The application and algorithm of fabric simulation in game industry

Shengjie Zhang

Donald Bren School of Information & Computer Sciences, University of California, Irvine, United States

shengjz9@uci.edu

Abstract. fabric simulation has emerged as a crucial aspect of computer graphics, particularly in applications such as film production, video games, and design. This paper provides a comprehensive review of fabric simulation methodologies, implementation techniques, and its diverse applications within the game industry. The fabric simulation algorithms are broken down into Fabric Modeling, Dynamic Simulation, and Collision Handling. The three common modeling strategies of Elasticity-Based Models, Particle-Based Models, and Mass-Spring Damper Models, are discussed in detail. Additionally, dynamic simulation strategies such as Position-Based Dynamics (PBD) and Extended Position-Based Dynamics (XPBD) are explored for their role in real-time simulations, particularly in the game industry. The paper also delves into collision handling algorithms, exploring the strategies for accurate collision detection and response. The application of fabric simulation in the game industry is highlighted, showcasing how it enhances realism and immersion by simulating fabric movement and interactions with the environment. The paper introduced various fabric simulation tools and systems within popular game engines like Unity 3D and Unreal Engine 5. Furthermore, optimization strategies, such as view-dependent adaptive simulation, are discussed to improve performance providing insight for future work.

Keywords: Application and algorithm, fabric simulation, game industry.

1. Introduction

The rapid development of Computer Graphics has enabled the once-distant human dream of building virtual worlds into a tangible reality. CG technologies with the support from advanced hardware allow digital arts to be widely used in multiple fields. Along with the improving algorithms, the digital rendering quality also constantly reaches a new level. As a projection of the real world, the virtual world is typically expected to follow the mechanisms in the real world, like physical principles. Though computer graphics would also conduct studies on how to implement different visual art styles, one of the most important goals for the research field is to present things as natural, as realistic, and as accurate as possible. That pursuit involves the rendering and simulation of the most common objects and phenomena in people's daily lives, such as fluid simulation. From a layer of the model with texture attached to the current choppy water waves, the Liquid simulation and rendering experienced huge progression. Another famous research topic in CG research is the simulation and rendering of fabric material. Similarly, from the initial stages of fabric textures to the current state where simulations capture the nuanced details of fabric behavior, this area also has witnessed substantial advancements.

Fabric stimulation is not just a concern point for the computer graphic research field; it is also an important part of the digital art industry in bringing audiences a realistic virtual world. Film and animation production has always been the main application fields of cloth simulation technology as the fabrics, like character costumes and fabric texture scene objects need to move based on character movement or outside forces in the scene. With the rapid development of the game industry, fabric simulation technology in video games is a crucial attraction to players by bringing more details and reality into the virtual game world. The technique would also be important in the engineering and design field by providing new ways for the producer to design and test the quality of their fabric product. The future trends in Augmented Reality and Virtual Reality would also benefit from this technique as users could experience more realistic interaction between the avatars and the scenes.

Based on the needs of different fields, the fabric simulation work could also be divided into two different categories: offline simulation and real-time simulation. Offline simulation, typically used in the film and animation industry, has an unlimited amount of time to just generate a frame of an image even for many trials. Thus, the algorithm behind it requires high believability and controllability [1]. In contrast, real-time simulations are used in situations where rendering is constantly required, like a video game. That of the most advanced video games in current days aim to reach 60, 144, and even 360 frames per second under high resolution, with each frame requires to be generated within 17, 7, and 3 milliseconds. Even though there are plenty of techniques, like frame buffer to optimize performance, the rendering cost is still a large burden on the hardware. A tiny part of the simulating algorithm could also bring an enormous amount of costs. Therefore, the real-time simulation algorithm that allows the instant interaction between user input and the virtual environment has the design principle to be fast and stable [1,2]. Following the concerns about the performance and cost of fabric simulation algorithms would also raise studies on the utilization of CPU and GPU. In either case, the rendering consists of several main problems: Fabric Modeling, Dynamic Simulation, and Collision Handling which the paper will review in detail later [3].

This paper endeavors to comprehensively review the intricate landscape of fabric simulation by delving into its implementation methodologies and exploring diverse application fields. In addition, an insightful discussion on the anticipated future development trends within the field will be presented, shedding light on the evolving trajectory and potential innovations that lie in the future.

2. Application in the Game Industry

The primary principle of fabric simulation in the game industry is to make rendering images more realistic and immersive. By precisely simulating the movement of the fabric and its interaction with the environment, developers can bring a more engaging gaming experience to players. One of the most common applications of fabric simulation is character clothing. Instead of static textures or rigid animations, the costume props could move based on the character's movement and external forces such as wind or gravity. This adds a level of vitality and authenticity to character animation. Fabric simulation is also used for environmental elements such as flags and curtains. These objects can truly respond to in-game physics, wind effects, and player interactions, enhancing the overall atmosphere and immersion of the game world. the player can interact with the fabric in the game world, like opening a curtain or tearing a piece of fabric. One famous example involving fabric interaction is the game Journey where players would frequently have physical contact with the floating ribbons that would react with the players' movement. In addition, Fabric simulation is also used in animation and cutscenes to create visually stunning sequences with cinematic styles (Figure 1).

