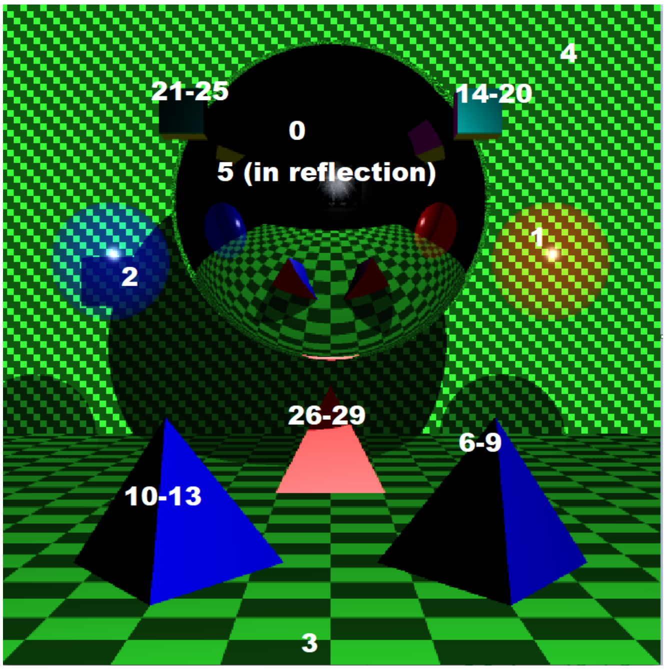
COSC363: Computer Graphics

Assignment 2

David Turton 81535137

Ray Tracer Description:

My ray-tracer was based off of the ray-tracer developed in labs 7 and 8. My ray-tracer features generated patterns, transparent objects, a reflective sphere, and shapes consisting of planes. In my scene there are two cubes, each consisting of 6 planes, three tetrahedrons consisting of 4 planes each, two transparent spheres and one reflective sphere. The image below indicates each objects position in the SceneObjects array.



Basic Features:

Lighting:

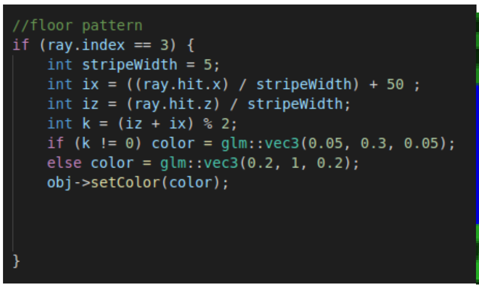
The scene has one lighting source located at (7, 20, -10). Objects have different colours depending on the lighting, cube 1 is lighter than cube 2 as it's closer to the light source.

Shadows:

Objects cast shadows based on the light source, transparent objects (1 and 2) cast lighter shadows than solid objects (i.e. object 0).

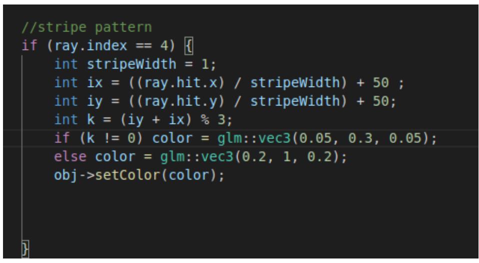
Chequered Pattern:

I generated the chequered pattern on the floor using the code shown below, based off of the code used for making stripes in lab 8. This code algorithmically generates a pattern based on when the z and x axis overlap in a repeating function.

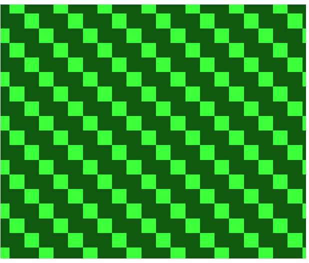


```
//floor pattern
if (ray.index == 3) {
    int stripeWidth = 5;
    int ix = ((ray.hit.x) / stripeWidth) + 50 ;
    int iz = (ray.hit.z) / stripeWidth;
    int k = (iz + ix) % 2;
    if (k != 0) color = glm::vec3(0.05, 0.3, 0.05);
    else color = glm::vec3(0.2, 1, 0.2);
    obj->setColor(color);



}
```

I used a modified version of this to create the diagonal stripes in the background.

