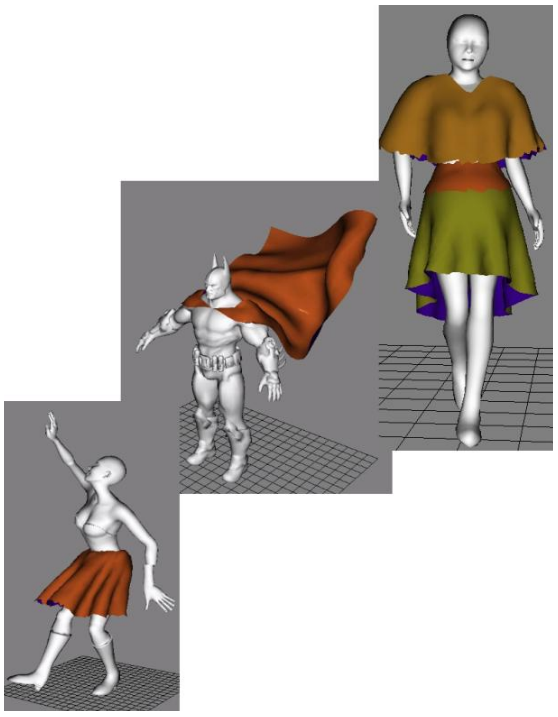
Real Time Simulation of Game Character Clothing

ANITHA RAJAGOPALAN

Master of Science,

Computer Animation and Visual Effects

Bournemouth University



August 16, 2013

Contents

Abstract

This thesis presents the concepts of developing a real time cloth simulator for game character clothing that reads the assets generated in Maya, and developing cloth authoring tools in Maya that generates the assets. The salient features of the project are real-time performance of the simulator with optimized self collision between cloth, importing the assets into the application, synching the animation with the cloth simulation and intuitive user interface to test and set the properties of the cloth to get best results.

The results achieved include real time simulation performance of cloth with physically plausible outputs on wide range of animated characters and successful development of basic cloth authoring tool which gives a completion to the game assets pipeline for clothing.

Chapter 1

Introduction

Cloth simulation and animation has been the topic of research since mid-80's in the field of computer graphics. There have been various methods proposed for achieving this such as physical methods, geometric methods, hybrid methods and data-driven methods. There is a trade-off between the realism and efficiency in such simulations. Off line simulations in movies expect more realism and compromise with the efficiency whereas applications like virtual reality in computer games, garment industry involving real-time simulation can compromise on the realism and require efficiency.

The motivation behind this project are some of the real time state of the art cloth simulators that are available like Syflex, APEX cloth, Havok cloth that provide cloth simulation solutions for games. The work flow for these software are predominantly divided into two parts – cloth authoring using 3D graphics package (like Maya, 3DS max) plugins and importing it into a game Development Environment (like UDE) where simulation tests are performed real time.

1.1 Objective

The objective of the project is to develop a standalone application that implements real time simulation of the cloth meshes on game characters that is imported from Maya and a Maya interface that aids in cloth authoring. The objective was successfully accomplished and the results are presented. The key features of the project are:

- Real time performance of the cloth simulation achieved using Position Based Dynamics methodology

- Artist friendly Maya Authoring tool for designing the cloth meshes and simulating the exported asset from Maya in the standalone application.

- Self-collision and collision with moving capsules to achieve realistic cloth behaviour with animated characters

1.2 Libraries Used

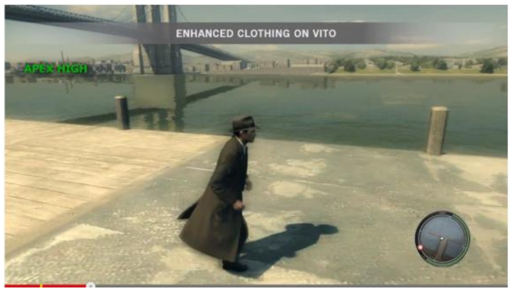
Following libraries have been used in the application that has helped in the successful completion of the project.

- NCCA Graphics Library: Open-source library developed and maintained by NCCA was the backbone of the application taking care of shading, providing containers to store graphical data like vector, matrices, face list amongst many other data structures.
- ASSIMP – Open Asset import Library: Open source library that can import various well known 3D model formats in a uniform manner, used for importing Maya Collada files exported from Maya.
- Boost Library: Open source C++ library used for parsing data, optimizing for loops and string handling

Chapter 2

Previous Work

“Real time simulation of games character clothing” - the project is based on current state of the art interactive simulation software such as PhysX and Bullet. These simulation software provide the platform for game studios to introduce real time cloth simulation solutions to their products. Fig (1) shows the cloth simulation achieved by Apex studio in their games.



(a)

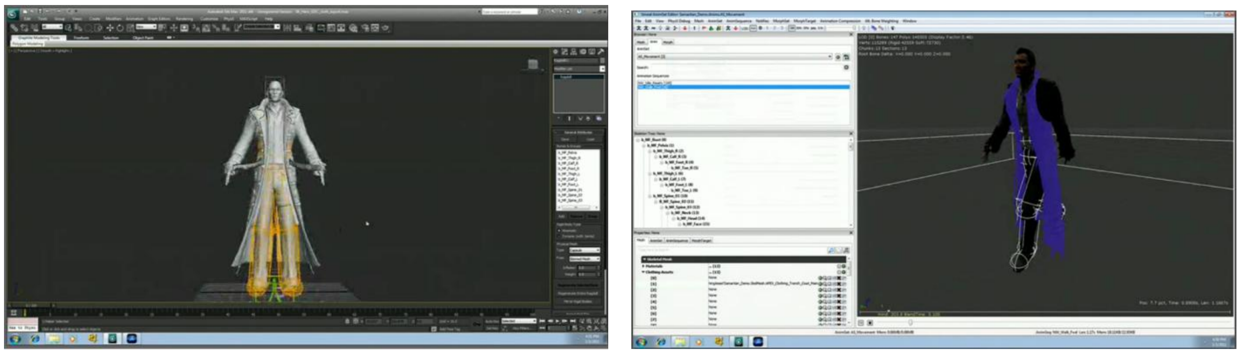
(b)

Syflex
The Cloth Simulator

(c)

Fig: 1 (a) Apex PhysX cloth simulation in Mafia II
(b) Apex PhysX cloth in Samaritan demo
(c) Syfflex real-time cloth simulation demo

The software solutions provided by the graphics development companies is generally in the form a Plugin created for commonly used 3D packages like Maya or 3DS Max. This is called the Authoring of game assets which are then exported as assets in formats readable by game Engines like UDK. Apart from plugin, there are some companies like Havok that provide a standalone cloth authoring application. Fig (2) illustrates the cloth authoring tool developed by Apex for 3DS max and the corresponding simulation of the cloth in UDK development environment.

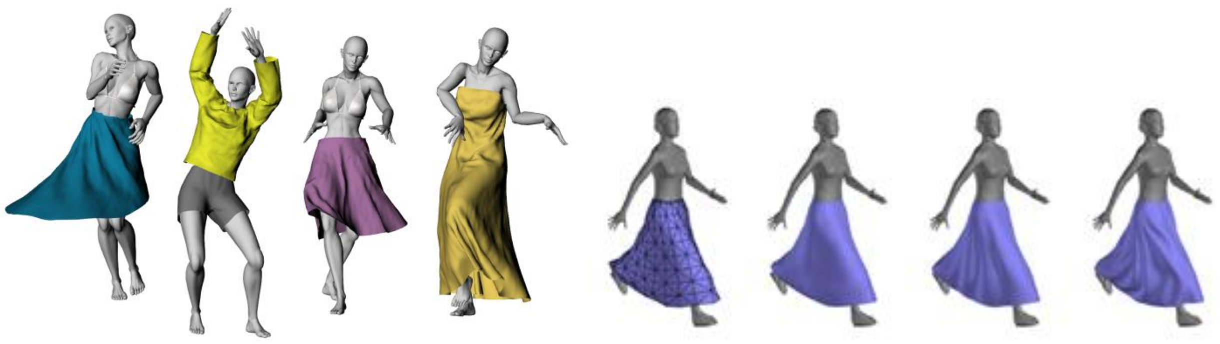


(a) (b)

Fig 2: (a) ApeX Cloth authoring plug-in for 3DS Max
(b) Authored asset imported into UDK Development environment

This project aims at developing a basic pipeline based on these solutions. For achieving this, a basic cloth authoring tool was decided to be developed in Maya that exports the most essential inputs for running the simulation. The real time simulation part was the main focus developed as a standalone C++ application mocking a Game Engine Platform like UDK that performs real time simulation tests. The methodology for achieving real time simulation of cloth to get the desired results has been analysed below.

A survey on real time cloth simulation had been done as part of Personal Enquiry course that was useful in deciding the technique to be adopted for the project. There are many techniques to achieve real time performance in cloth simulation like hybrid method, Data driven Method, GPU Implementation, Position Based Dynamics and light weight CPU method. All these methods have a trade off between accuracy, efficiency and stability. For offline cloth simulations mechanical accuracy and realism is important where as in real time, visual realism is important. *Fig: 3* illustrates some examples of real time cloth simulation achieved by these methods



(a) (b)

Fig 3: (a) Hybrid method proposed by [19]
(b) Light weight CPU method proposed by [6]

The solution chosen for this project is Position based dynamics proposed by [13]. This is the method used in PhysX real time cloth explained above. It is a recent technology also used in Bullet engine because of its simplicity and speed in generating physically plausible simulations. The paper describes the basic concept of position based dynamics with cloth simulation as an example which has been implemented that forms the base of the project.

Chapter 3

Technical Background

This section describes the technical aspects of the project. Since the application is aimed at real-time performance, the algorithms are chosen to be as efficient as possible. Firstly the Position Based Dynamics methodology is explained in general followed by the cloth simulation solution. Consecutively, efficient collision detection-resolution techniques and spatial optimization techniques are illustrated.

3.1 Position Based Dynamics

As the Position Based Dynamics methodology directly manipulates the positions of the particles of the simulation, they increase the control-ability of the system. The overshooting problems of explicit integration of force based methods can be easily controlled by adding corrections to the positions. The only point to keep in mind is that while applying the corrections the linear and angular momentum of the system should be preserved. If it is not preserved, the system never comes to equilibrium due to ghost forces. Following sections describe the position based simulation in detail.

3.1.1 Algorithm Overview

As explained by [13], every dynamic object in the simulation is represented by a set of N vertices and M constraints. The vertex i in [1,…, N] has a mass M_i, position P_i and velocity V_i.
A constraint C_j consists of following:

A constraint j in [1, …. , M] consist of

- A cardinality n_j
- A function C_j: $R^{3nj} \rightarrow R$,
- A set of indices $\{i_1,\dots, i_n\}$, i_k in [1,…., N]
- A stiffness parameter kj in [0,…,1] and
- A type of either *equality* or *inequality*

Where,
n_j - Cardinality defines the number of particles involved in the constraint,
C_j – Constraint function defines the relationship between the particles involved in the constraint.
$\{i_1, .. ,i_{nj}\}$ – is the list of indices corresponding to the particles involved in the constraint
K_j - defines the strength of the constraint ranging from 0 to 1.
Type - The constraint j with type equality is satisfied if $C_j(X_{i1},\dots,X_{inj}) = 0$ and type inequality is satisfied if $C_j(X_{i1},\dots,X_{inj}) \geq 0$.

The steps of the algorithm are defined below.

Step 1: Hook up the external forces - Calculate all External forces (F_{ext}) like gravity, wind and hook up the external forces using an explicit Euler step

```
for every particle i, ( with inverse mass $W_i$ and velocity $V_i$ )
    $V_i = V_i + dt \times W_i \times F_{ext}$
```

Step 2: damp the velocities – If this step is omitted system oscillates to infinity, explained in section 3.1.4.

Step 3: Calculate the estimated positions using the explicit Euler method

```
for every particle i, ( with position $X_i$ and estimated position $P_i$)
    $P_i = X_i + dt \times V_i$
```

Step 4: Add collision constraints – Collision is detected in this step, explained in section 3.3.
Step 5: Project the constraints – This is the core of solver where both static (like stretch and bend) and dynamic constraints (like collisions) are projected, as explained in sections 3.1.2 and 3.1.3.

```
for solver iteration times,
    $P_i = project(C_m + C_{coll}, P_i)$ ,
```

project() – Projects Static (C_m) and dynamic collision(C_{coll}) constraints maintaining the systems linear and angular momentum

Step 6: Update new velocity and position – The estimated positions (P_i) which is now projected to correct positions based on the constraints are assigned back to position variable (X_i) and the velocity is updated in exact correspondence to Verlet Integration Method.

```
for all particles i,
    $V_i = (P_i – X_i) / dt$
    $X_i = P_i$
```

Step 7: Update the velocity for collision constraint particles alone, in order to add friction and restitution to resulting velocity obtained in step 6.

Integration:

This scheme is unconditionally stable unlike the traditional explicit integration method because of step 6 that does not extrapolate blindly into the future. The particles are moved to a physically accurate position P*i* as computed by the constraint solver in step 5 which is explained in detail in the section below. The only source of instability is the solver itself that uses Newton-Raphson method to solve for valid positions. The key point is that the system stability is independent of the time step size but on the shape of the constraint function.

Though the integration step used is explicit Euler, due to the solver iteration involved in the process, the integration method cannot be clearly classified as implicit or explicit scheme. If only one solver iteration is used per step, it behaves like an explicit scheme but when the number of iteration is increased, the system can be made arbitrarily stiff and the algorithm behaves more like an implicit integration scheme.

3.1.2 The solver

The input to the solver is the list of constraints $C_M + C_{coll}$ and the estimate positions p_1… $p_{N.}$ The solver modifies the estimate positions such that they satisfy all the constraints (both static and dynamic) added to the system. The resulting system of equation is a non-linear. Also, the inequality constraint yields inequalities to the system. To solve such a generic system of equations, Gauss-Seidel (GS) algorithm is used. This iteration method is used to solve a set of linear equations but the idea of solving one constraint after the other in a linear fashion is borrowed from this method.

This process implies that the order of steps is important because the modification to point locations are visible to the process immediately, ie., the modified position at each constraint projection step is available to the next constraint projection process. Thus the force wave propagates through the material in a single solver step. If the order is not maintained, the process can lead to undesirable oscillations. The constraint projection process that modifies the estimates to valid positions is explained below.

3.1.3 Constraints Projection

Constraints are used to determine new velocities and positions of each vertex of the cloth. They may be generated by stretching, bending, external forces such as gravity, and collision avoidance. Each of these constraints has its own constraint function. The algorithm achieves it by projecting the points using the constraint functions. The whole idea here is to conserve linear and angular momentum. Let Δpi be the displacement of vertex i by the projection. Linear momentum is conserved if

$$\sum mi\ \Delta pi = 0, \qquad (1)$$

which amounts to conserving the centre of mass. Angular momentum is conserved if

$$\sum ri \times mi\ \Delta pi = 0 \qquad (2)$$

where the ri are the distances of the pi to an arbitrary common rotation centre. The algorithm introduces ghost forces dragging and rotation the object to make sure the positions of the vertices satisfy the constraints if a projection violates them. Nevertheless, conserving linear and angular momentum only applies to internal constraints and collision or attachment constraints are allowed to have global effects on the object [4]. The method used in this project for constraint projection conserves both momenta for internal constraints.

Let c be a constraint with cardinality n on the points
$p_1,\ldots,p_n$ with constraint function C and stiffness k. Let p be the concatenation $[p_1^T,\ldots,p_n^T]^T$. In order to conserve the momenta for internal constraints, the direction to project must not change the value of the function for internal constraints C because internal constraints are independent of translation and rotation. As a result, the direction must be perpendicular to rigid body modes because it is the direction of maximal change. The gradient ∇_p C is in such direction. Both momenta are automatically conserved if the correction Δp is chosen to be along ∇_p C and all masses are equal.

