Physically Based Simulation

Pro-Seminar WS 2015/16

Cloth Simulation

Elias Zischg, Daly Chea, Gerhard Aigner

Fri Feb 5 2016

1 Introduction

This project is about the simulation of a piece of cloth interacting with rigid bodies.

During the simulation, the cloth interacts with its environment i.e. self-collision and cloth-object collision must be regarded. We use collision detection between the cloth object represented as mass spring system and rigid body objects represented as triangle mesh. Self-collision of the cloth object is not considered.

The collision response method uses spring penalty forces for the mass points of the cloth object. The rigid objects do not respond to collisions. The equation of motion for the mass spring system is solved by a symplectic euler method.

The appearance of the cloth is primarily the response to these conditions. The simulation shows the cloth falling down and covering a sphere object. We use OpenGL to render the scene.

2 Set-up of the scene

The modeling of the scene has been done with the 3D-graphics software *blender*. Figure 1 shows a screen-shot with the model used for the cloth simulation. Once, the wire-frame and the texturing is done, the model is exported as wavefront object. The result is a *.obj*-file and a *.mat*-file. In a post-processing step with the *blender2oGL* [1] tool, also developed during the project, the *.obj*-file is translated to a set of C-source and header files, ready to include into the project. Thus, almost any scene setting is easily included into the cloth simulation program, as long as the target computer can handle the processing effort[2].

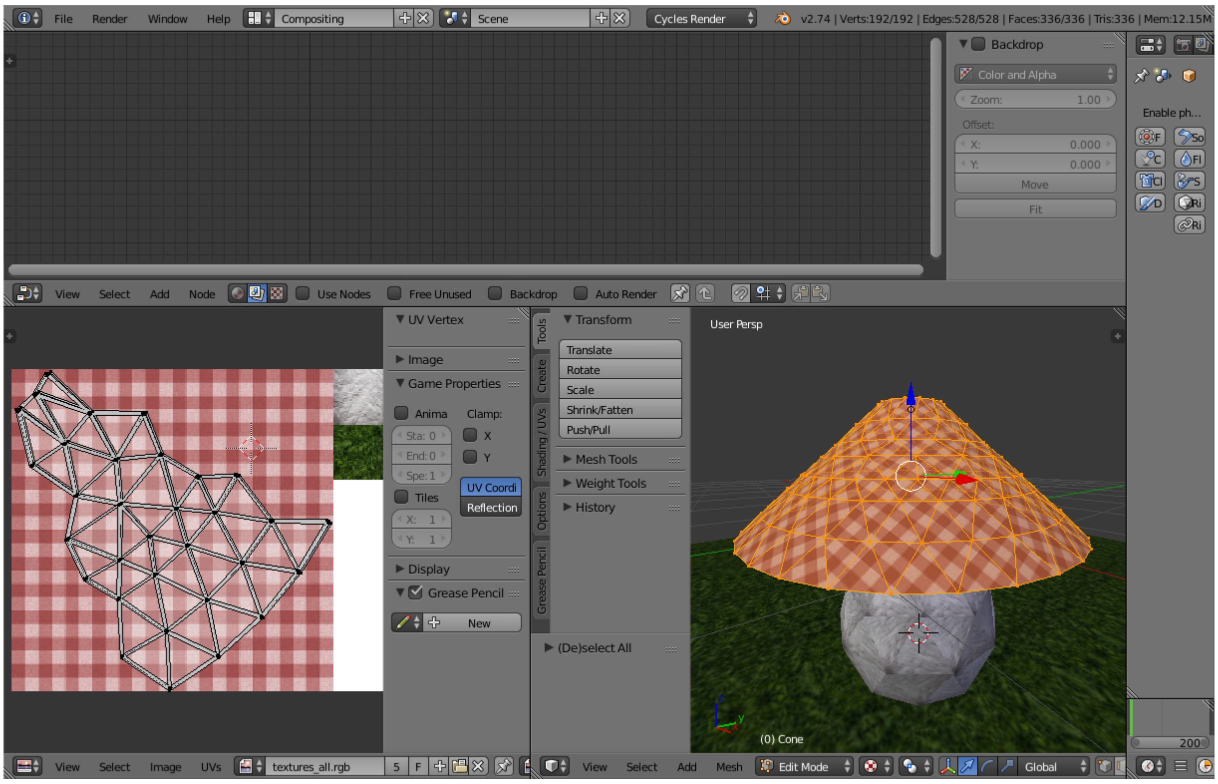


Figure 1: Blender is used to set up a model. Subsequently, the model is exported as wavefront object file and converted to C-files with the *blender2oGL* tool.

