Post Mortem – Alex Denford

Overview:

Overall I found this project very interesting and I am very happy I kept on track with this project from the start. After the formative I had a good base to build off of and I found working through each of the features one at a time was quite easy and not overwhelming. I had an initial drop in motivation when I couldn't get my mouse interaction to work, but I stuck with it and fixed it after doing some research on the internet. Once I had mouse interaction in, getting the rest of the small features in to get a passing grade was quite simple. As I achieved a passing grade roughly 2 weeks before the project was due, it left me a lot of time to play around with the cloth and add some of the more advanced features. I am happy with the result of the project and it is surprising how such a seemingly complex system (cloth simulation) can be done with relative ease. Prior to this project I would never have considered trying to code a cloth simulation and I had very little knowledge on it.

To extend the project further I would really like to set everything up with proper graphics and set it in a realistic environment. Showing non-programmers a cloth simulation is interesting, but when they see a wireframe environment it becomes less relatable. I would like to see how the cloth simulation would look when used with a towel, flag or some other real life object that would use the mechanics of the cloth.

I would also like to try one of the more advanced integrators such as RK4 in a future project, to see the different results.

Features:

- Mouse interaction with cloth
    - Stretching, pulling, ripping etc. (left click)
    - Cutting with cloth (right click)
    - Starting a fire (middle click)
- Wind force, gravity force & fan force affecting the cloth
- Solid ground
- Cloth folding & cloth held by pins at top
- Dynamic GUI allows resizing, resetting and changing values
- Improved cloth structure (more constraints)
- Collision with sphere, pyramid + capsule
- Control of each object with object selection

Lessons learnt:

I have learnt a great deal about physics and cloth simulations with this project. I found this project very interesting and it was not something I was expecting to do prior to being given the brief. Reading about physics engines and how they fit in with rendering systems was very interesting and I learnt a lot about how full 3D games are created/ setup.

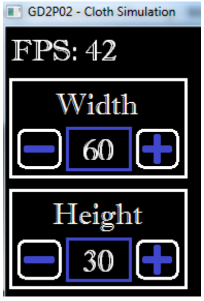
One of the biggest things I learnt with this project is efficiency. The MDS PC’s are quite powerful and therefore we generally do not need to worry about FPS nor efficiency. After running my simulation on my dual core laptop, I noticed that it was running at a very low FPS. I went through my code and found several areas that I could improve FPS. After changing a whole load of code I found that I managed to gain over 50 FPS! Small things like square roots (in distance checks) being called every frame can take away 3 frames per second. I knew about using square distances before (because you do not need to know the exact distance from A -> B) but I generally didn’t bother. Changing most of my distance checks to use squared distances helped a great deal with efficiency. I also added some primitive squared distance checks for collisions with the pyramid, to avoid doing all of the plane -> point calculations if the pyramid wasn’t even close to the cloth.

Design:

GUI:

I initially added 3 buttons / sprites to the interface for resetting, cloth width & cloth height. These were to fit the criteria of an interface to change the size of the cloth. I then added another to change the number of rods. I created a little function to find where the rods should be placed along the cloth (because not all cloth sizes will be divisible by the number of rods).

Later I added functionality to change the speed and area of the fan.

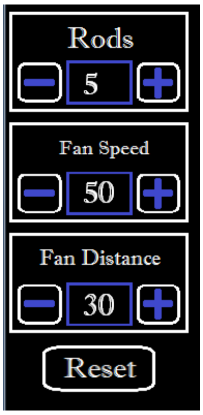


GD2P02 - Cloth Simulation

FPS: 42

Width: 60

Height: 30



Rods: 5

Fan Speed: 50

Fan Distance: 30

Fan:

I wrote a function to calculate the strength of the fan on a given piece of the cloth mesh. It finds the angle different between the fan and the direction vector from the cloth to the fan to check if the angle is within the bounds of the fan's angle setting. I then use an inverse function of the distance between the fan and cloth piece to create value used to make the fan's strength. (Closer = more powerful force, further = weaker).

Fire:

Before I set up fire I made it so my vertices / constraints were coloured based on their "stress value". This stress value was the larger of either constraint stress (how stretched or contracted the constraint is) or fire damage. After this I set up middle click to set the cloth on fire. Each constraint on fire increases in fire damage over time. I created a little formula to generate the change of it catching onto other constraints. I used a random function that is affected by how burnt the piece is already + a random chance. If the check passed, it would randomly set a nearby constraint on fire. Once a constraints stress is greater than 1.0f (100%) it breaks. This gave some interesting results and I am still playing with the values and way it works to create realistic fire spread.

