PVRUniSCo Editor

User Manual

Filename	:	PVRUniSCo Editor.User Manual.1.6f.External.doc
Version	:	1.6f External Issue (Package: POWERVR SDK 2.06.26.0557)
Issue Date	:	07 Jul 2009
Author	:	POWERVR

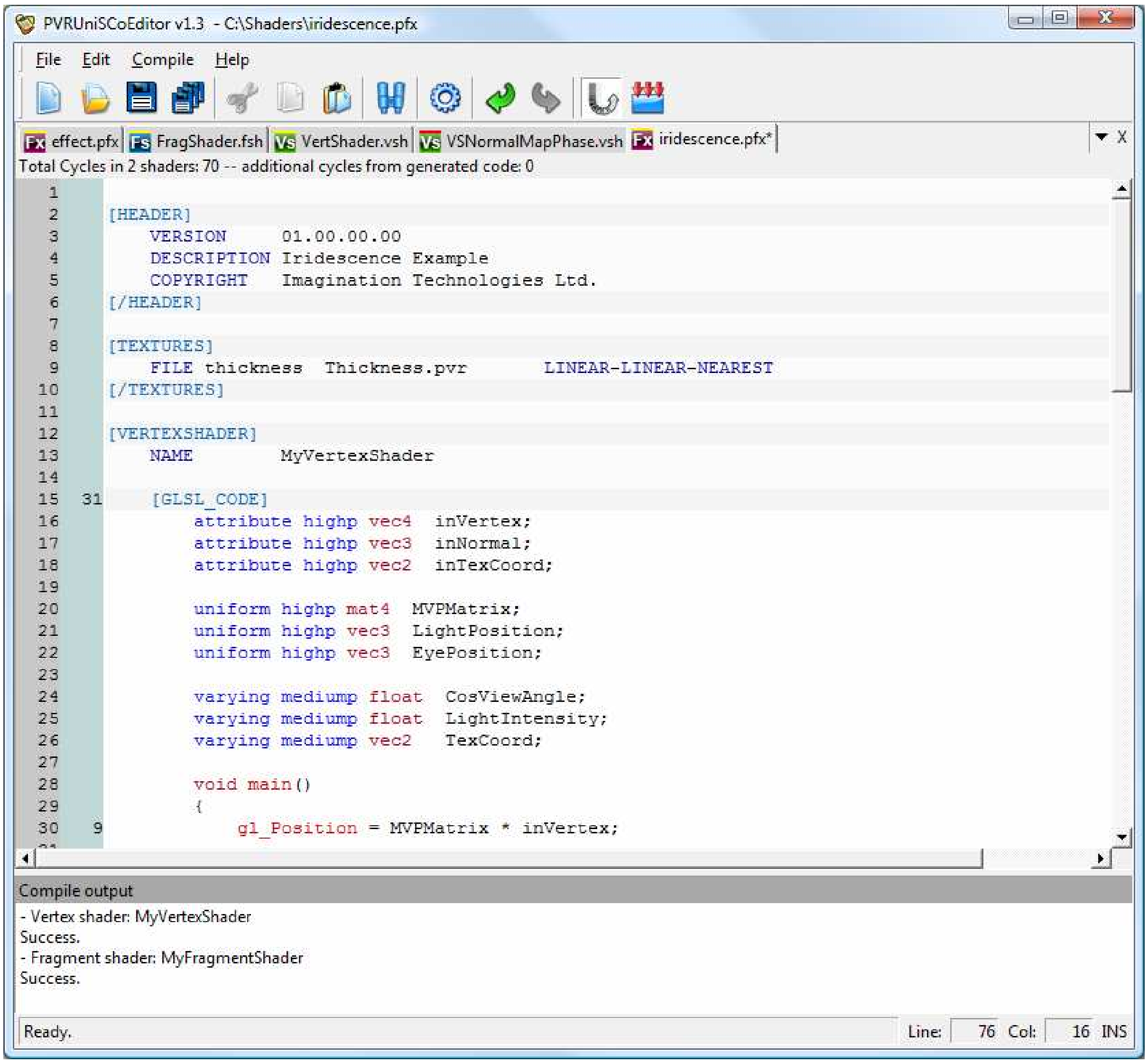
Contents

1. Introduction

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

1. PVRUniSCo Editor Interface

1.1. Overview



Total Cycles in 2 shaders: 70 -- additional cycles from generated code: 0

```

[HEADER]
    VERSION     01.00.00.00
    DESCRIPTION Iridescence Example
    COPYRIGHT   Imagination Technologies Ltd.
[/HEADER]

[TEXTURES]
    FILE thickness  Thickness.pvr       LINEAR-LINEAR-NEAREST
[/TEXTURES]

[VERTEXSHADER]
    NAME        MyVertexShader

31    [GLSL_CODE]
        attribute highp vec4  inVertex;
        attribute highp vec3  inNormal;
        attribute highp vec2  inTexCoord;

        uniform highp mat4  MVPMatrix;
        uniform highp vec3  LightPosition;
        uniform highp vec3  EyePosition;

        varying mediump float  CosViewAngle;
        varying mediump float  LightIntensity;
        varying mediump vec2   TexCoord;

        void main()
        {
9           gl_Position = MVPMatrix * inVertex;
```

```
- Vertex shader: MyVertexShader
Success.
- Fragment shader: MyFragmentShader
Success.
```

