Hair Simulation for Real-Time Applications

Computer Generated Imagery Techniques
Bournemouth University
Zoe Sams
i7698049

November 2014

Abstract

Hair simulation, especially in real time, is one of the most challenging areas of simulation for virtual humans. The human head has roughly 100,000 hairs, each of which has a complex mechanical behaviour that is simply not feesable to simulate in real time. This project proposes to implement the mass spring model to create guide hairs which will be used to drive the simulation. Rendering will be discussed and considered, but is not the main focus of this project.

1 Introduction

Creating realistic virtual humans has often been a prominent topic within the realm of computer graphics. Visual effects, games, engineering and the fashion industries all use virtual humans on a regular basis. These virtual humans require a high level of detail to be believable, but must still move and update in an authentic fashion. Animation and visual effects have the benefit of being able to use pre-rendered animation, however games and other interactive graphics applications must ensure that all parts of the virtual human must update in real-time. This, however, does not mean the animations and visual effects industries' pipelines would not benfit from real-time systems. Within the area of virtual humans, soft body dynamics are arguably the most challenging as hair and cloth face similar problems when being simulated. This project proposes to implement a mass spring model for hair simulation which will be suitable for use within real-time graphics applications. The mass spring model is straightforward to implement. It does, however, have issues with stability when using a large time step, which can be countered using certain integration methods. With the human head containing over 100,000 hairs, it would be impracticle to simulate all of these. Therefore, a select number of hairs will be implemented and used as guide hairs for the simulation.

2 Background

There are many aspects to creating realistic human hair for graphics. Although specifically referencing cloth, Kang et al. (2012) discuss that the way in which soft bodies move must be accurate and believable, while still looking aesthetically pleasing and realistic.
Rendering hair is a huge topic area, and arguably another full project in itself. Translucensy, shadows, opacity, shading and texturing are just some of the areas which need to be considered when rendering hair. The vast scope of potential implementations of rendering means it will not be looked at in depth in this project - the hair will be polygonally rendered to be seen, but not to be visually impressive. Instead, the main focus of this project will be the simulation.

2.1 Simulation

Simulation of hair can be vastly complex, with external forces such as collision, friction, wind and even static electricity affecting how hair behaves in the real world (Parent, 2012). Emulating this exact behaviour within real-time applications is still not possible, but technology is improving. Magnenat-Thalmann et al. (2008) discuss how hair simulation can be constructed in one of two ways - strand (or particle) based and volume based.

2.1.1 Volumetric Representation

One method of volumetric representation is the use of keyhairs to drive the main simulation, where the remainder of the hairs are interpolated from these keyhairs.
Petrovic et al. (2014) discuss representing these keyhairs as B-splines, and summing up the spatial influences of the keyhair control vertices to create a volume. Using a Cartesian voxel representation to simulate the inter hair interactions, they then present the equation

$$V_{xyz} = \frac{\sum_i (1 - |P_x^i - x|)(1 - |P_y^i - y|)(1 - |P_z^i - z|)v^i}{D_{xyz}}$$

to calculate the velocity at each vertex (x, y, z), with (P_x^i, P_y^i, P_z^i) being the position of the ith particle in world space and v^i the velocity. The method presented by Petrovic et al. is inexpensive, but will not be used in this project. Strand based techniques should be investigated as an alternative.

2.1.2 Follow The Leader

By first creating a chain of particles, from x_1 to x_n positions, with the distances in between each being the same, we can move the chain of particles using the Follow The Leader (FTL) algoritm. Müller et. al (2012) discuss how particle 2 in the chain should be considered to be upon the edge of a sphere, with the sphere's centre point being x_1, and the radius being the rest length in between the two. Particle 2 then moves in the direction of particle 1, much like a chain of rigid body particles constrained to the it's neighbours.

This method is, however, challenging to integrate within a dynamic simulation, and is unpopular in the field because of this. Müller (2012) states that as follow the leader is a geometric based technique, it is not as accurate as physically based techniques such as the mass spring model.

2.1.3 Mass Spring Model

The mass spring model is a method of simulating soft bodies often used within hair and cloth simulation. Among the first to propose a model for simulating indicidual strands were Rosenblum et al. (1991), suggesting that each strand can be modelled using particles connected by springs. A basic implementation of this method can treate vertices as masses and edges as springs (Parent, 2012). For hair simulations, the stretch parameter of the spring must be appropriately constrained, otherwise the hair simulation will appear unrealistic. Hällman (2011) discusses that stiff springs, however, mean the simulation can become unstable, with springs reacting unexpectedly.

