjToAl -i <infile> [-o <outfile>] [-h]

infile	the name of the input OBJ file; this must be specified.
outfile	the name of the output Alias wire file. If this is not specified, standard output (generally the screen) is used.
-h	displays help information, then exits.

Platforms

IRIX

Purpose

The Plug-in Manager script supports the loading, removal and distribution of plug-ins. The Plug-in Manager script is called *pim* (*Plug-in Manager*) and is located in `$ALIAS_LOCATION/bin/`.

Description

Plug-in archives

The plug-in manager script allows users to work with plug-in manager archives. The suffix used to represent a plug-in manager archives is *.pim*. Plug-in manager archives encapsulate the necessary files of a plug-in into one archive file. All scheme, icon, binary(.plugin) and other supporting files will be placed into the archive. Archives can be created, appended to, viewed and installed using the Plug-in Manager script.

Environment variables for the Plug-in Manger script

The Plug-in Manager script requires that `$ALIAS_LOCATION`, `$ALIAS_WORKENV` and `$HOME` be set. In addition, `$ALIAS_LOCATION/bin/aliasWarning` must exist.

Plug-in Manager script: Command Line and GUI mode

The Plug-in manager script supports both Command Line and GUI modes. The script is layered so that the user input code is isolated from the code that actually performs the generic operations required. GUI mode is utilized to perform operations that are called from within StudioTools.

Plug-in Manager script options

The Plug-in Manager script supports the following options:

-install <pim file, pim file, ... >	Used for local or global installation of plug-in manager archive files
-remove <pim file, pim file, ... >	Used to remove plug-in manager archive files
-view <pim file, pim file, ... >	View the files within a plug-in manager archive
-create <pim file> <plug-in files> <scheme files> <icon files> <other files>	Allows for the creation of plug-in manager archive files
-cR <pim file> <plug-in file directory>	Recursive creation of a plug-in manager archive from a directory
-append <pim file> <plug-in files> <scheme files> <icon files>	Appends to an already created plug-in manager archive file
-gui_install <pim file, pim file, ... >	Installs plug-in manager archive files through the desktop
-h	Help

The creation (-create, -cR) and append (-append) routines require the files that will be placed into the archive be in a specific format. This format is:

- *.plugin in ./lib
- *.scm in ./scm
- *.M in ./medium
- *.S in ./small

- other files can be stored in ./files

This rigid format has been chosen so that special logic is not required to install the individual files from the archive.

Workflow using the Plug-in Manager script

Creating a Plug-in Manager archive

The following is an example of how to create a plug-in manager archive

```
pim -c nothing.pim ./lib/nothing.plugin ./scm/
nothing.scm ./scm/nothing_init.scm ./medium/
nothing.M
```

The files that will be inserted into the archive must be relative to the current directory. Note that the created file's location could be specified in another directory.

```
pim -c /tmp/nothing.pim ./lib/nothing.plugin ./
scm/nothing.scm ./scm/nothing_init.scm ./medium/
nothing.M
```

If the directory NothingPlugin has the proper structure, then the recursive creation option could be used.

```
pim -Cr nothing.pim NothingPlugin
```

Installing a Plug-in Manager archive from the command line

The following is an example of how to install plug-in manager archives using the script.

```
pim -i /tmp/nothing.pim
```

You will be asked to choose between local or global installation. Local installations put the plug-in files into \$HOME/.Alias/plugins. This makes plug-ins available locally to any StudioTools session. Global installations place the plug-in files into \$ALIAS_LOCATION/ODS/OpenAlias/plugins. If it's a global installation, then the user must have the privileges required to place files within the Alias directories.

Installing a Plug-in Manager archive locally by using drag & drop

You can install the files of a plug-in manager archive locally into `$HOME/.Alias/plug-ins` by dragging and dropping the archive onto the StudioTools icon on the computer's desktop. The files are copied out of the archive and placed in the local plug-in directory, with warning prompts if there are duplicates. An "Installation Complete" dialog signals the end of the install process.

At the end of the installation process of the plug-in, StudioTools exits. In the next session of Alias, the plug-in installed will be displayed in the Plug-in Manager window.

StudioTools attempts to install all files dropped as Plug-in Manager Archives if the first file of the possible set that was dropped onto StudioTools has the suffix `.pim`. This implies the following:

- if a `.pim` is followed by a wire file, then StudioTools attempts to extract files from the wire file as it would from a `pim` file. The plug-in manager archive extraction from a wire file will fail.
- if a wire file is followed by a `.pim` file, then StudioTools starts and attempts to load the `.pim` file as a wire file. The plug-in manager archive will fail to read as a wire file.

Running the archive removal script

To simplify the removal of plug-ins, when the plug-in manager script executes an install, it also builds a removal script. This removal script can be run in two ways. It can be run from the command line:

```
pim -r helix.pim
```

The user will be asked if it is a local or global removal. If its a global removal, then the user must have the privileges required to remove files from the Alias directories.

The second way of deleting the files associated with a plug-in that was installed locally is to use the desktop to run the appropriate removal script located in `$HOME/.Alias/plug-ins/remove`. Double clicking on the script will execute it to run the removal code. As a result, plug-ins should not share scheme, icon or other supporting files.

pixdiff

Platforms

IRIX

Purpose

pixdiff compares two pix files, pixel by pixel, to determine where they differ.

