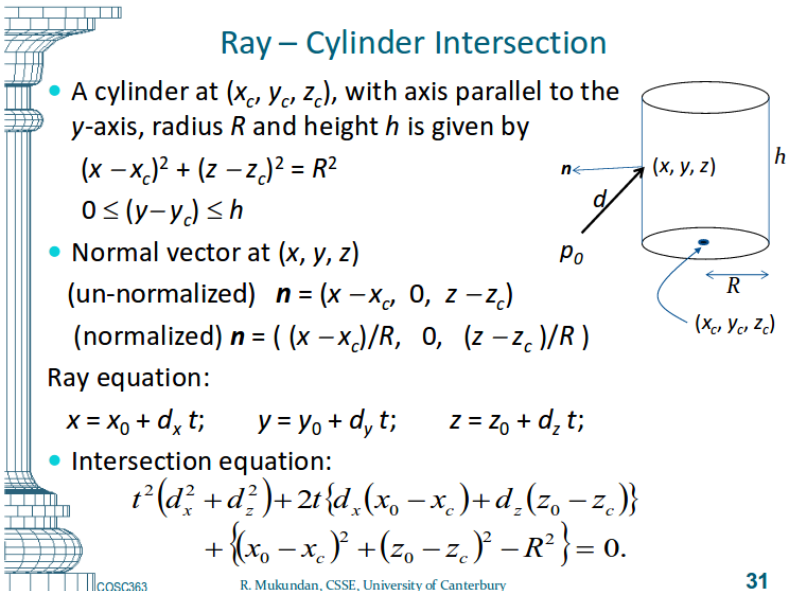
COSC363 Computer Graphics
Assignment 2: Ray Tracer

Student Name: Heyang Li
Student Number: 46522491

To run assignment from geany will take around 15 seconds. In this assignment, there are 9 objects has been created and several lighting effect is been demonstrated. I have found out it is hard to draw a simple object using ray tracing, not mention about complicated object. I have tried to write some code to generate a complicated object but failed on this. But I do managed to finish all basic requirements and some of the extension requirements for this assignment. This assignment is not fun as the first one, but I do feel that I has learnt some advanced computer graphic skill from the knowledge side.

Cylinder:



Ray – Cylinder Intersection

- A cylinder at (x_c, y_c, z_c), with axis parallel to the y-axis, radius R and height h is given by

  $(x - x_c)^2 + (z - z_c)^2 = R^2$

  $0 \le (y - y_c) \le h$

- Normal vector at (x, y, z)

  (un-normalized) $\mathbf{n} = (x - x_c,\ 0,\ z - z_c)$

  (normalized) $\mathbf{n} = (\ (x - x_c)/R,\ \ 0,\ \ (z - z_c)/R\ )$

Ray equation:

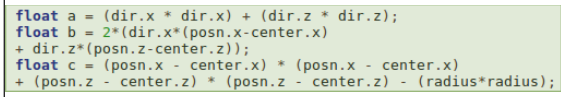
$x = x_0 + d_x t; \qquad y = y_0 + d_y t; \qquad z = z_0 + d_z t;$

- Intersection equation:

$$t^2(d_x^2 + d_z^2) + 2t\{d_x(x_0 - x_c) + d_z(z_0 - z_c)\} + \{(x_0 - x_c)^2 + (z_0 - z_c)^2 - R^2\} = 0.$$

COSC363 R. Mukundan, CSSE, University of Canterbury 31

I have used Prof. Mukundan's lecture slice as a reference and fingered out how to draw a cylinder using ray tracing.
As we can see from the image below, it shows a, b, c is the intersection equation,



```
float a = (dir.x * dir.x) + (dir.z * dir.z);
float b = 2*(dir.x*(posn.x-center.x)
+ dir.z*(posn.z-center.z));
float c = (posn.x - center.x) * (posn.x - center.x)
+ (posn.z - center.z) * (posn.z - center.z) - (radius*radius);
```

and un-normalized n which is glm::vec3 n = glm::vec3 (p.x-center.x,0,p.z-center.z);, and then we just need to normalize n, which use the build-in function glm::normalize(n) to implement it.

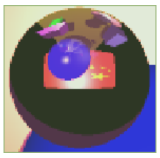
Box and orthogonal:

To draw a box is quite simple, I have drew 6 planars and make sure all the vertexes is in the correct positions. Then push all the planars into the scene.

