Oregon State University
College of Engineering

CS Capstone Final Report

June 11, 2019

Interactive 2D simulations to support Inquiry-Based Learning in Mechanical Engineering

Prepared for

Oregon State University, School of Chemical, Biological, and Environmental Engineering

Milo Koretsky
Tom Ekstedt

Prepared by

Group53
Education Simulations

Cameron Friel
Kelli Ann Ulep
Samuel Wilson

Abstract

This project involves solving the problem of some universities lacking resources to visually show physical and mechanical interactions for Mechanical Engineering concepts. This project is built on the research that shows that students can achieve a better understanding of difficult concepts by learning through interactive simulated environments. By implementing 2D simulations based on these concepts, students will be able to visually interpret the concepts in the course. The goal of this document is to document the design for these 2D simulations, explaining the implementations of the features which shall be expected for this project.

Contents

1 Project Introduction

1.1 Context

The clients for this project are Milo Koretsky and Tom Esktedt. The client runs *Concept Warehouse*, an educational tool which focuses on creating interactive learning opportunities for classrooms. In order to see how well this interactive learning works, exercises first ask students to answer abstract questions relating to the subject based on prior knowledge or preconceptions. Then, students engage with online simulations to naturally learn through experimentation. Afterwards, students answer the same questions as before, hopefully with obvious improvements. The data collected from these before and after quizzes have shown that this type of learning works very well and is efficient, considering the time required for students or instructors to set-up real-world labs. Subjects, such as chemistry, have already been implemented into the Concept Warehouse website; however, mechanical physics has not.

1.2 Importance

Mechanical Engineering students should have a solid understanding of key mechanical or physical concepts in their study. Students should be engaged in class in a way that effectively communicates challenging ideas. Students come in with pre-existing knowledge or preconceptions, engage with learning material, then they ask themselves or explain what they witnessed. With the idea of *cognitive conflict*, when students observe ideas that challenge or contradict their prior ideas or beliefs, they then are pushed to think more about it. Especially when concepts are non-intuitive, they may struggle with understanding. Typically, instructors can illustrate information with educational tools such as visuals or hands-on experiments. Classes may conduct simulations or experiments like dropping a ball at certain heights to understand gravity or rolling a cylinder down an inclined surface to understand motion. However, some phenomena are not as accessible or convenient to interact with physically.

Equipment used to observe some concepts could be inaccessible, or the physical motions are difficult to see in person. Students may want to be able to stop an simulation at a certain frame of time to observe or plot data points, but then the experiment could be progressing too quickly to show. Even if some equipment may be accessible, it still could prove inefficient to conduct an entire experiment that covers all points of curiosity around certain phenomena. Students may have difficulty explaining or envisioning how these calculations apply to real situations. With the example of the experiment to roll a cylinder down a ramp to observe motion, the students may be interested in seeing how the system is affected with a 1000 ft tall ramp with a steep angle and a 2000 lb cylinder. To do some calculations on a whiteboard and then imagine the scenario is not enough for some. While some of these outcomes may be impossible in the physical world, students should have the option to change values and see for themselves how it applies in some form.

1.3 Team Members

The members of this team are Cameron Friel, Samuel Wilson, and Kelli Ann Ulep. The main components of the project are

1) Physics Simulation
2) Graphing/Measurement Updates
3) User Interface

All members have contributed in some way to each of these sections. Responsibilities at the beginning of the term were given as Cameron Friel to the physics simulation, Sam Wilson with the Graphing/Measurements, and Kelli Ulep with the User Interface.

1.4 Client Role

The clients were Milo Koretsky, a Chemical Engineering professor at Oregon State University and Tom Ekstedt, a programmer analyst at Oregon State University. Milo's role in the project was to observe and approve of design decisions through the iterations of the project from a non technical side. Tom's role in the project was to guide the project in the right direction by providing input and giving the project team the requirements for the project. Tom also hooked up this project's simulations up to the Concept Warehouse to be used in the classroom.

:

Oregon State University
College of Engineering

CS CAPSTONE REQUIREMENTS DOCUMENT

APRIL 19, 2019

INTERACTIVE 2D SIMULATIONS TO SUPPORT INQUIRY-BASED LEARNING IN MECHANICAL ENGINEERING

PREPARED FOR

OREGON STATE UNIVERSITY, SCHOOL OF CHEMICAL, BIOLOGICAL, AND ENVIRONMENTAL ENGINEERING

TOM EKSTEDT Tom Ekstedt 19 April 2019
Signature *Date*

PREPARED BY

GROUP53
EDUCATION SIMULATIONS

CAMERON FRIEL _____ _____
Signature *Date*

KELLI ANN ULEP _____ _____
Signature *Date*

SAMUEL WILSON _____ _____
Signature *Date*

Abstract

This project involves solving the problem of some universities lacking resources to visually show physical and mechanical interactions for Mechanical Engineering concepts in the classroom. This project is built on the research that shows that students can achieve a better understanding of difficult concepts by learning through simulated environments that they can interact with. By implementing 2D simulations based on these concepts, students will be able to visually interpret the concepts in the course. The goal of this document is to set the requirements for these 2D simulations, listing all the features which shall be expected for this project.

CONTENTS

1 REVISION TABLE

TABLE 1
Revisions

Section	Original	New
3.3	Limitations: A high quality camera might also be needed to record a live demonstration of the simulations to be embedded within the web page	This part is removed as we decided that we do not need a live simulation along with the virtual one
4.2	Next to the animation shall be graphs or numbers that display the position where the weight started and where the weight ended up on the other side. This will be shown in as the angle and the resulting height.	This will be shown in as resulting height for Cases 1-5 and as the velocity and height for exploratory mode.

2 INTRODUCTION

2.1 Purpose

The purpose of this document is to present the requirements for the Interactive 2D simulations to support Inquiry-Based Learning in Mechanical Engineering project, as well as expand on the intent behind the project. This document is intended to be read by the clients of the project, as well as the projects team members in order to bind the two parties to a contract regarding the requirements of the project.

2.2 Scope

Our product will consist of six variations of a two dimensional simulation based on mechanical engineering concepts in order to support inquiry based learning for mechanical engineering students. The simulations will be hosted on the Concept Warehouse website, which is a website that provides interactive learning opportunities for classrooms. While the simulation is running, real time mathematical feedback will be provided in the form of graphs or simply values in the simulation. There will be start/stop, and reset button within each simulation so that the student can explore the simulations as many times or at as many points as they need to. The students will also be able to modify specific parameters within the simulations in the sixth variation of the simulation, such as mass, friction, and angles.

2.3 Product overview

This document contains two more sections. The second section provides an overarching description of the product. The third section contains the specific requirements for the project.

3 OVERALL DESCRIPTION

3.1 Product perspective

Our product is part of a larger system, that being the Concept Warehouse, so students will need to click on a hyperlink within the Concept Warehouse website in order to navigate to our product. Each simulation will be powered by a 2D physics engine on a standalone web page. Each web page will have a canvas window to the left hand portion of the screen where the simulation will occur. Below the canvas will live buttons to start, stop, and restart the simulation. On the right side of the canvas will be parameter fields a student could interact with to change the way the simulation interacts.

Above the parameter fields will be live mathematical feedback showing the student how the different parameters are changing during the simulation.

3.2 User characteristics

Our product is intended to be used by mechanical engineering students whose college is subscribed to the Concept Warehouse.

3.3 Limitations

Our product will need to be able to run on all modern web browsers, such as Chrome, Safari, and Firefox.

4 Specific Requirements

4.1 System interface

- The user shall only need a computer, working internet connection, and a URL to run the simulation.
- The simulation shall appear as a standalone web page.

4.2 Functional requirements

The project shall produce at least one inquiry-based learning activity (IBLA) simulation. The first simulation shall focus on a pendulum system: an Impact Pendulum. The following requirements are based off this pendulum physics system.

- The animation shall operate based on simulated physics.
- There shall be a button to trigger the simulation to start.
- When a student triggers the pendulum to fall, the weight shall go to the opposite side and the animation will freeze at its maximum height.
- Next to the animation shall be graphs or numbers that display the position where the weight started and where the weight ended up on the other side. This will be shown in as resulting height for Cases 1-5 and as the velocity and height for exploratory mode.

