ASSIGNMENT 04: RAYTRACING

COMPUTER GRAPHICS (CSE333)

Harsh Vardhan Singh
ECE
IIIT Delhi
2020202
harsh20202@iiitd.ac.in

ABSTRACT

In this assignment, we will modify a ray tracer to generate excellent images. We have been provided with a basic ray tracer, Lumina. We must extend the ray tracer to achieve the questions.

1. Extend the object class to implement a Triangle that can be rendered in the ray tracer.
2. Implement Blinn-Phong shading for the shapes.
3. Implement shadows in the ray tracer.
4. Implement reflective and dielectric materials (i.e., recursive the ray tracer).
5. Implement jittered supersampling to improve the quality of the rendering. Perform the same only around edges to ensure performance.
6. Implement transformed primitives.
7. Implement the generation of a depth map from the ray tracer (instead of a colour image). Encode a floating-point depth in an 8-bit integer.

PART 1:

1. We extended the objects class to implement a Triangle that can be rendered in the ray tracer. We can define a triangle by providing 3 points in the "vector3D" form. Then, we use the Moller–Trumbore intersection algorithm to find if the ray intersects with the triangle.

• Intersect a ray with a parametric surface

$xe + t(xd) = f(u, v)$

$ye + t(yd) = g(u, v)$

$ze + t(zd) = h(u, v)$

• Three equations and three unknowns (t, u, and v)

• For a parametric plane, the parametric surface can be represented in terms of any three points in the plane

• We utilise barycentric coordinates for ray-triangle test

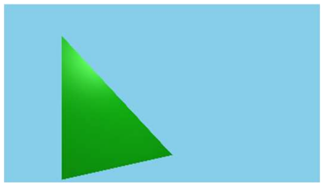
• For a triangle with vertices a, b, and c, an intersection will occur when

$e + td = a + \beta(b-a) + \gamma(c-a)$

• Intersection is inside the triangle if

$\beta>0$, $\gamma>0$, and $\beta+\gamma<1$

• Otherwise, the ray has hit the plane outside the triangle



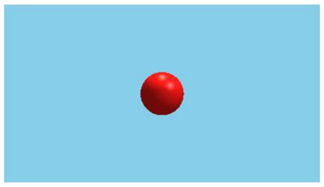
PART 2:

1. We implemented the Blinn Phong shading technique for multiple lights. We calculated ambient lighting, which considered all the lights and added an ambient component at the end. We calculated the diffused and specular components effect for each light separately and added them up, which considered both the object's and the light's colour before shading.
2. We also had to calculate normal for spheres and triangles separately

• Sphere: normal is obtained by subtracting the hit point from the sphere centre position.

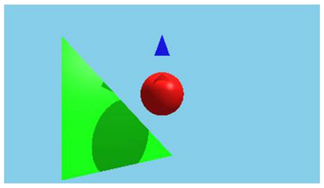
• Triangle: normal is the cross product of 2 edges, which can be calculated from the 3 points given.

•These normals are normalised before being set.



PART 3:

1. We implemented shadows in the ray tracer. We emit rays a bit above the hit point of a particular object (It is a bit above it to avoid self-shadow at the hit point itself). The direction of these rays points towards the light's position. If this probing ray intersects with another object before reaching the light, it will cast a shadow. To avoid making the shadow completely black, we add the ambient light to the regions where a shadow is being formed. If the probe ray reaches the light source without any intersection, no shadows are included, and shading is applied as before.



PART 4:

1. We implemented reflective and dielectric materials.
2. For specular reflection
   - Mirror reflections can be added by shading reflected rays

     r = d – 2(d.n) n

     where r is the reflected ray, d is the incident ray, and n is the normal of the object at that point
   - Some energy is lost during the reflection of light from the surface (loss depends on wavelength)
   - Following recursion adds reflection

     colour c = c + km (ray colour (p + tr, ε, ∞))

   km: specular RGB colour for mirror reflection
3. For dielectric material, we had to implement four methods … Refraction, Total internal reflection, Schlick Approximation, Beer's Law
   - Refraction:
     - Dielectric: a transparent material that refracts light
     - Light bends when it travels from a medium with refractive index n to one with nt
     - Snell's law: n*sin θ = nt*sin Φ
     - To compute transmitted ray t, we define it in terms of orthonormal basis b and n [t = sin Φb + cos Φn]
     - Incoming ray d can be written as [d = sin θb – cos θn]
     - Solving for t:

       t = n (d + n (d · n))/nt - n (1-(n^2*(1-(d. n) ^2))/(nt^2)) ^1/2 (we use a little changed from)
   - Total internal reflection:
     - Critical angle can be computed as:

       k = 1 - n^2*(1-(d. n) ^2)/(nt^2) < 0 -> cos(θ) = (1-nt^2/n^2) ^1/2
     - We can use this angle to shade the area to be reflective (if k<0). It will be refractive otherwise.
   - Schlick approximation:
     - The reflectivity of a dielectric varies with the incident angle as per the Fresnel Equations
     - Schlick approximation may be used instead