Orthogonal is a very similar to box, however, to create a orthogonal, I have to more careful about the vertexes in the right positions. I have tried a few times to put them into the right position.

Sphere:
I have used our lab7 and lab8 code as a reference to create 4 spheres and to show most of the lighting effect on the spheres.

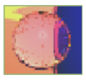
Refraction effect:



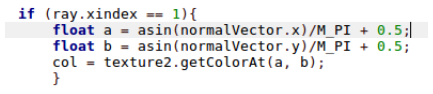
For the refraction effect, I have implemented it use recursion, because of it is need to refract multiple times to the different objects. glm::vec3 refCol = trace(outref, step+1); is the recursion part. The code basically says set up a refraction rate and if the point is inside the spheres, do refraction, otherwise return background colour.
From the right side image, it shows us the refraction effect.

Transparent effect:

Transparent effect is similar to the refraction effect except the refraction rate. I have changed the rate to 1.02 so that works as transparent. From the right side image we could see that sphere is transparent, and this why we could see the flag and sphere behind it. And the method I have used which is the same as refraction effect.



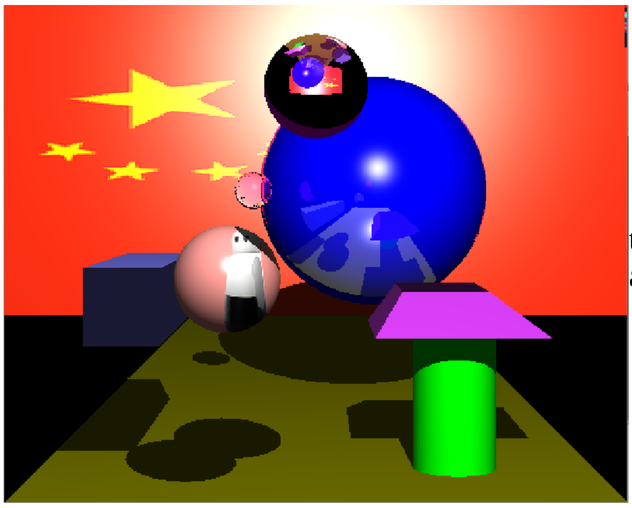
Non-planar object texture:
To texture this sphere, I have followed the lecture notes – Lec09a_MoreRayTracing. Successfully mapped the a and b point from the texture on the sphere x and y. I chose a 200*200 pixel image to fit into the sphere, the function and the sphere textured object we can see from the below.

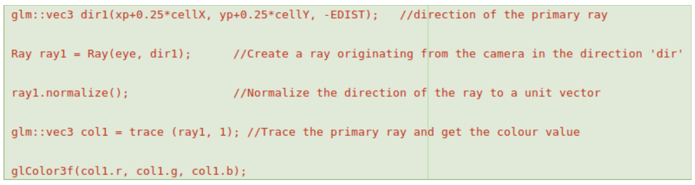


```
if (ray.xindex == 1){
    float a = asin(normalVector.x)/M_PI + 0.5;
    float b = asin(normalVector.y)/M_PI + 0.5;
    col = texture2.getColorAt(a, b);
    }
```

Planar texture:
To texture the planar is easier than texture sphere, on the lecture note it shows us how to texture on the planar, so I just followed every step and it is been successfully implemented on the planar.

Anti-Aliasing:

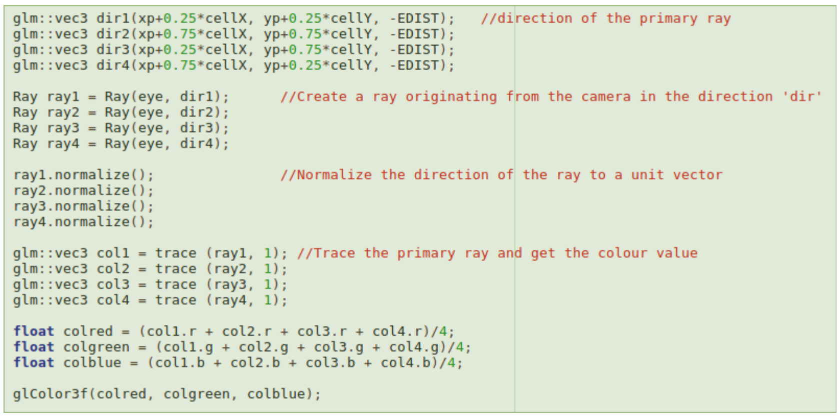




```
glm::vec3 dir1(xp+0.25*cellX, yp+0.25*cellY, -EDIST);   //direction of the primary ray

Ray ray1 = Ray(eye, dir1);      //Create a ray originating from the camera in the direction 'dir'

ray1.normalize();               //Normalize the direction of the ray to a unit vector

glm::vec3 col1 = trace (ray1, 1); //Trace the primary ray and get the colour value

glColor3f(col1.r, col1.g, col1.b);
```

the code above is the original code without anti-aliasing.

