AN INVESTIGATION INTO THE PERFORMANCE OF INTEGRATION METHODOLOGIES FOR REAL TIME MASS-SPRING CLOTH SIMULATIONS

by

CHRISTOPHER PHILLIPS
ID: 15022229

A dissertation submitted in partial fulfilment of the
requirements for the award of

MASTER OF SCIENCE IN COMPUTER GAMES PROGRAMMING

September 2016

Department of Computing
Staffordshire University
Stafford ST18 0AD

Supervised by: Christopher McCreadie

I declare that this dissertation is my own work and that the work of others is acknowledged and indicated by explicit references.

Christopher Phillips
September 2016

Abstract

There has been much research into teaching computers to play games effectively, resulting in programs which are capable of beating the best human players at chess and other board games. These programs typically make use of strongly defined rules to provide their intelligence. This project suggests a different technique; applying stochastic optimisation techniques to effectively learn the strategic rules for playing Connect 4.

Three popular algorithms, a genetic algorithm, evolution strategies and particle swarm optimisation, were implemented to play Connect 4 in The Arena, a Java based framework providing the implementation of the game. Following a training period to learn the rules of the game, the playing performance of these algorithms was tested. The test results showed limited success, with only one of the algorithms able to play relatively successfully. The time constraints for the development of this project may be the reason for the poor performance of the algorithms, therefore more research and testing should be considered.

Acknowledgements

With thanks to Dr James Heather, for his help during the development of this project, and Dr Johann A. Briffa, for no doubt saving the author countless arguments with word processing software whilst trying to format this dissertation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The visual fidelity of video games has increased dramatically in recent years largely due to developments in discrete graphics hardware, or graphics processing units (GPUs). As a result, there is a need for realistic cloth in order to sell the appearance of video game characters and scenes. It is essential that the cloth should react to dynamic behaviour, such as a character moving or the wind blowing, and it must be animated in real time, i.e. at a minimum of 30 frames per second (FPS). These requirements render many cloth simulation techniques inappropriate, as they are too computationally expensive to be run in real time. The Mass-Spring model is one technique that is appropriate for use in video games, and is the most popular method for handling cloth for games. Whilst not necessarily providing 100% accuracy, Mass-Spring models result in visually pleasing animations, good enough for games, at, most importantly, real time frame rates.

Mass-Spring models use forces, calculated using Newtonian mechanics, to animate the cloth. This results in a series of differential equations that must be approximated by a numerical integrator at discrete time intervals. There are many different integrators that can be used and the choice of integrator can directly affect the performance of the simulation. Some integrators must necessarily use small time intervals in order to maintain the stability of the cloth, and this increases the frequency of the integration calculations thus reducing performance.

This project will investigate the effect different integration methods have on a real time cloth simulation using the Mass-Spring model.

1.1 Project Aims

The aim of this project is to investigate the performance effects of different integration methods on real time cloth simulation using the Mass-Spring model.

Based on this aim, the objectives of the proposed project are as follows:

- To research cloth simulation and numerical integration techniques
- To implement the Mass-Spring model for cloth simulation and several integrators, including explicit Euler
- To investigate the performance effects of the implemented integration methods on the simulation

1.2 Hypotheses

Volino and Magnenat-Thalmann (2001) were the first to investigate the performance impact of different integrators on cloth simulation. Their results, however, are extremely outdated, due to the vast increase in CPU power since 2001; from a 200MHz workstation CPU to the 3GHz+ CPUs available in modern desktops and laptops. Later work by Wang, Hu, and Zhuang (2009) also investigated the impact of different integrators, and found that integrator choice still has an impact on simulation performance. However they make no reference to the hardware their experiments were run on.
As a result, two hypotheses are proposed for this project, a null and an alternative.

1.2.1 Null Hypothesis

The null hypothesis is that all integration methods result in a real time cloth simulation when running on modern hardware.
This was chosen because Wang, Hu, and Zhuang (ibid.) make no reference to the hardware used, and the simulation developed will be run on an Intel I7 4770K at 4.2GHz, so it may be the case that there are no performance concerns with this modern hardware.

1.2.2 Alternative Hypothesis

The alternate hypothesis is that some integration methods are prohibitively expensive for real time cloth simulation and other methods provide better performance.

Recently, some researchers, such as Zeller (2005) and Tang et al. (2013), have proposed GPU accelerated approaches to Mass-Spring models which suggests that performance is still a consideration. Third party physics engines, such as PhysX® also use GPU accelerated models (Kim 2011) again suggesting performance of Mass-Spring models are still a concern.

1.3 Report Structure

This remainder of this report will be structured as follows:

- Chapter 2: literature review. An overview of cloth simulation techniques will be given and the Mass-Spring models and some numerical integrators described in detail
- Chapter 3: project plane. This chapter will describe the development and testing plan of the project

Chapter 2

Literature Review

2.1 Cloth Simulation

The simulation of cloth is a relatively old and well studied field with applications in many different areas, including, but not limited to:

- Virtual Garment Design
- Virtual Fitting Rooms
- Films
- Video Games

Different use cases require different things from the cloth simulation. For example, in virtual garment design, the physical accuracy of the simulation is paramount whereas cloth simulation for video games prioritises real-time simulation, sacrificing accuracy. Hence, many different models for cloth simulation have been proposed.
Since this project is concerned with the real-time simulation of cloth, only those models appropriate for real-time simulations have been studied in detail. However, a brief overview of other techniques will be provided. For a more detailed overview of cloth simulation techniques, see Ng and Grimsdale (1996).

2.1.1 Cloth Properties

Cloth has several properties that should be considered for modelling.

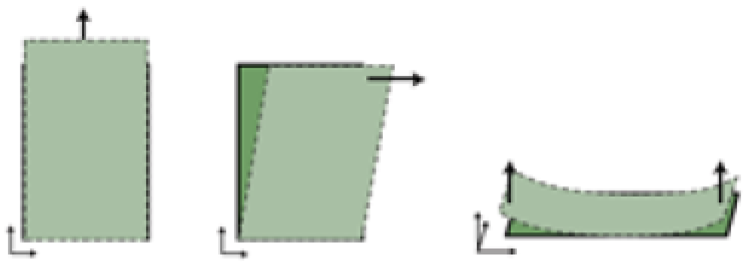


Figure 2.1: Mechanical properties of cloth (*Techniques for Animating Cloth*, p. 1)

2.1.1.1 Mechanical Properties

Cloth has three mechanical properties that control its behaviour; stretching, shearing and bending, fig. 2.1 shows how each property affects the cloth.
Stretching is the displacement of the cloth in either the horizontal or vertical direction. Most cloth has a high resistance to stretching and can typically only be stretched by 10% (*Techniques for Animating Cloth*, p. 1; Provot 1995, p. 4).
Shearing is the displacement of the cloth in a diagonal direction. Again, most cloths have high shearing resistance and this, coupled with high stretch resistance, makes cloth incompressible.
Finally, bending is the overall curvature of the cloth surface. Typically, cloth has low bending resistance and so is easily folded.

2.1.1.2 Visual Properties

The mechanical properties of cloth, discussed above, cause cloth to exhibit two visual properties. These properties arise from the fact that cloth is typically non-elastic, due to stretch and shear resistances, but highly flexible, due to low bend resistance.
Firstly, cloth will drape over objects and secondly the cloth will form many folds and wrinkles. Fig 2.2 demonstrates the visual properties of cloth.

2.1.2 Cloth Models

Techniques for modelling cloth are usually classified as either Geometric or Physically-based, and the choice of which modelling method to use depends on the use-case for the simulation.

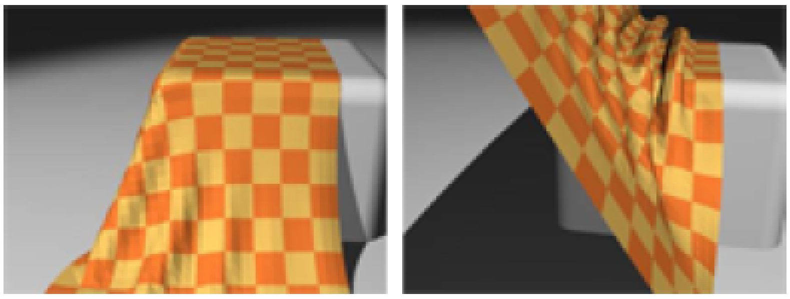


Figure 2.2: Visual properties of cloth (*Techniques for Animating Cloth*, p. 1)

2.1.2.1 Geometric Models

This family of techniques were the first models used to simulate cloth. They model the cloth using geometric equations and are especially good at modelling folds and wrinkles.
Weil was the first to propose a geometric model in 1986 and uses catenary curves to model the drape and folds of a hanging cloth. Following on from Weil, a number of other geometric models were proposed (see Ng and Grimsdale (1996) for more information).
All geometric techniques focus on simulating the appearance of cloth, rather than the physical properties. As such, geometric models are typically more computationally efficient than physically-based models, as there is no need to solve a series of complex equations. However, geometric techniques are unable to accurately simulate the motion of cloth, (Mongus et al. 2012, p. 1; Zhang and Yuen 2001, p. 2; Xinrong et al. 2009, pp. 1-2), and so are mostly useful for static cloth simulations.
As such, geometric models have not been considered for this project.

