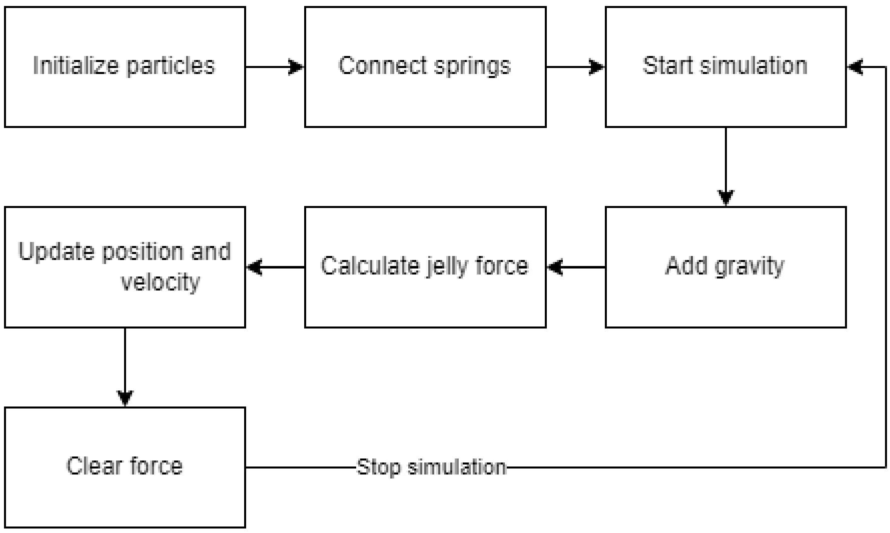
Computer animation hw1 report

1. Fundamentals



2. Implementation

a、Springs:

i. **Struct and bending**: connect to its neighbor and the particle behind its neighbor in x, y, z-direction respectively.

```
// struct
// z-direction
for (int i = 0; i < particleNumPerEdge; i++){ ... }

// y-direction
for (int i = 0; i < particleNumPerEdge; i++){ ... }

// x-direction
for (int i = 0; i < particleNumPerEdge - 1; i++){ ... }

// bending
// z-direction
for (int i = 0; i < particleNumPerEdge; i++){ ... }

// y-direction
for (int i = 0; i < particleNumPerEdge; i++){ ... }

// x-direction
for (int i = 0; i < particleNumPerEdge - 2; i++){ ... }
```

ii. **Shear**: connect to 10 particals in its diagnal.

```
// shear
// +y, +z
for (int i = 0; i < particleNumPerEdge; i++){ ... }

// +x, +y, +z
for (int i = 0; i < particleNumPerEdge - 1; i++){ ... }

// +x, +z
for (int i = 0; i < particleNumPerEdge - 1; i++){ ... }

// +x, -y, +z
for (int i = 0; i < particleNumPerEdge - 1; i++){ ... }

// -y, +z
for (int i = 0; i < particleNumPerEdge; i++){ ... }

// -x, -y, +z
for (int i = 1; i < particleNumPerEdge; i++){ ... }

// -x, +z
for (int i = 1; i < particleNumPerEdge; i++){ ... }

// -x, +y, +z
for (int i = 1; i < particleNumPerEdge; i++){ ... }

// +x, +y
for (int i = 0; i < particleNumPerEdge - 1; i++){ ... }

// +x, -y
for (int i = 0; i < particleNumPerEdge - 1; i++){ ... }
```

b、 Collision handle

i. **Plane terrain**: update the velocity to $\boldsymbol{v}'$ if the particle satisfies the following equations $f(\boldsymbol{x}) \wedge f_{plane}(\boldsymbol{x}) = \text{true}$ ($|\boldsymbol{x}|_{\text{x-z}}$ indicates the norm of $\boldsymbol{x}$ on x-z plane).

$$f(\boldsymbol{x}) \coloneqq [\boldsymbol{N} \cdot (\boldsymbol{x} - \boldsymbol{p}) < \varepsilon] \wedge [\boldsymbol{N} \cdot \boldsymbol{v} < 0]$$

$$f_{plane}(\boldsymbol{x}) \coloneqq [|\boldsymbol{x} - \boldsymbol{p}_{hole}|_{\text{x-z}} > r_{hole}]$$

$$\boldsymbol{v}_{\text{N}} = \boldsymbol{N} \cdot \boldsymbol{v} * \boldsymbol{N}, \boldsymbol{v}_{\text{T}} = \boldsymbol{v} - \boldsymbol{v}_{\text{N}}$$

$$\boldsymbol{v}' = -\text{k}_r \boldsymbol{v}_{\text{N}} + \boldsymbol{v}_{\text{T}}$$

(f_{plane} indicates whether the particle is outside the hole)

ii. **Bowl terrain**: likely to plane terrain, update the velocity when $f(\boldsymbol{x}) \wedge f_{bowl}(\boldsymbol{x}) = \text{true}$, but the normal of the collision plane is $\boldsymbol{N}_{bowl} = \frac{\boldsymbol{p}-\boldsymbol{x}}{|\boldsymbol{p}-\boldsymbol{x}|}$.