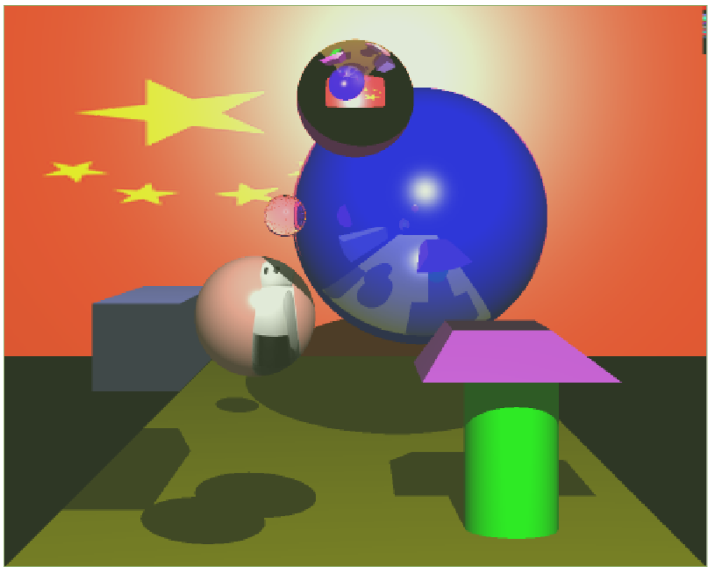
As we can see from the image above, the sphere and cylinder are not rendered very pretty – we can clearly see the edges, to improve it we could use anti-aliasing.



```
glm::vec3 dir1(xp+0.25*cellX, yp+0.25*cellY, -EDIST);   //direction of the primary ray
glm::vec3 dir2(xp+0.75*cellX, yp+0.75*cellY, -EDIST);
glm::vec3 dir3(xp+0.25*cellX, yp+0.75*cellY, -EDIST);
glm::vec3 dir4(xp+0.75*cellX, yp+0.25*cellY, -EDIST);

Ray ray1 = Ray(eye, dir1);      //Create a ray originating from the camera in the direction 'dir'
Ray ray2 = Ray(eye, dir2);
Ray ray3 = Ray(eye, dir3);
Ray ray4 = Ray(eye, dir4);

ray1.normalize();               //Normalize the direction of the ray to a unit vector
ray2.normalize();
ray3.normalize();
ray4.normalize();

glm::vec3 col1 = trace (ray1, 1); //Trace the primary ray and get the colour value
glm::vec3 col2 = trace (ray2, 1);
glm::vec3 col3 = trace (ray3, 1);
glm::vec3 col4 = trace (ray4, 1);

float colred = (col1.r + col2.r + col3.r + col4.r)/4;
float colgreen = (col1.g + col2.g + col3.g + col4.g)/4;
float colblue = (col1.b + col2.b + col3.b + col4.b)/4;

glColor3f(colred, colgreen, colblue);
```

So the way to approach anti-aliasing is to find out the closet 4 point’s and average them. As we can see the code above, I have calculated those point, add them all together and then averaged them by divide by 4.

Now we could see the image below, which is after anti-aliasing. We could say that is improved, the edge is eliminated, and the render is better.



In summary, I found it is a quite challenge assignment. I have learnt a lot and in this course. If I have couple of more days, I would like to finish more features for this assignment.

References:
1. Ray tracing lecture notes from Prof.Mukundan
2. Ray tracing labs from Prof.Mukundan
3. https://en.wikipedia.org/wiki/Flag_of_China
4. https://www.google.co.nz/search?q=robot+image&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjZ8t60o5nUAhUGn5QKHS1oCXoQ_AUICigB&biw=1920&bih=956#imgrc=2H4jSSBe3bMH2M: