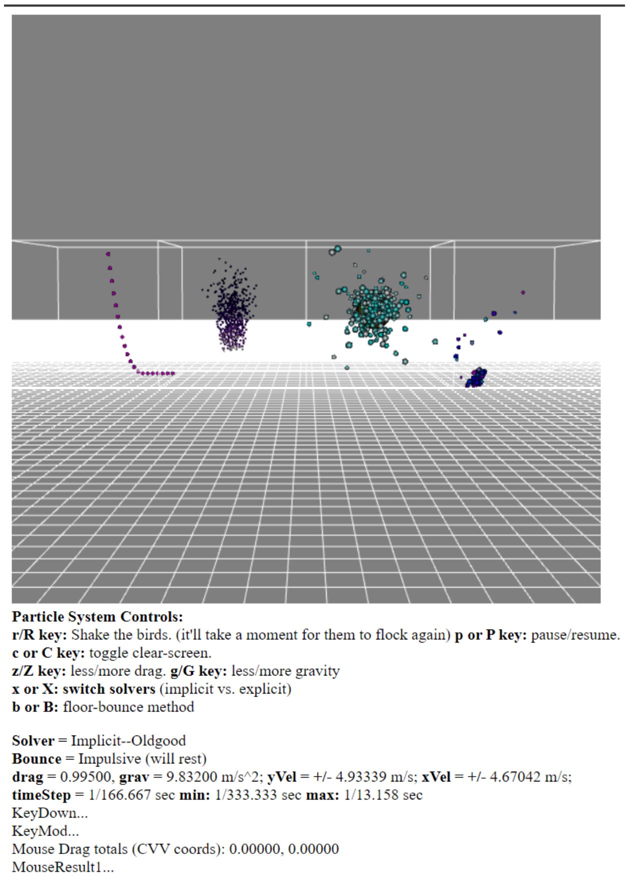
Alex Chen

Aac1384

Particle Systems

Intermediate Graphics

BrightBox: My God this Project And My Winter Quarter Schedule Hurts My Insides And Soul but Boy is That Rope Blowing, That Fire Flaming, Those Planets Orbiting, and Those Birds Flocking



Particle System Controls:
r/R key: Shake the birds. (it'll take a moment for them to flock again) **p or P key:** pause/resume.
c or C key: toggle clear-screen.
z/Z key: less/more drag. **g/G key:** less/more gravity
x or X: switch solvers (implicit vs. explicit)
b or B: floor-bounce method

Solver = Implicit--Oldgood
Bounce = Impulsive (will rest)
drag = 0.99500, **grav** = 9.83200 m/s^2; **yVel** = +/- 4.93339 m/s; **xVel** = +/- 4.67042 m/s;
timeStep = 1/166.667 sec **min:** 1/333.333 sec **max:** 1/13.158 sec
KeyDown...
KeyMod...
Mouse Drag totals (CVV coords): 0.00000, 0.00000
MouseResult1...

Figure 1: Project A initial screen on load.
Left to Right: Rope, Flame, Orbits, Flocks

Essential Instructions:
Camera Control + Movement:
WASD: Move Forwards, Left, Back, and Right respectively
Arrow Keys: Tilt Camera to various directions
F key: Change solvers for all particle systems
B key: Change the bounce method
R key: Give birds additional velocity (it will take a while for them to return to flocking)
P key: Pause
Z and G: modify drag elements

C: change if the screen gets cleared or not per frame

Goals: To simulate several particle systems within WebGL, and simulate the various solvers that can go with them. These will be summarized by a very fast grading guide, in case Jack/Jipeng don't want to read the entire document. Ideally, these systems and solvers show the intended implementation goals of my project.

Grading Stuff:

- File Naming, Report, etc: Hopefully yes
- Camera Navigation: camera functionality outlined in Instructions
- Interactive rates: on my computer, well over 3fps
- Constraints: There is a ground grid and cubes that enclose each PartSys, but sadly, there are no further ones. Could not figure out the math in time.
- System1: many particles orbit around a central, giant particle. The rest of the particles are all attached to not earth gravity, but the giant particle's and each other's. Further, makes the position-based vector-forcefield much clearer. Masses have been set to roughly translate into the Sun and many, many asteroids.
- System2: boids that demonstrate all 4 properties of boids.
- System 3: purple flame whose particle lifecycles continuously loops and goes from bright white to black as time continues. It's less noticeable, but the velocities and trajectories change slightly too.
- System 4: Rope of 15 particles that are blowing in the wind. The wind varies over time, like real wind, so it doesn't reach steady state.
- Stable and Unstable Solvers: both shown, along with semi-implicit ones, that can all be cycled through by virtue of the F key.
- OPTIONALS
  - Solvers Added (beyond the given two solvers):
    - Adams-Bashforth
    - Explicit: Midpoint
    - Velocity Verlet
    - Iterative Implicit: Euler
    - Iterative Implicit: Midpoint
    - Leapfrog
  - No additional constraint types
  - Additional Force-makers: Wind, Planetary Gravity
  - No novel rendering techniques

Code Guide:

Within the Alex_PartSys folder, the following .js files can be found. A brief description for each is attached.

