Report

Usage:

# must pip install pillow

Running Test Scripts

- Run ‘nmake all’ from the s2899221 folder on developer command prompt
- Render the test scripts with ‘.\render_test1.bat’ and ‘.\render_test2.bat’
- Alter the test script parameters as described in FeatureList.txt

Loading New Scene

- Copy export.py into Blender and change the output path to where you want
- Copy one of the render_test.bat files and change .json path to correct scene.

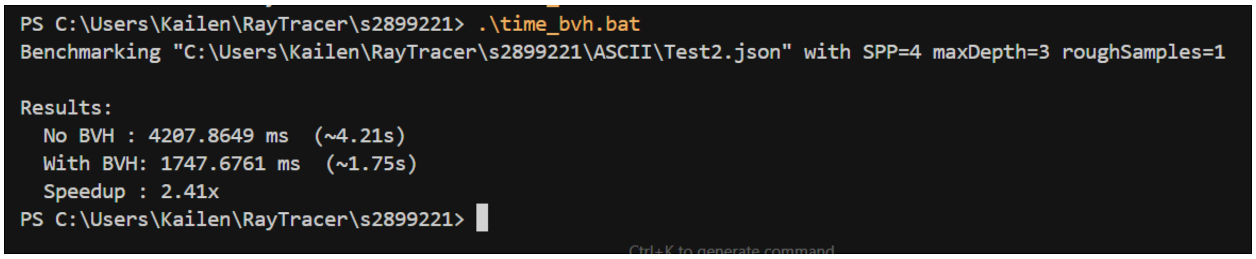
Prompting:

My general strategy for prompting the LLM to help me code was to break down tasks into smaller portions and to be specific. Instead of “Make me an export script to receive objects from blender,” I would prompt, “Make a export script to be run in blender that contains perspective cameras, point lights, and the following meshes: Cube, Sphere, Cylinder, Plane. Write these to a .json file with the following properties…”

The strength of LLMs was that I did not have to code everything. Overall, my project contains a few thousand lines of code. If I were to code this myself, it would probably take a few weeks. Additionally, I don’t know C++ very well, so I can use the LLMs knowledge of C++ syntax to my advantage. The weakness of LLMs is that certain tasks are not well represented in their training data, so implementing them is extremely difficult. I found that implementing reflections correctly was very difficult for the LLM. It insisted that objects be tinted heavily by the background, which caused whitening. Another weakness is that LLM code is hard to modify productively. I realised after module 3 that my rendering pipeline would not work with the tests. I tried to fix my pipeline using an LLM, but in the end I decided to rewrite the codebase from scratch.

Features:

