SMARTICLES: A METHOD FOR IDENTIFYING AND CORRECTING INSTABILITY AND ERROR CAUSED BY EXPLICIT INTEGRATION TECHNIQUES IN PHYSICALLY BASED SIMULATIONS

A Thesis

presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Susan Marano

June 2014

COMMITTEE MEMBERSHIP

TITLE:	Smarticles: A Method for Identifying and Correcting Instability and Error Caused by Explicit Integration Techniques in Physically Based Simulations
AUTHOR:	Susan Marano
DATE SUBMITTED:	June 2014
COMMITTEE CHAIR:	Zoë Wood, Ph.D. Department of Computer Science Associate Professor
COMMITTEE MEMBER:	Timothy Kearns, Ph.D. Department of Computer Science Associate Professor
COMMITTEE MEMBER:	Aaron Keen, Ph.D. Department of Computer Science Associate Professor

ABSTRACT

Smarticles: A Method for Identifying and Correcting Instability and Error Caused by Explicit Integration Techniques in Physically Based Simulations

Susan Marano

Using an explicit integration method in physically based animations has many advantages including conceptual and computational simplicity, however, it requires small time steps to ensure low numerical instability. Simulations with large numbers of individually interacting components such as cloth, hair, and fluid models, are limited by the sections of particles most susceptible to error. This results in the need for smaller time steps than required for the majority of the system. These sections can be diverse and dynamic, quickly changing in size and location based on forces in the system. Identifying and handling these troublesome sections could allow for a larger time step to be selected, while preventing a breakdown in the simulation.

This thesis presents Smarticles (smart particles), a method of individually detecting particles exhibiting signs of instability and stabilizing them with minimal adverse effects to visual accuracy. As a result, higher levels of error introduced from large time steps can be tolerated with minimal overhead. Two separate approaches to Smarticles were implemented. They attempt to find oscillating particles by analyzing a particle's (1) past behavior and (2) behavior with respect to its neighbors along a strand. Both versions of Smarticles attempt to correct unstable particles using velocity dampening. Smarticles was applied to a two dimensional hair simulation modeled as a continuum using smooth particle hydrodynamic. Hair strands are formed by linking particles together using one of two methods: position based dynamics or mass-spring forces.

Both versions of Smarticles, as well as a control of normal particles, were directly compared and evaluated based on stability and visual fluidity. Hair particles were exposed to various forms of external forces under increasing time step lengths. Testing showed that both versions of Smarticles working together allowed an average increase of 18.62% in the time step length for hair linked with position based dynamics. In addition, Smarticles was able to significantly reduce visible instability at even larger time steps. While these results suggest Smarticles is successful, the method used to correct particle instability may jeopardize other important aspects of the simula-

tion. A more accurate correction method would likely need to be developed to make Smarticles an advantageous method.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Physically Based Animation

The use of computer graphics in animation is rapidly growing. Physically based animation allows complex and realistic behavior to be automatically calculated using physics models. This reduces the tedious task of animation by hand in movies and allows interactive simulations in games. Some common items modeled with physically based animation include deformable solids, cloth, fluids, and in thesis hair.

Unlike more scientifically based applications that focus on accuracy, the goal in computer animation is to provide physically plausible movement that is visually appealing. This allows short cuts to be taken that sacrifice scientific accuracy for speed. If too much inaccuracy and error is allowed, however, the simulation may lose numerical stability and cause negative visual affects.

In this thesis, the short cut being taken is the use of an explicit integration technique with large time steps. This type of integration method is known for its simplicity and low computational complexity, however, the accuracy is inversely related to the size of the time step. Larger time steps reduce the number of evaluations per second resulting in a faster simulation, but result in more error. When too large of a time step is used, visual artifacts appear, such as oscillation and other unnatural behaviors. Smarticles, as presented by this thesis, aims to reduce the appearance of these artifacts. This way larger time steps can be used while maintaining the appearance of visual correctness to the viewer or user.

1.2 Hair Modeling

Smarticles is applied to a hair simulation in order to evaluate the effectiveness of Smarticles in reducing the appearance of error caused by explicit integration methods. For this reason, hair is modeled using techniques known to be ill-suited for explicit integration methods. This thesis models hair strands as chains of particles connected with strong spring forces. Strong spring forces are particularly sensitive to time step size when using explicit methods. Unless the time step is kept small, particles begin to oscillate around the true solution.

This hair implementation gave an ideal scenario for testing Smarticles, however, the strong spring forces require impracticably small time steps to maintain stablity. As a result, the simulation can not support real time speeds, even with Smarticles. Smarticles is intended for real time and interactive applications, therefore a second hair strand model is implemented to better meeting these time constraints. The second method models hair strands as a chain of particles constrained together using position based dynamics. This method allows significantly larger time steps than using spring forces, which allows the simulation to run at real time speeds. This method is still adequate for evaluating Smarticles given that large enough time steps still produce visible error. In this thesis, both methods for modeling hair strands are implemented and used to evaluate Smarticles.

In addition to modeling hair strands, the collective behavior of the hair must be addressed. Simulating hair is a particularly difficult feat due to the large number of strands and complexity of motion. A human head of hair can contain over 100,000 strands [33]. These strands easily bend but strongly resist stretching and shearing. Additionally, hair is constantly colliding and interacting with the body, air, and other hair strands. Hair on hair interaction is uniquely effected by static electricity. Rather than model the interaction of every hair strand with every other hair strand, one technique is to generalize the interaction using a fluid model.

This thesis uses smooth particle hydrodynamics to simulate the collective behavior of hair. This way each hair particle has a rest density, maintaing an adjustable distance from other hairs. This gives the appearance that hair strands rest on top of each other. In addition, viscosity is used to simulate the friction caused by hair

strands colliding and sliding along each other. A fluid model handles much of the collective hair behavior and allows control over global hair properties such as hair volume.

1.3 Numerical Integration

Mathematical models used to describe a dynamic physical system often involve a set of time-dependent differential equations. In general, these equations give the relation between an unknown function and one or more of its derivatives. When initial conditions are also given, it becomes an initial value problem, as shown in Equation 1.1. Here the unknown function is $x(t)$ and its relation to $x'(t)$ is described with the known function f. The initial value at time t_o is given as x_o. In most cases, x represents a vector.

$$\begin{cases} x'(t) = f(t, x(t)) \\ x(t_o) = x_o \end{cases} \tag{1.1}$$

In order to progress a simulation, a solution of x(t) must be found to give the state of the system. Finding the exact solution analytically would be ideal, allowing exact values to be known at any time with a relatively trivial calculation. Unfortunately, this approach is only practical in the most simplistic scenarios; the majority of physical systems result in differential equations that are extremely difficult or impossible to solve with analytic techniques. Therefore, numerical techniques are used, which increment forward in time finding an approximated solution at each discrete time step.

Generally, numerical integration methods can be divided into two main categories: explicit and implicit. Explicit methods have the advantage of often being computationally and intuitively simpler. As a result they are easy to implement and fast at evaluating each time step. One negative, however, is their potential to produce significant error if evaluations are insufficiently frequent. This is seen in situations sensitive to error, such as those involving stiff equations, where explicit methods often require impractically small time steps in order to maintain numerical stability. Implicit methods, however, do not produce increased error in relation to step size,

therefore, they can achieve the required accuracy with much larger step sizes. As a result, implicit methods can often advance the simulation faster even though a single time step evaluation takes longer than with an explicit method.

1.4 Summary

This thesis aims to reduce the error caused by explicit methods to allow the use of larger time steps even in situations involving stiff equations. This is accomplished with the main contribution of this thesis, Smarticles. Smarticles detects when a particle's updated position (determined with an explicit integration method) exhibits excess error that interferes with visual accuracy. This is done by comparing a particle's current movement with its past movement and/or the movement of adjacent particles in the structure. Smarticles then attempts to correct the particle so that instability is not seen or spread to other particles.

A two dimensional hair simulation is implemented and used to evaluate the effectiveness of Smarticles. The overall collective behavior of hair is modeled using smooth particle hydrodynamics. Hair strand structure is maintained with the use of constraints between adjacent particles along a strand. Two versions of the hair simulation are used, differing in the method used to enforce constraints. One method uses the position based dynamics technique presented in [21] while the other method uses spring forces.

Discrete tests are preformed on the hair simulation to determine how large the time step can be increased before signs of instability are seen. These finding are used to compare the effectiveness of Smarticles in allowing larger time steps, specially in two diminutional hair simulations that use SPH, PBD, and spring forces.

1.5 Contribution

The main contribution of this thesis is Smarticles, a method of reducing the instability caused by explicit methods in physically based simulations. This includes two particle instability detection methods: Past Behavior Comparison (PBC) and Strand

Neighbor Behavior Comparison (SNBC). In addition, Smarticles includes a correction method which attempts to stabilize the particles found to be unstable.

Testing showed Smarticles allowed at least a minimal increase in time step size for the hair simulation using position based dynamics. While Smarticles did not allow larger time steps for the hair simulation using spring forces, a reduction in stability was still exhibited. An in-depth discussion is given exploring the possible reasons better results could not be achieve. This leads into suggestions on possible areas to improve Smarticles in hopes to make it a more practical method.

In addition, this thesis implements a unique hair simulation utilizing smooth particle hydrodynamics, dynamic follow-the-leader (a position based dynamics method presented in [21]), and a two dimensional environment.

Chapter 2

BACKGROUND

2.1 Integration techniques

As mentioned in Section 1.3, numerical integration of time-dependent differential equations is a major concern in physically based animation. Producing fast results is highly desirable, however, a minimum amount of accuracy is needed not only for visual appeal, but also to maintain a stable simulation. This Section defines and describes various numerical integration techniques used in computer animation.

