Andrew Barba

Assignment 7

November 11, 2015

Particle System

Particle systems are an important concept in computer animation, even at their simplest form. A simple particle system is the starting point for complex animations like explosions, smoke or gun fire. At their simplest form, they describe how a point particle is affected by mass, gravity, and the other particles around it. Rigid body particle systems take into account the surface area of each particle in order to simulate realistic collisions as multiple particles come in contact with each other and other bodies in the system. For this assignment, I wanted to create a simple rigid body particle system that could emit particles from the mouse pointer as it was moved around on screen. Almost like a simulated firecracker, the particles would emit from the cursor and then follow the laws of gravity and fall to the surface, at which point they would be removed from the system.

When researching what program I could create the system in, I ultimately decided to choose between two: Racket and Processing. Racket provided a simple functional approach for creating a particle system with powerful tools for getting started quickly. Processing provided an object oriented approach and ultimately had a better set of API's for animating and drawing on screen. For this reason I chose Processing. Processing also had a great set of tutorials and documentation on their official website that really helped me understand some of the built in classes they have for dealing with physics. Simple classes, like PVector, made things like velocity and acceleration much easier to deal with than trying to create my own classes.

To start, I immediately opened up my first assignment, twerking robot, to refresh my memory on the draw and setup functions as well as the basic drawing functions for squares and circles (ellipses). I was aiming for a pretty basic system and began with two classes: a

ParticleSystem class and a Particle class. The system class I knew would just be an array of particles with at least on function for looping through the array and placing the particles on screen. To keep complexity low, I moved most of the math for particle movement into the particle class. The reasoning for this was, if one particle knows how to move itself, then all particles in the system will know how to move themself. I'm not convinced this is a good approach for complex animations, but it was a decent starting point for this assignment. Outside factors like wind, and collisions could easily poke a few holes into this approach where extra complexity is needed. At this point I stubbed out functions like draw in both the particle and system classes, and also added basic properties like color and x position and y position. Aside from that, I knew it was time to consult some documentation to see if there were higher level classes that could be of use. As I started looking through the examples I came across the following: https://processing.org/examples/simpleparticlesystem.html I reference this link in my code because it really helped piece together someof the math behind particle movement. Again, the PVector class was a standout here and I immediately took out my x, y vars and substituted for a vector. And because the class has the convenient .add function, modeling velocity and acceleration with the same class became dead obvious. This made the actual move function of the particle straightforward.

After modeling the particle and the system i added a basic check in the top-level draw function that would begin adding particles to my system as the mouse moved. From here I could actually watch particles be emitted on screen. Now it was time to add a polygon to the mix and also give it some angular velocity. This was a good time to go back to my first assignment and reference some of the rotation code I had for "twerking" my robot. The rotation was pretty straight forward after using pushMatrix and popMatrix and translating to the location of the particle. Finally, I added very rudimentary collision detection by adding another loop within my particle loop to see if any particles had similar x and y positions within their size. If they did, I would change the particles to a different color and swap their horizontal velocities so they would move in the opposite direction.

Overall my final system was pretty basic, and after researching further I found I had a very similar solution to some other examples online. Modeling the system with 2 classes seemed to be the de facto approach for many tutorials which lends me to believe I was on a starting track to something more complex. With that said, I wish I had more time to properly model collisions with more realistic behavior.