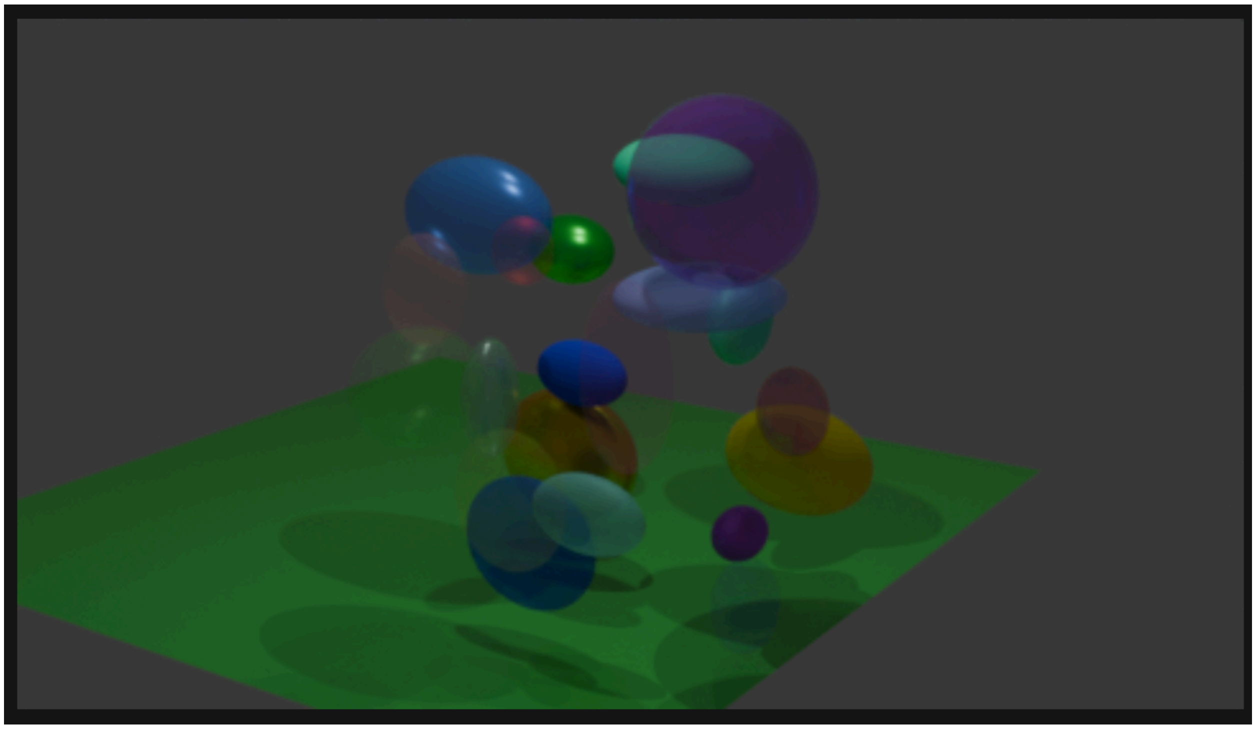
- Acceleration Hierarchy (Bounding Volume Hierarchy)



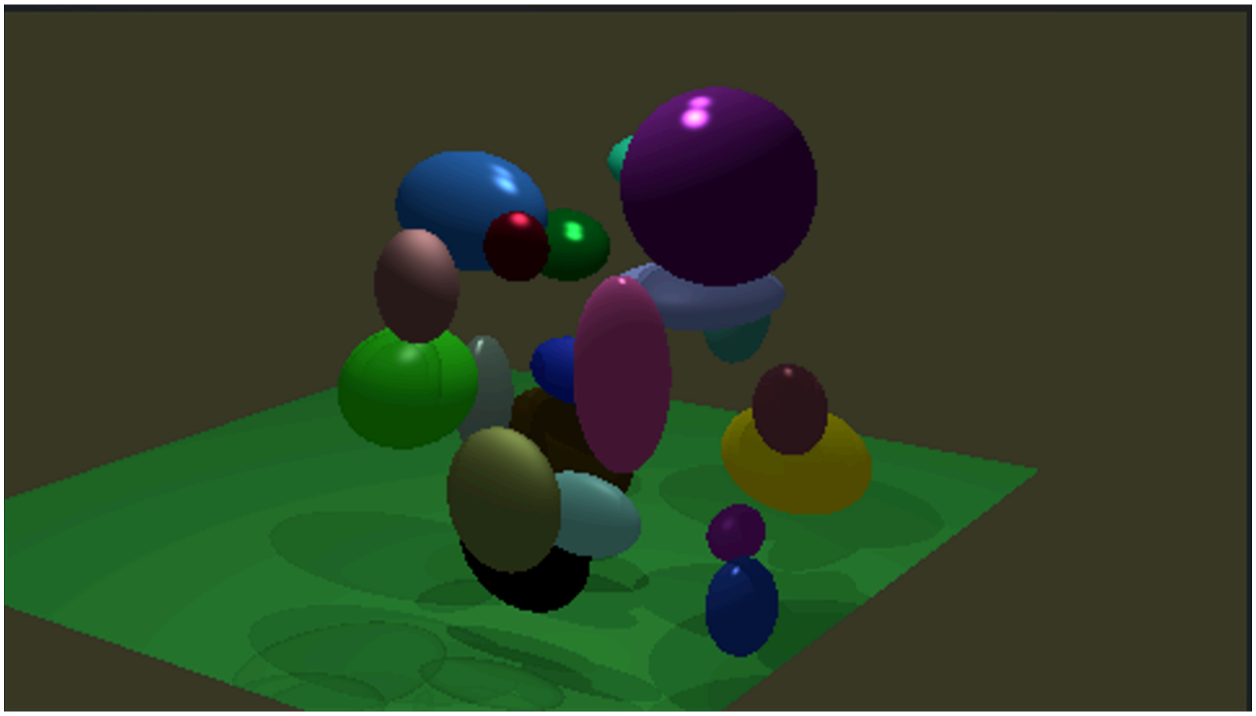
```
PS C:\Users\Kailen\RayTracer\s2899221> .\time_bvh.bat
Benchmarking "C:\Users\Kailen\RayTracer\s2899221\ASCII\Test2.json" with SPP=4 maxDepth=3 roughSamples=1

Results:
  No BVH : 4207.8649 ms  (~4.21s)
  With BVH: 1747.6761 ms  (~1.75s)
  Speedup : 2.41x
PS C:\Users\Kailen\RayTracer\s2899221>
```

