

PVRTrace

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 : PVRTrace.User Manual.1.0.12.External.doc
Version : 1.0.12 External Issue (Package: POWERVR SDK REL_3.0@2149525)
Issue Date : 13 Sep 2012
Author : Imagination Technologies Ltd

Contents

1. Introduction	4
1.1. Software Overview.....	4
1.1.1. Recording Libraries	4
1.1.2. PVRTrace GUI	4
1.2. Document Overview	4
1.3. File Types	4
2. Recording Libraries	5
2.1. Overview	5
2.2. Compatibility	5
2.2.1. API.....	5
2.2.2. Supported Extensions	5
2.3. Installation.....	6
2.3.1. Package Installation	6
2.3.2. Windows	6
2.3.3. Linux	6
2.3.4. Android	7
2.4. Configuration	8
2.5. Recording with PVRTrace	10
2.5.1. Network Recording	10
2.5.2. Offline Recording.....	10
2.5.3. Recording using PVRTraceSetup	10
2.6. PVRTraceSetup Overview.....	11
2.6.1. Setup Screen.....	11
2.6.2. Info Screen	11
2.6.3. Options Screen.....	12
3. PVRTrace GUI.....	13
3.1. Overview	13
3.2. Installation.....	13
3.2.1. From Installer	13
3.3. Main Interface.....	14
3.3.1. Frame Summary.....	14
3.3.2. Calls Summary	15
3.3.3. Image Analysis	15
3.3.4. Data Viewer	16
3.3.5. Render State Tab	17
3.3.6. Textures Tab	18
3.3.7. Shaders Tab	18
3.3.8. Function Call List.....	19
3.4. Toolbars.....	20
3.4.1. Data Viewer Toolbar.....	20
3.4.2. Function Call List Toolbar.....	20
3.4.3. Image Analysis Toolbar	21
3.5. Dialogs.....	22
3.5.1. Find.....	22
3.5.2. Filter Functions	22
3.5.3. Remote Controller	23
3.5.4. Image Analysis Preferences	23
3.6. Menus	24
3.6.1. File.....	24
3.6.2. Tools.....	24
3.6.3. Help	24
4. Related Materials	25
5. Contact Details	26
Appendix A. Regular Expression Syntax	27

List of Figures

Figure 2-1 Recording library overview	5
Figure 2-2 Setup Screen	11
Figure 2-3 Info Screen	11
Figure 2-4 Options Screen	12
Figure 3-1 PVRTrace Main Window	14
Figure 3-2 Frame Summary	14
Figure 3-3 Calls Summary	15
Figure 3-4 Image Analysis	15
Figure 3-5 Per-line Cycle Counts	16
Figure 3-6 Render State Tab	17
Figure 3-7 Textures Tab.....	18
Figure 3-8 Shaders Tab	18
Figure 3-9 Function Call List	19
Figure 3-10 Zoom Slider	20
Figure 3-11 Show Draw Calls	20
Figure 3-12 Toggle Filters	20
Figure 3-13 Show Filter Dialog.....	20
Figure 3-14 Find.....	20
Figure 3-15 Mode.....	21
Figure 3-16 Lock Draw Call Values	21
Figure 3-17 Draw Call Selector	21
Figure 3-18 Find Dialog.....	22
Figure 3-19 Filter Functions	22
Figure 3-20 Remote Controller.....	23
Figure 3-21 Image Analysis Preferences.....	23
Figure 3-22 File Menu	24
Figure 3-23 Tools Menu	24
Figure 3-24 Help Menu	24

1. Introduction

1.1. Software Overview

PVRTrace is a scene recording and analysis utility; it captures all the API calls made by an OpenGL ES application as it is running and records the data for analysis at a later date. It consists of two main components, a recording tool and an analysis tool.

1.1.1. Recording Libraries

The recording libraries are shim libraries that are installed on a platform and capture all calls to that platform's native graphics libraries. These calls are captured and written into a PVRT file for reading back by the analysis GUI.

1.1.2. PVRTrace GUI

PVRTrace GUI serves as the analysis interface of PVRTrace allowing 'human-readable' access to the contents of pre-recorded PVRT files.

1.2. Document Overview

The purpose of this document is to serve as a complete user manual for the PVRTrace analysis tool and its associated libraries. It includes compatibility information, installation instructions, a guide to the functionality of the application and a complete listing of all interface options and preferences, broken down by application.

1.3. File Types

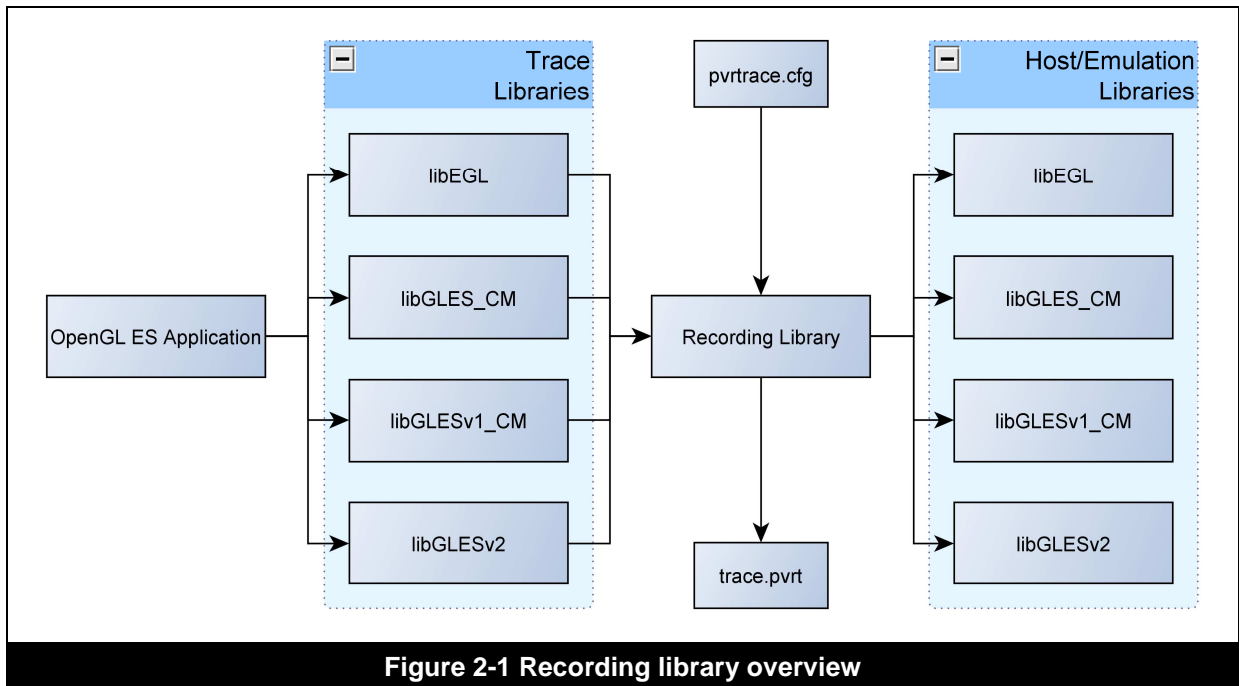
PVRT

A trace of a given application output by PVRTrace recording libraries at runtime.

