Minecraft: Ray Traced

Marco Jonkers

February 9, 2017

Abstract

This project attempts to implement a GPU ray tracer in the video game Minecraft, using CUDA. The ray tracing itself is done in CUDA. At first, I attempt to ray trace the vertex buffers given by Minecraft. Then, I implement my own data structures for increased performance.

1 Goal

Similar to other ray tracing research projects by Intel Corporation. Also shortly explain the benefits of ray tracing versus regular rasterization. Minecraft's world is made up of voxels, which is nice for ray tracing.

2 Setup

Here I explain how the project is set up and the technologies involved. Minecraft is coded in Java. The Minecraft Forge project allows me to do two things: Listen to specific events using pre-installed hooks Edit Minecraft bytecode before it is loaded Using Java Native Interface (JNI), I can call into C++ code from Java. The C++ code contains the CUDA kernel. Data is passed between OpenGL and CUDA using CUDA's graphics interop layer.

3 Minecraft Rendering System

There are four geometry groups:

Solid This is solid geometry.

Cutout Used for glass.

Mipped Cutout Identical to Cutout, but mipmapped.

Translucent Used for block which have partial transparency (alpha blending).

Every geometry group has its own vertex buffer.

4 Raytracing Minecraft's Vertex Buffers Directly

I explain something here that is found in Figure 1.

5 Challenges

The challenges of this project include:

Ray tracing performance Because of gameplay.

Acceleration structure rebuild speed Because of gameplay.

6 Static Geometry

Test for citing [Amanatides et al., 1987].

I am also citing [Ivson et al., 2009].

I am also citing [Reinhard et al., 2000].

7 Dynamic Geometry

This section describes ray tracing dynamic objects, such as NPCs.

8 Benchmarks

9 Future work

Future work is addressed here.

References

[Amanatides et al., 1987] Amanatides, J., Woo, A., et al. (1987). A fast voxel traversal algorithm for ray tracing. In *Eurographics*, volume 87, pages 3–10.

[Ivson et al., 2009] Ivson, P., Duarte, L., and Celes, W. (2009). Gpu-accelerated uniform grid construction for ray tracing dynamic scenes. *Master's thesis, Departamento de Informatica, Pontificia Universidade Catolica, Rio de Janeiro.*

[Reinhard et al., 2000] Reinhard, E., Smits, B., and Hansen, C. (2000). Dynamic acceleration structures for interactive ray tracing. In *Rendering Techniques 2000*, pages 299–306. Springer.