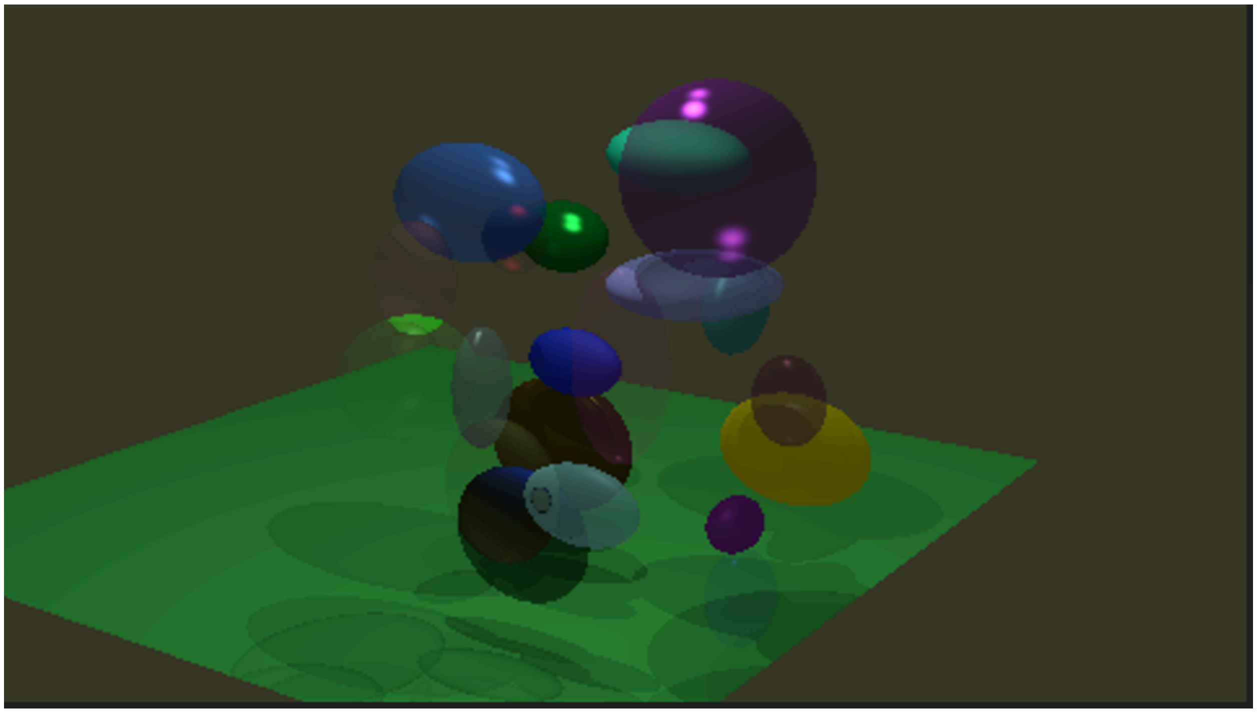
- Blender Modified Test 1 for Comparison



