OpenGL ES 2.0 on the iPhone 3G S

Dan Ginsburg

On June 8th, 2009 at the Apple Worldwide Developer Conference (WWDC) the new iPhone 3G S was announced. Apple confirmed that this new generation of iPhone would support OpenGL ES 2.0. Meanwhile, Apple made available to developers a seed version of the iPhone SDK 3.0 that includes support for OpenGL ES 2.0. The new iPhone SDK allows developers to write application for the iPhone 3G S that use OpenGL ES 2.0 for rendering 3D graphics.

The introduction of the iPhone 3G S to the market marks a major milestone in the adoption of OpenGL ES 2.0. This new iPhone will put the power of programmable 3D graphics in the hands of millions of consumers. With this power comes the challenge to developers to get the most out of the programmable hardware by developing high-performance shader-based 3D applications. The OpenGL ES 2.0 Programming Guide was written with exactly this goal in mind. The book covers every aspect of the API along with various advanced 3D rendering techniques to help you get the most out of OpenGL ES 2.0.

Upon first publication of the OpenGL ES 2.0 Programming Guide, the only implementations of OpenGL ES 2.0 that were available were PC-based emulators. While these emulators are very useful in developing the relevant OpenGL ES code for an application, many of the platform details that developers encounter working with real-world devices are missed. With the availability of the iPhone SDK 3.0, it now possible to develop OpenGL ES 2.0 applications for one of the world's most popular handheld devices.

The purpose of this chapter is to help you get started developing OpenGL ES 2.0 on the iPhone 3G S. Throughout the book, we provided a variety of sample code to demonstrate various techniques and rendering functionality in OpenGL ES 2.0. We have ported all of this sample code from the previous chapters in the book to run on the iPhone 3G S. This chapter details what was involved in doing this port and what information you need to know to get up and running to develop OpenGL ES 2.0 applications for the iPhone 3G S.

Getting Started with the iPhone SDK 3.0

The iPhone 3.0 SDK is available from the Apple developer website at http://developer.apple.com/. As of this writing, the iPhone SDK 3.0 requires that you have an Intel-based Mac running Mac OS X 10.5.7 or later. The iPhone SDK 3.0 includes everything you will need to get started with developing OpenGL ES 2.0 applications including the Xcode IDE as well as the iPhone Simulator. The iPhone Simulator allows you to run, test, and debug your iPhone applications directly on the Mac. The iPhone Simulator provides support for running both OpenGL ES 1.1 and OpenGL ES 2.0-based applications.

Getting the Sample Code for iPhone

In order to aid you in using the content of the OpenGL ES 2.0 Programming Guide on the iPhone platform, we have ported all of the C-based sample code to the iPhone SDK 3.0. The updated code can be downloaded from the book website at http://www.opengles-book.com. The

package on the book website includes all of the sample code along with Xcode projects for building each of the samples.

Throughout the book, we developed an ES Utility Framework that provides basic common functionality that is needed by most OpenGL ES 2.0 applications. This framework includes functions for generating simple geometry, compiling and linking shaders, computing modelview and projection transformation matrices, and creating rendering surfaces.

The ES Utility Framework that was developed for the original book has been adapted to run on the iPhone. Several of the function calls that were used for setting up EGL (esCreateWindow()) and setting up callbacks (esRegisterDrawFunc(), esRegisterUpdateFunc(), etc.) did not map particularly well to the iPhone and were deprecated. However, all of the code that was relevant to creating 3D geometry, transformations, and loading shaders/programs was ported without modification. These functions will be especially helpful to developers migrating from the OpenGL ES 1.1 fixed-function pipeline in generating equivalent transformation matrices.

Building the Sample Code

As with the Windows-based version of the sample code, the ES Utility Framework is built into a static library that each of the individual samples links against. Each sample contains only the relevant rendering and setup code. The project is organized in the following directory structure:

- /Common – holds the source code and Xcode project for building the ES Framework library (called libCommon.a)
- /Chapter_X – holds each of the individual sample code along with the Xcode projects

In order to build any of the sample programs, the first thing you will need to do is build the ES Utility Framework located in the *Common/* folder. Assuming you have installed the iPhone SDK 3.0, the only thing you need to do is open the *Common.xcodeproj* project and do a build. This will generate the *libCommon.a* file that contains the ES Utility Framework. All of the other sample code is setup to link against this library.

