CS405 Ray Tracing Project Report

Cavit Çakır
23657

Table of Contents:

File Structure & How to run:

project.zip -> Includes project source code
renders.zip -> Includes images for each task.
Naming as follows: Task_<task_no>_<enumeration>

You can run the project as follow:
First run this: g++ -std=c++11 main.cpp
Secondly: ./a.out > Task_<task_no>_<enumeration>.ppm

Task 1 [Required]: Basic Scene (15Pt)

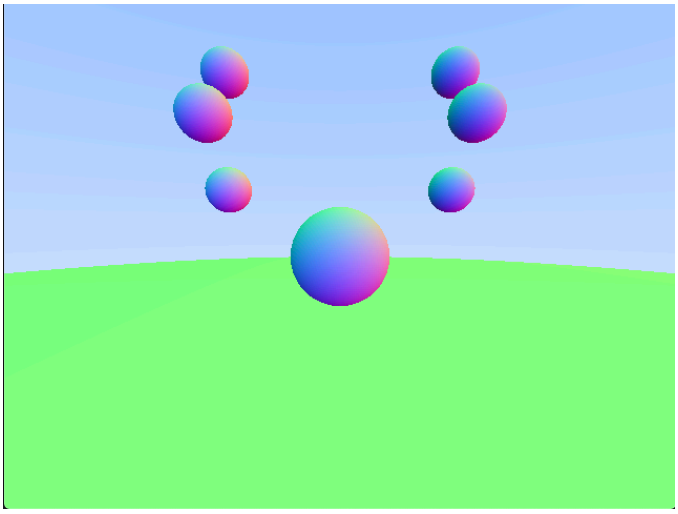


Image 1

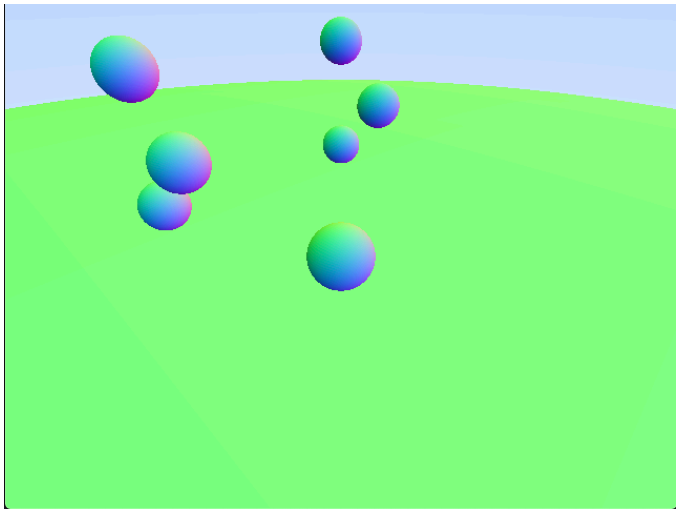


Image2

I created 8 spheres, green one for ground and others for observation.
Also added white-blue background with rays.

Position and Radius:

(0, 0, -1) & 0.3
(1, 1.5, -2) & 0.2
(1, 1, -1.5) & 0.2
(1, 0.5, -2) & 0.2
(-1, 1.5, -2) & 0.2
(-1, 1, -1.5) & 0.2
(-1, 0.5, -2) & 0.2
(0, -100.5, -1) & 100

Camera(eye) Positions:

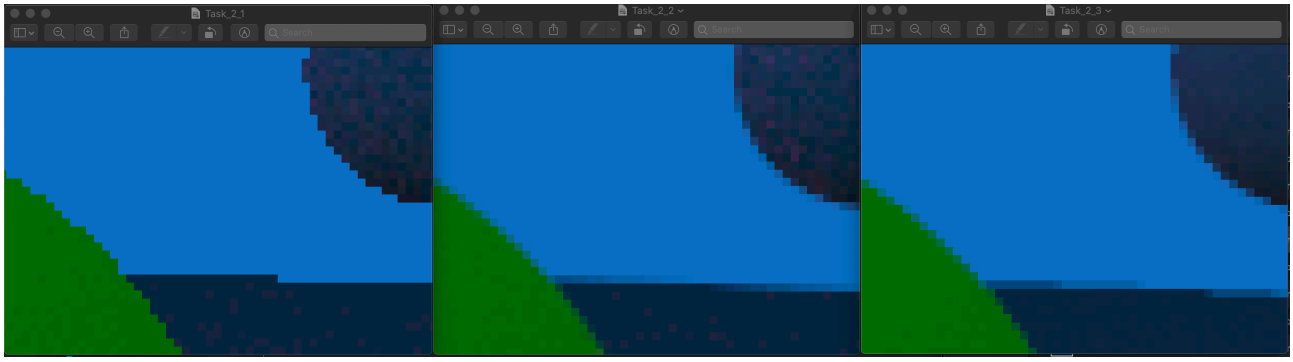
First Image:

Looks from: (0, 0.2, 1),
Looks at: (0, 0, -1)

Second Image:

Looks from: (-2, 2, 1)
Looks at: (0, 0, -1)

Task 2 [Optional]: Anti-Aliasing (10Pt)

