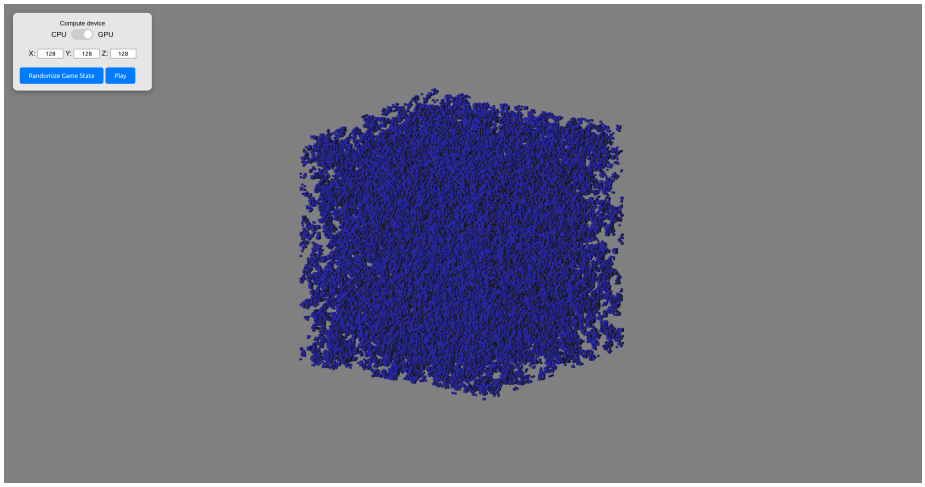
TÖL105M - 3D Game of Life

Bence Koczogh

October 2024

1 Introduction

The game can be found on this link: https://superb-souffle-fa1050.netlify.app/
It is a basic implementation of Conway's Game of Life in 3D with only minimal controls.



2 Features

- GPU and CPU compute paths
- Custom game board size
- Camera zoom and rotation
- Pause & resume simulation at any time
- Switch between CPU and GPU mid-game without pausing

3 Implementation

3.1 Game board

The game board is initialised with random values, where each cell has a 15% chance of being "alive". The default size is 10x10x10, but there is no technical limitation on size other than the hardware. It is treated as an infinitely repeating field. This means that cells on opposite edges are treated as if they were neighbouring each other, effectively wrapping around. While playing, the simulation runs every 500 milliseconds.

3.2 GPU compute

By far the most notable feature of the game is the GPU compute path. This allows the game logic to run on the GPU in the form of a fragment shader, allowing for far larger game fields. I originally wanted to use compute shaders for this task, but WebGL never fully implemented support for compute shaders. Instead, the game field is stored in a 3D texture, with each pixel representing one cell. The fragment shader checks neighbouring pixels, and outputs a colour (R0 or R1) depending on whether the cell is alive or not. This is rendered to a framebuffer for each vertical slice. Due to draw call overhead, the GPU implementation generally scales better in the X and Y dimensions than in Z, as computing each Z layer requires an additional draw call. This, however, is then outweighed by the even greater overhead added by parsing large textures in JavaScript in some situations.

3.3 CPU compute

While it is the default option, it is considered to be more of a fallback when GPU compute is not available. It is faster than the GPU on small game boards such as 10x10x10, which is the reason why it is the default. While I did some very basic optimisations to the code, a lot more could be done to speed it up. I originally added it only to validate the correctness of the GPU compute path, so it was not built with speed in mind.

3.4 Rendering

This game uses instanced drawing as there can be a lot of cubes on screen at once. It is still rather slow with 256x256x256 boards and greater (roughly 4 million active cells), not including the game logic. The shading is rather basic, with only diffuse and ambient light, the code for which I adapted from an earlier OpenGL project I did.