The initial setting shows a skirt above a sphere. The sphere is hovering above the ground. As shown in figure 2b, the skirt falls down, on top of the sphere and wraps around it. After the collision with the surface of the sphere,

[1] The *blender2oGL* tool is available from our project repository on GitHub: `https://github.com/3eee3/ez-dc-ga`.

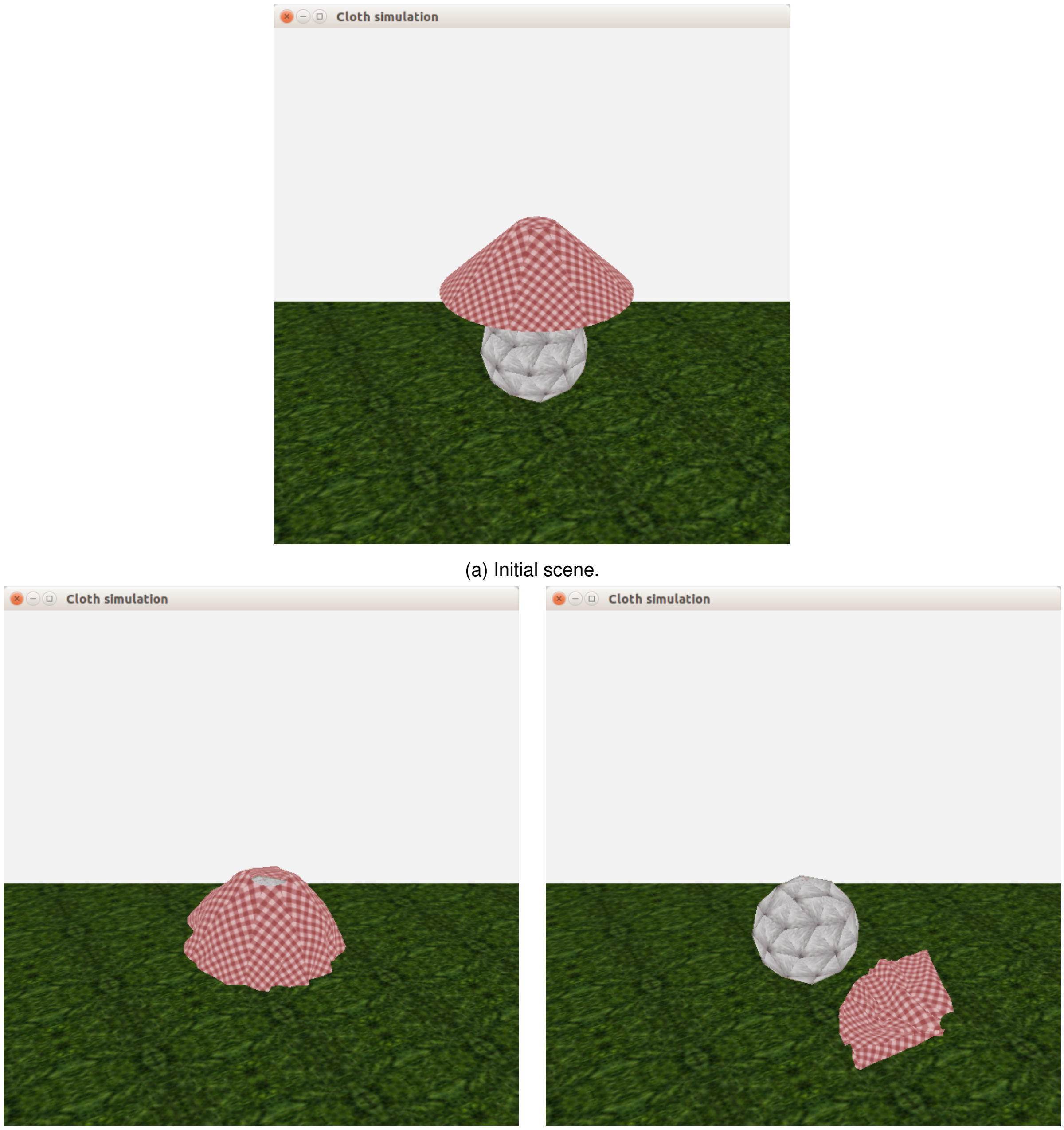
[2] The algorithms to compute the physics simulation and the collision detection are quite CPU-time expensive. Besides that, the computation is forced onto one single core, such that only small scenes will run smooth and seamless.

