PVRTrace

Quick Start Guide for Unrooted Android Devices

Filename	:	PVRTrace.Quick Start Guide for Android Unrooted
Version	:	PowerVR SDK REL_18.1@5080009a External Issue
Issue Date	:	31 May 2018
Author	:	Imagination Technologies Limited

Contents

List of Figures

1. Introduction

1.1. Document Overview

This document provides instructions for the installation, configuration and initial running of PVRTrace to debug and analyse API calls on an Android device with PowerVR graphics technology.

1.2. Software Overview

PVRTrace is a utility for recording and analysing scenes. 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 stage. It consists of two main components:

- **Recording Libraries**: These are shim libraries that are installed on the target device and capture all calls to the target device's native graphics libraries. These calls are captured and written into a PVRT file for reading back by the PVRTrace GUI.
- **PVRTrace GUI**: This is the analysis interface of PVRTrace. The GUI presents the contents of pre-recorded PVRT files in a 'human-readable' format. For more information on the PVRTrace GUI see the "PVRTrace User Manual").

Note: Applications that use both OpenGL ES 1.1 and OpenGL ES 2.0/3.0/3.1/3.2 can't be recorded with the unrooted method.

1.3. Prerequisites

This document assumes:

- The following packages are installed on the host machine:
  - PowerVR SDK and Utilities.
  - The Android SDK and Android device drivers.
  - The Android NDK.
- The following folders are in the `PATH` on the host machine:
  - The 'tool' folder of the Android SDK.
  - The Android NDK folder.
- The target device is not rooted.
- The Recording Libraries and the PVRHub application are not installed on the target device.

For full software installation instructions, see the PVRTrace User Manual.

2. Installation

The Recording Libraries should be statically linked and packaged with the user's application. The recording library functions as a 'shim' library in conjunction with the Android GL Library, as depicted in Figure 1.

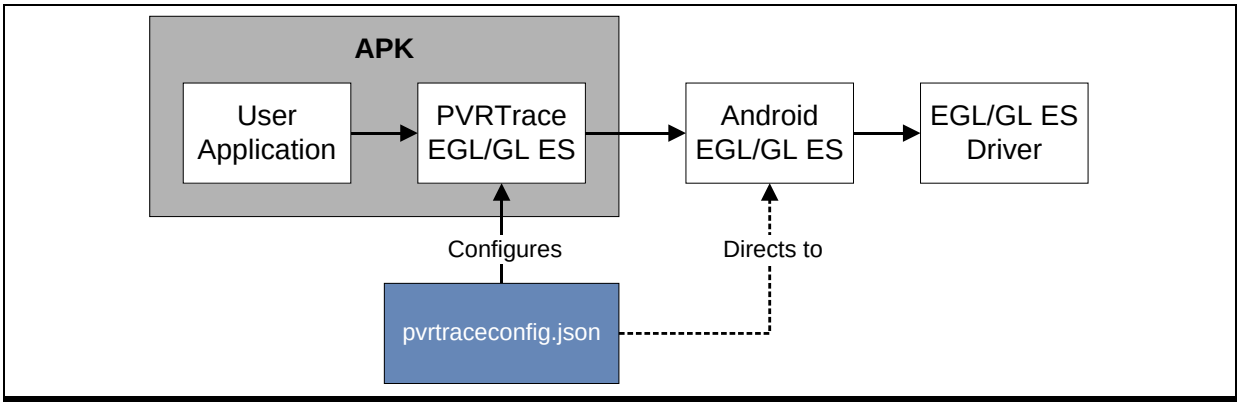

Figure 1. PVRTrace libraries

Note: The PVRTrace and Android GL Libraries are in reverse order compared to a rooted Android device.