Overview

pixdiff supports the following file formats:

- Alias
- SGI
- TIM
- TIFF and TIFF16
- RLA
- HARRY

Description

```
pixdiff [-d] [-l] file1 file2 [outfile]
```

-d	creates an intensity difference file rather than a half-intensity pix file.
-l	prints a long version of the pixel differences and creates a half-intensity file with the differences at full intensity.
outfile	specifies an output file containing the pixdiff image. The file format is the same as file1.

pixdiff checks the alpha channel for differences, in addition to the RGB channels if the input file type has a bit depth of 32. If file1 has fewer channels than file2, the outfile uses the smaller of the two channels. The smaller number of channels is checked

for differences. In addition to generating an image, the following statistics are also listed:

- number of differences
- maximum difference
- mean difference
- variance
- RMS (Root Mean Square)

print_wire_header

Platforms

IRIX, Windows

Purpose

print_wire_header prints the header and can be used to verify the named wire files.

Description

print_wire_header [-v] file [file...]

-v	sets the verify mode. If -v is included, the entire wire file is read and if the trailer record is not found, the file is reported as invalid. This option indicates that a wire file is not truncated. It is useful for verifying the success of a file transfer operation, such as a copy to or from tape or across NFS.
file [file...]	is a list of files to be processed. There must be at least one file listed. Wildcard listing of files is supported.

Example

The following is an example of output produced using the command `print_wire_header -v *` in a directory.

```
Filename:                               SkiBoot
-----
Alias Product Name:                      Alias Studio
-----
Alias Product Version:                   11
-----
Alias Software Date                      2001/07/18 13:08
-----
Alias File Version:                      V11-12
-----
File Creation Date:                      Thu Jul 29 17:09:25
                                           2001
```

```
File Type:                WIRE
-----
Coordinate System        Z Up
-----
Graphics Display Size:  830 x 840
-----
SkiBoot_strap.iges is not an Alias Wire File.
-----
Warning: Wire file truncated.als is
corrupted. Trailer record is missing!
```

psplotf

Platforms

IRIX, Windows

Purpose

Translates Alias plot files into PostScript for printing on any PostScript printer.

The plotter driver program `psplotf` is a filter that takes input from `stdin` or `<in_file>` and sends output to `stdout` or `<out_file>` instead of `lp`.

Description

```
psplotf [-f fontname] [-h] [-i] [-r #] [-s #,#,#,#] [-t #] [-u] [-w #] [<in_file> [<out_file>]]
```

<code>-f fontname</code>	specifies the font name to use.
<code>-h</code>	displays the on-line help.
<code>-i</code>	displays the version information.
<code>-r #</code>	specifies the rotation angle.
<code>-s #, #, #, #</code>	specifies the paper size. The # values represent x min, x max, y min, y max.
<code>-t #</code>	specifies the font size.
<code>-u</code>	specifies that the program generated initialization string is not to be used.
<code>-w #</code>	specifies the line width.
<code><in_file></code>	reads input from the file <code><in_file></code> . The default is to read from <code>stdin</code> .
<code><out_file></code>	writes output to the file <code><out_file></code> . The default is to write to <code>stdout</code> .

You can add more initialization commands to the file by defining the environment variables `ALIAS_PSLOT_INIT` and `ALIAS_PSLOT_INIT2`.

pst, pstget

Platforms

IRIX

Purpose

You use **pst** and **pstget** to create simple editor windows of the type used by StudioTools.

Overview

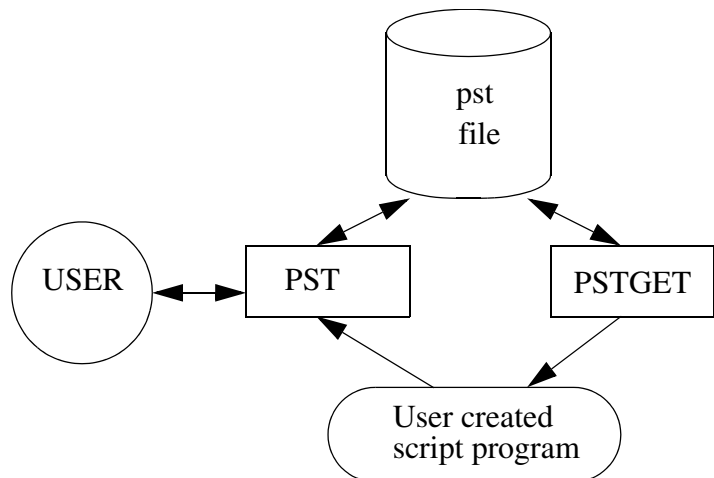
The PST editor system

The PST editor system comprises two stand alone programs: **pst** and **pstget**.

You should be familiar with programming concepts and know how to write UNIX shell scripts.

The editor windows created are of the *form* type. The editors always contain Save and Cancel buttons and can contain many editor fields, which can be interdependent.

This high-level diagram shows how a PST editor is used:



A PST file contains a description of the fields to be edited, and the current value of each field.