2.1.1 Euler's Method

Given Equation 1.1, a solution for $x(t)$ is desired. Starting at the initial value $x(t_o)$, future values of x are found by taking discrete times steps of size h. The approximate change in x, Δx, over the time interval h can be directly found using the derivative function f. Then, by incrementing $x(t_o)$ by Δx, a value for $x(t_o + \Delta t)$ can be found. This technique yields Euler's method, the simplest of numerical solutions, shown in Equation 2.1. Substituting $f(t, x(t))$ with $x'(t)$ from Equation 1.1 simplifies Equation 2.1 to Equation 2.2.

$$x(t + h) = x(t) + hf(t, x(t)) \tag{2.1}$$

$$x(t + h) = x(t) + hx'(t) \tag{2.2}$$

Given that $x'(t)$ is not the same for all values of t, the accuracy of Euler's method

is highly dependent on h, the size of the time step. In addition, each new time step is starting from an approximated value for $x(t)$ causing the error to accumulate. As time progresses the approximated solution will drift farther from the true solution resulting in divergence.

A clear example of this is seen when using spring forces. A simple method of enforcing constraints uses Hooke's law, which states the force required to deform a spring is equal to k times the displacement of the deformation. Applying the law to an object will produce a force in the direction that reduces displacement, resulting in the appearance of a constraint. This often requires a relatively large k value to allow the constraint to appear stiff when other forces are acting on the object. In order for Euler's method to converge, however, the step size must be less than $2/k$. This means h must be quite small for large values of k, otherwise the propagation of error will cause the found solution to overshoot the rest position and oscillate farther and farther from the true solution.

2.1.2 Mid-point Method

Note that Euler's method evaluates f at the beginning of the time step. A better estimate can be obtained by evaluating f at the middle of the time step, as shown in Equation 2.3. This requires Δx to be known, therefore Equation 2.4 is used to find an approximation for Δx as is done in Euler's method.

$$f_{mid} = \Delta t f(\frac{t + \Delta t}{2}, \frac{x + \Delta x}{2}) \tag{2.3}$$

$$\Delta x = \Delta t f(t, x(t)) \tag{2.4}$$

Replacing f with f_{mid} in Equation 2.1 results in Equation 2.5, which is known as the Mid-point method. This requires slightly more computation than Euler's method, but generally makes up for it by producing more accurate results.

$$x(t + h) = x(t) + h f_{mid} \tag{2.5}$$

The Mid-point method is more accurate because it approximates a second order term in the Taylor series resulting in a second order approximation rather than just first order. This leads into Runge-Kutta methods which continue to reduce error by approximating more terms in the Taylor series. A common Runge-Kutta method, known as RK4, provides a 4th order approximation.

2.1.3 Backward Euler

Backward Euler is the implicit version of Euler's method. Rather than using $f(t, x(t))$, which evaluates f at the x value determined in the previous time step (as in the explicit solution in Equation 2.1), Backward Euler uses $f(t + h, x(t + h))$. This way f is evaluated at the targeted value for x as shown in Equation 2.6.

$$x(t + h) = x(t) + hf(t + h, x(t + h)) \tag{2.6}$$

As a result, this method is time reversible, meaning it will return to the same initial conditions if time steps were run in reverse. This allows for an important property in physically based simulations: conservation of energy.

Unfortunately, $f(t+h, x(t+h))$ can only be directly solved if f is a linear function. Therefore, a linear approximation is usually found using Taylor series. This becomes a nontrivial task when considering f is a vector, which requires f' to be a matrix. As a result, a linear system needs to be solved at each time step. Clearly this requires significantly more computation than using an explicit method to solve a single time step, however, the stability of implicit methods allows longer time steps to be taken. Also, in many cases f' is sparse, allowing for a near linear time complexity.

2.1.4 Semi-Implicit

Physically based simulations typically utilize Newton's 2nd Law of motion, force equals mass times acceleration, to find an objects position. This leads to the second order problem described in Equation 2.7, where x is position, v is velocity, and a is

acceleration.

$$\begin{cases} x''(t) = v'(t) = a(t) = f(t, x) \\ x'(t) = v(t) \end{cases} \tag{2.7}$$

The previously described integration methods can easily be applied to these problems, usually by solving $x(t)$ and $v(t)$ separately as two first-order problems. Often better results can be achieved, however, by solving them together as a second order problem. For example, the Semi-Implicit method improves upon Euler's method by using the found value for $v(t + h)$ in the equation for finding $x(t)$ as seen in Equation 2.8 and Equation 2.9. This allows $x(t)$ to be found implicitly with out introducing more computation.

$$v(t + h) = v(t) + ha(t) \tag{2.8}$$

$$x(t + h) = x(t) + hv(t + h) \tag{2.9}$$

2.1.5 Leap-Frog

The Leap-Frog method is a popular method for solving second order problems, such as the one described in Equation 2.7, due to its simplicity and improved accuracy. It is unique in that it evaluates x and v asynchronously. The velocity is evaluated at the mid-points of each time step interval while position is evaluated on the interval as normal. As illustrated in Figure 2.1, the velocities and positions appear to leap over each other.

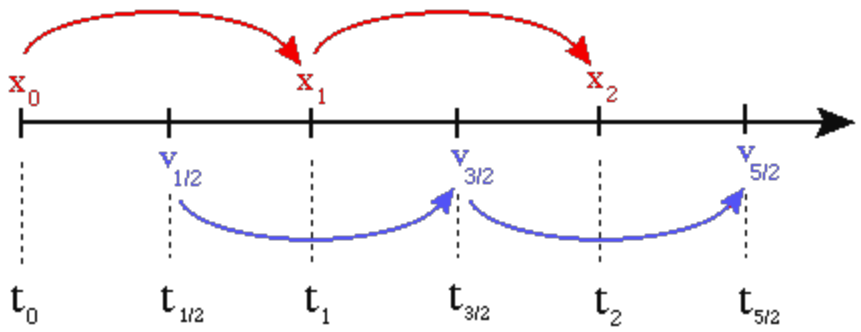

Figure 2.1: Leap-Frog method. Velocity and position is evaluated at times indicated by subscript. Source: `http://einstein.drexel.edu/students/courses/Comp_Phys/Integrators/leapfrog/`

The resulting solutions for position and velocity are shown in Equation 2.10 and Equation 2.11. The initial offset for velocity can be determined with another method, such as Euler's method. Like the Semi-implicit method, Leap-Frog must evaluate v first, so that it can be used to evaluate x.

$$v(t + \frac{1}{2}\Delta t) = v(t - \frac{1}{2}\Delta t) + \Delta t a(t) \tag{2.10}$$

$$x(t + \Delta t) = x(t) + \Delta t v(t + \frac{1}{2}\Delta t) \tag{2.11}$$

A major advantage of the Leap-Frog method, as compared to other explicit methods discussed in this thesis, is its time reversibility. Like Backward Euler, Leap-Frog would arrive at the same initial conditions if it were run in reverse. Therefore divergence is not a major concern, however, this method is still very sensitive to step size. A comparison of the error produced by the explicit methods Mid-point, RK4, and Leap-Frog can be seen in Figure 2.2.

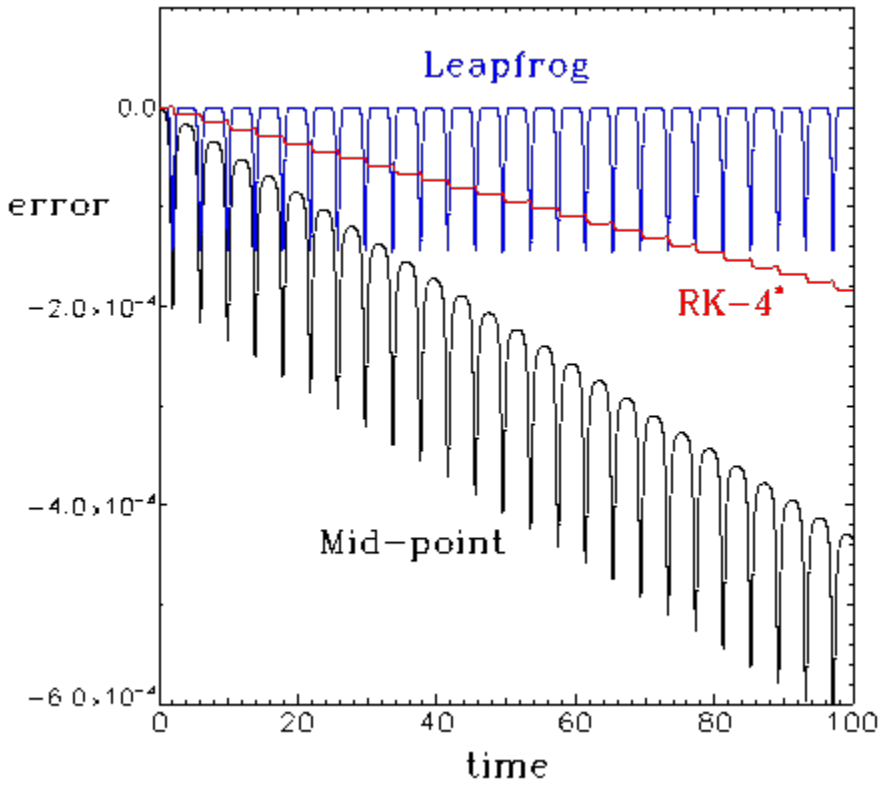

Figure 2.2: Error produced by explicit integration methods as time progresses. Source: `http://einstein.drexel.edu/students/courses/Comp_Phys/Integrators/leapfrog/`

2.2 Position Based Dynamics

Dynamic simulations are traditionally force based, where Newton's 2nd Law of motion is used to calculate accelerations from the forces acting on the system, as described in the previous Section. Position based dynamics (PBD) takes an alternate approach, which avoids much of the instability produced from the need to use numerical integration to update velocity and position from acceleration. Instead, position is directly manipulated, providing a high level of control. This provides an attractive method for enforcing constraints, such as those needed to model a strand of hair.

PBD has many of the qualities computer animation finds attractive, such as speed, stability, and visual accuracy. Additionally, it can be easily understood and implemented. In some applications, such as modeling a strand of hair, it has a comparable simplicity to that of a mass spring model. Unlike spring forces, however, PBD does

not have the issue of overshooting that leads to energy gain and instability.

2.2.1 Follow-The-Leader

A hair strand can be modeled as a chain of particles, each with position x_n (where x_1 is fixed). Each particle is attached to its adjacent particles with rest distance of l_o. If any particles violate these distance constraints, a PBD approach can be used to reposition them to valid locations.

