RENDERTIMES:

Image	**Time**	**Number of Rays**
Reflective (depth of 1, 2 reflective spheres)	12.692 seconds	1,073,317
Reflective (depth of 8, 2 reflective spheres)	12.557 seconds	1,075,366
Transparent Spheres (depth of 8, 2 transparent spheres with different refractive indexes)	9.6789 seconds	1,586,743

HARDWARE:
I ran this on my personal computer using Ubuntu 18.04. It's a Dell XPS 15 from 2016 with the following CPU: Intel® Core™ i5-6300HQ CPU @ 2.30GHz - Quad-core, an x86 64-bit processor. My ray tracer is a CPU ray tracer with only 1 thread.

LANGUAGES:
I used C++ for everything. Specifically, I am using clang++6.0 as the compiler. I used Python for MP1, but it became far too slow, so I switched to C++. It'd be interesting to see if PyCUDA and/or numpy could be used to make a reasonably fast Python ray-tracer… but I'm a bit swamped this semester :(

CODE CITATIONS:

Nearly all of the code I wrote in the ray tracer was heavily taken from the Ray Tracing From the Ground Up book by Kevin Suffern. However, I re-typed all of this code by hand / line-by-line. I did this to understand the code better and make it more similar to my coding style / deleted things I thought were unnecessary. I also used a few other websites for topics that were not covered well in Suffern's book.

Most of the code: Ray Tracing From the Ground Up: Kevin Suffern
Most of the code (Fork of Kevin Suffern's code):
https://github.com/tfiner/RTFGU/blob/f1d6c164afeb8bd38b74171582b06d9d30cc7e25/src/geo_obj/BBox.h
OBJ Parser: https://stackoverflow.com/questions/21120699/c-obj-file-parser
BVH: https://graphics.stanford.edu/~boulos/papers/efficient_notes.pdf

BVH:
https://www.scratchapixel.com/lessons/advanced-rendering/introduction-acceleration-structure/bounding-volume-hierarchy-BVH-part1