Launching an editor

To allow you to interact with the editor described by the PST file, the script launches the `pst` program, providing it with a specified path on the command line.

You must have write permissions for the `pst` file to make an editor window.

Description

The usage statement for `pst` is

```
$ALIAS_LOCATION/bin/pst <-geometry WxH+X+Y>  
PST_file_name
```

If `-geometry` is specified, the window is opened as a `WxH` pixel window, and located at the coordinates specified by `X` and `Y`.

The path to the `pst` file must be specified.

If you exit from `pst` by clicking the `Save` button, the return code is 0. If you exit from `pst` without clicking `Save`, the return code is 1. If an error occurs, the return code is a negative integer.

Querying values from an editor

A script can query values from an editor using `pstget`.

`pstget` accepts a specified `pst` filename as an argument and a symbol name specifying the field to be queried and then prints the current value of the field to the console. The output can be saved in the script that you created.

The usage statement for `pstget` is

```
$ALIAS_LOCATION/bin/pstget PST_file_name  
symbol_name
```

How to write PST files

You can format a `pst` file in any way you choose because white spaces (tabs, spaces, new lines) are irrelevant. When you click

the Save button, the file is recreated from scratch by a code generator in pst.

Edit window title

Each file must start with a name for the table (the string <Name> appears as the title of the editor window).

```
profile "<Name>" {  
LIST  
}
```

LIST is a list of entries. Each entry can be a block name (a group in the editor window) or a standard entry.

Blocks

A block, or group, is an open/close section of an editor window.

The usage statement for a group is:

```
group "<GroupName>" {  
LIST  
}
```

OR

```
group = "<GroupName>" {  
LIST  
}
```

A <GroupName> must be entered because it appears as the title for the block in the editor window.

If "=" is included in the statement, the block starts open. If "=" is not included, the block starts closed. Individual entries

In each of the following entries, if data replaces entry, the particular field does not appear in the editor window, but rather in the table. You can determine the value of the entry with pstget.

Visible comments

To insert comments in the editor window, use the following:

```
annotation( "<text>" );
```

The <text> string appears right justified.

Boolean entries (ON | OFF)

To insert Boolean entries in the editor window, use the following:

```
entry( "<Name>", type=boolean, value="<X>" );
entry( title="<Title>", "<Name>", type=boolean,
value="<X>" );
```

<Title> is optional and appears as the <Name> for the field in the editor window.

<Name> can be used to query and to refer to the value of the entry.

<X> is the default value.

0 is OFF and 1 is ON.

Integer sliders

To create an integer slider, use the following:

```
entry( "<Name>", type=integer, value="<val>" );
entry( title="<Title>", "<Name>", type=integer,
value="<val>" );
entry( "<Name>", type=integer {range "<low>" to
"<high>" }, value="<val>" );
entry( title="<Title>", "<Name>", type=integer
{range "<low>" to "<high>" }, value="<val>" );
```

If range is not specified, it defaults to [0,100].

<low>, <high>, and <val> should be integers. They represent the low and high boundary for the slider and the initial value.

Floating point sliders

To create a floating point number slider, use the following:

```
entry( "<Name>", type=float, value="<val>" );
entry( title="<Title>", "<Name>", type=float,
value="<val>" );
entry( "<Name>", type=float {range "<low>" to
"<high>" }, value="<val>" );
entry( title="<Title>", "<Name>", type=float {range
"<low>" to "<high>" }, value="<val>" );
```

If range is not specified, it defaults to [0.0,100.0].

<low>, <high> and <val> should be floating point numbers. These values represent the low and high boundary for the slider and the initial value.

Radio pop-up menus

To create a radio pop-up menu, use the following:

```
entry( "<Name>", type=choice { "<Name_0:symbol_0>",  
"<Name_1:symbol_1>", ... },  
value="<Name_x:symbol_x>" );  
entry( title = "<Title>", "<Name>", type=choice  
{ "<Name_0:symbol_0>", "<Name_1:symbol_1>", ... },  
value="<Name_x:symbol_x>" );
```

The value specifies the initial value and it should be the same as one of the entries in the choice.

String entry

To create a string entry in the editor window, use the following:

```
entry( "<Name>", type=string, value="<text>" );  
entry( title="<Title>", "<Name>", type=string,  
value="<text>" );
```

<text> appears as the initial value for the string entry.

Dependencies (conditional entries)

Editor windows look different depending on some of the values in the table. For example, you want to display the Pivots option in extrude only if extrude tube is chosen, because extrude flat does not use the Pivots option.

To make editors depend on the values in the table, use the test entry as follows:

```
test( { "<Name1>" * "<value1>", "<Name2>" *  
"<value2>" }, { ... }, ... );
```

* can be = or ! to signify equality or inequality. <Name> matches one of the <Name> entries in the table. The string right after the optional title=Title.

<value> is one of the possible values for that entry.

The conditional entry should be right before the entry for which the test is being done.

