Cameron Churchwell
Cnc5166
CS351-2

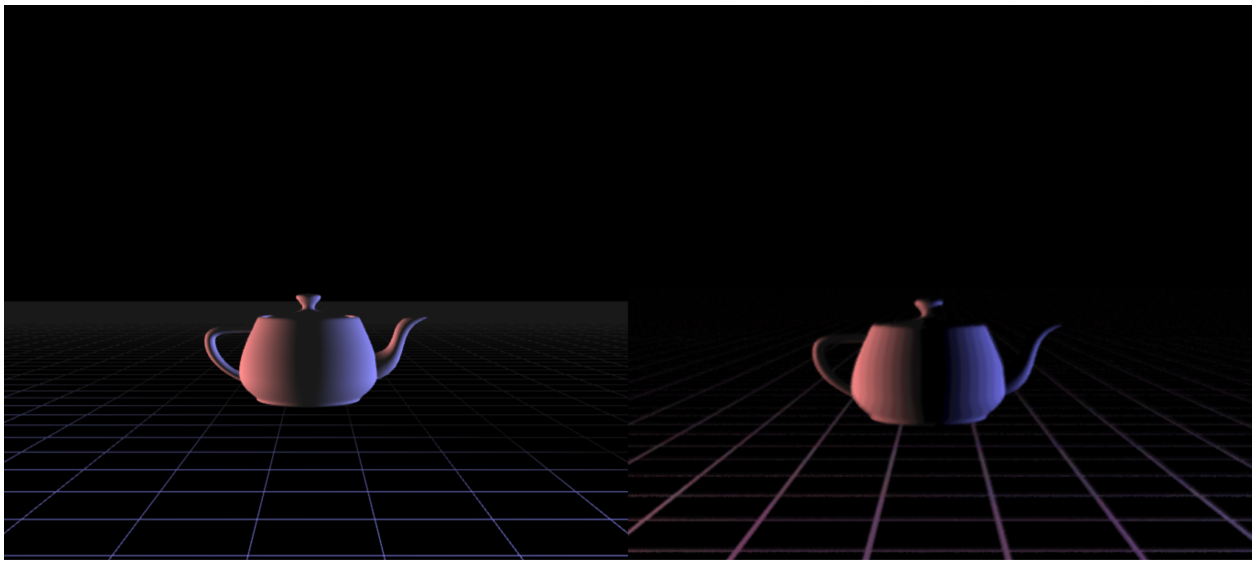
Project B: Ray Trace the World

With this project I really wanted to try to implement as general and capable of a ray tracer as possible. I also wrote almost all of the code from scratch, not using any starter code except very rarely as a reference. The majority of my codebase is written in TypeScript, with a small portion written in C which gets compiled down to WebAssembly. There are also shaders obviously written in GLSL. Vertex array objects are stored as JSON files. I use WebPack to bundle all of these into a single javascript file which gets included by the HTML file. The WebAssembly code uses SIMD to try to boost performance, however I only measured a very marginal performance from that. I considered rewriting again to use GPGPU to emulate a modern ray tracing pipeline, however due to lack of time and a lack of flexibility in even WebGL2, I did not follow through on this beyond writing a ray generation shader which is commented out in the main TypeScript file.

