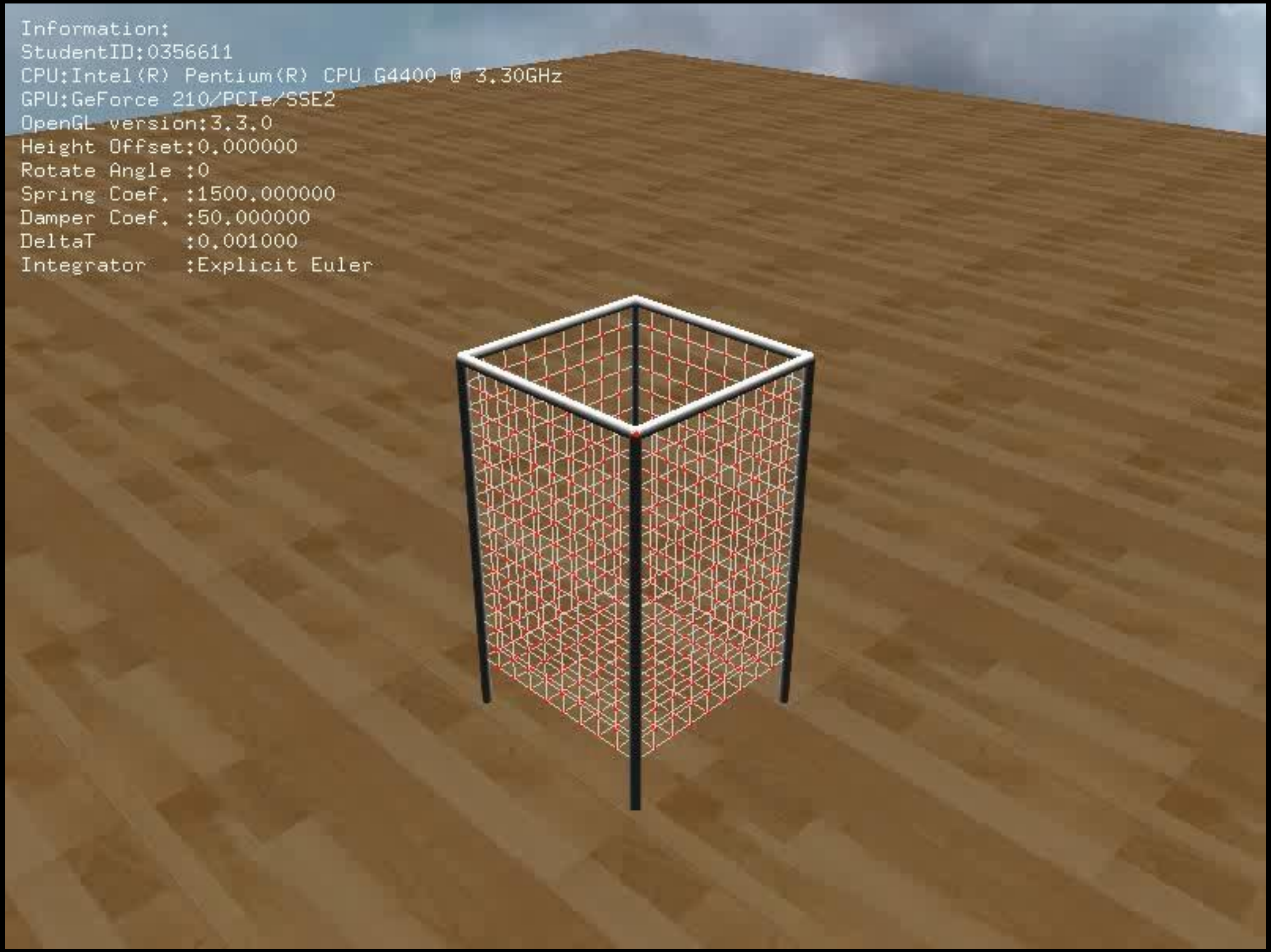
Particle System
Soft Body Simulation

TA: 劉彥廷

2018/03/29

Demo

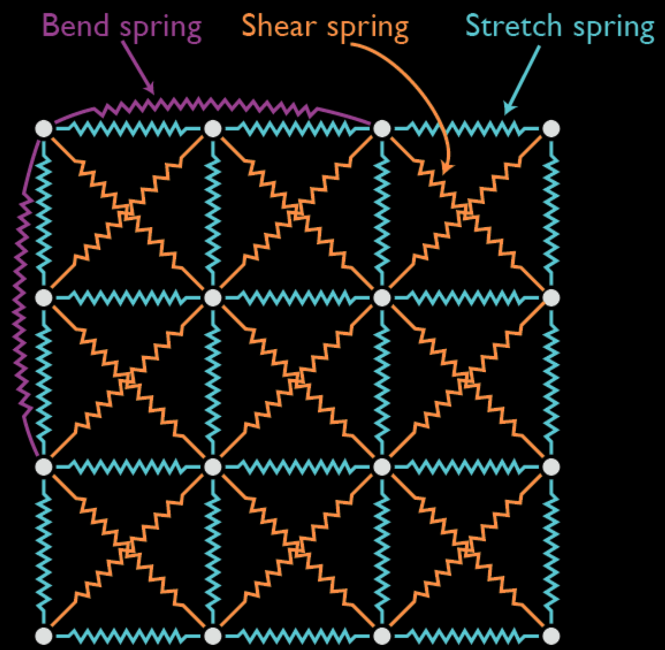

Information:
StudentID:0356611
CPU:Intel(R) Pentium(R) CPU G4400 @ 3.30GHz
GPU:GeForce 210/PCIe/SSE2
OpenGL version:3.3.0
Height Offset:0.000000
Rotate Angle :0
Spring Coef. :1500.000000
Damper Coef. :50.000000
DeltaT :0.001000
Integrator :Explicit Euler

Objective

- Simulate soft body by mass-spring system
  - Initialize the mass-spring system
  - Compute spring and damper forces
  - Handle collision
  - Integrate
  - Generate video