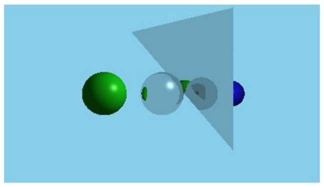
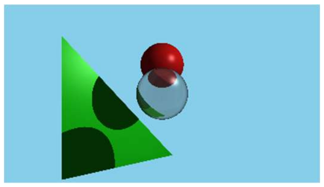
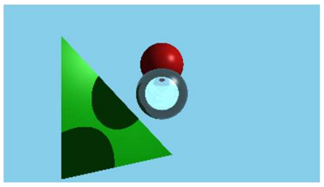
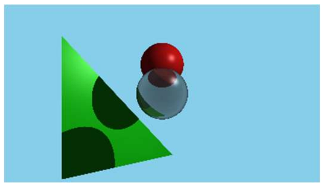
       R(θ) = R0 + (1 - R0) (1 – cos θ) ^5
     - R0 is reflectance at normal incidence

       R0 = ((nt-1)/(nt+1)) ^2
     - θ: angle in rarer medium (larger of the two angles)
   - Beer's law:
     - For homogeneous impurities in a dielectric, a light-carrying ray's intensity attenuates as per Beer's law
     - Loss of intensity in the medium: dI = -CIdx

• Solution: I = k exp (Cx) + k' with boundary conditions: I(s) = I(0)e^ln(a)s

s: distance from the interface, a: attenuation constant (attenuation after unit distance)

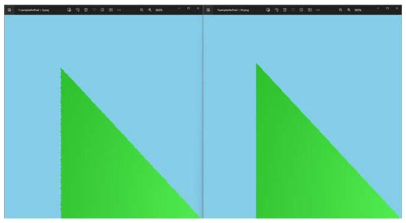
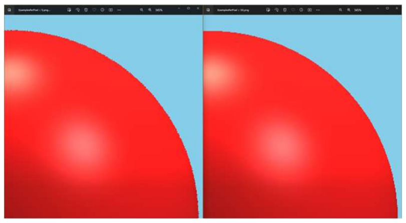
4. Here are examples of objects implementing Reflection, Refraction + Schlick's approx., TIR, and Beer's law, respectively

* Note: in reflection, after some reflections, we see the object turn black as the limit of recursion depth is reached (which we set). Also, the glass material becomes dark at specific areas in beer's law implementation due to impurities

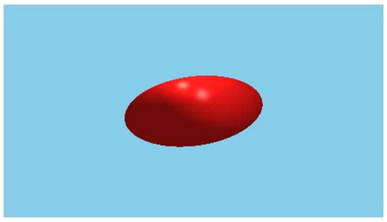
PART 5:

1. We implemented jittered super sampling to enhance the quality of rendered images. It involves taking multiple samples within each pixel and slightly displacing their positions in a jittered pattern. This mitigates aliasing artefacts and results in smoother, more realistic images.". The more the parameter is, the better the render. (We could not optimise it, so the super sampling occurs for the whole render rather than just the edges). (See effects of supersampling. The left side samples per pixel =1, and on the right side, samples per pixel=16 for each image).

PART 6:

1. We implemented transformed primitives. We can stretch, rotate, and move around primitive objects such as spheres and triangles with the help of a transform matrix (for which we created a script which allows us to create 4x4 identity matrices, apply scale and rotation, translation, and scaling, and multiply vectors and other matrices to it).
2. We had to handle shadows as well as we had to transform these probe rays as well. We detected if the object was of the transformed type and multiplied the transform matrix by the shadow ray.



PART 7:

1. We implemented a depth generation map. We calculate the distance of each ray being emitted from the camera. We set the maximum and minimum depth and then normalised the values of all the distances. It is converted to uint_8 and then used to give a colour with the same r, g, and b values set as the depth, which is between 0 and 1.

2. Below is an image of a scene in the ray tracer and its depth map.