Rotating Teapot

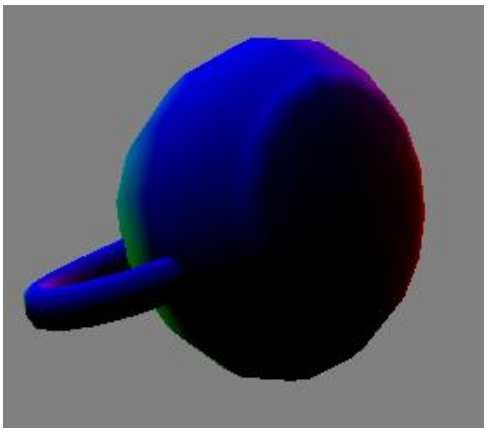
By multiplying the rotation matrices in the x, y and z directions and increasing the angle of rotation on each scene update, the teapot appears to be moving.

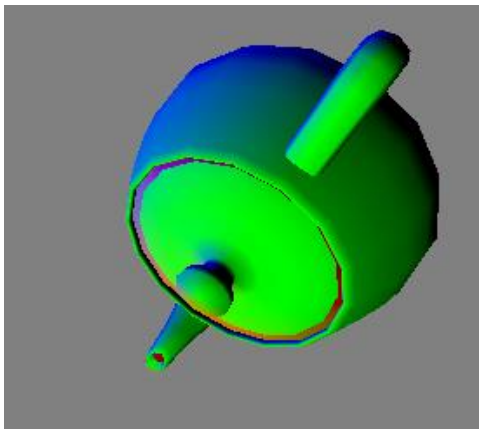
```
// bottom-left
mat4 view1 = translate (identity_mat4 (), vec3 (0.0, 0.0, -40.0));
mat4 persp_proj1 = perspective(45.0, (float)width/(float)height, 0.1, 100.0);
mat4 model1y = rotate_x_deg(identity_mat4(), angle / 5);
mat4 model1x = rotate_y_deg(identity_mat4(), angle / 6.56);
mat4 model1z = rotate_z_deg (identity_mat4 (), angle / 4);
mat4 model1 = model1x*model1y*model1z;
angle = angle + 0.1;

glViewport (0, 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

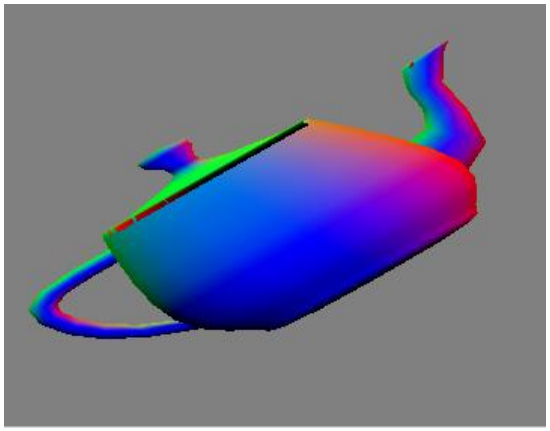
Using the orthographic projection matrix described in the notes, a function was created in Anton's Maths which allowed for an orthographic projection to be depicted in the viewport.

```
// bottom-right
mat4 view2 = translate(identity_mat4(), vec3(0.0, 0.0, -40.0));
mat4 model2 = rotate_z_deg(identity_mat4(), 45);
mat4 persp_proj2 = orthogonal(15.0f, -15.0f, 20.0f, -20.0f, 0.1f, -100.0f);

glViewport(300, 0, 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);
```



Perspective Projection

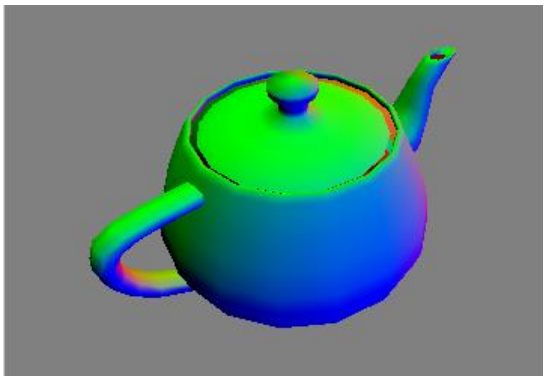
Using the perspective projection matrix function supplied in Anton's Maths, the teapot was rotated 40 degrees in the x and y direction in order to better display perspective in the viewport. The slowly moving teapot can also be seen to be using a perspective projection.

```
// top-left
mat4 view3 = translate(identity_mat4(), vec3(0.0, 0.0, -40.0));
mat4 persp_proj3 = perspective(45.0, (float)width / (float)height, 0.1, 100.0);
mat4 model3x = rotate_x_deg(identity_mat4(), 40);
mat4 model3y = rotate_y_deg(identity_mat4(), 40);
mat4 model3 = model3x*model3y;

glViewport(0, 300, width / 2, height / 2);

glUniformMatrix4fv(proj_mat_location, 1, GL_FALSE, persp_proj3.m);
glUniformMatrix4fv(view_mat_location, 1, GL_FALSE, view3.m);
glUniformMatrix4fv(matrix_location, 1, GL_FALSE, model3.m);

glDrawArrays(GL_TRIANGLES, 0, teapot_vertex_count);
```



Look-at Function

This viewport makes use of the look_at function in Anton's Maths in the view matrix, the function parameters contain three 3D vectors.

```
// top-right
vec3 camera = vec3(44.0, 30.0, -5.0);
vec3 target = vec3(0.0, 0.0, 0.0);
vec3 position  = vec3(1.0, 1.0, 0.0);

mat4 view4 = look_at(camera, target, position);
mat4 persp_proj4 = perspective(45.0, (float)width / (float)height, 0.1, 100.0);
mat4 model4 = rotate_y_deg(identity_mat4(), 90);

glViewport(300, 300, 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);
```