Bates College

Senior Thesis

Physics Simulations Using Javascript

Author:
Peter Krieg

Advisor:
Gene Clough

Presented to The Department of Physics, Bates College
In Partial Fulfillment of the Requirements for the
Degree of Bachelor of Science

Lewiston, Maine
December 1st, 2014

Declaration of Authorship

I, Peter Krieg, declare that this thesis titled, 'Physics Simulations Using Javascript' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this College.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

BATES COLLEGE

Abstract

Faculty Name
Department of Physics

Bachelor of Sciences

Physics Simulations Using Javascript

by Peter Krieg

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too Lorem ipsum dolor sit amet, consectetur adipisicing elit. Asperiores obcaecati provident aliquam, impedit quidem iusto voluptatibus temporibus nihil error id est! Adipisci atque aut nostrum, recusandae, quae magnam repellat est.Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sint distinctio autem, officia facilis molestiae veniam beatae, a deserunt laborum, optio earum laudantium atque aliquam, rerum incidunt hic repellat similique ratione?Lorem ipsum dolor sit amet, consectetur adipisicing elit. Incidunt atque amet, officiis consectetur adipisci debitis vero suscipit in doloribus, pariatur ipsa, accusantium aspernatur ab minima fugiat consequuntur distinctio, voluptates! Corrupti.Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sunt deleniti eos quibusdam enim excepturi. Ea, aspernatur sed excepturi beatae, quod consequatur nisi, debitis blanditiis tempore vero vitae asperiores necessitatibus iure?

Acknowledgements

First and foremost, I would like to thank my advisor, Gene Clough. Gene, thank you for your guidance along the way: you helped formulate my ideas and encourage me along the process. I appreciate your patience and dedication to working with me–I have always enjoyed discussing problems with you. I always feel relieved walking into your office knowing that you will have some book pertaining any problem I'm having. It was only fitting that I had you for FYS advisor, and finished with thesis. I've really enjoyed talking with you these past four years.

I would also like to thank Professor Hong Lin, who met with me multiple different times to discuss my thesis. Thank you professor Lin for patiently explaining concepts with me and for letting me borrow some of your books. I appreciate your willingness to talk during an especially busy semester of teaching for you! Additionally, thank you to Professor Mark Semon, for showing me past examples of theses, and outlining the formatting requirements for Bates.

Thank you to my parents for encouraging me thorughout the whole process, and for at least trying to understand what I was doing for my thesis. Lastly, I would like to thank my friends and everyone else who was supportive of my thesis.

Contents

Abbreviations

HTML	HyperText Markup Language
JS	JavaScript
API	Application Programming Interface

Dedicated to my parents

Introduction

0.1 What is a Physics Simulation?

The purpose of this thesis is to present a series of physics simulations, each modeling a specific problem of physics as realistically as possible. These simulations differ from *animations*, which can be seen as predictable representations that always display the same visual. Animations are analagous to a movie script: no matter how many times you watch the movie, it will always end in the same way. Simulations, on the other hand, need to adapt to variable conditions, and be based partly on random processes. This brings up the topic of *dynamic* vs. *static* animation. Most of the physics simulations in this thesis will be dyamic because they present a unique viewing each time they are run, and can also involve user feedback which influences the outcome of the simulation.

Any simulation requires creating the illusion of motion. Almost every form of projected motion media uses frames to accomplish this. Researchers have shown that to make the simulation look realistic, it must be presented at a rate of around 60-100 frames per second. Anything far slower than this, and the human eye will detect the "choppiness" of the simulation. People can't detect anything much faster than 100 frames per second, so there is no need to project media faster than that, with the exception of slow-motion videography.

0.2 Methods of Producing a Simulation

The physics simulations in this thesis differ greatly from common animations. Movies and cartoons, for example, operate by displaying a series of images similar to one another, and displaying them as many frames per second to create the illusion of motion. My simulations, on the other hand, function by providing the *information* for each frame, and then providing the data for *how* the animation can be created. These instructions are passed onto the HTML5 canvas API, which creates the visual which can be seen in the web browser. Because physics simulations contain instructions instead of a series of images, the files of code take up far less space than a movie file would, for example. This is one primary advantage of coded simulations. Every simulation follows a similar set of steps, which can be simplified below:

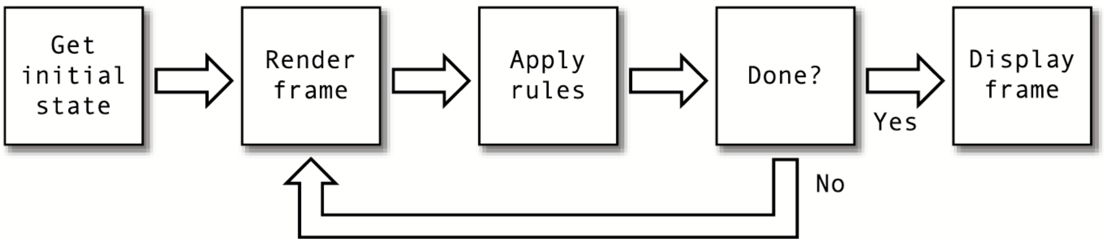

FIGURE 1: The frames of a general simulation

The canvas API gets the initial state of the simulation, which could the position of a ball, for example. Then, the frame is *rendered* by applying rules to the canvas element, and changing the initial state of the simulation. Once the rules have been applied, and all conditions are satisfied, the frame is rendered, and then displayed on the canvas element, to be seen in the web browser. The canvas is embedded into a web page with the ¡canvas¿ tag, like any other HTML tag. The positioning of objects in the canvas element is specified with a coordinate system, which uses pixels as its unit. Figure 2 shows the orientation of the canvas, which differs from the traditional cartesian coordinate system.