The next step is to simply open any of the individual sample projects in Xcode. For example, to build the MipMap2D texture sample from Chapter 9, simply open *Chapter_9/MipMap2D/MipMap2D.xcodeproj* in and then click *Build and Go*. The sample will come up in the iPhone Simulator as pictured in Figure 1.

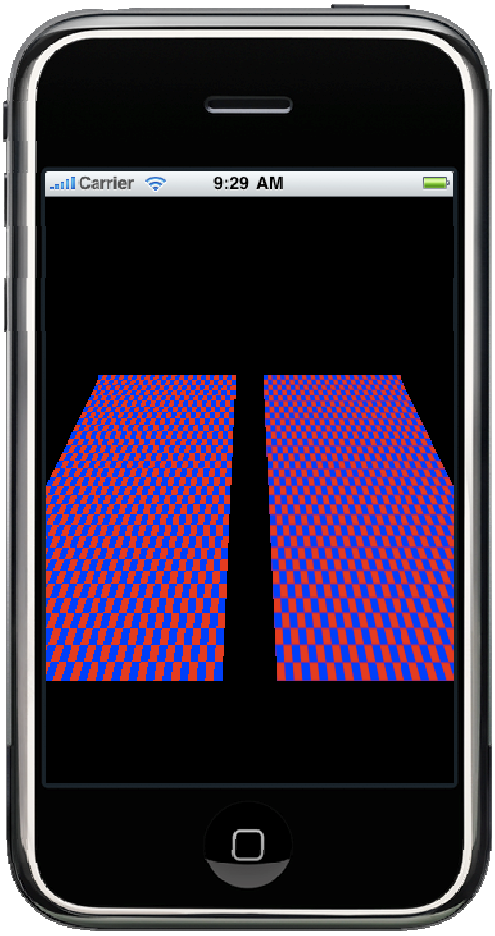

Figure 1 – MipMap2D sample on iPhone Simulator

That's all there is to getting the samples to run on the iPhone. The iPhone SDK 3.0 makes getting up and running very simple which allows you to focus on the relevant graphics code of each sample.

Porting the Sample Code to the iPhone

The process of porting the sample code from the book to the iPhone highlights some of the unique features of the iPhone. In particular, some of the unique features of the development environment include:

- The use of Objective C
- The process of creating an EGL rendering context using the `EAGLContext`.
- Detection of device capabilities and creating an OpenGL ES 2.0 context
- The use of a framebuffer object as the primary rendering surface

Once you have created and setup a rendering context, programming with OpenGL ES 2.0 on the iPhone is identical to any other platform. However, there are some unique aspects of the platform that are worth understanding before jumping into the code.

Objective C

Apple utilizes the Objective C language within their development framework. Objective C is an object oriented language that is a superset of C. Files that are in Objective C have the *.m* file extension. There are a number of great tutorials online that will help you understand the fundamental differences between C/C++ and Objective C. As Objective C is the primary language used on the iPhone, you will definitely want to get an understanding of it before moving too far in development.

One of the great things about Objective C is because it is a strict superset of C, any code written in C will also compile with the Objective C compiler. The sample code in our book was all written in C so this made it quite easy to port all of the code. The main changes to the code were around the event system and EGL initialization. By and large, the core of each sample ported to the platform without modification.

Creating an OpenGL ES 2.0 Rendering Surface

In Chapter 3 of the OpenGL ES 2.0 Programming Guide, we cover in detail how to create a rendering surface and context using EGL. We developed the *esCreateWindow()* function that uses EGL to initialize a rendering surface and create a context. On the iPhone, Apple provides their own framework for initializing with EGL and abstracts this from the developer. This framework is documented in the iPhone SDK.