Unlike animation or film production, the game developer needs to pay more attention to the performance of the game in the real-time running process. The fluent running of the program is more important than the visual rendering effects. Thus, game developers often use some tricks while implementing some functions so that they hold a better effect with a smaller acceptable time and memory complexity. Following this principle, many fabric movements are not simulated in real-time but are pre-made animation assets that play following certain instructions; instead, the developers would choose more cost-effective algorithms.

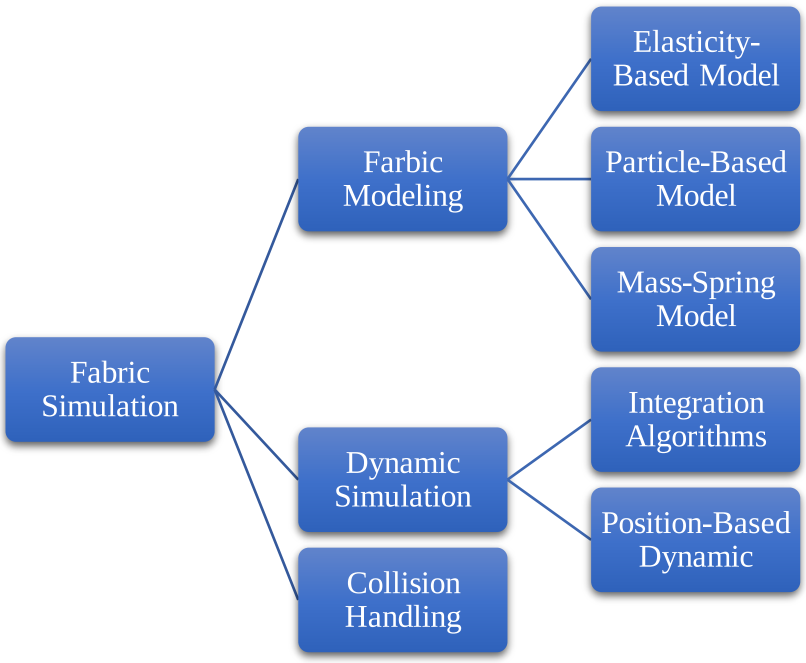


Figure 1. The basic structure for fabric simulation(Picture credit :Original)

2.1. Algorithm: Fabric Modeling

The fabric simulation algorithm consists of Fabric Modeling, Dynamic Simulation, and Collision Handling. The main fabric modeling strategies are the Elasticity-Based Model, Particle-Based Model, and Mass-Spring Damper Model.

2.1.1. Elasticity-Based and Particle-Based Methods. In considering the cloth as a continuous material, the Elasticity-Based Model will use the energy function of Elastic Theory to get the elastic forces in the cloth. In considering the cloth as a mechanism of particles representing thread crossing, the Particle-Based Model will consider the interaction of particles to simulate the draping and buckling behavior [4]. These two approaches bring realistic simulating results, but the algorithms are complex and time-consuming.

2.1.2. Mass-Spring Damper Model. The Mass-Spring Model is the most commonly used one as it provides an easily accessible simulation solution though not accurate enough. It consists of a grid of particles with mass connected by springs of different types (structural springs, shear springs, and bend springs) as shown in Figure 2. Each of the springs has its normal length, and springs would zoom in and out to move back to the normal length based on Hooke's Law of $F = -kx$. The spring system also needs to consider resistance force and gravity forces that exist in the real world.

The popularity of Mass-Spring Model is due to the better simulation effect and performance. The Elasticity-Based Model has the problem of elongation making them looks more stretchy, however, the problem does not apply to the Mass-Spring Model because of the enforcement of distance constraints. In comparison to the other two models, Mass-Spring Model also has a much better simulation performance [3]. That performance is further improved by taking the springs in one of the two diagonal directions in shear springs away. Wang, et al. prove that having one or two directions in the shear springs has little effect on the simulation performance [5].

