**Projective Dynamics**

R. Arietta, D. Rerucha, P. Huelin Merino

CIS563 Final Project

University of Pennsylvania 2014

1. **Different Aspects of the Simulation**

This project is based on the 2014 SIGGRAPH paper entitled “Projective Dynamics” by L. Kavan et al. Our final simulation code for this project has focused on successfully implementing the following aspects of the paper:

1. We implemented a stable local/global solver as described in the paper.

This solver followed the format established in the paper, in which each constraint was projected individually to minimize the potential energy imparted on the system by the positions of the particles. The implicit solver in the next phase then sought to globally reconcile the positions of each particle of each constraint and preserve a stable configuration for the mesh or model. This was done with a pre-factored Cholesky decomposition using the Eigen library.

In addition to the parallelization of this two-stage process described below, we worked hard to optimize this functionality to achieve real-time simulation frame rates. Much of the mathematical systems involved in both the local and global steps were pre-processed at the initialization of the simulation (or, similarly, with the addition or removal of any additional user-placed constraints and the subsequent resetting of the simulator).

1. Within this solver, we were able to successfully project and solve for the following various constraint types:
   1. Attachment Constraints – These constraints focus on maintaining the world space position of one single vertex of the mesh. In order to minimize the potential energy of this type of constraint, we sought to project the associated vertex directly to the fixed point position.
   2. Spring Constraints – These constraints, much like in Position Based Dynamics, focus on maintaining the distance between any two vertices of the mesh. To minimize potential energy of this type of constraint, we sought to move the two endpoint vertices proportionally towards (or away) from one another as a means of achieving the initial rest length of the spring.
   3. Volumetric Constraints – These constraints focus on preserving the volume of a deformable tetrahedral mesh unit as defined by four vertices and an initial rest volume. This constraint was projected by moving the vertices of the tetrahedron relative to one another to realize the initial rest volume but aligned with the deformed configuration.
2. In order to make the simulation run in real time, we focused on parallelizing the local projection stage of the solver. We did this with the OpenMP framework.

Unlike in traditional Position Based Dynamics simulators, each constraint in Projective Dynamics can be performed independently and separated from the global solution, avoiding iterative solver approaches. Since each constraint is so independent, this local projection lent itself to multi-threading. We chose to do this on the CPU with OpenMP rather than migrating any part of the code to the GPU, which would be more hardware-specific and require a lot of unnecessary build configuration to account for CUDA development.

1. Finally, we implemented collision detection of the simulated mesh objects with simple primitive mesh shapes.

This collision detection and correction is performed post-process after each time step is solved. If a cloth or tetrahedralized mesh vertex is discovered to be penetrating a mesh primitive beyond a defined threshold, its position is augmented by the penetration vector that defines its distance and direction along the normal to the nearest exterior surface.

