CPSC 587 — Fundamentals of Computer Animation
Assignment 3 Report — Winter 2021

Name: Sebastian Collard
Student ID: 30046208

The following concepts were used across all component implemented (mass spring 1, 2, 3 and 4):

Each mass spring implementation consists of a $reset()$, $step(float\ dt)$ and $createSprings(springs,\ lmin)$ function. Mass spring 3 also has an extra $CalculateCollisionForce(pi)$ function.

The properties of reset() vary per component since the state we want to reset to deal with different point configurations. For mass spring 1, we clear our points and initialize two points to the initial positions. After the points are created, we generate the spring that connects the two. For mass spring 2, we clear the points and add 15 new points that form a line and then generate the springs that connect them as a chain. For mass spring 3, we clear the points, create a 10x10x10 cube of points, rotate it slightly and then generate the springs. For mass spring 4, we clear the points, create a square 20x20 square of points, set two corners to zero mass to make them fixed, then generate the springs.

step(float dt) is the same for all mass springs, except for an added force calculation in the third one. First we loop over all of the springs, which all store a corresponding pair of integers that act as indices for the list of points. Using these indices, we access the corresponding points this spring connect together,pi and pj , and update both points' netforces as follows:

$$pi.netforce+ = F^s_{ij} + F^d_{ij}$$

$$pj.netforce+ = F^s_{ij} - F^d_{ij}$$

where

$$F^s_{ij} = -(spring.stiffness)(glm :: length(pi.pos - pj.pos) - spring.restLength)(\frac{pi.pos - pj.pos}{glm :: length(pi.pos - pj.pos)})$$

$$F^d_{ij} = -(spring.damping)\frac{glm :: dot((pi.velo - pj.velo), glm :: normalize(F^s_{ij}))}{glm :: dot(glm :: normalize(F^s_{ij}), glm :: normalize(F^s_{ij}))} glm :: normalize(F^s_{ij})$$

After this, we loop over all points pi and accumulate other forces:

$$F^{gravity} = pi.mass * vec3f\{0.f, -g, 0.f\}$$

$$pi.netforce+ = F^{gravity}$$

The force of gravity alone suffices for mass springs 1, 2 and 4. For mass spring 3, we also need to accumulate the collision force by calculating the spring force (with zero rest length) between points underneath an arbitrary plane and a point directly above them at the height of the plane:

$$F^{coll}_i = CalculateCollisionForce(pi)$$

Where

```
CalculateCollisionForce(pi)
    if pi.pos.y >= (plane.y) then
        return vec3f{0.f}

    float stiff = ...
    float damp = ...

    Point pj = Point(0.f, vec3f{ pi.pos.x, 0.f, pi.pos.z })

    $F^s_{ij}$=-(stiff)(glm::length(pi.pos - pj.pos))($\frac{pi.pos-pj.pos}{glm::length(pi.pos-pj.pos)}$)
    d = glm::normalize(pi.pos - pj.pos)

    $F^d_{ij}$=-damp($\frac{glm::dot(pi.velo-pj.velo),d}{glm::dot(d,d)}$)d

    return $F^s_{ij} + F^d_{ij}$
```

Lastly, we loop again over all points *pi* and update the velocity and position if the given points has a non-zero mass. If it does have zero mass, then we treat the point as being fixed in space. The last step in the loop resets the net force of the points to zero regardless of its mass:

$$pi.mass > 0 \longrightarrow pi.velo+ = \frac{pi.netforce}{pi.mass}dt$$

$$pi.mass > 0 \longrightarrow pi.pos+ = (pi.velo)dt$$

$$pi.netforce = vec3f\{0.f\}$$

createSprings(springs, l_{min}) follows the same procedure as the pseudocode provided and is used for all 4 mass springs to generate the desired spring network for the given set of points.

Mass Spring 1 Parameters: dt: 0.01; k_s: 1; k_d: 0.2; substeps: 7
Mass Spring 2 Parameters: dt: 0.01; k_s: 200; k_d: 3; substeps: 7
Mass Spring 3 Parameters: dt: 0.01; k_s: 750; k_d: 10; substeps: 11
Mass Spring 4 Parameters: dt: 0.01; k_s: 600; k_d: 3; substeps: 7