In Xcode, there is a project template for an iPhone OpenGL ES Application. This template creates an object named *EAGLView* in the files *EAGLView.m* and *EAGLView.h*. This object sub-classes the *UIView* object and implements most of the basic necessities for getting up and running with an OpenGL ES application. One change from the template that is necessary is to specifically request an OpenGL ES 2.0 context rather than an OpenGL ES 1.1 context. Unlike other platforms, the iPhone SDK will allow a program to compile and link with calls to both OpenGL ES 1.1 and OpenGL ES 2.0 functions. However, in order for these calls to actually be able to execute, one must request the appropriate context. To request an OpenGL ES 2.0 context in the *EAGLView* class that gets generated, we use the following code.

```
context = [[EAGLContext alloc] initWithAPI:kEAGLRenderingAPIOpenGLES2];
```

Note that in order to write an application that will safely run on either the iPhone 3G or iPhone 3G S, one must do proper runtime checks to detect device capabilities. As OpenGL ES 2.0 is only available on the iPhone 3G S, it is necessary to have an OpenGL ES 1.1 fallback path to run on previous generation iPhone. The call to *initWithAPI* will return *nil* on devices which do not support OpenGL ES 2.0. In this case, an OpenGL ES 1.1 context can be created and a fallback rendering path executed.

In addition to creating an OpenGL ES 2.0 context, the application will also need to include the OpenGL ES 2.0 header files (rather than the OpenGL ES 1.1 headers):

```
#import <OpenGLES/ES2/gl.h>
#import <OpenGLES/ES2/glext.h>
```

After making these changes, the default *drawView* method that draws an OpenGL ES 1.1 spinning cube can be modified to do your own OpenGL ES 2.0 rendering. This was the approach that was taken to creating the samples in the book. The default *EAGLView* class was modified to create an OpenGL ES 2.0 context and callback into the C sample code from the book.

Using a Framebuffer Object for Rendering

One other unique aspect to the iPhone SDK is that rendering is done to an offscreen framebuffer object. The full details of creating and rendering with framebuffer objects is covered in Chapter 12 of the book. On most platforms, one creates a displayable window surface using EGL and then uses *eglSwapBuffers* to present the surface to the screen. This was the approach that was taken by the sample code in the OpenGL ES 2.0 Programming Guide. However, on the iPhone an offscreen renderbuffer is used for all rendering and then presented to the screen using *EAGLContext presentRenderbuffer* method. It was a small change to the sample code to drop the call to *eglSwapBuffers* and wrap the main drawing function for each sample with the code to bind and present the renderbuffer.

Transitioning from OpenGL ES 1.1 to OpenGL ES 2.0

For developers that are currently writing applications that target OpenGL ES 1.1 on the iPhone 3G, the move to OpenGL ES 2.0 may be a bit difficult at first. While OpenGL ES 2.0 is significantly more powerful than OpenGL ES 1.1, it also pushes the responsibility to the developer to implement the vertex transformation and fragment shading pipeline. This means becoming intimately familiar with the fixed-function vertex and fragment pipelines in OpenGL ES 1.1 and translating those pipelines into OpenGL ES 2.0 shaders.

The OpenGL ES 2.0 Programming Guide was written with this type of developer in mind. In Chapter 8 Vertex Shaders, we provide a shader that implements the entire OpenGL ES 1.1 fixed-function pipeline (Example 8-8). This includes vertex transformation, texture coordinate generation, lighting, and vertex fog. In Example 8-6, we also show how to implement matrix palette skinning in a vertex shader. In Chapter 10 Fragment Shaders, we show how fixed-function state from OpenGL ES 1.1 can be translated into fragment shader code. In particular, we cover the texture environment, multitexturing, fog, alpha test, and user-clip planes.

In addition to these chapters, the ES Utility Framework provides functions that do the equivalent of glRotate, glTranslate, and glScale. It also contains functions for generating the perspective transformation matrix and the modelview matrix. These functions should be helpful to a developer that has relied on the fixed-function API of OpenGL ES 1.1 for generating transformation matrices.

Conclusion

The iPhone 3G S brings OpenGL ES 2.0-programmable hardware into millions of consumers' hands. The OpenGL ES 2.0 Programming Guide covers everything in the API to help you render efficient shader-based 3D graphics on the iPhone 3G S. In this chapter we covered the basics of how to get up and running with developing OpenGL ES 2.0 applications for

the iPhone 3G S. Throughout the OpenGL ES 2.0 Programming Guide, we developed a number of C-based samples to demonstrate the use of various features of the API. In order to help you utilize the content of the book on the iPhone, we ported these samples to the iPhone SDK which you can download off of the book website. In addition, we covered some of the unique aspects of programming with OpenGL ES 2.0 on the iPhone.