Cameron Churchwell

Cnc5166

Project A: Fever Dream

The broad goal of this project was to build a codebase for simulating particle systems in WebGL. I ended up adding the additional challenge of converting from javascript to typescript, adding an actual dev environment, optimizing some of the math, and adding abstractions and object-oriented ideas like classes and interfaces.

Users can use WASD to move around and the arrow keys to look around. Help information is displayed on screen under the canvas element. Users can explore the 3D world to see flying saucers that run away but never stray too far from the origin, a continuously burning voxel-like fire, a hanging rope swaying in the breeze, a tornado, and a set of red lanterns hanging in the air. Users can press R to reset all of the particle systems and P will play them again. The space bar moves a single step forward.

In addition, there are radio buttons to select which solver should be used for the rope. The default is the reverse iterative midpoint solver which I implemented at 20 backwards time steps. The naive solver is the next best but all of them are unstable except for the backwards iterative solver. I also implemented the forward midpoint method but it was not that much better than the default implicit solver.

In terms of additional constraints, I added ConstraintFixed which fixes specific particles in a system to specific positions (like the ends of the rope). I also added a kill plane constraint which resets a particle back to its initial state, which is used in the fire and the tornado.

In terms of additional force generating objects, I added a parametric time based force generator which can be thought of as a vector field with the additional component of time information. This allowed me to implement very different ideas like wind and the random launching of the fire very easily. I implemented the boids using two force generating objects, one with the flocking behavior and the other with user avoidance. I also abstracted my ForceGenerator class to ChangeGenerator, which allowed me to implement MassChangeGenerator and ColorChangeGenerator, though it is debatable whether or not those should count as additional force generating objects. I also added a containment force to send the saucers back towards the center when they got too far away.

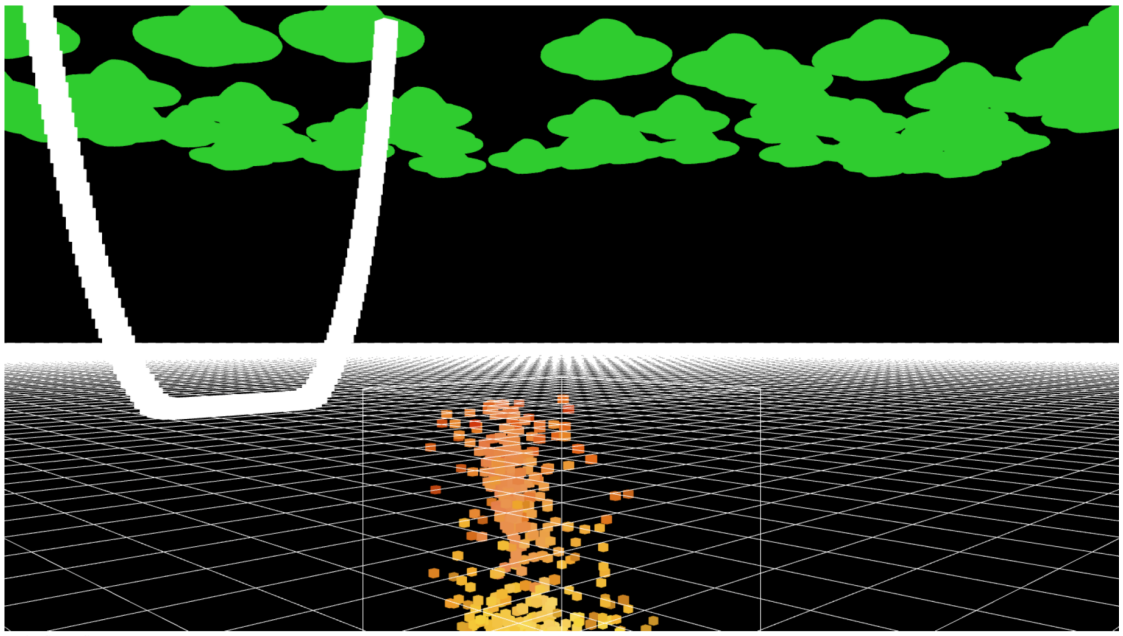
The greatest feature is likely the rendering of 3D objects at particle positions at very high frame rates. I use vertex pulling to read positions from a uniform buffer that I write with 100 positions at a time. This means that I can draw objects like the UFOs 100 at a time to reduce the time spent waiting on writes to GPU memory, while not having to rely on huge state vectors. In fact, all of my particles are stored separately and yet I still achieve this performance. On my machine, I notice no drop in FPS so it is likely running at at least 30fps.