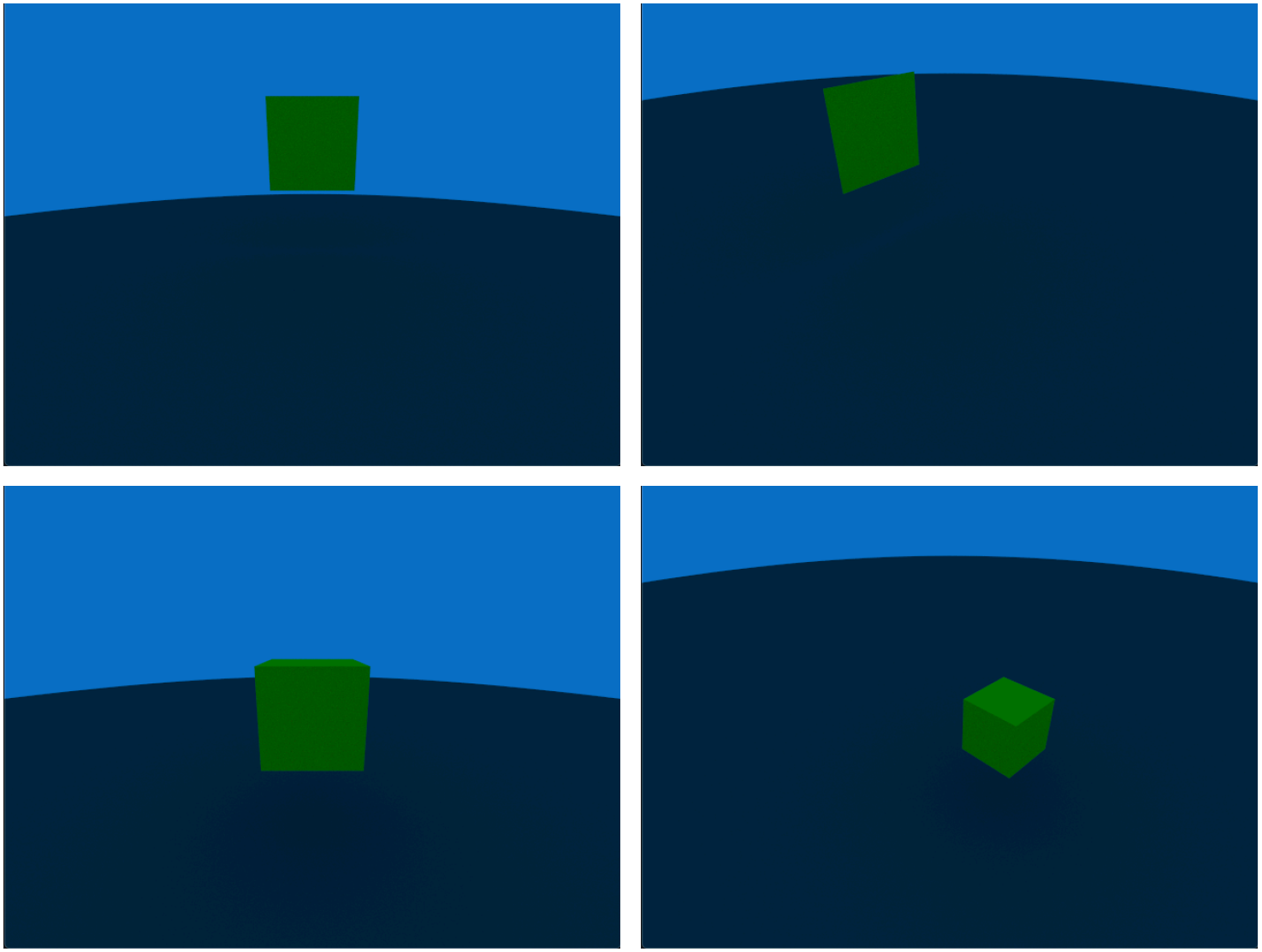


Anti-Aliasing Methods:

First image: No aliasing
Second image: Random Algorithm
Third image: Grid Algorithm

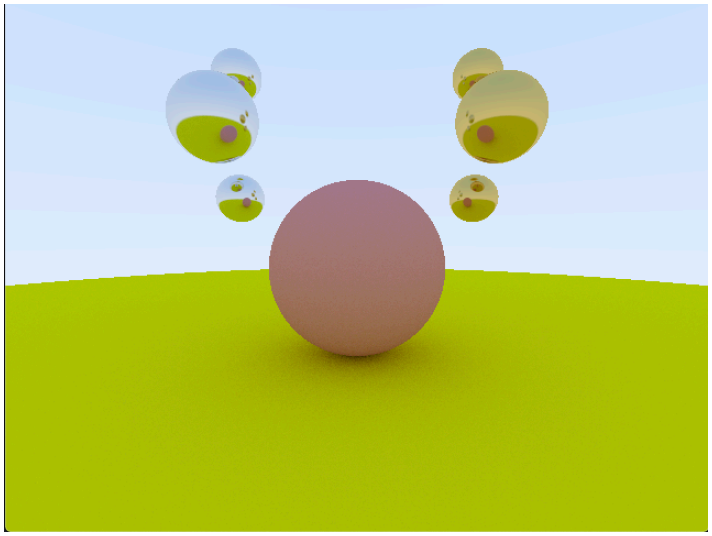
Grid Algorithm's effect on rendering an image is the best but in other hand it is most time consuming one.
If I have enough computational resource I would choose Grid Algorithm but if not, I would choose random algorithm because it works but not good as grid algorithm.
As we can see; anti-aliasing's contribution to render is: Algorithms are smoothened the sharp edges.

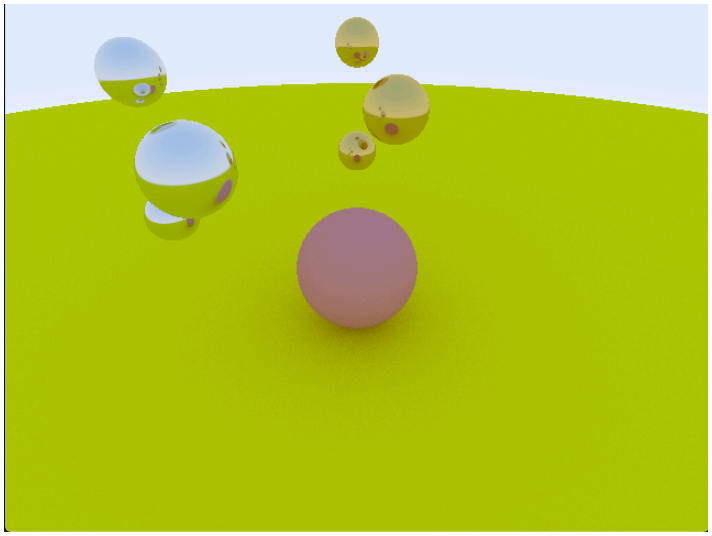
Task 3 [Extra]: More Shapes (5Pt)



I created a cube and plane. I found those formulas from "Ray Tracing The Next Week" article.