4.3 User modes

- The simulation shall have a normal mode. The inputs into the pendulum system (weight's mass, starting height/angle, and coefficient of restitution) are pre-set. The 5 variations of the pendulum are as follows:
    1) 1.5 lb weight held at 60 degrees and released
    2) 1.5 lb weight B held at 60 deg and the 1.5 lb weight A at rest at 0 deg. The weights stick together at collision
    3) 3 lb weight B held at 60 deg and the 1.5 lb weight A at rest at 0 deg. Weight B is released. The weights stick together at collision.
    4) 1.5 lb weight B held at 30 deg and the 1.5 lb weight A held at -30 deg. The weights are released. The weights stick together at collision
    5) 1.5 lb weight B held at 60 deg and the 1.5 lb weight A at rest at 0 deg. Weight B is released. The weights do not stick together at collision.
- The simulation shall have an exploratory mode. The user can adjust the inputs that affect the pendulum system: number of weights, mass of weights, starting angle, coefficient of restitution, and length of the pendulum arm.

4.4 Other simulations

4.4.1 Spool Simulation

This second simulation depicts a spool being pulled along a surface in various cases. Sections 3.2 and 3.3 will be modified to address a Spool simulation. This is a stretch goal of the project to happen after the pendulum simulation. This simulation follows these 4 cases:

- String pulled up
- String pulled down
- String pulled to the right
- String pulled to the right below the surface

These animations will show the direction in which the spool will move.

4.5 Software Attributes

4.5.1 Availability

The web page shall be supported on major modern web browsers: Google Chrome, Mozilla Firefox, and Safari.

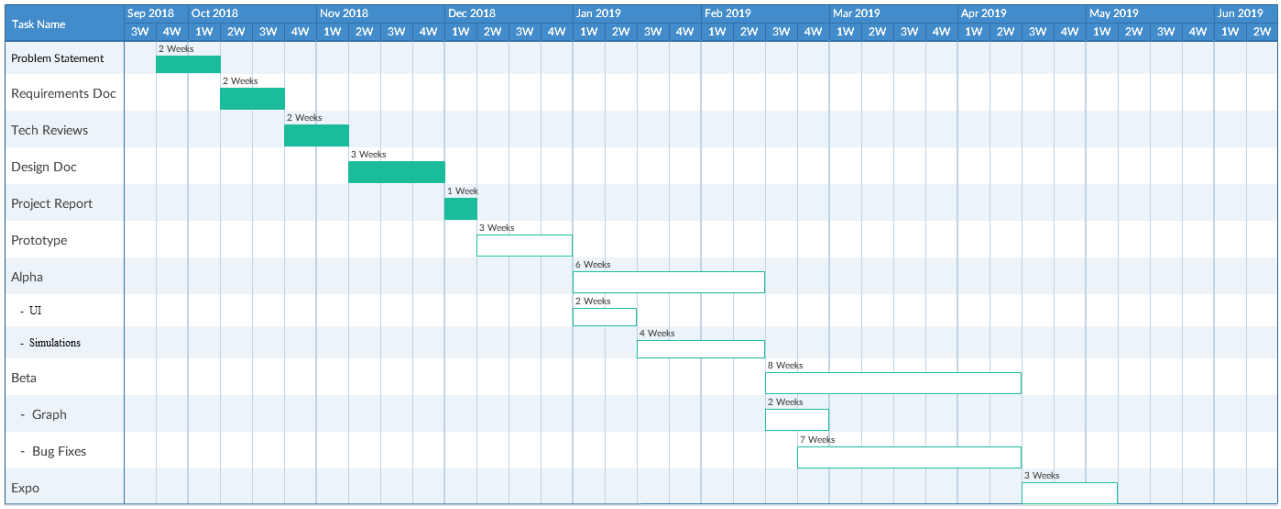
4.5.2 Maintainability

The software shall utilize as many existing technologies (i.e. JavaScript libraries) as possible. These technologies must be open source - that the original source code is freely available and can be redistributed and modified. The technologies must be able to be used in the context of Concept Warehouse without any charges or usage restrictions.

4.5.3 Portability

The framework shall be reusable and modular such that other IBLA simulations or their variants can reuse code.

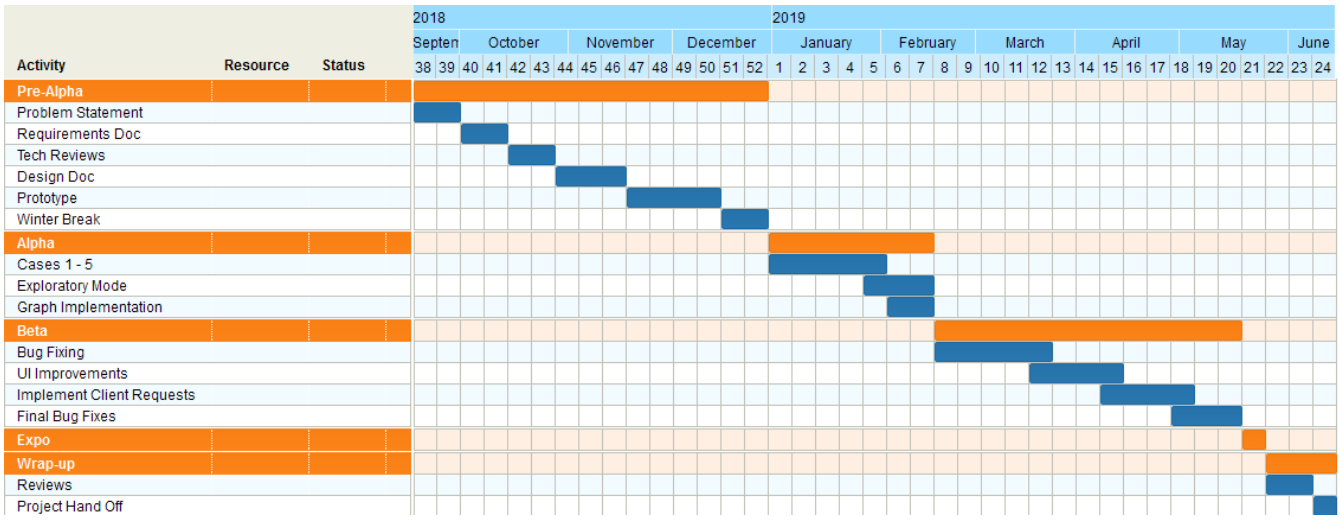
5 GANTT CHART



6 GLOSSARY

1) Inquiry-based learning - A form of learning that starts by posing questions rather than providing facts.
2) IBLA - Inquiry based learning activity
3) Parameter - a value is selected for the particular circumstances that affects the system.

2.5 New Gantt Chart



:

Oregon State University
College of Engineering

CS CAPSTONE DESIGN DOCUMENT

APRIL 19, 2019

INTERACTIVE 2D SIMULATIONS TO SUPPORT INQUIRY-BASED LEARNING IN MECHANICAL ENGINEERING

PREPARED FOR

OREGON STATE UNIVERSITY, SCHOOL OF CHEMICAL, BIOLOGICAL, AND ENVIRONMENTAL ENGINEERING

TOM EKSTEDT Tom Ekstedt *Signature* 19 April 2019 *Date*

PREPARED BY

GROUP53

EDUCATION SIMULATIONS

CAMERON FRIEL _____ *Signature* _____ *Date*

KELLI ANN ULEP _____ *Signature* _____ *Date*

SAMUEL WILSON _____ *Signature* _____ *Date*

Abstract

This project involves solving the problem of some universities lacking resources to visually show physical and mechanical interactions for Mechanical Engineering concepts. This project is built on the research that shows that students can achieve a better understanding of difficult concepts by learning through interactive simulated environments. By implementing 2D simulations based on these concepts, students will be able to visually interpret the concepts in the course. The goal of this document is to document the design for these 2D simulations, explaining the implementations of the features which shall be expected for this project.

CONTENTS

1 REVISION TABLE

TABLE 1
Revisions

Section	Original	New
4.3	Described the game engine, Defold	Describes the physics engine, Matter.js
4.4.3	• Exploratory mode description using Defold's input system and collision object feature • Use Bootstrap to design the webpage	• Used matter.js to control all the inputs into the pendulum system • Used the library noUiSlider to receive input • Used HTML and CSS to design the webpage of each simulation
4.6	Art Software	This section is removed since we are using the built in entities provided by the Matter.js library.

2 OVERVIEW

This document contains an in depth description of the designs to be used in the 2D Simulations for Inquiry-Based Learning project. This document will go over the UI design layouts needed for the simulation web page, the implementation of web page hosting, and the implementation of the animation. The components within each of these sections are also explained.

