

# **PVRShaman**

## **User Manual**

Copyright © Imagination Technologies Ltd. All Rights Reserved.

This publication contains proprietary information which is subject to change without notice and is supplied 'as is' without warranty of any kind. Imagination Technologies and the Imagination Technologies logo are trademarks or registered trademarks of Imagination Technologies Limited. All other logos, products, trademarks and registered trademarks are the property of their respective owners.

Filename : PVRShaman.User Manual.1.0.120.External.doc  
Version : 1.0.120 External Issue (Package: POWERVR SDK REL\_2.10@839132)  
Issue Date : 10 Feb 2012  
Author : Imagination Technologies Ltd

# Contents

<b>1.</b>	<b>Introduction .....</b>	<b>6</b>
1.1.	Software Overview.....	6
1.1.1.	Features .....	6
1.1.2.	Limitations .....	6
1.2.	Document Overview .....	7
1.3.	File Types .....	7
<b>2.</b>	<b>Compatibility .....</b>	<b>8</b>
2.1.	Operating System .....	8
2.2.	Graphics API.....	8
2.2.1.	OpenGL .....	8
2.2.2.	DirectX.....	8
2.3.	Graphics Card.....	8
<b>3.</b>	<b>Installation .....</b>	<b>9</b>
3.1.	From Installer.....	9
3.2.	From GZIP .....	9
3.3.	Additional Requirements .....	9
3.3.1.	PVRTexTool .....	9
3.3.2.	Collada2POD .....	9
3.3.3.	PVRUniSCo.....	9
<b>4.</b>	<b>Using PVRShaman.....</b>	<b>10</b>
4.1.	Adding Materials .....	10
4.1.1.	Using an Existing Effect File.....	11
4.1.2.	No Effects File .....	11
4.1.3.	Single Texture .....	11
4.1.4.	Blank Effects File.....	12
4.1.5.	Existing Library Effects.....	13
4.1.6.	Applying a Material .....	13
4.2.	Building an Effect.....	14
4.2.1.	PFX Blocks .....	14
4.2.2.	Semantics.....	17
4.2.3.	PVRUniSCoEditor .....	17
4.3.	Render To Texture & Post-Process Effects .....	19
4.3.1.	Overview.....	19
4.3.2.	Render-To-Texture.....	19
4.3.3.	Post-Process Effects .....	20
<b>5.</b>	<b>Interface Overview .....</b>	<b>22</b>
5.1.	Main Interface.....	22
5.1.1.	Scene Browser .....	23
5.1.2.	Scene Container.....	23
5.1.3.	Visualization Panel .....	24
5.1.4.	Effects Editor .....	24
5.1.5.	Debug Output Panel.....	25
5.2.	Menus .....	26
5.2.1.	File Menu.....	26
5.2.2.	Edit Scene Menu .....	28
5.2.3.	Edit Shader Menu.....	30
5.2.4.	Build Menu .....	31
5.2.5.	Render Menu.....	31
5.2.6.	View Menu.....	32
5.2.7.	Tools Menu.....	32
5.2.8.	Help Menu .....	32
5.3.	Toolbars.....	33
5.3.1.	Main Toolbar .....	33
5.3.2.	Visualization Panel Toolbar .....	34

5.3.3.	Effects Editor Toolbar.....	36
5.3.4.	Find/Replace Toolbar.....	37
5.4.	Dialogs.....	38
5.4.1.	Material Properties Dialog.....	38
5.4.2.	Add Material Dialog.....	39
5.4.3.	Light Properties Dialog.....	40
5.4.4.	Camera Properties Dialog.....	41
5.4.5.	Preferences Dialog.....	42
5.4.6.	POD Information Dialog.....	43
5.4.7.	Object Data Dialog.....	44
5.5.	Window Configuration Modes.....	45
5.5.1.	Dock Windows (Scene).....	45
5.5.2.	Dock Windows (Editor).....	45
5.5.3.	Floating Scene.....	45
5.5.4.	Floating Windows.....	46
5.5.5.	Split Window.....	46
5.6.	Render Modes.....	47
5.6.1.	Effects.....	47
5.6.2.	Wireframe.....	47
5.6.3.	Wireframe No Effects.....	47
5.6.4.	No Effects.....	48
5.6.5.	Depth Complexity.....	48
5.7.	Navigation Modes.....	49
5.7.1.	Select.....	49
5.7.2.	Pan.....	49
5.7.3.	Rotate.....	49
5.7.4.	Zoom.....	49
5.7.5.	FPS Navigation.....	49
5.8.	Texture Viewer.....	49
<b>6.</b>	<b>Preferences.....</b>	<b>50</b>
6.1.	General Preferences.....	50
6.2.	Viewport Preferences.....	52
6.3.	Compiler Settings.....	53
6.4.	Editor Colours.....	54
<b>7.</b>	<b>Related Material.....</b>	<b>55</b>
<b>8.</b>	<b>Contact Details.....</b>	<b>56</b>
<b>Appendix A.</b>	<b>PVRShaman PFX Semantics List.....</b>	<b>57</b>
A.1.	Attributes.....	57
A.2.	Uniforms.....	57
<b>Appendix B.</b>	<b>Regular Expression Syntax.....</b>	<b>61</b>

## List of Figures

Figure 4-1 The 'Add Material' Window .....	10
Figure 4-2 Blank PFX File .....	12
Figure 4-3 Total Cycle Counts .....	17
Figure 4-4 Per-line Cycle Counts .....	17
Figure 4-5 Profile Output Panel .....	18
Figure 4-6 Post-Process Effects Example .....	21
Figure 5-1 PVRShaman (Split Window View) .....	22
Figure 5-2 Scene Browser .....	23
Figure 5-3 Scene Container .....	23
Figure 5-4 Visualization Panel .....	24
Figure 5-5 Effects Editor .....	24
Figure 5-6 Debug Output Panel .....	25
Figure 5-7 File Menu .....	26
Figure 5-8 Edit Scene Menu .....	28
Figure 5-9 Edit Shader Menu .....	30
Figure 5-10 Build Menu .....	31
Figure 5-11 Render Menu .....	31
Figure 5-12 View Menu .....	32
Figure 5-13 Tools Menu .....	32
Figure 5-14 Help Menu .....	32
Figure 5-15 Open POD .....	33
Figure 5-16 Close POD .....	33
Figure 5-17 Save POD .....	33
Figure 5-18 Reload POD .....	33
Figure 5-19 Add Material .....	33
Figure 5-20 Delete Material .....	33
Figure 5-21 Assign Material .....	33
Figure 5-22 Preferences .....	33
Figure 5-23 Apply Shader .....	33
Figure 5-24 Fit Selected .....	34
Figure 5-25 Fit All .....	34
Figure 5-26 Unhide All .....	34
Figure 5-27 Hide All .....	34
Figure 5-28 Rotate Around Selected .....	34
Figure 5-29 Select .....	34
Figure 5-30 Pan .....	34
Figure 5-31 Rotate .....	34
Figure 5-32 Zoom .....	34
Figure 5-33 FPS Navigation .....	35
Figure 5-34 Camera Selection .....	35
Figure 5-35 Play/Pause .....	35
Figure 5-36 Frame Slider .....	35
Figure 5-37 Current Frame .....	35
Figure 5-38 Time Direction .....	35
Figure 5-39 Open File .....	36

Figure 5-40 Close File.....	36
Figure 5-41 Save File.....	36
Figure 5-42 Save All.....	36
Figure 5-43 Cut .....	36
Figure 5-44 Copy .....	36
Figure 5-45 Paste.....	36
Figure 5-46 Find.....	36
Figure 5-47 Undo .....	36
Figure 5-48 Redo .....	36
Figure 5-49 Cycle Counts .....	36
Figure 5-50 Find/Replace Toolbar .....	37
Figure 5-51 Find/Replace Regular Expression Error.....	37
Figure 5-52 Material Properties Dialog .....	38
Figure 5-53 Add Material Dialog .....	39
Figure 5-54 Light Properties Dialog .....	40
Figure 5-55 Camera Properties Dialog .....	41
Figure 5-56 Preferences Dialog .....	42
Figure 5-57 POD Information Dialog.....	43
Figure 5-58 Object Data Dialog .....	44
Figure 5-59 Dock Windows (Scene) .....	45
Figure 5-60 Dock Windows (Editor) .....	45
Figure 5-61 Floating Scene.....	45
Figure 5-62 Floating Windows .....	46
Figure 5-63 Split Windows .....	46
Figure 5-64 Effects Render Mode.....	47
Figure 5-65 Wireframe Render Mode .....	47
Figure 5-66 Wireframe No Effects Render Mode.....	47
Figure 5-67 No Effects .....	48
Figure 5-68 Depth Complexity .....	48
Figure 5-69 Select.....	49
Figure 5-70 Pan .....	49
Figure 5-71 Rotate .....	49
Figure 5-72 Zoom.....	49
Figure 5-73 FPS Navigation.....	49
Figure 5-74 Texture Viewer.....	49
Figure 6-1 General Preferences .....	50
Figure 6-2 Viewport Preferences .....	52
Figure 6-3 Compiler Settings .....	53
Figure 6-4 Editor Colours .....	54

# 1. Introduction

## 1.1. Software Overview

PVRShaman is a tool for the rapid prototyping, development, and testing of OpenGL ES 2.0, OpenGL, DirectX9 and DirectX10 shaders. It provides a universal shader compiler and editor with syntax highlighting and per line cycle counts for rapid shader development, and a 'Visualization Panel' where the results of shader changes can be seen in real time.

Shader development is done in either PowerVR Effects (.pfx) or Microsoft Effect (.fx) format, while geometry is handled in the PowerVR POD (.pod) format; a file format created using the Collada2POD or PVRGeoPOD file exporters also available from Imagination.

### 1.1.1. Features

- Windows, Linux and Mac OS support
- Geometry input from POD files generated using PVRGeoPOD
- Support for PowerVR FX files (PFX)
- Texture input from PVR files generated using PVRTexTool
- Support for all common texture formats
- Integrated version of PVRUniSCo Editor
- POD viewer
- WYSIWYG concept allows rapid prototyping of new shaders

### 1.1.2. Limitations

- Cannot open multiple .pod files simultaneously
- Cannot merge .pod files
- Does not allow for extra lights, cameras or objects to be added to a .pod, only materials
- The default shader (the shader applied to objects when no shader has been set by the user) only supports one light, for more lights the user must use a shader that supports multiple lights

## 1.2. Document Overview

The purpose of this document is to serve as a complete user manual for the PVRShaman Shader Development Environment. It includes compatibility information, installation instructions, a guide to the functionality of the application and a complete listing of all interface options and preferences.

## 1.3. File Types

Through the course of this document several file types will be used. The most important of these are:

### **PFX (.pfx)**

A file format for storing and setting up shaders and effects with runtime available in the PowerVR Insider SDK Tools and examples in the training course demos. .pfx files are editable in PVRShaman and PVRUniSCoEditor.

### **POD (.pod)**

A file format for storing complete scenes with meshes, lights, animations, materials and references to textures and effects. While .pods are not compressed, they store information in a format designed for speedy deployment to hardware.

### **PVR (.pvr)**

.pvr is a file format for storing API-friendly textures. .pvr files are produced using PVRTexTool and can be compressed as small as 2/4 bits per pixel, while still maintaining a good quality.

### **Collada (.dae)**

Collada is a royalty-free XML file scheme developed by the Khronos group that acts as an interchange format for digital assets and effects.

### **Microsoft Effects (.fx)**

Proprietary file format belonging to the Microsoft Corporation used for setting up effects for use with the DirectX API.

## 2. Compatibility

### 2.1. Operating System

PVRShaman is compatible with Windows 2000 onwards, Linux (requires X11), and Mac OS 10.6 onwards. The X11 environment, an optional install from Apple, is required for the Mac OS version to function.

### 2.2. Graphics API

#### 2.2.1. OpenGL

For the OpenGL and OpenGL ES 2.0 APIs the .pfx file format is used; OpenGL ES1.1 does not use shaders or effects files.

#### 2.2.2. DirectX

PVRShaman uses .fx files for DirectX9 and DirectX10 shaders; these modes are only available on Windows. To use the DirectX10 mode, a DirectX10 capable graphics card is required and the latest DirectX10 runtime must be installed (available from the Microsoft website).

### 2.3. Graphics Card

In order to use PVRShaman as a shader development environment a graphics card that supports the targeted API is required. E.g. If shaders are being written for DirectX10 a DirectX10 capable graphics card is required. For viewing .pod files, an older card can be used. In addition, when targeting the OpenGL ES 2.0 API a graphics card compatible with OpenGL 2.0 is required; and when targeting OpenGL ES 1.x API a graphics card compatible with OpenGL 1.5 is required.



## 3. Installation

### 3.1. From Installer

Download either the PowerVR Insider SDK or the individual PVRShaman package and follow the on screen instructions. Once the package has successfully installed the application will be available in the Windows 'Start Menu'.

### 3.2. From GZIP

Download either the PowerVR Insider SDK or the individual PVRShaman package. Unzip the .tar.gz file, and then untar the .tar file. From the ensuing folder browse to:

```
<SDK_ROOT>\Utilities\PVRShaman\<PLATFORM>\
```

This folder will contain the PVRShaman executable.

### 3.3. Additional Requirements

This section pertains only to additional requirements if PVRShaman is not downloaded as part of the full PowerVR Insider SDK. In instances where the full SDK is installed all of these utilities will be installed and the options for each set appropriately. PVRTexTool, Collada2POD and PVRUniSCo are all available from the PowerVR Insider section of the Imagination website (see section 8 Contact Details for more information).

#### 3.3.1. PVRTexTool

It is possible to plug PVRTexTool into PVRShaman, allowing for images to be directly opened from the Scene Container with a double click; for this to function PVRTexTool must be installed and its location set in 'Path to PVRTexTool' under 'Preferences -> Shaman Preferences'.

#### 3.3.2. Collada2POD

PVRShaman can directly import Collada (.dae) files through the 'File -> Import' option; this functionality uses Collada2POD, a file converter that comes packaged in the PowerVR Insider SDK. This program must be installed and its location set in 'Path to Collada2POD' in 'Preferences -> Shaman Preferences' for this functionality to be available.

#### 3.3.3. PVRUniSCo

In order for the shader editor to correctly output cycle count and register information for the shaders being edited, PVRUniSCo must be installed. Once this is installed, the OpenGL and OpenGL ES compilers must be linked to PVRShaman in 'Preferences -> Compiler Settings', as must the VGP Compiler if VGP information is required.

## 4. Using PVRShaman

### 4.1. Adding Materials

Materials are added to a scene in the same way regardless of whether the scene has been imported from another program or is based on a built-in object.

To add a material to PVRShaman click 'Edit Scene -> Add Material' or press 'Ctrl-M' on the keyboard. This will open the Add Material dialog box.

