Simulation and rendering of hair

Chaymae Acherki

Computer Science Student at Linköping University, chaac557@student.liu.se

January, 2023

Abstract

This report covers the simulation and rendering of hair using OpenGL. It has been developed as a project in the course TNM084 - Procedural images at Linköping University.
Hair rendering and simulation have always been challenging tasks, especially in real-time. Due to their high computational demands. However with recent advancements in both graphics hardware and software methods, real-time hair rendering and simulation are now possible with reasonable performance and quality. The project uses a method proposed by Dongsoo Han and Takahiro Harada in order to simulate hair. The report presents the theory behind it, describes the process of creating hair, and discusses potential improvements.

1 Introduction

Hair plays a crucial role in the overall appearance of virtual characters such as humans or animals. However, in the majority of real-time graphics programs, hair rendering has been avoided or replaced with a very simplified and frequently wildly unrealistic polygonal approximation. So instead of trying "fake" hair rendering techniques like these, we will focus on accurately generating hair in a manner that is similar to that of offline hair rendering.
One of the good approaches for modeling and representing hair is to consider hair as a collection of several hair strands. But the problem with this representation is that it is computationally heavy because of the huge number of hair strands. For instance, a person can have over 100,000 hair strands. Therefore, directly modeling each and every hair strand is not the best way. Instead, it is suggested to model only a part of all hair strands (master hairs) and populate the remaining hair model based on the ones that are explicitly modeled.
So, in this project, the idea of simulating and rendering hair is to create master hair strands on the CPU, and then send this data to GPU in order to be processed using the Han and Harada approach, then interpolate between them to create more hair. Finally, some lightning was added to give the hair a better look.
The project is related to the course 'Procedural images', by the fact that it is an application of the use of the tesselation and geometry shaders.

2 Background

2.1 Han and Harada approach

2.1.1 Simulation Overview

The Han and Harada approach is a method for simulating the dynamics of hair in computer graphics. It is based on the idea of simulating hair strands individually on GPU. Each hair strand is represented as a polyline (set of connected vertices). The vertex and edge information of hair polylines are loaded into memory, also all simulation control properties such as external forces or stiffness are fed from CPU to GPU.
One of the key features of the Han and Harada approach is its ability to simulate the interactions between individual hair strands, such as collisions and inter-strand forces. This allows for creating more realistic and dynamic hair simulations.
The simulation is composed of 3 steps: Integration , local and global constraints, length constraints.(1)

2.1.2 Integration

The integration step is used to apply forces to the hair strands such as gravity or wind. Practically, the positions of the hair strand vertices get updated using Verlet integration. Because according to the article (1), Verlet integration is simple and shows good numerical stability compared to the explicit Euler method.
The basic idea behind Verlet integration is to use the positions and velocities of the particles at two different times to calculate the new position of the particles at a later time. The specific equation used in Verlet integration in our case is:

$$P_{i+1} = P_i(1 - D) * V + force * \Delta t^2$$

Where P_{i+1} is the next position to compute, P_i is the current position, V is the velocity, D is damping coefficient is multiplied to velocity to simulate a damping effect, Δt is the time step. And $force$ is the sum of all the forces applied to the hair strand, in our case $force = gravity + windForce$.

2.1.3 Local and global constraints

The global and local constraints are used to preserve the initial hair shape and avoid getting tangled in weird shapes during fast moving gameplay.
It is similar to shape matching we apply Transforms to the initial

hair strands in order to match them with the simulated hair strands. See figure 1