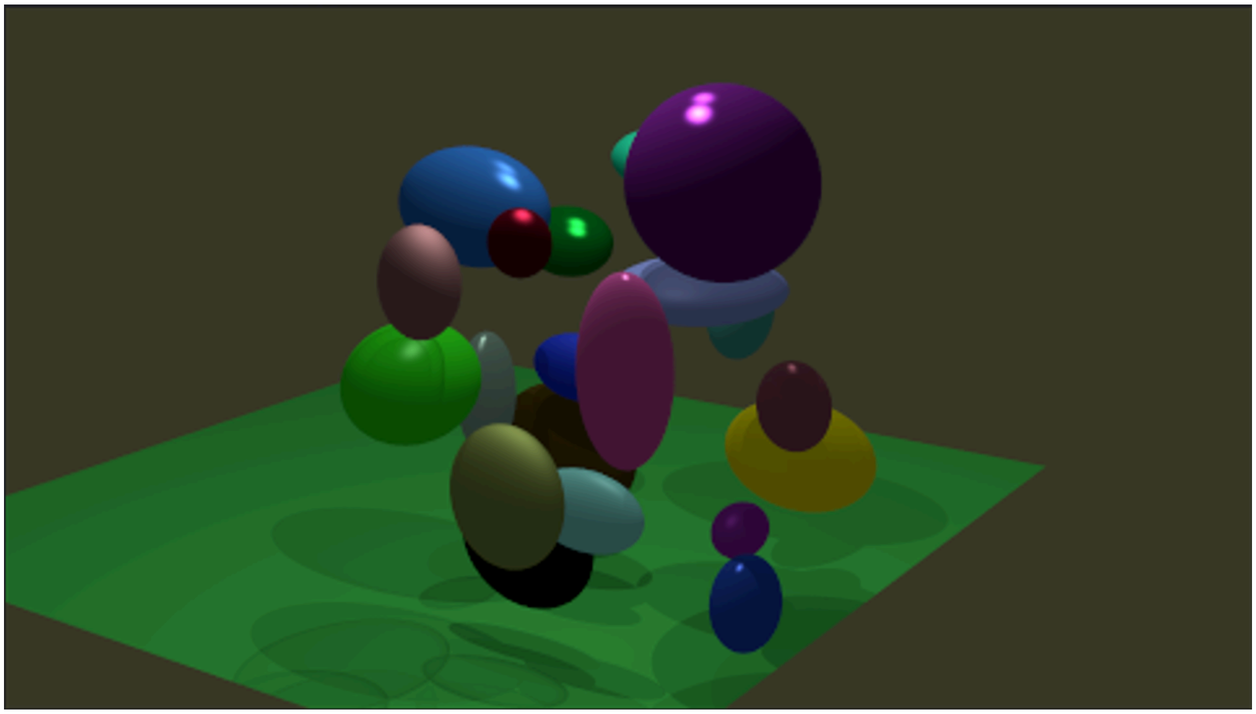
- **Whitted-Style Raytracing without Recursion and Anti-Aliasing**