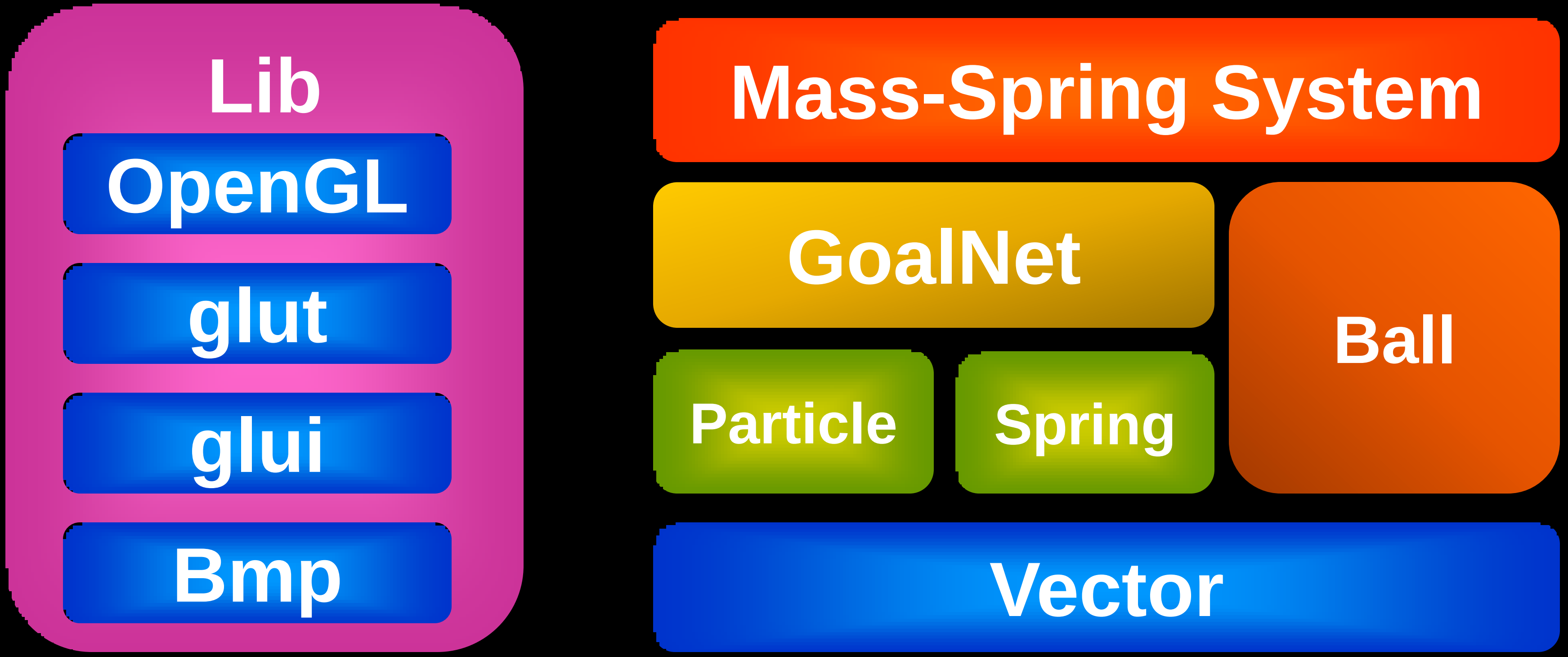
Mass-Spring System

- A set of particles are connected by springs
- A simple approach to simulate solid deformable objects



