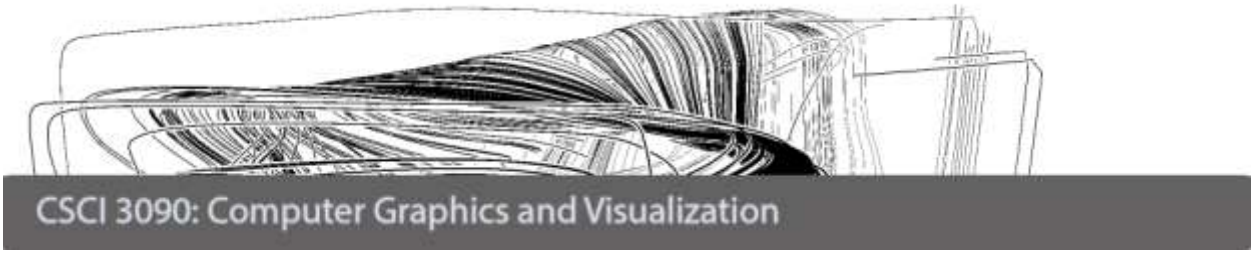

Laboratory Seven: Simple Texture Mapping

Introduction

The purpose of this laboratory is to gain some experience with the use of texture maps in vertex and fragment shaders. This laboratory is based on the example 7 program, so start by downloading this example and make sure that it compiles and runs. In this laboratory you will only be changing the vertex and fragment shaders.

Vertex Shader

We will start with the vertex shader code. The code that we have for the vertex shader is:

```
in vec4 vPosition;
in vec2 vTexture;

uniform mat4 modelView;

out vec2 texCoord;

void main(void) {

	gl_Position = modelView * vPosition;
	texCoord = vTexture;

}
```

We are passing the texture coordinates straight through to the fragment shader without changing them in any way. Try multiplying the value of vTexture by 4 and see how that impacts the resulting image. When we set up the texture we stated that it would repeat in both directions. When we multiply the texture coordinates by 4 we are repeating the texture 4 times in both directions.

Now try using the vertex position as the texture coordinates. The position is a vector of length 4, while the texture coordinates are 2D, so you need to extract two of the coordinates from the position vector. This is called swizzling and it's done by specify the coordinates as if they are fields of the position vector. For example, vPosition.xy gives the x and y coordinates of the

position vector. Try different combinations of the vertex coordinates and observe their impact on the image.

Finally rotate the texture coordinates by fixed angle within the vertex shader. There are cos and sin functions in the shading language, so you can use the standard 2D rotation matrix for this. The argument to both of these functions is in radians. Explain the result that you get.

Fragment Shader

Now we turn to the fragment shader and make some modifications to it. The code for the fragment shader is:

```
in vec2 texCoord;
uniform sampler2D tex;

void main(void) {

    gl_FragColor = texture(tex, texCoord);

}
```

Start by multiplying the texture value by red (the vector (1.0, 0.0, 0.0, 1.0)) and observe what the result is. Which of the squares does this impact? How would you change the colour of the other set of squares? Use this to construct a shader that changes the colours of both sets of squares. The texture that we have constructed is basically a template for a range of checkerboard like effects.

Bonus Activity (optional)

Add lighting to these boards.

Laboratory Report

Submit screenshots of your results along with an explanation of how the different effects were produced in the Blackboard dropbox.