2.1 Scope

Our product will consist of six variations of a two dimensional simulation based on mechanical engineering concepts in order to support inquiry based learning for mechanical engineering students. The simulations will be hosted on the Concept Warehouse website, which is a website that provides interactive learning opportunities for classrooms. While the simulation is running, real time mathematical feedback will be provided in the form of graphs or simply values in the simulation. There will be start, stop, and reset button within each simulation so that the student can explore the simulations as many times or at as many points as they need to. The students will also be able to modify specific parameters within the sixth variation of the simulation, such as mass, friction, and angles.

2.2 Purpose

The purpose of this document is to present the design specifications for the Interactive 2D Simulations to support Inquiry-Based Learning in Mechanical Engineering project. This document will expand on the technologies used to implement the project, which will include:

- The process of serving the simulations to the student
- The Physics Engine used to procure the simulations
- User Interface designs for the web page
- Data Visualization
- Software for creating the simulation assets

2.3 Intended Audience

The intended audience for this document is primarily for developers that will be working on the project and need to be on boarded. In addition to this, the document is used as verification to the client's of the project, Milo Koretsky and Tom Ekstedt, to make sure that the design falls inline with the expectations they have provided.

3 GLOSSARY

- Concept Warehouse - A platform for engineering students to answer questions inside or outside of class.
- User Interface (UI) - The aspect of a program that the user interacts with directly through keyboard and mouse.
- Simulation - A digital and visual representation of a physics model.
- Object - A particular instance of a class, where the object can be a combination of variables, functions, and data structures.
- JavaScript - A core scripting language used in the web. It is used to create interactive elements within a web page.
- Chart.js - A lightweight JavaScript library for creating interactive graphs and that is free to use for the web.
- Open-source - Software code that has been made available for public use and manipulation.
- Matter.js - A free, 2D physics engine, based on Box2D; a C++ physics engine.
- Coefficient of Restitution - ratio of the final to initial relative velocity between two objects after collision. Values range from 0 to 1
- Mockup - A prototype providing the design and functionality of a system

3.1 Design Timeline

- Jan 7 - Jan 20: UI creation
- Jan 20 - Feb 20: Simulation creation
- Feb 20 - Mar 22: Graph Add-on and Bug Fixes
- Mar 22 - April 10: Clean Up and Stretch Goals
- April 10 - Expo: Expo Preparation

4 DESIGN VIEWPOINTS

4.1 Introduction

The different components of the web page is the hosting information, physics engine, user interface, graphs, and art software. The following section explains each of the components and the implementation design.

4.2 Hosting

The project will consist of several files that will be delivered to the client. The client will then upload these files to their own server to be distributed to students. The end product will simply be a simulation that can be displayed to students. The simulation consists of 5 pendulum variation cases and an exploratory mode. The flow of the system is that the instructor dictates when students are able to view the simulation through Concept Warehouse. Then the server updates the content of the web page with a new case. The cases are shown in sequential order from 1 to 5. Exploratory mode, where users can modify parameters, is also unlocked by the instructor.

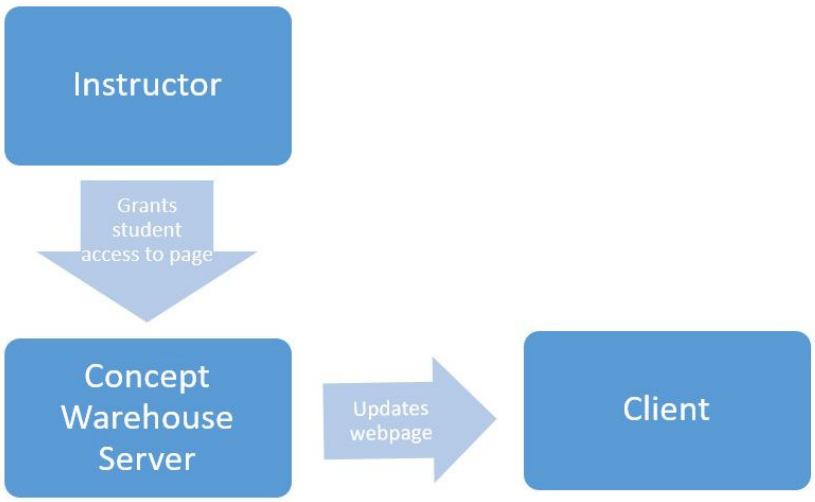


Fig. 1. Hosting Diagram

4.3 Viewpoint: Physics Engine

4.3.1 Design Concerns

Creating the physics needed to accurately portray pendulums acting upon each other would take more time and knowledge than what we currently have. Using an open source physics engine will allow us to create the simulations very quickly.

4.3.2 Design Elements

Physics engines are pre-made frameworks that allow for fast creation of physics objects, which respond to each other in a realistic way. Primarily, these objects are referred to as ”bodies,” and they hold specific properties, such as position, velocity, mass, and friction constants. The physics engine we chose is Matter.js, which is based off of Box2D, a well known, C++ physics engine. Matter.js is open source, which means we do not need to worry about a paid license to use the engine.

4.3.3 Function Attribute

The physics engine, Matter.js, will allow the team to quickly create multiple simulations of similar types. The ability to visually create physics objects with a few lines of code will greatly increase our productivity.

4.4 Viewpoint: User Interface

The User Interface is the gateway to connecting with the end user. It allows for interactivity with the simulation.

4.4.1 Design Concerns

A disorganized UI or a UI that is hard to understand can impede the student's learning. Even if the animation runs well in itself, if the web page it is placed in is hard to manipulate, it can take away from the animation.

4.4.2 Function Attribute: Normal Mode

The user interface mockup of Case 1 is shown in Figure 2 and Figure 3. Figure 2 will be the initial screen shown. This will be the normal mode of case 1. Normal mode is applicable to all 5 of the variations of the impact pendulum scenario. The normal mode components with their description are as such:

- Start/Pause Button - Starts the simulation or pauses it
- Reset Button - Resets the simulation
- Graph of angle vs. time
- Live measurement updates - Updates the angle and height of each weight, velocity, and the time every 0.1 seconds of the simulation
- Starting measurements - Length (m), weight (g), angle (degrees), height of weight off the ground (m), and coefficient of restitution

The graph component design is explained in section 3.5.

