HOCHSCHULE
RAVENSBURG-WEINGARTEN
UNIVERSITY
OF APPLIED SCIENCES

Simulaton of hair with the Tractrix curve

Master Thesis

at the University of applied science Ravensburg-Weingarten

by

Michael Zappe

August 2020

Student ID	29901
Supervisor	Daniel Scherzer
Secondary supervisor	Martin Zeller

Author's declaration

Hereby I solemnly declare:

1. that this Master Thesis, titled *Simulaton of hair with the Tractrix curve* is entirely the product of my own scholarly work, unless otherwise indicated in the text or references, or acknowledged below;
2. I have indicated the thoughts adopted directly or indirectly from other sources at the appropriate places within the document;
3. This Master Thesis has not been submitted either in whole or part, for a degree at this or any other university or institution;
4. I have not published this Master Thesis in the past;
5. The printed version is equivalent to the submitted electronic one.

I am aware that a dishonest declaration will entail legal consequences.

Ravensburg-Weingarten, August 2020

Michael Zappe

Abstract

The tractrix curve can be used to describe motion in 3D space of any flexible one dimensional object like hair, rope or similar in a follow the leader algorithm. In a simulation, where a strand is discretized into multiple straight segments, the tractrix algorithm accurately describes the motion of the strand if one point of any segment is moved. This simulation has been successfully implemented in non real time environments. The main purpose of this paper is to find a way to implement a real time hair simulation based on the tractrix algorithm.

Two approaches are described. The first one tries to emulate currently existing models for real time hair simulation. It is moderately realistic, but is computational very intensive and not feasible in real time environments. The second proposed approach is not as physically accurate as currently existing simulation models, but it is simple and easy. This should enable rudimentary hair simulation without extensive effort regarding implementation and configuration.

Contents

1 Introduction

Simulation and rendering of hair in real time rendering environments have been improved significantly over the last few decades, but there still is ongoing research to improve either the visual quality or the time needed to compute realistic results. The motivation of this paper is to test a new approach for the simulation of hair strands, that could improve performance or reduce the time needed to configure a realistic looking hair simulation.

There are multiple techniques for simulating hair, like rigid multi body serial chains, super helix simulation and mass spring systems. Many hair simulations are using simulation methods that are comparable to mass spring systems, as they are well understood and fast [Tar10, p.4].

As an example, the *TressFX* framework from Advanced Micro Devices (AMD), which is used to simulate and render Lara Crofts hair in *Tomb Raider*, uses a particle constraint system [Eng14]. A particle constraint system is simpler and more stable than a mass spring system, but not as accurate. In its core, mass spring systems and particle constraint systems use the same concepts of acceleration and integration to keep a simulated object in its desired shape [Tar10, p.5]. Nvidias current hair simulation and rendering framework *Nvidia Hairworks* also uses a particle constraint system [Tar08, p.8] as it is based upon the Nvidia Fermi Demo [Rap14, p.11].

As particle systems are often used in current hair simulation frameworks, this work is based on a particle system, but does not use any kind of traditional particle constraint or mass spring system. Instead, the motion of the hair is simulated with the help of the tractrix curve. This mathematical curve describes the motion of a one dimensional flexible object like hair, rope or similar if one point of the object moves. This might reduce the computational effort of the simulation and still produce realistic looking results. Additionally, the tractrix system should need less configuration then traditional mass spring systems [Men16].

The main contribution of this paper is showing the major problems of using the tractrix curve in a real time hyper realistic simulation of hair. These problems consist mainly of performance and stability issues, the general force propagation in the strand as well as the problem of maintaining a certain hair style. However, an approach is provided which simplifies rudimentary simulations of hair to the point where just one floating point number has to be adjusted, while maintaining decent visual quality, good performance and only needing minimal implementation effort. This approach is not physically accurate or realistic but might be sufficient for basic hair simulation. One example could be a game, where using hyper realistic looking frameworks like TressFX is either not necessary or needs to much performance, but simple hair simulation is a requirement. The paper is organized as follows: Section 2 describes the physical properties of hair and presents currently known methods of simulating it. Additionally, it covers the basics of the tractrix curve and how the tractrix algorithm works. In Section 3, the proposed approaches for using the tractrix algorithm to simulate hair and the problems with each approach are presented. In Section 5, implementation specifics for the tractrix algorithm on a Graphics Processing Unit (GPU) are shown as well as the frameworks and software used to implement the project. Section 6 concludes all findings and includes further ideas on how to use the tractrix in hair simulation.

2 Related Work

2.1 Physical properties of hair

Average humans have around 100 000 to 150 000 hair strands. Geometrically, a single hair can be viewed as a thin curved cylinder. It can either be straight, curly or something in between [Had06].

A typical hair strand is 50 - 120 µm thin and consists of two parts. The core of the strand is called the cortex, which is mostly made up by keratin filaments and contributes to about 90% of the weight of the strand. The region which encloses the cortex is called the cuticle and it forms scales. These scales interfere heavily with light that hits the hair strand and influences the visual look of hair. Because of the small diameter they are easily bend or twisted. On the other hand, the elastic modulus of hair is 3.89 GPa, comparable to the modulus of wood, which makes it difficult to shear and stretch strands [Rap14].