Task 4 [Required]: Diffuse and Metal Materials (25Pt)





I created 6 metal and 2 diffuse spheres. Metal spheres fuzziness values are 0.5, ground diffuse sphere's fuzziness value is 0 and center sphere's fuzziness value is 0.3.

Colors of objects:
Ground = (0.8, 0.8, 0.0)
Center = (0.7, 0.3, 0.3)
Metals left = (0.8, 0.8, 0.8)
Metals right = (0.8, 0.6, 0.2)

Position and Radius:

(0, 0, -1) & 0.5
(1, 1.5, -2) & 0.2
(1, 1, -1.5) & 0.3
(1, 0.5, -2) & 0.2
(-1, 1.5, -2) & 0.2
(-1, 1, -1.5) & 0.3
(-1, 0.5, -2) & 0.2
(0, -100.5, -1) & 100

Camera(eye) Positions:

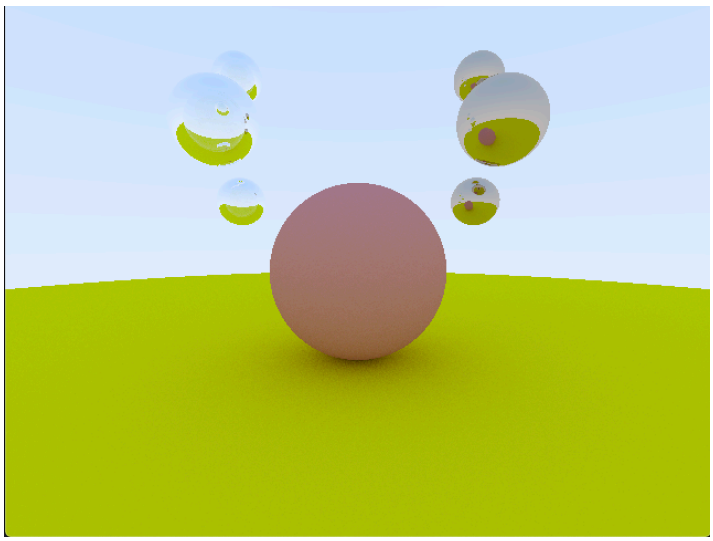
First Image:

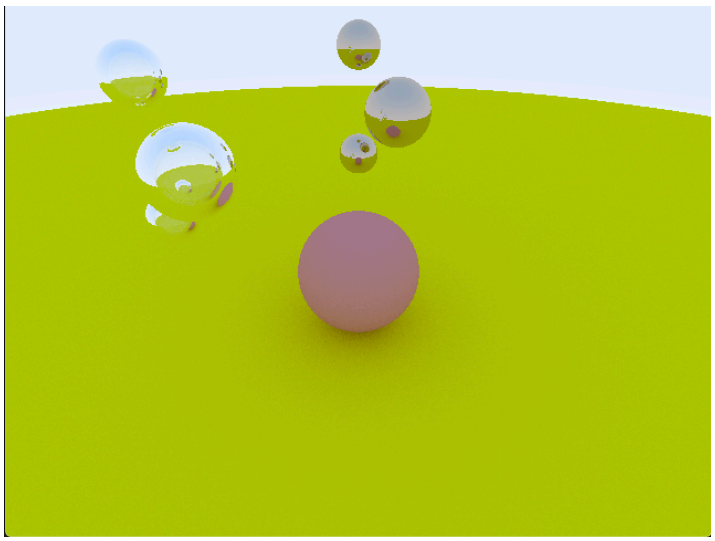
Looks from: (0, 0.2, 1),
Looks at: (0, 0, -1)

