POWERVR OpenGL ES SDK

User Guide

Filename	: OpenGL ES SDK.User Guide.1.20f.External.doc
Version	: 1.20f External Issue (Package: POWERVR SDK 2.05.25.0804)
Issue Date	: 13 Aug 2009
Author	: POWERVR

Contents

1. OpenGL ES SDK Contents

1.1. Introduction

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

Note: The support for OpenGL ES 1.0 has been discontinued from SDK version 2.3 onwards. If you're developing for the Dell Axim, please download the 2.2 version which is still available from our website.

1.2. Documents

The following documents are provided with the POWERVR SDK providing technology overviews, performance recommendations and specifications.

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

POWERVR OpenGL ES SDK.User Guide
Description of the POWERVR OpenGL ES SDK contents and installation instructions.

POWERVR Technology Overview
This document gives an overview of POWERVR technology from a hardware perspective. It is useful for developers that wish to learn more about the hardware they're programming for and can help to understand potential optimizations.

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

POWERVR MBX.3D Application Development Recommendations
Some recommendations and tricks to get best performance on POWERVR MBX devices.

PVRTextureCompression
Whitepaper describing PVRTC texture compression in detail. This whitepaper was issued at the presentation that was given at the Graphics Hardware 2003 conference.

Reference Driver.User Guide
Some notes about OpenGL ES 1.x drivers: Extension description, how to detect POWERVR hardware and some coding recommendations.

Extension specifications *.txt
Specifications for supported OpenGL ES extensions.

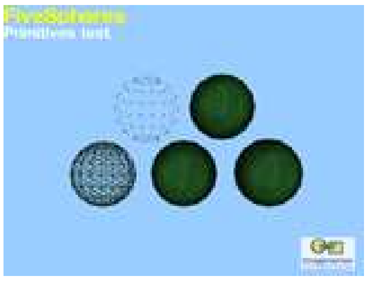
1.3. Demos

The POWERVR OpenGL ES SDK contains a variety of technology demos which aim to demonstrate a particular feature of the hardware or software API. Each demo is provided with an executable and associated source code. A variety of command-line options are available while running the demos, see 1.5 PVRShell and command-line options below.

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

Note: Some of the Demos or Training Courses listed below might not be present on your specific SDKs. This will depend on the capabilities and features of your target platform.

OGLESEvilSkull	
EvilSkull Morphing.	Demonstrates geometry morphing. Morphing creates the final object using weighted blending between a number of different configurations of the same object. Each configuration controls a certain aspect of the object. In the case of EvilSkull the different configurations represent the position of the eyebrows and cheekbones. By setting different weights it is possible to generate a huge variety of facial expressions. The background shows multitexturing with two independently animated layers.

OGLESFiveSpheres	
	Shows five different primitive types: GL_POINTS GL_TRIANGLES GL_LINE_STRIP GL_TRIANGLE_FAN GL_TRIANGLE_STRIP

OGLESLighting	
OGLESLighting 8 point lights.	Shows a sphere lit by eight different coloured spotlights using ambient, diffuse and specular components.

OGLESMouse	
Mouse VGP Toon Shading	Demonstrates cel-shading (cartoon) and animation. Cel-Shading is the "art" of rendering your objects to look like cartoons. This demo shows a dancing mouse with a hard black outline and banded shading to achieve the desired cartoon-look. Both effects use an optimised vertex shader program or, in case the hardware platform does not have a VGP unit, use optimised software routines.

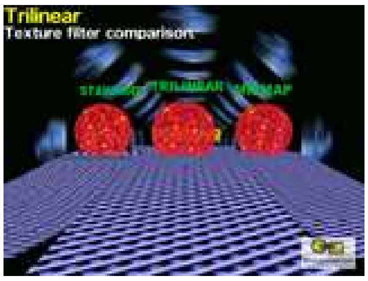
OGLESOptimizeMesh	
Optimize Mesh 104.0fps - No 7 Indexed Triangle List: Unoptimized	Demonstrates the performance gains possible from optimizing geometry. Shows a textured model rendered in two different ways: one using a non-optimized triangle list and one using an optimized triangle list. The demo switches mode after a few seconds or after the user presses the action key. The data for both models was generated using the PVRGeoPOD plug-in utility from the SDK. The optimized triangle list case was generated using the PVRTTriStrip triangle sorting option from PVRGeoPOD. MBX devices with up-to-date drivers should show higher performance in the optimized triangle list case. Note that the performance difference will not be visible on PC Emulation builds!

