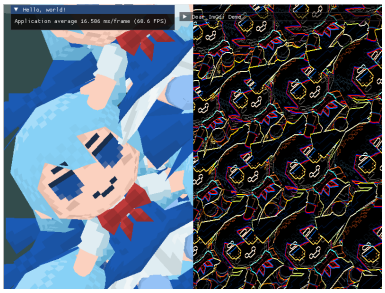
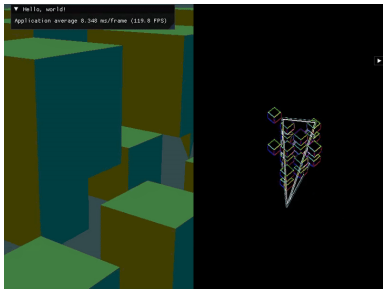
Project Proposal

Arrian Chi





***Abstract*— In this proposal, I describe my interest in developing a cloth simulation feature for my hobby project, AlienGLRenderer. I will first explain the motivation of the project, followed by the questions I aim to answer, and ending with the areas I will look into for these answers. I will also provide a timeline and extra topics this project may introduce.**

I. Introduction

Cloth simulation is a problem with extensive research in the field of computer graphics. Numerous methods and techniques have been posed to simulate cloth. One method proposes a geometric solution, simply approximating the appearance of cloth (wrinkles, fold, etc.) based on a geometric model from cable theory [1]. Another proposes a physical solution involving the division of the cloth into evenly distributed masses connected by mass-spring dampers [2]. A more well-known physically based method involves the consideration of elastic energies [3] and forces over the surface of the cloth, which is then solved using numerical methods [4]. More recently, machine learning techniques, such as deep neural networks and unsupervised learning, have been adapted as an optimization scheme [5], including the handling of cloth dynamics [6]. In this project, I aim to focus on modeling cloth with the discrete elastic plates/shells (DEP/S) method as presented in class, as well as explore how to make this simulation ”real-time”.

II. Motivation

The motivation for this project starts with a hobby project AlienGLRenderer I am working on as of late. AlienGLRenderer is a renderer, written in C++ and OpenGL, designed to support the creation of simple scenes (inspired by the Quake Engine). The overall goal of this project is to streamline the rapid creation of simple scenes for artistic expression and to showcase evidence of technical knowledge. Thus, each feature added is one step towards the overarching goal. Currently the renderer supports a basic 3D rendering pipeline, model/scene loading, model instancing, frustum culling, compute shaders, and post-processing. The addition of a ”real-time” cloth simulation feature would add many different features to the renderer and would be, by itself, a significant feature to showcase.

III. Questions

Before explaining what a cloth simulation would bring to my renderer, I would like to present the questions I am posing for this project:

- How do we simulate a cloth effectively and efficiently?
  - What are some modifications we can use to make the DEP algorithm ”real-time”?
    - Are these modifications reasonable to implement?
  - How does implicit and explicit integration differ in the context of cloth simulation?
    - Could we afford a tradeoff between stability and speed?
    - How would offline rendering impact the application in general?
- What technical problems/constraints arise when simulating cloth?
  - Where could the DEP algorithm benefit from optimization?
  - How do we render a cloth with good visual quality?

IV. Idea

The addition of a cloth simulation feature to AlienGLRenderer would involve steps that could be divided in the following categories:

Essential (High Priority) Features:

These include the implementation of features necessary for the setting up the simulation’s computation and rendering. For instance, the concept of a simulation implies the existence of time stepping and animation, so a time-stepping algorithm (and discretization scheme) would be necessary. The DEP algorithm’s computation of forces and energies implies the representation of cloth as a particle system (as also stated in [4]). The self-occlusion of cloth,

as in the case of folding, requires a perspective of depth, which could be solved with a lighting / shadow model or texture mapping. Finally, depending on the speed of the DEP implementation, the application may require a offline rendering feature to precompute the cloth's state (more on this in the next section).

Focus (MAIN PRIORITY) Feature:

The main focus of this project is to make the cloth simulation "real-time". The use of "Real-time" is in quotes because of the uncertain nature of how fast the DEP algorithm could be (when implemented in C++). For an online simulation, one iteration of the loop should take at most 33 milliseconds (30 frames per second was the common standard for games in the past). The computation time depends on a number of factors, such as the number of faces on the cloth, the size of the time step, the complexity of calculating the forces, the complexity of rendering the cloth, and more. Moreover, optimizations in the DEP algorithm are possible with GPU solvers [7], better numerical methods [8], and neural networks [6]. These are areas I would like to explore to answer whether or not the DEP algorithm can be made "real-time" (online).