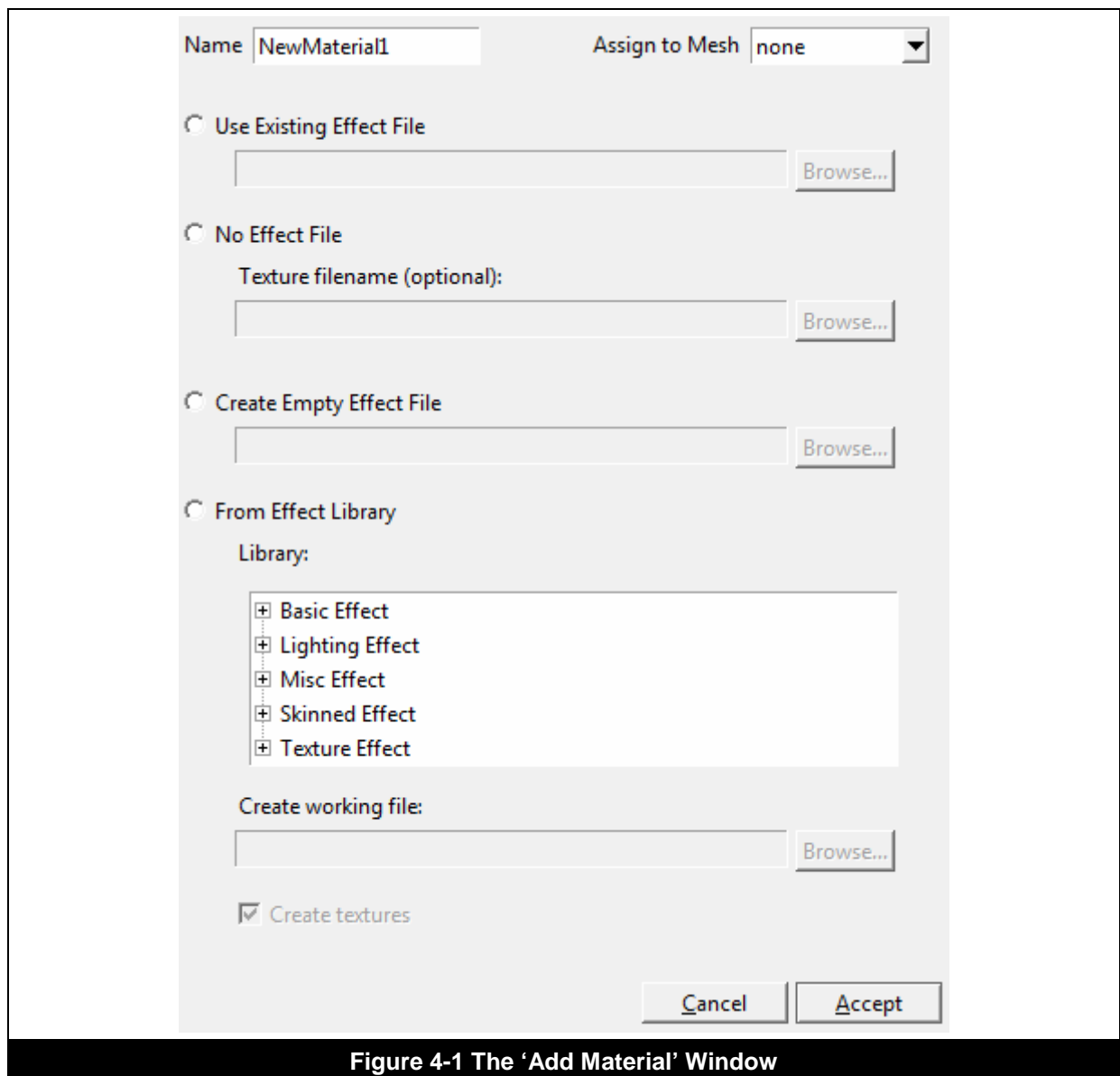


Figure 4-1 The 'Add Material' Window

Give the material a name relevant to what it does (e.g. ParallaxBumpMap). This name will be used to attach the material to objects within the scene.

#### 4.1.1. Using an Existing Effect File

An existing effects file is a .pfx file that contains pre-written shader code. The file may reference a number of textures in the form of .pvr files. These files must be present in the same folder as the .pfx file.

In order to use an existing effects file click the radio button next to 'Use Existing Effects File' and click 'Browse'. This will open an 'open file' dialog box. Browse to the location of the .pfx file, and either double click it, or single click it and click 'Open'. This will return you to the Add Material window. Click 'Accept' to add the effects file to the scene, it will now appear under 'Materials' in the 'Scene Container', the Visualization Panel will lose focus, the Effects Editor will gain it, and the shader code will now be visible.

#### 4.1.2. No Effects File

It is possible to add a material without an effect file. To do this, click the radio button next to 'No Effect File', and then click 'Accept'. The new material will now appear under 'Materials' in the Scene Container. This material can be assigned to an object; the object will retain its original colour but will now be affected by any lights within the scene.

#### 4.1.3. Single Texture

To add a single texture, without an effects file, as a material click the radio button next to 'No Effect File', click 'Browse'. This will open an 'open file' dialog box. Browse to the location of the texture file then either double click it, or single click it and click 'Open', then click 'Accept'.

It should be noted that the .pod file format does not store the full file path to texture files. In order to avoid errors textures should be stored in the same folder as the .pod file that uses them. A warning will appear when using this option on a built-in object or if the texture is not in the same folder as the .pod file you are editing/creating.

#### 4.1.4. Blank Effects File

A blank effects file is a .pfx file with some basic formatting information to help write shaders.

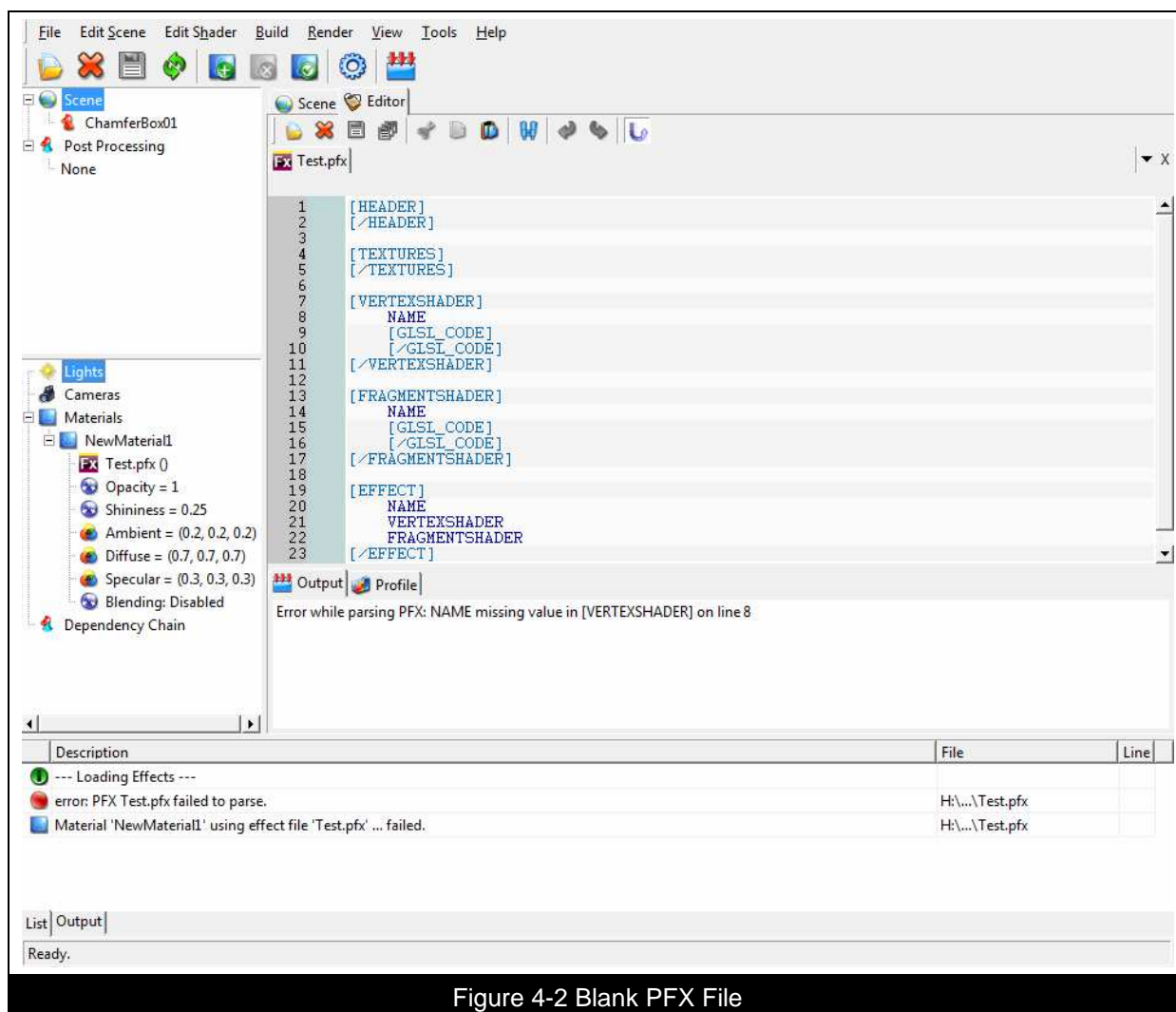


Figure 4-2 Blank PFX File

In order to add a material with a blank effects file click the radio button next to 'Create Empty Effect File', click 'Browse'. This will open a 'save file' dialog box. Using this dialog box browse to a suitable location, enter a name for the file, click 'Save', and then click 'Accept'.

The .pod file format does not store the full file path to effect files. In order to avoid errors, effects files should be stored in the same folder as the .pod file that uses them. A warning will appear when using this option on a built-in object or if the effects file is not in the same folder as the .pod file you are editing/creating.

### 4.1.5. Existing Library Effects

PVRShaman includes a small library of built in effects which can be automatically copied into a new .pfx file. These include:

- Anisotropic Lighting – A form of lighting used to mimic satin surfaces or brushed metal.
- Bump Mapping – A technique to fake complex geometry through the manipulation of reflected light.
- Cell Shading – A type of non-photorealistic rendering used to give a cartoon style effect.
- Complex Lighting – A shader for handling a number of standard lighting situations – this shader is not designed to be optimal for performance.
- Fast Transform and Lighting – An efficient algorithm for basic transformation combined with diffuse and specular lighting calculations.
- Fog – An implementation of linear, exponential and exponential squared.
- Fresnel Reflection – A demonstration of reflections based on the incidence of the view vector taking into account the different refractive indices between two media.
- Iridescence – A shader which mimics the behaviour of light reflecting off a surface covered by a translucent film with variable thickness.
- Refraction – A technique for mimicking the effect of light passing through a translucent surface like glass.
- Texturing – Basic texture functionality.

In order to make use of these examples, from the Add Material window click on the 'From Effect Library' radio button. From the library window select the desired effect and click 'Browse'. A Save File dialog box will open. Using this dialog box, browse to a suitable location; enter a name for the file, click 'Save', and then click 'Accept'.

As has already been stated; the .pod file format does not store the full file path to effect files. In order to avoid errors, effects files should be stored in the same folder as the .pod file that uses them. A warning will appear when using this option on a built-in object or if the effects file is not in the same folder as the .pod file you are editing/creating.

### 4.1.6. Applying a Material

Once a material has been added and appears in the 'Scene Container' it can be applied to a model. This can either be done as part of creating the effect, from the Add Material window, or by right clicking on a model within the 'Scene Browser', clicking on 'Assign Material' and selecting the material you wish to apply.

## 4.2. Building an Effect

The PowerVR Effects (.pfx) format is a small, simple, easy to use effects file format that allows for the declaration of 'application data' semantics so that shaders can be accurately simulated.

A single .pfx file consists of several blocks that describe how a given effect is put together (see Section 4.2.1 PFX Blocks below) and by default consists of a single effect. Multiple effects may exist within a single .pfx file, each with their own effect block; likewise multiple shaders may exist within a single .pfx, each referenced by name. Finally, it is also possible for separate .pfx files to reference the same shaders by placing those shaders in separate files and linking to them from the vertex shader and fragment shader blocks.

### 4.2.1. PFX Blocks

Each .pfx file is broken down into several blocks, each block with a different purpose. A basic effect will consist of a header, an effect block, a vertex shader block, a fragment shader block, and one or more texture and/or target blocks.

#### [HEADER]

```
[HEADER]
    VERSION      01.00.00.00
    DESCRIPTION   header example
    COPYRIGHT     Imagination Technologies
[/HEADER]
```

The header block contains information about the .pfx file being edited. Three keywords are used here:

- VERSION – The version of the PowerVR Effects File Specification the file uses.
- DESCRIPTION – A description of what the effect does.
- COPYRIGHT – A copyright string.

#### [VERTEXSHADER] / [FRAGMENTSHADER]

```
[FRAGMENTSHADER]
    NAME          FragShader
    [GLSL_CODE]
        uniform sampler2D      sampler2d;
        varying highp vec2     texCoordinateMain;

        void main (void)
        {
            gl_FragColor = texture2D(sampler2d, texCoordinateMain);
        }
    [/GLSL_CODE]
[/FRAGMENTSHADER]

[VERTEXSHADER]
    NAME          VertShader
    FILE           VertShader.vsh
[/VERTEXSHADER]
```

The vertex shader block describes a given vertex shader; it contains up to two keywords and one nested block:

- NAME – The name for the vertex shader, for use in the effect block.
- FILE – The FILE tag can be used to specify a file that contains the vertex shader in cases where the GLSL code block is not used.
- [GLSL\_CODE] – Code within the GLSL code block is passed verbatim to the GLSL compiler, and represents the actual shader code to be run.

## [TEXTURE]

```
[TEXTURE]
    NAME                Lena
    PATH                "Lena.pvr"
    MINIFICATION         LINEAR
    MAGNIFICATION        LINEAR
    MIPMAP               NEAREST
    WRAP_T               REPEAT
    WRAP_S               CLAMP
[/TEXTURE]
```

The texture block contains information about a texture to be loaded, and the flags to be used when drawing it. Seven keywords are used:

- NAME – The name of the texture, for use in the effect block.
- PATH – The path to the file the texture block represents. Currently this file must be in the same folder as the .pfx file.
- MINIFICATION – This option states how to handle texture minification, two valid values exist, either LINEAR or NEAREST.
- MAGNIFICATION – This option states how to handle texture magnification, two valid values exist, either LINEAR or NEAREST.
- MIPMAP – This option states how to pick the correct MIP-map, three valid values exist, either LINEAR, NONE, or NEAREST.
- WRAP\_T/WRAP\_S – These two options indicate which wrap mode is to be used for the texture, two valid values exist, either REPEAT or CLAMP.
- VIEW/CAMERA – Used when performing a render-to-texture, these tags indicate which camera should be rendered to this texture. Valid values are 'PFX\_CURRENTVIEW', the currently selected view in PVRShaman, or the name of any camera from the POD file.

When no values are provided the following values will be used:

- MINIFICATION – NEAREST
- MAGNIFICATION – NEAREST
- MIPMAP – NONE
- WRAP\_T/WRAP\_S – REPEAT

## [TARGET]

```
[TARGET]
    NAME                Mix
    SURFACETYPE         RGB888
    RESOLUTION           512 512
    MINIFICATION         LINEAR
    MAGNIFICATION        LINEAR
[/TARGET]
```

The target block contains information about a render target. The targets name is used in an effects block to signify that the effect associated with that block should render to the named off-screen buffer. Five keywords are used in the creation of a target block:

- NAME – The name of the target, for use in the effect block.
- SURFACETYPE – The pixel format of the target, currently the following formats are accepted:
  - RGBA 8888
  - RGBA 4444
  - RGB 888
  - RGB 565
- RESOLUTION – This specifies the resolution of the target buffer in the form '<X-SIZE> <Y-SIZE>', where size is a number of pixels.
- MINIFICATION – This option states how to handle texture minification, two valid values exist, either LINEAR or NEAREST.
- MAGNIFICATION – This option states how to handle texture magnification, two valid values exist, either LINEAR or NEAREST.