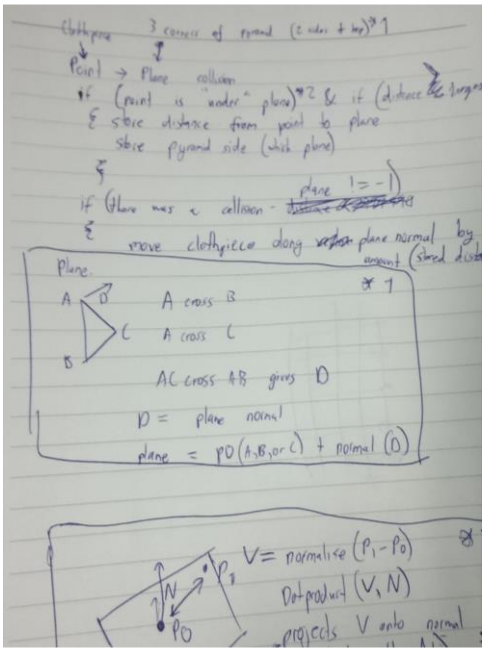
Collisions:

Sphere collision is just a simply distance check with each vertex of the cloth mesh. I solved the collision by finding the direction vector from the sphere to the cloth and moving the piece along this vector by the amount the piece is inside the sphere (essentially just moving the piece to the edge of the sphere).

Capsule collision is pretty simple too, (very similar to sphere collision) although you need to find the shortest distance from the point (cloth piece) to the line (capsule). The radii of the cloth piece sphere + capsule radius can then be subtracted from this distance to find if there was a collision. Solving the collision is the same as in sphere.

I found pyramid collision more tedious but my initial idea worked correctly. My method worked as follows:

```
For each plane of the pyramid (each side)
    Check if the piece is "under" the plane
    If it is, and the distance is the shortest found so far, store this new distance + which plane
If a piece is not under any of the planes, exit the collision checking
Otherwise, move the piece along the plane's normal vector by magnitude of the shortest distance
```



```
Clothpiece        3 corners of pyramid (2 sides + top)*1
   ↓                  ↓
Point → Plane collision
 if (point is "under" plane)*2 & if (distance [illegible]
  { store distance from point to plane
      store pyramid side (which plane)
  }
 if (there was a collision - ~~distance [illegible]~~ plane != -1)
  {
      move clothpiece along ~~vector~~ plane normal by
                                    amount (stored dist
```

Plane

A cross B

A cross C

AC cross AB gives D

D = plane normal

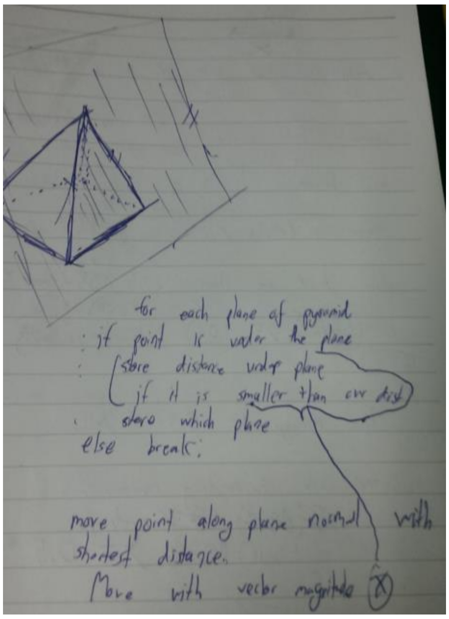
plane = P0(A,B,or C) + normal (D)

V = normalise $(P_1 - P_0)$

Dot product (V, N)

projects V onto normal

*1
A
B
C
D
*
N
P_1
P0



```
for each plane of pyramid
 if point is under the plane
  [store distance under plane
   if it is smaller than cur dist
    store which plane
 else break;

move point along plane normal with
shortest distance.
 Move with vector magnitude X
```

Known Issues:

Collisions are non-continuous in my cloth simulation (discrete collision closing). This means that at slow movements, the collisions are quite accurate and appear (visually) correct. If the objects move quickly through the cloth though, some clipping can occur due to the speed of the collision solving not keeping up with the movement of the object itself. If I were to extend this project I would do some more research on more advanced collision techniques and in particular collision solving methods.

Resetting the cloth with a height greater than that which reaches the floor (spawning in the floor) can cause lag spikes because of the collision solving working extremely hard to fix it.

The cloth can fall in on itself (no self-collision). I had a small attempt at adding more constraints that skip vertices to help improve the self-collision but it added quite a lot of calculations and thus dropped the FPS considerably. With a bit more time I would like to play around with the number of constraints and where they are used (more important for corners as opposed to the centre of the cloth etc.). I also didn't have time to try out a basic collision solver for each vertices against the others to help stop self-collision.

I am having some issues with the solving of the collision as it is making the cloth pieces to jitter or move away with significant force (as opposed to just placing the cloth piece on the edge of the pyramid / plane). If I had more time I would like to look further into what is causing this issue in order to fix it.

References:

http://en.wikipedia.org/wiki/Verlet_integration

http://gamedevelopment.tutsplus.com/tutorials/simulate-tearable-cloth-and-ragdolls-with-simple-verlet-integration--gamedev-519

http://www.braynzarsoft.net/index.php?p=Picking

http://www.toymaker.info/Games/html/picking.html

https://www.youtube.com/watch?v=8bAQbJzqa-0

http://en.wikipedia.org/wiki/Line%E2%80%93plane_intersection

http://www.flipcode.com/archives/Point-Plane_Collision.shtml

https://www.youtube.com/watch?v=YFLvA1aOZkw

http://web.archive.org/web/20140208014400/http://rastertek.com/