Free Flowing Particles

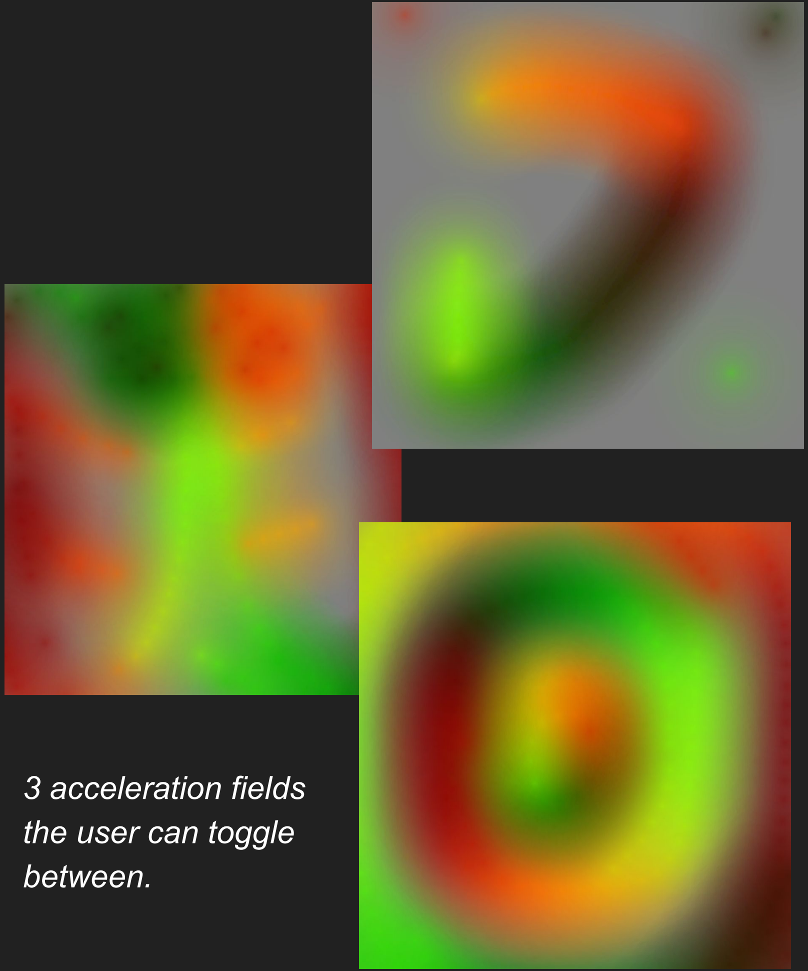
Skyler Krouse, Connor Pugh, Matthew Quinn, Zachary Schecter

The Inspiration...

What We Accomplished

Loaded Textures

Textures are loaded in advance to set the color, mass, and acceleration of the particles. Multiple acceleration field textures are loaded to allow the user to switch between them and see different motion effects.



3 acceleration fields the user can toggle between.