2. Recording Libraries

2.1. Overview

Recording of a trace is performed by multiple libraries; shim libraries for each graphics API, and one core recording library. Once installed on a platform, the shim libraries intercept graphics API calls and send them to the recording library to write them to a file. The calls are then passed on to the host libraries, as stated in the 'pvrtrace.cfg' file. Calls are streamed to the PVRT file during runtime so that even if an application crashes, data will still be recorded.



2.2. Compatibility

2.2.1. API

The PVRTrace recording libraries are compatible with EGL, OpenGL ES 1.1 and OpenGL ES 2.0.

2.2.2. Supported Extensions

OpenGL ES 1.1	OpenGL ES 2.0
GL_OES_vertex_array_object	GL_OES_vertex_array_object
GL_EXT_multi_draw_arrays	GL_EXT_multi_draw_arrays
GL_OES_mapbuffer	GL_OES_mapbuffer
GL_EXT_discard_framebuffer	GL_EXT_discard_framebuffer
GL_OES_framebuffer_object	
GL_OES_fixed_point	
GL_OES_EGL_image_external	
GL_OES_matrix_palette (GL_OES_extended_matrix_palette)	

2.3. Installation

2.3.1. Package Installation

From Installer

Download the PowerVR Insider SDK and follow the on screen instructions. Once the package has successfully installed the recording libraries can be found within the PVRTrace folder in the install directory as follows:

```
<InstallDir>\PVRTrace\Recorder\<API>\<PLATFORM>\
```

2.3.2. Windows

1. Copy: 'libEGL.dll', 'libGLESv2.dll' (OpenGL ES 2.0) or 'libGLES_CM.dll' and 'libGLESv1_CM.dll' (OpenGL ES 1.1), and 'PVRTrace.dll'
To: Executable folder
2. If 'pvrtrace.cfg' does not exist, make one manually (see Section 2.4 Configuration).
3. Place 'pvrtrace.cfg' into the executable folder.
4. Update 'pvrtrace.cfg' so that EglLibraryPath, Es1LibraryPath and Es2LibraryPath are set to the location of the systems graphics libraries (see Section 2.4 Configuration).

2.3.3. Linux

1. Copy: 'libEGL.so', 'libGLESv2.so' (OpenGL ES 2.0) or 'libGLES_CM.so' and 'libGLESv1_CM.so' (OpenGL ES 1.1), and 'libPVRTrace.so'
To: Any folder
2. Set the LD_LIBRARY_PATH to the folder containing the new libraries.
 - a. This can be set globally using 'set LD_LIBRARY_PATH=...' or 'export LD_LIBRARY_PATH=...', and reverted once the trace is complete.
 - b. This can also be set by using 'LD_LIBRARY_PATH=... ./<APPLICATION_NAME>' as part of running the trace.
3. If 'pvrtrace.cfg' does not exist, make one manually (see Section 2.4 Configuration).
4. Place 'pvrtrace.cfg' into the executable folder.
5. Update 'pvrtrace.cfg' so that EglLibraryPath, Es1LibraryPath and Es2LibraryPath are set to the location of the systems graphics libraries (see Section 2.4 Configuration).

2.3.4. Android

Using PVRTraceSetup

The PVRTraceSetup APK can be found in the following folder:

```
<InstallDir>\PVRTrace\Recorder\Installer\<PLATFORM>\
```

Once this APK has been installed on a rooted device it can be used to control and setup the Recording Libraries with no manual editing of the pvrtrace config file or filesystem. Using PVRTraceSetup is the preferred method of controlling the Android Recording Libraries.

Manual Installation

1. Copy: `'libPVRTrace.so'`
To: `/system/lib`
2. Copy: `'libEGL_PVRTRACE.so'`, `'libGLESv2_PVRTRACE.so'` (OpenGL ES 2.0) or `'libGLESv1_CM_PVRTRACE.so'` (OpenGL ES 1.1).
To: `/system/lib/egl` or `/system/vendor/lib/egl` (Android 2.3 onwards)
Specifically, the folder the OpenGL ES driver libraries are located in.
3. If `'pvrtrace.cfg'` does not exist, make one manually (see Section 2.4 Configuration).
4. Place `'pvrtrace.cfg'` into the root of the file system (`/`) or the SD card (`/mnt/sdcard`). It should be noted that the root of the file system (`/`) will be checked first and used in preference over the root of the SD Card.
5. Update `'pvrtrace.cfg'` so that `EglLibraryPath`, `Es1LibraryPath` and `Es2LibraryPath` are set to the location of the systems graphics libraries (see Section 2.4 Configuration).
6. File permissions are tightly regulated on Android, typically you can only write to a location either on an SD Card, or the process folder for your app (`/data/data/<PROCESS_NAME>/`). So the "TraceFile" variable must be set accurately with an absolute path. If it isn't correctly set, no file will be output. It is suggested that you check write permissions for your app before setting this. As all subsequent processes will be traced after installation, it is recommended that you use an output folder that allows write access for all processes so that PVRTrace output doesn't fail and potentially threaten the stability of your device.
7. Edit `'egl.cfg'` in `/system/lib/egl/` to use PVRTrace instead of the current driver. For example if the original file stated:

```
0 0 android
0 1 POWERVR_SGX_540_120
```

Edit it to:

```
0 0 android
0 1 PVRTRACE
```

This change can be done at any time and will only affect new processes that are launched after the change. It is recommended that two copies of `'egl.cfg'` be used, one referencing the PVRTrace libraries, one referencing the default libraries; each file should be named separately and copied over the original `'egl.cfg'` as required.

8. Once a trace is complete the changes to `'egl.cfg'` must be reverted to avoid any undesired behaviour. This is particularly important before rebooting your device as the OpenGL ES libraries are required to draw the UI, and if these are traced it can cause issues at boot time.

WARNING: If the `'egl.cfg'` file is inaccurate and points at libraries that do not exist or are in the wrong location and the device is rebooted it will be unable to boot correctly. If this happens the device may need to be reflashed unless external access to the file system is available.