2.1. Packaging the Recording Libraries with an Application

To package the Recording Libraries with the user's application, the following steps have to be followed:

1. Tell the Native Activity to load the Recording Libraries. For example:

```
package com.powervr.OGLES2Water

import android.app.NativeActivity

public class OGLES2Water extends NativeActivity
{
    @Override
    protected void onCreate (Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
    }
    static {
        try {
            System.load("/data/data/com.powervr.OGLES2Water/lib/libPVRTrace.so");
            System.load("/data/data/com.powervr.OGLES2Water/lib/libEGL_PVRTRACE.so");
            System.load("/data/data/com.powervr.OGLES2Water/lib/libGLESv2_PVRTRACE.so");
        }
        catch( UnsatisfiedLinkError e ) {
            System.err.println("Native code library failed to load.\n" + e);
        }
    }
}
```

2. Edit the Android.mk to point to the correct libraries:

```
LOCAL_LDLIBS :=  \
                -llog \
                -landroid \
                -L/data/data/com.powervr.OGLES2Water/lib/\
                -lEGL_PVRTRACE \
                -lGLESv2_PVRTRACE
```

3. Add the following line to the `AndroidManifest.xml` file. This gives the application permission to save the PVRTrace file to the SD card:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

4. Perform an NDK Build:

- Update the Android project by running the following command:

```
android update project -p . -t <target>
```

Where `<target>` corresponds to the Android API target ID number.

- Use `ndk-build` to compile the application code into a library. To compile for all supported platform types:

```
ndk-build
```

Or to compile for a specific platform you can compile using the `APP_ABI` option:

```
ndk-build APP_ABI=<platform>
```

Replacing `<platform>` with an entry from the following list of supported ABIs:

- `armeabi`
- `armeabi-v7a`
- `arm64-v8a`
- `x86`
- `x86_64`
- `mips`
- `mips64`

5. Copy the Recording Libraries to:

```
"<build_output>/libs/<platform>/"
```

Where `<build_output>` is the build output directory and `<platform>` is an ABI supported by the Android build system, for example: `armeabi`, `armeabi-v7a`, `x86` or `mips`.

6. Build the Java components and package the native libraries to create the `.apk` file for installing on a device. The quickest way of achieving this is to enter the command:

```
ant debug
```

2.2. Tracing an Application

To trace an application, perform the following steps:

1. Install an application on the target device. Using the `adb install` command in the Android SDK, enter the following command:

```
adb install <local>
```

Where `<local>` is the location of the APK application file on the host machine.

2. Create `pvrtraceconfig.json` in the root of `/sdcard`.
   PVRTrace looks for the config file at the following locations in the following order:
   i) /data/data/com.powervr.pvrhub/pvrtraceconfig.json
   ii) /data/data/com.powervr.PVRHub/pvrtraceconfig.json
   iii) /data/data/com.powervr.PVRTraceApp/pvrtraceconfig.json
   iv) /pvrtraceconfig.json
   v) /sdcard/pvrtraceconfig.json
   Make sure that your config file is not overridden by a higher priority config file. Also make sure your config file has sufficient permissions for example –rwxrwxrwx. Finally, make sure PVRTrace is enabled for your application by setting "Enabled": true in the config file at your application's entry (or globally for *).
   For example, for a Google Nexus S, recording frames 0 to 200 to a local file, `pvrtraceconfig.json` should look similar to the following:

```
{
  "*": {
    "Tracing": {
      "OutputFilename": "/sdcard/%pname.pvrtrace",
      "RecordData": true,
      "StartFrame": 0,
      "EndFrame": 200,
      "ExitOnLastFrame": true
    }
  }
}
```

Note: The flag `%pname` tells the libraries to use the process name.

3. Run the application. The application exits when it reaches the end frame (frame 200).

4. Once the application has exited copy the output file set in `pvrtraceconfig.json` into a location on the host machine accessible from the PVRTrace GUI. Using the `adb pull` command in the Android SDK, enter the following command:

```
adb pull <remote> [<local>]
```

Where `<remote>` is the location of the trace file on the target device and `<local>` is the name and location of the trace file to save on the host machine. If `<local>` is not provided, the trace file is saved in the current directory using the existing name. For example:

```
adb pull /sdcard/com.powervr.OGLES2Water.pvrt
```

An example output from this command is:

```
adb pull /sdcard/com.powervr.OGLES2Water.pvrt
3610 KB/s (3690237 bytes in 0.998s)
```

5. Run the PVRTrace GUI and open the output file to begin analysis. For more information about PVRTrace GUI, consult the “PVRTrace User Manual”.

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