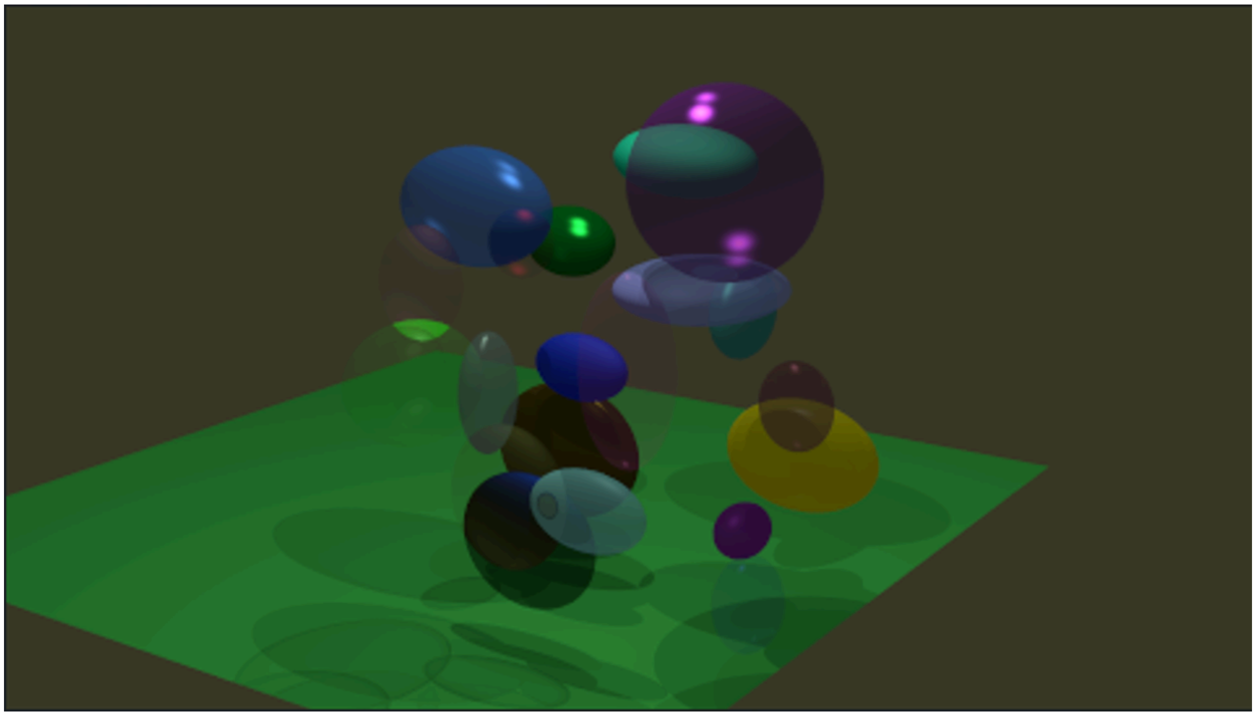
- **Whitted-Style Raytracing with Recursion**



