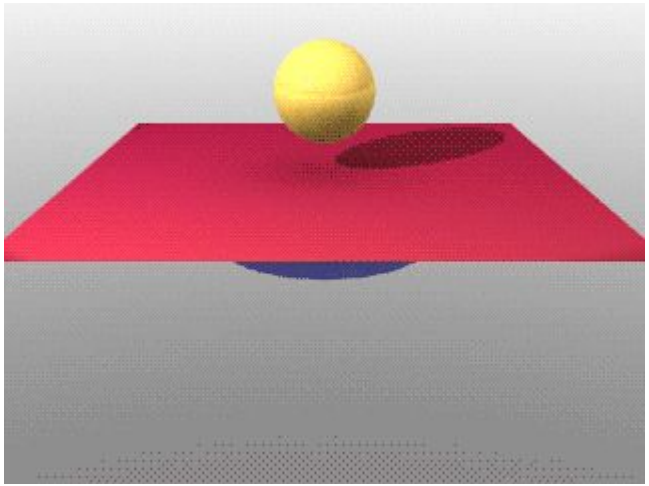
Compute Shader cloth

github.com/likangning93/GPU_cloth

Motivation

Animators need to see cloth motion quickly to make edits and control decisions.



Academics: “It’s physically accurate!”

Artists: “But what if I want her dress to swish like *this* instead? or like *this*? or like *this*? or like ...”

https://commons.wikimedia.org/wiki/File:Cloth_Simulation.gif

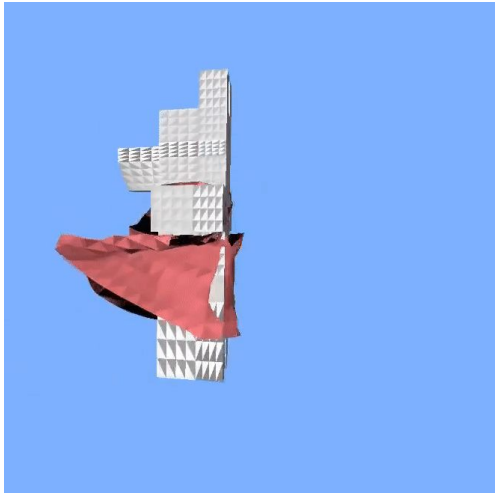
Disney Pixar, Inside Out:
https://media.giphy.com/media/uT2g55XqavoRy/giphy.gif

What I did:

- convertor from obj -> simulation internal format
  - including internal constraint generation
- basic rendering with a geometry shader
- simulation solver - caveats and details to follow
- performance analysis tools

How close did I get?

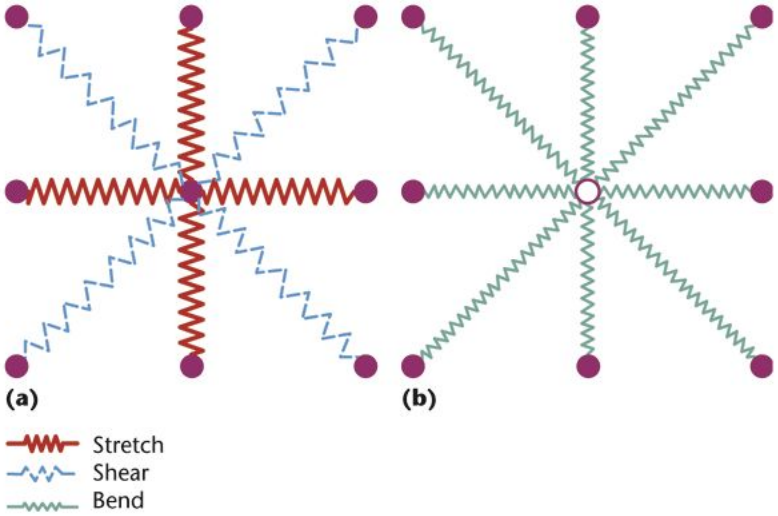
There’s still a lot of work to be done… but it’s a start!



