# CS171 Assignment 5: Cloth Simulation with the Mass Spring System

## Introduction

In this assignment, you are required to implement a simple mass spring system for simulating a piece of cloth, using OpenGL.

In the following, we will give you the specifics about what you need to accomplish, as well as some related guidelines in order to assist your programming.

## Programming Requirements

- **[must]** Define mass particle properties and initialize particles. [10%]
- **[must]** Define spring properties and create springs connecting particles. [30%]
- **[must]** Implement the mass spring simulation algorithm with gravity, including constraints that left up and right up corner of the cloth is fixed. [40%]
- **[must]** Under the default case, the cloth can become stable after some time of simulation. [20%]
- **[optional]** Apply external force to simulate the effect of wind. [5%]
- **[optional]** Add user interaction to the cloth, so that we can play with it. [10%]
- **[optional]** Couple the cloth with sphere colliders, and simulate the cloth falling on a sphere. [15%]
- **[optional]** Use implicit integration methods to achieve stability at time step of no less than 0.01 seconds. [15%]

## Notes

- We will offer a code skeleton for you. Please understand the functionality of each declared class and method in the code skeleton before you start.
- Carefully read README.md in the code skeleton before working on it.

## Submission

You are required to submit following things through GitHub repository.

- Project code in the Coding folder.
- A PDF-formatted report which describes what you have done in the Report folder.

Submission deadline: **22:00, Dec 16, 2024**

# Grading Rules

- You can choose to do the **[optional]** items, and if you choose to do it, you will get additional scores based on the additional work you have done. But the maximum additional score will not exceed 15% of the entire score of this assignment.
- **NO CHEATING!** If found, your score for the entire assignment is zero. You are required to work **INDEPENDENTLY**. We fully understand that implementations could be similar somewhere, but they cannot be identical. To avoid being evaluated inappropriately, please show your understanding of code to TAs.
- Late submission of your assignment will subject to score deduction.
- After the deadline of this assignment, we will schedule an on-site demonstration. The specific date and venue will be posted on Piazza.

# Skeleton Project/ Report Template

- The skeleton program and report template will be provided once you accept the assignment link of GitHub classroom. If you accept the assignment through the link properly, a repository which contains the skeleton project and report template will be created under your GitHub account.
- Please follow the template to prepare your report.

# Implementation Guide

Here we will instruct you to finish each part of this assignment. It is recommended to firstly overview the entire code skeleton and then fill each part by following your own understanding. This guide will not cover the optional parts, which mostly count on your individual study.

## Git Classroom

Accept the assignment in this link through Git classroom to start your assignment.

## 1. Understand the mass spring system

The mass spring system is a simple system for simulating deformable objects. It can also be used to simulate materials like cloth. Please read the tutorial from Stanford to learn how it works.

## 2. Mass particles

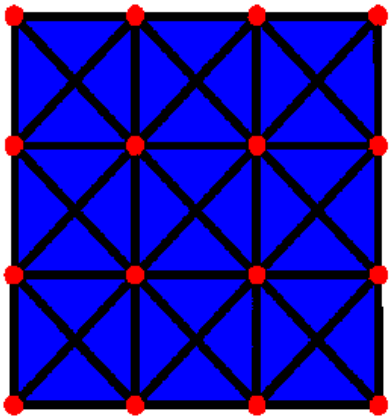
The first task is to create the mass particles, given a rectangular cloth. You should define both the constant and dynamic physical properties you need in the struct `RectClothSimulator::MassParticle`, for example velocity. Then initialize each particle in `RectClothSimulator::createMassParticles`, where part of the creation code is already given.

Related files: **cloth_simulator.hpp**, **cloth_simulator.cpp**

## 3. Springs

The next step would be creating springs to connect all the mass particles you created. You can use the simplest spring layout, shown in the figure below, where each particle is connected to its 8

neighboring particles. Of course, you can use more complex layout to achieve better results.



The simplest spring layout

Again, you should first define the physical properties you need in struct `RectClothSimulator::Spring`. Then you need to implement the whole `RectClothSimulator::createSprings` function. In this function, you should not only create the springs and store them, but also update the spring connection information for particles.

While actually not, the springs are directed in our program, because you need to store the direction of a spring to help calculate its forces. Thus, the spring have "from" and "to", and in the `RectClothSimulator::MassParticle` struct you can see two different lists for springs with its "start" and its "end" connected to it. You should correctly handle the connection information.

NOTE: The spring constant (stiffness) is given as "stiffnessReference", because we assume that not all springs share a single stiffness value, and the stiffness of difference springs should have some kind of relation. Thus a reference value of stiffness is given, and you should initialize the stiffness for each spring according to it.

Related files: **cloth_simulator.hpp**, **cloth_simulator.cpp**

## 4. Solver/Simulator

After both the mass particles and the springs are created, you can start implementing the simulation algorithm. As shown in the tutorial from Stanford, the equations of motion in the system reads (you can ignore the wind force part):

$$F_{\text{net}}(v) = \text{Mg} + F_{\text{wind}} + F_{\text{air resistance}} - \sum_{\text{Springs} \in v} k(x_{\text{current}} - x_{\text{rest}}) = \text{Ma}$$

$M$ = mass of vertex
$g$ = gravity vector = $(0, -9.8, 0)$
$k$ = spring constant
$x_{\text{current}}$ = current length of spring
$x_{\text{rest}}$ = rest (initial) length of spring
$F_{\text{wind}}$ = wind vector
$F_{\text{air resistance}} = -a * \text{velocity}(v)^2$

To solve this equation programmatically, you need to choose an integration method, as listed in the tutorial from Stanford. The (semi-implicit) symplectic Euler's method and the (explicit) Verlet

algorithm are easy to implement, but be careful that they are not stable on large time steps. The algorithms are shown below:

$$\mathbf{v}_{t+\Delta t} = \mathbf{v}_t + \Delta t \left(\frac{\mathbf{dv}}{\mathbf{dt}}\right)_t$$

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \Delta t\, \mathbf{v}_{t+\Delta t}$$

Symplectic Euler's Method (Semi-Implicit)

$$\mathbf{x}_{t+\Delta t} = 2\,\mathbf{x}_t - \mathbf{x}_{t-\Delta t} + \left(\frac{\mathbf{dv}}{\mathbf{dt}}\right)_t (\Delta t)^2$$

Verlet Algorithm (Explicit)

The function you need to implement is `RectClothSimulator::step`. You should update the particle positions using the integration method you chose, and then update the springs.

Related files: **cloth_simulator.cpp**

### 5. Constraints

After updating particles and springs, you need to apply constraints of the cloth simulation. In this assignment, the constraints are fixing the left up and right up corner of the cloth. This should be done in the same function, i.e. `RectClothSimulator::step` .

Related files: **cloth_simulator.cpp**

## Expected Results

Under the default case, after some time, the stabilized cloth looks like: