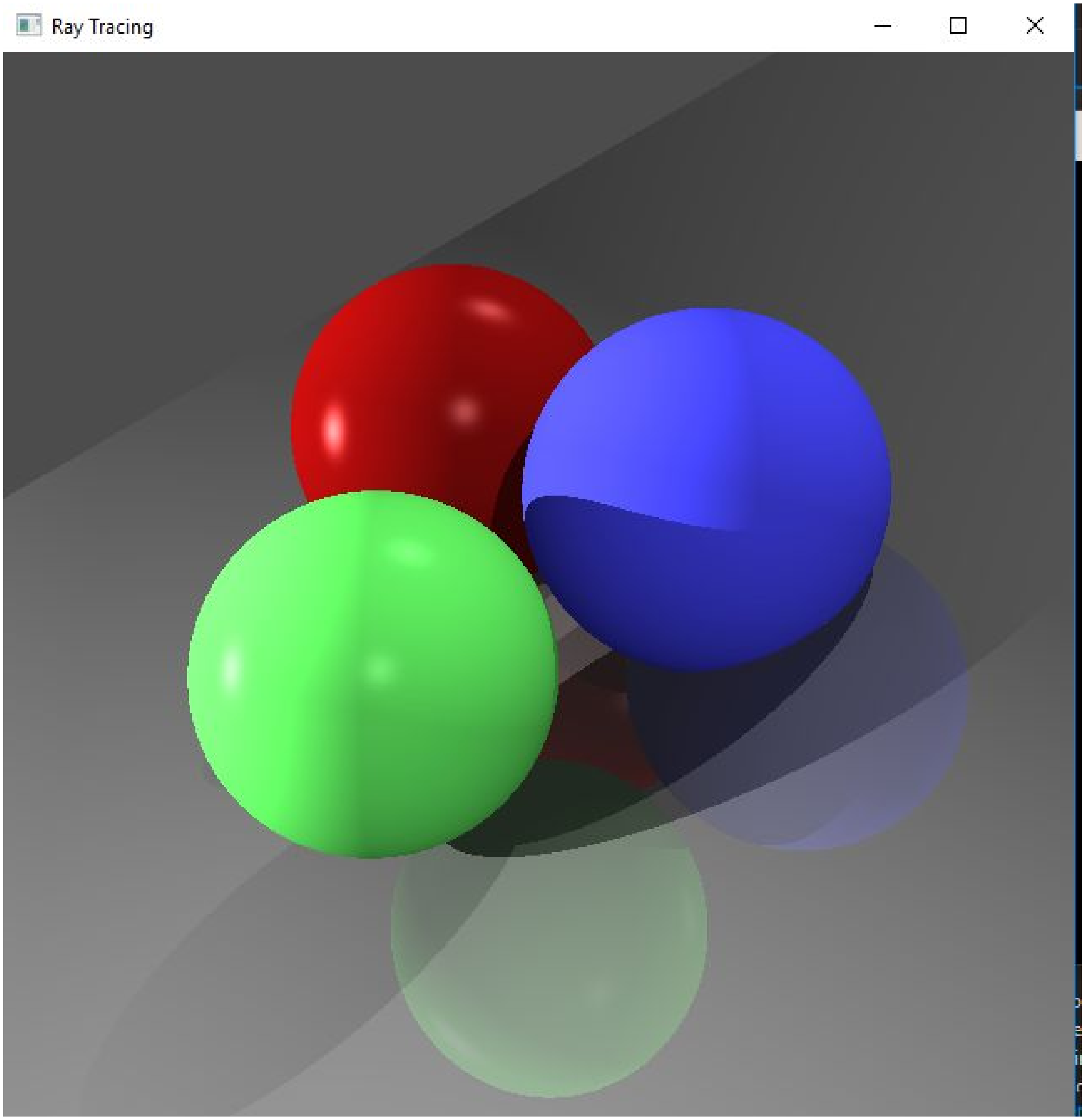
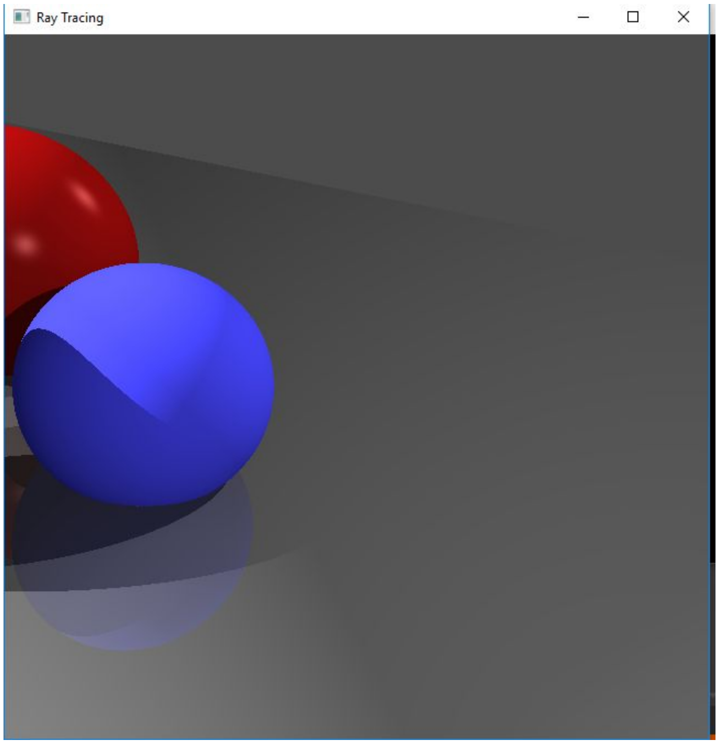
Feature #1: BVH

