Homework 4 - Position Based Dynamics (PBD)

Pair Submissions Allowed

Deadline: on canvas

Carefully Follow Instructions Below

In this assignment, we will implement components of the PBD paper, building on the PBD code derivation and pseudo-code.
Position Based Dynamics paper: https://matthias-research.github.io/pages/publications/posBasedDyn.pdf

Main algorithm loop:

(1) **forall** vertices i
(2) initialize $\mathbf{x}_i = \mathbf{x}_i^0, \mathbf{v}_i = \mathbf{v}_i^0, w_i = 1/m_i$
(3) **endfor**
(4) **loop**
(5) **forall** vertices i **do** $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i \mathbf{f}_{ext}(\mathbf{x}_i)$
(6) dampVelocities$(\mathbf{v}_1, \ldots, \mathbf{v}_N)$
(7) **forall** vertices i **do** $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
(8) **forall** vertices i **do** generateCollisionConstraints$(\mathbf{x}_i \rightarrow \mathbf{p}_i)$
(9) **loop** solverIterations **times**
(10) projectConstraints$(C_1, \ldots, C_{M+M_{coll}}, \mathbf{p}_1, \ldots, \mathbf{p}_N)$
(11) **endloop**
(12) **forall** vertices i
(13) $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$
(14) $\mathbf{x}_i \leftarrow \mathbf{p}_i$
(15) **endfor**
(16) velocityUpdate$(\mathbf{v}_1, \ldots, \mathbf{v}_N)$
(17) **endloop**

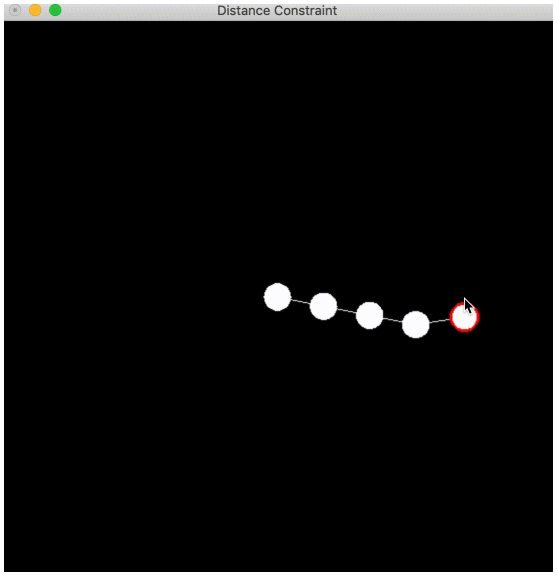
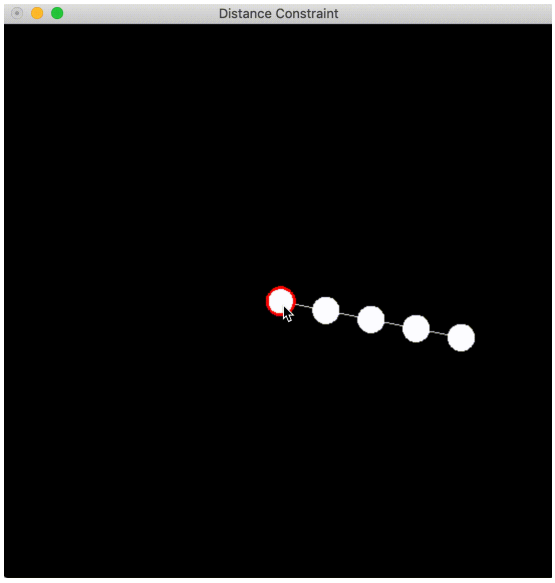
Startup Skeleton code:
https://www.dropbox.com/scl/fo/jaz8heppqpurpud3r39qw/h?rlkey=mwor8zubl65iupyfbe7isk6il&dl=0
For all the sub-assignments below:

- Your code must compile to be graded.

- We provide you with **startup skeleton** code, in which:
    - You will have instructions on where to implement the tasks requested below.
    - You may add helper functions, classes and other code as you see necessary.
    - You may modify existing code, but do so with caution. Bugs can easily be introduced in simulation code, e.g. by flipping + with a - sign or changing parameter values.

