RAYCASTING WITH SPHERES

ESTY THOMAS

FRAMES/MATLAB UNIT

1. Goals

The goal of this assignment was to create a functional ray-casting program to render three-dimensional spheres in a two-dimensional image - that is, given an image array, a light source at infinity, and a sphere floating out in the distance somewhere, project rays from the viewpoint through each pixel of the image array and calculate the light value of those rays that hit or do not hit the sphere. The camera ought to be able to track, pan, and tilt, and spheres should be of varying levels and sorts of reflectance.

2. Methods

2.1. **Image Array ('Retina').** The image array, or retina, as it's named in the code, for an NxN image is a 3xN^2+1 matrix; each column represents one point: $\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$. The final column is equal to the viewpoint.

2.2. **Sphere-rendering.** The sphere floating out in the void somewhere is given a center and a radius from user input. To render the sphere, a line is cast from the viewpoint through each pixel on the image array and tested to see if it intersects with the sphere. If it does, the brightness is set based on the semi-gloss reflectance equation (which, given the correct parameters, also gives Lambertian brightness values). If the line does not intersect the sphere, its brightness is set to the ambient brightness.

There are two intersection points with the sphere for almost all intersecting rays. When the closer is chosen as the point to light, the sphere appears completely unlit and entirely self-shadowed. When the farther is chosen, the sphere renders with shading, but if given gloss, the rendering becomes concave rather than convex.

2.3. **Camera Movement.**

2.3.1. *Tracking.* Side-to-side tracking adds a translation vector to every element of the retina.

2.3.2. *Panning.* Side-to-side rotation/panning multiplies each element of the retina by a rotation array: $\begin{pmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{pmatrix}$

2.3.3. *Tilting.* Up-and-down tilting multiplies each element of the retina by a rotation array: $\begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{pmatrix}$

2.4. **Brightness.** The brightness of a given point is calculated according to the equation helpfully included in the project materials: $B = a(sC^n + (1-s)(\vec{l} \cdot \vec{n}))$, where $C = 2(\vec{n} \cdot \vec{l})(\vec{n} \cdot \vec{v}) - (\vec{v} \cdot \vec{l})$, s is the proportion of specular (mirror-like) effect, n is the degree of perfection of the mirror effect, a is the albedo of the surface at the point, $\vec{n}$ is the surface normal vector of the point, $\vec{l}$ is the direction of the light, and $\vec{v}$ is the direction vector for the viewpoint. When s = 0, this gives the Lambertian brightness.

3. Results

All column vectors are written as row vectors (x, y, z) for the sake of simplicity. The ambient light is .9 for all images.

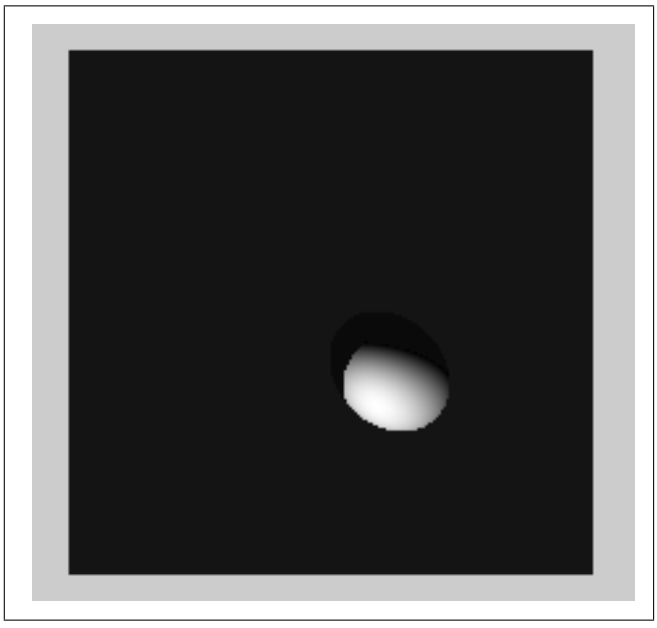

Figure 1. Circle A: a sphere with r = 100, centered at (100, 100, 200), viewed from (0, 0, -20), with the light coming from (1, 5, 0). No gloss. Albedo = 5.

