Ciaran O'Hagan

CS4031 Computer Graphics

Lab II Report

1. Keyboard Control

To enable the use of keyboard actions, I simply created a function called `keyPressed()` and used the `glutKeyboradFunc()` function in the main; this essentially tells the glut process that if it detects any keyboard input, it should execute the function which has been passed to it:

```
void keyPressed(unsigned char key, int x, int y)
{
	switch (key) { ... }
}

glutKeyboardFunc(keyPressed);
```

The `keyPressed()` function contains a large switch statement which reads in a certain key being pressed. This is used to display certain effects on screen when a button is pressed.

2. Transformations

2.1. Setup

In setting up my transformations, I declared a global identity matrix mat, which is the matrix used as a platform for the calculations to perform the transformations. Within the vertex shader, I declared a new uniform variable, which would correspond to the previous matrix. The uniform matrix would be multiplied by `gl_Position` within the shader, and when redrawn, results in the desired transformation being visible on screen.

```
uniform mat4 mat;
void main(){
	gl_Position = mat * vec4(vPosition, 2.0);
}
```

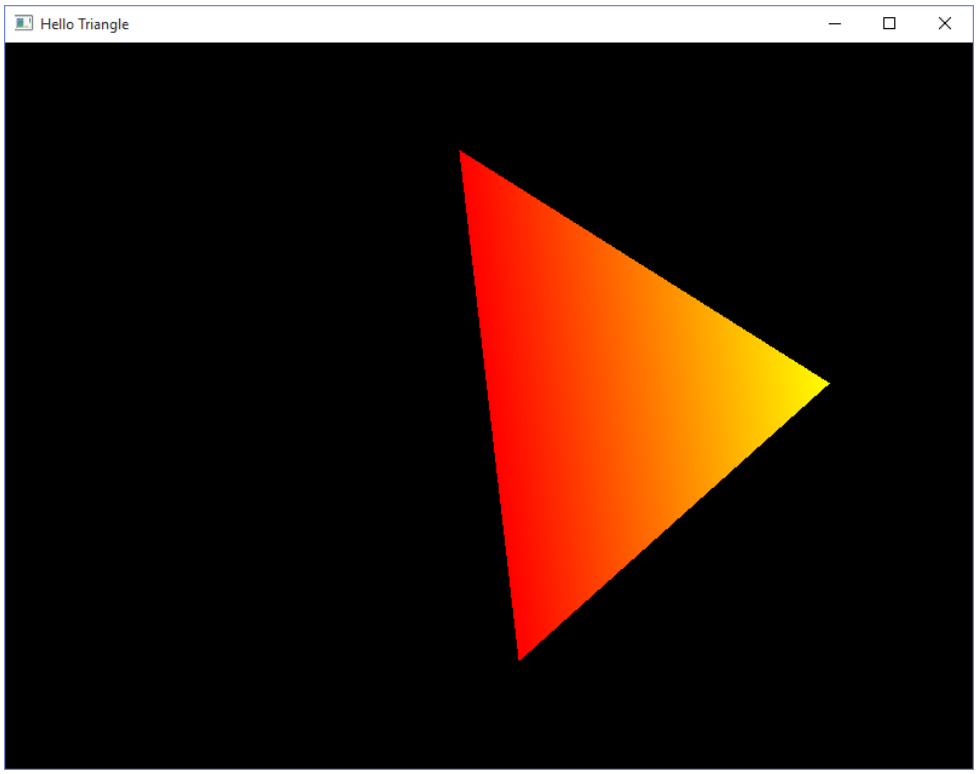
The following code takes the resulting transformed identity matrix and places it into the uniform variable located in the vertex shader:

```
GLint matrix_location = glGetUniformLocation(shaderProgramID, "mat");
glUniformMatrix4fv(matrix_location, 1, GL_FALSE, alterMat.m);
```

For the following sections, I will just paste the relevant code snippets and a screenshot as it's relatively straightforward.

2.2. Rotation around the x-, y- & z-axis

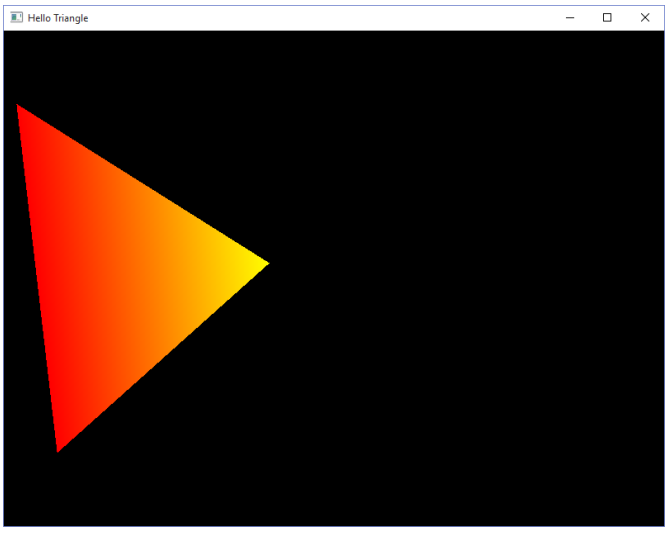
```
//Rotation
case('q') : //x
{
	alterMat = rotate_x_deg(alterMat, 10.0f);
	break;
}
case('e') ://y
{
	alterMat = rotate_y_deg(alterMat, 10.0f);
	break;
}
case('r') : //z
{
	alterMat = rotate_z_deg(alterMat, 10.0f);
	break;
}
```