Disney Pixar, Inside Out:
https://media.giphy.com/media/uT2g55XqavoRy/giphy.gif

Cloth Simulation

- cloth is often simulated as a mass-spring system
- basically, every step involves moving a bunch of points according to forces and maintaining distance relationships to neighbors



Position Based Dynamics - Overview

In each timestep:

1) apply external forces (gravity) and damping to velocities
2) compute predicted positions
3) correct predictions with internal constraints
   a) the “springs” in the mass spring system
   b) solved *N* times -> more stable as *N* increases
4) generate and resolve collision constraints
5) set start positions and velocities for the next timestep

details and math here:
http://matthias-mueller-fischer.ch/publications/posBasedDyn.pdf

Position Based Dynamics - On the GPU?

In one timestep:

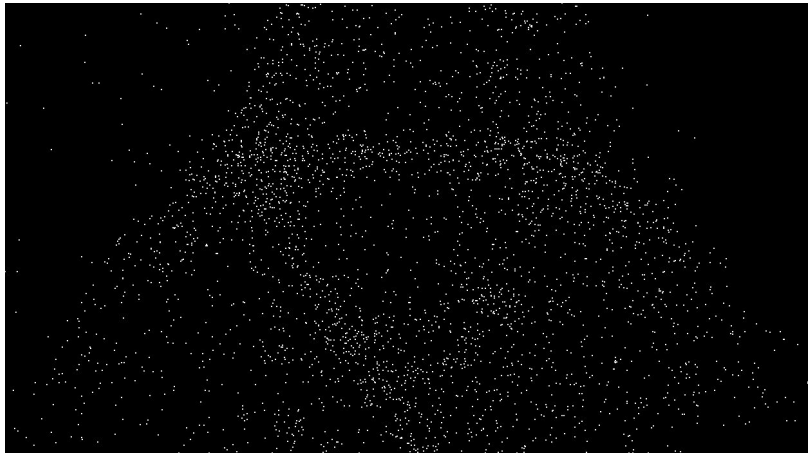
1) apply external forces (gravity) and damping to velocities - parallelize by vertex
2) compute predicted positions - parallelize by vertex
3) correct predictions with internal constraints - parallelize by constraint
    a) the “springs” in the mass spring system
    b) solved *N* times -> more stable as *N* increases
4) collision constraints - at most 1 per vertex, so parallelize by vertex
5) set start positions and velocities for the next timestep - by vertex

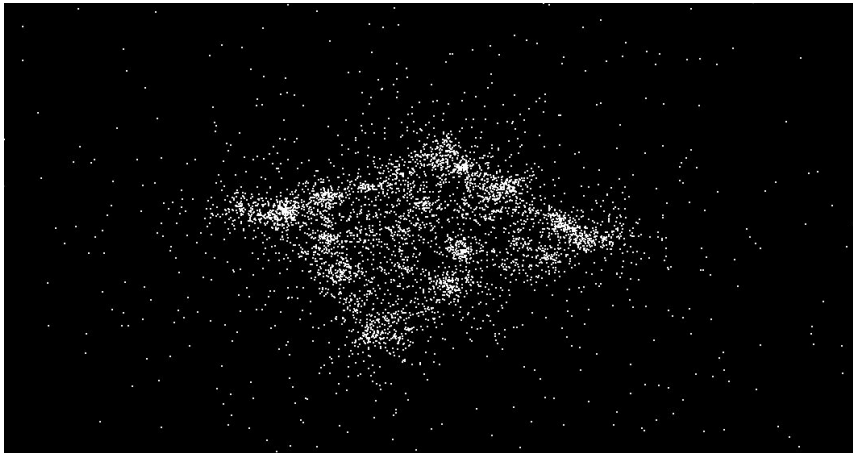
So basically the simulated cloth needs two types of data in buffers:

- vertex info (velocity, position, mass)
- constraint info (“these two vertices shouldn’t separate by more than x”)

Caveats and Details: how does SSBO data work?!