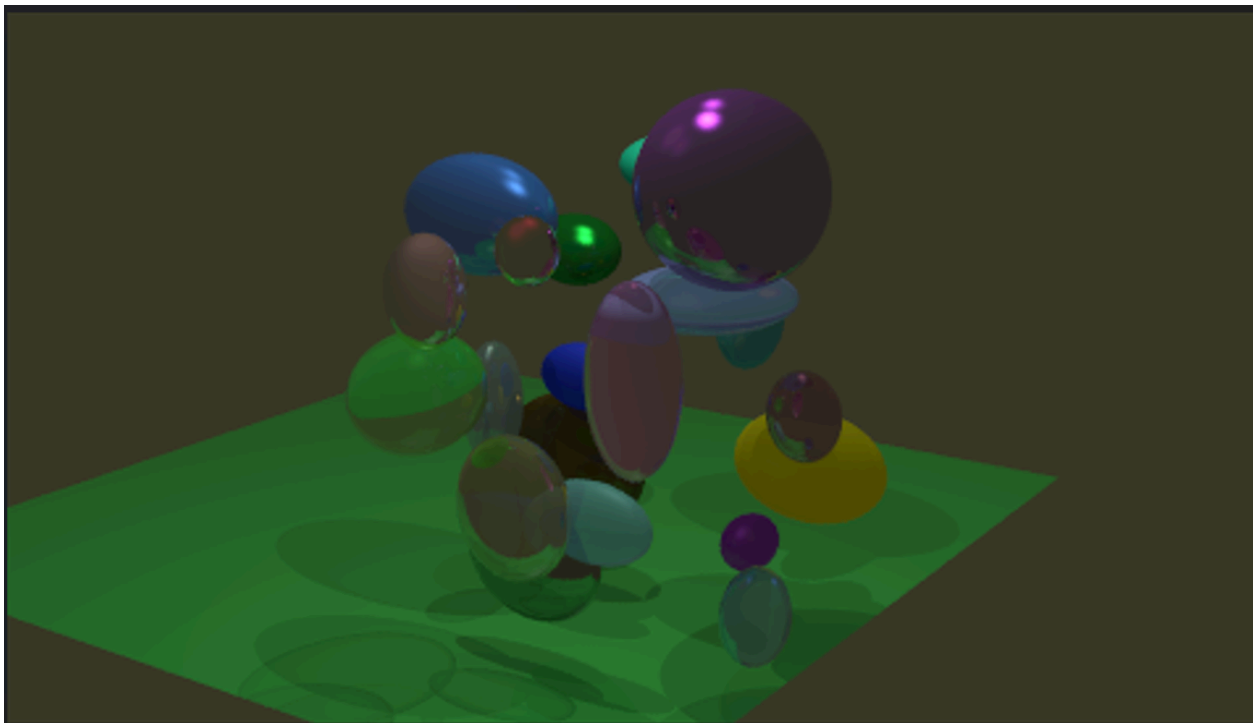
- **Whitted-Style Raytracing with Antialiasing**



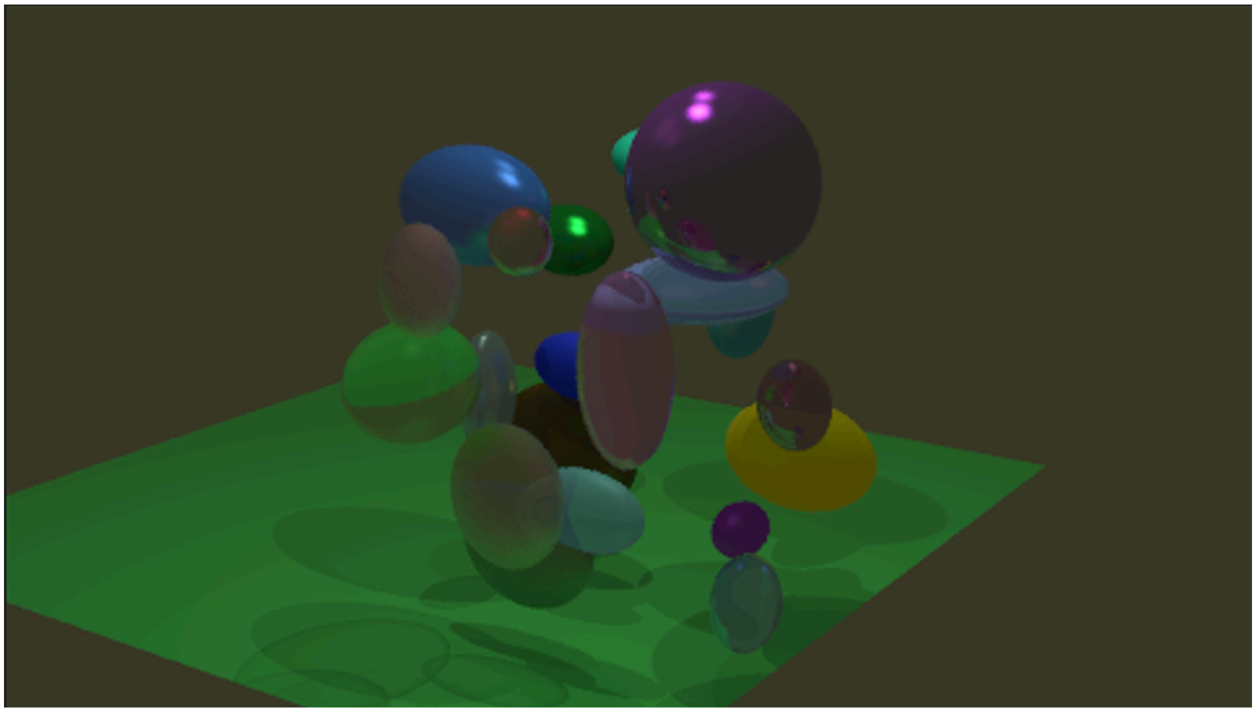
- **Whitted-Style Raytracing with Recursion and Antialiasing**



