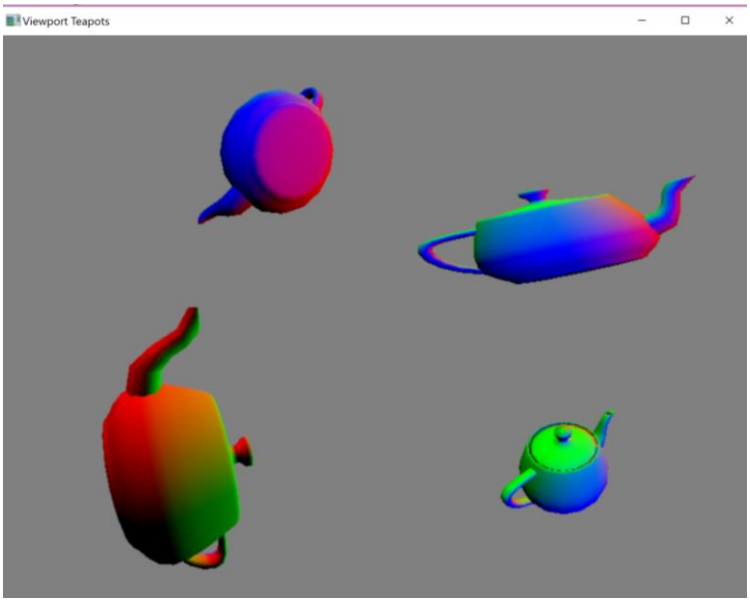
Lab 3 Viewing

Eva Nolan
14335043



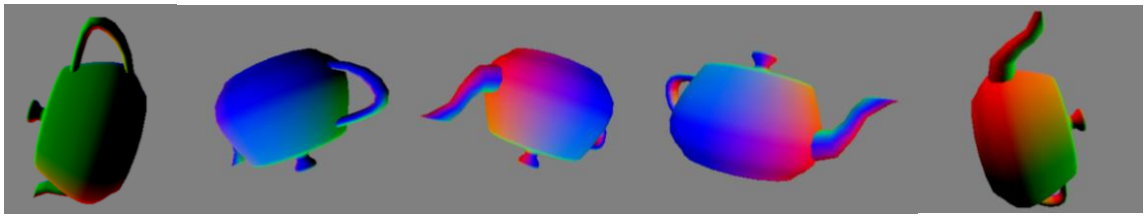
Perspective Projections

The bottom left and bottom right and top right viewports both help perspective projections. These were achieved using the perspective projection function in maths_funcs.cpp

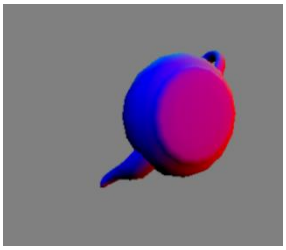
I created one teapot which appears to be moving. I achieved this by changing the angle at which the model was projected with respect to time using the getGetTime() function. The model appears to rotate about the x, y and z axis.

```
// bottom-left perspective projection and rotating
mat4 view = translate(identity_mat4(), vec3(0.0, 0.0, -40.0)); //sets camera at -40 on z axis
mat4 persp_proj = perspective(45.0, (float)width / (float)height, 0.1, 100.0); // persepective projection
mat4 model = rotate_z_deg(identity_mat4(), (timeGetTime() / 20)); // rotates object by 45 degrees
model=model*rotate_y_deg(identity_mat4(), (timeGetTime()/20));
model=model*rotate_y_deg(identity_mat4(), (timeGetTime()/20));
glViewport(0, 0, width / 2, height / 2);

glUniformMatrix4fv(proj_mat_location, 1, GL_FALSE, persp_proj.m);
glUniformMatrix4fv(view_mat_location, 1, GL_FALSE, view.m);
glUniformMatrix4fv(matrix_location, 1, GL_FALSE, model.m);
glDrawArrays(GL_TRIANGLES, 0, teapot_vertex_count);
```