System Overview

- OS: Windows
- IDE: Visual Studio 2013

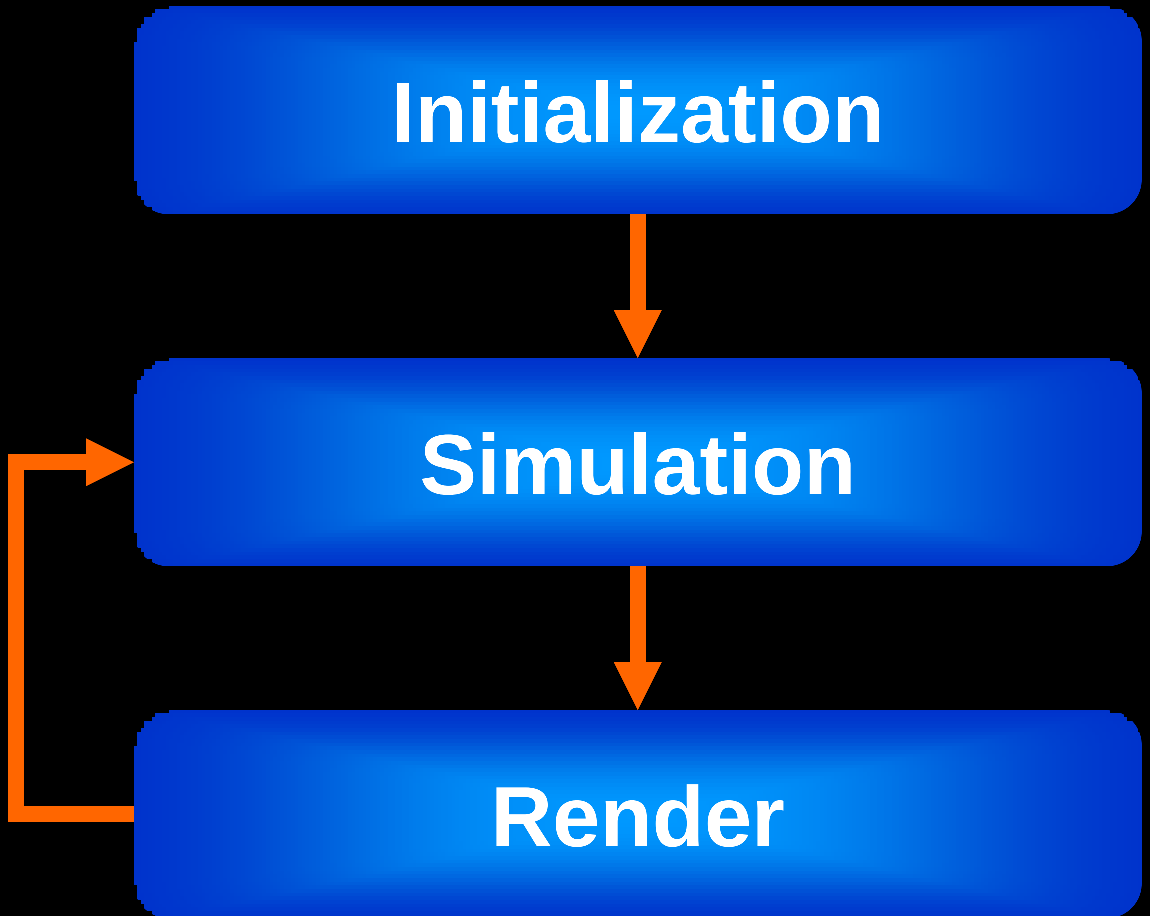


Note

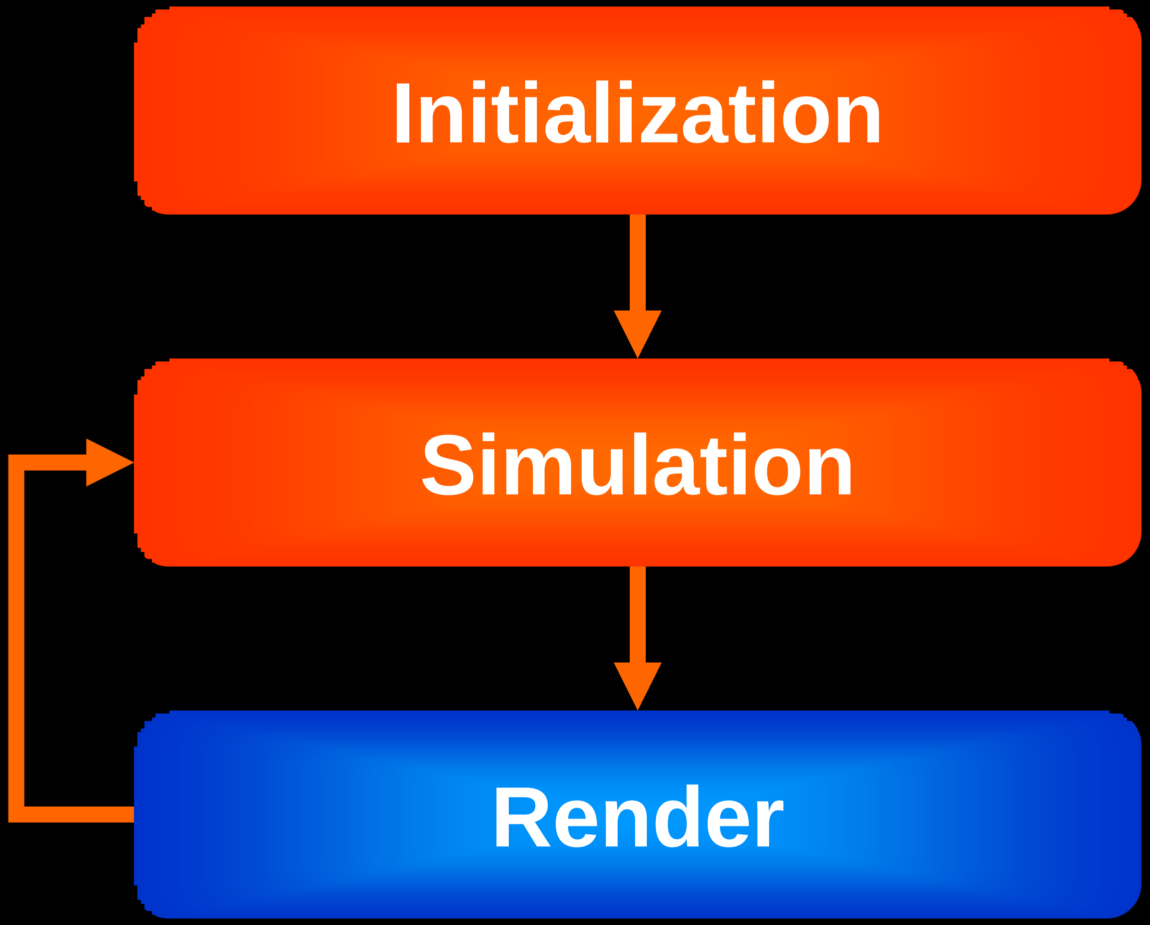
- Remember to modify you Student Id in the "Configuration.txt"

```
*StudentID
0356611
```

