Brendan Schneider
Xifeng Gao
CAP 5726
9/22/2020

Assignment 1 - Ray Tracing

Compilation and Implementation

All of my work ended up being done within main.cpp, so the project compilation should be identical to the instructions given in "general rules.pdf". Only the 6 required tasks were completed, and not the optional tasks.

Part 1 - Multiple Spheres

Part 1 gave me a lot of trouble, since it has been a while since I last used vectors. In the end I got it working fine, though, and added the ability to define up to 6 spheres (could do more easily since the limit is hard-coded). The light for this and all following scenes is located at (-ax, ay, az), where 'a' is the largest magnitude single coordinate on the surface of the spheres, plus 0.5. The size of the scene itself is similar, being a box with vertices at (+-ax, +-ay, +-az).

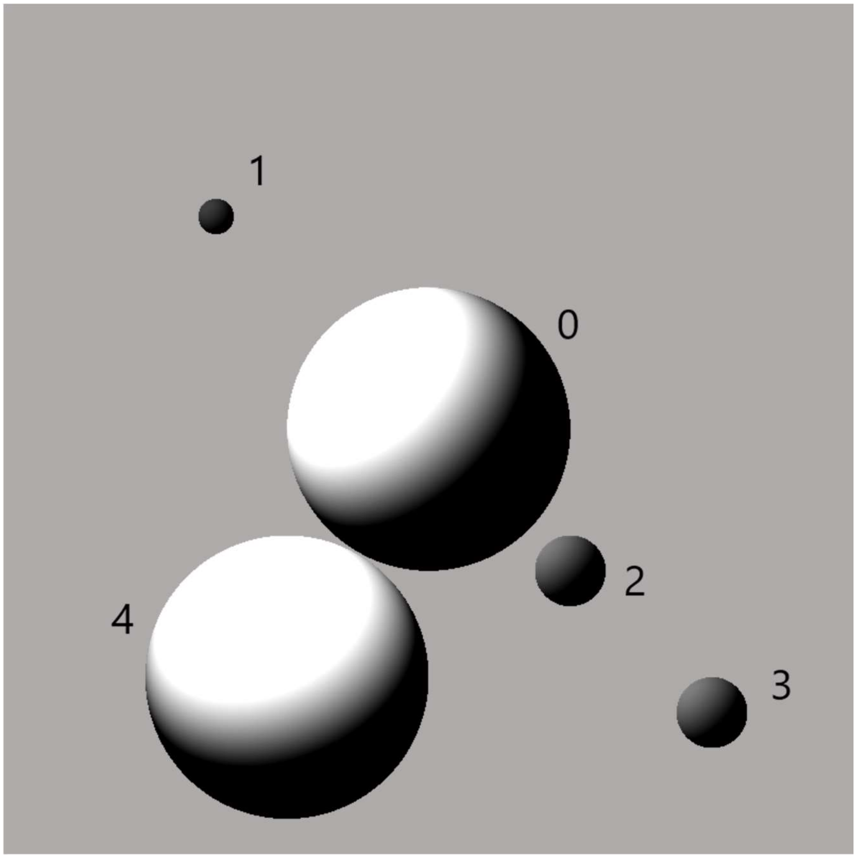

Figure 1

In figure 1, 5 spheres are defined as follows, identical to those spheres defined in parts 2, 3, 5, and 6:

Index	x	y	z	Radius
0	0	0	0	2
1	-3	3	3	0.25
2	2	-2	2	0.5
3	4	-4	0	0.5
4	-2	-3.5	-3.5	2

Part 2 - Shading and Color

The second light source for this and all future parts is located at (ax, -ay, 0), and it was given an intensity of 0.5 (compared to a default 1 for light source 1). The spheres were colored by simply rotating them through red, green, and blue for each new sphere (red, green, blue, red, etc.). Ambient lighting was given a coefficient of 0.25, while Lambertian shading was given a 0.5 and specular a 0.2. The Phong exponent for each sphere was given a value 1.5x that of the previous sphere. The difference in Phong exponent can clearly be seen by comparing sphere 0 to sphere 4, which has a much more “reflective” look.