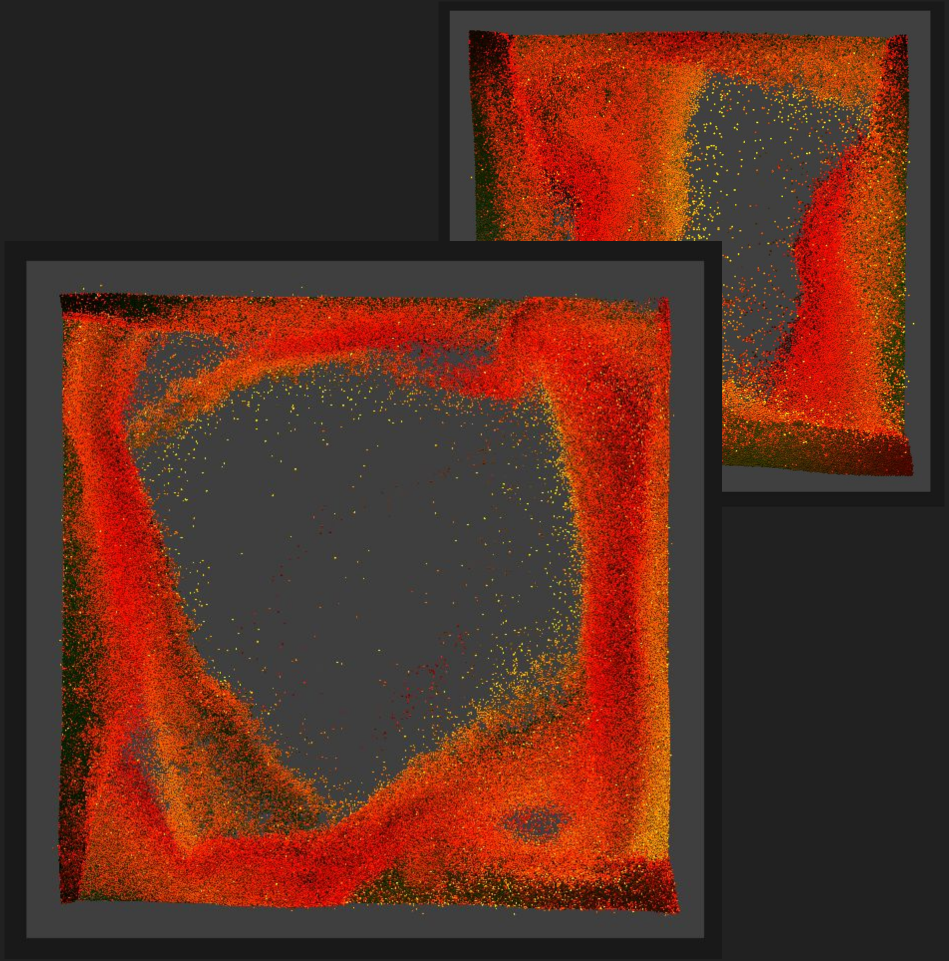
3D Collisions & Acceleration Fields

The particle space is in 3 dimensions, mimicking a box full of particles we are looking at from above.

Acceleration fields defined by textures drive movement in the scene.