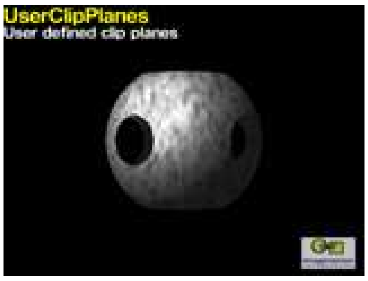
OGLESParticles	
	Shows a physics-based particle effect with alpha transparency, displaying a total of 900 particles (600 real + 300 reflected).

OGLESPolybump	
PolyBump DOT3 per-pixel lighting	Demonstrates the Polybump technology developed by Crytek to enhance 3D-rendering quality without increasing the overhead in real-time rendering. It allows users to create and render an extremely low poly model using per pixel DOT3 lighting. The DOT3 normal map is automatically generated from an ultra-high poly model. As a result when displaying the low polygon model with the DOT3 map generated from the high polygon object there is virtually no visible difference between the two. Rendering time is greatly decreased and bandwidth requirements reduced. In this demo a highly detailed human head is shown even though there are only 276 polygons in the model. This demo also demonstrates the use of the GL_IMG_texture_format_BGRA8888 extension (where available) to load a BGRA 32bit normal map to get the best quality per pixel DOT3 lighting.

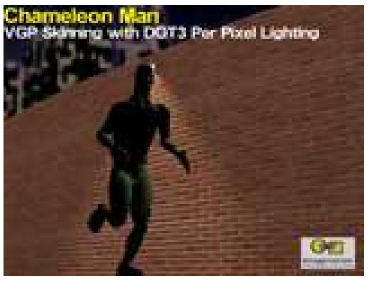
OGLESShadowTechniques	
	Demonstrates different three methods of rendering shadows: The dynamic blob method draws a transparent blob under the character and the blob moves based on the centre of the object. The projected geometry method draws the object "projected" onto the floor plane, using this method only pure black shadows are possible due to multiple polygons being projected into the same location (would give intensity differences in the shadow if transparency would be used). The final method is projected texture where the projected shadow texture is updated dynamically each frame (generated by rendering the scene from the point of view of the light source). This method, unlike the previous methods, also works correctly for shadows cast on non-planar objects.

OGLESSkybox	
	Shows how to render an environment skybox. The skybox is compressed using PVRTC4 using the PVRTextureTool skybox compression feature.

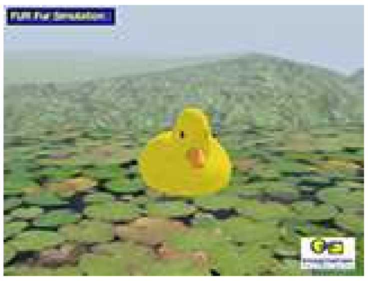
OGLESTrilinear	
 Trilinear Texture filter comparison STANDARD TRILINEAR MIPMAP	Shows the difference between three texture filtering modes, including trilinear texturing: GL_LINEAR (bilinear with no mip-mapping) GL_LINEAR_MIPMAP_NEAREST (bilinear with mip-mapping) GL_LINEAR_MIPMAP_LINEAR (trilinear)

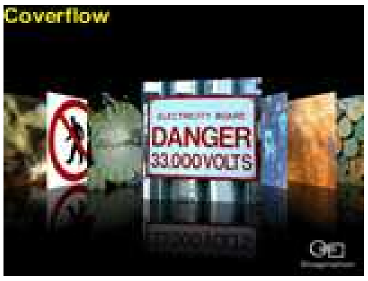
OGLESUserClipPlanes	
	Shows how to use the user-defined clip planes extension. A rotating sphere is shown cut by six user clip planes. This demo requires 3D hardware which supports user clip planes.