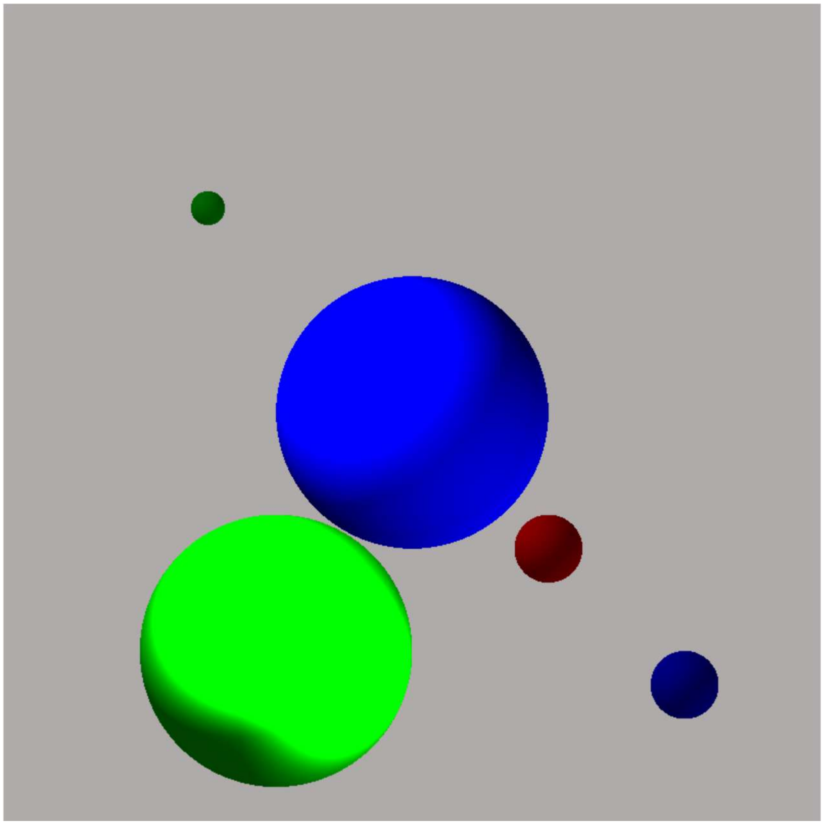

Figure 2

Part 3 - Perspective View

The camera for the scene's "perspective view was set to (0, 0, 2az), and the screen was set as the wall of the viewing box closest to the camera (the "positive z" wall). The results of this change can be seen in figure 3, where spheres 0 and 4 - which previously appeared to be next to one another - have a clear depth order to them.

