Name: Unna Pussayapaiboon (64462335) **Course:** COSC363 (Assignment 2)

1. Description

1.1 Ray Tracing

RayTracer.cpp comprises of many sub classes such as Object, Vector, Color, Sphere, Plane, Cylinder, and Cone. The code from lab8 was modified to create a screen which consisted of eight elements: reflected sphere, textured sphere, 3D-textured sphere, transparent sphere, textured plane, cube, opened-cap cylinder, and cone. Vector, Colour, Sphere, and Object classes were taken directly from the lab, while RayTracer and Plane classes were modified by the student. The screen created by the ray tracer is as shown in Figure1.

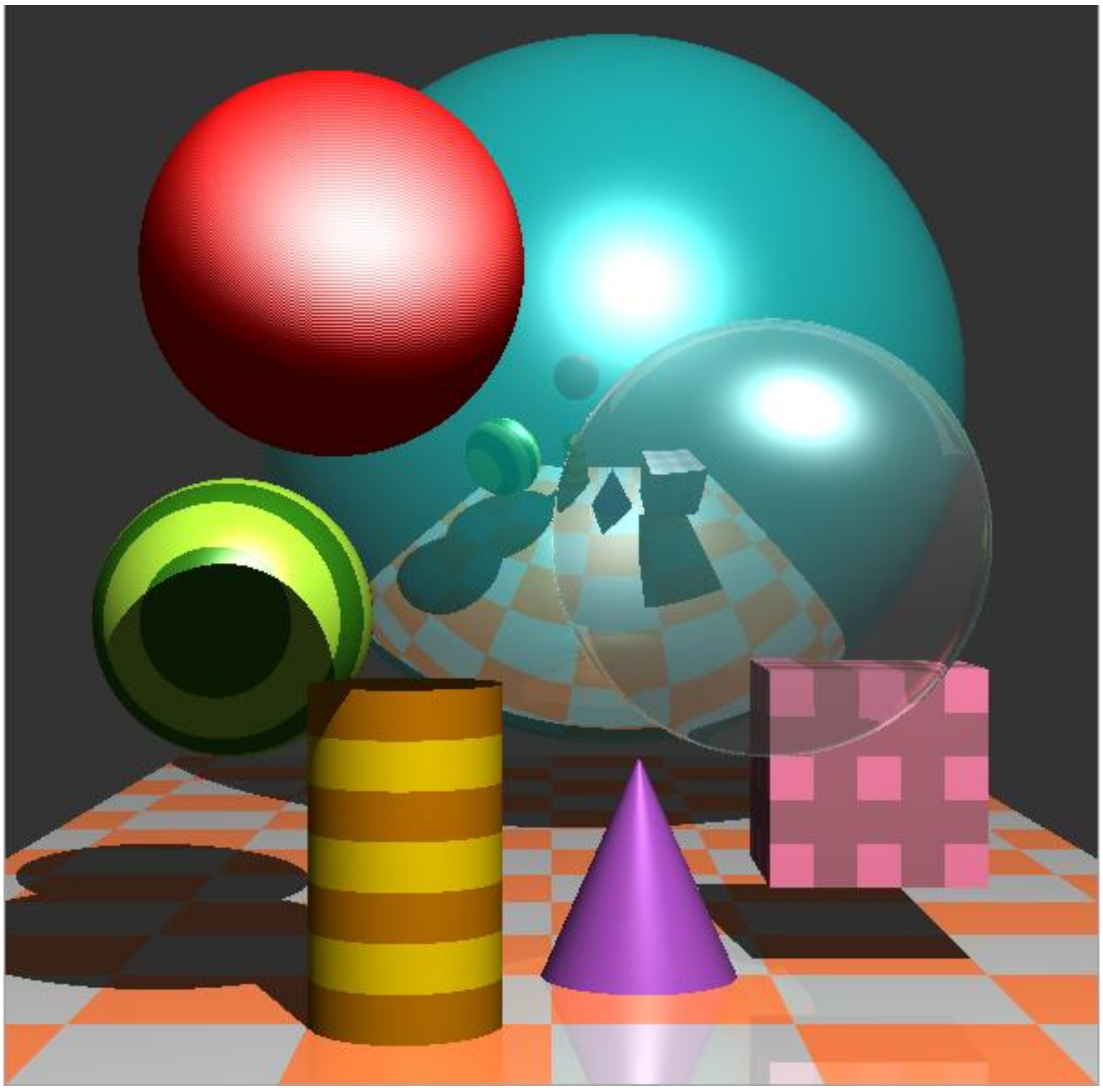

Figure 1: Screen

The primary ray is traced through each pixel on the screen, and it gets a RGB value out of the trace function. Inside the trace function, the point of the closest intersection between the ray and the object is calculated. If the ray does not intersect, the index value of -1 will be returned, and the colour of that pixel will be set to background colour.

1.2 Light and Shadow

After obtaining the point of intersection, the normal vector n is calculated as well as the light vector. The light ray l is calculated by subtracting the point of intersection q from the light source which is (10, 40, -5) in this case. The vector l is normalized, and is used to check whether the point is visible. If the dot product between l and n is less or equal than zero, it is not visible, so return the background colour.

The PointBundle q2 is created for the shadow ray. The reflection vector r is calculated by applying r = 2n(l·n) – l, and then normalized it. The index of q2 is checked and if it is -1, the phong exponent will be zero. Otherwise, the dot product of r and v will be calculated and if it is more or equal to zero, the phong exponent will be set to 10. The phong exponent results in a specular light on the surface of the object.

1.3 Successes

- The scene satisfies all basic criteria. It contains light, shadows, reflection, cube, and textured plane.
- The scene contains textured objects including 3D texture (bumpy surface).
- The scene contains cylinder and cone.
- The scene contains a refracted sphere.

1.4 Failures

- There is no anti-aliasing, so the edges around the sphere are pixelated.
