Maya Soft Body Deformer Plugin Using Shape Matching

SFX - Tricks of the trade: Project Report

Isabell Jansson isaja187@student.liu.se
Ronja Grosz rongr946@student.liu.se
Jonathan Bosson jonbo665@student.liu.se

December 12, 2016

Abstract

This report follows the approach proposed by Müller et al.[1] to simulate soft body deformation using shape matching and examines how it can be implemented into Autodesk Maya as a deformer plugin. The shape matching method does not require storage of any data about neighbouring vertices and is unconditionally stable. Due to its efficiency in terms of memory and computation complexity it is well suited for interactive applications.

1 Introduction

Soft body deformation is a computer graphics method for simulating the deformation of a soft body. There exists many techniques for achieving soft body deformation. In this paper we present the method and results from implementing a soft body deformer plugin for the 3D computer graphics software Maya. The implementation is based on the meshless shape matching method proposed by Müller et al. [1].

2 Background

Soft deformable bodies is a field in computer graphics that focuses on the physical and visual simulation of the motion and characteristics of a soft deformable body. Soft deformable bodies are objects whose bodies can change by being deformed without losing its characteristic shape. Several different methods have been proposed to simulate the deformation of soft deformable bodies. The spring mass model is one of the simpler methods where the soft body is approximated by a set of masses linked by springs. The method is popular for simulating the deformation of cloth. Two other methods are the finite element method and the energy minimization method.

The shape matching approach proposed by Müller et al. [1] is a meshless shape matching technique for deformation of soft bodies.

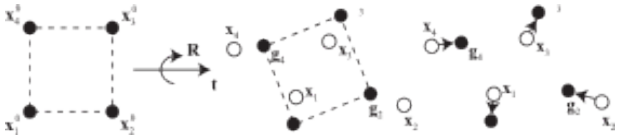


Figure 1: The shape matching process where the original shape $\mathbf{x}_i^0$ is matched to the deformed shape $\mathbf{x}_i$ and pulled towards the goal shape $\mathbf{g}_i$. Image source: [1].

It is a meshless technique since the object is only represented by a point cloud. There is no need for connectivity information between the points since the shape matching keeps the shape of the object. This is done by driving the deformed points towards the position of the points of the original shape, see figure 1.

3 Shape matching implementation

The soft body is represented by a point cloud, where each particle has a mass m_i, an initial position $\mathbf{x}_i^0$ representing the shape of the object and a deformed position $\mathbf{x}_i$. The soft body is affected by external forces such as gravity, collision impulse and friction impulse caused by a collision with an object. The external forces $f_{ext} = F_g + J_c + J_f$ is what causes the deformation of the soft body. The new velocities and positions for the particles are calculated using the external forces and Euler integration. The collision impulse J_c is calculated through Equation 1 where ϵ is the elasticity of the soft body, n is the contact normal and Δv is the velocity difference between the soft body and the static object it collides with. The friction impulse is calculated through Equation 2, where f is the friction.

$$J_c = -(1+\epsilon)(n(n \cdot \Delta v))m_i \tag{1}$$

$$J_f = -f(\Delta v - n(n \cdot \Delta v)m_i \tag{2}$$

The shape matching is made by finding a goal position g_i for every particle. The goal position is used to find the new velocities and positions for the particles for each time step, driving the soft body to take its original shape at $\mathbf{x}^0$. The new velocities and positions are calculated through Euler integration where α is the bounciness and stiffness of the soft body and h is the time step, see Equation 3 and 4.

$$\mathbf{v}_i(t+h) = \mathbf{v}_i(t) + \alpha\frac{\mathbf{g}_i(t) - \mathbf{x}_i(t)}{h} + h\frac{f_{ext}(t)}{m_i} \tag{3}$$

$$\mathbf{x}_i(t+h) = \mathbf{x}_i(t) + h\mathbf{v}_i(t+h) \tag{4}$$

The goal position is found through the rotation matrix $\mathbf{R}$, the initial position $\mathbf{x}_i^0$, the center of mass for the initial position $\mathbf{x}_{cm}^0$ and the center of mass for the current deformed shape $\mathbf{x}_{cm}$, see Equation 5.