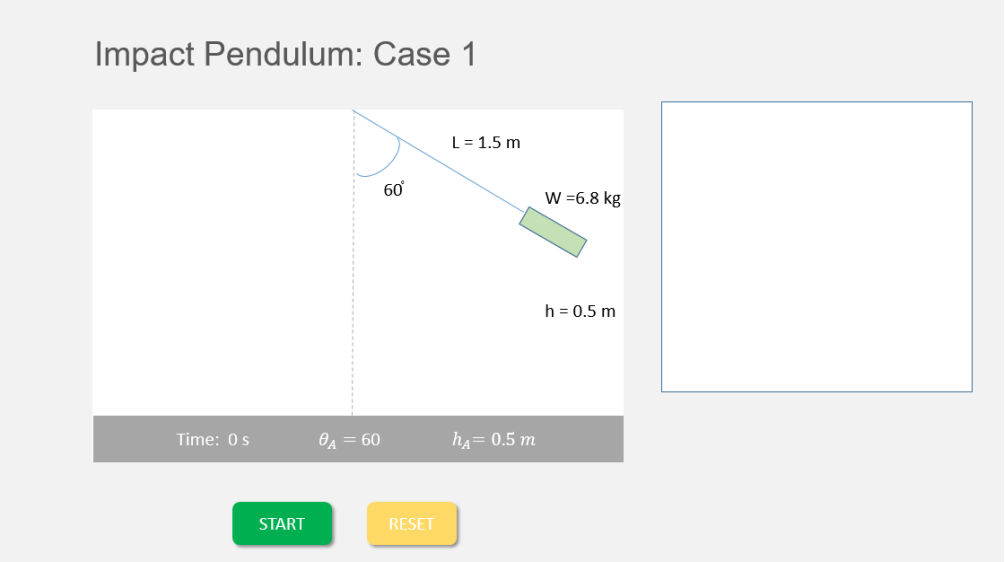


Fig. 2. Pendulum Case 1 - Before animation UI

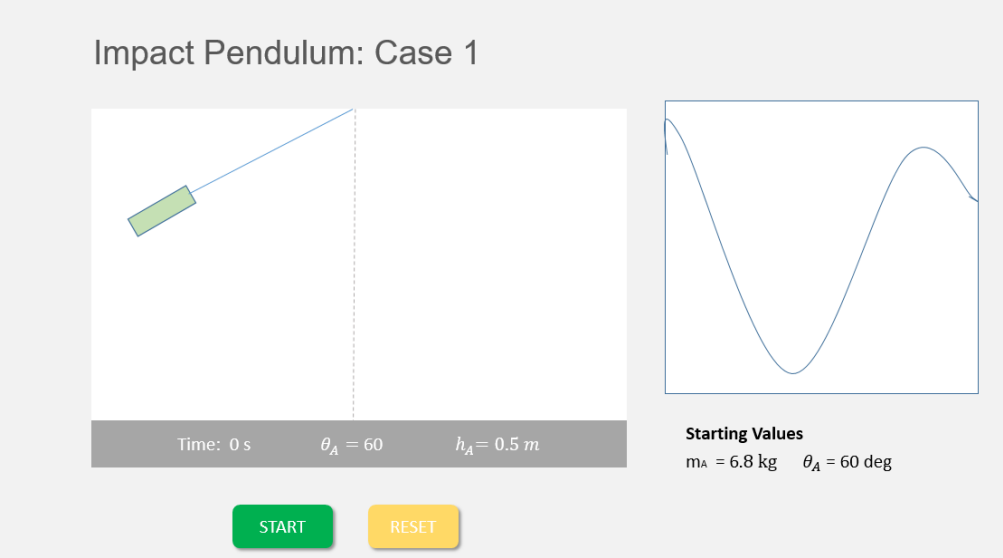


Fig. 3. Pendulum Case 1 - After animation completion UI

4.4.3 Function Attribute: Exploratory Mode

The mockup of the Exploratory mode where the user can modify parameters is shown in Figure 4 and Figure 5. The exploratory mode components are

- Start/Pause Button - Starts the simulation or pauses it
- Reset Button - Resets the simulation
- Graph of angle vs. time
- Live measurement updates - Updates the angle and height of each weight, velocity, and the time every 0.1 seconds of the simulation
- Option to choose between 1 weight or 2 weights
- Starting parameter input sliders for the angle (degrees), length (m), weight (g), and coefficient of restitution.

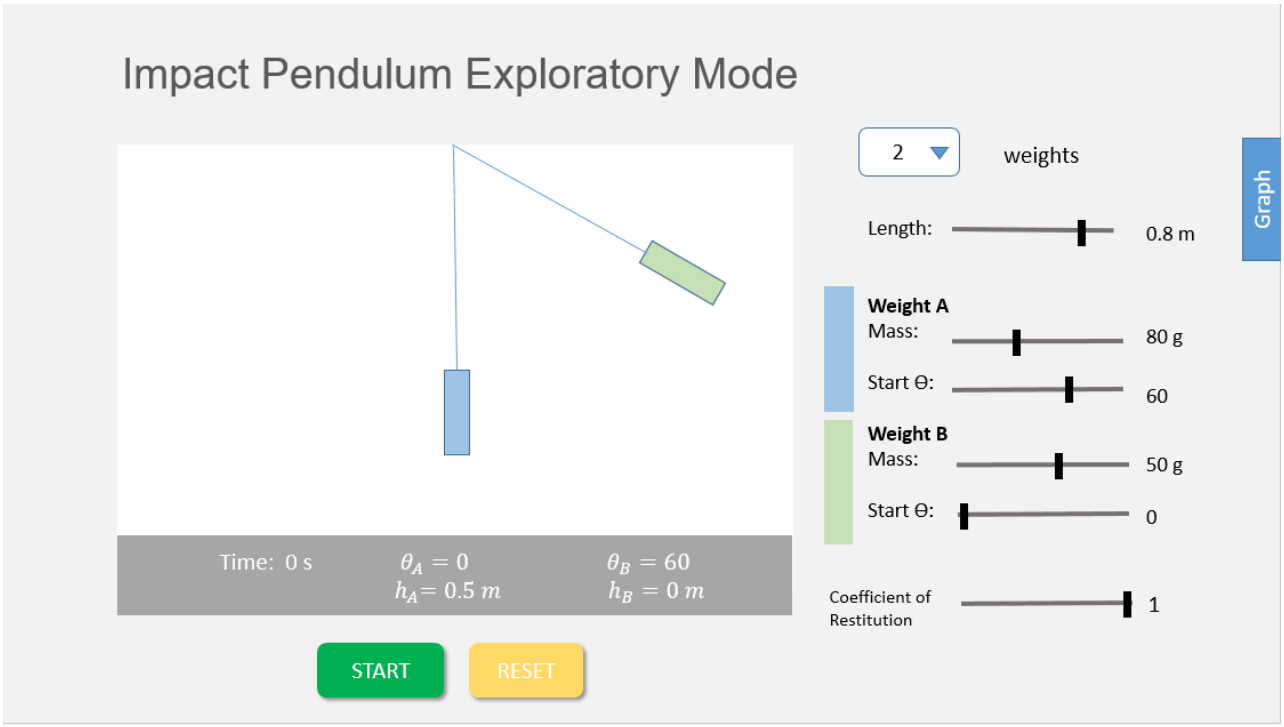


Fig. 4. Exploratory Mode Start Screen

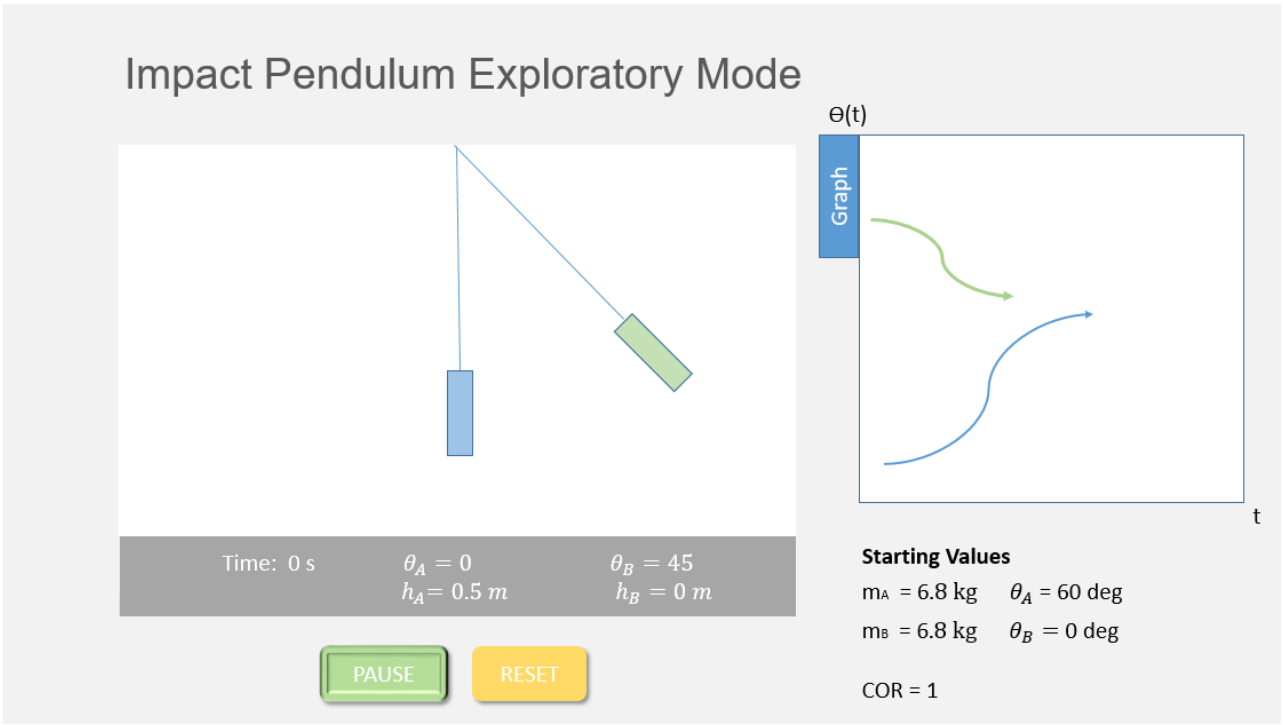


Fig. 5. Exploratory Mode Mid Animation

The webpages uses HTML/CSS and Javascript to construct the start/pause/reset buttons, live measurement update tables, menu options, and the input sliders. Everything but the input sliders are constructed by HTML features: *button*, *select*, *option*. The input sliders are constructed by the JavaScript library noUiSlider. This library provides callback functions, called whenever the users changes the slider. Whenever the user changes the slider button, the simulation world is updated to match the current values of that slider. Matter.js has features to set the object's mass and coefficient of restitution. The position of the pendulum weights are calculated based on the length of the pendulum arm and the angle.

$$x = x_p - l * sin(\theta) \tag{1}$$

$$y = y_p + l * cos(\theta) \tag{2}$$

The animation receives inputs from the input sliders and the drop down menu, so user input can be limited to reasonable measurements for the pendulum system. The coefficient of restitution input will affect the pendulum game object's setting *restitution* - affecting how the objects react after colliding (i.e. sticking together or not). Matter.js has a *collision object* feature that has built in physical behavior and properties.

The graph of the weight's angle over time will replace the input sliders after starting the simulation. When the start button is hit, the sliders are hidden, and the graph is shown. The graph shown is based on the user's choice between angle vs time, height vs time, or velocity vs time. The height of the weights is calculated by the equation

$$h = L - Lcos(\theta) \tag{3}$$

Where L is the length of the pendulum arm, and the angle is relative to the center of the simulation. The velocity vs time graph uses the calculated values from Matter.js.

4.5 Viewpoint: Graphing

4.5.1 Design Concerns

Graphs can be over stimulating if too much information is being presented to the viewer. Caution must be had when creating the graphs, so the layout is clear, and only important data points are being used.

4.5.2 Design Elements

Graphs will be displayed to the screen with Chart.js, which is a lightweight JavaScript library that is free to use. Once the library is included with the script tag, a graph can be included by creating a Chart object with modifiable fields to build up your graph, such as, label for the graph's title, data for the graph's data to be graphed, and backgroundColor. After the object is made, it can be referenced to a canvas tag in order to be embedded within the web page.

4.5.3 Function Attribute

Graphs will be used within the UI for data visualization to provide the student with another resource to understand the interactions within the simulation.

4.6 Conclusion

The core aspect of this project's development revolves around the use of the Matter.js physics library, which will be supported with the UI designs written with HTML,CSS, and JavaScripts - utilizing helper JavaScript libraries like Chart.js and noUiSlider.

:

CS CAPSTONE TECHNOLOGY REVIEW

JUNE 7, 2019

INTERACTIVE 2D SIMULATIONS TO SUPPORT INQUIRY-BASED LEARNING IN MECHANICAL ENGINEERING

PREPARED FOR

OREGON STATE UNIVERSITY

TOM ESTKEDT, MILO KORETSKY

PREPARED BY

GROUP 53

LEARNING SIMULATIONS

KELLI ANN ULEP

Abstract

The solution to create interactive 2D mechanical simulations involves many different components. The web application has been broken down in different components. This document compares and contrasts 3 different technologies for 3 different components used to complete the solution. Each technology has its own benefits and drawbacks, and these will be used to decide which technology to use.

CONTENTS

1 INTRODUCTION

This project involves solving the problem of some universities lacking resources to visually show physical and mechanical interactions for Mechanical Engineering concepts in the classroom. To solve this problem, the project aims to build at least one 2D simulation that accurately emulates a physical or mechanical idea and allows for student interactions with the system. The simulation will run on a web browser, and students will receive links to different situations within assignments. The web application has been broken up into three components: animation libraries, styling libraries, and video players.

2 ANIMATION LIBRARIES

The simulations should use an animation library so that they can be created with greater ease. Web pages can use animation libraries with HTML, CSS, or JavaScript. Animation libraries should be simple to use, well-documented, and versatile. The library should be easy to incorporate into the code. The web page should use these libraries for creating the mechanical animations itself or other parts of the user interface - the buttons, sliders, or status updates. The options the section will cover are Popmotion.io, Animate.css, and GreenSock Animation Platform (GSAP).

2.1 Popmotion.io

Popmotion.io - specifically Popmotion Pure - is an open source, low-level animation library for any JavaScript environment. It works with most popular browsers to render any target like the DOM. Popmotion Pure seems the most appropriate of the Popmotion libraries as it offers more animation freedom. The Popmotion pure site contains a detailed ”Getting Started” tutorial and simple library import instructions. The API (Application Programming Interface) is also documented well with many usage examples. It is available on npm (a node package manager) or can be simply imported or included in the code with one line. This would allow for more ease of entry. The library size is only 11.7kb which which is smaller than GSAP.

2.2 Animate.css

Animate.css is ”just-add-water CSS animation” library and is known for being simple to use. It has 55,181 stars on GitHub, so it is fairly popular. The library is compatible with different browsers (cross-browser), and only really involves CSS3. Like Popmotion.io, it is incorporated easily into a project by installing with npm and referencing the the file. It can be combined with jQuery, another common animation library, making it a little flexible to use [1]. Video tutorials also exist online. The library is open source and licensed under the MIT license, so it would not take much trouble to use. The library does not provide much documentation except a readme file on GitHub, but at a first glance, the API seems simple enough that it does not need much explanation. Because it is such a simple animation library, even if it can be combined with jQuery, this library seems to work best only for simple web page animations like input sliders, boxes, or labels. It would not be as effective to animate the actual mechanical simulation because it so simple, but it would be possible.

2.3 GreenSock Animation Platform (GSAP)

GSAP is an animation library based off JavaScript. GSAP is a collection of JavaScript files, compatible with many popular browsers. It claims to be ”The most robust animation library on the planet” [2]. GSAP is also known for its performance

and efficiency. The site provides a speed simulation in comparison to other popular animation libraries like jQuery, and tops the list in performance. The site shows much documentation and video tutorials, so it would not take much to learn to use. A draw back is that some parts of the library is only accessible for paid users. A paid business license is necessary for a product that is sold to multiple users - which would have to be the case for this project. The simulation would be part of a site you have to subscribe to. Other libraries can do what GSAP can do without having to pay, but this well known library is recommended if robustness and speed is one of the main concerns of the web page. [2]

3 Styling Libraries and Frameworks

The front-end of the web page should be organized that the user can easily understand the format of the web page. Styling libraries make organizing a web page easier. The library should be simple to use that it is appropriate for an educational site. Preferably, it should have responsive design functionality, but it is not a high priority for this project. Most importantly, it should be simple to use, robust, and compatible with most browsers.

3.1 Bootstrap

The framework is considered the best responsive front-end framework and one of the most popular open source projects. With its grid systems and different buttons, it is one of the most commonly known responsive design user interface framework. It is one of the most popular libraries, so many documentation and tutorials exist, making it easier to learn if one has no previous experience. Also it is compatible with most browsers. Other frameworks may have the same functionality as Bootstrap, but because it is such a popular library, much more resources are available like extra add-ons, plugins, or provided themes. For the educational web page this project aims to build, Bootstrap's main advantages are that its well-documented, and the simulation can easily be made for tablets or phones [3]. Responsive design is not one of the high priorities, and it can be one of the stretch goals. With Bootstrap, however, it could make it easier to change if the project ends up going in that direction. The code is released under the MIT License and released under Creative Commons and can be easily incorporated into the project.

3.2 Bulma.io

Bulma is another open source CSS framework that is used by over 100,000 developers. It is based on CSS's Flexbox, having one of its main benefits responsive design. The framework is designed so the site works on mobile platforms first. The grid system that Bulma uses is simple since the web page columns or sections resize themselves. The library is modular such that it consists of 39 .sass files that can be imported individually. Its modular nature makes this library more lightweight than Bootstrap, as developers can import only what they need. Bulma is structured as a collection of CSS classes, so that the HTML code has no impact on the page styling. The site shows well presented documentation on all the classes, and the classes are readable. It is not as popular as Bootstrap, so it might not have as many tutorials and online resources. The code, however, is still well documented and is simply stated. Only one file is required to use Bulma, so it can be easily included in the code base [4].

3.3 Semantic UI

Semantic is another open source framework helping to build responsive layouts. The library's main benefit is to make the code human-readable since it regards words and classes as ”exchangeable concepts.” Library syntax is readable and is

intuitive, so the code can be understandable. This is important for the project since the code must be understandable by all current developers and future developers. Then developers can easily create and debug code. Though not as popular as Bootstrap, the code is still well documented and there are tutorials. There are also many built in UI components that the others may not offer, but there are no extra plugins as it is so simple. This library is still in development in integrating with other common frameworks like Angular or React [5]. It may become issue if the project uses those potential frameworks or would eventually switch to them. However, it is compatible with most major web browsers. The library can be used simply, but there is not so much room for flexibility. The set-up is not as straightforward as Bulma or Bootstrap.

