PVRUniSCoEditor

User Manual

Filename	:	PVRUniSCoEditor.User Manual.1.0.4.External.doc
Version	:	1.0.4 External Issue (Package: POWERVR SDK REL_2.10@808188)
Issue Date	:	08 Feb 2012
Author	:	Imagination Technologies Ltd

Contents

List of Figures

1. Introduction

1.1. Software Overview

1.1.1. PVRUniSCoEditor

PVRUniSCoEditor is a shader editor and graphical front-end for the PVRUniSCo shader compiler. It currently offers syntax highlighting for GLSL, GLSL ES, HLSL shaders, VGP vertex programs, Microsoft Effects (.fx) and PowerVR FX (.pfx) files. The Editor is a standalone version of the shader editing functionality that can be found in PVRShaman.

1.1.2. PVRUniSCo

PVRUniSCo is a shader compiler that can be used for GLSL and GLSL ES shaders. It provides PVRUniSCoEditor with profiling output and per line cycle counts to assist in shader development and optimization.

1.2. Document Overview

The purpose of this document is to serve as a complete user manual for PVRUniSCoEditor. It includes installation instructions, a guide to the functionality of the application, and a complete listing of all interface options and preferences for the GUI.

2. Installation

2.1. PVRUniSCo

2.1.1. From Installer

Download either the PowerVR Insider SDK or the individual GLSL Compiler package and follow the on screen instructions. With the package successfully installed the compilers will be available in:

`<SDK_ROOT>\Utilities\PVRUniSCo\<API>\<PLATFORM>`

2.1.2. From GZIP

Download either the PowerVR Insider SDK or the individual GLSL Compiler package. Unzip the .tar.gz file, and then untar the .tar file. The compilers can be found in the following folder:

`<SDK_ROOT>\Utilities\PVRUniSCo\<API>\<PLATFORM>`

2.2. PVRUniSCoEditor

2.2.1. From Installer

Download either the PowerVR Insider SDK or the individual PVRUniSCoEditor package and follow the on screen instructions. Once the package has successfully installed the application will be available in:

`<SDK_ROOT>\Utilities\PVRUniSCoEditor\<PLATFORM>\`

2.2.2. From GZIP

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

`<SDK_ROOT>\Utilities\PVRUniSCoEditor\<PLATFORM>\`

2.2.3. Compiler Settings

Once PVRUniSCo and PVRUniSCoEditor are installed, run PVRUniSCoEditor and open `'Edit -> Preferences -> Compiler Settings'`. The file paths in the ensuing compiler settings dialog must be set to match the file paths as described in Section 2.1 PVRUniSCo above (e.g. `<SDK_ROOT>\Utilities\PVRUniSCo\OGLES\Windows_x86_32\PVRUniSCo_SGX540.exe` for the GLSL ES compiler running on 32 bit Windows X86 targeting SGX540 platforms).

2.2.4. Compatibility

On Linux and Mac OS X11 is required.