ChenAlex_ProjA: The main file that sets up the particle systems, takes care of initiating particle systems, draws particle systems, and handles things for the HTML file.

CForcer: file that handles code regarding adding force to particles in particle systems.

CLimit02: file used to determine the physical response to hitting a constraint within the program.

CubeAndGridVBO: VBO initialization and drawing for the ground grid and the borders of each cube for the particle systems. Much of the inner workings are remnants of ProjectC in CS351-1, though most of them are either removed or repurposed.

PartSys06: main file for particle system state handling, utilizing a current s1 state, a previous s0 state, a future s2 state, a middle sM state, and a derivative s1dot state all used to assist various solvers calculate the proper positions to render particles. The entire file is documented with thought processes and explanations of what is happening at a given block of code.

Results

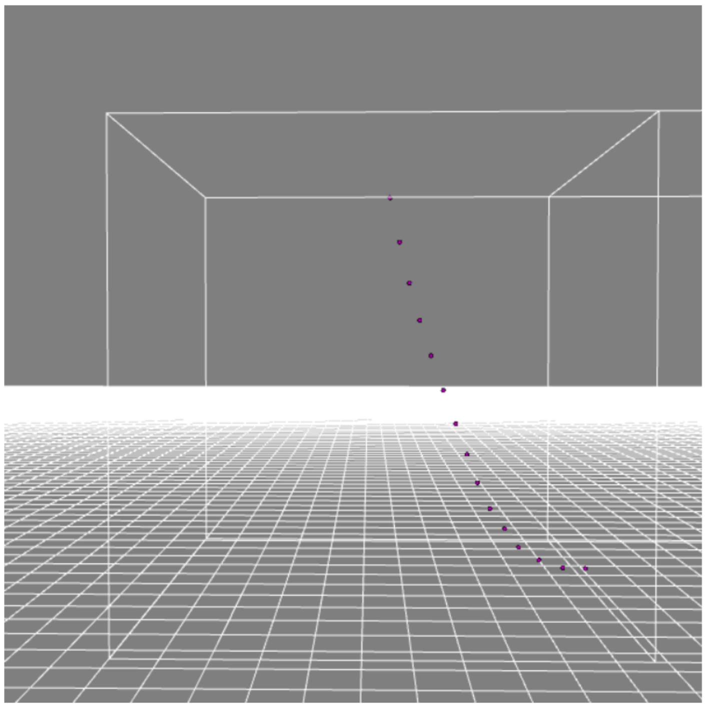


Figure 2: Rope at steadier state, in the wind

The rope, under a varying force at each time, constantly blows in the wind in order to not go to rapid equilibrium.