2.3. Translation in the x-, y & z direction

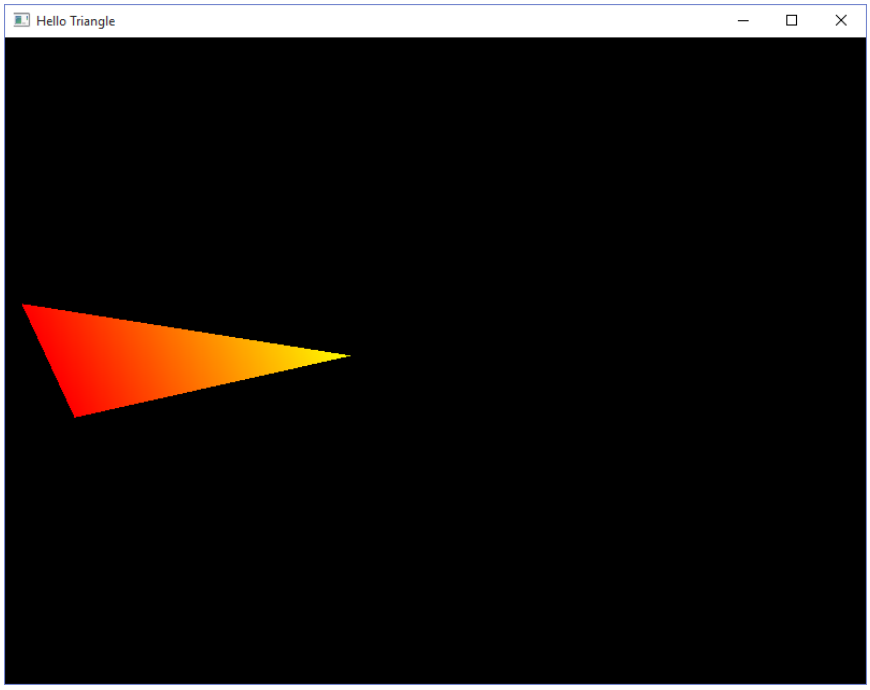
```
//Translation
case('w') : //+y
{
	vec3 v = { 0.0f, 0.1f, 0.0f };
	alterMat = translate(alterMat, v);
	break;
}
case('s') : //-y
{
	vec3 v = { 0.0f, -0.1f, 0.0f };
	alterMat = translate(alterMat, v);
	break;
}
case('a') : //-x
{
	vec3 v = { -0.1f, 0.0f, 0.0f };
	alterMat = translate(alterMat, v);
	break;
}
case('d') : //+x
{
	vec3 v = { 0.1f, 0.0f, 0.0f };
	alterMat = translate(alterMat, v);
	break;
}
case('z') : //-z
{
	vec3 v = { 0.0f, 0.0f, 0.1f };
	alterMat = translate(alterMat, v);
	break;
}
case('x') : //+z
{
	vec3 v = { 0.0f, 0.0f, -0.1f };
	alterMat = translate(alterMat, v);
	break;

}
```



2.4. Uniform & Non Uniform Scaling

```
//Uniform Scaling
case('f') : //Bigger
{
	vec3 v = { 1.5f, 1.5f, 1.5f };
	alterMat = scale(alterMat, v);
	break;
}
case('g') : //Smaller
{
	vec3 v = { 0.5f, 0.5f, 0.5f };
	alterMat = scale(alterMat, v);
	break;
}
//Non Uniform Scaling
case('t') :  //x
{
	vec3 v = { 1.0f, 0.5f, 0.5f };
	alterMat = scale(alterMat, v);
	break;
}
case('y') :  //y
{
	vec3 v = { 0.5f, 1.0f, 0.5f };
	alterMat = scale(alterMat, v);
	break;
}
case('u') : //z
{
	vec3 v = { 0.5f, 0.5f, 1.0f };
	alterMat = scale(alterMat, v);
	break;
}
```



2.5. Combined Transformations

```
case('.') : //Combined Transformaton - Translation + Rotation
{
	vec3 v = { 0.1f, 0.0f, 0.0f };
	alterMat = translate(alterMat, v);
	alterMat = rotate_x_deg(alterMat, 10.0f);
	break;
}
case(',') : //Combined Transformaton - Translation + Rotation
{
	vec3 v = { 0.1f, 0.0f, 0.0f };
	alterMat = translate(alterMat, v);
	alterMat = rotate_x_deg(alterMat, 10.0f);
	break;
}
```

3. Multiple Triangles

To draw multiple triangles using the same buffer, I added a new transformation matrix, pointed it towards the same uniform matrix within the vertex shader and simply redrew the triangle again. Calculations performed on the first triangle have no effect on the second one.

```
GLint matrix_location = glGetUniformLocation(shaderProgramID, "mat");
glUniformMatrix4fv(matrix_location, 1, GL_FALSE, alterMat.m);
glDrawArrays(GL_TRIANGLES, 0, 3);

GLint matrix_location2 = glGetUniformLocation(shaderProgramID, "mat");
glUniformMatrix4fv(matrix_location2, 1, GL_FALSE, alterMatNew.m);
glDrawArrays(GL_TRIANGLES, 0, 3);
```