Name: Nathaniel Wong
netID: nwy590
EECS 351-2 at Northwestern University
Intermediate Computer Graphics

Project A: A WebGL arena of a flock of Birds, spinning Tornado, volcano Fire, and draped Cloth

Goals

The goal of the project is to demonstrate the capabilities of the WebGL javascript particle system that has been developed in the context of the class, Intermediate Computer Graphics.

The particle system is generalizable to include any number and any type of forces, constraints, numerical solvers (ergo physics).

To demonstrate this, I developed 4 types of particle systems that showcases the possibilities provided by combining forces, and constraints and solvers in meaningful ways:

- Reeves-like bulb of **Fire** made of particles that alter in size and color as they age.
- A **Tornado** system that transports particles around a circular force field, with exponentially increasing distance as height increases.
- A boid particle system that simulates a flock **Birds;** they are attracted and repulsed special particles that simulate a predatory environment
- **Cloth** made up of particles connected by springs and responsive to light wind

User Guide

To run the program, unzip the project folder and open the webpage WongNathaniel_ProjectA.html. You will see a square canvas on the top of the screen that will display the simulation. The user controls are as follows:

W,A,S,D: Pan the camera left, right, up, or down.
Arrow Keys: Zoom the camera in/out or strafe the camera sideways
T: reset the simulation to its original state.
R or 3: run the simulation .
Space or 2: single-step through a frame of the simulation.
P or 1: pause/resume the simulation.
Q: cycles between different numerical solvers available

Code Guide

The ProjectA.js file is a typical webGL animation script that creates a webGL context. It includes **Shader** programs that instruct the hardware shader on the generation of the particles. An animation loop, **tick,** updates and renders the particles systems in turn.

The project is **interactive** since it contains mouse and keyboard event handlers that enable custom user input to have an effect on the simulation. See the functions **myKeyPress() myMouseMove()** and **myMouseUp()** for details.

A the heart of implementation is a prototype object for a generalized particle system, **PartSys**. The object requires access to a global object, **gl**, which is a webGL rendering context object.

PartSys is primarily made unique by its **State** arrays, mainly **s0** which holds the current state of the system, and **s1** is used to calculate the *next* state of the system. Other state arrays like s0dot, s1dot, and s2, all help to determine the value of s1.

PartSys has the following API methods:

- **init(TYPE, ...)** and related functions **initXYZ**: allows a user to initialize a specific type of particle system. Here, the system's **initial State**, **Force** objects, constraint objects (**Limits**), and Solver types are used to define the particle system.
- **solver():** updates the particle system to the next timestep. this involves solving the next state of the particle system given the environmental variables and settings that were set in init(). The implementations of differnt types of numerical physics solvers are defined here, eg. Euler.
- **render():** sends particle system data in **s0** to the Vertex Buffer Object (VBO) in the graphics hardware, and draws particles as described by the VBO and shader programs.

PartSys has the following internal functions which may be of interest to the reader:

- **initVBO():** called within init(). Creates a new VBO in the graphics hardware that will contain vertex data as the particle system is simulated.
- **applyForces(State, CForcer[ ]):** Calculates the total force acting on each particle based on the list of forces described in the CForcer array.
- **dotFinder(s, sdot)**: Calculates the rate of change of each attribute in the state array **s,** as determined by attributes such as force and the type of particle simulation. These values are stored in **sdot.**

- **doConstraints()**: checks the next state of the system, **s1** to see if any constraints have been violated as defined by the array of **Limits** objects. If a particle is in violation, a correction is applied to the particle's attributes.
- Other helper functions: these are quality of life methods support the work of the core functions

PartSys relies on a list of helper files that each define a child object needed to run the library:

- **CPart**: an object that describes a single particle. A number of particle attributes are defined here, include kinematic attributes, color, age, mass, charge and so on. It contains getter and setter methods that allows the data to be copied to the VBO.
- **CForcer**: Describes a force that will act on the particle system. Includes attributes like spring constant, gravitational constant, function of the force field, and the indexes of particles.
- **CLimit**: similar to CForcer, describes a constraint that is acting upon the particles system. Also contains attributes like location of the wall, and the corrective behavior to the particle.

Results

Implemented helpers:

Solvers - Forward Euler, Backward Euler (1 iteration), Midpoint Euler, Symplectic Euler, Verlet

Forcers - Gravity (planetary), gravity (Earth), spring pairs, customizable spring sets, drag, bubble force-fields, custom wind force fields, alignment forces.