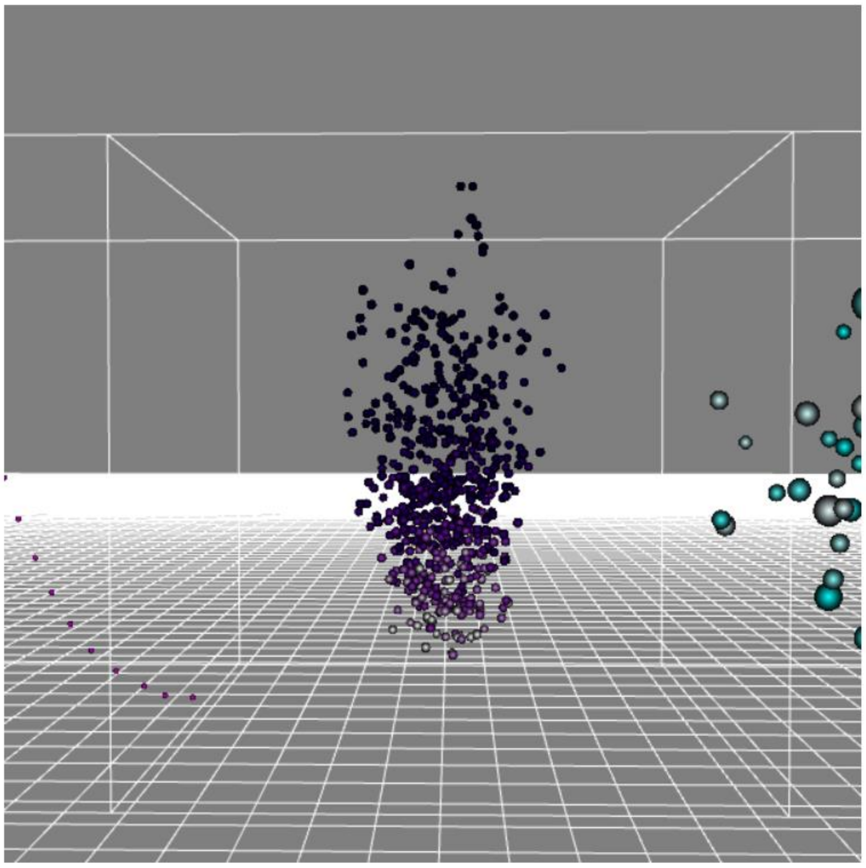


Figure 3: Fire implementation, but purple because go cats
Particles initialize white, but then as they continue on, they go from light purple to dark purple to black. Once they reach the end of their life, they reset at the beginning.

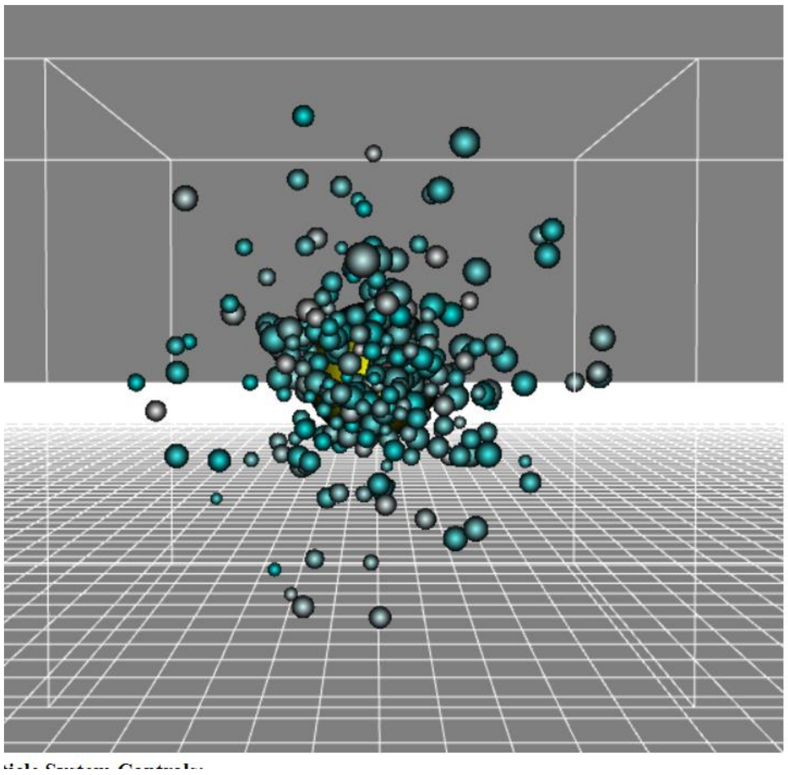


Figure 4: particles orbiting a really, really big particle

The particles themselves all generate with random mass, which affects their trajectory. In order to keep stability, particles that get too close to the center mass use a very small distance for calculations instead in order to not crash.