2.1.2.2 Physically-based Models

By contrast, physically-based models are concerned with the accurate modelling of the physical properties of the cloth and can therefore be used to produce realistic animations.
These models typically use a system of partial differential equations (PDE), or other differential equations, to model the cloth. These equations cannot be solved analytically and therefore the system requires discretisation to solve the equations at specific points in space and time. Following discretisation, a physically-based model typically requires the solving of an ordinary differential equation (ODE) of

the form (Baraff and Witkin 1998, p. 1):

$$\ddot{x} = M^{-1}\left(-\frac{\delta E}{\delta x} + F\right)$$

where:

x is a vector representing the geometric state of the system

M is a diagonal matrix representing the mass distribution of the system (2.1)

E is a function of x which yields the internal cloth energy

F is a function of x and $\dot{x}$ which describes other forces

Physically-based models can be classified as either Continuum or Discrete.

2.1.2.2.1 Continuum Models

Continuum models were the first physical models to be proposed. Techniques in this family model cloth as a continuous surface and utilise continuum mechanics to calculate its behaviour; the Lagrange equations are most commonly used.

To discretise the continuous model, a numerical technique, such as a finite element method, is used. This is one of the advantages of continuum methods; they allow the use of a low resolution discretisation without sacrificing the accuracy of the simulation (Wacker, Thomaszewski, and Keckeisen 2005, pp. 4-5).

Another advantage of continuum models are that they are accurate; "they provide accurate models of the material derived directly from mechanical laws and models of material properties" (Magnenat-Thalmann and Thalmann 2004, p. 200).

This accuracy comes at the cost of computational performance, the main disadvantage of these techniques. The accuracy also renders these models inappropriate for use in dynamic simulations; "the formal and analytical description they require for the mechanical behavior of the material cannot easily be altered to represent transitory and non-linear events. Hence, phenomena such as frequent collisions or other highly variable geometrical constraints cannot be conveniently taken into account" (ibid., p. 200).

Hence, continuum models are typically only considered appropriate for static simulations, or simulations where accuracy is paramount. As a result, continuum models have not been considered.

2.1.2.2.2 Discrete Models

According to Choi and Ko (2002, p. 2), "Cloth is not a homogeneous continuum. Therefore modeling fabrics as a continuum and employing FEM or FDM has several potential drawbacks". As a result several discrete, or particle, models have been proposed.

With these techniques the discretisation in space is carried out by modelling the cloth as a discrete mesh, either regular or triangular, of point masses, called particles. This sacrifices some of the accuracy of continuum models, as the accuracy will depend on the number of particles used. However, this loss in accuracy is traded off against better computational performance; physical models are the only models that can be used for dynamic, real-time simulations. The discretisation of the cloth has a direct affect on the performance of the simulation; Volino and Magnenat-Thalmann (2001, p. 5) have shown that simulation time varies cubically with mesh size. Hence, there is a trade off to be made between accuracy and performance when using discrete models, depending on the use-case.

By far the most popular technique is the mass-spring model. This model is popular as it is simple, easy to implement and offers a good balance between accuracy and efficiency. This model is the most common model used for dynamic, real-time simulations and as such, it has been chosen for this project and will now be described in more detail.

2.1.3 Mass-Spring Models

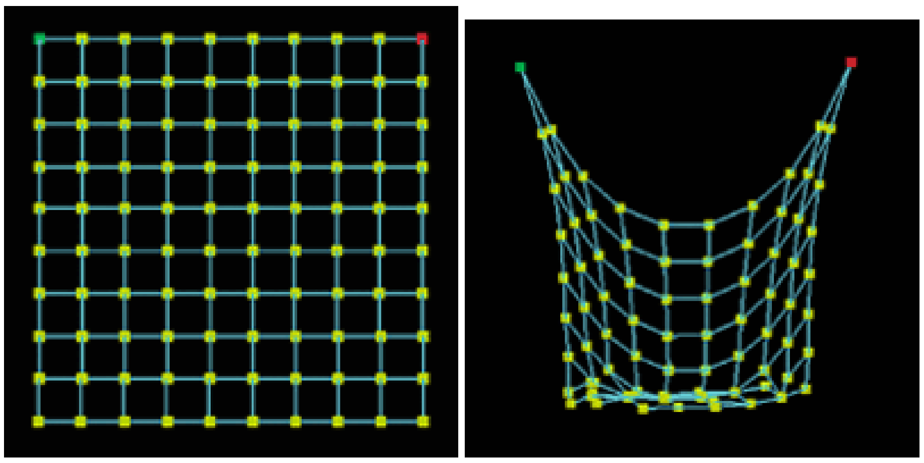
Mass-spring models were first proposed for use in cloth simulation in Provot (1995). Using these models, the cloth is discretised as a 2-dimensional mesh of point masses, either regular or triangular, connected by linear springs.

2.1.3.1 Provot Model

Using the mass-spring model proposed in Provot (ibid.) the cloth is modelled as a regular mesh of point masses. The points are connected together using three different types of springs:

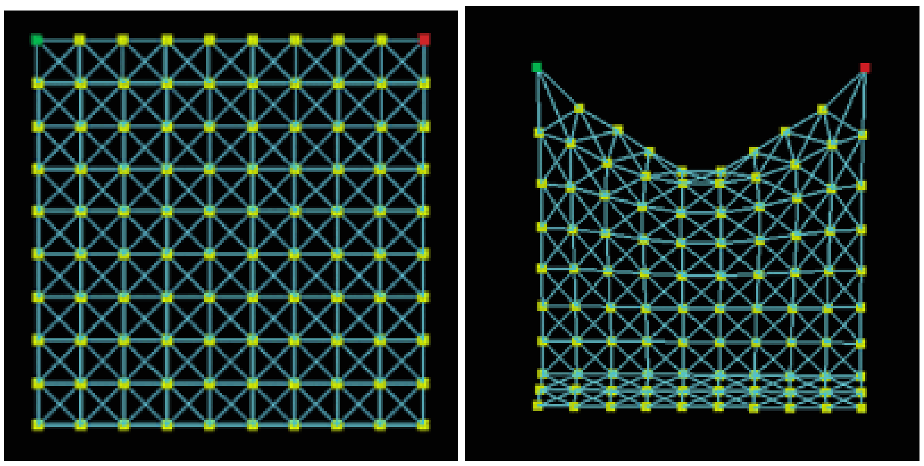
- Structural springs, connecting particle [i, j] to particles [i + 1, j] and [i, j + 1]. These springs resist structural deformations of the cloth, and provide the overall cloth structure. Structural springs are not enough to provide a realistic cloth model. Fig 2.3 shows the results of running a cloth simulation with structural springs only. As can be seen, this does not produce a realistic image

- Shear springs, connecting particle [i, j] to particle [i + 1, j + 1] and particle [i + 1, j] to particle [i, j + 1]. These springs provide shearing resistance for the cloth. By adding shear springs, the realism of the model is improved; Fig 2.4 shows the improved fidelity afforded by shear springs



(a) Initial configuration (b) Result of running the simulation

Figure 2.3: Cloth with structural springs only (Lander 2000b, p. 2)



(a) Initial configuration (b) Result of running the simulation

Figure 2.4: Cloth with structural and shear springs (Lander 2000b, p. 2)

- Bend springs, connecting particle [i, j] to particles [i + 2, j] and [i, j + 2]. These springs model bend resistance

Fig 2.5 shows the arrangement of these springs for a small cloth model.

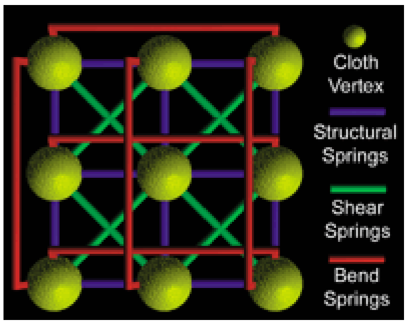


Figure 2.5: Provot cloth model (Lander 2000b, p. 2)

To animate the mesh, forces are applied to the particles and are calculated using Newton's second law:

$$F_{ij} = m_{ij}a_{ij}$$

where:

F_{ij} is the total sum of forces acting on particle ij (2.2)

m_{ij} is the mass of particle ij

a_{ij} is the acceleration of particle ij

The total force acting on a particle is defined as:

$$F_{total} = \Sigma F_{external} + \Sigma F_{internal} \tag{2.3}$$

F$_{external}$ are external forces acting on the mesh, such as gravity and wind.