The order in which constraint violations are resolved is nontrivial. Each particle is linked to two other particles (rather than one), therefore moving x_i so that it is l_0 away from x_{i-1} likely violates the constraint between x_i and x_{i+1}. Of course, this becomes an even bigger issue in applications that require high particle connectivity, such as cloth meshes. Many approaches find a sufficiently valid solution by iterating over all particles many times, leading to high computation costs.

Luckily, in a chain topology, all constraints can be resolved in a single iteration using Follow-The-Leader (FTL), where each particle is processed from 2 to n. This way, modifying particle x_i only affects x_{i+1} which is next in line to be modified.

2.2.2 Dynamic Follow-The-Leader

In order to actually resolve particle constraints, valid particle positions need to be found. In addition, particle velocities need to be updated to reflect the change in position so that the simulation is dynamic. This can be achieved with Dynamic Follow-The-Leader (DFTL), a PBD method presented in [21].

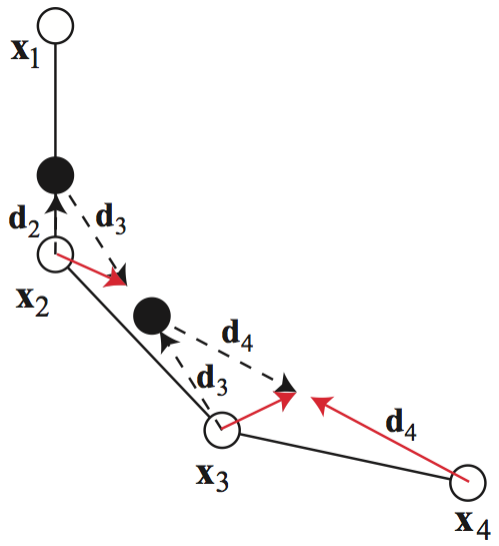

Figure 2.3: Position and velocity corrections using DFTL. The new velocity for each particle i is the vector $d_i - d_{i+1}$, seen as solid red arrows. [21]

At each time step, first the position and velocity are updated to account for any forces in the system. This is done using a time integration method, such as those described in Section 2.1. Using the updated position p_i, constraints are enforced using equation Equation 2.12. Velocity is then updated to account for the position change using Equation 2.13.

$$p_i = p_i + d_i \tag{2.12}$$

$$v_i = \frac{p_i - x_i}{\Delta t} + s_{damping}\frac{-d_{i+1}}{\Delta t} \tag{2.13}$$

The circle centered at x_{i-1} with radius l_o represents all possible locations of particle i that fulfills the constraint between particles $i-1$ and i. Therefore, if i is not on the circle, a new value for x_i must be found. A reasonable location for particle i is the point on the circle closest to the original position x_i. Once this is found, a correction vector, d_i, is formed from the old particle position to the new particle position.

2.3 Smooth Particle Hydrodynamics

A particularly difficult aspect of hair simulation is efficiently modeling the complex and numerous hair interactions. Thousands of hair strands are colliding and interacting with each other and the environment. One approach to modeling this collective hair behavior is to use a fluid model, such as smooth particle hydrodynamics (SPH).

In SPH, discrete sampling points (particles) are used to approximate values within a continuous field. Each particle contains specific information pertaining to the physical properties and state of the system at that point, such as mass, density, and velocity. This allows fluid flows to be modeled in a computer simulation. In this thesis, SPH provides a way of modeling the combined dynamics of many interacting hair strands.

SPH is a Lagrangian method, meaning the sampling points move with the fluid flow. This differs from a Eulerian method, where sampling is done along grid aligned cells. Originally developed for astrophysics, SPH is better able to handle areas of extreme density, as seen in astronomical phenomena. By having the sampling points move with the flow, SPH naturally supports variation in resolution of field qualities like density.

2.3.1 Interpolation Method

SPH provides a means for calculating field variables at any point in a system. This is achieved through interoperation, based on the identity given in Equation 2.14, where $A_I(x)$ is the integral interplant of any quantity function, $A(x)$, ranging over Ω. Additionally, W is a smoothing kernel with width h, derived from an approximated Dirac delta function.

$$A_I(x) = \int_{\Omega} A(x')W(x - x', h)dx' \tag{2.14}$$

A discrete form of Equation 2.14 is given in Equation 2.15 so that the continuous field is represented as a set of particles. The summation in Equation 2.15 iterates overall all particles j, where m_j and ρ_j are the mass and density attributes implicit

to particle j respectively. Additionally, A_j is the value of any quantity A at position x_j.

$$A_S(x) = \sum_j A_j \frac{m_j}{\rho_j} W(x - x_j, h) \tag{2.15}$$

With the use of Equation 2.15, the property A of the particle at x can be found as the sum of the properties of the surrounding particles. Each particle, j, contributes to the property weighed according to its density and distance. As would be expected, the larger the density and shorter the distance, the greater the impact.

As it turns out, only the smoothing kernel differs in derivatives of the summation interpolant in Equation 2.15. Therefore, the spatial derivative of a property value can be found with Equation 2.16. Subsequent derivatives can also be found with this way.

$$\nabla A_S(x) = \sum_j A_j \frac{m_j}{\rho_j} \nabla W(x - x_j, h) \tag{2.16}$$

In some cases, Equation 2.16 is too inaccurate of an approximation for the gradient field. In those cases, Equation 2.17 can be used.

$$\nabla A_S(x) = \rho \sum_j \left(\frac{A_j}{\rho_j^2} + \frac{A}{\rho^2} \right) m_j \nabla W(x - x_j, h) \tag{2.17}$$

2.3.1.1 Smoothing Kernel

The smoothing kernel dictates the rate at which particle j decreases in influence as distance increases. This rate is a reflection of the smoothing length, h. Once particle j is at a distance greater than the smoothing length, it no longer has an impact on the property value at x. Therefore, it is desirable for $W(x, h) = 0$, given $\|x\| > h$. Also, it is expected that a kernel function be normalized and even.

2.3.2 Lagrangian Fluid Dynamics

The Navier-Stokes equations describe the fluid flow for an incompressible, isothermal fluid. The Lagrangian formation of these equations is given in Equation 2.18, where v is velocity, ρ is density, p is pressure, μ is viscosity, and f represents the sum of external forces.

$$\rho\frac{dv}{dt} = -\nabla p + \mu\nabla^2 v + f \tag{2.18}$$

The right hand side of Equation 2.18 can be interpreted as the sum of internal and external forces: $f_{total} = f_{internal} + f_{external}$. This allows the acceleration of a particle to be found with equation Equation 2.19. Acceleration can then be used to find particle velocity and position using a time integration method.

$$a = \frac{f_{total}}{\rho} \tag{2.19}$$

2.3.2.1 Density

All other fluid properties depend on density, therefore density is the first property to be calculated. Equation 2.20 gives the simplified form of Equation 2.15 that can be used to find the density of each particle.

$$\begin{aligned}\rho(x_i) &= \sum_j \rho_j \frac{m_j}{\rho_j} W(x_i - x_j, h) \\ &= \sum_j m_j W(x_i - x_j, h)\end{aligned} \tag{2.20}$$

2.3.2.2 Pressure

Pressure contributes to the internal forces acting on each particle. The pressure at each particle can be found using the Ideal Gas Law:

$$pV = nRT \tag{2.21}$$

where V is volume per unit mass, n is amount of substance given in moles, R is the universal gas constant, and T is temperature. Given an isothermal fluid with constant mass, the right hand side of Equation 2.21 remains constant allowing it to be replaced with, k, a gas stiffness constant. Also, $V = \frac{1}{\rho}$, therefore, an equation for pressure can be written as Equation 2.22.

$$p = k\rho \tag{2.22}$$

An ideal gas will expand to fill a space, however, liquids prefer to maintain a constant rest density. As a result, Equation 2.22 is not ideal for liquids, due to always giving a positive pressure value. Instead, Equation 2.23 can be used, where ρ_0 is the rest density of the material.

$$p = k(\rho - \rho_0) \tag{2.23}$$

With the pressure at each particle determined using Equation 2.21, the force resulting from that pressure can be found as $\nabla p(x_i)$. Unfortunately, this can not be solved with Equation 2.16 as would be expected. Doing so results in asymmetry which does not conserve momentum, therefore the more accurate Equation 2.17 should be used. This results in Equation 2.24.

$$\nabla p(x_i) = \rho \sum_j \left(\frac{p_i}{\rho_j^2} + \frac{p_i}{\rho^2} \right) m_j \nabla W(x_i - x_j, h) \tag{2.24}$$

2.3.2.3 Viscosity

Particles within a fluid experience friction between themselves and surounding objects resulting in a loss of kinetic energy. This is known as viscosity, seen in fluids as a resistance to flow. From Equation 2.18, the term $\mu\nabla^2 v$ is the internal force associated with viscosity. Equation 2.25 gives the equation for calculating the viscosity force

action on a particle i.

$$\mu\nabla^2 v(x_i) = \mu \sum_j (v_j - v_i) \frac{m_j}{\rho_j} \nabla^2 W(x_i - x_j, h) \tag{2.25}$$

2.3.2.4 External Forces

External forces make up any other forces within the system acting on the fluid particles. This can include forces from many sources including gravity, collisions, and user interaction. Those chosen to be included in a SPH simulation highly depends on the application.

2.4 Quadtree

In many physically based simulations, spacial data structures are a critical component in reducing computational time complexities. For this thesis, a quadtree is used to increase the efficiency of the neighbor search required by the smooth particle hydrodynamics portion of the simulation. In SPH, all particles within every other particle's smoothing length must be found. This way only neighboring particles need to be iterated over in order to find a property value for a particle.

A quadtree is a tree data structure that recursively subdivides two demential space into four equal regions until a threshold is reached. For this reason, every node in the tree contains four children, excluding leaf nodes. The quadtree shown in Figures 2.4 and 2.5 for example, continued to divide the scene until each leaf node contained no more than two points. Alternatively, a quadtree could be designed to stop after a set number of divisions.