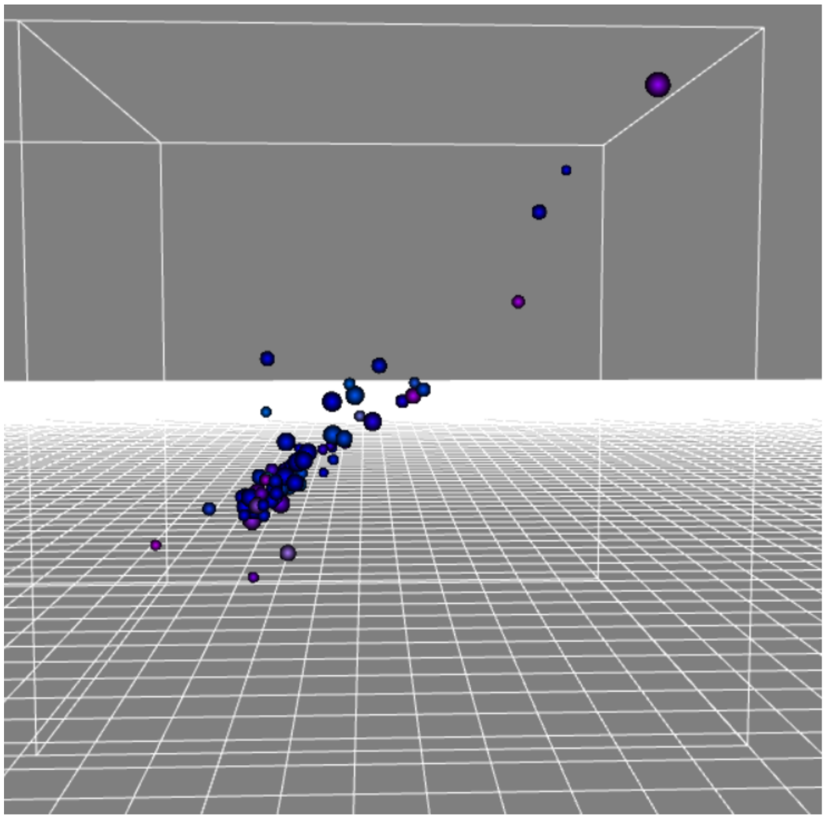
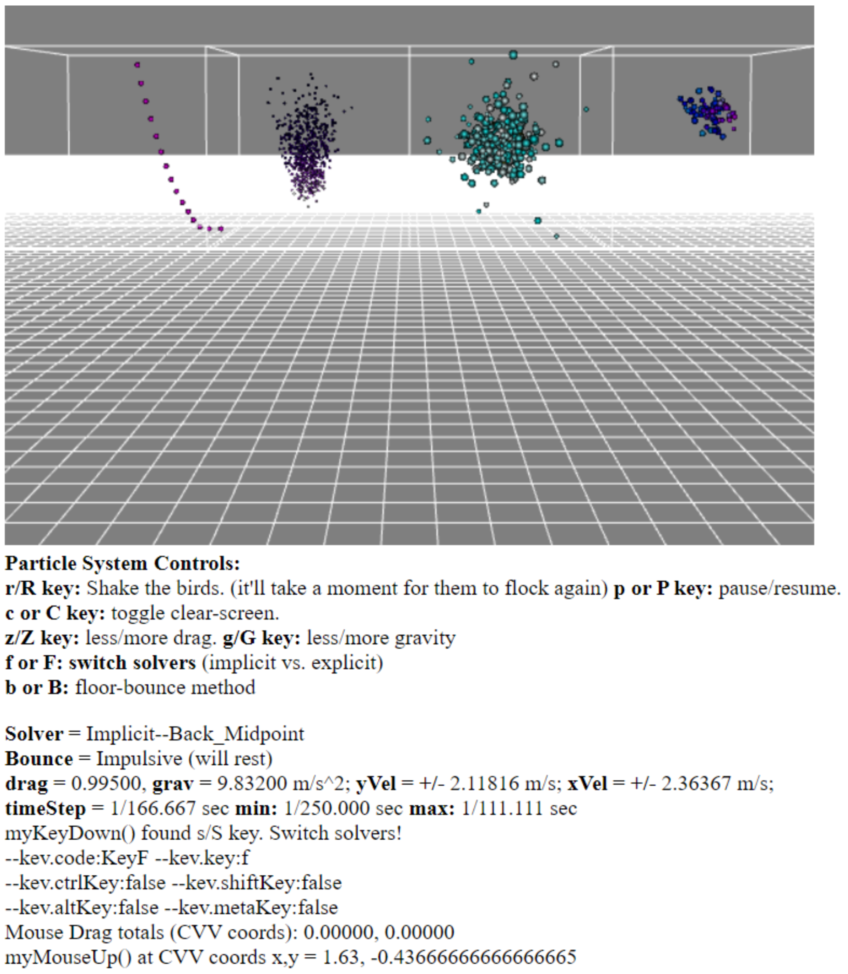


Figure 5: birds clustering

As the birds move away from each other, they feel a pull towards a weighted average position, and have a heading pointing towards the weighted average velocity, where force can then be added. There are several stragglers in the figure, but they are likely to return back once the flock heads their way once more.



Particle System Controls:
r/R key: Shake the birds. (it'll take a moment for them to flock again) **p or P key:** pause/resume.
c or C key: toggle clear-screen.
z/Z key: less/more drag. **g/G key:** less/more gravity
f or F: switch solvers (implicit vs. explicit)
b or B: floor-bounce method

Solver = Implicit--Back_Midpoint
Bounce = Impulsive (will rest)
drag = 0.99500, **grav** = 9.83200 m/s^2; **yVel** = +/- 2.11816 m/s; **xVel** = +/- 2.36367 m/s;
timeStep = 1/166.667 sec **min:** 1/250.000 sec **max:** 1/111.111 sec
myKeyDown() found s/S key. Switch solvers!
--kev.code:KeyF --kev.key:f
--kev.ctrlKey:false --kev.shiftKey:false
--kev.altKey:false --kev.metaKey:false
Mouse Drag totals (CVV coords): 0.00000, 0.00000
myMouseUp() at CVV coords x,y = 1.63, -0.43666666666666665

Figure 6: Solver changing

Though it is much clearer interacting with it, the Solvers can be iterated through by pressing the F key. Where in the first figure, we were on the old implicit solver, now we are on Back_Midpoint.

As a final note, credit goes to Seong Kim, my weekly assignments partner, since we discussed and collaborated on all 4 weekly assignments to this point, and on our ideas for the particle systems.