1. **Challenges Faced**

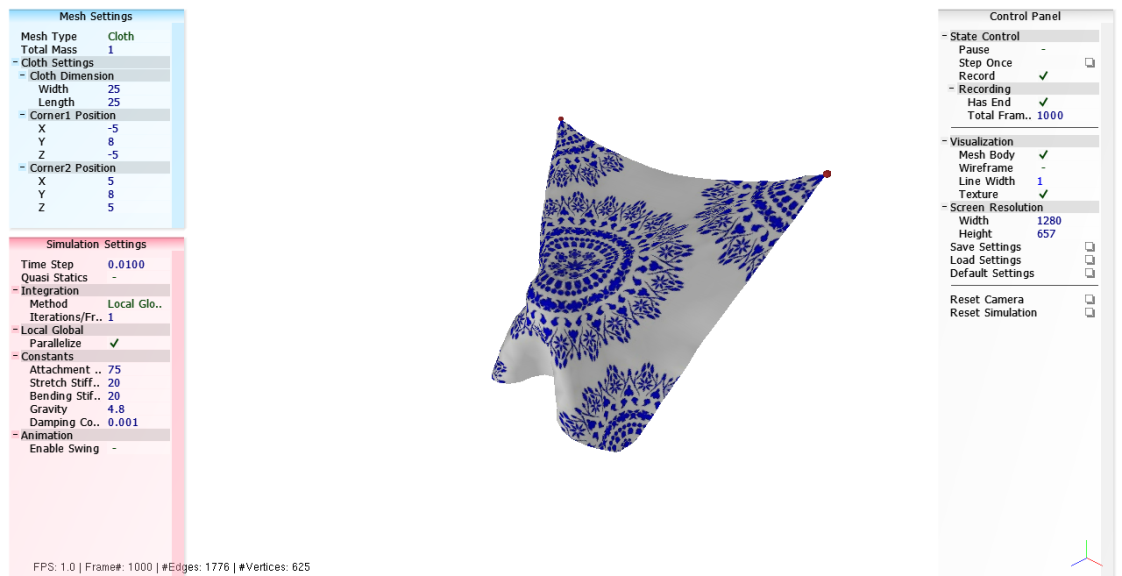
We faced several challenges in the course of development for this project. Initially, many of these challenges revolved around the simulation mathematics described in the paper. It took the three of us a fair amount of time to wrap our heads around the energy minimization involved in the local projection step, as well as the various matrix coefficients and solver decomposition of the global solver. Luckily, Tiantian Liu and Dr. Kavan were often available for clarification on this front.

Beyond this general mathematical understanding, our largest time commitment in the development of this project was probably spent in optimization. Our initial implementation was incredibly slow, with each frame running at nearly two-and-a-half minutes. We were able to improve this by orders of magnitude when we came to understand the parallelization of the projection and the ability this simulation allowed for preprocessing large parts of the implicit solver. The simulation now runs in real-time for meshes with upwards of 400 vertices.

We also ran into some issues with the collision detection. After implementing the post-process collision detection, we noticed that the simulation was not correctly accounting for intersections with primitives. We spent a lot of time testing and adjusting the code until finally realizing that the time step size needed to be made much smaller in order for this to work correctly. This is a limitation of the code structure defined in the paper, but this workaround provided accurate and reliable results.

1. **Results Obtained**

Here are some screen shots obtained from the simulation, along with a brief description of the image contents:



The above is an image rendered of a cloth simulation. The cloth is constrained at the top corners by attachment constraints and the cloth itself is structured with strain- and bend-constraints modeled with springs. You can see that the cloth is draped over a spherical primitive, collisions with which are successfully detected. This simulation is run at a vertex resolution of 25x25 with a time step size of 0.01 seconds. It runs in almost real time at 25 FPS.

**[INSERT IMAGE HERE]**

This rendered image shows an OBJ mesh whose deformable shape is constrained with tetrahedral volumetric constraints.

1. **Base Code Description**

Luckily for us, Tiantian Liu was able to provide us with a helpful framework for the simulation code. Tiantian Liu and Dr. Kavan were also present throughout the entire process for questions and implementation problems.

Much of the code for the simulation itself was stripped out due to an agreement with the other developers. However Tiantian’s framework did provide us with many of the necessary data structures (i.e., stripped down “Constraint” and “Mesh” classes were included which still provided us a fair amount of functionality), as well as all the necessary libraries and build structure.

This code also included functions for basic penetration detection of a point with a ground plane and with a sphere, which we worked into our post-process collision correction stage.

All of the simulation aspects described in the first section of this write-up were implemented by the development team and added to this provided base code.

1. **Breakdown of Project Work Done Per Teammate**

This project was largely split amongst the three people in the development team. We did most of the work for the local projection steps and the implicit global solver together, each contributing code and ideas. We also all worked as a unit to solve the collision detection and correction problem, and a lot of the code structuring, class definitions, simulation initialization, and optimization was done together as well.

Towards the end of the development process, Paula and Danny worked to successfully complete the volume preservation aspect of the simulation, while Ricky worked to parallelize the local solver using OpenMP.

1. **References**

This simulation implementation is based entirely on L. Kavan et al’s paper “Projective Dynamics: Fusing Constraint Projections for Fast Simulation,” SIGGRAPH 2014.