Soft Body Simulation

TNCG13 - SFX Tricks of the Trade

EMMA FORSLING PARBORG MARTIN GRÅD ELLEN HÄGER

Linköping University
December 17, 2015

1 Introduction

This is a project in the course TNCG13 - SFX Tricks of the Trade at Linköping University. The aim of the project is to create a soft body simulation plug-in for Maya. The method is largely based on the article *Pressure Model of Soft Body Simulation* by Matyka & Ollila [Matyka & Ollila 2003]. Another goal is to learn how to use the Maya interface and work with the API for creating plug-ins.

2 Method

The plug-in employs a mass-spring-damper system and a gas pressure model to make meshes retain their shape [Matyka & Ollila 2003]. The gas pressure model utilizes ideal gas approximation, and the entire system is simulated using forward Euler integration. In the following sections, these components will be described followed by a brief remark on the limitations imposed on the simulation.

2.1 Mass-Spring-Damper system

A simple mass-spring-damper system is used to generate the soft body dynamics. The system is a particle system that takes as input a mesh created within Maya, and is constructed using the properties of such a mesh.

When handling collisions for the vertices, an impulse force is calculated using Newton's second law, and is applied to the force acting on them. Newton's second law and the impulse equation are displayed in Equation 1 and Equation 2, respectively.

$$\vec{F} = m * \vec{a} \tag{1}$$

$$\vec{J} = \vec{F} * \Delta\vec{v} \tag{2}$$

Here, $\vec{F}$ represents the force, m the mass, $\vec{a}$ the acceleration, $\vec{J}$ the impulse, and $\Delta\vec{v}$ the is the change in velocity.

The mass-spring-damper system utilizes Hooke's law to determine the spring force, and also applies a damping force:

$$F_s = -k * x \tag{3}$$

$$F_d = c * \frac{dx}{dt} \tag{4}$$

$$F = F_s + F_d \tag{5}$$

The spring force F_s is calculated by multiplying the spring constant k with the elongation of the spring x. The damping force F_b is calculated by multiplying the damping constant b with the change in elongation over time.

2.2 Pressure method & Ideal gas approximation

A mass-spring-damper system constructed from only the vertices and edges of a hollow mesh will collapse on itself. To remedy this, the system needs an additional internal force of some kind. This is where the pressure model is introduced. Modeling internal gas pressure will keep a mesh inflated, and thus also from collapsing. The internal pressure force is applied in the direction of the normals of the object and is approximated by

$$\vec{P} = P * \hat{n}, \tag{6}$$

in which $\vec{P}$ is the pressure vector for a point, P is the pressure value and $\hat{n}$ is the normal vector for a face. The pressure vector can then be used in Equation 7 to calculate the pressure force $\vec{F}_p$.

$$\vec{F}_p = \vec{P} * A \tag{7}$$

The pressure force is calculated for each face in the mesh. The area of a face, A, is calculated using Equation 8

$$A = \frac{|\vec{v}_1 \times \vec{v}_2|}{2} \tag{8}$$

Here, v_1 and v_2 are two edges in the triangle.

The pressure value P, from Equation 6, is calculated using the Clausius Clapeyron equation, see Equation 9.

$$P = V^{-1} * n * R * T \tag{9}$$

Here, n is the gas mol number, R is the ideal gas constant and T is the temperature. V is the volume of the body, and for a triangle mesh it can be calculated by using equation 10.

$$V = \Big| \sum_{j}^{i=0} \frac{\vec{v}_0(i) \bullet (\vec{v}_1(i) \times \vec{v}_2(i))}{6} \Big| \tag{10}$$

Here, $\vec{v}_0(i)$, $\vec{v}_1(i)$ and $\vec{v}_2(i)$ are the vectors starting from the origin of the mesh and pointing to the vertices of the triangle face.

2.3 Numerical method

The numerical method used in this project is an explicit forward Euler integration, see Equation 11, which is a first order method that is conditionally stable [Hoffman & Frankel 2001].

$$f_{n+1} = f_n + h * f'_n \tag{11}$$

Here, the next value f_{n+1} is calculated from the current value f_n, the step length h and the value of the derivative f'_n. This method is used for both calculating the new positions and velocities.

2.4 Implementation

The implementation of the different techniques can be summarized in the following steps:

1. Loop over all vertices of the mesh
   - Add external forces such as gravity and collisions.
2. Loop through all the edges
   - Add the internal forces created by the springs and dampers to the vertices.
3. Loop through all faces. Calculate and add the pressure force acting on each face by:
   - Calculating the volume of the soft body
   - Calculating the ideal gas approximation
   - Calculating the area of the triangle
4. Update the velocities for the vertices with explicit Euler integration
5. Update the positions of the vertices with explicit Euler integration