For example:

```
entry( watch, title="Extrude Tube",
"mo_extrude_tube", type=boolean, value="0" );
...
test( {"mo_extrude_tube" = "1"} );
entry( watch, title="Extrude Pivots",
"mo_extrude_pivots",
type=choice { "COMPONENT:comp", "CLOSEST:clos" },
value="CLOSEST:clos" );
...
test( {"mo_extrude_pivots" ! "clos"} );
something else that appears only if closest is NOT
chosen as the pivot option.
```

In general, the line

```
test( { a = b, c ! d }, {e = f}, {g ! k, h = m} );
```

says that { (a equals b) OR (c does not equal d) } AND { e equals f } AND { (g does not equal k) OR (h equals m) }. This should be general enough for you to convert anything into this form.

To make this work, we tell the table that the value of the `mo_extrude_tube` variable is important as something else depends on it. This is what the `watch` keyword means.

You can put test entries before groups in which the whole block appears or does not appear depending on the condition tested.

Example

The Editor Window

The following is an example file for a curve rebuild editor window. You can copy this code and see what the result would look like by running `pst /fullpath/filename`.

```
profile "Rebuild Curve" {
entry( watch, title = "Rebuild Type",
"mo_rblld_type",
type=choice{"MATCH:match", "UNIFORM:uniform", "CURVAT
URE:curvature"},
value="UNIFORM:uniform" );
```

```

test( { "mo_rblld_type"! "match" } );
entry( title = "Tolerance", "mo_rblld_tolerance",
type=float {range "1e-05" to "0"}, value="0.01" );

test( { "mo_rblld_type"="uniform" } );
entry( watch, title="Keep Number of CVs",
"mo_rblld_keep_cvcs",
type=boolean, value="0" );

test( { "mo_rblld_type"! "match" },
{ "mo_rblld_type"! "uniform", "mo_rblld_keep_cvcs"="0" }
);
entry( watch, title = "Max Spans Type",
"mo_rblld_spans_flag",
type=choice{ "RELATIVE:relative", "ABSOLUTE:absolute"
},
value="RELATIVE:relative" );

test( { "mo_rblld_type"! "match" },
{ "mo_rblld_type"! "uniform", "mo_rblld_keep_cvcs"="0" },
{ "mo_rblld_spans_flag"="absolute" } );
entry( title = "Max Spans", "mo_rblld_spans_val",
type=integer{range "1" to "50"}, value="3");

test( { "mo_rblld_type"! "match" },
{ "mo_rblld_type"! "uniform", "mo_rblld_keep_cvcs"="0" },
{ "mo_rblld_spans_flag"="relative" } );
entry( title = "Max Spans Factor",
"mo_rblld_spans_fact",
type=float {range "0.1" to "10"}, value="1.0");

test( { "mo_rblld_type"! "match" } );
entry( watch, title = "Change Degree",
"mo_rblld_deg_flag", type=boolean, value="0" );

test( { "mo_rblld_deg_flag"="1" },
{ "mo_rblld_type"! "match" } );
entry( title = "Degree", "mo_rblld_degree",
type=integer {range "1" to "9"}, value="3" );
}

```

The above code creates an editor window that looks like the following:

Querying a value

To query a value (such as `mo_rblld_type`), from the above table, assuming it was called `/usr/tmp/table`, the usage statement is as follows:

```
pstget /usr/tmp/table mo_rblid_type
```

In this example, provided that no fields had been changed, the following would be written to stdout:

```
uniform
```

To capture this value into a c-shell variable called `buildtype`, use the following:

```
set buildtype=`pstget /usr/tmp/table mo_rblid_type`
```

The single quotation marks above are a forward single quotation mark (to the left of the 1 key on an SGI keyboard).

rb_stereo

Platforms

IRIX

Purpose

rb_stereo combines a stereo pair of images into a single image file suitable for viewing with red & blue eye glasses.

Overview

rb_stereo is useful for previewing stereo images on a standard display. All that is required are a stereo pair of images, **rb_stereo**, and a pair of red/blue or green/red "3-D" stereo glasses.

rb_stereo accepts a single stereo pair, or a sequence of stereo pairs.

rb_stereo does a luminance calculation on each image to reduce it to a grey-scale, 8-bit image. The 8-bit channels are then combined into a single image file by assigning the 8-bits to the red, green, or blue channel of the new file as specified by the user options.

rb_stereo maintains the image file type between the input files and the output files. For example, if SGI image files are used as input, then SGI image files are output.

Description

```
rb_stereo [-a # # #] [-b] [-g] [-h] <infile> <outfile>
```

-
- | | |
|----------|--|
| -a # # # | converts an animation, specifying the startframe, endframe, and byframe. The #s must be integers. |
| -b | blends in a bit of green in order to improve red/blue viewing. This is necessary to improve the appearance of the images since the blue gun of the monitor does not exactly match the blue on most red/blue glasses. |
-

-g	creates a red/green stereo pair for use with red/green glasses. This option precludes the -b option.
-h	displays the on-line help.

Examples

If you want to combine the files `image_left` and `image_right` into a single red/blue stereo pair image with a bit of green blend, the command would be:

```
rb_stereo -b image image.3d
```

This command creates a file called `image.3d` in your current directory.

