OpenGL ES 2.x SDK

User Guide

Filename	:	OpenGL ES 2.x SDK.User Guide.mht
Version	:	1.4f (PowerVR SDK 2.03.23.1162)
Issue Date	:	15 Aug 2008
Author	:	PowerVR

Contents

1. OpenGL ES SDK Content

1.1. Introduction

The PowerVR OpenGL ES 2 SDK provides a set of documentation, source code and utilities to help developers to create applications using the OpenGL ES 2 graphics library on PowerVR platforms. This document describes the contents of the SDK and gives guidelines for installing it on different platforms.

1.2. Documents

These documents cover some technology overviews, performance recommendations and specifications.

Documents are located in `\SDKPackage` or in `\SDKPackage\Documents`.

Name	OpenGL ES SDK.User Guide
Description	Description of OpenGL ES SDK contents and Installation.

Name	Migration from software rendering to 3D accelerated graphics using OpenGL ES.Khronos Kolumn
Description	An article posted in the Khronos Kolumn: http://www.gamedev.net/columns/kk/kk1/ This article will help developers who are now starting to code for hardware accelerated devices.

Name	PowerVR Effect File (PFX)
Description	This document describes the PFX file format. This format is used to encapsulate vertex shaders, fragment shaders and textures for an easy definition of a material. It is used by PVRShaman. The API to handle this format in OpenGL ES 2.0 is part of PVRTools.

Name	PowerVR SGX.Application Development Recommendations
Description	This document gives performance recommendations for PowerVR SGX hardware and explains how to analyse an application to detect bottlenecks.

1.3. Demos

The OpenGL ES SDK contains a variety of technology demos whose aim is to demonstrate a particular feature of the hardware or software API. Each demo is provided with an executable and associated source code. For a description of the command-line options supported by the demos see Shell section below.

All demos are located in the `\SDKPackage\Demos` folder.

	Shaders
	Shaders library. This demo demonstrates several procedural texturing methods like 'wood', 'phong shading', 'environment mapping', etc. using fragment shaders. These shaders are applied to different geometry objects. Use the cursor key to select. Please keep the folder called 'Shaders' (which contains all the shaders files) in the same location as the application. The application will quit if it cannot find these files. This demo supports command-line input where '-s=' specifies the shader (from 0 to 5) and '-m=' the mesh (from 0 to 5): `e.g. OGLES2Shaders -s=2 –m=0`

	SkyBox2
	This is a new version of the SkyBox demo from our OpenGL ES1.1 SDK but, in this case, using different vertex and fragment shaders to show the capabilities of OpenGLES2.0 API.

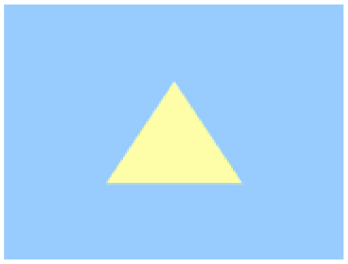
1.4. Training course

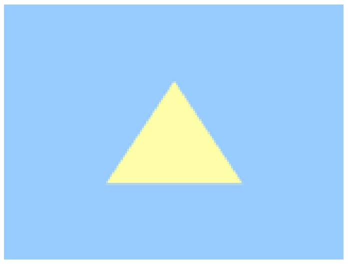
Training Course is composed of several simple applications to show specific features in a simplified form. The code has been thoroughly commented to help developers to understand the API and get started.

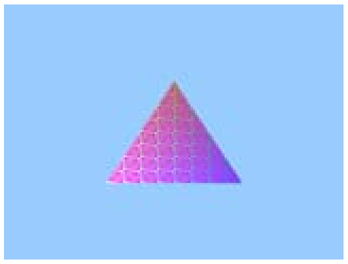
All Training Course demos are located in the `\SDKPackage\TrainingCourse` folder.

Note: The demos in the TrainingCourse do not handle screen rotation to keep the code as simple as possible. On devices with a portrait display the images below might be shown stretched.

	Initialization
	Shows how to initialise OpenGL. It does a simple background clear.

