UNIVERSITY OF TEESSIDE

Procedural Animation

Maya Cloth Simulation plug-in

Paul Demeulenaere
15/05/2008

Contents

Introduction

Maya, originally developed by Alias Systems Corporation, is Autodesk software. It is a 3D modeling software such as 3D studio Max. There are two ways to develop Maya plug-in, the first is the MEL (for Maya Embedded Language) which is a scripting language, the second is given by the C++ Maya API (Application Programming Interface) which permit to give compiled code in a kind of dynamic link library name, a "mll" file. In this project, the most of computation are done thanks to Maya API and the user interface is created by the MEL language.

The goal of this project is the implementation of a simple cloth simulation integrated in the Maya Software. There are two main methods to simulate a cloth. The first one is really simple; we define a particle system where each particle is linked to their neighbours by springs. The second one, is more complex, but the effect is faster and more realistic, it is difficult to explain in some words, this method is describe in "Large Steps in Cloth Simulation" from David Baraff and Andrew Witkin and use an implicit integration method. This last one could be really interesting, but, due to a leak of time, it was the first which has been chosen.

This report will be divided in a few parts. First, the mathematical concept and an idea of the algorithms implemented will be described in the analysis part. Then, there will be some words about the design and organisation of the application. To finish, the implementation and the application resulted will be presented.

Chapter 1 – Analysis

This section will explain the mathematical concept to simulate a cloth thanks to spring links, and how it is possible to detect the collision in this representation.

I. Cloth representation

1. General Idea

There is a lot of way to represent the simulation of a cloth. One of them is done thanks to spring connected particles. The cloth is represented with a particle field, each particle are connected to their neighbours by spring. It is possible to give only four links per particle but it is more interesting to have more connections for a realistic effect even if, with this technique, the cloth become very slow and tensile.

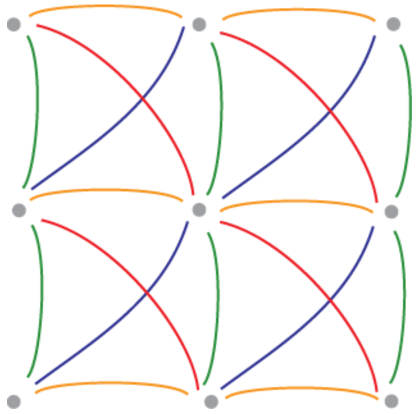


Figure 1 : Springs links

2. Hooke's law

The Hooke's law describes forces given by a spring when there is a compression. It is name and the end of the 17th century by Robert Hooke. This law can be sum up by "ut tensio sic vis", or in English, "as the extension, so the force". In this cloth simulation, the force of a spring will be described by this equation.

$$F_{P_1 \to P_2} = \frac{L_{current} - L_{normal}}{L_{normal}} \times \frac{P_1 - P_2}{\|P_1 - P_2\|} \times \Delta t$$

L current represent the current length between the particular P1 and P2, L normal, the normal length. Δt is more a force scale than a value depending of the time.

3. Euler Integration

The Euler method is the simplest method to solve a basic integration. It generate an approximation of the movement of the particles displace with spring. A very small time step is defined. The forces insides the cloth are computed and particles displaced for each iteration. This method is easy but also very slow because, to be correct, the time step should be very small. Furthermore, it is not stable along the time, if there is an error of approximation, the next iteration can keep this error and, later, the result could be really farther of an acceptable result.

Even if the Euler Integration is not perfect, it is easy to implement, for the cloth, the equation described previously thanks to the Hooke's law will be used.

II. Obstacles

The cloth is represented by a field of particles. It there isn't any interaction of the cloth with the environment, the one force which is uniformly applied to the particles is the gravity. Furthermore, there isn't any consideration of the viscosity of air then, without interaction with obstacles, the cloth falls like a metal plane.

1. Collision detection

For each particle, there is a position and a velocity vector. The position follows the velocity vector. The ray tracing become very useful in this situation. A ray is throwing from the position of the particle in the direction of the velocity. If there is a collision detected, the velocity vector is corrected adding the reaction force.