OGLESVase	
Vase Translucency and reflections	Shows texturing with transparency and environment mapping using a transparent vase with dynamic reflections on its metallic parts.

OGLESChameleonMan	
	Demonstrates per-pixel DOT3 lighting and skinning animation using custom made VGP code. This demo requires 3D hardware which supports GL_IMG_vertex_program and will be not present in the SDK for platforms with no VGP core.

OGLESPhantomMask	
PhantomMask VGP Spherical Harmonics Lighting	Shows spherical harmonics lighting using custom made VGP code. Spherical harmonic lighting is a real-time rendering technique that uses a pre-process step - the projection of the lights, of the model and of the transfer function onto the spherical harmonic basis - to render realistic scenes using any type of light source. It is primarily used to reproduce diffuse lighting. This demo shows a mask lit using spherical harmonics and regular diffuse vertex lighting. The diffuse vertex lighting case requires at least four light sources to produce a similar effect to the spherical harmonics case for this scene. Implementing four diffuse lights requires more vertex program instructions than the spherical harmonics calculations. Realistically many more vertex lights would have to be added to fully match the result of the spherical harmonics lighting. This demo requires 3D hardware which supports GL_IMG_vertex_program and will be not present in the SDK for platforms with no VGP core.

OGLESFur	
	Demonstrates a method for rendering fur. Multiple translucent "shells" of the original model are rendered, each slightly larger than the previous. The texture applied to these layers contains a "dot" in the alpha channel to show where a hair passes through the "shell."

Coverflow	
	Demonstrates how to implement a simple version of 'cover flow'.

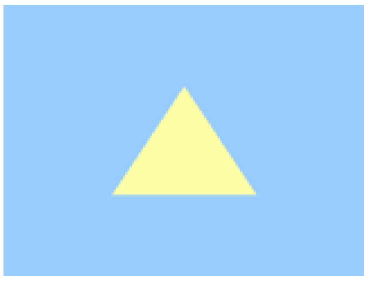
1.4. Training Course

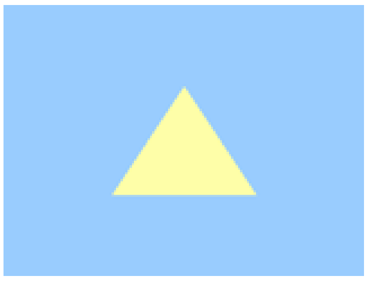
The Training Course folder contains several simple applications to show specific features in a simplified form. The code has been thoroughly commented to help developers to understand the API and get started.

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

Note: The demos in the TrainingCourse folder 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 ES. It does a simple background clear.

HelloTriangle	
	Shows how to draw simple untextured geometry.

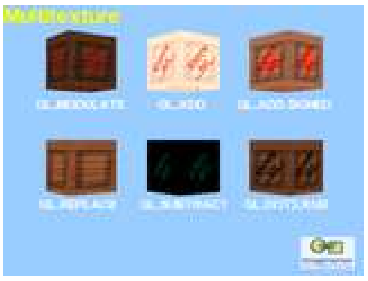
IntroducingPVRShell	
	Shows how to use PVRShell as the OS and API initialisation framework using the same geometry used in the previous example.

Texturing	
	Shows how to load and use textures.

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

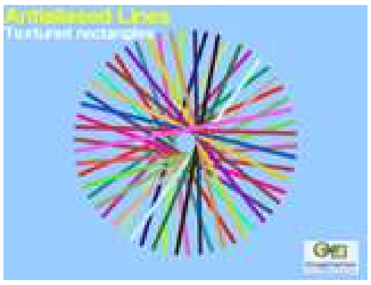
IntroducingPVRTools	
	Shows some of the features available in PVRTools supplied with the SDK. In this case shows how to load textures and display text on the screen.

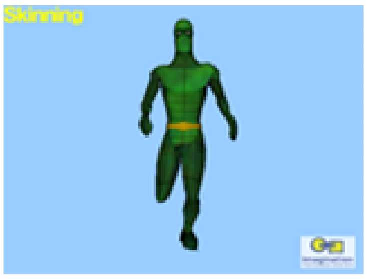
IntroducingPOD	
	Shows how to load and playback a POWERVR POD files exported from 3ds Max.

