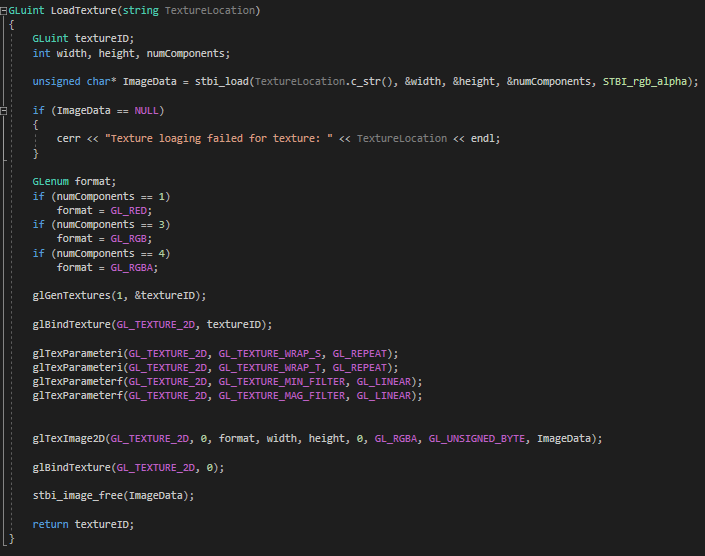
# Normal Mapping

## Loading the map

You’ve seen this before, review the texturing tutorial for more clarity. I’m going to do this quickly so we can move on to the meat of this topic.

Since we are now loading more than one texture, we will need to change LoadTexture to support this. Currently, this function writes to a global variable, textureID.

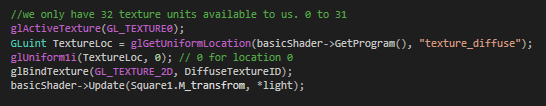
Cut this variable and put it in the loadTexture function, near the top, change the functions definition to return a GLuint and at the bottom of the function, return the textureID.



In the main function, make a new GLuint, DiffuseTextureID, and set it to the output of LoadTexture.



And update the bind function call in the main loop



You program should still run. Test it now.

Now create another GLuint for the normal map texture.

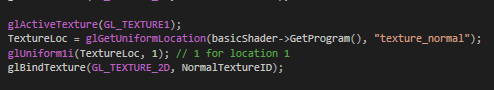


Now Lets Bind it to our shade

Add a new uniform sampler2D to the fragment shader, for our normal map.



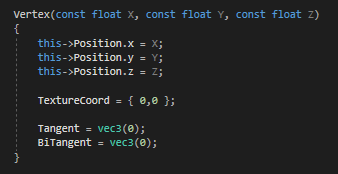
And bind it in the main loop



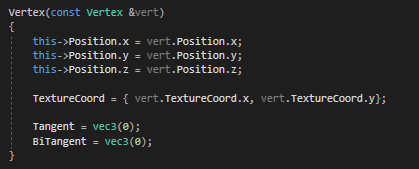
## Calculating tangents

In the vertex.h, add to new vec3, Tangent and Bitangent. In the constructor, set them to 0.





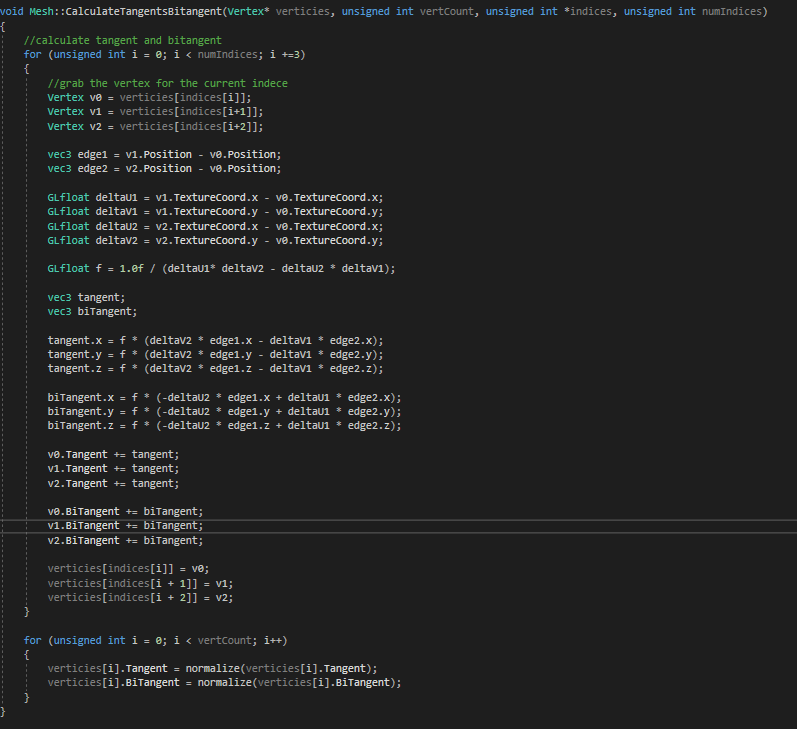
You may also want a copy constructor



Now we need to generate some tangents and bit tangents

Calculating the tangents and bitangents requires some math and quite honestly, I can’t explain it better then OGL dev have so if you want to know how this math works, read their article. http://ogldev.atspace.co.uk/www/tutorial26/tutorial26.html

As for the code, I created a new function called CalculateTangentsBitangents().

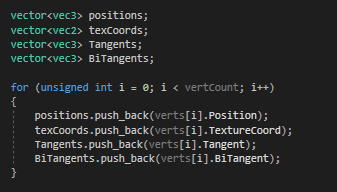


All the new tangent and bitangent data will be stored in the vertices array, since the vertices array is a pointer, we don’t have to return anything,

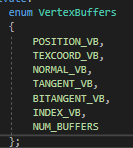
To use it, add the following code to the top of the mesh constructor



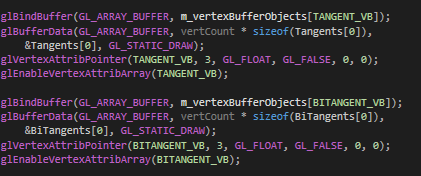
Now we need to get them over to the GPU.



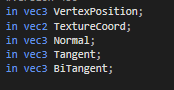
Add TANGENT\_VB and BITANGENT\_VB to the VertexBuffers in mesh.h



And bind the data in the mesh constructor

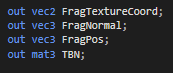


Add two new in vec3’s to the vertex shader for the tangent and bitangent.



## Using the normal map and TBN

In the vertex shader, we need a new out mat3 called TBN. This is our, tangent, bitangent, normal matrix and we will us it to shift all our normal map vectors from world space to tangent space.



Now we need to give it some data to send to the fragment shader. To get this data, we just need to take our tangent, bitangent and normal and multiply them by our model.



Then we use these to set our TBN matrix.



Over in the fragment shader, we need a new in mat3 for the TBN.



Comment out this line:



And replace it with the following:



This code will read in the values stored in the normal map, normalising them for good measure.

Then it will multiply them by 2 and subtract one to transform the normal information from vales of 0 to 1 band in to real normal values of -1 to 1.

And finally, we multiply the normal by the TBN matrix to shift it to tangent space.

That’s it!

Run your program and you should see the following:



