3D Object Rasterization and Rendering with CUDA - Milestone Report

Minyu Cai (minyuc), Jinyan Zhu (jinyanz)

webpage: `https://jerrycmy2001.github.io/3d-object-rendering/`

1 Summary

We are going to implement 3D object rasterization and render a 360-degree camera view of the object. The entire project will be built on CUDA. We will first model the 3D object and camera position, and then use rasterization and z-buffering to render the view.

2 Current Progress

In the past 3 weeks, we have modeled the 3D object data structures, implemented the camera rotation and the rasterization algorithm, and successfully set the program on CUDA. Currently, we're able to break down simple 3D objects to triangles, render those triangles from the perspective of the camera, and rotate the camera to have arbitrary views.

3 Preliminary Results

Currently, we have made some demos with our 3D object renderer for simple objects. These are posted on our web page.

4 New Goals and Deliverables

Reviewing our schedule for the first 3 weeks, we think we have finished all tasks and achieved the basic goals of our project.

For the next step, we believe we can continue to follow our plan and produce all the other deliverables. As we believe the current result is satisfying enough, we think it is not necessary to implement the anti-aliasing algorithm. Instead, we plan to build new scenes more related to the real world.

4.1 New goals

- Model 3D objects, camera position and direction. This is the prerequisite to performing rendering and creating the camera view. The 3D objects may be polygons and we need to design how to model them in a correct and efficient way. (Completed)
- Render camera view with 3D object projection with rasterization and z-buffering. This is the second step after modeling. When rendering the camera view, we need to focus on each view with specific positions and directions. Therefore, rasterization and z-buffering are a must for our project. (Completed)
- Rotate the camera to get a 360-degree view of the objects. After we achieve the first two goals, the final step is to use the techniques we have implemented to get the entire view. Till this, we believe the whole object is finished. (Completed)
- Build different scenes. We want to verify the usage and performance of the 3D object renderer in various scenarios. We hope it can not only build simple objects but can also help in some real-world situations.
- Experiment on PSC machine to retrieve data for measuring speedup.

4.2 Deliverables - Poster Session

We plan to demonstrate the following parts during the poster session:

- Show the video of rendering simple and complex 3D objects in 360 degree view.
- Provide data analytics results of the speedup to prove the performance gain from parallelizing on batch of GPUs.

5 Updated Schedule

- Week 4 Part 1 (12/02 - 12/04)
  - Bugfix the random scene (Jinyan)
  - Use `OpenMP` to parallelize the rendering across multiple GPUs (Minyu)
- Week 4 Part 1 (12/05 - 12/08)
  - Optimize the course title page, especially the shape and color (Jinyan)
  - Experiment on PSC machine to collect data and analyze speedup (Minyu)
- Week 5 Part 1 (12/09 - 12/12)
  - Build a larger scene with good visual effects for demo purposes (Minyu and Jinyan)
- Week 5 Part 2 (12/13 - 12/15)
  - Prepare for poster session and write final report (Minyu and Jinyan)

6 New Challenge

- How to use openMP to parallelize the task among multiple GPUs. We believe it's similar to what we've done in the previous projects, but it remains an unknown.
- How speedup would be like on PSC machine remains? We've experimented on local desktop GPU and found it has significant speedup over the CPU implementation, so most likely we can also achieve great speedup on PSC machines.