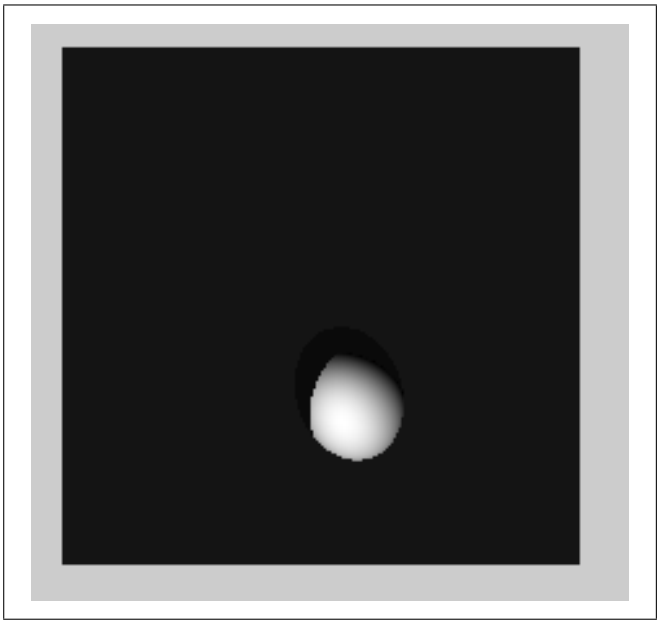

FIGURE 2. Circle A: translated from Figure 1 by the vector (50, -50, 0)

4. Discussion

Figures 1-8 were rendered such that when the cast ray intersected the sphere at two points, the farther of the two points is chosen; this accounts for the distinctly odd shading patterns, and the incredibly odd shading pattern of Figure 8 itself, the only glossy sphere of those.

Figures 9 and 10 demonstrate the reason for this: they are rendered with every visible point in shadow, so the brightness of every sphere point is set to half the ambient light. Therefore the mat2gray function in Matlab renders the ambient light as very bright and the shadow as very dark, there being no highlight to offset the contrast. I am not sure why the spheres are rendered so incorrectly.

It is apparent that spheres not centered in the field of view are skewed, as though painted upon a cone emanating out from the center of the view. I am not sure why this should be.

5. References

For the mathematics used to find points of intersection: Bourke, Paul. "Intersection of a Line and a Sphere (or circle)." N.p., Nov 1992. Web. 15 Dec 2010.

http://local.wasp.uwa.edu.au/ pbourke/geometry/sphereline/.

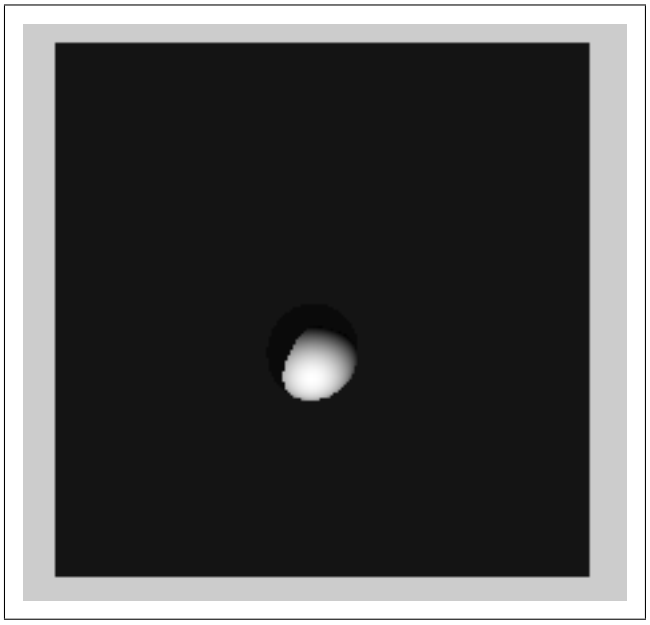

FIGURE 3. Circle A: view panned 15 degrees (right) and tilted -15 degrees (down) from Figure 2

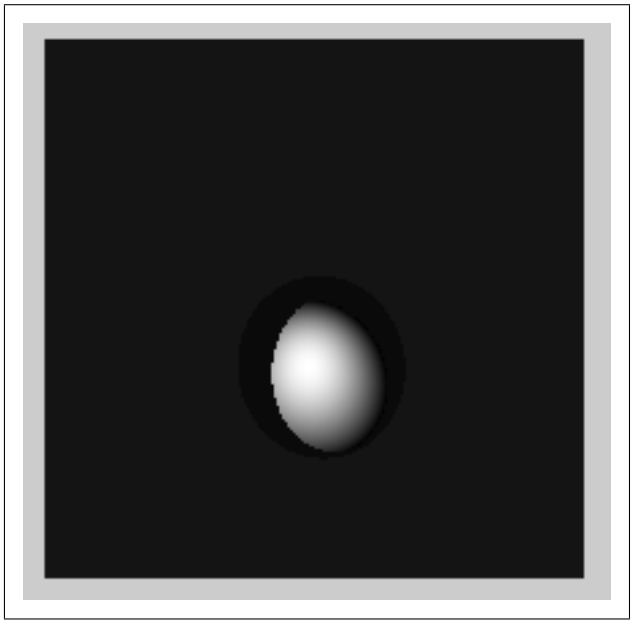

FIGURE 4. Circle A: view translated (zoomed) by the vector (0, 0, 100) Figure 3

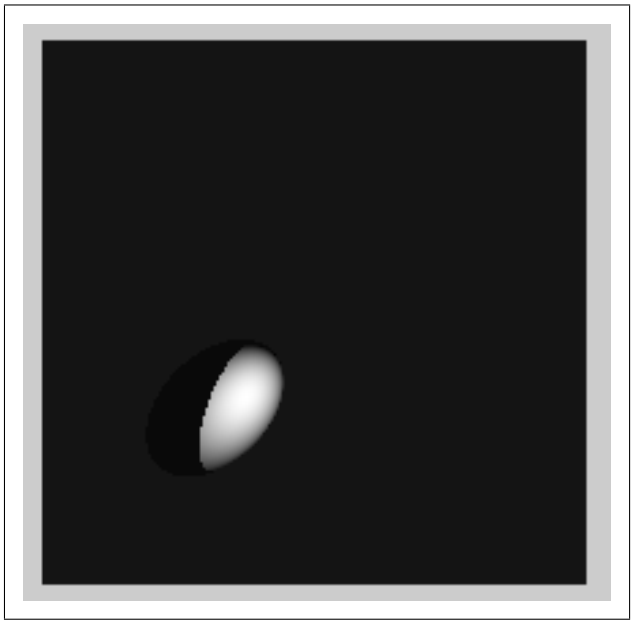

FIGURE 5. Circle A: view translated by (150, -150, 0) from Figure 4

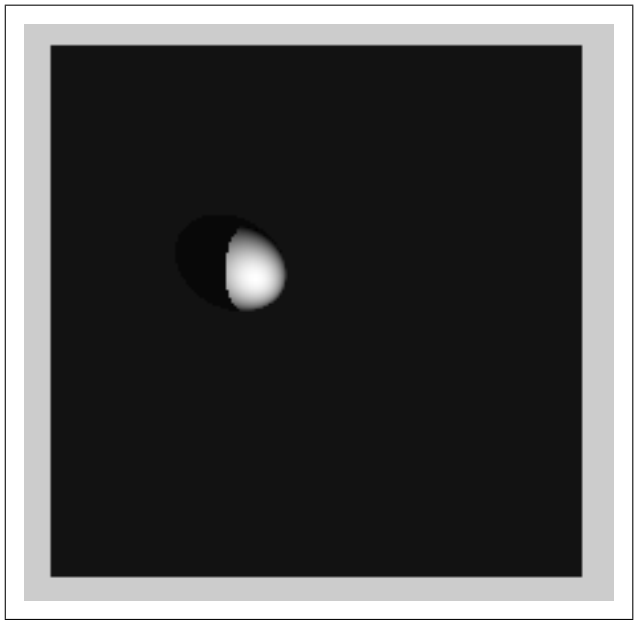

FIGURE 6. Circle A: view tilted by -50 degrees (down) from Figure 5

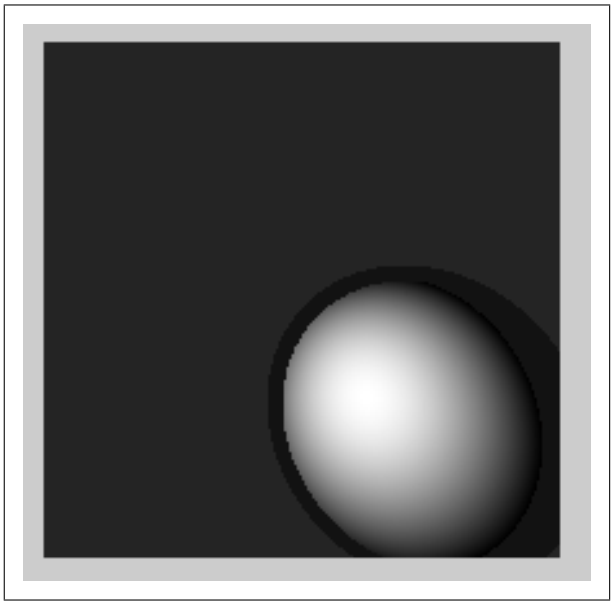

FIGURE 7. Circle B: a sphere with r = 100, centered at (75, 75, 150), viewed from (0, 0, -50), with the light coming from (-1, 5, 10). No gloss. Albedo = 6.