1. Distance Constraint

   Implement the distance constraint we discussed in class.
   The distance constraint should link between all the particles. When a user moves one of the particles, it should drag all the other particles with it.
   User interaction instructions: left click with mouse button to select particle. Then after releasing the left click, move the particle around. Left click again to deselect the particle.
   Start from the python skeleton code here: worm.py
   You may only need to modify distance_constraint() function
   See examples here:

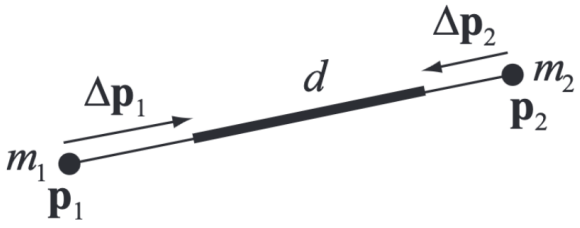
Before	After
	

See page 4 in the paper for more details on distance constraint:

To give an example, let us consider the distance constraint function $C(\mathbf{p}_1, \mathbf{p}_2) = |\mathbf{p}_1 - \mathbf{p}_2| - d$. The derivative with respect to the points are $\nabla_{\mathbf{p}_1} C(\mathbf{p}_1, \mathbf{p}_2) = \mathbf{n}$ and $\nabla_{\mathbf{p}_2} C(\mathbf{p}_1, \mathbf{p}_2) = -\mathbf{n}$ with $\mathbf{n} = \frac{\mathbf{p}_1 - \mathbf{p}_2}{|\mathbf{p}_1 - \mathbf{p}_2|}$. The scaling factor s is, thus, $s = \frac{|\mathbf{p}_1 - \mathbf{p}_2| - d}{w_1 + w_2}$ and the final corrections

$$\Delta \mathbf{p}_1 = -\frac{w_1}{w_1 + w_2} (|\mathbf{p}_1 - \mathbf{p}_2| - d) \frac{\mathbf{p}_1 - \mathbf{p}_2}{|\mathbf{p}_1 - \mathbf{p}_2|} \quad (10)$$

$$\Delta \mathbf{p}_2 = +\frac{w_2}{w_1 + w_2} (|\mathbf{p}_1 - \mathbf{p}_2| - d) \frac{\mathbf{p}_1 - \mathbf{p}_2}{|\mathbf{p}_1 - \mathbf{p}_2|} \quad (11)$$