Gravity is calculated by:

$$F_g = m_{ij}G$$

where: (2.4)

G is the gravitational constant

Wind is calculated by:

$$F_{wind} = w(n_{ij} \bullet \overrightarrow{W})$$

where:

n$_{ij}$ is the surface normal of particle ij (2.5)

$\overrightarrow{W}$ is the wind direction vector

w is the wind constant

F$_{internal}$ are the resultant forces of the springs connecting the mesh.

The spring force is calculated using the Hooke equation (Parent 2012, p. 201):

$$F_{spring} = -k_s(L_c - L_r)\frac{p_2 - p_1}{\| p_2 - p_1 \|} \tag{2.6}$$

where:

k$_s$ is the spring stiffness coefficient

L$_c$ is the current length of the spring

L$_r$ is the initial, or rest, length of the spring

p$_1$ & p$_2$ are the positions of the two connected particles

Using this equation, the cloth will be modelled with pure elastic springs and will oscillate indefinitely. However, as mentioned in 2.1.1.2 and Provot (1995, p. 1), cloth is a non-elastic medium and therefore the model needs to account for the energy lost due to internal friction in the cloth.

This is typically modelled as an extra internal damping force, calculated using (Parent 2012, p. 201):

$$F_{damping} = -k_d(\dot{p}_2 - \dot{p}_1) \bullet \left(\frac{p_2 - p_1}{\| p_2 - p_1 \|}\right)\left(\frac{p_2 - p_1}{\| p_2 - p_1 \|}\right) \tag{2.7}$$

where:

k$_d$ is the spring damping coefficient

$\dot{p}_1$ & $\dot{p}_2$ are the velocities of the two connected particles

2.1.3.2 Choi Ko Model

A different mass-spring model was proposed in Choi and Ko (2002) and aims to improve the buckling behaviour of the cloth, resulting in more realistic draping and wrinkling behaviour.

The cloth is modelled in a similar way to the Provot method, but additional bend springs are added, connecting particle [i, j] to particle [i + 2, j + 2] and particle [i + 2, j] to particle [i, j + 2]; fig 2.6 shows the arrangement of springs.

This model uses an energy-based approach, of the general formula 2.8(Bartels 2014, p. 3), to calculate the forces acting on individual particles.

$$F = \left(\frac{\delta E(S)}{\delta x}, \frac{\delta E(S)}{\delta y}, \frac{\delta E(S)}{\delta z}\right) \tag{2.8}$$

where:

E(S) is an energy function of S, a representation of the cloth's state

Two types of interactions are defined, type 1 and type 2. Type 1 interactions model stretch and shear resistances, the red lines in 2.6, and are represented by a linear spring model. Type 2 interactions model

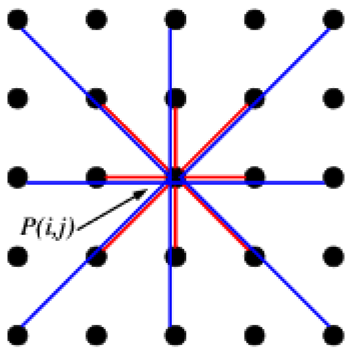


Figure 2.6: Choi Ko cloth model (Choi and Ko 2002, p. 2)

bend forces, the blue lines in 2.6, and helps prevent the so called post-buckling instability problem. The interested reader should see Choi and Ko (2002) for more information on the energy functions for each interaction type.

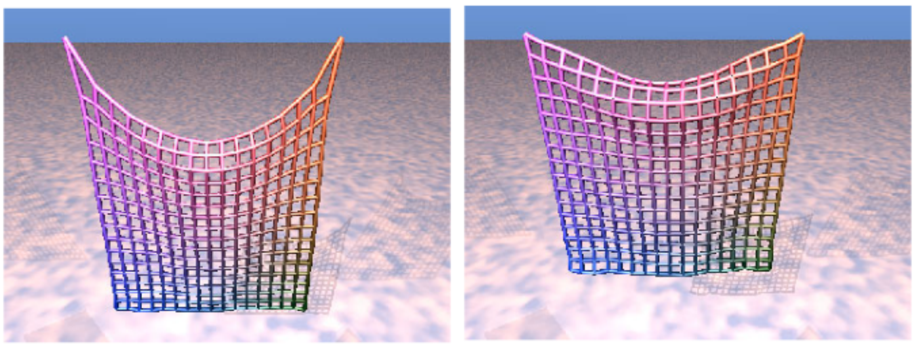
2.1.3.3 Justification of Choice

Mass-spring models are suitable for use in this project as they are efficient and simple to implement; "Mass-spring models are the most efficient as well as the simplest of the cloth models. This method is one of the most popular techniques for simulating cloth, especially when interactive frame rates are required" (Zink and Hardy 2007, p. 2). As this project is concerned with the real time animation of cloth, mass-spring models are therefore the obvious choice.

Mass-spring models are also the most common method of modelling cloth in video games, used in games such as Alan Wake, (Enqvist 2010, p. 2), and Hitman: Codename 47, (Jakobsen 2005, p. 1), as well as commercial physics engines for games, such as Havok® and PhysX®. Again, since this project is concerned with the simulation of cloth for use in a video game, mass-spring models are the logical choice.

In particular, the Provot model was used for this project as the force-based approach requires the solving of a much simpler series of equations than the energy-based approach of the Choi Ko model, and is therefore likely to be more computationally efficient.

One disadvantage of the mass-spring model is that it does not achieve realistic animation of cloth as "mass-spring systems do not model any specific material and are not related to measured properties of real clothes" (Wacker, Thomaszewski, and Keckeisen 2005, p. 3). However, by careful tuning of the spring stiffness coefficients pleasing results can be achieved; Mongus et al. (2012) have shown that,



(a) Super-elastic effect (b) Result of correcting super-elasticity

Figure 2.7: Super-elasticity problems (Provot 1995, pp. 4,6)

with tuning, mass-spring models can reproduce the drape of a cloth with an accuracy of 97%.

Another disadvantage of mass-spring models is what Provot calls 'super-elasticity'; springs are allowed to deform too much, leading to unrealistic looking cloth. This is caused because "the springs are “ideal” and they have unlimited linear deformation rate" (Vassilev, Spanlang, and Chrysanthou 2001, p. 3). To counter this effect, Provot suggests enforcing length constraints on structural and shear springs. First the position of the particles are updated, then the deformation of the springs are calculated. If this deformation value is greater than some threshold, τ_c, then the position of the connected particles are adjusted so the deformation rate equals τ_c. Fig 2.7 shows the super-elasticity problem and the results of employing Provot's corrective method. As can be seen, correcting the super-elasticity results in a much more realistic model.

2.1.4 Numerical Integration for Mass-Spring Models

As mentioned above, physically-based models for cloth simulation require solving a series of differential equations, discretised in space and time. According to Wacker, Thomaszewski, and Keckeisen (2005, p. 5), "since particle systems already represent a discretization in space, only a system of ordinary differential equations has to be solved". For the Mass-Spring model using Newtonian mechanics a series of second order ODEs, of the form 2.9, must be solved (Zink and Hardy 2007, p. 5).

$$\frac{\delta^2 x}{\delta t^2} = M^{-1} F(x,v) \tag{2.9}$$

This can be converted into a coupled series of first order ODEs by separating the position and velocity (ibid., p. 5):

$$\frac{\delta}{\delta t}\begin{pmatrix} x \\ v \end{pmatrix} = \begin{pmatrix} v \\ M^{-1}F(x,v) \end{pmatrix} \tag{2.10}$$

Equations 2.9 and 2.10 cannot be solved analytically, and therefore it is necessary to use a numerical method, or integrator, to approximate them at discrete time intervals. There are many integration methods that could be chosen, typically classified as either explicit or implicit, and the most popular choices will be described now.

2.1.4.1 Explicit Integrators

Most of the early work on physically-based cloth simulation used explicit integrators as they are simple and easy to implement; they only require information about the state of the system at the previous interval to calculate the current state.
The most commonly used explicit integrators are the Runge-Kutta family of integrators.