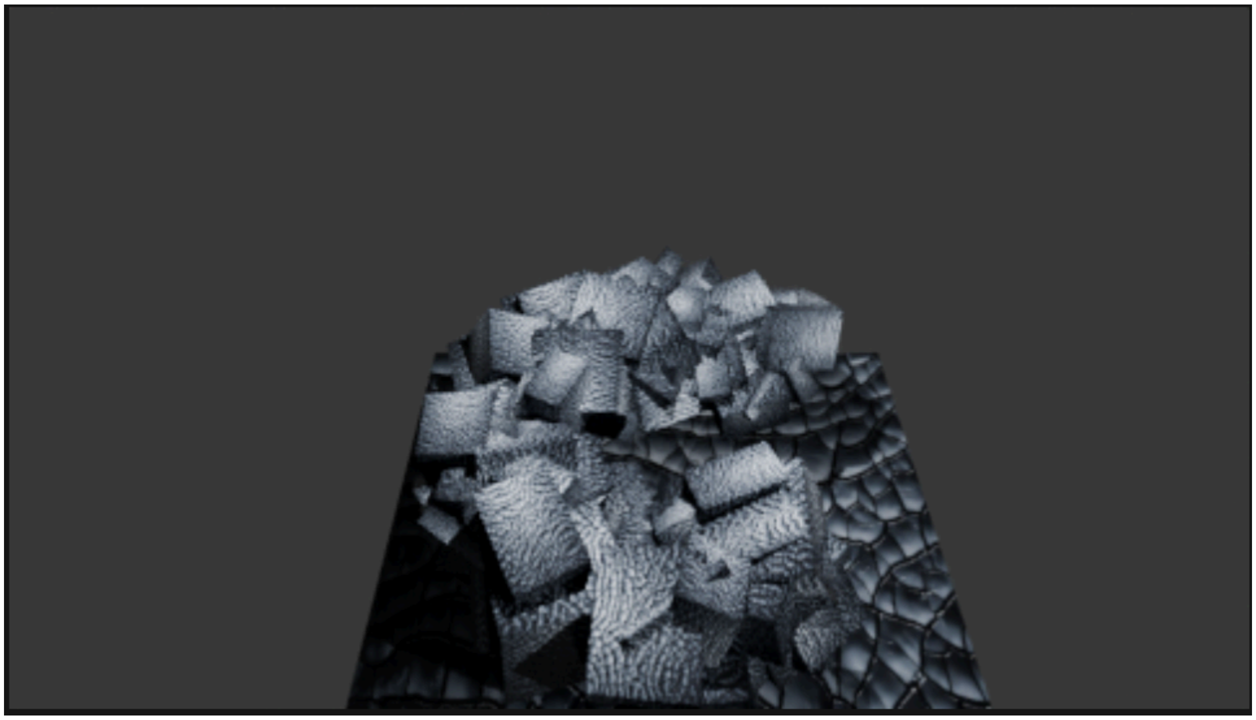
- **Glossy Reflections**



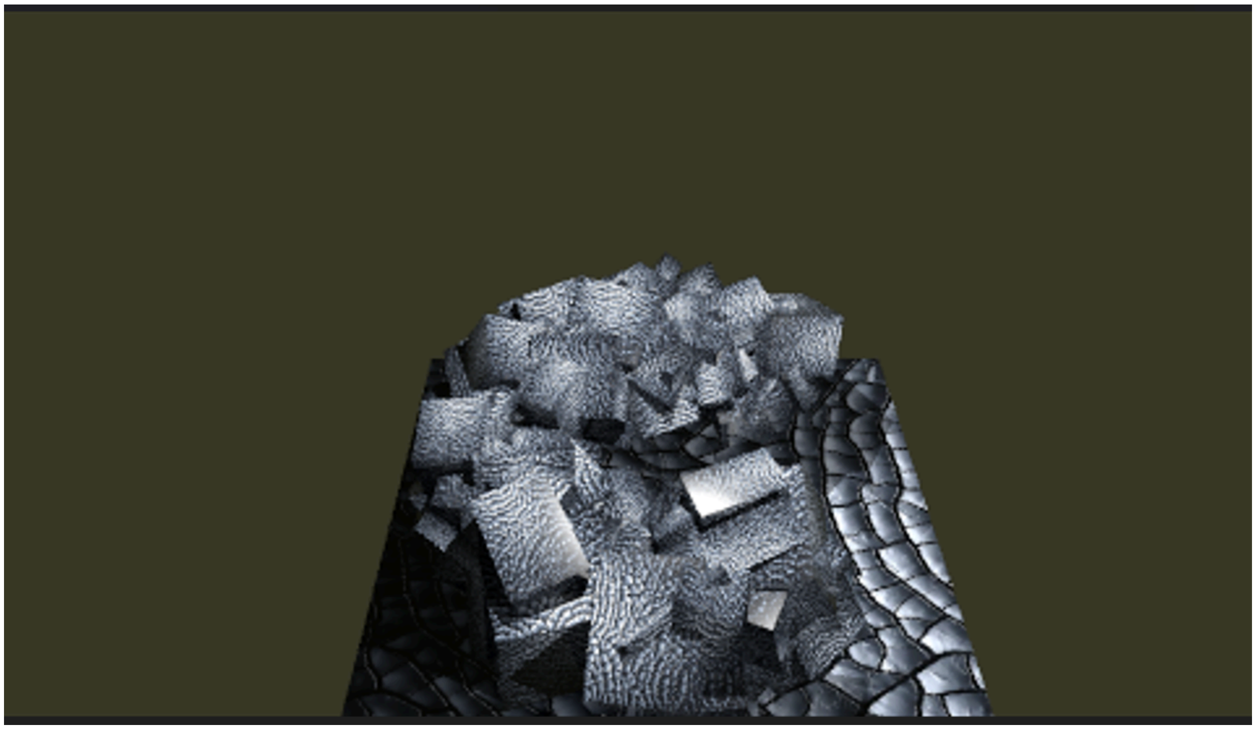
- **Roughness**



- **Blender Modified Test 2 for Comparison**



- **Image Texture**



Parts Completed:

Module	**Completed Parts**	**Incomplete Parts**	**Percent**
1	Export, Camera, Image	None	100%
2	Intersection, BVH Acceleration	None	100%
3	Whitted style, Antialiasing, Texturing	None	100%
4	Distributed RT	Motion Blur, Soft Shadows	50%

Timeliness:

I submitted all the modules on time. Up until the 4th module, no major changes occurred between modules. While completing the 4th module, I noticed that my rendering pipeline needed to be changed in order to run the test scenes. I tried to prompt an LLM to make these changes, but the code was messy at that point and the LLM struggled. Thus, I decided to clean up the code by restarting the project from scratch. I was able to prompt so that the organization of the code base improved. At that point, I knew exactly what should happen for each module, so prompting the LLM was easy.