Another thing that helped with performance was that I added many functions to the matrix library, most notably in-place operations like in-place copying, addition, subtraction, and scaling. This reduces the number of Vertex3 allocations that are needed for each time step across the particle systems massively, and I saw very real performance gains from this as well. All of this together allows me to render thousands of particles on screen. For example, my tornado system is 4000 particles and the fire system is 1000 particles.

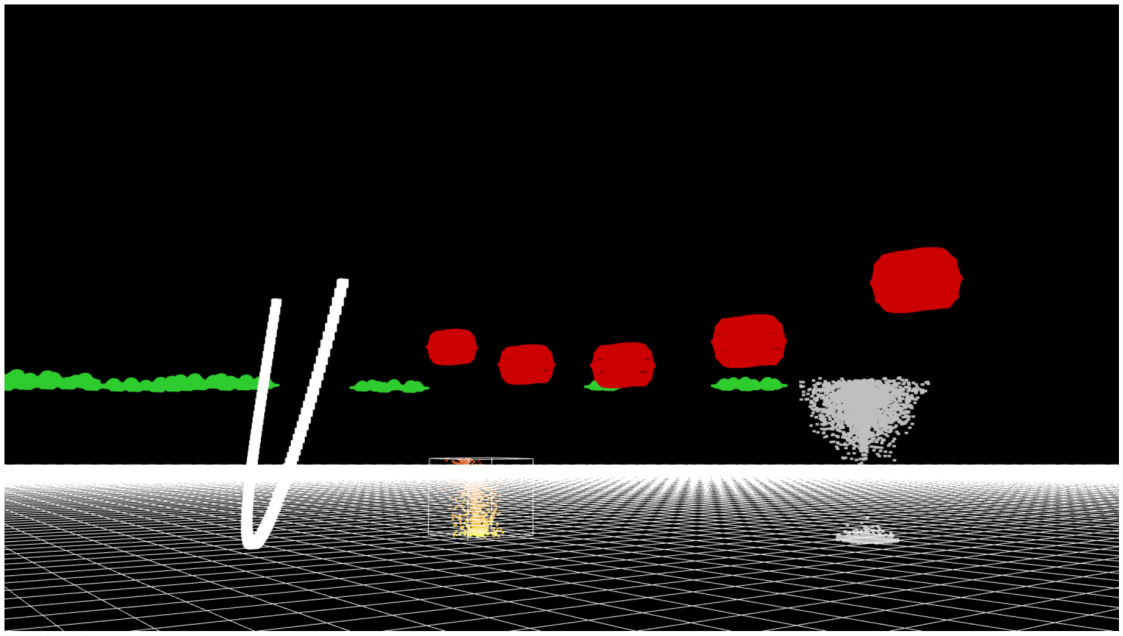
In addition to adding all of these abstractions, I was able to write to many different files instead of a few very large files. This means that my code base should be fairly easy to navigate at a glance as I tried to name all of the directories sensible things. I accomplished this by drawing on my previous experience with web development and I used webpack to bundle all of the files together into a single javascript file which is included in the HTML file, preventing CORS errors while allowing for intelligent code completion and automatic import generation between files.

Webpack also handles the transpiling between typescript and javascript, and also allows me to store vertex information in json files and then require it. The benefit of this is that IDEs will not try to parse json files in the same way, which I found to kill performance in VS code and make webpack compile times much longer. I also installed a loader for GLSL, which allowed me to have my shaders in their own '.glsl' files which yields a much nicer development environment. Note that webpack can be improved further to cache changes when in "watch" mode, which could reduce build times even further, but I was able to build in under 3 seconds even when my project was complete.