2.1.4.1.1 Euler

The first order Runge-Kutta integrator, or explicit Euler, was used in Provot (1995) to approximate a Mass-Spring system.
Equation 2.10 is approximated by (Wang, Hu, and Zhuang 2009, p. 3):

$$\begin{aligned} &v_{i+\Delta t} = v_i + \Delta t F(t_i, v_i) \\ &\text{where:} \\ &v_i \text{ is the velocity of a particle at time interval i} \\ &\Delta t \text{ is the time step} \end{aligned} \tag{2.11}$$

When applied to the Provot Mass-Spring model, this gives (Provot 1995, p. 3):

$$\begin{aligned} &a_{i,j}(t+\Delta t) = \frac{1}{m_{i,j}} F_{i,j}(t) \\ &v_{i,j}(t+\Delta t) = v_{i,j}(t) + \Delta t a_{i,j}(t+\Delta t) \\ &x_{i,j}(t+\Delta t) = x_{i,j}(t) + \Delta t v_{i,j}(t+\Delta t) \\ &\text{where:} \\ &\quad a_{i,j}, m_{i,j}, v_{i,j} \text{ and } x_{i,j} \text{ are the acceleration, mass, velocity and position of particle i, j respectively} \end{aligned} \tag{2.12}$$

The explicit Euler method is computationally cheap but can result in numerical instability if too large a time step is used. Mathematically, the explicit Euler method is stable only if the time step is less than the natural period of the system, approximated as $\pi\sqrt{\frac{m}{K}}$, where K is the maximum stiffness in the system. Vassilev, Spanlang, and Chrysanthou (2001, p. 2) found that in fact explicit Euler is only stable for Δt

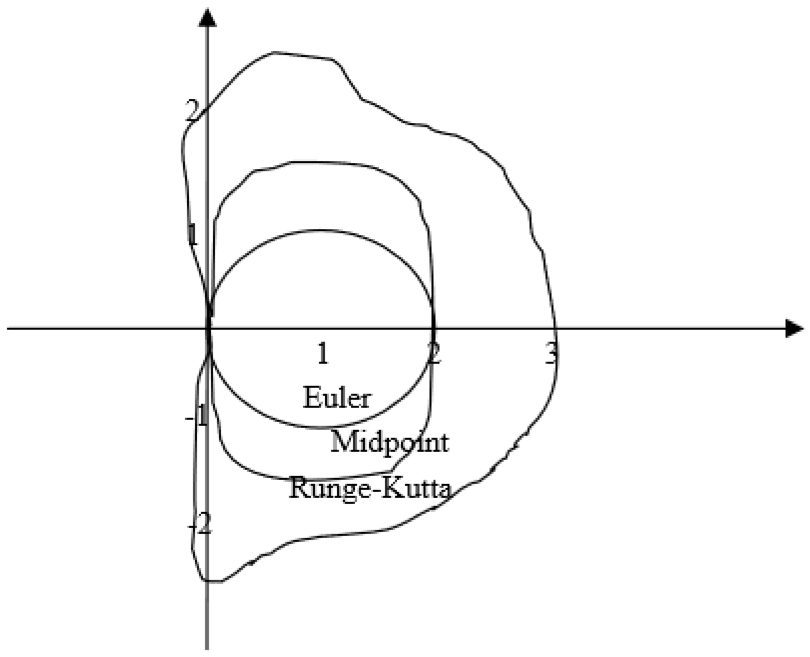


Figure 2.8: Explicit integrator stability regions (Wang, Hu, and Zhuang 2009, p. 4)

values less than $0.4\pi\sqrt{\frac{m}{K}}$.

As cloth generally does not stretch easily, this results in high stiffness in the structural and shear springs, which necessitates the use of a small time step if the explicit Euler integrator is chosen. This can impact the overall performance of the simulation, as while this integrator is cheap, the frequency of calculations is high as a result of the time step limitations.

2.1.4.1.2 Midpoint

The explicit Midpoint integrator is a second order Runge-Kutta method and modifies the Euler integrator to give greater stability.

Equation 2.11 is modified to give the following (Wang, Hu, and Zhuang 2009, p. 3):

$$v_{i+\Delta t} = v_i + \Delta t F\left(t_i + \frac{\Delta t}{2}, v_i + \frac{\Delta t}{2} F(t_i, v_i)\right) \tag{2.13}$$

Since this method requires two derivatives, the computational cost is greater than Euler. However, because the midpoint method affords greater numerical stability (see fig 2.8), a larger time step can be used which increases the overall simulation performance; Wang, Hu, and Zhuang (ibid.) have shown that the midpoint integrator offers close to twice the simulation performance over explicit Euler.

2.1.4.1.3 Fourth order Runge-Kutta

The Fourth order Runge-Kutta integrator offers greater stability using larger time steps over the midpoint method and is formulated as (Wang, Hu, and Zhuang 2009, p. 3):

$$\begin{aligned}
v_{i+\Delta t} &= v_i + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\
k_1 &= F(t_i + v_i) \\
k_2 &= F\left(t_i + \frac{\Delta t}{2}, v_i + \frac{\Delta t}{2}k_1\right) \\
k_3 &= F\left(t_i + \frac{\Delta t}{2}, v_i + \frac{\Delta t}{2}k_2\right) \\
k_4 &= F\left(t_i + \Delta t, v_i + \frac{\Delta t}{2}k_3\right)
\end{aligned} \tag{2.14}$$

This integrator has a significantly higher computational cost than the other methods discussed, however Volino and Magnenat-Thalmann (2001, p. 4) have shown that the Runge-Kutta integrator supports time steps almost six times larger than the midpoint method. Therefore, given that Runge-Kutta is three times as computationally expensive as midpoint, this suggests that this integrator can lead to twice the overall simulation performance. Wang, Hu, and Zhuang (2009, p. 4) have also shown that the fourth order Runge-Kutta integrator offers simulation performance over midpoint and explicit Euler.

2.1.4.1.4 Verlet

Verlet integration is an alternative to the Runge-Kutta family of integrators. It avoids velocity calculations by approximating the velocity of a particle using its previous positions.
A particle's new position is approximated by (Mongus et al. 2012, p. 2):

$$x_{i+\Delta t} = 2x_i - x_{i-\Delta t} + a_{i+\Delta t}\Delta t^2 \tag{2.15}$$

This method is computationally fast and reasonably stable, as "velocity is implicitly given and consequently it is harder for velocity and position to come out of sync" (Jakobsen 2005, p. 1). However it does still suffer from time step issues; figures 11 and 13 in Wacker, Thomaszewski, and Keckeisen (2005, pp. 14-15) show that below a certain threshold Verlet integration is less stable than explicit Euler, but more stable over that threshold.

2.1.4.2 Implicit Integrators

According to Baraff and Witkin (1998, p. 1), "Explicit methods are ill-suited to solving stiff equations because they require many small steps to stably advance the simulation forward in time". Therefore they propose an implicit integrator for use in cloth simulation since implicit integrators are unconditionally stable regardless of step size.

However implicit integrators are computationally more expensive than their explicit equivalents, as "they involve the resolution of a large and sparse linear equation system for each iteration" (Volino, Cordier, and Magnenat-Thalmann 2005, p. 4). Since they are unconditionally stable however, this can be countered by simply using a larger time step. The guaranteed stability of these methods also reduces the accuracy of the simulation, as they introduce inherent numerical damping, which increases as the time step increases (see Volino and Magnenat-Thalmann (2001, p. 4)). As such, there is a balance to be found between performance and accuracy of the simulation with implicit integrators.

The most common implicit integrator is the implicit, or backward, Euler method which will be described now. It should be noted that there are many other implicit integrators available.

2.1.4.2.1 Euler

First proposed for use in cloth simulation in Baraff and Witkin (1998), the implicit, or backward, Euler method is an adaptation of the explicit Euler method.

Equation 2.12 is modified to give (Kang, Choi, and Cho 2000, p. 3):

$$v_i^{t+\Delta t} = v_i^t + F_i^{t+\Delta t}\frac{\Delta t}{m_i} \tag{2.16}$$

$F_i^{t+\Delta t}$ cannot be calculated at the current time step, and so must be approximated as (ibid., p. 3):

$$F^{t+\Delta t} = F^t + \frac{\delta F}{\delta x}\Delta x^{t+\Delta t} \tag{2.17}$$

$\Delta x^{t+\Delta t}$ can be written as $\Delta t(v^t + \Delta v^{t+\Delta t})$ and so eq. 2.17 can be rewritten, giving the series of linear equations (ibid., p. 3):

$$\left(I - \frac{\Delta t^2}{m}\frac{\delta F}{\delta x}\right)\Delta v^{t+\Delta t} = F^t\frac{\Delta t}{m} \tag{2.18}$$

where:

I is the identity matrix

Thus, implicit Euler involves calculating $\Delta v^{t+\Delta t}$ every iteration using:

$$\Delta v^{t+\Delta t} = \left(I - \frac{\Delta t^2}{m}\frac{\delta F}{\delta x}\right)^{-1} F^t\frac{\Delta t}{m} \tag{2.19}$$

$\frac{\delta F}{\delta x}$ is the negated Hessian matrix, denoted as H, and can be approximated as (Kang, Choi, and Cho 2000, p. 3):

$$H_{ij} = \begin{cases} k_{ij} & \text{if } i \neq j \\ -\sum_{i \neq j} k_{ij} & \text{if } i = j \end{cases} \tag{2.20}$$

where:

k_{ij} is the spring stiffness

As a result, the implicit Euler method requires the computation of an n x n matrix each iteration, and thus increasing the size of the mesh greatly impacts the performance of this method.
However, if the stiffness of the springs and mass of the particles are constant throughout the simulation then $\left(I - \frac{\Delta t^2}{m} H\right)^{-1}$ can be precomputed, giving performance gains.
This method has been shown by Volino and Magnenat-Thalmann (2001) to be stable for any time step value, however as the time step increases, the accuracy decreases rapidly over a certain threshold. Using the method described above, it is also not possible to use an adaptive time step or to vary the mass or stiffness of the model as the cost of computing H every iteration is high. As such, many researchers, such as Mesit, Guha, and Chaudhry (2007) and Kang, Choi, and Cho (2000), have presented faster approximation methods for the implicit Euler integrator.