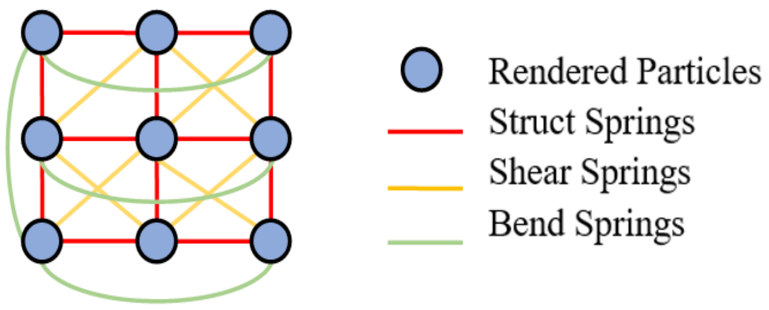


Figure 2. The illustration on the Stretch, Shear, and Bend Springs(Picture credit :Original)

2.2. Dynamic Simulation

2.2.1. Integration Algorithms. One main strategy in dynamic simulation is to perform integration on the Ordinary Differential Equation (OED) of Newton's Second Law, F = MA, to get the result of particle velocity and position. Many explicit and implicit integration algorithms could be used to solve this problem with their advantages and disadvantages. Overall, the principle in the integration strategy is to get accurate and realistic simulation results based on physics, which leads to problems like time performance. It is common to simulate a short video clip taking hours. That is not an acceptable cost for the game developers, so they are willing to give up some of the realistic simulation quality in exchange for the performance.

2.2.2. Position-Based Dynamics. To deal with the performance issue, Position-Based Dynamics (PBD) is widely used, especially in real-time simulation areas (Table 1) [6]. Unlike the various integration algorithms trying to get the most physically realistic simulation, PBD is based on the previous positions, instead of integration of the Forces in Newton's Second Law.

Table 1. PBD simulation loop

Algorithm 1 PBD simulation loop

```
1:  for all vertices i
2:      initialize $x_i = x_i^0, v_i = v_i^0, w_i = \frac{1}{m_i}$
3:  endfor
4:  loop
5:      forall vertices i do $v_i \Leftarrow v_i + \Delta t w_i f_{ext}(x_i)$
6:      dampVelocities($v_1, \ldots, v_N$)
7:      forall vertices i do $p_i \Leftarrow x_i + \Delta t v_i$
8:      forall vertices i do generateCollisionConstraints($x_i \to P_i$)
9:      loop solverIterations times
10:         projectConstraints($C_1, \ldots, C_{M+M_{coll}}, p_1, \ldots, p_N$)
11:     endloop
12:     forall vertices i
13:         $v_i \Leftarrow (p_i - x_i) / \Delta t$
14:         $x_i \Leftarrow p_i$
15:     endfor
16:     velocityUpdate($v_1, \ldots, v_N$)
17: endloop
```

The simple and concise PBD algorithm consists of only three steps. In the initialization process, A new particle position is calculated each time based on the previous position and velocity, and no physical simulation is performed. To control the dynamic of the cloth, constraints are then added to adjust the new position. At last, the new velocity and new position are calculated and updated to replace the old ones. After examining the algorithm, it is clear that the constraints, such as distance constraint, collision constraint, and volume constraint, are important in determining the simulation result. More constraints are introduced later for specific situations, including stretching, bending, long range attachments, strands, etc. [7]. Aside from the easy implementation, the PBD is also cost-effective because it can run parallelable on GPU thus reducing the burden on the CPU. Besides, it holds a faster simulation speed in low-resolution simulation tasks, making it a good choice for smaller games as the game models typically utilize low-poly models along with texture maps from the high-poly models to get better visual quality.

Table 2. XPBD simulation loop [8]

```
Algorithm 2 XPBD simulation loop
1:  predict position $\tilde{x} \Leftarrow x^n + \Delta t v^n + \Delta t^2 M^{-1} f_{ext}(x^n)$
2:  Initialization
3:  initialize solve $x_0 \Leftarrow \tilde{x}$
4:  initialize multipliers $\lambda_0 \Leftarrow 0$
5:  while i < solverIterations do
6:      for all constraints do
7:          $\Delta\lambda_j = \frac{-c_j(x_i) - \tilde{\alpha}_j \lambda_{ij}}{\nabla C_j M^{-1} \nabla C_j^T + \tilde{\alpha}_j}$
8:          $\Delta x = M^{-1} \nabla C(x_i)^T \Delta\lambda$
9:          update $\lambda_{i+1} \Leftarrow \lambda_i + \Delta\lambda$
10:         update $x_{i+1} \Leftarrow x_i + \Delta x$
11:     end for
12:     $i \Leftarrow i + 1$
13: end while
14:
15: update position $x^{n+1} \Leftarrow x_i$
16: update position $v^{n+1} \Leftarrow \frac{1}{\Delta t}(x^{n+1} - x^n)$
```

There are also drawbacks to the PBD like the physically unrealistic simulation because only the positions and constraints are considered in the simulation process [9,10,11]. The final stiffness effect is determined not solely by the stiffness parameter but more by the number of constraint iterations and the object mesh resolution. The more constraint iterations and the lower mesh it has, the more stiffness it holds [12].

To mitigate the issue, Extended Position Based Dynamics (XPBD) was invented with an additional scalar per-constraint to allow the simulation "on arbitrary elastic and dissipative energy potentials" [5]. This new version of XPBD also suits the high-resolution simulation tasks that PBD would have low performance (Table 2).