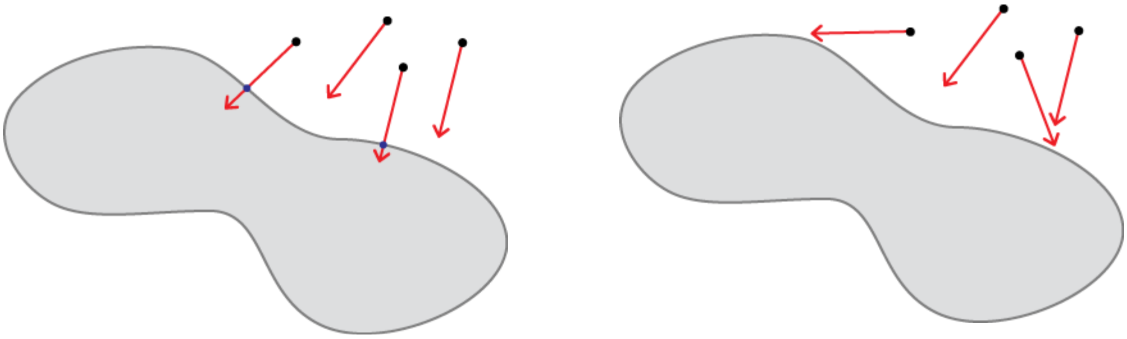


Figure 2: Collision detection

2. Reaction forces

To compute the reaction force applied on a particle when there is a collision, the third Newton law is applied: “for every action, there is an equal and opposite reaction”.

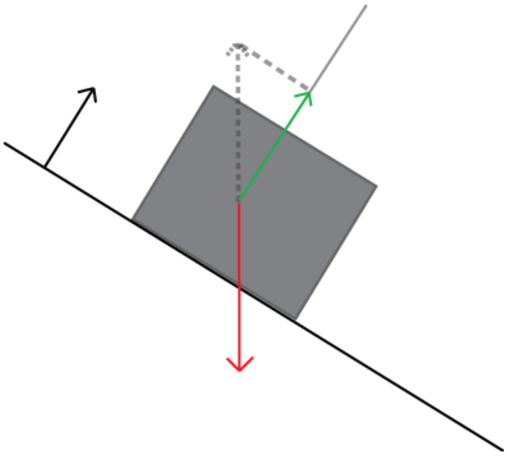


Figure 3: Geometrical representation of the reaction force (in red, the original force)

The previous figure presents the reaction force in green. This force depends of the normal of the surface collided. This reaction force is added to the velocity and multiplied by a value to simulate the friction.

The collision is only tested on the particles and not on the edge of the cloths; furthermore, the cloth is tensile with this resolution, so, it is more realistic to keep the velocity at null when the particle is collided because the cloth could stretch until to thread the obstacle between two particles.

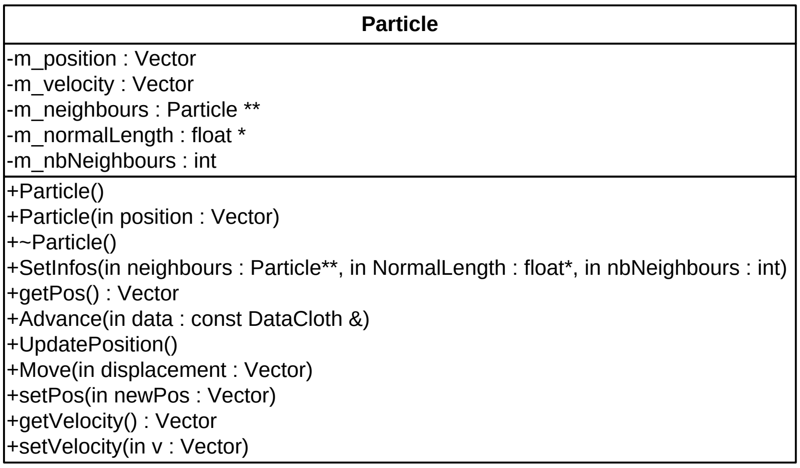
Chapter 2 – Design

The application has to have a correct design to ease the development; the cloth particle system is a black box for Maya plug-in classes. First, I will present the cloth classes, then, an overview about Maya classes.