2.1.4.3 Chosen Integration Methods

Four integrations methods were chosen for investigation by this project.

- Explicit Euler. This is the simplest integrator and one of the most popular in the literature but it is also most likely to impact simulation performance due to its reliance on small time steps
- Fourth order Runge-Kutta. Computationally more expensive than explicit Euler, but stable with larger time steps, therefore offering an interesting comparison between computational cost and simulation performance
- Verlet. Chosen as it is an explicit integrator that is not part of the Runge-Kutta family
- Implicit Euler. Unconditionally stable regardless of time step, Chosen to provide a contrast between explicit and implicit integrators

Chapter 3

Design and Implementation

3.1 Development Methodology

3.1.1 Development

Since RAD was the chosen development methodology, the artefact was developed over several cycles.

3.1.1.1 First Cycle

In the first cycle the Mass-Spring model was implemented. Springs were implemented using the Hooke equation (Equation 2.6) to calculate the internal spring force with linear damping as an additional internal force using Equation 2.7. Gravity, implemented as Equation 2.4, was the only external force. Rendering of the cloth was also implemented.
Table 3.1 shows the requirements of the first development cycle.

Before designing this cycle, an initial investigation into the rendering component was carried out, as the choice between DirectX11 or OpenGL would affect the design of the system.
As this project is not concerned with rendering cloth, but simulating it, it was decided that the cloth would be rendered as a mesh of lines, representing the structural and shear springs (as in Figures 2.4 and 2.7). Since the positions of the particles, and therefore vertices, will change over time, a DirectX11 implementation would have to use a dynamic vertex buffer and remap the vertex data every frame. This is not an issue in OpenGL, since it is possible to draw vertices directly with the glVertex3f function. It was unknown by the author what the cost of this remapping would be so simple OpenGL and Di-

Requirement	Type
Must model cloth using the Provot Mass-Spring model	Functional
Must be able to render cloth to the screen	Functional
Must be able to pin particles so they are unaffected by forces	Functional
Must perform in real time	Non-functional

Table 3.1: First cycle requirements

rectX11 implementations were created and their performance compared. For a 500 by 500 mesh, the OpenGL implementation rendered the structural and shear springs at 80FPS and the DirectX11 version at 450FPS. Hence, DirectX11 was chosen as the rendering language. It should be noted that the OpenGL implementation was very naive, due to the author's inexperience, and this is most likely the reason for the poor performance. It is probable that a more robust implementation, using more advanced features, would perform much more closely to the DirectX11 implementation.

The initial design can be seen in Fig A.1; the rendering component, constructors, accessors and mutators have been excluded for brevity.
Since DirectX11 was the rendering language, DirectXMath types were used for the key variables in the Particle class. This gives performance gains over manually implementing a 3-dimensional vector and functions, as the functions that operate on DirectXMath types are compiled into SIMD instructions, allowing the calculations to be completed in less CPU cycles.

During implementation, the initial design had to be adapted as a result of performance concerns.
Firstly the std::vector was replaced with dynamically allocated arrays in the Cloth class. Iterating though the vector to calculate the spring forces proved prohibitively expensive for meshes over a certain size, even calcSpringForce was an empty function; a 100 by 100 mesh was the largest mesh that supported real time frame rates, running at 40FPS. Switching to arrays gave a 4x FPS increase, for the same mesh size, and a 2x increase in the maximum real time mesh size.
Secondly, XMVECTORs were used in place of XMFLOAT3s in the Particle class. XMFLOAT3s must be converted into XMVECTORs using XMLoadFloat3 in order to be able to use the DirectXMath vector functions. Therefore, the position and velocity of every particle had to be converted every frame in order to implement Equations 2.6 and 2.7. Through profiling, it was found that this conversion was the most expensive part of calcSpringForce and so XMVECTORs were used instead, giving performance gains of 40FPS for a 50 by 50 mesh. Changing the accessors and mutators in Particle to return and pass by

reference improved performance slightly also, giving gains of roughly 5FPS.

The final design of the first development cycle can be seen in Fig A.2

3.1.1.2 Second Cycle

During the second development cycle the model developed in the first cycle was adapted to use different integration methods. The explicit Euler and Verlet integrators were implemented as described in 2.1.4.1. Wind was also added as an additional external force.

3.1.1.3 Third Cycle

In the final development cycle the fourth order Runge-Kutta and implicit Euler integrators were implemented.

3.2 Testing Plan

3.2.1 Test Data

To evaluate the performance impact of different integration method a number of data points will be captured from the simulation

- Simulation frame rate. This will be affected by the cost and frequency of the integration calculations and is essential to capture to determine if the simulation is running in real time
- Average time spent on integration calculations. Extracting this allows investigations into the affect of more expensive, but less frequent, integrators
- Time taken to reach the equilibrium point. The equilibrium point is defined as the point at which there is close to zero total force for all particles, i.e. the external and internal forces are balanced. Wang, Hu, and Zhuang (2009) have shown that using higher order Runge-Kutta integrators reduces the time taken to reach equilibrium and thus, extracting this data will allow comparisons with their results

In addition, the average time spent updating and rendering the cloth will also be extracted, in order to gain a better understanding of where the time each frame is spent.

3.2.2 Test Parameters

3.3 System Requirements

Before proceeding with the design and implementation of any software project it is important to define the requirements of the system to be developed. The requirements of a system define what the system must be able to do and are normally split into functional and non-functional requirements. Functional requirements of a system specify the explicit behaviour of the system and non-functional requirements specify other properties of the system that aren't explicitly related to the functionality, e.g. performance, file types supported etc.

In a professional software engineering environment the requirements gathering process would be conducted with the end user of the system, in order to ensure that what is developed matches what the end users actually want, and not what the developers think they might want. In this situation, a variety of requirement gathering techniques would be employed, some of which may include

- Document analysis; analysing any existing documents the end users currently use. This can help to establish the structure of database tables, or simply the datatypes the system should use
- Interviews. By interviewing end users, it is easy to get a list of what they want as well as an idea on how the system currently works
- Observation. By observing the current system, an idea of the flow of data etc. can be gathered
- Research. Research into the technologies available, or other similar solutions, can be a key factor in specifying a system

Due to the nature of the project there is no specific end user, so the requirements were based on the research described in the previous section.

3.3.1 Functional Requirements

- The system must be able to play Connect 4 in The Arena as effectively as possible
- The system must be able to optimise any unconstrained real valued maximisation or minimisation problem
- The system must optimise within the bounds defined by the problem only

- The system must implement the genetic, evolution strategies and particle swarm optimisation algorithms as described in the previous section
- The genetic and evolution strategies algorithms must be implemented supporting a number of the different selection and recombination methods described in the previous section
- The state of the system must be able to be written to and loaded from a file
- The system must store the fitness of each individual at each generation, and must allow this data to be plotted on a scatter graph

3.3.2 Non-functional Requirements

- The parameters of the algorithms must be easily changeable
- The system must allow new problems to be defined as easily as possible
- The system must allow problem specific stopping criteria to be easily defined
- The system must allow new binary or real value coded population based algorithms to be added easily
- The system must be fully documented using JavaDoc
- The system must be machine and operating system independent

3.4 Development Methodology

Using a software development methodology helps to keep any large scale development project in check. Development methodologies split development into a number of phases, which build on top of each other and guide the development of the software. Typical phases in a development methodology are analysis, design, implementation and testing.

The primary development methodology chosen for this project was phased development. In this methodology, the system is developed in phases of design, implementation and testing, with each phase building on the previous one, as shown by figure 3.1**phased** The use of this methodology is supported by the project's first and second aims. These aims can be used to split the project into two developmental phases; first, to develop a generic optimisation library and second to adapt this library to The Arena.