**[EFFECT]**

```
[EFFECT]
    NAME Bumpmapping
    VERTEXSHADER VertShader
    FRAGMENTSHADER FragShader

    TEXTURE 0 base
    TARGET COLOR0 rendertarget

    ATTRIBUTE    inVertex    POSITION
    ATTRIBUTE    inNormal    NORMAL
    ATTRIBUTE    inTexCoord  UV
    ATTRIBUTE    inTangent   TANGENT
    UNIFORM      MVPMatrix   WORLDVIEWPROJECTION
    UNIFORM      LightPosition LIGHTPOSMODEL0
    UNIFORM      sampler2D    TEXTURE0
[/EFFECT]
```

The effect block is used by PVRShaman to set up the shader. It defines all the attribute and uniforms to be passed to the shader, a name for the shader and the names of the vertex and fragment shaders that are to be used for this given effect. It can also be used to set annotations, and to set the texture number for a given texture. Seven keywords, and one nested block are available for use:

- **NAME** – The name for the overall effect.
- **UNIFORM** – Used multiple times to set-up a given uniform in the shader. This is written in the form `'UNIFORM <NAME> <SEMANTIC>'` where the name represents the name of the PVRShaman semantic list (see Appendix A. PVRShaman PFX Semantics List).
- **ATTRIBUTE** – Used multiple times to set-up a given attribute in the shader. This is written in the form `'ATTRIBUTE <NAME> <SEMANTIC>'` where the name represents the name of the attribute as it is written in the shader and the semantic represents a semantic from the PVRShaman semantic list (see Appendix A. PVRShaman PFX Semantics List).
- **TARGET** – Used multiple times; this is used to set a target number and name for a given render-to-texture target represented by a [TARGET] block. It is declared in the form `'TARGET COLOR<NUMBER> <NAME>'` where number represents the ID of the target and the name represents the name of the texture as set in the [TARGET] block. Its purpose is to specify the render target to be used by an effect.
- **TEXTURE** – Also used multiple times; this is used to set a texture number and name for a given texture set by a texture block. It is declared in the form `'TEXTURE <NUMBER> <NAME>'` where number represents the ID of the texture and name represents the name of the texture as set in the texture block.
- **FRAGMENTSHADER** – Used to specify the fragment shader to be used for this effect. This is required as multiple fragment shaders can be specified within a single .pfx file.
- **VERTEXSHADER** – Used to specify the vertex shader to be used for this effect. This is required as multiple vertex shaders can be specified within a single .pfx file.
- **[ANNOTATION]** – All text within the annotation sub block will be read into a string, this string is not used in PVRShaman, though it can be used to pass information to other applications that may later read the .pfx file.



### 4.2.2. Semantics

In order for a shader to function its parent application must pass certain values to it; these are uniforms and attributes.

A uniform is a value that is read-only, and does not change during a render, for example light position or colour. Attributes are input values that are different for every vertex, they are also read-only, and are only available in the vertex shader.

From shader to shader many of these values will change, and will represent radically different things. An example might be that one shader requires only the ModelViewProjection matrix, whereas another might need the inverse transpose of the ModelView matrix. Given this wide area of requirement, PVRShaman must have a means to understand what a given uniform or attribute represents so that the value can be provided to the shader when executing. This is the purpose of the PVRShaman semantics, a full list of which can be found in Appendix A. PVRShaman PFX Semantics List.

### 4.2.3. PVRUniSCoEditor

PVRUniSCoEditor is the PowerVR Universal Shader Compiler and Editor, available as both a standalone application included in the PowerVR Insider SDK, and as the PVRShaman effect and shader editor.

PVRUniSCoEditor's main features are:

- syntax highlighting
- on-the-fly cycle counts
- register information

By default the cycle count for the currently selected file (the total of all available shaders, as well as all paths through those shaders) is visible on the top bar.

#### Cycle Counts

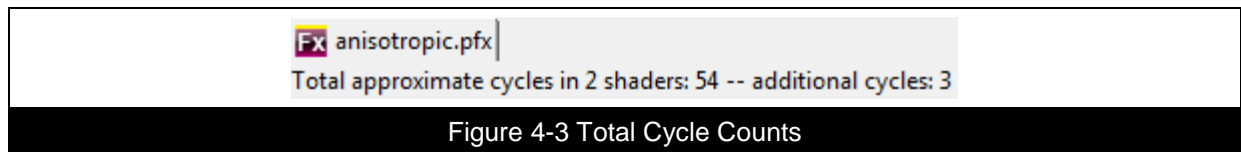


Figure 4-3 Total Cycle Counts

Cycle counts are also available on a per line basis allowing you to quickly spot bottle necks and see the effect of optimizations as they are carried out.

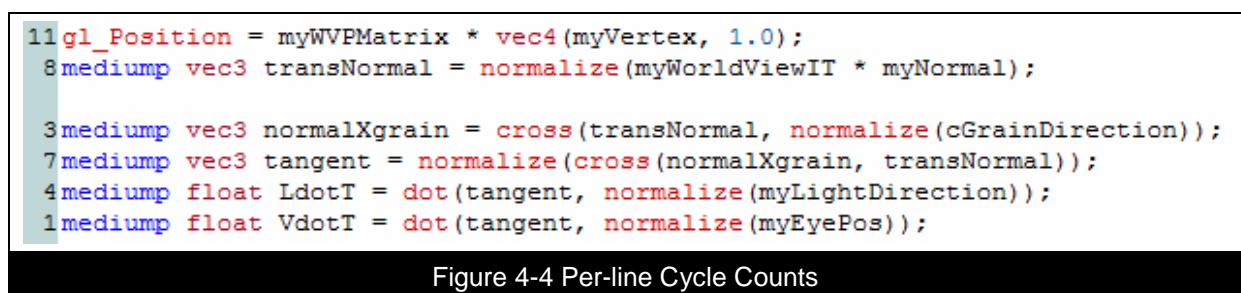


Figure 4-4 Per-line Cycle Counts

It should be noted that these cycle counts are only approximations, and only available once a shader is capable of being compiled (i.e. functional). These cycle counts can be updated in one of several ways, either by clicking the 'Apply Shaders' button on the toolbar or in the 'Build' menu, by ticking the 'Generate cycle counts in background' option in the 'Build' menu, by clicking 'Rebuild Shaders' in the 'Build' menu, or by pressing the 'F5' key.

## Profile Output Panel

### Compiler

This value identifies the GPU name targeted by the compiler.

### Version

This value identifies the compiler version.

### Emulated Cycles Best

This number represents the number of cycles the shader will use when all conditional branches fail and are not processed.

Emulation gives a much more accurate measuring of cycles than the per-line cycle count.

### Emulated Cycles Worst

This number represents the number of cycles the shader will use when all conditional branches succeed and are processed.

In many cases the best and worst cycle counts will be identical.

### Emulated Cycles

This number represents the number of cycles the shader will use, and is only displayed when there are no conditional blocks present.

### Primary Attributes Used

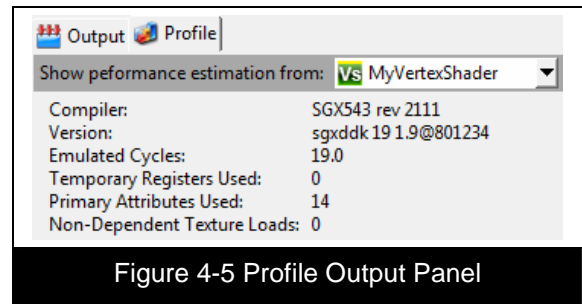
'Primary Attributes' are the number of logical input entities. The number of registers consumed by a primary attribute will depend on the number of elements and on its precision.

For indication only; input data storage typically has 1280 registers allocated (varying from platform to platform). These registers are shared by all the shaders being run at any one time. Running out of registers will force reading and writing from external memory which might affect performance.

### Temporary Registers

'Temporary Registers' are the extra data storage required to process a shader.

As in the case of primary attributes, overflowing the storage allocated might cause degradation in performance. Temporary data storage typically has 384 registers allocated (this number might vary depending on the platform).



## 4.3. Render To Texture & Post-Process Effects

### 4.3.1. Overview

The phrase 'Render to texture' is used to describe any scenario during the rendering of a scene where it is required that rendering be directed to another buffer other than the frame, specifically, to a texture.

Post-processing refers to any full screen technique performed on a scene after the scene has been fully composited and rendered. Instead of rendering 3D objects directly, the scene is first rendered to a texture; pixel shaders are then used to apply filters to the texture before displaying it. Post-processing allows effects to be used that require awareness of the entire image such as:

- High dynamic range
- Bloom
- Motion blur
- Screen Space Ambient Occlusion
- Crepuscular rays
- Film grain simulation
- Depth of field
- Shadow mapping

### 4.3.2. Render-To-Texture

Rendering to a texture is done in much the same way as rendering to the framebuffer. Vertex shader, fragment shader, effect, and texture blocks are created as usual, with the exception that the texture is given a 'VIEW' entry, representing the name of the camera within the loaded POD file whose view is to be rendered into the texture. An example of this can be seen below:

```
[TEXTURE]
    NAME          renderTest
    RESOLUTION     512 512
    MINIFICATION   LINEAR
    MAGNIFICATION  LINEAR
    MIPMAP         NONE
    VIEW          "Camera02"
[/TEXTURE]

[EFFECT]
    NAME          RenderToTextureEffect

    ATTRIBUTE      inVertex          POSITION
    ATTRIBUTE      inTexCoord        UV0
    UNIFORM        sTexture          TEXTURE0
    TEXTURE        0                 renderTest
    VERTEXSHADER   VertShader
    FRAGMENTSHADER FragShader
[/EFFECT]

[VERTEXSHADER]
    NAME          VertShader
    FILE          VertShader.vsh
[/VERTEXSHADER]

[FRAGMENTSHADER]
    NAME          FragShader
    FILE          FragShader.fsh
[/FRAGMENTSHADER]
```

### 4.3.3. Post-Process Effects

Post process effects are created by chaining multiple effect blocks together; using the output of one effect block, set using the 'TARGET' flag, as an input to the next, set using the 'TEXTURE' flag. An example of this, with an explanation of each stage, can be found below:

```
[TEXTURE]
    NAME                Lena
    PATH                "Lena.pvr"
    MINIFICATION        LINEAR
    MAGNIFICATION        LINEAR
    MIPMAP              NEAREST
[/TEXTURE]

[TARGET]
    NAME                InputEffectOutput
    SURFACETYPE         RGB888
    RESOLUTION          512 512
    MINIFICATION        LINEAR
    MAGNIFICATION        LINEAR
[/TARGET]
[TARGET]
    NAME                GreyscaleEffectOutput
    SURFACETYPE         RGB888
    RESOLUTION          512 512
    MINIFICATION        LINEAR
    MAGNIFICATION        LINEAR
[/TARGET]
[TARGET]
    NAME                MixEffectOutput
    SURFACETYPE         RGB888
    RESOLUTION          512 512
    MINIFICATION        LINEAR
    MAGNIFICATION        LINEAR
[/TARGET]

[EFFECT]
    NAME                InputEffect

    ATTRIBUTE           inVertex          POSITION
    ATTRIBUTE           inTexCoord        UV0
    UNIFORM             sTexture          TEXTURE0
    TEXTURE             0                 Lena
    TARGET              COLOR0            InputEffectOutput
    VERTEXSHADER        ScreenAlignedVS
    FRAGMENTSHADER      InputFS
[/EFFECT]
[EFFECT]
    NAME                GreyscaleEffect

    ATTRIBUTE           inVertex          POSITION
    ATTRIBUTE           inTexCoord        UV0
    UNIFORM             sTexture          TEXTURE0
    TEXTURE             0                 InputEffectOutput
    TARGET              COLOR0            GreyscaleEffectOutput
    VERTEXSHADER        ScreenAlignedVS
    FRAGMENTSHADER      GreyScaleFS
[/EFFECT]
[EFFECT]
    NAME                MixEffect

    ATTRIBUTE           inVertex          POSITION
    ATTRIBUTE           inTexCoord        UV0
    UNIFORM             sOriginal         TEXTURE0
    UNIFORM             sGreyscale        TEXTURE1
    TEXTURE             0                 InputEffectOutput
    TEXTURE             1                 GreyscaleEffectOutput
    TARGET              COLOR0            MixEffectOutput
    VERTEXSHADER        ScreenAlignedVS
    FRAGMENTSHADER      MixFS
[/EFFECT]
```

### Step-by-Step Explanation

Assuming that 'MixEffect' has been selected in the Scene Browser the following steps will occur:

1. 'Lena.pvr', a texture file, is used as an input for the 'InputEffect' block.
2. The 'InputEffect' block is run, outputting to 'InputEffectOutput'.
3. 'InputEffectOutput' is used as an input for the 'GreyscaleEffect' block.
4. The 'GreyscaleEffect' block is run, outputting to 'GreyscaleEffectOutput'.
5. 'InputEffectOutput' and 'GreyscaleEffectOutput' are used as inputs for the 'MixEffect' block.
6. The 'MixEffect' block is run, rendering to 'MixEffectOutput'.

### Outputting to the Framebuffer

When a post-process effect is selected from the PVRShaman GUI (see Section 5.1.1 Scene Browser) a dependency tree is built; this tree is used to determine which renders must be run in which order to correctly display the selected effect. Once this effect has been run, its output is displayed in the Visualization Panel.

It should be noted that the post-process effects described in this section cannot be applied to a model, unlike the effects described in Section 4.3.2 Render-To-Texture.

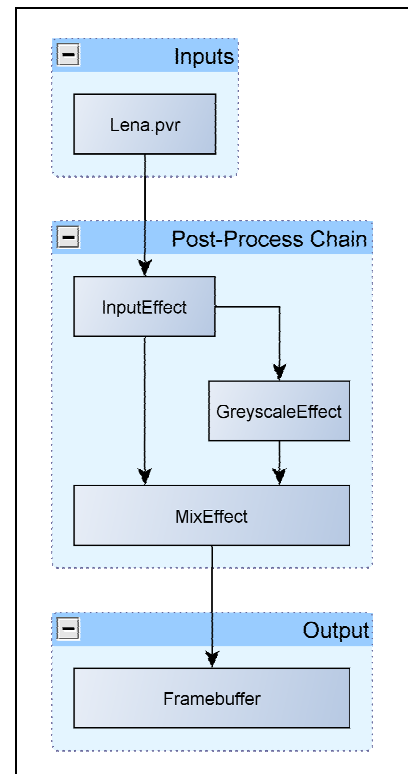


Figure 4-6 Post-Process Effects Example

## 5. Interface Overview

### 5.1. Main Interface

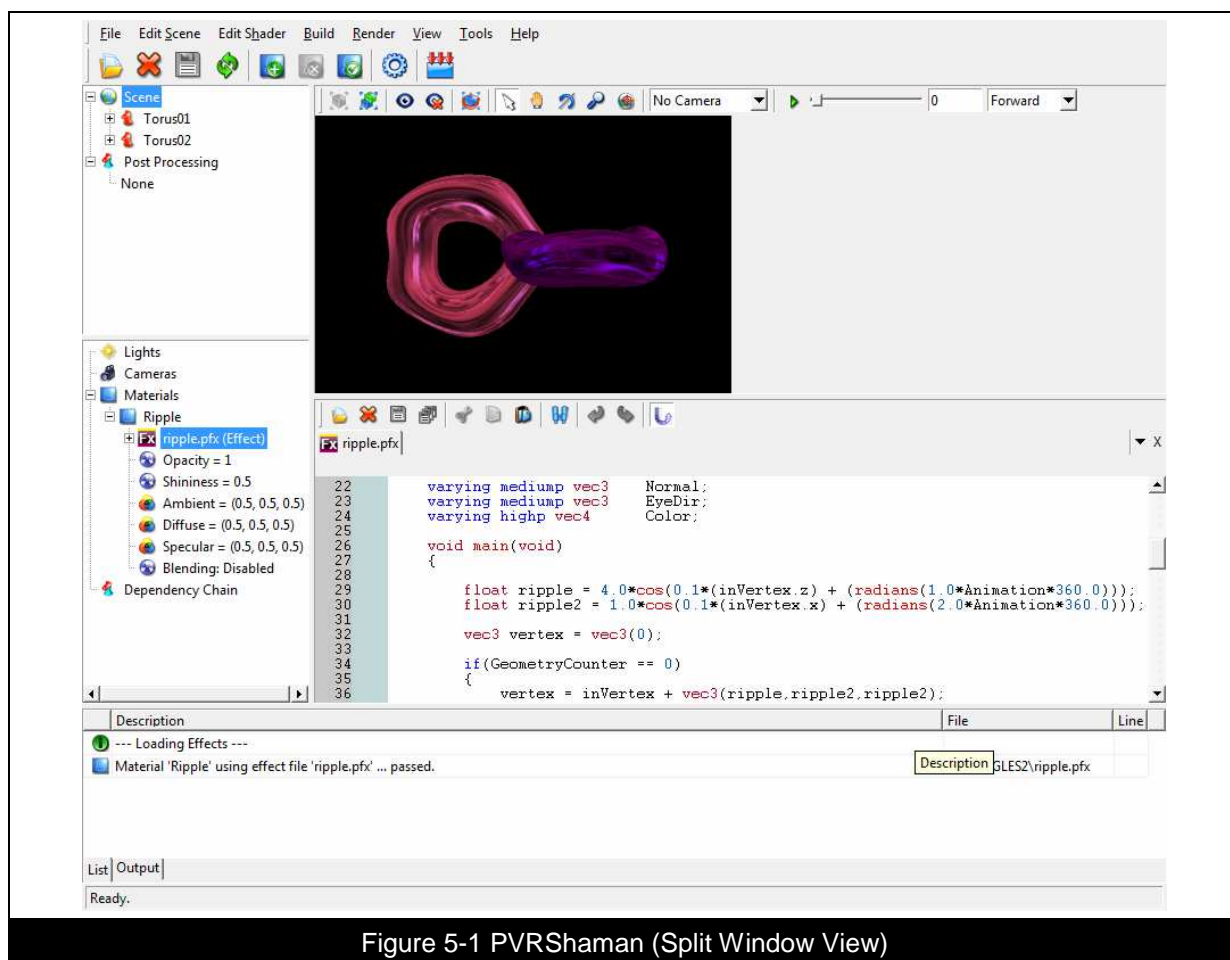


Figure 5-1 PVRShaman (Split Window View)

