PVRShaderEditor

User Manual

Filename	:	PVRShaderEditor.User Manual
Version	:	PowerVR SDK REL_3.5@3523383a External Issue
Issue Date	:	17 Apr 2015
Author	:	Imagination Technologies Limited

Contents

List of Figures

List of Tables

1. Introduction

1.1. Document Overview

The purpose of this document is to serve as a complete user manual for PVRShaderEditor. It includes installation instructions, functionality explanations and useful tips on how to make the most out of the application.

1.2. Software Overview

The application is made up of two major components:

- **PVRShaderEditor:** PVRShaderEditor is a shader editor and graphical front-end for the GLSL profiling compilers 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.
- **GLSL Profiling Compilers:** The GLSL profiling compilers are a series of shader compilers that can be used for gaining profiling information for GLSL and GLSL ES shaders. They provide PVRShaderEditor with profiling output and per-line cycle counts to assist in shader development and optimization.

1.3. Installation

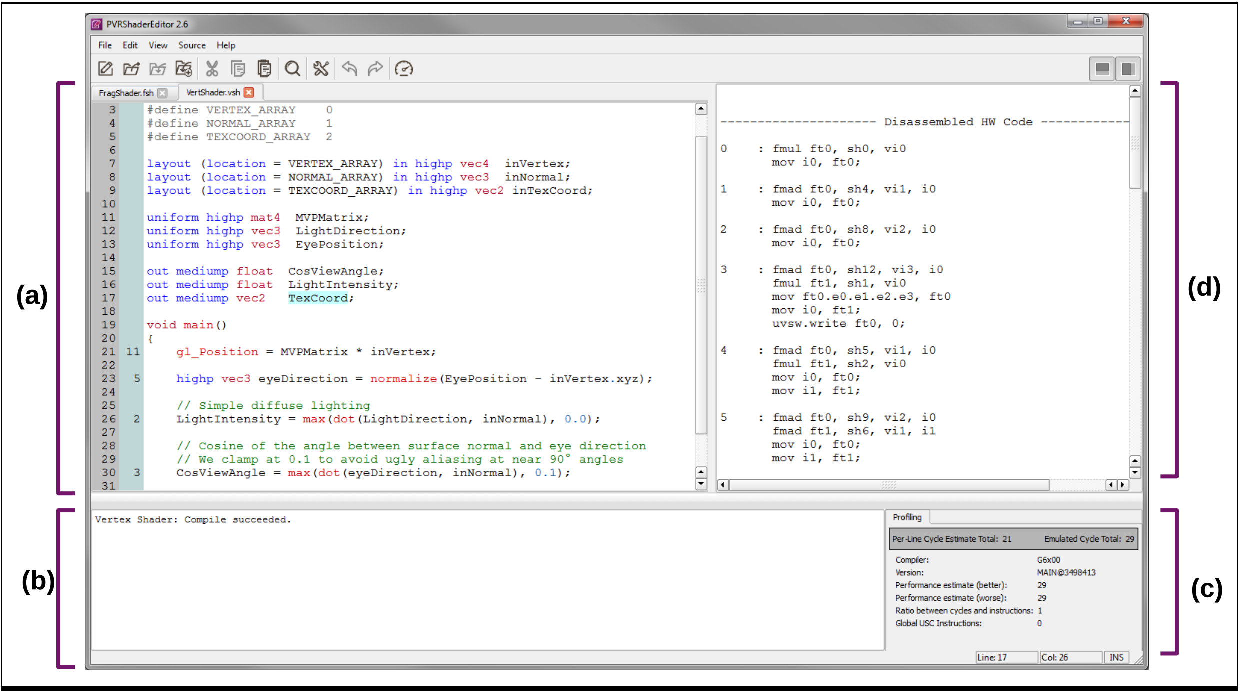
Download the PowerVR Graphics SDK and follow the on-screen instructions. Once the package has successfully installed, the application is available in:

```
<InstallDir>\PVRShaderEditor\<PLATFORM>\
```

2. PVRShaderEditor Graphical User Interface

2.1. User Interface Layout

The main interface of PVRShaderEditor consists of three parts, the `Effects Editor`, the `Effects Debug Panel`, and the `Profile Output Panel`, as displayed in Figure 1. As a tab-based MDI (Multiple Document Interface) application, PVRShaderEditor may have multiple files open simultaneously, displaying a tab for each. In these instances any action performed is performed on the tab that currently has focus.



```
    #define VERTEX_ARRAY    0
    #define NORMAL_ARRAY    1
    #define TEXCOORD_ARRAY  2

    layout (location = VERTEX_ARRAY) in highp vec4  inVertex;
    layout (location = NORMAL_ARRAY) in highp vec3  inNormal;
    layout (location = TEXCOORD_ARRAY) in highp vec2 inTexCoord;

    uniform highp mat4  MVPMatrix;
    uniform highp vec3  LightDirection;
    uniform highp vec3  EyePosition;

    out mediump float  CosViewAngle;
    out mediump float  LightIntensity;
    out mediump vec2   TexCoord;

    void main()
    {
11      gl_Position = MVPMatrix * inVertex;

 5      highp vec3 eyeDirection = normalize(EyePosition - inVertex.xyz);

        // Simple diffuse lighting
 2      LightIntensity = max(dot(LightDirection, inNormal), 0.0);

        // Cosine of the angle between surface normal and eye direction
        // We clamp at 0.1 to avoid ugly aliasing at near 90° angles
 3      CosViewAngle = max(dot(eyeDirection, inNormal), 0.1);
```

```
--------------------- Disassembled HW Code -----------

0    : fmul ft0, sh0, vi0
       mov i0, ft0;

1    : fmad ft0, sh4, vi1, i0
       mov i0, ft0;

2    : fmad ft0, sh8, vi2, i0
       mov i0, ft0;

3    : fmad ft0, sh12, vi3, i0
       fmul ft1, sh1, vi0
       mov ft0.e0.e1.e2.e3, ft0
       mov i0, ft1;
       uvsw.write ft0, 0;

4    : fmad ft0, sh5, vi1, i0
       fmul ft1, sh2, vi0
       mov i0, ft0;
       mov i1, ft1;

5    : fmad ft0, sh9, vi2, i0
       fmad ft1, sh6, vi1, i1
       mov i0, ft0;
       mov i1, ft1;
```

```
Vertex Shader: Compile succeeded.
```