I. Cloth

1. Particles

The cloth is represented thanks to particles, each particles has a position and a fixed number of neighbours. The function "Advance" applies by the Euler method the Hooke's law to store the new velocity in "m_velocity", the function "UpdatePosition" displace the particle following the velocity vector.

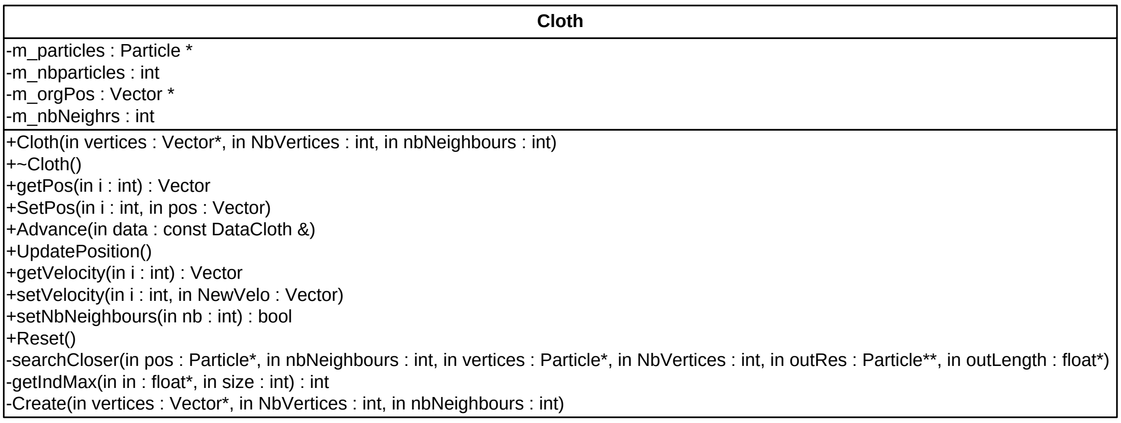


```
Particle
-m_position : Vector
-m_velocity : Vector
-m_neighbours : Particle **
-m_normalLength : float *
-m_nbNeighbours : int
+Particle()
+Particle(in position : Vector)
+~Particle()
+SetInfos(in neighbours : Particle**, in NormalLength : float*, in nbNeighbours : int)
+getPos() : Vector
+Advance(in data : const DataCloth &)
+UpdatePosition()
+Move(in displacement : Vector)
+setPos(in newPos : Vector)
+getVelocity() : Vector
+setVelocity(in v : Vector)
```

Figure 4: The particle class

2. Cloth

The cloth class contain an array of particles. This movement of the cloth is done by the call of the “Advance” function. To create a cloth, we just need to send an array of vector, the cloth class create link between the particles.



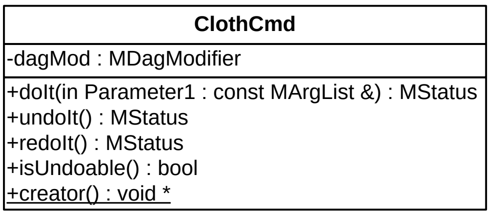
```
Cloth
-m_particles : Particle *
-m_nbparticles : int
-m_orgPos : Vector *
-m_nbNeighrs : int
+Cloth(in vertices : Vector*, in NbVertices : int, in nbNeighbours : int)
+~Cloth()
+getPos(in i : int) : Vector
+SetPos(in i : int, in pos : Vector)
+Advance(in data : const DataCloth &)
+UpdatePosition()
+getVelocity(in i : int) : Vector
+setVelocity(in i : int, in NewVelo : Vector)
+setNbNeighbours(in nb : int) : bool
+Reset()
-searchCloser(in pos : Particle*, in nbNeighbours : int, in vertices : Particle*, in NbVertices : int, in outRes : Particle**, in outLength : float*)
-getIndMax(in in : float*, in size : int) : int
-Create(in vertices : Vector*, in NbVertices : int, in nbNeighbours : int)
```

Figure 5: The cloth class

II. Maya

1. Command

The Maya command class is a child of MPxCommand. It became a component of Maya. When the command cloth is called, the selected object is linked to a new cloth node.