2.4. Configuration

The PVRTrace recording libraries require the configuration file 'pvrtrace.cfg' to be present. If this file is not present then one will be created the first time the application being traced is run. An example of a configuration file can be found below:

```
EglLibraryPath = %SYSTEMDIR%\libEGL.dll
Es1LibraryPath = %SYSTEMDIR%\libGLES_CM.dll
Es2LibraryPath = %SYSTEMDIR%\libGLESv2.dll

TraceFile = trace-%pid.pvrt
RecordData = 0
StartFrame = 0
EndFrame = 100

Network = 1
NetworkWait = 1
SaveFileToDisk = 0
NetworkBufferSize = 65535
```

EglLibraryPath

'EglLibraryPath' refers to the location of the original (non-PVRTrace) EGL library on the host system; this is usually found in /system/lib/egl or /system/vendor/lib/egl. The library in /system/lib should not be used as it is merely a 'shim' library used to redirect calls to the correct library based on the settings in 'egl.cfg'. On Windows, environment variables can be used (such as %SYSTEMDIR%). This functionality is not supported on Linux or Mac OS.

Es1LibraryPath

'Es1LibraryPath' refers to the location of the original (non-PVRTrace) OpenGL ES 1.1 library on the host system; this is usually found in /system/lib/egl or /system/vendor/lib/egl. The library in /system/lib should not be used as it is merely a 'shim' library used to redirect calls to the correct library based on the settings in 'egl.cfg'. On Windows, environment variables can be used (such as %SYSTEMDIR%). This functionality is not supported on Linux or Mac OS.

Es2LibraryPath

'Es2LibraryPath' refers to the location of the original (non-PVRTrace) OpenGL ES 2.0 library on the host system; this is usually found in /system/lib/egl or /system/vendor/lib/egl. The library in /system/lib should not be used as it is merely a 'shim' library used to redirect calls to the correct library based on the settings in 'egl.cfg'. On Windows, environment variables can be used (such as %SYSTEMDIR%). This functionality is not supported on Linux or Mac OS.

TraceFile

'TraceFile' sets the name and location of the trace file that will be output by the recording libraries. The process ID of an application can be used as part of its trace file name by using %pid in this field. It should be noted that applications being traced with this config file must have permission to access the location given or no trace file will be output.

RecordData

This option states whether the trace libraries should record the data associated with each call or not. With 'RecordData' set to '1', texture information and buffer data etc. is recorded along with the graphics API calls. With 'RecordData' set to '0' only the graphics API calls are recorded. In general, 'RecordData' should be set to '1' unless file size is a problem.

StartFrame

'StartFrame' states the frame at which recording should begin.

EndFrame

'EndFrame' states the frame at which recording should stop.

Network

With this flag set to '1' the PVRTrace GUI will be able to connect to the application over a network allowing for remote recording. It should be noted that while this flag is set to '1' the 'StartFrame' and 'EndFrame' flags will be ignored. With this flag set to '0' offline recording will be performed and the 'NetworkWait', 'SaveFileToDisk', and 'NetworkBufferSize' flags will be ignored.

NetworkWait

With this flag set to '1' the application being recorded will pause during the first OpenGL call until an instance of the PVRTrace GUI has connected and sent the command to continue playing. With this flag set to '0' the application being recorded will continue to play regardless of whether a client has connected or not.

SaveFileToDisk

This flag set to either '1' or '0', states whether the recording libraries should still save the trace to a local file while remote recording is set. If this flag is set to '1' then a local recording will still be performed and output to the location set in 'TraceFile'; if this flag is set to '0' no output file will be created.

NetworkBufferSize

'NetworkBufferSize' sets the volume of data that will be sent, from the recording libraries to the PVRTrace GUI, in each packet. The default value is '256'.

2.5. Recording with PVRTrace

2.5.1. Network Recording

1. Install the PVRTrace libraries on the target device, as per the installation instructions (see Section 2.3 Installation).
2. Ensure that `'egl.cfg'` and `'pvrtrace.cfg'` are set correctly (see Section 2.4 Configuration).
3. Ensure that `'pvrtrace.cfg'` has `'Network'` set to `'1'`.
4. If required, set `'NetworkWait'` to `'1'`. This allows the initial setup calls of the application to be remotely recorded, but will cause the application to become unresponsive until `'Play'` is pressed in the Remote Controller.
5. If an offline recording is desired in addition to the remote recording, set `'SaveFileToDisk'` to `'1'` and ensure that the application has permission to write to the output location set under `'TraceFile'` in `'pvrtrace.cfg'`.
6. Run the PVRTrace GUI and launch the Remote Controller by clicking `'File -> Show Network Panel'`.
7. Set an output file for the recording by clicking the `'Output File Trace'` button.
8. Enter the IP address of the target device in the `'Server'` box.
9. Run the application to be traced.
10. Connect to the target device by clicking the `'Connect'` button.
11. Click the `'Record'` button at any time to begin recording, if `'NetworkWait'` has been set to `'1'`, click the `'Play'` button to signal the application to continue running.
12. Click `'Stop'` to end recording.
13. Click `'Open Trace'` to open the recording and begin analysis.

2.5.2. Offline Recording

1. Install the PVRTrace libraries on the target device, as per the installation instructions (see Section 2.3 Installation).
2. Ensure that `'egl.cfg'` and `'pvrtrace.cfg'` are set correctly (see Section 2.4 Configuration) and that the application has permission to write to the output location set under `'TraceFile'` in `'pvrtrace.cfg'`.
3. Ensure that `'pvrtrace.cfg'` has `'Network'` set to `'0'`.
4. Run the application to be traced.
5. Once the application has exited copy the output file set in `'pvrtrace.cfg'` into a location accessible from the PVRTrace GUI.
6. Run the PVRTrace GUI, and open the output file to begin analysis.

2.5.3. Recording using PVRTraceSetup

1. Install the PVRTraceSetup APK on the target, rooted, device.
2. Run PVRTraceSetup and click `'Install'`
3. If using network recording, click on `'Options'`, click on `'Network'`, click on `'Network Wait'` if desired.
4. On the `'Options'` screen select the desired start and end frame, and set the output file.
5. On the `'Setup'` page click `'Start Tracing'` to trace all application launched after the button is clicked, then launch whichever applications you wish to trace.
6. If you wish to trace a specific application click `'Trace an Application'`, this will launch the desired application and close it once the end frame is reached.
7. Run the PVRTrace GUI, and either open the output file, or follow steps 6 – 13 of Network Recording above.

2.6. PVRTraceSetup Overview

PVRTraceSetup is an application for Android that allows for the control and setup of the PVRTrace Recording Libraries so that manual setup and control can be avoided.

2.6.1. Setup Screen

Install

Installs the PVRTrace Recording Libraries

Uninstall

Removes the PVRTrace Recording Libraries

Start/Stop tracing

'Start Tracing' sets the PVRTrace Recording Libraries as the currently active graphics libraries; from the point of activation onwards any application launched will be traced.

'Stop tracing' restores the original graphics libraries, disabling recording for any application launched after 'Stop tracing' is hit.

Trace an application

'Trace an application' sets the PVRTrace Recording Libraries as the current active graphics libraries for a specific application, launches that application, then exits it once the end frame set in the Options Screen is hit. Only the selected application will be traced.