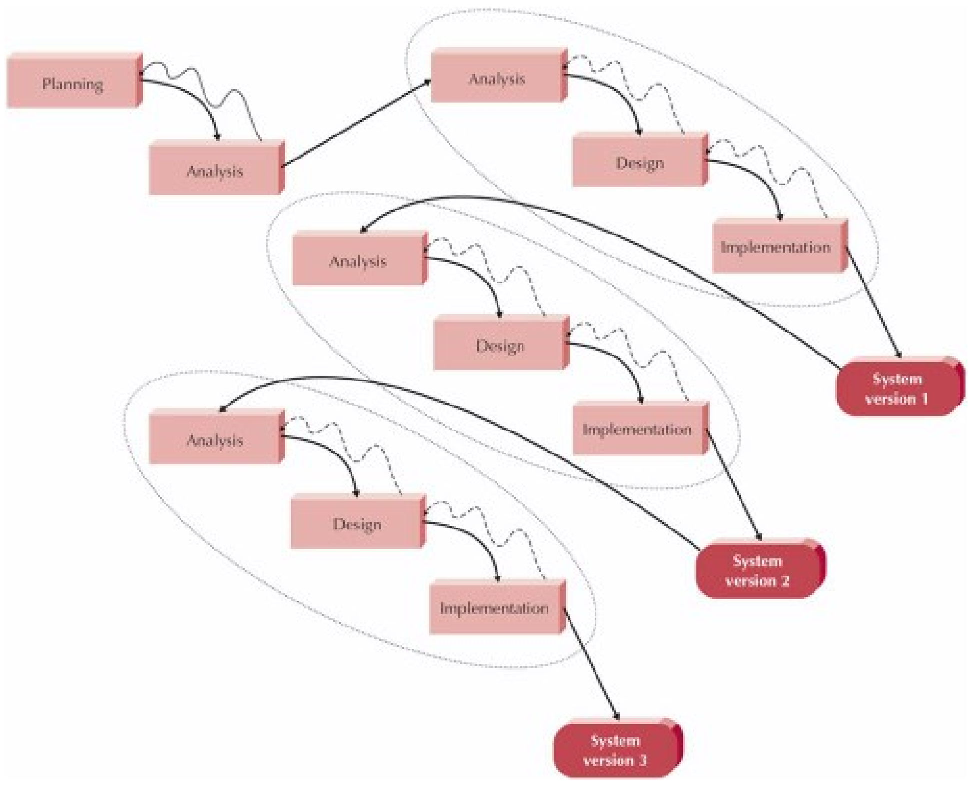


Figure 3.1: Phased development methodology

Appendix A

Class Diagrams

A.1 First Cycle Design

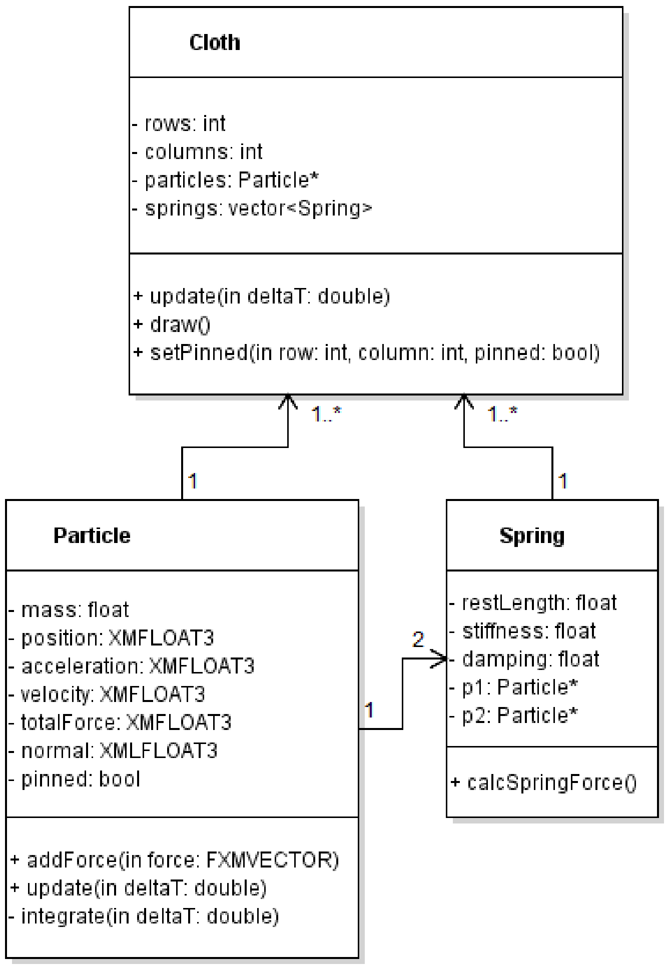


Figure A.1: Initial design for the first development cycle

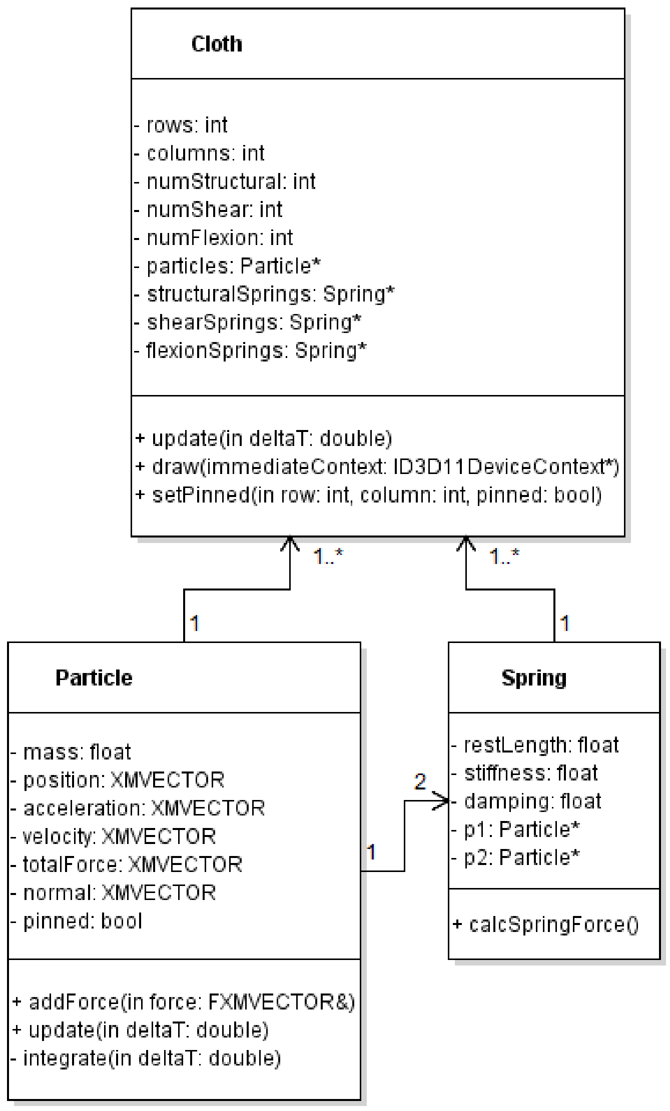


Figure A.2: Final design for the first development cycle

A.2 The Arena Class Diagram

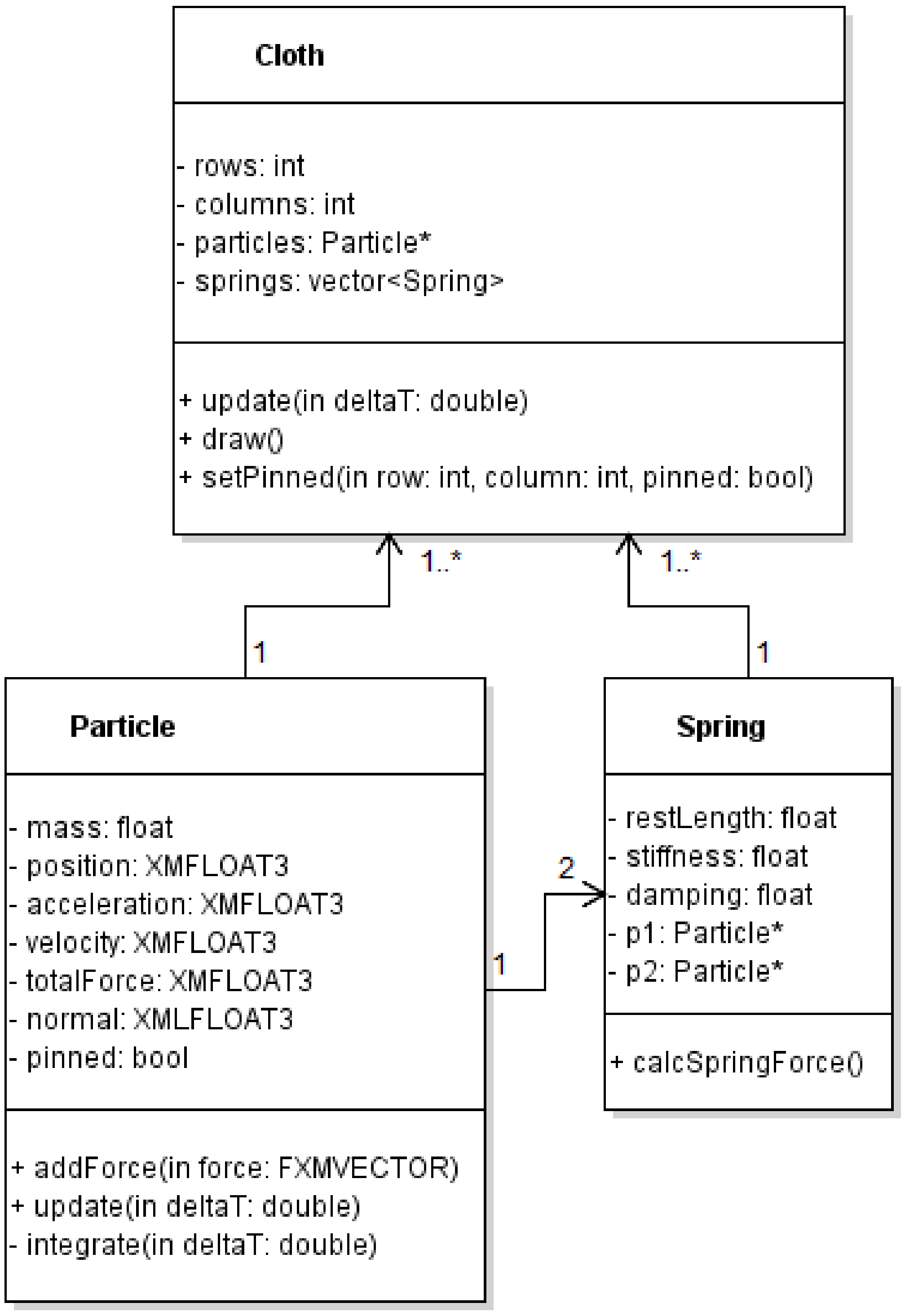


Figure A.3: Class diagram showing the additional Arena classes

References

Baraff, David and Andrew Witkin (1998). “Large Steps in Cloth Simulation.” In: *SIGGRAPH*, pp. 43–54.

Bartels, Pieterjan (2014). *Simulation of Tearable Cloth.* URL: `https://nccastaff.bournemouth.ac.uk/jmacey/CGITech/reports2015/PBartels{\_}PieterjanBartels{\_}CGITECH{\_}final.pdf` (visited on 03/10/2016).

Choi, Kwang-Jin and Hyeong-Seok Ko (2002). “Stable but Responsive Cloth.” In: *SIGGRAPH*, pp. 604–611.

Enqvist, Henrik (2010). *The Secrets Of Cloth Simulation In Alan Wake.* URL: `http://www.gamasutra.com/view/feature/132771/the{\_}secrets{\_}of{\_}cloth{\_}simulation{\_}in{\_}.php?page=1` (visited on 03/03/2016).

Jakobsen, Thomas (2005). *Advanced Character Physics.* URL: `http://www.gotoandplay.it/{\_}articles/2005/08/advCharPhysics.php` (visited on 03/03/2016).

Kang, Young-Min, Jeong-Hyeon Choi, and Hwan-Gue Cho (2000). “Fast and Stable Animation of Cloth with an Approximated Implicit Method.” In: *Computer Graphics International*. Geneva.

Kim, Tae-Yong (2011). *Character Clothing in PhysX-3.*

Lander, Jeff (2000b). *Devil in the Blue Faceted Dress: Real Time Cloth Animation.* URL: `http://www.gamasutra.com/view/feature/131851/devil{\_}in{\_}the{\_}blue{\_}faceted{\_}dress{\_}.php` (visited on 03/03/2016).

Magnenat-Thalmann, Nadia and D. Thalmann (2004). *Handbook of Virtual Humans.* 1st. Chichester: Wiley.

Mesit, Jaruwan, Ratan Guha, and Shafaq Chaudhry (2007). “3D soft body simulation using mass-spring system with internal pressure force and simplified implicit integration.” In: *Journal of Computers*.

Mongus, D. et al. (2012). “A hybrid evolutionary algorithm for tuning a cloth-simulation model.” In: *Applied Soft Computing Journal*.

Ng, Hing N. and Richard L. Grimsdale (1996). "Computer Graphics Techniques for Modeling Cloth." In: *IEEE Computer Graphics & Applications* 16, pp. 28–42.

Parent, Rick (2012). *Computer Animation Algorithms and Techniques*. Third. Waltham: Elsevier. URL: `https://www.dawsonera.com/readonline/9780124159730/startPage/20`.

Provot, Xavier (1995). "Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behaviour." In: *Graphics Interface'95*, pp. 141–155.

Tang, Min et al. (2013). "A GPU-based streaming algorithm for high-resolution cloth simulation." In: *Computer Graphics Forum*.

Vassilev, T, B Spanlang, and Y Chrysanthou (2001). "Fast Cloth Animation on Walking Avatars." In: *Eurographics*.