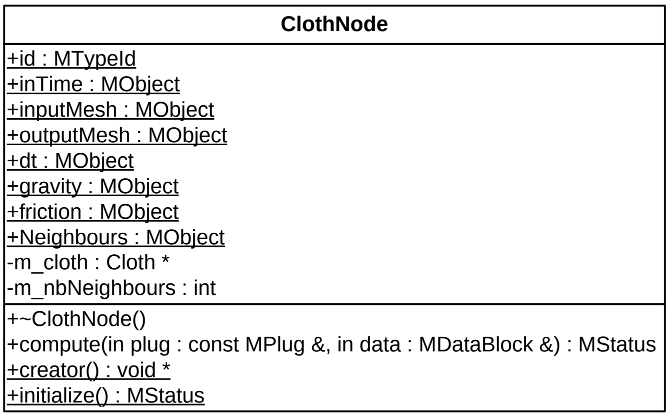


```
ClothCmd
-dagMod : MDagModifier
+doIt(in Parameter1 : const MArgList &) : MStatus
+undoIt() : MStatus
+redoIt() : MStatus
+isUndoable() : bool
+creator() : void *
```

Figure 6: Cloth command class

2. Node

Like the command, the node is a child of a particular Maya class, “MPxNode”. This node take a mesh in parameter to create a cloth, for each modification of time, the advance function is called and the cloth animated. It is also in this component which computes collision between the cloth and obstacles.



ClothNode

+id : MTypeId
+inTime : MObject
+inputMesh : MObject
+outputMesh : MObject
+dt : MObject
+gravity : MObject
+friction : MObject
+Neighbours : MObject
-m_cloth : Cloth *
-m_nbNeighbours : int

+~ClothNode()
+compute(in plug : const MPlug &, in data : MDataBlock &) : MStatus
+creator() : void *
+initialize() : MStatus

Figure 7: Node class

Chapter 3 – Implementing and result

I. The OpenGL application

To ease the development and make the cloth classes more portable, the cloth hasn't been implemented directly as a Maya plug-in. The first approach was to create a simple application to test the algorithms. This application is on a QT environment with OpenGL and creates a mesh of a hundred vertices to debug the cloth.