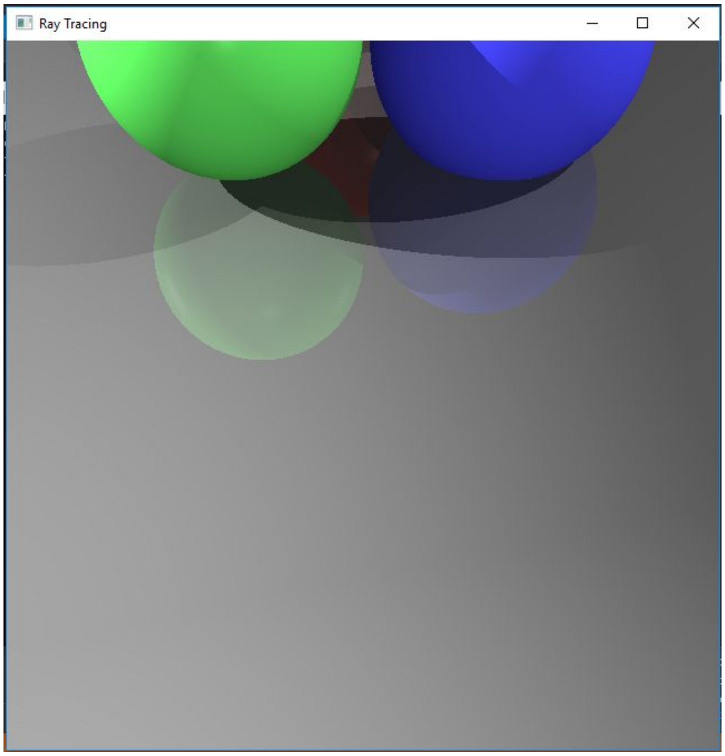
- An acceleration structure that speeds up the ray tracing process.
- Works fine, run scene bvh.json (or just click run since it is the default scene) to check out the result. Render time: ~12 sec (200 spheres)
- Checkout the Appendix section below to the resulting image.

Feature #2: Camera

- Able to view objects in different views.
- Works fine, I’ve made some fixed camera position. To select them, go to q1.cpp, set bool cam = true (line 38) then select the camera position from 0 to 3, then run the solution to see the result.