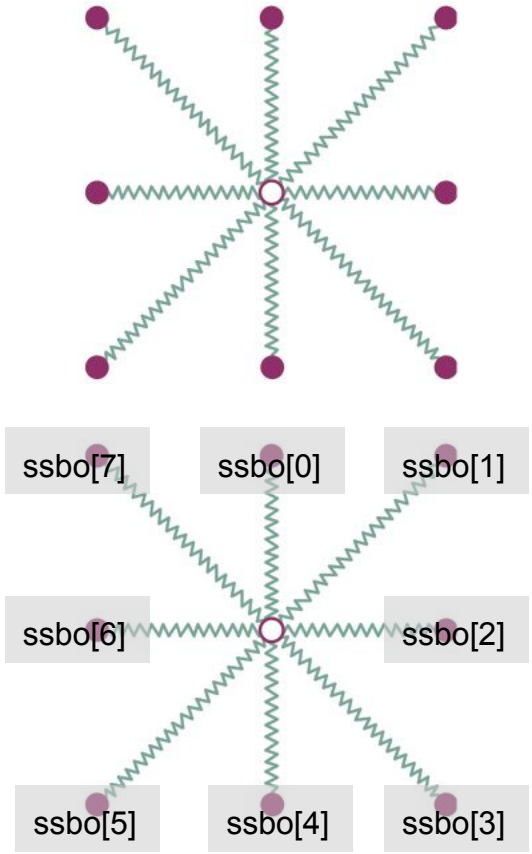
- SSBO: shader storage buffer object
- like a more generalized version of a VBO
- compute shaders expect data to be transferred as vec4s - transfer positions as vec3s and you get something like this instead of this:





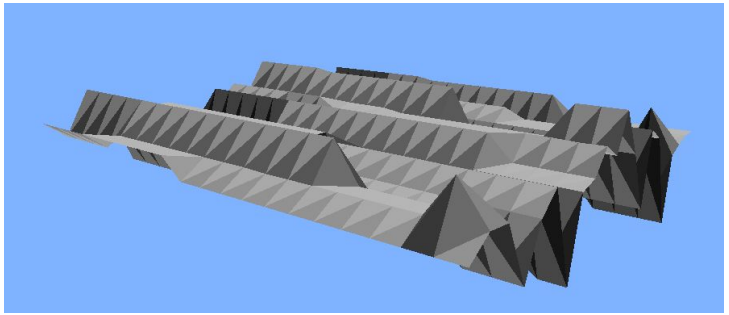
Caveats and Details: Constraints

- a vertex's position is corrected by **constraints**, which in turn are based on its neighboring vertices
- each vertex may have up to 8 constraints!
- how to parallelize without race conditions?
  - atomics? OpenGL compute only offers atomics for ints!
- **solution**:
  - build multiple buffers of non-conflicting constraints
  - evaluate constraint sets in separate passes



Caveats and Details: incoherent memory, tragedy

- Compute Shaders operate on “incoherent” memory
- so compute shader invocation *B* might not wait for a previous invocation *A* to finish up with data before starting
- 3-week long bug: with only gravitational influence, my cloth did this:



- solution: glMemoryBarrier
- prevents access to *A*’s memory
- until *A* is done with it
- PBD’s stability: even before memory barriers it *kind of* worked!

Caveats and Details: incoherent memory, tragedy

How did I find out about this?