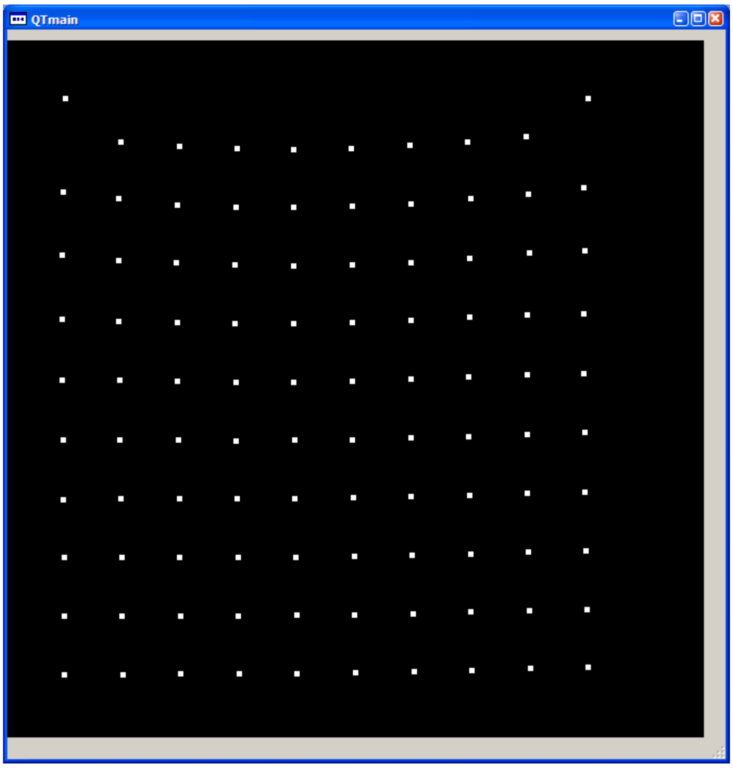
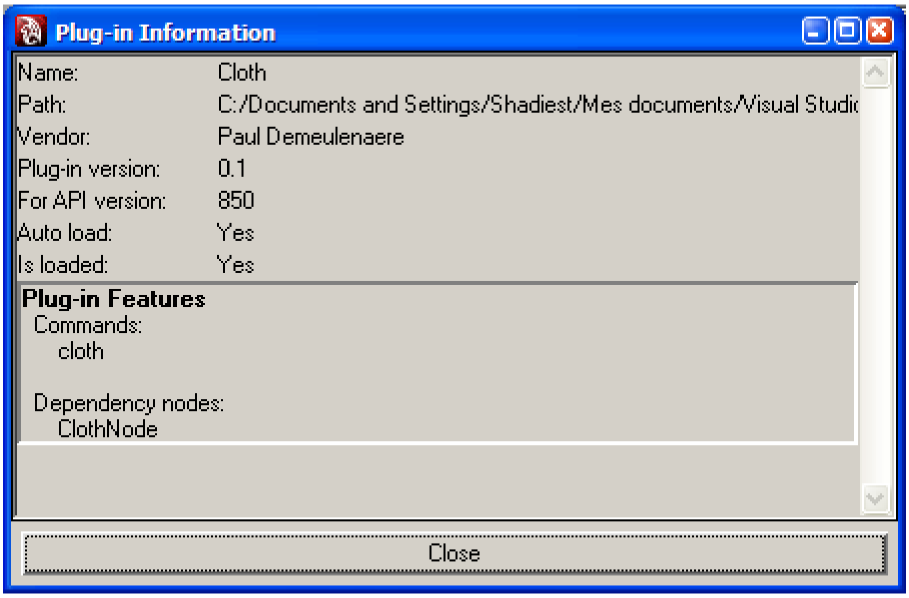


Figure 8: Overview of the first application

This technique is very useful because it is not easy to use possibilities of debug tools when a Maya plug-in is compiled. When cloth classes are enough tested, it is possible to begin the project for integration in Maya.

II. Maya plug-in

The Maya plug-in is composed of two elements, a command called “cloth” and a node called “ClothNode”.



Plug-in Information

Name:	Cloth
Path:	C:/Documents and Settings/Shadiest/Mes documents/Visual Studi
Vendor:	Paul Demeulenaere
Plug-in version:	0.1
For API version:	850
Auto load:	Yes
Is loaded:	Yes

Plug-in Features
Commands:
cloth

Dependency nodes:
ClothNode

Figure 9: Plug-in informations

When the cloth command is called with a shape selected, this last became a cloth, and the animation is played on each new key frame and reset at the frame 0. The obstacles are not linked to the node, every object which has been called “obs*”are considered as obstacle.

