Particle System

HW1
2023 Computer Animation and Special Effects

Outline

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

Overview (cont.)

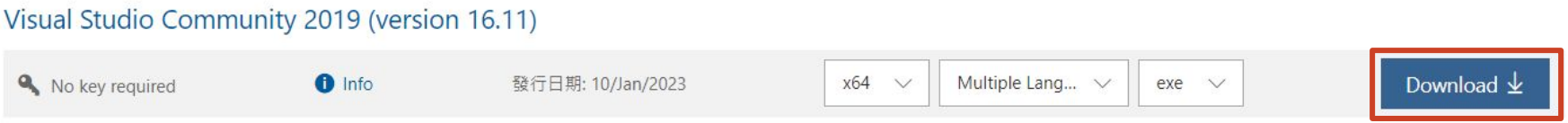
- Demo link

Overview (cont.)

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

Environment Setup

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



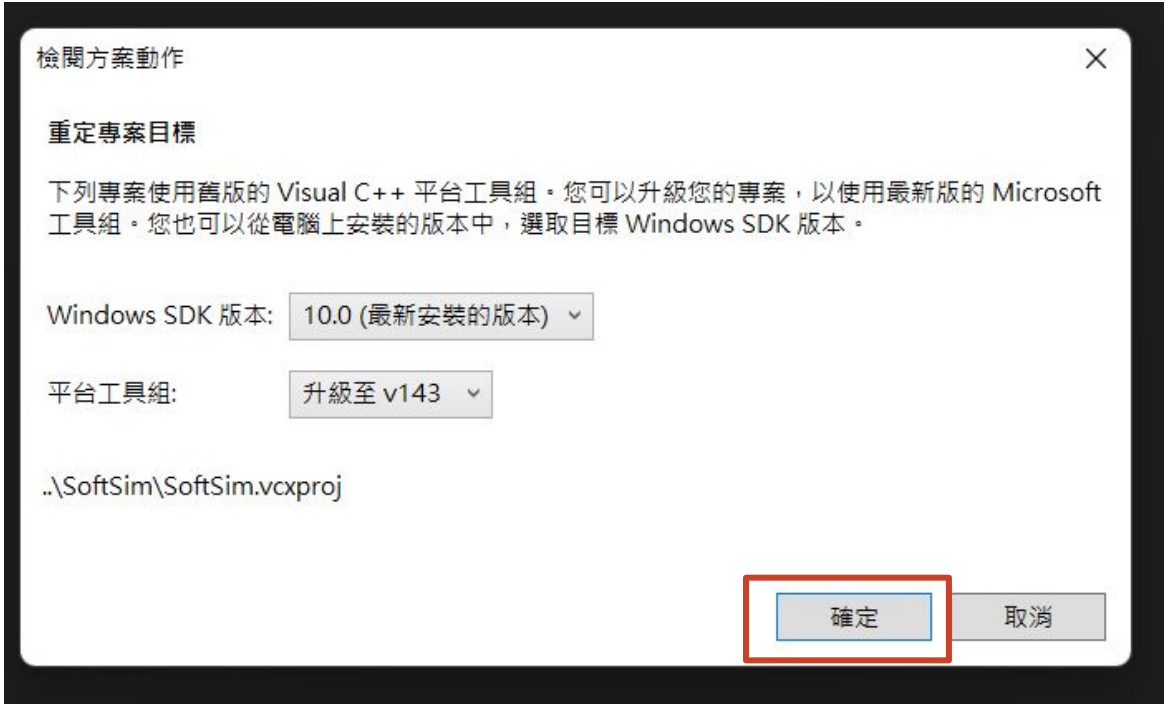
Environment Setup (cont.)

- Launch Visual Studio Installer

Environment Setup (cont.)

