### Gravity

### By Peter Milkman

### Instructions

Click or touch anywhere in the central area to place object.

Objects will combine masses and momentums when they get too close.

Object radii are logarithmic to their mass.

Drag to give your object direction.

### What have I created?

This app is an interactive rudimentary gravitational physics simulator. It allows the user to create as many objects with their own masses, velocities and angles, then watch the momentum and gravity of the objects interact with each other.

### Why did I decide to create this?

Without being a designer, this felt like the most fun and compelling interactive experiment relating to gravity that I could create that didn't require real art assets. I did come up with the original concept independently before I discovered [this Flash app](http://www.nowykurier.com/toys/gravity/gravity.html" \t "_blank) and [this demo page for PhysicsJS](http://wellcaffeinated.net/PhysicsJS/" \l "demo-1" \t "_blank), but those pages did serve as useful reference points.

### Process and Tools

With the concept in my head, the following is the step-by-step process I took to complete the app:

* Created a private code repository on [GitHub](http://github.com" \t "_blank).
* Cloned it to my local computer and web server.
* Opened the repo in [my editor](http://www.sublimetext.com/3" \t "_blank).
* Set up the basic index.html with a canvas basic css and javascript files.
* Set up the click/touch events to plot objects on the canvas.
* Set up dragging lines from those objects around the canvas.
* Set up basic requestAnimationFrame animation rendering loop with canvas clearing.
* Add real-time timer and algorithm to move objects along selected angles and speeds.
* Spend some time looking over many web pages for formulas to inform gravity/momentum algorithms.
* Implement and refine object gravitational interaction loop.
* Implement collision/combining algorithm.
* Add header UI with size selectors and implement the logic behind them.
* Add footer reset button and logic.
* Test IE10/Firefox/Safari/iPhone, fix bugs.
* Have friends test. One friend complained the animation flickered. Added canvas double-buffering as a result.
* Find a few areas for optimization and possible bugs/edge cases.
* Write this documentation.

The only third party lib I used wound up being basic jQuery. I was already underway with the project when I discovered [the PhysicsJS](http://wellcaffeinated.net/PhysicsJS/" \l "demo-1" \t "_blank) experiment. In a real-world exercise, I would have happily halted development and opted to take the code there and simply modified it. However, this wouldn't have been much different than a simple copy/paste job and not a compelling example of my capabilities in an application project.

### Why is this a compelling example of my talents?

This project, due to massive time-constraints from client projects, took me roughly 6 hours total. In that time I was able to develop a cross-browser compliant, touch-friendly, responsive, interactive, fun, fast, double-buffered animated simulation of momentum plus gravitational attraction of n objects in a 2D space.

I think it's particularly useful that this underscores my understanding of the low-level technology behind creating this kind of animation/app. An understanding that is only exponentionally enhanced in real-world usage when employing lots of 3rd party libraries.