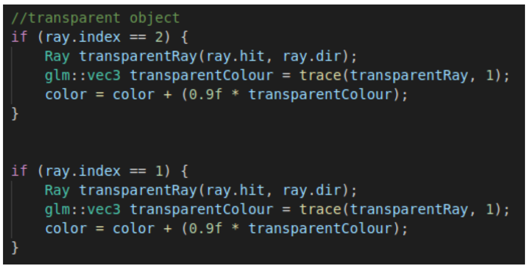


```
//stripe pattern
if (ray.index == 4) {
    int stripeWidth = 1;
    int ix = ((ray.hit.x) / stripeWidth) + 50 ;
    int iy = ((ray.hit.y) / stripeWidth) + 50;
    int k = (iy + ix) % 3;
    if (k != 0) color = glm::vec3(0.05, 0.3, 0.05);
    else color = glm::vec3(0.2, 1, 0.2);
    obj->setColor(color);



}
```



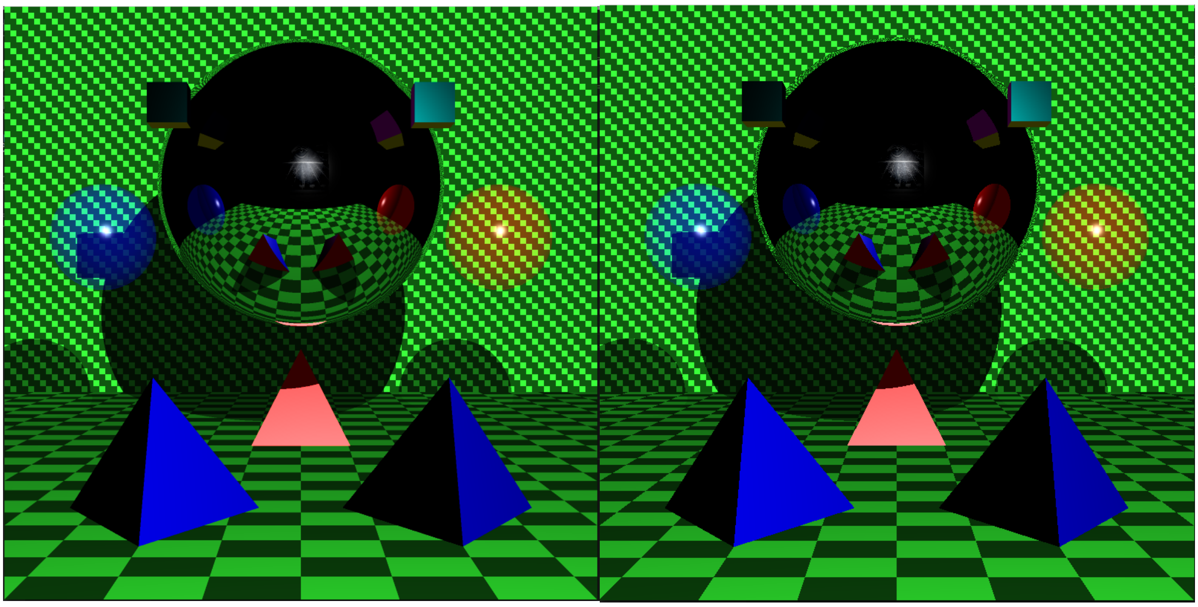
Transparency:

Objects 1 and 2 are both transparent spheres that cast lighter shadows. I made them transparent using the code below.



```
//transparent object
if (ray.index == 2) {
    Ray transparentRay(ray.hit, ray.dir);
    glm::vec3 transparentColour = trace(transparentRay, 1);
    color = color + (0.9f * transparentColour);
}


if (ray.index == 1) {
    Ray transparentRay(ray.hit, ray.dir);
    glm::vec3 transparentColour = trace(transparentRay, 1);
    color = color + (0.9f * transparentColour);
}
```