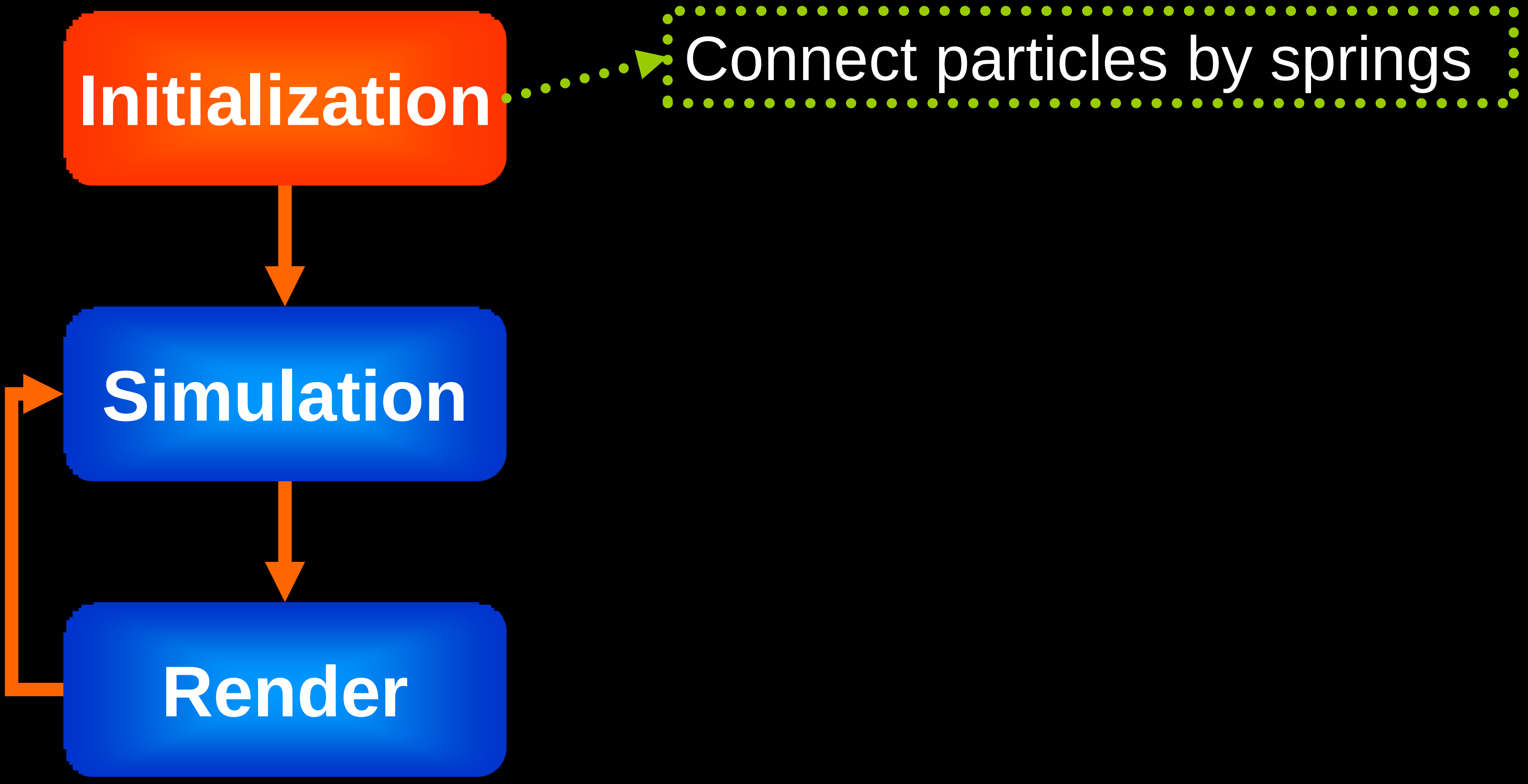
Algorithm Overview



What You Should Do

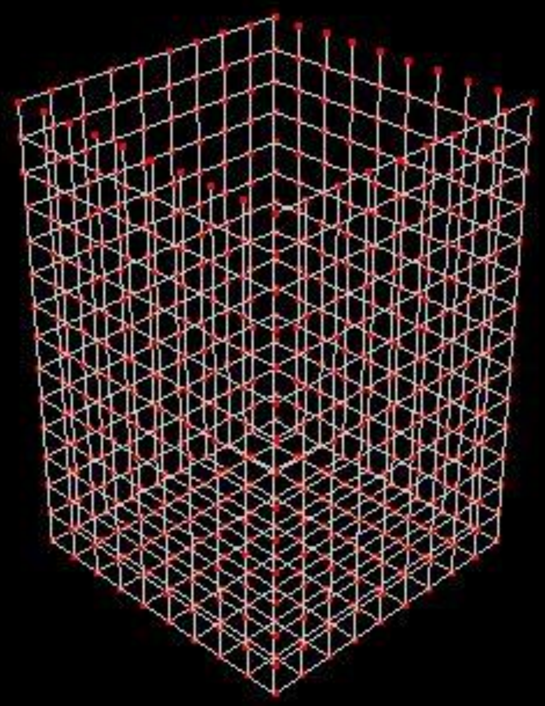


Initialization

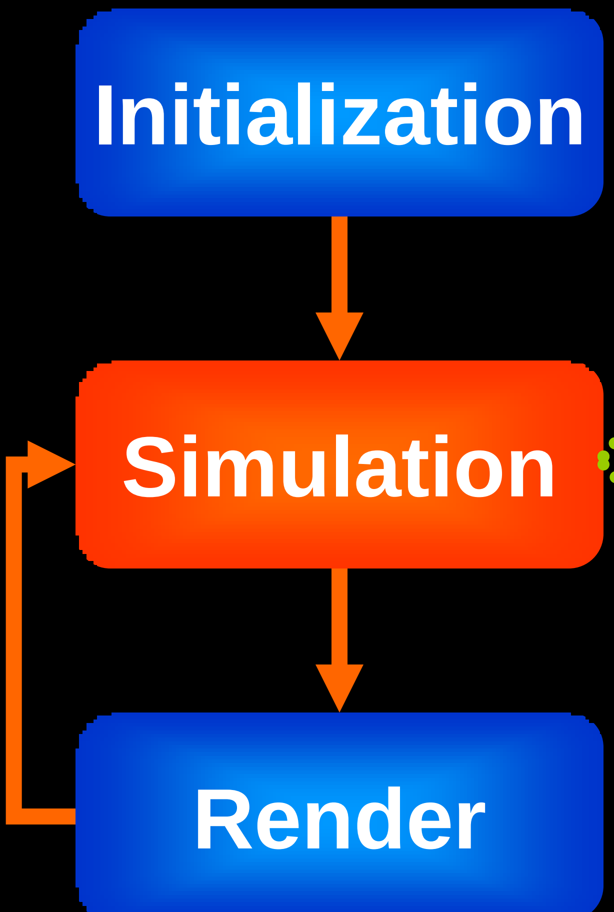


Structural Spring

- Establish the basic structure of the net
- Connect every particle to its direct 5 neighbors



