DH2323 Computer Graphics and Interaction

Project report
- Interactive Cloth in 3D

Johanna Gustafsson, jgu7@kth.se
Marcus Wallberg, mwallb@kth.se

Abstract

In this paper we have implemented a real time cloth simulation in our own SDL based graphics engine, that handles cloth as particles. The engine is CPU-based and written in C++. Running on an Macbook Pro (late 2013), we managed to interact with a grid of 20 by 20 nodes with a framerate of 41 in average. The camera is movable and the cloth material can be interacted with from different angles.

1 Introduction

The goal of this project was to implement an interactive real time cloth in 3D. The intention was to create a cloth which reacts realistically to movement when dragged by the mouse. The progress of this project can be followed on the blog: *http://xml.csc.kth.se/~mwallb/CGIproject*

2 Background

Cloth simulation is usually modelled as a mass spring model where each vertex is connected to adjacent vertices via a spring. The goal is usually to render as accurate, fast and as high-resolution as possible [3].

The physics of cloth rendering takes a lot of computation power to get right. This is a big challenge to do in real time when you ramp up the polygon count of the cloth surface. Most of the models that we found used nodes with three different types of springs. It seems like the bests results have been implemented with the GPU.

3 Technology overview

3.1 IDE:

- Visual Studio 2015
- Xcode

3.2 Renderer:

- Rasterization using the raytracer from lab3, based on SDL in C++
- (maybe some Open GL for GPU acceleration)

4 Implementation

When beginning the implementation, we used the knowledge as well as some of the code we had gotten from the previous rendering labs in this course. We created a SDL surface as a screen and started off by implementing the nodes of the cloth. Each node was given a set of neighbours - a set of neighbouring nodes, between which we drew lines by interpolating the distance between each pair of nodes. The lines were drawn to the closest neighbours in horizontal and vertical directions. These lines represents our structural springs.

Often when implementing cloths, it is typical to use more than one type of springs between the nodes [3]. See figure 1 for example.

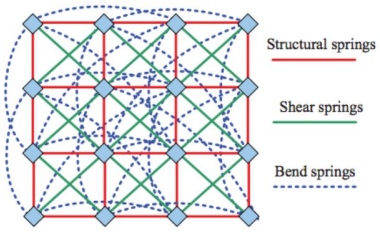

Figure 1: *Implementing the structural model with 3 types of springs would look something like this. We only used structural springs.*

When we got the grid working, we added the ability to move around in the room with the camera to see different perspectives. It is possible to move forward and backward, and to rotate the camera to the left and right. To do this we connected the keyboard steering to the VertexShader function we had from a previous lab. This function controls the rotation and projects the 3D coordinates into 2D.

The next step was to implement the mouse movement. Since we had trouble with the mouse click functionality from SDL on one of our computers, we decided not to rely on mouse clicks. To move the cloth, you ought to press the space bar on the keyboard at the same time as moving the mouse or moving the touch pad. When the space bar is being hold down more than one frame, we save the movement of the mouse and loop through all the nodes to find out which of the nodes that are

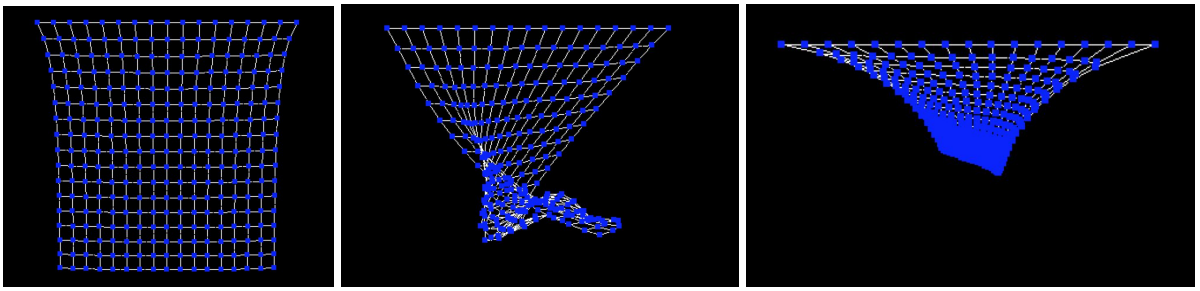

Figure 2: *Real time render screenshots from the cloth material in our graphics engine. The image to the left shows the initial condition of a 17 x 17 (289 vertices, 1159 springs) cloth with the parameters shown in figure 3. The same configuration is shown in the middle image with some mouse interaction. The right image is the initial condition with the spacing variable set to 0.05, this is the control of how strong the springs between each vertex would be.*

close enough to the mouse to be affected by the movement. We use a parameter set to a distance away from the mouse so that all the nodes within this radius is directly affected by this pulling force. We also save the exact distance to these nodes, so that the force is dependent on the distance.

In order to implement the movement simulation of the cloth, we had to set some parameters that affects the pulling force and the gravitation. These parameters are shown below in figure 3.

```
float gravitation = 9.82 * 1e-4;
float gForce = 0;
float spacing = 0.1;
float lineForce = 0.5;
```

Figure 3: *These are the global parameters that control the behaviour of the cloth simulation. Since the particles don't have an implemented mass, the gravitation variable is multiplied with a small number which works the same as if every particles (vertex) would be given a mass.*

When it has been detected that the mouse is being dragged, we update the physics. The physics in our implementation is not an accurate model of the real world, it's rather a simulation of that is optimized to feel realistic and most importantly run very fast on a CPU based physics engine.