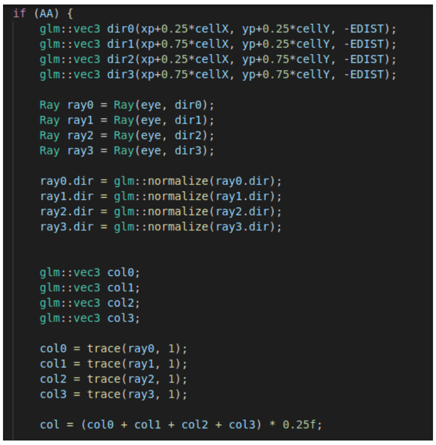
Anti-Aliasing:



(left: AA On, right AA off)

The scene uses anti-aliasing to smoothen out edges, such as the edges of the triangles, shadows, and the places in which the chequered tiles meet. I increased the number of number of divisions in the assignment from 500 to 800 to make things smoother. The version with anti-aliasing took four minutes to render, whereas the regular version took less than a minute. The texture reflected in object 0 is also much clearer to see in the anti-aliased version. The reflection of the chequered pattern is also far more visible in the anti-aliased version.

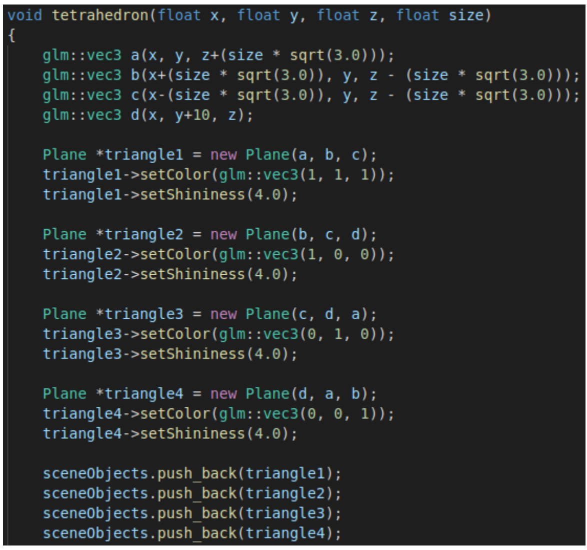
To generate the anti-aliased version I divided the cells in the scene into four and generated each primary ray respectively as seen in the code below.



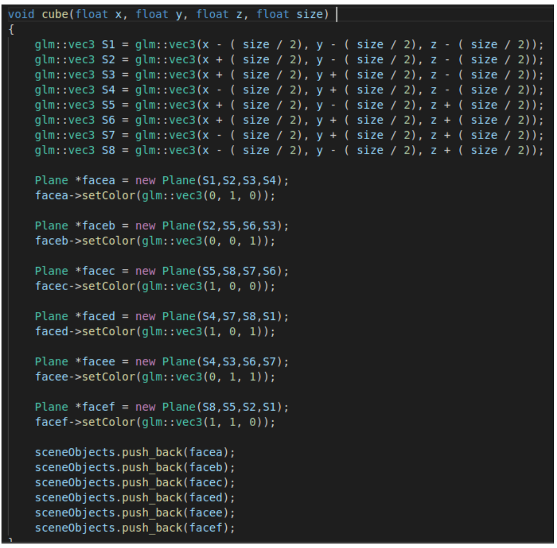
```
if (AA) {
    glm::vec3 dir0(xp+0.25*cellX, yp+0.25*cellY, -EDIST);
    glm::vec3 dir1(xp+0.75*cellX, yp+0.25*cellY, -EDIST);
    glm::vec3 dir2(xp+0.25*cellX, yp+0.75*cellY, -EDIST);
    glm::vec3 dir3(xp+0.75*cellX, yp+0.75*cellY, -EDIST);

    Ray ray0 = Ray(eye, dir0);
    Ray ray1 = Ray(eye, dir1);
    Ray ray2 = Ray(eye, dir2);
    Ray ray3 = Ray(eye, dir3);

    ray0.dir = glm::normalize(ray0.dir);
    ray1.dir = glm::normalize(ray1.dir);
    ray2.dir = glm::normalize(ray2.dir);
    ray3.dir = glm::normalize(ray3.dir);


    glm::vec3 col0;
    glm::vec3 col1;
    glm::vec3 col2;
    glm::vec3 col3;

    col0 = trace(ray0, 1);
    col1 = trace(ray1, 1);
    col2 = trace(ray2, 1);
    col3 = trace(ray3, 1);

    col = (col0 + col1 + col2 + col3) * 0.25f;
```