Below are screenshots of my project:

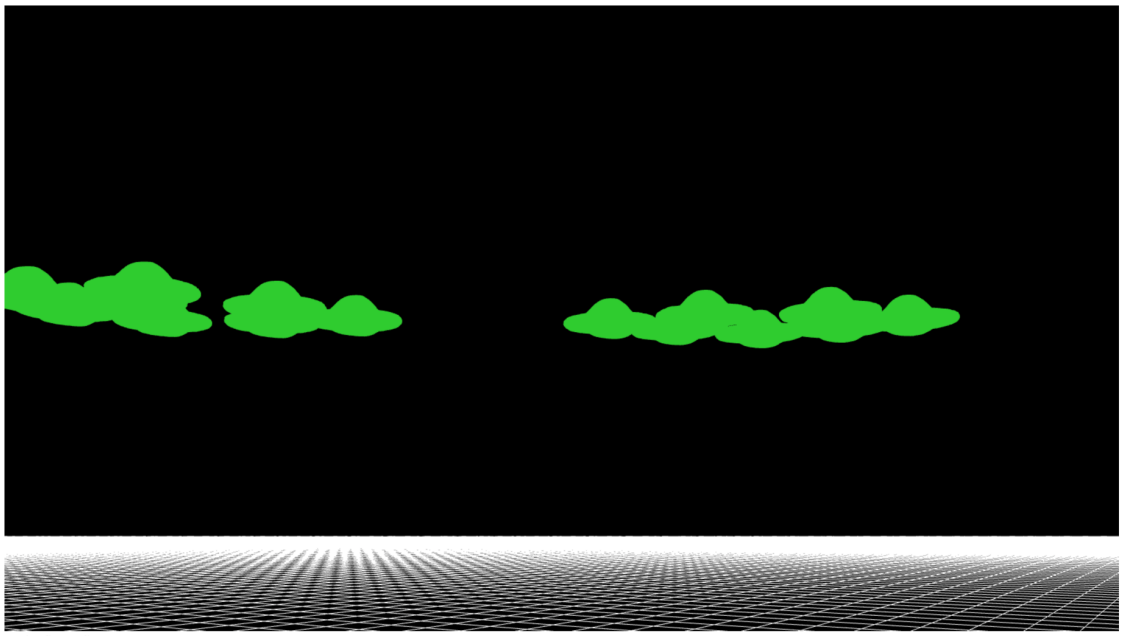


Particle System Controls:
R key: hard reset. p or P key; pause/resume. Space-bar: single-step.
WASD to move, Arrow keys to look around. Note that moving vertically is disabled
The flying saucers will run away from you if you move under them (aliens are secretive)
Select the solver for the chain system: ○ euler ○ naive ○ Forward midpoint ◉ Backward iterative



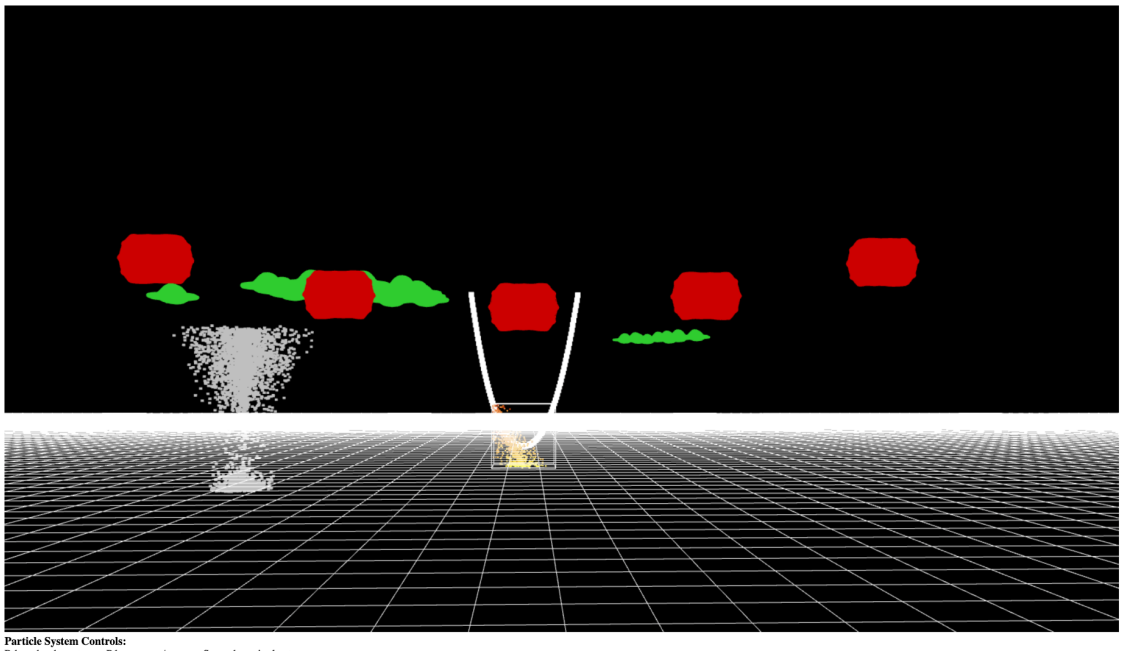
Particle System Controls:
R key: hard reset. p or P key; pause/resume. Space-bar: single-step.
WASD to move, Arrow keys to look around. Note that moving vertically is disabled
The flying saucers will run away from you if you move under them (aliens are secretive)
Select the solver for the chain system: ○ euler ○ naive ○ Forward midpoint ◉ Backward iterative