Multitexture	
 Multitexture GL_MODULATE GL_ADD GL_ADD_SIGNED GL_REPLACE GL_SUBTRACT GL_DOT3_RGB	Shows how to combine two textures in six different ways using different multitexture modes.

RenderToTexture	
	How to perform render to texture using pbuffers (pixel buffers). It renders to two surfaces to create a nice recursive fractal effect.

MatrixPalette	
	Shows the loadings of a 3D scene from a pod file and then drawing and animating the skinned character using internal bones. Matrix Palette refers to the set of matrices which are used to calculate the position of every vertex in the skin. Every vertex has information about which matrices (bones) affect it and the amount of deformation. *Note: This demo might not be supported in all platforms.*

AntialiasedLines	
	This training course shows how to render anti-aliased lines with round caps using texture filtering and blending.

Skinning	
	Demonstrates how to use matrix palette with bone batching.

1.5. PVRShell and command-line options

The POWERVR shell is used in all demos and training course applications 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 also provides a limited keyboard input option which may vary depending on your actual platform. If your 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 full-screen mode(1) or windowed (0)
-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 that help developers achieve common tasks associated with 3D graphics rendering, e.g. model loading, extensions handling, matrix functions, common/common-lite profile functions abstraction, 3D font, etc. All SDK demos make use of the OpenGL ES 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.

PVRTextureTool
Tool to convert bitmap files (e.g. BMP, TGA, etc.) to any texture type supported by MBX hardware. Both a Windows and Linux version are supplied.

PVRTC
PVRTC library for MBX texture compression. Provided as both a Linux and Windows library.

3DSMAX Plugin
Legacy plug-in, now superseded by PVRGeoPOD. PVRexp_v4.dle can be used with 3DSMax 4 and 3DSMax 5 while PVRexp_v6.dle can only be used with 3DSMax 6 and PVRexp_v7.dle can only be used with 3DSMax 7.

PVRGeoPOD
Official POWERVR plug-in currently available for all versions of 3DSMAX and Maya Exports model data from the packages listed above in requested formats. Also supports special features, e.g. tangent space generation and bone batching.

VGPCompiler
Command-line utility to compile VGP programs for use with the GL_IMG_Vertex_Program extension. Note: This utility might not be present for platforms which do not have VGP support.

PVRUniSCo Editor
Editor to be used with the VGPCompiler. It offers a very friendly and powerful GUI user interface.

PVRShaman
PowerVR Shader composer and POD file viewer.

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 the 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 to 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

SDK installation and build notes for all platforms supported by POWERVR cores are described in the following sections.

3.1. PC Emulation Windows

Requires a PC equipped with the Windows XP operating system with 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.

1. Unpack the contents of this package to a local folder on 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.
2. Copy the following drivers to a DLL-accessible directory prior to running the SDK applications (e.g. standard Windows directory or local directory).

   The OpenGL ES emulation “drivers” are called gles_cm.dll (Common profile) and gles_cl.dll (Common-Lite profile) and are located by default in the following directory: SDKPackage\Builds\OGLES\WindowsPC\Lib

Note: Tools and Demos can be launched directly from their project files.

3.2. PC Emulation Linux

1. Unpack the contents of this package to a local folder on a PC equipped with a Linux operating system using “tar –zxvf”.
2. Copy the following drivers to somewhere accessible by the OS. To do this put them into your library path by using: “export LD_LIBRARY_PATH=(folder containing the .so files)”

   The OpenGL ES emulation “drivers” are called libgles_cm.so.0 (Common profile) and libgles_cl.so.0 (Common-Lite profile) and are located by default in the following directory: SDKPackage\Builds\OGLES\LinuxPC\Lib
3. To build a component, first set the platform to LinuxPC using “export PLATFORM=LinuxPC”
4. Go into the component Build/LinuxGeneric folder and type ‘make’. You can also build a CommonLite version by typing ‘make CommonLite=1’ or a Debug version by typing ‘make Debug=1’. The resulting executable will go into a subfolder of Build/LinuxPC.
5. In some of distributions of Linux you encounter a problem when running applications using PCEmulation then you may need to install glew libraries. On Ubuntu it may be done with commands :

   sudo apt-get install libglut3-dev -y

   sudo apt-get install libglew-dev -y

   sudo apt-get install glew-utils -y

Note: Demos and benchmarks can be launched directly from the Binary folder.