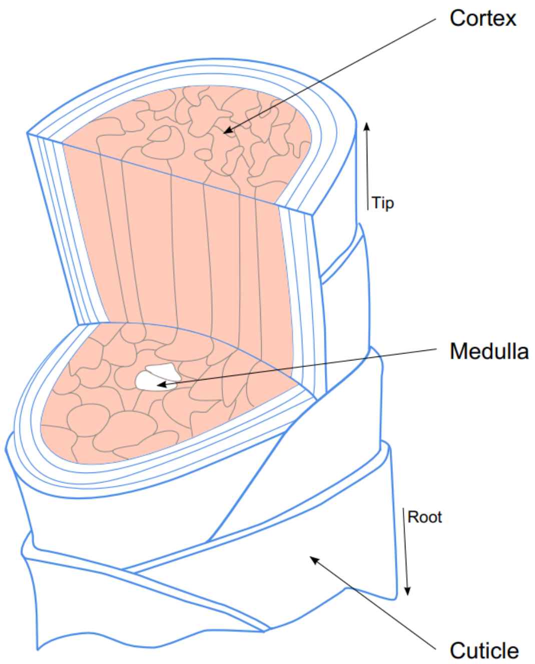


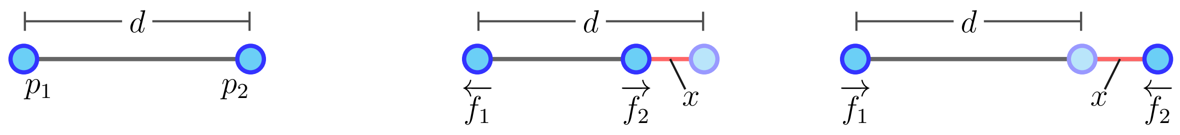
Figure 2.1: Hair strand cross section [Rap14]

2.2 Known hair simulation techniques

A variety of very different approaches exist to simulate hair. One of the earliest attempts was to simulate hair as a continuum and using fluid dynamics to reduce the computational effort that would be needed to simulate each and every strand [HMT01]. Continuum based models are best suited for smooth and fluid hair, while struggling with wavy or curly hair [Ber+06]. Another approach is to use serial rigid multibody chains to simulate hair dynamics. Each strand is divided into segments of similar length and connected by spherical joints. [Had06] This method is good for realistic hair motion, but it is hard to calculate the collision response of the system and it is difficult to parallelize the calculations [Tar10]. An even more robust solution is the super helix model. It uses the Kirchhoff equations for dynamic, inextensible elastic rods to realistically predict hair motion. However the simulation is computational intensive [Ber+06]. Furthermore, mass spring systems are often used for simulation. Some modern real time hair simulation frameworks use a comparable approach to mass spring systems e.g. particle constraints systems. Because of this relationship, mass spring systems will be explained in more detail in the following section [Tar10].

2.2.1 Mass spring systems

Mass spring systems are commonly used in cloth simulation, but their versatility allows them to be also used for hair simulation. They are very well understood, easy, efficient and can produce realistic results. Each hair strand, which is simulated by a mass spring system, is divided into multiple particles. Each particle has its own velocity and mass. To control the distance of neighboring particles, they are connected by coil springs.



We can use these springs to apply forces to each particle according to Hooke´s law.

$$F = kx$$

The factor k defines the stiffness of the spring. A higher k value will result in stronger forces. To use the equation in a mass spring system, x is defined as the difference of the wanted distance d and the current distance between the particles. Resulting in the equation:

$$F = k(||p_1 - p_2|| - d)$$

This represents the strength of the force response. By multiplying this force with the normalized vector between p_1 and p_2 we can accelerate the particles in the desired direction.

$$\overrightarrow{f} = -k(||p_1 - p_2|| - d)\frac{p_1 - p_2}{||p_1 - p_2||}$$

As the force is split evenly between the particles p_1 and p_2, the result of this equation is divided by 2 and assigned to either particle.

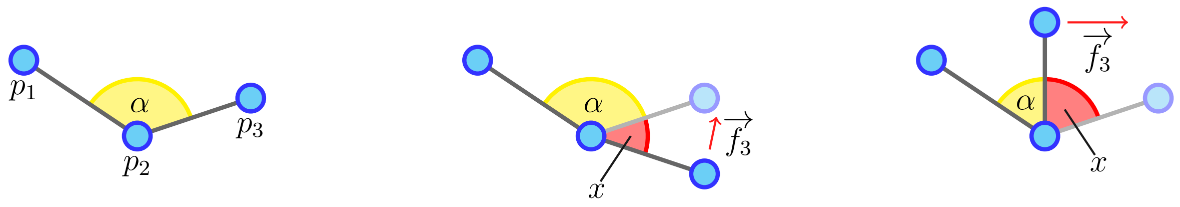
$$\overrightarrow{f_1} = -0.5k(||p_1 - p_2|| - d)\frac{p_1 - p_2}{||p_1 - p_2||}$$
$$\overrightarrow{f_2} = -\overrightarrow{f_1}$$

Knowing the force which is applied to each particle, we can now use Newton´s second law of motion to calculate the resulting acceleration $\overrightarrow{a}$. [Rap14]

$$\overrightarrow{F} = m\overrightarrow{a} \Leftrightarrow \overrightarrow{a} = \frac{\overrightarrow{F}}{m}$$

With the knowledge about the acceleration of every particle, any known integration method can be used to calculate the velocity and thus the movement of the particle.

To maintain hair strands in a certain hair style, like a mohawk, angular springs can be used. These work the same way as coil springs, only the distance d is now measured as a difference in radiants of the wanted angle α and the current angle. Additionally, the corresponding force is perpendicular to the strand direction.



The biggest problems of simulating hair with mass springs systems is that the spring factors k have to be chosen correctly. If e.g. k is too small, coil springs will easily extend or contract, which would contradict the properties of hair, as it does not easily stretch.

However if k is too big, the time between simulation steps has to be small, otherwise the system might not converge and become unstable. The same problem is present in angular springs [Rap14]. This is especially problematic in real time environments, as the time steps are not constant and only a complicated, and thus computational intensive, integration scheme such as implicit backward integration could prevent the system from becoming unstable. Because of this, pure mass spring systems are popular in Visual Effects (VFX) but not in real time environments [Eng14].