If you have the following animation sequence:

```
pix/image_left.1  pix/image_right.1  
pix/image_left.2  pix/image_right.2  
pix/image_left.3  pix/image_right.3
```

the command to create a sequence of green/red stereo images is as follows:

```
rb_stereo -g -a 1 3 1 pix/image pix/image
```

This command creates files called `pix/image.1`, `pix/image.2`, and `pix/image.3` in your current directory.

renderer/raytracer/powercaster/powertracer

Platforms

IRIX, Windows

Purpose

The renderer creates raycast or raytraced image files from StudioTools Scene Description Language files while in a command line. The StudioTools render programs create image files as follows:

- **renderer**: raycasts images
- **powercaster**: multi-threaded version of the renderer images (purchaseable)
- **raytracer**: raytraces images
- **powertracer**: multi-threaded version of the raytracer images (purchaseable)

StudioTools Renderer for Windows is a command-line tool for rendering Alias files. The rendered files produce a sequence of images that can be played back as an animation.

The functions **powercaster** and **powertracer** might not be present if they have not been purchased for your system.

Description

All four render programs use the same arguments. The usage statement for **renderer**, **raytracer**, **powercaster**, or **powertracer** is described below.

```
<command> [-a#] [-b#] [-B#] [-c <quantized_output_file>]
[-C <color_map_filename>] [-d <filename>] [-e#] [-E#]
[-f <script>] [-h#] [-H] [-J] [-k] [-K#] [-m <filename>] [-p <filename>] [-P] [-q#] [-Q#] [-r#]
[-R#] [-s#] [-S#] [-t#] [-T#] [-w#] [-W#] [-x#] [-y#] [-Y#] [<sdl_filename>]
```

<command>	is renderer or raytracer or powercaster or powertracer
-a#	sets the anti-aliasing level (aalevel) to the integer #. aalevel is the maximum anti-aliasing level per pixel.

-b#	sets the by frame number for animation sequences to the floating point number #.
-B#	sets the by extension for animation sequences to the integer #.
-c <quantized_output_file >	outputs the quantized image to the file <quantized_output_file> after each frame.
-C color_map_filename	uses the SGI image format file <color_map_filename> as the color map to refer to for quantizing after each frame.*
-d <filename>	uses <filename> as the depth filename.
-e#	sets the ending frame number for animation sequences to the floating point number #.
-E#	sets the size extension for animation sequences to the integer # where # indicates the number of 0 padding before the extension number. For example, -E 4 produces file extensions such as <file>.0001 indicating frame 1.
-f <script>	invokes the program <script> after each frame.
-h#	sets the image height for the partial image to be rendered to the integer # without changing the view port. However, this sub-region to be rendered always originates from the lower left hand corner of the image. The integer # moves the origin of this window around the view port.
-H	displays the on-line help.
-J	creates a depth file called 'timing' representing time per pixel.
-k	keeps depth maps in memory after reading them once.**
-K#	turns depth maps on disk usage to #. 0 is OFF. Any number other than zero is ON. ***
-m <filename>	produces a matte file and uses <filename> as the filename.
-n#	sets, to the integer #, the number of processors to render on. This option is only available with powertracer and powercaster.
-p <filename>	uses <filename> as the pix filename.
-P	preserves the non-glowed image unless DOF or Quantize are on. This option allows you to save both a glowed and non-glowed image on disk. The non-glowed image will have the same name as the glowed image, but will have the suffix .ng.
-q#	sets the quiet flag to #. # can be 0, 1, or 2.

-Q#	sets the resolution in the X direction and the view port to the integer #. This option is useful for overriding the resolution specified in a given SDL file. For example, it is useful for switching between rendering NTSC and 1/4 NTSC for a quick preview render.
-r#	sets the aspect ratio to the floating point number #.
-R#	sets the resolution in the Y direction and the view port to the integer #. This option is useful for overriding the resolution specified in a given SDL file. For example, it is useful for switching between rendering NTSC and 1/4 NTSC for a quick preview render.
-s#	sets the starting frame number for animation sequences to the floating point number #.
-S#	sets the start extension for animation sequences to the integer #.
-t#	sets the aathreshold to the integer #. aathreshold is the anti-aliasing threshold value that adaptively super-samples pixels based on color difference. The higher the value, the more sensitive the super-sampling is to color difference.
-T#	sets the number of Y pixels in a tile to the integer #. A tile is a row of pixels to be rendered together. The main reason to use this option is to reduce the amount of memory used by lowering the Y value. This controls the tile size the image is broken up into for rendering and has no effect on the final image or its resolution.
-v	render normally outside of viewport region
-V	render image with hidden lines
-w#	sets the image width for the partial image to be rendered to the integer # without changing the view port. However, this sub-region to be rendered always originates from the lower left hand corner of the image. The integer # moves the origin of this window around the view port.
-W#	sets the ylow for backgrounds to #. The ylow and yhigh define the region of the rendering, specified in pixels, where the background should appear.
-x#	sets the xleft to the integer #. xleft is the left corner of the partial image to be rendered.
-y#	sets the ylow to the integer #. ylow is the left corner of the partial image to be rendered.

`-Y#` sets the yhigh for backgrounds to #. The ylow and yhigh define the region of the rendering, specified in pixels, where the background should appear.

`sdl_filename` sets the SDL filename to a specific filename. If no filename is specified, standard input is used.

