Tony Lu (tianyua1)
Julian Tutuncu-Macias (jtutuncu)

Making Minecraft Fast

URL: rex-ludorum.github.io

SUMMARY: We are going to optimize Craft, a lightweight version of Minecraft written in C with OpenGL, with a predictive rendering scheme in which users' paths through the environment are predicted and the scenes along those paths rendered ahead of time.

BACKGROUND: Minecraft is a popular sandbox game in which the world is composed of cube-shaped blocks. Users can destroy and build parts of the world by breaking or placing blocks, respectively. While Minecraft is closed-source, Craft is a lightweight open-source version that supports less features. Its renderer already makes some optimizations, like partitioning the world into 32 by 32 block "chunks" and only rendering exposed faces. Our optimization will involve precomputing the visible chunks along various paths the user could take to speed up the rendering process. Since there will likely be multiple paths, these can all be computed in parallel. Even along a single path, we can split up the world so that chunks in different sections of the world can be computed in parallel to determine which are visible. We will also port the renderer to Vulkan, a lower level graphics API that offers parallel tasking.

THE CHALLENGE: There is no clear way to predict user movement, so devising a good prediction scheme will be challenging. Rendering the scene itself presents opportunities for parallelism, since we will have to determine a scheme to split up the pixels on the screen. We must also find opportunities to exploit spatial and temporal locality in the user's motion. There is much more computation than communication, so we must find as many ways to parallelize the computation as possible.

RESOURCES: We will use the existing implementation of Craft and add our code to it (https://github.com/fogleman/Craft). We will use machines with different levels of graphics computing power to collect statistics about our renderer.

GOALS AND DELIVERABLES: We plan to at least match the performance of OpenGL in the base implementation with our Vulkan implementation, and successfully implement a predictive rendering scheme using player pathfinding. If time allows, we will adapt our rendering to be able to handle other players in a multiplayer game mode, and make our implementation performant on as constrained hardware resources as possible. If our work progresses more slowly, we will aim to match the base implementation's frame rate as closely as possible. For our demo, we will have graphs featuring our speedup, and computers for people to play on to compare implementations.

Tony Lu (tianyua1)
Julian Tutuncu-Macias (jtutuncu)

PLATFORM CHOICE: We will use different machines to benchmark our implementation. Some will have dedicated GPUs and some will only have integrated graphics, like our laptops. Since the base Minecraft implementation is lightweight and is performant on our Macbook Pros with only integrated graphics, we will focus on systems with less resources to benchmark our implementation.

SCHEDULE:

November 4 - 10:

- Familiarize ourselves with current implementation of Craft, develop performance metrics & tests
- Study specifications of computers we are benchmarking on, especially integrated graphics

November 11 - 17:

- Familiarize ourselves with Vulkan
- Begin porting to Vulkan
- Draft prediction schemes

November 18 - 24:

- Test prediction schemes

November 25 - December 1:

- Finalize prediction schemes

December 1 - 8:

- Finish poster and report