Therefore, given p, the correction Δp is chosen such that C(p + Δp) = 0. This equation can be approximated by

$$C(p+\Delta p) \approx C(p) + \nabla_p C(p)\cdot \Delta p = 0 \quad (3)$$

Since Δp has to be in the direction of ∇pC, it means choosing a scalar λ such that

$$\Delta p = \lambda \nabla_p C(p). \quad (4)$$

Substituting Eq. 4 in Eq. 3, solving for λ and substituting it back into Eq. 4 yields the final formula for Δp,

$$\Delta \mathbf{p} = -\frac{C(\mathbf{p})}{|\nabla_{\mathbf{p}} C(\mathbf{p})|^2} \nabla_{\mathbf{p}} C(\mathbf{p}) \quad (5)$$

And, for each of the individual point pi, the equation becomes

$$\Delta p_i = -s\nabla_{p_i} C(p_1,\ldots,p_n) \quad (6)$$

with scaling factor

$$s = \frac{C(\mathbf{p}_1,\ldots,\mathbf{p}_n)}{\sum_j |\nabla_{\mathbf{p}_j} C(\mathbf{p}_1,\ldots,\mathbf{p}_n)|^2} \quad (7)$$

This scaling factor is the same for all points. Moreover, the corrections Δpi can be weighed by the inverse mass $w_i = 1/m_i$ if each of the point has different mass. If a point is fixed, the mass of the point is simply set to infinite. With its inverse mass being 0, the corrections Δpi is 0, and the estimate position is the same as before, it means that the point does not move. Therefore, Eq. 4 is replaced by

$$\Delta pi = \lambda wi \, \nabla_{pi} C(p) \quad (8)$$

Eq. 7 becomes

$$s = \frac{C(\mathbf{p}_1,\ldots,\mathbf{p}_n)}{\sum_j w_j |\nabla_{\mathbf{p}_j} C(\mathbf{p}_1,\ldots,\mathbf{p}_n)|^2} \quad (9)$$

and finaly Eq. 6 becomes

$$\Delta_{pi} = -s \,_{x}\, w_i \nabla_{pi} C(p_1,\ldots,p_n) \quad (10)$$

This formula will be used in projecting the constraints of cloth as explained in section 3.2

3.1.4 Damping

A global damping is added without influencing the rigid body modes of the object in step 2 of the Algorithm. There are many methods to do this but the method proposed by [13] is implemented. The algorithm steps are explained below:

Step 1: calculate the center of mass of the particles
$x_{cm} = (\sum_i x_i m_i) / (\sum_i m_i)$

Step 2: calculate the center of mass of the velocity
$V_{cm} = (\sum_i v_i m_i)/(\sum_i v_i)$

Step 3: Compute the global Linear Velocity
$L = \sum_i r_i (m_i v_i)$
Step 4: Compute the angular velocity of the system omega

$I = \sum I \tilde{r}_i \tilde{r}^T_i m_i$
$\omega = I^{-1} L^i$

Step 5: Damp the individual deviation Δv_i of the velicity v_i from the global motion ($v_{cm} + \omega_x r_i - v_i$)

For all vertices i

$\Delta v_i = v_{cm} + \omega_x r_i - v_i$

$v_i = v_i + k_{damping} \Delta v_i$

Where,

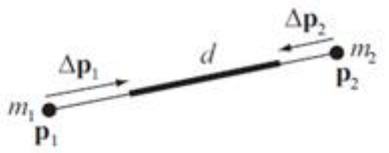
$r_i = x_i - x_{cm}$, $\tilde{r}_i$ is the 3 by 3 matrix with the property $\tilde{r}_i v = r_i v$, and $k_{damping}$ *in* [0…1] is the damping coefficient.
This preserves the rigid body modes of the object because in the in the extreme case $k_{damping} = 1$, only the global motion survives and the set of vertices behaves like a rigid body. For arbitrary values of $k_{damping}$, the velocities are globally dampened but without influencing the global motion of the vertices.

3.2 Cloth Representation

In a cloth model there are 2 types of static constraints that reproduce the natural cloth behaviour – Stretch and Bending constraints. As their names suggest, they take care of the stretchy and bendy nature of a cloth The stretching constraints is added as a *distance constraint* function as distance constraints preserve the stretching behaviour of the cloth. For the bending constraint, there are two methods, one as proposed by [13] is called *dihedral bending constraint* and other one is called Triangle bending constraint as proposed by [8]. The two methods are analysed below and the better of the two is chosen.

3.2.1 Distance constraint/ Stretch Constraint

This constraint tries to maintain the particles connecting it within a minimum distance between them similar to a spring (fig: 4), thus taking care of the stretching stiffness of cloth. This implies the constraint has to be projected throughout the simulation unconditionally making it an *equality* constraint.



Projection of the constraint $C(p_1, p_2) = |p_1 - p_2| - d$. The corrections Δp_i are weighted according to the inverse masses $w_i = 1/m_i$.

Fig: 4

The constraint function for this as generated in [13] is,

Constraint function: $C(p1,p2) = |p1 – p2| - d$ (11)

with stiffness $K_{stretch}$ and d = rest length

Using this constraint function, the projections are calculated using the equations derived in section 3.1.3. So deriving the gradients of the function with respect to p1, and p2, we get,

$$\nabla_{p1}C(p1,p2) = n \text{ and } \nabla_{p2}C(p1,p2) = -n \qquad (12)$$

Where , $\mathbf{n} = \frac{\mathbf{p}_1-\mathbf{p}_2}{|\mathbf{p}_1-\mathbf{p}_2|}$.

Substituting this in the scaling factor equation derived in Eq. (9), we get

$$\text{Scaling factor, S} = \frac{|\mathbf{p}_1-\mathbf{p}_2|-d}{w_1+w_2} \qquad (13)$$

Substituting these in equation (10), we get the final projections for the two particles as,

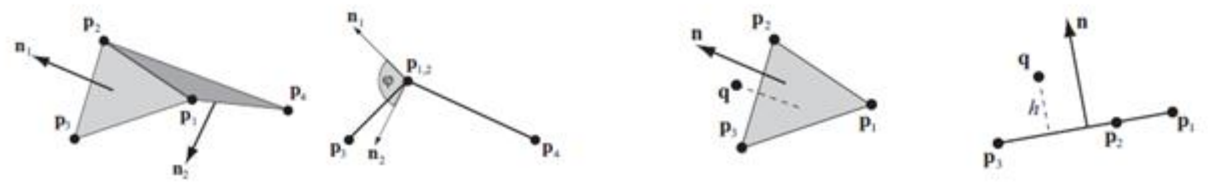
$$\Delta\mathbf{p}_1 = -\frac{w_1}{w_1+w_2}(|\mathbf{p}_1-\mathbf{p}_2|-d)\frac{\mathbf{p}_1-\mathbf{p}_2}{|\mathbf{p}_1-\mathbf{p}_2|}$$
$$\Delta\mathbf{p}_2 = +\frac{w_2}{w_1+w_2}(|\mathbf{p}_1-\mathbf{p}_2|-d)\frac{\mathbf{p}_1-\mathbf{p}_2}{|\mathbf{p}_1-\mathbf{p}_2|} \qquad (14)$$

3.2.2 Bending constraint

The two types of bending constraints are analysed in this section. First the projections for the two constraints are calculated following which they are compared based on the results obtained.

Dihedral Bending constraint

Dihedral bending constraint is constructed form triangle pairs sharing a common edge as illustrated in fig: 5



For bending resistance, the constraint function $C(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3,\mathbf{p}_4) = \arccos(\mathbf{n}_1\cdot\mathbf{n}_2)-\varphi_0$ *is used. The actual dihedral angle* φ *is measure as the angle between the normals of the two triangles.*

Constraint function $C(\mathbf{q},\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3) = (\mathbf{q}-\mathbf{p}_1)\cdot\mathbf{n}-h$ *makes sure that* $\mathbf{q}$ *stays above the triangle* $\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3$ *by the the cloth thickness* h*.*

Fig: 5

As proposed by [7], for each pair of adjacent triangles (p_1,p_2,p_3) and (p_1,p_2,p_4) the bending constraint is generated with the following constraint function:

Constraint function:

$$C_{bend}(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3,\mathbf{p}_4) = \operatorname{acos}\left(\frac{(\mathbf{p}_2-\mathbf{p}_1)\times(\mathbf{p}_3-\mathbf{p}_1)}{|(\mathbf{p}_2-\mathbf{p}_1)\times(\mathbf{p}_3-\mathbf{p}_1)|}\cdot\frac{(\mathbf{p}_2-\mathbf{p}_1)\times(\mathbf{p}_4-\mathbf{p}_1)}{|(\mathbf{p}_2-\mathbf{p}_1)\times(\mathbf{p}_4-\mathbf{p}_1)|}\right)-\varphi_0 \tag{15}$$

For deriving the gradients of the function with respect to p1, p2, p3 and p4, the constraint function for bending is rewritten as ,

$$C = \arccos(d) - \phi_0$$

Where
$d = n_1.n_2 = n_1^T n_2$

With loss of granularity we put $p_1 = 0$ and get for the normal,
$\mathbf{n}_1 = \frac{\mathbf{p}_2\times\mathbf{p}_3}{|\mathbf{p}_2\times\mathbf{p}_3|}$ and $\mathbf{n}_2 = \frac{\mathbf{p}_2\times\mathbf{p}_4}{|\mathbf{p}_2\times\mathbf{p}_4|}$
With $\frac{d}{dx}\arccos(x) = -\frac{1}{\sqrt{1-x^2}}$, we get the following gradients,

$$\nabla_{\mathbf{p}_3}C = -\frac{1}{\sqrt{1-d^2}}\left(\left(\frac{\partial\mathbf{n}_1}{\partial\mathbf{p}_3}\right)^T\mathbf{n}_2\right) \tag{16}$$

$$\nabla_{\mathbf{p}_4}C = -\frac{1}{\sqrt{1-d^2}}\left(\left(\frac{\partial\mathbf{n}_2}{\partial\mathbf{p}_4}\right)^T\mathbf{n}_1\right) \tag{17}$$

$$\nabla_{\mathbf{p}_2}C = -\frac{1}{\sqrt{1-d^2}}\left(\left(\frac{\partial\mathbf{n}_1}{\partial\mathbf{p}_2}\right)^T\mathbf{n}_2+\left(\frac{\partial\mathbf{n}_2}{\partial\mathbf{p}_2}\right)^T\mathbf{n}_1\right) \tag{18}$$

$$\nabla_{p1}C = -\nabla_{p2}C - \nabla_{p3}C - \nabla_{p4}C \tag{19}$$

Using the gradients of normalized cross product as explained in Appendix A, we compute,

$$\mathbf{q}_3 = \frac{\mathbf{p}_2\times\mathbf{n}_2+(\mathbf{n}_1\times\mathbf{p}_2)d}{|\mathbf{p}_2\times\mathbf{p}_3|} \tag{20}$$

$$\mathbf{q}_4 = \frac{\mathbf{p}_2\times\mathbf{n}_1+(\mathbf{n}_2\times\mathbf{p}_2)d}{|\mathbf{p}_2\times\mathbf{p}_4|} \tag{21}$$

$$\mathbf{q}_2 = -\frac{\mathbf{p}_3\times\mathbf{n}_2+(\mathbf{n}_1\times\mathbf{p}_3)d}{|\mathbf{p}_2\times\mathbf{p}_3|}-\frac{\mathbf{p}_4\times\mathbf{n}_1+(\mathbf{n}_2\times\mathbf{p}_4)d}{|\mathbf{p}_2\times\mathbf{p}_4|} \tag{22}$$

$$q_1 = -q_2 - q_3 - q_4 \tag{23}$$

Substituting these gradients in Eq (10), projection formula is derived as follows:

$$\Delta\mathbf{p}_i = -\frac{w_i\sqrt{1-d^2}(\arccos(d)-\varphi_0)}{\sum_j w_j|\mathbf{q}_j|^2}\mathbf{q}_i \tag{24}$$

Triangle bending constraint

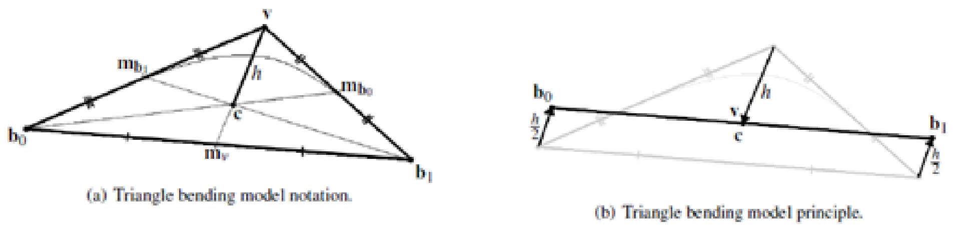


Fig 6: The triangle bending model (a) The intersection point of the three medians is the centroid c. The radius of curvature is represented by the length h from the centroid to the tip of the triangle. By employing the length difference, a simple geometric constraint can be formulated. (b) The triangle in a collapsed state where the vertices have been displaced along the direction of the center median [8]

This bending model as proposed in [8] arises from idea of collapsing a triangle with a few and as cheap operations as possible, introducing the potential energy where possible. This uses the centroid of the triangle as illustrated in fig: 6 as it has a couple of useful properties for defining the constraint. The three medians meet exactly at the centroid in a ratio 2:1, i.e. $||v - c|| = 2\ ||m_v - c||$. In order to collapse the triangle while conserving the linear and angular momenta, the base line vertices (b_o and b_1) are displaced by ½ (v-c) and v updated as v = c. The distance between the tip of the triangle v and the centroid c is used to generate the constraint function as follows:

$$C_{triangle}\ (b_0, b_1, v) \geq ||v - c|| - (K + h_0) \qquad (25)$$

Where, K – global bending param;

h_0 = rest length,

c = 1/3 (b_0 + b_1 + v), centre of medians of the triangle

This is an inequality projection constraint whose *Projection is evaluated only when $C_{triangle} < 0$.* This allows the underlying model to become more curved in rest shape like for plants and leaves. The corresponding equality function is,

$$C_{\text{triangle}}(\mathbf{b}_0, \mathbf{b}_1, \mathbf{v}) = \| \mathbf{v} - \mathbf{c} \| - h_0. \qquad (26)$$

In the above equation the global damping K=0 that helps preserve the original shape of the underling model by maintaining its curvatures as required in fabrics. Thus equality triangle bending is used in the project. The projection for this constraint function yields,

$$\Delta \mathbf{b}_0 = \frac{2w_{b_0}}{W}(\mathbf{v}-\mathbf{c})\left(1-\frac{\kappa+h_0}{\|\mathbf{v}-\mathbf{c}\|}\right)$$

$$\Delta \mathbf{b}_1 = \frac{2w_{b_1}}{W}(\mathbf{v}-\mathbf{c})\left(1-\frac{\kappa+h_0}{\|\mathbf{v}-\mathbf{c}\|}\right)$$

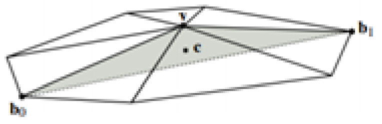
$$\Delta \mathbf{v} = -\frac{4w_{v}}{W}(\mathbf{v}-\mathbf{c})\left(1-\frac{\kappa+h_0}{\|\mathbf{v}-\mathbf{c}\|}\right) \tag{27}$$

Where, W is generalized inverse mass for the triangle, $W = w_{b_0} + w_{b_1} + 2w_v$

Comparison between Dihedral and Triangle Bending Constraint:

Comparing the behaviour of the, the Triangle bending constraints proves to be more efficient and a better approach than dihedral bending constraint. The reasons are analysed below.

- The first most evident advantage of Triangle bending constraint is the projection computation. Form Eq. 24 and Eq. 27 it is clear that dihedral bending is computationally more involved than Triangle bending method.



The triangle bending constraint must span at least two triangles to model curvature.

Fig: 7

- Triangle bending constraint is not directly dependent on topology of the mesh. It can be created between any three unique vertices as long as their contribution will model “curvature”, i.e. for a triangle mesh any triangle bending constraint must span at least two triangles as illustrated in fig: 7.
- Depending on how the triangle bending constraints are assigned to the mesh topology, a variety of different material properties can be obtained. This is different from the dihedral bending constraints which by design can only be created on shared triangle edges.
- The convergence rate of triangle bending model has been compared against the dihedral bending model in [8] and the convergence plots are shown in fig 8. This illustrates how triangle bending model converges faster that dihedral bending model.

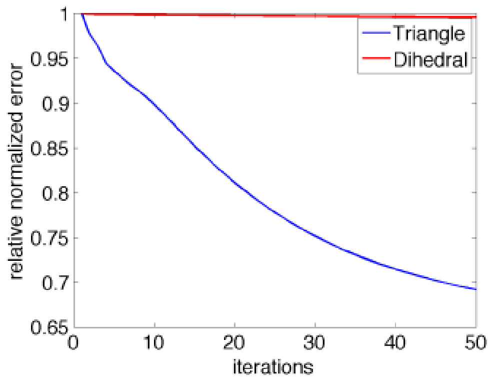


Fig: 8 Convergence plots showing how fast the triangle bending model converges, compared to dihedral-based bending model.

3.3 Collision Constraints

Collisions are created as constraints in PBD .These are dynamic constraints that change in every time step unlike the stretch and bending constraints. For collision constraints, the constraint functions and corresponding projection is not derived as explained in [13] but the traditional methods have been used as discussed below. It was observed that the equilibrium state of the system was not affected by these methods.

There are three types of collision that was required for the application

- Plane collision – collision with floor to keep the cloth from falling infinitely.
- Capsule collision – collision volume used in games, the game character will be made of capsule collision volumes.
- Self - collision – point triangle collision. Spatial Partitioning required.

In Step 4 of the simulation, the above three collision detection checks are performed and collision constraints are added to the simulator. During the projection of the constraints in Step 5, these collisions are resolved. In the last step of the simulation Step 7, the velocities of the collided particles are updated to handle restitution and friction to the collision. The methods to do this are discussed below.

3.3.1 Collision Detection

Plane Collision:

The plane collision check is the simplest as illustrated in the fig 9. Plane collision detection is used to check the collision of the object with the floor.

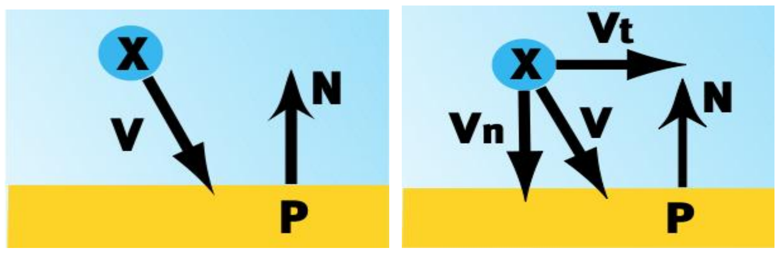


Fig: 9 Plane collision detection and resolution [4]

For a particle at position X moving with a velocity V towards a surface with normal N, the collision detection is evaluated as follows:

Contact condition: $(X-P).N < \varepsilon$
Penetrate condition: $(X-P).N < -\varepsilon$
Where, $\varepsilon \approx 0$

Once a contact collision is established the point may be moving in a velocity towards the surface or away from the surface. Thus a further check is done to determine if the velocity is away from the surface using the formula $V.N < 0$

Capsule Collision

Since the application is aimed at real time performance, complex object collision detection is not necessary. Collision with bounding capsules generated on the game characters body parts is sufficient as incorporated in most of the CPU powered games. Thus sphere - capsule collision detection is performed with a negligibly small radius assigned to the sphere that represents the cloth particles.