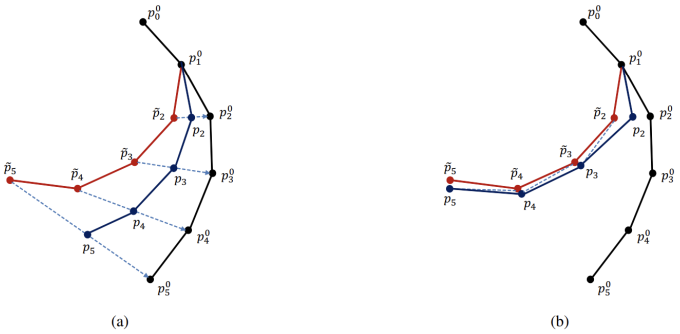


Figure 1 (a) represents how global constraints work, (b) represents how local constraints work.
The black hair strand is the initial hair strand that we try to match with the red one that represent the simulated hair strand in order to get the blue one as a result. (figure from (1))

For global transform, we use the equation below:

$$P_i += S_G(TP_i^0 - P_i)$$

For local transform we use the equations below:

$$d_i = P_i^0 - P_i$$

$$P_{i-1} -= \frac{1}{2} S_L d_i$$

$$P_i += \frac{1}{2} S_L d_i$$

Where P_i is the current position of the hair vertex of index i.
P_i^0 is the initial position of the hair vertex of index i.
S_G is a stiffness coefficient for the global shape constraint. It ranges between 0 and 1. If S_G is 0, there is no effect; if it is 1, the hair becomes completely rigid, frozen to the initial shape.
S_L local stiffness coefficient.
T is the transform matrix.

2.1.4 Length constraints

Length constraints are used to control the distance between each hair vertices P_i and P_j of a hair strand. This is important for maintaining the overall shape and structure of the hair, as well as for preventing the hair from stretching or contracting in an unrealistic way.

The equations used in this constraint are:

$$\delta = P_i - P_j$$

$$d = |P_i - P_j| - l$$

$$P_i -= \frac{m_i}{m_i + m_j} * d * \delta$$

$$P_j += \frac{m_j}{m_i + m_j} * d * \delta$$

Where P_i and P_j the positions of the hair vertices, l is the initial length between them, and m_i and m_j are their masses, but they can be simplified by considering all the hair vertices have the same mass.

2.2 light model

The most common technique used to simulate the appearance of hair in computer graphics is the Kajiya-Kay light mode, also known as the Kajiya-Kay reflection model.(2)
The Kajiya-Kay light model is based on the idea that hair behaves like a translucent medium, and that the color of hair is determined by the color of the light that is transmitted through it, as well as the color of the light that is reflected by it. Also, the reflection of light off of hairs in not just one direction, but a cone formed by rotating the tangent vector of the hair by the surface normal at the intersection of the ray and the hair (3). See figure 2.

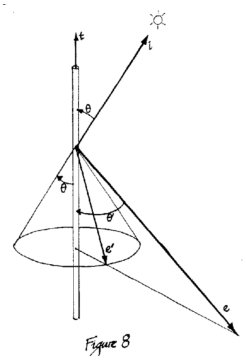


Figure 2 Reflection of the incoming light in Kajiya-Kay model (figure from (3))

The model takes into account a diffuse and specular component.
The diffuse part is the light that scatters uniformly in all directions after hitting the hair surface. This component of the model describes the overall color of the hair and is determined by the color of the light source and the albedo (surface reflectivity) of the hair. This component is mathematically equivalent to the

Blinn-Phong diffuse. The equation below is used:

$$\Psi_{diffuse} = K_d \sin(t, l)$$

The specular component is the light that reflects off the hair surface in a specific direction. This component of the model describes the highlights and shininess of the hair and is determined by the shininess of the hair and the angle of the light source. It differs from Blinn-Phong and is obtained by the equation:

$$\Psi_{specular} = K_s(t.l * t.e + \sin(t, l)\sin(t, e))^p$$

We have t as the tangent vector to the hair cylinder, l as the vector to the light source, and e as the vector to the camera. K_d is the diffuse scalar component, K_s is the specular scalar component, and p is the exponent to the specular component.