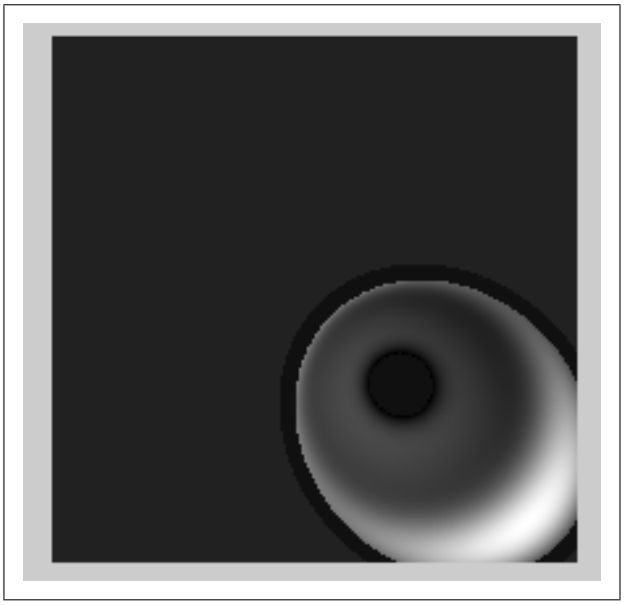

FIGURE 8. Circle C: a sphere with r = 100, centered at (75, 75, 150), viewed from (0, 0, -50), with the light coming from (-1, 5, 10). Gloss proportion = .7; gloss sharpness = 5. Albedo = 6.

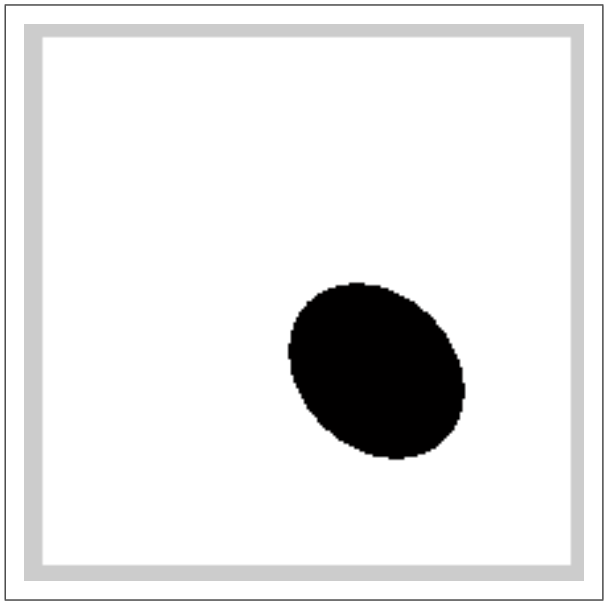

FIGURE 9. Circle D: a sphere with r = 100, centered at (75, 75, 150), viewed from (0, 0, -50), with the light coming from (-1, 5, 10). Gloss proportion = .7; gloss sharpness = 5. Albedo = 6.

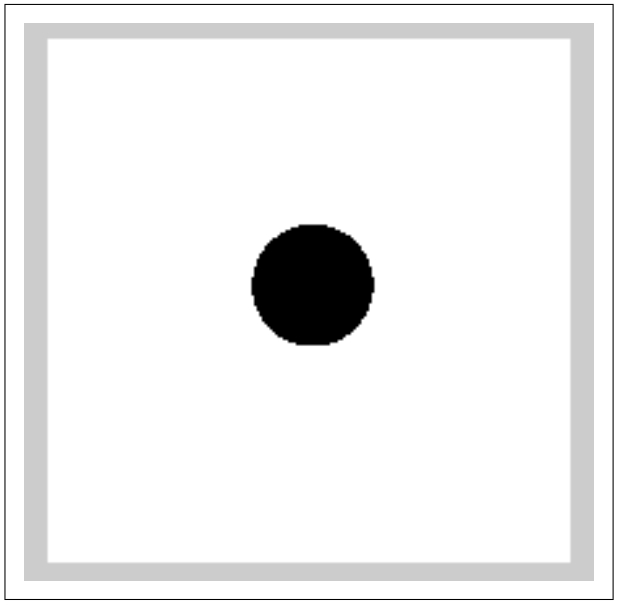

FIGURE 10. Circle D: view panned by 20 degrees (right) and tilted by -30 degrees (down) from Figure 9