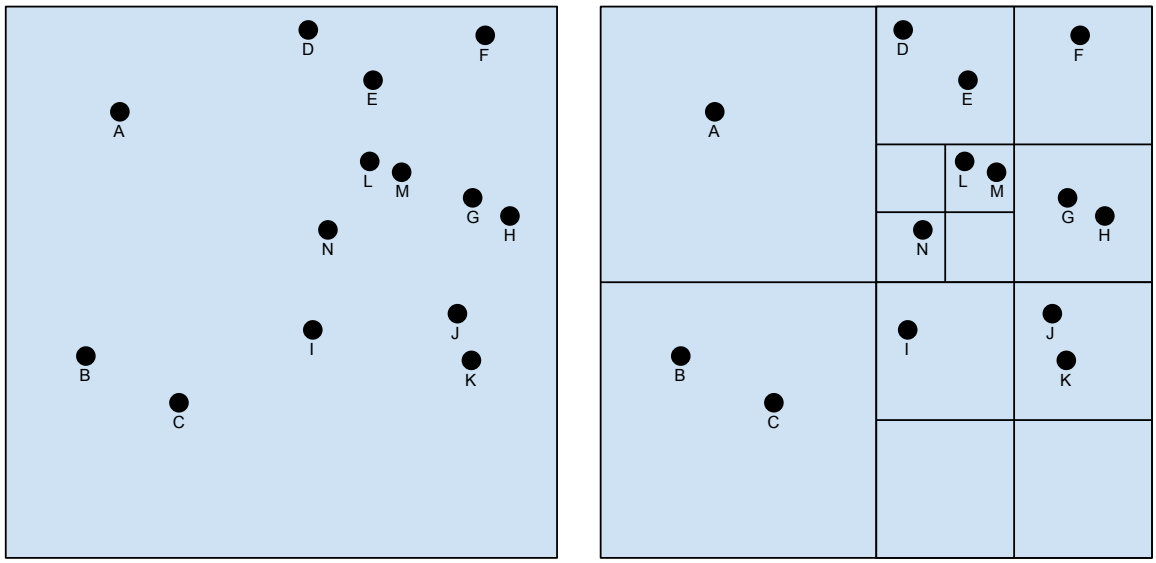

Figure 2.4: An example of how a scene is divided by a quadtree. The tree representation is shown in Figure 2.5.

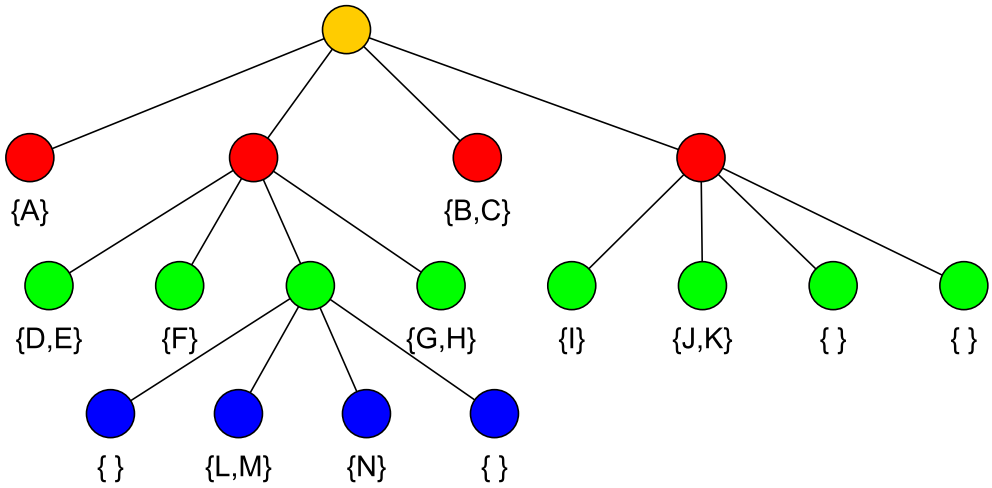

Figure 2.5: An example quadtree for the scene shown in Figure 2.4.

Once a quad tree is constructed, a particle's neighbors can be found by traversing the tree. Starting at the root, each child is traversed if the region it represents intersects the circle centered at the particles position with a radius equal to the smoothing length, h. Once a leaf is reached, the particles within that region are found to be the neighbors of the original particle if they are within the circle. This

way, a quadtree can often save computation time by eliminating regions at a time rather than each particle individually.

Chapter 3

RELATED WORKS

Combatting the instability that arises from stiff equations, while minimizing computational complexity has been a major concern in simulating deformable objects. Section 3.1 explores the main methods developed to combat this problem and their major contributors. Section 3.2 then briefly describes the major approaches to simulation hair dynamics. In addition to often facing stiff equations, hair is particularly difficult to simulate due to its complex physical properties and large number of interacting strands.

3.1 Numerical Integration with Stiff Equations

3.1.1 Post Correction Methods

Integration methods are a significant concern in cloth simulation due to its wide use of mass-spring systems. Stiff equations quickly become an issue due to cloth's tendency to resist stretching but allow bending and shearing. Many implementations using explicit integration methods have combatted this issue by simply using smaller spring forces than the material properties would suggest. The unavoidable over elasticity seen in high stress areas are then handled with post correction methods.

One of the fist post correction methods, presented in [25], iteratively checks and limits the elongation of each spring connecting the nodes of a cloth mesh using position modification. Two nodes found too far apart (exceeding a predefined threshold) are repositioned closer together. This solution is based on the assumption that the Euler integration method used provides correct values with respect to direction, but not

necessarily the distance between the two endpoints.

A pitfalls of this method is the potential for an undesirable propagation effect. When a spring is readjusted, it has likely modified the elongation of adjacent springs sharing the same endpoints. Additionally, the current implementation iterates through the spring evaluations and adjustments in an order dictated by data structure. Provot [25] argues these are not issues for the targeted situations in which problem springs are in highly concentrated and localized regions. Arguably one of the benefits of this method is its ability to help distribute a deformation throughout the structure, rather than it remaining in a concentrated region.

Vassilev et al. [30] improved upon this method by modifying the velocity rather than position of the particles. The forces acting on the cloth to produce elongation, such as gravity and object collisions, are taken into account to determine the new endpoint velocities of over elongated springs. When applied to all springs, the resulting velocities no longer contribute to stretching. This method improves upon [25] by showing promising result for both global and local deformations.

3.1.2 Implicit Methods

For the most part, cloth simulation utilized explicit methods until [5] presented a practical implicit solution that allowed for significantly larger time steps. Their method includes a modified conjugate gradient method that both solves the implicit integration and enforces particle constraints.

This is achieved using a mass modification method to enforce constraints. By using an inverse mass vector and modifying it appropriately, a particle can be constrained in any dimension and consequentially any direction. As a result, constraint enforcement is an inherent property of the first order differential equation describing the system.

This equations is solved using the implicit integrator, backward Euler, resulting is a nonlinear equation. Solving an implicit nonlinear equation is computationally expensive, therefore, a Taylor series expansion is used to create a first order approximation. Fortunately, the accuracy of this approximation is not effected by the time step. Baraff and Witkin [5] solve the final equation using their modified conjugate

gradient method, specially derived for efficiency with the constraint enforcement.

As a result of [5], implicit methods have been widely used due to their stability even when using large time steps. Future work has mainly focusing on reducing the high cost of solving a linear system at each time step. Desbrun et al.[10] provides an approximated solution, sacrificing accuracy for speed. The result is similar to an explicit integration scheme, but able to retain stability with large time steps. This succeeded in a speed up, however, the computational complexity is still much more significant than an explicit method.

Kang et al. [16] continued simplify computation by eliminating the need to solve a large linear system. This provided a fast and visually realistic solution, however, physical correctness was not considered. As a result, some materials experience incorrect behavior, such as over elongation.

3.1.3 Implicit-Explicit Methods

Implicit-Explicit (IMEX) integrators, first developed by [9], provide a combined approach to solving differential equations. The main objective is to solve the stiff portions with an implicit method, and the non-stiff portions with an explicit method. This way both the stability of an implicit method and computational simplicity of an explicit method are utilized.

A main concern of IMEX methods is determining when and how to most effectively separated the differential equations for each integration scheme. [28] takes an extreme approach, determining separation before the differential equations are even formed. Implicit integration is responsible for the fast forces that are typically oscillatory and linear, while explicit integration handles the slow forces that tend to result in non-linear equations.

Boxerman and Ascher [6] use an adaptive IMEX scheme along with a parallel approach to improve computational efficiency at each time step. By decomposing the cloth mesh into smaller components, implicit and explicit methods can better be applied based on local characteristics.

3.2 Approaches to Hair Simulation

The simulation of hair dynamics is often divided into two main stages: strand dynamics and collective dynamics. Strand dynamics determines the structure and motion of individual hairs, while collective dynamics is concerned with the hair as a collection including how hair interacts with itself and the environment. Often a method from one stage is compatible with any method from another stage, however, some methods work better together than others.

The major contributors to stand and collective dynamics are described in Section 3.2.1 and Section 3.2.2. For a more comprehensive and trough overview see [33] and [13], which include other aspects of hair simulation such as styling and rendering.

3.2.1 Strand Dynamics

An early attempt by Rosenblum et al. [26], uses a mass-spring system to model hair. Each hair strand is made up of a set of particles attached with springs and hinges. The simplicity of this model is highly appealing, however, stiff equations quickly become an issue. More so than cloth, hair resists stretching which requires strong spring forces to prevent. Mass-spring systems have been given more consideration for hair simulation since [5] provided a practical integration technique that allowed for larger time steps, as seen in [27] and [23].

Another early attempt, presented by Anjyo et al. [3], utilizes projective dynamics for modeling individual strands. One-dimensional projective differential equations are used to determine the angular momentum of linked rigid sticks. This method preserves the length of each segment, eliminating the issue of hair stretching, however, handling collisions and constraints is much more difficult.

Forward kinematics, borrowed from robotics, has been applied to hair strand simulation by several works including [14], [24], [8], and [12]. A rigid multi body serial chain reduced to spatial coordinates is used to represent a strand of hair. Motion is then computed using an articulated body method as described in [11]. Being more computationally expensive per strand than mass-spring systems or projection dynamics, this method is often used to simulated a reduced number of hairs. The

remaining hairs are then animated using interpolation methods, such as continuum or wisp as described in Section 3.2.2.