Simulation



1. Compute spring and damper force
2. Handle collision
3. Explicit Euler Integration

Spring and Damper Forces

- Review “particle.ppt” from p.9-p.13

Point-Plane Collision

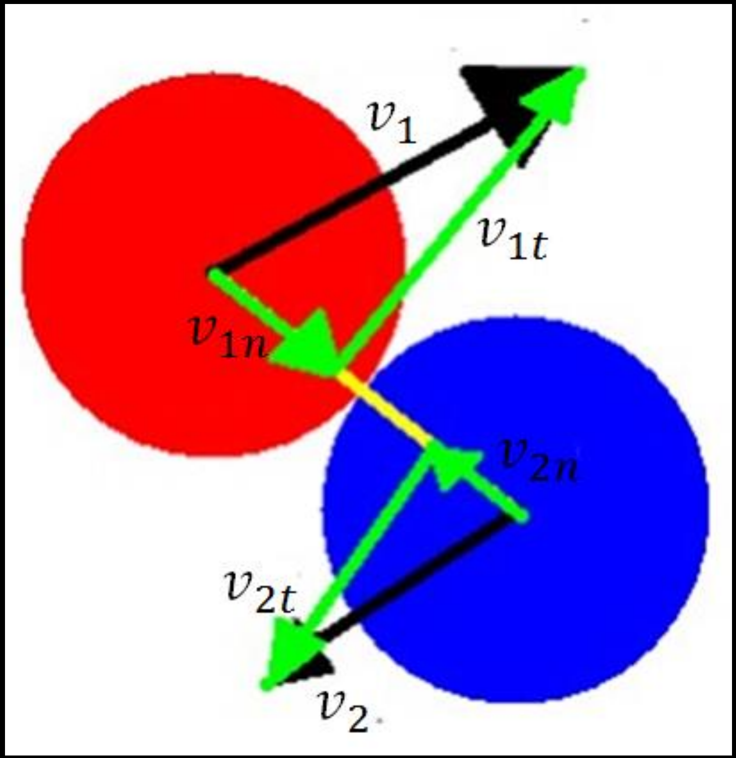
- Review “particle.ppt” from p.14-p.19
- The default plane is y=-1
- Recommend ε is 0.01
- Coefficient of restitution $k_r = 0.5$
- Coefficient of friction $k_f = 0.3$

Sphere-sphere collision

- handle ball-ball collision and ball-particle collision

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

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



Explicit Euler Integration

- Review “ODE_basic.ppt” from p.15-p.16
- The default $\triangle$t is 0.001

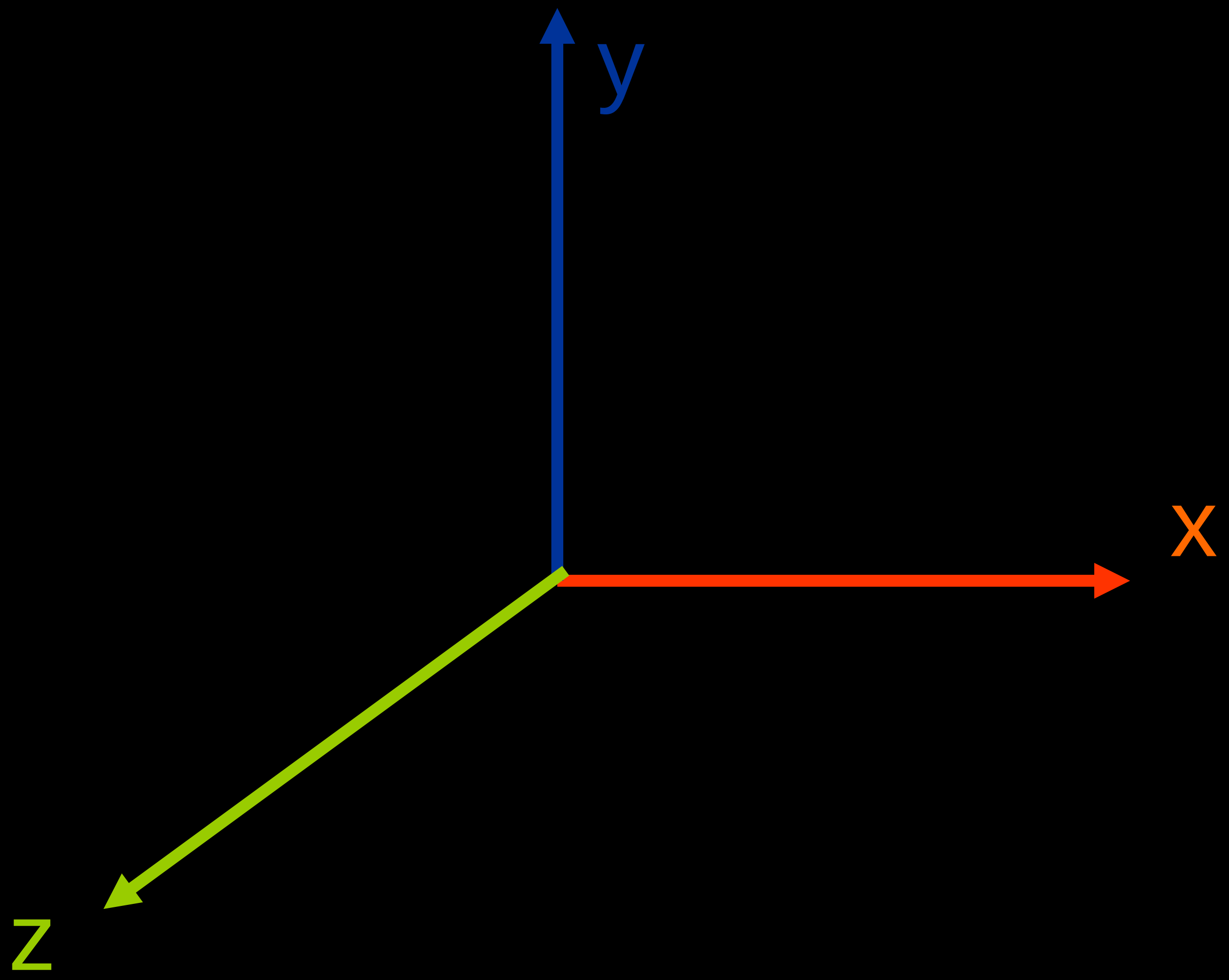
Runge Kutta 4^{th}(RK4) integration

- Review “ODE_basic.ppt” from p.21
- Or “Physically Based Modeling” from p.B5-p.B6