Shapes:

I created the tetrahedrons and cubes out of flat planes as seen in the code below.



```
void tetrahedron(float x, float y, float z, float size)
{
    glm::vec3 a(x, y, z+(size * sqrt(3.0)));
    glm::vec3 b(x+(size * sqrt(3.0)), y, z - (size * sqrt(3.0)));
    glm::vec3 c(x-(size * sqrt(3.0)), y, z - (size * sqrt(3.0)));
    glm::vec3 d(x, y+10, z);

    Plane *triangle1 = new Plane(a, b, c);
    triangle1->setColor(glm::vec3(1, 1, 1));
    triangle1->setShininess(4.0);

    Plane *triangle2 = new Plane(b, c, d);
    triangle2->setColor(glm::vec3(1, 0, 0));
    triangle2->setShininess(4.0);

    Plane *triangle3 = new Plane(c, d, a);
    triangle3->setColor(glm::vec3(0, 1, 0));
    triangle3->setShininess(4.0);

    Plane *triangle4 = new Plane(d, a, b);
    triangle4->setColor(glm::vec3(0, 0, 1));
    triangle4->setShininess(4.0);

    sceneObjects.push_back(triangle1);
    sceneObjects.push_back(triangle2);
    sceneObjects.push_back(triangle3);
    sceneObjects.push_back(triangle4);
```



```
void cube(float x, float y, float z, float size)
{
    glm::vec3 S1 = glm::vec3(x - ( size / 2), y - ( size / 2), z - ( size / 2));
    glm::vec3 S2 = glm::vec3(x + ( size / 2), y - ( size / 2), z - ( size / 2));
    glm::vec3 S3 = glm::vec3(x + ( size / 2), y + ( size / 2), z - ( size / 2));
    glm::vec3 S4 = glm::vec3(x - ( size / 2), y + ( size / 2), z - ( size / 2));
    glm::vec3 S5 = glm::vec3(x + ( size / 2), y - ( size / 2), z + ( size / 2));
    glm::vec3 S6 = glm::vec3(x + ( size / 2), y + ( size / 2), z + ( size / 2));
    glm::vec3 S7 = glm::vec3(x - ( size / 2), y + ( size / 2), z + ( size / 2));
    glm::vec3 S8 = glm::vec3(x - ( size / 2), y - ( size / 2), z + ( size / 2));

    Plane *facea = new Plane(S1,S2,S3,S4);
    facea->setColor(glm::vec3(0, 1, 0));

    Plane *faceb = new Plane(S2,S5,S6,S3);
    faceb->setColor(glm::vec3(0, 0, 1));

    Plane *facec = new Plane(S5,S8,S7,S6);
    facec->setColor(glm::vec3(1, 0, 0));

    Plane *faced = new Plane(S4,S7,S8,S1);
    faced->setColor(glm::vec3(1, 0, 1));

    Plane *facee = new Plane(S4,S3,S6,S7);
    facee->setColor(glm::vec3(0, 1, 1));

    Plane *facef = new Plane(S8,S5,S2,S1);
    facef->setColor(glm::vec3(1, 1, 0));

    sceneObjects.push_back(facea);
    sceneObjects.push_back(faceb);
    sceneObjects.push_back(facec);
    sceneObjects.push_back(faced);
    sceneObjects.push_back(facee);
    sceneObjects.push_back(facef);
}
```

Build Instructions:

Run the folder on a computer with OpenGL capability (I used the VM setup from the learn page)

Open the assignment 2 files folder in an ide (I used VisualStudioCode) and run the file raytracer.cpp.

Resources:

Eye texture in reflection taken from textures.com.

Lecture slides, Labs 7 and 8.