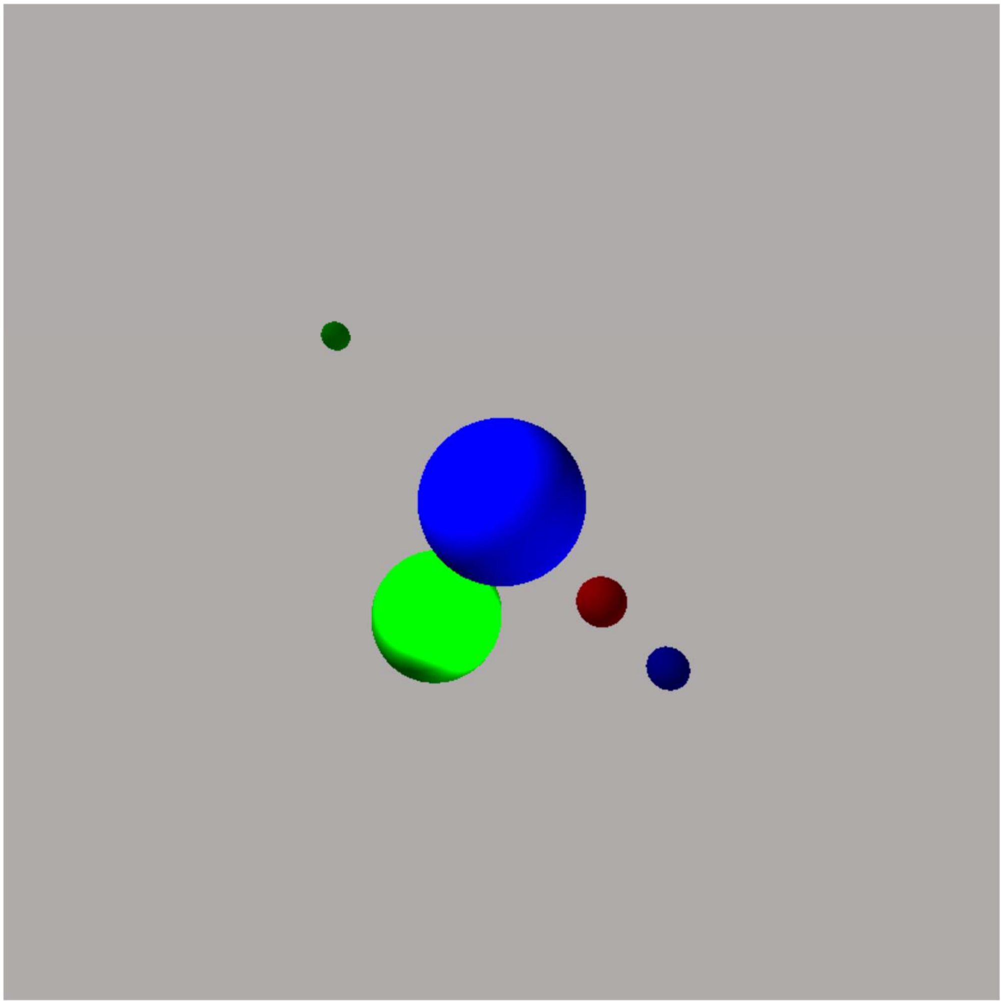

Figure 3

Part 4 - OFF Format Files

Not realizing that I had been working in debug mode made this part take way longer than it should have. As a result, this portion of the project is standalone and does not benefit from the shadow and mirror code in parts 5 and 6 (which are instead rendered with respect to the previous sphere scene). Figure 4 shows both given OFF files rendered side-by-side, with the bunny scaled up by 20 times.