Per-Line Cycle Estimate Total: 21   Emulated Cycle Total: 29

Compiler:	G6x00
Version:	MAIN@3498413
Performance estimate (better):	29
Performance estimate (worse):	29
Ratio between cycles and instructions:	1
Global USC Instructions:	0

Line: 17   Col: 26   INS

Figure 1. Main interface

There are three main aspects to the PVRShaderEditor GUI:

- **Effects Editor:** This is the main area of the GUI that allows for editing shader files (Figure 1a).
- **Effects Debug Panel:** This area of the GUI functions if auto-compile is active, and provides immediate feedback regarding edits in the shader file (Figure 1b).
- **Profile Output and Compiler Settings Panel:** This area of the GUI (Figure 1c) displays profiling information (compiler and language dependent), as well as compiler settings for OpenCL and custom compilers.
- **Disassembly Area:** This area of the GUI shows disassembly code for PowerVR hardware compilers (Figure 1d).

2.2. Menu Bar

2.2.1. File Menu

Selecting `File` on the `Menu Bar` opens up the `File` menu, displayed in Figure 2. The options listed deal with opening and saving files.

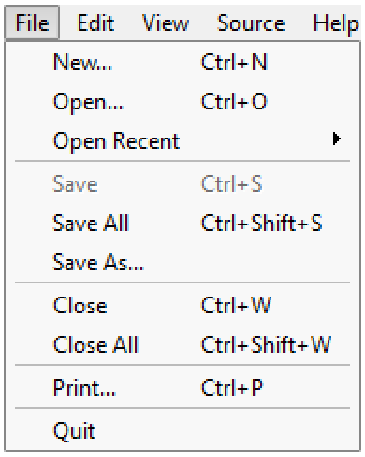


Figure 2. File menu

Create a New File

`New…` creates a new text file and opens it in the `Effects Editor`. The `New` icon in the toolbar can also be used for this function.

Open a File

The `Open…` option in the menu and the `Open File` button on the toolbar open files for editing.

Open a Recent File

`Open Recent` contains a list of the ten most recent files opened. Clicking these files will open them for editing.

Save a File

The `Save` option saves the file that currently has focus. This option will be greyed out if the file cannot be saved for any reason (e.g., the file is marked as read-only). In addition to the menu option, this function can also be triggered by selecting the `Save` icon in the toolbar.

Save All Open Files

`Save All` saves all the currently open files. The `Save All` icon in the toolbar can also be used to perform this function.

Save as a New File

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

Close a Currently Opened File

`Close` closes the file that currently has focus.

Close all Opened Files

`Close All` closes all the currently opened files.

Print a File

Selecting `Print…` will print the file that currently has focus.

Exit the GUI

`Quit` closes the application.

2.2.2. Edit Menu

Selecting `Edit` from the main `Menu Bar` opens the `Edit` menu, illustrated in Figure 3, which is chiefly used when editing the data in the shader files.

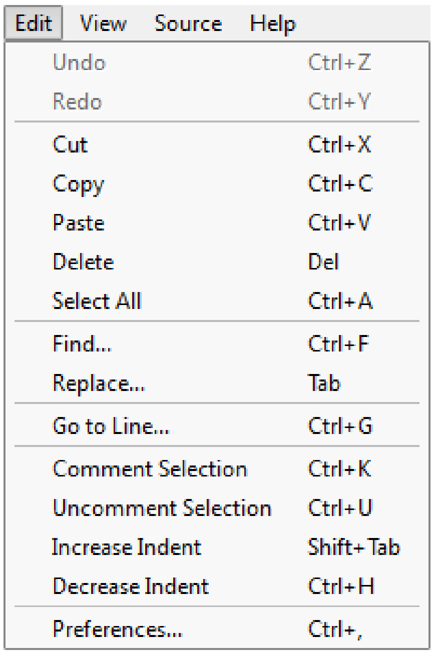


Figure 3. Edit menu

Undo an Action

Selecting `Undo` undoes the last performed action. This function can also be triggered from the `Undo` icon in the toolbar.

Redo an Action

The option `Redo` redoes the last undone action, and can also be achieved by clicking the `Redo` button on the toolbar.

Cut Text

The `Cut` option cuts the selected text to the clipboard. The `Cut` icon on the toolbar can be used as a shortcut for this function.

Copy Text

Selecting `Copy` (or clicking the `Copy` icon on the toolbar) copies highlighted text to the clipboard.

Paste Text

Selecting `Paste` from the menu, or through the toolbar icons, pastes the contents of the clipboard.

