HW1 - Particle System

2022 Computer Animation and Special Effects

Outline

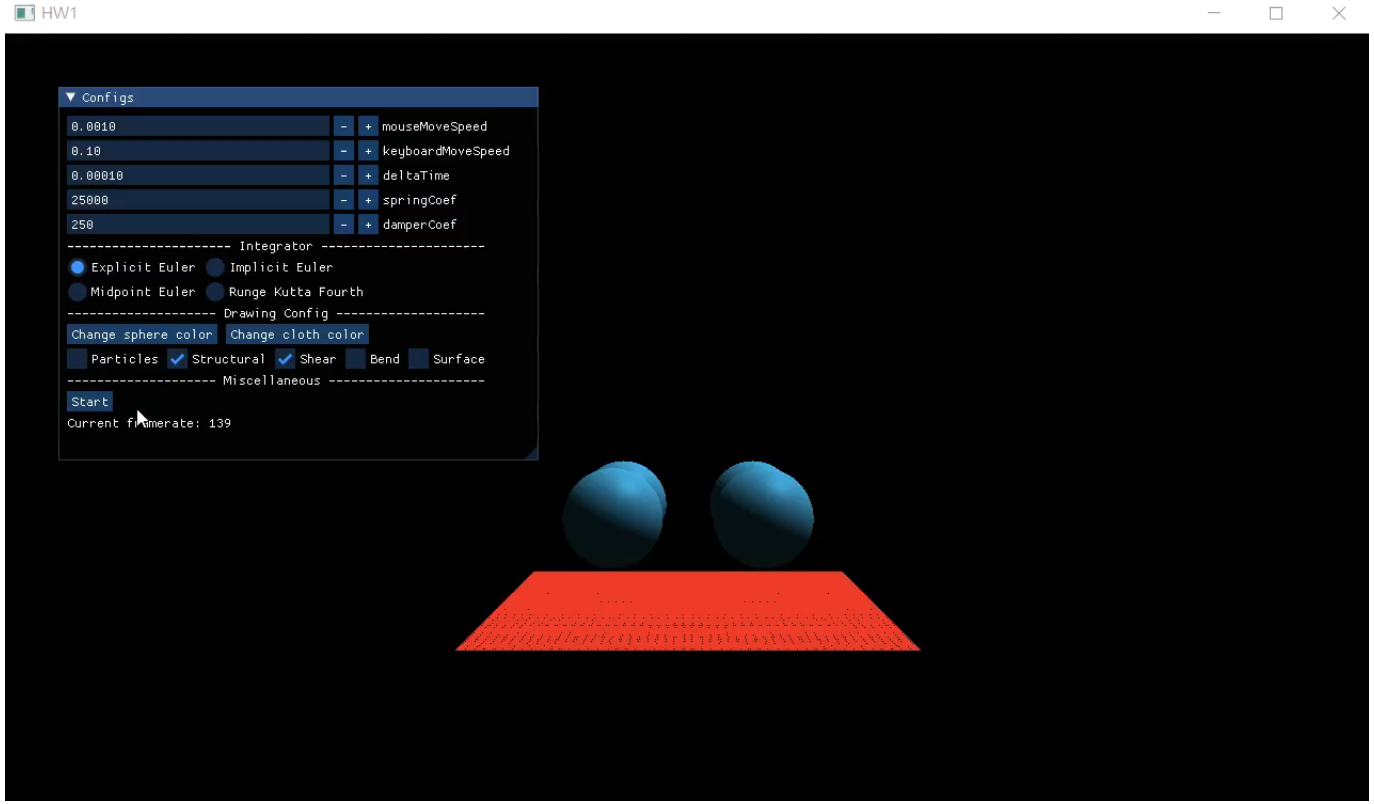
- Overview
- Environment Setup
- Objective
- Report
- Scoring
- Submission

Overview

- IDE: Visual studio 2019 / Visual studio 2022
- Graphics API: OpenGL
- Dependencies
  - Eigen
  - glfw
  - glad
  - Dear ImGui

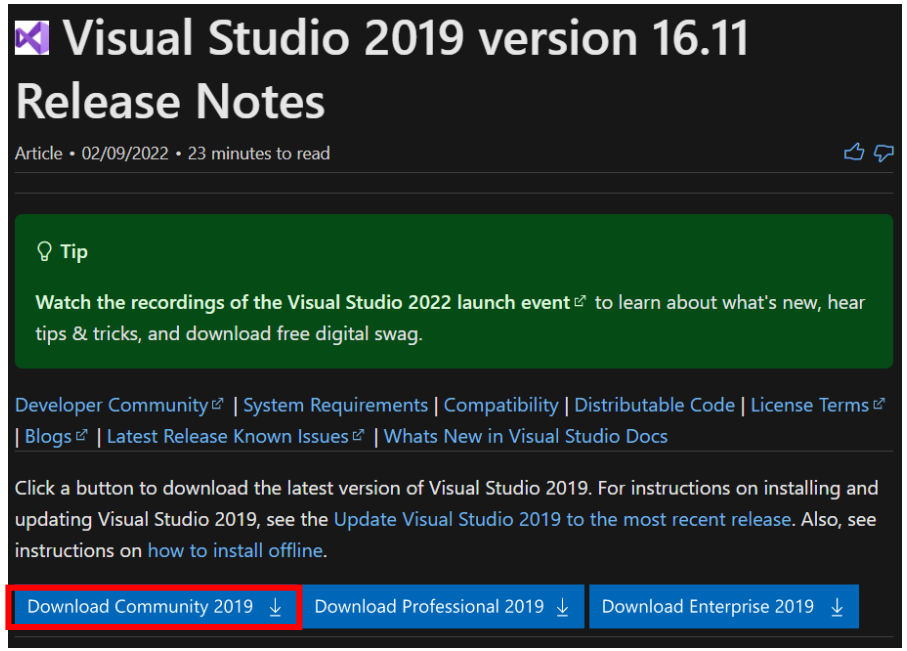
Overview (cont.)