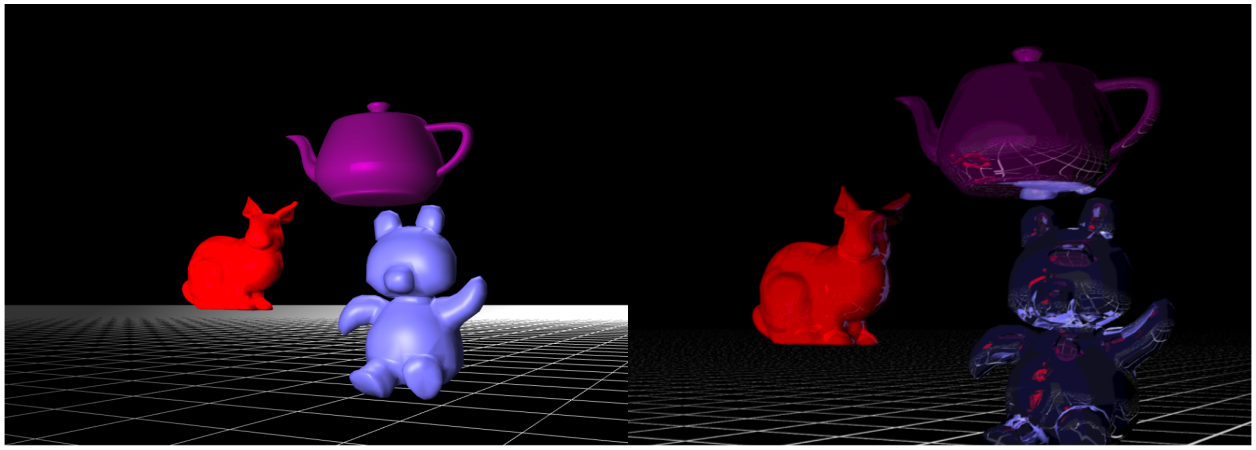
The primary features of my shader are the mesh tracing (see the second scene) and the Phong lighting I implemented in the WebGL preview. The mesh tracing also has Phong lighting with manually interpolated normals. It also is noteworthy that I ported all of the code to GLSL_ES version 300. It also was important to me that everything be as general as possible, so things like scenes, lights, and meshes are very easy to add (as opposed to something hardwired like the vboBox code).

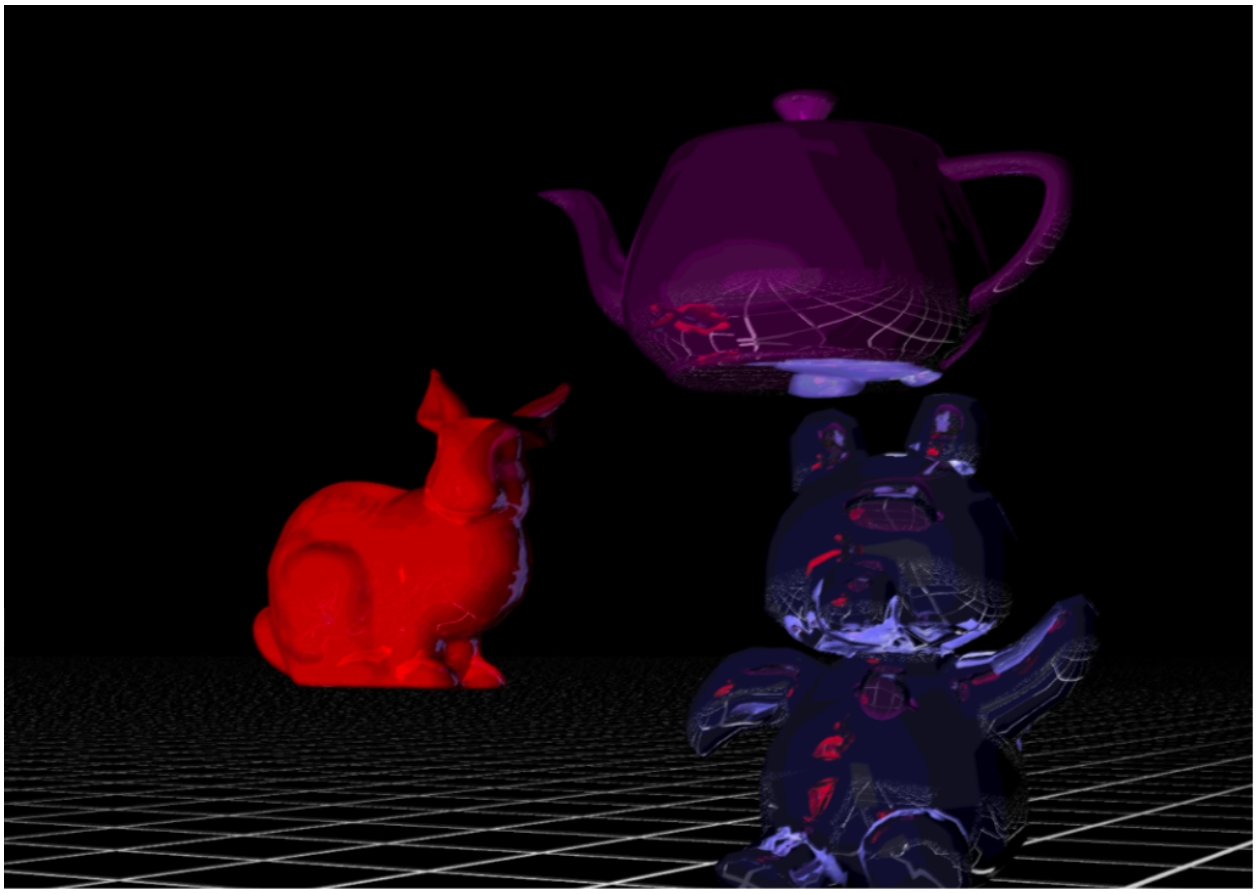
Users can move around using WASD and look around using arrow keys. The next most important key is T, which ray traces the scene. If you open the console you can see a kind of percentage remaining countdown for tracing, as some traces can take a while depending on hardware. M and N are used to increase and decrease the number of reflections respectively. P and O similarly increase and decrease supersampling. J iterates through jitter modes. There are radio buttons to select between scenes and checkboxes to toggle on and off lights. The window can also be resized.

In terms of optional items on the rubric, it's a bit up for interpretation. I added multiple new kinds of geometries: Disc, Triangle, Composite, Mesh, and BoundingSphere. I used the Mesh implementation to create the teapot, bunny, bear, etc. The majority of the work I did does not fall directly under any of the categories listed, but I believe that it is substantial. For example, the mesh geometries are automatically created as bounding volume hierarchies (using spheres) which exhibit tree-like structure to reduce the number of intersection checks necessary to be closer to logarithmic. This took a long time to implement but I learned so much doing this.

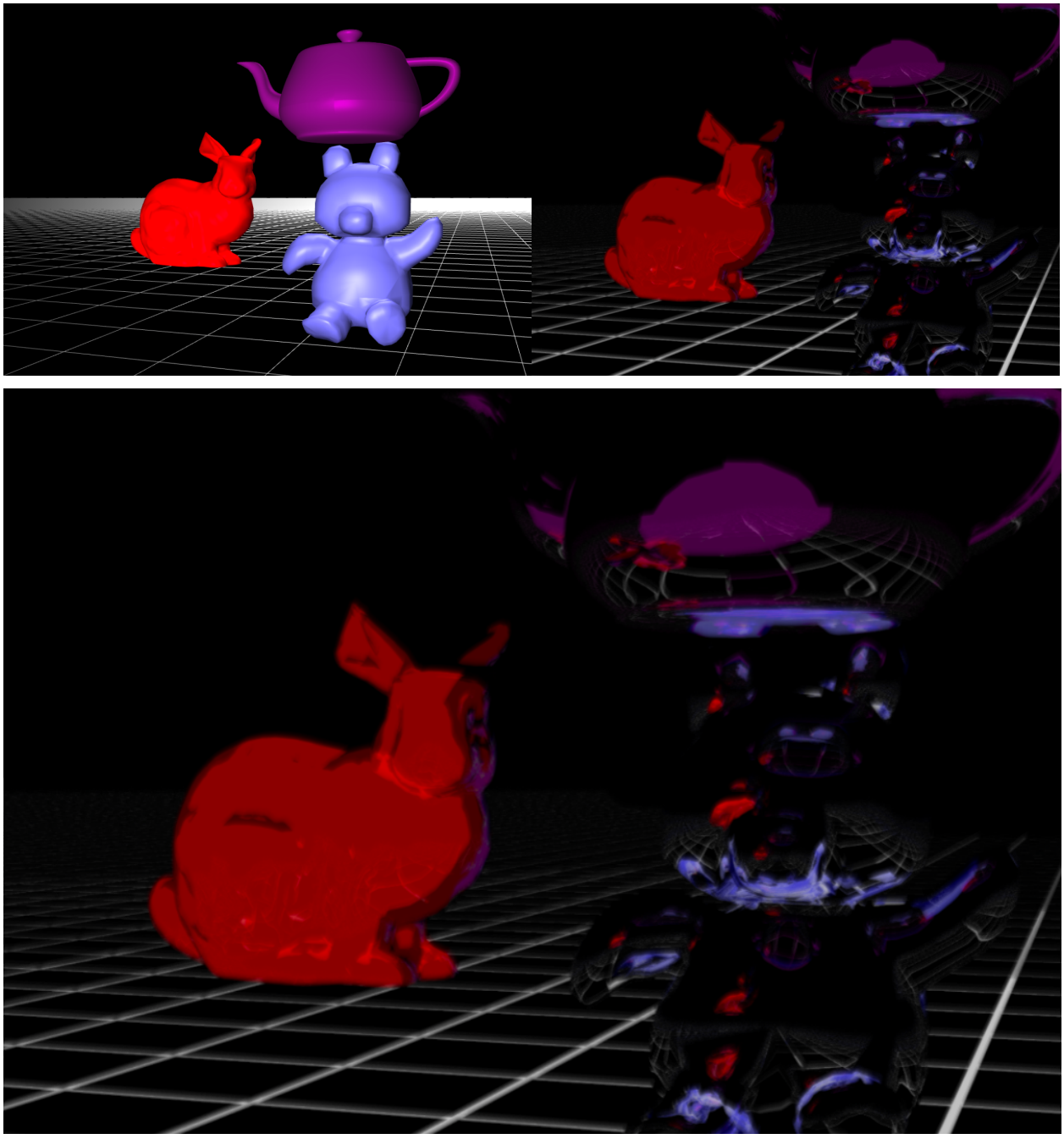


We'll start with a simple render. The teapot here is given a matte texture. You can see that in the Phong preview some blue appears on the handle however in the ray traced image this does not happen as the body of the teapot is occluding that geometry as one would expect, and casts a shadow.

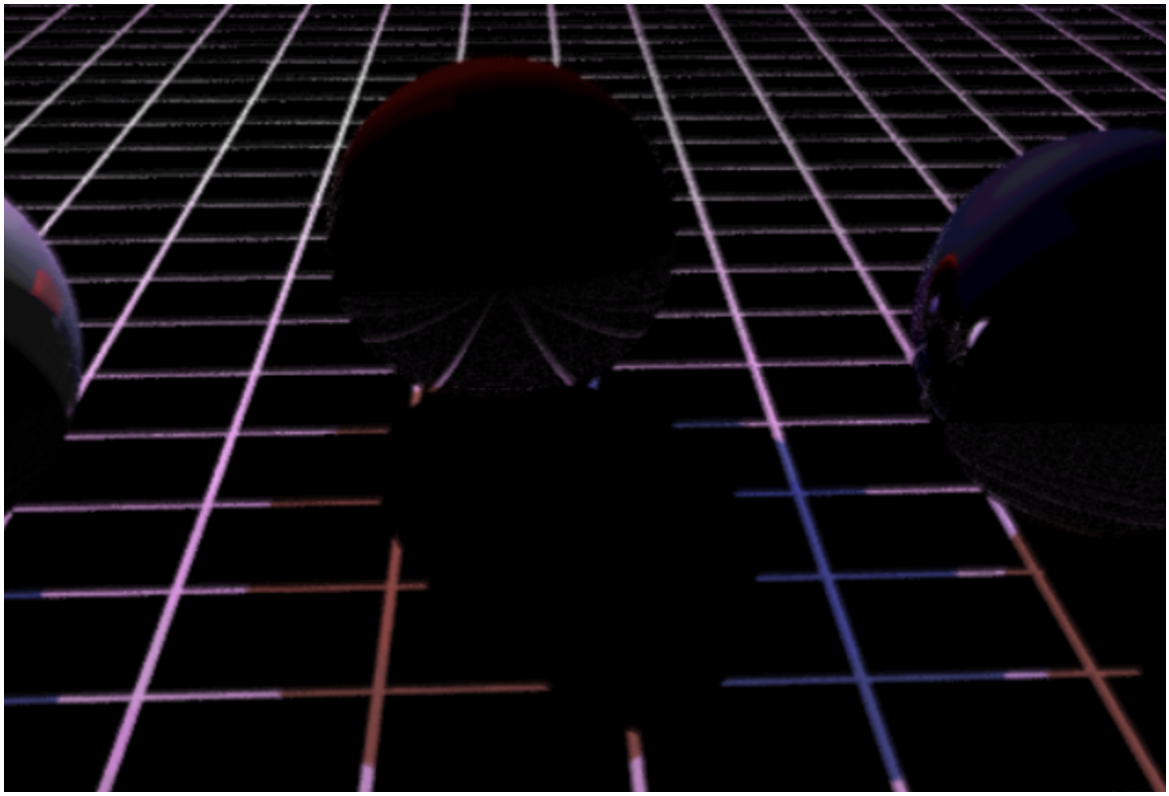




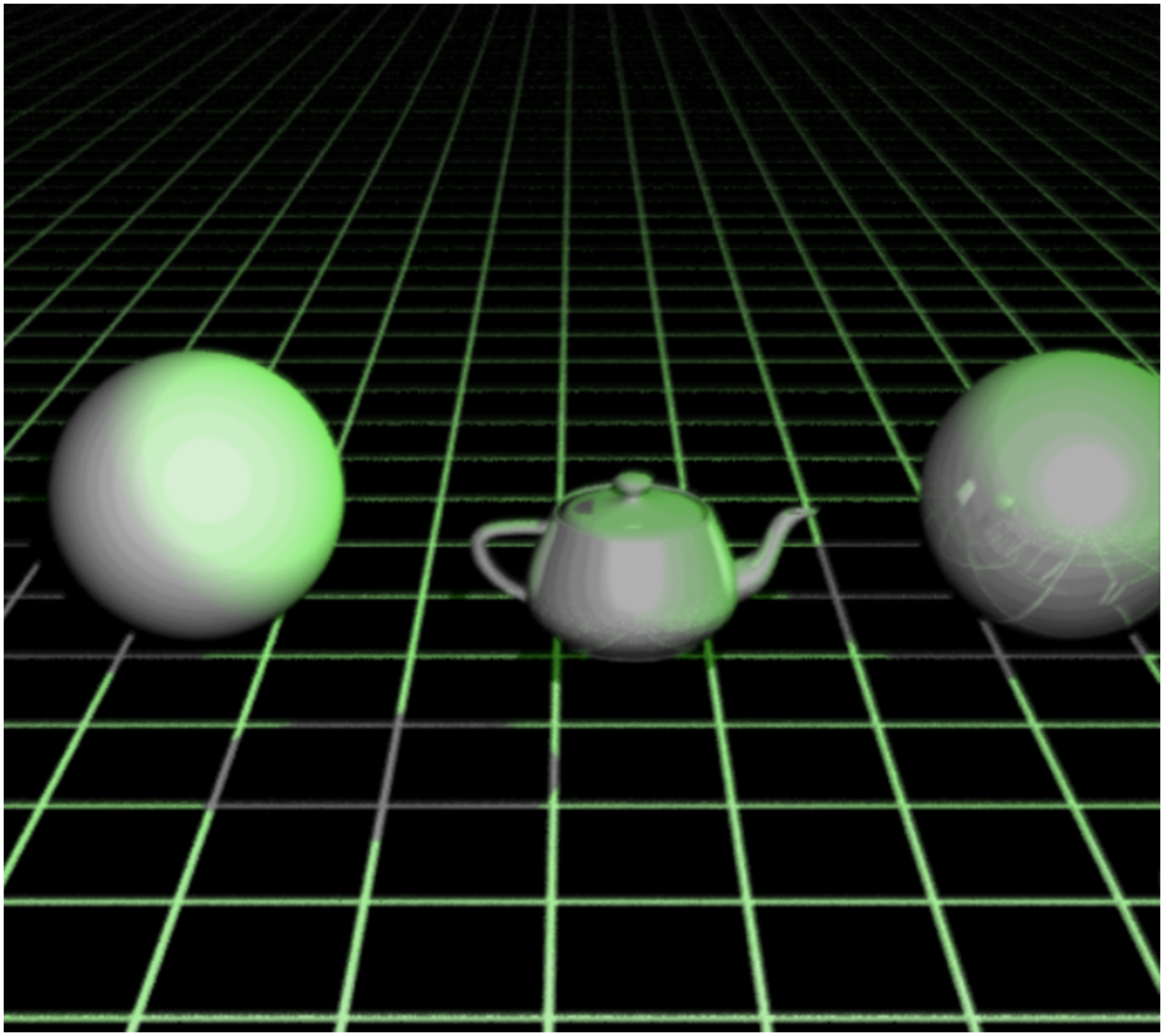
This render demonstrates tracing, Phong lighting, Phong materials, and mirror reflectance on three different mesh geometries. You can clearly see the reflection of the bunny in both the bear and the teapot, a reflection of the bear and ground grid in the teapot. You can even see a reflection of the ground on the teapot on the bear, in two different places (forehead and chest). This render was generated at 1024x1024 with no jitter and I believe 6x6 supersampling, and I think 3 levels of recursive reflection. At this high quality it took 48 minutes on my desktop CPU.



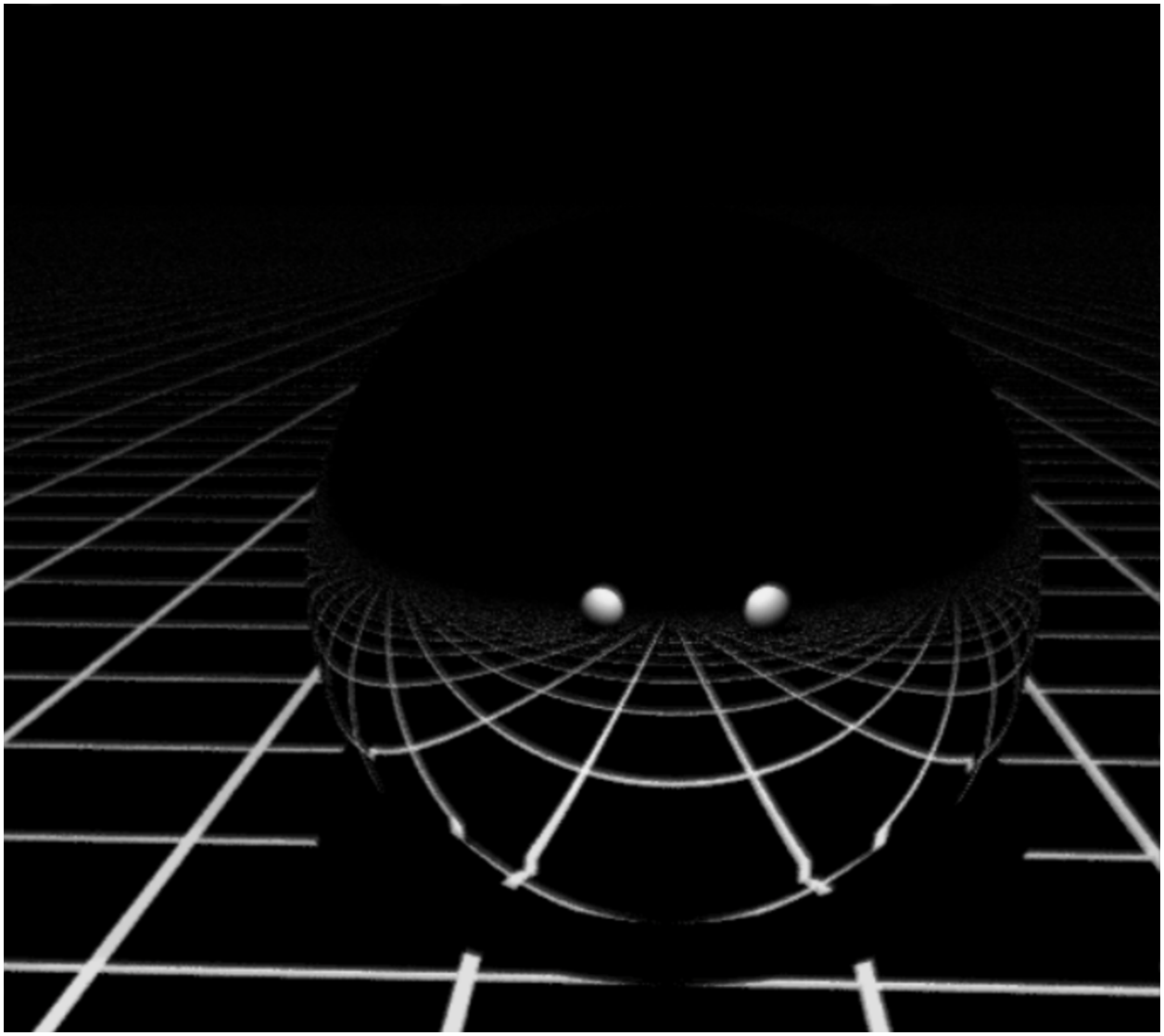
This is the same as the previous except with 8x8 supersampling and 4 reflections. This one took around 6 hours to render.



This image is great because it demonstrates the overlapping shadows of two colored lights behind a sphere.



Here I show that I added reflectance as a material parameter. The sphere on the left has a matte texture and no reflectance, which stands out starkly from the shere on the right. And yet you can see that both spheres appear as reflections on the teapot, showing that this property is disjoint across materials and objects as you would expect.



This render shows a perfectly reflective sphere.