Delete Text

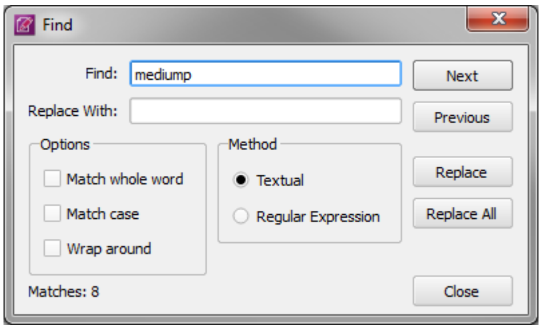
`Delete` deletes the currently selected text.

Select all Files

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

Find and Replace Text

The Find dialog (illustrated in Figure 4) can be accessed via the Find or Replace options in the Edit menu, the Effects Editor Toolbar, or by using the necessary shortcut keys (e.g., on Windows use Ctrl+F to find and Ctrl+H to replace).



Find: mediump
Replace With:
Options: Match whole word [ ], Match case [ ], Wrap around [ ]
Method: [✓ Textual] Regular Expression
Matches: 8

Figure 4. Find dialog

The Regular Expression method enables the interpretation of regular expressions in the Find and Replace with fields. The Find field is shown in red if the search term is not a complete regular expression. Information on which regular expressions are supported can be found in Appendix A. Back references (\1 to \9) can be used in the Replace with field as well as the Find field. The find and replace functionality works as follows:

1. Upon opening the dialog, type in the text to be found in the Find field, and select Next.
2. This will highlight the input text in the displayed shader file.
3. To replace found text, input the new text that is meant to replace the old text in the Replace With field, and select Replace.
4. To replace all instances of the found text, select Replace All.

Commenting Text

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 /* */.

Uncommenting Text

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

Indent Text

Selecting Increase Indent indents the selected text by a single tab.

Decrease Text Indent

The option Decrease Indent decreases the indent on the selected text by a single tab.

User Preferences

Selecting Preferences… from the menu or the Editor Toolbar opens the General Preferences Dialog, detailed in Section 2.3.

2.2.3. View Menu

Selecting `View` from the `Menu Bar` will open the `View` menu, which is primarily used to modify the appearance of the GUI, with the options displayed in Figure 5.

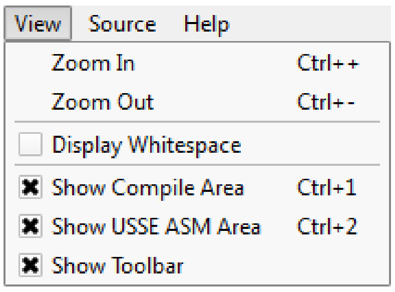


Figure 5. View menu

Zoom in and out

To change the zoom level, select either `Zoom In` to enlarge the size of the text in the `Effects Editor`, or `Zoom Out` to reduce the size of the text in the `Effects Editor`.

Display Whitespaces

Selecting `Display Whitespaces` shows characters in the `Shader Editor` to illustrate white space characters (i.e., spaces, tabs, etc.).

Show or Hide Compile Panel

Toggle the `Show Compile Panel` option to show or hide the `Effects Debug Panel` and the `Profile Output Panel`, detailed further in Sections 2.3.2 and 2.3.4.

Show or Hide Disassembly Panel

Shows or hides the disassembly instructions panel. The display of this panel is dependent on the compiler selected by the user.

Note: Disassembly compilers for PowerVR Series6 hardware family are public and packaged with PVRShaderEditor. The Series5 hardware family disassembly compilers are made available upon signing of an NDA. See Section 3 for details on how to contact us.

Show or Hide Toolbar

This option shows or hides the main toolbar. This displays a variety of shortcut icons that cover the most common user actions.

2.2.4. Source Menu

Choosing the `Source` option in the `Menu Bar` will open the `Source` menu, shown in Figure 6, which is primarily used for optimizing code.

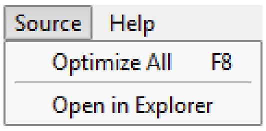


Figure 6. Source menu

Optimize the Code

`Optimize All` optimizes the code in all files for better performance.

Open in Explorer/Finder

Selecting `Open in Explorer` opens the folder which contains the effect file or shader file in Windows Explorer or Finder (for OS X).

2.2.5. Help Menu

To open the `Help` menu (shown in Figure 7), select the `Help` option displayed in the `Menu Bar`.

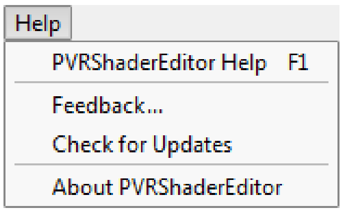


Figure 7. Help menu

View PVRShaderEditor User Manual

To view the “PVRShaderEditor User Manual”, click `Help -> PVRShaderEditorTrace Help`.

Submit Feedback

To provide feedback, click `Help -> Feedback`… This will open a dialog box where instructions are displayed on how to post feedback and request for support.

Check for Updates

As of SDK release 3.0, PVRShaderEditor is able to auto-update. However, to force-check for software updates, click `Help -> Check for Updates`.