More recently, hair strand dynamics has been successfully simulated using position based dynamics. Presented by [20], PBD was originally developed for a mesh topology, such as cloth. This required an iterative method for resolving constraints. Brown et al. [7] avoided this by developing an algorithm (Follow the Leader) for chain topologies that require only a single pass. Unfortunately this only applied to static simulations, such as knot tying. The challenge of creating a dynamic version was solved by [21]. This method guarantees both stability and inelasticity of hair with only one iteration per frame.

3.2.2 Collective Dynamic

A human head of hair can consist of hundreds of thousands of hair strands, each interacting with each other, the body, and the air. Some approaches do simulate and render every hair, such as [26], [3], and [27], however, this can be cumbersome. Rather than simulate every individual hair strand, many implementations take advantage of the observation that hair tends to move together in a fluid motion. Therefore, a collection of hair can then be modeled as a continuum.

This lead Hadap and Magnenat-Thalmann [14] to model hair as a fluid using smooth particle hydrodynamics. Hair on hair interactions are handled with Lagrangian fluid dynamics, while individual hair strands are still modeled using forward kinematics with rigid chains (as described in Section 3.2.1). Bando et al. [4] also uses SPH, but does not explicitly define hair strands. Instead, each particle represents a portion of hair with a direction. Particles are then grouped with neighboring particles that share similar orientations. This method provides for a more real-time application, however, visual accuracy can be hindered by lack of connectivity and detail.

Another continuum method, presented by [22], uses a fixed grid volumetric approach. Hair is rasterized to a Eulerian velocity field to capture hair on hair interactions. A main focus is on reducing computational complexity rather than depicting a high level of detail. Unfortunately, this fixed grid model does not allow for fluid

compression.

Attempting to gain the benefits of both Lagrangian and Eulerian approaches, McAdams et al. [19] presents a hybrid model. A Eulerian fluid model is used to minimize computational complexity. Hair self collisions are then handled using a high resolution Lagrangian method, ensuring high fine detail. The implementation also gives explicit volumetric density control.

Many approaches also use a sparse set of guide hairs to reduce the number of strand simulations needed, including [24], [12], and [23]. Resolving collisions between these hairs then becomes a more practical task. Additional hairs can then either be interpolated between the guides or clumped with the guides. Choe and Ko [8] created guide hairs using a hybrid method of mass-spring and rigid multi body serial chains. A statistical model was then used to create more hairs from the guides.

Chapter 4

DYNAMIC HAIR SIMULATION

4.1 Hair Structure

A collection of hair is stored as an array of particles. Each of these particles are a root node for a linked list of particles representing a single hair strand. A illustration of this data structure can be seen in Figure 4.1.

The SPH portion of the simulation is computed with no regard to this hair strand arrangement. Excluding root nodes, which remain fixed, all particles are allowed to interact together as a normal fluid. The hair strand structure and dynamics is therefore handled with one of two independent mechanisms: PBD or mass-spring. Both methods were implemented to provide additional testing scenarios for evaluating the Smarticles algorithms.

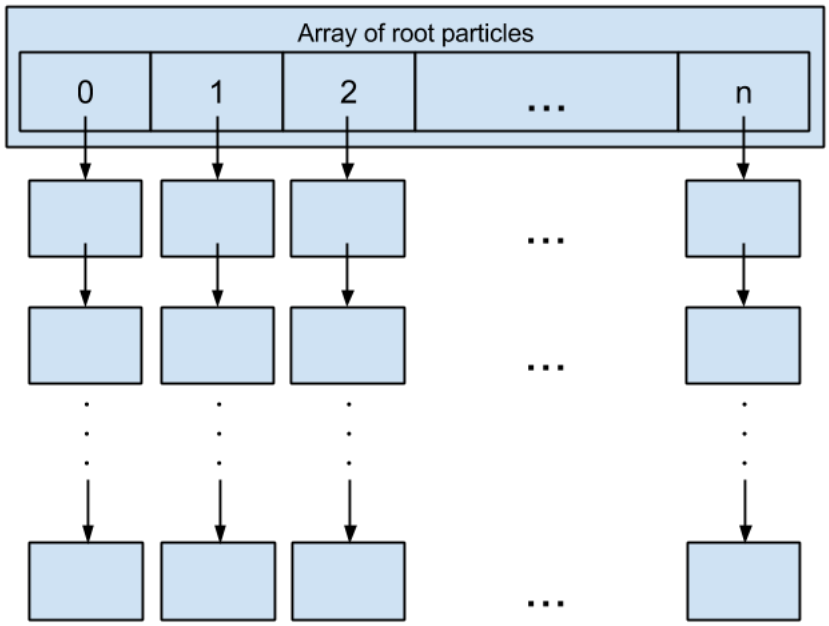

Figure 4.1: Data structure for hair containing n strands

4.2 Hair Dynamics

4.2.1 Collective Dynamics

In this thesis, collective dynamics is handled with Smooth Particle Hydrodynamics, most closely following the implementation given in [17]. Figure 4.2 gives an overview of the SPH process preformed for each time step.

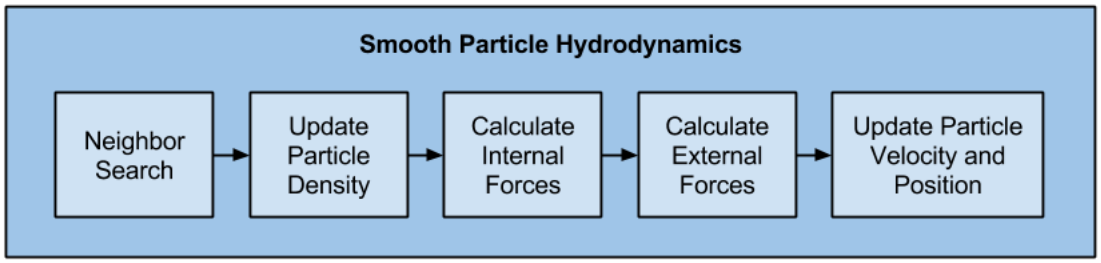

Figure 4.2: SPH process for a single time step

4.2.1.1 Neighbor Search

When evaluating a property value of a particle, there is no need to iterate over all other particles in the system. Instead, only a particle's neighbors are relevant, specifically the particles within its smoothing radius. For this reason, all smoothing kernels given in Section 4.2.1.2 evaluate to zero for distances greater than h.

A neighbor search needs to be preformed for every particle at the start of a time step. For the purposes of this thesis, neighbor refers to any particle within another particle's smoothing length h. By extension, a neighbor search finds all the neighbors of a particle. Preforming a neighbor search for every particle can be a time consuming task. A brute force approach would require comparing the distances of every particle to every other particle, resulting in a time complexity of $O(n^2)$. Implementing and utilizing an quadtree as discussed in Section 2.4 can significantly reduce this time complexity.

All memory for the quadtree is allocated during the initialization stage of the simulation. This way the memory for the tree does not need to be freed and reallocated every time step. To accommodate the possible tree constructions, memory is allocated for a balanced tree with a fixed depth. The chosen depth is based on the number of particles in the scene. It is unlikely the hair particles will always be distributed equally through out the scene, therefore the depth is biased to be deeper than likely necessary.

4.2.1.2 Smoothing Kernels

The following smoothing kernels were chosen based off of the recommendations given in [17]. It is assumed $\|x\| < h$ for all kernels.

$$W_{density}(x, h) = \frac{315}{64\pi h^9}(h^2 - \|x\|^2)^3 \tag{4.1}$$

$$\nabla W_{pressure}(x, h) = -\frac{45}{\pi h^6}\frac{x}{\|x\|}(h - \|x\|)^2 \tag{4.2}$$

$$\nabla^2 W_{viscosity}(x,h) = \frac{45}{\pi h^6}(h - \|x\|) \tag{4.3}$$

Density is computed using the kernel given in Equation 4.1. This kernel is chosen for its balance between accuracy and speed. The gradient of this kernel is not chosen for the pressure force calculation, instead the gradient given in Equation 4.2 is used. This is to prevent particle clustering that would otherwise occur if the gradient of the density kernel is used. The viscosity force calculation also uses a unique kernel. This is because the viscosity force should only dampen velocity, not increase it. This is achieved if the Laplacian of the smoothing kernel is always positive. Because this it not true for the Laplacian of the density or pressure kernel, the Laplacian given in Equation 4.3 is used.

4.2.1.3 Internal Forces

Internal forces are determined as described in Section 2.3.2. Specifically, the forces associated with pressure and viscosity are found using Equation 2.24 and Equation 2.25 respectively.

4.2.1.4 External Forces

Three main forms of external forces may be applied to each particle: gravity, spring, and user interaction. Gravity is applied simply as a constant force in one direction. Spring forces are only applied in simulations utilizing a mass-spring method for strand dynamics, as discussed in Section 2.1.1. User interaction allows forces to be applied in the form of mouse movement. These movements can be interpreted as a global force, affecting all particles, or a local force, affecting particles within a chosen radius of the mouse.

To create the user forces from mouse movement, the location of the curser is recorded at each time step. A vector is then created using the current and previous time step locations. The magnitude of the force vector is made reasonable with a user defined constant multiplier. The user is also given the choice of using a unit vector version of the force vector so that the speed of the simulation does not affect

magnitude. For example, if the time steps occur more frequently, the users curser locations will be recorded more frequently and therefore closer together. The resulting force vector will have a smaller magnitude unless a unit vector is used. This is mainly used to allow consistency across simulation scenarios during testing. A user may prefer the original vector, however, so that the speed to curser movement reflects stronger forces.

4.2.2 Strand Dynamics

In order to simulate a hair strand, each particle needs to be attached to the particles adjacent to it in the strand. Therefore a constraint is placed between adjacent particles so that they are kept at a distance of l_o to each other.

This thesis implements two separate methods of enforcing these constraints: Position Based Dynamics and Mass-Spring. In general, PBD is the more advantageous method; it is stable under smaller time steps and does not exhibit elasticity. In the case of this thesis, however, instability is somewhat desirable in order to test the effectiveness of Smarticles.

4.2.2.1 Position Based Dynamics

The Dynamic Follow-The-Leader algorithm given in [21] enforces hair strand structure and dynamics using a position based approach. After new particle velocities and positions are updated through time integration to reflect the forces produced from the SPH portion of the simulation, DFTL corrects any positions that violate hair strand constraints and updates the velocities to reflect the change in position. Equation 2.12 and Equation 2.13 are used to make the position and velocity changes, iterating through all the particles in the order dictated by Follow-The-Leader.

