Colorful Natural Disasters

Adrien Katsuya Tateno

Goals

Implement a state-based particle system using the parallel processing capabilities of graphics hardware. Create a tornado by means of a vector field. Make particles demonstrate flocking patterns. Implement particles with a limited lifetime. Render the particle system in three dimensions with meaningful color. Allow the user to interact with the particle system. Create areas in the system through which particles cannot pass. Demonstrate the effects of explicit and semi-implicit methods for differential equations.

Technologies Used

WebGL OES_texture_float Khronos Ratified Extension Specification
Render to texture is impossible without the use of floating point textures.

stats.js - https://github.com/mrdoob/stats.js/
Used to display rendering Frames Per Second. Helps to identify performance issues.

dat-gui - https://code.google.com/p/dat-gui/
Commonly used control panel for WebGL applications. Used for interacting with the flame, playing/pausing the simulation, and switching between the Runge-Kutta 4 and Euler's method.

glMatrix - http://glmatrix.net/
Fast matrix library for working with the camera and user controls.

User Guide

The program starts paused. Click the PlayPause button on the control panel to start. If at any time you would like to switch from Euler's method to Runge-Kutta 4 or vice-versa, click ToggleSolver. You may change the emission point of the flame via the fire_x and fire_y sliders also located on the control panel.

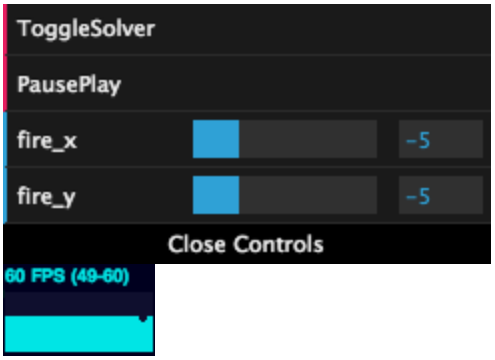
- W to move forward
- S to move backward
- D to move right
- A to move left
- E to move up
- Q to move down

- Change direction of facing with the arrow keys

Code Guide

The code uses a method known as render to texture in which WebGL uses framebuffer objects to store state and differential data in texture, so it can be later accessed through that texture in another shader program. This program implements 5 shader programs.

The physics program calculates the time differential of the state given a state. The calculation program takes a time delta and calculates a new state from a state and a state time differential. The solver program combines multiple state time differentials into one weighted state time differential using Runge-Kutta 4. The draw program interprets a state texture as a list of positions corresponding to each point in the system. It does some color calculation as well as scaling the points depending on their distance from the "screen". The static program simply draws static objects such as the grid, axes, and cubes.