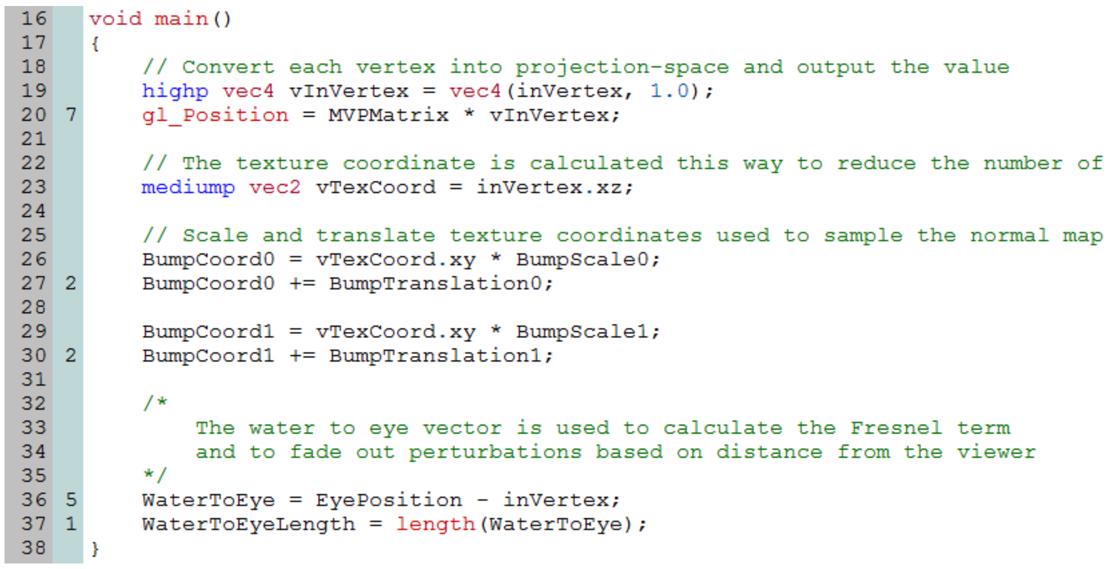
About PVRShaderEditor

To view basic information about PVRShaderEditor release information such as versioning and contact details, click `Help -> About PVRShaderEditor`.

2.3. Editing and Compiling a Shader Program

2.3.1. Edit a Shader Program

The `Effects Editor` is the area used to edit shader code, PFX files, and FX files, as well as plain text files, shown in Figure 1a, and focused on in Figure 8. Opened files are arranged as tabs so multiple files can be used at any given time. The editor is syntax highlighted, based on the type of file identified (a setting that can be overridden by either selecting the file type from the dropdown menu on the bottom bar of the screen). The `Effects Editor` is also the home of the beginning of PVRShaderEditor's profile output.



```
16    void main()
17    {
18        // Convert each vertex into projection-space and output the value
19        highp vec4 vInVertex = vec4(inVertex, 1.0);
20 7      gl_Position = MVPMatrix * vInVertex;
21
22        // The texture coordinate is calculated this way to reduce the number of
23        mediump vec2 vTexCoord = inVertex.xz;
24
25        // Scale and translate texture coordinates used to sample the normal map
26        BumpCoord0 = vTexCoord.xy * BumpScale0;
27 2      BumpCoord0 += BumpTranslation0;
28
29        BumpCoord1 = vTexCoord.xy * BumpScale1;
30 2      BumpCoord1 += BumpTranslation1;
31
32        /*
33            The water to eye vector is used to calculate the Fresnel term
34            and to fade out perturbations based on distance from the viewer
35        */
36 5      WaterToEye = EyePosition - inVertex;
37 1      WaterToEyeLength = length(WaterToEye);
38    }
```

Figure 8. Effects Editor

2.3.2. Automatic Compiling

Automatic compiling is on by default but can be disabled by deselecting the option `Compile shaders in background when possible` (see Section 0).

Cycle Counts

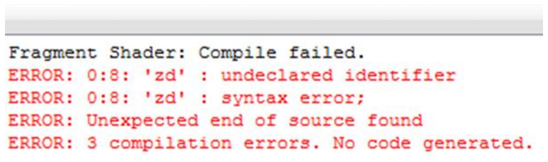
If PVRShaderEditor has the correct compiler settings and the `Compile shaders in background when possible` option is ticked in the preferences, then PVRShaderEditor can display approximate cycle count information for the currently opened shader. Two sets of cycle counts are available:

- **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 in the `Profile Output Panel` and give the overall cost for that shader.

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. It is important to bear in mind that texture sampling is not included in the cost.

Compiler Output

The `Effects Debug Panel` (Figure 1b) contains the compile output of the PVRShaderEditor compiler. Feedback is broken down per shader with line numbers and error details provided, as shown in Figure 9.

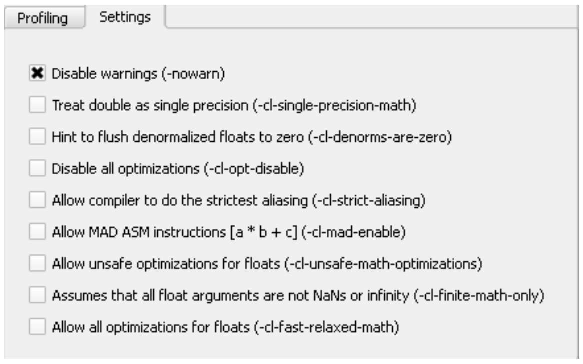


```
Fragment Shader: Compile failed.
ERROR: 0:8: 'zd' : undeclared identifier
ERROR: 0:8: 'zd' : syntax error;
ERROR: Unexpected end of source found
ERROR: 3 compilation errors. No code generated.
```

Figure 9. Effects Debug Panel

2.3.3. OpenCL and Custom Compiler Options

Compiler options are shown in the `Compiler Settings Panel` when choosing the OpenCL compiler or a custom compiler. An example of compiler options for the OpenCL compiler is shown in Figure 10, with their explanation in Table 1.