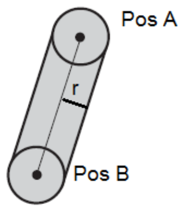


Fig 10: representation of a Capsule [4]

A capsule is also referred to as capped cylinders or sphereocylinders. They are represented by two end points of the medial line passing through the capsule and a radius as illustrated in the fig 10.

The pseudo code for the intersection test between a sphere and capsule is *[4]*:

```
SphereCapsuleCollision(Sphere s , Capsule c)
    i.   Compute the square distance between sphere centre and capsule line
         segment
    ii.  If squared distance is smaller than sum of radii of capsule and sphere,
         they collide.
```

Self-Collision

Self-collision is checked between particles and triangles of itself or other cloth meshes. Thus *sphere - triangle* collision detection is performed with a negligibly small radius assigned to the sphere.

For Sphere - Triangle collision detection, first, the point P on the triangle closest to the sphere center is computed. The distance between P and the sphere center is then compared against the sphere radius to detect possible intersection. The testing function returns true or false based on the outcome along with a point P on ABC that is closest to the sphere. This is the intersection point of the collision which is required for collision resolution.

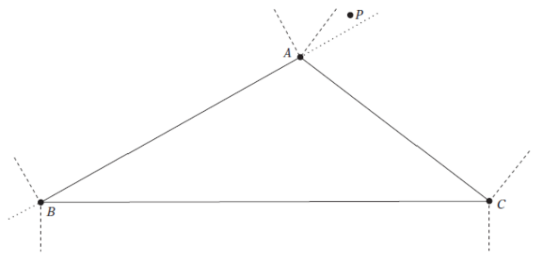


Fig:11 The test for Point P inside Triangle ABC

SphereTriangleCollision(Sphere S, Point A, point B, point C, point &P)

i. Find point P on ABC closest to the center of S
ii. Sphere and triangle intersect if the squared distance from sphere center to point P is less than squared radius of the Sphere S.

3.3.2 Collision Resolution

Collision resolution in Position Based Dynamics system happens in the constraint projection step of the solver and it is done in the form of position correction. Thus the correction Δp is calculated in each of the sections.

Plane Collision resolution:

The position correction for plane collision detection is computed as a change in position of the particle to project it back to the surface of the plane as illustrated in the fig 12. Thus the projection is computed as

$$\Delta q = -(pq.n)n$$

Applying a bouncing to the position, the correction is applied as

$$q \mathrel{+}= (1 + \varepsilon)\Delta q$$

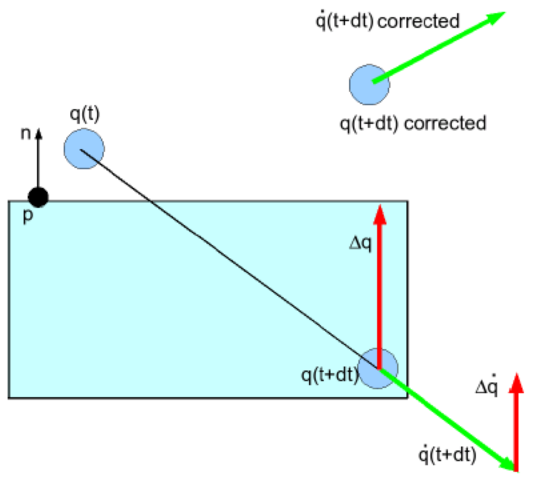


Fig: 12 point q(t) is colliding with plane P with normal n. It is inside the plane at time t+dt. A correction of Δq is applied to the particle

Capsule Collision Resolution:

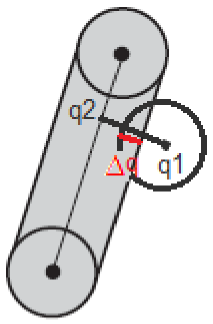


Fig 13: Capsule colliding with a sphere

For calculating the position increments for capsule collision, the contact point of collision is required. As discussed in collision detection section, this point is returned by the collision detection function. This is denoted by q1 as illustrated in fig 13.

Thus the correction increment can be computed as

$$\Delta q = (r1 + r2 - ||q1q2||)n_{12}$$

$$\text{Where, } n_{12} = q2 - q1$$

In order to maintain the center of mass for the collision, the inverse masses of the capsule and sphere are used.

$$\Delta q1 = w1/(w1 + w2)\, \Delta q$$

$$\Delta q2 = -w2/(w1 + w2)\, \Delta q \qquad (28)$$

But since the position of the capsule is not affected by the collision, its mass is infinite according to the laws of conservation of energy, and thus inverse mass w2 is 0. Hence the projection equation becomes,

$$\Delta q1 = \Delta q$$
$$\Delta q2 = 0$$

Self-Collision Resolution:

For self-collisions the same rules apply for the position increment calculation. The collision normal n_{12} is computed with the nearest position P on the triangle ABC returned by the collision detection module. The only difference is that both triangle and sphere are affected by the collision and hence w2 does not become zero and the projection is given by Eq. 28. The inverse mass w2 of the triangle is the average inverse mass of the three particles constituting the triangle.

$$w2 = (wa + wb + wc)/3$$

3.3.3 Friction and Restitution

Friction and restitution is applied to the colliding particles based on the velocity of the particle. Friction is the tangential drag exerted on the particle and restitution is push applied along the collision normal of the particle. Thus the velocity of the particle is divided into two components V_n and V_t. as described below.

$$V_n = (N \cdot V)\, N$$
$$V_t = V - V_n$$
$$V' = K_f V_t - K_r V_n$$

where,

K_r = *Coefficient of restitution,*

K_f = *Coefficient of friction,*

V = *velocity of the particle*

V' = *resultant velocity of the particle*

In the case of capsule and plane collision this method is sufficient as only the particle is influenced by the collision. On the contrary, in self-collision the colliding particle and the particles constituting the triangle are affected. Hence the relative velocity between the two colliding bodies is divided into two components and the coefficients are applied to it. The velocity of the triangle is the average velocity of the all three particles.

3.4 Spatial Hashing

Spatial hashing is a spatial partitioning technique that is required in the simulation to optimize the self-collision checks. Spatial Partitioning is the method of dividing the simulation space into grids or cells that help in optimizing the queries related to spatial proximity of the objects involved in the simulation. An object is classified into different grids based on their position in the simulation space which is done by dividing its position by grid cell size. When the objects are divided into different grids, proximity query is easily done by picking the member present in and around the grid of the querying object. A table of objects is maintained that store the objects based on their grid positions and from this table, the query returns the data.

Spatial hashing is chosen over other spatial partitioning methods as this gives infinite grid partitioning of the simulation space as opposed to the others that work inside a fixed bounding Space. But this flexibility comes with the pain of choosing appropriate values for the parameters involved in hashing. *Hash function*, Hash *table size* and *grid cell size* are the most important parameters. The hash function should be such that it computes unique hash table location for a unique grid coordinate in order to avoid hash collisions. Also the grid size should be optimum to hold only a few particles inside a cell to get best performances.
The algorithm for spatial hashing as proposed by [16] has been incorporated along with some custom set conditions which is described in section 4.1.2

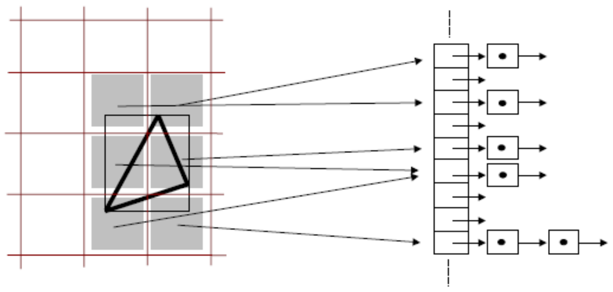


Fig: 14 Spatial query of the triangle using the BBox of the triangle

Step 1: Spatial Hashing of Vertices

The positions of all vertices are discretized with respect to a user–defined cell size. Therefore, the coordinates of the vertex position (x, y, z) are divided by the given grid cell size and rounded down to the next integer,

$$(i, j, k):\ i = \lfloor x/l \rfloor,\ j = \lfloor y/l \rfloor,\ k = \lfloor z/l \rfloor.$$

The hash indices of these vertices are then computed using the hashing function,

hash(x,y,z) = (x p1 xor y p2 xor z p3) mod n

where, (x,y,z) = (i,j,k) that was computed
P1, P2 and P3 are large prime numbers like
73856093, 19349663, 83492791.
n = Hash table size

Based on this hash index, the vertex positions are pushed into the hash table.

Step 2: Spatial hashing of triangles

First the BBOX of the triangles is calculated and minimum and maximum values of the BBox are determined. This minimum and maximum value is discretised into (I, j, k) coordinates as explained in step 1. This value is then used to identify the neighbours of the querying particle as illustrated by the fig 14 and the colliding particles and triangles are identified.

Step 3: Intersection Test

The sphere Triangle collision detection is performed here with the set of spheres and triangles that share a common spatial region.

3.5 Wind Force

Wind usually acts on faces instead of individual particle as illustrated in the fig 15. Thus it uses the normal at the points to affect the face and can be given by the following formula:

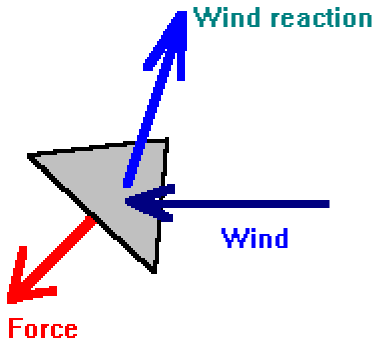


Fig 15: The direction of forces on acting on a triangle face due to wind

F = c*(n.d)*n

Where c = wind strength,

d = wind direction

n = the normal of the face at that point

Chapter 4

Implementation

The project implementation has been divided into two major parts, standalone application that will implement the real time simulation of the cloth mesh and a Maya Tools set for designing and authoring of the cloth in Maya. The basic pipeline for the project is illustrated in fig 16.

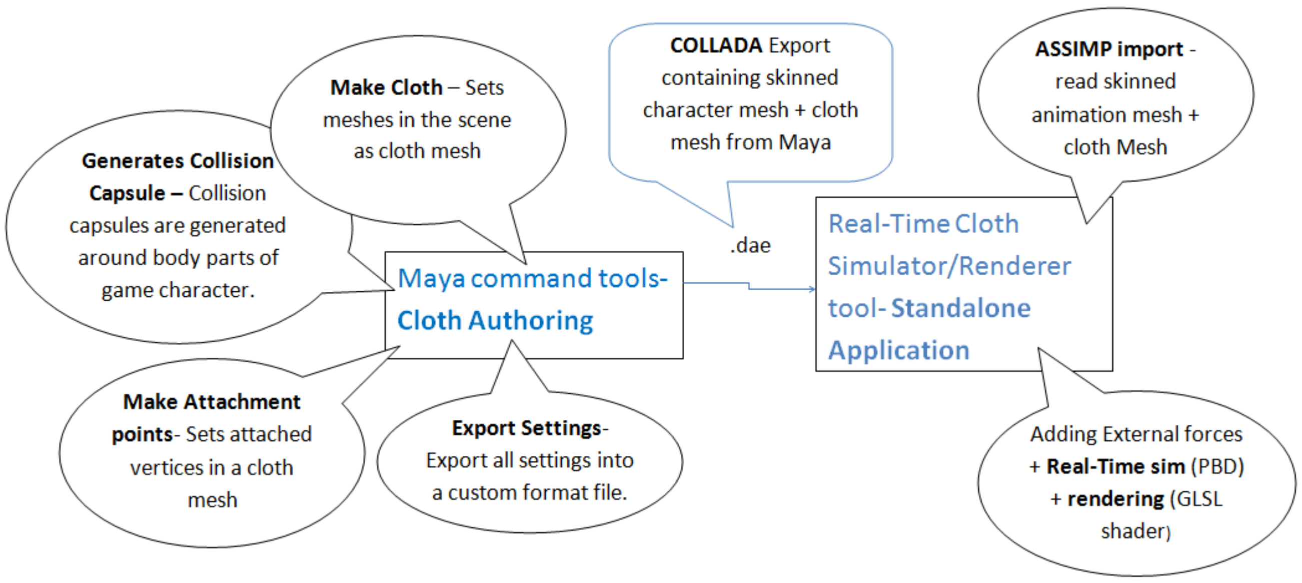

Make Cloth – Sets meshes in the scene as cloth mesh
Generates Collision Capsule – Collision capsules are generated around body parts of game character.
COLLADA Export containing skinned character mesh + cloth mesh from Maya
ASSIMP import - read skinned animation mesh + cloth Mesh
.dae
Maya command tools- Cloth Authoring
Real-Time Cloth Simulator/Renderer tool- Standalone Application
Make Attachment points- Sets attached vertices in a cloth mesh
Export Settings- Export all settings into a custom format file.
Adding External forces + Real-Time sim (PBD) + rendering (GLSL shader)

Fig 16: Project pipeline

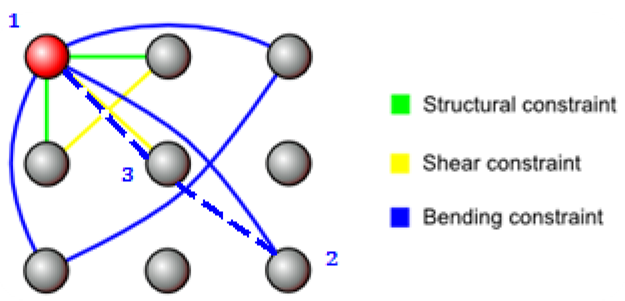
4.1 Stand alone application Development

The application was first built to simulate rectangular cloth grids. Consecutively arbitrary cloth meshes were imported and finally cloth meshes along with the skinned or static character mesh was imported combined with the settings exported using the custom authoring tools in Maya to get the final results of a clothed game character.

4.1.1 Cloth Models

Cloth Grid

The cloth grid is constructed from 2D grid of particles whose resolution and dimensions is user defined. The stretching and bending constraints are constructed based on the mass spring cloth model as illustrated in image below. The structural and shear springs are converted into stretch constraints and bending springs are converted into Triangle Bending constraints. The triangle for the bending constraint is constructed as shown in the image below.



The dotted lines represnt the connections that completes the triangle (1,2,3) that is used for the construction of Triangle Bending Constraint from the Bend Springs of the mass spring cloth model.

Fig 17

Cloth Mesh

The simulator accepts arbitrary triangle meshes as input. The only restriction imposed is that it requires representing a manifold, ie each edge should be shared by maximum two triangles. If not, due to duplicate edges, the cloth mesh will tear. Each vertex in the mesh becomes the simulated particle. The stretching and bending constraints are then generated using the edges as explained below. The edges had to be generated from the list of vertices and faces obtained from the import files (collada or obj) as 3D packages do not export edge list for the models.

Stretch Constraints Construction

The edges of the imported model represent the stretch constraint of the cloth. The edges are extracted from the face list and vertices list obtained from the Assimp importer of the application. The pseudo code is described below. This also generates a per vertex neighbour list for every vertex in the mesh.

```
for all faces in the object,  fi(i1,i2,i3)
    if(neibourList[i1] != i2)
        create StretchConstraint(i1,i2)
        neighbourList[i1] += i2;
        neighbourList[i2] += i1
    if(neibourList[i1] != i3)
        create StretchConstraint(i1,i3)
        neighbourList[i1] += i3;
        neighbourList[i3] += i1
    if(neibourList[i3] != i2)
        create StretchConstraint(i3,i2)
        neighbourList[i3] += i2;
        neighbourList[i2] += i3
```

This per vertex neighbour list is also required for the triangle bending constraint construction and thus serves the purpose of two. This list is generated as a string of vertex indices separated by ’|’ symbol, for instances “1|2|15|17|80” is generated.

Triangle Bending Constraint Construction

A fast algorithm is proposed in [14] for construction of the triangle bending constraints for a triangulated mesh. The pseudo code of the algorithm is as follows:

```
for each vertex v in the mesh
    V  = getNeighbours(v)
    for each vi in V
        $\cos(\phi_{best}) = 0$
        $v_{best}$ = vi
        for each $v_i$ in V where $v_i$ != $v_j$
            $\cos(\phi) = (vi – v) . (vj – v) / | vi – v | | vj – v |$
            if cos(phi) < $\cos(\phi_{best})$
                $\cos(\phi_{best}) = \cos(\phi)$
                $v_{best}$ = vj
    if vi not connected to $v_{best}$
        create Triangle Bending Constriant
```

A visualization done using the application of the stretching and bending constraint generated for a cloth mesh is illustrated in the images below.