The PVRShaman interface is composed of several panels which hold information about the current scene, the materials within the scene, and any debug output from those materials.

### 5.1.1. Scene Browser

At the top left is the Scene Browser. This shows all the objects in a scene and any materials applied to its meshes. If a mesh has a parent, this will be reflected in the tree structure. An object can be selected with a single click, highlighting the model, while double clicking the model name will centre the model in the Visualization Panel.

Finally, right clicking the model name will open a context menu with options to assign a material to the currently selected object, open it in the Object Data Dialog, fit the object to the screen, centre the object on the screen, hide the selected object, hide all other objects, or unhide all objects. There are no context menus on right click for lights or cameras, however double clicking a camera will adjust the Visualization Panel to look through the associated camera.

Finally, the Scene Browser contains the post-processing controls. Right clicking on the currently active effect will allow the selection of the post-process (if any exist) that is currently rendered to the Visualisation Panel. If 'None' is selected then no post-processing is performed and the scene is rendered to the Visualisation Panel as normal.

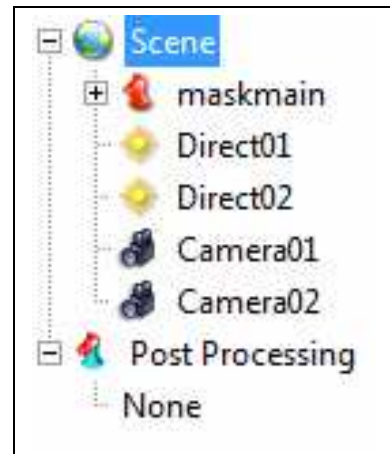


Figure 5-2 Scene Browser

### 5.1.2. Scene Container

Below the Scene Browser is the Scene Container where the components of the scene are stored. Expanding a component will display its properties and double clicking on a property will bring up the Material Properties Dialog, Light Properties Dialog, or Camera Properties Dialog as appropriate.

There are two types of materials: A single textured material, which can either be exported with a scene from 3ds Max or Maya, or setup as described in Section 4.1.3 Single Texture; and a 'shader' material composed of a single .pfx file.

Double clicking on a .pfx file will bring up the integrated PVRUniSCoEditor interface. Expanding the .pfx item will show any external GLSL or texture files used within the .pfx. Double clicking on a GLSL file will open it in the integrated editor. Double clicking on a texture file will open it in PVRTexTool if its location has been set in 'Tools -> Preferences -> Path to PVRTexTool' or in the Texture Viewer if it has not.

Pressing 'Delete' when a material is selected deletes the material. Cameras, lights and meshes cannot be deleted.

Finally, 'Dependency Chain' displays an ordered list of the render dependencies of the currently selected post-process effect.

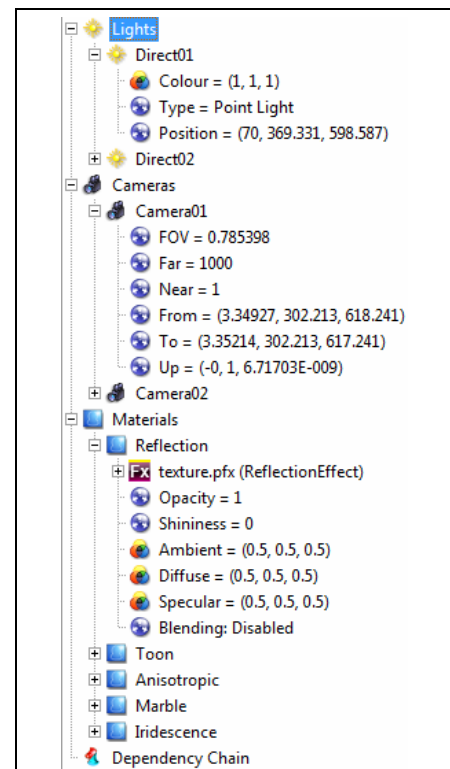


Figure 5-3 Scene Container

### 5.1.3. Visualization Panel

The Visualization Panel is the area that displays the output of the combined scene data and materials; displaying any meshes in the scene with the corresponding materials applied. Lights and camera are also displayed, lights as a small light bulb, cameras as a small green camera. In instances where an object is drawn without a material it will appear entirely grey; this can be due to an error in the material that is applied or due to a lack of an applied material, more information on the cause is available in the Debug Output Panel as described in Section 5.1.5 Debug Output Panel.

Clicking an object in the scene selects it, when an object is selected right clicking will bring up a contextual menu identical to that described in Section 5.1.1 Scene Browser. Finally, scenes can be explored using the mouse and keyboard, for more information see Section 5.7 Navigation Modes.

In Figure 5-1 PVRShaman (Split Window View) the Visualization Panel can be seen at the top of the window; this position is not set and can be adjusted using the 'View' menu.

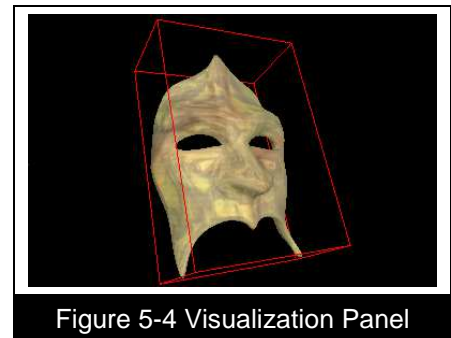


Figure 5-4 Visualization Panel

### 5.1.4. Effects Editor

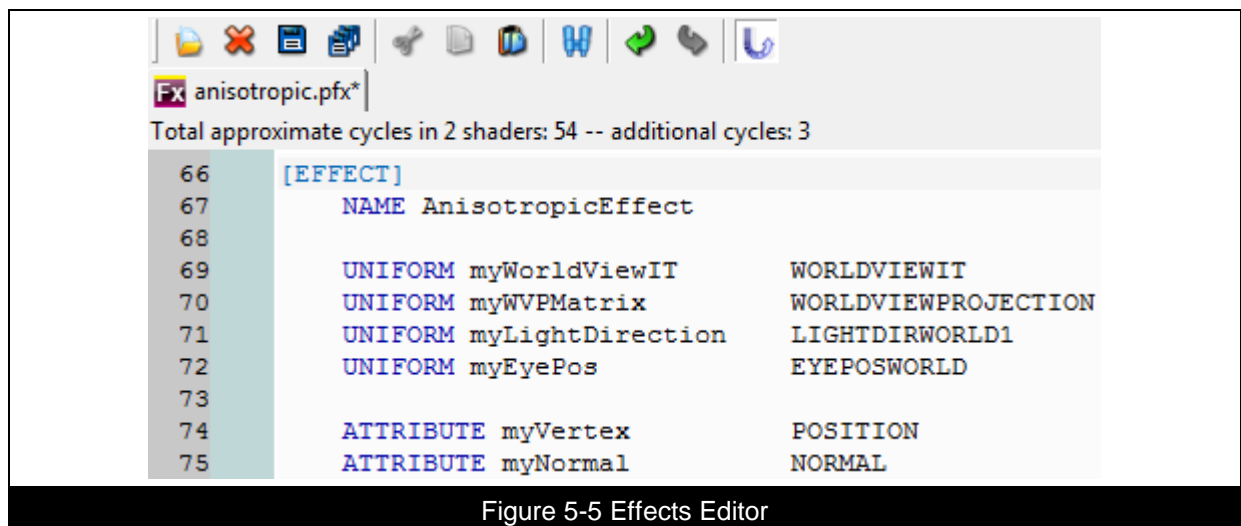


Figure 5-5 Effects Editor

The effects editor panel, output tab, and profile tab are the home of PVRUniSCoEditor(see Section 4.2.3 PVRUniSCoEditor). PVRUniSCoEditor can be used to edit GLSL code, .pfx and .fx files as well as plain text files. In addition to the functionality already described, open files are arranged as tabs so multiple files can be open at any given time; and right clicking within an open file brings up a context menu that allows for easy insertion of attributes or uniforms as well as the normal array of copy and paste options.

Full information on the functionality found in PVRUniSCoEditor can be found in the PVRUniSCoEditor User Manual.



### 5.1.5. Debug Output Panel

Description	File	Line
--- Loading Effects ---		
Material 'Reflection' using effect file 'texture.pfx' ... passed.	C:\...\OGLES2\texture.pfx	
Material 'Toon' using effect file 'toon.pfx' ... passed.	C:\...\OGLES2\toon.pfx	
Material 'Anisotropic' using effect file 'anisotropic.pfx' ... passed.	C:\...\anisotropic.pfx	
Material 'Marble' using texture 'maskmain.pvr' ... passed.		
warning: Semantic unknown to application: MinThickness MINTHICKNESS	C:\...\iridescence.pfx	
warning: Semantic unknown to application: MaxVariation MAXVARIATION	C:\...\iridescence.pfx	
Material 'Iridescence' using effect file 'iridescence.pfx' ... passed.	C:\...\iridescence.pfx	
Material 'RenderToTexture' using effect file 'RenderTexture.pfx' ... passed.	C:\...\RenderTexture.pfx	

Figure 5-6 Debug Output Panel

The debug output panel appears at the bottom of the main window and contains information as to the state of the loaded effects and meshes. In the case of errors in shaders or effects files the line number and a description of the error will be seen. Warnings are also given for errors which are 'non-fatal' and should not affect the running of the shader/effect in which the error occurs.

## 5.2. Menus

### 5.2.1. File Menu

This menu is composed of the file input and output options for POD files and shader files.

#### Open POD

'Open POD...' is used to open a .pod file.

#### Open Recent POD

This drop down menu contains a list of the ten most recently opened .pod files; clicking on an entry from this list will open the specified .pod file.

#### Open Recently Exported

The 'Open Recently Exported' drop down menu contains a list of the ten most recently exported .pod files; clicking an entry from this list will open the specified .pod file.

#### Open Built-in Object

'Open Built-in Object' creates a new .pod file from the existing object library.

#### Import File

'Import File...' allows for the importing of .dae files providing the path to Collada2POD has been set in 'Tools -> Preferences -> Path to Collada2POD'.

#### Save

'Save' saves the currently open .pod file.

#### Save As

'Save As...' saves the currently open .pod file to a new file.

#### Close POD

'Close POD' closes the currently open .pod file.

#### Reload POD File

'Reload POD File' reloads the currently open .pod file, this is particularly useful if the file has been updated from an external application, for example 3D Studio Max.

#### POD info

'POD Info' displays information as to the contents of the .pod file; the number of materials, meshes, and frames of animation, the total number of vertices and faces, including a per mesh breakdown.

#### New Shader

The 'New Shader' menu contains a list of shader types which, when selected, create a new file of the chosen type.

#### Open Shader

'Open Shader' opens an existing shader file.

#### Save All Shaders

'Save All Shaders' saves all the currently open shaders.

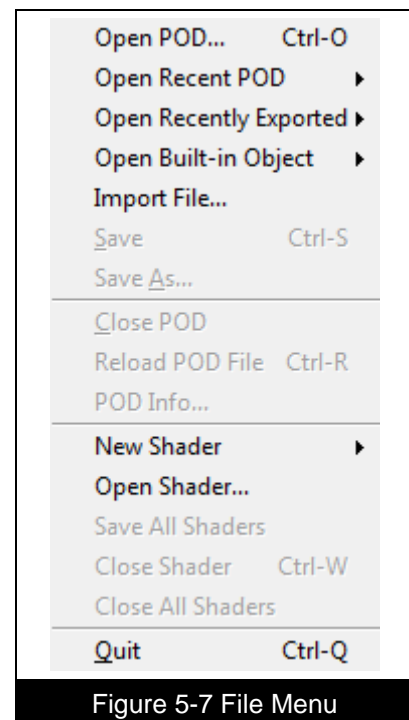


Figure 5-7 File Menu

**Close Shader**

'Close Shader' closes the currently selected shader.

**Close All Shaders**

'Close All Shaders' closes all the open shader files.

**Quit**

'Quit' closes PVRShaman.

### 5.2.2. Edit Scene Menu

This menu contains options to edit the scene, and navigate the Visualization Panel.

#### Fit Selected

'Fit Selected' zooms the visualisation panel in/out to fit the currently selected object, and centres the object within the panel.

#### Fit All

'Fit All' zooms the visualisation panel in/out to fit the entire scene, and centres the scene within the panel.

#### Centre Selected

'Centre Selected' centres the current view in the visualisation panel on the currently selected object.

#### Hide All

'Hide All' hides all objects in the scene.

#### Unhide All

'Unhide All' unhides all objects hidden with 'Hide All'

#### Current View Properties

'Current View Properties' displays the Camera Properties Dialog for the currently selected camera.

#### Add Material

'Add Material' brings up the Add Material dialog as described in Section 4.1 Adding Materials.

#### Delete Material

'Delete Material' deletes the currently selected material.

#### Assign Material

'Assign Material' is used to assign a given material to a given mesh.

#### Background Colour

'Background Colour...' changes the background colour of the Visualization Panel.

#### Save Screenshot

'Save Screenshot' saves a screenshot of the current contents of the Visualization Panel.

#### Select

This radio button changes the control scheme of the Visualization Panel so that objects can be selected by clicking on them.

#### Pan

This radio button changes the control scheme of the Visualization Panel so that it is possible to pan around the scene using the mouse while the left mouse button is down.

#### Rotate

With 'Rotate' active it is possible to rotate the scene around the currently selected object, the world axis, or the centre of the Visualization Panel using the mouse while the left mouse button is down.

