Introduction to OpenGL Development using C++ 2: Electric Boogaloo

By Anna Wall (Skuld)

What We will be using:

Before we introduced:

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

But now we will introduce:

- libpng

Prepare Your Environment

```
sudo apt-get install libpng-dev
```

Most likely, you will already have this.

How To Compile the Example

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

Includes For Your Libs

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

Why libpng?

Well, we need a way to load textures some how. So it decompresses png files into an array of pixels. Each pixel ends up 4 bytes. A bytes being a value between 0-255, or… 8-bits! We will be using 32-bit color, which means, 8-bits for each color (32/4). But you notice I wrote 4, what's the 4th?

32bit color

So this is what 32-bit color means:

- Red 8-bits (0-255)
- Blue 8-bits (0-255)
- Blue 8-bits (0-255)
- Alpha 8-bits (0-255)

Which you will see written out at times as:

RGBA8

We will be using this format for our tutorial.

16bit color

So this is what 16-bit color means:

- Red 4-bits (0-15)
- Blue 4-bits (0-15)
- Blue 4-bits (0-15)
- Alpha 4-bits (0-15)

Which you will see written out at times as:

RGBA4

Older games used this color format, it just looks bad.

Setting Up libpng

png_structp ptr;

This is a pointer to a context for the png decompressor.

png_infop info;

This is a pointer to a context that supplies meta-data for the png file.

(I honestly don’t know why this needs to be separated, old libraries do weird things like this.)

Just check the code.

I could go over each piece by slides, but it’ll be easier to explain line by line. So check loadFromDisk() in texture.cpp to see how to use libpng.

Setting Up OpenGL for texturing

To tell OpenGL to use textures, we need to first, enable it with:

```
glEnable( GL_TEXTURE_2D );
```

You'll notice we used GL_TEXTURE_2D, more modern you could use GL_TEXTURE_2D_MULTISAMPLE, which takes multiple textures and blend them.

https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/glTexImage2DMultisample.xhtml

Creating the texture

So now that everything is all setup, we need to create the texture. Start by telling OpenGL to reserve place for it in your video memory:

```
glGenTextures( n, &id );
```

n is the number of textures to generat;

Id is the id openGL will assign to your texture.

Creating the texture

Next, we have to tell Open GL to USE the texture.

```
glBindTexture( GL_TEXTURE_2D, id );
```

Creating the texture

And now, we need to setup how the texture renders along the surface of our polygons.

```
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE );
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE );
```

This sets up the GL_TEXTURE_2D renderer's wrap method to clamp, which creates a hard edge for the texture with no repeating. S,T are coordinate variables for the texture. Similar to U,V.

Creating the texture

Next we can finally transfer the data we collected earlier from our png file to video memory.

```
glTexImage2D( GL_TEXTURE_2D, 0, GL_RGBA8, width, height, 0, GL_RGBA,
GL_UNSIGNED_BYTE, (GLubyte *)data );
```

This sends the data through to GL_TEXTURE_2D this looks complicated but is actually pretty straight forward. Check out: https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/glTexImage2D.xhtml
For more about it's parameters.

A little side note.

n^2 textures:

Older video cards required height and width to be n^2 textures. GLEW will expose the ability to do this still, but on these old cards it can be VERY slow.

RGBA:

Older video cards didn’t always support RGBA exclusively, but instead ARGB, or even BGRA. GLEW will swap around the bytes to fit the format the card supports, but also is slow.

Creating the texture

Last, we need to tell OpenGL how to render MipMaps.

```
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR );
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR );
```

Mipmaps are used internally by the video card to scale the image for it’s various angles, including when looking up close or from far away.

Using the texture

Now to use it, all we do is have to call the below before you draw your mesh. Your mesh should have each vertex setup with a matching (u,v) coordinate. You will remember in the previous tutorial, we already set these up.

```
glBindTexture( GL_TEXTURE_2D, id );
```

Now Explore

Take the time to read through the rest of the example and explore what it is doing and how it is drawing the textures. 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 still. But you can also now use ‘c’ to disable the color coding.

Summary

Now you can begin to render a mesh. Try to take this further and make your own 3D environment to explore.