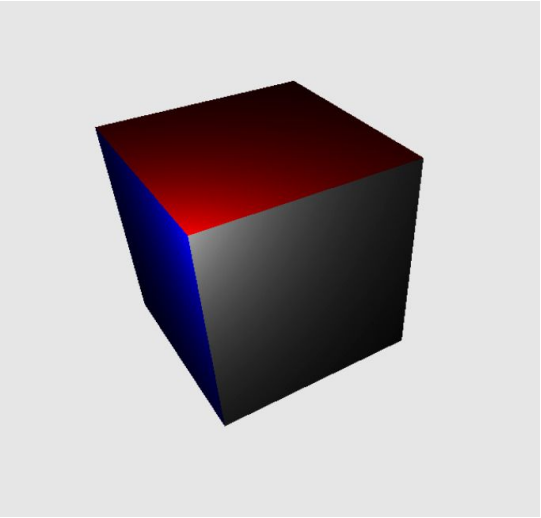
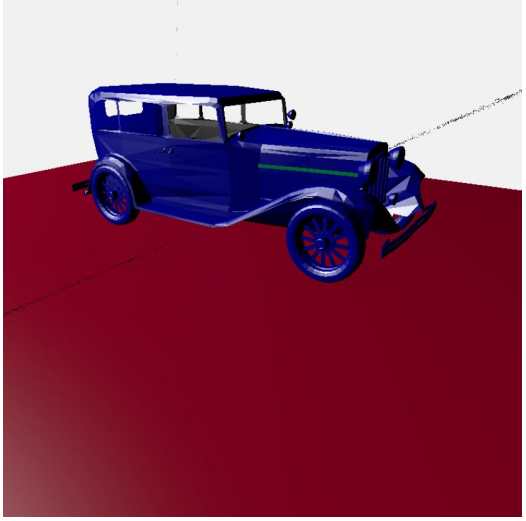
Ray Tracing Project Report - Group 42

Minimum Features

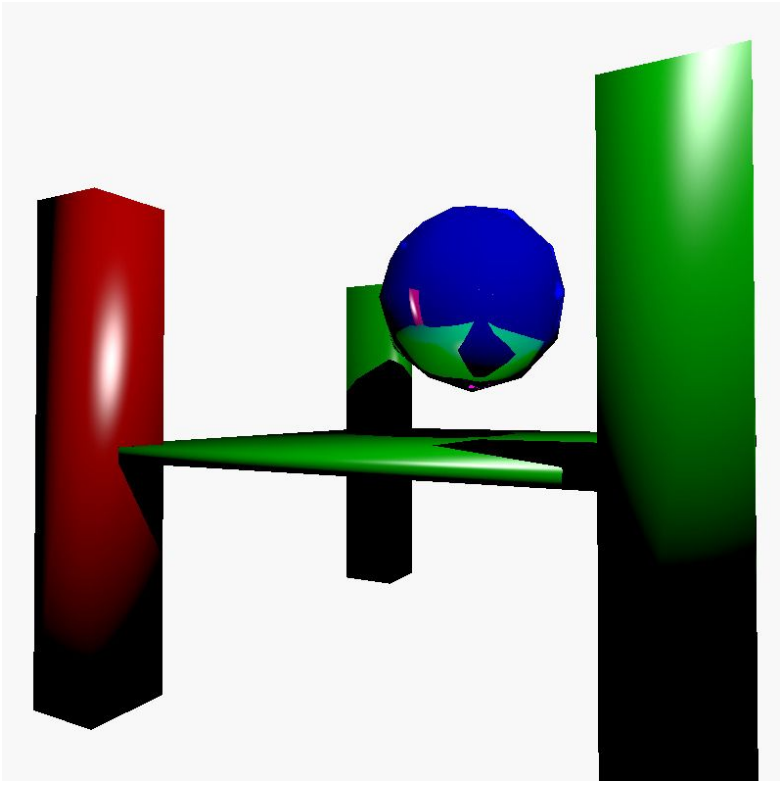
- Perform ray intersections with planes, triangles, and bounding boxes.
  Martin, Ivaylo, Rohan, Jakub