3. Interface Overview

3.1. Main Interface

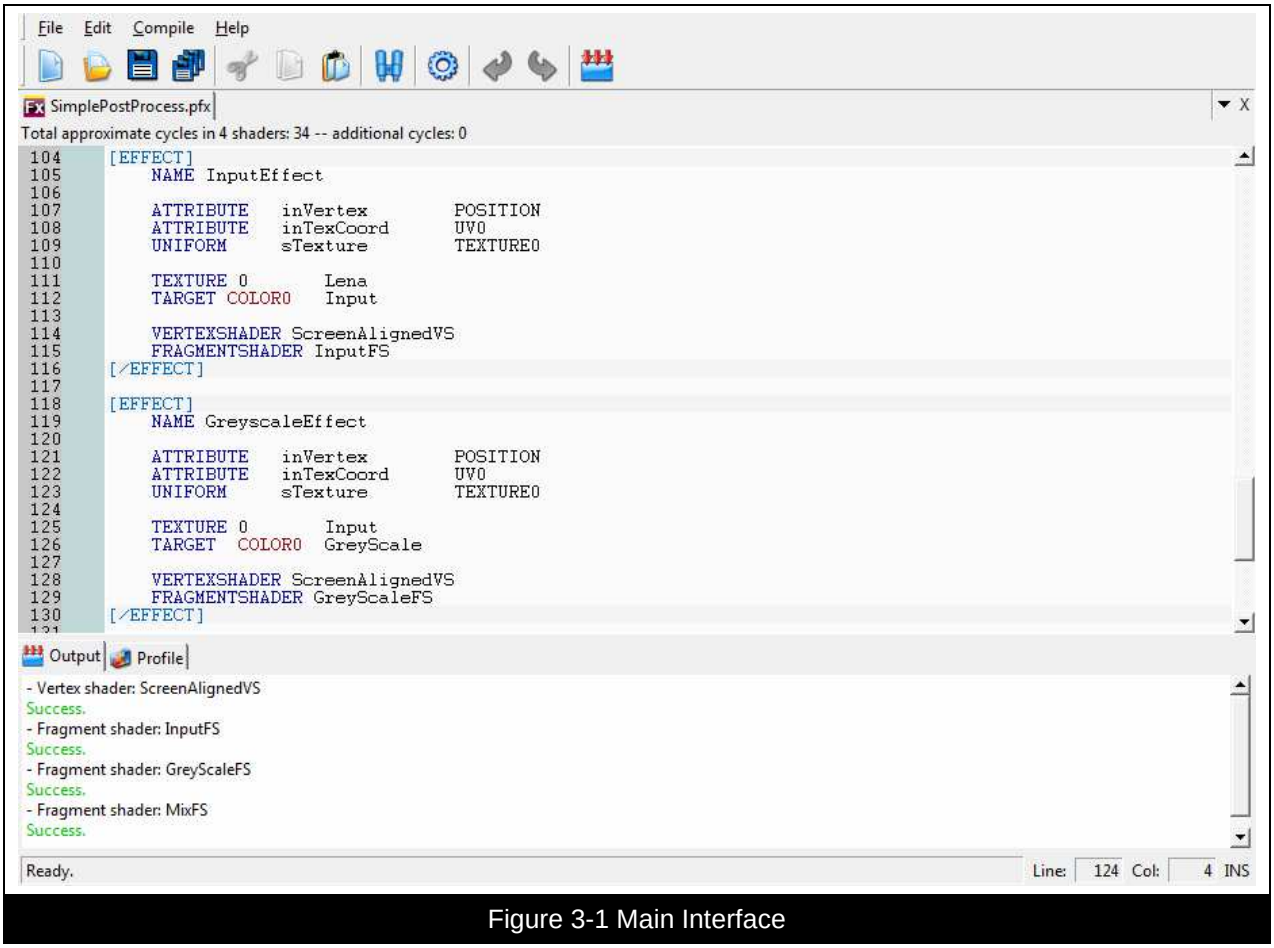

Figure 3-1 Main Interface

3.1.1. Multiple Document Interface

The main interface of PVRUniSCoEditor consists of three parts, the Effects Editor, the Effects Debug Panel, and the Profile Output Panel. As a tab-based MDI (multiple document interface) application PVRUniSCoEditor may have multiple files open simultaneously, displaying a tab for each. In these instances any action performed will be performed on the tab that currently has focus.

3.1.2. Effects Editor

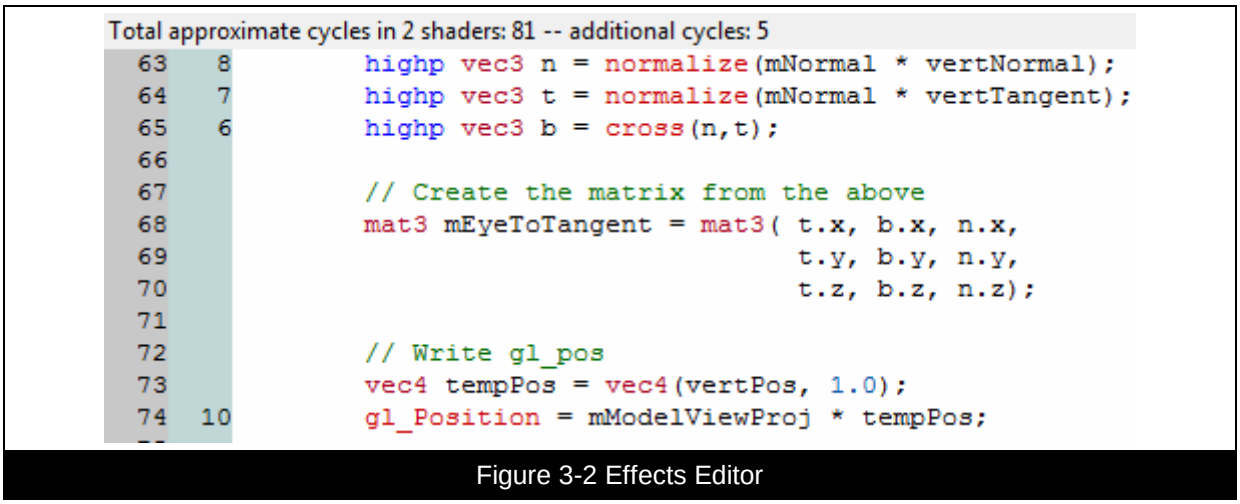

Figure 3-2 Effects Editor

The effects editor is the area used to edit shader code, .pfx and .fx files as well as plain text files. Open files are arranged as tabs so multiple files can be open at any given time. 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. The editor is syntax highlighted, based on the type of file identified (a setting that can be overridden either by right clicking on the tab and selecting the file type or by using `'Compile -> Set File Type'`). The effects editor is also the home of the beginning of PVRUniSCoEditor's profile output.