In general, PBD is suitable for real-time cloth simulation of its performance, controllability, and easy implementation. It also has some drawbacks: the stiffness is not fully dependent on the user-defined stiffness parameters, instead, it gets influenced by the time step size and solver iteration number, making parameters hard to adjust independently. The introduction of XPBD helps to solve those problems to a certain degree, but the problem is not completely solved [7]. After all, the fast performance of PBD makes up for the drawbacks it holds.

PBD gains a wide range of acceptance across the community and industry in real-time simulation and rendering, such as PhysX, Maya nCloth [2]. The PBD simulation is applied in the Unreal Engine's Chaos Cloth system, and the 5.3 version further incorporates the XPBD algorithm.

As for Collision Detection, the different algorithms would suit different situations, but the main principle is to reduce the minimized amount of work by quickly cutting off the objects that are impossible to collide and take calculations on the other. Based on when the collision is checked, the detection algorithms could be classified into two categories discrete and continuous collision.

2.3. Collision Handling

Collision handling is a crucial part of the cloth simulation for the objects to react to each other in the virtual world. Collision handling involves the following parts: collision detection, collision determination, and collision response (Figure 3). Collision detection could be divided into different phases: the broad phase which checks the potential collisions and the narrow phase which checks the overlapping convex between objects, and it is sometimes divided into three phases with a mid-phase that takes parts of the function of the narrow phase.

The Discrete Collisions Detection would look at the objects that collide at that instant. Due to the checking mechanism of discrete detection, this method is only applicable when the objects still have contact with the collided objects within that frame. That leads to problems named Tunneling when the colliding object has a high velocity that could pass through the collided object within that frame, in which the Discrete Collisions Detection would be inapplicable [9]. The solutions would be to lower the object speed or increase the detecting frequency. Either way is not a good strategy in situations with extremely high velocity, which would take a huge cost to solve by using the increased detecting frequencies.

To have accurate detection in these extreme cases, continuous collision detection (CCD) is been applied. CCD would check the collision along the trajectory of objects and advance until the response for the collision is resolved. The mechanism of CCD implies that it has a higher cost and complexity than the Discrete Collisions Detection.

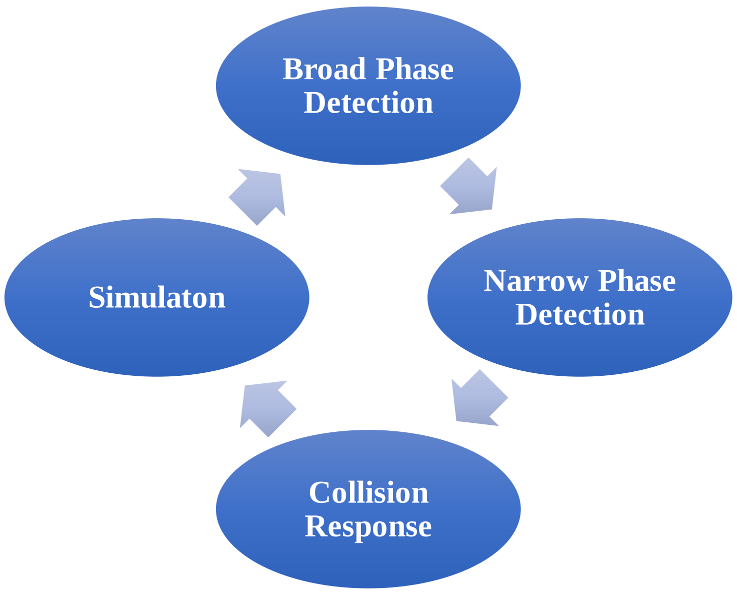


Figure 3. The basic structure for Collision Handling (Picture credit: Original)

The category of discrete and continuous is also referred to as posterior and priori because the intersecting collision indeed happens in Discrete Collisions Detection which requires further collision response to adjust the object position. In contrast, the colliding objects in CCD stop at the point of contact instead of intersecting with others.

Once the collision is detected and determined, responses need to be made. For the Collision Response, one main task is to separate the intersecting objects from each other. There are several ways to reach that effect: changing the velocity and acceleration of the objects, applying repulsive force, or directly updating the position and velocity of the objects.

The collision scenarios could be generally classified into two cases: cloth-cloth and object-cloth collisions [1]. In situations where layers of cloth collide with each other, also known as self-collision, a temporary repulsive spring to particles could be added to the mass-spring model to prevent them from intersecting each other. Bridson et al. publish another solution that applies corresponding repulsive forces based on the distance between cloth particles [13].

