Position based dynamics in Houdini using VEX

https://github.com/TaroAndMulan/HoudiniPhysicsSimulation-Vex-/

This project is a compilation of small simulation that I made throughout this semester. Most of these simulations are based on position base dynamics, with some slight variation between them.

Why Houdini

I want to focus on the physics simulation part, so I decide to use 3D software instead of c++. I tried blender, but I hated it, so I end up choosing Houdini.

Restriction

All solvers are written from scratch using VEX scripting language. Houdini is only used for geometry creation and rendering.

Position based dynamics

while simulating
  for all particles i
    $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t\, \mathbf{g}$
    $\mathbf{p}_i \leftarrow \mathbf{x}_i$
    $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta t\, \mathbf{v}_i$
  for n iterations
    for all constraints C
      solve$(C, \Delta t)$
  for all particles i
    $\mathbf{v}_i \leftarrow (\mathbf{x}_i - \mathbf{p}_i)/\Delta t$

$\Delta t_s \leftarrow \Delta t/n$
while simulating
  for n substeps
    for all particles i
      $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t_s \mathbf{g}$
      $\mathbf{p}_i \leftarrow \mathbf{x}_i$
      $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta t_s \mathbf{v}_i$
    for all constraints C
      solve$(C, \Delta t_s)$
    for all particles i
      $\mathbf{v}_i \leftarrow (\mathbf{x}_i - \mathbf{p}_i)/\Delta t_s$

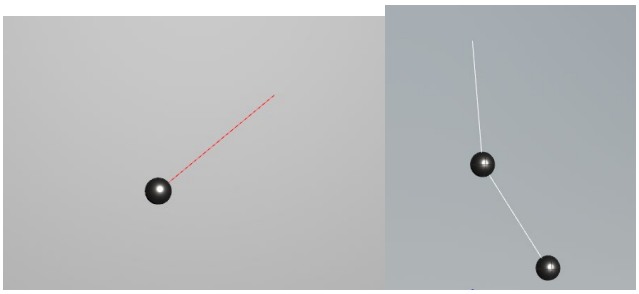
https://matthias-research.github.io/pages/tenMinutePhysics/09-xpbd.pdf

#1 Pendulum and double pendulum (PBD)

My first simulation in Houdini, using distance constraint to keep the ball in circular path.

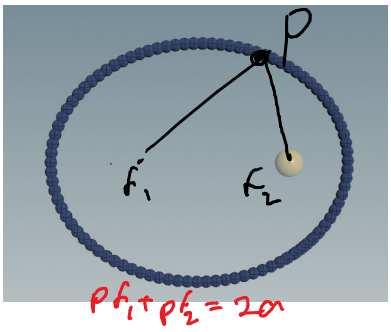
Important stuff that I learn from this simulation

More sub step → less energy dissipation



#2 Ellipse constraint (PBD)

After playing around with sim#1 and understand how the distance constraint and position-based dynamics work in general, I derived an ellipse constraint that confine particle to move in elliptical orbit using an ellipse constant sum property.

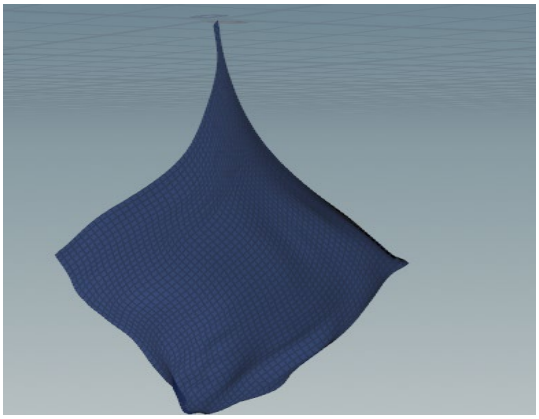


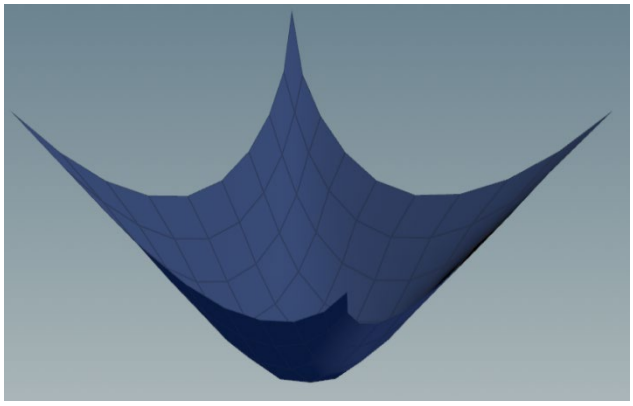
$$C = |\vec{p} - \vec{f}_1| + |\vec{p} - \vec{f}_2| - 2a$$

$$\nabla C = \frac{(\vec{p} - \vec{f}_1)}{|\vec{p} - \vec{f}_1|} + \frac{(\vec{p} - \vec{f}_2)}{|\vec{p} - \vec{f}_2|}$$