	HelloTriangle
	Shows how to draw simple untextured geometry.

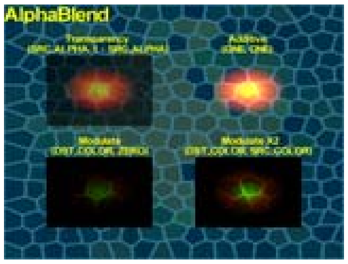
	IntroducingPVRShell
	Shows how to use PVRShell as the OS and API initialisation framework. Uses the same geometry used in the previous example.

	Texturing
	Shows how to load and use textures.

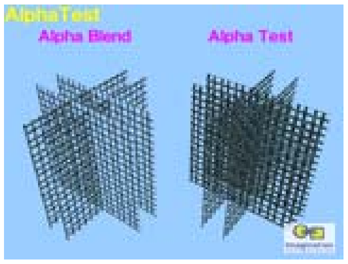
	BasicTnL
	Shows how to transform and light simple geometry. Uses the same geometry used in the previous example.

	IntroducingPVRTools
	Shows some of the features available in the PVR tools supplied with the SDK. In this case shows how to load textures and display text on the screen.

	IntroducingPOD
	Shows how to load and playback a PowerVR POD file exported from 3ds Max.

	AlphaBlend
 AlphaBlend Transparency (SRC.ALPHA, 1 - SRC.ALPHA) Additive (ONE, ONE) Modulate (DST.COLOR, ZERO) Modulate X2 (DST.COLOR, SRC.COLOR)	Shows 4 different types of blending mode.

	RenderToTexture
	How to perform render to texture using the back buffer.

	AlphaTest
	Demonstrates how to use the Alpha Test.

	StencilBuffer
	Demonstrates how to use the Stencil Buffer.

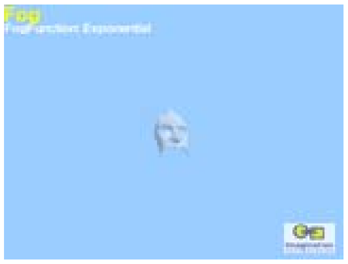
	IntroducingPFX
	Shows how to use our tools to load a PFX file.

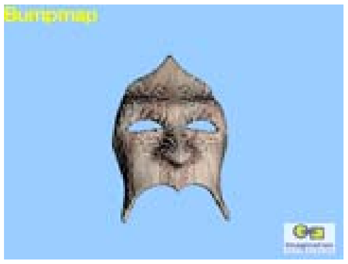
	FastTnL
	Shows how to use our fast transformation and lighting vertex shader. This technique is appropriate when a high performance shader is required.

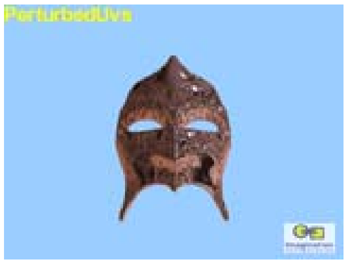
	LightMap
	Shows how to use Light Maps.

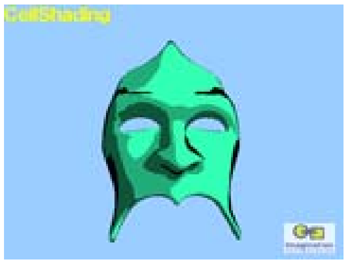
	Reflections
	How to use reflections, 2D Mapping and Cube Mapping.

	ComplexLighting
	Shows how to implement directional, point, and spot Lighting.

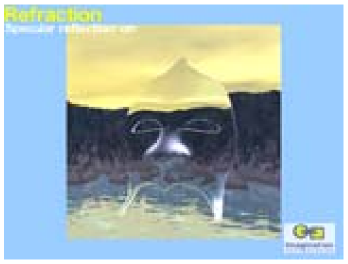
	Fog
	Shows how to implement a fog effect using shaders.

	Bumpmap
	Shows how to use a normal map for bump mapping.

	PerturbedUVs
	Shows how to use a normal map to perturb uvs.