4 Video Players

The website can have an option of displaying a video of the real system that the animation is based off. The way the video players are embedded into the website should be robust and simple, and the video must work.

4.1 HTML5 Video Player

This option allows more compatibility if the users come from different devices. All modern browsers support this option - so videos can also be streamed on mobile devices. This a simple option in comparison to other video services. This option also requires the video be prepared in three different formats, which can be troublesome. Also the videos must be hosted on the server which is a potential drawback. This is a possible option, as we do not have to worry about compatibility or ads, but also it is more complicated to add to the webpage.

4.2 YouTube

The option is the most common way to embed a video. It is the simplest to implement as all it requires is to click on the *Embed* button and copy the code into the web page. YouTube is also easily accessible on its own. This option is highly compatible with many browsers. There is also no need to host videos on the server, but then the website would have to depend on YouTube to host which is a potential drawback. YouTube videos may also have ads. However, this is the simplest and most accessible solution for this project.

4.3 Vimeo

This site offers free video hosting. The player is customizable unlike the YouTube player (if you pay for Premium), and it also just as simple to upload and embed. Paying for premium for a customizable player is not necessary as the video is not much of a priority for the simulation web page. It also does not show in HD unless paying for premium which is also a drawback. Vimeo also offers compatibility for other browsers and free video hosting like the previous two options[6].

5 Conclusion

There are many libraries and methods to use when developing a web page. Each one has its own benefits that pertain to each requirement priority of creating a 2D mechanical simulation. Popmotion.io is a suitable animation library since its robust, well documented, and versatile. Bulma.io and Bootstrap are both well documented and versatile styling libraries, but Bulma.io is simpler and modular, and does not take much to incorporate into the library. For video players, YouTube is the most common and easiest solution to embed videos.

REFERENCES

[1] daneden. (2018) animate.css. [Online]. Available: https://github.com/daneden/animate.css

[2] G. Sock. (2018) Gsap, the standard for javascript html5 animation. [Online]. Available: https://greensock.com/gsap

[3] (2018) Introduction - bootstrap. [Online]. Available: http://getbootstrap.com/docs/4.1/getting-started/introduction/

[4] (2018) Overview — bulma. [Online]. Available: https://bulma.io/documentation/overview/

[5] (2018) Semantic ui. [Online]. Available: https://semantic-ui.com/

[6] (2018) Embed video on website: 5 easy ways. [Online]. Available: https://www.freemake.com/blog/embed-video/

:

Tech Review

Cameron Friel

Interactive 2D Simulations Group 53

Senior Software Engineering Project I

Term 1

Abstract

This project involves solving the problem of some universities lacking resources to visually show physical and mechanical interactions for Mechanical Engineering concepts in the classroom. This impedes the learning process for Mechanical Engineers by forcing them to take an auditory approach to learning when the subject matter is very difficult to understand in the first place. This project is built on the research that shows that students can achieve a better understanding of difficult concepts by learning through simulated environments that they can interact with. By implementing two dimensional simulations based on these concepts, students will be able to visually interpret the concepts in the course. The solution being implemented is to create two dimensional simulations using primarily client side Java Script. The student will be able to modify certain values like the mass of an object or the angle that an object is dropped within a simulation in order to understand how interactions that occur in the real world are rationalized.

1 INTRODUCTION

Currently, some university's Mechanical Engineering classes lack the resources to visually demonstrate the physical and mechanical interactions that need to be learned in the classroom. This creates ambiguity for the students trying to understand Mechanical Engineering concepts, such as something rudimentary like how rolling cylinders of various sizes and materials down a ramp demonstrates the concepts of mass and acceleration. This project is built off of research that has shown that creating environments where students can visualize the concepts they are learning increases their understanding of the material. With the idea of cognitive conflict, when students observe ideas that challenge or contradict their prior ideas or beliefs, then they are pushed to think more about it. Especially when the concepts are non-intuitive, they may struggle. This project hopes to solve this problem by creating several two dimensional simulated environments that include these concepts. The students will be able to modify some of the variables within the simulation in order to enhance their understanding of the concepts being presented. These environments will be available online for students to see and interact with in order to attain a stronger grasp of the material. The simulated environments will go along with lessons hosted on the Concept Warehouse website, a platform to help supplement student's education.

In order to begin the development for this project, Cameron Friel has been tasked with conducting research in the areas of Software Development methodologies, Graphing Libraries for the web, and 2D Physics Engines. His role is to make sure the team sticks to deadlines and are made aware of problems that might come up. He is also responsible for making sure that any data within the simulation is able to be graphed in real time with readable data points. His third task is to make sure that the physics engine used is capable of simulating physical properties that would happen in real life into a 2D space on a web page.

2 SOFTWARE DEVELOPMENT METHODOLOGIES

The Software Development methodology chosen will be used throughout the entire project to make sure the team stays in line with the schedule set up by both the class and our own development specifications. The methodology will benefit us by setting up expected deadlines and give responsibility to each team member. It will also help make sure the client can see our progress and be able to steer us in the right direction if what we are doing is not align with what is expected.

2.1 Agile Development

Agile Development excels at being adaptive to the ever changing requirements the client can give during a project, as well as splitting development into short sprints so that errors can be caught sooner rather than later. By splitting up tasks into short sprints, the development that happens during that sprint will also be tested so errors are caught early on in the process. These short sprints also allow for constant communication with the client to show the current progress in the project to make sure what is being delivered is what the client wants. Where Agile Development falls short is that the outcome is never clear because the development is always changing and sprints are planned after the previous one ends [1].

2.2 Waterfall Development

The Waterfall Model is a very classic Software Development methodology because it is very structured and easy to follow. The Waterfall Model focuses on maintaining a sequential linear progression for project management [2]. Every project that follows the Waterfall Model starts by defining the requirements and creating the system design. After this, the project goes into implementation phase and then testing follows. Once the project has wrapped up

testing the project is deployed and maintained for the rest of its life cycle. This cycle is good because it is structured and quick, but it also falters in many ways since it fails in projects that go on for a long time or when new features need to be added [1].

2.3 Spiral Development

The Spiral Model is used mainly in big projects that require tremendous risk management where failure cannot occur. The Spiral Model is similar to the Waterfall since it takes on the approach of a linear progression. This model has a high overhead since the project is slowly and meticulously defined by assessing risks and creating prototypes to prove that the system can be created. If the risk assessment is too much during any point in the project the project can be completely stopped [3].

3 Graphing Libraries

The project will require that within the simulations data be graphed in order to display a visual demonstration of what is going on. This will provide students with a secondary way to understand the material being presented that would not be possible with a physical version of it. The user will be able to visually see the graph as well as see the exact values for each point on the graph.

3.1 Highcharts

Highcharts is an interactive JavaScript graphing API that is free for non-commercial use. It is used by 72 of the top 100 corporations in the world. Additionally, Highcharts provides 20 different graphs to choose from including heat maps, scatter plots, and log graphs. The documentation is very robust and has plenty of demos for pretty much any use case. Since the library is built with pure JavaScript, there is no need for a user to have Flash or Java downloaded. It also has support for all modern browsers and is responsive to changes in screen size. When it comes to live data, Highcharts is able to dynamically graph points that come in from Ajax request or client side JavaScript [4].

3.2 Chart.js

Chart.js is an open source Graphing Library boasting 8 different graph types that can be used. It has responsive web capabilities and has excellent performance across all major web browsers. Chart.js provides incredible transitions when graphing and when new data comes in via animations. Chart.js is easy to use and the documentation covers its uses thoroughly. This library provides its users with many customization options including, but not limited to, a title, legend, and tool tip for its graphs. Furthermore, it has the most active Github repository when it comes to both making new content and fixing bugs when compared to other graphing libraries for the web. Lastly, Chart.js is very lightweight and is free for anyone to use [5].