Constraints - X/Y/Z axis bounding walls (low and high), Anchor particles, Age constraints

Illustrations of the particles systems:

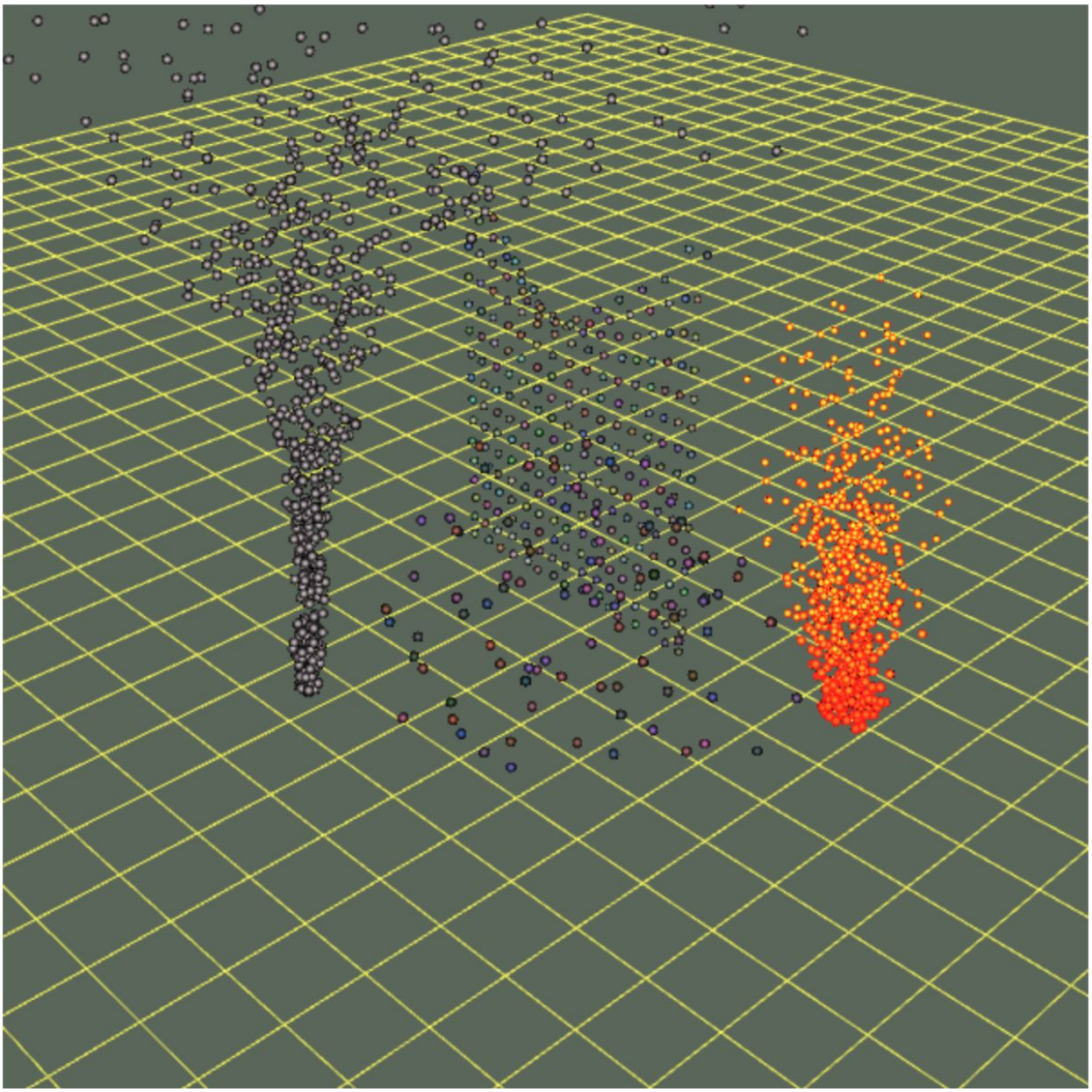

Figure 1: all 4 particle systems in unison on a ground plane grid overlay.