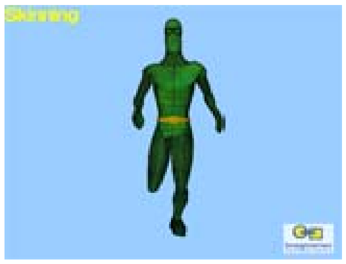
	CellShading
	How to implement cell shading.

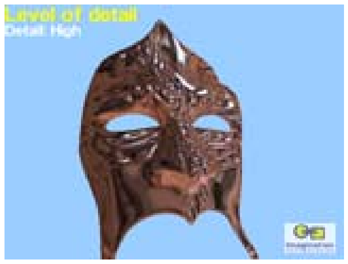
	FresnelReflections
FresnelReflections	Shows how to implement fresnel reflections. It looks similar to a car body surface with wax.

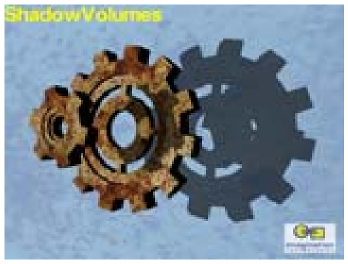
	Refraction
	How to create glass like materials and Phong's specular reflection.

	AnisotropicLighting
AnisotropicLighting	This example demonstrates two techniques for generating an anisotropic lighting effect on the mask. One calculates the shading for each pixel on the fly; the other is optimised by using a texture to lookup the shading values. Anisotropic lighting simulates the effect of surfaces with a grain, or grooves. This includes materials such as brushed metal, velvet and records.

	Iridescence
Iridescence Minimum Thickness 100 nm Maximum Variation 100 nm	How to implement an iridescence effect by mimicking the behaviour of light reflecting off a surface covered by a translucent film of variable thickness.

	Skinning
	Demonstrates how to perform skinning.

	LevelOfDetail
	Demonstrates the use of a low detail method for displaying an object where you won't notice all the detail i.e. far away.

	ShadowVolume
	Shows how to use our tools to create and use shadow volumes.

1.5. PVRShell

The PowerVR shell is used in all Demos and Training Courses to provide a common framework for developing OpenGL ES applications on any PowerVR platform. The shell takes a set of command-line arguments which allow things like the position and size of the demo to be controlled. The table below shows these options.

PVRShell provides also with a limited keyboard input which may vary depending on your actual platform. If you platform has full keyboard support, the Q key will quit the application and the cursor keys will allow looping through the application options.

Table 1 - Shell Command-Line Options

Option	Description
-width=N	Sets the viewport width to N.
-height=N	Sets the viewport height to N.
-posx=N	Sets the x coordinate of the viewport.
-posy=N	Sets the y coordinate of the viewport
-FSAAMode=N or –aa=N	Sets full screen anti-aliasing. N can be: 0=no AA , 1=2x AA , 2=4x AA
-fullscreen=[1,0]	Runs in fullscreen mode(1) or windowed (0)

Option	Description
-powersaving=[1,0]	Where available enable/disable power saving.
-quitaftertime=N or –qat=N	Quits after N seconds
-quitafterframe=N or –qaf=N	Quits after N frames
-vsync=N	Where available modify the apps vsync parameters
-version	Output the SDK version to the debug output.
-info	Output setup information to the debug output.

1.6. PVRTools

The tools library consists of various modules whose task is to help developers achieve common tasks associated with 3D graphics rendering, e.g. model loading, extensions handling, matrix functions, 3D font, etc. All SDK demos make use of the OpenGL ES 2 Tools library.

The Tools library is located in the `\SDKPackage\Tools` folder.

1.7. Utilities

These are utility programs or libraries useful for 3D application development.

All utilities are located in the `\SDKPackage\Utilities` folder.

Name	PVRTexTool PVRTexLib PVRTC
Description	Tools to convert bitmap files (e.g. BMP, TGA, etc.) to any texture type supported by PowerVR hardware.