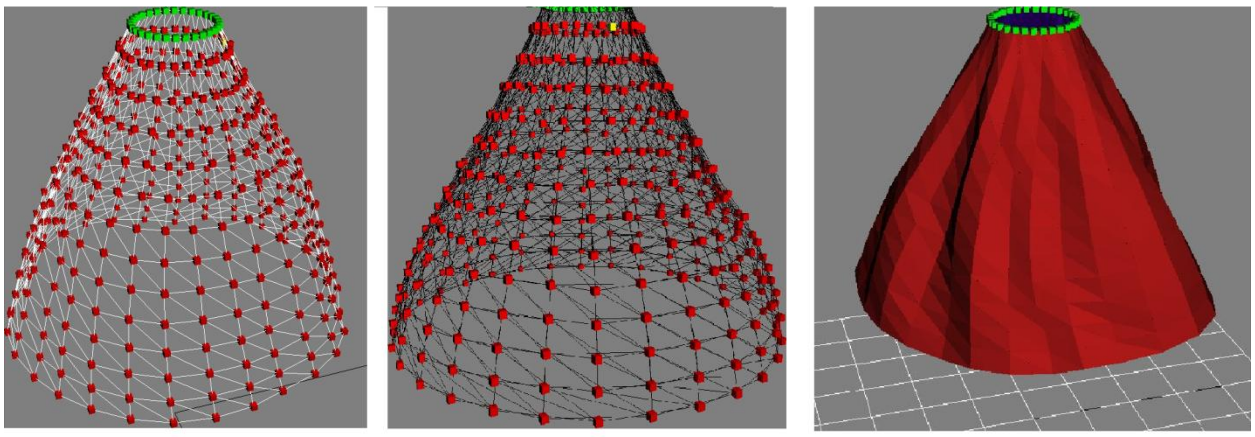

(a) Stretch cosnstraints genertaed on cloth mesh (b) Bending cosnstraints genertaed on cloth mesh (c) Simulated cloth mesh

Fig 17

The image below illustrates the behaviour of the PBD cloth model for various stiffness values for the two constraints. In case (a) the cloth is stretchy for a stretching stiffness value of 0.5. Thus on increasing the stretching stiffness to 0.9 in case (b) the mesh does not stretch but the cloth has stiffer wrinkles due to a stiffer bending constraint. Hence in case (c) the cloth looks good with high stiffness stretching constraint and low stiffness bending constraint.

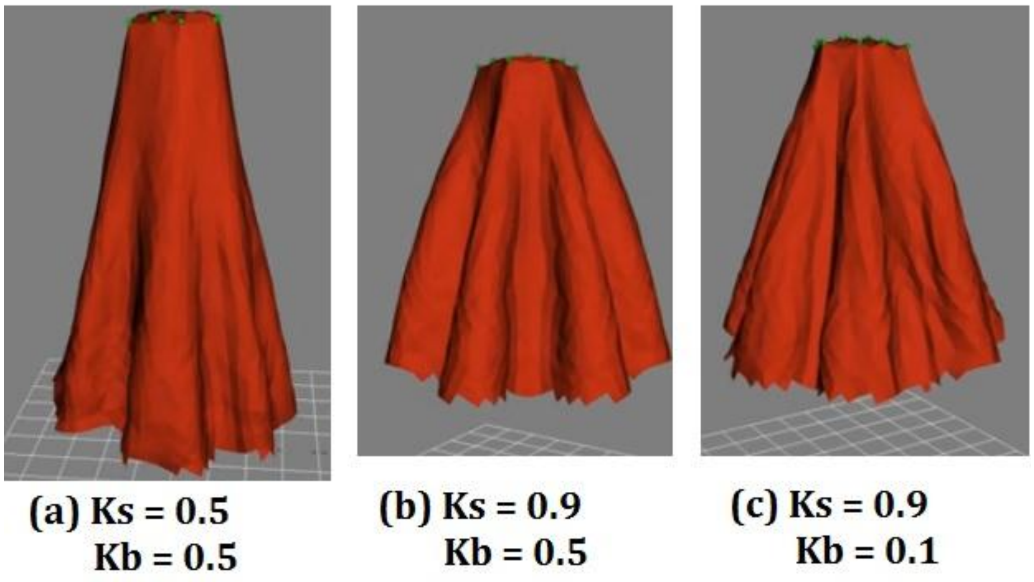


Fig 18: Comparison of cloth behaviour based on the constraint stiffness

4.1.2 Collision

Once the technical background of collisions was understood as explained in chapter 3, the implementation was straight forward. The debug visualization feature of the application helped speed up the development process of collision detection and resolution modules. Image below illustrates the different colour coding used it determining the collisions.

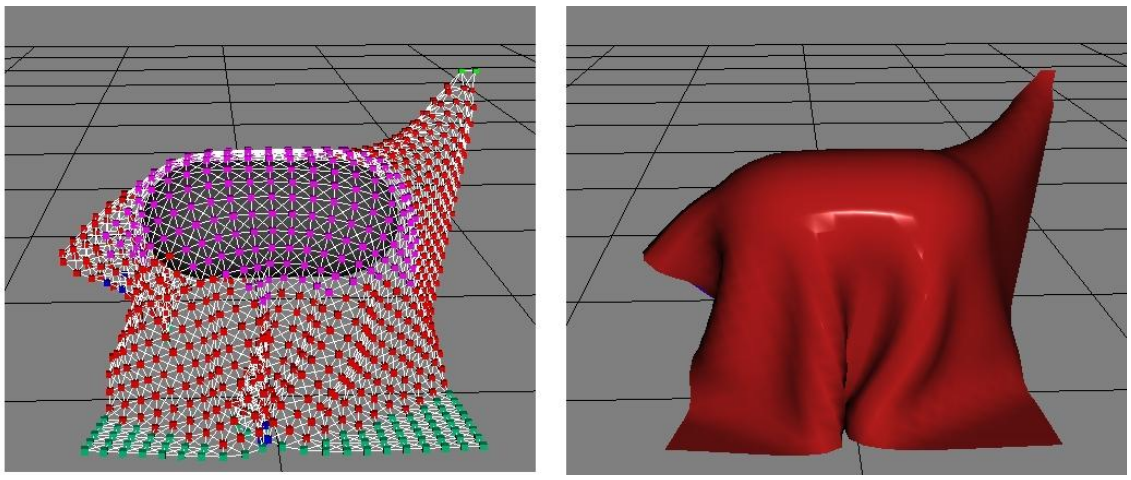


Fig 19: Visualization of collision detection: pink particles represent capsule collision, green floor collision and blue self-collision. Red coloured particles are unaffected by any of the collisions.

AABB Optimization:

For all the three collision detection viz sphere Plane, sphere capsule and sphere triangle collision detection, AABB optimization has been implemented. The sphere particle is always tested against the AABB of the plane, Capsule or mesh triangles before the collision check is performed which is the costlier operation thus increasing the code efficiency. The following sections describe observations and specific features implemented in capsule and self-collision detection.

Capsule collision

The application has an inbuilt feature of creating static and kinematic capsules to test the cloth behaviour and performance to static and dynamic collisions. This helped in testing the cloth behaviour and its response to collision with moving limbs of the game character that was later imported into the application. The moving capsules are simulated using the basic PBD simulator of the application. The images below illustrate the tests performed with moving capsule

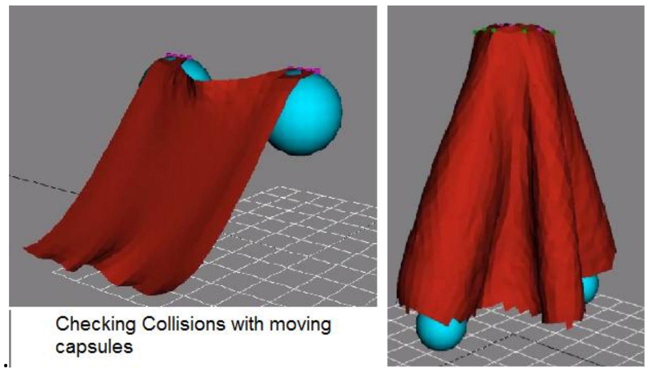
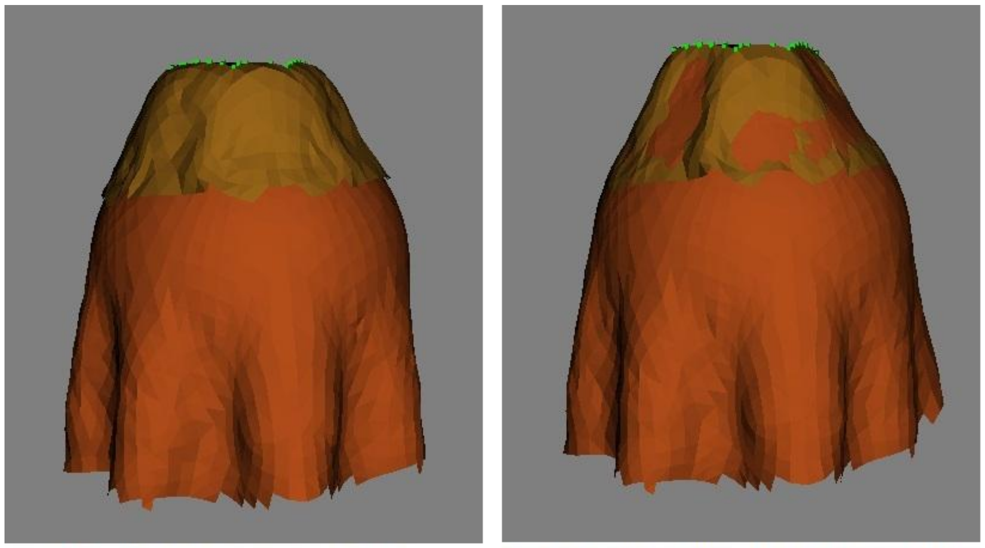


Fig 20:

It was also observed that with fast moving capsules, the cloth sometimes entered into the capsule due to the fast movement. This is not a problem up to a limit as the capsules will be sufficiently huge to not let the cloth reach the underlying character mesh.

Self-collision

This is the costliest operation of the all the algorithms implemented in the application as self-collision test is performed between all the vertices and triangles of one or more cloth meshes. Fig 21 is an output generated from the application, illustrates the scenarios when self-collision is ON and OFF between two cloth meshes. Handling collision with itself or with another cloth mesh is one and the same for the simulator as multiple cloth meshes are input as a single list of vertices. In order to optimize number of checks performed, spatial hashing technique is used as discussed in chapter 3. The implementation details and observations are described in the next section.


(a) Self collision turned ON between two cloth meshes @ 70 ms per frame for approximately 1800 particles

(b) Self collision turned OFF between two cloth meshes @ 55 ms per frame for appoximately 1800 particles

Fig 21

The initial implementation results of self-collision test performed without spatial hashing is tabulated below. The system could handle up to 400 particles at real time performance of > 29 fps which decreased the performance drastically with increase in particles count.

Number of particles (grid resolution)	Time taken per frame in s (FPS)
400 (20 x 20)	Approx= 35 ms (≈ 29 fps)
625 (25 x 25)	Appr 80 ms (≈ 13 fps)
900 (30 x 30)	Appr 135 ms (≈ 7 fps)

Table 1

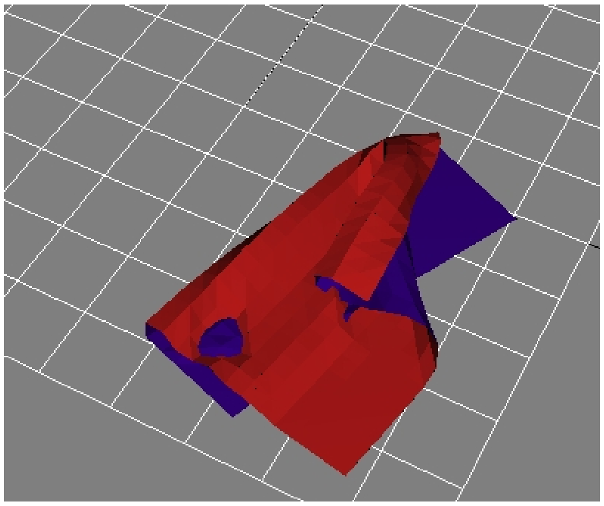
Self Collision Performance Comparison:

After implementation of spatial hashing optimization for self-collision, the performance of the system was contrasted with and without Self collision. The computation time increases by 15 ms for approximately 1800 particles when the self-collision is switched on as illustrated by fig 21. For 1000 particles the increase in computation time was negligibly small around 2 to 3 ms. Thus it can be concluded that with increase in number of particles the excess time taken per step for self-collision increases exponentially. But when compared with the self-collision performance of 3D packages like Maya, this is a good performance exhibited by the simulator, suitable for real time solutions.

Cloth Tangling

The self-collision detection is not sufficient to keep the cloth vertices on the original side. During extreme cases like pinning the cloth by a single point under gravity or fast moving cloth hitting with itself, self-collision fails.

This is called cloth tangling. When the tangling happens, the simulator is unaware of it and it dutifully keeps the cloth facing in the wrong direction, without letting it to untangle. The images below illustrate the tangles encountered during the implementation.


(a) tangled cloth fallen on gronud. The simulator dutifly keeps the cloth in the wrong side.

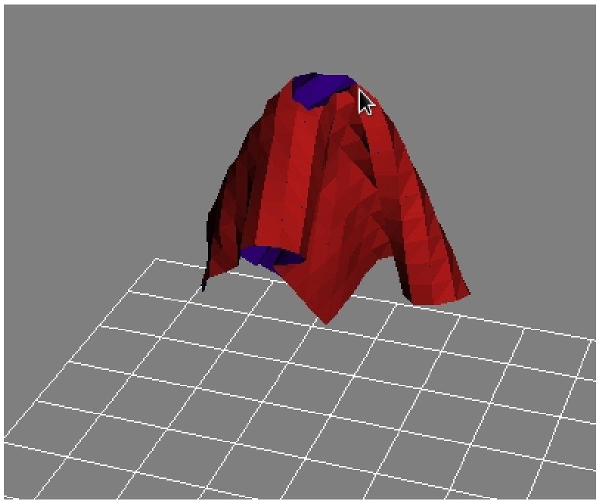

(b) cloth tangled in mid air, prevents the cloth from falling

Fig 22

A solution to this problem is to implement cloth un-tangling algorithms as proposed by [1]. But these algorithms are computationally very costly for real time requirements. From the observation made, cloth tangling could be minimized under following conditions in the application:

- Increasing the friction, restitution and stiffness of the self-collision constraint. But this increased the repulsive nature between the particles to such an extent that a falling cloth stopped in mid-air due to the upward thrust applied by the cloth underneath it.
- Another option is to increase the solver iterations within which the penetration gets resolve. Though increasing solver iterations slows down the process, this method is better than implementing a costly algorithm like cloth untangling.

Friction, Restitution and collision stiffness coefficients

The collision behaviour is highly dependent on friction restitution and the constraint stiffness. From the observations made during the implementation, the following results are arrived.

Coefficient	Value range	Cloth behaviour	
		Sticky Cloth	Slippery Cloth
Constraint Stiffness K	(0 to 1)	Approx. 0	Approx. 1
Collision restitution K_r	(0 to 1)	Approx. 0	Approx. 1
Collision friction K_f	(0 to 2)	Approx. 2	Approx. 0

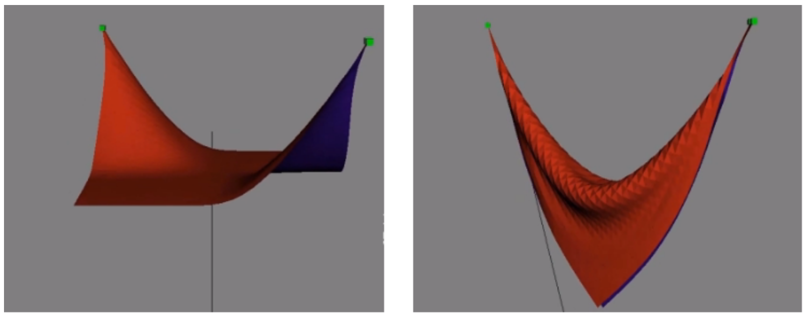
Table 2:

From the table it is evident that the cloth is slippery when the constraint is very stiff, restitution is high and friction is very low and vice-versa for sticky cloth behaviour.

When the cloth collides with two overlapped capsules, the collision response is applied multiple times to the same cloth particle causing jittering cloth response. This can be avoided by decreasing the Collision Constraint Stiffness to approx 0. But this extra jittery push is required when collision with moving intersecting capsule is performed without which the cloth tends to enter the capsule easily. The jumpy cloth response becomes invisible to eyes when the capsules are animating.

4.1.3 Spatial Hashing

The implementation of spatial hashing is pretty straight forward as explained in chapter 3. The major implementation bottlenecks faced during implementation are listed below:



(a) (b)

Comparison of self collision with and without spatial hashing for 900 particles, 6670 edges after equal number of frames

(a) Without spatial hashing, 136ms per simulation step

(b) With spatial hasing, 59ms per simulation step

Fig 23

Grid Size: Setting grid length as the average edge length of the cloth mesh as described in chapter 3 works best when the triangles in the mesh are equal in size which is not possible always. When the grid size decreases below 1, there is no improvement in performance. A grid cell size of 1 worked best for the system.

Hash Table Size: As explained in chapter three, the hash table size is determined based on the number particles present in the simulation space. The number that best suited the implementation are numbers like 99 199 299. This number is chosen as the next greatest number to the total number of particles in the system. For instance, a system 1500 particles will have a Hash table size of 1999.

Hash table Update Optimization: For large hash tables, re-initialization of the table at every time step proved to reduce the efficiency. This has been optimized by avoiding the hash table clearance step and updating the cells based on time stamps assigned to every hash map instead.

