Position Based Dynamic Cloth Simulation

A Simple Position Based Dynamics System For Modeling Fabric

Matt Stewart
University of Victoria
Victoria, British Columbia
stewartm@uvic.ca

ABSTRACT

This paper is a discussion of the solution and results of a simple position-based dynamic simulation for modeling fabric in real-time, the final project for Computer Science 473, Fall 2018.

ACM Reference Format:
Matt Stewart. 2018. Position Based Dynamic Cloth Simulation: A Simple Position Based Dynamics System For Modeling Fabric. In *Proceedings of Computer Science 473 (CSC473 '18).* ACM, New York, NY, USA, 3 pages. https://doi.org/00.000/000_0

1 INTRODUCTION

While traditional force-based simulations are able to render fabric and cloth in reasonable detail for real-time applications, historically there has been difficulty in matching the effects of these simulations to their real-world counterparts. A position based dynamics approach, modeling a mesh of particles and utilizing a constraint solver to predict and average positional data, can more realistically model folds, collisions, the effects of external forces such as wind and gravity, and stretching and tearing of the fabric, without leading to unrealistic behaviour such as clipping, jittering, and self-intersection.

This project generates an environment for modeling simple, and potentially more complex, fabric simulation in real-time using position-based dynamics and constraint satisfaction.

1.1 Method

The method follows the pattern set by Müller et al. [2006] for creating a position based dynamics system, with a focus on constraints necessary for realistic cloth simulation as well as rudimentary collision simulation to model a simple object for the cloth to interact with. The calculation pipeline follows Algorithm 1 from Bender et al. [2017][1]

Constraints are handled with a non-linear Gauss-Seidel solver (likely non-hierarchical, as there are will be relatively few constraints to solve for). Constraints handled are distance preservation between adjacent cloth vertices and collisions with a sphere.

1.2 Expectations

At minimum, the simulation will be able to handle natural hanging due to gravity and collisions with another object in the scene. Ideally (time permitting), it will realistically model folding behaviour when dropped or moved, the effects of external forces other than gravity, natural movement and hanging due to the attachment of heavier rigid bodies, stretching of the fabric according to a variable elasticity, tearing of the fabric when stretched above a tensile threshold, spherical volume-preserving elastic cloth "balloons", and potentially further features as they are considered.

It is expected that modeling constraints realistically will be the primary challenge in this project, and success in modeling each constraint in turn will allow for the potential of others to be added, while also unfortunately contributing to slow-down of the entire system.

Algorithm 1 Position-based-dynamics

```
1: for all vertices i do
2:     initialize
3: end for
4: loop
5:     for all vertices i do vi ← vi + δtwifext(xi)
6:     for all vertices i do pi ← xi + δtvi
7:     for all vertices i do genCollConstraints(xi → pi)
8:     loop solverIteration times
9:         projectConstraints(C1, ... ,C(m+mColl), p1, ... , pn)
10:    end loop
11:    for all vertices i do
12:        vi ← (pi - xi) / δt
13:        xi ← pi
14:    end for
15:    velocityUpdate(v1, ..., vn)
16: end loop
```

2 SOLUTION

2.1 Outline

The system is modeled in OpenGL and C++ using the Atlas framework, available at https://github.com/marovira/atlas. It follows Algorithm 1 from Bender et al. [2017], in which a mesh of particles are created, representing vertices at equal distances on the surface of the cloth. At each time step, the system sums the forces acting on each particle and applies them to that particle's velocity, then uses that velocity to generate an initial estimate for the particle's updated position.

The estimate is then refined iteratively to better conform to the constraints imposed upon it by neighbours, collisions, etc., arriving at a final approximation. The particle's velocity is updated according to the distance between the previous position and this

[1] Apologies for the lack of subscripts and proper mathematical notation; Overleaf appears to have issues with inserting certain mathematical statements inside tables.

CSC473 '18, December 2018, University of Victoria
2018. ACM ISBN 000-0000-00-000/00/00…$-
https://doi.org/00.000/000_0

approximation, and the final position is updated to match the approximation. Successive iterations should bring the approximation closer in line with the ideal positions according to the constraints.

2.2 Details

The program consists of four relevant C++ files: Particle.cpp, Cloth.cpp, ClothScene.cpp, and Sphere.cpp.

Particle.cpp defines the Particle class, with data members mMass (float), mPosition (atlas::math::Point), mPrediction (atlas::math::Point), mVelocity (atlas::math::Vector), and mMovable (bool). It contains one member function, setMovable(bool b), which indicates whether the Particle object is affected by constraints and forces acting on it during the simulation loop, or is fixed in space.