List of Figures

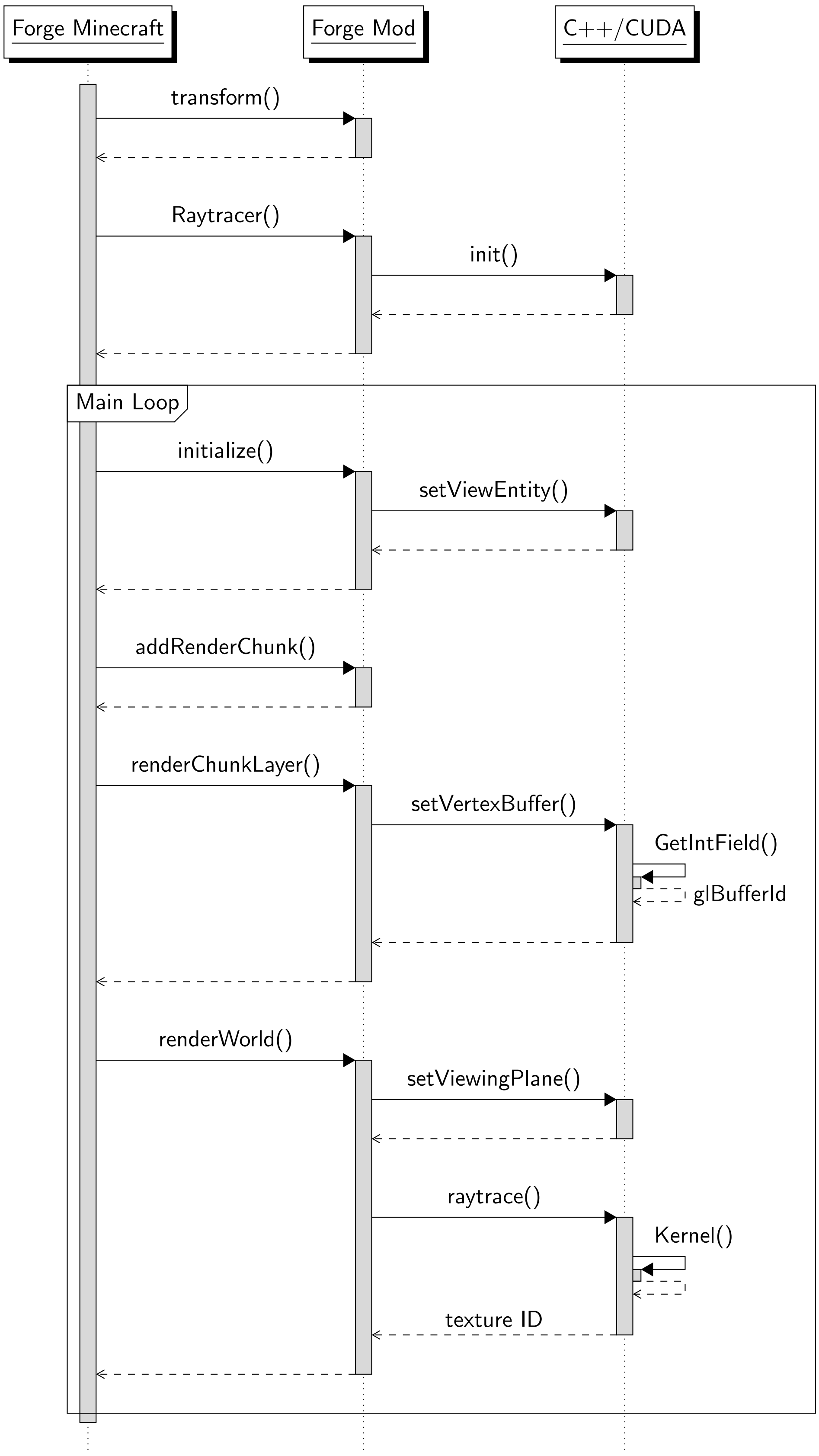

Figure 1: Sequence diagram showing interaction between the modules

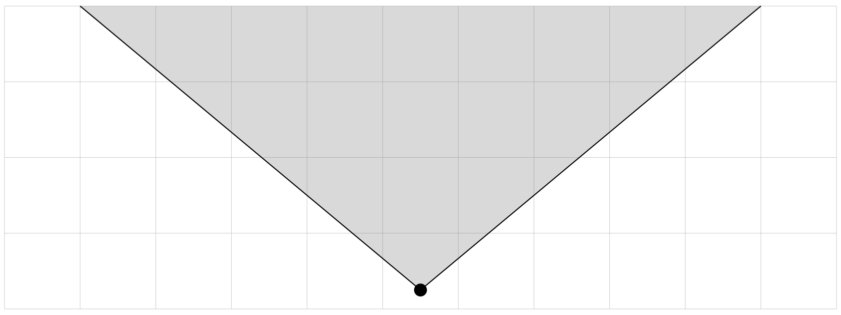

Figure 2: A player's default field of view, facing north. Diagonal rays traverse more cells.

