COSC363 – Assignment Two

By Andrew Dallow, ID: 56999204

1 Scene Description

The ray traced scene consists of a large reflective green sphere in the centre sitting on top of a yellow cone, an orange rotated cube, and four other smaller spheres distributed about the scene. One of the smaller spheres has a procedurally generated texture giving it the colour of a bluish rainbow, while another sphere is transparent with a distorted view of what is behind it, which was achieved via refraction. Two identical orange pillars are placed at the back each of the scene and were created via cylinders. The entire scene is enclosed in a boxed area with a yellow back wall, two red side walls and a reflective floor with a blue and white checkerboard pattern

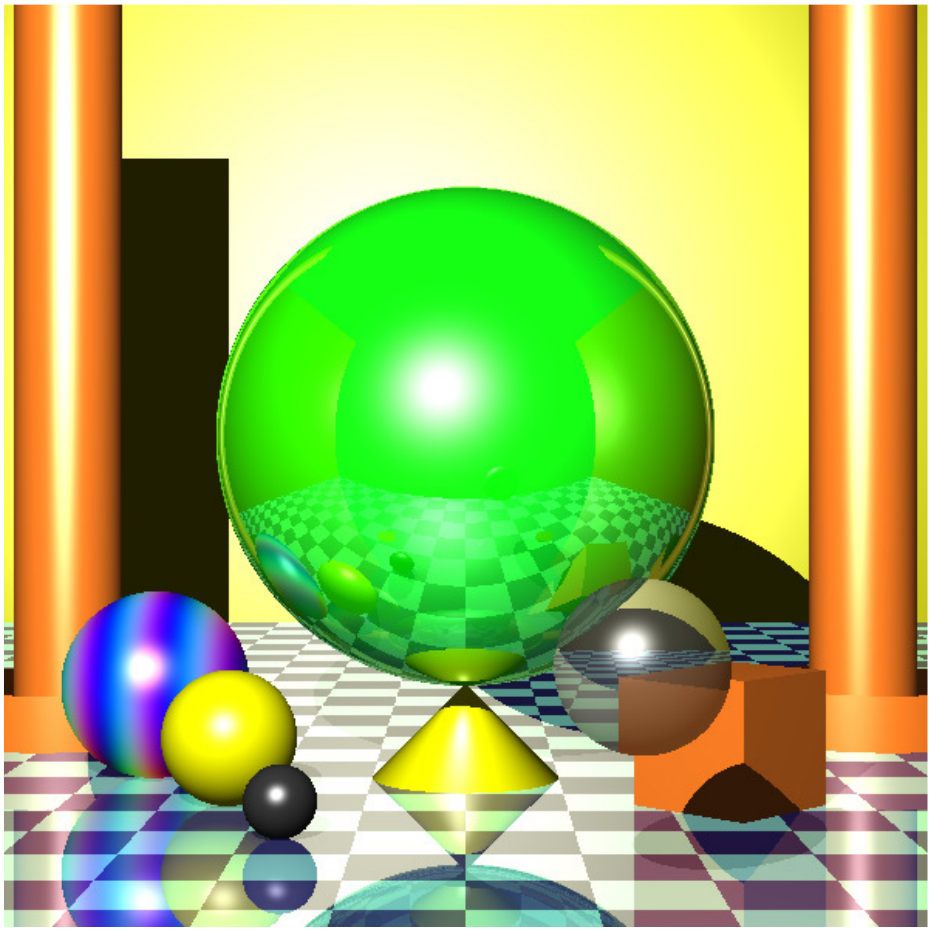

Figure 1 – Image of the ray traced scene

2 Features

- **Cube (rotated)**
  - Created using six planes with a height and width of 4 units.
  - A method called 'rotation' was added to the 'Vector' class which can be used to rotate a point about a given axis using the rotation equations.
  - The cube was rotated about the y-axis by 45 degrees using the equations:

$$x' = x\cos(\theta) + z\sin(\theta),$$
$$y' = y,$$
$$z' = -x\sin(\theta) + z\cos(\theta).$$

- **Reflections**
  - The large green sphere in the middle of the scene and the floor have been made reflective.
- **Shadows**
  - All objects in the scene have shadows.
- **Refracting Sphere**
  - The grey sphere on the right of the scene uses refraction to make the area behind it visible through the sphere. This view is also distorted.
  - This was achieved by using Snell's law to determine the angle of refraction within the sphere, and then as a ray passes out of the sphere. To calculate the vector of the refracted ray ($\boldsymbol{g}$), the following equations were used:

$$\boldsymbol{g} = \left(\frac{\eta_1}{\eta_2}\right)\boldsymbol{d} - \left(\frac{\eta_1}{\eta_2}(\boldsymbol{d}\cdot\boldsymbol{n}) + \cos(\theta_t)\right)\boldsymbol{n},$$

$$\cos(\theta_t) = \sqrt{1 - \left(\frac{\eta_1}{\eta_2}\right)^2 (1 - (\boldsymbol{d}\cdot\boldsymbol{n}))^2}$$