- Computate shading at the first impact point (diffuse and specular).
  Martin



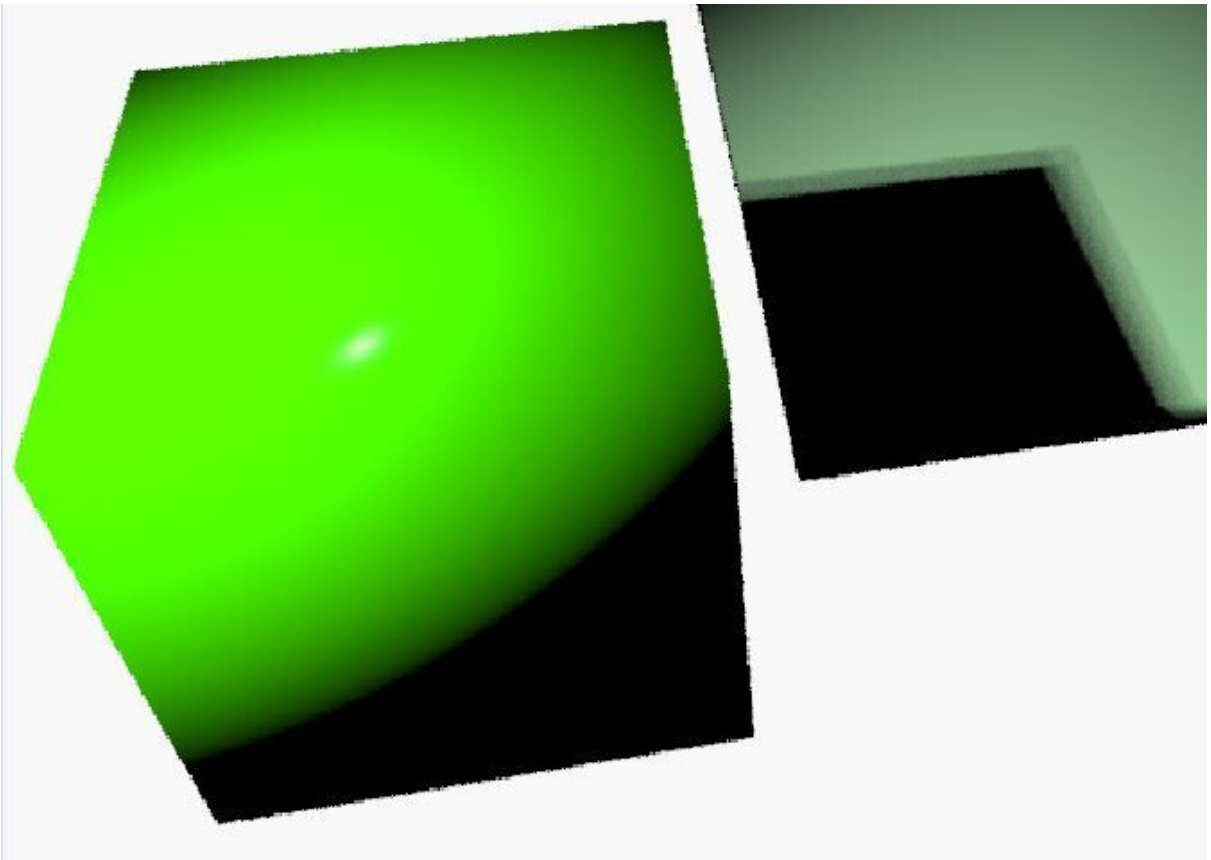


- Perform recursive ray tracing for reflections to simulate specular materials.
  Ivaylo, Martin