2.2.2 TressFX

TressFX is a hair rendering and simulation framework developed by AMD. It is used in games like *Tomb Raider* [Eng14] and *Deus Ex: Mankind Divided* [Par15].

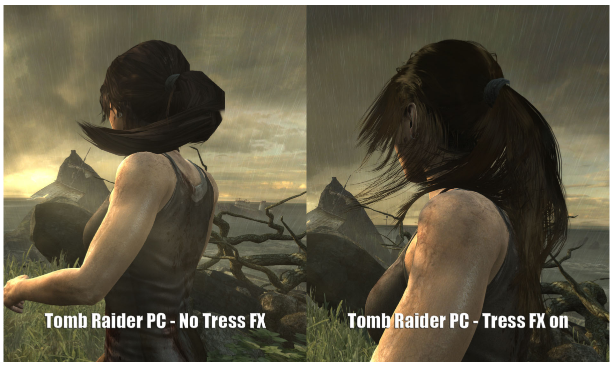


Figure 2.2: TressFX Comparison [Trea]

For its simulation it uses a particle constraint system [Eng14], which is a system that is similar to a mass spring system in its approach [Tar10]. Every hair strand is simulated as a poly line with the following algorithm.

```
Input: hair data
while simulation running do
    integrate;
    apply Global Shape Constraints;
    while itertation do
        apply Local Shape Constraints;
    end
    while iteration do
        apply Edge Length Constraints;
        if too much acceleration then
            apply ad-hoc constraints;
            break;
        end
    end
    apply wind;
    collision handling;
    render;
end
```

Algorithm 1: Simplified TressFX algorithm

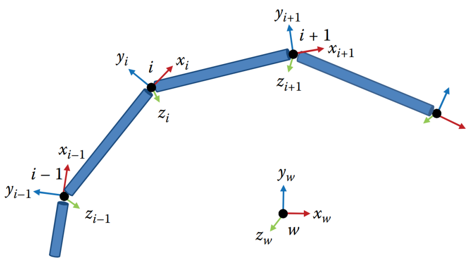


Figure 2.3: Strand as Poly Line [Eng14]

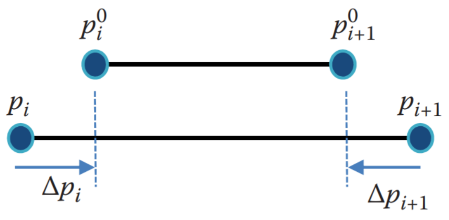


Figure 2.4: Edge Length Constraints [Eng14]

The Global Shape Constraints (GSC) try to preserve the initial hair shape by moving every particle in the direction of its initial hair style position. Local Shape Constraints (LSC) simulate bending and twisting forces by applying a distance constraint where two vertices are connected across the bent edge. To keep the system simple, bending and twisting are combined into one singular effect. The Edge Length Constraints (ELC) work similar as coil springs, however they not only apply force, but also move the particle in the direction of its neighboring particle. Both LSC and ELC are iterative processes with poor convergence. During fast moving sections, the acceleration put on the particles can reach a critical level. In a mass spring system this could lead to an unstable system. In *TressFX* this results in worse convergence of the ELC. To circumvent this problem if high accelerations are present, ad-hoc constraints are applied. These constraints work from the root of the strand and pull or push back the next particle in the poly line to the desired position. This method allows ELC to be met in one step, but it can add extra energy to the system which is removed by adding high damping. After applying GSC, LSC and ELC, external forces like wind, gravity and collision are added and the hair is rendered.

Following this algorithm *TressFX 1.0* was able to simulate around 19 000 strands in less than a millisecond on high-end GPUs. But because stability and performance were the highest concerns, the physical accuracy was lacking [Eng14].

A similar particle constraint system is used in the Fermi Hair Demo, which is the basis for another popular hair simulation and rendering framework *Nvidia HairWorks* [Tar08] [Rap14].

After examining a working approach, we can conclude that a believable hair simulation does not require complete physical accuracy. It has to be primarily stable and performant, and only secondarily physical accurate.

2.2.3 Tractrix

The tractrix curve can be described as the curve that results from the following setup. A string (or generally every one dimensional flexible object) is placed on the Y-axis of a cartesian coordinate system and is pulled in an orthogonal direction of the initial position e.g. along the X-axis. Due to the influence of friction when pulled along this path, the typical tractrix curve is produced. [Traa]

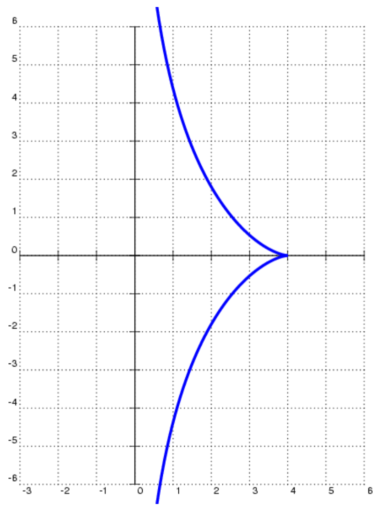
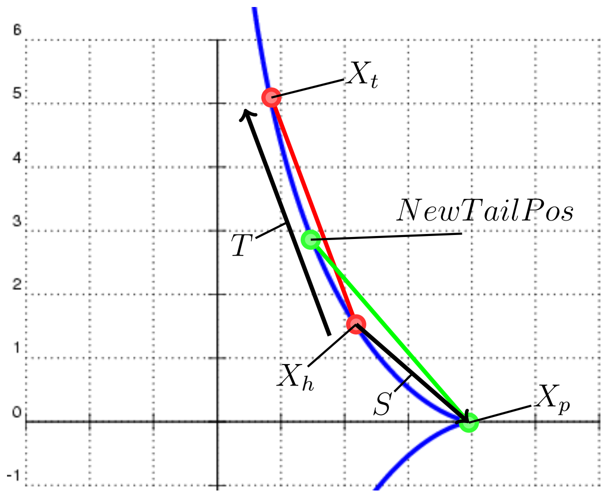