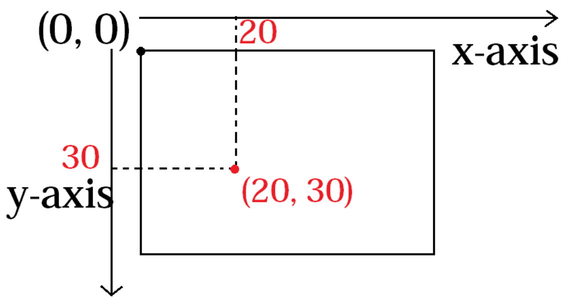

FIGURE 2: The canvas coordinate system. The canvas is displayed on the screen as an invisible white rectangle by default. A sample point of (20,30) is shown for clarification.

To produce any realistic simulation, the steps in figure 1 must be repeated multiple times per second. In fact, these steps must be repeated 60 times per second to achieve the desired 60 frames per second outlined in the previous section. Luckily, the canvas API is capable of running the instructions very quickly to make this simulation possible.

0.2.1 The Code

To program the simulations in this thesis, I chose to write the code in javascript. This scripting language is easy to view in any modern browser: therefore, all the simulations of this thesis can be viewed online. Javascript combines seamlessly with HTML5, which is why I mostly decided to use it for this thesis. The evolution of HTML (HyperText Markup Language) has progressed from simple web documents to complex web applications. For this thesis, every simulation utilizes the HTML5 <canvas>element, which has been used since around 2011. The HTML5 canvas API allows programmers to write javascript code that accesses the element and runs visual displays through a web browser The HTML needed to include a canvas can seen below:

```
<!doctype html>
<html>
 <body>
  <canvas id="canvas" width="500" height="500" >
 </body>
 <script>
  var canvas = document.getElementById('canvas');
  var context = canvas.getContext('2d');
 </script>
```

```
10 </html>
```

Listing 1: The bare bones code necessary for an HTML document to include the canvas element

The above code displays the most basic HTML combined with javascript necessary to begin any simulation. Lines 7-8 are the only ones that actually contain javascript: this is the simple step necessary for the canvas API to recognize the HTML document. These two steps are necessary for any physics simulation. The step on line 7 initializes a JS variable and sets it equal to the canvas element on the web document object. The second step

All web browsers include some form of javascript interpreter: whenever the browser encounters a <script >element, it "passes" the code onto the JS interpreter. In listing 1, the HTML and JS code are written in the same document for clarity. While this is an acceptable practice, all future simulations will involve the HTML referencing to external JS documents to keep the contents separate. The appendices in this thesis have full code files from all the simulations.

While this thesis can contain code excerpts, figures, and screen-shots of various simulations, it obviously can't contain the flow of images itself. Therefore, I have put the entire thesis and its simulations on my personal website, which can be found at: http://www.peterkrieg.com/thesis. You can navigate by each chapter and view the simulations outlined in thesis.

Bibliography

[1] ”Air Friction.” Air Friction: Viscous Resistance. Georgia State University. Web. 2 Nov. 2014. ¡http://hyperphysics.phy-astr.gsu.edu/hbase/airfri.html¿.

[2] Flanagan, David. JavaScript: The Definitive Guide. 6th ed. Beijing: O’Reilly, 2011. Print.

[3] Hawkes, Rob. Foundation HTML5 Canvas. New York: Friends of ED :, 2011. Print. Lamberta, Billy, and Keith Peters. Foundation HTML5 Animation with Javascript. New York: Friends of Ed :, 2011. Print.

[4] Kleppner, Daniel, and Robert J. Kolenkow. An Introduction to Mechanics. New York: McGraw-Hill, 1973. Print.

[5] Ramtal, Dev, and Adrian Dobre. Physics for JavaScript Games, Animation, and Simulations: With HTML5 Canvas. Apress. Print.

[6] Rauschmayer, Axel. Speaking JavaScript. Cambridge: O’Reilly, 2011. Print.

[7] Roux, A., & Dickerson, J.(2007). ISB Journal of Physics *Coefficient of Restitution of a Tennis Ball.* Retrieved November 10, 2014, from http://www.isb.ac.th/HS/JoP/vol1/Papers/Tennis.

[8] Serway, Raymond A., and John W. Jewett. Physics For Scientists and Engineers with Modern Physics. 9th ed. Boston, MA: Brooks/Cole, Cengage Learning, 2014. Print.

[9] Young, Hugh D., and Roger A. Freedman. University Physic with Modern Physics. 13th ed. Pearson, 2011. Print.