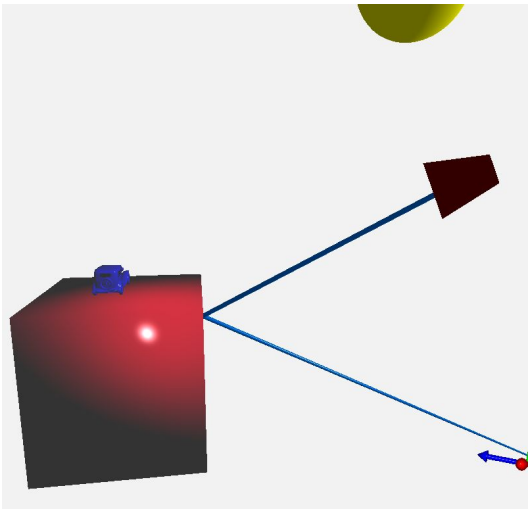


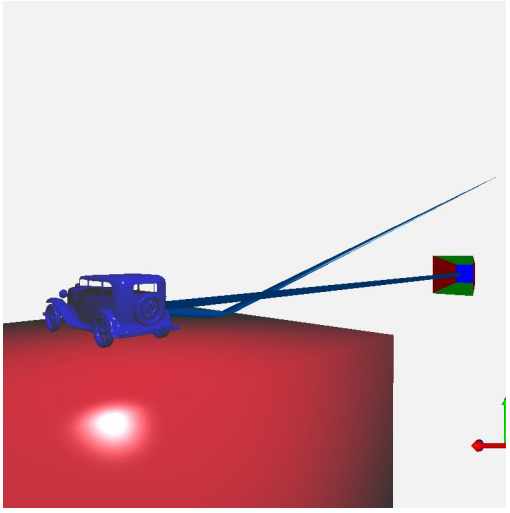
- Calculate hard shadows from a point light.
  Martin, Lucas

- Calculate soft shadows from a spherical light centered at a point light.
  Lucas

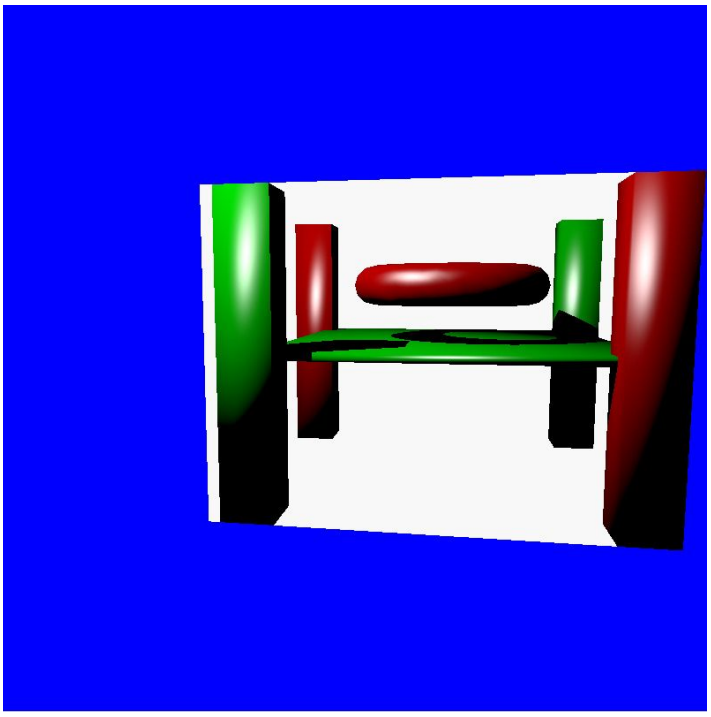


- Show an interactive display in OpenGL of the 3D scene and a debug ray tracer. A ray from a chosen pixel should be shown via OpenGL, illustrating the interactions with the surfaces.
  Rohan, Lucas, Jakub

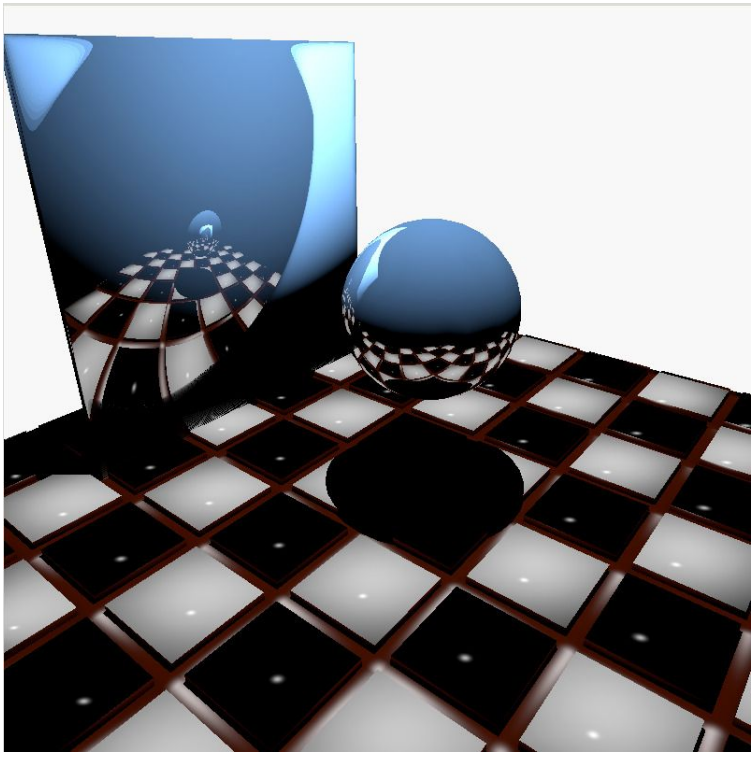




- Implement a (simple) acceleration structure.
  Martin (Ghiyath)