What You Should Know

- Coordinate
- Class “Vector3d”
- Class “CParticle”
- Class “CSpring”
- Class “GoalNet”
- Class “Ball”
- Class “CMassSpringSystem”
- GUI
- Configuration file
- Output video

Coordinate



Class “Vector3d”

- Provide basic vector operators and methods
  - +, -, *, / , DotProduct, CrossProduct, Length, Normalize
- “Length()” returns the length of the vector
- Operator “*” is neither dot nor cross product

Class “Vector3d” (con’t)

- “Normalize()” normalizes itself and returns the length of the original vector
- “NormalizedCopy()” returns its normalized vector
- Reference “Vector3d.h” in the directory “Math” for details

Class “CParticle”

- Important member variables

```
double   m_dMass;
Vector3d m_Position;
Vector3d m_Velocity;
Vector3d m_Force;
```

Class “CParticle” (con’t)

- No acceleration
- Set and Get acceleration will modify “m_Force”
- Reference “CParticle.h” in the directory “MassSpringSystem” for details

Class “CSpring”

- Important member variables

```
int   m_iSpringStartID;
int   m_iSpringEndID;
double m_dRestLength;
double m_dSpringCoef;
double m_dDamperCoef;
enType_t m_nType;
```

Class “CSpring” (con’t)

- Constructor

```
CSpring(const int a_ciSpringStartID,
        const int a_ciSpringEndID,
        const double a_cdRestLength,
        const double a_cdSpringCoef,
        const double a_cdDamperCoef,
        const Vector3d &a_rcColor,
        const enType_t a_cType);
```

Class “CSpring” (con’t)

- “a_ciSpringStartID” and “a_ciSpringEndID” are the index in the 1D array which stores particles

Class “GoalNet”

- Important member variables

  vector<CParticle> m_Particles;

  vector<CSpring> m_Springs;

  int m_NumAtWidth;

  int m_NumAtHeight;

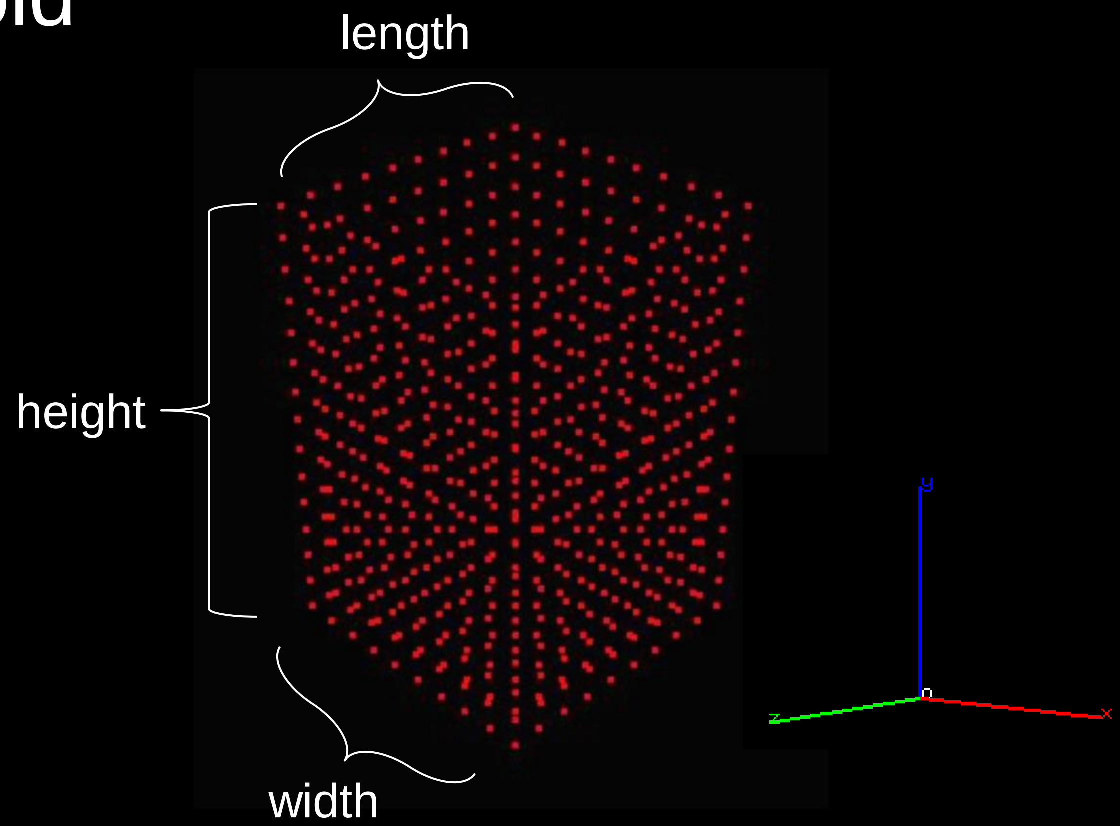
  int m_NumAtLength;

Class “GoalNet” (Con’t)

- The “vector” is an STL container, it is not the “Vector3d”.
- “m_Springs” is not initialized, you should construct springs in correct way as mentioned previously.

