Physics-Based Cloth Animation

Final Assignment for Numerical Analysis Discipline

Felipe Bernardi, Matheus Venturelli, Pedro Ferraz

Department of Informatics
PUC-Rio
Brazil
December 18th, 2020

1 Introduction

This project consists in a 3D simulation of cloth physics using Verlet Integration. It consists of two parts: the physics simulation algorithm written in C++, and the visualization created with OpenGL 4+ using Qt Creator.

The goal of the project is to create animations of cloth under the influence of wind and gravity. To simulate it, numerous particles are created with bars of specific length between them. Before drawing every frame, each particle has its position updated according to the forces exerted upon them. As particles move, their distances relative to one another change, and the bars between them deform. To maintain the aspect and structure of cloth, an algorithm tries to readjust each bar to its original length, which creates a visual motion effect that resembles real life cloth movement.

2 Methodology

The first thing to take note is how to create an object that behaves like cloth. One way to do so is to create a mesh of 3D particles connected by bars. Each particle should have a bar connecting it to all of its immediate neighbours and should be defined as fixed or free to move. Also, each bar should have a fixed length equal to the distance between the particles that it links. This allows for the composition of generic shapes. An example of a rectangular mesh is the grid represented in Figure 1, where white and black dots represent fixed and free particles, respectively.

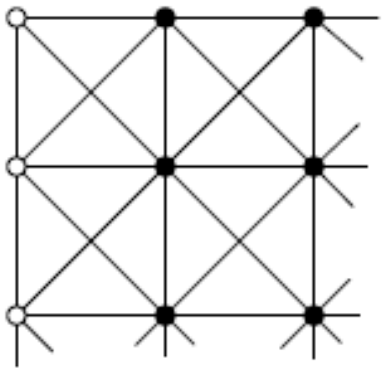

Figure 1: Rectangular cloth mesh.

In order to simulate proper behavior and prevent too much bending, it is also necessary to make connections to particles that are two bars away, as represented by the dashed lines in Figure 2. Note that these bars must also have length equal to the distance between the particles.

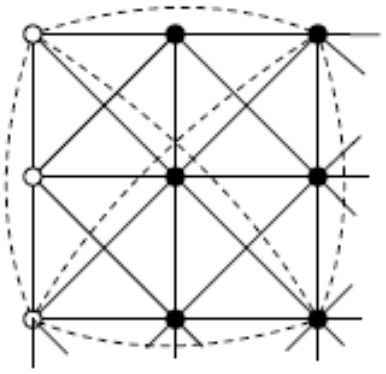

Figure 2: Rectangular cloth mesh with more connections.

Having this structure, it is necessary to make it behave like it should. To achieve this goal, we can firstly simulate the action of forces in each particle using Verlet Integration, and then use a relaxation algorithm to try restoring the original length of each bar, maintaining the cloth structure.

The method of Verlet Integration is broadly used in 2D and 3D particle simulation in games and can be described as in Equation 1, where x_i is the position of each particle on step i, δ is a smoothing coefficient, h is the step's size, m_j is the mass of particle j and f_j is the sum of the forces acting on particle j.

$$\mathbf{x_{i+1}} = \mathbf{x_i} + (1 - \delta)(\mathbf{x_i} - \mathbf{x_{i-1}}) + \frac{h^2}{m_j}\mathbf{f_j} \tag{1}$$

As this equation depends on the current and previous position of each particle, we can start the simulation for a specified initial position $\mathbf{x_0}$ and velocity $\mathbf{v_0}$ by performing one Euler step to determine the next position $\mathbf{x_1}$, as described in Equation 2.

$$\mathbf{x_1} = \mathbf{x_0} + h\mathbf{v_0} \tag{2}$$

Once the next position for each particle has been computed by Verlet Integration, it is necessary to apply a process of relaxation to try restoring each bars' size. This is done by an algorithm that iterates over all the bars readjusting the positions of particles as shown in Figure 3.

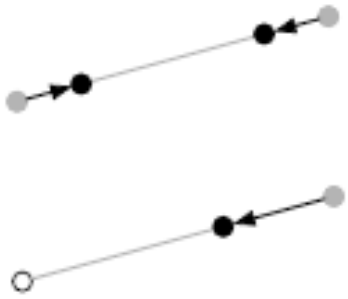

Figure 3: Relaxation applied to free and fixed bars.

It is proven that, after a large enough amount of relaxations, the system converges, and every bar is brought back to it's original size. By adopting smaller number of iterations, the bars aren't able to restore their original length and the cloth behaves more loosely.

3 Results and Analysis

To create a simulation close to real behavior, it is necessary to adjust the variables in Equation 1, as well as the number of relaxations on the bars and forces applied. On the following simulations, we will test the system's behaviour while varying the amount of relaxations, assuming values of $\delta = 0.02$, $h = 0.05$ and $m = 0.2$ for every particle. The force $\mathbf{f_r}$, applied to every particle is described on Equation 5, which is the sum of gravity (Equation 3), and wind (Equation 4), where t represents time, in milliseconds, since the day's start.

$$\mathbf{f_g} = (0, -9.8, 0) \tag{3}$$

$$\mathbf{f_w} = (5 + 2sin(\frac{t}{2}), 6 + 6sin(\frac{t}{2}), -2 + sin(\frac{t}{2})) \tag{4}$$

$$\mathbf{f_r} = \mathbf{f_g} + \mathbf{f_w} \tag{5}$$

With the previous definitions, it is possible to start testing the simulation.