Figure 2.5: Traditional tractrix curve [Trab]

Sreenivasan et al. [SGG10] showed that the movement of a strand can be simulated with the tractrix curve in a *follow the leader* simulation. They developed an algorithm which solves the tractrix motion in 3D space. Every strand that should be simulated is discretised into many straight segments. The strand is moved segment by segment, starting with the leading one. The tractrix resolver function takes the head (X_h) and tail (X_t) position of the segment, and the desired head position (X_p). The output of the function is the new position of the tailing end ($NewTailPos$) and the head will be moved to the desired head position. This movement will be length preserving. One important property of the tractrix method is that the movement of the tail segment will always be smaller or equal to the head movement. This guarantees the movement will die out the further it travels from the leading particle.



The algorithm developed by Sreenivasan et al. [SGG10] is the following:

1. Define the vector $S = X_p - X_h$ where X_h is the current location of the head particle and X_p is the destination point of the head.
2. Define the vector $T = X_t - X_h$ where X_t is the current location of the tail particle.
3. Define the new reference coordinate system $\{r\}$ with the X-axis along S. Hence $\hat{X}_r = \frac{S}{|S|}$.
4. Define the Z-axis as $\hat{Z}_r = \frac{S\times T}{|S\times T|}$.
5. Define the rotation matrix ${}^0_r[R] = \left[\hat{X}_r \quad \hat{Z}_r \times \hat{X}_r \quad \hat{Z}_r\right]$.
6. Define L as $L = |X_h - X_t|$.
7. The y-coordinate of the tail (lying on the tractrix) is given by $y = \hat{Y}_r \cdot T$ and the parameter p can be obtained as $p = L\ sech^{-1}(\frac{y}{L}) \pm |S|$.
8. From p we can obtain the X and Y-coordinate of the point on the tractrix in the reference coordinate system as
   $x_r = \pm|S| - L\ tanh(\frac{p}{L})$
   $y_r = L\ sech(\frac{p}{L})$

9. Once x_r and y_r are known, the point on the tractrix $(x, y, z)^T$ in the global fixed coordinate system $\{0\}$ is given by $(x, y, z)^T = X_h +_r^0 [R] (x_r, y_r, 0)^T$

This movement can now be propagated along the chain of segments with a simple loop.

```
Input: desiredHeadPos
get head/tail position from leading segment;
for each segment do
    newTailPos = TractrixStep(headPos, tailPos, desiredHeadPos);
    save newTailPos;
    desiredHeadPos = newTailPos;
    get head/tail positions from next segment;
end
apply new positions;
```

Algorithm 2: Recursive Tractrix algorithm

This loop results in natural looking movement of a strand, if only the leading particle of the strand is moved with an algorithmic complexity of $\mathcal{O}(n)$, where n is the number of particles in the strand [SGG10].

3 Contribution

Goals of the simulation

The main goal of the approaches is to use the tractrix curve in any way to simulate hair. The metrics used to measure the success of each approach are similar to current hair simulation frameworks (see: TressFX). Primarily stability and performance are considered with physical accuracy as a secondary metric. Performance is important, because in a real time environment time is a limited resource. A 60 fps application only has 16.66 ms to simulate and render the next frame. A stable system is desirable, because in real time applications with user input, many variables that influence the simulation can not be fully controlled. For example the time steps between frames can vary or the acceleration on particles can be high, because the user input generated these accelerations. A stable system will converge to acceptable results even under these conditions.

However, it is a prerequisite that the simulated strand has to be able to maintain a certain hair style. Both approaches share that the strand is discretised into multiple particles which are connected by rigid segments. Furthermore, there is no collision detection. A typical starting configuration looks like this, with the segments connecting the particles in green and blue and the resulting spline in red. This starting configuration was used, because it was easy to imagine a realistic strand in this position and how gravity or other forces will influence it.