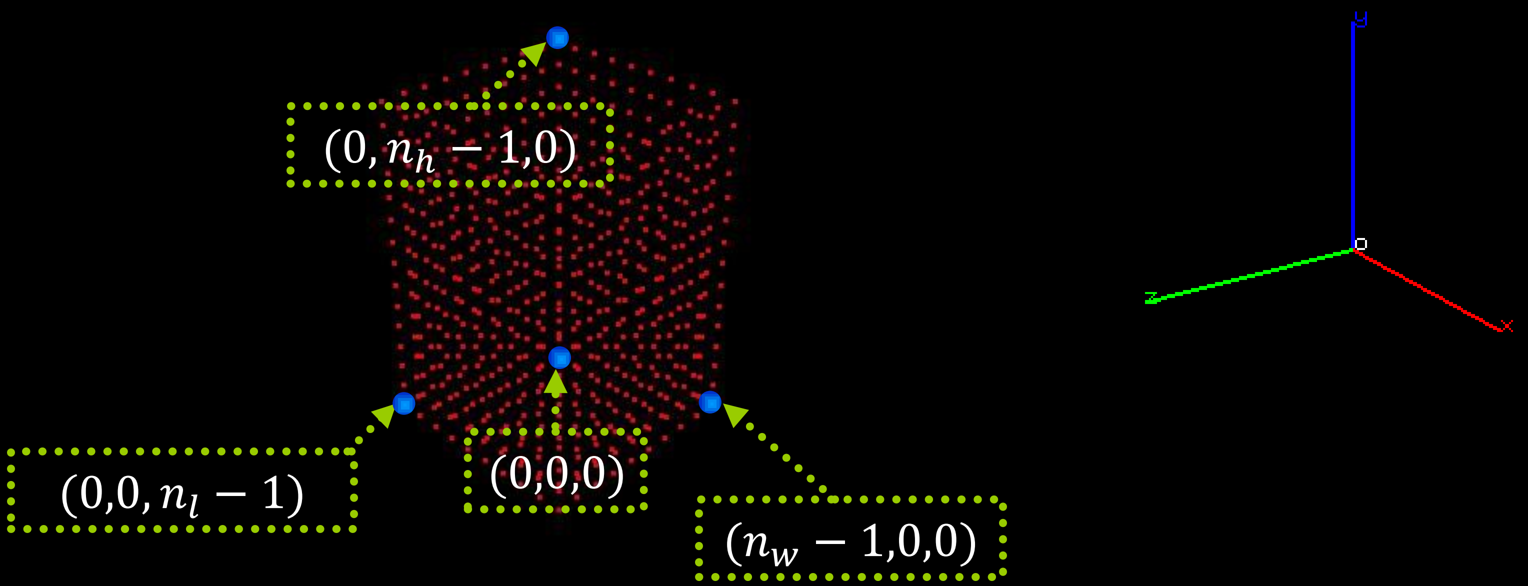
Class “GoalNet” (con’t)

- For simplicity, you can think of an goal net as a cuboid



Class "GoalNet" (con't)

- Use 'GetParticleID' to transform index between 1D and 3D array
  - n_w : number of particles at width
  - n_h : number of particles at width
  - n_l : number of particles at length



Class “GoalNet” (Con’t)

- Finish the following methods:

  InitializeSpring()
  ComputeInternalForce()
  ComputeSpringForce()
  ComputeDamperForce()

- Search “TO DO”

Class “Ball”

- similar to ‘particle’

```
double  m_dMass;
double  m_dRadius;
Vector3d m_Position;
Vector3d m_Velocity;
Vector3d m_Force;
```

Class “CMassSpringSystem”

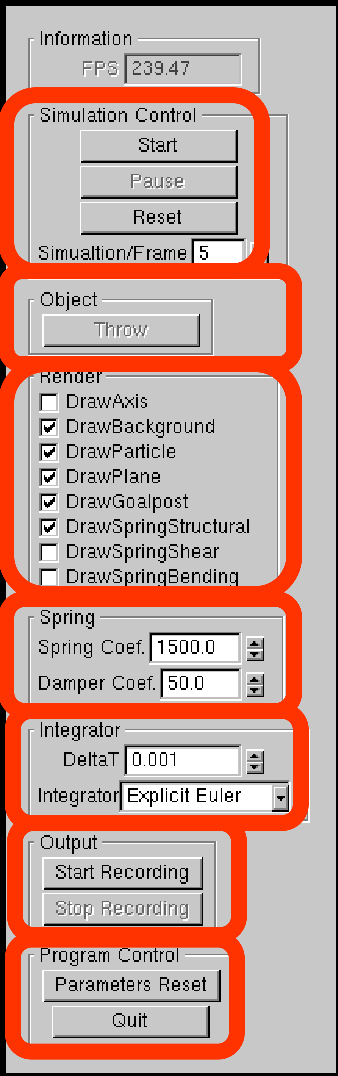
- Important member variables

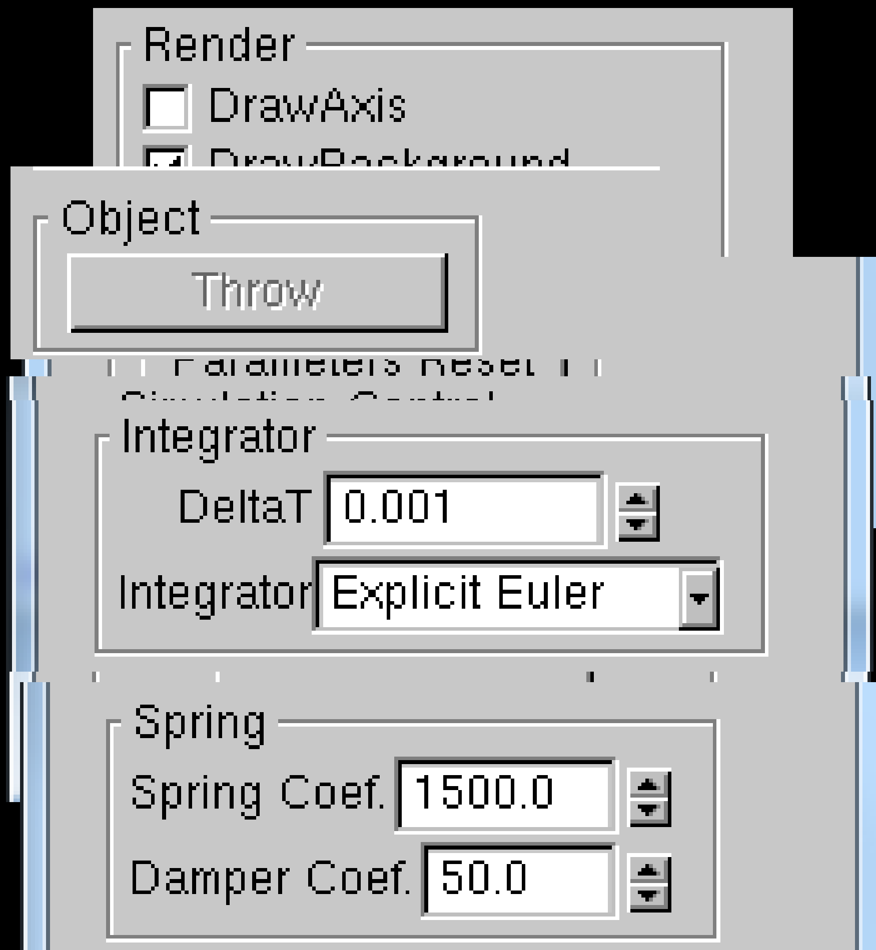
```
int m_iIntegratorType;
double m_dDeltaT;
GoalNet m_GoalNet;
vector<Ball> m_Balls;
```