Volino, Pascal, Frederic Cordier, and Nadia Magnenat-Thalmann (2005). "From early virtual garment simulation to interactive fashion design." In: *Computer-Aided Design* 37.6, pp. 593–608.

Volino, Pascal and Nadia Magnenat-Thalmann (2001). "Comparing Efficiency of Integration Methods." In: *Conference on Computer Graphics International*.

Wacker, Markus, Bernhard Thomaszewski, and Michael Keckeisen (2005). "Physical Models, Numerical Solvers for Cloth Animation and Virtual Cloth Design." In: *Eurographics*.

Wang, Jianchun, Xinrong Hu, and Yi Zhuang (2009). "The dynamic cloth simulation performance analysis based on the improved spring-mass model." In: *International Conference on Wireless Networks and Information Systems, WNIS 2009*, pp. 282 –285.

Xinrong, Hu et al. (2009). "Review of cloth modeling." In: *2009 Second ISECS International Colloquium on Computing, Communication, Control, and Management, CCCM 2009*.

Yalçın, M Adil and Cansın Yıldız. *Techniques for Animating Cloth*. URL: `http://www.cs.bilkent.edu.tr/{~}cansin/projects/cs567-animation/cloth/cloth-paper.pdf` (visited on 05/25/2016).

Zeller, Cyril (2005). "Cloth Simulation On The GPU." In: *SIGGRAPH*.

Zhang, Dongliang and Matthew M F Yuen (2001). "Cloth simulation using multilevel meshes." In: *Computers and Graphics (Pergamon)*.

Zink, Nico and Alexandre Hardy (2007). "Cloth Simulation and Collision Detection using Geometry Images." In: *AFRIGRAPH*. New York: ACM, pp. 187–195.

Bibliography

Akiya, Naoko, Scott Bury, and John M. Wassick (2011). “A General Model for Soft Body Simulation in Motion.” In: *Winter Simulation Conference*. 2010, pp. 2194–2205.

Baraff, David and Andrew Witkin (1998). “Large Steps in Cloth Simulation.” In: *SIGGRAPH*, pp. 43–54.

Bartels, Pieterjan (2014). *Simulation of Tearable Cloth*. URL: `https://nccastaff.bournemouth.ac.uk/jmacey/CGITech/reports2015/PBartels{\_}PieterjanBartels{\_}CGITECH{\_}final.pdf` (visited on 03/10/2016).

Bender, Jan, Daniel Weber, and Raphael Diziol (2013). “Fast and stable cloth simulation based on multi-resolution shape matching.” In: *Computers and Graphics (Pergamon)*.

Bourg, David M. and Bryan Bywalec (2013). *Physics for Game Developers*. Second. Sebastopol: O’Reilly. ISBN: ISBN 9781449361051. URL: `https://www.dawsonera.com/abstract/9781449361051`.

Boxerman, Eddy (2003). “Speeding Up Cloth Simulation.” Master’s Thesis. The University of British Columbia.

Bridson, Robert, Ronald Fedkiw, and John Anderson (2002). “Robust Treatment of Collisions, Contact and Friction for Cloth Animation.” In: *SIGGRAPH*, pp. 594–603.

Choi, Kwang-Jin and Hyeong-Seok Ko (2002). “Stable but Responsive Cloth.” In: *SIGGRAPH*, pp. 604–611.

— (2005). “Research problems in clothing simulation.” In: *Computer Aided Design* 37, pp. 585–592.

Conger, David (2004). *Physics Modeling for Game Programmers*. First. Boston: Course Technology / Cengage Learning. URL: `http://site.ebrary.com/lib/staffordshire/reader.action?docID=10065752`.