- https://youtu.be/BgJ4UWFfxFw



Environment Setup

- Download Visual Studio 2019 – Community or Visual Studio 2022 - Community



Environment Setup (cont.)

- Launch Visual Studio Installer

Environment Setup (cont.)

- Download HW1.zip and unzip
- Open HW1.sln

Environment Setup (cont.)

- Run the project

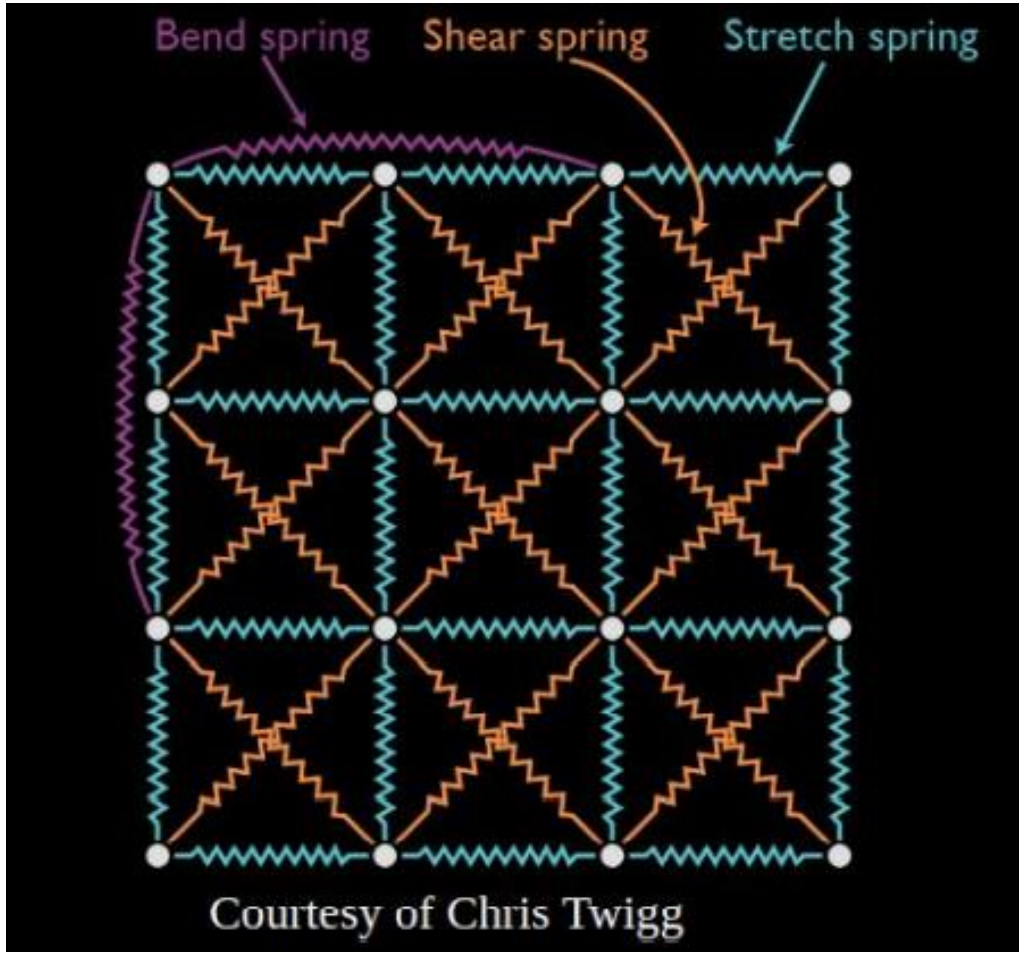
- Select config then build (CTRL+SHIFT+B)
- Use F5 to debug or CTRL+F5 to run
  - It will spend a lot of time to debug so release is recommended unless you need debugger

Objective

- src
  - cloth.cpp
    - void Cloth::initializeSpring()
    - void Cloth::computeSpringForce()
  - sphere.cpp
    - void Spheres::collide(Cloth* cloth)
    - void Spheres::collide()
  - integrator.cpp
    - void ExplicitEuler::integrate(…)
    - void ImplicitEuler::integrate(…)
    - void MidpointEuler::integrate(…)
    - void RungeKuttaFourth::integrate(…)

Objective (cont.)