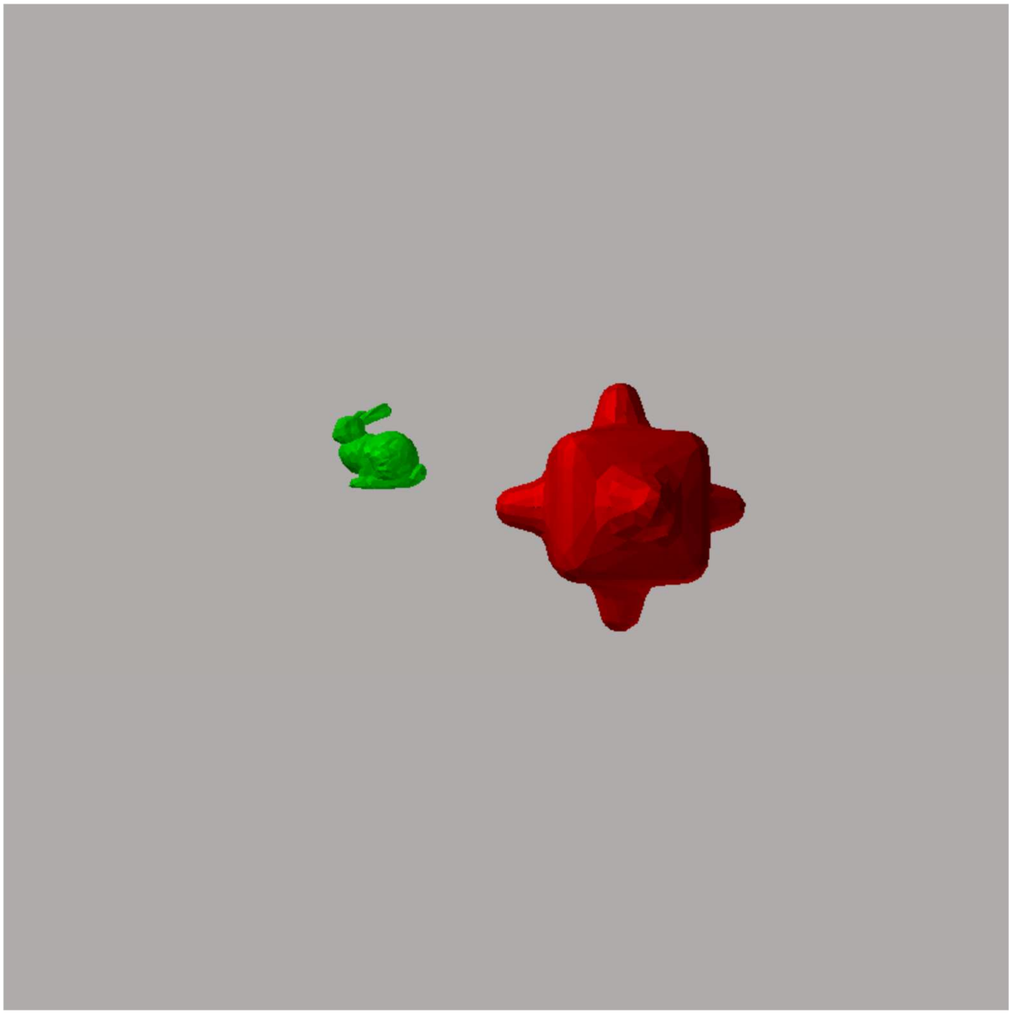

Figure 4

Part 5 - Shadows

Using the same light sources described above, figure 5 is the sphere scene re-rendered to show shadows. Spheres 1 and 3 in particular best demonstrate the shadows, which can be seen on sphere 0. A small amount of shadow can be seen cast from sphere 2 to sphere 3 as well.