PVRUniSCo Editor interface is composed of three parts. The top part consists of the typical menu and tool bars. Below these there is the shader editor with tabs which can be used to edit shaders, POWERVR FX files, and ordinary text files. Open files are arranged as tabs, so multiple files can be open at any time.

At the bottom there is the “Compile log” output panel which shows information about the most recent compilation, including error output from the compiler.

1.2. The Menu Bar

1.2.1. File Menu

This menu contains the options to open, save and close files, and gives access to a list of recently opened files.

1.2.2. Edit Menu

The edit menu contains all the typical text editing commands, copy, cut, paste and delete. In addition, undo/redo, find/replace, comment and uncomment selected text, as well as indent and outdent commands can be reached from here. The preferences window can also be accessed from this menu.

1.2.3. Compile Menu

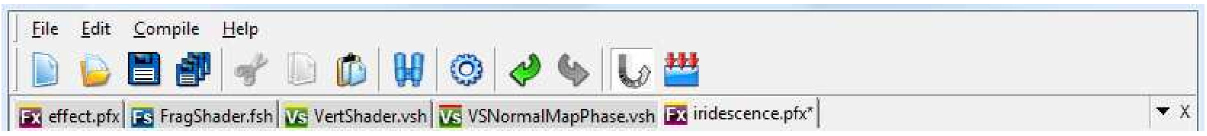
The compile menu gives access to the compile operation as well as the possibility to set the file type of the currently selected tab. This is used to determine the right compilation step for the file in question as well as to enable syntax highlighting. When trying to compile a file that is marked as plain text, you will be prompted to select the compilation type. Selecting Compile will also save the file you are currently working on.

There is also an option to generate per-line cycle counts in the background. See section 2.3 for details.

1.2.4. Help Menu

The help menu provides a shortcut to open this document.

1.3. The Toolbar

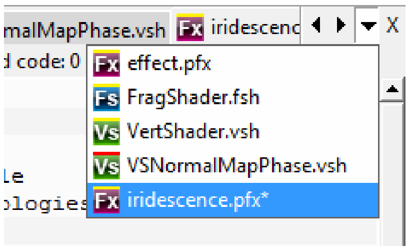


In the initial window layout for PVRUniSCo Editor, the icon toolbar can be found right below the menu bar (described further down in this document) and above the tab bar. The icons have the following meanings (from left to right):

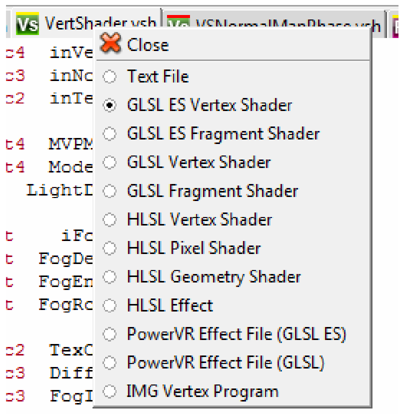
- **New File**
  Creates an empty new tab with type set to plain text (denoted by a white paper sheet icon in the tab bar)
- **Open Files**
  Lets you select one or more files to open from disk.
- **Save Current Tab**
  Saves changes made to the currently selected file to disk.
- **Save All**
  Saves all open files to disk.
- **Cut Selection**
  Moves the currently selected text to the clipboard.
- **Copy Selection**
  Copies the currently selected text to the clipboard.
- **Paste From Clipboard**
  Inserts the text from the clipboard at the current cursor position, replacing any selected text.
- **Find and replace**
  Opens a dockable find and replace dialog window.
- **Preferences**
  Opens the preferences window where you can set compiler paths and editor colors.
- **Undo**
  Undoes previous text changes.
- **Redo**
  Redoes text changes.
- **Generate cycle counts in background**
  Toggles background cycle count generation
- **Compile**
  Compiles the currently selected tab.

1.4. The Tab Bar

The tab bar gives you 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 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 below. The X button closes the currently shown tab.