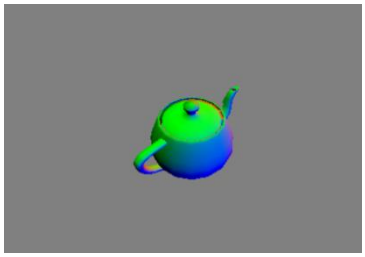
The second perspective projection I achieved was static. It also used the provided look_at function. I also chose to rotate the model about the x and y axis to give an image of the model from a different side and make the four viewports together more interesting to look at.



```
// top-left prespective and look at
//                      camera pos              target pos                up
mat4 view2 = look_at(vec3(0.0, -20.0, -60.0), vec3(10.0, 20.0, 10.0), vec3(1.0, 60.0, 90.0));
mat4 persp_proj2 = perspective(45.0, (float)width / (float)height, 0.1, 100.0); // persepective projection
mat4 model2 = rotate_y_deg(identity_mat4(), 20); // rotates object by 45 degrees
model2 = model2*rotate_x_deg(identity_mat4(), 45); // rotates object by 45 degrees

glViewport(0, height / 2, width / 2, height / 2);
glUniformMatrix4fv(proj_mat_location, 1, GL_FALSE, persp_proj2.m);
glUniformMatrix4fv(view_mat_location, 1, GL_FALSE, view2.m);
glUniformMatrix4fv(matrix_location, 1, GL_FALSE, model2.m);
glDrawArrays(GL_TRIANGLES, 0, teapot_vertex_count);
```

The third presepective projection was was again rotated to show another angle of the teapot. I also changed the fovy so the teapot appears father away in the scene.

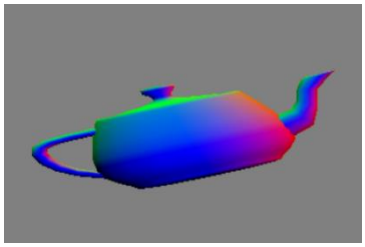


```
//bottom right persepctive projection and some rotation
mat4 view1 = translate(identity_mat4(), vec3(0.0, 0.0, -45.0)); //sets camera at -40 on z axis
mat4 persp_proj1 = perspective(70.0, (float)width/ (float)height, 0.1,50.0); // persepective projection
mat4 model1 = rotate_x_deg(identity_mat4(), 45); // rotates object by 45 degrees
model1 = model1*rotate_y_deg(identity_mat4(), 45);

glViewport(width / 2, 0, width / 2, height / 2);
glUniformMatrix4fv(proj_mat_location, 1, GL_FALSE, persp_proj1.m);
glUniformMatrix4fv(view_mat_location, 1, GL_FALSE, view1.m);
glUniformMatrix4fv(matrix_location, 1, GL_FALSE, model1.m);
glDrawArrays(GL_TRIANGLES, 0, teapot_vertex_count);
```

Orthographic Projection

The orthographic projection was the most difficult to achieve as it was necessary to create a projection matrix. I wrote an orthographic projection function based on the projection functions in maths_funcs.cpp and orthographic projection algebra. This function was then called to create the projection matrix.



```
mat4 Ortho(float l, float r, float b, float t, float n, float f) {
    mat4 m= zero_mat4();
    m.m[0] = 2 / (r - l);
    m.m[5] = 2 / (t - b);
    m.m[10] = -2 / (f - n);
    m.m[12] = -(l + r) / (r - l);
    m.m[13] = -(t + b) / (t - b);
    m.m[14] = -(f + n) / (f - n);
    m.m[15] = 1;

    return m;
}
```

```
// top-right orthographic
mat4 view4 = translate(identity_mat4(), vec3(40.0, 40.0, -40.0));
mat4 persp_proj4 = Ortho(float(height / 30), float(height/10), float(width / 30), float(width/10), 0.1f, 100.0f);
mat4 model4 = rotate_z_deg(identity_mat4(), 20);

glViewport(width / 2, height / 2, width / 2, height / 2);
glUniformMatrix4fv(proj_mat_location, 1, GL_FALSE, persp_proj4.m);
glUniformMatrix4fv(view_mat_location, 1, GL_FALSE, view4.m);
glUniformMatrix4fv(matrix_location, 1, GL_FALSE, model4.m);
glDrawArrays(GL_TRIANGLES, 0, teapot_vertex_count);
```