4.2.2.2 Mass-Spring

Spring forces are used to maintain hair strand structure by placing an additional external force on each particle. Constraints between adjacent particles in a hair

strand are therefore enforced as part of the SPH portion of the simulation.

As described in Section 1.3, the magnitude of the applied spring force is the product of displacement, X, and the spring constant, k. In the case of this hair simulation, displacement is calculated as the difference between the true distance of the adjacent particle and the rest distance, l_o. A force vector is then found as the vector pointing towards the adjacent particle position with magnitude kX.

4.3 Time Integration

Once particle accelerations are found using Equation 2.19, the Leap-Frog integration scheme is used to find particle velocities and positions using Equation 2.10 and Equation 2.11 respectively. This method was chosen due to its advantages over other explicit method described in Section 2.1.5

The Leap-Frog method evaluates particle velocities at the midpoint of each time step interval, however, some force calculations, such as the viscosity force, require velocities occurring on the interval. Therefore, Equation 4.4 is used, which finds a midpoint approximation of the velocity at t.

$$v(t) = \frac{v(t - \frac{1}{2}\Delta t) + v(t + \frac{1}{2}\Delta t)}{2} \tag{4.4}$$

Chapter 5

SMARTICLES ALGORITHM

The main contribution of this thesis is the introduction of Smarticles, a method aimed at reducing the negative visual artifacts that result from error and instability produced by explicit integration techniques. The basic concept of Smarticles includes looking each particle individually, analyzing its behavior, determining if it is unstable, and correcting it if necessary.

5.1 Overview

Smarticles consists of two main stages: identification and correction. As such, identification and correction methods do not rely on each other allowing them to be interchanged. This thesis implements two identification methods referred to as Past Behavior Comparison (PBC) and Strand Neighbor Behavior Comparison (SNBC). One correction method is used, which uses velocity dampening. These methods are further discussed in the following sections.

The Smarticles methods are applied at the end of each time step, after final particle position and velocity updates have been preformed. Either one or both of the two identification methods can be used. An overview of a single time step of the complete hair simulation including the Smarticles portion is given in Figure 5.1.

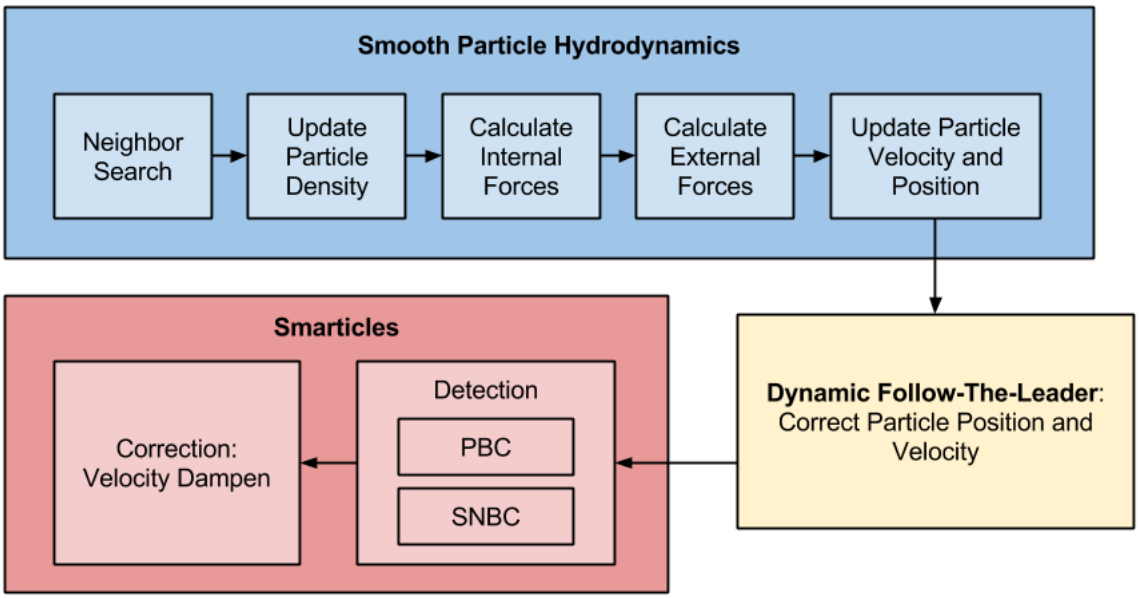

Figure 5.1: Complete hair simulation process for a single time step

5.2 Past Behavior Comparison (PBC)

Oscillation is a common and apparent visual indication of excess error. When using explicit integration with large time steps, it is common for resting points to be over shot leading to oscillation that continues to increase in energy. PBC attempts to identify oscillating particles by comparing a particle's past behavior to current behavior. If a particle moves in one direction and immediately after moves in an opposite direction, it is suspected to be oscillating and therefore unstable.

PBC is described in detail in Algorithm 1. For each particle p, two vectors are found: one defined as the position change resulting from the current time step and one defined as the position change that resulted from the last time step. This is given in Algorithm 1 as $p.curChange$ and $p.lastChange$ respectively. The Dot Product of these two vectors is then found and used to determine if the angle between them suggests they are pointing in opposing directions. If so, the particle is considered to be oscillating and therefore showing signs of instability. The particle is then marked

to be corrected in the next stage of Smarticles.

Rather than requiring the vectors to be in complete opposite directions, a range of angles are accepted. The range of 90 to 270 degrees was chosen due to the significant direction change it suggests in only a single time step. (maybe put ending sentence here and remove next paragraph to other section)

In general, Smarticles biases towards over identification. It is considered much less desirable for a particle to exhibit instability than be corrected when unnecessary. However, the benefit of this bias depends heavily on the correction method used. Over correction may create more error, negative visual artifacts, or waist computation time. The correction method given in this thesis does exhibit some of these negative consequences which are further discussed in SECTION.

```
for each particle p do
    // update position change vectors
    p.lastChange ← p.curChange;
    p.curChange ← p.curPosition − p.lastPosition;

    // angle between vectors is between 90 and 270 degrees
    if dotProduct(p.lastChange,p.curChange) < 0 then
        particle p is unstable;
    else
end
```

Algorithm 1: Smarticles identification method: Past Behavior Comparison

Originally, a longer history of a particle's past behavior was captured and stored in a queue. The queue held the position change vector of a particle over several time steps that were then compared in a similar manner as shown in Algorithm 1. This way a particle would be identified as unstable only after exhibiting concerning behavior repeatedly, thus reducing the likelihood of false positives.

Preliminary tests were preformed to help determine the optimal queue length. These tests revealed that indeed bad particles were identified more accurately with larger queue sizes, however, larger queue sizes also increased the amount of time before a misbehaving particle was identified and corrected. As a result, the particle clearly exhibited instability before Smarticles took action. The need to identify unstable

particles immediately made this queue method unsuitable. Instead, Past Behavior Comparison (PBC) only looks at the position change of the last and current time step, therefore requiring only a single time step delay.

5.3 Strand Neighbor Behavior Comparison

When particles are coupled together, such as in a hair simulation, they are expected to generally behave similar to their neighbors. Therefore, if a particle is moving radically differently as compared to its direct neighbor in the hair strand, it is likely the particle is experiencing significant error. In a mass-spring model for example, particles often oscillate along the strand, propagating error to their neighbors. Stand Neighbor Behavior Comparison (SNBC) attempts to detect this type of oscillation by comparing the behavior of a particle and its immediate hair strand neighbor.

Both SNBC and PBC attempt to identify oscillation caused by error, therefore they have the same basic steps. The main difference is the vectors being compared. SNBC, shown in Algorithm 2, only requires the current position change vector, $p.curChange$ (defined the same as in PBC). This vector is than compared to the neighboring particle's position change vector. In a hair strand, a particle typically has two neighbors, one closer to the root and one closer to the loose end. Here those neighbors are referred to as the particle's *parent* and *child* respectively. Unless the

particle has no child, the particle's parent vector is used.

```
for each particle p do
    p.curChange ← p.curPosition − p.lastPosition;
end
for each particle p do
    if particle has no child then
        p.neighborChange = p.parent.curChange;
    else
        p.neighborChange = p.child.curChange;
    end
    // angle between vectors is between 90 and 270 degrees
    if dotProduct(p.curChange, p.neighborChange) < 0 then
        particle p is unstable;
    else
end
```

Algorithm 2: Smarticles identification method: Strand Neighbor Behavior Comparison

5.4 Smarticles Correction

Once particles have been labelled as stable or unstable by PBC or NSBC, the correction stage of Smarticles is preformed. This thesis implements a simple dampening method which reduces the velocity of unstable particles in hopes to counteract the energy gain caused by oscillation.

Each particle has the attribute *dampen*, which contains a floating point number between 0 and 1. During time integration, the updated velocity is multiplied by the *dampen* value before the position has been updated. This way the velocity is reduced if *dampen* is less than 1, and the position update is affected by this new velocity. Ideally, this stops the particle from exhibiting unstable behavior and reduces the potential for it to amplify.

As seen in Algorithm 3, a particle's *dampen* attribute is changed according to its

current stability. During a time step in which a particle is labeled unstable, the velocity dampening is increased based off of the predefined dampening rate, *dampenRate*. Any time steps in which a particle is labeled as stable again, the velocity dampening is reduced until *dampen* returns to a value of 1.

Changing dampening in this manner forces a particle to exhibit stable behavior for more than a single time step before its movement is no longer dampened. This helps to ensure a particle is truly stable before it is allowed to move freely again.

```
if particle p is unstable then
    p.dampen ← p.dampen ∗ dampenRate;
else
    if p.dampen < 1.0 then
        p.dampen ← p.dampen ∗ undampenRate;
        if p.dampen > 1.0 then
            p.dampen ← 1.0;
        else
    else
end
```

Algorithm 3: Smarticles Correction

Chapter 6

RESULTS

This section evaluates Smarticle's ability to reduce instability and allow larger time step sizes without interfering with visual accuracy. This is done using discrete tests that apply various external forces to hair simulations at increasing time step sizes. Smarticle's effectiveness is tested for both the hair simulation method that uses spring forces and the method that uses position based dynamics to constrain particles in a hair strand. Video recordings of each test are made to keep a record and allow further analysis and comparison.