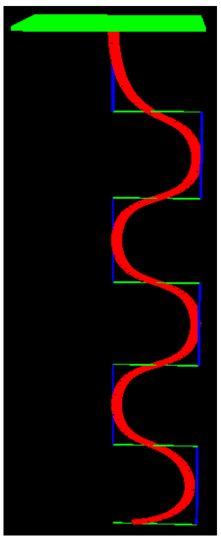


Figure 3.1: Starting configuration

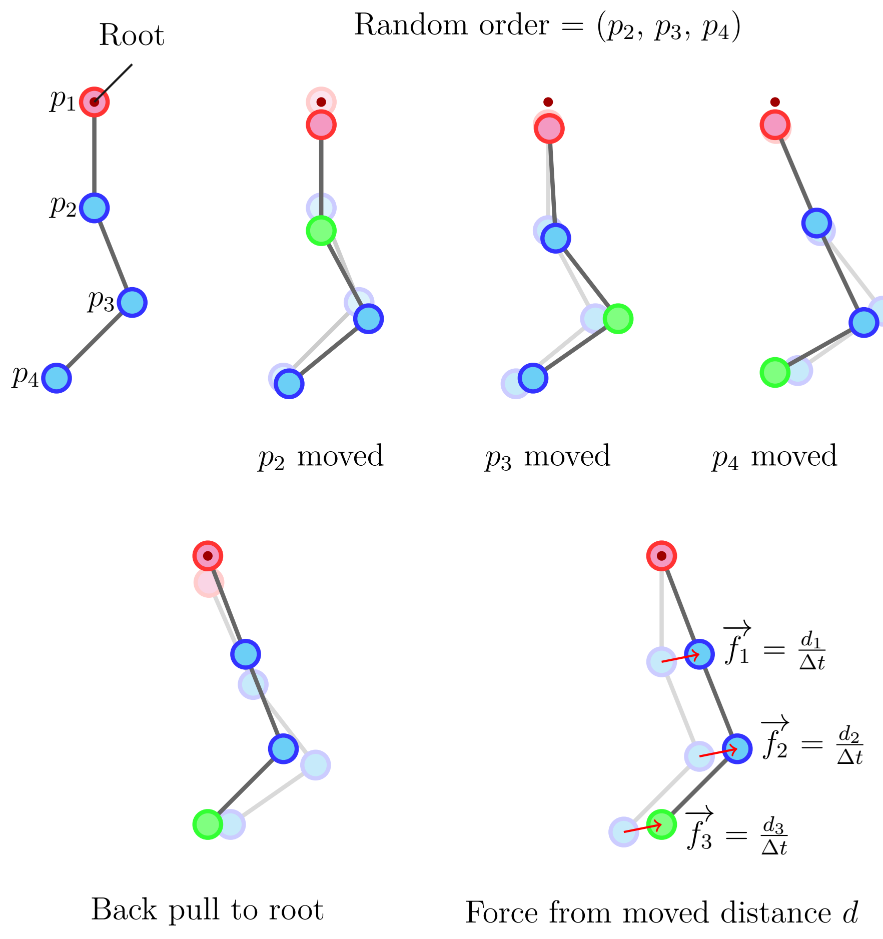
3.0.1 Particle constraint system emulation

The first approach was to emulate a particle constraint system to a certain degree. Every particle has its own velocity and position, which is updated every frame by this algorithm.

```
Input: strand data
apply gravity;
calculate current velocity;
for every particle do
    save desired position;
end
generate random order;
for every index i in random order do
    read desired position i;
    move particle i to desired position;
    apply tractrix move to rest of the strand;
end
pull strand back to root position via tractrix;
for every particle do
    apply forces generated by the movement;
    apply forces to maintain a hair style;
end
```

Algorithm 3: Emulation algorithm

Firstly, global forces are added and the current velocity is updated according to these forces. Based on this updated velocity, the desired position in this frame of each particle can be calculated and saved in a structure. These updated positions can not be applied instantly, as the tractrix move is a *follow the leader* method. Only one particle can move at a time and the rest of the particles will follow its movement, thus the updated positions are applied one after another. To reduce artifacts, the particles are not moved in a linear order, but in a random one. At the end of this loop, one particle will be at its desired position and the rest will be approximately at its own desired one. This is due to the fact that, as mentioned in section 2.2.3, movement will die out the further it is away from the leading particle. After all particles are moved, the strand will most likely not be attached to the scalp anymore. This step can not be skipped by not moving the root particle, as the root particle is not moved on its own, but rather its position is changed by the movement of every other particle in the strand. If the movement of the root would be suppressed, the length between it and the next particle in the strand would change. This would break the length preserving property of the tractrix move and the physical property of hair as it does not strecht.


Root
Random order = (p_2, p_3, p_4)
p_1
p_2
p_3
p_4
p_2 moved
p_3 moved
p_4 moved
$\overrightarrow{f_1} = \frac{d_1}{\Delta t}$
$\overrightarrow{f_2} = \frac{d_2}{\Delta t}$
$\overrightarrow{f_3} = \frac{d_3}{\Delta t}$
Back pull to root
Force from moved distance d

As most particles will not be at their desired position, the difference in movement is divided by the time step Δt. This will result in a velocity, because as mentioned in 2.2.1 $Velocity * \Delta t = Movement$ thus $\frac{Movement}{\Delta t} = Velocity$. The generated velocity will be added onto to the velocity of the particle to account for the forces that the particles apply to each other during the movement. Afterwards, a force is added that moves every particle in the direction of the desired hair style. This force is needed to maintain said hair style.

3.0.2 Splitting physical and hair style simulation

This approach does not try to emulate a physical accurate system. It tries to use the *follow the leader* property of the tractrix movement to simplify approximations of hair subjected to only global forces, like wind and gravity. The simulation splits the position of the particles into the actual rendered position of the hair and a physical representation. Just the leading particle X_L of the strand is physically simulated.