3.3 Chartist.js

Chartist.js is another JavaScript solution for web. It supports SVG image format allowing for beautiful graphs on any resolution the end user might have. This library works on the basis of separation of concerns, meaning that customizing the graphs are dependent on CSS, while the functionality of the graph will be done with JavaScript. This library provides several animations that can be added to any of the SVG elements, though there is no support on Internet Explorer. Overall, Chartist.js works with all major web browsers and supports responsively to the user's screen size. Its drawbacks come from its true potential being shown only to users with modern web browsers. It also is one of the least contributed repositories compared to its competitors [6].

4 2D Physics Engines

In order to create 2D simulations for mechanical engineering concepts, there will need to be an engine to render the elements of the simulation, as well as account for all of the physical properties that must happen within the simulation. A physics engine solves both of these issues since it includes libraries to add rigid bodies to elements, account for collision detection, add gravity to the world, and so much more. Utilizing a 2D physics engine will also negate a lot of development time reinventing the wheel with physical properties.

4.1 Matter.js

Matter.js is a free open source 2D physics game engine created for the web. Its main purpose is to create realistic 2D simulations on the web. It includes the necessary elements for a physics engine, such as rigid bodies, so each element that interacts with the world has collision detection added to it. Each element created can be tuned to the exact parameters it would have in the real world. For example, Matter.js allows one to set the objects friction and restitution so as to create a real life simulation within the browser. Matter.js has a plethora of documentation and demos for ease of use. Matter.js is written in JavaScript, making it web friendly, and does not require any third party libraries to use.

4.2 Phaser.io

Phaser.io is a HTML5 game engine which utilizes WebGL to allow for hardware acceleration. Phaser.io is open source and has an amazing set of examples and documentation for beginners. It recently integrated 2D physics into its code base in order to allow for high level physics simulations. It allows for rigid bodies, constraints, springs, restitution, and so much more. Since Phaser.io is a game engine, it is very easy to integrate assets into any simulation that we create. Additionaly, Phaser.io is built with JavaScript, so no third party libraries are needed, however it does need to have a web server up in order to test it. The game engine uses HTML5 canvas to display its games and is responsive depending on the size of the screen.

4.3 p2.js

p2.js is an open source 2D physics engine for the web. Written in JavaScript, it is easy to integrate and requires no third party libraries to be used. p2.js was show cased at the Google IO event in 2015. Its main use case is for game development, but can be used for anything requiring realistic physics properties. p2.js has all of the attributes one would need in a physics engine, such as rigid bodies, advanced constraints, collision detection, springs, and much more. A real application can be seen in one of its demos where objects of different masses are dropped into a pool of water to demonstrate buoyancy.

5 Conclusions

The best option out of the three software development methodologies would have to be Agile Development for us. Since there is very little risk management needed in our project, and we will most likely be dynamic with our design decisions, an Agile project management makes the most sense. Our client will also enjoy being able to see our additions to the project so that they can give feedback as well. All three graphing libraries chosen provide a very similar feature set, however, Chart.js is the most lightweight and provides exactly what we need. It is able to update in real time and the only graphing we will need are line graphs. Couple this with an active community, it will have the smoothest development experience. Matter.js will be our 2D physics engine of choice. It has the best documentation and has the most features that we need for our project.

REFERENCES

[1] "12 best software development methodologies with pros and cons," Acodez, September 2018.

[2] "What the waterfall project management methodology can (and can't) do for you," Lucid Software Inc., August 2017.

[3] "spiral model (spiral lifecycle model)," TechTarget.

[4] "Highcharts review," FinancesOnline, 2018.

[5] "Comparing the most popular javascript charting libraries," Medium, October 2017.

[6] "Compare the best javascript chart libraries," Sicara, June 2017.

:

CS CAPSTONE TECHNOLOGY REVIEW

JUNE 11, 2019

INTERACTIVE 2D SIMULATIONS TO SUPPORT INQUIRY-BASED LEARNING IN MECHANICAL ENGINEERING

PREPARED FOR

OREGON STATE UNIVERSITY

TOM ESTKEDT, MILO KORETSKY

PREPARED BY

GROUP 53

LEARNING SIMULATIONS

SAMUEL WILSON

Abstract

This document seeks to review several technologies, which will be integral in the creation of a project dedicated to creating a 2D simulation website. The 2D simulation will be used for the education of mechanical engineering students in simple physics. Examples would be: pendulums, pulleys, rolling cylinders, etc. Game engines, art software, and project management software are the technologies being reviewed. The goal will be to choose the most well fitting software for the project, based on a set of criteria.

1 INTRODUCTION

This project aims to solve the problem of some universities lacking resources to visualize physical and mechanical interactions for Mechanical Engineering concepts in the classroom. By implementing 2D simulations based on these concepts, students will be able to visually interpret the concepts in the course. The solution to be implemented is the creation of 2D simulations using client side Javascript. The student will be able to modify certain values like the mass of an object or the angle that an object is dropped within a simulation in order to understand how interactions that occur in the real world are rationalized. This document looks over technology that can be used in the project's creation, in order to decide the best options. Game development engines will be compared, along with 2D art software and project management software.

2 GAME DEVELOPMENT ENGINES

The main feature of this project requires 2D physics simulations, along with several important UI controls. A possible solution to building this would be to use game development engines. When choosing one, the key aspects to look for are:

- Number of features
- Ease of exporting
- Clear user interface
- Easy coding language to learn
- Quality of Physics implementation

2.1 GDevelop

The first engine is GDevelop, which is an open-source game creator with an online UI. This engine seems to have all necessary features, though simple. The website promises web exporting with one click. Its designed for 2D games, so sprites and particle effects are implemented well. The UI can be accessed in a browser, as well as in a desktop app, which both are very simple, with a focus on visual programming. Objects in this engine can also use a built-in physics system with the ability to change friction, mass, and forces like gravity. Overall, this engine has a solid set of features, but they are simplistic and the visual programming does not seem intuitive. [1]

2.2 Defold

At first glance, Defold seems to be more feature complete then GDevelop, possibly due to its use by professional companies. Defold allows for easy development to the web using HTML5, also having a single click export. The software is only available on desktop, with no browser development. Defold also promises easy tools for version control. The UI for Defold is much more robust than GDevelop, with the layout of a typical IDE. Lua is used for scripting, which is a lightweight language with some influence from C++, so there should be a small learning curve for our team. Finally, Defold uses Box2D [2] for its physics, which is a well known C++ physics engine. Defold seems like a very promising option, with higher quality features than GDevelop. [3]

2.3 Godot

Godot is an open-source, desktop application. It has a similar look to Defold, with a professional layout and set of features. Godot doesnt seem as clear on whether its web exporting is quick and easy. As stated, the UI for Godot is clear and has a well organized layout. This engine uses a custom language called GDScript, which is based on Python. It also has the ability to use visual scripting. Generally, Python is a very simple language, but some learning may be required. While not promoted as a feature, it does seem to have a physics engine based on Box2D. Godot seems to be similar in features to Defold, though the custom programming language may get in the way of quick production. [4]

3 2D ART SOFTWARE

In order to give feedback for the physics simulations, there needs to be a visible representation of the model. To do this, the team will need graphics software to create simple 2D art. In order to decide which software to use, the following criteria will be used.

- Software purpose
- Image manipulation
- Clean interface
- Ease of shape creation

3.1 Inkscape

Inkscape is a vector-based graphics software, which means objects created in the program can be easily scaled and modified. It seems to be highly recommended for making logos. It lacks in tools for manipulating images that were imported, compared to other software. Inkscape has a good interface for quickly making images which are designed with shapes. This project will only need basic shapes and lines for each element of the simulation, so Inkscape seems like a very good option. [5]

3.2 Krita

Krita is designed for digital artist and illustrators. It has very few features that revolve around image manipulation. It's expected that the user will be an artist who will want to customize everything. Most of the interface is moveable to function as an efficient canvas. These features are not very important for the project, unless the quality of the art needs to be improved later. Krita has some functions for creating images from shapes, but its functions are clearly for more artistic pursuits. Overall, Krita would be a good option if more detail was needed for the project, but based on the current criteria its too complex. [6]

3.3 GIMP

GIMP advertises itself as an open source, image manipulation program. Its tools focus on modifying imported photos and art with a large variety of graphical effects. This makes it great for creating textures and interesting graphics, however GIMPs tools may be more than what is needed. Due to the vast array of the tools and effects, GIMPs interface is not as well put together, especially compared the previous two options. For creating simple graphics made of shapes, GIMP seems overly complex, though it has the features to support it. For the purpose of this project, GIMP seems to more than what is needed. It is also for more general purpose image editing; not having a specific design goal, when compared to the other options. [7]

