COSC 342 Assignment 2A

Ray Tracer

Rhianne Price 1953770

Cone and Cylinder Intersections

To find the intersection of a ray with a unit cone, I have split the cone into two parts. To find intersections along the curved surface of the cone I have used the unit cone and ray:

$$x^2 + y^2 - z^2 = \mathbf{p.p} - 2p_z^2 = 0 \text{ and } \mathbf{p} = \text{ray point} + \lambda\,(\text{ray direction})$$

to get a quadratic in λ, the distance along the ray to the intersection point. To solve for λ my implementation uses the quadratic formula. The sign of the term under the square root splits the problem into three cases, no real intersections , one intersection, and two intersections. Once we know how many intersections the transformed ray has with the cone, we can find the distance along the ray to the intersection/s and then the intersections themselves by plugging λ back into our equation for **p** . If this intersection is in front of the camera ($\lambda > 0$), and if the intersection's z component is the appropriate range for a unit cone ($0 < z < 1$), the intersection is transformed before being added to the result to be returned. For the base of the cone I have checked whether the ray intersects the plane $z = -1$ where the distance to the z axis is less than 1.

My implementation deals with cylinder intersections in a similar way. To calculate the point where the ray intersects with the barrel part of the cylinder, I used the cylinder and ray:

$$x^2 + y^2 = \mathbf{p.p} - p_z^2 = 0 \text{ and } \mathbf{p} = \text{ray point} + \lambda\,(\text{ray direction})$$

and solved the resulting quadratic in the same way as for a cone to get the intersection/s if any, and then checked whether the intersection's z component is in the interval (-1, 1). For the end caps I have checked whether the ray intersects the plane $z = -1$ or $z = 1$ where the distance to the z axis is less than 1.

To test these intersections I rendered a scenes with unit cones and cylinders rotated at various angles about the x and y axes to check whether the geometry looked correct. It took a few adjustments to get the geometry right for both cones and cylinders, so testing proved important.

The normals to the barrel part of the cylinder are just the normals to the rings around the cylinder. I took the hit point and zeroed out the z component to get a perpendicular vector to the z axis. For the normals to the points on the cone's surface, I started with an incorrect calculation that gave the normal to the rings around the cone, which essentially gave the correct x and y directions of the normal with no direction in z. This looked plausible in my initial tests as I hadn't implemented specular or diffuse lighting. After implementing specular lighting, my cone was showing improbable dark patches in test scenes, so I went back to change the calculation of the normals to cone surface to include the appropriate z component. My implementation now calculates the normal to the curved part of the cone's surface as $2x + 2y - 2z$, where x, y and z represent the intersection.

Lighting

I have implemented diffuse and specular shading. My implementation calculates the diffuse part for each light using the object's surface normal at the hit point, the unit vector from the hit point to the light and the object's material's diffuse component. I have calculated the specular part for each light using the unit direction from the hit point to the camera, the reflected direction of the light around the hit point's normal, and the material's specular colour and specular coefficient. Adding these two components and multiplying by the light intensity and colour for each light before summing across all lights and bounding the result, gives the colour of the object at the hit point.

To test diffuse shading I rendered a scene with spheres with diffuse components. They appeared consistent with what I expected. I tried scenes with different coloured light sources and multiple light sources to check whether the materials still looked reasonable.

At first when I tested specular lighting, I tested materials with both specular and diffuse components. This looked reasonable until I checked just the specular component. I realised I was using the wrong reflected ray in my specular calculation. In my initial renders the interaction between the specular component and the diffuse component made this mistake hard to detect. I found it was important to test one thing at a time, but also the interaction between different things. I have one persisting issue with my specular lighting. If the specular exponent is less than or equal to two, I am seeing unexpected behaviour. I haven't been able to fix the mistake but I think it is something to do with clipping the colour at the wrong time.

Shadows

To implement shadows cast by objects, for each light, a new ray is cast from the hit point to the light and checked for intersections. If the distance along this ray to the closest intersection is in the in the interval [0, 1], this hit point is in shadow to this light, so the light's intensity is overridden by zero. To test shadows, I rendered scenes with multiple objects and placed lights in a way which would make objects cast shadows and checked whether these looked reasonable. I tried this with multiple light sources to check the interaction between shadows.

Reflection

I have implemented reflection with a recursive approach. After computing and clipping the colour at the hit point, I compute the final hit colour as a weighted sum of that colour and the colour of the hitpoint on the ray reflected around the objects normal at the original hit point. This sum is weighted by the surfaces reflective colour or mirror component.

To test this, I rendered a scene with a sphere sitting on the end cap of an enlarged cylinder with a reflective material. I was expecting to see the reflection of the sphere in the flat surface of the cap, but initially didn't. In this test scene, the cylinder's material also had a specular component.

I was seeing unexpected behaviour, with some dark spots and no reflection. At this stage I hadn't corrected the mistake in my implementation of specular shading. Specular materials looked plausible without reflection, so I thought there was something wrong with my implementation of reflection. This is where it was useful to separately test features of my ray tracer to work out where the unexpected behaviour was coming from. Once I spotted the problem in my specular calculation, reflections started appearing as expected. This unit-testing like approach was a useful way to debug my ray tracer.

Spotlight

The points in the scene lit by spotlight are the points that make an angle with the spotlight's axis that is less than the spotlight's angle of illumination. This angle is equal to the inverse cosine of the dot product between the unit vector between the light location and the point and the spotlight's unit direction. My implementation calculates this angle and returns the light that reaches that point according to the inverse square law if the calculated angle is less than the spotlights angle, or zero if the calculated angle is greater than the spotlights angle.

To test this I rendered a scene with spotlights of varying angles pointed directly towards the flat surface of an object, and checked whether the circles are of expected proportions. I also tried spotlights with different directions.I have also rendered a scene where the spotlight lights the edge of one object and part of another.