If you chase the saucers, you can part them as they attempt to both flee from you and return towards the center.



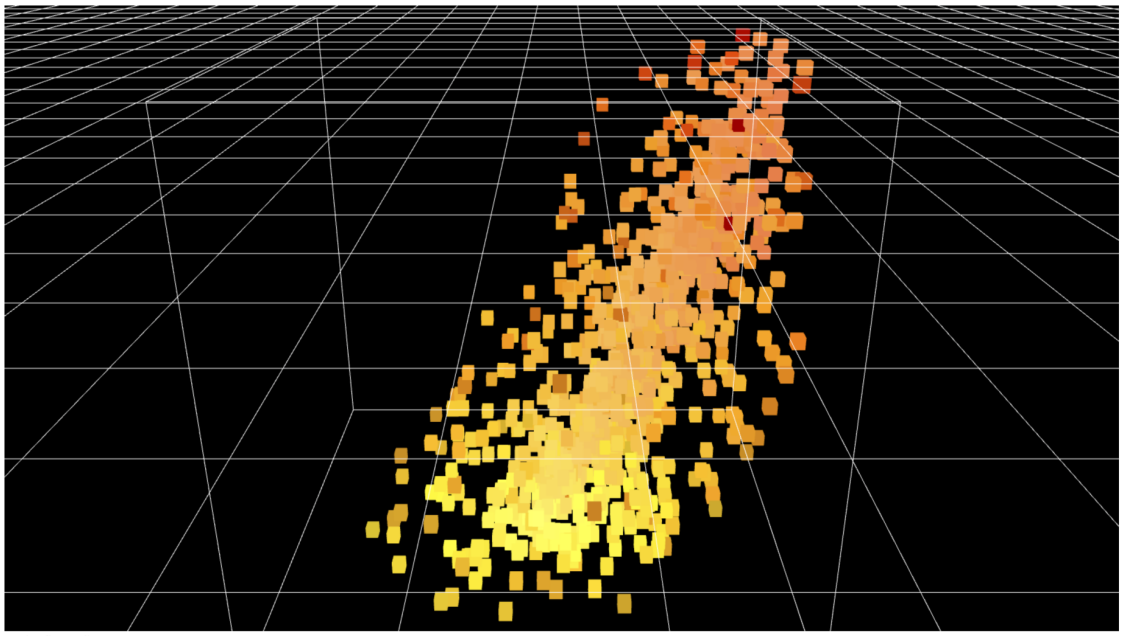
Particle System Controls:
R key: hard reset. p or P key; pause/resume. Space-bar: single-step.
WASD to move, Arrow keys to look around. Note that moving vertically is disabled
The flying saucers will run away from you if you move under them (aliens are secretive)
Select the solver for the chain system: ○ euler ○ naive ○ Forward midpoint ◉ Backward iterative

I really like how the lanterns turned out! The saucers too.



Particle System Controls:
R key: hard reset. p or P key; pause/resume. Space-bar: single-step.
WASD to move, Arrow keys to look around. Note that moving vertically is disabled
The flying saucers will run away from you if you move under them (aliens are secretive)
Select the solver for the chain system: ○ euler ○ naive ○ Forward midpoint ◉ Backward iterative

You can see that the fire particles are rendered as cube objects of varying colors. This is because I also write color information as a uniform during draw calls.



Particle System Controls:
R key: hard reset. p or P key; pause/resume. Space-bar: single-step.
WASD to move, Arrow keys to look around. Note that moving vertically is disabled
The flying saucers will run away from you if you move under them (aliens are secretive)
Select the solver for the chain system: ○ euler ○ naive ○ Forward midpoint ◉ Backward iterative