Darker particles are heavier

Particles that are moving fast turn yellow/green

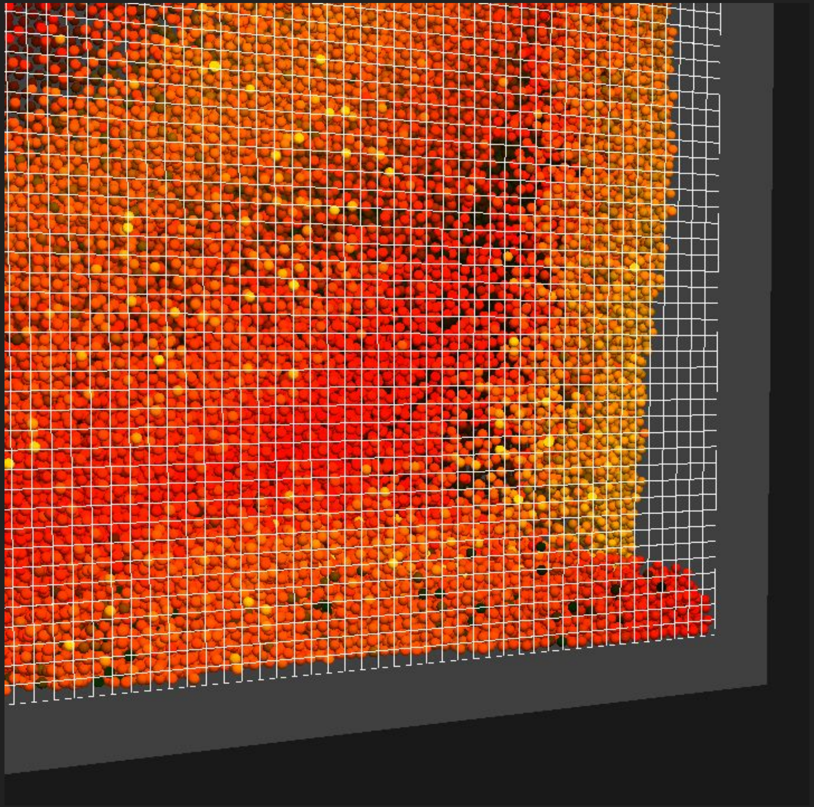


Collision Optimization

To improve collision detection, implemented 2D spatial subdivision; indices are stored in cells within a partitioned list, only particles in neighboring cells are checked for collision.

Massive performance boost- 60+fps with 200k particles

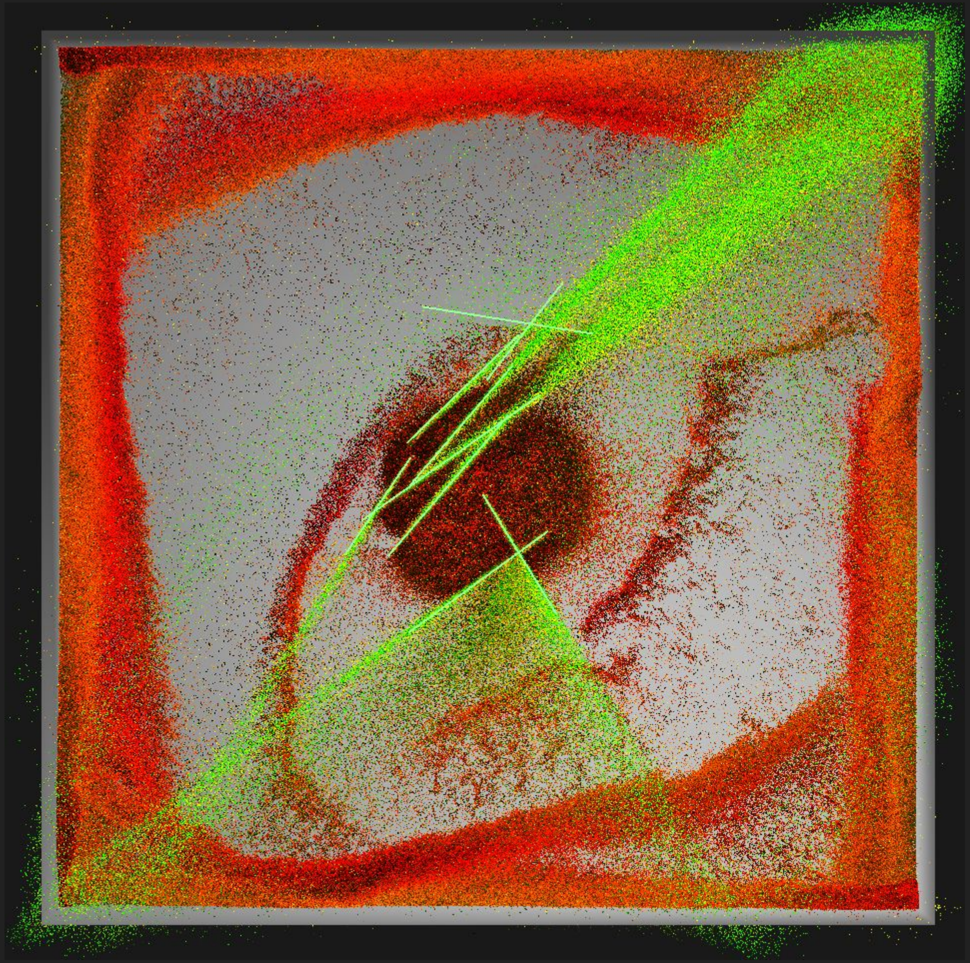
Collision response is a single step solver instead of the iterative solver