3.3. Windows Mobile 5, PocketPC

1. Download and install the Windows Mobile 5 SDK for Pocket PC from the following location: http://msdn.microsoft.com/mobility/downloads/sdks/default.aspx
2. Unpack the contents of this package to a local folder on a PC equipped with the Windows XP operating system. The project and solution files provided (.vcproj and .sln, for Microsoft Visual Studio 2005) do not require the SDK to be installed in a pre-defined location and are configured to use relative-paths.

   The OpenGL ES libraries gles_cm.lib (Common profile) or gles_cl.lib (Common-Lite profile) should be in the folder Builds/OGLES/<platform>/Lib.

3. Demos and Training Courses can be launched directly from their Visual Studio project files:
    - In the "Configuration Manager" dialog box, make sure the "Active solution platform" is set to "Windows Mobile 5.0 Pocket PC SDK (ARMV4I)".
    - Set the "Target Device" to "Windows Mobile 5.0 Pocket PC Device" (it may have defaulted to "Emulator."

3.4. Windows Mobile 5, SmartPhone

1. Download and install the Windows Mobile 5 SDK for Smartphone from the following location: http://msdn.microsoft.com/mobility/downloads/sdks/default.aspx
2. Unpack the contents of this package to a local folder on a PC equipped with the Windows XP operating system. The project and solution files provided (.vcproj and .sln, for Microsoft Visual Studio 2005) do not require the SDK to be installed in a pre-defined location and are configured to use relative-paths.

   The OpenGL ES libraries gles_cm.lib (Common profile) or gles_cl.lib (Common-Lite profile) should be in the folder Builds/OGLES/<platform>/Lib.
3. Demos and Training Courses can be launched directly from their Visual Studio project files:
    - In the "Configuration Manager" dialog box, make sure the "Active solution platform" is set to "Windows Mobile 5.0 Smartphone SDK (ARMV4I)".
    - Set the "Target Device" to "Windows Mobile 5.0 Smartphone Device" (it may have defaulted to "Emulator."

3.5. PocketPC 2003

1. Download Embedded Visual C++ 4.0 from the following location: http://msdn.microsoft.com/mobility/othertech/eVisualc/
2. Download Embedded Visual C++ 4.0 Service Pack 3 or above from the same location.
3. Download the SDK for Windows Mobile 2003-based Pocket PCs from the following location: http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=9996b314-0364-4623-9ede-0b5fbb133652
4. Install Embedded Visual C++ 4.0.
5. Install the Embedded Visual C++ 4.0 Service Pack you've downloaded.
6. Install the SDK for Windows Mobile 2003-based Pocket PCs.
7. Unpack the contents of this package to a local folder on your PC. The Embedded Visual C++ 4.0 project files provided do not require the SDK to be installed in a pre-defined location and are configured to use relative-paths.

   The OpenGL ES libraries gles_cm.lib (Common profile) or gles_cl.lib (Common-Lite profile) should be in the folder Builds/OGLES/<platform>/Lib.

   Demos and Training Courses can be launched directly from these Embedded Visual C++ 4.0 project files (.vcw and .vcp). Before building make sure the Active WCE Configuration is set to POCKET PC 2003 and the Default Device to POCKET PC 2003 Device.

3.6. Windows CE 5

1. Download Embedded Visual C++ 4.0 from the following location: http://msdn.microsoft.com/mobility/othertech/eVisualc/
2. Download Embedded Visual C++ 4.0 Service Pack 4 from the same location.
3. Export an SDK for application development from Platform Builder. This process is described here: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wceosdev5/html/wce50howHowtoCreateanSDKforaCustomPlatform.asp
4. Install Embedded Visual C++ 4.0.
5. Install the Embedded Visual C++ 4.0 Service Pack you've downloaded.
6. Install the exported application-development SDK.

7. Unpack the contents of the POWERVR SDK package to a local folder on your PC. The Embedded Visual C++ 4.0 project files provided do not require the SDK to be installed in a pre-defined location and are configured to use relative-paths.