#### Zoom

With 'Zoom' active, mouse movement controls the zoom level of the Visualization Panel, pulling back will zoom out, pushing forwards will zoom in, while the left mouse button is down.

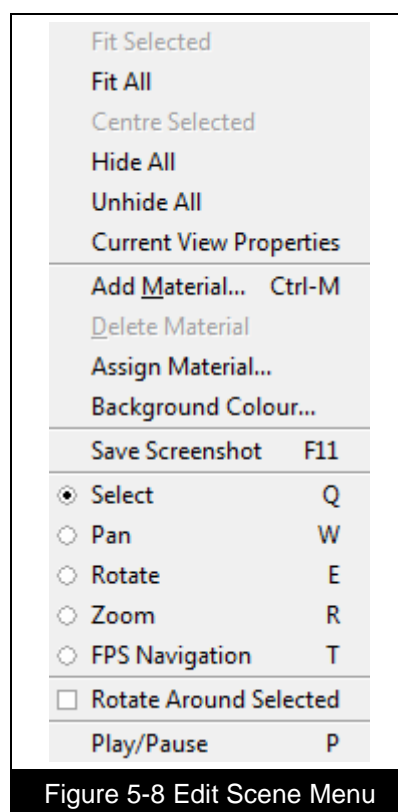


Figure 5-8 Edit Scene Menu

**FPS Navigation**

With this option selected the mouse and keyboard now allow for First Person Shooter (FPS) like movement. Pressing the 'Esc' key will exit this mode, returning the control scheme to 'Select'.

**Rotate Around Selected**

When this option is ticked the 'Rotate' button on the Visualization Panel Toolbar rotates around the selected object rather than around the origin of the POD scene.

**Play/Pause**

'Play/Pause' is used to control the animation currently playing in the Visualization Panel.

### 5.2.3. Edit Shader Menu

#### Undo

'Undo' undoes the last performed action.

#### Redo

'Redo' redoes the last undone action.

#### Cut

'Cut' cuts the selected text to the clipboard.

#### Copy

'Copy' copies the selected text to the clipboard.

#### Paste

'Paste' pastes the contents of the clipboard.

#### Delete

'Delete' deletes the currently selected text.

#### Select All

'Select All' selects the entire contents of the file that currently has focus.

#### Comment Selection

'Comment Selection' comments out the selected text from the file; if a complete line is selected that line will begin with '///  
'; if part of a line is selected that part will be surrounded in '/\* \*/'.

#### Uncomment Selection

'Uncomment Selection' removes the commenting out from the selected text, either removing a surrounding '/\* \*/' or the '///  
' at the beginning of the line.

#### Indent Selection

'Indent Selection' indents the selected tax by a single tab.

#### Outdent Selection

'Outdent Selection' removes an indent from the selected text.

#### Find/Replace

'Find/Replace' opens the **Error! Reference source not found..**

Undo	Ctrl-Z
Redo	Ctrl-Y
Cut	Ctrl-X
Copy	Ctrl-C
Paste	Ctrl-V
Delete	Del
Select All	Ctrl-A
Comment Selection	Ctrl-K
Uncomment Selection	Ctrl-U
Indent Selection	Tab
Outdent Selection	Shift-Tab
Find/Replace	Ctrl-F

Figure 5-9 Edit Shader Menu

## 5.2.4. Build Menu

### Apply Shaders

'Apply Shaders' compiles all shaders in the current .pod file that have been edited and applies those shaders to the scene in the Visualization Panel.

### Rebuild Shaders

'Rebuild Shaders' rebuilds all shaders in the current .pod file irrespective of whether that shader has been edited.

### Generate Cycle Counts

The 'Generate cycle counts in the background' tick box enables the calculation and display of cycle counts in PVRUniSCoEditor if a compiler has been specified in 'Tools -> Preferences -> Compiler Settings'.

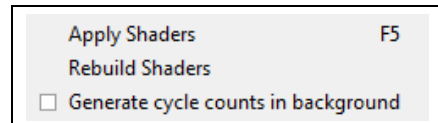


Figure 5-10 Build Menu

## 5.2.5. Render Menu

### Render Modes

The first five options in the Render menu represent the available Render Modes, more details on these render modes can be found in Section 5.6 Render Modes.

### Default Lighting

'Default Lighting' toggles on or off the default light, when this is toggled off only lights from the .pod file are used.

### Show Grid

'Show Grid' displays a grid around the origin within the Visualization Panel.

### Show Axis Helper

'Show Axis Helper' displays a small X, Y, Z axis diagram around the origin within the Visualization Panel.

### Depth Test

'Depth Test' allows for the enabling or disabling of depth testing within the scene.

### Backface Culling

'Backface Culling' tells PVRShaman to enable or disable backface culling.

### Frontface Culling

'Frontface Culling' enables or disables frontface culling. This allows scenes that have different winding orders for their polygons to be rendered correctly by PVRShaman.

### Anti-Aliasing

'Anti-aliasing' activates or deactivates anti-aliasing for the current scene. Toggling this option requires the scene to be saved and reopened.

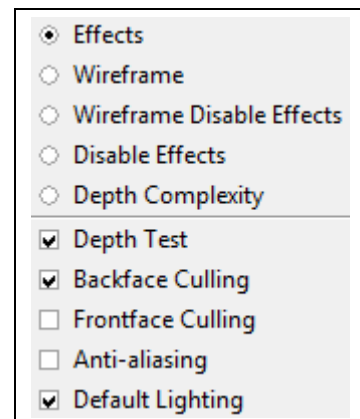


Figure 5-11 Render Menu

### 5.2.6. View Menu

The View Menu contains options for adjusting the window configuration. Full information on these window configurations can be found in Section 5.5 Window Configuration Modes.

In addition it also includes the options 'Compile/Profile Output', and 'Scene Output' which enable and disable the Output/Profile bar within PVRUniSCoEditor and the Debug Output Panel respectively.

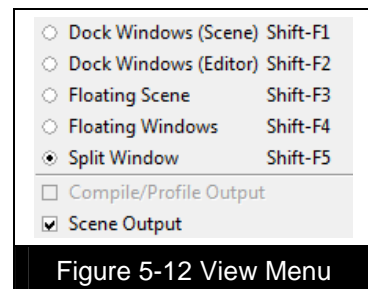


Figure 5-12 View Menu

### 5.2.7. Tools Menu

#### Preferences

'Preferences' opens the 'Preferences' dialog, a full description of which can be found in Section 6 Preferences.

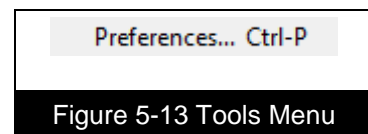


Figure 5-13 Tools Menu

### 5.2.8. Help Menu

#### Help

'Help' opens this user manual.

#### Show Semantics

'Show Semantics' opens a window containing a list of all of the semantics available to PVRShaman. This list is also available in Appendix A. PVRShaman PFX Semantics List.

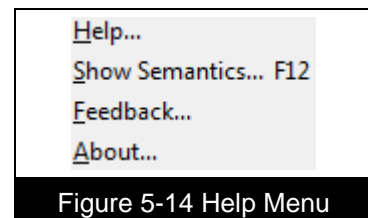


Figure 5-14 Help Menu



## 5.3. Toolbars

### 5.3.1. Main Toolbar

#### Open POD

Opens a new .pod file.



Figure 5-15 Open POD

#### Close POD

Closes the current .pod file.



Figure 5-16 Close POD

#### Save POD

Saves the current .pod file.



Figure 5-17 Save POD

#### Reload POD

Refreshes the current .pod file. This is useful if shader or .pod files have been edited outside PVRShaman.



Figure 5-18 Reload POD

#### Add Material

Adds a material to the scene using the Add Material dialog.



Figure 5-19 Add Material

#### Delete Material

Deletes the currently selected material from the Scene Container.



Figure 5-20 Delete Material

#### Assign Material

Launches a dialog that allows materials to be assigned to meshes.



Figure 5-21 Assign Material

#### Preferences

Launches the preferences dialog.



Figure 5-22 Preferences

#### Apply Shader

Compiles the current shader and applies it to the scene in the Visualization Panel.



Figure 5-23 Apply Shader

### 5.3.2. Visualization Panel Toolbar

#### Fit Selected

Alters the view so that the selected object is fitted to the Visualization Panel.



Figure 5-24 Fit Selected

#### Fit All

Alters the view so that the whole scene to fit the Visualization Panel.



Figure 5-25 Fit All

#### Unhide All

Unhides all objects hidden with 'Hide All'.



Figure 5-26 Unhide All

#### Hide All

Hides all objects in the scene.



Figure 5-27 Hide All

#### Rotate Around Selected

Indicates the 'Rotate' action will now rotate around the currently selected object rather than the entire frame.



Figure 5-28 Rotate Around Selected

#### Select

Enables selection mode in the scene. Clicking on an object will select the object.



Figure 5-29 Select

#### Pan

Enables pan mode. Use this mode to displace the scene within the Visualization Panel.



Figure 5-30 Pan

#### Rotate

Enables rotation mode.



Figure 5-31 Rotate

#### Zoom

Enables zoom mode. Use this mode to zoom the Visualization Panel in and out.



Figure 5-32 Zoom

### FPS Navigation

Enables "First Person Shooter" navigation mode. This mode uses the mouse and the W, A, S & D keys to navigate the Visualization Panel.



Figure 5-33 FPS Navigation

### Camera Selection

Pull down menu to select the current camera for the viewport. 'No Camera' means that the current view is not from a camera defined in the scene. 'Follow Selected' follows the currently selected object - useful for observing one object through an animated scene.

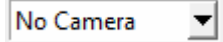


Figure 5-34 Camera Selection

### Play/Pause

Play or pause the animation if there is any.



Figure 5-35 Play/Pause

### Frame Slider

A slider representing the progress through the current scene.



Figure 5-36 Frame Slider

### Current Frame

The number entered in here represents the frame currently being displayed in the Visualization Panel.



Figure 5-37 Current Frame

### Time Direction

Using this dropdown, the direction of time within an animated scene can be selected. The three options are:

- Forward – The standard direction of the scene.
- Reverse – The reverse of the direction indicated by 'Forward'.
- Ping-Pong – The scene will play 'Forwards' until completion, then will play 'Reverse' until the first frame is returned to.

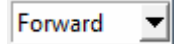


Figure 5-38 Time Direction

### 5.3.3. Effects Editor Toolbar

#### Open File

Opens an effect file or shader file.



Figure 5-39 Open File

#### Close File

Closes the file currently open in the editor.



Figure 5-40 Close File

#### Save File

Saves the file currently open in the editor.



Figure 5-41 Save File

#### Save All

Saves all files open in the editor.



Figure 5-42 Save All

#### Cut

Cuts the selected text to the clipboard.



Figure 5-43 Cut

#### Copy

Copies the selected text to the clipboard.



Figure 5-44 Copy

#### Paste

Pastes text currently in the clipboard.



Figure 5-45 Paste

#### Find

Opens the Find/Replace Toolbar.



Figure 5-46 Find

#### Undo

Undoes the last action.



Figure 5-47 Undo

#### Redo

Redoes an action that has been undone.



Figure 5-48 Redo

#### Generate Cycle Counts

Enables calculation and display of shader cycle counts.



Figure 5-49 Cycle Counts

### 5.3.4. Find/Replace Toolbar

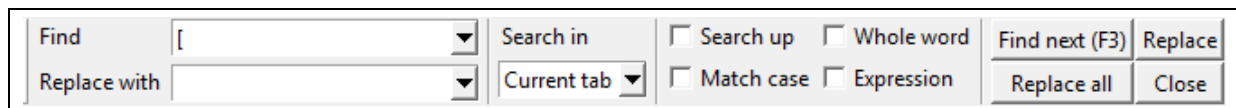


Figure 5-50 Find/Replace Toolbar

The find and replace dialog can be accessed via the 'Edit Shader' menu, the Effects Editor Toolbar, or by pressing 'Ctrl-F'. It will appear below the toolbar. Selected text or the word closest to the cursor will automatically become the suggested search term. The current tab, all open tabs, or the current selection, if any can be searched. Searching up instead of down, searches for whole words only, and case sensitive searches are all possible.

#### Regular Expressions Syntax

When you tick the 'Expression' box PVRUniSCoEditor will interpret the search term as a regular expression. The 'Find' field will be shown in red if the search term is not a complete regular expression. More information on the type of error will be shown in the tooltip that appears when you hover the mouse cursor over the field as shown below.

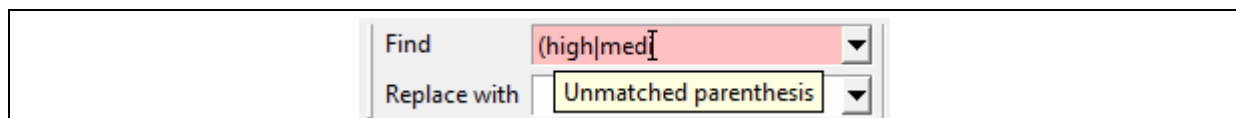


Figure 5-51 Find/Replace Regular Expression Error

Information on what regular expressions are supported can be found in Appendix B. Regular Expression Syntax. It is important to note that back references (\1 to \9) can also be used in the 'Replace with' field as well as the 'Find' field.

## 5.4. Dialogs

### 5.4.1. Material Properties Dialog

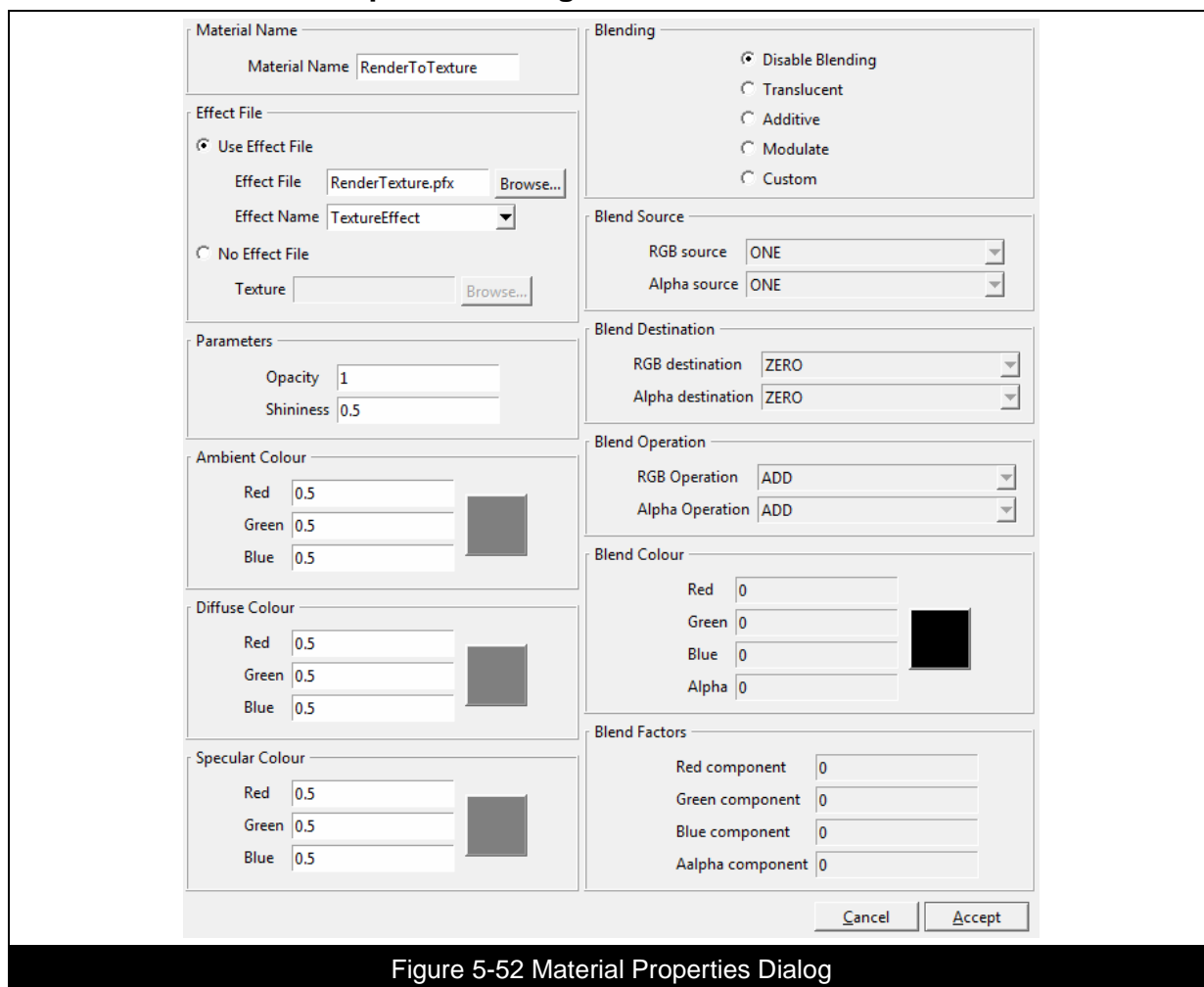


Figure 5-52 Material Properties Dialog

The Material Properties dialog box displays a material's properties and allows them to be edited. The effect file associated with a given material can be changed as can the specific effect being used. Values for opacity, shininess, ambient colour, diffuse colour and specular colour can be adjusted in the range 0 to 1. Clicking a colour panel brings up a Colour Selector dialog, which serves as an alternative way to select the colour. Blend mode data can also be changed at a very precise level. Finally, all changes to the material are applied to the scene as the values are set.

### 5.4.2. Add Material Dialog

Figure 5-53 Add Material Dialog

When adding a new material it must be given a name, and can optionally be assigned directly to a mesh. There are four options when adding materials:

- 'Use Existing Effect File'
- 'No Effect File'
- 'Create Empty Effect File'
- 'From Effect Library'

Once the new material has been created it will be added to the list of materials in the scene container. More information on how to add materials can be found in Section 4.1 Adding Materials.

### 5.4.3. Light Properties Dialog

#### Colour

'Colour' values must be in the range 0 to 1.

#### Type

The type of light can be either 'Point', 'Directional', or 'Spot'.

#### Position

'Position' refers to the coordinates of the light source (disabled for directional lights).

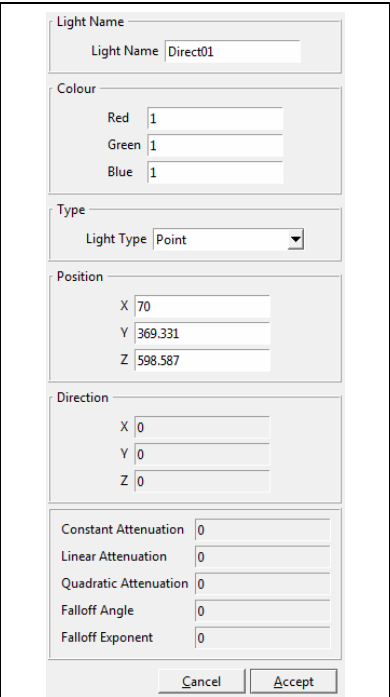
#### Direction

'Direction' refers to the direction of the light (disabled for point lights).

#### Falloff and Attenuation

'Falloff' and 'Attenuation' are for spot lights only and refer to the rate at which the light from a spot light falls off to zero based on distance and angle from the centre of the light respectively.

Changes will be displayed as they are made and saving the .pod file will store the changes.



The screenshot shows the 'Light Properties Dialog' with the following fields and values:

Light Name	
Light Name	Direct01

Colour	
Red	1
Green	1
Blue	1

Type	
Light Type	Point

Position	
X	70
Y	369.331
Z	598.587

Direction	
X	0
Y	0
Z	0

Constant Attenuation	0
Linear Attenuation	0
Quadratic Attenuation	0
Falloff Angle	0
Falloff Exponent	0

Buttons: Cancel, Accept

Figure 5-54 Light Properties Dialog



#### 5.4.4. Camera Properties Dialog

##### Parameters

'FOV' refers to the field of view.

'Near' and 'Far' are the distances of the view frustum clip planes.

##### From

'From' is the co-ordinate of the camera.

##### To

'To' is the coordinate of the point the camera is looking towards.

##### Up

'Up' refers to the direction of the up axis.

As with previous dialogs; changes are displayed as they are made.

Figure 5-55 Camera Properties Dialog

### 5.4.5. Preferences Dialog

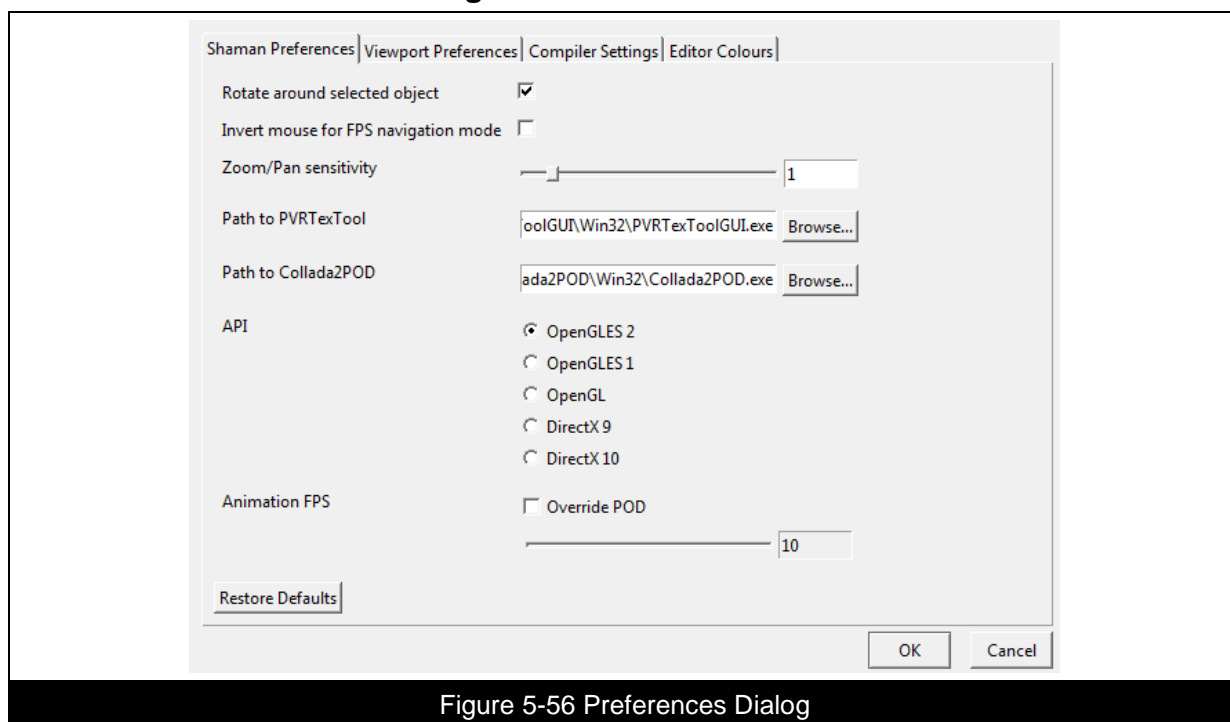


Figure 5-56 Preferences Dialog

The preferences dialog can be accessed through the Tools menu. Full details on all the available options can be found in Section 6 Preferences.

### 5.4.6. POD Information Dialog

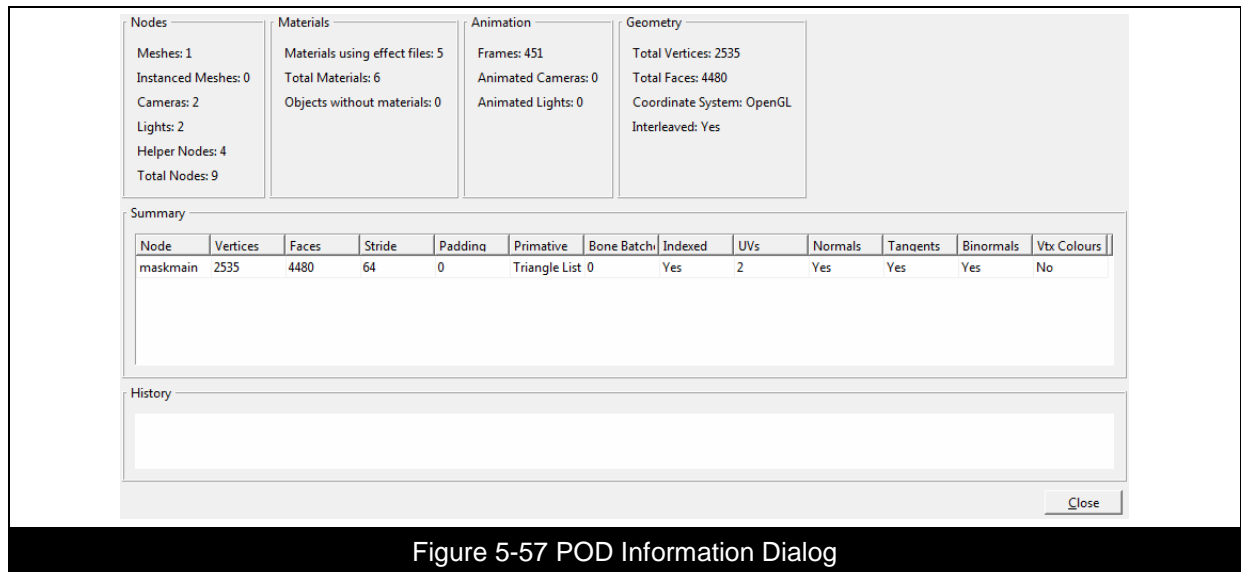


Figure 5-57 POD Information Dialog

This information displays a variety of information about the currently open .pod file. The total combined information about the file is displayed at the top with per mesh information at the bottom (including what information was exported by Collada2POD/PVRGeoPOD). The history section can also be used to keep track of changes made to the file.

## 5.4.7. Object Data Dialog

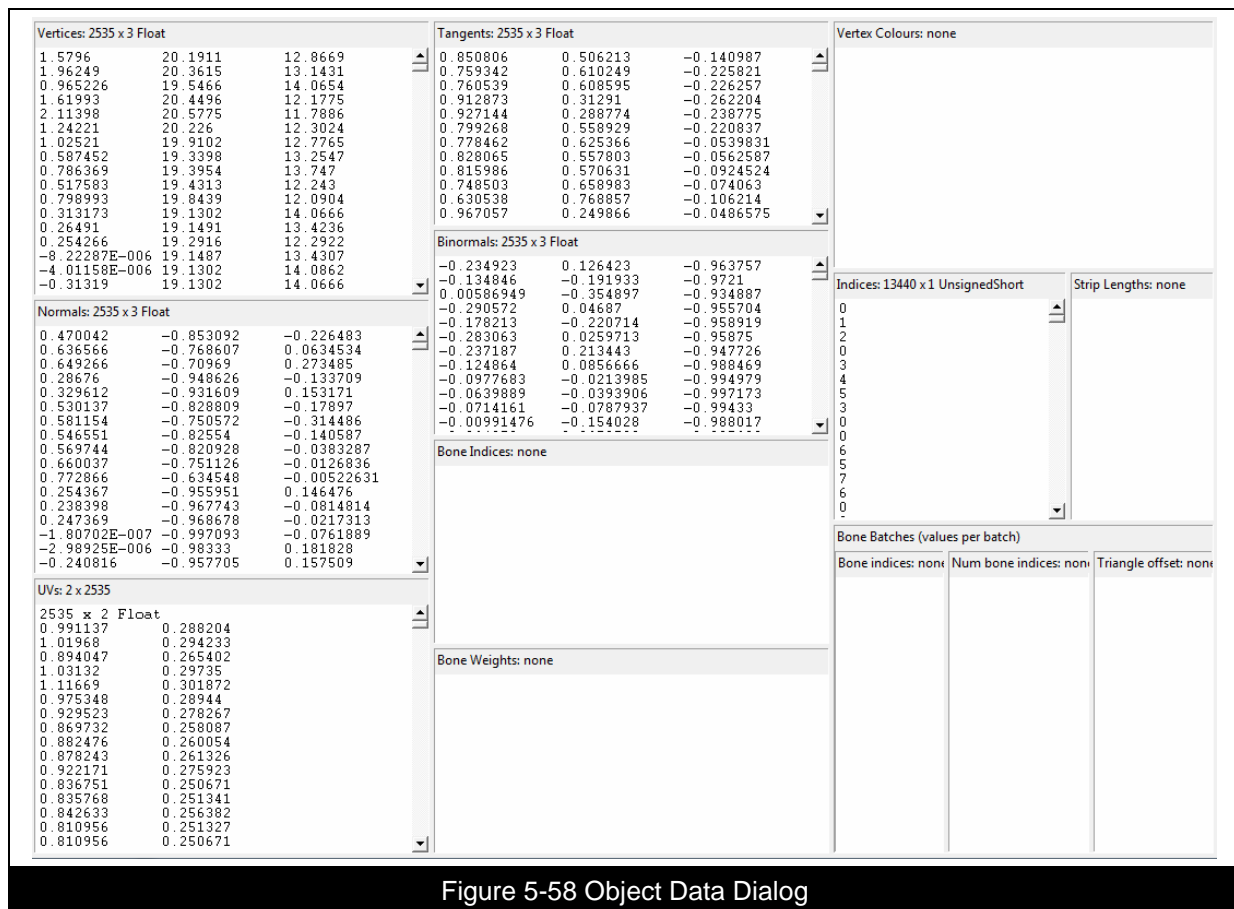


Figure 5-58 Object Data Dialog

This dialog displays all of the data associated with a particular mesh, including vertex positions, normal directions, bone indices, vertex colours etc. This is particularly useful for spotting errors in models or in the exporting process.

## 5.5. Window Configuration Modes

Various window configurations are available for PVRShaman; these can be set either from the 'View' menu, or by using the shortcuts 'Shift+F1' through to 'Shift+F5'.

### 5.5.1. Dock Windows (Scene)

In this view the Scene and Editor are both docked as tabs in the main window. The Scene tab is displayed.

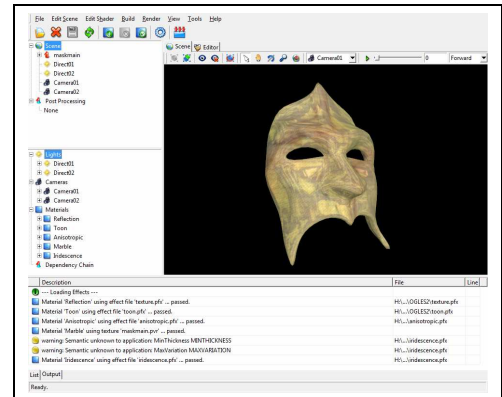


Figure 5-59 Dock Windows (Scene)

### 5.5.2. Dock Windows (Editor)

In this view the Scene and Editor are both docked as tabs in the main window. The Editor tab is displayed.