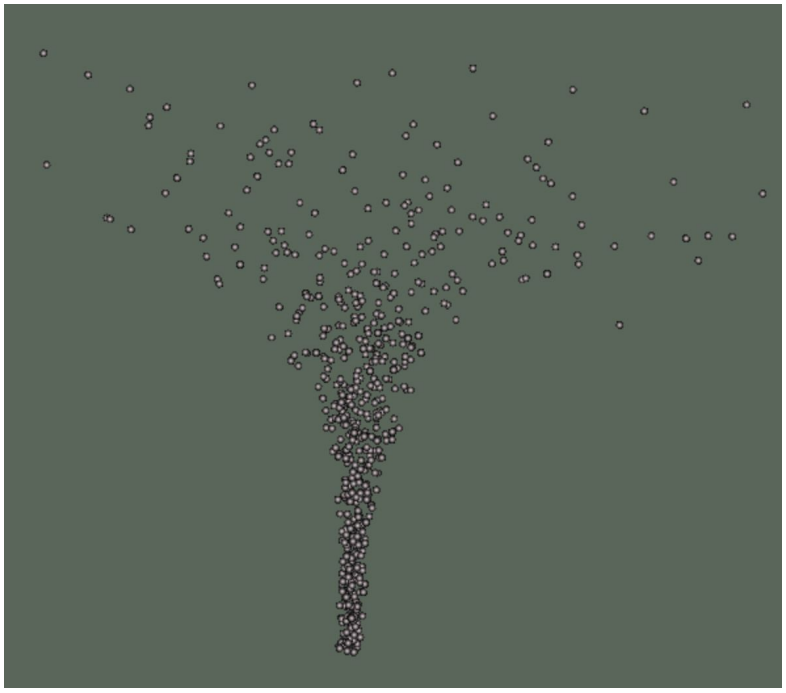

Figure 2: the tornado system featuring a spinning funnel of about 600 particles

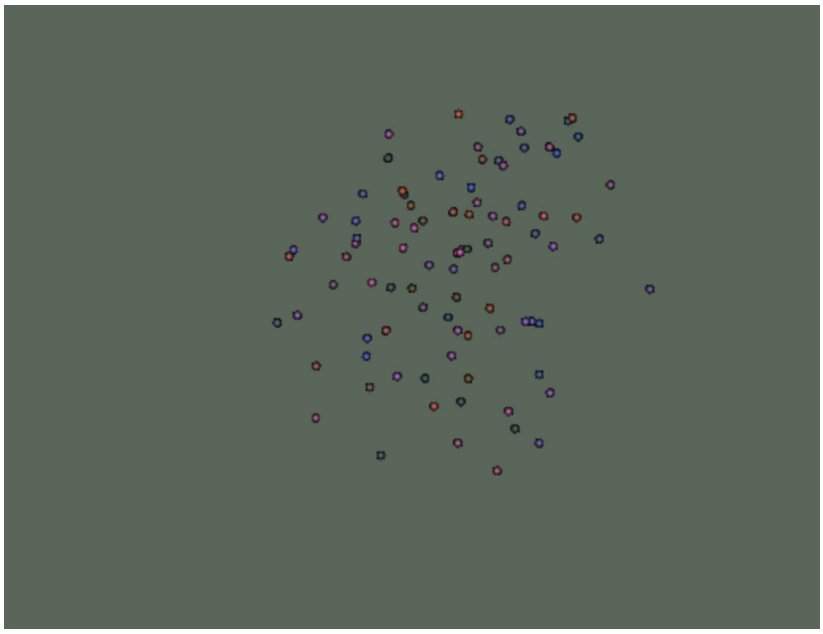

Figure 3: the boids system; 90 birds flock together in space

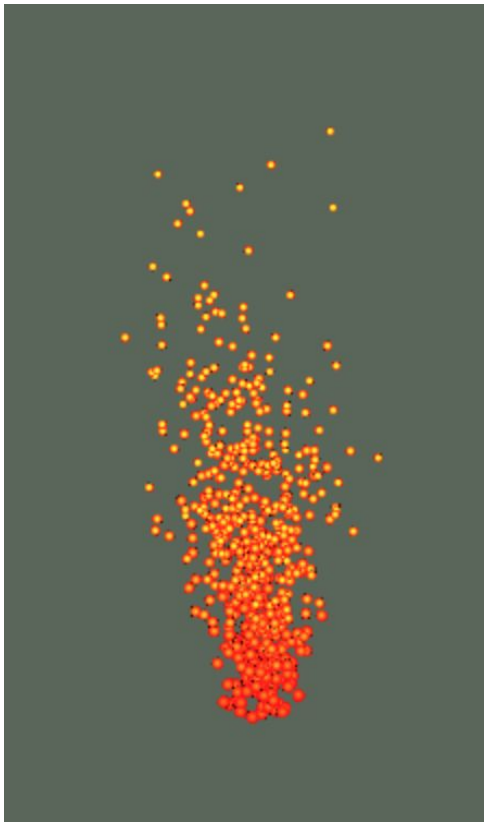

Figure 4: Volcano reeves-like fire the ages with time, color and size of particles change

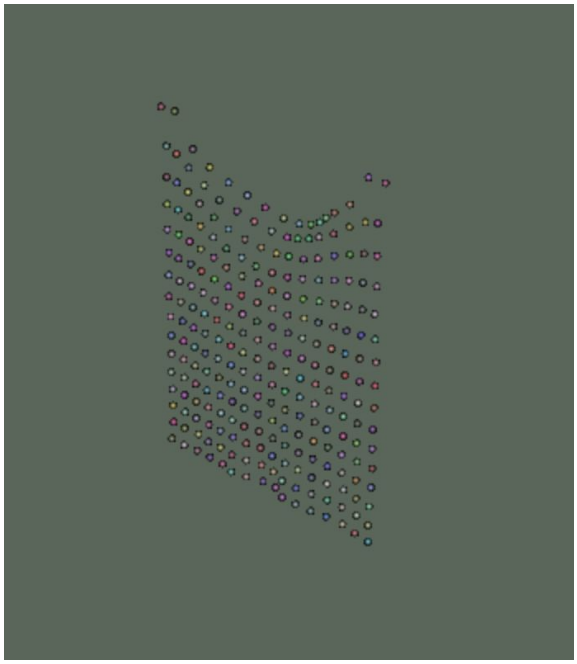

Figure 5: 2D web of particles form a cloth-like system. Particles are held togeth by arrays of spring constraints, namely structural, shear, and bending.