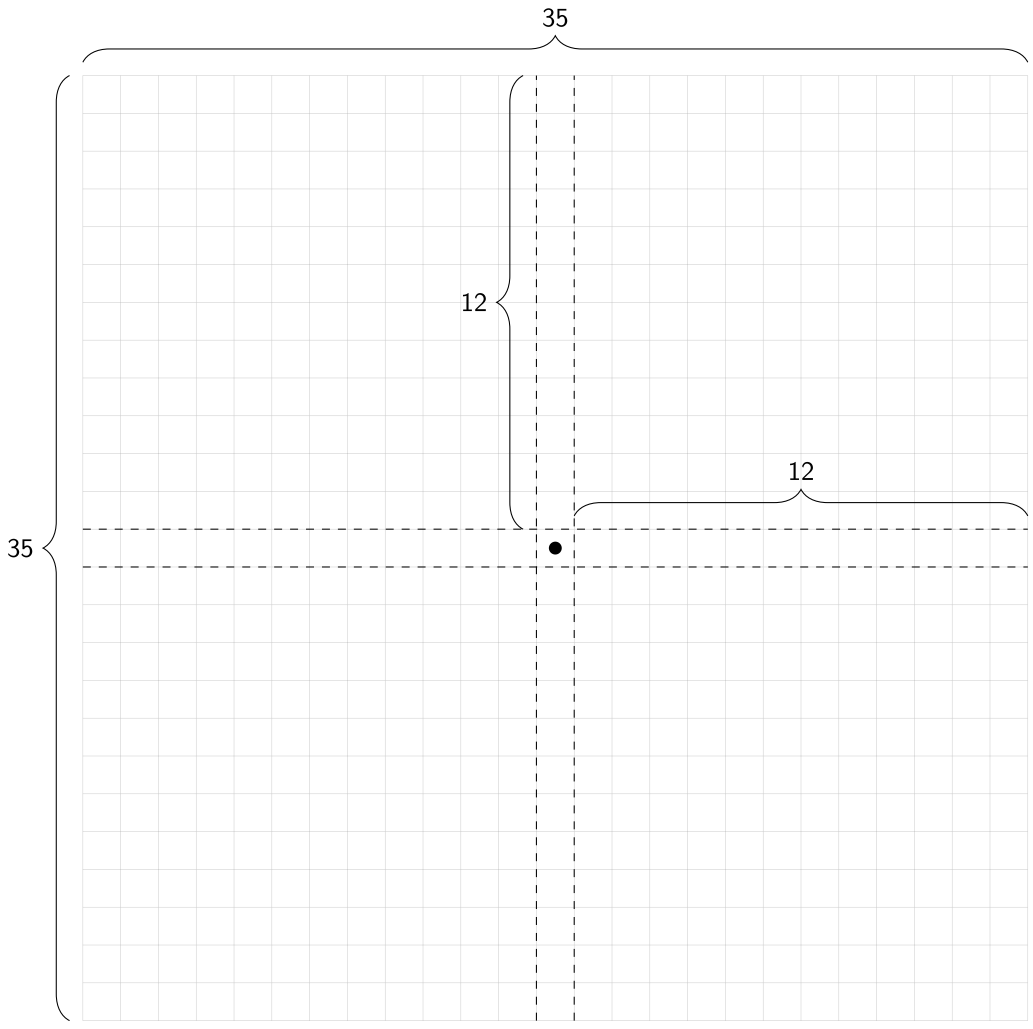

Figure 3: A two-dimensional view of the vertex buffer array, assuming a render distance of 12. Each cell contains 16 RenderChunks.

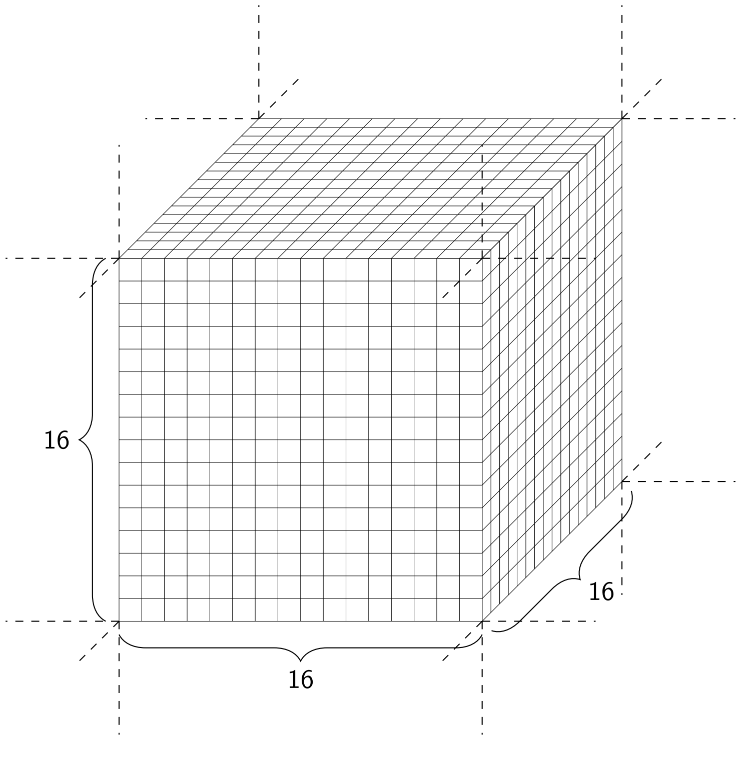

Figure 4: A single, completely filled RenderChunk.