   The OpenGL ES libraries gles_cm.lib (Common profile) or gles_cl.lib (Common-Lite profile) should be in the folder Builds/OGLES/<platform>/Lib.

   Demos and Training Courses can be launched directly from these Embedded Visual C++ 4.0 project files (.vcw and .vcp). Before compiling make sure the Active WCE Configuration is set to your exported application-development SDK.

3.7. Linux

1. OpenGL ES libraries are not distributed with the POWERVR Linux SDKs. Ask your platform provider for these libraries if you do not have them.
2. Define the environment variable LIBDIR to point to the directory containing the OpenGL ES libraries, e.g. “export LIBDIR=/home/user/Linux_Baseport/target/usr/lib”
3. Install the latest platform toolchain for your target platform in your Linux system.
4. Define the TOOLCHAIN environment variable to the toolchain directory, e.g. “export TOOLCHAIN=/data/omap_binaries/linux/arm/”, or add the path to the toolchain to the PATH environment variable (i.e. run “export PATH=”<path to the toolchain>:$PATH”).
5. If you want an X11 build and it is available, define the environment variable X11ROOT to point to the freedesktop directory (i.e. “export X11ROOT=/usr/X11R6_SGX”)

   To build individual components go to the directory

   Demos/<DemoName>/OGLES/Build/LinuxGeneric or

   TrainingCourse/<TCName>/OGLES/Build/LinuxGeneric and run the command**:**

   LinuxRaw: “make PLATFORM=<platform> <ESProfile>=1”

   LinuxX11: “make PLATFORM=<platform> X11BUILD=1 <ESProfile>=1”

   where: ESProfile is “Common” or “CommonLite” and platform consists of the word “Linux” followed by the name of the device you are building the application for, e.g. LinuxOMAP3, LinuxX86

   Depending on the chosen ESProfile , platform, and the value of X11BUILD, the executables for the demos and training courses will go to:
   Demos/<DemoName>/OGLES/Build/<platform>/<ReleaseDir> or
   TrainingCourse/<DemoName>/OGLES/Build/<platform>/<ReleaseDir>, where

   ReleaseDir is one of: ReleaseRaw, ReleaseRawLite, ReleaseX11, ReleaseX11Lite
6. Ensure that the POWERVR drivers are installed on your target device (please refer to DDK / driver installation instructions).
7. If the standard c++ libraries are not present on your target device, copy libc++ from the toolchain into /usr/lib. libdl and libgcc may also be required.
   Note: *libc++ lives at /usr/lib if you have installed the drivers, or can be found as part of a binary driver release package.*
8. Ensure the drivers are running (e.g. type /etc/init.d/rc/pvr start, then run an X session if required).

   Under X11 window sizes can be specified for the executables using the command line arguments -posx=n and -posy=n to define the top right hand corner, and -width=n and -height=n to define width and height respectively. E.g.
   ./OGLESEvilSkull -posx=10 -posy=10 -width=100 -height=100
9. If you attempt to run an SDK training course or demo and it fails with the message: “Can’t open display” produced by the X client then make sure that the DISPLAY variable is set with the shell command: “set | grep –e DISPLAY”. If this command doesn’t yield any output then type (in shell): “DISPLAY=:0.0; export DISPLAY”

3.8. Symbian

1. Install latest Symbian release.
   Note: In the following [SYMBIAN] is used to refer to the Symbian installation directory.

2. To build individual components go to the /Build/SymbianTextShell or Build/SymbianTechView sub-directory of the component to build and type:

   ```
   bldmake bldfiles
   abld build <platform> urel
   ```

   Note: For OMAP systems use armv5 as platform

   The .exe will be automatically copied in the correct place in the Symbian SDK tree to be included in the image through the OBY file.

   Pre-built binaries can be found in the Binaries folder ready to be dropped in your Symbian SDK (exes and IBY for TechView and TextShell, and resources for TechView)

   Refer to your platform BasePort documentation to know how to build an image and how to download it to the platform. We supply in the package the OBY and IBY files that will allow you to include the SDK demos in the image.

