25/02/2019 BLOG – C++ OpenGL

As promised in the last update this one will have more C++ (and OpenGl).

Graphics and Shader:

-OpenGL and C++

-What is it for?

This project is a graphics and shaders module which requires a scene to use C++ and OpenGL to create different looks using programs and shaders.

-What is it made in?

This project is using C++ with OpenGL (4.2) as the rendering pipeline.

I will be demonstrating two examples I this update. The first being with lighting applied to a model and the second using the vertex shader to animate a mesh.

-Challenges?

-Example 1?

-Basic lighting?

The first example is of basic lighting being applied to a model. The scene only has one light. The scene is being rendered with forward rendering. The lighting is being calculated using Phong shading.

The gif below shows the result of applying Phong shading to a fbx model with normal.

.gif

Here is the fragment shader which handles all the lighting.

<code>

#version 150

//Inputs from the vertex shader

in vec4 vPos;

in vec4 vNormal;

in vec4 vColour;

in vec2 vUV;

in vec4 vLightDir;

in vec4 vViewDir;

//Output

out vec4 fragColour;

//Uniforms

uniform vec4 lightColour = vec4(1.0, 0.85, 0.05, 1.0);;

uniform vec4 lightAmbient = vec4(0.4, 0.4, 0.4, 1.0);

uniform vec4 lightSpecular = vec4(1.0, 1.0, 1.0, 1.0);

uniform float specularPower = 1.0;

uniform sampler2D DiffuseTexture;

uniform sampler2D NormalTexture;

uniform int useNormalMap;

void main()

{

//Get the albedo

vec4 albedo = texture(DiffuseTexture, vUV);

//Check if we have a normal map as the tank model does not have a

//normal baked into the file

vec4 normalTex;

if(useNormalMap == 1)

{

//If we have loaded a normal map. Use it

normalTex = texture(NormalTexture, vUV) \* 2.0 - 1.0;

}

else

{

//If we have not loaded a normal map the use the vertex values passed

//though

normalTex = vNormal;

}

//Get the ambient light

vec4 ambient = lightAmbient;

//Get the lambertian term

float NdL = max(0.0, dot(normalTex, -vLightDir));

//Use the light colour as the diffuse colour of the light

vec4 diffuse = lightColour \* NdL;

//Use the built in reflect function to reflect the light direction around the surface normal

vec4 R = reflect(vLightDir, normalTex);

//Get the specular term

float specularTerm = pow(max(0.0, dot(R, vViewDir)), specularPower);

//Get the specular colour

vec4 specular = lightSpecular \* specularTerm;

//Combine all values together

fragColour = vec4(albedo.rgb \* (ambient.rgb + diffuse.rgb + specular.rgb), 1.0);

}

</code>

[Here is an interactable demo on phong shading. Based on a [WebGL applet](http://www.mathematik.uni-marburg.de/~thormae/lectures/graphics1/code/WebGLShaderLightMat/ShaderLightMat.html) by [Prof. Thorsten Thormählen](http://www.uni-marburg.de/fb12/grafikmultimedia-en/team/thormae-en). Modified by [Johannes Kehrer](http://wwwcg.in.tum.de/group/persons/kehrer.html) for educational purpose.] (http://multivis.net/lecture/phong.html)

-Example 2?

-Vertex animation?

The second example I want to show is using the vertex shader to move objects. This to me is very interesting as you can take any object you want within a scene and modify it with great performance. Here is a link to how Pico Tanks use the vertex shader in Unity to move tress when a tank is driving though (<https://www.reddit.com/r/PicoTanks/comments/apbi67/in_pico_tanks_we_push_the_trees_with_a_vertex/>). On to my example. My example shows how to create waves using the vertex shader. This might be standard these days, but it is a great way to show how the vertex shader can be used.

.gif

<code>

//Inputs

//Outputs

//Uniforms

void main()

{

…

//Get the offset which must be applied to the vertex

float yOffset = sin((Time \* 2.5) + offsetOne + offsetTwo \* 0.5);

//Transform the vertex into view space

vec4 viewPos = ProjectionView \* Model \* Position + vec4(0, yOffset, 0, 0);

…

//Get the new position for this vertex in world space

vPosition = Model \* Position + vec4(0, yOffset, 0, 0);

…

}

</code>

The code above is a snippet of the vertex shader.

That’s it for this update. I made sure to leave out any Unity stuff this week and just do C++ as the past two updates have been mainly Unity based. However, the next update (hopefully on time 😊) will contain more networking as that is what I have been getting headaches over.