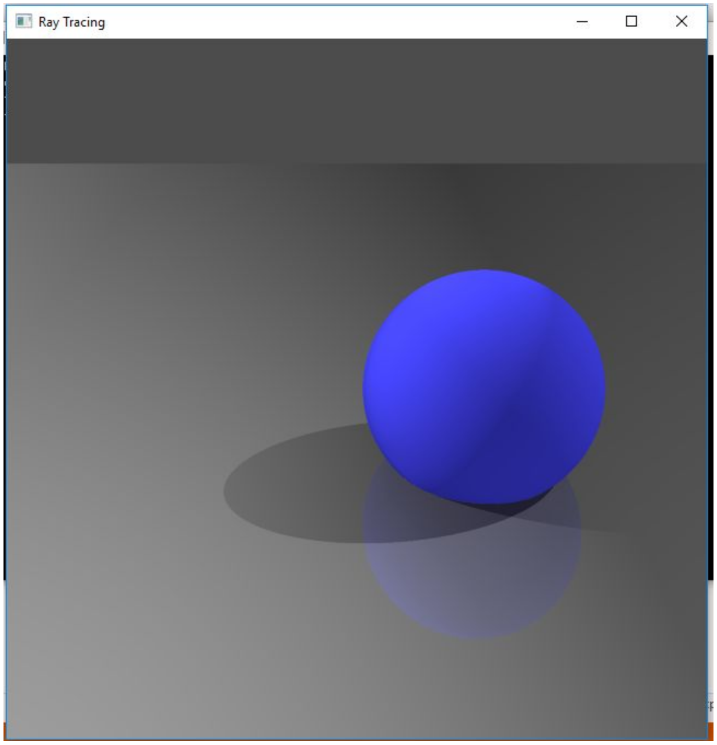


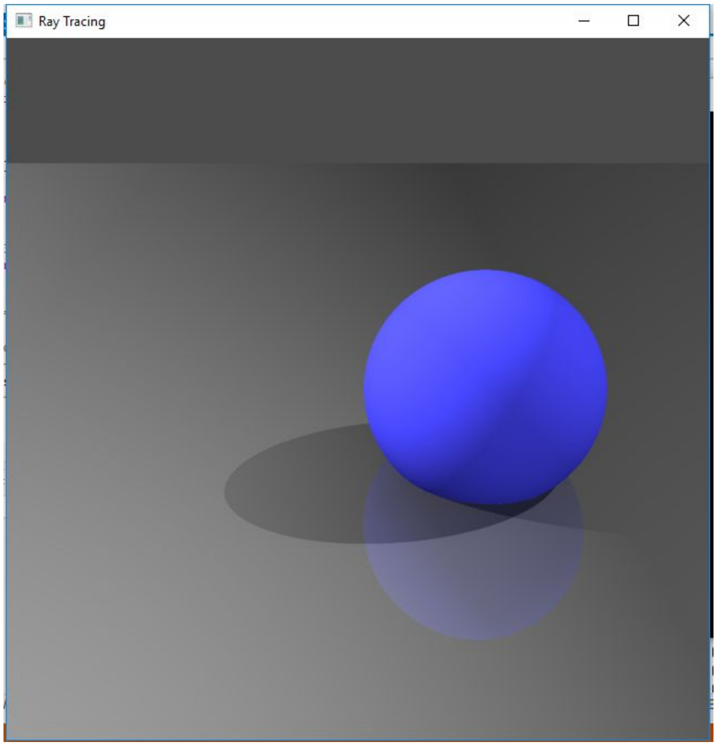


Render time: around ~10 seconds (no antialiasing)