Note: Due to a Symbian DevKit problem with long path names the TrainingCourse demos will not build out of the package. Please, install SDKPackage/ in you root path and rename it to a shorter name (e.g. SP/) to be able to build the TrainingCourse applications.

3.9. Symbian UIQ3.x

Install the UIQ 3.x SDK that you wish to use from http://developer.uiq.com/devtools_uiqsdk.html. Also, if required install the platform extension pack.

3.9.1. Building for hardware

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

   ```
   bldmake bldfiles
   abld build gcce urel
   ```

2. To create an installation package (.SIS file), type:

   ```
   makesis –d%EPOCROOT% [packagename].pkg
   ```

   Where [packagename] is the name of the .pkg file in the /Build/SymbianUIQ directory.

   Note:

   Some versions of makesis do not fully support the –d option so you may experience some problems when creating the .SIS file. If this happens either install the newest UIQ SDK or update makesis with the one from http://developer.uiq.com/devtools_other.html#makesis.

3. On some UIQ devices the .SIS file may need to be signed before it can be installed. To do this type:

   ```
   signsis [sisname] [sisname] [certificate] [key] [pass]
   ```

   where [sisname] is the name of the .SIS file, [certificate], [key] and [pass] are the certificate, key and password to use. You can either create and use your own certificate and key for signing or you can use the ones provided in /Builds/Symbian which have the password “ImgTec”. To do this type

```
signsis [sisname] [sisname] ..\..\..\..\..\Builds\Symbian\ImgTec.cer
..\..\..\..\..\Builds\Symbian\ImgTec.key ImgTec
```

4. Pre-built and signed .SIS files can be found in the ‘Binaries’ folder.
5. Please follow the manufacturer’s instructions on how to install the .SIS files onto your phone.

3.9.2. Building for the emulator

1. To build individual components go to the /Build/Symbian_S60_3rd sub-directory of the component to build and type:

```
bldmake bldfiles
abld build winscw udeb
```

2. An executable is then created in Epoc32\release\winscw\udeb in your Symbian installation that can be ran by double-clicking it.

Note:

Due to a Symbian DevKit problem with long path names the TrainingCourse demos will not build out of the package. Please, install SDKPackage/ in your root path and rename it to a shorter name (e.g. SP/) to be able to build the TrainingCourse applications.

Also some demos and training courses may not run on the Symbian emulators because they use special hardware extensions that the emulators do not support.

3.10. Symbian S60 3rd Edition

Install the Symbian S60 3rd Edition SDK, and the OpenGL ES 1.1 Plug-in available from http://www.forum.nokia.com/

3.10.1. Building for hardware

1. To build individual components go to the /Build/Symbian_S60_3rd sub-directory of the component to build and type:

```
bldmake bldfiles
abld build armv5 urel
```

2. To create an installation package (.SIS file), type:

```
makesis –d%EPOCROOT% [packagename].pkg
```

[packagename] is the name of the .pkg file in the /Build/Symbian_S60_3rd directory.

3. Before the .SIS file can be installed on the phone it needs to be signed. To do this type:

```
signsis [sisname] [sisname] [certificate] [key] [pass]
```

where [sisname] is the name of the .SIS file, [certificate], [key] and [pass] are the certificate, key and password to use. You can either create and use your own certificate and key for signing or you can use the ones provided in /Builds/Symbian which have the password “ImgTec”. To do this type

```
signsis [sisname] [sisname] ..\..\..\..\..\Builds\Symbian\ImgTec.cer
..\..\..\..\..\Builds\Symbian\ImgTec.key ImgTec
```

4. Pre-built .SIS files can be found in the ‘Binaries’ folder.
5. Please follow the manufacturer’s instructions on how to install the .SIS files onto your phone.

3.10.2. Building for the emulator

1. To build individual components go to the /Build/Symbian_S60_3rd sub-directory of the component to build and type:

```
bldmake bldfiles
abld build winscw udeb
```

2. An executable is then created in Epoc32\release\winscw\udeb in your Symbian installation that can be ran by double-clicking it. If the demo uses external files then these need to be copied to Epoc32\winscw\c\system\data.