$$\mathbf{g}_i = \mathbf{R}(\mathbf{x}_i^0 - \mathbf{x}_{cm}^0) + \mathbf{x}_{cm} \tag{5}$$

The rotation matrix is approximated through the linear transformation $\mathbf{A}$, see Equation 6. The linear transformation is composed of the relative locations $\mathbf{q}_i = \mathbf{x}_i^0 - \mathbf{x}_{cm}^0$ and $\mathbf{p}_i = \mathbf{x}_i - \mathbf{x}_{cm}$, see Equation 7 and 8. The rotation $\mathbf{R}$ is the rotational part of $\mathbf{A}_{pq}$

which is found through singular value decomposition.

$$\mathbf{A} = (\sum_i m_i \mathbf{p}_i \mathbf{q}_i^{\mathbf{T}})(\sum_i m_i \mathbf{q}_i \mathbf{q}_i^{\mathbf{T}})^{-1} = \mathbf{A}_{pq}\mathbf{A}_{qq} \tag{6}$$

$$\mathbf{x}_{cm}^0 = \frac{\sum_i m_i \mathbf{x}_i^0}{\sum_i m_i} \tag{7}$$

$$\mathbf{x}_{cm} = \frac{\sum_i m_i \mathbf{x}_i}{\sum_i m_i} \tag{8}$$

Rigid deformation

For a rigid deformation α is set to one, moving the points to the goal position $\mathbf{g}_i$ exactly for each time step. The goal position represents a rotated and translated version of the initial shape.

Linear deformation

For linear deformation, the linear transformation $\mathbf{A}$ in combination with the rotation matrix $\mathbf{R}$ according to Equation 9 is used instead of the rotation matrix to calculate the goal position, see Equation 5. The linear deformation $\mathbf{A}$ is divided by $\sqrt[3]{det(\mathbf{A})}$ to keep the volume conserved.

$$\beta\mathbf{A} + (1-\beta)\mathbf{R} \tag{9}$$

Quadratic deformation

To extend the motion to twisting and bending a quadratic transformation is applied. $\mathbf{q}$ is extended to $\tilde{\mathbf{q}} = [\mathbf{q}_x, \mathbf{q}_y, \mathbf{q}_z, \mathbf{q}_x^2, \mathbf{q}_y^2, \mathbf{q}_z^2, \mathbf{q}_x\mathbf{q}_y, \mathbf{q}_y\mathbf{q}_z, \mathbf{q}_z\mathbf{q}_x]^{\mathbf{T}}$. With the new $\tilde{\mathbf{q}}$, a new linear transformation $\tilde{\mathbf{A}}$ can be calculated the same way as Equation 6. To keep the volume, $\tilde{\mathbf{A}}$ is divided by $\sqrt[9]{det(\tilde{\mathbf{A}})}$. Together with the rotation matrix $\tilde{\mathbf{R}} = [\mathbf{R}\ \mathbf{0}\ \mathbf{0}]$, the goal position is calculated according to Equation 10.

$$\mathbf{g}_i = (\beta\tilde{\mathbf{A}} + (1-\beta)\tilde{\mathbf{R}})\tilde{\mathbf{q}} \tag{10}$$

4 Maya Implementation

The plugin is built on top of the deformer node, $MPxDeformerNode$, in Mayas C++ API which can control the position of all vertices of an object with the deformer loaded. By copying the vertice positions and storing them as individual particles in a custom particle system in the plugin is more control and freedom gained. This allows the use of high quality linear algebra libraries for efficient and complex calculations.

As seen in figure 2, the deformation of a shape can be controlled by changing the following parameters:

- Gravity Magnitude and Direction
- Mass, m
- Stiffness, $\alpha_s \in [0,1]$ - variable controlling how much deformation is allowed
- Bounciness, $\alpha_f \in [0,1]$ - variable controlling the strength of the overshoot
- Friction, f
- Deformation, $\beta \in [0,1]$ - variable controlling the strength of the deformation
- Elasticity, ϵ
- Mode (Rigid, Linear or Quadratic) - Controls the kind of desired deformation

- Initial Velocity

The local time variable is connected to the global time so that the deformation will update when the timeline changes. The plugin stores the value of the last time step so the simulation behaves correctly regardless of how the global time is changed. Further variables can be linked with other parameters in Maya such as the gravity magnitude and direction can take their values from a gravity field.