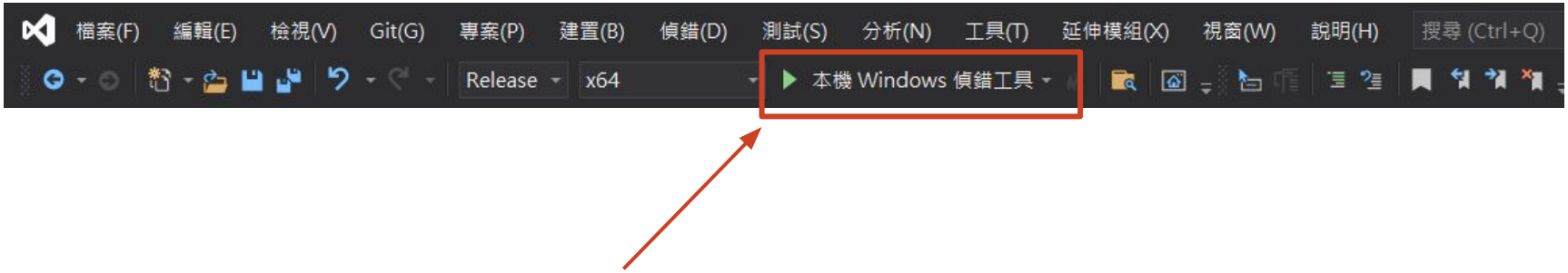
- Download HW1.zip and unzip
- Open SoftSim.sln

Environment Setup (cont.)



Environment Setup (cont.)

- Run the project



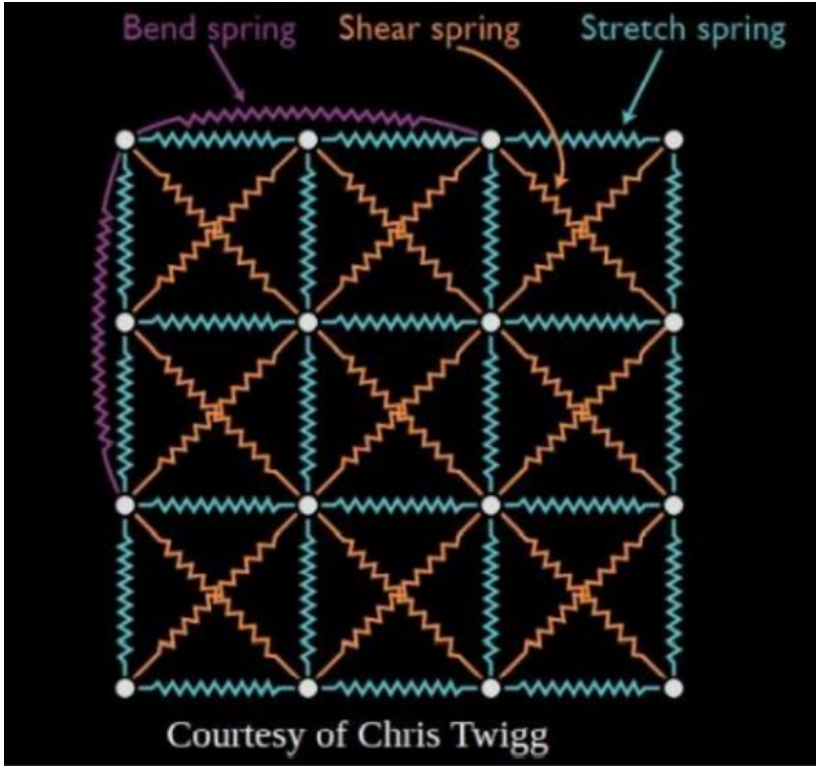
- Select config then build (CTRL+SHIFT+B)
- Use F5 to debug or CTRL+F5 to run
  - It will spend a lot of time to debug so release is recommended unless you need debugger

Objective

- src
  - main.cpp
    - change your studentID
  - jelly.cpp
    - void Jelly::initializeSpring()
    - Eigen::Vector3f Jelly::computeSpringForce(...)
    - Eigen::Vector3f Jelly::computeDamperForce(...)
    - void Jelly::computeInternalForce()
  - terrain.cpp
    - void PlaneTerrain::handleCollision(...)
    - void BowlTerrain::handleCollision(...)
  - integrator.cpp
    - void ExplicitEulerIntegrator::integrate(...)
    - void ImplicitEulerIntegrator::integrate(...)
    - void MidpointEulerIntegrator::integrate(...)
    - void RungeKuttaFourthIntegrator::integrate(...)