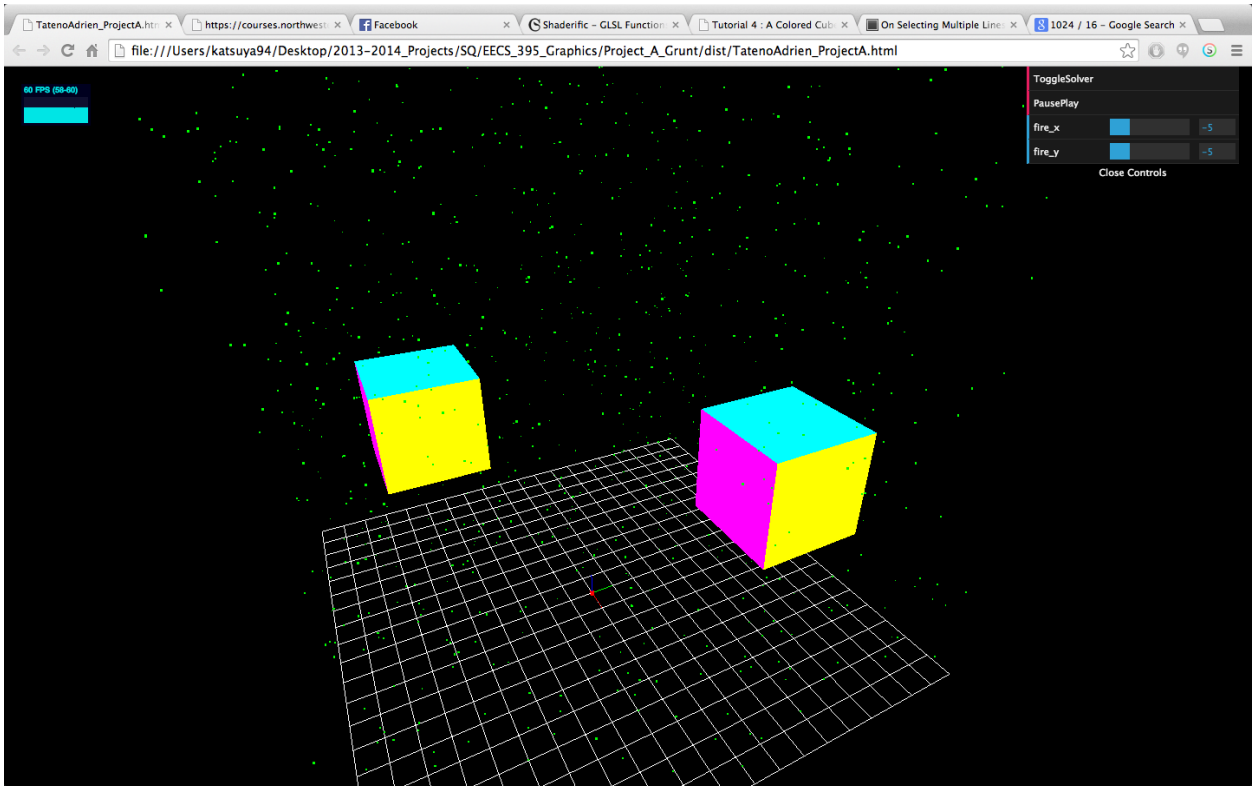


Results

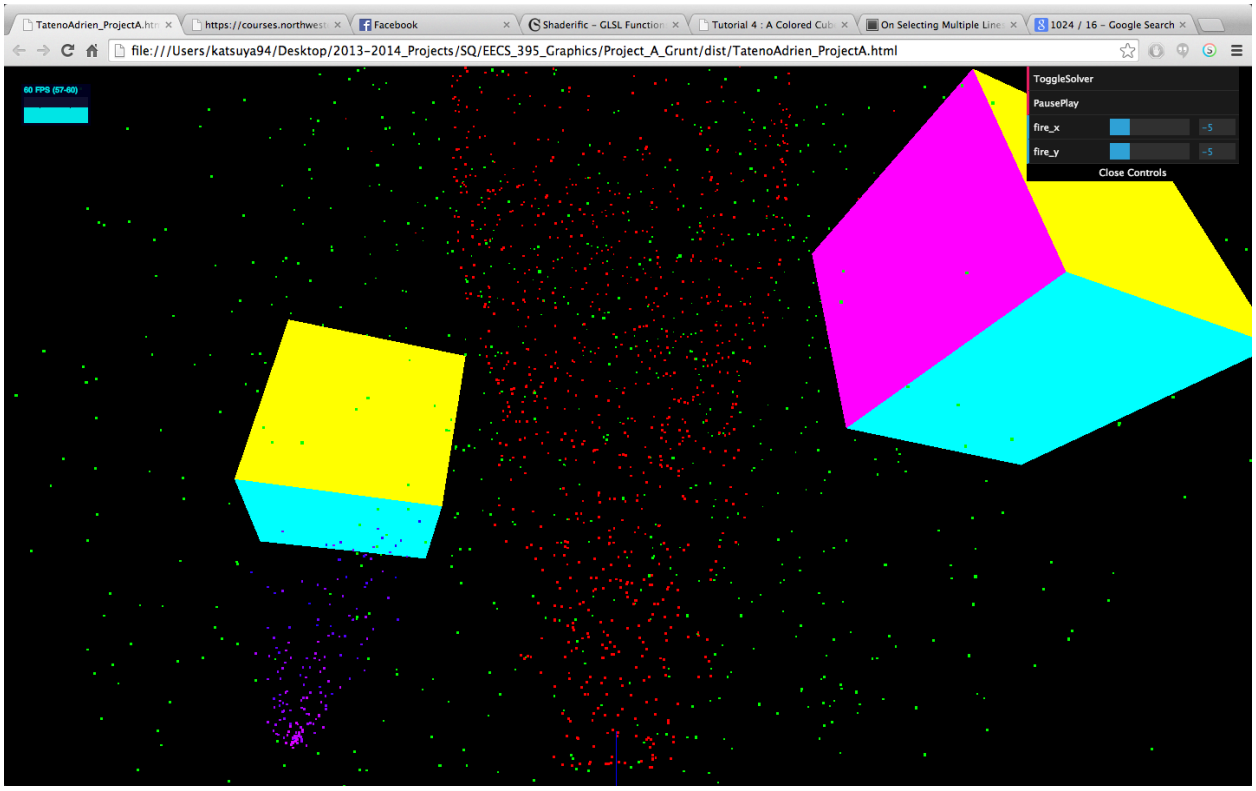
The FPS meter seemed to be consistently getting 60 FPS using Euler's Method, but only about 9 FPS using Runge Kutta 4.