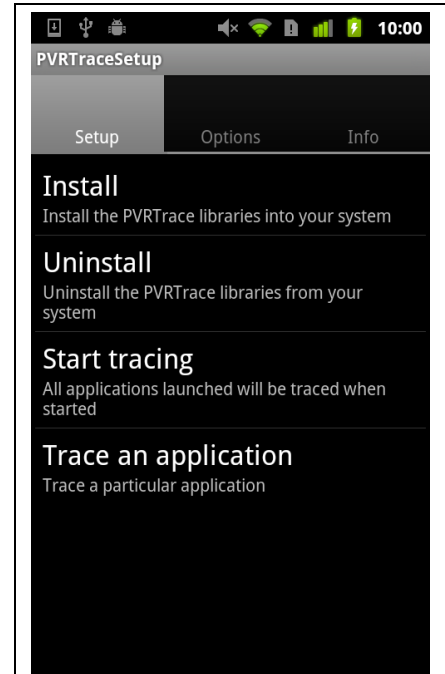


Figure 2-2 Setup Screen

2.6.2. Info Screen

The Info Screen provides information as the platform PVRTraceSetup is installed on, the SDK version the recording libraries are from, and the version of the PVRTrace being used. It also shows the location of the host graphics libraries.

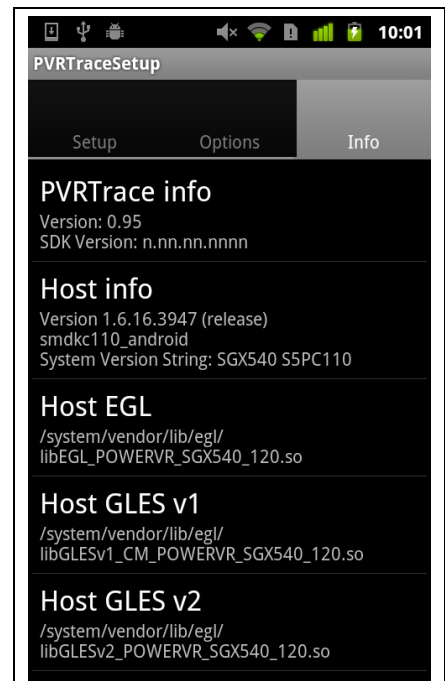


Figure 2-3 Info Screen

2.6.3. Options Screen

Set Trace Output File

The location and file name of the output trace file.

Start Frame

The first frame of a traced application to record.

End Frame

The last frame of a traced application to record.

Record Data

With this option ticked the PVRTrace Recording Libraries will record all buffer contents, all render state changes, every shader, every texture etc. this allows the Image Analysis Window of the PVRTrace GUI to be used.

SaveFrameBuffer

With this option ticked the PVRTrace Recording Libraries will save the contents of the frame buffer at the end of every frame.

Network

This option states whether the PVRTrace Analysis GUI will be recording the trace directly via the network.

Network Wait

With this option ticked the PVRTrace Recording Libraries will cause the application to wait in the first OpenGL call until the PVRTrace Analysis GUI signals to continue.

Network Buffer Size

'Network Buffer Size' sets the size of the buffer that is filled before sending data to the PVRTrace Analysis GUI when using Network recording. In general the default option is sufficient, but in some cases this option may need to be changed depending on network settings.

Debug Info

The 'Debug Info' dropdown selects whether debug data should be output to logcat. Three options exist:

- 'Off' – No data is written to the logcat.
- 'Basic' – Only the start of a recording, the end of a recording, and the current frame number are written to the logcat.
- 'Verbose' – Outputs all data to logcat.

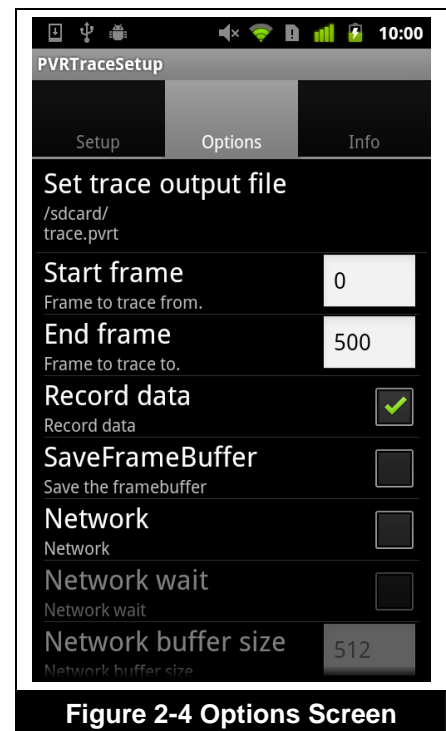


Figure 2-4 Options Screen

3. PVRTrace GUI

3.1. Overview

PVRTrace GUI is the graphical interface to the PVRTrace analysis tool. It opens PVRT files created with the PVRTrace recording libraries and displays them in a human readable and easy to understand format. It provides a wealth of information on a trace, broken down frame by frame; everything from the number of times a given call occurs in a trace to the exact values of a specific matrix.

PVRTrace currently supports the following APIs:

- OpenGL ES 1.1
- OpenGL ES 2.0
- OpenGL ES 3.0 (Partial: Function Call List only)

3.2. Installation

3.2.1. From Installer

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

```
<InstallDir>\PVRTrace\Analysis\<PLATFORM>\
```