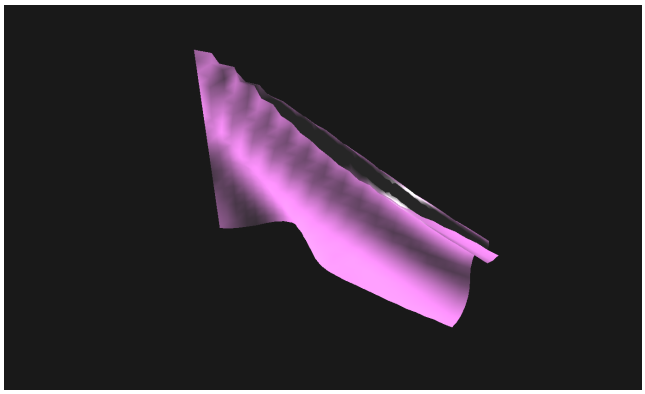

Figure 4: Simulation with 5 relaxations.

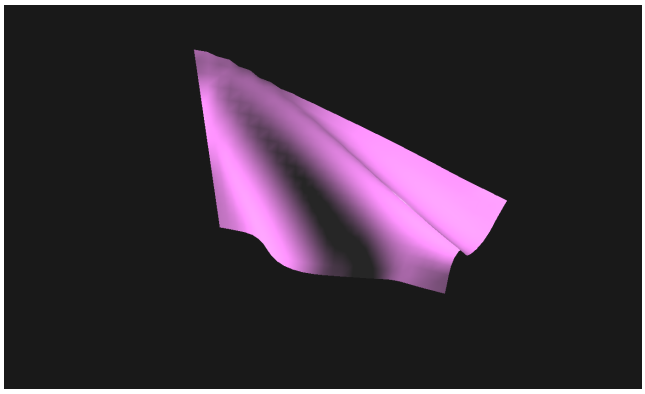

Figure 5: Simulation with 20 relaxations.

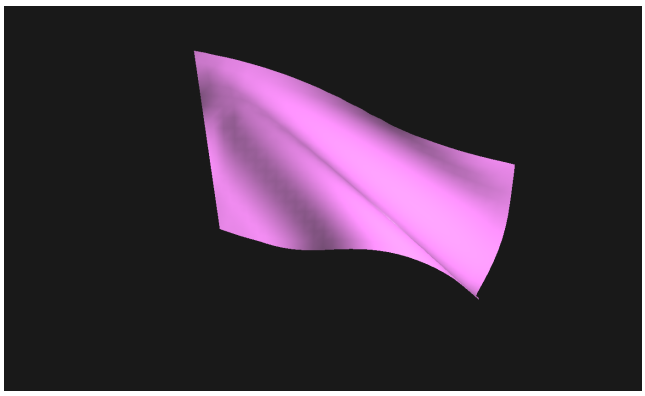

Figure 6: Simulation with 40 relaxations.

Analyzing the figures, we can verify that for a smaller number of relaxations, the simulation starts to form edges that shouldn't occur in real cloth. As we increase the number of relaxations, it starts to behave more like cloth. However, for large enough values, it becomes too rigid.

Another important aspect to verify is performance. The data on Table 1 represents the performance tests made on a Macbook Pro, with a Intel Iris Pro graphics card with 1536 MB VRAM (Dynamic, Max) and 16 GB RAM. The table compares different values for number of relaxations, specifying the mean time to take one step of simulation and render a frame, and the mean frames per second (FPS) of 1000 frames.

Number of Relaxations	Mean Simulation Time in Seconds	Mean FPS
05 Relaxations	0.0177711	57.2139
10 Relaxations	0.0179318	56.8465
20 Relaxations	0.0251436	40.3116
30 Relaxations	0.0367839	27.3911
40 Relaxations	0.0488167	20.5959

Table 1: Simulation time and FPS for different numbers of relaxations.

Taking note of similarity to reality and performance, it is reasonable to choose between 20 and 30 relaxations to make it possible to run in real time with good results.

An important observation is that the adequate value for the number of relaxations depends on the size of the mesh and which points are fixed. Different configurations may require larger values for number of relaxations or may be good enough with lower values.

The previous simulation was made with a 20 by 30 particles rectangular mesh. To simulate generic meshes, the algorithm can be initialized with a list of particles containing their position and a Boolean variable indicating if they are free or fixed. It must also receive a graph, in the form of adjacency list, representing the bars connecting each particle. This implementation supports the simulation of any generic mesh. However, the visualization tool implemented only supports rectangular meshes.

4 Conclusion

With all the techniques involved, it was possible to create a smooth animation that resembles real life motion. We found that a number of relaxations between 20 and 30 is a good fit between a convincing animation and demanding performance. By using lower values, the mesh would move in unnatural ways, while with larger values, it would not be possible to run in real time.