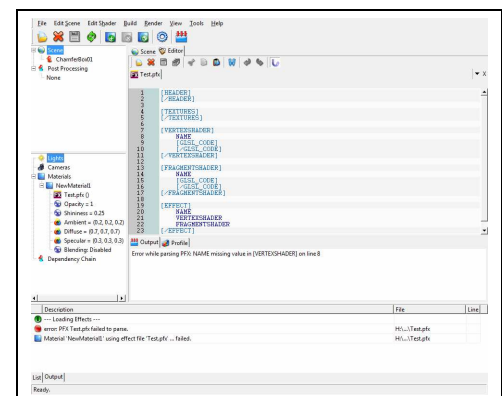


Figure 5-60 Dock Windows (Editor)

### 5.5.3. Floating Scene

In this view the scene is in a floating window and the editor is docked into the main window. This mode can be useful when using multiple monitors.

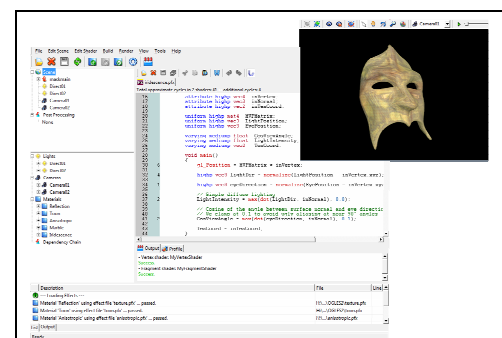
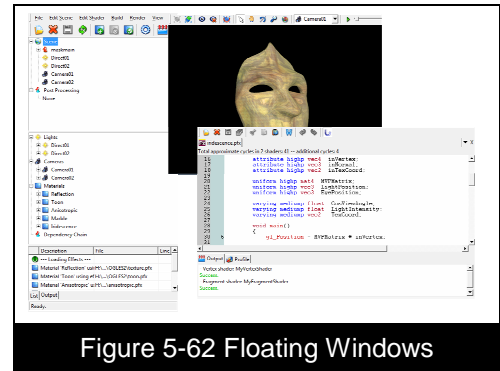


Figure 5-61 Floating Scene

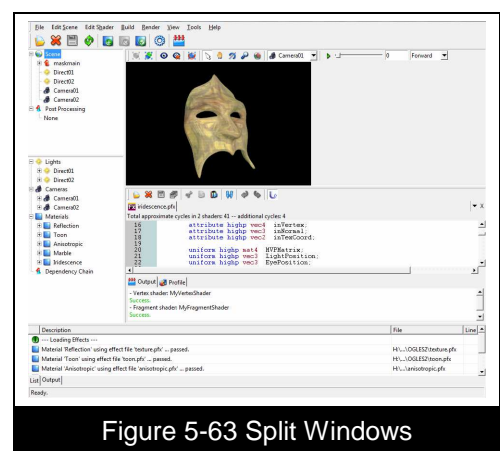
### 5.5.4. Floating Windows

In this view both the scene and editor are in floating windows and the main window is reduced in size. As with Floating Scene, this can be useful when using multiple monitors.



### 5.5.5. Split Window

In this view the main window is split, with the scene in the top half and the editor below.



## 5.6. Render Modes

These modes change the way meshes are rendered and effects are applied within the Visualization Panel.

### 5.6.1. Effects

This is the default mode, displaying any PFX and texture effects which are applied to the meshes.



Figure 5-64 Effects Render Mode

### 5.6.2. Wireframe

This mode shows the effects applied to a wireframe version of the mesh.



Figure 5-65 Wireframe Render Mode

### 5.6.3. Wireframe No Effects

This mode shows the wireframe mesh without any effects applied.



Figure 5-66 Wireframe No Effects Render Mode

#### 5.6.4. No Effects

This mode shows the meshes without the effects applied. If no material is applied to a mesh, or the material fails to load, the mesh will be displayed in this way.

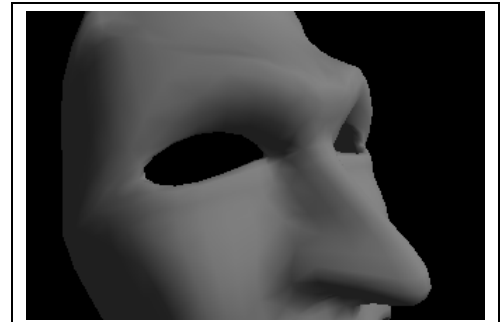


Figure 5-67 No Effects

#### 5.6.5. Depth Complexity

This mode shows the depth complexity of the scene. The brighter white means greater complexity.



Figure 5-68 Depth Complexity



## 5.7. Navigation Modes

These modes are used to navigate the scene in the Visualization Panel.

### 5.7.1. Select

This mode allows for selection of meshes within the scene.



Figure 5-69 Select

### 5.7.2. Pan

This mode allows the whole scene to be moved up, down, left and right.



Figure 5-70 Pan

### 5.7.3. Rotate

This mode allows rotation around the centre of the scene, or the centre of the current object, if one is selected.



Figure 5-71 Rotate

### 5.7.4. Zoom

'Zoom' moves the view in or out as the mouse is moved forward or backward.



Figure 5-72 Zoom

### 5.7.5. FPS Navigation

This mode allows navigation of the scene similar to the movement in a first person shooter. The mouse cursor is grabbed by the visualization window and can no longer be used until the mode is quit. This can be achieved by left clicking the mouse, or by pressing Escape. In this mode the mouse is used to change the view direction and the keyboard is used to move:

- W – Forward
- S – Backward
- A – Strafe Left
- D – Strafe Right



Figure 5-73 FPS Navigation

## 5.8. Texture Viewer

The Texture Viewer displays a given texture within PVRShaman. It can be accessed by right clicking on a texture from within the Scene Container.

The Texture Viewer can also show render to texture targets in real time as they update by double clicking on a the render to texture within the Scene Container.

Finally, it is possible to save a texture to a file, either a pre-loaded texture, or a given frame of a render to texture target; to reload a pre-loaded texture, and to zoom any texture.

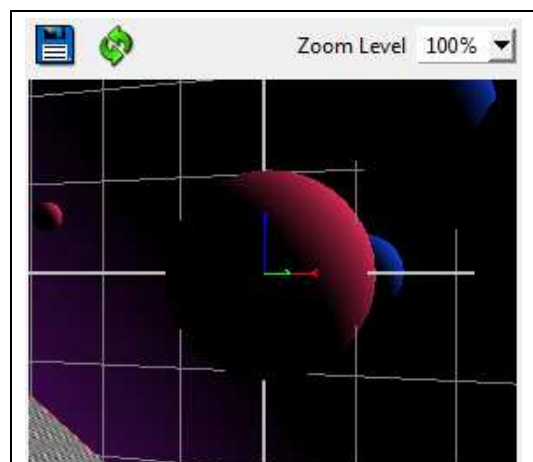


Figure 5-74 Texture Viewer

## 6. Preferences

### 6.1. General Preferences

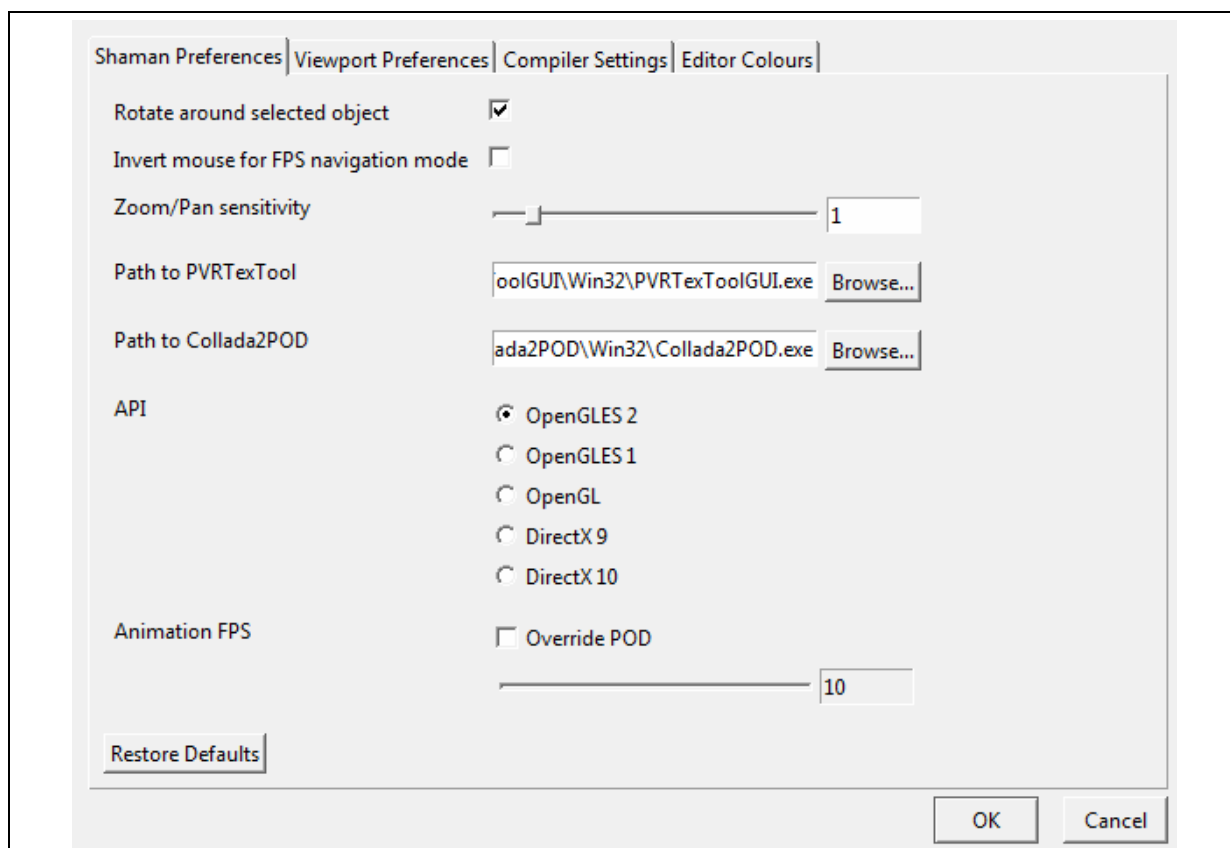


Figure 6-1 General Preferences

#### Rotate around select object

'Rotate around selected object' changes the Rotate navigation option to rotate around any object that is selected as opposed to the origin.

#### Invert mouse for FPS Navigation mode

'Invert mouse for FPS Navigation mode' inverts the Y-Axis when in FPS Navigation mode.

#### Zoom/Pan sensitivity

'Zoom/Pan sensitivity' adjusts the rate at which the Zoom and Pan navigation modes function.

#### Path to PVRTexTool

'Path to PVRTexTool' sets the path to the PVRTexTool binary; if this is set, double clicking on a texture will open it in PVRTexTool.

#### Path to Collada2POD

'Path to Collada2POD' sets the path to the Collada2PODGUI binary; if this options is set the 'File -> Import' function can be used to import .dae files.

**API**

'API' sets the API currently being used by PVRShaman. If this option is changed a restart of PVRShaman will be required. It should be noted that OpenGL/OpenGL ES require effects in .pfx format while DirectX requires effects to be in the .fx format. Winding order is also important when the API is changed; files previously exported to target OpenGL must be re-exported to target DirectX as OpenGL uses a left handed coordinate system while DirectX uses a right handed.

**Animation FPS**

'Animation FPS' overrides the frames per second (FPS) rate of loaded .pod files and will play them back at the FPS set using the slider.

## 6.2. Viewport Preferences

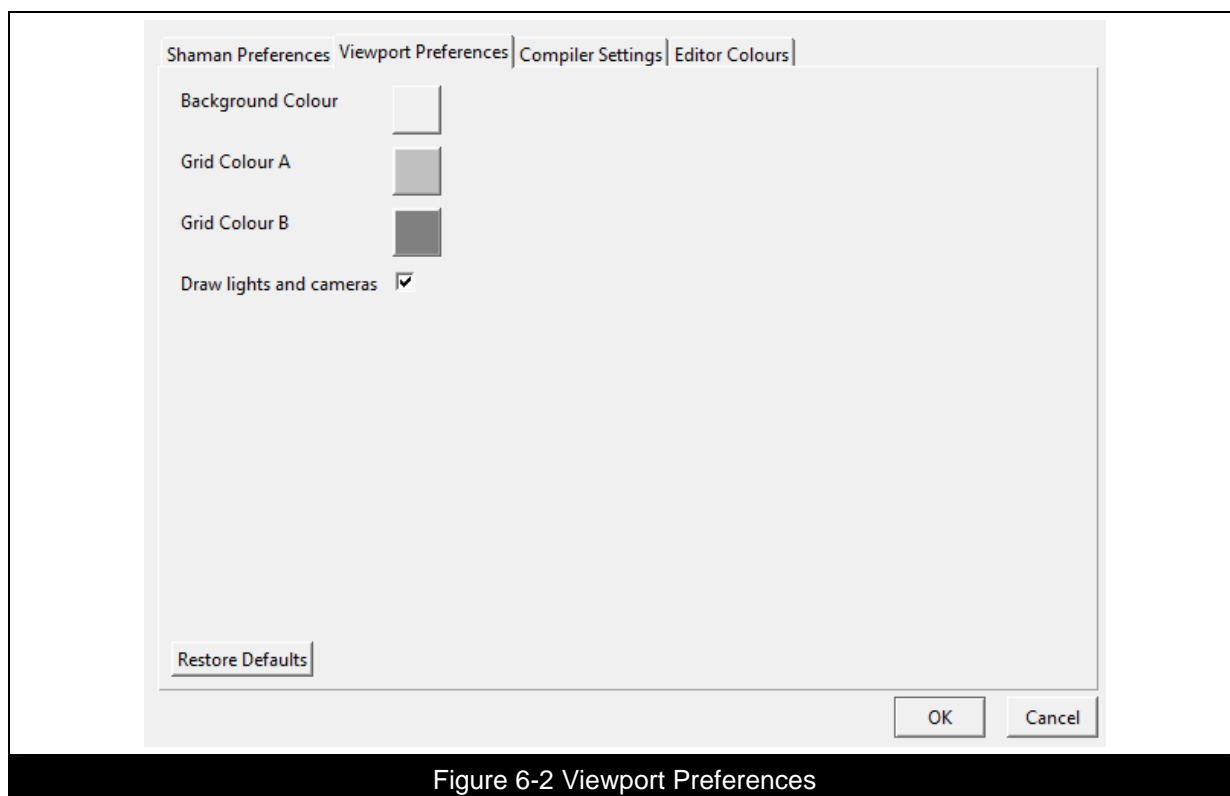


Figure 6-2 Viewport Preferences

### Background Colour/Grid Colour

These options allow the colouration of the grid and background within the Visualization Panel to be changed.

### Draw lights and cameras

'Draw lights and cameras' toggles whether lights and cameras have icons displayed in the Visualization Panel.

