CS 116x Winter 2015
Craig S. Kaplan

# Assignment 03: Physical simulation

Due date: Wednesday, 4 February, 12:00pm

In this assignment you will develop a sketch that incorporates a basic simulation of Newton's laws of motions (at least the first and second laws; we won't worry about collisions).

## Question 1: Dynadraw

Paul Haeberli, who brought us the ideas behind the Impressionist sketch in Assignment 02, produced another lovely demo at around the same time: Dynadraw.

Dynadraw is a drawing program based on a simple physical simulation. In a normal drawing program (like Impressionist without randomness), the paintbrush follows the mouse location exactly. Imagine that instead of that, the paintbrush is attached to the mouse location with a *spring*. When you first press down with the mouse, the spring is at its rest length (of zero), and the paint stroke appears exactly at the mouse pointer. As you drag the mouse, the paintbrush doesn't track it exactly. Instead, the spring stretches, and it takes a bit of time for the spring to pull the paintbrush back to the mouse's current location. When it does so, the paintbrush can be moving with enough speed that it will overshoot the mouse and spring outward in the opposite direction, only to return again later. The overall effect is that the paintbrush can seem to "orbit" the mouse. Add in damping, and the orbit will decay until the paintbrush returns to the mouse position. The simulation can be varied through the two standard parameters that control the spring: its stiffness and a damping factor.

If you drag the mouse while this is happening, the paintbrush can trace out graceful, smooth curves. There are many implementations of Dynadraw online. One simple but effective one is a Javascript-powered version by Roger Allen (try it!). That version has fixed values for stiffness and damping; you'll have sliders to control them.

In this question you will create your own implementation of Dynadraw. The basic requirements are as follows:

- The sketch should open a window of dimensions $800 \times 600$, with a white background.
- Pressing any key should clear the screen to white so that the user can begin a new drawing.
- When the user presses the mouse button, begin a spring simulation. Initially, the pen is located at the mouse position and has zero velocity. At every time step, these get updated as discussed in class, except that the spring's displacement is measured relative to the current mouse position, not a fixed point in the sketch. When the user releases the mouse button, the simulation stops and no more drawing happens (until the next button press).
- At every time step, your simulation will use the information about the previous paintbrush location (and velocity, etc.), combined with the current mouse location, to compute a new paint-

brush location. Build a stroke by drawing a line segment from the old paintbrush location to the new one. You don't have to vary the thickness of the segment (but see the bonus below).

- The sketch should include two sliders: one that controls the spring's stiffness and one that controls the damping factor. You must use ControlP5 to create these sliders and process events from them. Put the sliders where you think they work best (presumably somewhere near the edge of the sketch window). Make sure to draw a dark-coloured rectangle underneath the sliders (as in Assignment 02). Choose reasonable initial values for stiffness and damping (I use 0.05 and 0.1, respectively). Make sure the sliders show those initial values when the sketch starts.

- **Bonus:** One way to add a lot of visual appeal to Dynadraw's curves is to use the paintbrush's speed to control the width of the painted stroke. Define a maximum brush width for when the brush isn't moving. As the brush moves faster, draw a progressively thinner stroke, down to some predefined minimum for any speeds beyond a threshold. I'll award a small bonus to implementations that include effective automation of brush width.

- **Bonus:** If you build the algorithm as described, you discover a deficiency in the output. When the stroke curves around too tightly, it looks "polygonal"—that is, it looks like it's made from a sequence of short lines, instead of a smooth curve. For another bonus, figure out an effective way to make the curve look truly smooth, or at least a lot smoother. There's no one way to do this, and it's even harder if you vary the line thickness too.

This problem is made up of several components, but they're all based directly or with small variations on ideas discussed in class and used in labs and assignments. Here are a few specific pointers:

- The ControlP5 aspects of this assignment are nothing new, they're just the usual sliders used in Lab 02 and Assignment 02. If you had trouble implementing them previously, be sure to refer to the sample solutions.

- The spring simulation is very similar to the `DampedSpring` sample sketch. The big difference is that the simulation operates in two dimensions, not just one. But converting from one dimension to two can be done as in the other examples discussed in class (see the transition from `SecondLaw` to `SecondLaw2D`, for example). Another difference is that this spring has a rest length of zero: it wants to return to having no length at all. Finally, the displacement of the spring is computed differently in this sketch, because it depends on the current mouse location.

- Remember to reset the values that control the simulation every time the user presses the mouse button. That means the brush starts at the mouse location, with no initial velocity.

- As with the problems in Assignment 02, it will help to maintain an explicit global variable that tracks whether the user is currently dragging (style hint: what type should the variable have? Not `int`...). And as in the Impressionist example, it will help to check explicitly whether mouse presses happen in the ControlP5 part of the window, so that presses there don't trigger painting events.

**What to submit:** On LEARN, you should submit a single sketch entitled `A03`. Submit the entire sketch folder.