3 Implementation

The simulation and rendering of hair were implemented using C++, GLSL and OpenGL library. The implementation follows the steps below:

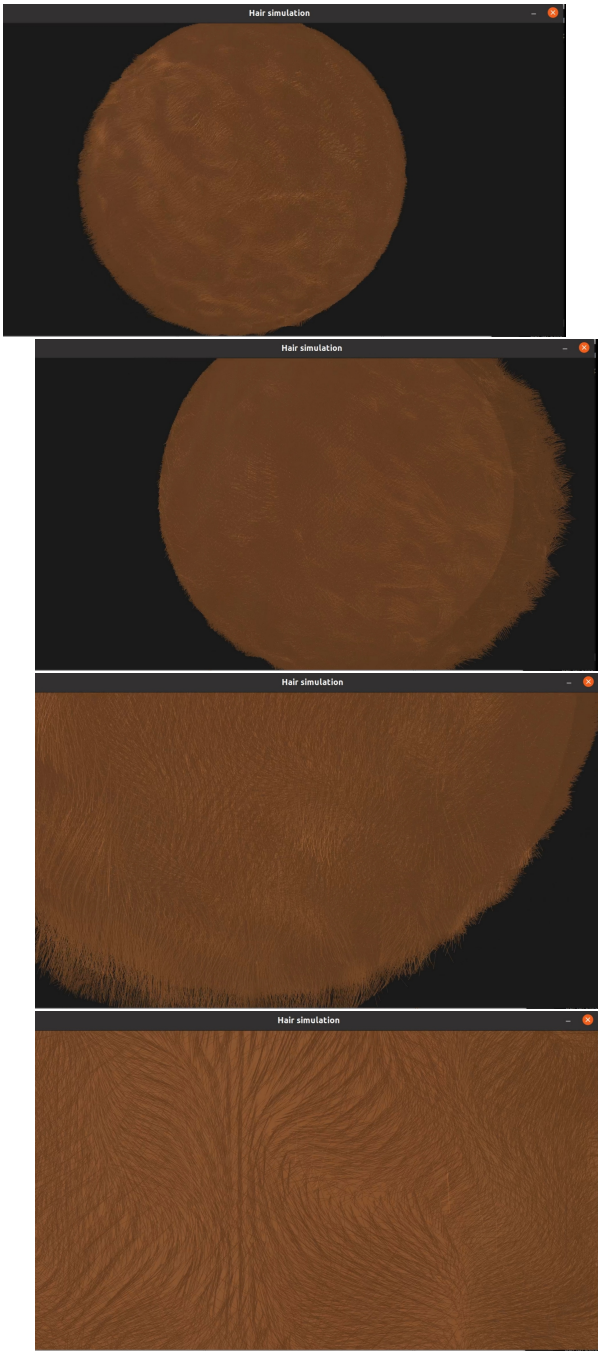
(1) **Creating master hair strands:** It is done on CPU. Based on the object we have (it can be a human head or an animal...) the first hair vertex is created to be the one that is attached to the object, then the second hair vertex is created following the normal of the object, and we continue like this by adding as much as we want of hair vertices.
After creating the master hair strands of the form of a list containing multiples vertices. this list is uploaded to a texture that we send to the GPU.

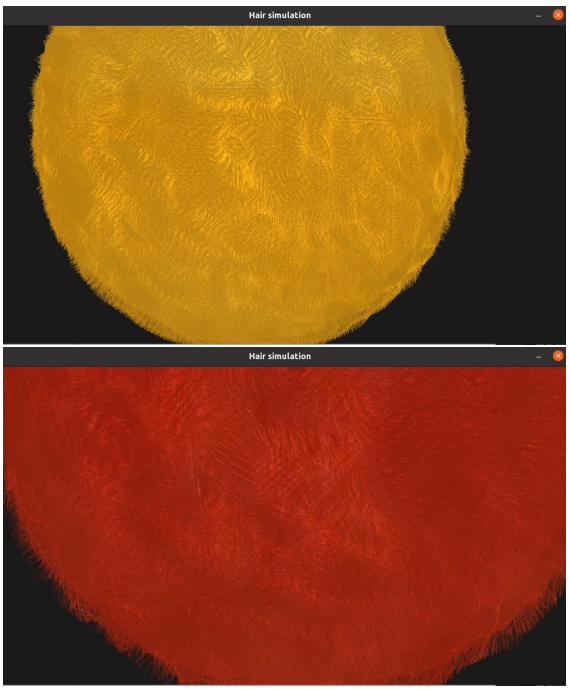
(2) **Processing master hair strands:** It is done on GPU, particularly in the compute shader, using the approach of Han and Harada explained in section 2.1.
In the compute shader we read the data from the texture and we write the result (the new simulated hair strands) in a texture too.