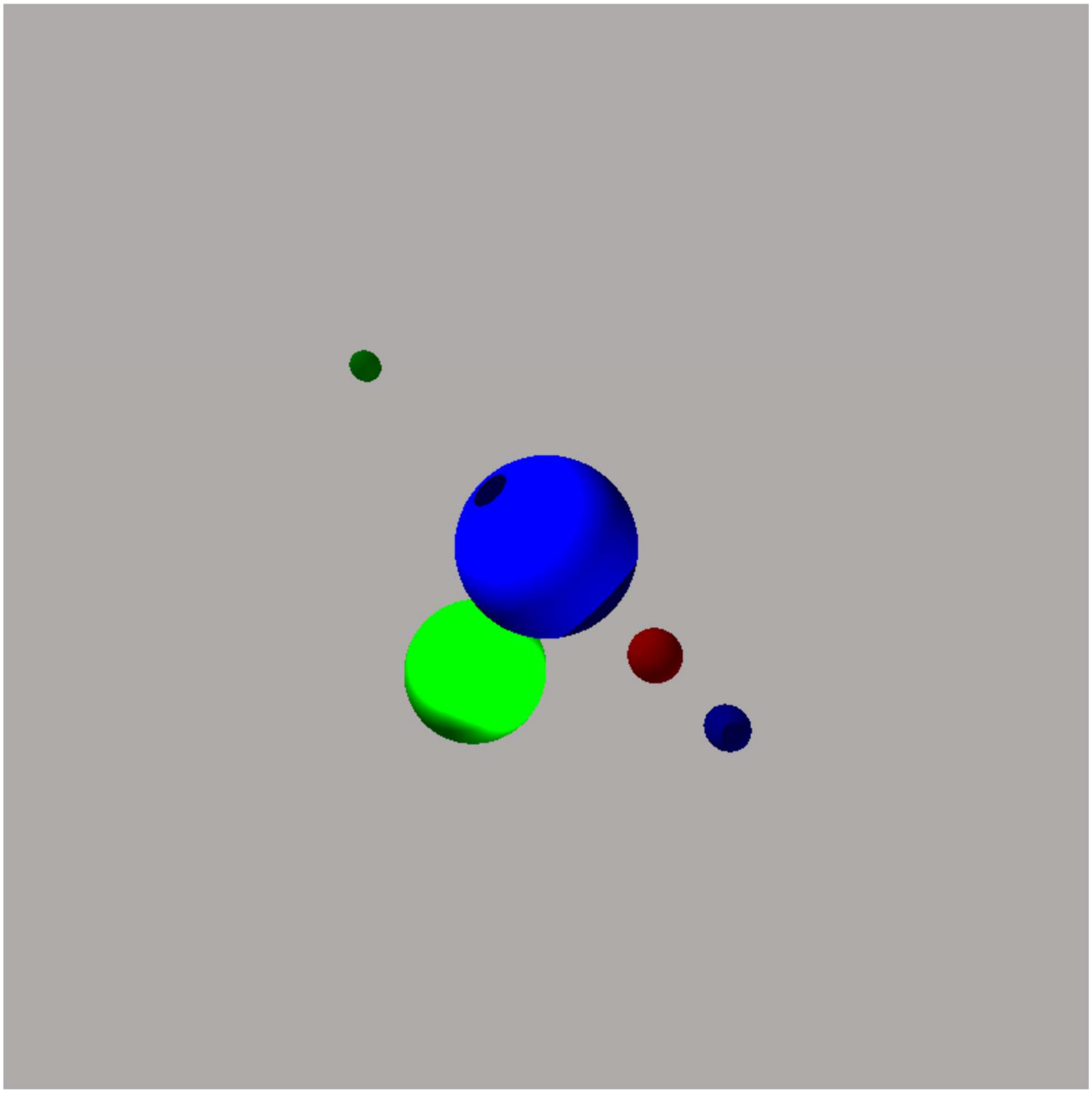

Figure 5

Part 6 - Mirroring

Finally, figure 6 shows the implementation of a mirror into the scene, created by mirroring the bottom of the viewing box. Several of the spheres can be seen mirrored across the bottom of the image, with sphere 4 being the clearest due to it being closest to the floor of the scene. This also gives a better perspective on the shadow cast onto the bottom of sphere 0.

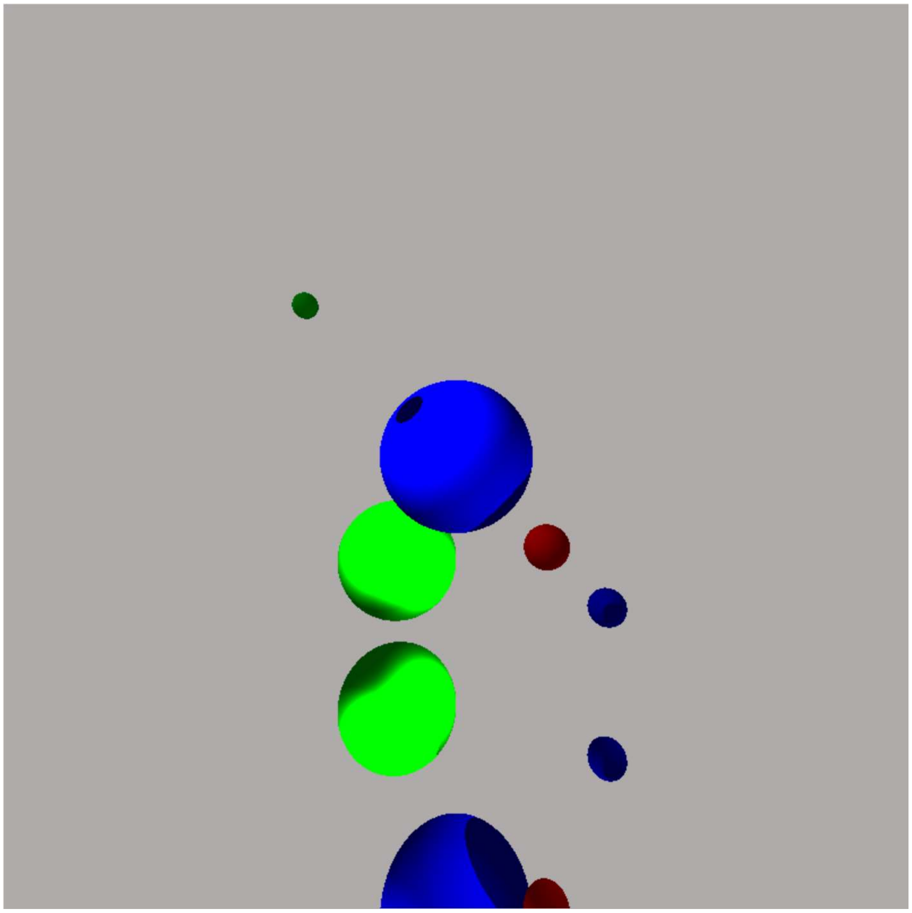

Figure 6