

OpenGL ES 2.x SDK

User Guide

Copyright © 2009, Imagination Technologies Ltd. All Rights Reserved.

This document is confidential. Neither the whole nor any part of the information contained in, nor the product described in, this document may be adapted or reproduced in any material form except with the written permission of Imagination Technologies Ltd. This document can only be distributed to Imagination Technologies employees, and employees of companies that have signed a Non-Disclosure Agreement with Imagination Technologies Ltd.

Filename : OpenGL ES 2.x 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 2 SDK Content.....	3
1.1.	Introduction.....	3
1.2.	Documentation	3
1.3.	Demos	3
1.4.	Training course.....	4
1.5.	PVRShell	12
1.6.	PVRTools	12
1.7.	Utilities	12
2.	Memory File System and FileWrap	14
2.1.	Using the Memory File System in your own Projects.....	14
3.	OpenGL ES 2 SDK Installation	15
3.1.	PCEmulation.....	15
3.1.1.	Windows PC Emulation.....	15
3.1.2.	Linux PC Emulation	15
3.2.	Linux	15
3.3.	Symbian.....	16
3.4.	ulTRON	17
3.5.	Symbian S60 5th Edition	17
3.5.1.	Building for hardware	17
3.1.	Windows Mobile 7	18
3.2.	Windows Mobile 6.1, Standard.....	19
3.3.	Windows CE 6 CEPC (x86).....	19
3.3.1.	Note	19
3.4.	iPhone OS	20
4.	Support Contact.....	21

1. OpenGL ES 2 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. Documentation

The following documents are located in `\SDKPackage` or in `\SDKPackage\Documents`.

OpenGL ES 2 SDK.User Guide
Description of OpenGL ES 2 SDK contents and installation.

Migration from software rendering to 3D accelerated graphics using OpenGL ES.Khronos Kolumn
An article posted in the Khronos Kolumn (http://www.gamedev.net/reference/articles/article2268.asp) which will help developers who are now starting to code for hardware accelerated devices.

POWERVR Effect File (PFX)
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.

POWERVR SGX.Application Development Recommendations
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 the Shell section below.

The 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.

Shaders



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 OpenGL ES 2.0 API.

Coverflow



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

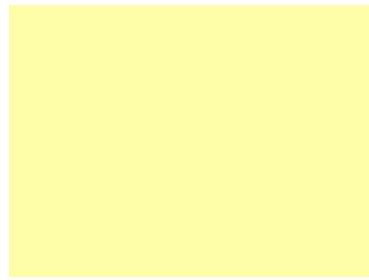
1.4. Training course

Several simple applications are provided to show specific features in a simplified form. The code has been thoroughly commented to help developers to understand the API and get started.

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

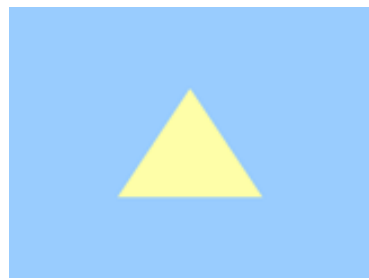
Note: The demos in the TrainingCourse folder do not handle screen rotation to keep the code as simple as possible. On devices with a portrait display the images below might be shown stretched.

Initialization



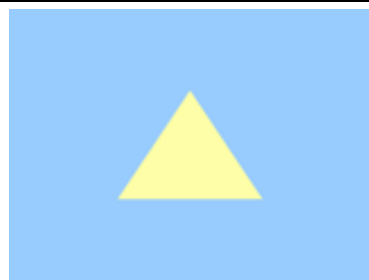
Shows how to initialise OpenGL ES. It does a simple background clear.

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 as the previous example.

Texturing



Shows how to load and use textures.

BasicTnL



Shows how to transform and light simple geometry using the same geometry as the previous example.

IntroducingPVRTools



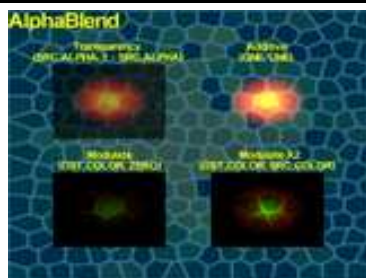
Shows some of the features available in the PVR tools supplied with the SDK. In this case shows how to load textures and display text on the screen.

IntroducingPOD



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

AlphaBlend



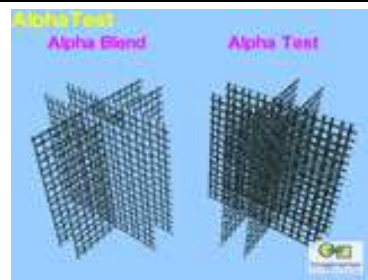
Shows 4 different types of blending mode.

RenderToTexture



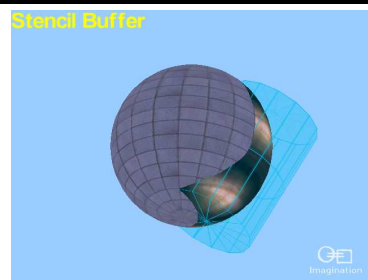
How to perform render to texture using the Frame Buffer Objects. It renders to two surfaces to create a nice recursive fractal effect.

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 'pre-baked' 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



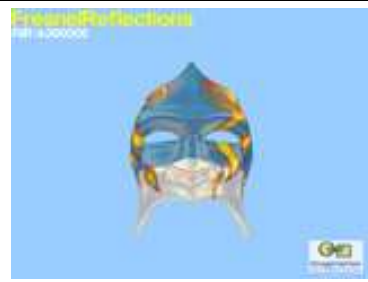
Shows how to use a normal map to perturb uvs.