Cloth.cpp defines the Cloth class, with data members mPosition (atlas::math::Point), mMass (float), mWidth (float), mLength (float), mHeight (float), mG (float), mRest (float), and mRadius (float). In addition to the Atlas/OpenGL wrapper functions necessary for rendering, updating, and resetting the object's geometry, it contains member functions setPosition (atlas::math::Point const pos), mag (atlas::math::Vector v), which returns the magnitude of the input vector as a float, and constrainDistance (int p1, int p2), which updates the position predictions of Particles at indices p1 and p2 according to the current and rest distances between them.

A Cloth object's constructor method creates a VAO, vertex buffer, and index buffer for a single sphere which will be rendered at each vertex of the cloth. It creates a grid mesh for the cloth, represented by a 1-dimensional vector of Particle objects of length mWidth*mLength, each with a mass equal to mMass/mWidth*mLength and a position in 3D space whose x and z values represent its position in the row and column of the mesh, and whose y value is mHeight.

In the Cloth object's updateGeometry method, for each Particle in the vector grid, a predicted velocity is calculated based on the forces acting on it (gravity, unless the Particle is designated immovable), and a predicted position is calculated based on the predicted velocity. Both predictions use Symplectic Euler integration to arrive at a rough estimate for each.

Over a fixed number of iterations (currently 100), this estimate is then procedurally refined by applying the system's constraints, constrainDistance() and a collision test with a large sphere. constrainDistance() acts upon each Particle object and compares it to 2-4 of its neighbours (depending on its position in the mesh; Particles at the corners have only 2 adjacent neighbours, and particles on the edges have only 3).

The distance constraint is enforced by measuring the current magnitude of the length of the line segment between each Particle and each of its neighbours, and producing a correction vector by multiplying that line segment by the ratio between the magnitude and the cloth's natural resting distance.

To update both Particles' prediction symmetrically, the correction vector is halved, and both predictions are brought closer together by the resulting half correction vector. Each pair of particles is compared twice in each step (eg. Particle 1 compared with Particle 2, then Particle 2 compared with Particle 1 later), for a total of 200 comparisons for each shared edge.

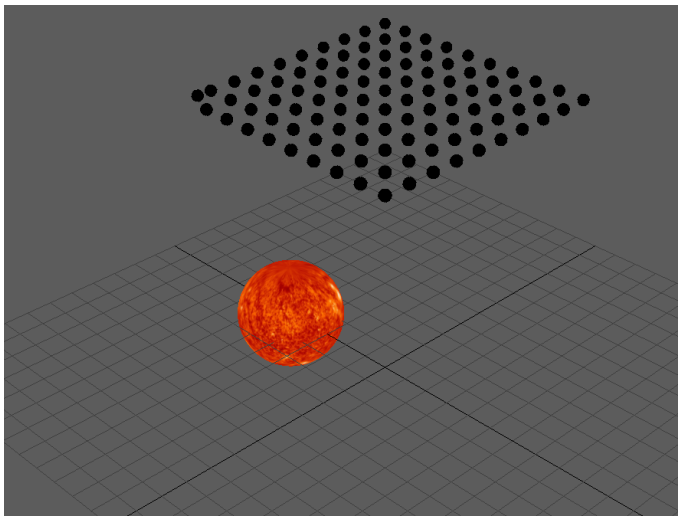
The collision constraint is modeled by testing if any Particle is within the radius of the large sphere, and projecting its prediction outward onto the surface of the sphere if that is the case. After 100 iterations, the resulting prediction is used to update the velocity vector and the position of each Particle.

ClothScene.cpp instantiates and renders a scene involving one Cloth object, comprised of Particle objects, and one Sphere object.

Sphere.cpp defines a simple Sphere class for rendering purposes, with a centre and a radius. Although the sphere rendered in the scene is a Sphere object, the collisions generated with the sphere are in fact based on the non-rendered sphere defined by Cloth.mPosition and Cloth.mRadius, for simplicity's sake.

3 RESULTS

Figure 1: Initial Conditions



The simulation renders a variable-size grid (modified by Cloth.mWidth and Cloth.mLength, both set to 10) of Particle spheres at a point above the larger sphere indicated by Cloth.mHeight (set to 10.0). The Particles at the highest left and right corners of the grid are fixed in place. When the simulation is run, the movable Particles in the grid are pulled downwards by gravity and constrained in their motion by the fixed corners, resulting in realistic and emergent folding and rippling behaviours in the whole fabric. At the same time, the bottom edges of the cloth collide with the larger sphere, draping over it in a realistic manner and swaying as directed by the distance constraint.