$$f_{bowl}(\boldsymbol{x}) := \left[\left||\boldsymbol{x}-\boldsymbol{p}| - r\right| \leq \varepsilon\right] \wedge \left[\boldsymbol{x}_y \leq \varepsilon\right]$$

(f_{bowl} indicates whether the particle is contacting the bottom of the bowl).

c、 Contact force

i. **Plane terrain**: add force $\boldsymbol{f}'$ to a particle when it satisfies the equations $g(\boldsymbol{x}) \wedge f_{plane}(\boldsymbol{x}) = \text{true}$.

$$g(\boldsymbol{x}) \coloneqq [\boldsymbol{N} \cdot (\boldsymbol{x}-\boldsymbol{p}) < \varepsilon] \wedge [\boldsymbol{N} \cdot \boldsymbol{v} < \varepsilon] \wedge [\boldsymbol{N} \cdot \boldsymbol{f} < 0]$$

$$f_c = -(\boldsymbol{N} \cdot \boldsymbol{f})\boldsymbol{N}, f_f = -\mathrm{k}_f(-\boldsymbol{N} \cdot \boldsymbol{f})\boldsymbol{v}_{\mathrm{T}}, \boldsymbol{f}' = f_c + f_f$$

ii. **Bowl terrain**: add force $\boldsymbol{f}'$ when it satisfies the equations $g(\boldsymbol{x}) \wedge f_{bowl}(\boldsymbol{x}) = \text{true}$.

d、 Integrator

i. **Explicit Euler**: direct update

$$\boldsymbol{x} \leftarrow \boldsymbol{x} + h\boldsymbol{v}, \boldsymbol{v} \leftarrow \boldsymbol{v} + h\boldsymbol{a}$$

ii. **Implicit Euler**: clear every force and recalculate all force, then update by Explicit Euler.

iii. **Midpoint Euler**: first update

$$\boldsymbol{x}' \leftarrow \boldsymbol{x} + \frac{h}{2}\boldsymbol{v}, \boldsymbol{v}' \leftarrow \boldsymbol{v} + \frac{h}{2}\boldsymbol{a}$$

then calculate the force according to $\boldsymbol{x}'$ and $\boldsymbol{v}'$, get $\boldsymbol{v}_{t+\frac{h}{2}}$ and $\boldsymbol{a}_{t+\frac{h}{2}}$, and update the particle by $\boldsymbol{v}_{t+\frac{h}{2}}$ and $\boldsymbol{a}_{t+\frac{h}{2}}$.

$$\boldsymbol{x} \leftarrow \boldsymbol{x} + h\boldsymbol{v}_{t+\frac{h}{2}}, \boldsymbol{v} \leftarrow \boldsymbol{v} + h\boldsymbol{a}_{t+\frac{h}{2}}$$

iv. **Runge-Kutta Fourth**:

$$k_1 = (\boldsymbol{v}, \boldsymbol{a})$$

Calculate $(\boldsymbol{v}', \boldsymbol{a}')$ according to $(\boldsymbol{x}, \boldsymbol{v}) + \frac{h}{2}k_1$

$$k_2 = (\boldsymbol{v}', \boldsymbol{a}')$$

Calculate $(\boldsymbol{v}'', \boldsymbol{a}'')$ according to $(\boldsymbol{x}, \boldsymbol{v}) + \frac{h}{2}k_2$

$$k_3 = (\boldsymbol{v}'', \boldsymbol{a}'')$$

Calculate $(\boldsymbol{v}''', \boldsymbol{a}''')$ according to $(\boldsymbol{x}, \boldsymbol{v}) + hk_3$

$$k_4 = (\boldsymbol{v}''', \boldsymbol{a}''')$$

Finally, update

$$(\boldsymbol{x}, \boldsymbol{v}) \leftarrow (\boldsymbol{x}, \boldsymbol{v}) + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

3. Result and Discussion

a、Difference between integrators

i. Implicit Euler has the highest velocity after colliding and sliding in the bowl, while Runge-Kutta has the lowest velocity.
ii. Midpoint Euler and Runge-Kutta slow down very early, while Implicit Euler sliding in the bowl for a long time.
iii. Only Explicit Euler jelly become concave at y=30.
iv. Implicit Euler is the best against concave.

b、Effect of parameters

i. **SpringCoef**: spring coefficient can help the jelly to maintain its shape. Higher value jumps higher and concave less, while lower value causes the jelly concave a lot.
ii. **damperCoef**: damper coefficient helps the jelly to cancel the effect of outer force. Low damper coefficient causes the jelly keep shaking and become unstable, but setting the value too high would cause the jelly acts like a stone, not responding to outer force.
iii. **coefResist**: this coefficient decides how much force is given when collision. High value gives the jelly bigger contact force, letting it jumps higher.
iv. **coefFriction**: coefficient of friction decides how much friction the jelly would take on the surface. Higher value would cause the jelly stops earlier.
v. **particleCountPerEdge**: setting this value higher can let the jelly respond to the environment better, but the chance of concave also become higher, and my computer is crying.

4. Conclusion

This homework is not easy and has lots of details, but very interesting. I have taken neither Numerical Methods nor Computer Animation courses, so I found myself learning a lot by completing this homework.