To determine the next position of the particles in the cloth simulation, the Euler method is commonly used to predict the future values from a function [7]. It is less accurate than calculating the function values, but have an upside in performance. In our implementation we used an even more simple model:

$$F_{res} = const * g + const * sf + const * mf$$

Formula 1: *g is the gravitation, sf is the spring force and mf is mouse force.*

As can be seen in (1), we implemented a simple cloth physics where the F_{res} is applied iteratively on each vertex.

5 Result

Our physics engine manage to simulate the cloth with interaction. Avarage framerates are being presented in Table 1 below. Figure 2 shows some screenshots from the end result.

Particles	Springs	FPS (average)
9	36	62
100	400	51
400	1600	41
1600	6400	26
2500	10000	18

Table 1. *These are the average frame rates that we got on a Macbook Pro (late 2013) Processor: 2,3 GHz Intel Core i7 and memory: 16 GB 1600 MHz DDR3. FPS, frames per second, are the averages we got when the program ran for 60s with some user interaction. The screen size was 640 × 480 pixels.*

6 Discussion

The frame rates from the result is based on the initial distance from the cloth material. When the camera is closer the frame rates drops rapidly. This is a cause of the limitations of the graphics engine built for this project. If this method was implemented in another engine such as Unity or Unreal engine, this would most likely not have the same effect and we believe that the frame rates in this result might hold.

If we were to do the same project again, there is not too many things we would have changed. Even though there are some features we would have liked to add, we did not encounter any troubles that had any big negative effects on the outcome.

We were able to reuse some bits of code from previou labs, which was to our advantage, and we were also able to do some of the work in parallel, through communication and the ability to work on different functions at the same time. Most of the process went smoothly and merching the code was sometimes tricky, but due to good structure and planning of the code, it was manageable.

It was both interesting and motivating to be able to use previous knowledge from this course when doing a project of our own. While we had some use of previous knowledge, we also learn a lot about particles and animation. We did not do the animation track on the labs, but leant a lot by doing this project. We also learnt about physics and mechanics, and we also had the opportunity to develop our skills in C++.

7 Conclusion

All in all, our implementation method worked as expected and the as Table 1 shows. It is however difficult to compare the results that we got in our project to other projects. Our project ran in real time compared to most of the other that we looked at. Most of the other also rendered every polygon with light included, which we did not. [6]

The closest to our project is the Nvidia physics engine Flex [7] which outperforms our implementation. Flex is implemented on the GPU with most of the benchmark tests run on computers with very high performance. We believe that both our engine from this project and cloth physics would benefit a lot by running on better hardware.

7.1 Future work

One improvement would be to add collision detection. We would also like to make it GPU-based, so that it can run faster.

In this implementation, we only used structural springs. One way to make it look more realistic would be to also implement shear springs as well as bond springs (see Figure 1).

Another way to make it seem more realistic is to keep trying to tweak the parameters to give them more appropriate values. This could give a whole other feel to the cloth. The parameters used now are chosen since they, in our perception, seem realistic even though it can always be improved. One way to get a better result is to do a user study, where the subjects are being presented with cloths with different parameter settings, while being asked to choose which of the cloths they think look more realistic. This could be done by an A/B-testing for example.

When the mouse is being dragged close enough to some of the nodes of the cloth, the cloth is moving. There is no check to see if the mouse is placed within the edges of the cloth. So it is possible to move the cloth without actually “touching it”. We would have liked to implement a better functionality concerning this.

When we chose this project, one of our goals was to make the cloth tearable. Unfortunately, we did not have enough time to look deeper into how this could have been implemented.

There were also some minor fixes that we would have wanted to add, which was left out due to lack of time for this project, but on the whole, we are pleased with the outcome.

References

[1] Heath, I. Accelerating Cloth Simulation with CUDA. http://www.andrew.cmu.edu/user/iheath/418/cloth/CUDAClothSimulatorFinalReport.html

[2] Provot, X. (1996). Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior. *In Graphics Interface,* 147-154. https://graphics.stanford.edu/courses/cs468-02-winter/Papers/Rigidcloth.pdf

[3] Tang, M., Tong, R., Narain, R., Meng, C., & Manocha, D. (2013). A GPU‐based Streaming Algorithm for High‐Resolution Cloth Simulation. *Computer Graphics Forum*, *32*(7), 21–30. http://doi.org/10.1111/cgf.12208

[4] Tearable Cloth. http://codepen.io/dissimulate/pen/KrAwx/

[5] Unite 2016 - GPU Accelerated High Resolution Cloth Simulation. https://www.youtube.com/watch?v=kCGHXlLR3l8&t=1825s

[6] Rahul Narain, Armin Samii, and James F. O'Brien. "Adaptive Anisotropic Remeshing for Cloth Simulation". ACM Transactions on Graphics, 31(6):147:1–10, November 2012. Proceedings of ACM SIGGRAPH Asia 2012, Singapore.

[7] David Baraff Andrew Witkin. “Large Steps in Cloth Simulation” Robotics Institute Carnegie Mellon University. COMPUTER GRAPHICS Proceedings, Annual Conference Series, 1998

[8] Nvidia Flex. https://developer.nvidia.com/flex