Cycle Counts

If PVRUniSCoEditor has the correct compiler settings, and the 'Compile in the background' option is ticked in General Preferences, or compilation is performed through the Compile Dialog with the '-profile' flag (by placing a tick in 'Output cycle count per source line') then PVRUniSCoEditor can display approximate cycle count information for the currently open shader.

Three sets of cycle counts are available:

- per-line cycle counts. In the blue column, next to the line numbers, a line by line breakdown of the cycle cost of the shader will be displayed.
- per-shader cycle counts; these counts are available at the beginning of each individual shader within a file and give the overall cost for that shader.
- per-file cycle count; this value appears at the top of the editor window and gives an approximate total cost for all shaders within the current file.

It should be noted that all cycle count values in the effects editor window are approximations intended to assist in identifying areas for optimization. Hardware may be able to reduce these numbers through instruction scheduling etc. that cannot be taken into account by this offline analysis. More accurate information can be found in the Profile Output Panel.

3.1.3. Effects Debug Panel

The Effects Debug Panel contains the compile output of PVRUniSCoEditor's compiler, either via the 'Compile in the background' option set in General Preferences or through the Compile Dialog.

Feedback is broken down per shader with line numbers and error details available when using the PVRUniSCo compiler.

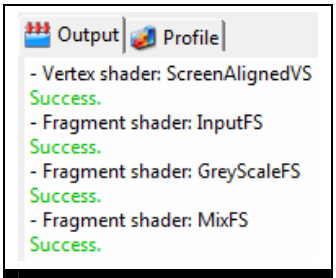

Figure 3-3 Effects Debug Panel

3.1.4. Profile Output Panel

The Profile Output Panel contains more detailed and more accurate profiling information for the shader currently selected from the drop down menu. This information is gained by running the shader through a cycle-accurate emulator.

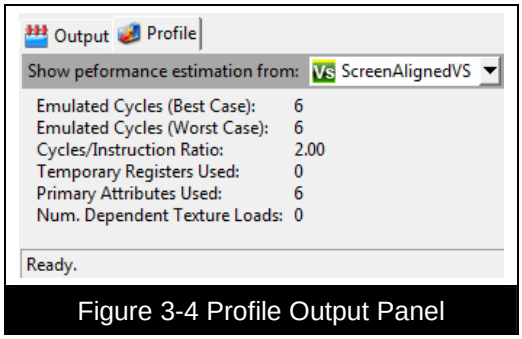

Figure 3-4 Profile Output Panel

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 us, 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).

Non-Dependent Texture Loads

`'Non-Dependent Texture Loads'` represents the number of texture loads within the selected shader that can be pre-fetched; they are 'independent' on the processing of the shader. Dependent texture loads can be calculated based on this value, and identified by the values they use for their 'U' and 'V' co-ordinates, specifically:

- If the co-ordinates of the texture load use the 'Z' and 'W' values of a 'Vec4', even if that 'Vec4' is directly read from a varying, the read will be considered 'dependent'.
- If the co-ordinates are based on a calculation within the shader the read will be considered dependent.

While PowerVR hardware contains many optimisations to reduce the cost of dependent texture reads, they should be avoided whenever possible.

3.2. Menus

3.2.1. File Menu

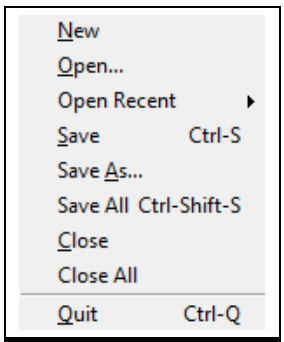

Figure 3-5 File Menu

New

'New' creates a new text file and opens it in the Effects Editor.

Open

'Open…' opens a file for editing.

Open Recent

'Open Recent' contains a list of the ten most recent files opened, clicking on these files will open them for editing.

Save

'Save' saves the file that currently has focus.

Save As

'Save As…' saves the file that currently has focus as a new file.

Save All

'Save All' saves all the currently open files.

Close

'Close' closes the file that currently has focus.

Close All

'Close All' closes all the currently open files.

Quit

'Quit' closes the application.

3.2.2. Edit Menu

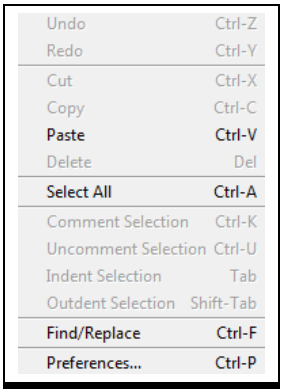

Figure 3-6 Edit Menu

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 from the selected text, either removing a surrounding '/* */' or the '//' at the beginning of the line.

Indent Selection

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

Outdent Selection

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

Find/Replace

'Find/Replace' opens the Find/Replace Toolbar.

Preferences

'Preferences…' opens the Preferences Dialog.