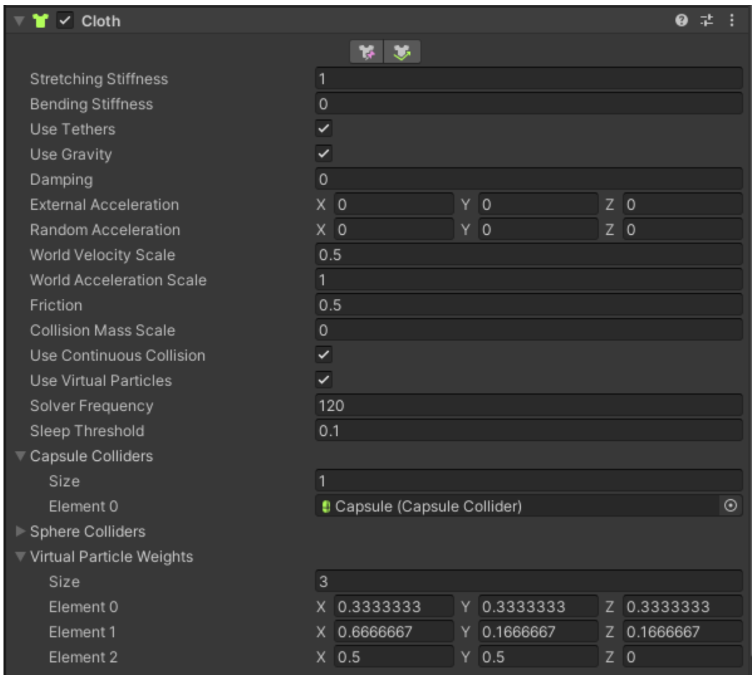
Similar techniques of changing particle position and velocity are also useful in situations where the cloth collides with other objects. However, the problem of un-smooth simulation results is annoying for the audience. Bridson et al proposed another solution to construct a grid for the scene where each cell is assigned a number [14]. By finding and comparing the number of cells corresponding to collision, it can create a popping effect on the cloth to show wrinkles.

The collision response is another important step aside from the simulation and collision detection stage, meaning that the challenge of having better performance still exists here. The collision response process may bring further collisions or other complicated scenarios. Thus, the application of collision response also could not leave the optimization for better performance. To analyze the performance of algorithms and hardware, a study has been conducted by implementing Unity fabric collision solutions with different factors: the result shows that solution two of moving mass-point position on the object surface at the collision detected time step would result in more natural and feasible simulation effect than the solution one that only reflecting the velocity vector during the collision response. In several tests, the cloth model is penetrated by the object model under solution one, making solution two to be the favored choice [15]. Both solutions are both over 350 frames per second with a difference of roughly 10 frames of solution one over solution two. The difference is acceptable as the solution has better simulation results. Besides, the performance test also shows that the result of implementing shaders in Unity with GPU that provides parallel computing ability is 60% faster than CPU-only calculation, providing more insight in how to optimize cloth simulation and rendering.

3. Discussion

The two popular game engines, Unity 3D and Unreal Engine 5 all have their cloth simulation system (Figure 4). Unity has a Cloth simulation component with various attributes to adjust the simulation effect. There are also other cloth simulation add-ons in the Unity Asset Store, signaled by the Magica Cloth 2 with comprehensive functions. Unreal Engine 5 also has its own Chaos Cloth system.

To implement cloth simulation in Unity 3D, the developer simply needs to add the Cloth component to the cloth model asset and adjust the attributes in the panel. The two buttons with cloth icons provide a painter to directly draw on the cloth surface to modify the simulation setting on each mesh point around max (displacement) distance, surface stiffness, and self-collision. Among all of them, Gravity and Damping are two important parameters that apply gravity force on the cloth and determine the damping speed back to a stable state. The combination of all the parameters would reach various effects for different cloth textures.



Cloth

Stretching Stiffness	1
Bending Stiffness	0
Use Tethers	✓
Use Gravity	✓
Damping	0
External Acceleration	X 0 Y 0 Z 0
Random Acceleration	X 0 Y 0 Z 0
World Velocity Scale	0.5
World Acceleration Scale	1
Friction	0.5
Collision Mass Scale	0
Use Continuous Collision	✓
Use Virtual Particles	✓
Solver Frequency	120
Sleep Threshold	0.1
Capsule Colliders	
Size	1
Element 0	Capsule (Capsule Collider)
Sphere Colliders	
Virtual Particle Weights	
Size	3
Element 0	X 0.3333333 Y 0.3333333 Z 0.3333333
Element 1	X 0.6666667 Y 0.1666667 Z 0.1666667
Element 2	X 0.5 Y 0.5 Z 0

Figure 4. Unity 3D Cloth component (Picture credit: Original)

Note that the Unity Cloth component only accepts capsule and sphere colliders, which is inconvenient for many kinds of models. Kim, Minsang, et al. created a cloth plugin that follows the same strategy of using sphere colliders which could be generated according to model shape. Along with the technique of parallel processing of the mass-spring model using shader on GPU, the simulation result is more accurate and faster than the Unity cloth component [16].