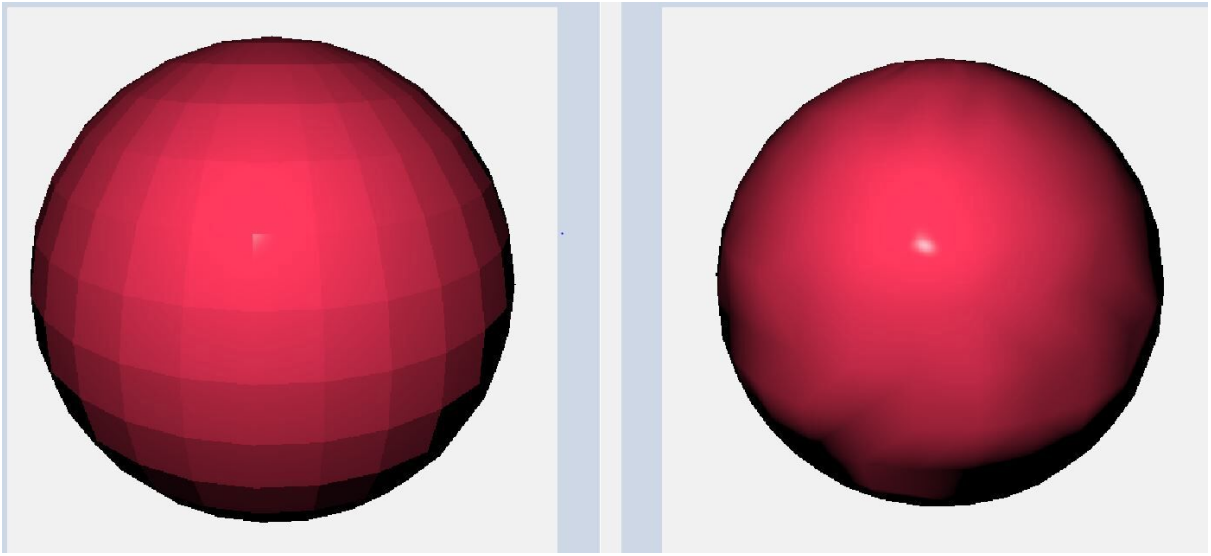


- Show a scene created by the group, exported as a wavefront object (OBJ) and directly loaded into the application.
  Martin, Rohan

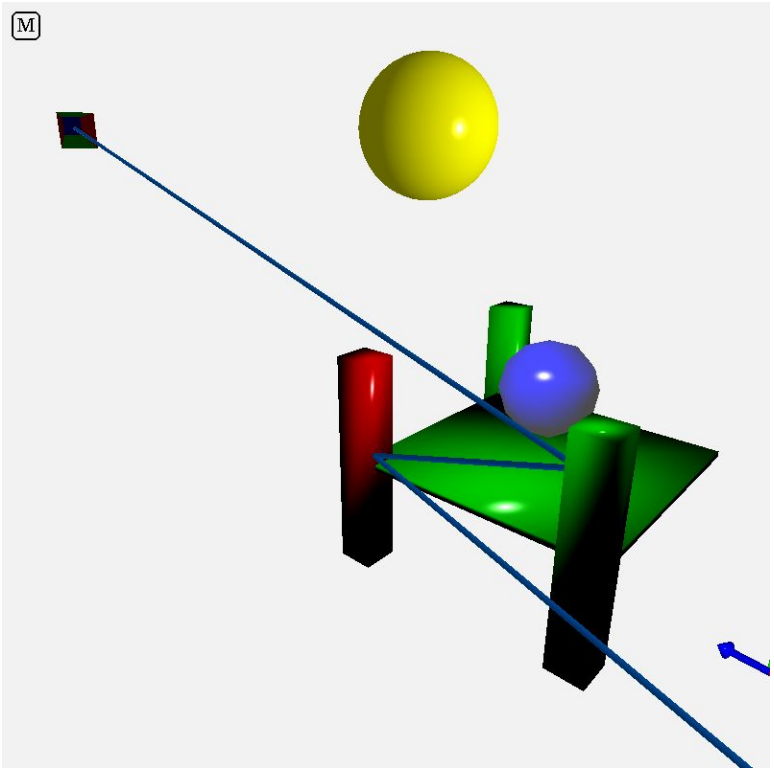


Extra features:

- Utilizing interpolated normals to smooth objects.
  Martin



- Extending the debugger to show the n_{th} reflection of a ray via the key-board, or triggering a ray highlighting and showing command line output of the selected ray's properties.
  Jakub, Rohan



- A numerical evaluation of the performance of your ray tracer.
  Jakub, Ivaylo

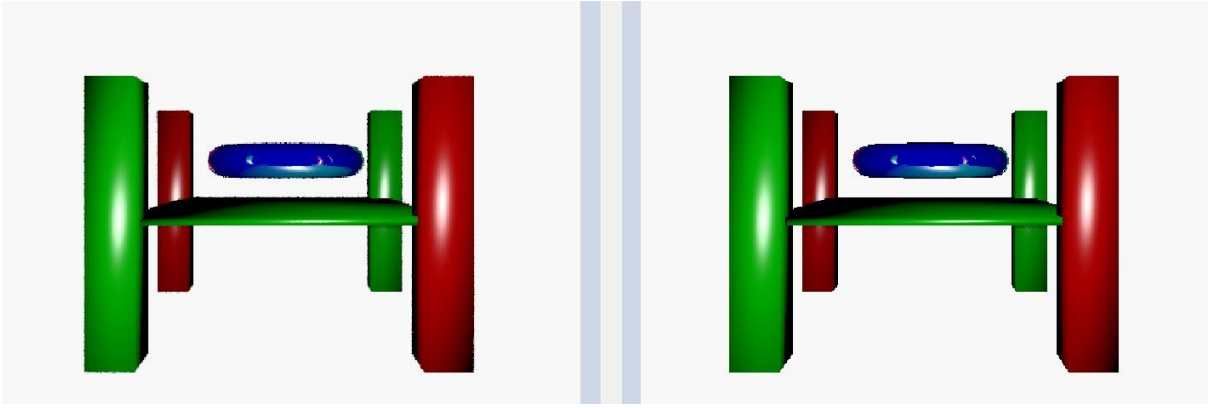
```
275532828 Faces checked (not including reflections)
9492480000 Faces to check w/o Acc structure (not including reflections)
2.90264% Faces checked in comparison to no Acceleration structure
ray tracing done!
It took 47 seconds to raytrace the Scene
Resulting in the speed of 3404pixels/s
```

- Multicore support of the ray tracer (implementing additional threads).
  Martin

```
12 concurrent threads are supported.
Using 12 threads.
Starting thread 0 of 12
Starting thread 1 of 12
Starting thread 2 of 12
Starting thread 3 of 12
Starting thread 4 of 12
Starting thread 5 of 12
Starting thread 6 of 12
Starting thread 7 of 12
Starting thread 8 of 12
Starting thread 9 of 12
Starting thread 10 of 12
Starting thread 11 of 12
```

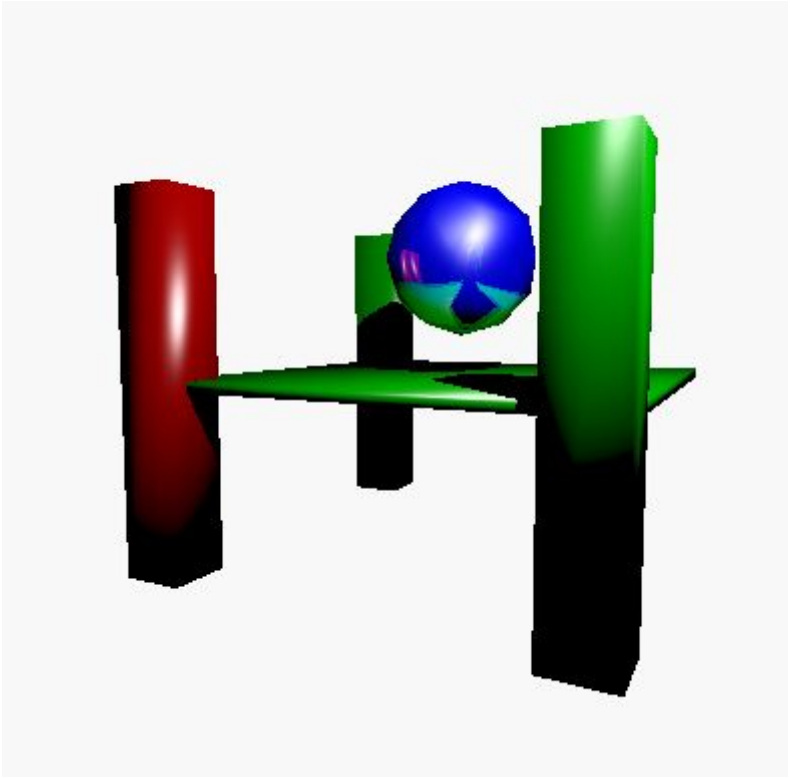
- Implement KD Tree: we noticed that kd tree didn't accelerate the code more than Bounding box, therefore we handed in a code that works with bounding box.
  Ghiyath