Class “CMassSpringSystem” (Con’t)

- Finish the following methods:
  - ParticlePlaneCollision
  - BallPlaneCollision
  - BallToBallCollision
  - BallParticleCollsion
  - ExplicitEuler
  - RungeKutta
- Search “TO DO”

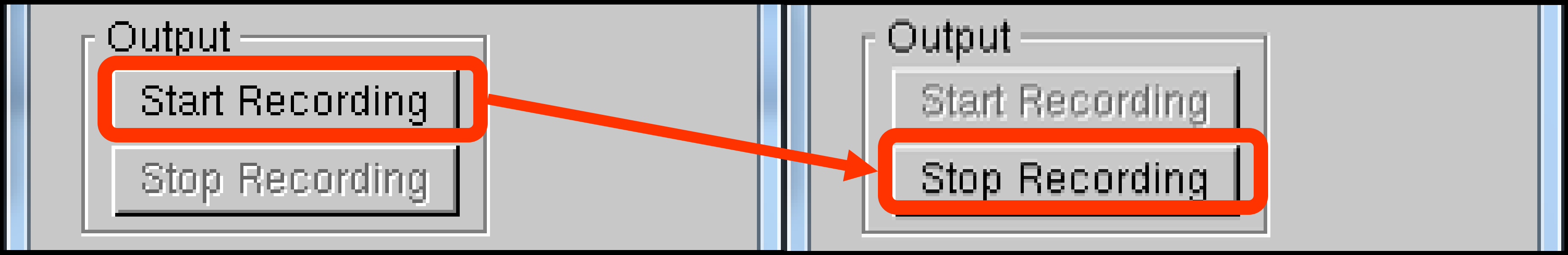
GUI


Information
FPS 239.47
Simulation Control
Start
Pause
Reset
Simualtion/Frame 5
Object
Throw
Render
DrawAxis
DrawBackground
DrawParticle
DrawPlane
DrawGoalpost
DrawSpringStructural
DrawSpringShear
DrawSpringBending
Spring
Spring Coef. 1500.0
Damper Coef. 50.0
Integrator
DeltaT 0.001
Integrator Explicit Euler
Output
Start Recording
Stop Recording
Program Control
Parameters Reset
Quit


Render
DrawAxis
Object
Throw
Integrator
DeltaT 0.001
Integrator Explicit Euler
Spring
Spring Coef. 1500.0
Damper Coef. 50.0

Output Video

- “Start Recording”
  - output a serial of images after clicking “Start” button
- “Stop Recording”
  - Stop recording and then output video



Output Video (con't)

- After few seconds, you will get a video in the directory "ParticleSystem"
- Be careful of the new video will replace the old video if their filenames are same

Configuration File

- All important parameters are in the "Configuration.txt"
- You can modify some of parameters to get different initialization
- For example,

```
*DrawPlane
true
```

```
*DrawPlane
false
```

In this case, plane will not be showed in the beginning

Recommended approach

1. Implement intergrator
   – Explicit Euler (or Runge Kutta)
2. Construct the spring
   – Draw the connected spring to confirm the constrained relations
3. Calculate internal force of goal net
   – compute spring force and damper force
4. Detect/resolve the collision

Grading

- Construct the connection of springs 10%
- Compute spring and damper forces 20%
- Handle collision 30%
  - particle-plane collision 5%
  - ball-plane collision 5%
  - ball-particle collision 10%
  - Ball-ball collision 10%
- Explicit Euler integration 5%
- Runge Kutta 4^{th} integration 20%
- Report: 20%
- Bonus 15%

Bonus

- Material of the net
- Rotating ball
- Any other fun (describe them in your report)

Suggested Outline of Report

1. Introduction/Motivation
2. Fundamentals
3. Implementation
4. Result and Discussion
   – the difference between Explicit Euler and RK4
   – effect of parameters
5. Conclusion

Submission

- Compress all the materials into a zip file
  - Naming rule: CA1_StudentID_Version,
  e.g., CA1_0356611_v001.zip
- Your zip file shall contain
  - Source code (ensure your project build successfully)
  - 2 videos(include Euler and Runge Kutta)
  - Report in pdf or MS word format, no more than 10 pages
- Upload all your materials to E3
  - No limit to the no. of times of upload
  - The latest version is your final submission

Late and Cheating

- Late policies
  - Penalty of 10 points of the value of the assignment/day
- Cheating policies
  - 0 points for any cheating on assignments
  - Allowing another student to examine your code is also considered as cheating

Deadline

- Deadline
  - Thursday ,2018/04/12 ,23:59

You can find TA in…

- Email:
  - liou8308@gmail.com
- Lab EC229B
  - Need appointment
- Please briefly mention your question when you contact me

Reference

- Slides on E3
- Real Time Physics (Siggraph 2008 class)
  - http://www.matthiasmueller.info/realtimephysics/index.html

Q&A