- [x] Disable warnings (-nowarn)
- [ ] Treat double as single precision (-cl-single-precision-math)
- [ ] Hint to flush denormalized floats to zero (-cl-denorms-are-zero)
- [ ] Disable all optimizations (-cl-opt-disable)
- [ ] Allow compiler to do the strictest aliasing (-cl-strict-aliasing)
- [ ] Allow MAD ASM instructions [a * b + c] (-cl-mad-enable)
- [ ] Allow unsafe optimizations for floats (-cl-unsafe-math-optimizations)
- [ ] Assumes that all float arguments are not NaNs or infinity (-cl-finite-math-only)
- [ ] Allow all optimizations for floats (-cl-fast-relaxed-math)

Figure 10. OpenCL compiler settings

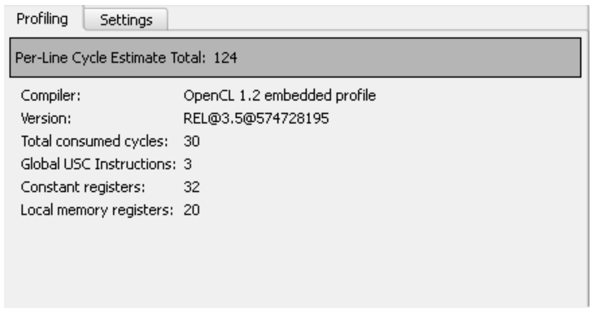
Table 1. OpenCL compiler options

Setting	Detail
`Disable warnings (-nowarn)`	Allows the user to hide warnings.
`Treat double as single precision (-cl-single-precision-math)`	Treat double precision floating-point constants as single precision constants.
`Hint to flush denormalized floats to zero (-cl-denorms-are-zero)`	Hint to the compiler to flush denormalized floating-point numbers to zero.
`Disable all optimizations (-cl-opt-disable)`	Disable all optimizations.
`Allow compiler to do the strictest aliasing (-cl-strict-aliasing)`	Allow the compiler to assume the strictest aliasing rules.
`Allow MAD ASM instructions [a * b + c] (-cl-mad-enable)`	Allow a * b + c to be replaced by a ‘mad’. May reduce accuracy.
`Allow unsafe optimizations for floats (-cl-unsafe-math-optimizations)`	Allow unsafe optimizations for floating-point arithmetic.

Setting	Detail
`Assumes that all float arguments are not NaNs or infinity (-cl-finite-math-only)`	Allow optimizations for floating-point arithmetic that assume that arguments and results are not NaNs or infinity.
`Allow all optimizations for floats (-cl-fast-relaxed math)`	Allow all optimizations for floating-point arithmetic.

2.3.4. View Profiling Information

The `Profile Output Panel` (Figure 11) displays profiling information that is dependent on the chosen compiler and language. Profiling details are available for Series5 GLSL ES, Series6 GLSL ES and Series6 OpenCL.



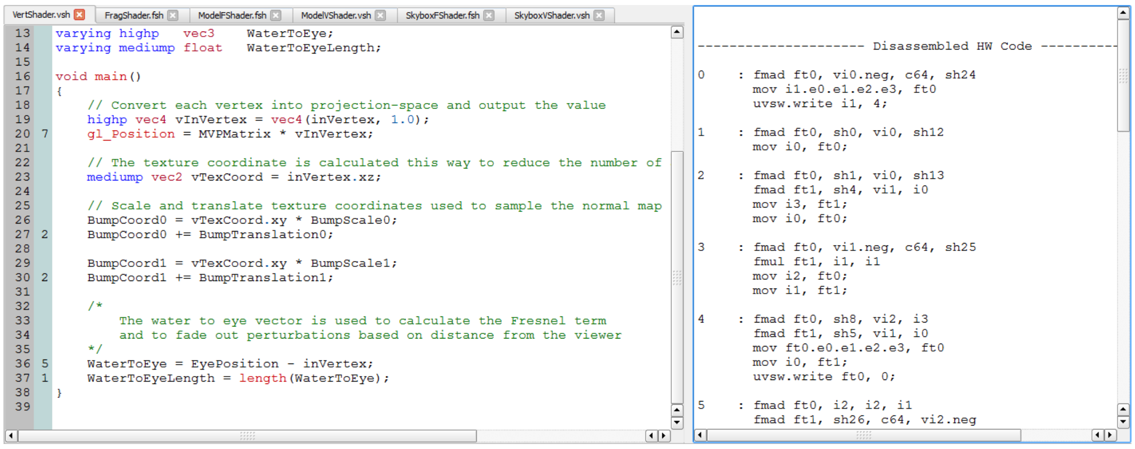
```
Per-Line Cycle Estimate Total:  124

Compiler:                 OpenCL 1.2 embedded profile
Version:                  REL@3.5@574728195
Total consumed cycles:    30
Global USC Instructions:  3
Constant registers:       32
Local memory registers:   20
```