- The cylinder and the cone have no caps.
- The cylinder and the cube have no highlight.
- The texture is done by checking the index, so the order of sceneObjects.push_back() cannot be changed

2. Mathematical and Implementation Aspect

2.1 Sphere

The equation of a sphere can be written as $(p - C) \cdot (p - C) = r^2$. By applying Equation1, the equation for calculating t can be formed as shown in Equation 2.

Equation 1: $$p = p_0 + td$$

Equation 2: $$t = -(s \cdot d) \pm \sqrt{(s \cdot d)^2 - (s \cdot s) + r^2}$$

The value of the smallest positive t is needed in order to find the intersection of the ray and the sphere. The normal of the sphere is equal to the nomalised vector p_0-C where p_0 is the eye position and C is the center of the sphere.

2.2 Plane

The plane can be drawn by applying the equation $(p-a)\cdot n = 0$ and Equation1. This creates Equation 3.

Equation 3: $$t = \frac{(a-p_0)\cdot n}{d\cdot n}$$

The value of t is used to find the point q or the point of intersection. The value of t is returned only if point q is inside the plane. The checking is done by finding the surface normal vector of point q and uses it to compute vectors u and v for each corner on the plane. The corner is calculated in anti-clockwise direction or from point A, B, C, and D respectively. Figure 2 shown the plane with letter labeled.

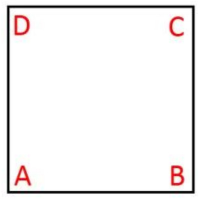

Figure 2: Plane with letter labeled.

Vector u is the different of the next point and the current point, and v is the different between q and the current point. If $(u \times v)\cdot n \geq 0$, point q is inside the plane. The unit normal vector of the plane can be computed by applying $n = (b-a)\times(c-a)$.

2.3 Cube

The cube is drawn by computing six planes. Function draw_cube() takes 8 cornered points ABCDEFGH which are used for creating planes. The points are as shown in Figure3.

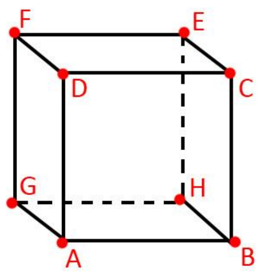

Figure3: Cube Configuration

In order to make all planes visible, the planes have to be rendered in a correct order, or from the side that is closest to the camera. The order is from top, front, right, back, left, and bottom respectively; therefore the input points for creating the planes are (DCEF), (ABCD), (BHEC), (HGFE), (GADF), and (GHBA).