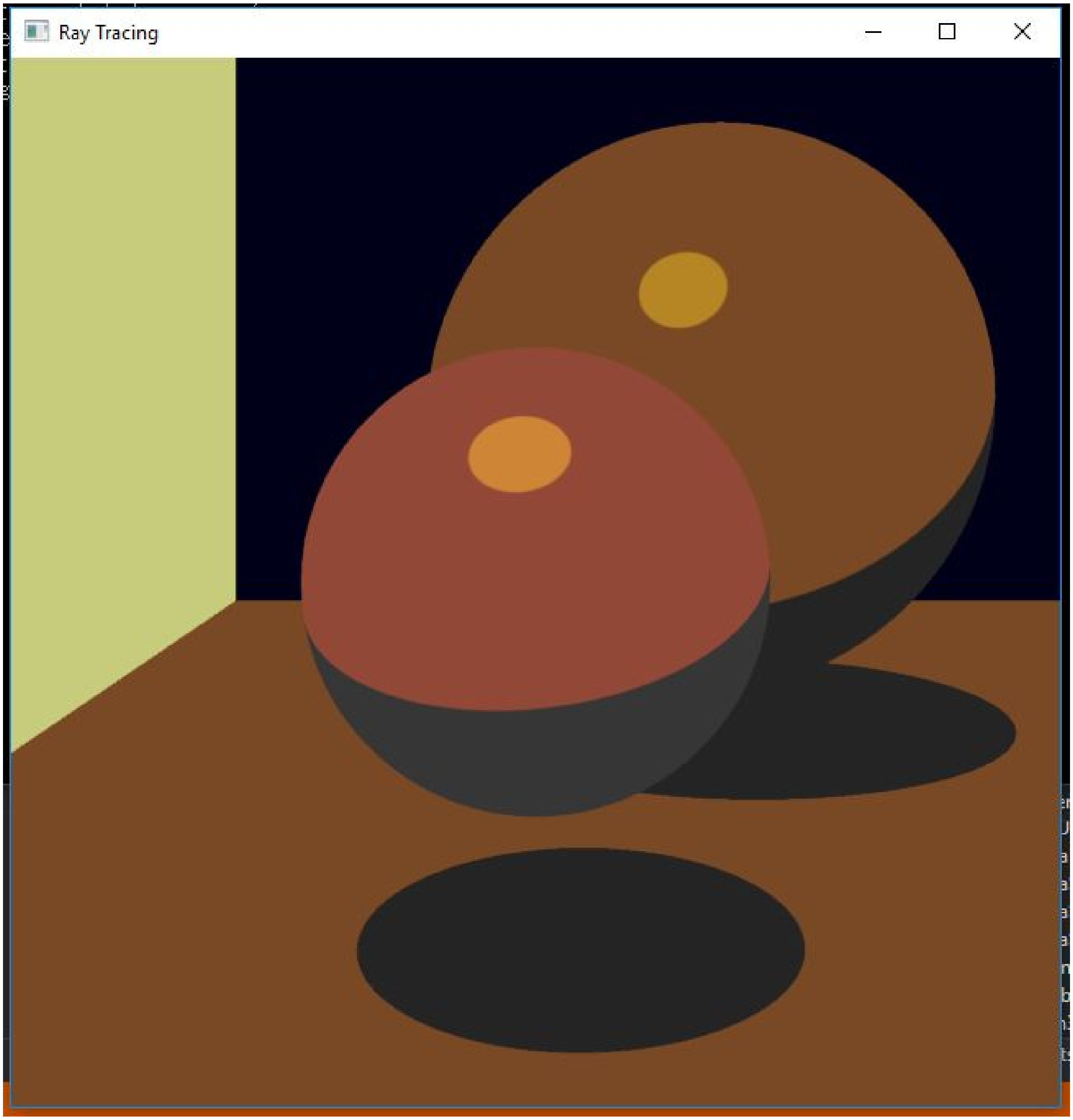
Feature #3: Improved Quality

- **Oren - Nayar diffuse reflectance:**
  - Used Oren-Nayar diffuse lighting to get better colours on rough surface objects.
  - Works fine, looks good
  - To have Oren - Nayar diffuse, an object must have a variable “oren” : {amount} in the material
  - Render time: ~ 20 seconds (with antialiasing)
  - External source: https://en.wikipedia.org/wiki/Oren%E2%80%93Nayar_reflectance_model
  - Oren Nayar vs Lambertian Diffuse (scene o.json):