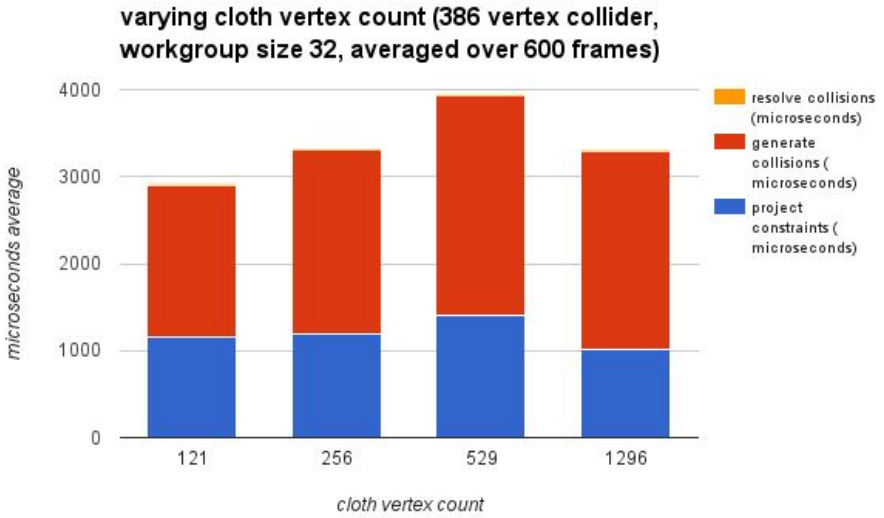
- ran simulation with debug code between shader calls: problem disappeared!
- ran simulation on a machine with a slower CPU: problem disappeared!
- finally read up on the OpenGL compute memory model: https://www.opengl.org/wiki/Memory_Model

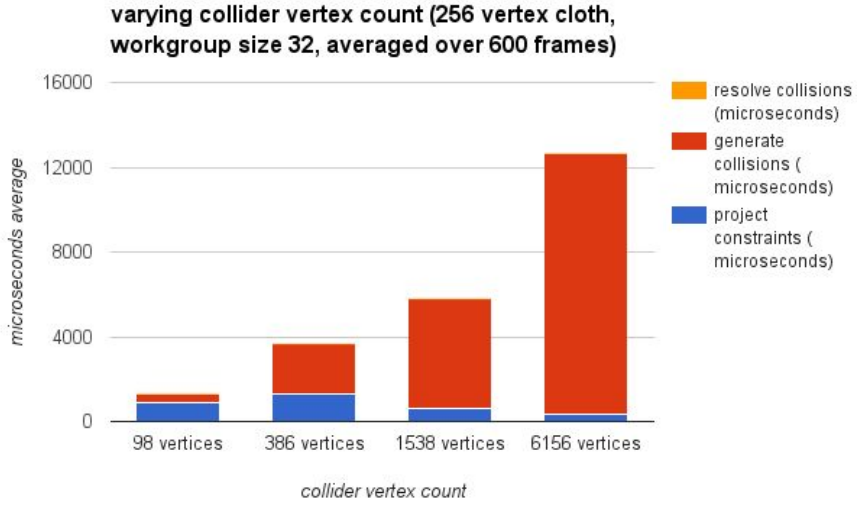
Performance: Analysis Overview

- data collected using OpenGL Timer Queries on specific stages
  - internal constraints stage
  - collision detection stage (naive detection)
  - collision resolution stage
- technique: http://www.lighthouse3d.com/tutorials/opengl-timer-query/
- varied:
  - Cloth size
  - Rigidbody collider size
  - compute shader work group size

```
GLint64 startTime, stopTime;
unsigned int queryID[2];

// generate two queries
glGenQueries(2, queryID);
...
// issue the first query
// Records the time only after all previous
// commands have been completed
glQueryCounter(queryID[0], GL_TIMESTAMP);

// call a sequence of OpenGL commands
...
// issue the second query
// records the time when the sequence of OpenGL
// commands has been fully executed
glQueryCounter(queryID[1], GL_TIMESTAMP);
...
// wait until the results are available
unsigned int stopTimerAvailable = 0;
while (!stopTimerAvailable) {
    glGetQueryObjectiv(queryID[1],
                                GL_QUERY_RESULT_AVAILABLE,
                                &stopTimerAvailable);
}

// get query results
glGetQueryObjectui64v(queryID[0], GL_QUERY_RESULT, &startTime);
glGetQueryObjectui64v(queryID[1], GL_QUERY_RESULT, &stopTime);
```