Particle Life

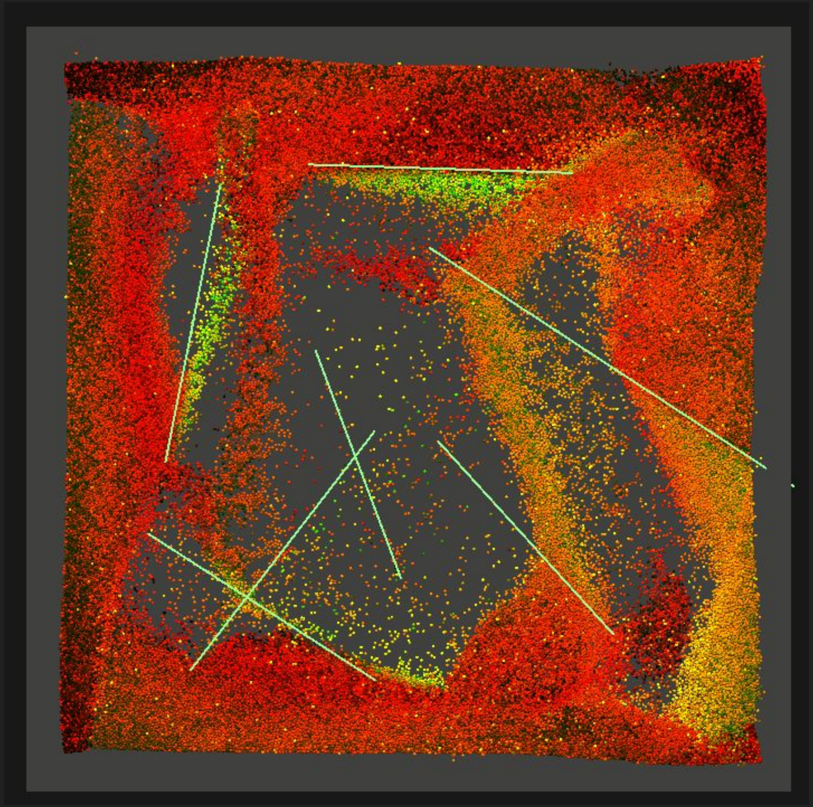
When feature is enabled, particles can have a random lifespan (selected from a predetermined range) and respawn when their lifespan ends.

The user can set the spawn point of the particles to be on a radius around a given point.



Interactive Acceleration Lines

Users can draw lines on the screen to accelerate particles along them, as if creating a current in the particle field. This is done by taking the mouse input of users dragging a line and accelerating particles that touch the line accordingly.