Second Image:

Looks from: (-2, 2, 1)
Looks at: (0, 0, -1)

Task 5 [Optional]: Refraction (10Pt)





I created 3 glass, 3 metal and 2 diffuse spheres. Glass spheres refraction value is 1.5 (I played with refraction value and found out that 1.5 seems good.), Metal spheres fuzziness values are 0.5, ground diffuse sphere's fuzziness value is 0 and center sphere's fuzziness value is 0.3.
I used negative radius for glass spheres and got resulted as surface normals point inward and got hollow glass spheres.

Colors of objects:
Ground = (0.8, 0.8, 0.0)
Center = (0.7, 0.3, 0.3)
Metals right = (0.8, 0.6, 0.2)

Position and Radius:

(0, 0, -1) & 0.5
(1, 1.5, -2) & 0.2
(1, 1, -1.5) & 0.3
(1, 0.5, -2) & 0.2
(-1, 1.5, -2) & -0.2
(-1, 1, -1.5) & -0.3
(-1, 0.5, -2) & -0.2
(0, -100.5, -1) & 100

Camera(eye) Positions:

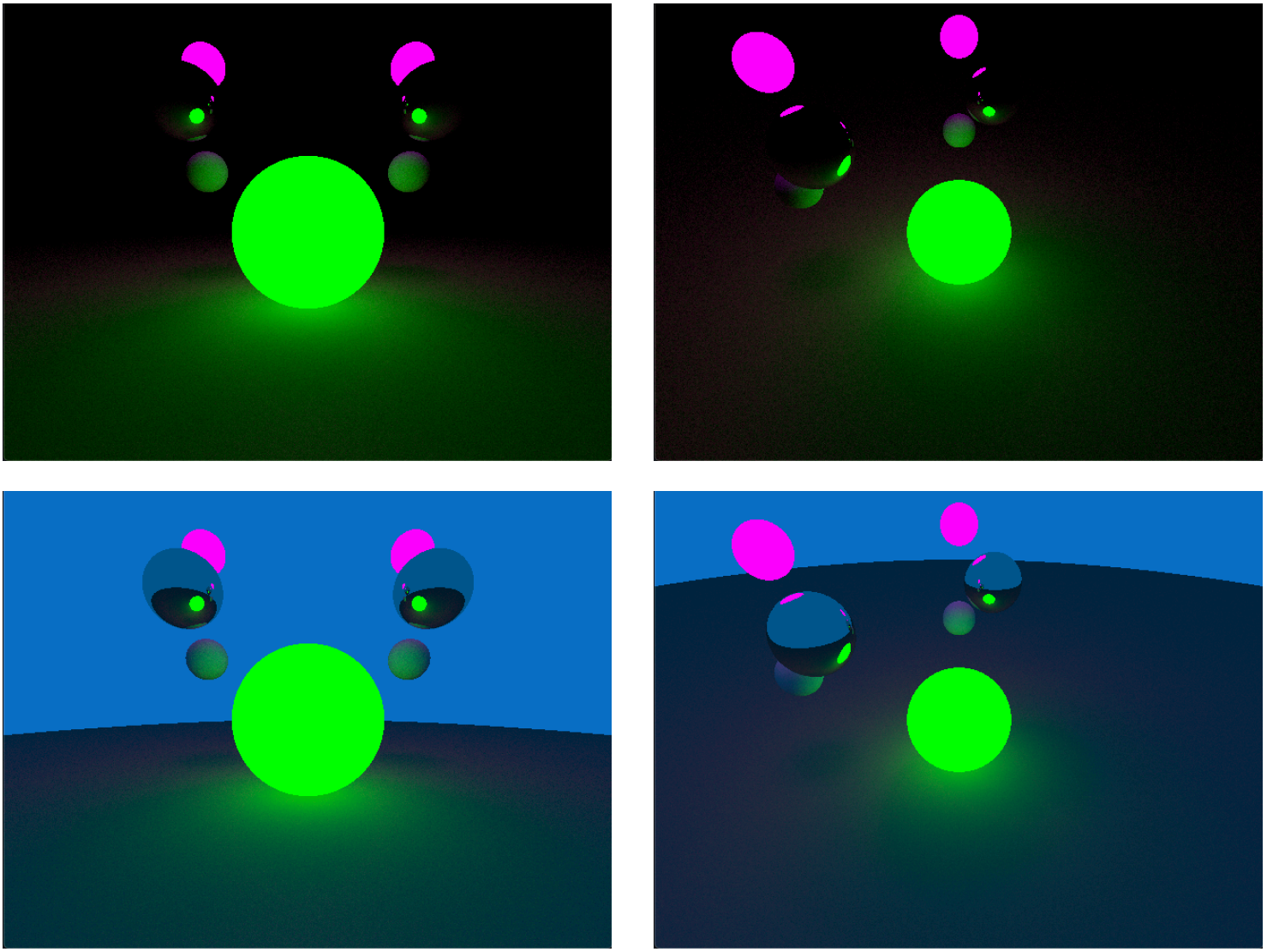
First Image:

Looks from: (0, 0.2, 1),
Looks at: (0, 0, -1)

Second Image:

Looks from: (-2, 2, 1)
Looks at: (0, 0, -1)

Task 6 [Required]: Lights (15Pt)



I created 2 metal, 3 diffuse and 3 emissive spheres. Metal spheres fuzziness values are 0, ground diffuse sphere's fuzziness value is 0.
The shadows are really satisfying if we give bright light sources.

Backgrounds:

No background: (0, 0, 0)
Blueish background: (0, 0.2, 0.6)

Colors of objects:

Ground = (0.8, 0.8, 0.0)
Green emissive = (0, 6, 0) -> Giving bigger than 1 values as RGB values results brighter spheres.
Purple emissive = (2, 0, 6) -> Giving bigger than 1 values as RGB values results brighter spheres.
Metals = (0.7, 0.6, 0.5)
Diffuse = (0.7, 0.6, 0.5)

Position and Radius:

(0, 0, -1) & 0.5
(1, 1.5, -2) & 0.2
(1, 1, -1.5) & 0.3
(1, 0.5, -2) & 0.2
(-1, 1.5, -2) & 0.2
(-1, 1, -1.5) & 0.3
(-1, 0.5, -2) & 0.2
(0, -100.5, -1) & 100

Camera(eye) Positions:

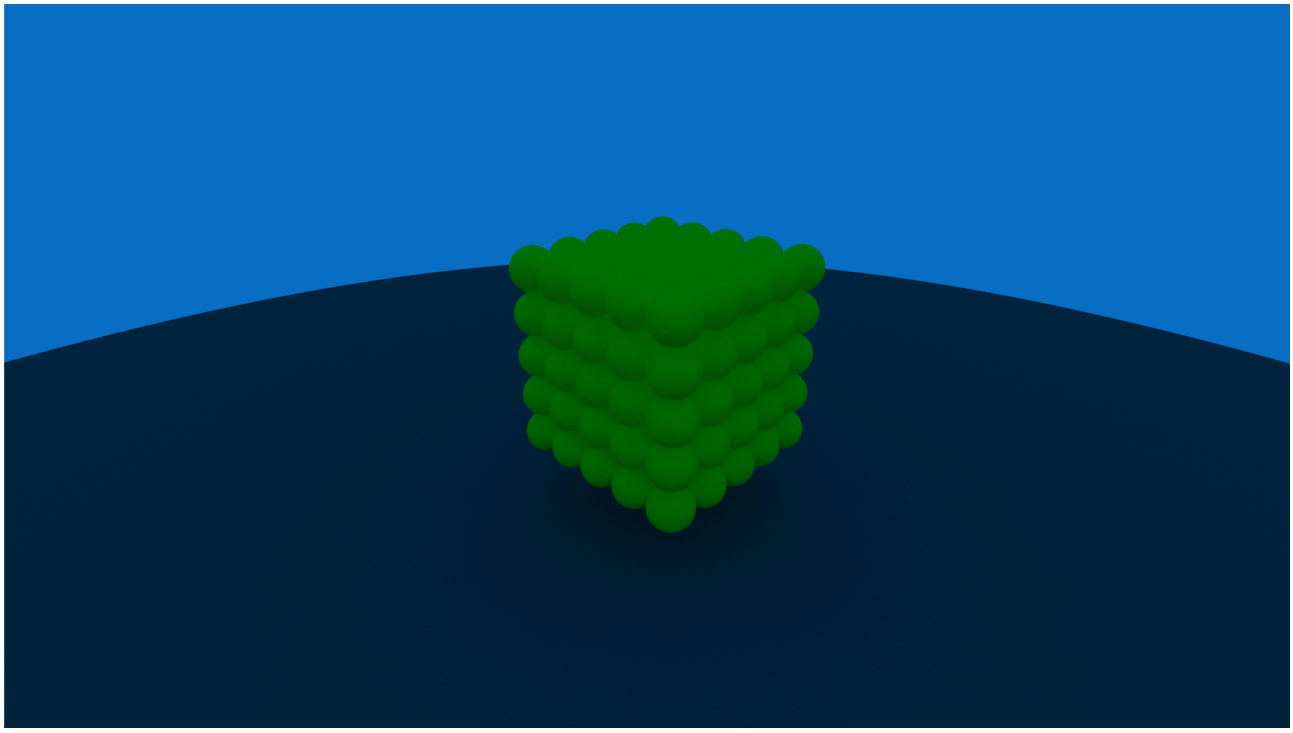
First Image:

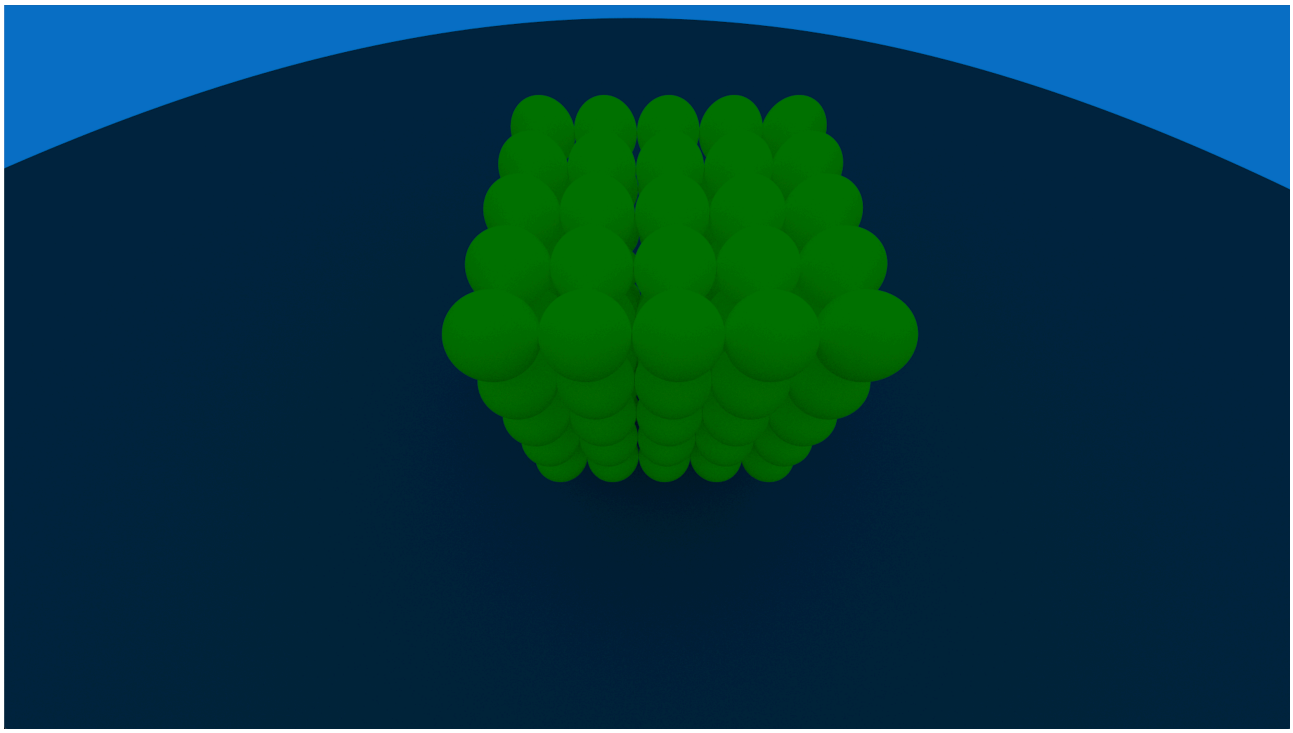
Looks from: (0, 0.2, 1),
Looks at: (0, 0, -1)