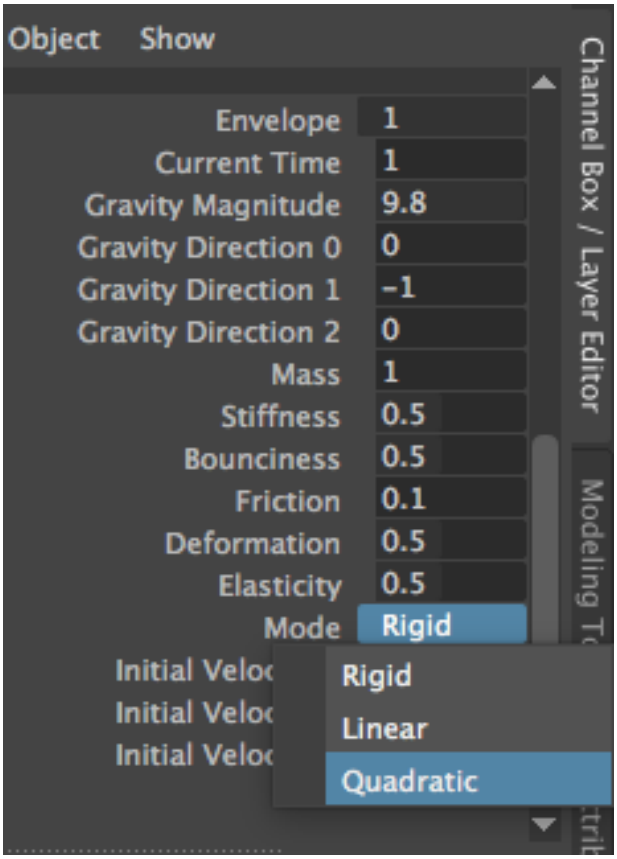


Figure 2: The user interface in Maya which is used to modify the deformation.

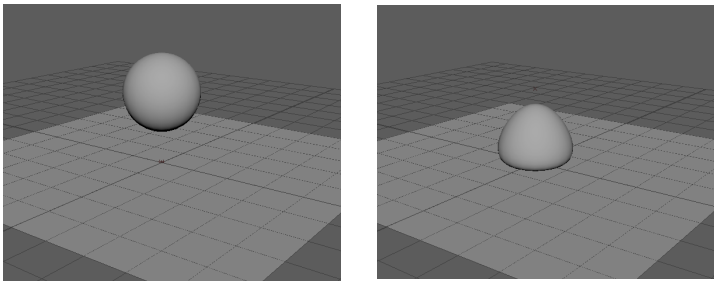
5 Conclusion

The deformer node plugin can handle rigid as well as soft body (linear and quadratic) deformations with a low computation cost. Figure 3 illustrates the deformation that happens when a sphere collides with the ground plane. After impact, in figure 3b, the mesh will transform into a egg-like shape to later settle down as the original sphere.

6 Discussion and Future Work

The shape matching method can be useful in rigid and moderately soft body deformations even though the algorithm is not physically grounded. It can model twists and bends well but struggles with stability with large deformations. If the parameters *bounciness* or $deformation$ is set too high the deformed positions cannot adequately be translated to their goal position, which causes the shape to be flattened upon impact with the ground plane. Proper parameter management is therefore required.

Another possible extension to the plugin could be to implement support of object to object collision. Current solution only handles collision with the ground plane by naively checking if the a particles y-value is below the y-value of the plane to then respectively updates the forces.



(a) (b)

Figure 3: A quadratic deformation of a sphere using the plugin.

References

[1] M. Müller, B. Heidelberger, M. Teschner, and M. Gross, "Meshless deformations based on shape matching," *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 471–478, 2005.