CS 3451 Project 4 writeup

Jim Wu, Raymond Garrison

1. The velocity field we ended up using was a combination of the tangent at the curve (previous point and next point) and a very small vector towards C[s], the actual point on the curve. The idea being that a particle will slowly come back towards the line of the curve, with the pull being stronger the farther away a particle is. The gravitating element is fairly weak to prevent adding too many collisions.

2. Random surface sampling was done by generating random floats between -1,1 for x, y, z to create a random direction, normalizing it to ||v|| = 1, then scalar multiplying it by R to get a displacement vector from the center to the sphere shell.

3. Collision detection loops between calculating next collision (time s), moving all particles forward by s, then adding s to another variable called T which is reset when T reaches 1. s is capped at 1-T to ensure that steps are taken 1.0 at a time. However due to some difficulty we had with correctly updating and propagating the bounced velocities, our program sometimes freezes when too many particles bunch up together and hit the obstacle at once. They also bounce off quite forcefully.

4. Gravitation is implemented by scaling a simple V(P,obstacle.center) vector by 1/d(P, obstacle.center). This points in the direction of the Earth and falls off quickly with distance.

Controls:

c: select and drag nearest point in viewer’s XY plane. C: drags in the Z axis.

I: to insert point after nearest point, big I to append a new point.

S: save the control curve L: load the control curve

Q: subdivides the curve towards 200

X: deletes the whole curve

g: moves the particle generator in XY plane, G moves it along Z axis.

o: moves the obstacle in XY plane, O moves it along Z axis.

D for rotate/zoom

A for animate