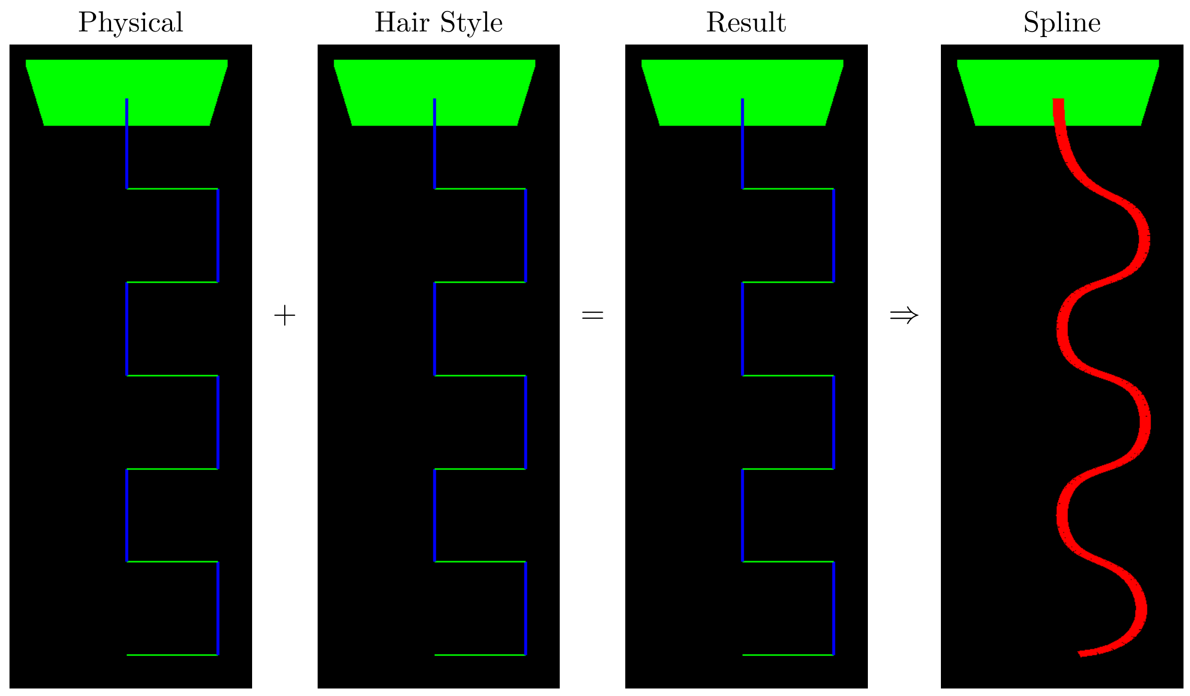
```
Input: strand data
apply gravity and wind to $X_L$;
calculate velocity of $X_L$;
move $X_L$;
apply tractrix move to rest of the strand;
pull strand back to root position via tractrix;
reduce velocity of $X_L$ based on the back move;
currentPos $X_c$ = hairRoot;
for every particle $i$ do
    mix direction of physical and desired strand direction;
    add mixed direction onto $X_c$;
    update position of rendered particle i to $X_c$;
end
```

Algorithm 4: Split algorithm

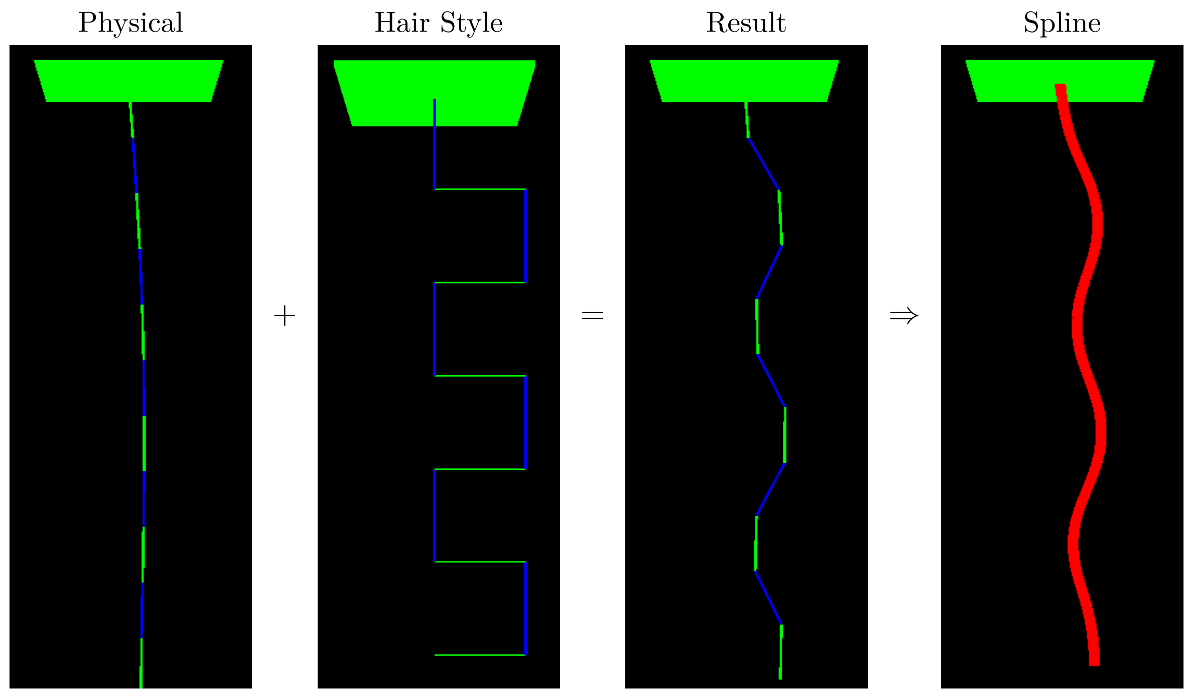
The first part of the algorithm applies forces to particle X_L and moves it accordingly. The rest of the strand follows this movement based on the tractrix calculation. The strand is then pulled back to the root of the hair, to stay attached to the scalp. As the system does not include any kind of constraint or spring, the velocity v_L of the particle X_L will simply grow, because every frame the global forces are added. To compensate for the lack of these elements v_L will be reduced based on the length of the movement X_L experiences by the back move to the hair root. E.g. if every segment of the strand points solely in the -y direction and v_L is pointed in the same direction the displacement of the leading particle d_L will be identical to the displacement of the root d_R. If we now subtract the velocity the root experienced from v_L the velocity of the leading particle will be zero. This allows us to limit v_L in some way.

The loop presented in the algorithm builds up the rendered strand starting from the root. The mix is done by rotating the direction of the physical simulated segment towards the desired direction saved in the hair style. This can be configured by simply changing one floating point value ranging from 0.0 to 1.0 of how far the segment should be rotated. After a segment was rotated, the next position is calculated by adding the resulting direction onto the current position X_c. The position of particle i can now be updated to X_c. The loop continues until every particle was updated with this method.