Aside from the pre-existing tools, it is also possible to build the cloth simulation system within the engines through coding. Hongly, et al. utilize a mass-spring system, constraint enforcement, and adaptive constraint activation and deactivation techniques to construct their cloth simulation system in Unity, with a better time performance than the default Unity cloth system [10].

Future work in this field will focus on optimization as the hardware evolving speed is slower than the previous decades. One straightforward methodology is to improve the algorithms while balancing between performance and cost. Aside from that, studies have been done on view-dependent adaptive simulation that uses reduced resolution for invisible model areas to reduce the calculation cost, and the result shows significantly smaller values in model faces, vertices, and time cost [10,11].

Besides, the frequently mentioned GPU parallel computing techniques are useful. A study presents a cloth simulation that utilized multiple GPUs parallelly to get faster computation and handle high-resolution mesh, and it gains simulation performance linear speedups based on the number of GPUs in work. Another study using GPU-based cloth simulation, which further presents the parallel algorithms for time integration and collision handling on GPU, shows high-quality and high-performance simulation [17]. However, these simulation solutions take seconds for each frame. Further study on how to implement real-time parallel GPU-based simulation solutions needs to be taken [18].

Other optimizations focus on the simulation solution and algorithms. The introduction of block coordinate descent to solve the time integration of mass-spring systems gains huge growth in simulation performance [19]. In comparison to the traditional exact solution of Newton's Method that generates an image at 13 seconds per frame, the advanced method could get similar results through 10 iterations and within 50ms (which is 20 fps). The fps could be further improved by reducing the iterations going through. Though the simulation result is less natural, running one iteration only requires 5ms (which is 200 fps), which meets the current game player expectation.

