Project Milestone Report: Parallel Terrain Generation

By Petros Emmanouilidis and Victor Zayakov

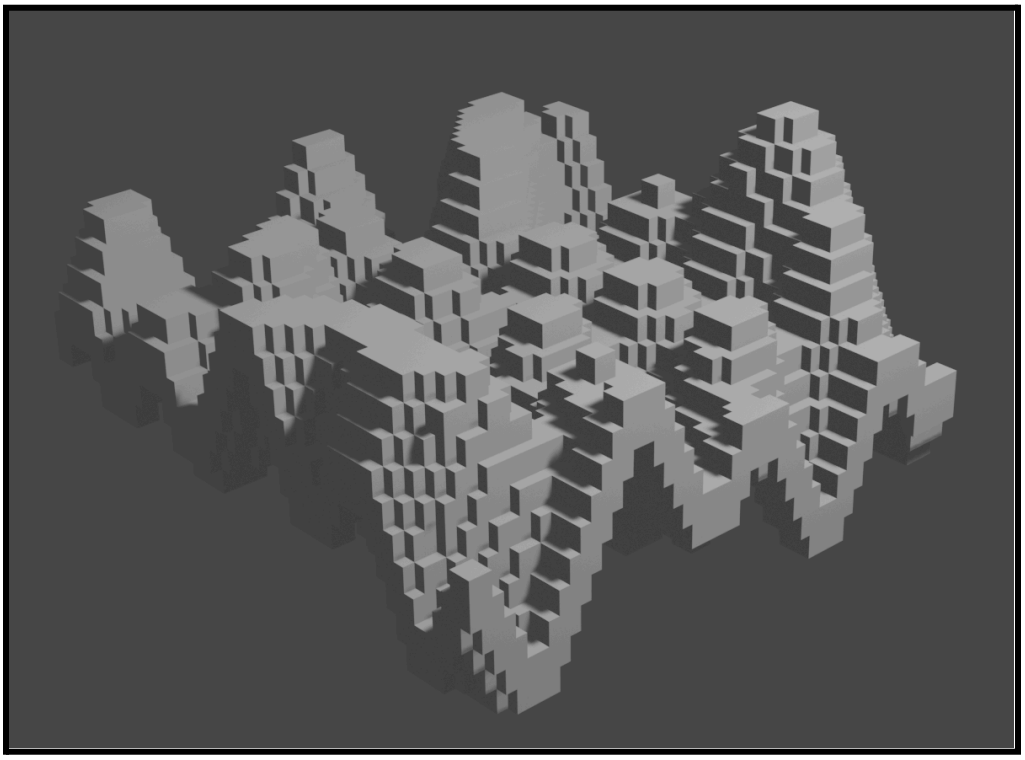
Summary of Progress

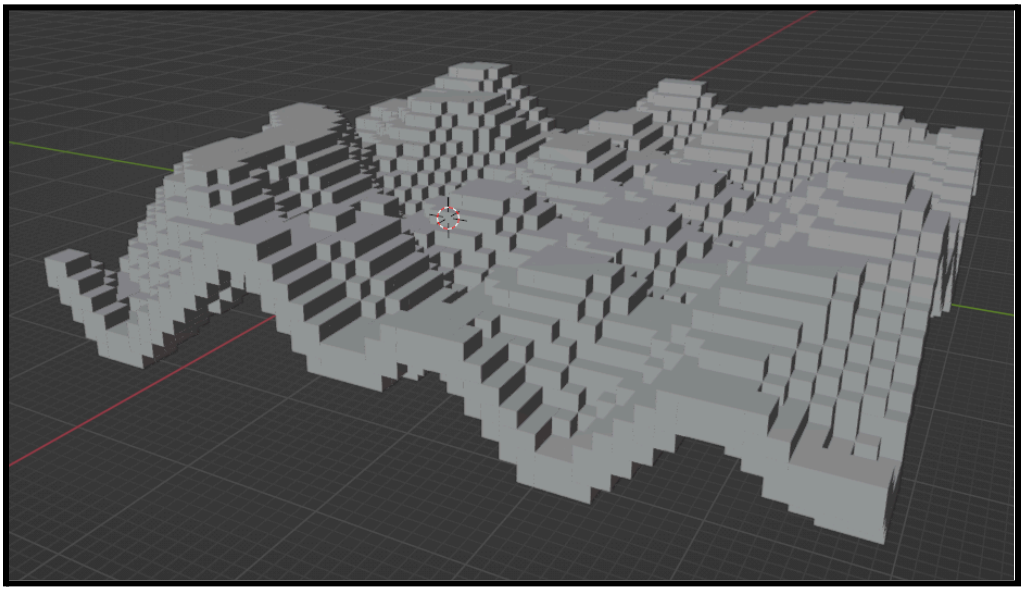
Up until now, we have developed a working implementation of the Spatial Partitioning approach on CUDA as stated in our initial proposal schedule. Our implementation allows for tuning hyperparameters such as scale, persistence, lacunarity, and octaves. Moreover, we have deployed a Blender script prototype used for visualizing the outputs of our CUDA program. The script is fully functional and produces renders displayed in the following sections. We have also implemented Fractal Brownian Motion (FBM), where different layers of Perlin Noise can be layered on top of each other – this is controlled by the octave, lacunarity, and persistence parameters. Overall, this amounts to roughly 40%-50% of our project being complete at this time.

