2018

DIY Science Sims

James Hedberg
CUNY City College

DIY Science Sims

James Hedberg
The City College of New York
Dept. of Physics
(Dated: April, 2018)

A new javascript library is presented that enables students and teachers with little coding experience to build interactive science sims. We discuss key parts of the framework and outline the structure of how to build a simulation.

I. INTRODUCTION

Interactive simulations (sims) for science education have been around for decades.[1, 2] Traditionally, the coding skills required to build these sims were well out of reach for beginning students. Efforts have been made to create high level, easy to use frameworks and libraries to built physics related sims.[3] This project focuses on using even more elementary javascript, which is a ubiquitous language driving much of the internet, to code basic to advanced sims that show interesting science without having to spend years learning software development. Students with little to no coding experience can pick up the basics of the language and start building interactive simulations. Furthermore, the skills developed while building a sim are immensely transferrable to other disciplines and are not confined to physics related education or research.

II. INSIDE A SIM

A. p5.js

The basic element we chose to build our sims off of is the HTML5 canvas element, which is simply an element of a standard webpage, as shown in Figure 1. As defined by the W3C standards:

> The canvas element provides scripts with a resolution-dependent bitmap canvas, which can be used for rendering graphs, game graphics, art, or other visual images on the fly. [4]

There are many javascript libraries that can be used to draw and animate canvas elements. We have chosen to build off of the p5.js library, which has as its mission

> ...to make coding accessible for artists, designers, educators, and beginners, and reinterprets this for today's web.[5]

p5.js is based on the Processing language which has been used for many visualizations and artistic projects since its inception.[6] This is in contrast to many previous sims frameworks that relied on standalone applets or embedded Java applications to create interactive animations. By building off the standard web page structure, we have a system that is widely distributable, platform independent, and accessible to anyone with a modern web browser. Our sims require no extra downloads, no security setting adjustments and can be embedded as `iframe` elements in any web page, just like an image or video. This makes deployment and distribution of the sims about as easy as publishing a standard webpage.

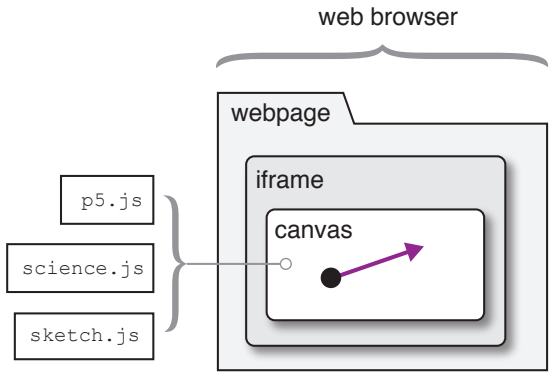

FIG. 1. The schematic layout of a sim as embedded element in a webpage. The sim, where the code is stored in the `sketch.js` file, draws to the canvas using the other libraries shown.

B. What's in a sketch

Every sketch consists of at least two functions, `setup()` and `draw()`. The `setup()` function is called once when the sketch loads, and the `draw()` function is called over and over, at rate of the animation (usually at least 30 frames per second).

```
// basic sketch outline
function setup() {
// things that happen once
}

function draw() {
// things that happen repeatedly
}
```

To make a circle located at the origin (0,0), with a diameter 20 pixels, we can call the `ellipse()` function from within the draw routine.

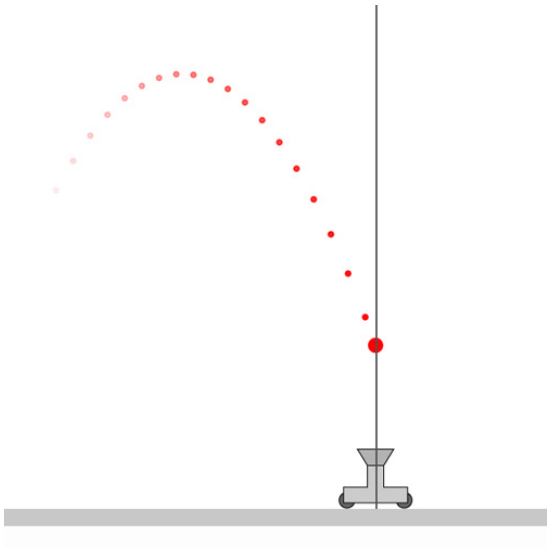

FIG. 2. A simulation of a 2D kinematics cart available on science sims.

```
function draw() {
   ellipse(0,0,20)
}
```