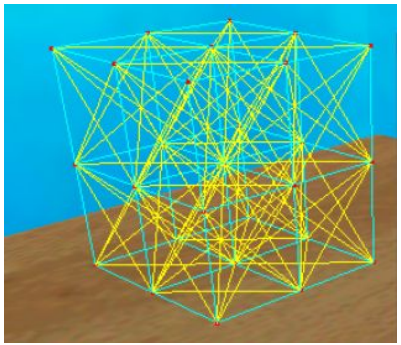
Objective (cont.)

- void Jelly::initializeSpring()
  - Construct the connection of springs
    - three types of spring
      - struct, shear and bending



Objective (cont.)

- void Jelly::initializeSpring()
  - Take a 3x3x3 jelly (27 particles) for example and observe the center particle
    - struct / bending: 3 directions
      - Total 6 directions: up, down, left, right, front and back. But if each particle is responsible for all 6 directions, there will have duplicate connection. Thus, each particle will be responsible only for 3 directions.
    - shear: 10 directions
      - Center particle is surrounded by 26 particles. 26 - 6 (up, down, left, right, front and back) = 20, and each particle will be responsible only for half part of directions.



Objective (cont.)

- void Jelly::initializeSpring()
  - Put all springs in std::vector<Spring> springs, which is a class member in class Jelly
  - You can also check class Spring in spring.h
    - “springStartID” and “springEndID” are the index in the std::vector<Particle> particles, a class member in class Jelly

```
Spring(
    int springStartID,
    int springEndID,
    float restLength,
    float springCoef,
    float damperCoef,
    SpringType type
);
```

Objective (cont.)

3x3x3 shear spring (yellow), struct spring (blue)

Objective (cont.)

- void Jelly::initializeSpring()
  - Sample code of connecting struct spring along z-axis

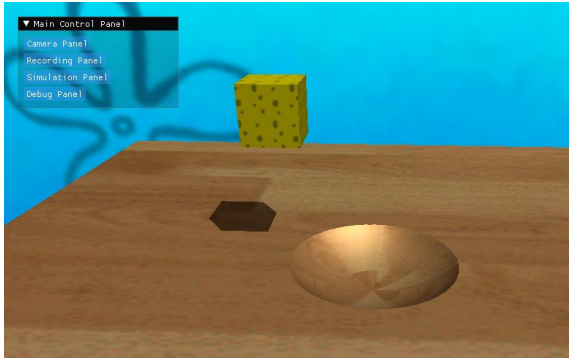
```
for (int i = 0; i < particleNumPerEdge; i++) {
    for (int j = 0; j < particleNumPerEdge; j++) {
        for (int k = 0; k < particleNumPerEdge - 1; k++) {

            int iParticleID = i * particleNumPerFace + j * particleNumPerEdge + k;
            int iNeighborID = iParticleID + 1;
            Eigen::Vector3f SpringStartPos = particles[iParticleID].getPosition();
            Eigen::Vector3f SpringEndPos = particles[iNeighborID].getPosition();
            Eigen::Vector3f Length = SpringStartPos - SpringEndPos;
            float absLength = sqrt(Length[0] * Length[0] + Length[1] * Length[1] + Length[2] * Length[2]);

            springs.push_back(Spring(iParticleID, iNeighborID, absLength, springCoefStruct, damperCoefStruct,
                                     Spring::SpringType::STRUCT));
        }
    }
}
```

Objective (cont.)

- void Jelly::computeInternalForce()
  - Trace every spring and apply the force accordingly
    - Eigen:: Jelly::computeSpringForce()
      - compute spring forces
    - Eigen:: Jelly::computeDamperForce()
      - compute damper forces
  - The values of parameter springCoef and damperCoef are defined in massSpringSystem.cpp
  - Hint: review “particles.pptx” from p.9 - p.13

Objective (cont.)