3.2.3. Compile Menu

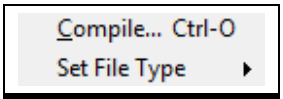

Figure 3-7 Compile Menu

Compile

'Compile…' opens the Compile Dialog.

Set File Type

'Set File Type' is a sub menu which allows the shader class to be chosen; this has the effect of setting the highlighting and compilation type for the shader.

3.2.4. Help Menu

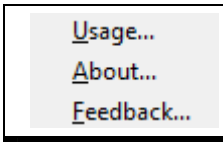

Figure 3-8 Help Menu

Help

'Help…' opens this document.

About

'About…' opens an about page containing version information, contact details etc.

Feedback

'Feedback' opens a panel for giving feedback on the application.

3.3. Toolbars

3.3.1. Main Toolbar

Open File
Opens an effect file or shader file.


Figure 3-9 Open File

Save File
Saves the file currently open in the editor.


Figure 3-10 Save File

Save All
Saves all files open in the editor.


Figure 3-11 Save All

Cut
Cuts the selected text to the clipboard.


Figure 3-12 Cut

Copy
Copies the selected text to the clipboard.


Figure 3-13 Copy

Paste
Pastes text currently in the clipboard.


Figure 3-14 Paste

Find
Opens the Find/Replace Toolbar.


Figure 3-15 Find

Preferences
Opens the Preferences Dialog.


Figure 3-16 Preferences

Undo
Undoes the last action.


Figure 3-17 Undo

Redo
Redoes an action that has been undone.


Figure 3-18 Redo

Compile
Opens the Compile Dialog.

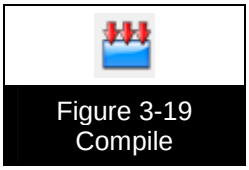

Figure 3-19 Compile

3.3.2. The Tab Bar

Figure 3-20 The Tab Bar

The tab bar gives quick access to all open shader and profile tabs. The currently shown tab is highlighted. The arrow buttons at the sides of the tab bar allow scrolling left and right if there is not enough space to show all the open tabs at once. The down arrow button brings up a list of all open tabs for easier navigation when a large number of tabs are open, as shown above. The `'X'` button closes the currently shown tab.

Right-clicking on a tab brings up a context menu that lets you close the tab or identify the shader class. This has the effect of setting the highlighting and compilation type for the shader.

When enabled in Preferences, a middle-click on a tab will close it. If the file contents have been modified since the last save, a dialog will launch requesting the changes be either saved or discarded.

3.3.3. Find/Replace Toolbar

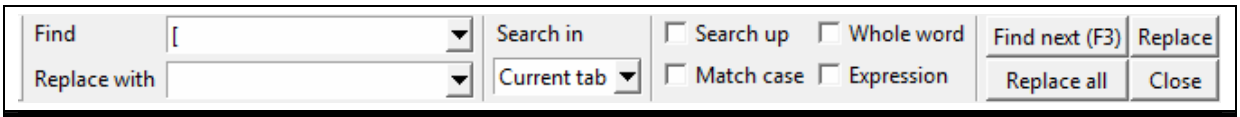

Figure 3-21 Find/Replace Toolbar

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 Find 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.

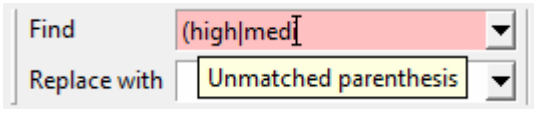

Figure 3-22 Find/Replace Regular Expression Error

Information on what regular expressions are supported can be found in Appendix A. 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.