    Where, η_1 and η_2 are the index of refraction for each medium, $\boldsymbol{d}$ is the incident ray vector, $\boldsymbol{n}$ the normal vector to the surface, and θ_t is the angle of transmission.
  - In the ray tracer these calculations were done using a separate function called 'refraction' which returns the refracted vector.
  - This function is called once when the incident ray intercepts the sphere's surface, then the 'closestPt' function is used with this vector to determine the exit point on the other side of the sphere.
  - Then the 'refraction' function is called again to calculate the exit vector, but with a reversed normal to get an outward pointing vector.
  - Finally, the 'trace' function is called to obtain the colours intercepted behind the sphere and is added to the colour sum for the sphere's surface.
  - The indexes of refraction have been artificially modified to get a transparent-like sphere, instead of a reversed image of the background.
- **Cone**
  - The yellow cone underneath the large green sphere was created using the following interception equation:

$$(x - x_c)^2 + (z - z_c)^2 = \left(\frac{R}{h}\right)^2 (h - y + y_c)^2$$

    Where, the subscript 'c' represents the points at the cone centre, R the radius of the cone, and h the height of the cone.
  - The ray equations were then substituted in for x, y, and z giving the interception equation:

$$\left(d_x^2 + d_z^2 - d_y^2\right)t^2 + 2\left((x_0 - x_c)d_x + (z_0 - z_c)d_z - (y_0 - y_c)d_y\right)t + ((x_0 - x_c)^2 + (z_0 - z_c)^2 - (y_0 - y_c)^2 = 0$$

    Where, d is the direction vector of the ray, (x_0, y_0, z_0) is the origin point of the ray, and *t* it the distance to the intersection point.

  - This interception equation was then solved by the quadratic formula to give two values of *t*.
  - In order to create a finite cone with the correct height and orientation the following rules where implemented, first calculating the two possible y-intercepts y1(t1) and y2(t2) (where, t1 < t2):
    - If $-h \leq y1 \leq 0$,
      - If $t1 \leq 0, t = -1$, no intercept
      - Else return $t = t1$
    - Else if $y1 > 0$
      - If $y2 > 0, t = -1$, no intercept

        - Else $t = t2$
    - Otherwise $t = -1$, no intercept
- One failure of the cone is that it has no base with this representation.
- The normal for the cone is calculated as follows (**centre** = (x_c, y_c, z_c)):

$$\boldsymbol{n} = (\boldsymbol{p}_0 - \boldsymbol{centre})\left(\frac{h}{R}\right), \qquad y_c = \frac{R}{h}$$

- **Cylinder**
    - The cylinder is created in a similar way to the cone but uses the following equation:

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

    - Substituting in the ray equations and solving for zero, gives the following intercept equation:

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

    - This interception equation was then solved by the quadratic formula to give two values of t.
    - In order to obtain a finite cylinder with end caps, the following rules where used, first calculating the two possible y-intercepts y1(t1) and y2(t2) (where, t1 < t2):
        - If $y1 < 0$
            - If $y2 < 0$, $t = -1$, no intercept
            - Else (hit cap) $t = t1 + \frac{(t2-t1)(y1+h)}{y1-y2}$, where $t > 0$ otherwise $t = -1$.
        - Else if $0 \leq y1 \leq h$
            - If $t1 \leq 0$, $t = -1$, no intercept
            - Else $t = t1$
        - Else if $y1 > h$
            - If $y2 > h$, $t = -1$, no intercept
            - Else (hit other cap) $t = t1 + \frac{(t2-t1)y_1}{y1-y2}$, where $t > 0$ otherwise $t = -1$
        - Otherwise $t = -1$, no intercept
    - The normal is calculated by the following:

$$\boldsymbol{n} = (\boldsymbol{p}_0 - \boldsymbol{centre}), \qquad y_c = 0.$$

    - Note: the code at [1] was used as a guide for implementing this code.
- **Textured Floor**
    - The floor texture is procedurally generated as a checkerboard pattern.
    - This is achieved by adding a specific colour to the colour sum via the ray tracer depending on the value of the interception point on the plane in the x and z direction.
    - The following conditions were implemented to get the checkerboard pattern:
        - If $remainder(q.point.z, 6) \geq 0$ *and* $remainder(q.point.z, 6) < 3$
            - $remainder(q.point.x, 6) \geq 0$ && $remainder(q.point.x, 6) < 3$
                - $colour = (0.5, 1, 1)$
            - Otherwise
                - $colour = (0, 0, 0.4)$
        - Otherwise (reverse pattern)
            - $remainder(q.point.x, 6) \geq 0$ && $remainder(q.point.x, 6) < 3$
                - $colour = (0, 0, 0.4)$
            - Otherwise
                - $colour = (0.5, 1, 1)$
    - The 'remainder' function is a standard c++ function for calculating the modulus of float numbers.
- **Textured Sphere**
    - The bluish sphere on the far left of the scene was procedurally textured to give it a rainbow-like texture.
    - This was done in the ray tracer by assigning each RGB colour based on the x position and the following equations:
        - $RED = \frac{1}{2}(1 + \sin(q.point.x * 3))$

        - $GREEN = \frac{1}{2}(1 + \cos(q.point.x * 3))$
        - $BLUE = 0$
    - This colour was then added to the colour sum for that position.

3 Runtime and File List

The Ray Tracer takes approximately 10-15 seconds to run before displaying the rendered scene.

File List:

- Color.cpp
- Color.h
- Cone.cpp
- Cone.h
- Cylinder.cpp
- Cylinder.h
- Makefile
- Object.ccp
- Object.h
- Plane.cpp
- Plane.h
- RayTracer
- RayTracer.cpp
- Sphere.cpp
- Sphere.h
- Vector.cpp
- Vector.h

4 References

[1] D. PenFold, “Cylinder intersection,” woo4.me, 29 01 2014. [Online]. Available: http://woo4.me/wootracer/cylinder-intersection/. [Accessed 06 01 2015].