(3) **Create more hair by interpolation:** It is done on GPU. Tesslation shader is used to determine how many interpolated hair strands we want to create. Then geometry shader is used to read data from the texture and from tesselation shader, to create new hair strands.

(4) **Add color/light:** This is done in fragment shader, where light is added using the Kajiy-Kay model explained in section 2.2.
To choose the color of the hair , we use initially a texture contained in a file of format TGA. Or we can change the light color directly.

4 Result





5 Discussion and Future work

As seen in the figures of section 4, the methods used in this project give good results.
To improve the project, in terms of simulation, it would be nice if we can simulate the hair strands using some kind of splines so we can be able to simulate different styles of hair, like a curly one, this idea is presented in (4) and seen in figure 3.

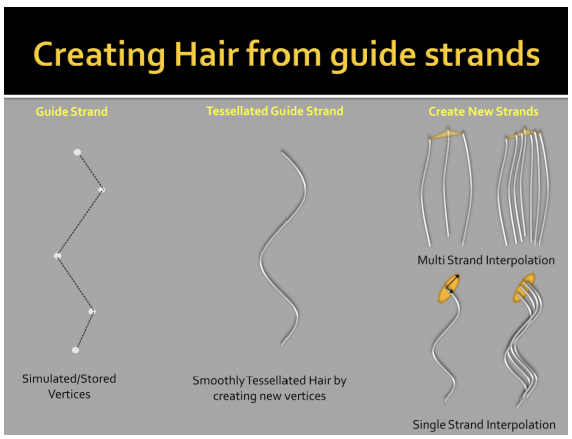


Figure 3 Use of tesselation shader to make smooth hair strands (figure from (4))

One of the key challenges in hair simulation is creating a realistic model of the behavior of individual hair strands. This can involve simulating the interactions between hair strands and other objects in the scene, as well as the effects of forces such as gravity and wind. In our project, only gravity and wind were added, so it would be nice to add collision handling between hair strands and between hair strands and other objects.
Additionally, shadows can also be added since hair shading models are crucial for creating more realistic images of hair.

6 Conclusion

The aim of this project was to show one of the applications of the use of tesselation and geometry shader, namely 'Simulation and rendering of hair".
Simulation and rendering of hair in computer graphics is a complex task that involves modeling the behavior of individual strands of hair as well as the overall appearance of hair. We introduced Han and Harada approach for simulation, this method involves creating a realistic model of how hair behaves in response to various forces, such as gravity, wind, and interactions with other objects. For rendering, we used Kajiya-kay model. This model involves creating a realistic image of the hair based on the simulated behavior and the lighting conditions of the scene.
Personally, at the beginning of this project, I found it difficult because there was a lot of research on this topic and many different approaches were suggested. As a result, I had to choose one based on popularity, one that many people had tried and assumed to be effective.
Overall, it was a good idea to learn more about shaders through this project. I was also able to go beyond the course and learn about the compute shader and its applications.

References

1 Dongsoo Han and Takahiro Harada. Real-time hair simulation with efficient hair style preservation. 2012.

2 James T Kajiya and Timothy L Kay. Rendering fur with three dimensional textures. *ACM Siggraph Computer Graphics*, 23(3):271–280, 1989.

3 Josh Pechner Jacob Lopez, Isaac Lee. Rendering realistic human hair.

4 Sarah Tariq Cem Yuksel. Advanced techniques in real-time hair rendering and simulation.