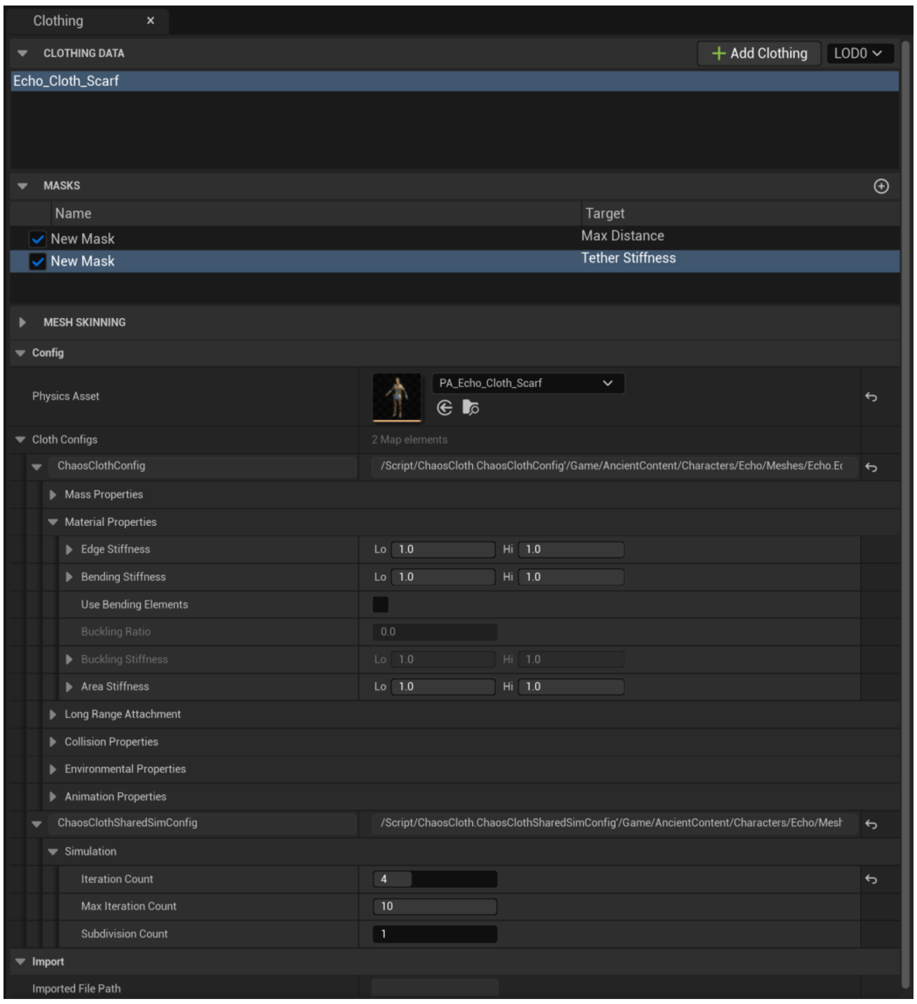

Clothing ×
CLOTHING DATA
Add Clothing
LOD0
Echo_Cloth_Scarf
MASKS
Name | Target
New Mask | Max Distance
New Mask | Tether Stiffness
MESH SKINNING
Config
Physics Asset: PA_Echo_Cloth_Scarf
Cloth Configs: 2 Map elements
ChaosClothConfig: /Script/ChaosCloth.ChaosClothConfig'/Game/AncientContent/Characters/Echo/Meshes/Echo.Ec
Mass Properties
Material Properties
Edge Stiffness: Lo 1.0 Hi 1.0
Bending Stiffness: Lo 1.0 Hi 1.0
Use Bending Elements
Buckling Ratio: 0.0
Buckling Stiffness: Lo 1.0 Hi 1.0
Area Stiffness: Lo 1.0 Hi 1.0
Long Range Attachment
Collision Properties
Environmental Properties
Animation Properties
ChaosClothSharedSimConfig: /Script/ChaosCloth.ChaosClothSharedSimConfig'/Game/AncientContent/Characters/Echo/Mesh
Simulation
Iteration Count: 4
Max Iteration Count: 10
Subdivision Count: 1
Import
Imported File Path

Figure 5. Unreal 5 Clothing (Picture credit: Original)

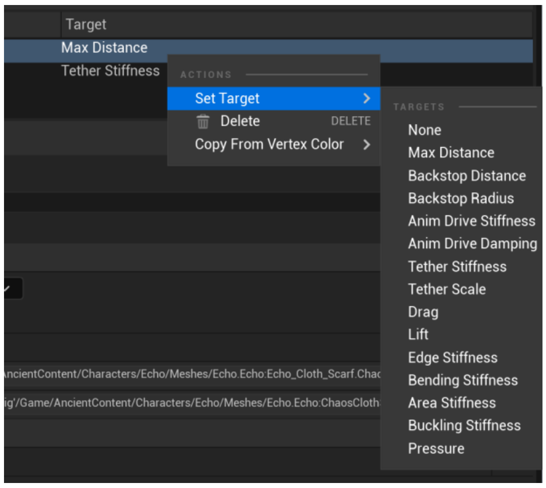

Target
Max Distance
Tether Stiffness
ACTIONS
Set Target
Delete DELETE
Copy From Vertex Color
TARGETS
None
Max Distance
Backstop Distance
Backstop Radius
Anim Drive Stiffness
Anim Drive Damping
Tether Stiffness
Tether Scale
Drag
Lift
Edge Stiffness
Bending Stiffness
Area Stiffness
Buckling Stiffness
Pressure
ncientContent/Characters/Echo/Meshes/Echo.Echo:Echo_Cloth_Scarf.Chac
ig'/Game/AncientContent/Characters/Echo/Meshes/Echo.Echo:ChaosCloth

Figure 6. Unreal 5 Clothing (Picture credit: Original)

Unreal also has a default cloth simulation system called Chaos Cloth, similar to the Cloth Component in Unity, as shown in Figure 5. It provides all the attributes and functions Unity Cloth holds, like settings for painter tools, and detailed parameters for simulation. In comparison to the painter tool in Unity, Unreal 5 contains more modes to control the cloth simulation more accurately at different Mask of cloth surface, like Pressure, Tether Scale, etc. That is done by creating a new Mask at the Masks panel and set its target into different modes as shown in Figure 6. The Unreal Clothing contains further specific parameters like edge stiffness and area stiffness, some of which are made of two values, Lo and Hi. If the cloth Weight Map with values in the range of 0 to 1 supports the parameter, Low and High values will be used in conjunction with each particle's weight in the weight map to interpolate the final value, Otherwise, Low value is applied. Multiple clothing data could be created and changed among them.

As for video games, interacting with game objects with fabric textures and characteristics could increase the immersion of players. It is useful in many scenarios like the cloth effect in swimming and the blood sputtering effect or control over certain fabrics. Many simulation studies on such interactions with fabric material have been taken and gained huge progression. Fei, Yun, et al. propose a physics-based method, considering factors like velocity fields and scalar saturation variables, to reach realistic simulation animation scenarios involving splashing, wringing, and collision between liquid and fabric [20]. However, the method is time-consuming taking 1 to 18 seconds per simulation step. Future studies should also focus on how to achieve such interaction between game scenes and fabric. Following along the direction physical-based method may not be plausible to get such a method for real-time level simulation, so more considerations on other strategies need to be taken. The optimization strategies are waiting for more exploration in the future.

4. Conclusion

In conclusion, the evolution of fabric simulation within computer graphics has been marked by significant advancements in both technology and application domains. From its humble beginnings of representing fabric textures to the current state of simulating intricate fabric behaviors, this field has become indispensable in various industries, including film production, video games, and design. In the game industry, fabric simulation plays a pivotal role in enhancing the realism and immersion of gaming experiences. It brings character costumes, environmental elements like flags and curtains, and interactive fabric objects to life, allowing players to engage more deeply with virtual worlds. Despite the performance challenges inherent in real-time simulations, developers have adopted clever strategies to balance visual fidelity with computational efficiency, such as using pre-made animation assets and cost-effective algorithms. The fabric simulation algorithm encompasses Fabric Modeling, Dynamic Simulation, and Collision Handling. Techniques like Mass-Spring Models and Position-Based Dynamics offer viable solutions, albeit each with its own set of methodologies and trade-offs. The ongoing development of Extended Position Based Dynamics (XPBD) showcases the industry's commitment to refining simulation techniques for high-resolution tasks and more realistic visual effects.