Currently, we have not yet implemented the second proposed approach to Perlin noise generation in CUDA, Temporal Partitioning (unlike what we had originally planned in our proposal schedule). Though this is a daunting task (and will be our focus this week), we are confident that we can come up with a working implementation faster than the Spatial Partitioning approach. That is because Temporal Partitioning shares a lot of functionality with Spatial Partitioning, meaning that implementing TP will be equivalent to modifying SP rather than producing new code from scratch.

Preliminary Results

We successfully tested the “Spatial Partitioning” (parallel) version of our Perlin Noise Terrain Generation program, running on an Nvidia GPU on the Gates machines. We rendered the resulting noise map using Blender, here are some of the renders:





Overall, we are pleased with this result. Our program generated realistic terrain within a reasonable time frame. It runs very quickly

Issues & Concerns

We are currently treating Voronoi noise as a far-end goal which we would like to implement if time permits. With that in mind, we recognize that to properly integrate Voronoi into our current project, we will need to research the topic in more depth and detail. Specifically, we need to learn more about the mathematics and algorithms behind Voronoi Noise generation. Because of this, there are still some "unknowns floating about" regarding a potential Voronoi implementation. However, there appear to be plenty of resources available online regarding Voronoi Noise, and we are hopeful that we'll get it to work over the next two weeks.

Other than that, we have done all of the necessary research regarding Perlin Noise, and have spent a lot of time developing both of our parallel approaches. It's just a matter of coding to get our Temporal Partitioning implementation up and running. We need to ensure that we're efficiently using shared memory and correctly indexing the appropriate values.

Current and Future Goals/Deliverables

Current Deliverables (Completed)

We have completed the following deliverables, that we originally planned in our project proposal:

- **Implemented** a fully functional implementation of the CUDA Spatial Partitioning approach (1 of 2 approaches)
- **Wrote** a functional Blender script for the visualization of CUDA results
- **Developed** infrastructure for properly building and launching our CUDA implementation and passing in hyperparameters
- **Implemented** Fractal Brownian Motion and included relevant hyperparameters for that (octaves, lacunarity, persistence)
- **Created** a serial reference implementation of Perlin Noise

Overall, we have completed about 40-50% of our planned deliverables, and are on track to complete 100% by the end of next week.

Future Deliverables

- **Implement** a fully functional CUDA Temporal Partitioning approach in addition to the existing SP approach
- **Improve** Blender visualization for more compelling renders (include textures and better lighting), and make rendering faster
- **Integrate** more robust infrastructure that fully encloses both CUDA approaches and ties them together to the Blender script (allowing an automatic transition between CUDA and Blender visualization)
- **Introduce** some form of Voronoi noise with the purpose of biome generation (Far-end goal)

Poster Session Plan

During the poster session, we plan to show various renders of the terrain we have generated that will look similar to the images in the "Preliminary Results" section. This will include renders with different terrain features. In addition, we will be presenting graphs and analyses of our two different Perlin Noise implementations. We will include speedup graphs where various hyperparameters are varied, including the image size, initial grid size, number of octaves, persistence, and lacunarity. Additionally, we will compare the sensitivity of each of the two approaches due to changes in the problem size, and the specific hyperparameters mentioned above.

Ideally, we also want to perform a demo at the poster session, where we are able to render an image of terrain from our Perlin noise kernel. We can't render terrain in real-time, but we would like to have a demonstration of our Perlin noise generation and rendering during the poster session.

\

Schedule for the Coming Weeks

Week 4 Half 1, Dec 3 - Dec 5:	● Minor improvements on currently fulfilled deliverables: ○ Timing code ○ Command line arguments ○ Blender script improvements ● Begin implementing Temporal Partitioning ○ Have an outline of the algorithm ○ Determine strategy for use of shared memory ○ Begin writing code
Week 4 Half 2, Dec 6 - Dec 8	● Complete/debug Temporal Partitioning ● Investigate the feasibility of introducing Voronoi
Week 5 Half 1, Dec 9 - Dec 11:	● Finalize infrastructure code for automating noise map generation/visualization ● Begin writing a comparative analysis between different approaches to Perlin Noise
Week 5 Half 2, Dec 12 - Dec 13:	● Wrap up analysis and prepare poster presentation ● Tie any remaining loose threads

Note: We will be completing all of the above tasks together, while both present in person or virtually