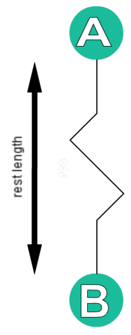

Figure 1: Basic spring between particle A and particle B - both with mass m

Tariq (2010) states that one of the benefits of the mass spring model is it's ability to be used in conjuntion with an implicit integration scheme. This makes for a more stable simulation, which will be discussed in section 3.2. Although the mass spring model is computationally inexpensive, alone it cannot simulate twist in hairs. Selle et al. (2008) suggest the addition of tetrahedral altitiude springs to the mass spring model. This can be used to stop the simulation colapsing in on itself to zero volume making for a more stable twist simulation. This technique is rather complex, and will therefore not be implemented within this project.

3 Method

3.1 Mass Spring Model

To implement the mass spring model, first the vertices - also referred to as nodes - must be given a mass. The mass depends on what kind of simulation is being implemented, however for believable hair the mass of each particle should be the same. For instance a strand could be given a mass and the mass could be in turn evenly distributed throughout the nodes in that strand. Changing the distribution within that mass could be used for other purposes, such as simulating wet hair which will be discussed in Section 4 - Possible Additions. A spring connects these two nodes, and the internal force F_{intern} of the spring must then be calculated using

$$F_{intern} = k_{spring} \left(|B - A| - d\right) \left(\frac{B - A}{|B - A|}\right) - k_{damper}(B(t) - A(t))$$

where k_{spring} is the spring constant, k_{damper} is the damper constant, d is the rest length between node A and node B, A and $A(t)$ are the postion and velocity of the first node respectively, much like B and $B(t)$ are the position and velocity of the second. This equation is based off of the equations presented by Lengyel (2012). First, the spring force must be calculated using Hooke's Law - measuring change in the springs state from rest length. If the change between the nodes is positive, the distance is greater than the rest length meaning the force applied will be negative in order to pull the nodes back together. If the change in nodes is negative, however, the neighbouring node must have a positive force applied to it in order to push it back towards it's inertial length, or rest length.
The external force of each node should then be calculated using Newton's Laws of Motion. The sum of all forces acting upon the particle should be

$$F = ma$$

according to Newton's Second Law of Motion, with m being the mass of the particle and a being the acceleration. This total force is the summation of the internal and external forces, with external forces being those such as gravity, air resistance and wind. To compute the dynamics of the system, an integration method should be used.

3.2 Integration

The Verlet method is a commonly used integration method within mass spring systems. Müller et al. (2008) discuss how the internal and external forces calculated are applied to each particle. To calculate the dynamic simulation itself, the Verlet method of integration can be applied to each system in the following steps.

$$\dot{\mathbf{x}}(t) = \frac{[\mathbf{x}(t) - \mathbf{x}(t - \Delta t)]}{\Delta t}$$

$$\ddot{\mathbf{x}}(t) = \frac{F(\mathbf{x}(t), \dot{\mathbf{x}}(t))}{m}$$
$$\mathbf{x}(t + deltat) = 2\mathbf{x}(t) - \mathbf{x}(t - \Delta t) + (\ddot{\mathbf{x}}(t)\Delta t^2)$$

In these equations, $F(t)$ is the total force applied to the particle, m is the mass, $\ddot{\mathbf{x}}(t)$ is the acceleration, $\dot{\mathbf{x}}(t)$ the velocity, $\mathbf{x}(t)$ the current position, and Δt the time step. The Verlet method is popular within real-time applications as it is both simple to implement and more accurate thant the explicit Euler method. (Fratarcangeli, M. 2011).

Implicit integration can also be used with the mass spring model to help stabalise it. The Backward Euler method can be implemented in numerous ways. Foster presents the main equations as

$$f(x_n, t_n) = \frac{(x_n - x_{n-1})}{\Delta t}$$
$$x_n = \Delta t f(x_n, tn) + x_{n-1}$$

And by reindexing, we can get the equation

$$x_{n+1} = \Delta t f(x_{n+1}, tn + 1) + x_n$$

With x_{n+1} on either side of the equation, the function is implicit. The advantage of an implicit integration method is that it is unconditionally stable (Foster, J., 2008) which could vastly improve the mass spring model when implemented.

3.3 Collision

Implementing collision detection and response within real-time hair simulations can be difficult, as caluclating interhair collisions can be extremely complex and computationally expensive. One method of responding to a collision is through the addition of small repulsive forces. Liu (2012) discusses that if a particle intersects with an object, a repulsive force in the negative direction should be applied until the collision is no longer true.

