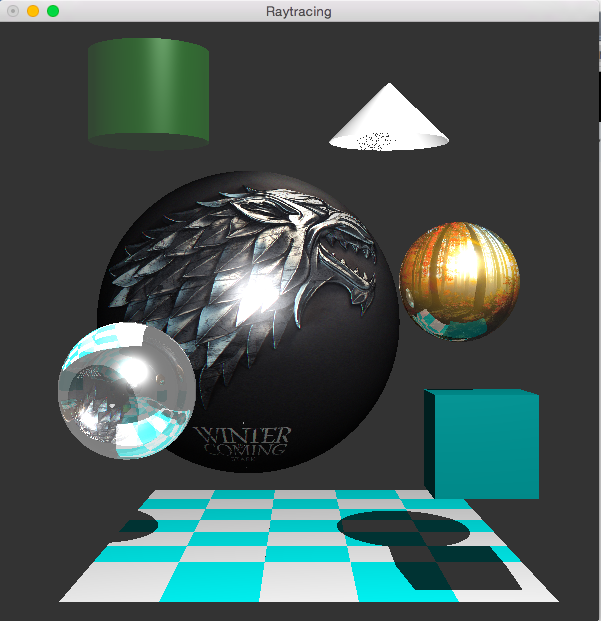
Ray Tracer Assignment

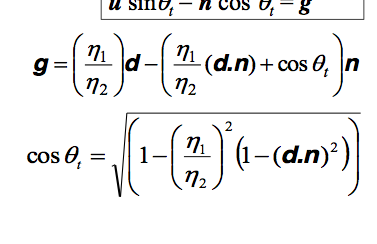


**The ray tracer implements the following features:**

1. Lighting, including diffuse and specular
2. Shadows
3. Reflections
4. Checkered floor composed of several planes
5. Cube composed of 6 planes
6. Transparent and refractive sphere
7. Transparent cylinder
8. Cone
9. 2 texture mapped sphere surfaces

**Implementation details of Extra features:**

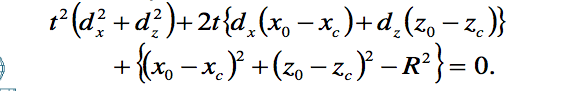
Refraction and Transparency:

The transparent sphere uses the following equation to compute the refraction direction - 

Where g is the direction of the refractive ray. n1/n2 is computed by checking whether we point is currently inside or outside the sphere, which can be found by computing the dot product of the direction ray and the normal. The trace function is then called recursively and the return of this is combined with the current material color.

Cylinder:

The rendering of a sphere was similar to that of a sphere, except the equation for the intersection and normal functions were different.

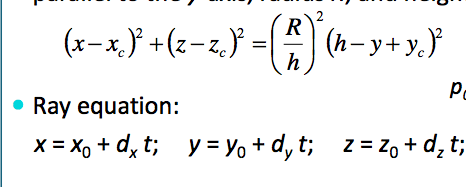


The above equation was used to find the intersection points, of which there were at most 2. Once a point of intersection was found, it was necessary to compute the y hit point using the equation y = y0 + dy t, in order to check whether the intersection lied within the range of the height of the cylinder. If the point of intersection was either above or below the cylinder, then it was necessary to similarly test the second point of intersection for a valid point.

To make it transparent I simply called the trace function again with the direction.

Cone:

the intersection points of a cone were found using the equations:



Once solved for t, the y hit point was tested similar to how it was for the cylinder.

Texture mapping a sphere:

To obtain a procedural texture mapping, it was necessary to obtain the coordinates of the texture based on the point on the sphere.

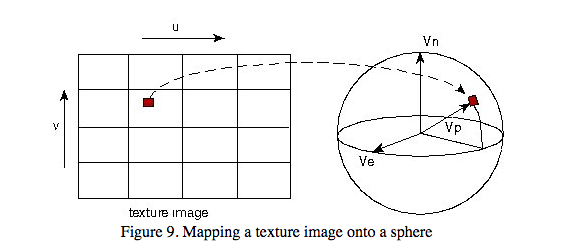


Fig.1 texture mapped to a sphere (Rademacher)

To do so, first I obtained the the following vectors shown in the figure above, Vn, Ve, and Vp. These were calculated using the center of the sphere, and the point of intersection.

It was then necessary to compute the angle phi between the vectors Vn and Vp, which was obtained using the arcos of the their dot product.

The second angle to compute was theta, which obtained from this formula:

Macintosh HD:Users:ceciliavillatoro:Desktop:Screen Shot 2016-06-01 at 7.08.48 PM.png

the coordinates of the texture (u,v) were obtained as follows:

v = phi / PI;

u = theta, if Vn.cross(Ve).dot(Vp) > 0

u = 1 – theta otherwise

Once these were found, the rgb values of the texture could be found by accessing the array holding these values. The code for reading in a bmp image was found in the article "Read Pixel Value in Bmp File." referenced below.

References:

"Flipping the 2D Texture on a Sphere with Ray-Tracing." *Stackoverflow*. Web. 01 June 2016.

<http://stackoverflow.com/questions/11780147/flipping-the-2d-texture-on-a-sphere-with-ray-tracing>

"How Do I Use Texture-mapping in a Simple Ray Tracer?" *Stackoverflow*. Web. 01 June 2016.

<http://stackoverflow.com/questions/2649674/how-do-i-use-texture-mapping-in-a-simple-ray-tracer>

"Read Pixel Value in Bmp File." *Stackoverflow*. Web. 01 June 2016.

<http://stackoverflow.com/questions/9296059/read-pixel-value-in-bmp-file>

Rademacher, Paul. "Ray Tracing: Graphics for the Masses." *Ray Tracing: Graphics for the Masses*. Web. 01 June 2016**.**

<https://www.cs.unc.edu/~rademach/xroads-RT/RTarticle.html>