The following observations were made comparing the performance before and after the optimization. For a cloth grid of size 8x8 with 40x40 = 1600 particles and a hash table size of 1999,

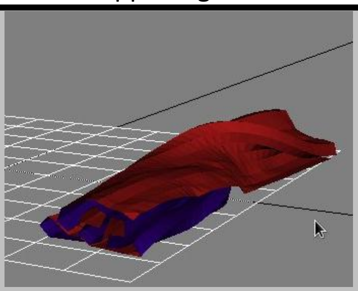
	Simulation step time With self-Collision not happening	Simulation step time With Self-collision happening
Image illustrating the scene	Hanging cloth – no self-collision checks happening	 Cloth fallen on ground – self-collision checks happening
Without Optimization	Approx. 65 ms	Approx. 80ms
With Optimization	Approx 60 ms	Approx 80ms

Table 3

From the table it is evident that the optimization has no effect when there is large number of collision checks happening because the time stamp of almost all particles would keep changing in every time step there by making the scenario same as clearing the full hash table. Whereas there is some performance improvement when there is no collision checks happening in the cloth as some calls to clear the hash table has been saved.

4.1.4 Attachment

With Position Based Dynamics method, attaching vertices to static or kinematic object is straight forward. For static attachment, the vertex position is simple set to the static target position. For kinematic objects, the vertex position is updated every time step to coincide with the position of the kinematic target object. The two cases are handled in general by the following pseudo code

```
Step 1: Pre Simulation Step,
For each fixed vert Vf,
        Relative pos =  fixed vert Position – Target Object Position.

Step 2: During every update
For each fixed vert Vf,
 Fixed Vert Position = Target Object Pos + Relative Pos
```

The application has an inbuilt feature to select attachment points for the cloth mesh. Also a cloth mesh can be attached to a moving capsule which is described in section 4.1.6.

4.1.5 Wind Forces

The application of wind forces is straightforward once the technical background was understood as explained in chapter 3. In order to give maximum flexibility to user in handling the wind effects, wind direction and wind strength is obtained from user inputs using the UI explained in section 4.5. Fig 24 below illustrates the results obtained using the tool.

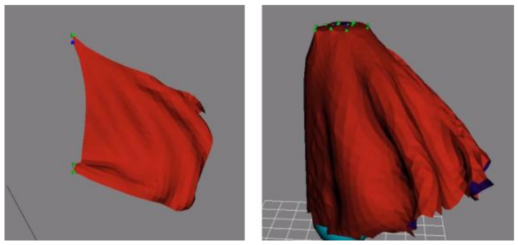


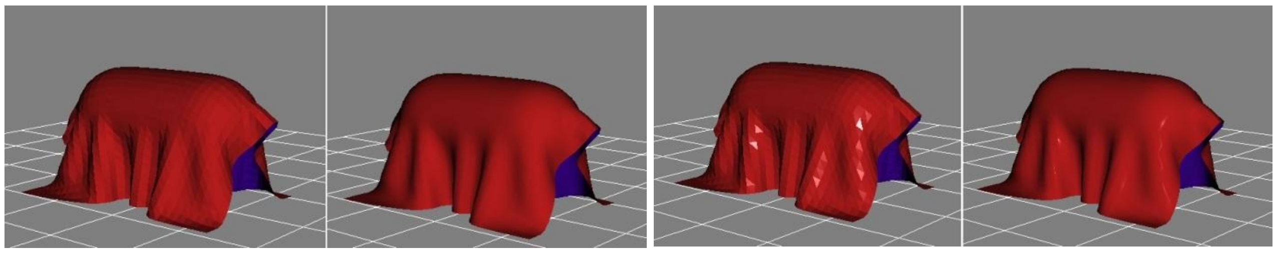
Fig 24: Initial wind effect tests with cloth grid and cloth mesh

4.1.5 Shading Models

This section throws light on the shading models and its features used in the application.

Per-fragment ADS GLSL shader:

This is the shader used for cloth rendering in the application that uses a basic phong reflectance model. This gives a glossy effect to the cloth. The user is also given the freedom to turn off the gloss and give the cloth a lambert look. This is achieved using ADS shader with zero specular coefficient set. Also being per fragment OpenGL shader, it imparts efficiency to the render cycle. The fig 25 below illustrates the difference achieved by ADS shader with and without the gloss (specular) effect.



(a) (b)

Fig 25: The left side images are cloth without smooth shading applied. Image (a) is diffuse shader and (b) is ADS shader. The glossy effect added to the cloth is evident

Smooth shading:

The shading of the cloth was always blocky as that the normals of the particles that are shared by multiple faces where always overridden by the normals of the latest triangle that is pushed into the shader. In order to avoid this, the normals per shared vertex are averaged before loading into the shader. The fig below illustrates the difference achieved by smooth shading.

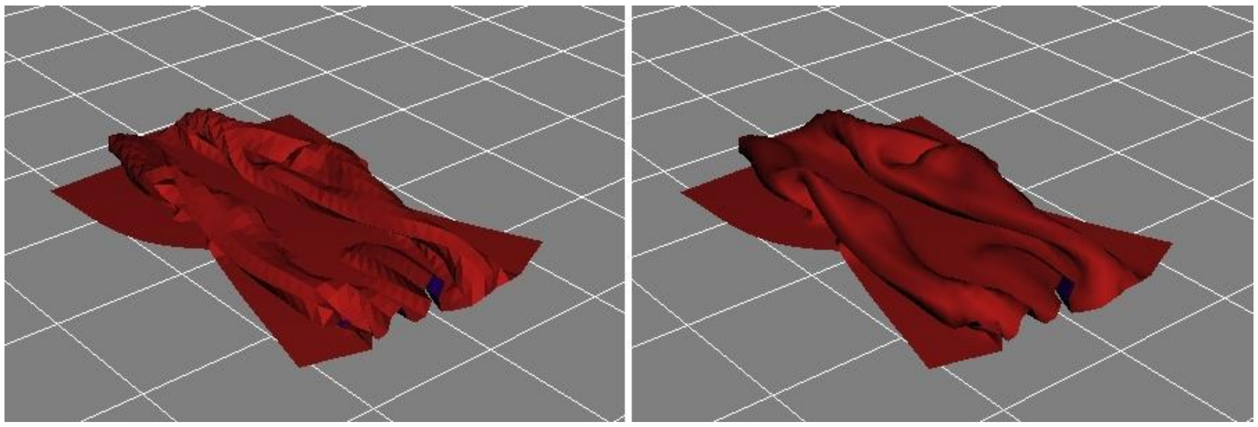


Fig 26: The smooth shading algorithm has given a good visual upgrade to the cloth

Double sided shading:

The cloth had to be shaded on both sides of the mesh which is not generally enabled in shaders as only the surface normal facing the viewer is pushed into the glsl shader for rendering. In order to shade the other side of the mesh, the normal have to be flipped and the mesh redrawn which is inefficient. Hence the option gl_FrontFacing was used in the fragment shader to enable two sided shading.

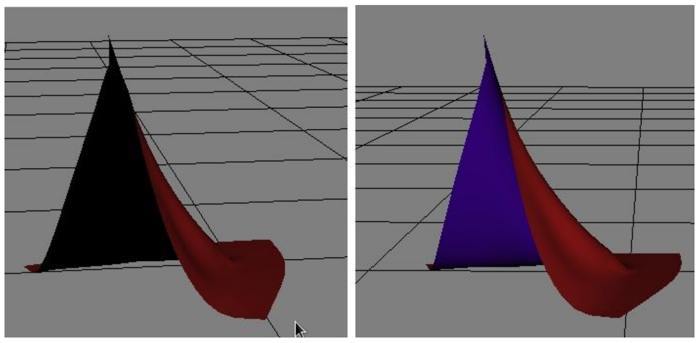


Fig 27: image on the left is without double sided shading and right image is with double sided shading

Skinning shader:

This shader is used for rendering the skinned animation mesh imported from Maya. The bone transform matrices of the skinned mesh are obtained from the assimp importer which is fed into the skinning shader where the corresponding vertex transformations are calculated. Per fragment diffuse shading model is used.

Adding a texture shader to the cloth was the intended goal of the project but it could not be achieved due to the limitations of the assimp parser used in the application. The reasons are elaborated in section 4.3.

4.2 Maya Authoring Tools

A simple cloth authoring Maya command tool set has been developed using Maya Python Commands. Due to time constraints and since this project is an attempt to test the application pipeline and real time performance of the C++ application, the idea of developing a full-fledged authoring plug-in is left to future works. The key feature of the tool is that more than one capsule can be attached to a cloth for implementing scenarios like curtain animation and more than one cloth can be attached to a single capsule useful for scenarios like multi layered skirt simulation. With no prior experience in Maya commands except 1 to 2 class room tutorials during the MSc course, a very basic set of tools has been created for the following:

- Capsules generation around animated bones of the game character
- Saving the per animation frame capsule data
- Mark meshes as Cloth mesh to be simulated
- Make attachment points on cloth mesh
- Make capsules to cloth attachment
- Dynamic undo options for all the above
- Export the settings

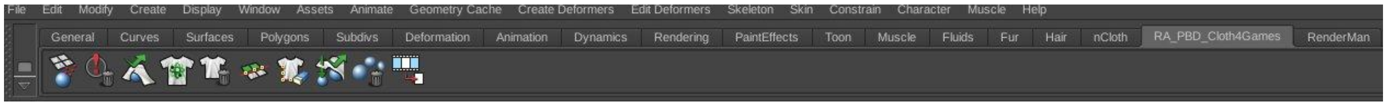


Fig 28: Custom Maya commands tool set Shelf developed for Maya cloth Authoring

Appropriate warnings and errors are generated by the tool which was understood to be an essential part of tools development for Maya.

4.2.1 Capsule Generation

This module takes care of generating capsules around the limbs of the character, saving the per animation frame capsule data and deleting the capsules generated by this tool. The user input is a set of selected bones. The pseudo code for the implementations is,

GenerateCapsule (jointsList):

```
i.   Check if more than one joint selected
ii.  Generate joint pairs to generate a bones list of the form [[joint1Pos, joint2Pos], [Joint2Pos,
     Joint3Pos],….] using joint Maya command.
iii. For each bone in the list
         o  Generate a capsule between the two joint positions of a bone using polycylinder
            Maya command with axis = JointPos1 - JointPos2 , name = JointName_ClColliCyl ,
            radius = 1, height = distance between JointPos1 & JointPos2, roundCaps = true.
         o  Move the cylinder to the centre of JointPos1 & JointPos2
         o  Parent the cylinder to parent Joint of the bone
         o  Lock the cylinder attributes like transformation, rotation order…
```

The user is not allowed to edit the height and position of the capsule. The radius of the capsules can be altered to cover up the underlying mesh. The un wanted capsules can be deleted manually by selectin one by one or delete them all using *DeletCapsule* tool. This deletes all the capsules generated.

```
DeleteAllCapsules()
    i.    Query the list of capsules generated in the scene using ls Maya command with the
          help of 'Tag names' appended to the capsules.
    ii.   Delete them using delete() Maya command.
    iii.  Delete the corresponding global data structure
```

Once the capsules are generated, deleted where necessary and sized to cover up the mesh, they need to be saved. The save capsules tool prompts the user to enter the starting and ending frame numbers for which the capsule information has to be saved. Pseudo code:

```
SaveCapsules ()
    i.    Prompt user to enter start and end frame numbers
    ii.   Query the list of capsules generated in the scene using ls Maya command
          with the help of 'Tag names' appended to the capsules.
    iii.  For each capsule in List
                Get,
                joint1 = parent of capsule using ListRelatives command
                joint2 = child of joint1
```

Any changes made to the capsuled in scene after saving the capsule will not make effect. So if any changes are made in the capsules, saveCapsule() has to be done again.

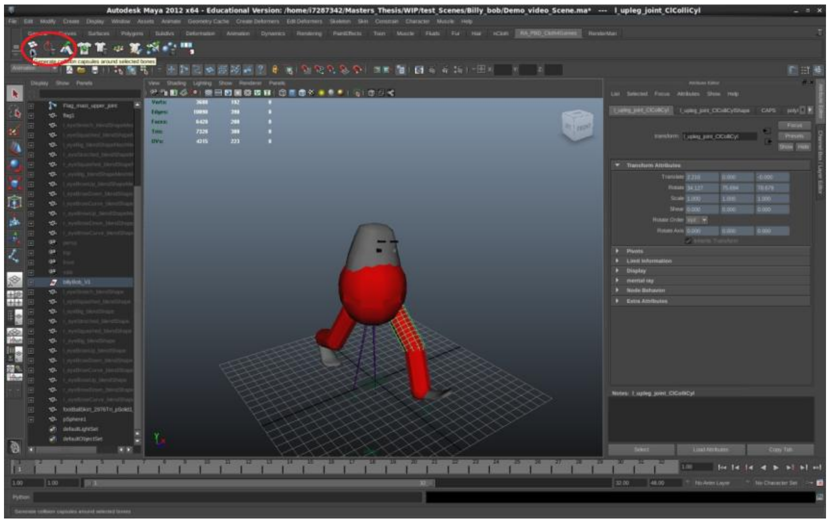


Fig 29: illustrates a Maya scene with capsule generation tools in action.

4.2.2 Make cloth

This tool marks the selected meshes in the scene to be a cloth mesh which will be fed into the simulator. This is done by adding a unique tag name '_MyPBSCloth' to the mesh which is identified in the cloth simulation application. The input from the user is a bunch of selected meshes.

Pseudo code:

```
MakeCloth()
    i.   Query the list of cloth meshes using ls command
    ii.  For every mesh in the list
             • If it is already a cloth, display warning and return
             • If the mesh is not triangulated, display warning an return
             • Rename the mesh
             • Save data in global data structure
```

For the triangulation check the number of faces and number of triangles are retrieved for the mesh using *polyEvaluate* maya command. If the two are same, the mesh is triangulated. This works fine in most cases but fails during some special cases.

Undo make cloth tool is used to undo the make cloth operation. The user input for this is a set of cloth meshes that have to be unmarked from being a cloth mesh.

```
UndoMakeCloth()
    i.   Query the list of cloth meshes using ls command
    ii.  For every mesh in the list
             • Rename it by removing the tag name using rename()
             • Clear the corresponding data structures
```

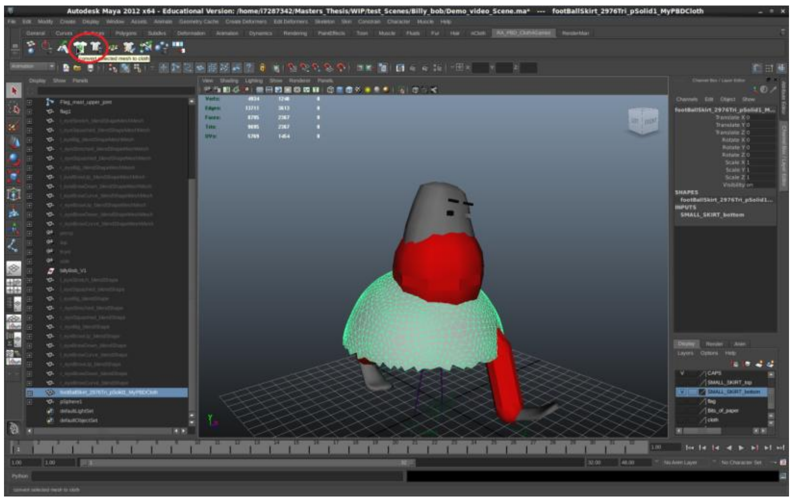

Autodesk Maya 2012 x64 - Educational Version: /home/i7287342/Masters_Thesis/WIP/test_Scenes/Billy_bob/Demo_video_Scene.ma* --- footBallSkirt_2976Tri_pSolid1_MyPBDCloth

Fig 30: illustrates a Maya scene with make cloth tools in action. The cloth mesh is highlighted.

4.2.3 Make attachment

This set of tools helps the artist in defining the attachment points on cloth mesh and the corresponding capsules to which the cloth can be attached to.

- **Point attachment:**

  The user is intended to select the set of vertices on a cloth mesh for using this tool. The attached vertices are colored black to help the artist visualize the attachment points made so far. Also the attachments can be made on more than one cloth mesh at a time.

  Pseudo code:

```
MakeAttachmentPoint()
    i.   Query the list of selected vertices using ls command
    ii.  If they are not on a cloth mesh, return
    iii. Get the position of the vertices using pointPosition Maya command
    iv.  Colour the vertices black using polyColorPerVertex Maya command
    v.   Save data into global Data structures
```

The attachment points are always appended to the data structure per cloth mesh making it more convenient to the user. The attachment points can be deleted completely on a cloth mesh using the *UndoMakeAttachmentPoint()* function. The user has to select the mesh whose attachment points have to be removed.

```
UndoMakeAttachmentPoint()
    i.   If the selected mesh is not a cloth mesh, return
    ii.  Rest the colour of the mesh using polyColorPerVertex Maya command
    iii. Clear the corresponding data structures
    iv.  Save data into global Data structures
```

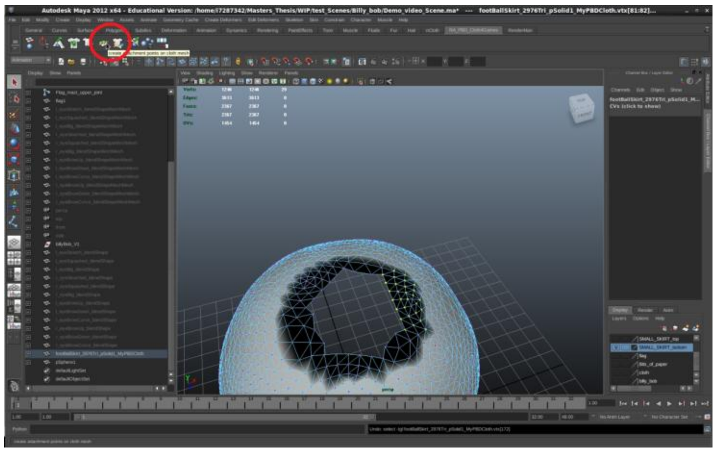


Fig 31: Illustrates a Maya scene with attachment points set on the cloth mesh

- **Capsule attachment**

  The user can attach more than one capsule to a cloth mesh using this tool. The inputs from the user must be a single cloth mesh and more than one capsule. The attachment points on the cloth mesh are mapped to the capsules based on the proximity of it from the capsule.

