COSC363: Computer Graphics
Assignment 2

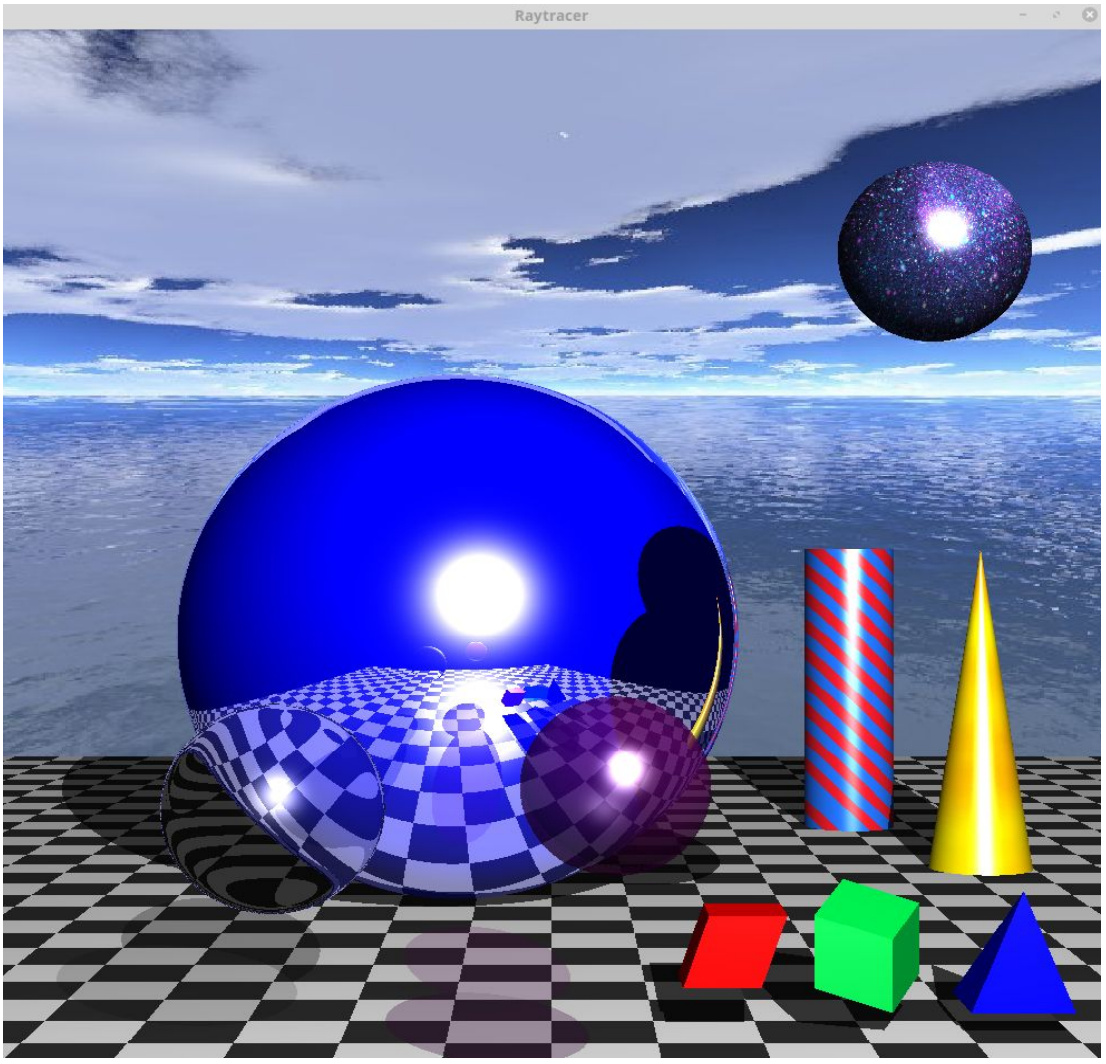
Ray Tracer

Student Name: Phuong Nam Vu
Student Number: 54781288

Build detail: Run Qt Creator and Open Project then Select 'CMakeLists.txt' (Make sure your run and build directory is correct). Run RayTracer.out

I/ Brief Description

The program uses ray tracer method to generate a picture of three spheres and a cylinder, a cone, two cubes and a pyramid.



II/ Extra Features

1. **Objects:**

Cylinder:

The cylinder is generated by Cylinder class which calculates the point of intersection and surface normal vector between the cylinder and traced ray.