Liu et al. (2013) discuss another method of collision response using fast approximate collision detection based on spacial subdivison to calculate collisions on a cloth. The response to this collision is to simply stop the external force being applied to the particle and moving the particle to a collision free state. This method is basic, but if used correctly can still achieve a believable behaviour.

4 Potential Additions

4.1 Wet Hair

Once the base hair simulation is complete, there are many additional characteristics that could be implemented in order to create a more interesting simulation. Wet hair simulation is an interesting topic which has come to light recently. From basic methods of changing the mass distribution throughout the hair particles in a strand if using the mass spring model, the characteristics of the hair's movement can be fundamentally changed. Rungjiratananon (2012) discusses how strands begin to clump when wet, and proposes a method of simulating this. This method is not yet used in real-time simulations, with Rungjiratananon discussing it's use within pre-rendered animations.
Silva (2010) also discusses use of the GPU and vertex shaders in order to implement a clumping effect, which takes the position of a droplet of water and creates a fur clump in collision. This method uses a mass-spring model which is embedded into the fur texture, and updates the texture and vertex shader in order to control the clumping and curling of hair.

4.2 Adaptive Mesh Tearing

As discussed within section 3.1 (Mass Spring) of the Methodology, Hooke's law is used to calculate the spring force to be applied to each particle. The stress measured from using the distance from the rest length could be used in further implementations, such as adaptive mesh tearing. For instance, it would be possible to add some sort of constant to show a springs "breaking point". If the change of position is too large, the stress on the string could be too much and a point in which the hair breaks or snaps could be decided. Bertails et al. (2003) discuss the use of adaptive wisp trees in order to separate particles depending on their acceleration and if their child node has already been split.

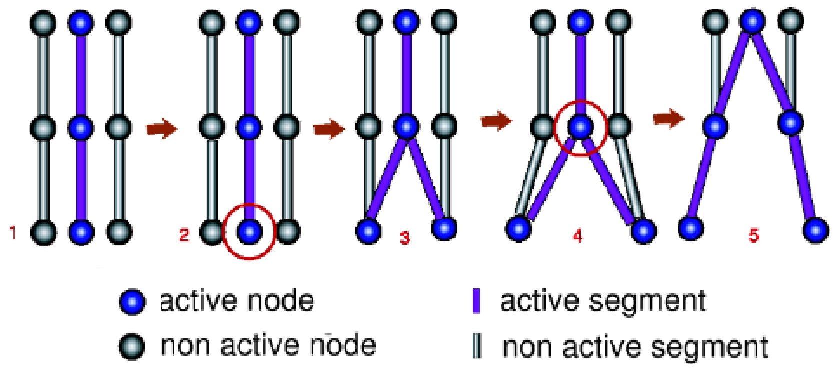

Figure 2: Hair splitting (Bertails et al. 2003)

4.3 Hair Webbing

McAdams et al. (2009) discuss an interesting technique to simulate the webbing of straight hair, usually caused by contact. To do this, they take a volumetric approach to self-interaction between strands, in which they take 5 steps detailed below.

1. Save collision-free position and velocity.
2. Velocity step $v_i^{*n+1} = v_i^n + \Delta t a_i(x_i^n, v_i^{*n+1})$
3. Use the volume technique to modify v_i^{*n+1} into a corrected velocity $\hat{v}_i^{n+1}$
4. Modify $\hat{v}_i^{n+1}$ for collisions to get v_i^{n+1}
5. Compute final position $x_i^{n+1} = x_i^n + \Delta t v_i^{n+1}$

where x_i^n, v_i^n is the time t_n position and velocity of the particle and Δt is the time step (McAdams et al., 2009). This would be an extremely interesting addition to the project, although it is unclear whether this is yet to be implemented in real-time applications.

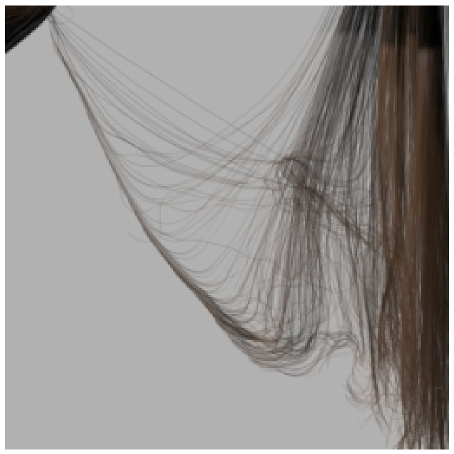

Figure 3: Hair webbing (McAdams et al., 2009)

5 Conclusion

With fast and straightforward implementation, the mass spring model is a good basis for this project. The exact method implemented within the final application should be iterated upon to create the best simulation possible. This means focusing on areas where the mass spring model has issues, such as stability and zero volume collapses. It would also be interesting to investigate potential implementations of wet hair simulations and investigating how proximity algorithms could be used to find the closest neighbours to particles. The more real time methods that can be implemented within the games, animation, and visual effects pipelines, the more time and cost efficient the pre-visualisation process can become.