#3 Cloth simulation (PBD)

Houdini is used to create points grid. Each point is connected to it's neighbor like a network. Only the distance constraint between particles is used (no bending and collision). The algorithm is based on PBD but with some slight modification. There are 2 versions, first version loop over all points and perform correction for each constraint attached to that point. Second version loop over all constraint and perform correction for each particle in that constraint.



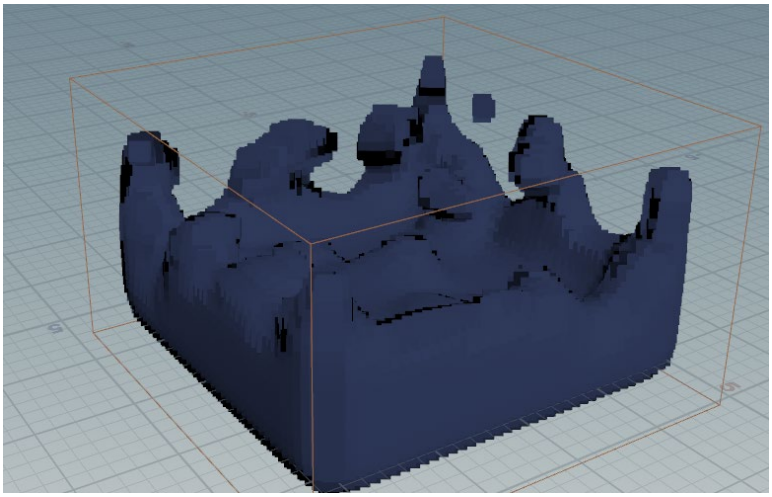


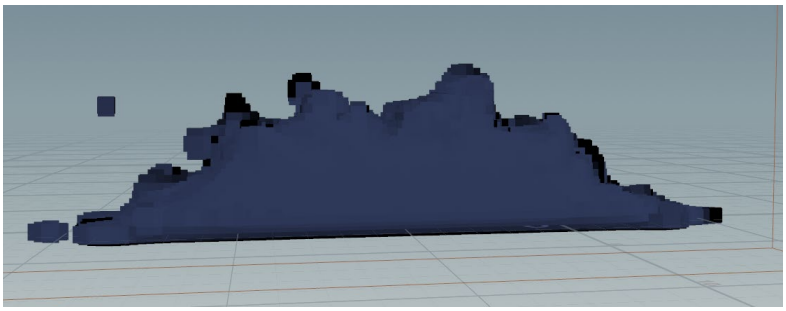
Fluid based dynamics

Based on this paper

Position Based Fluids, Miles Macklin, Matthias Müller, ACM Transactions on Graphics (SIGGRAPH 2013), 32(4)
[PDF] [Slides] [Project Page]

Implementing this paper in Houdini is straightforward, but it was hard to debug, since there is so much stuff that could go wrong and I didn't even know where to begin debugging. In the end, I have to omit the vorticity confinement and XSPH viscosity out of the algorithm since it breaks the simulation or have no effect at all, it also slows down the simulation significantly.





This link also provides some guideline that help me a lot

http://graphics.stanford.edu/courses/cs348c-20-winter/HW_PBF_Houdini/index.html

Snail chasing snail chasing snail

This simulation is inspired by a problem in the book "200 Puzzling Physics Problems With Hints and Solutions"

> **P1** Three small snails are each at a vertex of an equilateral triangle of side 60 cm. The first sets out towards the second, the second towards the third and the third towards the first, with a uniform speed of 5 cm min^{-1}. During their motion each of them always heads towards its respective target snail. How much time has elapsed, and what distance do the snails cover, before they meet? What is the equation of their paths? If the snails are considered as point-masses, how many times does each circle their ultimate meeting point?

According to the book's solution, the snail will move in a logarithmic spiral and will eventually collide at the center. However, the book only contains the calculation, but no actual path is drawn. What if there are more snails? It is unclear which kind of curve its motion would produce. Therefore, I decide to simulate this problem in Houdini to answer this question.