6.1 Test Cases

The goal of Smarticles is to allow a simulation to utilize a larger time step without showing signs of excess error or instability. Therefore, testing is designed to find the highest allowable time step for a simulation. This is done for a control simulation containing no Smarticles, a Smarticles simulation utilizing PBC, a Smarticles simulation utilizing SNBC, and a Smarticles simulation utilizing both PBC and SNBC. Ideally the highest time step for each of these would be found for both the Mass-Spring and PBD version of the hair simulation, however, only the PBD version showed a difference in this respect to the control, as further discussed in Section 6.2.2.

Ten test cases were created, each exposing the hair simulation to various external forces. These forces were applied to the simulation in the form of various mouse movements. The mouse movements were user provided and prerecorded to allow repeatability and consistency across tests. Section 4.2.1.4 further describes how mouse movements are interpreted by the simulation as external forces. For each test, the

time step was incrementally increase until clear signs of error were seen. Once this occurred, the last visually accurate time step was recorded.

6.2 Results and Analysis

6.2.1 Hair Simulation using PBD

Smarticles allowed at least a minor increase in the allowable time step size for the ten test cases used, shown in Table **??**. In addition, PBC Smarticles significantly reduced the visible error at much larger time steps, increasing in instability at a much slower rate than the control. Unfortunately, this is not particularly useful given only a small amount of visible error is unacceptable.

	Avg Delta t	Avg Increase
Control	34.8 ms	
PBC	40.5 ms	16.01%
SNBC	35.3 ms	4.82%
Both	41.3 ms	18.62%

Table 6.1: Time step sizes and percent increases as compared to the control averaged over 10 test cases.

PBC Smarticles provided a much larger increase as compared to SNBC Smarticles. This can largely be explained by the effectiveness of the PBD method, DFTL, to constrain each particle in a hair strand. Each particle is kept at the correct distance from its neighbor within the strand even when explicit time integration techniques causes significant error. This is because PBD directly manipulates positions making it a particularly stable method in this case. As a result, it is unlikely for a particle to ever move significantly differently than its strand neighbor, which is what SNBC Smarticles uses to identifying unstable particles. Instead, SNBC Smarticles was designed to identify the oscillation that often occurs between particles within a strand constrained by spring forces.

Unfortunately, the more effective method, PBC Smarticles, may only be successful because it is suppressing the SPH portion of the simulation. Unlike PBD, SPH is a force based method that does relays on integration methods to update position values. For this reason, the SPH portion of the simulation becomes the source of instability that prevents large time steps. Smarticles corrects the instability by dampening particle velocities. Because PBD directly changes the position of the particles, Smarticles does not significantly impact the PBD portion of the simulation. Instead, the dampening imposed by Smarticles has a much larger impact on the SPH portion of the simulation. This is concerning because more Smarticle corrections creates more dampening which eventually results in a purely PBD simulation with the majority of SPH suppressed. In this particular implementation, the affect of this is seen as a lose of hair on hair interactions. In a more realistic and practical application, this would be a more apparent and significant concern due to the need of handling collisions such as with the air, body, or other objects in the scene.

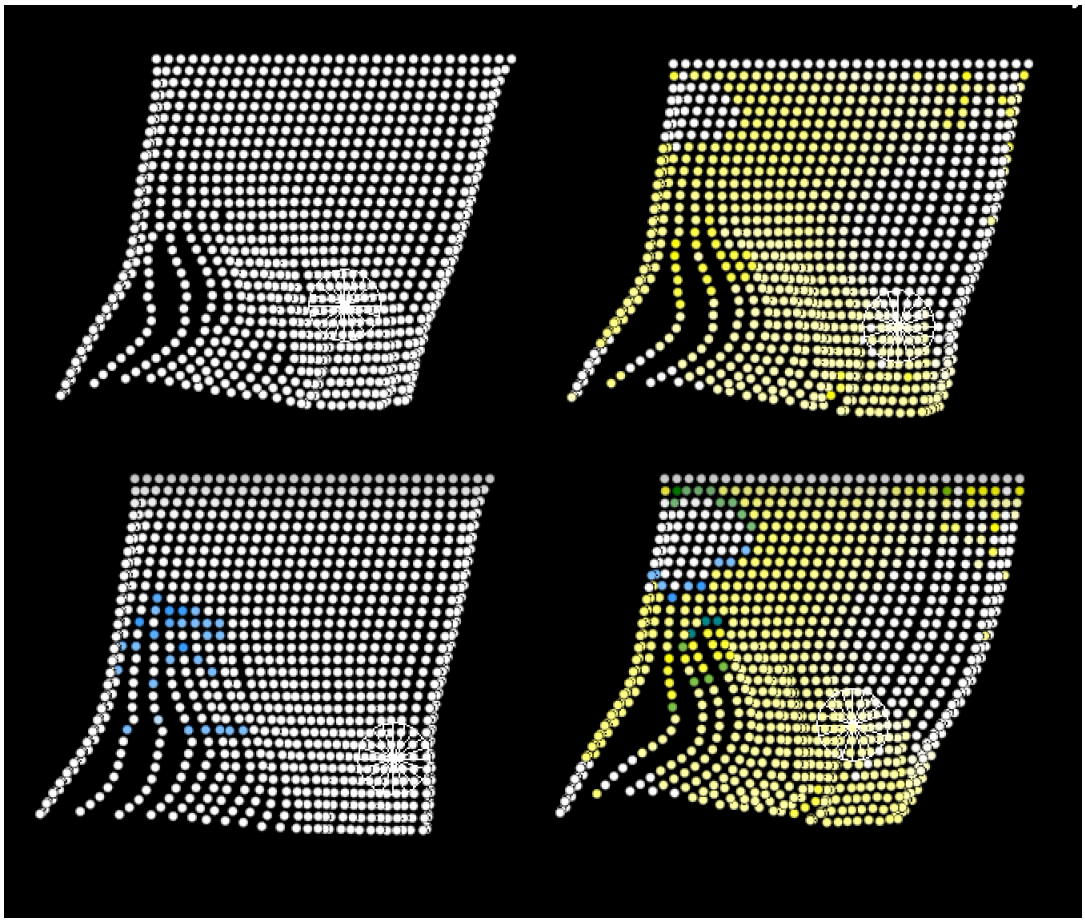

Figure 6.1: A snapshot of a test at time step size equal to 43.0ms for the control (upper left), PBC Smarticles (upper right), SNBC Smarticles (lower left), and both Smarticles (lower right).

In Figure 6.1, a snap shot of a test case can be seen. Although difficult to see in a still image, the control and SNBC Smarticles simulations begin to show signs of instability where the hair strands are separating. Particles identified as unstable by PBC Smarticles are colored yellow, by SNBC Smarticles are colored blue, and by both Smarticles are colored green. A higher saturation of color is used to show stronger velocity dampening.

6.2.2 Hair Simulation using Mass-Springs

Both PBC and SNBC Smarticles worked well at identifying the unstable particles, however, the Smarticles correction method was unable to correct all particles well

enough to allow for larger time steps. In addition, when Smarticles attempted to improve stability, visual accuracy was often negatively affected.

This failure can largely be attributed to the ineffectiveness of using velocity dampening to correct undesired particle oscillation along hair strands. This type of oscillation is caused by the spring forces over correcting particle positions when attempting to maintain the appropriate distance between neighboring hair strand particles. If the time step is large enough, this over correction causes the particle to oscillate between positions too close and too far from the adjacent particle. The amplitude of oscillation increases causing an energy gain and propagates the affect along the hair strand.

Dampening the velocity of these particles can help reduce the energy gain and the amount of propagation, however oscillation is rarely completely stopped without affecting hair structure. By dampening the velocity, particles do not return to a correct place within the hair strand as quickly. This results in over elongation of hair strands, a property that is also occurs when weaker spring forces are used.

In the PBD implementation, Smarticles is more affective due to the type of oscillation and situations in which it occurs. As discussed in Section 6.2.1, PBD prevents oscillation within the hair strand. Instead, instability is more often seen in isolated areas where external forces are most extreme. Dampening particle velocity in these situations often just has the affect of reducing the magnitude of the external force, having little affect on the apparent accuracy of hair structure and movement.

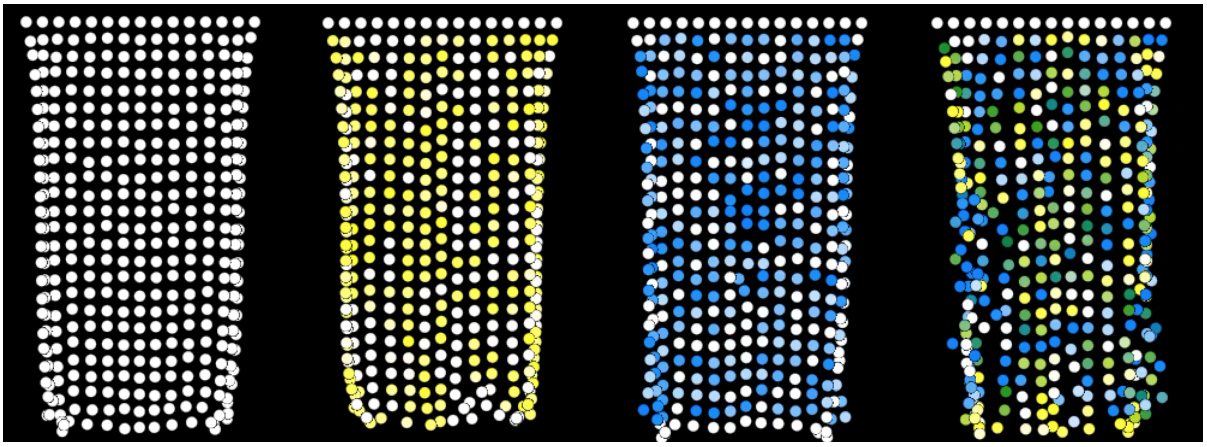

Figure 6.2: A snapshot of a test at time step size equal to 4.0ms for the control (left), PBC Smarticles (mid-left), SNBC Smarticles (mid-right), and both Smarticles (right). Difficult to see in the still image, all simulations are experiencing oscillation. SNBC Smarticles begins showing visual inaccuracies.