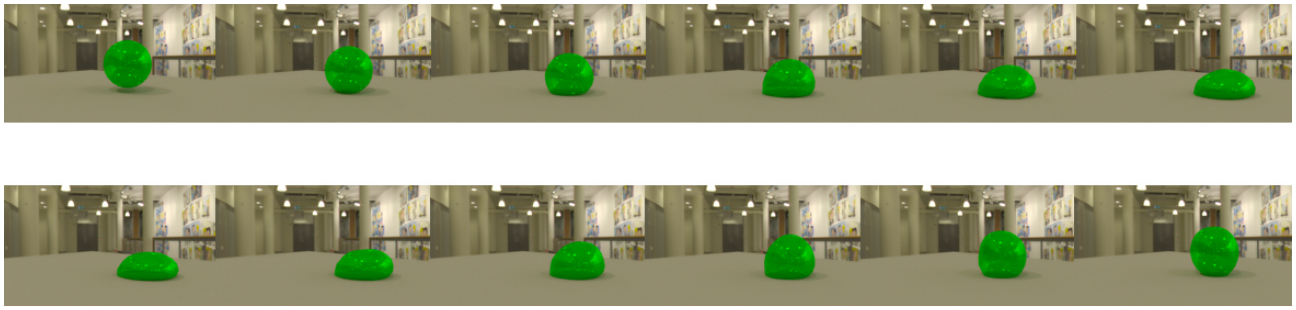

Figure 1: *Illustration of how the sphere moves when a force is applied from the start. From Left to right, top to bottom: the sphere lands on the floor and then jumps back in the same direction it came from.*

These steps are implemented in a Maya plug-in written in C++. A MEL script is used to create a default scene, consisting of a `polyPlatonicSolid` in the shape of a sphere falling down. In the final renderings, image-based lighting has been applied in Maya, by utilizing the mental ray renderer.

2.5 Limitations

One limitation for this project concerns the collision handling. The implementation explicitly defines the floor at the position $y = 0$ and if a vertex collides with it, a new force is calculated by adding the impulse force previously presented in Equation 2.

3 Results

The implemented Maya plug-in consists of two classes, `SoftBodyDeformer` and `ParticleSystem`. The first class extends `MPxDeformerNode`, and is used to extract the vertices, faces and edges of a mesh. The second class contains all functionality pertaining to the actual simulation. This is where meshes at every fram are converted, simulated and used to update the original mesh.

Figure 1 shows results for a sphere when an additional, external force is applied.

4 Discussion

The plug-in can handle triangulated MFnMeshes, although the same algorithm could be used for quadratic faces as well. The plug-in can handle one mesh at a time. When the plug-in is activated for a mesh, depending on which shape is loaded, the system may appear unstable sometimes when starting it. This is due to that the initial values for the spring constant, damping constant, gas pressure, mass and initial velocity are not optimized by default. The user can however manipulate these values and get a stable system. An improvement for this could be that instead of having the user find what parameters are appropriate for the mesh, they could be correctly computed and initialized by the program.

The different gas properties were combined into a single value. This was done to simplify the usage of the plug-in.

The collision handling for the plug-in is very basic and it does not handle self collision. Due to the limited amount of time for this project, this was not prioritised and could be improved in future work.

For future work a more stable numerical method could be implemented, instead of the Euler method, in order to minimize the error.

Another aspect of this implementation that requires work is achieving rotation of entire meshes. When an object with a horizontal velocity component hits the ground, it does not continue to move in this direction, and instead it bounces back, toward its initial position. This seems to happen since no rotation is generated around the center of mass of the mesh. This could be due to the simple collision handling. The fact that the impulse force generated from collisions with the ground is a simplified solution to this problem might cause these errors.

4.1 Technical and Theoretical challenges

The biggest challenge was to learn how to utilize the Maya API and use the correct functions. It was especially tricky to find the correct vertices and apply the appropriate forces to make the shape behave correctly. Due to this, different approaches were made in trying to create a foundation for the mass-spring-damper system. One approach was to take advantage of the built-in `nCloth` functionality in Maya, and then extend this to work with the pressure method. However, this proved to be difficult, and was ultimately discarded in favor of implementing a custom-made particle system.

The other approach was to explore the functionalities of the class `MFnMesh`, which came to be the final and implemented approach, since there are iterator classes that provide specific control over the vertices, edges and polygons of a mesh [Autodesk Inc. 2015]. Using these tools made it easier to implement the algorithm.

5 Conclusions

For this project, a soft body simulation plug-in has been created for Maya. By completing this project, the group has learned how to create and use plug-ins and scripts for Maya. Implementing the soft body simulation method has also been a learning experience.

References

HOFFMAN, J.D. & FRANKEL, S. (2001). Numerical Methods for Engineers and Scientists, Second Edition *CRC Press*, New York, USA.

MATYKA, M. & OLLILA, M. (2003). Pressure Model of Soft Body Simulation, *Proc. of Sigrad*, Umeå, Sweden.

AUTODESK INC. (2015). MFnMesh Class Reference, *http://goo.gl/TA8j4Z*, Latest update: 14 October 2015.