Second Image:

Looks from: (-2, 2, 1)
Looks at: (0, 0, -1)

Task 7 [Optional]: Let’s get creative! (10Pt)





I created a cube with 125 spheres. It is not original idea to create a cube with spheres but looks good in my opinion.

Task 8 [Required]: Questions (15Pt)

- **Question:** The approximate time complexity of ray tracing on models with triangles.
  - **Answer:**

    The algorithm for triangle ray tracing as follows:

    For each pixel of the frame:

    -> A ray is cast to determine which triangles are the closest,

    -> For each triangle, the distance is calculated from the triangle to the image plane.

    We consider each pixel and look for corresponding triangle polygon.
    So, when we are trying to determine which triangle is closest; we should look for all the triangles. With good data structures and algorithms we could get **O(logn) (where n is number of triangles)** complexity.

- **Question:** What is the difference between preprocessing and computing the image? Why?
  - **Answer:**
    - Image preprocessing is type of process that prepares image to render. There are some methods to preprocess image;
      - General normalizations for pixels.
      - Adjusting brightness to get good looking colors.
    - Computing image is the generating colors for each pixel according to rendering methods like ray-tracing, rasterization and etc.

- **Question:** What are the critical parameters for the ray tracer algorithm's performance?
  - **Answer:**
    - **Sample per pixel** parameter effects algorithms performance very critically.
      if sample size is bigger we get better renders but rendering time is getting longer.
    - If there are any algorithms for **blurring, anti-aliasing etc**.; Efficiency of algorithms are playing very important role for the ray tracers performance.

- **Question:** Imagine you are a systems engineer at Pixar. There is a new super-resolution in 6000×6000 pixels and you have to estimate the maximum rendering time per frame. Assume that the scenes are static, so you are not going to spend any CPU on animation, scene hierarchy, etc. The average scene has 500 objects with a total of 5.000.000 triangles to check intersection with.
  - **Answer:** The time complexity of triangle ray tracing is O(logn) so we have 6000x6000 pixels and 5 million triangles. In order to estimate maximum rendering time;
    We have to multiple total pixel with log(5million).
    So our equation will be: 6000 * 6000 * log(5000000)

References

P. Shirley, "Ray Tracing in One Weekend," [Online]. Available: https://raytracing.github.io/books/RayTracingInOneWeekend.html. [Accessed October 2020].

"Supersampling," [Online]. Available: https://en.wikipedia.org/wiki/Supersampling. [Accessed October 2020].

P. Shirley, "Ray Tracing The Next Week," [Online]. Available: https://raytracing.github.io/books/RayTracingTheNextWeek.html. [Accessed October 2020].