3.3. Main Interface

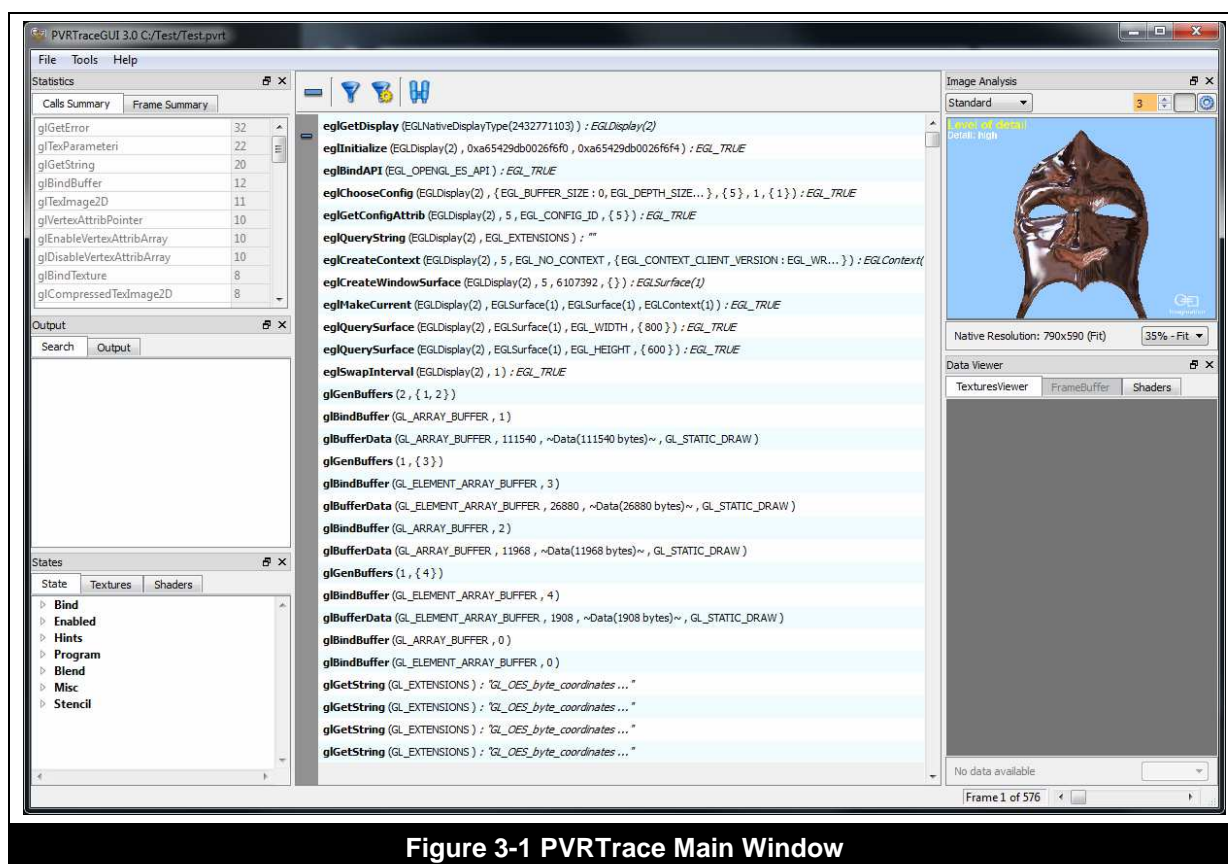


Figure 3-1 PVRTrace Main Window

3.3.1. Frame Summary

The Frame Summary tab gives an overall summary of the currently selected frame. The following information is displayed:

- total number of function calls
- total number of draw calls
- total number of triangles sent to OpenGL before any HSR/Culling
- total number of lines sent to OpenGL before any HSR/Culling
- total number of texture loads

Calls Summary Frame Summary	
Function calls	253
Draw calls	3
Triangles	4530
Lines	0
Texture Loads	19
Shaders Loads	4

Figure 3-2 Frame Summary

3.3.2. Calls Summary

The Call Summary tab serves as a list of all the function calls within the currently selected frame and the number of times each function has been called within that frame. Functions can be selected by left clicking; selected functions can be set as 'watched' by right clicking and clicking 'Add Watch (Selected)'. It is also possible to set a single function as 'watched' by right clicking a single function and clicking 'Add Watch'. 'Watched' functions always appear at the top of the window.

Calls Summary	
glGetError	32
glTexParameter	22
glGetString	20
glBindBuffer	12
glTexImage2D	11
glVertexAttribPointer	10
glEnableVertexAttribArray	10
glDisableVertexAttribArray	10
glBindTexture	8
glCompressedTexImage2D	8

Figure 3-3 Calls Summary

3.3.3. Image Analysis

By default, if the trace being analysed was recorded with data, the Image Analysis window displays the overall render output from the current frame, from first draw call to last. It is possible, using the 'Draw Call Selector', to change these values to draw only a subset of the overall draw calls, down to a single draw call if required. Draw calls can be scrolled through using the mouse wheel. Options exist to draw single draw calls, or ranges of draw calls; these options can be accessed via the Image Analysis Preferences Dialog, accessible from the blue cog icon in the top right of the Image Analysis window. Finally, three visualisation modes are available:

- Standard – The Image Analysis will display the output as a standard, coloured, shaded, etc. render.
- Wireframe – The Image Analysis will only display the wireframe of each object within the scene being rendered.
- Depth Complexity – A standard depth complexity render will be displayed, with each object being drawn partially transparent, brighter areas signalling greater depth complexity.

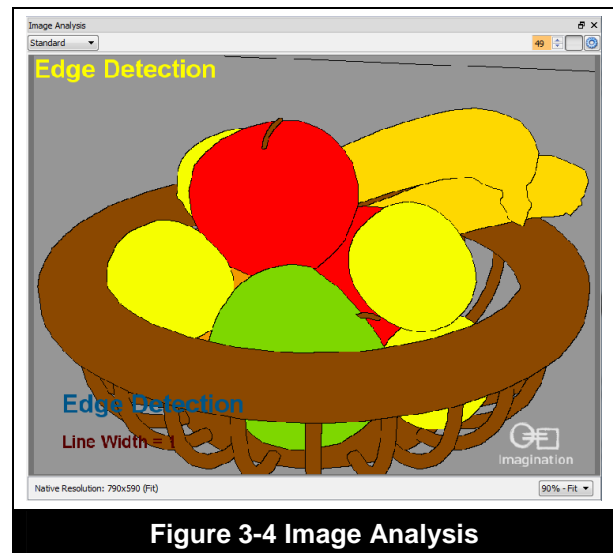


Figure 3-4 Image Analysis

It should be noted that Tiling and Hidden Surface Removal are not simulated in the Image Analysis, the output should not be considered indicative of how PowerVR hardware performs rendering; this information is given for debugging information only. Information on the process of Tile Based Deferred Rendering can be found in the 'Architecture Guide for Developers' on the PowerVR Insider SDK website.

3.3.4. Data Viewer

The Data Viewer consists of three areas, one for displaying textures, one for displaying the framebuffer, and one for displaying shader.

Texture Data Viewer

The texture data viewer displays the currently selected texture (selected through either the Render State Tab, or the Function Call List), its texture ID, format, and size.

Framebuffer Data Viewer

If the trace was recorded with the SaveFrameBuffer option set the contents of the final framebuffer, from the original device, will be displayed in this tab.