3.4. Dialogs

3.4.1. Compile Dialog

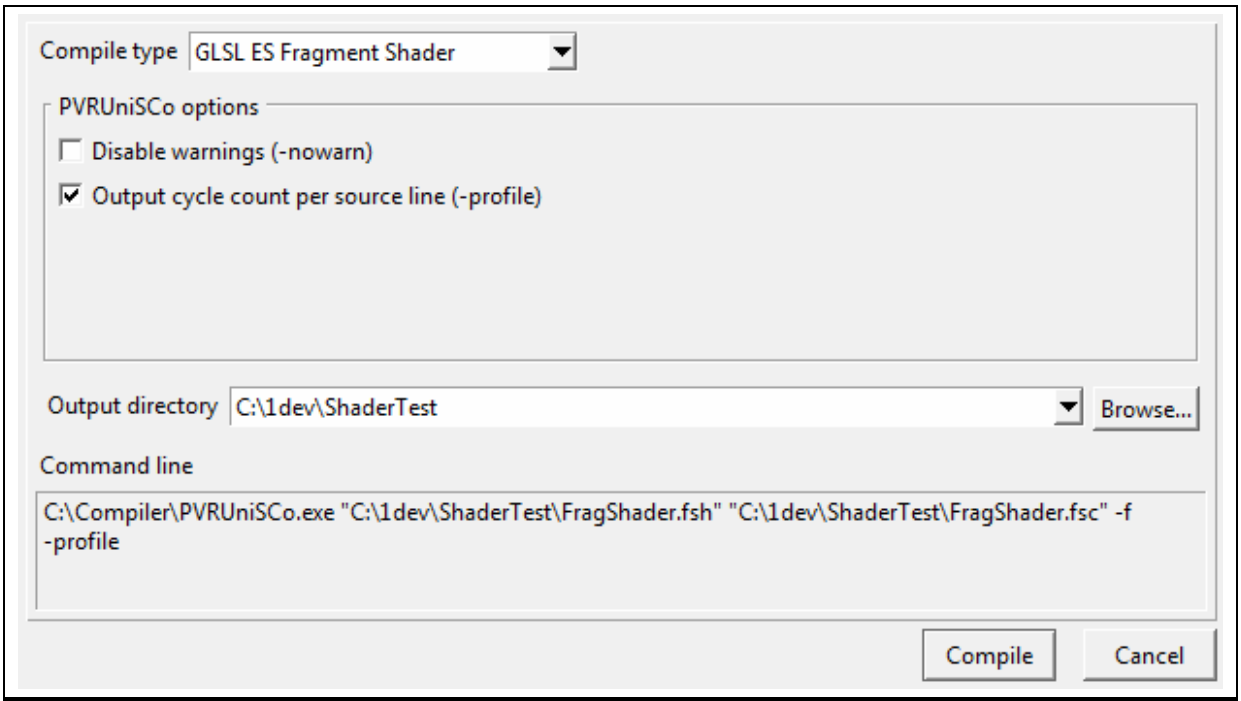

Figure 3-23 Compile Dialog

Compiler Type

'Compile type' states the type of shader to be compiled, as well as giving the option of using a custom shader compiler through the use of the 'Custom command line' option (see Section 3.4.2 Custom Command Line Dialog).

Disable Warnings

'Disable warnings' passes a '-nowarn' flag to the compiler; when using the PVRUniSCo compiler supplied in the PowerVR Insider SDK this option supresses warnings, ensuring that none appear in the Effects Debug Panel.

Output Cycle Count

'Output cycle count per source line' passes a '-profile' flag to the compiler; when used with the PVRUniSCo compiler supplied in the PowerVR Insider SDK this option returns the per line cycle counts as discussed in Section 3.1.2 Effects Editor.

Output Directory

'Output directory' specifies the directory the compiled shader will be output to.

Command Line

'Command line' is a window that contains the final combined command line.