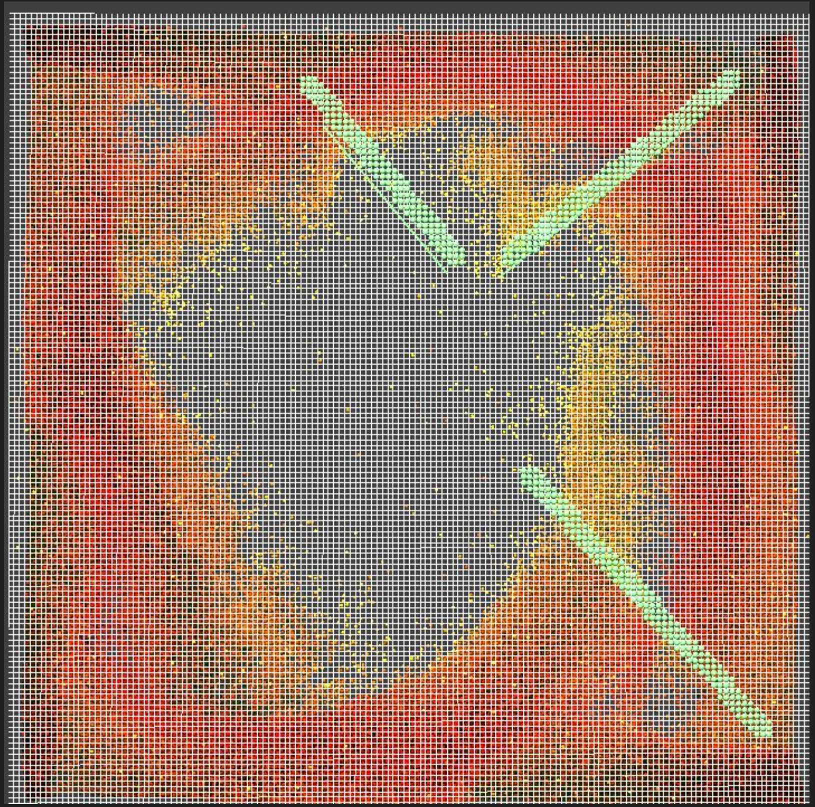


Line Optimization

Lines reuse collision cell data, only iterating over particles within nearby cells

List of nearby cells is built and retained on line creation; each frame, iterates over particles in each cell created during collision checks

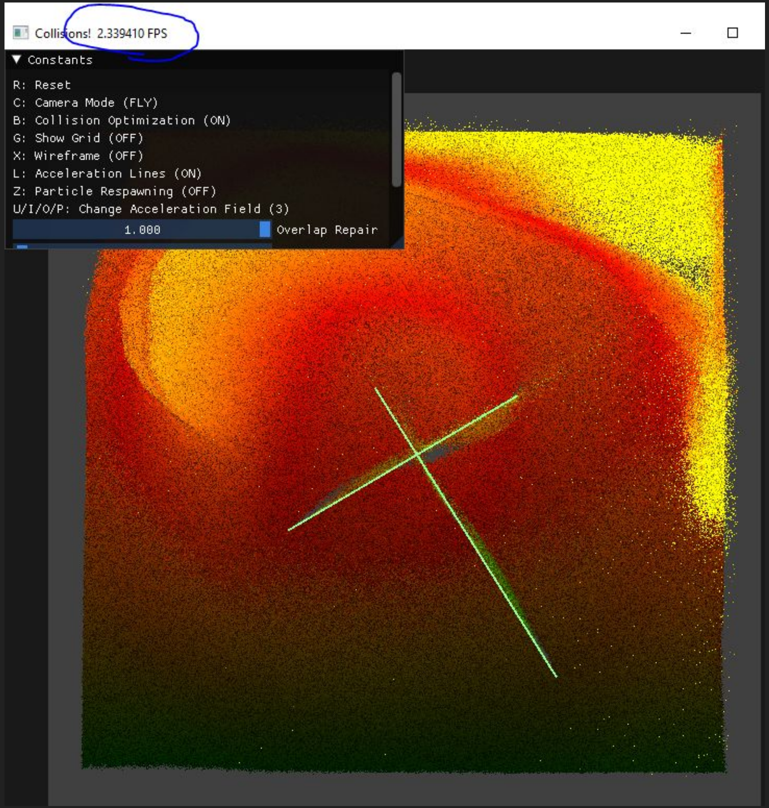
Uses Bresenham's line algorithm



Further Potential Optimizations

We are currently not storing our particle data inside of textures; with the amount of particles we are reaching, it would probably be optimal to switch to a texture-based system.

Finer control over the shader code would probably allow for some minor optimizations, as Taichi only parallelizes the outer loops in kernels. More performance could definitely be achieved with a bespoke C++ implementation to get rid of the Taichi overhead.



```
Collisions! 2.339410 FPS
▼ Constants
R: Reset
C: Camera Mode (FLY)
B: Collision Optimization (ON)
G: Show Grid (OFF)
X: Wireframe (OFF)
L: Acceleration Lines (ON)
Z: Particle Respawning (OFF)
U/I/O/P: Change Acceleration Field (3)
1.000 Overlap Repair
```

Running the sim at 10 million particles… quite slow!