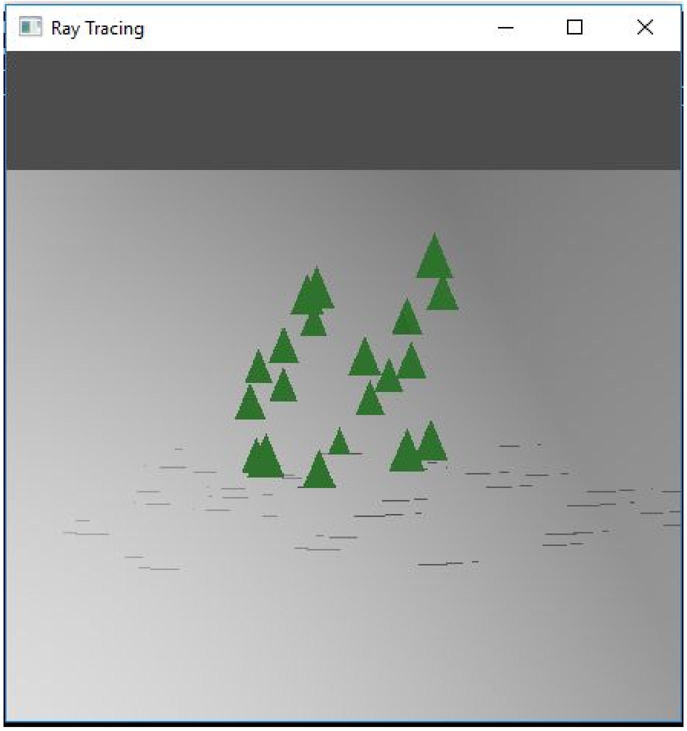


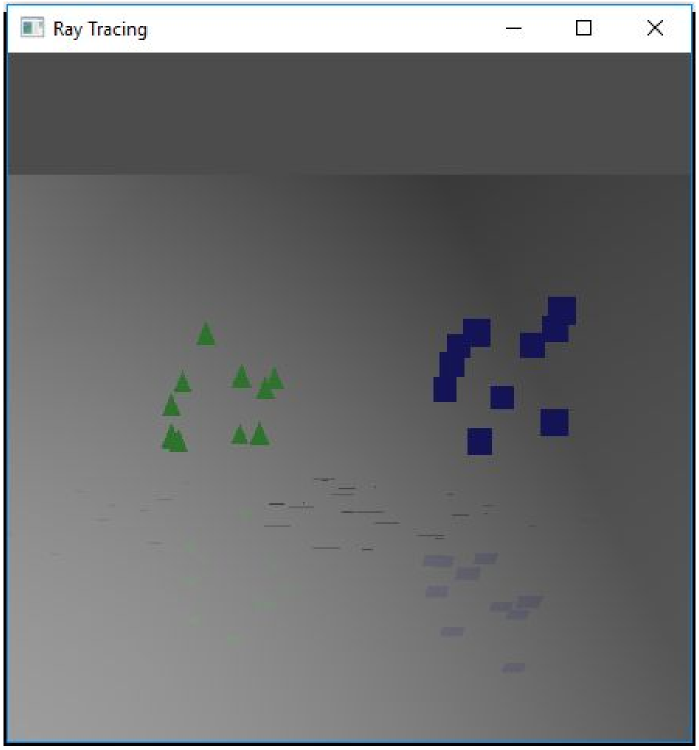
- **Toon shading:**
    - Rendering technique that makes 3D objects looks 2D (this was used in Legend of Zelda: BotW)
    - Looks good to me
    - To run, selects scene s.json then run it. To have toon shading on, in the json file, an object must have a variable “toon” : true in the material
    - External source: https://roystan.net/articles/toon-shader.html
    - Render time: ~28 sec (with antialiasing) (scene s.json)