  Pseudo code:

MakeAttachmentToCapsule()

```
i.   Query the list of selected meshes using ls command
ii.  Query the list of selected capsules using ls command
iii. For every attachment point on the selected cloth
         minSqDist = 100000
         closestCaps = capsuleList[0]
         For every capsule from list
             sqDist = square distance between center of capsule and
             attachment point.
             If sqDist < minSqDist
                 minSqDist = sqDist
                 closestCaps = capsule
iv.  Save data into global Data structures
```

The attachment to the capsules can be removed if necessary using the Remove Attachment capsule tool. The user is intended to select a bunch of cloth meshes whose attachment to capsules have to be cleared.
Pseudo code:

RemoveAttachmentCapsule()

```
i.   If the selected mesh is not a cloth mesh, return
ii.  Clear the corresponding data structures
```

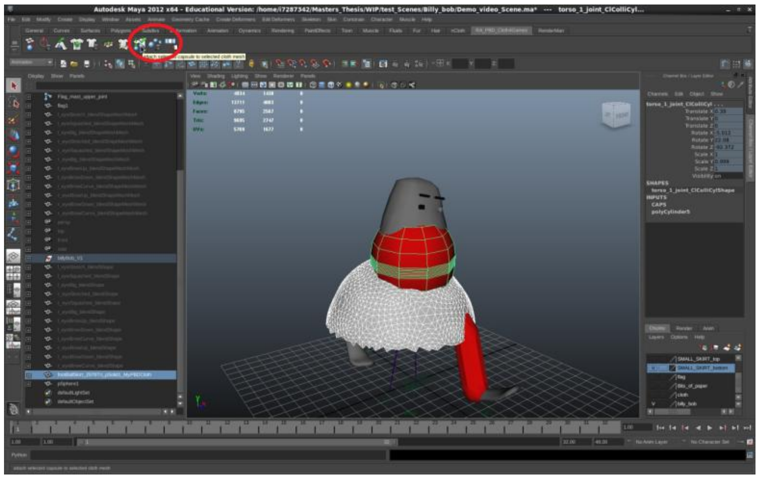

Autodesk Maya 2012 x64 - Educational Version: /home/i7287342/Masters_Thesis/WIP/test_Scenes/Billy_bob/Demo_video_Scene.ma* --- torso_1_joint_ClCollICyl...

Fig 32: Illustrates a Maya scene with Capsule and cloth selected that have been attached

4.2.4 File Export

This custom command exports all the setting made so far in the scene into a settings file. The format of the settings file is custom defined that is understandable by the standalone application. The format of the custom file is:

```
CAPS_DATA
COUNT 0
CAPSULE 0
        NAME joint1_ClColliCyl
        RADIUS 1.0
        TOTAL_FRAMES 1
        FRAME 0
                POS_A 1.0 1.0 1.0
                POS_B 1.0 1.0 1.0
        FRAME 1
                POS_A 1.0 1.0 1.0
                POS_B 1.0 1.0 1.0 . . .
CLOTH_MESH
COUNT 0
MESH 0
        NAME solid1_MyPBDCloth
        IS_ATTACHED 1
        POS 0 1.0 1.0 1.0
        POS 1 1.0 1.0 1.0
        CAPSULE 7
        POS 2 1.0 1.0 1.0
        POS 3 1.0 1.0 1.0
        CAPSULE 0 …..
```

During file export only the capsules and cloth meshes present in the scene are queried and their data is written out into settings file rather than saving all the data in the global data structures. Since Python Maya Commands was used, managing the data structures became easy using pythons Dictionary and Lists. The exported settings file is saved in the same location as the scene file with a tag name "_AR_PBDCloth4GameChar_Settings" appended to it.

4.3 Maya Import

The Maya data imported into standalone application is of two types – Settings file written out by the custom authoring Maya tools + collada export of the scene with the cloth meshes and an animated mesh. The export process of settings file was explained in previous section. The import process is explained below.

4.3.1 Maya Settings Parser

The settings file is parsed using boost tokenizer based on the format described in section 4.2.4. The parsed data is stored within the parsers data structure and queried later by the scene class. The data saved by the parser are tabulated below:

Capsule Data	• Total number of Capsules, • Capsule Name, • Total Number of Frames for which data is available • Per frame end position of capsule
Cloth Mesh Data	○ Total Number of cloth meshes ○ Mesh Name ○ Attached vertex position list on the cloth ○ Attached vertex Capsules list per cloth, for a set of attached points.

Table 4: List of asset data imported into application

4.3.2 ASSIMP Parser

Assimp (Open Asset Import Library) is a portable open source library to import various well known 3D model formats in a uniform manner. In this application, assimp is used to read the collada format files exported from Maya containing Cloth meshes and animated character skinned mesh.

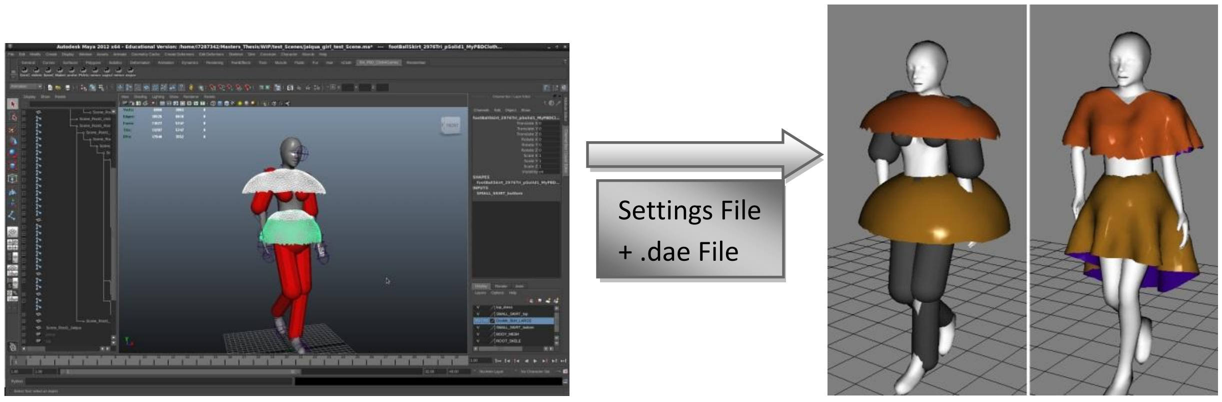


Fig 33: Illustrates the results achieved through the pipeline

The above image illustrates the result achieved through the pipeline. The assimp parser distinguishes cloth meshes from other meshes in the scene using the name of the mesh that was exported into the settings file. The cloth mesh is sent to the cloth simulator while the other meshes are just rendered.
The capsules are not exported in the .dae file. The information is read from the settings file and fed into the data structures of the simulator to perform the collision checks and to display them.
The attachment point positions read from the settings file is compared with the positions of the cloth mesh vertices read from assimp to identify them. This is not an efficient method but dues to the issues faced with the assimp importer the implementation adhered to it. A better solution would been to write out a binary texture map for

every cloth mesh with colour coding the attachment points from the rest of the vertices, which could not be achieved using assimp. The reason for that is explained in the next section.

Issues Faced

Among many tiny issues encountered following are some major ones.

- Assimp is an importer that post processes the data, ready to be rendered. Hence it duplicates vertices that are shared by more than one face to yield per vertex normal information to the users. In order to avoid the duplication, per vertex data such as normal and textures were stripped off during the import operation. This is the reason texture maps could not be used to store the attachment point information.

- During the .dae file export, transformations of the cloth meshes had to be frozen. If export as .obj, this was not necessary as OBJ preserved the transformations.

4.4 Managing Frame Rates

Managing Frame Rates was a crucial requirement for the project as it has to deal with three types of frame rates- Cloth Simulation FPS, Capsule Animation FPS and Render FPS and synchronize between them to achieve the desired results.

Before importing Maya animation into the application, the built-in capsule animation of the tool was running at the frame rate of the simulation. For a cloth with 1000 particles, the capsules animated at 30-35 FPS and for >2500 particles the capsules animated at 10-15 fps. But this should not be the case when animated capsule data is imported from Maya. The capsules should animate at the FPS set by the game character design artist in Maya and the cloth should behave accordingly. Also the render frame rate should be independent of the cloth simulation and capsule animation rates.

In order to handle this, firstly, the simulation of the capsule was decoupled from the cloth Simulation. Consecutively, the render rate was decoupled from the capsule Animation and the cloth simulation. Finally, the dt value sent to the simulator for integration was matched with the rate at which the simulation step is invoked. This is called fixing the time step which is very important for physical simulation and math intensive games (Real-time), where the project is both of the kinds. This is achieved as follows based on [5]:

- Fix the time step (dt) at which the simulation (cloth or capsule) has to run which can be determined as dt = 1/fps seconds . So if the cloth sim has to operate at 24 fps, set dt = 1/24 = 0.04 s.

- At every render cycle (standard being 30 FPS corresponding to a dt = 0.033 s), the simulate() function is called and updated positions are rendered. If simulate (update) functions takes more than 0.033s for instance for its computations, the display frame rate will fall below 30 FPS. Ideally the simulation should take exactly the same time as the render step which is never possible. In order to overcome this dt decided in the previous step comes in handy. This gives an extra time of 0.01 s = (0.04 – 0.03) s (10ms) per frame.

- If the simulation happens faster than 24 fps, many calls to Simulate function is skipped and interpolated values between previous and current positions are rendered instead. If the simulation is slower than 24 fps multiple calls to the simulate function is called per render cycle, again which affects the render rate. Hence the dt in step 1 should be set to *the worst case frame rate* of the simulation. Hence a dt = 0.1 s will cover all scenarios and the render rate will always remain the same.

  After implementing this, it was observed that, though the render rate remained constant for any type of cloth, the simulation was rendered very slow with the interpolated values instead of the original simulated values as many simulation steps were skipped as explained. The only option to overcome this is to accelerate the simulation using multi-threaded or GPU implementations.

  In the context this project, after multiple tests performed with the frame rates, the following results were inferred based on the observation illustrated in table 5 below.

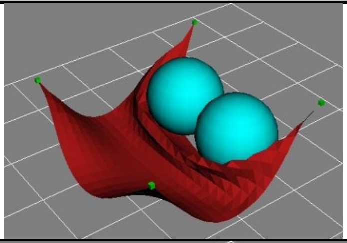
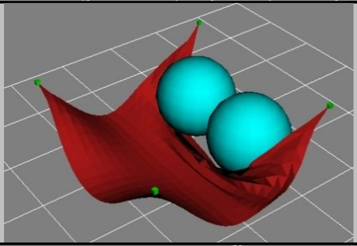
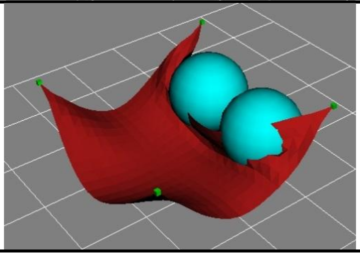
Render FPS = 60. Cloth grid of resolution 20 x 20 is used which simulates at 60 FPS			
Fixed Cloth FPS	Capsule FPS	OBSERVATIONS	Image Output
60	24	Perfect Cloth behaviour with proper wrinkles and folds generated	
24	24	Some artefacts visible.	
10	24	Capsule penetration into cloth is visible.	

Table 5: Results and observations made by varying the Capsule FPS and Cloth simulation FPS

Inference:

- The FPS of cloth Sim should be > FPS of Capsule Animation to avoid penetration while collision.
- If the simulation FPS drops, in-order to handle collision penetrations, the FPS of animation should be dropped which is undesirable. This drops the FPS of whole simulation.
- Irrespective of the above two scenarios, the render FPS can be maintained constant using the fix your time step method explained above. The fixed time step should be set such that, fixed dt <= simulation dt.

Having managed the frame rates in the system, following provisions are made in the application for the user to meddle with.

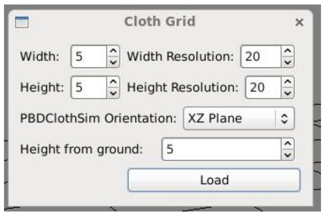
- Render FPS of the simulator can be set based on the FPS of the hosting system.
- FPS of the imported Maya animation can be doubled or halved. This feature is useful in testing the cloth behaviour with different FPS of the animation.
- Option provided to fix the time step of the simulation which can be done when the worst case FPS to the system is known which is dependent on the maximum load the user wants to impart for his requirement.

4.5 User Interface

The user interface was developed to be highly intuitive and Artist friendly. Images bellows demonstrate the various features of user interface developed for the application.

Dialogs:

- Create grid dialog for creating sample cloth grids useful for testing cloth behaviour (fig 34 (a)). It is a non modal dialog
- Create Capsule dialog for creating capsules for collision tests. The options provided in these dialogs are intuitive and self explanatory. It is a non modal dialog. (fig 34 (b)).



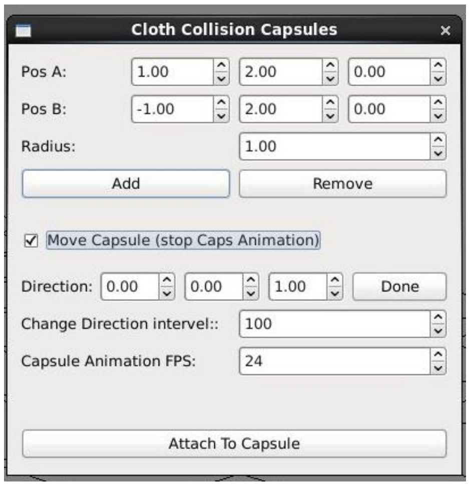
Cloth Grid

Width: 5 Width Resolution: 20

Height: 5 Height Resolution: 20

PBDClothSim Orientation: XZ Plane

Height from ground: 5



Cloth Collision Capsules

Pos A: 1.00 2.00 0.00

Pos B: -1.00 2.00 0.00

Radius: 1.00

☑ Move Capsule (stop Caps Animation)

Direction: 0.00 0.00 1.00

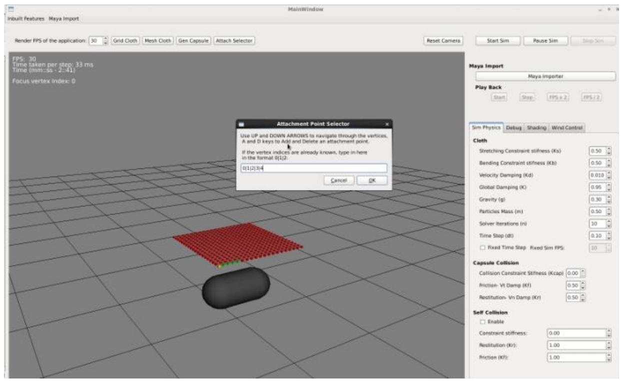
Change Direction intervel:: 100

Capsule Animation FPS: 24

(a) (b)

Fig 34: (a) Cloth grid Dialoge
(b) Create Capsule Dialog

- **Cloth Mesh** dialog for importing cloth mesh in obj or collada file formats. It’s a Modal dialog
- **Attach selector** dialog for setting attachment points on cloth grid and cloth mesh. It has the option of getting list of attachment point indices from user or intuitively selects the points using a point selector interface. It’s a modal dialog. Fig 35



Render FPS of the application: 30

FPS: 30
Time taken per step: 33 ms
Time (mm::ss - 2::41)

Focus vertex index: 0

Use UP and DOWN ARROWS to navigate through the vertices, A and D keys to Add and Delete an attachment point.