A further simulation involves the same initial conditions, but no fixed points in the cloth grid. Instead, when the simulation is run, the entire cloth drops to the plane at y=0, where it collides with the larger sphere and drapes over it entirely. Since friction is not modeled in the system, this collision proves quite unrealistic, and the cloth eventually slides off the sphere and continues to slide out of the scene unimpeded.

Figure 2: Gravity And Collisions With 2 Fixed Points

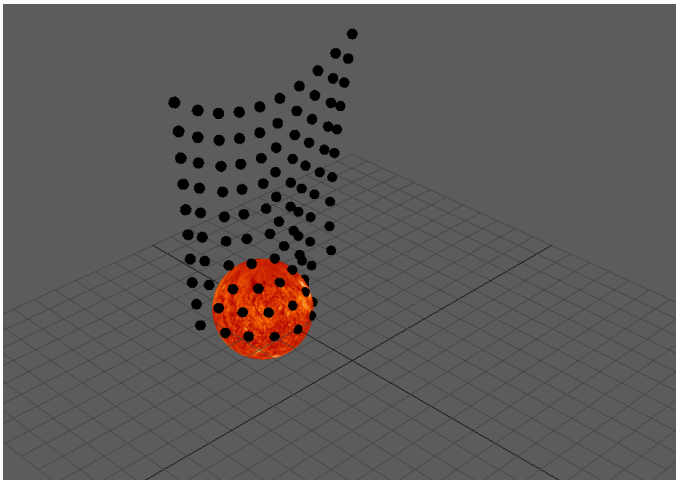
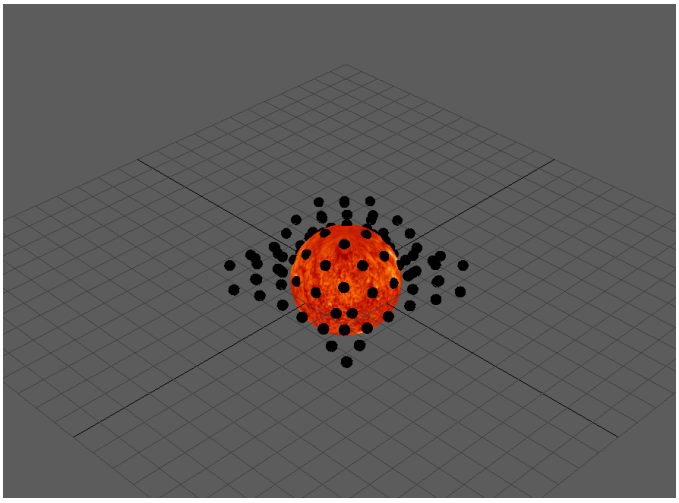


Figure 3: Gravity And Collisions With No Fixed Points



3.1 Discussion

Realistic behaviour in the system (absent friction and further constraints) - or more accurately, *verisimilitudinous* behaviour - is dependent entirely on the number of iterations performed in the constraint satisfaction loop. The more the distance constraint is checked and the positions updated, the closer they come to a near-perfect approximation of the "best" position the cloth vertices can be in to satisfy the constraints imposed on them. Naturally, further iterations also increases the slowdown of the system, since the distance constraint involves a nested pair of for loops and thus performs in $O(n^2)$ time.

Further, a finer grid of Particle objects would more closely approximate the behaviours of a real continuous fabric mesh, but again would drastically slow down the computations during the constraint satisfaction loop.

4 CONCLUSION

The position based dynamics approach to modeling cloth particle meshes and solving constraints to predict and average positional data has the potential to more easily and more realistically model forces and constraints in a physical system than the standard velocity based approach. Though the satisfaction of constraints via mathematical solvers can introduce further complexity and inefficiency into the simulation, for simple constraints and simple forces a position-based system would appear to be a promising choice.

4.1 Future Work

The obvious direction to pursue for the continuation of this project would be the introduction of additional forces and constraints into the system. Friction, wind, stretching, bending (including isometric bending), natural movement and hanging due to the attachment of heavier rigid bodies, tearing of the fabric when stretched above a tensile threshold, self-collision forbiddance, and volume-preservation could all be added and modeled to produce a more robust, and potentially more verisimilitudinous fabric simulation.

The system would also benefit from the addition of a hierarchical data structure to better handle constraint satisfaction and collision detection without requiring checks between objects spaced too far apart in the scene to be at risk of colliding.

REFERENCES

[1] Müller M., Heidelberger B., Hennix M., Ratcliff J. (2006), "Position Based Dynamics," VRIPHYS 2006.
[2] Bender J., Müller M., Macklin M. (2017), "A Survey on Position Based Dynamics," EUROGRAPHICS 2017.
[3] Macklin M, Müller M., Chentanez N., Kim T. (2017), "Unified Particle Physics for Real-Time Applications," ACM TOG 33(4), 2014.