Figure 11. Profile Output Panel

2.3.5. Disassembly Information

PVRShaderEditor supports disassembly information for PowerVR Series6, Series6XT and Series6 FP16, as well as OpenCL compilers for those pieces of hardware. The `Disassembly Panel` (see right hand panel in Figure 12) displays this information for the currently selected shader opened in the `Effects Editor`.



```
   varying highp   vec3    WaterToEye;
   varying mediump float   WaterToEyeLength;

   void main()
   {
       // Convert each vertex into projection-space and output the value
       highp vec4 vInVertex = vec4(inVertex, 1.0);
7      gl_Position = MVPMatrix * vInVertex;

       // The texture coordinate is calculated this way to reduce the number of
       mediump vec2 vTexCoord = inVertex.xz;

       // Scale and translate texture coordinates used to sample the normal map
       BumpCoord0 = vTexCoord.xy * BumpScale0;
2      BumpCoord0 += BumpTranslation0;

       BumpCoord1 = vTexCoord.xy * BumpScale1;
2      BumpCoord1 += BumpTranslation1;

       /*
           The water to eye vector is used to calculate the Fresnel term
           and to fade out perturbations based on distance from the viewer
       */
5      WaterToEye = EyePosition - inVertex;
1      WaterToEyeLength = length(WaterToEye);
   }
```

```
-------------------- Disassembled HW Code ---------

0    : fmad ft0, vi0.neg, c64, sh24
       mov i1.e0.e1.e2.e3, ft0
       uvsw.write i1, 4;

1    : fmad ft0, sh0, vi0, sh12
       mov i0, ft0;

2    : fmad ft0, sh1, vi0, sh13
       fmad ft1, sh4, vi1, i0
       mov i3, ft1;
       mov i0, ft0;

3    : fmad ft0, vi1.neg, c64, sh25
       fmul ft1, i1, i1
       mov i2, ft0;
       mov i1, ft1;

4    : fmad ft0, sh8, vi2, i3
       fmad ft1, sh5, vi1, i0
       mov ft0.e0.e1.e2.e3, ft0
       mov i0, ft1;
       uvsw.write ft0, 0;

5    : fmad ft0, i2, i2, i1
       fmad ft1, sh26, c64, vi2.neg
```

Figure 12. Disassembly information

2.3.6. Shader Optimization

We have included in PVRShaderEditor a full source-level GPU-independent shader optimizer developed by Aras Pranckevičius of Unity Technologies. To access this tool click on the speedo-like icon in the menu bar.

The optimizations include function inlining, dead code removal, copy propagation, constant folding, constant propagation, arithmetic optimizations, etc. The current version supports only OpenGL ES 2.0 (GLSL ES 100). Newer versions of GLSL ES will be supported in future releases.

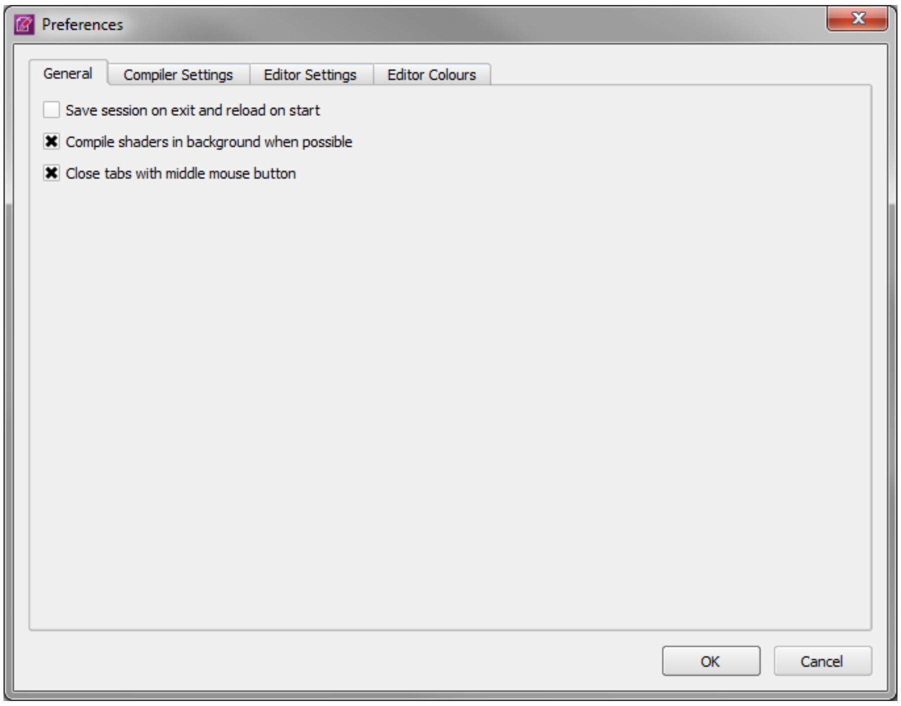
Bear in mind that this tool is a generic optimizer and, in some situations, it might not reduce the number of instructions. In worst case scenarios it might increase the cost of the shader for specific compilers. Please, check whether the results after optimization suit your needs before deploying it.

Follow this link for more information about GLSL Optimizer.

2.4. User Preferences

2.4.1. General Settings

Selecting `Preferences` from the `Edit` menu opens up the `General` tab, illustrated in Figure 13, which is used to toggle options that automate several processes (Table 2).



- [ ] Save session on exit and reload on start
- [x] Compile shaders in background when possible
- [x] Close tabs with middle mouse button