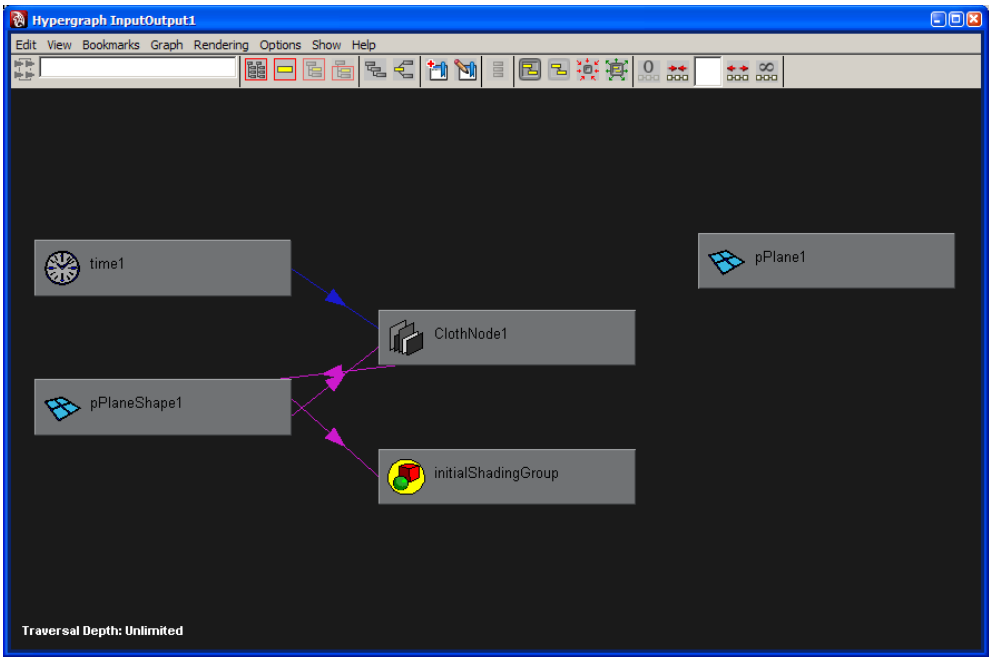
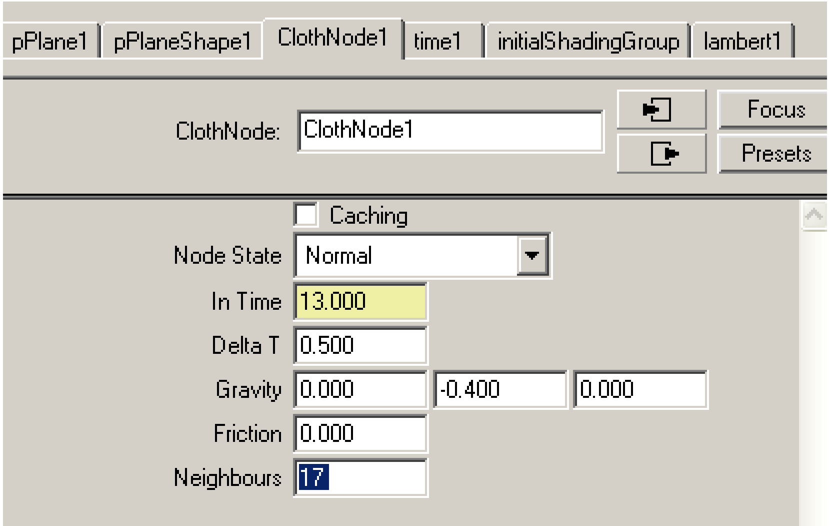

Hypergraph InputOutput1
Edit View Bookmarks Graph Rendering Options Show Help
time1
pPlane1
ClothNode1
pPlaneShape1
initialShadingGroup
Traversal Depth: Unlimited

Figure 10: Overview of the hypergraph of the cloth

This node has several settable parameters. “Delta T” represents the value which scales the force vector of springs. “Gravity” is a vector which defines a force applied uniformly on every particle. When the friction term is equal to 0, it does not mean that there is not any friction, the friction is infinite and a particle which collide an obstacle “stick” to it. Finally, it is possible to change the numbers of neighbours for each particle but this modification will reset the cloth because it is necessary to reconstruct all links between particles. The cloth is more realistic with a lot of neighbours but it became also really slower.