6.3 Future Work

The weakest aspect of Smarticles is likely its correction method. Velocity dampening gave a simple solution that showed some positive results, however, it often took too long to correct an unstable particle. Also, it likely dampened dynamic aspects of the simulation that are visually desirable or required. The use of Smarticles may be advantageous if more accurate correction methods are developed. This thesis suggests several areas to be explored for improving the effectiveness and practicality of Smarticles.

For the hair simulation using PBD where the SPH portion is what limits the time step size, a dynamic smoothing length could be used. Thus higher resolutions of fluid properties can be calculated where instability occurs, as indicated by the Smarticles identification methods.

Multiple time stepping integrators as described in [28] could be used to reduce integration error when necessary. These integrators allow different time steps sizes to be used during a simulation. Smarticles could be used to indicate when particle instability is high and time steps sizes need to be reduced.

A similar approach could be made using implicit integration. Smarticles could be

used to indicate when explicit methods are sufficiently accurate. If more accuracy is need, integration can be switched to an implicit method. This shows some similarities to hybrid implicit-explicit integration techniques.

In the PBD portion of the simulation, the FTL algorithm is less applicable to two-dimensional environment due to the loss of many potential particle correction positions. In this simulation, the FTL algorithm has a circles worth of potential points, whereas in three-dimensional coordinate system there is a sphere of potential positions, allowing for many more reasonable positions. When Smarticles applies velocity dampening, the limited number of potential positions may prevent the change in velocity to have a significant affect. It is worth considering that as a result Smarticles would have more success in a 3D environment.

6.3.1 Evaluation Improvements

The goal of Smarticles is to reduce the appearance of error and instability to a end viewer, rather than reduce the actual mathematical error. As a result, it is difficult to collect concrete, quantitative results in order to evaluate the effectiveness of Smarticles. For this reason, the evaluation method used by this thesis collects results for the most practical application of Smarticles: allowing a larger time step while showing no visible error or instability. This allows concrete results in the form of time step sizes, however, this does not fully capture the effect of Smarticles. In addition, this only evaluates Smarticles as a whole. More preferably, the identification and correction portion of Smarticles could be evaluated separately.

A more informative evaluation method would involve knowing the visual stability of each particle at each time step for all the tests. This way the actually stability of each particle could be compared to the stability determined by Smarticles. This would give a percent accuracy of Smarticles in identifying unstable particles for each test. In addition, the effectiveness of the correction could be evaluated similarly. A major advantage of this evaluation method is the ability to evaluate the two stages of Smarticles separately. The largest down side of this evaluation method is the tedious task of labeling each particle as stable or unstable by hand.

Another consideration not adequately tested in this thesis is whether the SPH

portion of the simulation is being suppressed by the Smarticles correction method. One method for testing this could involve placing obstacles within the scene so that SPH is used to resolve the collisions between the hair and the obstacle. If the collisions are not handled and instead the hair simply passed through the object, it can be concluded the SPH portion is being suppressed.

6.4 Conclusion

This thesis presented, Smarticles which attempted to improve stability of explicit integration methods so that larger time steps could be used. The effectiveness of Smarticles was tested on two separate physically based simulations of hair, one consisting of PBD and the other consisting of spring forces. For both simulations, the collective dynamics of hair was modeled using SPH. Regardless of Smarticles, the simulation utilizing PBD clearly provided a more practical method of hair simulation. It supported much larger time step sizes and provided an appearance of stiff hair strands.

Smarticles was designed to have two discrete steps performed for each particle: detection and correction of instability. Two detection methods were implemented: Past Behavior Comparison (PBC) and Strand Neighbor Behavior Comparison (SNBC). One correction method was implemented that uses velocity dampening.

Smarticles only demonstrated the ability to support a larger time step in the PBD hair simulation, giving the largest increase when both Smarticles identification methods were used together. This achieved an increase in maximum stable time step size by 18.62% as compared to a control hair simulation using no Smarticles. Unfortunately, Smarticles was unable to support a larger time step in the hair simulation using spring forces without significantly affecting visual accuracy. In both hair simulations, Smarticles significantly reduced (but did not eliminate) visible instability at high time steps.

The weakest point of Smarticles is likely the correction method. Velocity dampening rarely provided a sufficient means of correcting oscillations caused by spring forces. In addition, dampening disrupted the desired stiffness of spring forces and

likely other aspects of the simulation. Although Smarticles did not show negative visual affects in the PBD hair simulation, it is likely the dampening reduced dynamic aspects of the simulation that would be required in more complex simulations. For example, the dampening may remove SPH affects preventing the desired hair on hair, hair on air, and hair on body interactions. Therefore, in order for Smarticles to be a practical method for improving stability and supporting larger time steps when using explicit integration methods, an improved correction method would need to be developed.

BIBLIOGRAPHY

[1] *Comparison of several parallel API for cloth modelling on modern GPUs*. ACM, 2010.

[2] ACM. *A Robust Method for Real-Time Thread Simulation*, 2007.

[3] K.-i. Anjyo, Y. Usami, and T. Kurihara. A simple method for extracting the natural beauty of hair. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, pages 111–120. ACM, 1992.

[4] Y. Bando, B.-y. Chen, and T. Nishita. Animating hair with loosely connected particles. *COMPUTER GRAPHICS FORUM (EUROGRAPHICS 2003 PROC.) 22*, 2003.

[5] D. Baraff and A. Witkin. Large steps in cloth simulation. pages 43–54. ACM, 1998.

[6] E. Boxerman and U. Ascher. Decomposing cloth. pages 153–161. Eurographics Association Aire-la-Ville, 2004.

[7] J. Brown, J.-C. Latombe, and K. Montgomery. Real-time knot-tying simulation. *The Visual Computer*, 20(2-3):165–179, 2004.

[8] B. Choe and H.-S. Ko. A statistical wisp model and pseudophysical approaches for interactive hairstyle generation. *IEEE Transactions on Visualization and Computer Graphics*, 11(2):160–170, 2005.

[9] M. Crouzeix. Une méthode multipas implicite-explicite pour l'approximation des équations d'évolution paraboliques. *Numerische Mathematik*, 35(3):257–276, 1980.

[10] M. Desbrun, P. Schroder, and A. Barr. Interactive animation of structured deformable objects. pages 1–8. Proceedings of the 1999 conference on Graphics interface '99, Morgan Kaufmann Publishers Inc., 1999.

[11] R. Featherstone. *Robot Dynamics Algorithms*. Kluwer Academic Publishers, 1987.

[12] S. Hadap, editor. *Oriented Strands: Dynamics of Stiff Multi-body System*. Eurographics Association, 2006.

[13] S. Hadap, M.-P. Cani, M. Lin, T.-Y. Kim, F. Bertails, S. Marschner, K. Ward, and Z. Kacic-Alesic, editors. *Strands and Hair: Modeling, Animation, and Rendering*, volume 33, 2007.

[14] S. Hadap and N. Magnenat-Thalmann. Modeling dynamic hair as a continuum modeling dynamic hair as a continuum. *Computer Graphics Forum*, 20(3):329–338, 2001.

[15] Y.-M. Kang, J.-H. Choi, H.-G. Cho, and D.-H. Lee. An efficient animation of wrinkled cloth with approximate implicit integration. *The Visual Computer*, 17(3):147–157, April 2001.

[16] Y.-M. Kang, J.-H. Choi, H.-G. Cho, and C.-J. Park. Fast and stable animation of cloth with an approximated implicit method. *Computer Graphics International*, pages 247 – 255, 2000.

[17] M. Kelager. Lagrangian fluid dynamics using smoothed particle hydrodynamics. Department of Computer Science, University of Copenhagen Department of Computer Science, January 2006.

[18] H.-Y. Liu, Y.-Q. Zhong, and S.-Y. Wang. Numerical stability of integration methods used in cloth simulation. 3(2), 2010.

[19] A. McAdams, A. Selle, K. Ward, E. Sifakis, and J. Teran. Detail preserving continuum simulation of straight hair. ACM SIGGRAPH, 2009.

[20] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, April 2007.

[21] M. Müller, T. Y. Kim, and N. Chentanez. Fast simulation of inextensible hair and fur. volume Virtual Reality Interactions and Physical Simulations, pages 39–44. Erographics Association, December 2012.

[22] L. Petrovic, M. Henne, and J. Anderson. Volumetric methods for simulation and rendering of hair.

[23] E. Plante, M.-P. Cani, and P. Poulin. Capturing the complexity of hair motion. *Graphical Models*, 64(1):40–58, 2002.

[24] Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation. *A Practical Model for Hair Mutual Interactions*, 2002.

[25] X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. *In Graphics Interface*, pages 147–154, 1996.

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

[27] A. Selle, M. Lentine, and R. Fedkiw. A mass spring model for hair simulation. *ACM Trans. Graph.*, 27(3):64:1–64:11, August 2008.

[28] A. Stern and E. Grinspun. Implicit-explicit variational integration of highly oscillatory problems. *Multiscale Modeling & Simulation*, 7(4):1779–1794, 2009.

[29] D. Terzopoulos, J. Platt, and K. Fleischer. Elastically deformable models. volume 21, pages 205–214. SIGGRAPH, ACM, July 1987.

[30] T. Vassilev, B. Spanlang, and Y. Chrysanthou. Fast cloth animation on walking avatars. *Computer Graphics Forum*, 20(3):260–267, 2001.

[31] P. Volino, M. Courchesne, and N. Magnenat-Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. *SIGGRAPH '95 Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 137–144, 1995.

[32] P. Volino and N. Magnenat-Thalmann. Comparing efficiency of integration methods comparing efficiency of integration methods for cloth simulation.

[33] K. Ward, F. Bertails, T. yong Kim, S. R. Marschner, M. paule Cani, and M. C. Lin. A survey on hair modeling: styling, simulation, and rendering. In *IEEE TRANSACTION ON VISUALIZATION AND COMPUTER GRAPHICS*, pages 213–234, 2006.