International Core Journal of Engineering Volume 7 Issue 4, 2021
ISSN: 2414-1895 DOI: 10.6919/ICJE.202104_7(4).0007

A Procedural Approach for Cloth Simulation

Wei Guo[1], Yihao Wang[2], Kai Zhang[3]

[1]Department of Information Technology, James Cook University Singapore, 387380 Singapore;

[2]Dalian No.36 High School, Dalian, Liaoning 116010, China;

[3]School of Information and Communication Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China.

Abstract

Position Based Dynamics has been widely adopted as an effective method in real time simulation due to its merits of stability and low computational cost. However, its accuracy has always been questioned, especially in cloth simulation. In order to ameliorate the situation, the authors propose an idea to procedurally generate mesh topology similar to knitted clothes in the real-world. Several algorithms are compared and implemented to achieve the desired generation results. Experiments are conducted to demonstrate the effectiveness of this method and authenticate the improvement on accuracy in cloth simulation. Potential applications of this approach are later discussed.

Keywords

Computer Graphics; Cloth simulation; Position Based Dynamics; Wave Function Collapse; Procedural Vertex Generation.

1. Introduction

1.1 Background

CG, Computer Graphics, contributes significantly to people's daily entertainments and professional training, such as animations, video games, surgery simulations, and three-dimensional modeling. One of its most important research topics is cloth animation since cloth can be found everywhere in our daily life. Tasks of cloth animation include both appearance rendering and dynamic interaction. To generate more realistic effects, it is more than significant in constructing one model which could well represent the properties of clothes.

1.2 Existent Cloth Simulation System

Among all of the related works, Finite Element Method is able to get very realistic effects. However, as a continuum model, it requires too much calculation in solving the kinetic equations as well as collision detection. To simplify the whole process, models with discreet particles emerge. Provot [1] came up with the mass-spring system, in which he used particles and quadrilateral meshes to represent clothes' structure. Other scholars also have proposed some improving models to solve problems like spring-overstretching in the mass-spring system [2-4]. And Müller, etc. [5], put forward the Position-Based Dynamics, which uses several positional constraints to describe the inner forces, performing both accurately and efficiently. In these discrete particle models, most mesh in cloth simulation have evenly distributed vertices. In the real world, nevertheless, cloth often has random nodes interacting with different objects to deform and displace. Thus the even distribution could not describe cloth very well.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202104_7(4).0007

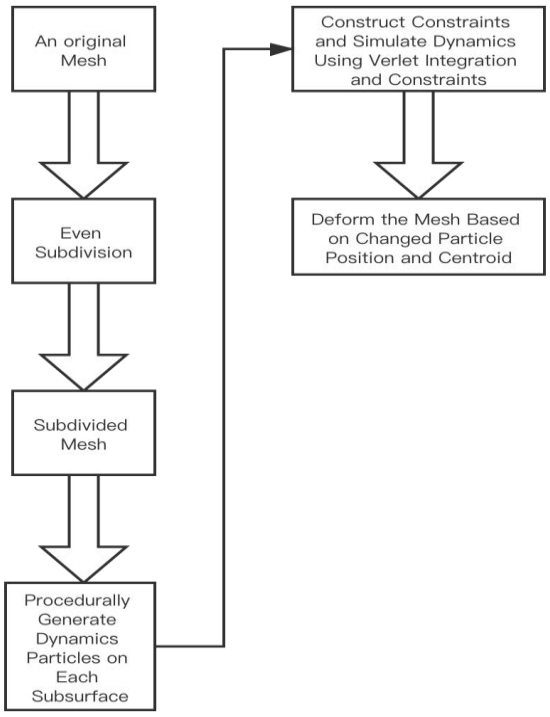

Figure 1. Flow chart

1.3 Our Simulation Approach

Considering that, the authors apply Wave Function Collapse to simulate this natural pattern of cloth, and have presented a relatively effective and efficient cloth simulation system. In order to balance computational cost and quality of generated image, Position Based Dynamics is adopted as the basic method. The authors divide the project into five parts. The first step is to evenly subdivide input mesh into many small faces, which is beneficial to the calculation of deformation. The second step is to procedurally generate dynamic particles on each subsurface to simulate this natural pattern of cloth. The third step is to construct constraints in the normal way so that this software can mimic the pattern users put in automatically, such as a silk piece knitted in a certain way. The fourth part is to simulate dynamics using Verlet Integration and Constraints[6] and finally, the fifth step is to deform the original mesh based on changed particle position. Based on test results, this method can effectively generate random points in a given mesh and the authors believe the method will improve the accuracy of cloth simulation. A flow chart is presented as figure 1.

2. Simulation

2.1 Wave Function Collapse

When a wave function, initially in a superposition of several eigenstates, reduces to a single eigenstate because of interactions with the external world, wave function collapse occurs. Observables represent classical dynamical variables, and when one is measured by a classical observer, the wave function is projected onto a random eigenstate of that observable. Models of wave function collapse have been widely adopted in generating random maps of games. The generated map can realistically mock the terrain in a real-world setting, which is mostly random.

2.2 Algorithm Selection of Procedural Vertex Generation

Most existing procedural algorithms utilize a monte carlo approach combined with pattern regulation to produce desired data. Such approaches include simple noise functions such as Perlin Noise [7], or more advanced methods as GAN generation [8] if a dataset is available. The mechanism behind them is straightforward: a set of patterns is predetermined from generation context, and data is populated by random noises regulated within these patterns.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202104_7(4).0007

For the purpose of generating PBD vertices in this scenario, the procedural algorithm devised should be compact and efficient enough to be operated in a real time application while maintaining a realistic simulation. Common noise functions alone have few regulations and couldn't meet these standards. Therefore, wave function collapse, a compact, light-weighted algorithm, comes into the picture. Wave function collapse is a procedural generation algorithm which produces bitmaps by arranging a collection of tiles according to rules about which tiles may be adjacent to each other, and relatively how frequently each tile should appear [9]. The algorithm simulates phenomena in quantum physics, in which several wave functions exist as a probability distribution of particles and until a certain state in the distribution is confirmed, the functions collapse into determined data points.

In procedural generation, a WFC algorithm repeatedly collapses tiles to limit probability distribution of adjacent tiles, thus determining and populating the bitmap as it walks through each tile. When an adjacent tile couldn't be collapsed because either it breaks the frequency or adjacency rule, the whole generation starts over to find an alternative placing pattern. These steps can be represented as the pseudo code below:

```
class tile{

tile_type type;
List<tile> adjacent_tiles;
tile root_tile;

bool is_collapsable(){...};

void collapse(){
    for(tile_type t in tile_type.types){
    this.type = t;
    bool in_noise = generate_noise() < noise_range/2;
    if(this.is_collapsable() && in_noise) break;
        }
}

void generateWFC(pattern, seed){
    if(this.is_collapsable()) this.collapse();
    else root_tile.generateWFC();

    for(tile in adjacent_tiles){
    tile.generateWFC();
        }
}

}

tile.generateWFC(pattern, seed);
```

ISSN: 2414-1895 DOI: 10.6919/ICJE.202104_7(4).0007

Note here that the generate_ noise() function refers to any noise function that could simulate a probability distribution to pick probable tile type to collapse. The is_ collapsable function determines if the collapsed tile satisfies frequency and adjacency requirements. The recursive structure of the pseudo code is for better illustration purposes. The nature of the WFC algorithm is iterative since it walks through each tile only once, and in implementation should respect iteration design since it yields more efficiency.

Once a bitmap is generated according to the corrugation pattern given for a cloth type, it could be mapped on the subdivided mesh's vertices. This mapping could be a simple displacement of each subdivided vertex on each face. The mapped mesh could then be used by PBD simulation for realistic cloth deformation purposes.

2.3 Implementation of Procedural Vertex Generation

Implementation of WFC and its utilities follow a modular software design pattern in the simulator. The module takes in a list of subdivided meshes with original topology information before subdivision and respective corrugation samples. It then performs procedural generation of each corrugation sample for each face and maps the subdivided vertices. After these steps the module produces the corrugated meshes and sends them to PBD deformer and animator for further processing.

Several points are worth noting here for implementation details. First, the original face's edge vertices are kept in position from mapping to ensure the original mesh's topology and shape is respected. This also solves the "stitching problem" of procedural generating separately on each face also, since the edge vertices are not moved and contradicted with each other. Also, some of the vertices are filtered in order to reduce face number, which benefits collision detection and deformation algorithms later.

2.4 Results of Procedural Vertex Generation

Here some of the sample results from the WFC generator are showcased. The algorithm treats a png image as a binary bitmap with black signaling fabrics and white signaling space. In this implementation other color channels are converted and grayscale is not considered. However, it would be an interesting experiment to upgrade the implementation in the future and observe how more colors could be used to improve vertex layouts. For example, if five colors are permitted, each one could represent a different fabric material and pattern. This could be useful for a more realistic PBR rendering and simulation.

The first sample generated is from a pattern simulating rough fabric found on muslin [10], bandage or other similar products. It has a simple cross knitting and much space between fibers. The pattern generated shows this corrugation with space and linearity. (Figure 2 and Figure 3).

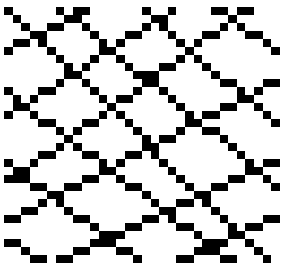

Figure 2. Sample of Loose Weave Generation

Figure 3. Result of Loose Weave Generation

ISSN: 2414-1895 DOI: 10.6919/ICJE.202104_7(4).0007

The second attempt aims at simulating a more rigid grid pattern. This pattern of weaving is common in machine produced fabrics. The result simulates the grid-like organization, only more naturally and with variations of space between fibers. (Figure 4 and Figure 5).

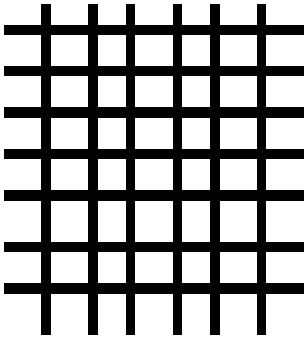

Figure 4. Sample of Grid Weave Generation

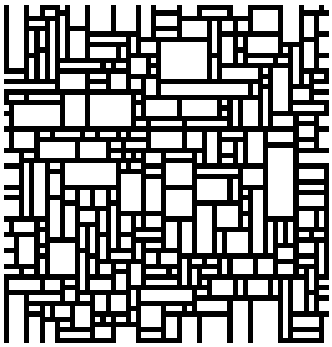

Figure 5. Result of Grid Weave Generation

A variation of cross weave is called diamond weave. It has the same basic pattern of crosses from the name, but doubles the structure and is much denser. This yields a stronger fabric than only one layer of knitting. The simulated result shows the corresponding increase in density and patterns of fibers. (Figure 6 and Figure 7)

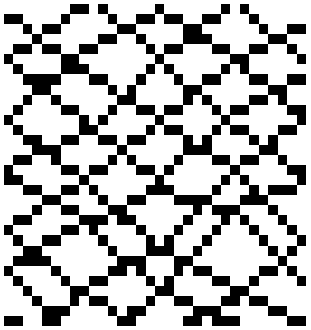

Figure 6. Sample of Diamond Weave Generation

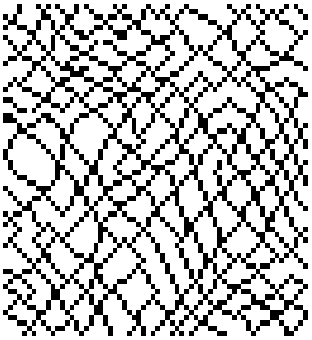

Figure 7. Result of Diamond Weave Generation

ISSN: 2414-1895 DOI: 10.6919/ICJE.202104_7(4).0007

Theoretically the WFC core implemented could generate any pattern given a sample layout. However, it is recommended by the authors that the sample should be carefully selected to represent the actual corrugation of realistic material. A poor sketch of these fabrics can yield unsatisfactory results. The readers are encouraged to try WFC generating themselves by accessing the author's implementation on github[11].

3. Deformation

3.1 Subdivision

Subdivider introduces a method of subdividing a square into any given times. The purpose of this part is to subdivide one mesh into multiple parts and calculate the interacting forces of these parts with rigid body or cloth so that more realistic simulations can be achieved.

In this phase, the authors use numerical values stored in vertices of each mesh to calculate the position, centroid, and norms of subdivided meshes, and pass new values to the actuator. The process of subdivision is relatively simple: in a given mesh, every square is calculated independently. Using coordinates of each vertex, the program can calculate the midpoint of each line and the centroid of the original square. Then, with the coordinate of the centroid and each vertex, the program can calculate the centroid of every new square respectively. In addition, based on linear interpolation, the program can calculate norms of every point. Finally, these elements are stored in Element* e so the actuator can use these elements to simulate deformations and motions of particles. Besides, in order to subdivide meshes into any given times, a variable is added. Every loop, subdivided mesh is passed to the variable and in the next loop the program uses the elements in the variable to do further calculation.

```
class Subdivider{

class Mesh, face;

void subdivide(Mesh& out, Mesh& in,int sub_time){
        for(auto face: in.faces){
            out.vertices.push_back;
        }

void sd(Mesh& out, Mesh& in,int sub_time){
      while(time>0){
            subdivide(out,in,time);
            in=out;
            time--;
            if(time!=1){
                out.faces.erase(out.faces.begin(),out.faces.begin()+n);
                n*=4;
            };
        };
    };
}
```

ISSN: 2414-1895 DOI: 10.6919/ICJE.202104_7(4).0007

3.2 Linear Interpolation of PBD Vectors and Mesh Integration

In the actual PBD dynamics the simulated points are performed on each face's centroid rather than on its vertices. This is done to avoid boundary problems, in which the vertices are deformed directly and may cause inner subdivided faces to deform with unnatural constraints since the faces are subdivided based on face topology. By using the centroids as PBD points, the simulation respects the original mesh topology.

To transport deformation to the actual vertices, a linear interpolation follows for each centroid's PBD vector. The face is treated as a microfacet of the mesh object, and displacement is evenly sampled on it. Further consideration on this deformation step includes utilizing monte carlo method to spread the simulated deformations less evenly. This prevents the deformed mesh from appearing micro faceted, achieving a smoother outcome and avoiding common PBD problems such as divergent dynamic equations and collision misses.

4. Conclusion

Based on input data, vertices similar to real material can be successfully generated in any given mesh with this method. This development can enhance the accuracy of cloth simulation because currently, most deformation methods still treat faces of cloth as triangles or squares, both of which don't follow real-world settings. Further discussions regarding this topic include the possibility of applying GAN on vertices' procedural generation to better simulate corrugation and appropriate subdivision algorithms for realistic mesh deformation. The authors will continue to conduct experiments to test the effectiveness of this method in various scenarios and improve the efficiency and performance of this software.

References

[1] Provot, X. (1995) Deformation constraints in a mass-spring model to describe rigid cloth behavior. Graphics Interface, 23(19): 147-154.

[2] Goldenthal, R, Harmon, D., Fattal, R., Bercovier, M., Grinspun, E.. (2007). Efficient simulation of inextensible cloth. ACM Transactions on Graphics, 26: 49.

[3] Ye, J.. (2010). Simulating inextensible cloth using impulses. Computer Graphics Forum, 27(7): 1901-1907.

[4] Ye, J., Webber, R. E., & Wang, Y.. (2009). A reduced unconstrained system for the cloth dynamics solver. Visual Computer, 25(10): 959-971.

[5] Matthias Müller, Heidelberger, B., Hennix, M. , Ratcliff, J. . (2007). Position based dynamics. Journal of Visual Communication & Image Representation, 18(2): 109-118.

[6] Baltman, R., Radeztsky, R.. (2004). Using Verlet integration and constraints in a six degree of freedom rigid body physics simulation. In: Game Developers Conference. San José.

[7] Perlin, K.. (2002). Improving noise. ACM Transactions on Graphics, 21(3): 681-682.

[8] Brummelen, J. Chen, B.. (2016). Procedural Generation. http://www.mit.edu/~jessicav/6.S198/Blog_Post/Procedural Generation.html.

[9] Grid Bugs. (2019) Procedural Generation with Wave Function Collapse. https://www.gridbugs.org/wave-function-collapse/

[10] MasterClass. (2020). Fabric 101: What Is Muslin? How to Use and Care for Muslin. https://www.masterclass.com/articles/fabric-101-what-is-muslin-how-to-use-and-care-for-muslin.

[11] Guo, W. (2020). WFC-Cloth-Mesh-Corrugator. https://github.com/Wei-Parker-Guo/WFC-Cloth-Mesh-Corrugator.