3.4.2. Custom Command Line Dialog

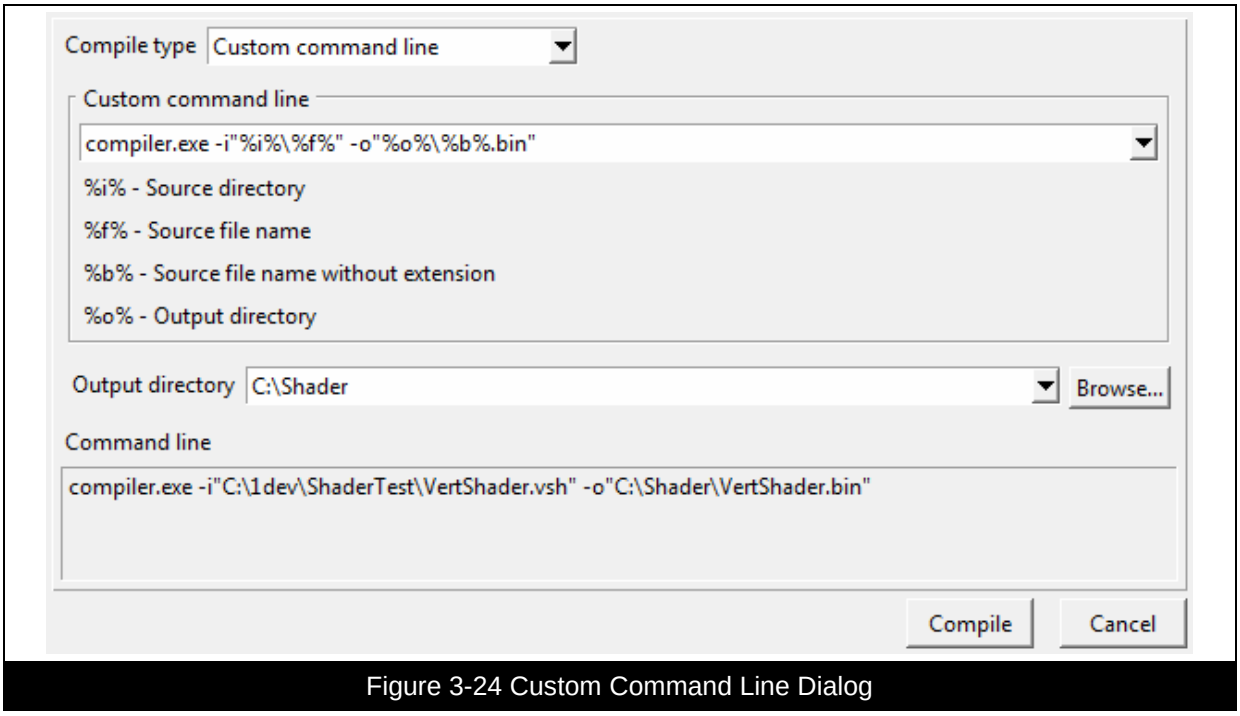

Figure 3-24 Custom Command Line Dialog

By selecting the compile type ‘Custom command line’ in the compile dialog another compiler can be called from within PVRUniSCoEditor. Enter the command line you want in the ‘Custom Command Line’ field. PVRUniSCoEditor will remember the last 10 command lines used.

The following special variables will be replaced depending on the source file and output directory.

Custom Command Line Special Variables	
%i%	Path to directory that contains the source file
%f%	Source filename, without path
%b%	Source file base name, without file extension
%o%	Output directory

The actual command line that will be used can be seen at the bottom of the dialog window in the ‘Command line’ box.

3.4.3. Preferences Dialog

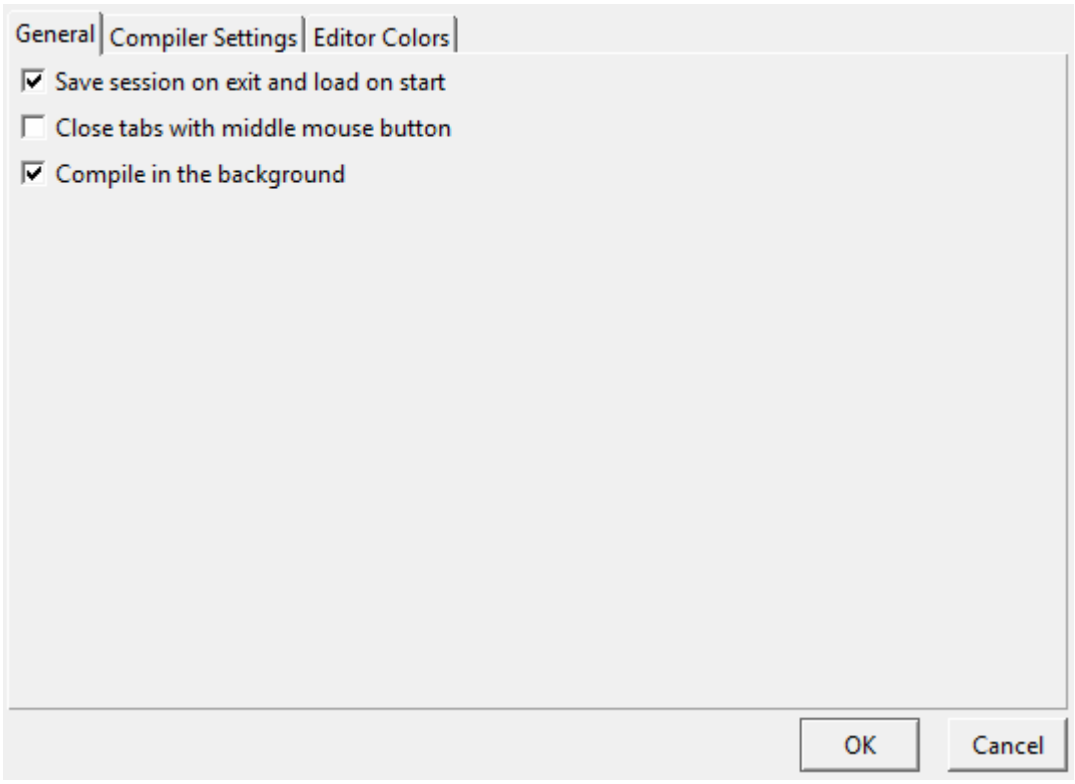

Figure 3-25 Preferences Dialog

The Preferences Dialog can be accessed through the ‘Tools’ menu. Full details on all the available options can be found in Section 4 Preferences.

.