De Farias, Thiago S M C et al. (2008). “A high performance massively parallel approach for real time deformable body physics simulation.” In: *Proceedings - Symposium on Computer Architecture and High Performance Computing*.

Desbrun, Mathieu, Peter Schröder, and Alan Barr (1999). “Interactive Animation of Structured Deformable Objects.” In: *Graphics Interface*. San Francisco: Morgan Kaufmann Publishers Inc., pp. 1–8.

Enqvist, Henrik (2010). *The Secrets Of Cloth Simulation In Alan Wake*. URL: `http://www.gamasutra.com/view/feature/132771/the{\_}secrets{\_}of{\_}cloth{\_}simulation{\_}in{\_}.php?page=1` (visited on 03/03/2016).

Ertekin, Burak (2014). *Cloth Simulation*. URL: `https://nccastaff.bournemouth.ac.uk/jmacey/CGITech/reports2015/BErtekin{\_}burakertekin-cgitech.pdf` (visited on 03/10/2016).

Garre, Carlos and Alvaro Pérez (2008). *A simple Mass-Spring system for Character Animation*. URL: `http://www.gmrv.es/{~}cgarre/APO{\_}MassSpring{\_}Jun08.pdf` (visited on 05/25/2016).

Gibson, Sarah F. F. and Brian Mirtich (1997). *A Survey of Deformable Modeling in Computer Graphics*. Tech. rep. Cambridge: MERL.

Harada, Takahiro (2006). “Real-time Cloth Simulation Interacting with Deforming High-Resolution Models.” In: *SIGGRAPH*.

— (2008). “Real-Time Rigid Body Simulation on GPUs.” In: *GPU Gems 3*. Pearson Education Inc. Chap. 29. URL: `http://http.developer.nvidia.com/GPUGems3/gpugems3{\_}ch29.html`.

Jakobsen, Thomas (2005). *Advanced Character Physics*. URL: `http://www.gotoandplay.it/{\_}articles/2005/08/advCharPhysics.php` (visited on 03/03/2016).

Kang, Young-Min, Jeong-Hyeon Choi, and Hwan-Gue Cho (2000). “Fast and Stable Animation of Cloth with an Approximated Implicit Method.” In: *Computer Graphics International*. Geneva.

Kim, Tae-Yong (2011). *Character Clothing in PhysX-3*.

Lander, Jeff (2000a). *Collision Response: Bouncy, Trouncy, Fun*. URL: `http://www.gamasutra.com/view/feature/3427/collision{\_}response{\_}bouncy{\_}.php` (visited on 03/24/2016).

— (2000b). *Devil in the Blue Faceted Dress: Real Time Cloth Animation*. URL: `http://www.gamasutra.com/view/feature/131851/devil{\_}in{\_}the{\_}blue{\_}faceted{\_}dress{\_}.php` (visited on 03/03/2016).

Magnenat-Thalmann, Nadia and D. Thalmann (2004). *Handbook of Virtual Humans*. 1st. Chichester: Wiley.

Mesit, Jaruwan, Ratan Guha, and Shafaq Chaudhry (2007). “3D soft body simulation using mass-spring system with internal pressure force and simplified implicit integration.” In: *Journal of Computers*.

Mesit, Jaruwan and Ratan K. Guha (2008). “Soft Body Simulation with Leaking Effect.” In: *2008 Second Asia International Conference on Modelling & Simulation (AMS)*. IEEE, pp. 390–395.

Miguel, E et al. (2012). "Data-Driven Estimation of Cloth Simulation Models." In: *Computer Graphics Forum* 31.2, pp. 519–528.

Mongus, D. et al. (2012). "A hybrid evolutionary algorithm for tuning a cloth-simulation model." In: *Applied Soft Computing Journal*.

Müller, Matthias et al. (2006). "Position Based Dynamics." In: *VRIPHYS*, pp. 71–80.

Ng, Hing N. and Richard L. Grimsdale (1996). "Computer Graphics Techniques for Modeling Cloth." In: *IEEE Computer Graphics & Applications* 16, pp. 28–42.

NVidia (2007). *Cloth Simulation*. URL: `http://developer.download.nvidia.com/SDK/10/direct3d/Source/Cloth/doc/Cloth.pdf` (visited on 05/25/2016).

O 'connor, Corey and Keith Stevens (2003). *Modeling Cloth Using Mass Spring Systems*. (Visited on 03/10/2016).

Ozgen, Oktar et al. (2010). "Underwater Cloth Simulation with Fractional Derivatives." In: *ACM Trans. Graph* 29.23.

Parent, Rick (2012). *Computer Animation Algorithms and Techniques*. Third. Waltham: Elsevier. URL: `https://www.dawsonera.com/readonline/9780124159730/startPage/20`.

Plath, Jan (2000). "Realistic modelling of textiles using interacting particle systems." In: *Computers & Graphics* 24, pp. 897–905.

Provot, Xavier (1995). "Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behaviour." In: *Graphics Interface'95*, pp. 141–155.

Santos Souza, Marco, Aldo Von Wangenheim, and Eros Comunello (2014). "Fast Simulation of Cloth Tearing." In: *SBC Journal on Interactive Systems* 5.1, pp. 44–48.

Schmitt, N et al. (2013). "Multilevel Cloth Simulation using GPU Surface Sampling." In: *Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS*.

Selle, Andrew et al. (2009). "Robust High-Resolution Cloth Using Parallelism, History-Based Collisions and Accurate Friction." In: *IEEE Transactions on Visualization and Computer Graphics* 15, pp. 339–350.

Tang, Min et al. (2013). "A GPU-based streaming algorithm for high-resolution cloth simulation." In: *Computer Graphics Forum*.

Vassilev, T, B Spanlang, and Y Chrysanthou (2001). "Fast Cloth Animation on Walking Avatars." In: *Eurographics*.

Vassilev, Tzvetomir and Bernhard Spanlang (2002). "A Mass-Spring Model For Real Time Deformable Solids." In: *East-West Vision*.

Vassilev, Tzvetomir Ivanov (2010). “Comparison of Several Parallel API for Cloth Modelling On Modern GPUs.” In: *CompSysTech*, pp. 131–136.

— (2011). “Comparison of Parallel Algorithms for Modelling Mass-springs Systems with Several APIs on Modern GPUs.” In: *CompSysTech*, pp. 204–209.

Volino, Pascal, Frederic Cordier, and Nadia Magnenat-Thalmann (2005). “From early virtual garment simulation to interactive fashion design.” In: *Computer-Aided Design* 37.6, pp. 593–608.

Volino, Pascal, Martin Courchesne, and Nadia Magnenat-Thalmann (1995). “Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects.” In: *SIGGRAPH*, pp. 137–144.

Volino, Pascal and Nadia Magnenat-Thalmann (1997a). “Developing Simulation Techniques for an Interactive Clothing System.” In: *International Conference on Virtual Systems and MultiMedia*. IEEE, pp. 109–118.

— (1997b). “Interactive Cloth Simulation: Problems and Solutions.” In: *JWS97-B*. Geneva.

— (2001). “Comparing Efficiency of Integration Methods.” In: *Conference on Computer Graphics International*.

Wacker, Markus, Bernhard Thomaszewski, and Michael Keckeisen (2005). “Physical Models, Numerical Solvers for Cloth Animation and Virtual Cloth Design.” In: *Eurographics*.

Wang, Jianchun, Xinrong Hu, and Yi Zhuang (2009). “The dynamic cloth simulation performance analysis based on the improved spring-mass model.” In: *International Conference on Wireless Networks and Information Systems, WNIS 2009*, pp. 282 –285.

Wang, Xiuzhong and Venkat Devarajan (2004). “2D Structured Mass-spring System Parameter Optimization based on Axisymmetric Bending for Rigid Cloth Simulation.” In: *SIGGRAPH*, pp. 317–323.

Xinrong, Hu et al. (2009). “Review of cloth modeling.” In: *2009 Second ISECS International Colloquium on Computing, Communication, Control, and Management, CCCM 2009*.

Yalçın, M Adil and Cansın Yıldız. *Techniques for Animating Cloth*. URL: `http://www.cs.bilkent.edu.tr/{~}cansin/projects/cs567-animation/cloth/cloth-paper.pdf` (visited on 05/25/2016).

Zeller, Cyril (2005). “Cloth Simulation On The GPU.” In: *SIGGRAPH*.

Zhang, Dongliang and Matthew M F Yuen (2001). “Cloth simulation using multilevel meshes.” In: *Computers and Graphics (Pergamon)*.

Zhenfang, Cao and He Bing (2012). “Research of Fast Cloth Simulation Based on Mass- Spring Model.” In: *National Conference on Information Technology and Computer Science*, pp. 323–327.

Zink, Nico and Alexandre Hardy (2007). “Cloth Simulation and Collision Detection using Geometry Images.” In: *AFRIGRAPH*. New York: ACM, pp. 187–195.