2.4 Cylinder

The equation of the cylinder is $(x-x_c)^2 + (z-z_c)^2 = R^2$ for $0 \leq (y-y_c) \leq h$, where (x_0, y_0, z_0) = eye position, and (x_c, y_c, z_c) = center of the cylinder. According to Equation1, the equation for cylinder can be written as shown in Equation4.

Equation 4: $$t^2(d_x^2 + d_z^2) + 2t\{d_x(x_0-x_c) + d_z(z_0-z_c)\} + \{(x_0-x_c)^2 + (z_0-z_c)^2 - R^2\} = 0$$

Equation4 can be written in form of $at^2 + bt + c = 0$, so the value of t can be solved by using the quadratic formula: $t = \frac{-b\pm\sqrt{b^2-4ac}}{2a}$. Based on the same rule as the sphere, the value of the smallest and positive t is chosen as the closest intersection value. Before returning the value of t, the height of the cylinder is checked. The value of t will be returned only if the point q calculated from $q = p_0 + td$ is within range for $0 \leq (y-y_c) \leq h$. The checking is done in checkTall function. The maximum value of y is calculated by y_max = y center + height of the cylinder. The q point is also calculated using Equation 1 and if q is within y center and y_max, the value of that t is accepted. The normal of the cylinder is (x-x_c, 0, z-z_c).

2.5 Cone

The equation of cone is $(x - x_c)^2 + (z - z_c)^2 = \left(\frac{R}{h}\right)^2 (h - y + yc)^2$ which is very similar to the cylinder. By substituting Equation1 to the cone's equation, the right hand side can be simplified into:

$$\{(h - y_0 + y_c)^2 - 2d_y t(h - y_c - y_c) + (d_y^2)t^2\} \times (\frac{R^2}{h})$$

Where R is the radius of the bottom of the cone and h is the height of the cone.

The equation of the cone is then simplified into a form at^2 + bt + c = 0 with the components:

$a = d_x^2 + d_z^2 - d_y^2(\frac{R^2}{h})$, $b = 2\{d_x(x_0 - x_c) + d_z(z_0 - z_c) + \ d_y(h - y_0 + y_c)\left(\frac{R^2}{h}\right)\}$,

$c = (x_0 - x_c)^2 + (z_0 - z_c)^2 - (h - y_0 + y_c)^2(\frac{R^2}{h})$

The quadratic formula is applied again for finding the value of t, and the height of the cone is checked by using the same method as the cylinder. The normal of the cone is (x-x_c, r tan(θ), z-z_c) where $r = \sqrt{x^2 + z^2}$, and $tan\theta = \frac{R}{h}$.

2.6 Texture

The colour of the object is obtained from getColor() function. The specific texture is painted on the object by checking the index and the point of intersection q.

The striped texture on the sphere is produced by checking the z coordinate of q. If q.point.z%2 is more than 1 for positive case and less or equal than 1 negative case, the colour will be changed. The striped texture on the cylinder also uses the same method but instead of checking the z coordinate, the y coordinate is checked. Chequered texture on the plane is done by checking the z and x coordinate at the same time; this is for checking the row and column. The texture on the cube is a modification of the chequered texture, but the colour is done in the opposite way. Also, the x y and z coordinates are all checked. Some texture uses combineColor() function to get a more appealing colour. The snapshot of textured objects is as shown in Figure4.

2.7 Bumpy Texture

The bumpy texture is implemented on the red sphere. This 3D texture is created by changing the normal of the sphere according to the current_bump and the noise coefficient. The current bump increases from 0 to 2, and will toggle between 1 and 2 to generate a noise on the sphere's surface. The new normal is then calculated by: $n = n(current_{bump}\ ) + noiseCof(current_{bump}\ )$ where noiseCofx = 0.1, noiseCofy = 0.1, and noiseCofz = 1.5. The bumpy sphere is as shown in Figure 5.