Shader Data Viewer

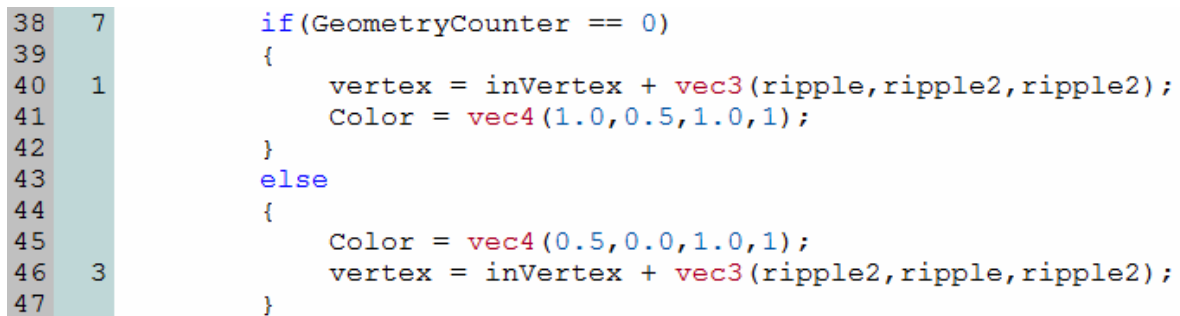
The Shader Data Viewer displays the currently selected shader (selected through either the Render State tab, or the Function Call List) inside of a build-in instance of PVRShaderEditor

PVRShaderEditor

PVRShaderEditor is the PowerVR Shader and Kernel Editor, available as both a standalone application and as a built-in component of PVRTrace.

PVRShaderEditor's main features are:

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



```
38 7      if(GeometryCounter == 0)
39      {
40 1      vertex = inVertex + vec3(ripple,ripple2,ripple2);
41      Color = vec4(1.0,0.5,1.0,1);
42      }
43  else
44  {
45      Color = vec4(0.5,0.0,1.0,1);
46 3      vertex = inVertex + vec3(ripple2,ripple,ripple2);
47      }
```

Figure 3-5 Per-line Cycle Counts

It should be noted that per-line cycle counts are only approximations, and only available for functional shaders.

3.3.5. Render State Tab

The render state tab contains information regarding the current frame's render state. This information changes based upon the position within the current frame selected in the Function Call List.

Bind

'Bind' contains a list of the currently bound textures; these textures can be selected and will be displayed in the Data Viewer.

Enabled

'Enabled' lists the options that are currently active, having been set through the use of 'glEnable'.

Hints

'Hints' lists the options currently set using 'glHint'.

Stencil

'Stencil' lists the current settings for the stencil buffer.

Misc

This section contains miscellaneous information that does not fit into the various other sections. This covers pixel byte alignment; polygon offset values, and scissoring information along with a variety of other useful information.

Program (OpenGL ES 2.0/3.0 Only)

'Program' details the settings for the currently running program, including the program's ID, the ID of the vertex and fragment shaders, and what the currently bound attributes and uniforms are. The vertex and fragment shader IDs can be selected and will appear in the Data Viewer.

Blend (OpenGL ES 2.0/3.0 Only)

'Blend' contains information pertaining to the current blend state, what blending equations and functions are used, the current clear depth and colour mask etc.

Clip Planes (OpenGL ES 1.1 Only)

'Clip Planes' details information of the currently set OpenGL clip planes as set with glClipPlane.

Lights (OpenGL ES 1.1 Only)

This menu lists all information pertaining to all the available lights within a scene, its specific colours, attenuations etc.

Tex Env (OpenGL ES 1.1 Only)

'Tex Env' lists all the information pertaining to each bound texture within the current frame.

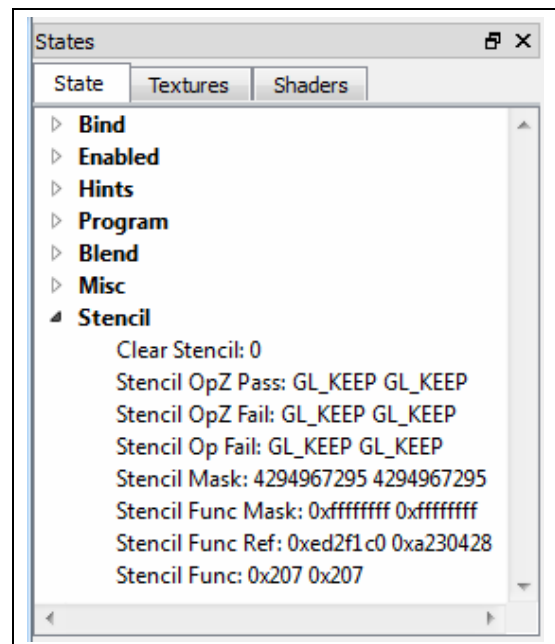


Figure 3-6 Render State Tab

3.3.6. Textures Tab

The Textures tab lists all the textures that are currently loaded. Each texture has a thumbnail and a unique ID, and contains its type, dimension, format (e.g. GL_RGBA), and storage type (GL_UNSIGNED_SHORT_4_4_4_4). Finally, it is possible to right click on a texture; this will open a menu with the options 'Show Loading Call' and 'Show Binding Call'.

'Show Loading Call' will move the Function Call List to the function call that loads that given texture.

'Show Binding Call' will move the Function Call List to the next instance of glBindTexture that binds that specific texture.

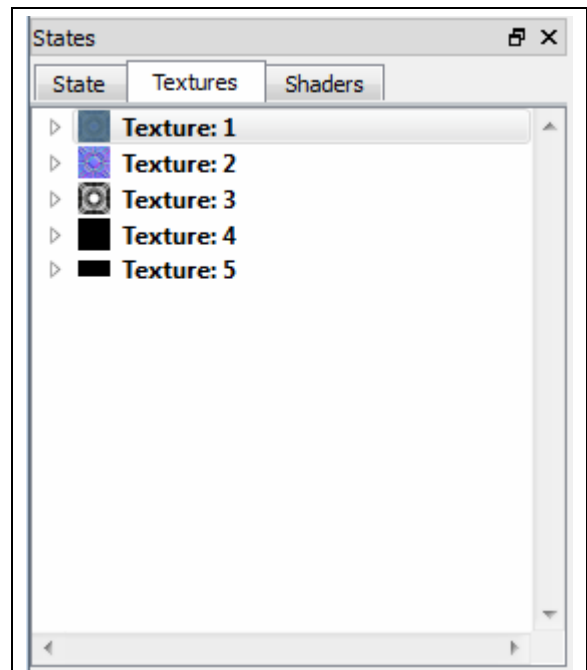


Figure 3-7 Textures Tab

3.3.7. Shaders Tab

The Shaders tab lists all the currently loaded shaders. Each shader is given a unique identifier, and if clicked on will open in the Data Viewer.