Name	PVRGeoPOD
Description	Plug-in for 3DStudio MAX and Maya to export optimised 3D data in POD format.

Name	Collada2POD
Description	Command-line utility to convert Khronos intermediate format to PowerVR POD format.

Name	PVRShaman
Description	PowerVR Shader Composer for OpenGL ES 2

Name	PVRUniSCo
Description	SGX shader compiler. *Note: This tool might not be available in all releases.*

Name	PVRUniSCo Editor
Description	Shader editor and GUI frontend for PVRUniSCo.

Name	PVRTune
Description	Remote hardware performance profiler.

Name	PVRScope
Description	Platform specific library to access the hardware performance counters.

Name	Filewrap
Description	Utility to wrap a text or binary file into a C++ source file that can be compiled and linked to an application. See description of memory file system below.

2. Memory File System and FileWrap

All file read access in the PVRTools library uses the CPVRTResourceFile class. This not only simplifies code, it also allows the use of a “memory file system”. Any file can be statically linked into the application and looked up by name at run time. This is useful for platforms without a file system and keeps all required data in one place, the application executable.

When looking for a file, CPVRTResourceFile will first look in the read path the application set by calling CPVRTResourceFile::SetReadPath. This is usually a platform dependent path which can be obtained from PVRShell. If the file is not there, CPVRTResourceFile will look for the file in the memory file system. This way “internal” files linked in the executable can be overridden by external ones. When using the memory file system filenames passed to PVRTools functions should always be given without a path.

2.1. Using The Memory File System In Your Own Projects

In order to use the memory file system in your own projects, you need to link against the PVRTools library. Furthermore, you need to turn the files you want to add into C++ sources using the Filewrap utility. Typically this would be done with a command line like this:

```
filewrap –o Memfilesystem.cpp file1 file2 file3 ...
```

Note that the name of the .cpp file is not important; the files will be registered in the memory file system under their original name (in this case file1, file2, file3). You can also generate multiple .cpp files and compile them into the same application. However, the memory file system is just a list of files without any directory structure, so make sure not to use duplicate file names within an application.

After you have generated the .cpp files, all you need to do is add them to your project like you would add other C++ source files. Now you are ready to use CPVRTResourceFile to read files from the memory file system. Of course this wrapping can also be automated using makefiles or custom build steps in Visual Studio projects.

To automate Filewrap using custom build steps in Visual Studio, first add the file you want in the memory file system to your project (“Add existing item…”). Then select this file in the Solution Explorer and open the property page for it. Choose “All Configurations”, and then select “Custom Build Step”.

Set “Command Line” to:

```
Filewrap.exe -o "$(InputDir)Outputfile.cpp" "$(InputPath)"
```

Set “Outputs” to:

```
$(InputDir)Outputfile.cpp
```

Make sure Filewrap.exe is on your PATH environment variable or use an absolute path to Filewrap.exe. Replace Outputfile.cpp with a file name of your choice.

Following this, close the property page, right-click on the file and select “Compile”. This should produce the .cpp file which you should also add to your project.

3. OpenGL ES SDK Installation

See below the SDK installation and build notes for all platforms supported by PowerVR cores. Please, refer to the platform you are using.

3.1. PCEmulation

Note: PCEmulation library is not an accurate representation of PowerVR hardware. This tool is a wrapper around desktop OpenGL so its capabilities will depend on the 3D acceleration present in your system.

3.1.1. Windows PC Emulation

1. Beginning with a PC equipped with the Windows XP operating system install Microsoft Visual Studio .NET 2003 or .NET 2005. Note: If you use Microsoft Visual C++ Express Edition be sure to follow all of the instructions in the Microsoft Platform SDK.
2. Unpack the contents of this package to a local folder in the system. The project and solution files provided (.vcproj and .sln, for Microsoft Visual Studio 2003) do not require the SDK to be installed in a pre-defined location and are configured to use relative-paths.
3. The OpenGL ES 2 emulation "drivers" are called libEGL.lib and libGLESv2x.lib and are located by default in the following directory:
   SDKPackage\Builds\OGLES2\WindowsPC\Lib
   Those files must be copied to a DLL-accessible directory prior to running the SDK applications (e.g. standard Windows directory or local directory).
4. Tools and Demos can be launched directly from their project files.