- void Cloth::initializeSpring()
  - Goal
    - Construct the connection of springs
  - Hint
    - You should initialize three types of spring
      - struct, shear and bending



Objective (cont.)

- Take a 3x3 cloth(9 particles) for example and observe the center particle
  - Struct / bending: 2 directions
    - 4 directions: up, down, left and right. But if each particle is responsible for all 4 directions, there will have duplicate connection. Thus, each particle will only be responsible for 2 directions.
  - Shear: 2 directions
    - Center particle is surrounded by 8 particles. 8 - 4 (up, down, left and right) = 4, and each particle can be only responsible for half part of directions.

Objective (cont.)

- Put all springs in std::vector<Spring> _springs, which is a class member in class Cloth
- You can also check class Spring in spring.h

```
Spring(unsigned int start, unsigned int end, float restLength, Type type) :
    _startParticleIndex(start), _endParticleIndex(end), _length(restLength), _springType(type) {}
```

- The parameter particlesPerEdge, which is defined in config.h, might be used
- Here is a sample code of connecting one of struct springs

```
float structrualLength = (_particles.position(0) - _particles.position(1)).norm();
_springs.emplace_back(0, 1, structrualLength, Spring::Type::STRUCTURAL);
```

Objective (cont.)

- void Cloth::computeSpringForce()
  - Goal
    - Compute spring and damper forces
  - Hint
    - Review “particles.pptx” from p.9 - p.13
    - Trace every spring and apply the force accordingly
    - Set acceleration to apply the force
    - You can use float Particles::inverseMass(int i) to get the inverse of mass
    - The parameter springCoef and damperCoef are defined in config.h

Objective (cont.)

- void Spheres::collide(Cloth* cloth) / void Spheres::collide()
  - Goal
    - Handle collision between spheres and cloth / spheres and spheres
  - Hint
    - Review “particles.pptx” from p.14 - p.19
    - You can use their radius and distance to determine whether they are collided
    - The radius of particles of cloth can be regarded as 0

Objective (cont.)

- To compute the velocity after collision
  - You only need to worry about the component of the velocity that is in the direction of the collision
  - The other component of the velocity (tangent to the collision) will stay the same for both particles
  - You can refer to this website for detailed information

$$v_a = \frac{m_\mathrm{a}u_\mathrm{a} + m_\mathrm{b}u_\mathrm{b} + m_\mathrm{b}C_R(u_\mathrm{b} - u_\mathrm{a})}{m_\mathrm{a} + m_\mathrm{b}}$$

and

$$v_\mathrm{b} = \frac{m_\mathrm{a}u_\mathrm{a} + m_\mathrm{b}u_\mathrm{b} + m_\mathrm{a}C_R(u_\mathrm{a} - u_\mathrm{b})}{m_\mathrm{a} + m_\mathrm{b}}$$

C_R is the coefficient of restitution

Objective (cont.)

- Integrator
  - Hint
    - You should update particles' velocity and position.
  - void ExplicitEuler::integrate(…)
    - Hint: Review “ODE_basics.pptx” from p.15 - p.16
  - void ImplicitEuler::integrate(…)
    - Hint
      - Review "ODE_implicit.pptx" from p.18 - p.19
      - You probably need to call simulateOneStep() for getting future information
  - void MidpointEuler::integrate(…)
    - Hint: Review “ODE_basics.pptx” from p.18 - p.20 and “pbm.pdf” from B.5 - B.6
  - void RungeKuttaFourth::integrate(…)
    - Hint: Review “ODE_basics.pptx” from p.21 and “pbm.pdf” from B.5 - B.6

Objective (cont.)

- Bonus
  - Add friction force in void Spheres::collide function
  - Add rolling the sphere in void Spheres::collide function
  - Remind: You should mention it in your report

Report

- Suggested outline
  - Introduction/Motivation
  - Fundamentals
  - Implementation
  - Result and Discussion
    - The difference between integrators
    - Effect of parameters
  - Bonus (Optional)
  - Conclusion

Scoring

- Construct the connection of springs - 15%
- Compute spring and damper forces - 20%
- Handle Collision - 20%
  - spheres and cloth - 10%
  - spheres and spheres - 10%
- Integrator - 25%
  - Explicit Euler - 5%
  - Implicit and Midpoint Euler - 5%
  - Runge-Kutta 4th - 15%
- Report - 20%
- Bonus - 15%
  - Add friction force - 5%
  - Add rolling the sphere - 10%

Submission

- Please upload hw1_<your student ID>.zip and report_< your student ID >.pdf respectively
- hw1_<your student ID>.zip (root)
  - src
  - include
- Late policies
  - Penalty of 10 points on each day after deadline
- Cheating policies
  - 0 points for any cheating on assignments
- Deadline
  - Monday, 2022/03/21, 23:59