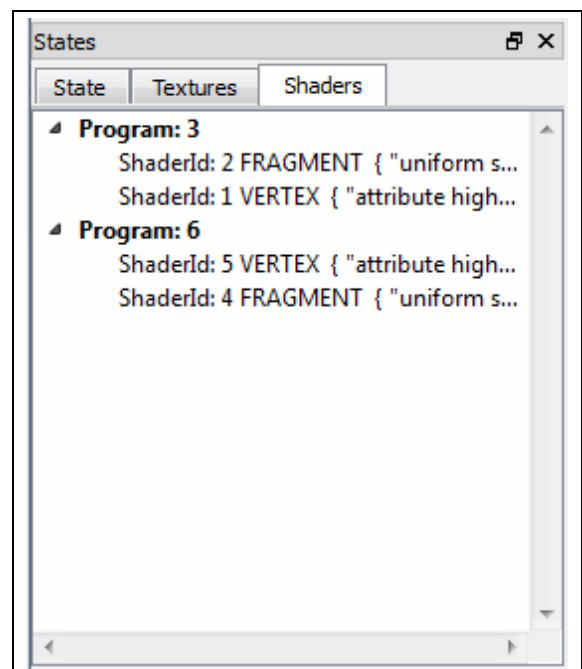


Figure 3-8 Shaders Tab

3.3.8. Function Call List

```

eglGetDisplay (EGLNativeDisplayType(302066616) ) : EGLDisplay(2)
eglInitialize (EGLDisplay(2) , 0x48ceb3b5003cf900 , 0x48ceb3b5003cf904 ) : EGL_TRUE
eglBindAPI (EGL_OPENGL_ES_API ) : EGL_TRUE
eglChooseConfig (EGLDisplay(2) , { EGL_BUFFER_SIZE : 0, EGL_DEPTH_SIZE... } , { 5 } , 1 , { 1 } ) : EGL_TRUE
eglGetConfigAttrib (EGLDisplay(2) , 5 , EGL_CONFIG_ID , { 5 } ) : EGL_TRUE
eglQueryString (EGLDisplay(2) , EGL_EXTENSIONS ) : ""
eglCreateContext (EGLDisplay(2) , 5 , EGL_NO_CONTEXT , { EGL_CONTEXT_CLIENT_VERSION : EGL_WR... } ) : EGLContext(1)
eglCreateWindowSurface (EGLDisplay(2) , 5 , 6238464 , { } ) : EGLSurface(1)
eglMakeCurrent (EGLDisplay(2) , EGLSurface(1) , EGLSurface(1) , EGLContext(1) ) : EGL_TRUE
eglQuerySurface (EGLDisplay(2) , EGLSurface(1) , EGL_WIDTH , { 800 } ) : EGL_TRUE
eglQuerySurface (EGLDisplay(2) , EGLSurface(1) , EGL_HEIGHT , { 600 } ) : EGL_TRUE
eglSwapInterval (EGLDisplay(2) , 1 ) : EGL_TRUE

```

Figure 3-9 Function Call List

The Function Call List shows all of the functions that have been recorded. Each function can be selected; this action adjusts the Render State Tab so that its information matches the render state at the position in the trace marked by the selected function. It is also possible to select shaders and textures from the parameters of a function call within this view; these textures and shaders will appear in the Data Viewer. In addition function parameters can be hovered over; doing this will produce a tooltip which will identify both the contents of the parameter and what the parameter represents (e.g. stride/size/pointer etc.). Any function call that has spawned a glError will be highlighted with red text after the function parameter list. Finally, function calls can be right clicked; this opens a menu with four available options: 'Add/Remove All Filters', 'Add/Remove All Highlights', 'Trace', and 'Function Quick Help'.

Add Filter

The 'Add Filter' option filters out the currently selected function call from the Function Call List for all frames. This filter can be removed using the Filter Functions dialog, or with the Remove All Filters option.

Remove All Filters

'Remove Filters' removes all filters from the Function Call List that are currently set.

Add Highlight

This option adds highlighting of the selected function to the Function Call List.

Remove All Highlights

This option removes all highlights.

Trace

'Trace' is a sub-menu consisting of two options:

- 'Disable Call'
- 'Clear Disabled Calls'

Clicking the first option disables the selected call, this call will no longer be processed when determining the output of the Image Analysis Window.

The second option will re-enable all previously disabled calls.

Function Quick Help

'Function Quick Help' opens the documentation for the selected function at <http://www.khronos.org>.

3.4. Toolbars

3.4.1. Data Viewer Toolbar

The Data Viewer Toolbar appears below the Data Viewer Image Analysis, on top of the Data Viewer when a texture is being displayed. It consists of six items:

Zoom Slider

This slider is used to set the zoom level of textures within the Data Viewer Toolbar.

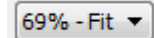


Figure 3-10
Zoom Slider

3.4.2. Function Call List Toolbar

The Function Call List toolbar is a small toolbar that appears above the Function Call List. It consists of four items:

Show Draw Calls

'Show Draw Calls' toggles whether to show all functions in the current frame or only the draw calls.



Figure 3-11
Show Draw Calls

Toggle Filters

'Toggle Filters' toggles on or off filters set with either the Filter Functions or 'Add Filter' right click option from within the Function Call List.



Figure 3-12
Toggle Filters

Show Filter Dialog

This option opens the Filter Functions dialog.



Figure 3-13
Show Filter Dialog

Find

'Find' opens the Find dialog.



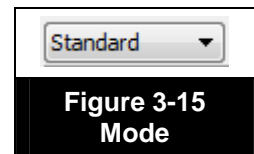
Figure 3-14 Find

3.4.3. Image Analysis Toolbar

Mode

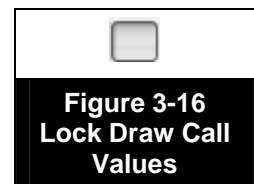
This drop-down selector allows the user to change what is displayed in the Image Analysis. Three modes are available:

- Standard – The Image Analysis will display the output as a standard, coloured, shaded, etc. render.
- Wireframe – The Image Analysis will only display the wireframe of each object within the scene being rendered.
- Depth Complexity – A representation of the depth complexity of the render will be displayed, with each object being drawn partially transparent, brighter areas signalling greater depth complexity.



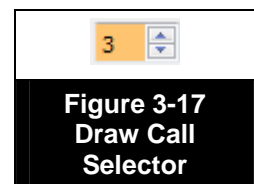
Lock Draw Call Values

The 'Lock Draw Call Values' button locks or unlocks the values set in the Draw Call Selector. With this option locked the output of a specific call can be followed from frame to frame. With this option unlocked, when another frame is displayed the Draw Call Selector will return to its default start and end frame.



Draw Call Selector

The 'Draw Call Selector' displays which draw call is to be drawn to. When 'Range' is selected a second draw call selector appears, this sets the starting draw call.



Depth Increment (Depth Complexity Mode Only)

This option, only displayed in 'Depth Complexity Mode', allows the depth complexity value to be adjusted. This has the effect of increasing or decreasing the brightness and contrast of the depth complexity render. This is particularly useful in very complex scenes

3.5. Dialogs

3.5.1. Find

The Find dialog searches the current frame for function calls containing the text entered in the 'Find' box.

'Exact Match' states that the search must match the case of the given string precisely.

'Whole Trace' informs Find that you wish the whole trace file to be searched rather than just the current frame.

Regular Expressions can be used in this search box. See Appendix A. Regular Expression Syntax.