the cloth slips off, driven by the gravity, applied to the not exactly centered cloth. The penalty force of the collision detection algorithm has also an effect which drives the skirt off the sphere. Finally, the skirt falls down to the floor, where it collides with many wrinkles.

We decided to implement different scenes for debugging reasons[3]. So, a second scene with two dice is available by uncommenting the line

```
#define _DEBUG_OGL_MODEL
```

in the file *model_mapping.cpp*.



(a) Initial scene.

(b) Sphere wrapped by the cloth.

(c) Finally the skirt falls down to the floor.

Figure 2: The cloth is initially placed above a sphere. It falls down onto the sphere and slips off to the floor.

[3]During the implementations happened some obstacles which led to non-running code. Because of that, we created a reduced version of the *skirt* program without the connection to the *blender* generated model. It contains a hard-coded grid of masses and springs with two corners fixed on top. It is also available on our project repository.

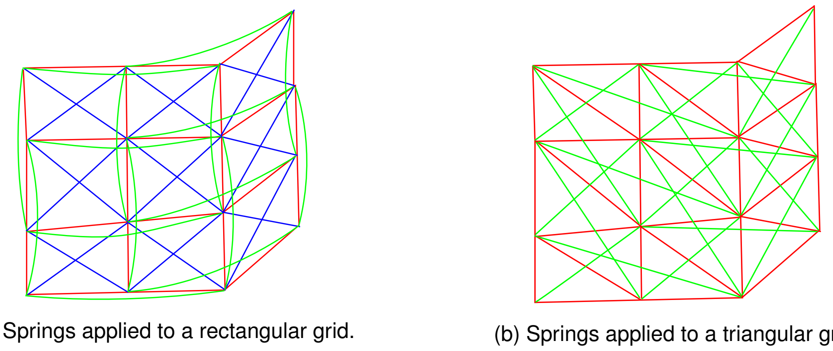
3 Mass Spring System implementation on cloth

The cloth model is represented by a grid of triangles with a given mass to each point and those triangles are connected by a series of springs.

There are three types of spring which are needed to get the characteristics of cloth:

- Structural springs: Handle extension and compression and are connected vertically and horizontally.
- Shear springs: Handle shear stresses and are connected diagonally.
- Bend springs: Handle bending stresses and are connected vertically and horizontally to every other point.

In our cloth simulation, only structural springs and bend springs are used. We use structural springs for all triangles and bend springs for all the springs connected the neighboring triangles. As the two neighboring triangles can always maintain rectangular shapes, there is no need to use the Shear springs on it. See figure 3 for an example.


(a) Springs applied to a rectangular grid.

(b) Springs applied to a triangular grid.

Figure 3: Example of a springs set up for a rectangular and a triangular grid. The structural springs are drawn in red color, bend springs are green colored. The blue shear springs are only applied to a rectangular grid.

When the simulation is initiated, the rest length of each spring is set to the original length of the springs. The value of the masses and the stiffness of the springs are also set.

Once the masses and springs are all set up, and an environmental force, for example gravity, is applied to the points in the model over a specified time step, it produces a resulting acceleration for each point. This acceleration gives rise to a velocity which causes the point to update its position. The new position of each point in turn causes a change in length to each connected spring. The combination of the point and spring forces is integrated with respect to time to provide a new acceleration for each point. Calculating forces and updating positions iteratively provides the motion of the cloth object.