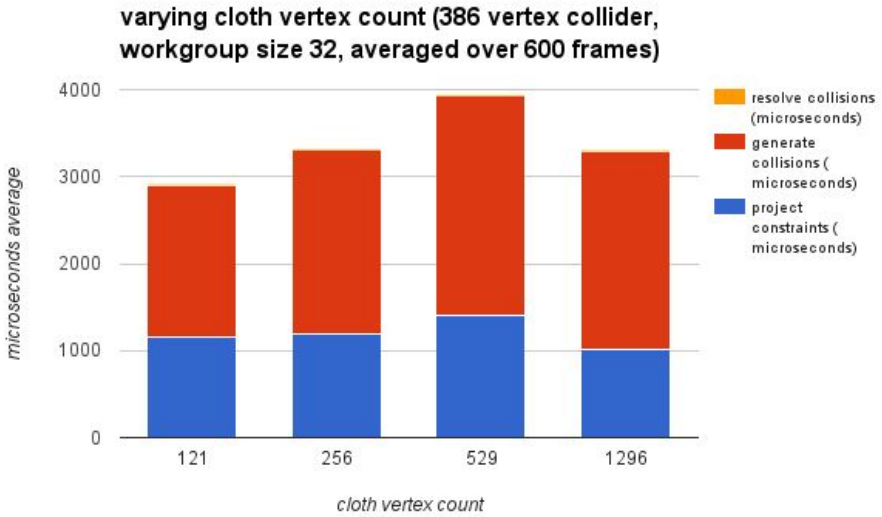
Performance: varying cloth size and collider size





- In general, overall computation time increases with more vertices
- however, drops in constraint solving with increasing vertices needs further investigation
- collision generation time increases as cloth vertices increase, but not as dramatically as when rigidbody collider vertices increase
- probably because increasing rigidbody vertices increases memory access per cloth vertex

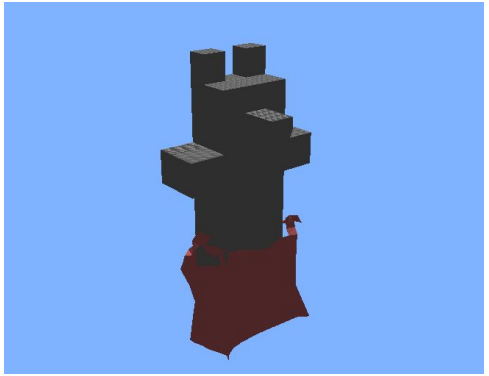
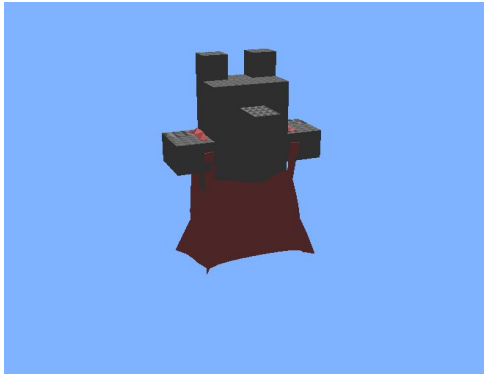
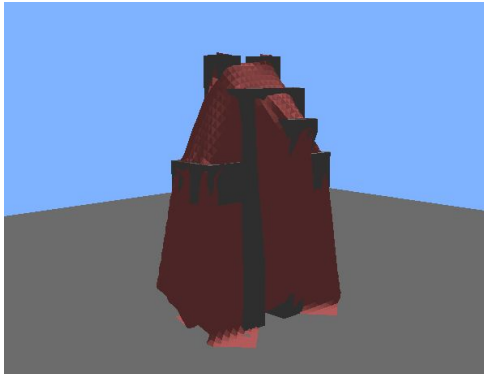
Performance: varying cloth size and work group size