If the vertex indices are already known, type in here in the format 0|1|2:

0|1|2|3|4

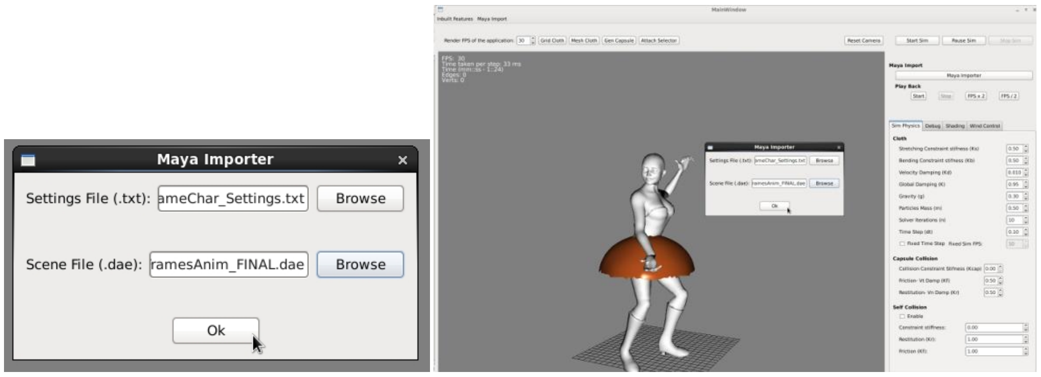
Cloth
Stretching Constraint stiffness (Ks): 0.50
Bending Constraint stiffness (Kb): 0.50
Velocity Damping (Kd): 0.010
Global Damping (K): 0.95
Gravity (g): 0.30
Particles Mass (m): 0.50
Solver Iterations (n): 10
Time Step (dt): 0.10
☐ Fixed Time Step   Fixed Sim FPS: 10

Capsule Collision
Collision Constraint Stifness (Kcap): 0.00
Friction- Vt Damp (Kf): 0.50
Restitution- Vn Damp (Kr): 0.50

Self Collision
☐ Enable
Constraint stiffness: 0.00
Restitution (Kr): 1.00
Friction (Kf): 1.00

Fig 35: Attachment points selector dialog

- **Maya Importer dialog and Play back options:** is used for importing the exported asset from Maya after cloth authoring. In their play back options, there is provision for the user to playback the animation faster or slower by doubling or halving the animation frame rate fig 36



Settings File (.txt): ameChar_Settings.txt

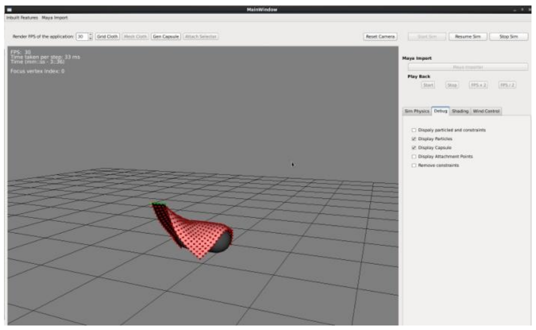
Scene File (.dae): ramesAnim_FINAL.dae

Fig 36: Maya Importer dialog

Tabs

There are four main tabs in the UI-

- **Physics tab** that contains all the physics related constants and coefficients of the simulation.
- **Debug tab** that has options for visualizing the simulation (fig 37)


(a)

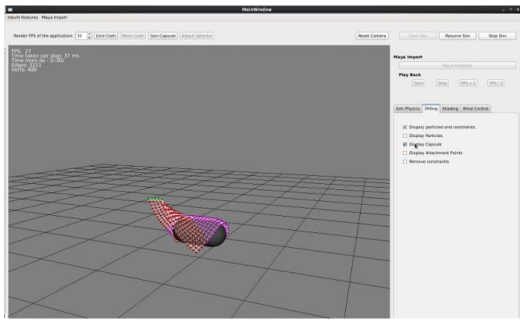

(b)

Fig 37: (a) Particle visualization
(b) Particles + constraint points visualization

- **Shading tab** that has shading options like smooth shading, glossy effect.
- **Wind Controls** Tab that provide options to enter wind control data like strength and direction.

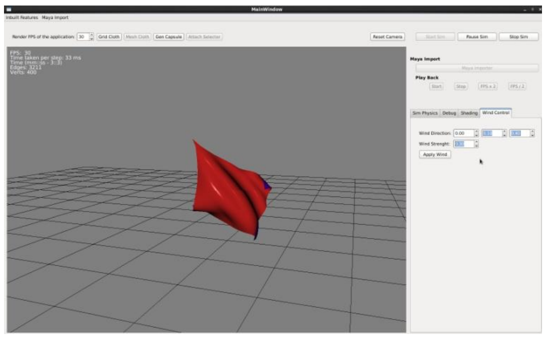


Fig 38: Wind Controller tab

Reset Camera button allows user to reset the camera to initial position as camera view changes based on the imported Maya asset.
Render FPS allows user to set the render frame rate of the system.
Menu Bar contains two menus, *Inbuilt Features* and *Maya Import.* The menu items in *Inbuilt features* menu is same as the dialogs already explained. The menu items in *Maya Import* menu is same as the Maya Importer button.

Chapter 5

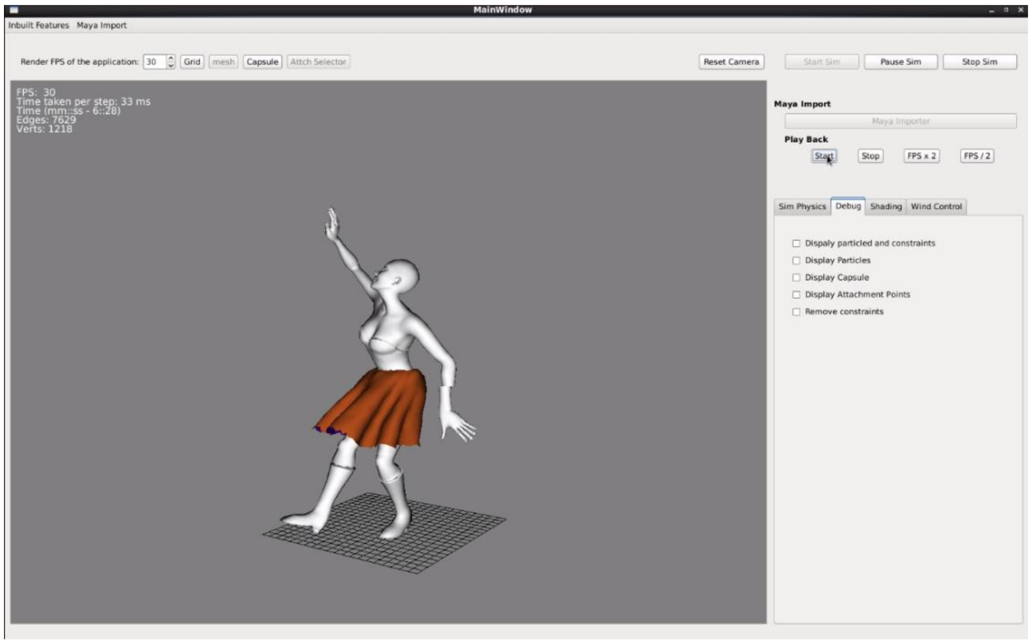
Results

5.1 Results from the application

The results generated using the simple pipeline implemented have been analysed in this section. The clothing of characters has been done on static and animated character meshes. The physics parameters have been tweaked for the various outputs based on the asset imported. The real time performance has been analysed which is dependent predominantly on the number of particles in the system, the number of cloth constraints and the presence of self-collision.

Animated Characters clothing outputs

i. Samba dance animation applied to animated mesh wearing a cloth of 1218 particles WITHOUT self collision is illustrated by image below. The frame rate of the animation is 6 FPS, which is halved twice the original FPS. This was necessary to achieve proper cloth response. The performance is quite high running at >30FPS speed. But collision with capsules failed during fast movements in the animation. Animations taken from [10]

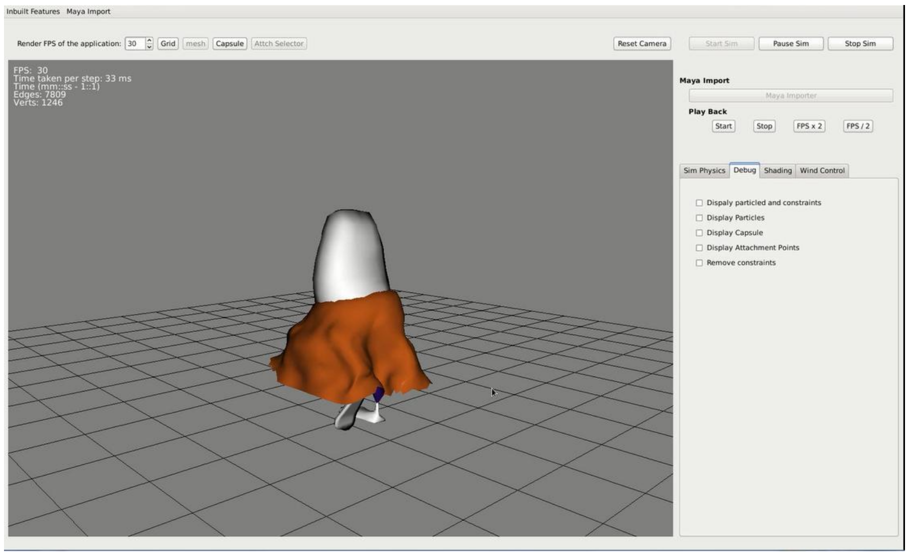


```
FPS:  30
Time taken per step: 33 ms
Time (mm::ss - 6::28)
Edges: 7629
Verts: 1218
```

Render FPS of the application: 30

Fig 39:

ii. The cloth behaviour on this animated character responses as expected in any animation frame rate it is run unlike the previous case. The performance achieved is the same as previous example. Hence the simulation highly depends on the nature of the asset which includes nature of animation, nature of character model and the physical settings such as mass, damping, gravity and constraint coefficients. Animation taken from NCCA assets.



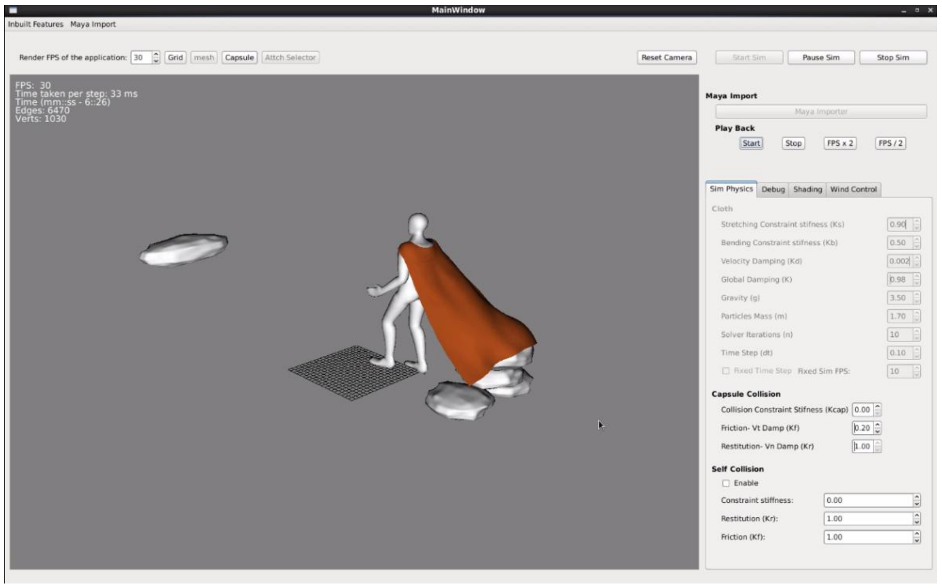
Render FPS of the application: 30

```
FPS:  30
Time taken per step: 33 ms
Time (mm::ss - 1::1)
Edges: 7809
Verts: 1246
```

☐ Dispaly particled and constraints
☐ Display Particles
☐ Display Capsule
☐ Display Attachment Points
☐ Remove constraints

Fig 40:

iii. Cape simulation on a walking character with 1030 particles WITHOUT SELF collision. This illustrates the ability of the application to interact with the environmental obstacles imported as an asset from Maya. Quite good performance of 30 FPS is achieved. Animation taken from Maya Visor.



Render FPS of the application: 30

```
FPS:  30
Time taken per step: 33 ms
Time (mm::ss - 6::26)
Edges: 6470
Verts: 1030
```

Cloth

Stretching Constraint stifness (Ks)	0.90
Bending Constraint stifness (Kb)	0.50
Velocity Damping (Kd)	0.002
Global Damping (K)	0.98
Gravity (g)	3.50
Particles Mass (m)	1.70
Solver Iterations (n)	10
Time Step (dt)	0.10
☐ Fixed Time Step Fixed Sim FPS:	10

Capsule Collision

Collision Constraint Stifness (Kcap)	0.00
Friction- Vt Damp (Kf)	0.20
Restitution- Vn Damp (Kr)	1.00

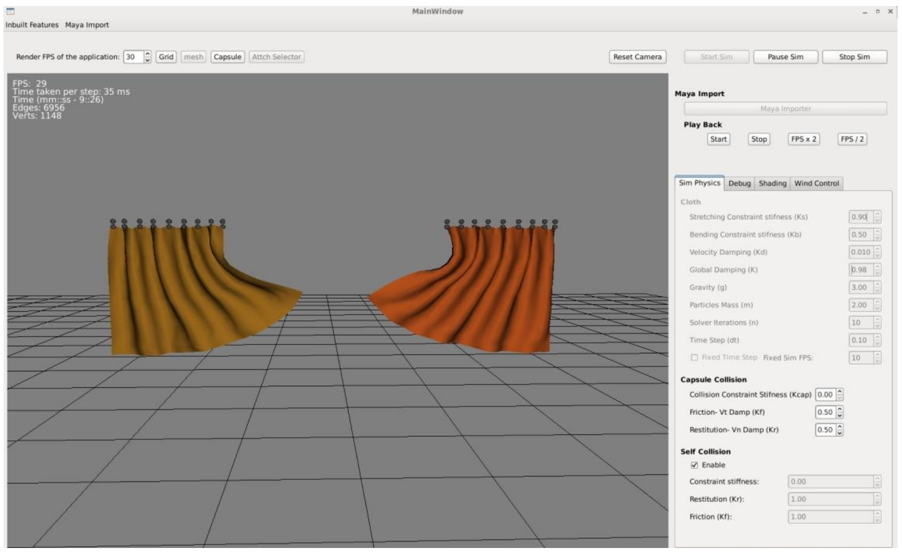
Self Collision

☐ Enable

Constraint stiffness:	0.00
Restitution (Kr):	1.00
Friction (Kf):	1.00

Fig 41

iv. Simulation of curtain with two cloth meshes loaded with 1148 vertices. The animation is running at 48FPS which is double the speed of its original frame rate that was imported from Maya. Simulation FPS of 29 is achieved WITH self collision. The cloth mesh was taken from Maya's nCloth demo.



Render FPS of the application: 30

```
FPS:  29
Time taken per step: 35 ms
Time (mm::ss - 9::26)
Edges: 6956
Verts: 1148
```

Maya Import

Play Back

Cloth

Stretching Constraint stifness (Ks)	0.90
Bending Constraint stifness (Kb)	0.50
Velocity Damping (Kd)	0.010
Global Damping (K)	0.98
Gravity (g)	3.00
Particles Mass (m)	2.00
Solver Iterations (n)	10
Time Step (dt)	0.10
☐ Fixed Time Step Fixed Sim FPS:	10

Capsule Collision

Collision Constraint Stifness (Kcap)	0.00
Friction- Vt Damp (Kf)	0.50
Restitution- Vn Damp (Kr)	0.50

Self Collision

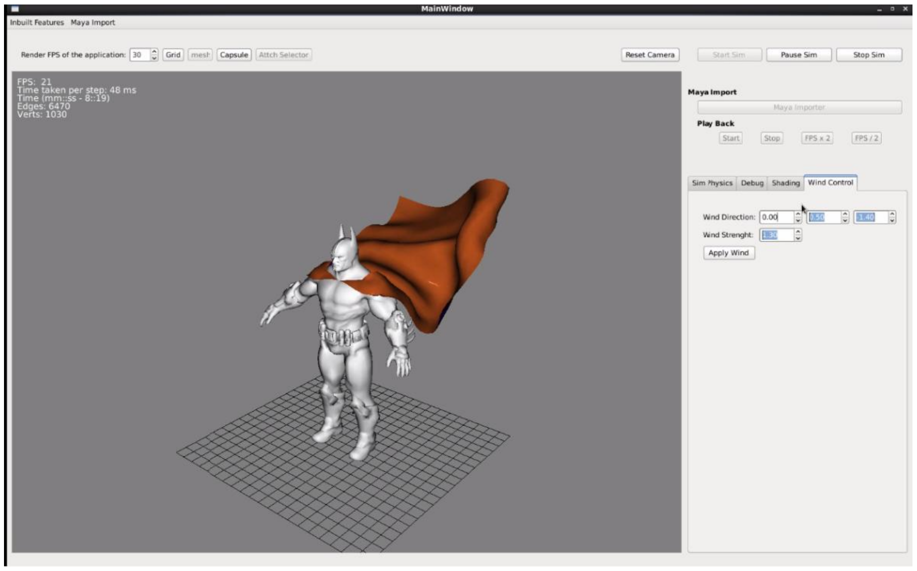
☑ Enable

Constraint stiffness:	0.00
Restitution (Kr):	1.00
Friction (Kf):	1.00

Fig 42:

Static Character clothing outputs

i. Cape simulation with self collision turned on. The system has 1030 particles and 6470 cloth constraints which simulated at 21 FPS. A wind effect has been applied to the cloth. Model taken from [17]



Render FPS of the application: 30

```
FPS:  21
Time taken per step: 48 ms
Time (mm::ss - 8::19)
Edges: 6470
Verts: 1030
```