Feature #4: Particle System

- A fixed state of a particle system, where particles are generated randomly and spread throughout a center point. It could have looked better if I used a more complex mesh for a single particle.

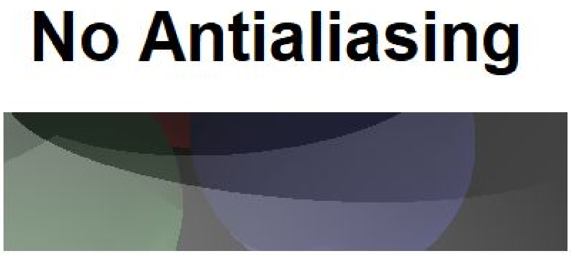


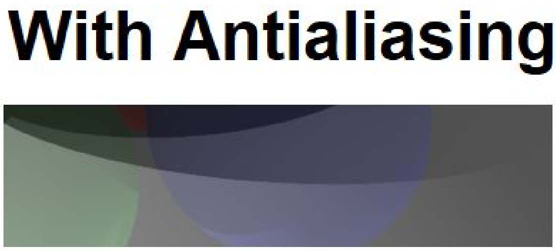


Render time: <10 seconds (this also depends on the number of particles) (scene p.json, check how this json file to see how particle is implemented)

Feature #5: Techniques

- **Antialiasing**: smoothing the edges of objects. Work perfectly

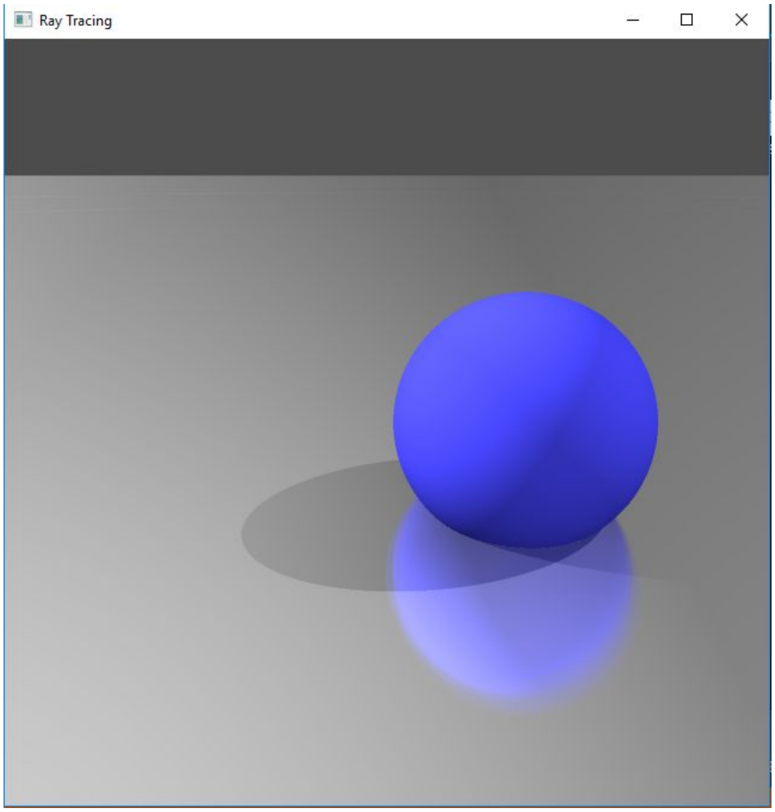


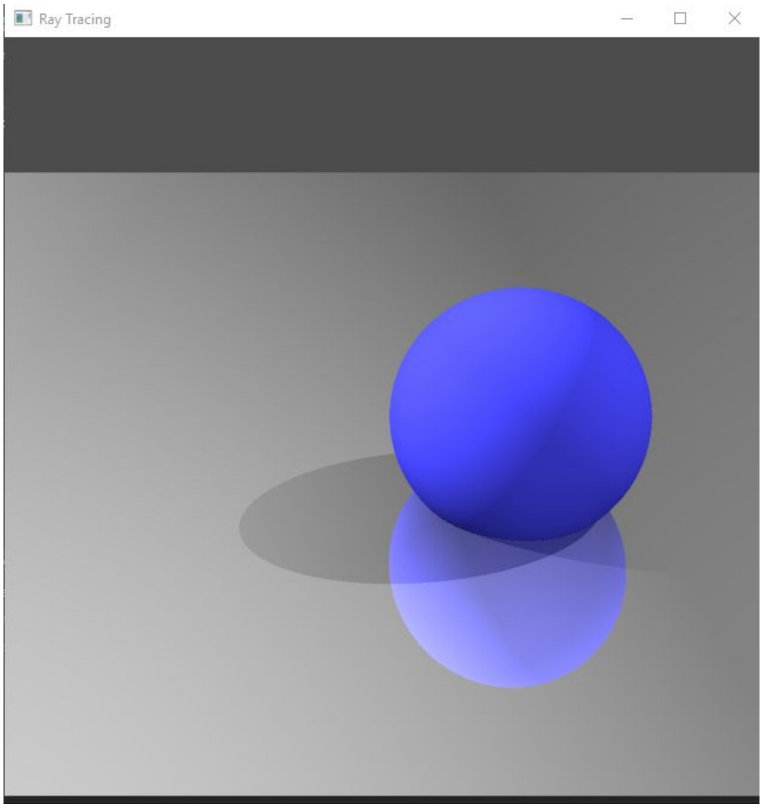


- To enable: go to q1.cpp, set bool antialiasing = true (line 33) then run

- Several images above already have antialiasing, the particles scenes and camera scenes didn't have antialiasing so you can clearly see the difference based on the shadow and the spheres.
- For image and time comparison, it will be in the appendix (down in the last section of the report)

- **Glossy Reflection:** added glossy reflection of a rough surface. It looks good to me.
- With and without glossy (scene t.json):