## 6.3. Compiler Settings

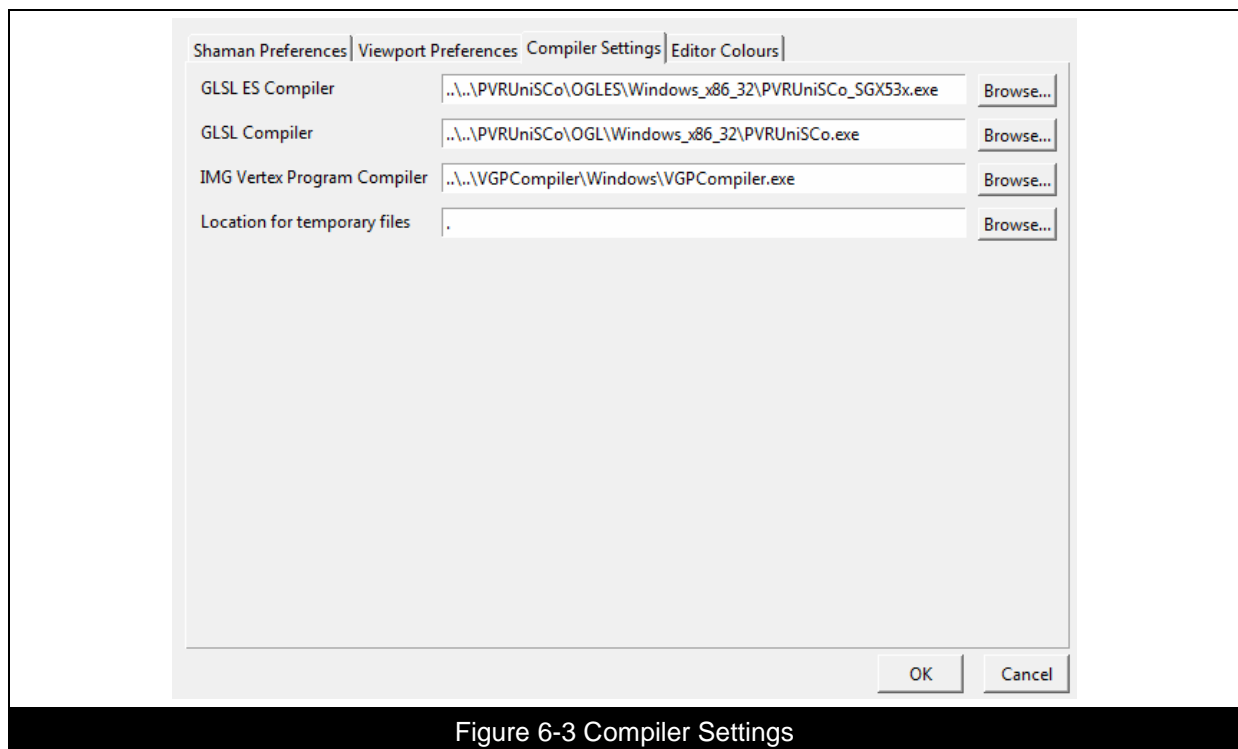


Figure 6-3 Compiler Settings

This window sets the location of the compilers that PVRShaman uses. Each of these must be set for PVRShaman to work correctly.

### GLSL ES Compiler

'GLSL ES Compiler' refers to the OpenGL ES compiler location. The compilers can be found in the 'Utilities' folder of the SDK, under 'PVRUniSCo'.

### GLSL Compiler

'GLSL Compiler' refers to the OpenGL compiler location. The compilers can be found in the 'Utilities' folder of the SDK, under 'PVRUniSCo'.

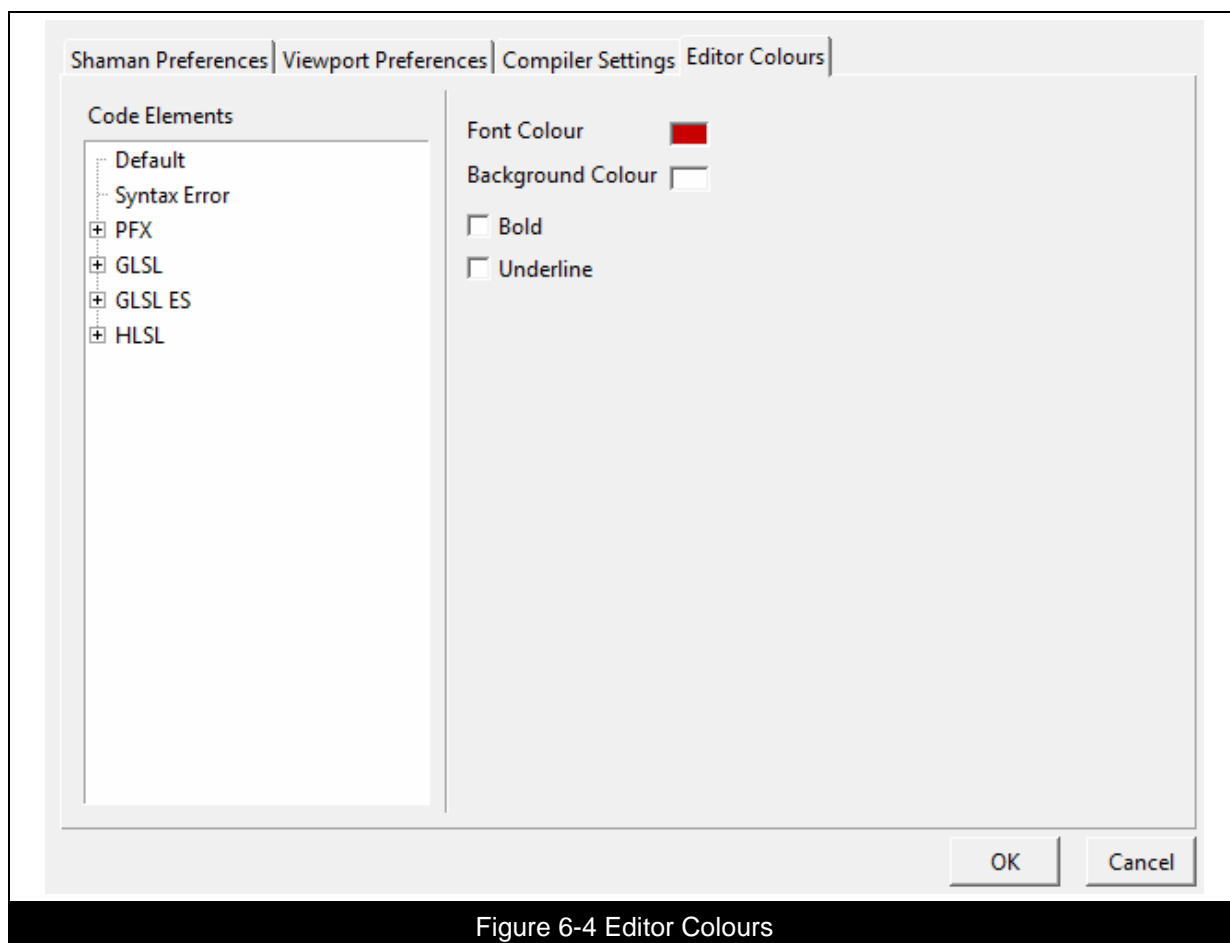
### IMG Vertex Program Compiler

'IMG Vertex Program Compiler' refers to the location of the VGP compiler. This option is deprecated.

### Location for temporary files

'Location for temporary files' refers to the location that PVRShaman will use to store any temporary files it creates.

## 6.4. Editor Colours



This window allows for control of the appearance of the Effects Editor; everything from background colour to syntax highlighting colours for the various supported languages.

## 7. Related Material

### Software

- Collada2POD
- PVRGeoPOD
- PVRTexTool
- PVRUniSCoEditor

### Documentation

- PVRGeoPOD & Collada2POD Getting Started
- PVRGeoPOD & Collada2POD User Manual
- PVRTexTool User Manual
- PVRUniSCo User Manual
- PVRUniSCoEditor User Manual

## 8. Contact Details

For further support contact:

[devtech@imgtec.com](mailto:devtech@imgtec.com)

PowerVR Developer Technology  
Imagination Technologies Ltd.  
Home Park Estate  
Kings Langley  
Herts, WD4 8LZ  
United Kingdom

Tel: +44 (0) 1923 260511

Fax: +44 (0) 1923 277463

Alternatively, you can use the POWERVR Insider forums:

[www.imgtec.com/forum](http://www.imgtec.com/forum)

For more information about POWERVR or Imagination Technologies Ltd. visit our web pages at:

[www.imgtec.com](http://www.imgtec.com)



## Appendix A. PVRShaman PFX Semantics List

PFX Semantics are keywords that are used within the '[EFFECT]' block to signify what a given 'uniform' or 'attribute' refers too. In a normal application this information would be calculated on the CPU during the render loop and passed to the shader; as editing the render loop to control what data is passed to the shader is not possible in PVRShaman these keywords allow for this information to be simulated correctly.

The following style is used to describe each keyword:

### KEYWORD

Format. Description.

### A.1. Attributes

#### POSITION

vec4. Position.

#### NORMAL

vec3. Normal.

#### TANGENT

vec3. Tangent.

#### BINORMAL

vec3. Binormal.

#### UV[n]

vec2. n-th set of UVs. Example UV0.

#### VERTEXCOLOR

vec4. Vertex colour attribute.

#### BONEINDEX

vec4. Bone Index.

#### BONEWEIGHT

vec4. Bone Weight.

### A.2. Uniforms

#### WORLD

mat4. World matrix.

#### WORLDI

mat4. World Inverse matrix.

#### WORLDIT

mat3. World Inverse Transpose matrix.

#### VIEW

mat4. View matrix.

#### VIEWI

mat4. View Inverse matrix.

**VIEWIT**

mat3. View Inverse Transpose matrix.

**PROJECTION**

mat4. Projection matrix.

**PROJECTIONI**

mat4. Projection Inverse matrix.

**PROJECTIONIT**

mat3. Projection Inverse Transpose matrix.

**WORLDVIEW**

mat4. World-View matrix.

**WORLDVIEWI**

mat4. World-View Inverse matrix.

**WORLDVIEWIT**

mat3. World-View Inverse Transpose matrix.

**WORLDVIEWPROJECTION**

mat4. World-View-Projection matrix.

**WORLDVIEWPROJECTIONI**

mat4. World-View-Projection Inverse matrix.

**WORLDVIEWPROJECTIONIT**

mat3. World-View-Projection Inverse Transpose matrix.

**UNPACKMATRIX**

Mat4. Matrix used to scale and offset vertex positions if the data has been exported with an unpack matrix.

**VIEWPROJECTION**

mat4. View-Projection matrix.

**VIEWPROJECTIONI**

mat4. View-Projection Inverse matrix.

**VIEWPROJECTIONIT**

mat3. View-Projection Inverse Transpose matrix.

**OBJECT**

mat4. Object matrix, without any parent node transformations.

**OBJECTI**

mat4. Object Inverse matrix, without any parent node transformations.

**OBJECTIT**

mat3. Object Inverse Transpose matrix, without any parent node transformations.

**MATERIALOPACITY**

float. Opacity of material.

**MATERIALSHININESS**

float. Shininess of material.

**MATERIALCOLORAMBIENT**

vec3. Ambient colour of material.

**MATERIALCOLORDIFFUSE**

vec3. Diffuse colour of material

**MATERIALCOLORSPECULAR**

vec3. Specular colour of material.

**BONECOUNT**

int. Number of bones.

**BONEMATRIXARRAY**

mat4[]. Array of bone transformation matrices.

**BONEMATRIXARRAYIT**

mat4[]. Array of bone inverse transpose transformation matrices.

**LIGHTCOLOR[n]**

vec3. Colour of light n (RGB). Example LIGHTCOLOR5.

**LIGHTPOSMODEL[n]**

vec3. Position of light n in model space. Example LIGHTPOSMODEL1.

**LIGHTPOSWORLD[n]**

vec3. Position of light n in world space. Example LIGHTPOSWORLD1.

**LIGHTPOSEYE[n]**

vec3. Position of light n in view space. Example LIGHTPOSEYE1.

**LIGHTDIRMODEL[n]**

vec3. Direction of light n in model space. Example LIGHTDIRMODEL1.

**LIGHTDIRWORLD[n]**

vec3. Direction of light n in world space. Example LIGHTDIRWORLD1.

**LIGHTDIREYE[n]**

vec3. Direction of light n in view space. Example LIGHTDIREYE1.

**LIGHTATTENUATION[n]**

vec3. Attenuation for spot lights (constant, linear, quadratic).

**LIGHTFALLOFF[n]**

vec2. Falloff for spot lights (angle, exponent).

**EYEPOSMODEL**

vec3. Eye position in model space.

**EYEPOSWORLD**

vec3. Eye position in world space.

**TEXTURE[n]**

sampler2D. Sampler for texture n. Example TEXTURE2.

**ANIMATION**

float. Contains the objects distance through its animation. Range 0 to 1.

**GEOMETRYCOUNTER**

Int. Resets to 0 at the beginning of each render frame and increases by one for each submission of geometry.

**VIEWPORTPIXELSIZE**

vec2. Size of the viewport in pixels

**VIEWPORTCLIPPING**

vec4. Near distance, far distance, width angle (radians), height angle (radians)

**TIME**

float. The current time

**LASTTIME**

float. The last frame's time

**ELAPSEDTIME**

float. The time between adjacent frames

**BOUNDINGCENTER**

vec3. Bounding box centre

**BOUNDINGSPHERERADIUS**

Float. Bounding sphere radius

**BOUNDINGBOXSIZE**

vec3. Bounding box size

**BOUNDINGBOXMIN**

vec3. Bounding minimum for x, y, z

**BOUNDINGBOXMAX**

vec3. Bounding maximum for x, y, z

**RANDOM**

float. A random value (Range 0 to 1)

**MOUSEPOSITION**

vec3. The mouse position on screen (x, y, time)

**LEFTMOUSEDOWN**

vec4. The left mouse down state, and its position at that time (x, y, isdown, timedown)

**RIGHTMOUSEDOWN**

vec4. The right mouse down state, and its position at that time (x, y, isdown, timedown)

## Appendix B. Regular Expression Syntax

Special Constructs	
(?i X )	Match sub pattern case insensitive
(?I X )	Match sub pattern case sensitive
(?n X )	Match sub pattern with newlines
(?N X )	Match sub pattern with no newlines
( X )	Capturing parentheses (use with back references, see below)
(?: X )	Non-capturing parentheses
(?= X )	Zero width positive look ahead
(?! X )	Zero width negative look ahead
(?<= X )	Zero width positive look behind
(?<! X )	Zero width negative look behind
(?> X )	Atomic grouping (possessive match)
Logical Operators	
X Y	X followed by Y
X   Y	Either X or Y
Quantifiers	
X *	Match 0 or more
X +	Match 1 or more
X ?	Match 0 or 1
X { }	Match 0 or more
X { n }	Match n times
X { , m }	Match no more than m times
X { n , }	Match n or more
X { n , m }	Match at least n but no more than m times
These quantifiers are greedy. By following them with '?' you can turn them into lazy quantifiers, or follow them by '+' for possessive (non-backtracking) quantifiers.	
Boundary Matching	
^	Match begin of line [if at begin of pattern]
\$	Match end of line [if at end of pattern]
\<	Begin of word
\>	End of word
\b	Word boundary
\B	Word interior
\A	Match only beginning of file
\Z	Match only end of file

Character Classes	
[abc]	Match a, b, or c
[^abc]	Match any but a, b, or c
[a-zA-Z]	Match upper- or lower-case a through z
[]	Matches ]
[-]	Matches -
Predefined Character Classes	
.	Match any character
\d	Digit [0-9]
\D	Non-digit
\s	Space
\S	Non-space
\w	Word character [a-zA-Z_0-9]
\W	Non-word character
\l	Letter [a-zA-Z]
\L	Non-letter
\h	Hex digit [0-9a-fA-F]
\H	Non-hex digit
\u	Single uppercase character
\U	Single lowercase character
\p	Punctuation (not including '_')
\P	Non punctuation
Characters	
\\	Back slash character
\033	Octal
\x1b	Hex
\t	Tab
\n	Newline
Back References	
\1 to \9	Reference to 1 <sup>st</sup> to 9 <sup>th</sup> capturing group

Imagination Technologies, the Imagination Technologies logo, AMA, Codescape, Enigma, IMGworks, I2P, PowerVR, PURE, PURE Digital, MeOS, Meta, MBX, MTX, PDP, SGX, UCC, USSE, VXD and VXE are trademarks or registered trademarks of Imagination Technologies Limited. All other logos, products, trademarks and registered trademarks are the property of their respective owners.