- void PlaneTerrain::handleCollision(...)/void BowlTerrain::handleCollision(...)
  - Handle collision between plane and jelly / bowl and jelly
    - constexpr float eEPSILON = 0.01;
    - constexpr float coefResist = 0.8;
    - constexpr float coefFriction = 0.2;
  - You can assume the terrain will not move under any circumstances
  - You can use their radius and distance to determine whether they are collided
    - the radius of particles of jelly can be regarded as 0
    - other related parameters can be found in class member
  - Hint: review “particles.pptx” from p.14 - p.19

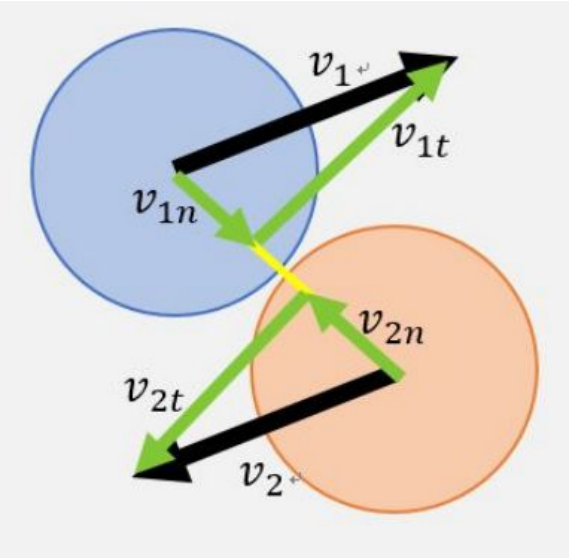


Objective (cont.)

- To compute the velocity after collision - bowl collision
  - You can assume the terrain will not move
    - that is, the velocity is 0

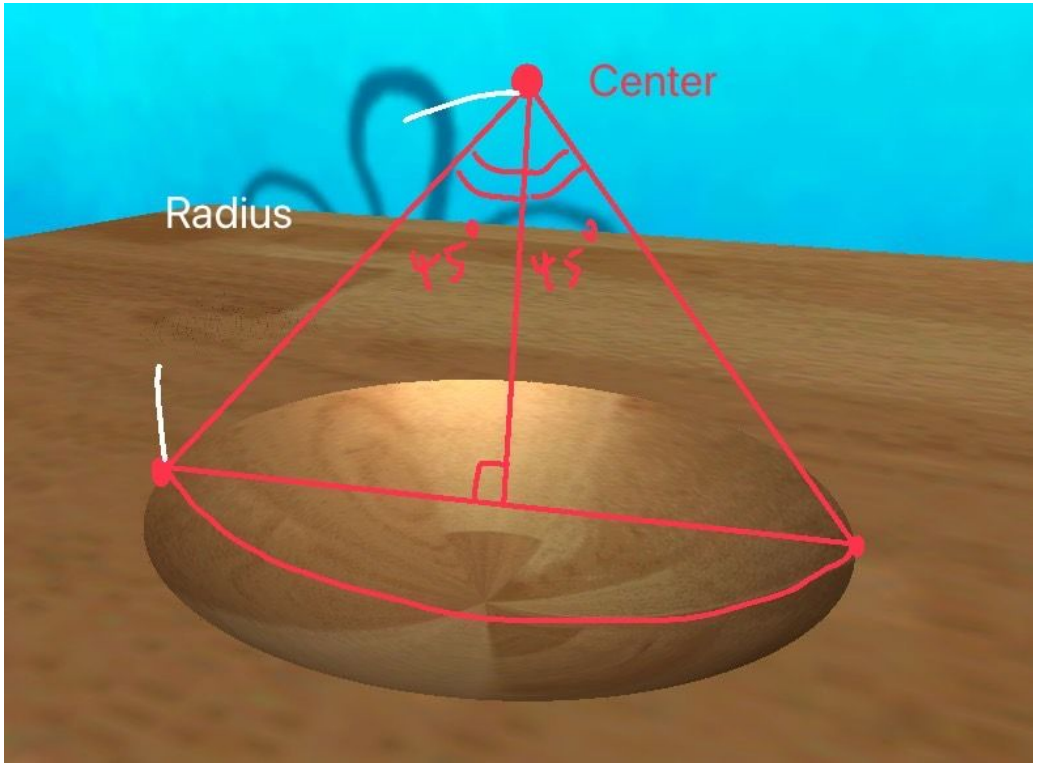
$$v_1' = \frac{v_{1n}(m_1 - m_2) + 2m_2 v_{2n}}{m_1 + m_2} + v_{1t}$$