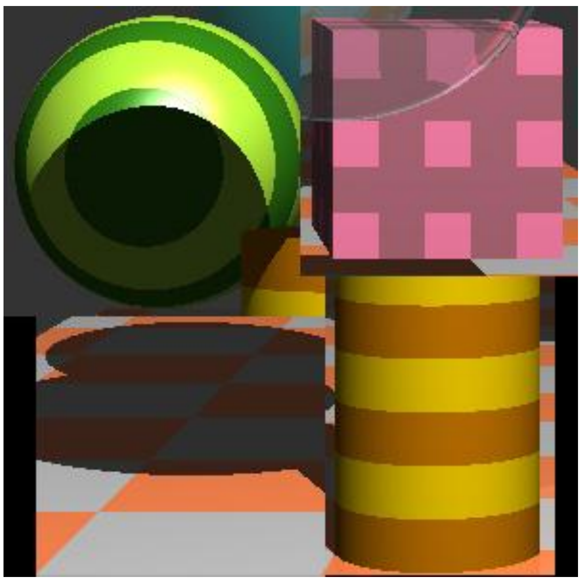
Figure4: Textured Objects

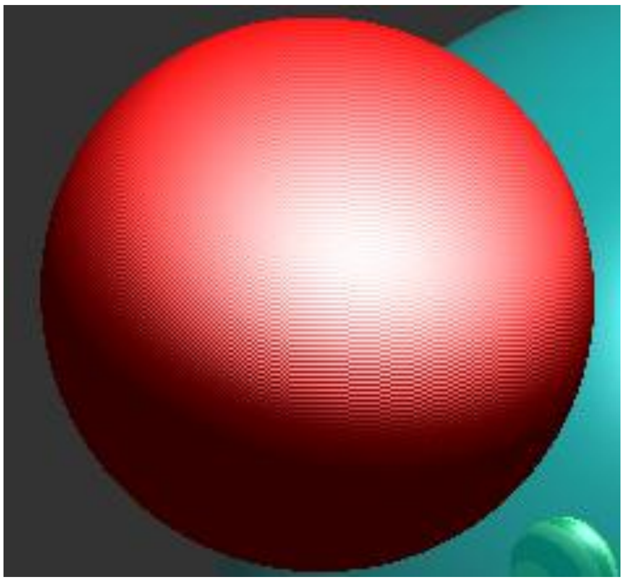
Figure 5: Bumpy Sphere

2.8 Reflection

The reflection is done by calculating $r = 2(v \cdot n)n - v$ and put r in the trace function again to calculate the reflected colour. The reflected colour is then combined with the original colour. The reflection is applied on a blue sphere, a plane, and refracted sphere with a different reflection coefficient.

2.9 Refraction

The refraction is divided into two paths: when the light enters the sphere and when the light exits the sphere. The medium n1 is defined as 1.00 which is air and n2 is 1.001. If the light enters the sphere, the value of $d \cdot n$ will be less than zero; else it exits the sphere. If the light exits the sphere, the values of two mediums have to be swapped (n1=n2, n2=n1), and the normal will be set to negative. According to this rule, the refracted ray g can be calculated by using Equation 5.

Equation 5:
$$g = \left(\frac{n1}{n2}\right) d - \left(\frac{n1}{n2}(d \cdot n) + cos\theta_t\right) n$$
$$\text{where } cos\theta_t = \sqrt{1 - \left(\frac{n1}{n2}\right)^2 (1 - (d \cdot n)^2)}$$

After this, ray g is traced again to get the refracted colour which will be combined with the original colour. The combination of reflection and refraction on a sphere at index=2 makes the sphere looks like bubble.

3. References

COSC363 Lecture 8-Ray Tracing, Lab7, and Lab8
http://www.codermind.com/articles/Raytracer-in-C++-Part-I-First-rays.html
http://www.cl.cam.ac.uk/teaching/1999/AGraphHCI/SMAG/node2.html
http://www.codermind.com/articles/Raytracer-in-C++-Part-III-Textures.html