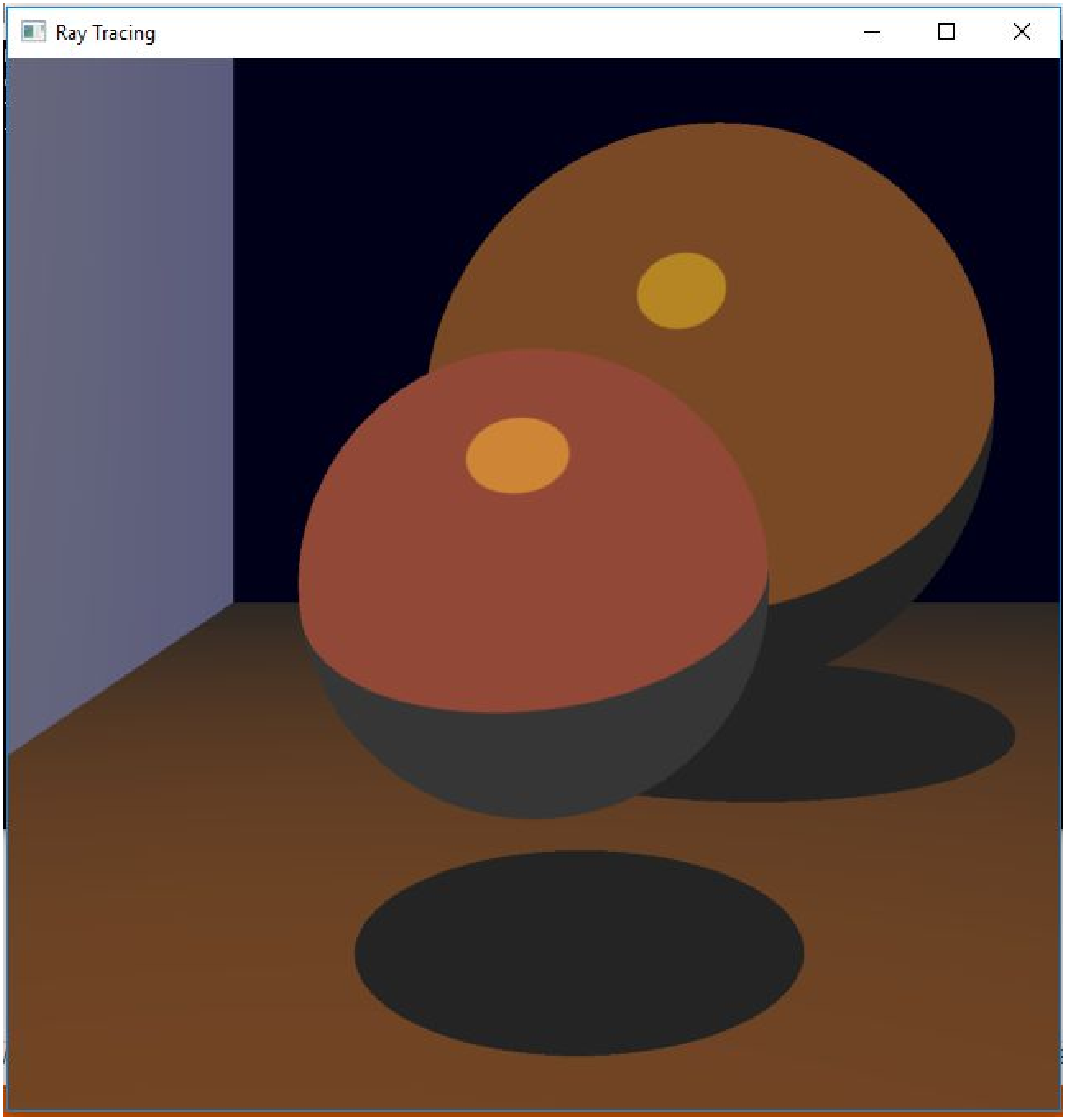


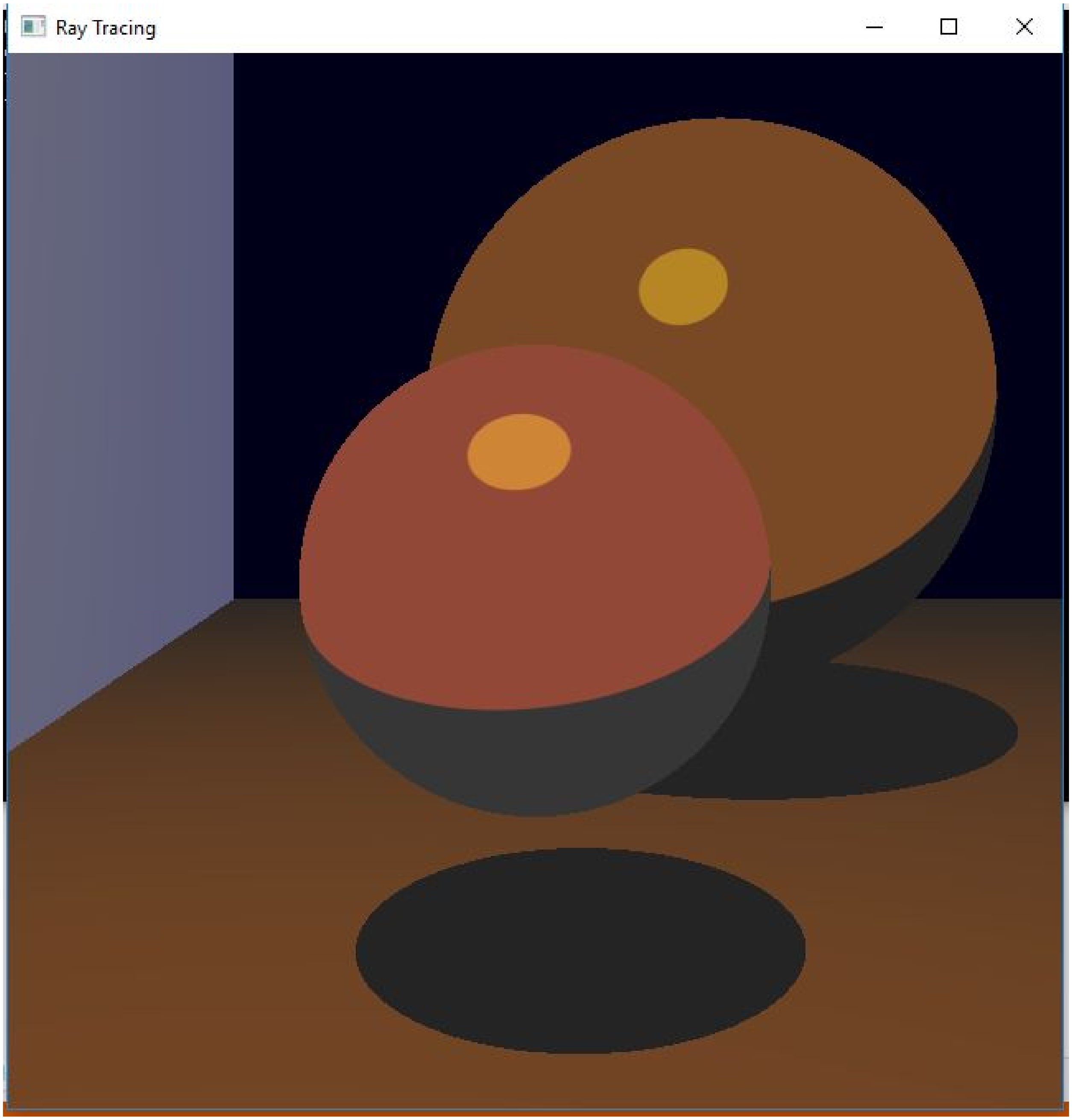
- To enable, in json file, an object must have a variable "glossy" : {amount} in the material
- Render time: ~2 min (this depends on the number of rays that I shoot, I have ~30 rays so it takes a while, also this is with antialiasing)

APPENDIX:

Since some effects need to be seen in full resolution to see the difference, I will put some pictures in here in full res.

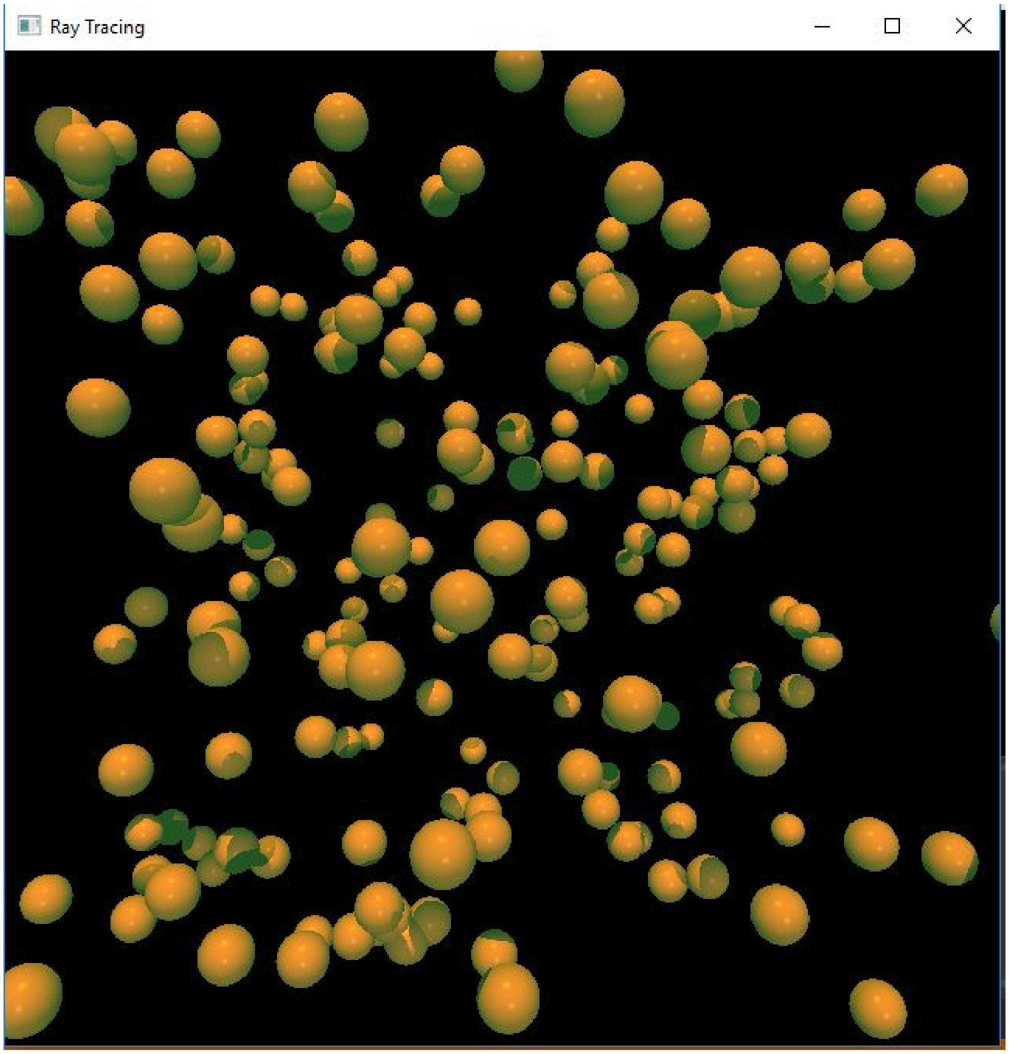
With/Without Anti-aliasing:





- Render time: ~28 sec for antialiasing and 7.5 sec for non-antialiasing

BVH:



Render time: ~12 seconds (without anti-aliasing, 200 objects)

Interesting fact:

- In the glossy scene, it takes 2 min with anti-aliasing, however it only takes 26 seconds without anti-aliasing.

Reference:

- Only for Oren-Nayar and Toon shading that I use external sources, all other techniques are from the video lectures.