Srayan Jana

CS 112

Shuang Zhao

Fall 2021

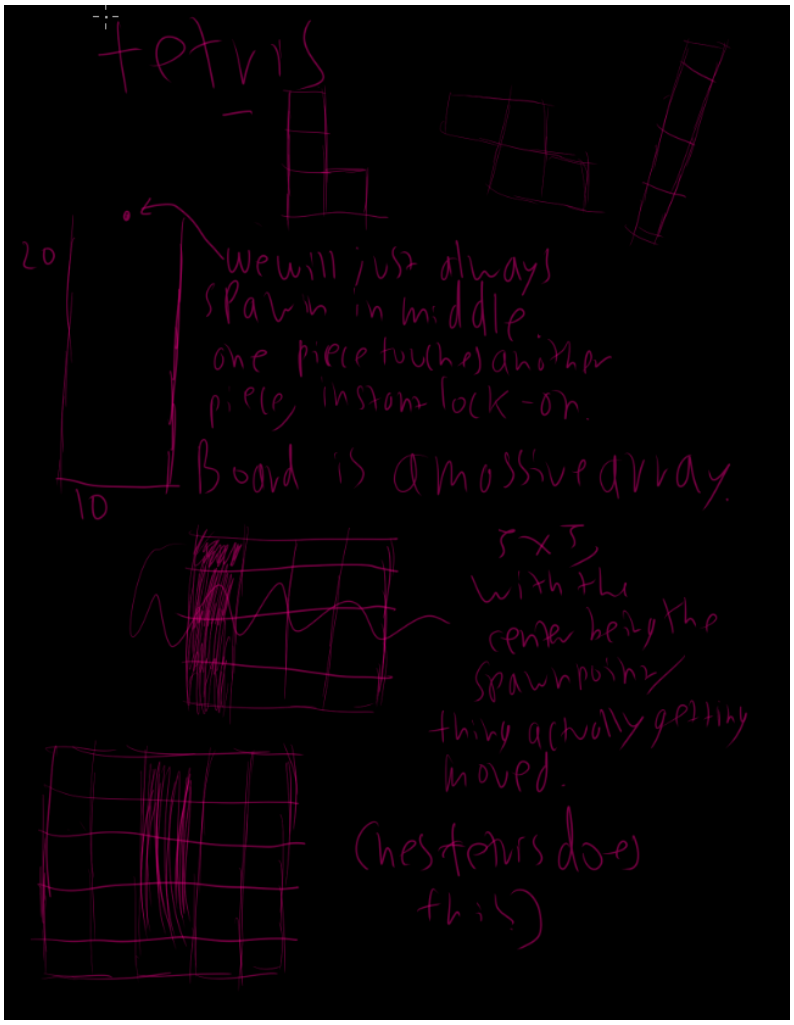
Hi, my name is Srayan Jana and I attempted to make a Tetris clone in wgpu-rs. Wgpu-rs is a Rust implementation of the upcoming WebGPU standard along with the WGSL shading language. Since this is a graphics class and not a games class, I'll largely be porting over existing tetris projects like https://github.com/DavideCanton/rust-tetris over to use my renderer instead.

In order to accomplish this, I went through the Learn wgpu tutorial (https://sotrh.github.io/learn-wgpu/#what-is-wgpu), and I've learned a lot about how the API works. It reminded me a lot of project 1 and how webgl2 draws vertices. However, I think I might have overscoped my workload, since I'm not super sure how to take the simple textured hexagon the tutorial helps you make and turn it into a full game.  First of all, it took a while to figure out how to get a textured square working in the first place, since I had a lot of trouble trying to figure out how to adjust the indices to make the shape of the square. As soon as I did that, I used wgpu's instancing API to spawn squares in the shape of a grid. I was even able to test out my grid's ability to have squares added dynamically by having my program spawn random placements of squares, which proved I had a good proof of concept.

Originally, I was just going to port this: https://github.com/DavideCanton/rust-tetris and use my wgpu renderer, but I found that difficult. Then, I realized that the codebase I wanted to port was much too complicated, especially since webgpu didn't seem to integrate nicely with the existing code. I thought that maybe integrating my rendering code for instancing would be good with this rust tetris version https://github.com/mikejquinn/rust-tetris since it worked in the terminal and didn't seem to have much in the way of rendering code but that failed as well. The

input system and the gameplay loop that codebase used was too different from how wgpu organizes its event loop.

I then realized that it might actually be easier to build my own custom tetris game rather than porting someone else's since I can better customize it to how gpu works. I started looking at this great article on the nes tetris source code: https://meatfighter.com/nintendotetrisai/, and did a mockup of the algorithm I could use to implement Tetris NES style.




Eventually, I realized that I was spending too much time getting the game to work than actually doing any graphics programming. So i decided that I would instead switch to implementing the game of life, as seen here in this article.https://dev.to/dineshgdk/game-of-life-in-rust-4mfc I was able to retool the game of life

function into an update method into what was supposed to be my tetris board, but then I kept running into runtime errors because iterating through the vectors wasn't working.

Eventually, I got so tired that I just decided to have the program spawn in a random group of textured squares on screen every second. I gave up that night and went to sleep, but in the morning, I got a flash of inspiration and finally got the game of life code working. There's still some more I can add, such as a text on the screen saying what generation the game is in right now.

Additionally, even though in my original proposal I said that I was going to have a web build, I have changed my mind and I'm just going to have a native build instead. The reason is that WebGPU on browsers is still incredibly experimental and buggy, and the webgl2 export for wgpu-rs is also unstable right now.

Overall, I think this has been a really fun yet extremely frustrating learning experience, and while I think this has cemented my love for Rust, I don't think I'd like to work in graphics. I find the process of working on graphics not very fun, since there seems to be a lot of boilerplate for the simplest functions. Maybe it's just because this is very low level graphics, but this isn't for me.