If we wanted to make the circle move, we can simply change the position every frame by using a counter, `dx`:

```
function draw() {
   ellipse(dx,0,20)
   dx = dx+1
}
```

Now, every time the draw routine runs, the x-position of the ellipse will advance by 1 pixel. This is the basis for motion in animations.

If we also want to change the speed of the motion, that is, create an acceleration, then we just need to change the rate at which the counter increases.

```
function draw() {
   ellipse(dx,0,20)
   dx = dx + dv;
   dv = dv + 1
}
```

This circle will now accelerate as it moves across the canvas.

Basic kinematics animations involving velocities and accelerations can be constructed to demonstrate the physics of motion. For example, the classic kinematics cart can be constructed given a mass and a cart the appropriate velocities and accelerations, and one can see how the ball will always remain at the same position as the cart, ensuring it land back in the cap after it is ejected.[7] It's a classic lecture demonstration that is nice to be able to see in slow motion. See Figure 2 for a still frame from the simulation.

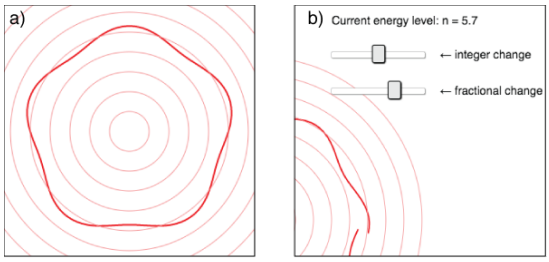

FIG. 3. A simulation of the Bohr - de Brolgie model showing how a standing wave can only take on certain integer values. a) The n = 5 configuration showing a complete standing wave. b) If however, the sliders are adjusted to create a non-integer value, then the standing wave is not continuous.

C. Interactivity

Watching balls move across screens can only be so stimulating. Creating interactivity within sims offers a much more powerful tool for learning physics. We can create a sim to illustrate a certain concept, but then add a slider to control one variable and observe the difference. The standard compliment of interactive mechanisms that users are familiar with from the internet are all available here: sliders, check boxes, radio buttons, etc. All can be 'wired up' to control a given parameter in a sim. For example, the Bohr - de Broglie model of the hydrogen atom energy levels, with the relation between wavelength and radius set by an integer counter n:

$$n\lambda = 2\pi r, \tag{1}$$

relies on the student being able to visualize what happens when the orbital is changed by integer values. It's harder to visualize what happens when the n value takes on non-integer values - this makes for a compelling sim. See Figure 3.

III. THE CATALOG

A. Overview

The catalog currently consists of several dozen sims, organized into thematic structures. We offer sims related to Math, Astronomy, Physics. Physics is the largest category and is broken down into smaller sub categories such as kinematics, waves, or modern physics.

B. Sim Design

Previous physics sims by other groups have attempted to recreate entire physics labs in a virtual environment, allowing for students to manipulate many aspects of a physical phenomenon and see the results.[2] Our sims aim for a much simpler target. The sims described herein are

meant to be visualizations of a single concept or principle, not a full quasi-realistic simulation of reality. If there are parameters to be adjusted, then we try to minimize the number of such parameters. The designs are minimal, with no distracting skeuomorphic elements.

C. User Submissions

A major goal of the DIY Science Sims project is to enable students and teachers to contribute sims to the public facing catalog. We employ the web service and version control of https://github.com to allow anyone to submit pull requests if they have built a sim they would like to be shared with the community. The website http://sciencesims.com is a curated collection of these sims.

IV. SCIENCE.JS

Since many sims will use similar components (i.e. a mass under the influence of gravity) we have built a library to extend p5.js called `science.js` .[8] This library has several objects that can be used to build sims. We'll examine a few:

A. Vectors

Students often struggle with the concepts associated with vectors. They can handle the algebra just fine, but when it comes to basic conceptual questions, their understanding is often lacking. With `science.js` we can attach vectors to objects and the vectors will respond to their motions.