Figure 13. General tab in the Preferences dialog

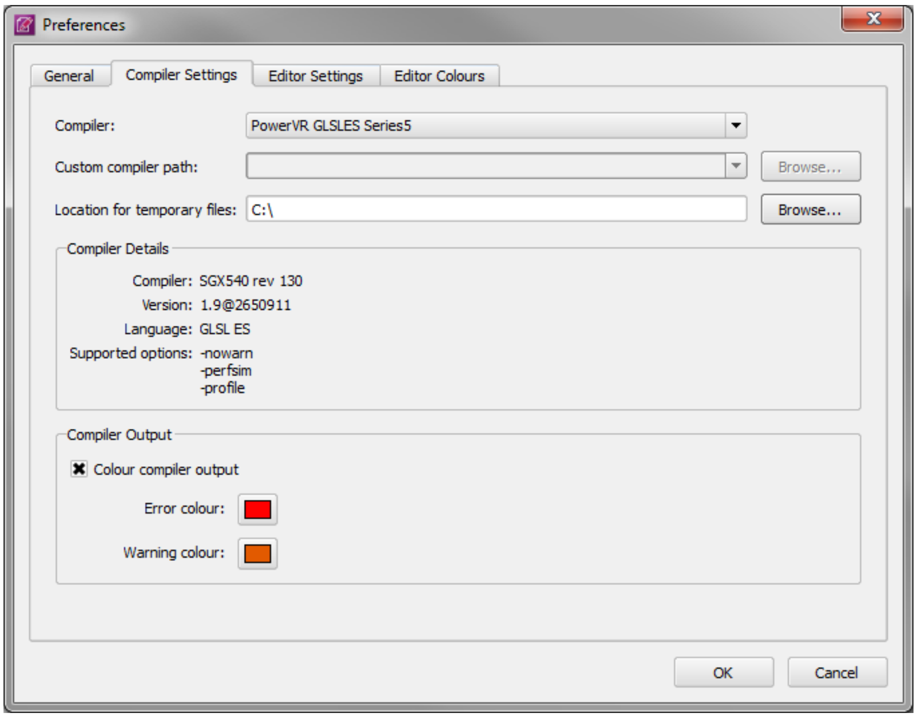
Table 2. General settings

Setting	Detail
`Save session on exit and reload on start`	With `Save session on exit and reload on start` enabled, PVRShaderEditor saves the session on exit and restart. Next time PVRShaderEditor is started the previous session will be reloaded.
`Compile shaders in background when possible`	With the `Compile shaders in background when possible` option enabled, PVRShaderEditor attempts to compile shaders in the background allowing it to display per-line cycle counts and profiling output as the user types.

Setting	Detail
`Close tabs with middle mouse button`	Enabling `Close tabs with middle mouse button` allows middle clicking an opened tab to close it.

2.4.2. Compiler Settings

Compiler settings are accessed by selecting the `Compiler Settings` tab, as shown in Figure 14. Options can be set in order to specify the output of the compiler, as explained in Table 3.


Preferences

General | Compiler Settings | Editor Settings | Editor Colours

Compiler: PowerVR GLSLES Series5

Custom compiler path:

Location for temporary files: C:\

Compiler Details

Compiler: SGX540 rev 130

Version: 1.9@2650911

Language: GLSL ES

Supported options: -nowarn
-perfsim
-profile

Compiler Output

[✓ Colour compiler output]

Error colour: 🟥

Warning colour: 🟧

Figure 14. Compiler Settings tab in the Preferences Dialog

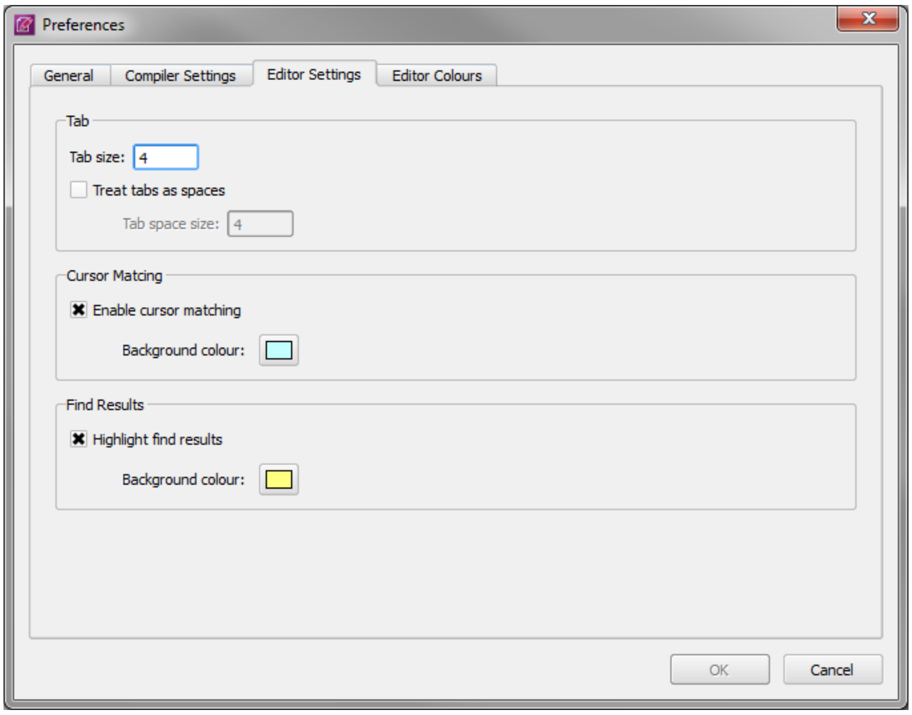
Table 3. Compiler settings

Setting	Detail
`Compiler`	The `Compiler` dropdown box provides a list of compilers from which to choose from. GLSL and OpenCL compilers for different PowerVR hardware are supported. It is also possible to specify a custom compiler.
`Custom compiler path`	`Custom compiler path` allows choosing the path to the compiler when the `Custom Compiler` option is chosen from the `Compiler` dropdown box.
`Location for temporary files`	`Location for temporary files` specifies the directory where PVRShaderEditor temporarily saves the profiling information.

Setting	Detail
`Colour compiler output`	With this option ticked the colouration of compiler warnings and errors can be adjusted. Possible options to choose from are: • **Error colour:** This button opens a colour selection window. The selected colour is used in the `Effects Debug Panel` when an error occurs. • **Warning colour:** This button opens a colour selection window. The selected colour is used in the `Effects Debug Panel` when a warning occurs.