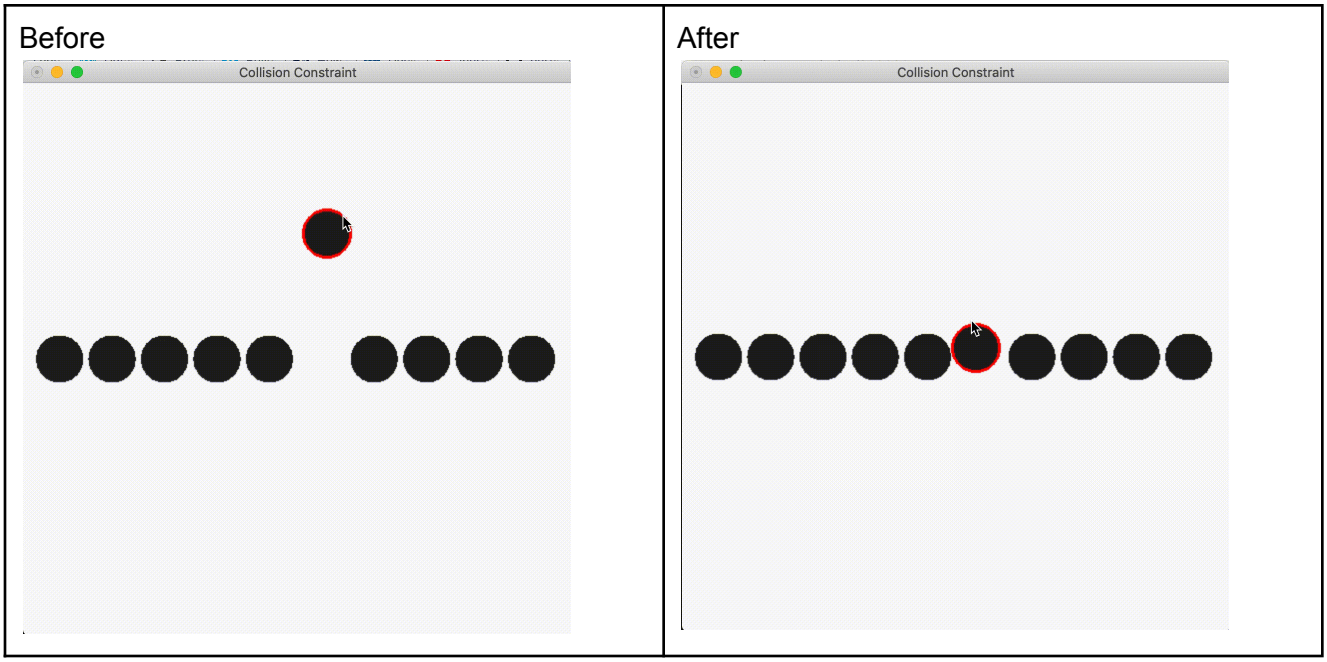


2. Collision Constraint

   This constraint is similar to the distance constraint, but uses inequality instead, as discussed in class. That means, this constraint will only be activated in case of a collision, and will act similarly as a distance constraint to resolve the collision.

   Start from the python skeleton code here: collision.py

   You may only need to modify collision_constraint() function



3. Point Constraint

   Implement the point constraint we discussed in class. The point constraint should lock one of the particles to the center coordinates (0.0, 0.0)

   Start from the python skeleton code here: point.py

   You may only need to modify point_constraint() function

   Windows users: note that there may be a stuttering with the main point constraint particle. This is normal.

   User interaction instructions: left click with mouse button to select particle. Then after releasing the left click, move the particle around. Left click again to deselect the particle.

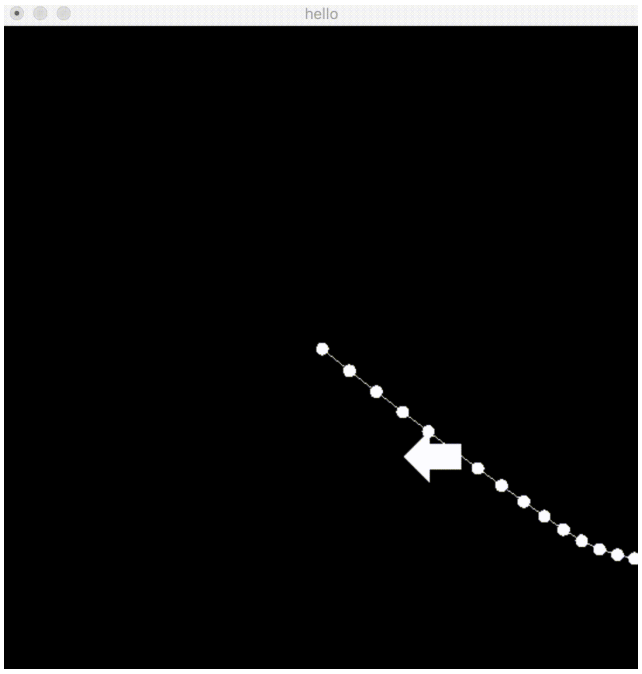
4. Rope Simulation

   Integrate your point and distance constraints to get a rope simulation.

   Start from the python skeleton code here: rope.py

   See hints in starter code. Try to get your code to imitate the result below – this means you may need to change your parameter settings.

Completed Result:



Source code and Implementation Instructions

- Start early! It might take some time to config your pipeline to compile properly.
- Your solution must include source code in Python.
- Packages used in previous Python homework are allowed. Other packages are not allowed. Contact the instructor if you are not sure.

General Instructions

- **You are allowed to perform this assignment either in pairs, or by yourself.**
- You can share ideas and discuss general principles with others, but all the code that you submit must be your own work; do not share your homework code with others, and do not look for previous solutions by using a search engine or visiting sites like GitHub. Please consult a TA or the instructor if you have any questions about this policy.
- Work incrementally. Start with a small and simple example and work your way to full functionality.
- Your program should not crash in normal operation as defined by spec above.
- Submit your solution as a zipped file, which is a file that ends with ".zip", to the class website on the class website at canvas
- The file should be named "<your full name1> <your full name2 (if working with in a pair)>.zip"

End :-)