* If required, the renderer quantizes images after rendering them. The option `-C` allows a previously generated color table to be used for quantizing images. However, aquant quantizes images much more quickly.

** Generally, when a `depth_input` file is specified for a shadowing spotlight, it is read every frame. `-k` forces the renderer to read the shadow map only once during the first frame of rendering.

*** Rather than editing your SDL files to add `depth_input` or `depth_output` commands to shadowing spotlights, this command line option may be used. When set to a non-zero value (for example `-K 1`), the renderer automatically creates depth output files named after the spotlights in your current directory, or if these files already exist, they are used.

Using any of the above options overrides any equivalent SDL keyword settings in the SDL file.

Examples

The following are examples showing the uses of the rendering utilities.

This command:

```
renderer testframe.sdl
```

renders an SDL file called `testframe.sdl`. All parameters and keywords set within the SDL file will be used. If any of the keywords is missing and requires a value, the default value for that parameter is used.

This command:

```
renderer -a0 -s1 -b2 -e20 -p testpix -q0 -h512 -w512 -x0 -y0 scene.sdl
```

renders an SDL file called `scene.sdl`. The scene to be rendered is an animation.

Anti-aliasing is turned OFF by setting the `aalevelmax` to 0. The animation specified in the SDL file is overridden by the specification of `-s1 -b2 -e20`, which renders with a frame step of 2 the first twenty frames of animation. Normal messages are output, and the test image size is 512x512 pixels square. The animation sequence is output to a series of files starting with the name `testpix`.

This command:

```
rendererer -s 1.5 -e 3.5 -b .25 -S 1 -B 2 -E 3 sdl/foo
```

produces the following:

```
foo.001 (a snap shot of the animation at time 1.5)
foo.003 (a snap shot of the animation at time 1.75)
foo.005 (a snap shot of the animation at time 2.0)
...
foo.019 (a snap shot of the animation at time 3.25)
foo.021 (a snap shot of the animation at time 3.5)
```

Assuming a 512x512 image, the following command:

```
rendererer -h 255 -w 255 -x 255 -y 255 sdl/foo
```

produces an image of the top right 255 x 255 pixel region of the original image.

Important notes

Saved geometry is not compatible with motion blur.

You cannot stop and start a render process that uses saved geometry and retain the saved geometry.

Saved geometry uses a significant amount of memory. You should have at least 100 megabytes of swap space available before attempting to use it.

Alias backgrounds interpolate between the top and bottom of the current rendering pixel span in Y. If you are rendering only a sub-region, this might not be what you intend, because the background is repeated many times in Y. Therefore, use the options `-W` and `-Y` to specify a final resolution from which StudioTools can calculate background positioning.

File formats

SDL files are binary, and are compatible across platforms. To edit an SDL file, you must first convert it to text using the standalone utility *bsdl* (page 33). The TIFF file format is an image display format that is also compatible across platforms. All other image formats are automatically read and saved in SGI image format.

You must uncompress any texture-map files before you render.

renderit

Platforms

IRIX

Purpose

renderit is a shell script that executes an SDL script. **renderit** copies files to the proper locations and can start a render on a remote machine.

Description

```
renderit [-e <engine:loc>] [-f <pixfile>] [-r <renderer>] [-t  
<targ>] [<sdlfile>]
```

-e <engine:loc>	specifies where the rendering is to be done. If this option is not used, the rendering is done on the local host in the current directory.
-f <pixfile>	specifies the name to give the resulting pix files.
-r <renderer>	specifies which renderer should execute the SDL script. The renderer can be one of the following: renderer raytracer powercaster powertracer. renderer is the default option.
-t <targ>	saves the rendered images in the specified file. <targ> can be in host:pathname format or pathname format. If this option is not used, the rendered images are left where they are rendered.
<sdlfile>	specifies the input files to use. If nothing is specified, the standard input file is used by default. Note that this must be the last option specified.

There are a few places in the `renderit` shell script where the standard StudioTools layout of projects is assumed. This is unfortunate, but necessary to make things work with StudioTools. The major assumption is that the renderer is running in a project directory (or at least in a directory with a `pix` subdirectory).

Because `renderit` is a shell script, it can be customized. However, we recommend that you give the customized version a different name so that the StudioTools interactive application runs the `renderit` script as supplied.

setupacct

Platforms

IRIX

Purpose

setupacct is used to customize pre-existing user accounts or create new user accounts to use StudioTools.

Usage

`/usr/aw/alias/bin/setupacct`

Description

Setupacct is used to update or create a user account for use with StudioTools. Creating a new account requires superuser status.

When run by an ordinary user, **setupacct** prompts the user to customize their account to use StudioTools. If they respond “yes”, the account is customized.

If **setupacct** is run by the superuser, it prompts for the name of a user account. If the specified account does not exist, it is created and customized to use StudioTools. If the user account exists, **setupacct** prompts the user to customize the specified account to use StudioTools. If they respond “yes”, the account is customized.

If an existing user account is customized to use StudioTools, some of the “dot” files in the account (i.e.: `.cshrc`, `.login`) may be moved aside in favor of the customized StudioTools software versions. These files are “saved” in a subdirectory of the user account named `.env_MMMDD`, where MMM are the first three letters of the current month and DD is the day of the month. For example, `.env_Jan06`.