2.4.3. Editor Settings

Editor settings are accessed by selecting the `Editor Settings` tab, which is used to alter the appearance and utility of the `Shader Editor`. Refer to Figure 15 for the appearance of the dialog and to Table 4 for the list of available options.


Preferences

General | Compiler Settings | Editor Settings | Editor Colours

Tab
Tab size: 4
☐ Treat tabs as spaces
Tab space size: 4

Cursor Matcing
☒ Enable cursor matching
Background colour:

Find Results
☒ Highlight find results
Background colour:

Figure 15. Editor Settings tab in the Preferences dialog

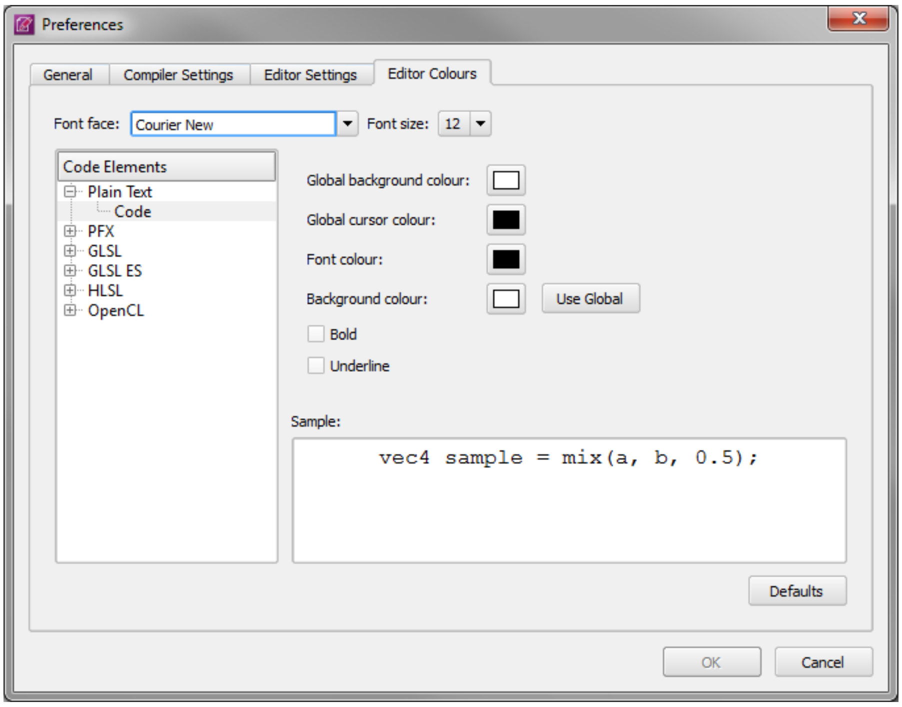
Table 4. Editor settings

Setting	Detail
`Tab`	If `Treat tabs as spaces` is enabled, PVRShaderEditor treats `[tab]` as a sequence of spaces, the exact amount of spaces being the same as the value input into `Tab space size`.
`Cursor Matching`	With the `Enable cursor matching` option enabled, all text in the `Effects Editor` matching the text under the cursor is highlighted in the colour chosen in the `Background Colour` window.

Setting	Detail
Find Results	With the Highlight find results option enabled, text that matches the contents of the Find dialog box is highlighted in the Effects Editor. The Background colour button opens a colour selection dialog box. The selected colour is used in the Effects Editor for highlighting text highlighted by Highlight find results.

2.4.4. Editor Colour Settings

Editor colour settings are accessed by selecting the Editor Colours tab, as displayed in Figure 16. This window allows for controlling the appearance of the Shader Editor including everything from background colour to syntax highlighting colours for the various supported languages, as detailed in Table 5.


Preferences

General | Compiler Settings | Editor Settings | Editor Colours

Font face: Courier New
Font size: 12

Code Elements
- Plain Text
  - Code
- PFX
- GLSL
- GLSL ES
- HLSL
- OpenCL

Global background colour: ⬜
Global cursor colour: ⬛
Font colour: ⬛
Background colour: ⬜ Use Global
☐ Bold
☐ Underline

Sample:

```
vec4 sample = mix(a, b, 0.5);
```

Figure 16. Editor Colours tab in the Preferences dialog

Table 5. Editor Colour settings

Setting	Detail
Font face	Using this dropdown menu allows for the changing of the font being used in the Shader Editor.
Font size	Using this dropdown menu allows for the changing of the size of the font used in the Shader Editor.
Global background colour	This option permits the modification of the background colour in the Shader Editor.
Global cursor colour	This option permits the modification of the cursor colour in the Shader Editor.
Font colour	This option permits the modification of the font colour in the Shader Editor.

Setting	Detail
`Background colour`	This option permits the modification of the text background in the `Shader Editor`.
`Bold`	This option bolds the text in the `Shader Editor`.
`Underline`	This option underlines the text in the `Shader Editor`.

3. Contact Details

For further support, visit our forum:
http://forum.imgtec.com

Or file a ticket in our support system:
https://pvrsupport.imgtec.com

To learn more about our PowerVR Graphics SDK and Insider programme, please visit:
http://www.powervrinsider.com

For general enquiries, please visit our website:
http://imgtec.com/corporate/contactus.asp

Appendix A. Regular Expression Syntax

Table 6 lists the regular expression syntax used by PVRShaderEditor.

Table 6. 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

Special Constructs	
\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