- Super Sample Anti-Aliasing (left with, right without)
  Jakub

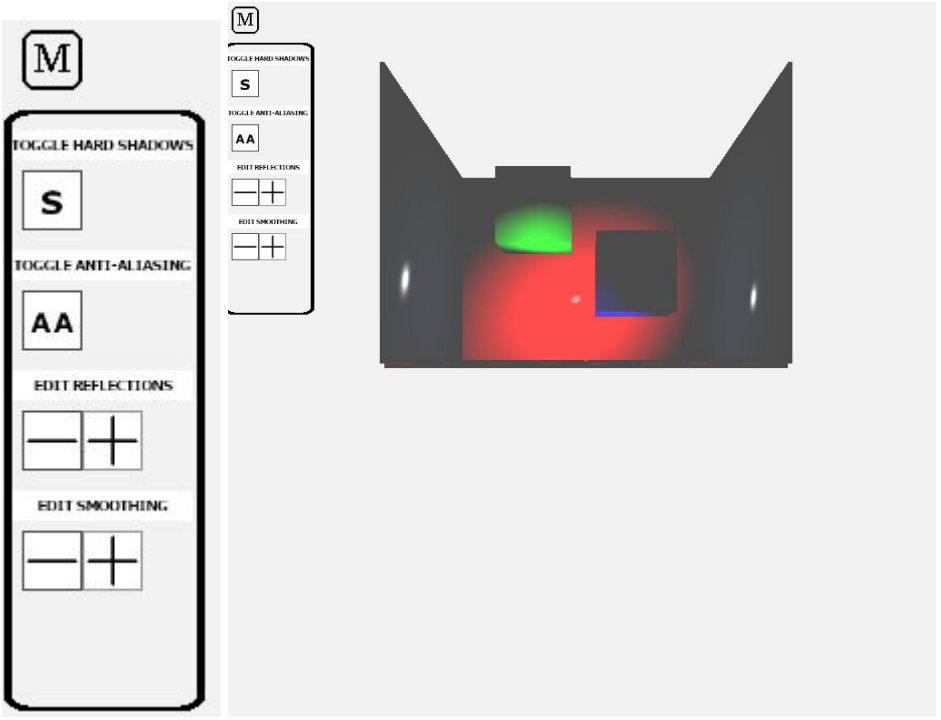


- Supporting refraction and the display of transparent objects
  Ivaylo

  We tried implementing refractions, but we do not know if they actually work. The code can be found in the project, but is not used. Here is an image we produced using the refraction code on:



- Interactive GUI
  Lucas



References

These materials were used while developing this raytracer.

1. Lecture material - Brightspace

2. [3D Intersections tutorial] (http://geomalgorithms.com/a06-_intersect-2.html)

3. [Ray Tracing in a Weekend] (https://github.com/RayTracing/raytracing.github.io)

4. [Phong reflection model] (https://en.wikipedia.org/wiki/Phong_reflection_model)

5. [Diffuse point lighting]
(https://www.tomdalling.com/blog/modern-opengl/06-diffuse-point-lighting/)

6. [Hard shadows]
(https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-shading/ligth-and-shadows)

7. [Fast AABB-Ray intersections]
(https://medium.com/@bromanz/another-view-on-the-classic-ray-aabb-intersection-algorithm-for-bvh-traversal-41125138b525)

8. [MTL material format (Lightwave, OBJ)] (http://paulbourke.net/dataformats/mtl/)

9. [Introduction to Shading]
(https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-shading/reflection-refraction-fresnel)