The point of intersection is calculated by intersection equation given in lecture note 9-37:
$t^2(d_x^2 + d_z^2) + 2t\{d_x(x_0 - x_c) + d_z(z_0 - z_c)\} + \{(x_0 - x_c)^2 + (z_0 - z_c)^2 - R^2\} = 0$
The point of intersection will be discarded if $y - coordinate$ (*float y-coordinate = posn.y + dir.y * t;*) greater than $y_c + height$.

The unit normal vector is generated by following equation given in lecture note 9-37:

```
glm::vec3 d = p - center;
glm::vec3 n = glm::vec3 (d.x,0,d.z);
n = glm::normalize(n);
```

Cone:

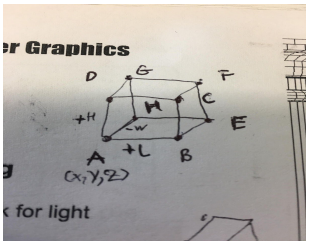
The Cone class also do the same as Cylinder class. The Cone class has different intersection equation which is:
$t^2(d_x^2 + d_z^2 - (\frac{R}{H})^2 * d_y^2) + 2t\{d_x(x_0 - x_c) + d_z(z_0 - z_c) + (\frac{R}{H})^2 * (h - y_0 + y_c) * d_y\} +$
$\{(x_0 - x_c)^2 + (z_0 - z_c)^2 - (\frac{R}{H})^2 * (h - y_0 + y_c)^2\} = 0$

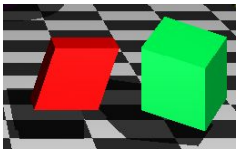
The unit normal vector is generated by following equation given in lecture note 9-43:

```
glm::vec3 d = p - center;
float r = sqrt(d.x * d.x + d.z * d.z);
glm::vec3 n = glm::vec3 (d.x,r*(radius/height),d.z);
n = glm::normalize(n);
```

Cube:

The cube is constructed by using five planes which follow picture beside

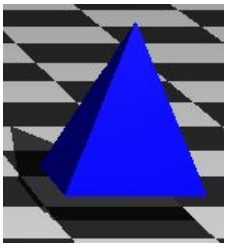




Pyramid:

The pyramid is also constructed by using four planes. The plane is created by 4 points but the 4th point is duplicated of the 3th point. So it makes a triangle plane.

The base is a rectangle plane.

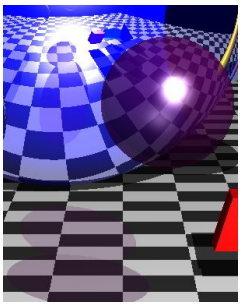


2. Transparent object:

The dark pink sphere is transparent object. This is achieved by creating a new transparent ray(g) from coming ray(d) with the same direction as d. Then trace (g) to get transparent colour.

```
Ray stranp(ray.xpt, ray.dir);
glm::vec3 stranCol = trace(stranp, 1);
colorSum = colorSum + (0.5f*stranCol);
```

The sphere is 50 percentage transparent.
The shadow of the sphere is pink and lighter because I reduce the material color and add the color of the sphere to the shadow



3. Refraction light:

The refractive sphere beside is created by following procedure(lecture note 9-21):
First calculate a refraction direction vector(g) from incoming ray(d) with the eta of the sphere:

```
glm::vec3 g = glm::refract(ray.dir, normalVector, 1.0f/eta);
```

Then create a new ray that is in the sphere with the direction as above and normalize it.

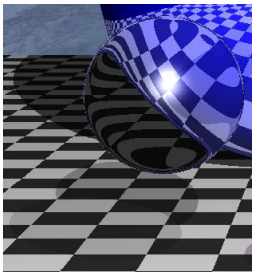
```
Ray refrRay1(ray.xpt, g);
refrRay1.closestPt(sceneObjects);
glm::vec3 m = sceneObjects[refrRay1.xindex]->normal(refrRay1.xpt);
```

From the new ray, we will calculate a direction vector and create new ray go out of the sphere by:

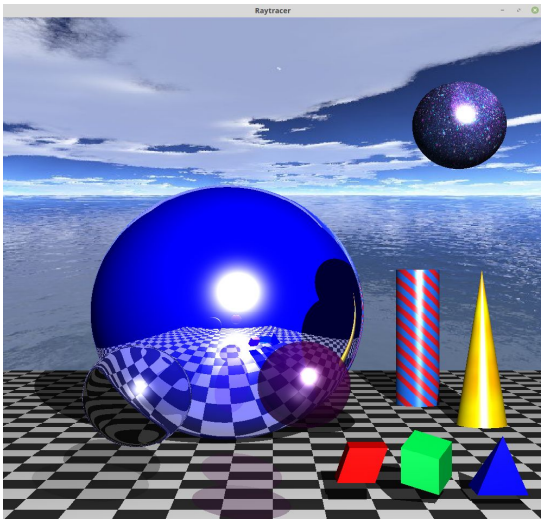
```
glm::vec3 h = glm::refract(g, -m, eta);
Ray refracRay2(refrRay1.xpt,h);
```

Then trace the new ray to get refraction colour.

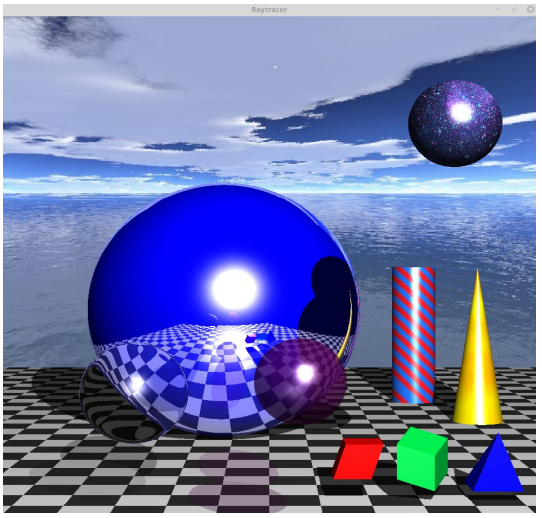
The shadow of the refraction sphere is lighter because of reducing material color on the shadow



4. Anti-aliasing:



Without-anti-aliasing



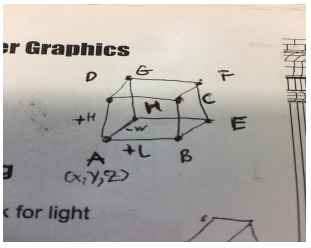
With-anti-aliasing

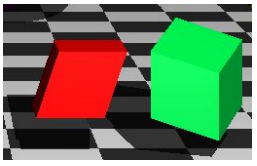
Anti-aliasing is implemented by using supersampling method. The method generates four rays through each square pixel and calculate the average color values:

5. Cube rotation and shear transformation:

The red shear cube is created by adding one more length to the top plane points of the cube which is D, C, F, E. It makes the cube shear.

The shear red cube is a rotated cube achieved by moving the edge CB down by 1 length, moving the edge GH up by 1 length and pushing back plane(G, F, E, H) to the right with the length of 1.5.

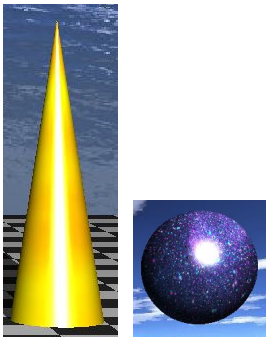




6. Texture object:

The cone and sphere is textured by map the coordinates of the closest point of intersection to image coordinates and assign the colour to the pixel to that point.

```
glm::vec3 n = normalize(normalVector);
float s = n.x /(2 * M_PI) + 0.5;
float t = n.y /(2 * M_PI) + 0.5;
materialCol = texture2.getColorAt(s,t);
```



7. Procedural pattern:

The cylinder and floor are textured by following procedural pattern:
If the sum of y and x coordinates of the closest point of intersection is even then the color of that point will be red. Otherwise the color will be blue.

```
if ((int(ray.xpt.y + ray.xpt.x) % 2 == 0)){
    materialCol = glm::vec3(0.133, 0.431, 0.945);
}else{
    materialCol = glm::vec3(0.894, 0.105, 0.156);
}
```

8. Multiple light sources

There are two light sources on my program. The first light source is at location of (20, 40, -3). The other light source is at (30, 60, 50). The two light source generate two different shadow for each object. Because two light sources are quite close in x-coordinate and y-coordinate so it is not obvious.

III/ Success and failure

- In the big green sphere, There is a blur on the curve of the sphere, because there is a reflection of the background.
- Background images also quite blur.
- The cylinder does not have lift on the top and the bottom.

VI/ References

COSC 363 labs and lecture notes(especially lecture note 9)