For each timestep, I move all particle at the same time rather than doing it one by one. (This guarantee that the particle will stay in a symmetric configuration with no bias on which point the program move first)

P1.old ← P1

P2.old ←P2

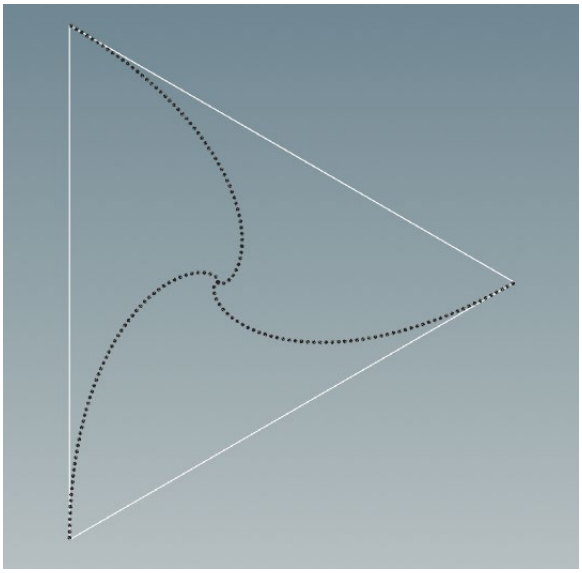
P3.old ← P3

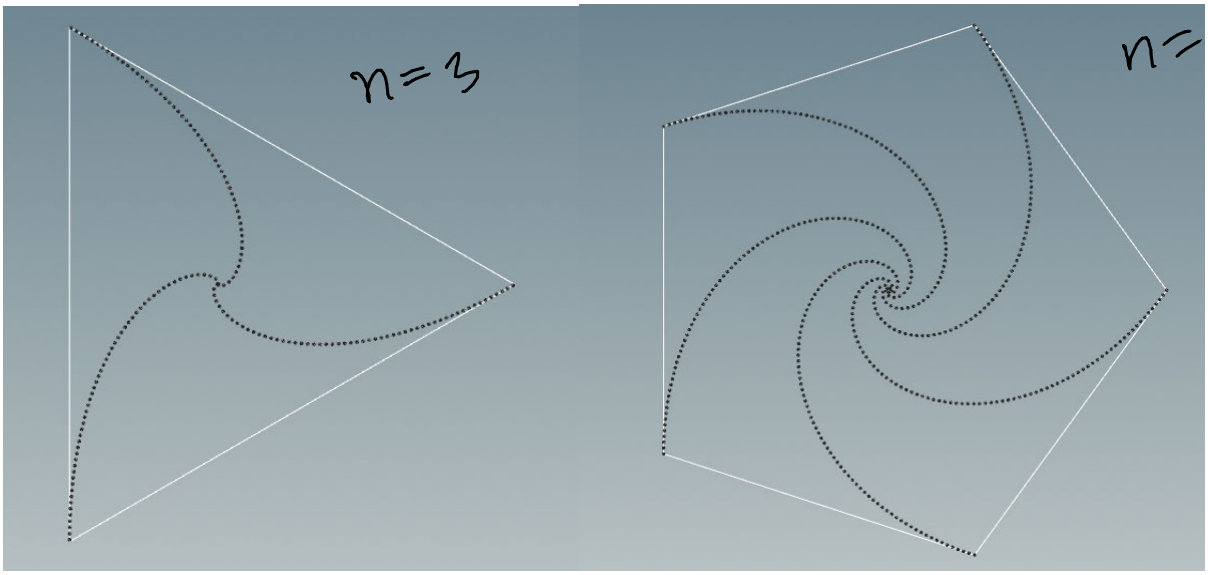
For all particle

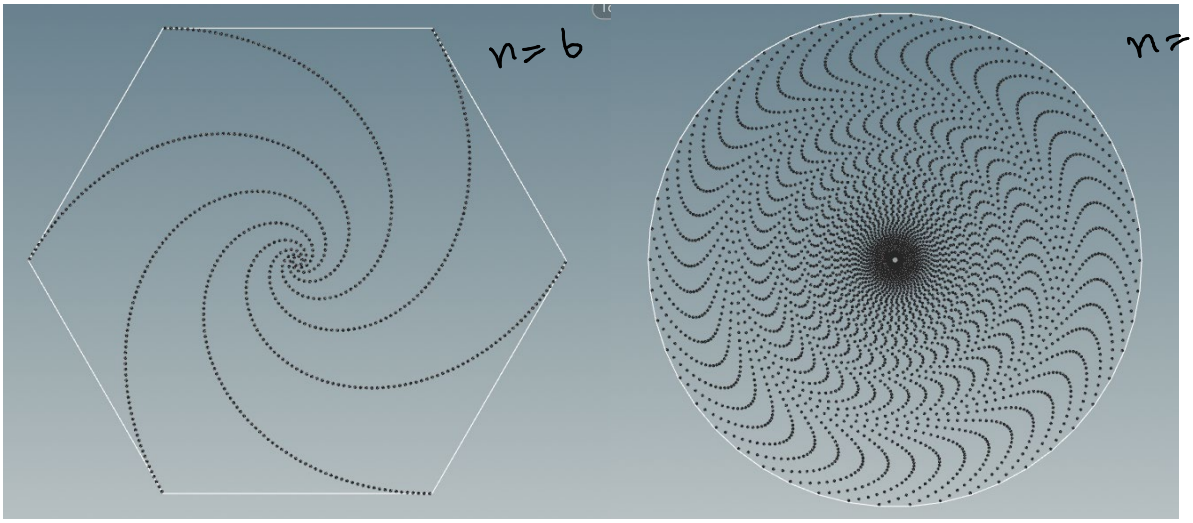
Move P1 toward P2.old

Move P2 toward P3.old

Move P3 toward P1.old




n=3
n=5


n=6
n=50

Soft body simulation (XPBD)

I used Houdini to transform an object into tetrahedron meshed.

This simulation used tetrahedron volume and distance constraint. The implementation is based on this video tutorial.

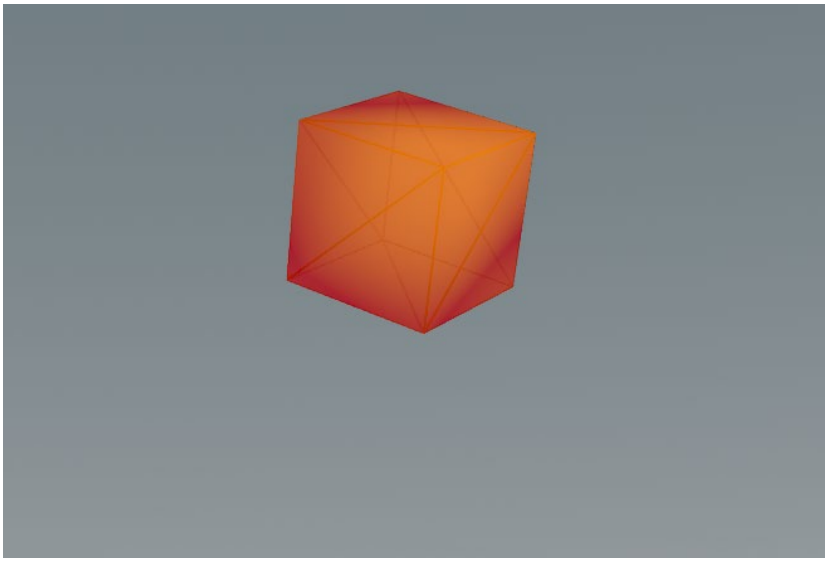
https://www.youtube.com/watch?v=uCaHXkS2cUg&t=3s

https://matthias-research.github.io/pages/tenMinutePhysics/10-softBodies.html

In the demo, the cube is made up of 12 tetrahedron and 25 edges.

There is a bug, if a geometry has too many points, the simulation will blow up.

Adjusting volume and edge compliance is crucial, the video tutorial use 0.0 for volume and 100.0 for edge, however that number doesn't work for me, without volume compliance my cube spin uncontrollably.



My Problem with houdini

Looping (attribute wrangle) in Houdini is tricky. For example, if you tell Houdini to run over all point in the geometry, it will do it in parallel, which make a certain algorithm fail.

For example,

Loop over all points:

Do A

Do B

Do C

How I thought Houdini works

Point 1 Do A

Point 1 Do B

Point 1 Do C

Point 2 Do A

Point 2 Do B

Point 2 Do C

But Houdini run in parallel like this

Point 1 Do A

Point 2 Do A

Point 1 Do B

Point 2 Do B

Point 1 Do C

Point 2 Do C

Moreover, if you run a for loop and change something in the geometry, all change will only be updated at the end of the final loop.

For example,

Point1.Color = blue

Loop:

iteration 1: Point1.Color = red

iteration 2: X = Point1.Color

.

.

.

End loop

X will return blue instead of red since Houdini only update geometry attribute at the end of the final loop. There is a work around and there are many types of for loop to use in Houdini, but it isn't obvious at all for an absolute beginner. I end up wasting several hours wondering why some of my simulation doesn't work.