Maya Import

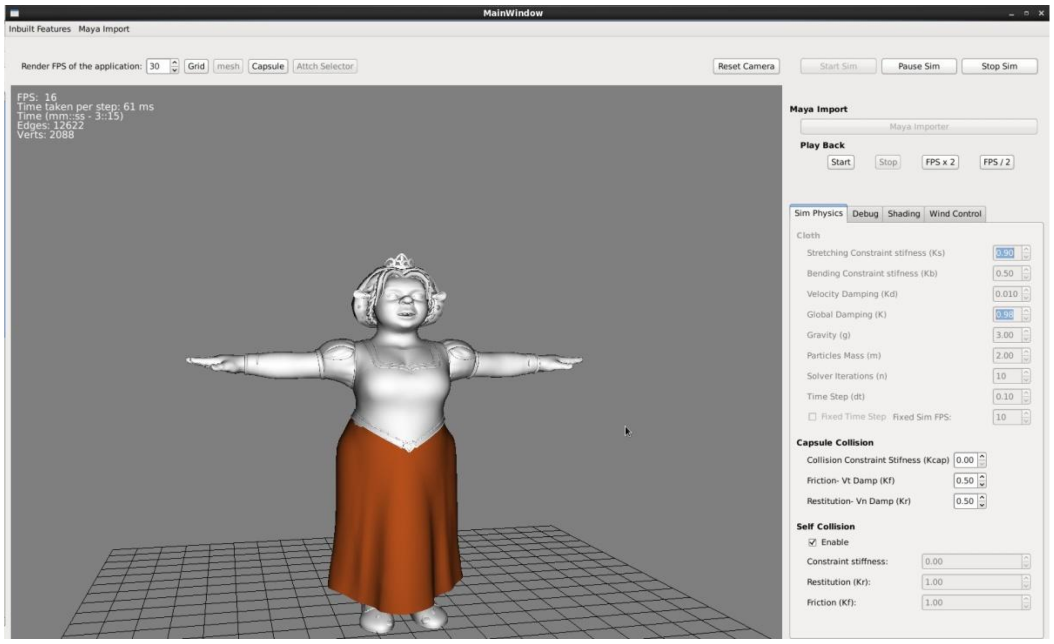
Play Back

Wind Direction: 0.00 0.50 1.40

Wind Strenght: 1.30

Fig 43:

ii. Skirt simulation with self collision turned on. The system has 2088 particles and 12622 cloth constraints running at 16 FPS. Model taken from [11]



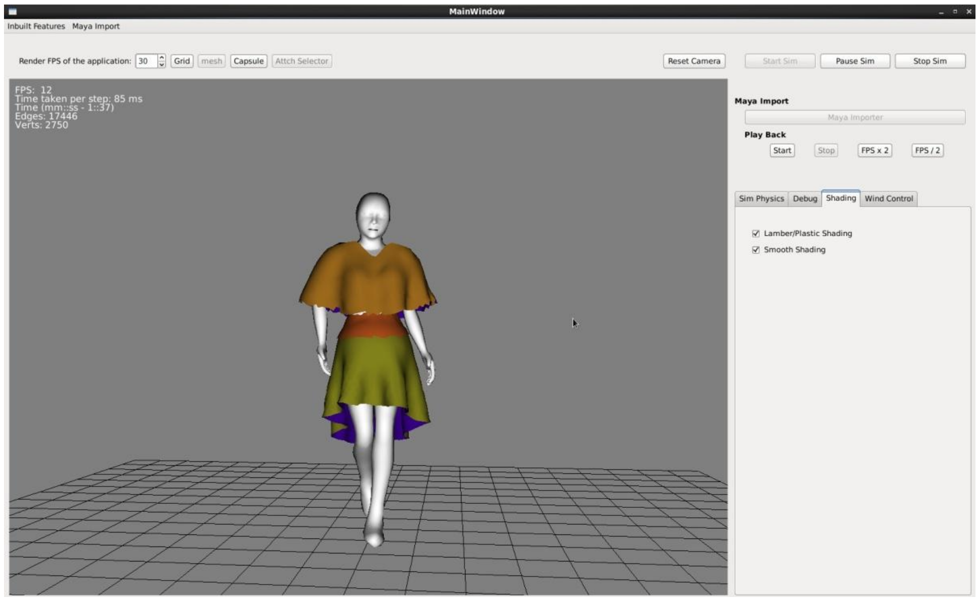
```
Render FPS of the application: 30

FPS:  16
Time taken per step: 61 ms
Time (mm::ss - 3::15)
Edges: 12622
Verts: 2088

Cloth
  Stretching Constraint stifness (Ks)          0.90
  Bending Constraint stifness (Kb)             0.50
  Velocity Damping (Kd)                        0.010
  Global Damping (K)                           0.98
  Gravity (g)                                  3.00
  Particles Mass (m)                           2.00
  Solver Iterations (n)                        10
  Time Step (dt)                               0.10
  ☐ Fixed Time Step  Fixed Sim FPS:            10
Capsule Collision
  Collision Constraint Stifness (Kcap)  0.00
  Friction- Vt Damp (Kf)                0.50
  Restitution- Vn Damp (Kr)             0.50
Self Collision
  ☑ Enable
  Constraint stiffness:   0.00
  Restitution (Kr):       1.00
  Friction (Kf):          1.00
```

Fig 44:

iii. Simulation of more than one cloth mesh on a character. The system has 2750 particles, 17446 cloth constraints, with self collision turned on between the cloth meshes, runs at 12 FPS. Model taken from NCCA assets.



```
Render FPS of the application: 30

FPS:  12
Time taken per step: 85 ms
Time (mm::ss - 1::37)
Edges: 17446
Verts: 2750

☑ Lamber/Plastic Shading
☑ Smooth Shading
```

Fig 45:

5.2 Comparison with existing tools

Cloth authoring tools

The image on the left is a ApeX clothing pugin for 3DS Max and the developed Maya cloth authoring window in the right. The collision capsules generated for the human character has been visualized in the developed tools in Maya, similar to the capsules generated in the Apex pulgin visualised in yellow wireframe.

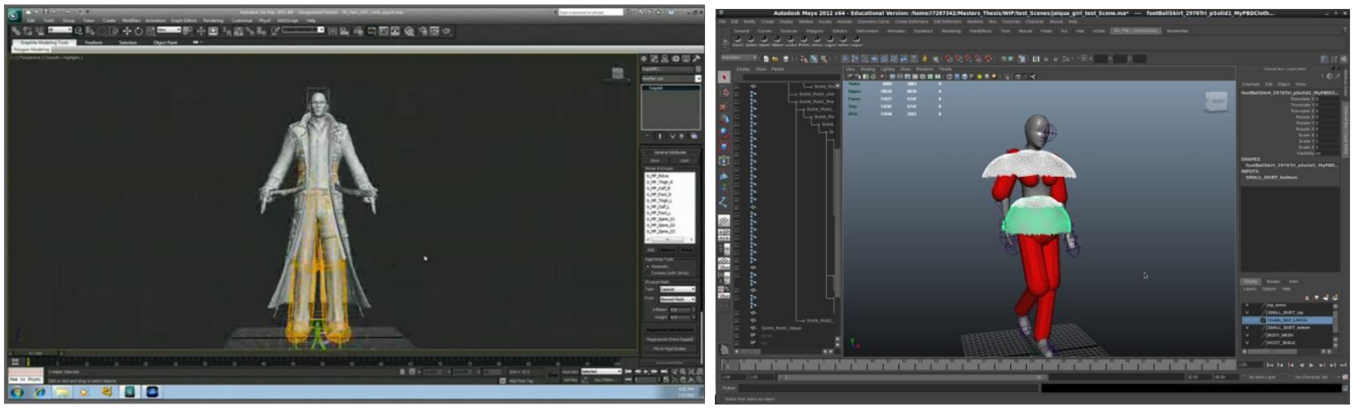


Fig 46: Apex Cloth authoring tool on the left and
the developed Maya authoring tool on the right.

Standalone application

The image on the left is Havok Previewer application that previews the assets imported from Havok Cloth authoring tool. The image on right is the developed application that simulates and visualizes the assets imported from the Maya cloth Authoring tools.

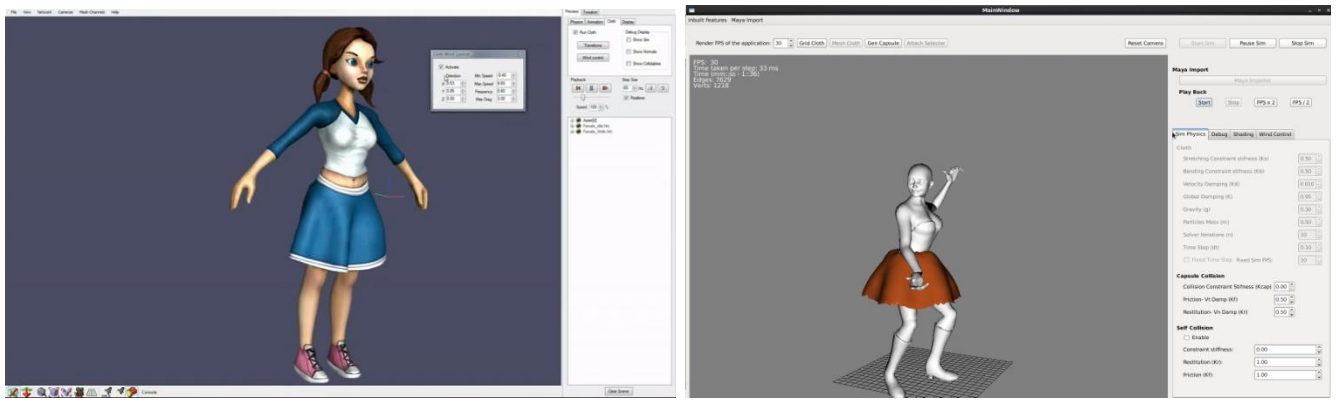


Fig 47: Havok asset viewer on the left and
the developed real time simulator Application on the right.

Chapter 6

Future work

The pipeline has scope for lots of extension and future works in both the application and cloth authoring tools. Some of the important points are listed below.

Cloth Authoring:

- A full fledge cloth authoring Plug-in with more user friendly tools can be developed like paint tools for setting attachment points.
- Options to set the cloth behaviour like cloth physics, can be provided with in the Maya plugin
- A preview of the simulation can be rendered with in Maya for easing the authoring process.
- When the scene is loaded back the settings should also be loaded back so that artist can resume his work from where he left.
- Texture maps to represent attachment points on cloth mesh rather than importing their position data which will become inefficient with larger assets.
- The cloth is currently attached to the moving capsule which is not an accurate method. The cloth should instead be attached to the animated mesh.

Stand alone application:

- Method such as Hierarchical PBD can be used to speed up the simulation.
- Cloth untangling to be handled with real time techniques.
- Handling stretchy cloth using Lang range Attachment method [9]. With increase in resolution of the cloth as illustrated by the images below for the same number of solver iterations the cloth stretches unrealistically

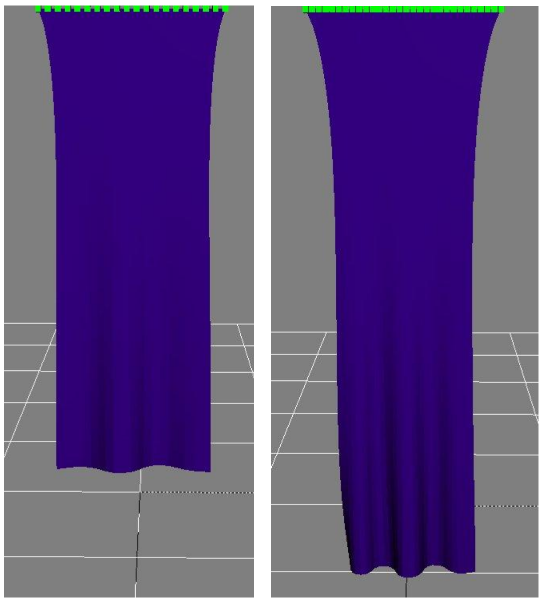

(a) (b)

Fig 48:

(a) Cloth Resolution: 30X60 is stretchy @ 20 iterations

(b) Cloth Resolution: 20X40 @ 20 iterations

- The implementation is easily extensible to run on multithreaded system because of the parallelism of cloth models and the PBD methodologies. Thus can be extended to run on GPUs or any multi threaded architecture that will improve the efficiency.
- Textures can be added to the cloth and the models

Chapter 7

Conclusion

With the duration of three months provided for the project, the time had to be utilized efficiently. Thus sufficient time of two weeks was allocated for the initial research of the project. This project being an extension of the Persona inquiry course, the initial research helped in laying a very strong understanding of the concepts and the background studies. With a clear goal in mind the development process was subdivided into smaller tasks and accomplished one by one.

With no prior experience with Maya Commands, learning and implementing it was a challenge that was accomplished in a week. By the end of the Maya tools development, it was fun to use the custom developed tools and generate the desired outputs using the pipeline as portrayed in the results section and the videos submitted. Many bugs were encountered during the process making the code cleaner and more reliable iteratively.

With the code base size increasing after two months of continues development, the time spent on initial structuring of the code based on design patterns proved to be very useful for projects of such size. The code structure has emerged to be highly flexible and extendible especially with Position Based dynamics simulations. The application can be extended into a complete Position Based Dynamics Engine.

On the whole the project development period had been a complete learning process with improved understanding on real-time simulations and graphics pipeline. The accomplishment of the desired results has given a feeling of success and content though the urge on extending the product further in making it as equipped as the industry standard tool like Apex and Havok keeps the mind ablaze.

References

[1] Baraff, D., Witkin, A., Kass, M., 2003. Untangling cloth. ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2003, 22 (3), 862-870.

[2] Bridson, R., Fedkiw, R., Anderson, J., 2002. Robust Treatment of Collisions, Contact and Friction for Cloth Animation. ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2002, 21 (3), 594-603.

[3] Darwin., (2001). Darwin 3D Game tech articles. Available: http://www.darwin3d.com/gamedev/articles/col0599.pdf Last accessed 10 May 2013.

[4] Ericson, C., 2005. Real Time Collision Detection. San Francisco: Morgan Kaufmann Publishers.

[5] Glenn Fiedler, 2006, Fix Your Timestep!.Gafferongames,2 September 2006.Avaialble from: http://gafferongames.com/game-physics/fix-your-timestep/ [Accessed 17 July 2013]

[6] Goldenthal, R., Harmon, D., Fattal, R., Bercovier, M., Grinspun, E., 2007. Efficient simulation of inextensible cloth. ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2007, 26 (3).

[7] Kavana,L., Gerszewski,D., Bargteil, A.W., Sloan,P.P., 2011, Physics-Inspired Upsampling for Cloth Simulation in Games, Computer Graphics Proceedings, Annual Conference Series, 29(5)

[8] Kelager, M., Niebe S., Erleben ,K., 2010, A Triangle Bending Constraint Model for Position-Based Dynamics. In: Bender.J Ed. Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2010).

[9] Kim, T.Y., Chentanez, N., Muller, M., 2012. Long Range Attachments - A Method to Simulate Inextensible Clothing in Computer Games. In: Kry, P., Lee, J., eds. Eurographics/ ACM SIGGRAPH Symposium on Computer Animation, July 2012 Switzerland. Aire-la-Ville: Eurographics Association, 305-31

[10]mixamo. , 2013. Animation in seconds: Mixamo.org. Available from http://www.mixamo.com/motions: [Accessed 4 August 2013].

[11]model3DFree,2013. Shrek in the model Fiona: 3DModelFree.com. Available from http://www.3dmodelfree.com/models/26931-0.htm: [Accessed 4 August 2013]

[12]Muller, M., Chentanez, N., 2010. Wrinkle meshes. In: Otaduy, M., Popovic, Z., eds. Eurographics/ ACM SIGGRAPH Symposium on Computer Animation 2010, July 2010 Spain. Aire-la-Ville: Eurographics Association, 85-92.

[13] Muller, M., Heidelberger, B., Hennix, M., Ratcliff, J., 2007. Position Based Dynamics. Journal of Visual Communication and Image Representation, 18 (2), 109-118.

[14]Rosenberg,O., 2011. OpenCL Overview. Beaverton: Khronos Group. Available from: http://www.khronos.org/assets/uploads/developers/library/overview/opencl-overview.pdf [Accessed 13 June 2013].

[15] Surazhsky, V., Surazhsky, T., Kirsanov, D., Gortler, S.J., Hoppe, H., 2005. Fast exact and approximate geodesics on meshes. ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2005, 24 (3), 553-560.

[16]Teschner, M., Heidelberger, B., Muller, M., Pomeranets, D., Gross, M., Optimized Spatial Hashing for Collision Detection of Deformable Objects. In: Ertl, T., ed. Proceedings of the Vision, Modeling, and Visualization Conference, 19-21 November 2003 Munich. Germany: Aka GmbH, 47-54.

[17]Tf3DM., 2013. Batman 3d model: tf3dm.com. Available from http://tf3dm.com/3d-model/batman-50670.html: [Accessed 4 August 2013].

[18] Thunderfirst, 2012. Capsule-Capsule Collision in Games. Blogger. 10 February 2012. Available from: http://thunderfist-podium.blogspot.co.uk/2012/02/capsule-capsule-collision-in-games.html [Accessed 6 June 2013].

[19]Vassilev, T.I., 2010. Comparison of several parallel API for cloth modelling on modern GPUs. In: Rachev, B., ed. 11th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing on International Conference on Computer Systems and Technologies, 17-18 June 2010 Bulgaria. New York: ACM, 131-136.

[20] Yu ,Feng, Kim, 2011. A deformation transformer for real-time cloth animation. ACM Transactions on Graphics (TOG), 29(4).

[21]Zogrim, 2011. Cloth Simulation Solutions for games. PhysXInfo, 22 March 2011. Available from: http://physxinfo.com/articles/?page_id=389 [Accessed 11 June 2013].