Starting configuration



Nearly fully extended physical strand



4 Evaluation

4.1 Performance

For performance comparisons the approaches simulated hair assets used in the *TressFX 4.1* Demo. The Demo is split into two separate parts. Additionally four more test sets were generated, which consisted of segments, that were rotated into random directions.

Name	Number Strands	Number Particles per Strand	Particle count
Mohawk	2228	32	71296
Short (Fur)	37747	8	301976
1k16	1000	16	16000
1k32	1000	32	32000
10k16	10000	16	160000
10k32	10000	32	320000

Table 4.1: Performance test set details.

Figure 4.1: TressFX 4.1 Demo Scene [Treb]

The system used for testing consists of a AMD Ryzen 3600 and a Nvidia GForce 1080 Ti. An important factor regarding the implementations used in this paper, is that they are not fully optimized and performance improvements are most likely to be gained through more detailed analysis.

	TressFX	Emulation	Split
Mohawk & Short	<1 ms	105 ms	2.7 ms
1k16	—	3.5 ms	0.4 ms
1k32	—	42.5 ms	0.75 ms
10k16	—	31 ms	1.4 ms
10k32	—	410 ms	5.9 ms

Table 4.2: Performance comparison.

Emulation

Evaluating the measured times it is clear, that the performance of this method is subpar. As mentioned in section 3 a 60 fps application has a 16.6 ms time frame to simulate and render the next frame. The emulation method breaks these time constraints in multiple tests. The emulation method uses the tractrix algorithm, which has a complexity of $\mathcal{O}(n)$, for n strands. This results in a complexity of $\mathcal{O}(n^2)$, which is not suited for an algorithm in a real time environment, that has to be able to process thousands of particles.

Split

The split approach does significantly better than the emulation approach. It only uses the tractrix move twice per strand resulting in a complexity of $\mathcal{O}(n)$. The simulation is slower than TressFX, but as mentioned the used code is not optimized. Considering this and the reasonable complexity of the approach it seems feasible that it could be used in real time environments.

4.2 Stability

Emulation

The stability of the emulation approach relies heavily on the implementation of it. A multitude of forces is added into the system to either simulate how each particle influences other particles, or that try to maintain a hair style. In theory these force responses could be configured to create a stable system. However this no trivial task and the implementation used in this paper is not stable. A typical observed artifact is that the strand, that is only influenced by gravity and a hair style, quickly converges to the desired hair style. After reaching the hair style the particle positions slowly start to drift away from it towards a fully extended strand, that one would expect if no hair style, but only gravity, is applied. A general downside considering stability for the emulation approach is that no iterative processes, like in particle constraints systems, can be used. The performance of the approach is simply not good enough to complete multiple iterations.

Split

Test showed, that the split approach is stable.

4.3 Physical Accuracy

As physical accuracy is only a secondary metric it was not analyzed in detail. In all tests both approaches behave roughly like hair, but the emulation approach looked better. This is expect, as in this approach every particle is moving on its own and is influencing other particles in the strand. It would be possible to implement more physical properties, like e.g. collision detection into the emulation approach, by adding more forces. The physical accuracy of the emulation approach depends on how many of these physical properties are simulated in the specific implementation.

In the split approach only the head of the strand is moving. If the strand is exclusively affected by gravity, the particles closer to the root don't move until the physical strand is mostly extended. Additionally some physical properties, like e.g. collision detection are simply not possible at all with the split approach, as most of the particles can not move on their own. These examples show, that the split approach can't be fully physically accurate. However considering the simplicity of the approach, the degree of realism is sufficient, as it still looks decently like hair.

5 Implementation

The implementation of the project was realized on a Windows 10 device with C++ as the programming language, DirectX 11 as the graphics framework and HLSL as the shader language. The simulation runs in a singular compute shader. The rendering of the scene is accomplished by drawing the segments and splines as lines lists, which are then extended into camera facing quads by a geometry shader. All source code of the project can be found under https://github.com/Michael-Zp/TractrixHairSimulation. To implement the tractrix in a compute shader on a GPU with the algorithm presented in 2.2.3, some special cases have to be considered.

5.1 Undefined Z-Axis

If the movement of the head is exactly in or opposing the direction of the segment, $S \times T$ will result in the vector $(0, 0, 0)$ for the $\hat{Z}_r$-axis. In HLSL normalizing the $(0, 0, 0)$ vector returns an indefinite result. [Hls] To combat this problem, the following error handler was implemented:

```
    float3 crossST = cross(S, T);
    float3 normCrossST = normalize(crossST);
    if(any(isnan(normCrossST)))
    {
        newHeadPosition = desiredHeadPosition;
        newTailPosition = desiredHeadPosition - (tailPos - headPos);
    }
```

In this error handler, the current head position is moved to the desired head position and the tail is the desired head position minus the segment itself. This guarantees, that the resulting segment is the same length as it was before. The error handler might not produce a fully realistic looking results in every situation. However the chance that a segment is pushed or pulled parallel to the strand direction is low, thus this solution is acceptable in these rare occasions.

5.2 Inverse hyperbolic secant

The inverse hyperbolic secant $sech(x)$ will converge to ∞ if $x \to 0$.