Once a user account has been updated, the user must log out and back in again to make changes take effect.

The user’s environment might need to be customized further. For example, if users wish to place the StudioTools working

area somewhere else than their home directories, they will need to edit their `.cshrc` file and change the value of `ALIAS_WORKENV`. Other customizations might be necessary if users wish to use their accounts for other than working with the interactive version of StudioTools. See the *Installation Notes* for more information.

showstereo

Platforms

IRIX

Purpose

showstereo is used to view stereo pix files on StudioTools systems equipped with stereo viewing hardware by Stereographics Corporation. Note that if the images are shown on other monitors, they are likely to appear half-height. To view the image on a standard monitor, you will need to interlace the images.

Description

This usage statement is to view a pair of pix files that have been rendered with the stereo keyword in SDL:

```
showstereo [-h] [-i] [-s #] <left> <right>
```

-h	displays the help messages and then quits.
----	--

-i	identifies the version and then quits.
----	--

-s #	toggles the SGI monitor in and out of 60Hz mode. 1 is on and 0 is off. The default is 1, on. If # is 0, the SGI monitor displays both images one on top of the other.
------	---

<left> and <right>	specifies the filenames assigned to the left and right images.
--------------------	--

Images that are smaller than full resolution (1280x480) are enlarged to fill the screen by pixel replication. Images larger than full screen resolution are truncated with as much as possible of the picture from the upper left corner preserved.

Platforms

IRIX, Windows

Purpose

Converts ISO10303 files to Alias wire files, specifically:

- application protocols ISO10303-203, or Configuration Controlled Design, conformance classes 1 to 4, and
- ISO10303-214, or Core Data for Automotive Mechanical Design Process, conformance classes 1 and 2.

The import and export of this data is supported via ISO10303-21 Physical File exchange.

Description

StToAI [-s] <infile> <outfile>

-s	does not stitch the model on input. (The default is to stitch the model.) This flag does not exist on Windows versions of the utility.
<infile>	specifies the STEP file to use as input. If not specified, input comes from stdin.
<outfile>	specifies an Alias wire file to write output to. If not specified, output goes to stdout.

systemInfo

Platforms

IRIX

Purpose

systemInfo displays information about your system in a window.

Overview

The preferred ways of getting the same information are by using [Help > About StudioTools](#) within StudioTools or the stand-alone `getid` at the UNIX shell level.

For further information about using `getid`, see [getid](#) in this manual.

Description

`systemInfo`

An information window is displayed. It varies depending on the operating system and hardware that you are using.

UGToAI, UGToAI19, UGToAI20, UGToAI21

UGToAI

Platforms

- UGToAI: for IRIX
- UGToAI19, UGToAI20, UGToAI21: Use directly for the Windows platform. In IRIX, these executables are called automatically by UGToAI.

Purpose

Convert Unigraphics files to StudioTools wire files.

Description

UGToAI [options] <infile> <outfile>	
(For Windows, substitute UGToAI with UGToAI19, UGToAI20, or UGToAI21.)	
infile	The name of the input Unigraphics file.
outfile	The name of the output StudioTools wire file.
Options:	
-i	Input Unigraphics file.
-o	Output Unigraphics file.
-s	Do not stitch model on input. Default is to stitch.
-e	Echo logfile to console.
-l	Create extended logfile.
-g	Convert categories. Default is not to convert.

vda_info

Platforms

IRIX

Purpose

vda_info lists the number of occurrences of entities in the specified VDAFS files.

Description

```
vda_info <file>
```

where <file> is any VDAFS format file.

Example

The following is an example of a vda_info usage statement:

```
vda_info sample.vda
```

The following information is displayed on the screen:

```
Summary of file sample.vda
```

```
*****
```

Entity Name	Number of Occurrence
SURF	108
CURVE	231
CONS	336
FACE	72
number of sets	5

wrl

Platforms

IRIX

Purpose

wrl is used to view image files.

Overview

wrl allows interactive placement and display of one or more image files.

Description

```
wrl [-c] [-C] [-f] [-F] [-h#] [-i] [-k] [-m] [-n] [-o] [-r#] [-s#] [-t#] [-w#] [-x#] [-x] [-X#] [-y#] [-Y#] [-display hostname:0.0] [files]
```

-c	draws the images sequentially in the same window.
-C	is the same as option -c, but cycles through the images forever. Press the Esc key to stop the cycling.
-f	fits the image to the window size. The image is full screen size if -w and -h options are not specified.
-F	is the same as option -f but it does not fit the image to the window if the image is smaller than the window. If the image is smaller than the window, -F just centers the image.
-h#	specifies # as height of image shown.
-i	identifies the version number of wrl and then quits.
-k	redraws images instead of closing.
-m	shows only the alpha channel of a TIFF, TIFF16, or RLA image. If the image file does not have an embedded alpha channel, the -m option is ignored.
-n	draws no borders around window.
-o	stops the default action of periodically re-reading incomplete images.
-r#	specifies # as the scaling factor.
-s#	scrolls the image into a window in steps of a specified number of pixels.
-t#	times out after a specified number of seconds. This option is useful with option -C.