$$v_2' = \frac{v_{2n}(m_2 - m_1) + 2m_1 v_{1n}}{m_1 + m_2} + v_{2t}$$



Objective (cont.)

- Shape of bowl



Objective (cont.)

- Integrator
  - Update particles' position and velocity
  - void ExplicitEulerIntegrator::integrate(...)
    - Hint: review “ODE_basics.pptx” from p.15 - p.16
  - void ImplicitEulerIntegrator::integrate(...)
    - Hint: review "ODE_implicit.pptx" from p.18 - p.19
  - void MidpointEulerIntegrator::integrate(...)
    - Hint: review “ODE_basics.pptx” from p.18 - p.20 and “pbm.pdf” from B.5 - B.6
  - void RungeKuttaFourthIntegrator::integrate(...)
    - Hint: review “ODE_basics.pptx” p.21 and “pbm.pdf” from B.5 - B.6

Objective (cont.)

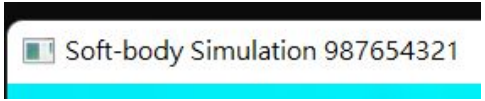
- Bonus
  - Any creativity
  - For example
    - Improve graphic
    - Change jelly's shape
    - Other type of terrain
    - …
  - Don't break original requirements (if it does, make an toggle for switching between requirement parts and bonus parts)
  - Mention it in your report

Report

- Suggested outline
  - Introduction/Motivation
  - Fundamentals
  - Implementation
  - Result and Discussion
    - The difference between integrators
    - Effect of parameters (springCoef, damperCoef, coefResist, coefFriction, etc.)
  - Bonus (Optional)
  - Conclusion

Scoring

- Change window title to “Soft-body Simulation **STUDENT_ID**” (0%)
  - -10% if title is wrong
- Construct the connection of springs - 15%
- Compute spring and damper forces - 20%
- Handle Collision - 20%
  - plane - 10%
  - bowl - 10%
- Integrator - 25%
  - Explicit Euler - 5%
  - Implicit and Midpoint Euler - 5%
  - Runge-Kutta 4th - 15%
- Report - 20%
- Bonus - up to 15%



Submission

- Please upload hw1_<your student ID>.zip and report_< your student ID>.pdf respectively
- hw1_<your student ID>.zip (root)
  - src
  - main.cpp
- Late policies
  - Penalty of 10 points on each day after deadline
- Cheating policies
  - 0 points for any cheating on assignments
- Deadline
  - Sunday, 2023/03/26, 23:59

Note

- Read TODOs in the template and follow TODOs' order

```
// TODO#1: Connect particles with springs.
//   1. Consider the type of springs and compute indices of particles which the spring connect to.
//   2. Compute rest spring length using particle positions.
//   2. Iterate the particles. Push spring objects into `springs` vector
// Note:
//   1. The particles index can be computed in a similar way as below:
//   ==========================================
//   0 1 2 3 ... particlesPerEdge
//   particlesPerEdge + 1 ....
//   ... ... particlesPerEdge * particlesPerEdge - 1
//   ==========================================
// Here is a simple example which connects the structrual springs along z-axis.
```

- How to contact TAs?
  - please ask your questions on new E3 forum
    - or send email to ALL TAs via new E3
  - if you need to ask questions face-to-face, please send an email for appointment