dat-gui has proved to be very useful for WebGL in many projects.

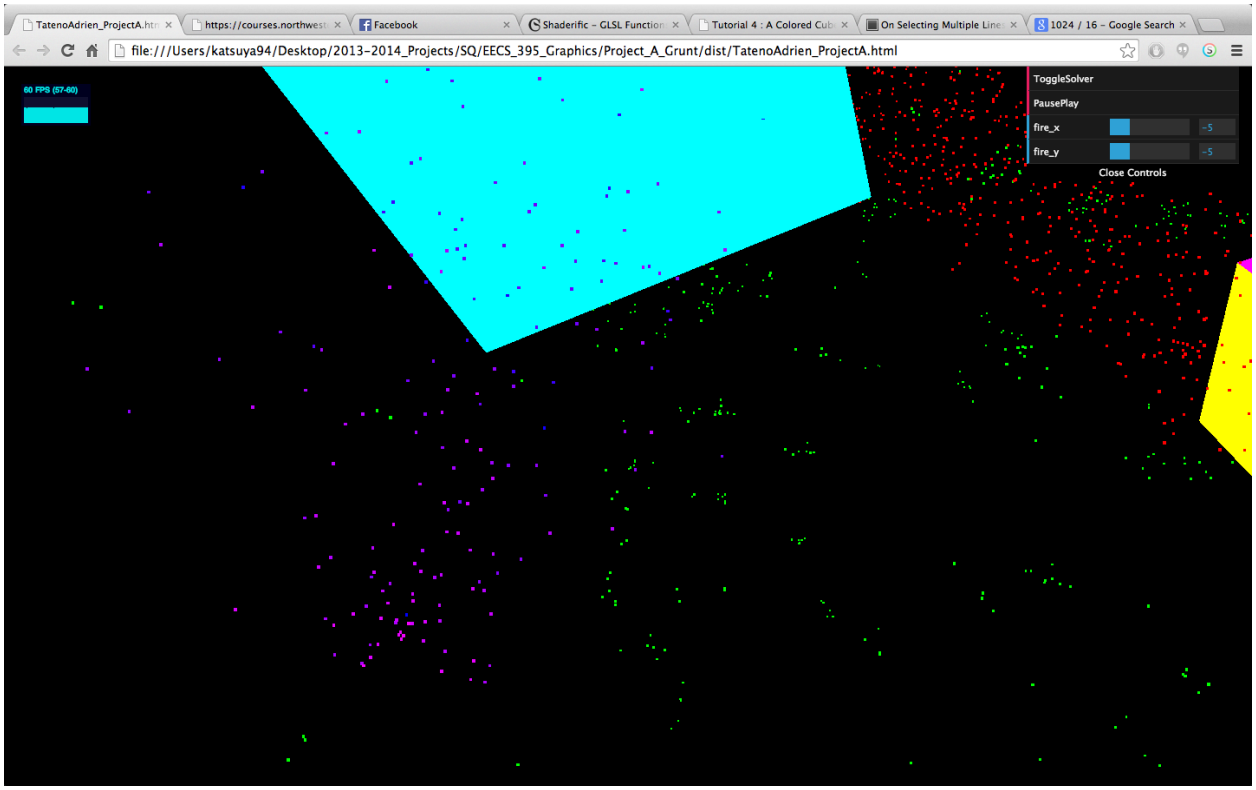
The Initial State of The System



The Tornado in Action



Particles from the Flame and Other Sources Bounce off the Box



As they Settle Down, Boids cluster up and move in the same direction.