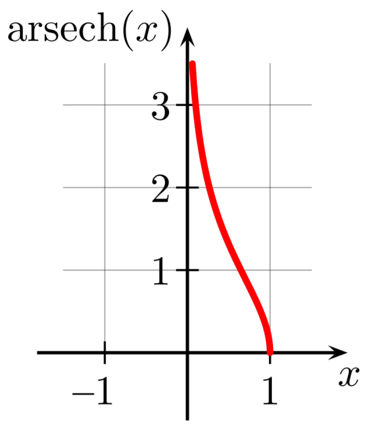


Figure 5.1: Inverse Hyperbolic secant graph [Sec]

This is a problem on a GPU, because a floating point number can not store indefinitely big numbers, but it is easily circumvented by minimising the result of the function.

```
    float p = L * sechInverse(y / L) + lengthS;
    p = min(100, p);
```

6 Conclusion

The tractrix curve is not suitable to simulate hair in a realistic, stable and performant way on its own. A moving hair will apply forces to itself based on inherent constraints of the strand or caused by properties like a hair style. The tractrix curve does not provide a reliable way to calculate these forces. Still, it consistently and realistically describes the movement of a hair and it was shown in the *Splitting physical and hair style simulation* approach that this fact can be used to simplify basic hair simulation and reduce the workload that has to be put into developing these systems. The performance and stability of this simpler approach are shown to fit the requirements for real time environments. It is not physically correct, but still should be sufficient for rudimentary hair simulation. Additionally the effort needed to configure this tractrix based hair simulation is reduced to adjusting one singular floating point number. On the other hand, implementing a traditional mass spring or particle constraints system needs a configuration, which does not become unstable and behaves realistically. This requires much more effort than adjusting the proposed simple tractrix simulation.

A future topic for research about the tractrix curve in hair simulation, could be its integration in fully functioning particle constraints systems. These systems rely on numerical methods to converge and if the tractrix move is used in some of these iterations, it might be possible to cut down the number of times the calculations have to be repeated. This could reduce computational cost or even improve the simulation by making it more physically accurate.

Acronyms

AMD	Advanced Micro Devices
ELC	Edge Length Constraints
GPU	Graphics Processing Unit
GSC	Global Shape Constraints
HLSL	High Level Shading Language
LSC	Local Shape Constraints
VFX	Visual Effects

List of Figures

List of Tables

References

[Ber+06] Florence Bertails et al. “Super-helices for predicting the dynamics of natural hair”. In: (2006), pp. 1180–1187.

[Eng14] Han Engel. *GPU Pro 5: Advanced Rendering Techniques - Hair Simulation in TressFX*. Taylor & Francis, 2014. Chap. VI.1, pp. 407 –417. ISBN: 9781482208634. DOI: https://doi.org/10.1201/b16721.

[Had06] Sunil Hadap. *Hair Simulation*. John Wiley and Sons, Ltd, 2006. Chap. 8, pp. 161–191. ISBN: 9780470023198. DOI: 10.1002/0470023198.ch8.

[Hls] *Hlsl - Normalize*. URL: https://docs.microsoft.com/en-us/windows/win32/direct3dhlsl/dx-graphics-hlsl-normalize (visited on 08/12/2020).

[HMT01] Sunil Hadap and Nadia Magnenat-Thalmann. “Modeling Dynamic Hair as a Continuum”. In: *Computer Graphics Forum* 20.3 (2001), pp. 329–338. DOI: 10.1111/1467-8659.00525.

[Men16] Ghosal Menon Gurumoorthy. “Efficient simulation and rendering of realistic motion of one-dimensional flexible objects”. In: *SODA ’10 Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms* (2016), pp. 13 –26. DOI: http://dx.doi.org/10.1016/j.cad.2016.02.003.

[Par15] Paras. *Dawn Engine – First Slide Showcasing AMD’s TressFX 3.0*. 2015. URL: https://wccftech.com/dawn-engine-slide-showcasing-amds-tressfx-30/ (visited on 08/11/2020).

[Rap14] Rapp. *Real-Time Hair Rendering*. 2014. URL: http://markusrapp.de/wordpress/wp-content/uploads/hair/MarkusRapp-MasterThesis-RealTimeHairRendering.pdf (visited on 08/10/2020).

[Sec] *Inverse Hyperbolic Secant*. URL: https://upload.wikimedia.org/wikipedia/commons/thumb/6/69/Inverse_Hyperbolic_Secant.svg/626px-Inverse_Hyperbolic_Secant.svg.png (visited on 08/13/2020).

[SGG10] S. Sreenivasan, Piyush Goel, and Ashitava Ghosal. "A real-time algorithm for simulation of flexible objects and hyper-redundant manipulators". In: *Mechanism and Machine Theory* 45.3 (2010), pp. 454 –466. ISSN: 0094-114X. DOI: https://doi.org/10.1016/j.mechmachtheory.2009.10.005.

[Tar08] Bavoil Tariq. *Real-Time Hair Simulation and Rendering on the GPU*. 2008. URL: http://developer.download.nvidia.com/presentations/2008/SIGGRAPH/RealTimeHairSimulationAndRenderingOnGPU.pdf (visited on 08/10/2020).

[Tar10] Yuksel Tariq. *Hair Dynamics for Real-time Applications*. 2010. URL: http://www.cemyuksel.com/courses/conferences/siggraph2010-hair/S2010_HairCourseNotes-Chapter7.pdf (visited on 08/10/2020).

[Traa] *Tractrix*. URL: https://en.wikipedia.org/wiki/Tractrix (visited on 08/12/2020).

[Trab] *Tractrix Curve*. URL: https://en.wikipedia.org/wiki/Tractrix#/media/File:Tractrix.png (visited on 08/12/2020).

[Trea] *TressFX 1.0 Tomb Raider*. URL: https://mygaming.co.za/news/wp-content/uploads/2013/03/Tomb-Raider-2013-PC-TressFX-comparison.jpg (visited on 08/12/2020).

[Treb] *TressFX 4.1 Demo*. URL: https://github.com/GPUOpen-Effects/TressFX/tree/4.1 (visited on 08/12/2020).