3.1.2. Linux PC Emulation

1. Unpack the contents of this package to a local folder on a PC equipped with a Linux operating system using "tar –zxvf".
2. The OpenGL ES emulation "drivers" are called libEGL.so and libGLESv2.so and are located by default in the following directory: SDKPackage\Builds\OGLES2\LinuxPC\Lib
   Those files must be accessible by the OS. To do so put them into your library path by using: "export LD_LIBRARY_PATH=(folder containing the .so files)"
3. Demos and benchmarks can be launched directly from the "Binaries" folder.
4. To build a component, first set the platform to LinuxPC using "export PLATFORM=LinuxPC"
5. Then go into the component's Build/LinuxGeneric folder and type 'make'.
6. The resulting executable will go into a subfolder of Build/LinuxPC.

3.2. Linux

1. OpenGL ES 2.0 libraries might not be distributed with the PowerVR Linux SDKs. Ask your platform provider for these if you do not have them. Copy all the libraries (*.so) into: /SDKPackage/Builds/OGLES2/<platform>/Lib
2. To build individual components go to directory
   Demos/<DemoName>/OGLES2/Build/LinuxGeneric or
   TrainingCourse/<TCName>/OGLES2/Build/LinuxGeneric and run command "make PLATFORM=<platform>".
3. The executables for the demos and training courses will go to:
   Demos/<DemoName>/OGLES2/Build/<platform>/Release
   TraininCourse/<DemoName>/OGLES2/Build/<platform>/Release respectively.
4. Be sure that the PowerVR drivers are in your discimage and that the LD_LIBRARY_PATH environment variable points to them. (please refer to DDK / Driver installation instructions).
5. If the standard c++ libraries are not present on your discimage, copy libc++* from the toolchain into /usr/lib on the discimage, libdl and libgcc may also be required.
6. Ensure the drivers are running (eg. type /etc/init.d/rc/pvr start, then run an X session if required).

7. Copy the executables to your ‘discimage’ to run them from your platform system.

3.3. uITRON

To build for uItron you will need to setup the following environment variables, alternatively they can be passed to the makefile on the command line.

WORKROOT	Refers to the folder that contains the Eurasia folder of the DDK.
PLATFORM	Refers to the name of the folder within *\eurasia\eurasiacon\build\uitron*.
SHC_DIR	The tools you wish to use, e.g. \Tools\Renesas\sh\8_05.

The following variables only need to be defined when using some versions of the DDK

SGXBUILD	SGXBUILD’s value is taken from the first 6 characters of the suffix attached to the folder name *uiton_shmobile_release* in \PLATFORM\tools. It will be of the form SGX***xxx***
SGX_CORE_REV	SGX_CORE_REV takes its value from the remainder of the suffix.

1. To build individual components go to the /Build/uitronSHMobile sub-directory of the component to build and type:

```
nmake Makefile.mak
```

3.1. Windows Mobile 7

1. Firstly, open up a command prompt and setup your Windows Mobile 7 build environment.
2. Set the environment variable SDKROOT to the root of the installed SDKPackage

```
e.g. set SDKROOT=c:\SDKPackage
```

3. To build all SDK apps at the root of the SDKPackage type

```
build –c
```

4. To build an individual app navigate to the application’s folder and type

```
build –c
```

The makefiles for the majority of apps are kept in [app]\OGLES2\Build\PlatformBuilder and the executables are built in [app]\OGLES2\Build\PlatformBuilder\oak\target\ARMV4I\retail.

4. Support Contact

For further support contact:

devtech@imgtec.com

Imagination Technologies Ltd.
FAO: Developer Technology
Home Park Estate
Kings Langley
HERTS WD4 8LZ
United Kingdom

Tel: +44 (0) 1923 260511
Fax: +44 (0) 1923 277463

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

www.imgtec.com
www.powervr.com
www.powervrinsider.com