Note:

Due to a Symbian DevKit problem with long path names the TrainingCourse demos will not build out of the package. Please, install SDKPackage/ in your root path and rename it to a shorter name (e.g. SP/) to be able to build the TrainingCourse applications.

Also some demos and training courses may not run on the Symbian emulators because they use special hardware extensions that the emulators do not support.

3.11. Windows Mobile 7

1. Open 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]\OGLES\Build\PlatformBuilder and the executables are built in [app]\OGLES\Build\PlatformBuilder\oak\target\ARMV4I\retail.

3.12. Windows Mobile 6.1, Standard

1. Download and install the Windows Mobile 6.1 Standard SDK from the following location: http://www.microsoft.com

2. Unpack the contents of this package to a local folder on a PC equipped with the Windows XP operating system. The project and solution files provided (.vcproj and .sln, for Microsoft Visual Studio 2005) do not require the SDK to be installed in a pre-defined location and are configured to use relative-paths.

   The OpenGL ES libraries should be in the folder Builds/OGLES/<platform>/Lib.
3. For building in the "Configuration Manager" dialog box, make sure the "Active solution platform" is set to "Windows Mobile 6 Standard SDK (ARMV4I)".

3.13. Windows CE 6 CEPC (x86)

1. Copy the POWERVR and third party folders from the DDK into your `\WINCE600\PUBLIC` folder.
2. Create a new OS Design Project in Visual Studio 2005 (with Platform Builder 6 installed).
   a. When choosing the BSP select "CEPC: x86".
   b. Choose a device and press finish.
3. In the "Catalog Items View" window under Third Party tick "POWERVR XXX – OpenGL ES 1.1".
4. In the "Catalog Items View" window tick
5. Core OS -> CEBASE -> Applications and Services Development -> C Libraries and Runtimes -> C++ Runtime Support for Exception Handling and Runtime Type Information.
6. Build the OS Design.
7. Copy the SDK Binaries into the release directory and execute from the command line when your device has been attached.

3.13.1. Note

1. To build the SDK examples you need to install "**WindowsCE 6 CEPC Default SDK.msi**" included in the package.
2. When building our tools if you get an error message similar to

   PVRTGeometry.cpp
   cwchar(75) : error C2039: 'wcsftime' : is not a member of '`global namespace''
   cwchar(75) : error C2873: 'wcsftime' : symbol cannot be used in a using-declaration
   malloc.h(45) : error C2143: syntax error : missing ',' before '*'
   ..\..\..\PVRTGeometry.cpp(1064) : error C3861: 'min': identifier not found

   then go to

   Tools -> Options -> Projects and Solutions -> VC++ Directories

   and set **"Platform"** to **"WindowsCE 6 CEPC Default (x86)"** and **"Show directories for"** to **"Include files"**

   Then move the **"WindowsCE6 CEPC Default"** directories above **"$(VCInstallDir)ce\include"**.

3. For examples that use external files (i.e. .pod, .pfx) these files need to be copied across to the root of the device using the **"file viewer"** under

   Target->Remote Tools

3.14. iPhone OS

1. Download a version of Apple's iPhone SDK from the following location: http://developer.apple.com/iphone/. You will need to become a member of Apple's developer programme in order to access this page. You can find details of how to join this at: http://developer.apple.com
2. Install the Apple SDK on your Mac as specified by Apple's hardware/software requirements. This should also install Xcode and the other development tools required.
3. Expand the POWERVR iPhone SDK to a location that you have read/write access for.
4. To build the demos, training course and other projects of the SDK, find the various OGLESXXX.xcode projects available within the SDKPackage directory and double click these to launch them in Xcode.
5. To build for an iPhone OS device you will need a valid developer certificate in your machine's keychain. You may also have to change the Properties:Identifier property from Project:Edit Active Target… to match that which you have set up for yourself through Apple's Program Portal.
6. If you don't have a developer certificate from Apple then you can still build and launch applications in the iPhone Simulator. Choose this configuration from the drop-down menu in the top left and then choose "Build and Run" from the drop-down menu.

4. Support Contact

For further support contact:

devtech@imgtec.com

Imagination Technologies Ltd.
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 or Imagination Technologies Ltd. visit our web pages at:

www.imgtec.com