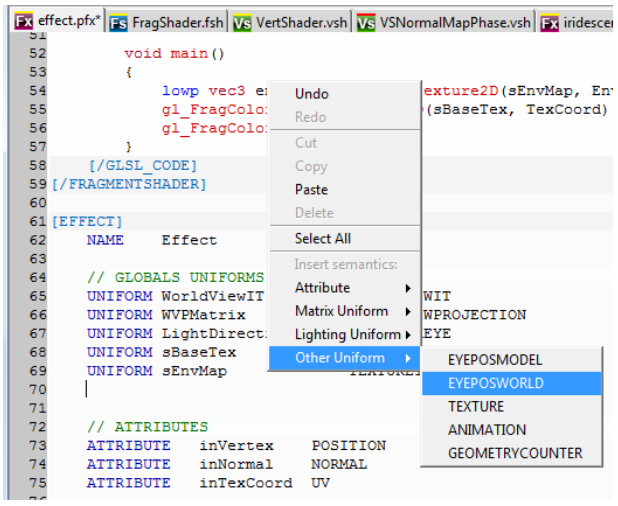
Right-clicking on a tab brings up a context menu that lets you close the tab, or choose the highlighting and compilation type for the shader.



When enabled in Preferences, a middle-click on a tab will close it. If the file contents have been modified since the last save, you will be asked whether you want to save or discard the changes.

1.5. Inserting PFX Semantics

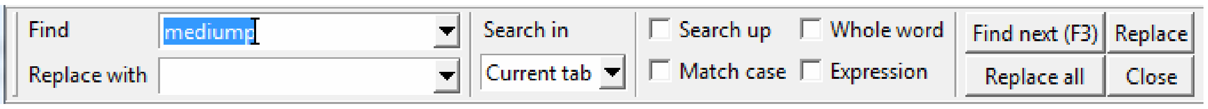
When editing a PFX file you can use the context menu (right-click) to quickly insert PFX semantics as shown in the screenshot below. Selecting one of the menu options will insert a new semantic description for a uniform or attribute at the current cursor location.



```
        void main()
        {
            lowp vec3 e                 exture2D(sEnvMap, En
            gl_FragColo                 (sBaseTex, TexCoord)
            gl_FragColo
        }
    [/GLSL_CODE]
[/FRAGMENTSHADER]

[EFFECT]
    NAME     Effect

    // GLOBALS UNIFORMS
    UNIFORM WorldViewIT                 WIT
    UNIFORM WVPMatrix                   WPROJECTION
    UNIFORM LightDirect                 EYE
    UNIFORM sBaseTex
    UNIFORM sEnvMap
    |

    // ATTRIBUTES
    ATTRIBUTE   inVertex    POSITION
    ATTRIBUTE   inNormal    NORMAL
    ATTRIBUTE   inTexCoord  UV
```

1.6. Find and Replace

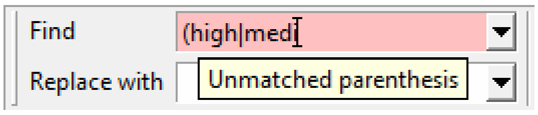
The find and replace dialog can be accessed via the edit menu, the toolbar, or by pressing Ctrl-F. It will appear below the toolbar. Selected text or the word closest to the cursor will automatically become the suggested search term. You can search the current tab, all open tabs, or the current selection, if any.



You can also search up instead of down, search for whole words only, and perform case sensitive search.

1.6.1. Regular Expressions Syntax

When you tick the *Expression* box PVRUniSCo Editor 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.



You can also use back references (`\1` to `\9`) in the *Replace with* field.

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 lookahead
`(?! X )`	Zero width negative lookahead
`(?<= X )`	Zero width positive lookbehind
`(?<! X )`	Zero width negative lookbehind
`(?> 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

Quantifiers	
`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

Predefined Character Classes	
`\U`	Single lowercase character
`\p`	Punctuation (not including '_')
`\P`	Non punctuation

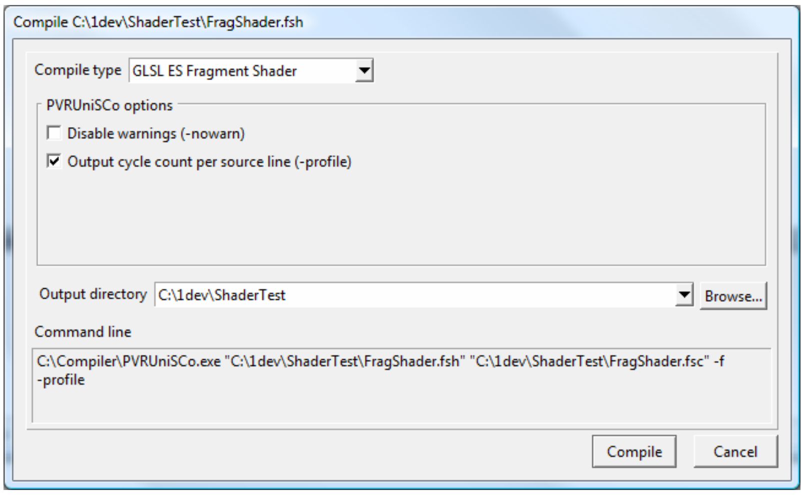
Characters	
`\\`	Back slash character
`\033`	Octal
`\x1b`	Hex
`\t`	Tab
`\n`	Newline