However, even if it is not possible to hit the 30 FPS target, it is still possible to animate the cloth in real-time with offline-rendering (precomputing the cloth's state for set time frame). Given ample memory and time, the application would calculate all the positions for the cloth particles for every time step, store them in memory, and then render the entire animation (this is exactly how some movies ray-trace their scenes). This offline computation scheme could still benefit from optimizations; the reduction in time would cut the precompute time. That being said, although the offline rendering feature is a good fallback, I ideally want to keep my renderer online.

Extra (Low Priority) Features:

These are features that are not necessary for the simulation to work, but would be nice to have. To name one, collision detection would make our simulation more realistic, preventing the cloth from intersecting with other objects and itself[4]. Another feature could be the simulation of wind, water, and other dynamic natural phenomena affecting the external forces of the system, which would make the simulation more lively and interesting. This could also be extended to adding destructive behaviors, such as burning and tearing, which directly impact the geometry and DOFs of the cloth. Lastly, we could add human interaction into the simulation, such as grabbing and pulling the cloth, which is essentially raycasting onto the cloth and fixing DOFs/applying forces to the selected area.

It is important to note that although these features would be exciting to implement, they may contribute to not only the complexity of the project, but also the time of the computation. Take self collision detection, for example. Without the use of acceleration data structures (spatial hash maps, bounding volume hierarchies, etc.), detecting whether an edge intersects another edge is an $O(n^2)$ operation [4]. It is likely that this added feature could become a bottleneck if all of the other computation has a lower amortized time complexity. In addition, some collision detection methods could contribute to the instability of the simulation if not implemented correctly. For instance, the use of a repulsive force field around each cloth particle is reported to give subpar results[9]. Thus, care must be taken before deciding to implement any of these extra features.

V. Plan

Here, I provide an overview of what the next five weeks of the project's development will look like:

Week 1: Proposal

- Implement a simple particle system, lighting, and the DEP algorithm in C++.
- Spend some time looking into various ways to optimize the algorithm.

Week 2:

- Determine whether the DEP algorithm can be made "real-time" (choose between offline and online rendering).
- Prepare the midterm report and presentation for next week.

Week 3: Midterm

- Present the current results and determine what is left for the project.
- If time permits, prioritize implementing interaction features before collision.

Week 4:

- Complete additional features in addition to polish of visual aesthetics.
- Prepare the final report and presentation for next week.

Week 5:

- Present final results and reflect on the project.
- Figure out if the project can be extended further into capstone.

VI. Conclusion

In this proposal, I have described my interest in developing a cloth simulation feature for my hobby project, AlienGLRenderer. I have explained what questions I hope to answer and the methods and areas I will explore first. I have also introduced the possibilty of extending this project with extra features and possibly making this my capstone. All in all, I am excited to see how this project will turn out and what it will bring to my renderer.

REFERENCES

[1] J. Weil, "The synthesis of cloth objects," *ACM Siggraph Computer Graphics*, vol. 20, no. 4, pp. 49–54, 1986.

[2] X. Provot *et al.*, "Deformation constraints in a mass-spring model to describe rigid cloth behaviour," 1995.

[3] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically deformable models," *SIGGRAPH Comput. Graph.*, vol. 21, p. 205–214, Aug. 1987.

[4] D. Baraff and A. Witkin, *Large Steps in Cloth Simulation*. New York, NY, USA: Association for Computing Machinery, 1 ed., 2023.

[5] Y. J. Oh, T. M. Lee, and I.-K. Lee, "Hierarchical cloth simulation using deep neural networks," in *Proceedings of Computer Graphics International 2018*, pp. 139–146, Association for Computing Machinery, 2018.

[6] H. Bertiche, M. Madadi, and S. Escalera, "Neural cloth simulation," *ACM Trans. Graph.*, vol. 41, Nov. 2022.

[7] J. Bolz, I. Farmer, E. Grinspun, and P. Schröder, "Sparse matrix solvers on the gpu: conjugate gradients and multigrid," *ACM Trans. Graph.*, vol. 22, p. 917–924, July 2003.

[8] R. Tamstorf and E. Grinspun, "Discrete bending forces and their jacobians," *Graph. Models*, vol. 75, p. 362–370, Nov. 2013.

[9] M. Fisher, "Matt's webcorner," 2014. Available at https://graphics.stanford.edu/ mdfisher/cloth.html.