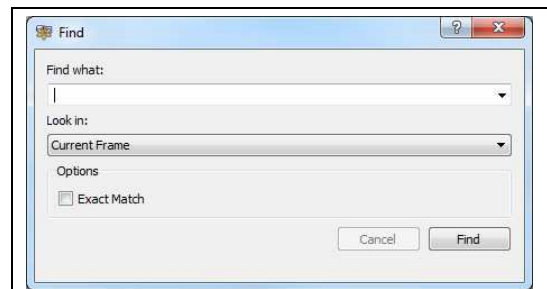


Figure 3-18 Find Dialog

3.5.2. Filter Functions

The Filter Functions dialog allows for specific OpenGL ES and EGL functions to be filtered in or out of the trace. A tick in the 'F' column of a given item indicates that the item (and any sub items) will be filtered out of the Function Call List; a tick in the 'H' column of a given item indicates that the item (and any sub items) will be highlighted in the Function Call List. A greyed out box shows that the item does not exist within the current trace (e.g. In an OpenGL ES 2.0 trace, the OpenGL ES 1.1 item, and all its sub items, will appear greyed out).

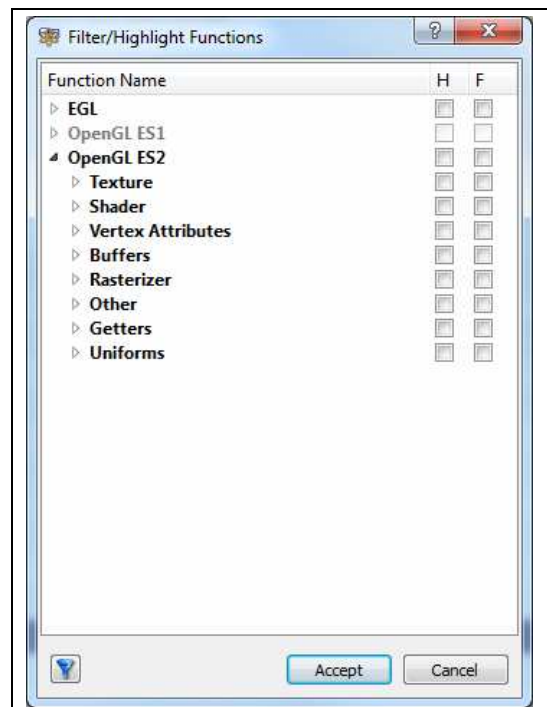


Figure 3-19 Filter Functions

3.5.3. Remote Controller

The Remote Controller dialog is used to connect to an application whose recording libraries are set up for remote recording.

In order to use this functionality an output file must be set by clicking the 'Output File Trace' button. Once this is done, the PVRTrace GUI can be connected to the remote application by clicking the 'Connect' button.

The remaining buttons allow for control over the remote recording; from left to right, these buttons are as follows:

- **Start/Stop Recording** – Used to signal the remote application to start or stop recording a trace.
- **Play/Pause** – This button signals the remote application to begin, pause, or continue, rendering. This is very important when the remote application has 'NetworkWait' set to '1' as the application will pause at the first OpenGL call until a 'Play' signal is received.
- **Open Trace** – Once recording has been stopped 'Open Trace' will open the recording in the PVRTrace GUI for analysis.

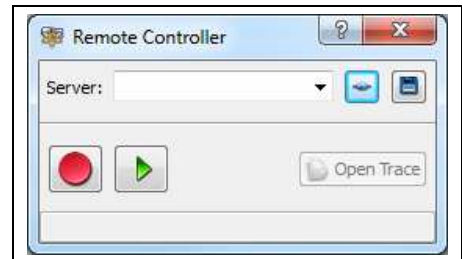


Figure 3-20 Remote Controller

3.5.4. Image Analysis Preferences

Selection

'Scrubber' sets the Image Analysis window to its default behaviour, drawing from the first draw call to the draw call indicated in the Image Analysis window.

'Single Primitive' sets the Image Analysis window to only draw the primitive associated with the draw call indicated in the Image Analysis window.

'Range' sets the Image Analysis window to draw all the primitives between two draw calls. It creates a second draw call selector in the Image Analysis window, this selector represents the start frame of the range, and the original selector represents the end of the range.

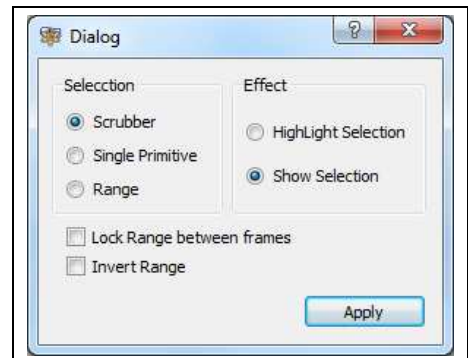


Figure 3-21 Image Analysis Preferences

Effect

'Highlight Selection' highlights the selected functions in the Image Analysis Window by false colouring them..

'Show Selection' displays only the selection function range in the Image Analysis Window.

Lock Range

'Lock Range between frames' locks the range of draw calls displayed in the Image Analysis window from frame to frame.

Invert Range

'Invert Range' inverts the range of draw calls displayed in the Image Analysis window.

3.6. Menus

3.6.1. File

Open

'Open...' opens an 'open file' dialog; from here a PVRT file can be selected and opened.

Open Recent

'Open Recent' contains a list of the most recently opened files; these list items can be selected and opened.

Save

'Save -> Save Current Frame (.txt)' saves the contents of the Function Call List for the current frame, to a TXT file.

'Save -> Save All Frames (.txt)' saves the contents of the Function Call List for all frames, broken down frame by frame, to a TXT file.

'Save -> Save Frame Statistics (.csv)' saves the contents of Frame Summary for all frames, as a .csv file.

'Save -> Save Call Statistics (.csv)' saves the contents of the Call Summary tab for all frames, as a .csv file.

Quit

'Quit' quits the application.

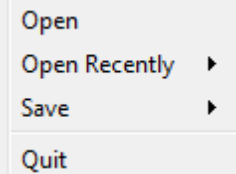


Figure 3-22 File Menu

3.6.2. Tools

Remote Controller

'Remote Controller' launches the Remote Controller dialog.

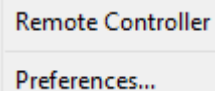


Figure 3-23 Tools Menu

3.6.3. Help

PVRTrace Help

'Help...' opens this document.

Feedback

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

About

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

Check for Updates

As of SDK release 3.0 PVRTrace can auto-update. 'Check for Updates' is used to force an update.

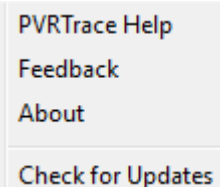


Figure 3-24 Help Menu

4. Related Materials

Software

- PVRTune

Documentation

- PVRTune User Manual
- Architecture Guide for Developers

5. 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 1 st to 9 th 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.