Introduction to OpenGL Development using C++

What We will be using:

- GLUT
- GLEW
- Open GL (> 2.0 standards)
- C++

Prepare Your Environment

- sudo apt-get install freeglut3-dev
- sudo apt-get install libglew-dev

(Ensure you have -dev files, since you will be compiling!)

How To Compile the Example

```
g++ main.cpp -lglut -lGL -lGLU -lGLEW -o meme
```

- -l tells C++ to include a library in your compile.
- -o is the name of the binary you want to create

Includes For Your Libs

```
#include <GL/glew.h>
#include <GL/gl.h>
#include <GL/glut.h>
```

What is GLEW?

Many cards have different extensions available, GLEW supplies methods for those extensions by probing the video card and altering GL.h. It will create methods for the extensions that exist, and supply software methods for those that don’t.

Why GLEW?

- You won’t have to probe the card for support yourself and implement methods
- Hardware compatibility
- Just saves a shit ton of time.

Setting Up GLEW

```
GlewInit();
```

And that’s it. Just include it, and initialize it, and GLEW will automatically expose OpenGL2+ extensions to you.

What is GLUT?

GLUT supplies us with a window, input handler, and most importantly, a context.

An OpenGL context is what allows you to call openGL methods. This needs to exist before calling any openGL methods, or the method will fail. (So don’t put OGL methods in your constructors, unless your class is dynamic).

Create the Context

- The first thing we'll need to do is create a context.
- Notice the callback methods.
- See the createContext() method in the example.

GLUT Callback methods

- glutIdleFunc(void *(method));
  - Triggers after the buffer is swapped. This is the one we care about the most.
- glutKeyBoardFunc(void*(method(unsigned char key, int x, int y)));
  - This is triggered when a key is pressed.
- glutDisplayFunc(void *(method));
  - Triggers when the window has been altered.
- glutReshapeFunc(void*(method));
  - Triggers when the windows is resized.

Setup OpenGL

After GLUT is configured, begin your configuration of your OpenGL context. See the example for what you usually would do.

Lights:

- You are allowed 8 hardware lights (usually).
- Can be turned on and off, as well as can be positioned and altered on the fly.

Old Way vs New Way Explained

In OpenGL, it was required to call methods to draw each vertex from system memory every frame. Only textures were originally stored in the video memory. The new, and proper way to now render objects in openGL is to load your vertexes to a video memory buffer, and tell openGL to draw from that buffer. So as you will see in the example the method:

- loadCubeToMemory();

Which will load the vertexes for our cube into a video buffer.

Now Explore

Take the time to read through the example and explore what it is doing and how it is drawing the cube. When you get it to compile and run, you should be able to press ‘b’ to switch to the bad way of drawing, and ‘g’ to switch back to the proper way of drawing your cube.

Summary

I hope this tutorial / example gets you started in building an engine in openGL, now you are ready to begin learning more advance openGL rendering tricks and methods.