3.1 Symplectic Euler method for mass-spring simulation

To simulate the mass-spring network, a Symplectic Euler solver is implemented. Its benefit is a low computation effort in comparison to the other methods used during the pro-seminar. Unfortunately, the drawback is a less accurate result for each simulation step, which leads to integrated numeric errors. In our project accuracy is not important and so, they may be ignored as long the simulation stays stable.

3.1.1 Algorithm

For each time-step h do the following steps:

For all mass-points compute position $\vec{x}$ at time $t + h$:

$$\vec{x}(t+h) = \vec{x}(t) + h \cdot \vec{v}(t) \tag{1}$$

Compute forces $\vec{f}$ for all springs with end-points $\vec{x}_1$ and $\vec{x}_2$ at time t considering the already updated values from $\vec{x}(t+h)$:

$$\vec{f}(t) = \vec{f}_{int}(t) + \vec{f}_{ext}(t) \text{ with} \tag{2}$$

$$\vec{\Delta x}(t) = \vec{x}_1(t+h) - \vec{x}_2(t+h), \tag{3}$$

$$\vec{f}_{int}(t) = \frac{\vec{\Delta x}(t)}{||\vec{\Delta x}(t)||} \cdot k \cdot \left(l - ||\vec{\Delta x}(t)||\right) \text{ and} \tag{4}$$

$$\vec{f}_{ext}(t) = m \cdot \vec{g} + \vec{F}_P + \vec{f}_{interact} \tag{5}$$

The parameters k and l are the spring constants and rest lengths of the springs respectively. A ratio of $1:8$ for the spring constant k between the bending- and structural springs gave reasonable results. Interactive forces $\vec{f}_{interact}$ may be generated by any input devices like keyboard or mouse. Actually no interactions are implemented.

Compute acceleration $\vec{a}$ for each point with mass m at time t:

$$\vec{a}(t) = \frac{1}{m} \cdot (\vec{f}(t) - \gamma \cdot \vec{v}(t)) \tag{6}$$

The damping parameter γ is used to simulate friction and air pressure. Finally, update the velocity $\vec{v}$ of each point at time $t+h$:

$$\vec{v}(t+h) = \vec{v}(t) + h \cdot \vec{a}(t) \tag{7}$$

4 Collision

For the project we used collision detection between the cloth object represented as mass spring system and rigid body objects represented as triangle mesh.

4.1 Collision detection

During collision detection every mass point of the cloth is compared with every triangle of the rigid object. If the point lies within an ε-area of the triangle the collision functions returns true, otherwise false. The comparison between a single Point P and a triangle with points A, B and C is explained below.

At first the triangle normal n and the distance d of the Point P to the triangle plane is computed:

$$n = (B-A) \times (C-A) \tag{8}$$

$$d = n \cdot P - n \cdot A \tag{9}$$

If the absolute value of the distance d is greater than ε the point does not collide with the triangle. Otherwise, further checks are necessary. The point P is projected on the plane of the triangle with points A, B and C and normal n.

$$P' = P - (n \cdot (P-A)/(n \cdot n))n \tag{10}$$

The projected point P' can then be expressed in barycentric coordinates u,v:

$$P' = A + u(B-A) + v(C-A) = (1-u-v)A + uB + vC \tag{11}$$

The point P' lies in the triangle area if $u \geq 0$ and $v \geq 0$ and $u+v \leq 1$.

4.2 Collision response

The implemented collision response method uses spring penalty forces for the mass points of the cloth object. The collision detection function returns the distance d of the point P to the colliding triangle T. The normal n of the colliding triangle is returned as well. The distance d can be negative or positive depending on which side of the triangle plane the point is located. In both cases the penalty force F_P is computed as

$$F_P = k_{rep} \cdot (-1) \cdot (d - \varepsilon) \cdot n \tag{12}$$

where k_{rep} is the repulsive spring constant used for the penalty force and ε is the distance from the triangle in which a collision can occur.