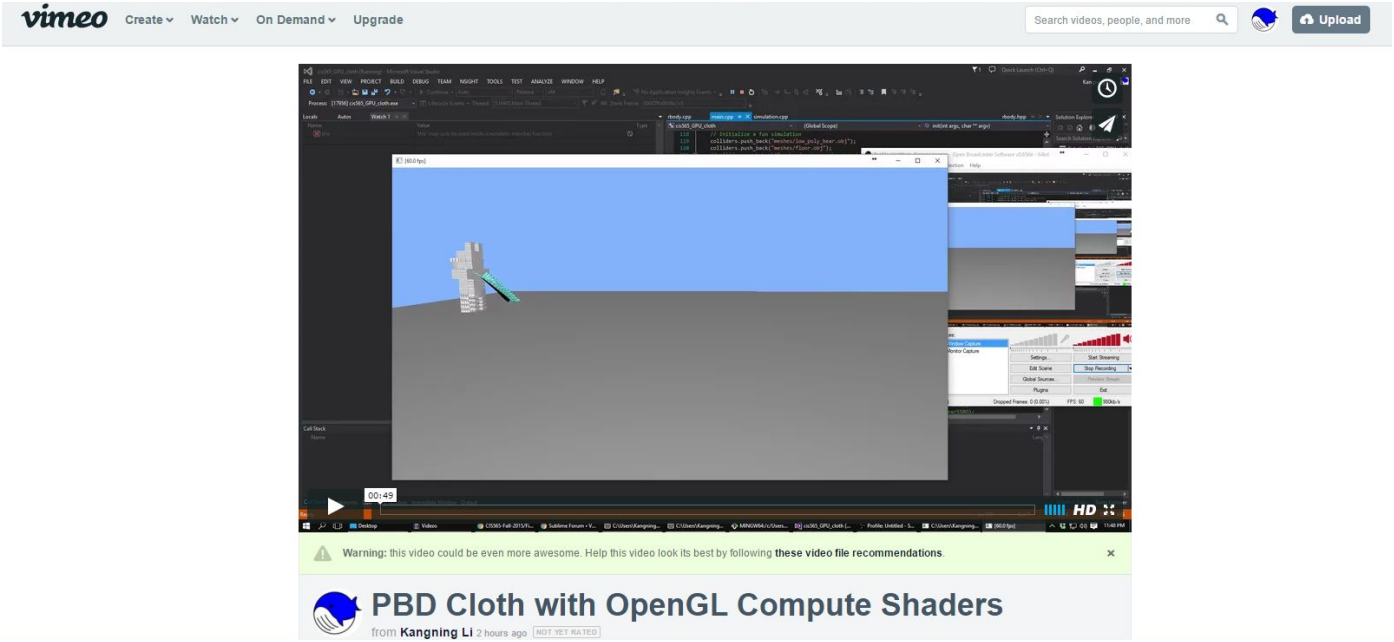
- again, drop in constraint solving time between tests needs further investigation
- increasing work group size seems to help up to a point
- however, more data needs to be collected

Future Work

- “real” animation system - BVH player?
- octree acceleration
  - construction with space filling curves
  - short stack traversal
  - https://www.cse.iitb.ac.in/~rhushabh/publications/octree
- cloth self collision

Questions?


vimeo
Create
Watch
On Demand
Upgrade
Search videos, people, and more
Upload
00:49
HD
Warning: this video could be even more awesome. Help this video look its best by following these video file recommendations.
PBD Cloth with OpenGL Compute Shaders
from Kangning Li 2 hours ago
NOT YET RATED