CellShading



How to implement cell shading.

FresnelReflections



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



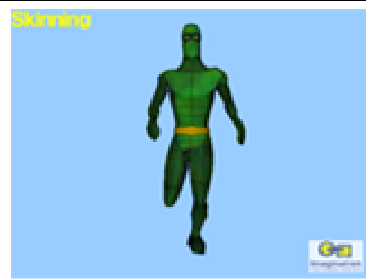
This example demonstrates two techniques for generating an anisotropic lighting effect on the mask. One calculates the shading for each pixel on the fly; the other is optimised by using a texture to lookup the shading values. Anisotropic lighting simulates the effect of surfaces with a grain, or grooves.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

Iridescence



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 stencil buffer shadow volumes.

DisplacementMap



Shows how to displace geometry in the vertex shader using a texture.

ShadowMapping



Demonstrates shadow mapping using GL_OES_depth_texture extension.

1.5. PVRShell

The POWERVR shell is used in all Demos and Training Courses to provide a common framework for developing OpenGL ES 2 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 designed 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.

PVRTexTool PVRTexLib PVRTC

Tools to convert bitmap files (e.g. BMP, TGA, etc.) to any texture type supported by POWERVR hardware.

PVRGeoPOD

Plug-in for 3DStudio MAX and Maya to export optimized 3D data in POD format.

Collada2POD

Command-line utility to convert Khronos intermediate format to POWERVR POD format.

PVRShaman

POWERVR Shader Composer for OpenGL ES 2

PVRUniSCo

SGX shader compiler.

Note: This tool might not be available in all releases.

PVRUniSCo Editor

Shader editor and GUI front end for PVRUniSCo.

PVRTune

Remote hardware performance profiler.

Note: This tool might not be available in all releases.

PVRScope

Platform specific library to access the hardware performance counters.

Note: This tool might not be available in all releases.

Filewrap

Utility to wrap a text or binary file into a C++ source file that can be compiled and linked to an application. See the 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 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 2 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

Requires a PC equipped with the Windows XP and 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 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.
2. 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).

Note: The 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 2 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’. The resulting executable will go into a subfolder of Build/LinuxPC.
6. 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: Running OGLES2 PC Emulation is currently not supported in 64 bit Linux environments

3.2. Linux

1. OpenGL ES 2 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 2 libraries, e.g. "export LIBDIR=/home/user/Linux_Baseport/target/usr/lib"
3. Install the latest platform toolchain for your target platform.
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>/OGLES2/Build/LinuxGeneric or

TrainingCourse/<TCName>/OGLES2/Build/LinuxGeneric and run the command:

LinuxRaw: "make PLATFORM=<platform>"

LinuxX11: "make PLATFORM=<platform> X11BUILD=1"

where 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 platform and the value of X11BUILD, the executables for the demos and training courses will go to: Demos/<DemoName>/OGLES2/Build/<platform>/<ReleaseDir> or TrainingCourse/<DemoName>/OGLES2/Build/<platform>/<ReleaseDir>, where ReleaseDir is one of: ReleaseRaw, ReleaseX11

6. Ensure that the POWERVR drivers are installed on the 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.
./OGLES2IntroducingPOD -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"

3.3. 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 SDK apps may not build out of the package. To keep the paths short please install SDKPackage/ in your root path and/or rename it to a shorter name (e.g. SP/) to be able to build the applications.

3.4. ulTRON

To build for ulTRON 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 <code>leurasia\leurasiacon\build\uitron</code> .
SHC_DIR	The tools you wish to use, e.g. <code>\Tools\Renesas\sh\8_05</code> .

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 <code>uiton_shmobile_release</code> in <code>PLATFORM\tools</code> . It will be of the form <code>SGXxxx</code>
-----------------	---

SGX_CORE_REV SGX_CORE_REV takes its value from the remainder of the suffix.

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

```
nmake Makefile.mak
```

3.5. Symbian S60 5th Edition

3.5.1. Building for hardware

1. If you haven't already done so you will need to download and install the Symbian S60 5th Edition SDK from www.forum.nokia.com. Currently the SDK doesn't come with the required libraries for compiling OGLES2 applications. You should be able to get these from your mobile phone manufacturer and these will go in `epoc32\release\armv5\lib`.
2. From a command-line go to the `/Build/SymbianS60` sub-directory of the component to build. If it is not already you need to make the Symbian S60 5th Edition SDK the default device for building. To check that it is type:

```
devices
```

The default SDK will have “-default” next to it. If it is not the default then type

```
devices -setdefault [symbianSDK]
```

where [symbianSDK] is the SDK you want listed by devices

e.g. S60_5th_Edition_SDK_v1.0:com.nokia.s60.

Once ready you can build the app by typing:

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

3. 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/SymbianS60 directory.

4. 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
```

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

3.1. 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]\OGLES2\Build\PlatformBuilder and the executables are built in [app]\OGLES2\Build\PlatformBuilder\oak\target\ARMV4\retail.

3.2. 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 2 libraries should be in the folder Builds/OGLES2/<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)”.

3.3. 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 2.0”.
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.

3.3.1. 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

3.4. 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 OGLES2XXX.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

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

For more information about POWERVR or Imagination Technologies Ltd. visit our website
www.imgtec.com.