A vector with arrowhead can be created with one line by calling:

```
aVector = new Arrow(startPoint, endPoint);
```

The startpoint and endpoint are p5.Vector objects. The arrows can be stylized and can also respond to user inputs such as dragging or changing angles.

B. KineticMass

Introductory physics is replete with balls experiencing various type of motion. There is of course constant velocity linear motion. Some accelerate due to constant forces such as gravity near the surface of the Earth. Some experience more complicated sinusoidal motions during simple harmonic motion. We have a simple point-like object that responds to forces and can be used to create 2D kinematic sims for many of the common situations encountered in introductory kinematics. Simply create a `KinematicMass` object and provide the kinematic variables that describe it, position, velocity and acceleration.

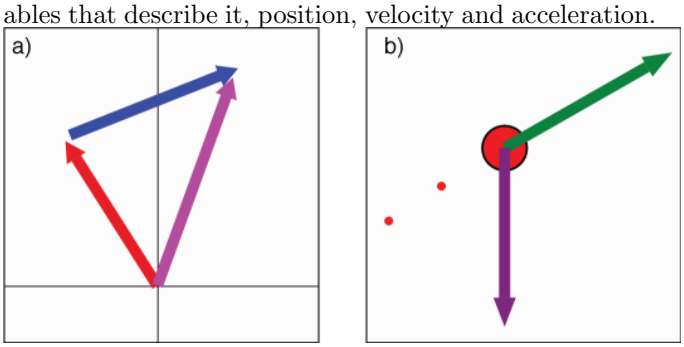

FIG. 4. a) Vector addition sim. Users can drag the red and blue vectors and the purple arrow updates to portray the vector sum. b) A `KineticMass()` object with velocity and acceleration vectors added.

```
ball = new KineticMass(pos,vel,accel,mass,color);
```

The `pos`, `vel`, and `accel` references are all p5.Vector objects that dictate the behaviour of the mass.

C. ...and many more

Many more reusable objects have been created and more are in development. For example: a simple spring, a pendulum, a Free Body Diagram generator, and a rotating wheel are all objects currently available in the `science.js` library.

V. CONCLUSION

It should be mentioned again that interactive physics simulations have been around for quite some time in various incarnations. The current project aims to offer a novel approach by lowering the bar to development even further. We want to let students build sims and by doing so, learn more physics and math as well as learn the basics of coding. Furthermore, our goals are not to create full simulations of lab experiments, in the vein of the PHET group, but rather to create small, single topic and single concept visualizations, where perhaps one or two variables might be adjustable. We anticipate expanding this work to include other disciplines besides physics. Already, we have many sims related to math and astronomy. Hopefully, chemistry and biology educators will be able to use the framework as well.

[1] John Richards, William Barowy, and Dov Levin. Computer simulations in the science classroom. *Journal of Science Education and Technology*, 1(1):67–79, Mar 1992.

[2] Katherine Perkins, Wendy Adams, Michael Dubson, Noah Finkelstein, Sam Reid, Carl Wieman, and Ron LeMaster. Phet: Interactive simulations for teaching and learning physics. *The Physics Teacher*, 44(1):18–23, 2006.

[3] F. G. Clemente. Deployment of physics simulation apps using easy javascript simulations. In *2017 IEEE Global Engineering Education Conference (EDUCON)*, pages 1093–1096, April 2017.

[4] w3. HTML5 Canvas Tag. https://www.w3.org/TR/html5/scripting-1.html#the-canvas-element. Accessed: 2018-03-23.

[5] p5.js. p5js.org. https://p5js.org/. Accessed: 2018-03-23.

[6] Processing Foundation. processing.org. https://processing.org/. Accessed: 2018-03-23.

[7] J. Hedberg. The Kinematics Cart Sim. http://sciencesims.com/sims/kinematic-cart-flat/. Accessed: 2018-04-02.

[8] DIY Science Sims. science.js. http://sciencesims.com/sciencejs/. Accessed: 2018-03-23.