References

Bertails, F., Kim, T-Y., Cani, M-P., Neumann, U. 2003. Adaptive Wisp Tree - a multiresolution control structure for simulating dynamic clustering in hair motion. *SIGGRAPH Symposium on Computer Animation.* The Eurographics Association.

Foster, J. 2008. *Brief Explanation of Integration Schemes* [online]. Available from: http://engineering.utsa.edu/ foster/me4603/files/integration.pdf [Accessed 17th November 2014].

Fratarcangeli, M. 2011. GPGPU Cloth Simulation using GLSL, OpenCL and CUDA. *In:* Lengyel, E., ed. *Game Engine Gems 2* MA: A. K. Peters, Ltd. 367-368.

Hällman, M. 2011. *Simulation and Visualization of Hair for Real-Time Games* [online]. Thesis (MSc). Chalmers University of Technology.

Kang, Y., Cho, C. 2012. Photorealistic cloth in real-time applications. *Computer Animation and Virtual Worlds.* 23(3-4), 253-265.

Lengyel, E. *Mathematics for 3D Game Programming and Computer Graphics.* 3rd edition. Boston, MA: CENGAGE Learning. 458-459.

Liu, F. *CIS563 Final Project: Hair Simulation* [online]. Available from: http://www.fannieliu.com/hairsim/hairsim.html [Accessed 5th November 2014]

Liu, T., Bargteil, A.W., O'Brien, J.F., Kavan, L. 2013. Fast Simulation of Mass-Spring Systems. *ACM Transactions on Graphics - Proceedings of ACM SIGGRAPH Asia 2013.* Hong Kong. 32(6), 1-7.

Magnenat-Thalmann, N., Kmoch, P., Bonanni, U. 2008. Hair Simulation Model for Real-Time Environments. *Proceedings of the 2009 Computer Graphics International Conference.* 5-12.

McAdams, A., Selle, A., Ward, K., Sifakis, E., Teran, J. 2009. Detail preserving continuum simulation of straight hair. *Proceedings of ACM SIGGRAPH 2009.* 28(3).

Müller, M., Stam, J., James, D., Threy, N. 2008. Real Time Physics. *ACM SIGGRAPH 2008 classes.* ACM. 88.

Müller, M., Kim, T.Y., Chentanez, N. 2012. Fast Simulation of Inextensible Hair and Fur. *Workshop on Virtual Reality Interaction and Physical Simulation.* The Eurographics Assosciation. 39-44.

Rosenblum, R., Carlson, W., Tripp, E. 1991. Simulating the stucture and dynamics of human hair: Modelling, rendering and animation. *The Journal of Visualization and Computer Animation*, 2(4), 141-148.

Rungjiratananon, W., Kanamori, Y., Nishita, T. 2012. Wetting Effects in Hair Simulation. *Computer Graphics Forum* 31(7) 1993-2002.

Parent, R. 2012. *Computer Animation Algorithms and Techniques.* 3rd edition. Elsevier, MA.

Petrovic, L., Henne, M., Anderson, J. 2014 (last modified). *Volumetric Methods for Simulation and Rendering of Hair* [online]. Available from: http://graphics.pixar.com/library/Hair/paper.pdf [Accessed 30th October 2014]

Selle, A., Lentine, M., Fedkiw ,R. 2008. A Mass Spring Model for Hair Simulation. *ACM Transactions on Graphics - Proceedings of ACM SIGGRAPH 2008.* 27(3).

Silva, P. Bando, Y., Chen, B., Nishita, T. 2010. Curling and Clumping Fur Represented by Texture Layers. *The Visual Computer - Proceedings of Computer Graphics International 2010.* 26(6), 659-667.

Tariq, S. 2010. Hair Dynamics for Real-time Application. *Advanced Techniques in Real-time Hair Rendering and Simulation - SIGGRAPH 2010 Course* [online]. Available from: $http : //sarahtariq.com/Chapter7 - HairCourseNotes_{H}airDynamicsforRealTimeApplications.pdf$ [Accessed 12th November 2014]