4. Preferences

4.1. General Preferences

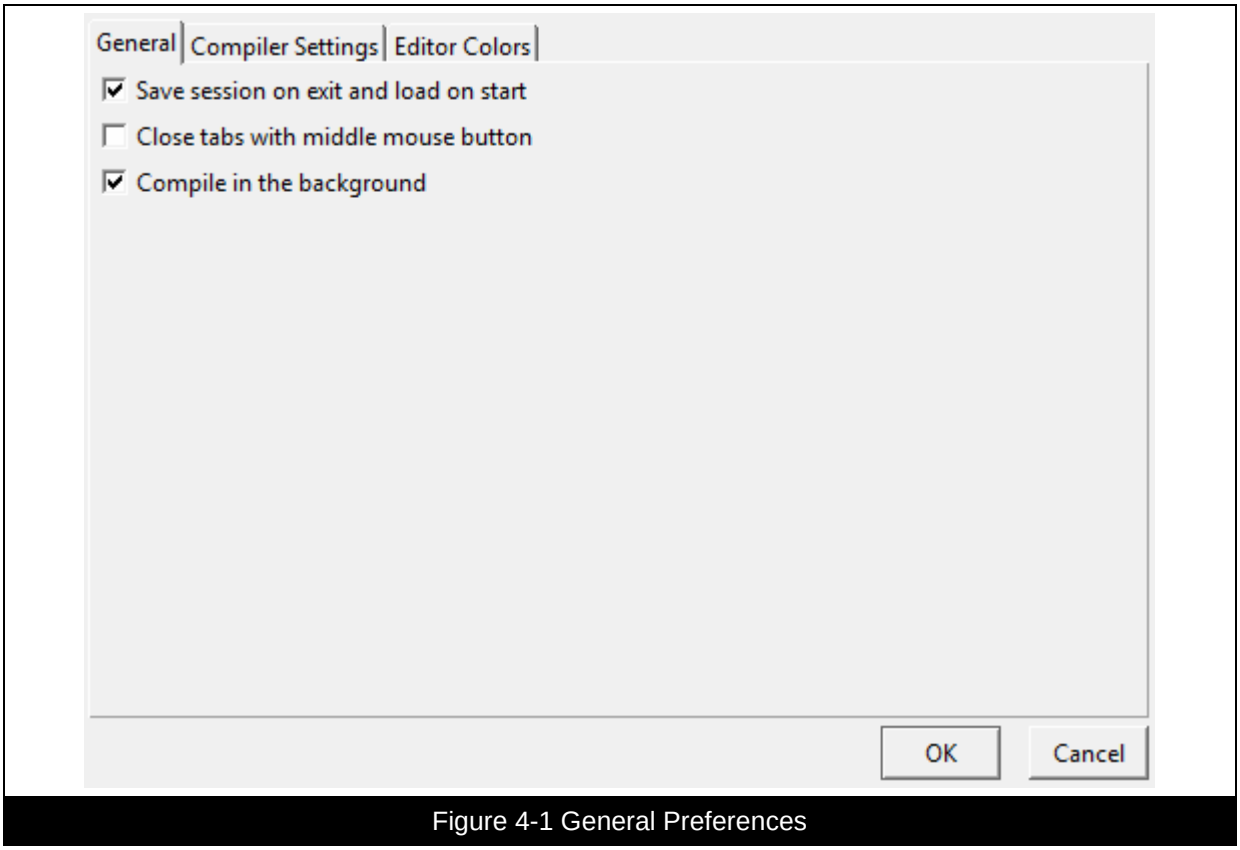

Figure 4-1 General Preferences

Save Session

`'Save session on exit and load on start'` causes PVRUniSCoEditor to save all active files when it is closed, and re-open those same files next time it is launched.

Close Tabs

`'Close tabs with middle mouse button'` allows tabs to be closed with a middle click when activated.

Compile in Background

`'Compile in background'` causes PVRUniSCoEditor to compile the currently active shader in the background, allowing for up to the minute profiling information to be available.

