Real Time Cloth Simulation using Extended Position Based Dynamics

Matthew Lenathen [1]

[1]Edinburgh Napier University

Summary

This project aimed to implement Extended Position Based Dynamics (XPBD), a specific method of simulating cloth, within the Unity physics framework. It would be implemented alongside a mass-spring system and the original Position Based Dynamics (PBD) to compare each method. The findings revealed that XPBD improved upon PBD by making stiffness independent of time step. It also fixes the issue of solver iterations affecting stiffness by replacing them with substeps. Overall, XPBD is a very useful method in simulating cloth and other deformable bodies.

Introduction

Realistic cloth simulation is integral to a number of applications across many industries such as film, games development and virtual reality. Combining maths, classical mechanics and programming, cloth simulation aims to accurately replicate the behaviour of these objects in real time environments. Among the many approaches available, Extended Position Based Dynamics (XPBD) stands out due to its simplicity and robustness in simulating cloth and other deformable bodies. This poster presents the implementation and results of utilising this method within the Unity physics simulation framework. Unity scripts are written in C#, but a C++ dll plugin was written to handle the simulation code, as C++ is much more efficient with vector calculations.

Methodology

In physics based animation, objects are represented with vertices that make up a mesh that is displayed to the screen. PBD and XPBD treat these vertices as particles with properties such as mass, position and velocity. Using the forces acting on the particles such as gravity and wind, the predicted particle position can be calculated. All constraints are then worked on, and a new velocity is calculated by taking the difference of the current particle position, and the predicted particle position divided by delta time. Delta time is fixed at 0.02 seconds but experiments were performed at different time steps.

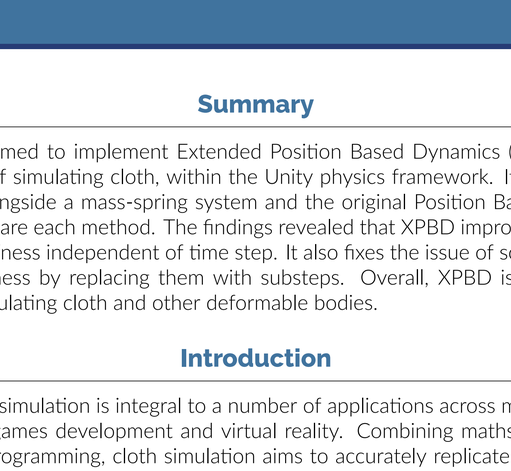


Figure 1. A wireframe of a cloth mesh.

Constraints are an important part of this method, it creates a relationship between particles that are close together and aims to keep these particles together during the simulation. It achieves this by working on the predicted position during the simulation to manoeuvrer the particles to satisfy the condition. In this case, the distance between particles must be equal to the rest length.

Figure 2. Projection of a constraint between two particles. Figure adapted from [1].

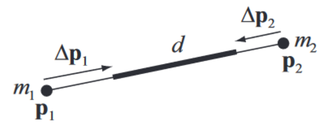


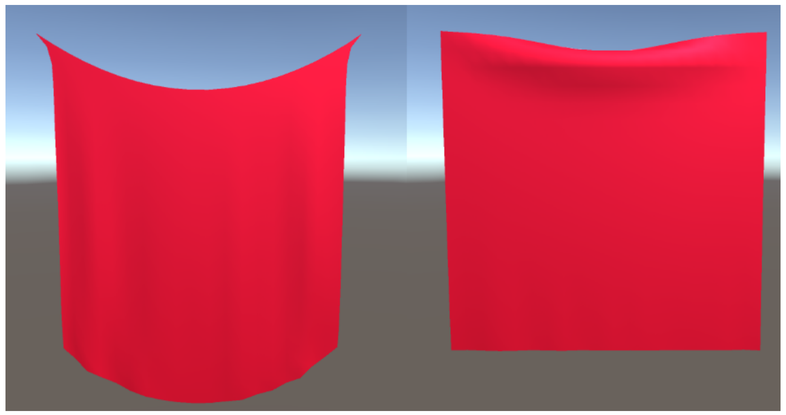
Figure 2 shows the general idea behind constraints. During the current time step of the simulation, the predicted positions are modified by some vector Δp so that they better satisfy the distance constraint d. In XPBD, the simulation loop iterates through multiple substeps. Each substep involves recalculating the predicted positions and adjusting them to satisfy the constraints more accurately. This iterative process ensures the constraints are satisfied to a higher degree and results in a more precise and accurate simulation.
The distance corrections Δp_1 and Δp_2 are calculated as:

$$\Delta p_1 = -\frac{w_1}{w_1 + w_2 + \frac{\alpha}{\Delta t^2}}(|p_1 - p_2| - d)\frac{p_1 - p_2}{|p_1 - p_2|}$$

$$\Delta p_2 = +\frac{w_2}{w_1 + w_2 + \frac{\alpha}{\Delta t^2}}(|p_1 - p_2| - d)\frac{p_1 - p_2}{|p_1 - p_2|}$$

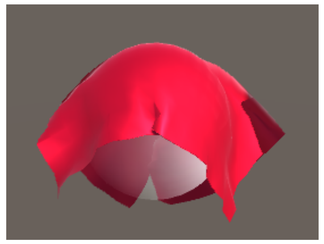
Where w is the inverse mass of the particles, and d is the rest or goal distance, and α is the compliance factor.
The stiffness of the simulation can be adjusted by modifying the compliance factor. The simulation is infinitely stiff when $\alpha = 0$, the effects of stiffness can be seen in Figure 3.

Figure 3. Two cloths with different stiffness values



The user is also able to interact with the cloth by clicking and dragging it. This works by casting a ray from the camera to the cloth, finding the nearest particle to the click and making it static. This functionality enables users to test extreme movements and assess the behaviour of the cloth simulation more effectively.
Another scenario was also implemented with collision code, where a horizontal cloth falls and collides with a sphere. This was to test how well the constraints perform when collisions are also happening in the same time step.
Collision works by checking if the distance from the particle to the centre of the sphere, is lower than the sphere's radius, indicating the particle has collided with the surface. It then calculates a vector to adjust the particle's predicted position out of the sphere. This scenario can be seen in Figure 4.

Figure 4. The horizontal cloth falling on a static sphere



Results

Once all three methods were implemented they were tested through different experiments. The mass-spring system is very unstable at high stiffness and can cause the simulation to explode, whereas PBD and XPBD are unconditionally stable. The mass-spring system was implemented in both C# and C++ to gauge how much faster C++ was, and it was at least a 60fps gain in most scenarios.

The time step stiffness issue with the original PBD is completely solved with the compliance factor in XPBD, saving the developer from manually tuning the stiffness of the system for different time steps. The solver iterations of PBD is also fixed in XPBD by utilising substeps for the whole simulation loop, instead of just looping over the constraints.

A recorder feature was implemented that allows many experiments to be set up in code, and the recorder will queue them up and record videos of each experiment. This was useful in observing how the cloth changes when variables such as substeps, compliance and time step are changed.

Conclusion

In this project, multiple cloth simulation methods were implemented and tested, the mass-spring system, Position Based Dynamics and Extended Position Based Dynamics. The results suggest that the two position based methods offer significant advantages over traditional methods such as increased stability and performance. The implementation of XPBD successfully resolves the long-standing issues present in PBD, such as time step dependant stiffness. The integration of a C++ simulation within the C# Unity environment allowed for the utilisation of helpful features provided in Unity, while retaining the improved performance of using C++.

Future Work

While the base XPBD implementation is good, there are some improvements that could be made with future work. One such improvement would be self collision. This would allow the cloth to interact and collide with itself, lending to increased realism. It is not a trivial problem to solve and one such solution could be the use of spatial hash maps to efficiently check which particles are colliding.

Another scenario could be implemented to further test the method. An example scenario could be clothing on a game character such as a dress. This would test how the cloth would look like in animation or a game. This would most likely require self collision first.

References

[1] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007.