<code>-w#</code>	specifies # as the width of image shown.
<code>-x#</code>	specifies # as the pixel to be shown on the farthest left.
<code>-X#</code>	specifies # as the x position of the window.
<code>-y#</code>	specifies # as the bottom pixel shown.
<code>-Y#</code>	specifies # as the y position of the window.
<code>- display hostname :0.0</code>	shows the image on a specified display. You can have other options before or after this option, but not between the display and the hostname. You must have usage permission for the display you want to show the image on. See the UNIX manual page for xhost for more information.
<code>files</code>	specifies the files to display. If no files are specified, stdin is used.

wrl generates a red square the size of the image to be shown. Move the mouse to place the red square where you want it and click the mouse button. The image is then displayed. Continue placing the red squares and displaying the images until all the files in the list have been placed on the screen.

If you type `r`, `g`, `b`, or `a` in a wrl window, wrl displays only the corresponding channel: red, green, blue, or alpha. Pressing any key other than `r`, `g`, `b`, or `a` toggles the wrl window back to red, green, blue display mode.

If the image file is only 3 channels, red, green, and blue, typing `a` toggles the window back to red, green, and blue display mode.

Once an image is displayed, it can be removed from the screen by clicking the mouse anywhere inside the image.

Pressing `Esc` with the cursor in any of the images removes all the image windows.

If the image to be displayed is exactly the same size as the console screen, it is placed so that it entirely fills the screen. The screen cursor is also blanked.

If the image is larger than the screen, the bottom left corner of the image is displayed.

Examples

The following is an example of a usage statement for wrl:

```
wrl -F-C -t30 <Files>
```

In this example, using the `-C`, `-F`, and `-t` options together cause `wrl` to continuously display images on the full screen, resizing them as necessary, and cycling to the next image in `<file>` every 30 seconds.

Press the `Esc` key to stop the images from cycling and to clear the screen.

The following is an example of a usage statement for `wrl`:

```
wrl -display remotehost:0.0 <Files>
```

In this example, `wrl` shows the image on another display. Alternatively, you can set your `DISPLAY` environment variable.

Index

A

- Abekas 64
- Alias batch file translator 10
- Alias plot files 81
- Alias preference variables 52
- AliasBatch 10
- AliasToRenderman 12
- AIToC5 21
- AIToCa 20
- AIToIv 24
- AIToObj 27
- AIToSt 30
- AIToUG 31
- AIToUG utility 31
- animation viewer 43
- Apple QuickTime 45
- ASCII SDL 33

B

- beep 32
- binary SDL 33
- block ordered texture (BOT)
files 70
- bsdl 33

C

- C5ToAI 35
- CAI format 20, 34
- CATIA Part (.CATPart)
format 21, 35
- CATIA product (.CATProduct)
document format 35
- CATIA/Alias Interoperability
format 20
- CaToAI 34

- command line programs 1
- command line utilities
 - on IRIX 6
 - on Windows 6
- conversion of image formats 61
- create new user accounts 100
- customize pre-existing user
accounts 100

D

- dxfl entities 38
- dxfl_info 38

E

- editor windows, creation of 82
- Explore 64

F

- FieldAssembler 40
- findit 42
- FLEXIm 69
- flipbook 43
- fmovie 45
- from100to97 47
- fstats 50

G

- get_alias_variable 52
- getid 54
- GIF 64

H

- host machine processors 71

- hp_gl2plotf 57
- hp_glplotf 57
- HP-GL2 plotters 57

I

- iges entities 59
- iges_info 59
- image format conversion 61
- imgcvt 61
- interlaced images 40
- ISO10303 format 28, 103
- IvToAI 66

J

- JPEG 64

K

- Kodak Cineon 64

L

- license file 67
- lmgd 67
- lmutil 69
- Lucas Film format 64

M

- makebot 70
- Maya IFF 64
- multi-processor machines 71

N

nprocs 71

O

OBJ format 27, 72

ObjToAI 72

P

picture file, locating 42

pim 73

pix files

 comparing 77

pixdiff 77

Pixibox PXB 64

plug-in manager 73

PostScript 81

powercaster 92

powertracer 92

PPM raw/ascii 64

print_wire_header 79

Prisms 64

psplotf 81

pst 82

pstget 82

PXN 64

Q

Quantel 64

R

raycaster 92

raytracer 92

rb_stereo 90

red/blue stereo images 90

renderer 92

renderit 98

Rendeman Interchange

 Bytestream (RIB) format 12

S

scan-line image statistics 50

setupacct 100

SGL 64

SGL Inventor format 24, 66

SGL movie format 45

showstereo 102

Softimage 65

stereo pair 90

Stereographics Corporation 102

StToAI 103

StudioTools 9.7 wire file,
 converting 47

StudioTools version 54

systemInfo 104

T

Targa 65

TIFF 65

U

UGToAI 105

UGToAI utility 105

Unigraphics format 31, 105

using command line utilities

 on IRIX 6

 on Windows 6

V

vda_info 106

VDAFS entities 106

view image files 107

view stereo pix files 102

Vista 65

W

Wavefront RLA 65

Windows bitmap 65

wrl 107