ClothNode: ClothNode1

□ Caching

Node State: Normal

In Time: 13.000

Delta T: 0.500

Gravity: 0.000 -0.400 0.000

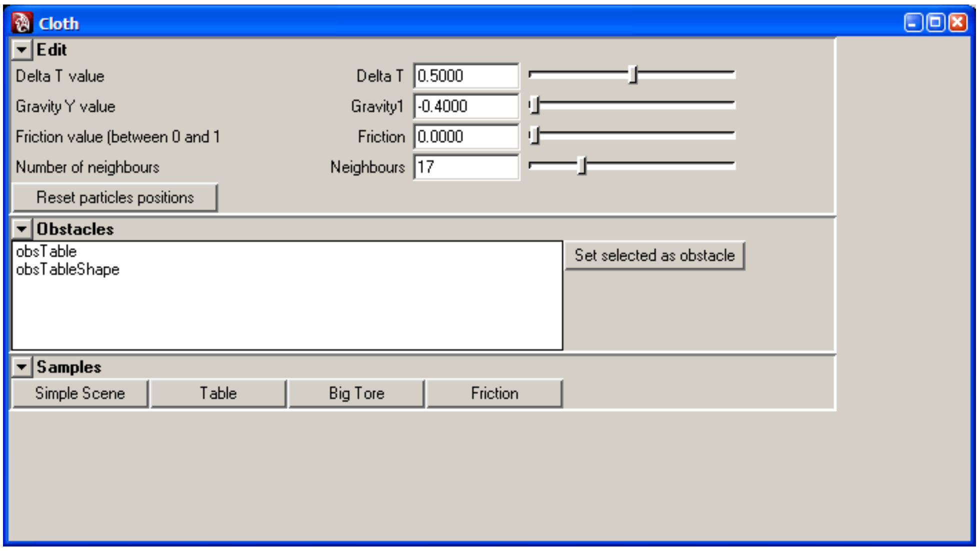
Friction: 0.000

Neighbours: 17

Figure 11: Parameters of the cloth node

III. MEL

A user interface has been created to ease to use of the nodes. The obstacles simply are a mesh with the name which begins by “obs”. The MEL interface indicates the list of elements which are obstacles. It is possible to convert a selected object as an obstacle; in fact, this option will rename the shape.



Cloth

Edit

Delta T value — Delta T 0.5000

Gravity Y value — Gravity1 -0.4000

Friction value (between 0 and 1 — Friction 0.0000

Number of neighbours — Neighbours 17

Obstacles

obsTable
obsTableShape

Samples

Figure 12: User interface created thanks to a MEL script

This user interface is a shortcut to the editable values of the cloth node; the current frame became 0 when the user decided to reset the cloth. It is possible to load some sample scenes but these actions will delete the entire object on the current scene.

Conclusion

There are still some bugs not resolved in the final application. There isn't enough check for errors but the plug-in is functional and can render interesting scenes. Like it is expected, the cloth is tensile and slow, but the effect is nice.

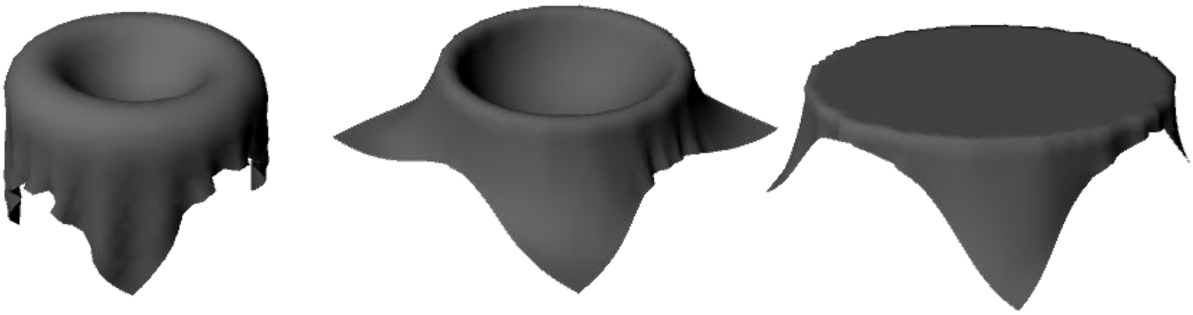


Figure 13: Some render samples

It could be interesting to implement the large step simulation, but it was an ambitious objective, the Maya API and the cloth simulation were new for me. If I have to the same kind of project later, I will choose the large step simulation because it is more stable, realistic and faster.

This project was very interesting; I hope I will have again the possibility to work for this kind of project. I have learnt a lot about the development of plug-in for software, the error checking should be more integrated but it was my first attempt of this kind of programming. My skills in procedural animation are really improved; the cloth simulation is a good example of possibilities of a simple mathematical concept applied in computer sciences to generate impressive visual effects.

Figure list