Overall, fabric simulation technology continues to evolve, driven by advancements in algorithms, hardware capabilities, and application demands. The future development of fabric simulation is anticipated to focus on improving realism, optimizing performance, and expanding its applications. As we look towards the future, advancements in fabric simulation will continue to drive innovation across industries, particularly in emerging technologies such as augmented reality, virtual reality, and real-time rendering applications. The ongoing quest for more efficient algorithms, coupled with advancements in hardware capabilities, promises a future where fabric simulations can seamlessly integrate with ever-expanding digital worlds, offering users unparalleled levels of immersion and realism.

References

[1] Stuyck, Tuur. Cloth Simulation for Computer Graphics. Switzerland, Springer International Publishing, 2022.
[2] Wallner, Guenter. "Simulating, animating and rendering clothes." ACM Transactions on Graphics 26.2 : 1-23, 2007
[3] Breen, David E., et al. "A physically-based particle model of woven cloth." Computer Graphics , 26.2 : 365-372,1992.

[4] Müller, Matthias, et al. "Position Based Dynamics." Proceedings of the 2006 ACM Eurographics Symposium on Computer Animation. Eurographics Association, 2006.
[5] Wang, Meiliang, et al. "Study on Improving Three-Dimensional Dynamic Simulation Model and Equation of Flexible Fabric." Advances in Mechanical Engineering (Print), vol. 11, no. 6, SAGE Publishing, June 2019, p. 168781401985284..
[6] Tomas Akenine-Moller, et al. Real-Time Rendering, Fourth Edition. CRC Press, 2018.
[7] Bender, Jan, et al. Position-Based Simulation Methods in Computer Graphics. Jan. 2015, https://doi.org/10.2312/egt.20151045. Accessed 2 Apr. 2024.
[8] Macklin, Miles, et al. "XPBD: Position-Based Simulation of Compliant Constrained Dynamics." ACM Transactions on Graphics, https://doi.org/10.1145/2994258.2994272.
[9] Zhang D. 2001. Efficient algorithms for cloth simulation. Ph.D. Dissertation. Hong Kong University of Science and Technology (People's Republic of China). Advisor(s) Matthew M. Yuen. Order Number: AAI3018351.
[10] Va, Hongly, et al. "Real-Time Cloth Simulation Using Compute Shader in Unity3D for AR/vr Contents." Applied Sciences, vol. 11, no. 17, p. 8255, Sept. 2021.
[11] Koh, W., et al. View-Dependent Adaptive Cloth Simulation. , pp. 159–66,July 2015.
[12] Bender, Jan, et al. "Position-Based Simulation of Continuous Materials." Computers & Graphics, vol. 44, pp. 1–10, Nov. 2014.
[13] Bridson, R., et al. "Simulation of Clothing with Folds and Wrinkles." ACM SIGGRAPH 2005.
[14] Bridson, Robert, et al. "Robust Treatment of Collisions, Contact and Friction for Cloth Animation." ACM Transactions on Graphics, vol. 21, no. 3, pp. 594–603, July 2002.
[15] Min Sang Kim, et al. Real-Time Collision Response between Cloth and Sphere Object in Unity. no. 6, pp. 53–62, Jan. 2018.
[16] Kim, Minsang, et al. "Parallel Cloth Simulation with Effective Collision Detection for Interactive AR Application." Multimedia Tools and Applications, vol. 78, no. 4, Springer Science+Business Media, pp. 4851–68, May 2018.
[17] Li, Cheng, et al. "P-Cloth: Interactive Complex Cloth Simulation on Multi-GPU Systems Using Dynamic Matrix Assembly and Pipelined Implicit Integrators." ACM Transactions on Graphics, vol. 39, no. 6, Association for Computing Machinery, pp. 1–15, Nov. 2020.
[18] Liu, Tiantian, et al. "Fast Simulation of Mass-Spring Systems." ACM Transactions on Graphics, vol. 32, no. 6, pp. 1–7, Nov. 2013.
[19] Tang, Min, et al. "A GPU-Based Streaming Algorithm for High-Resolution Cloth Simulation." Computer Graphics Forum, vol. 32, no. 7, pp. 21–30, Oct. 2013.
[20] Fei, Yun (Raymond), et al. "A Multi-Scale Model for Simulating Liquid-Fabric Interactions." ACM Transactions on Graphics, vol. 37, no. 4, pp. 1–16, Aug. 2018.