4.2. Compiler Settings

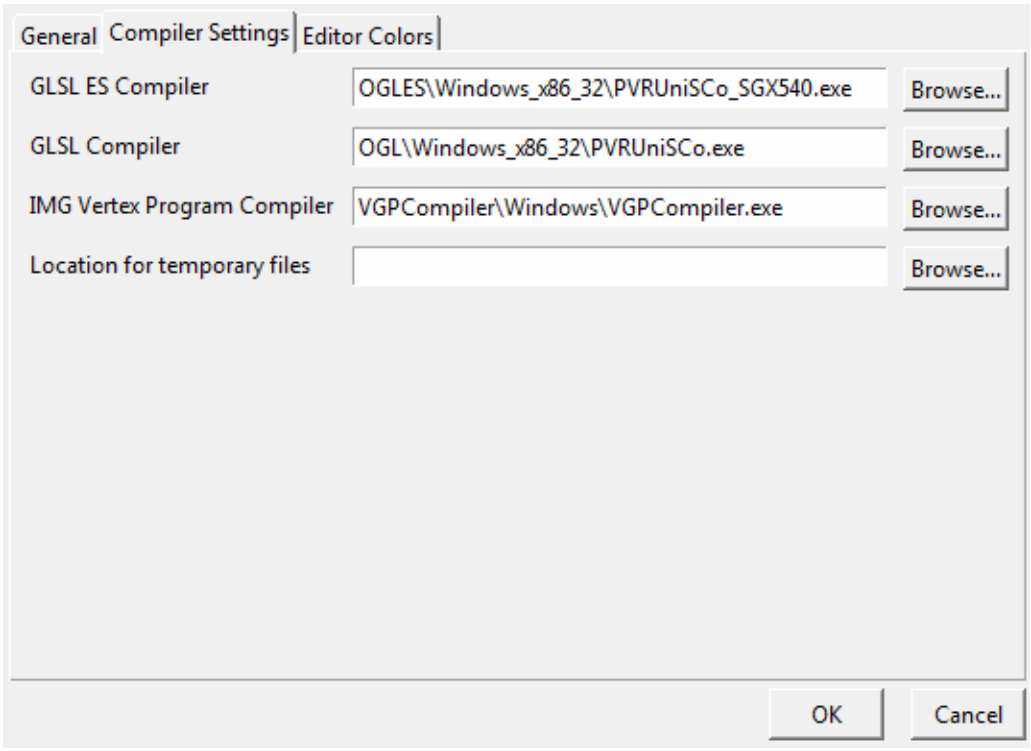

Figure 4-2 Compiler Settings

This window sets the location of the compilers that PVRUniSCoEditor 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.

4.3. Editor Colours

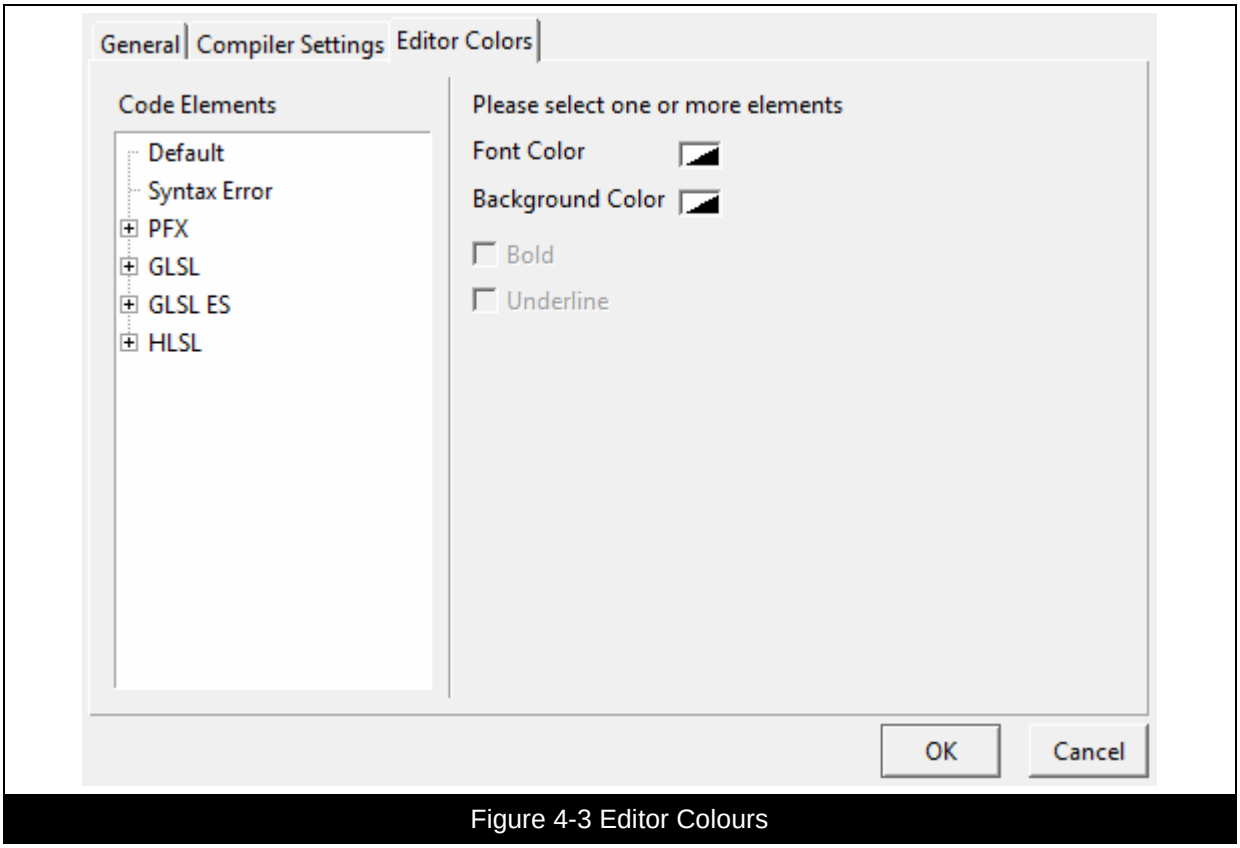

Figure 4-3 Editor Colours

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.

5. Related Materials

Training Courses

- Introducing PFX

Software

- PVRShaman

Documentation

- PVRShaman User Manual
- PVRShaman Getting Started

6. Contact Details

For further support contact:

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

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

www.imgtec.com

Appendix A. 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 1st to 9th capturing group