Back References	
`\1` to `\9`	Reference to 1st to 9th capturing group

2. Compiling

2.1. Using the PVRUniSCo Compiler

The PVRUniSCo Compiler can be used for GLSL, GLSL ES and PFX shaders. You have to set the path to the compiler in the Preferences so PVRUniSCo Editor can find the right compiler executable.


Compile C:\1dev\ShaderTest\FragShader.fsh

Compile type: GLSL ES Fragment Shader

PVRUniSCo options
- [ ] Disable warnings (-nowarn)
- [✓] Output cycle count per source line (-profile)

Output directory: C:\1dev\ShaderTest

Command line

```
C:\Compiler\PVRUniSCo.exe "C:\1dev\ShaderTest\FragShader.fsh" "C:\1dev\ShaderTest\FragShader.fsc" -f
-profile
```

When attempting to compile a shader you will be presented with this dialog. Select the compiler options you want

2.2. Showing Source Line Cycle Counts

If you enabled the option *Output cycle count per source line* when compiling and compilation of the shader was successful, the editor will show cycle count information for the shader. Next to the line number column there will appear another column representing the number of USSE instructions that are generated for each source line (this will be blank for lines that generate no instructions).

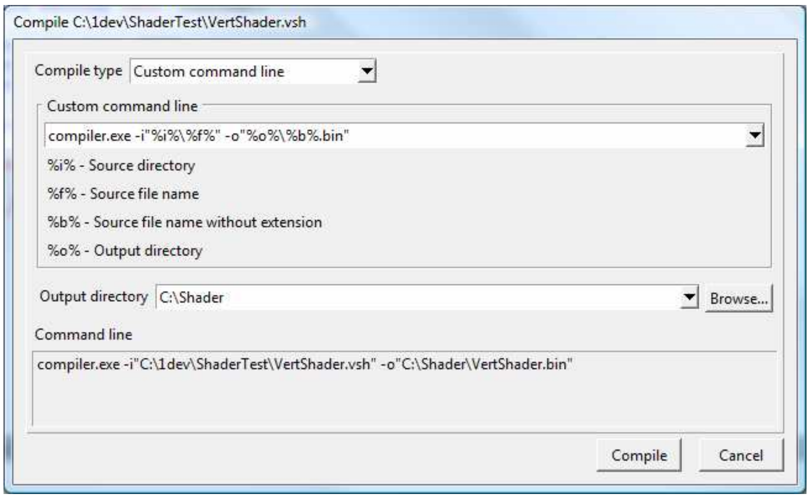
The total number of instructions as well as the number of instructions that could not be associated with a specific source line is presented at the top. For shaders in PFX files the total number of instructions for each shader is shown next to the opening `[GLSL_CODE]` tag. This information will disappear when you modify the shader.

2.3. Generating Cycle Counts in the Background

Per-line cycle counts can also be generated in the background, as you are typing. If this is enabled (either in the Compile menu or from the toolbar), PVRUniSCoEditor will compile the shader you are working on every time you modify it, and generate per-line cycle count information which is shown next to the source. This option uses a temporary file and therefore does not save the changes in the original file. As compilation happens in the background there may be a slight delay between typing and the cycle counts and compile output being updated. This option is only available for GLSL and GLSL ES shaders as well as PFX files, and requires a PVRUniSCo compiler that can generate shader profiling information.

It is important to note that the instruction counts generated are only an approximation of the number of cycles each shader takes to execute. Especially for short shaders the numbers may be inaccurate.

2.4. Custom Command Line



Compile type: Custom command line

Custom command line

compiler.exe -i"%i%\%f%" -o"%o%\%b%.bin"

%i% - Source directory

%f% - Source file name

%b% - Source file name without extension

%o% - Output directory

Output directory: C:\Shader

Command line

compiler.exe -i"C:\1dev\ShaderTest\VertShader.vsh" -o"C:\Shader\VertShader.bin"

By selecting the compile type "*Custom command line*" in the compile dialog you can call another compiler from within PVRUniSCo Editor. Enter the command line you want in the *Custom Command Line* field. PVRUniSCo Editor will remember the last 10 command lines you 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 file name, without path
`%b%`	Source file base name, without file extension
`%o%`	Output directory

You can see the actual command line that will be used at the bottom of the dialog window in the *Command line* box.