4 Project Management Software

In order to keep the project on time, and organize the tasked assigned to each member of the team, our group will be using project management software. The criteria for choosing the best choice are:

- Designed for small groups
- Feature Variety
- Ease of task management
- Integration with other software

4.1 Clubhouse

Clubhouse follows a Kanban board style, using a 'Story' for each task to complete. Clubhouse has a simple interface and flow, although it lacks some features that the other options have. It is designed for Agile development, which may be important, depending on the team's design goals. This software only allows three people in a project when using the free version, but that shouldn't be a problem with our team size. Finally, Clubhouse has the ability to connect with Github and Slack, which our team uses regularly. [8]

4.2 Pivotal Tracker

Similar to Clubhouse, Pivotal Tracker focuses on Agile development and uses 'Stories' to designate tasks. It is not specifically designed for small groups, but the free version is only limited to three people per project. A key feature of Pivotal Tracker is the analytic tracking to show progress for members and the team as a whole. This would be useful to keep each member accountable. Task management is simple, using Kanban boards, and the interface is clean and organized. Pivotal Tracker's integration is extensive, including the teams most used platforms, Github and Slack. [9]

4.3 Redbooth

Redbooth promises good organization for both small and large groups. It uses a Kanban board, like the other options, but it does have some features that the others don't. One of the features that the team would be most likely to use is the Gantt chart integration. The other options don't seem to have this basic function. Redbooth is just as easy to use as the other options, but with more functionality. Oddly, Redbooth does not have Github integration, though it does have Slack and Box integration. Our client wishes to use Box, so it could be a fair trade off for the Github integration. [10]

5 Conclusion

After analyzing the technologies of game development engines and 2D graphics software, there are pretty clear choices for the best in each category. For game engines, the goal is to find something robust, with accurate simulations, so the amount of problems will be minimal. The team should not run into something that is not supported by the engine. It should also be easy to script in the engine, with little learning needed. Both Defold and Godot have been used by professional developers, and were released to the public. They both have a good set of features, but Godots custom language may have more a learn curve for the team, so Defold will be the first choice for this technology category.

For the choice of 2D graphics software, the goal is the opposite of game engines. The project needs a very simple and quick image creation tool. Krita and GIMP have a wide set of tools, but between them, Krita seems to have a better

user interface. Unlike these two, Inkscape fits the projects need for simplicity. It also has a clear user interface with just enough tools for the teams needs. Unless more interesting art is needed, Inkscape is the best choice for this project.

Finally, for the choice of best project management software, the options are all very similar. All of them use Kanban boards for organization, with some focus on Agile development. Redbooth has the most features out of the choices, with the addition of some very useful ones. All of the software choices had very clean interfaces, so there is not clear winner for this criterion. For integration, Redbooth was missing Github, but the inclusion of Box integration somewhat negates this flaw. Overall, Redbooth promises the most usefulness out of the choices, despite all of them being very similar.

REFERENCES

[1] Gdevelop. [Online]. Available: https://gdevelop-app.com/
[2] Box2d. [Online]. Available: https://box2d.org/
[3] Defold. [Online]. Available: https://www.defold.com/
[4] Godot. [Online]. Available: https://godotengine.org/
[5] Inkscape. [Online]. Available: https://inkscape.org/
[6] Krita. [Online]. Available: https://krita.org/en/
[7] Gimp. [Online]. Available: https://www.gimp.org/
[8] Clubhouse. [Online]. Available: https://clubhouse.io/product
[9] Pivotal tracker. [Online]. Available: https://www.pivotaltracker.com/features
[10] Redbooth. [Online]. Available: https://redbooth.com/features

5 Blog Posts

5.1 Cameron's Weekly Blog Posts - Fall Term

5.1.1 Week 4

My group and I got together to make a master copy of our problem statement and I believe we are all on the same page for how the project should work. From the meeting with the client on Thursday I believe he is on board with what we said as well. We are leaning towards either Matter.js or Godot for the 2D physics engine to use within the browser.

Coming up this week we are not planning on meeting with the client but have made 3:30 pm on Thursdays to be the set meeting time if we ever need to meet. The group and I are going to be meeting up later next week to work on the requirements document and begin playing with the physics engines to see which will work with the project the best.

5.1.2 Week 5

The group and I have made our GitHub repository so that we can start uploading documents. We have shared this repository with our TA as well. We went into office hours to look at some example requirement documents to give us a better idea on the size of the paper as well as the content and structure. We met with the TA to find out more about the tech review requirements, so that we can start thinking about the technologies we could use for our project.

We have divided the work up for the requirements document so I do the introduction section, Kelli does the requirements section, and Sam does the verification and time line. We have also brainstormed our 9 potential areas to study for the tech review.

5.1.3 Week 6

We decided to wait until next week to have another assigned meeting with the client. We did however email them our rough draft of the requirements document for him to look over for next Thursday. We went over the nine technologies together and finalized what each person will do. We did not have a chance to ask Kevin if physics engines with IDE's vs ones with not was too much overlap.

We plan to meet with the client next week to go over and refine the requirements document, as well as make the changes our TA put forth to us. These changes will include adding a definitions section and putting more detail into what the requirements entail.

5.1.4 Week 7

We all completed our tech review's and think we have found the technologies we want to use. We met with our client and did any areas he disagreed with. He did give some suggestions on word structure but that is expected. During the meeting we also discussed that we were beginning to write the design document, so we got to go over in more detail what we should plan to develop in order to avoid problems down the road.

We plan to meet with the other groups in the coming weeks as well as with the client roughly two weeks into the design document to make sure they agree with what we are planning.

5.1.5 Week 8

Met with client and went over more details on the project to begin writing the design document.

We plan to meet the client Tuesday to show our progress on the design document. We also will be having a voice conference with the ta Saturday to update him on our design document.

5.1.6 Week 9

We have begun work on the Design Document finishing the wire frames for the UI design and have come up with the different viewpoints to write for the design document.

We plan to meet with all the groups this coming Friday at 3.

5.2 Cameron's Weekly Blog Posts - Winter Term

5.2.1 Week 1

This week we got together with our client to figure out which times work for all of us as well as planning to meet the three other capstone groups to show our alpha build to each other week 6. This time ended up being 1:00 pm on Friday. We also planned a meeting with Brian Self, the content creator for the project, week 3.

We discovered the engine we were using would not work out for the project like we hoped because it did not include UI elements necessary for the project.

We are currently addressing the issue by making more thorough assessments of two of the other engines to find one that will work for the project.

5.2.2 Week 2

We went ahead and went with Matter.js for our 2D physics engine in the web. We created the skeleton that each of our simulations will be based on and assigned tasks that each member will be doing for the project roughly in our wiki.

We will be meeting with our client this coming Friday to go over specifics for the simulations.

5.2.3 Week 3

This week we finished working on features for the reset, pause, and start buttons. We also implemented functionality to keep track of time for the simulation and track the angle. We met with client and he looked over our prototype and gave us suggestions on what to change. We have yet to make the graph responsive with the rest of the page.

We plan to finish the case 1 simulation this week and begin working on the case 2 simulation.

5.2.4 Week 4

This week we were able to get the start button functionality working as well as meet with the client to get their critique on our project. We were able to fix some of the things they suggested like the rounding of numbers

There is an error is the Matter.js library which does not let us have a pendulum that swings forever even when setting no air friction and infinite inertia in the world. This energy loss can only be set back to original levels by artificially adding more force back to the pendulum.

We plan on meeting our client this Friday to update him on our progress. We plan to finish up simulation 1 and work and hopefully finish simulation 2 later this week.

5.2.5 Week 5

We were able to get a lot of progress done this week. We completed the first simulation by getting the height and fixing the angle calculation. We started working on exploratory mode working on the sliders.

The current problem is the library does not calculate the swing of the pendulum when air friction is zero correctly. We will have to add an extra force to the pendulum to counteract this.

We will finish the exploratory mode the and second simulation this week for the alpha meeting on Friday.

5.2.6 Week 6

Over the week we were able to accomplish getting a basic template of each case and the exploratory mode. Everything is working as of now, but there are bugs and quality of life updates to come.

Just minor bugs and questions about exact math calculations vs the current simulations.

We plan to send over the current simulations for the content creator to look at and give his second round of criticisms and suggestions. Next week we want to work on fixing some bugs and making it so the graph comes into view in exploratory when the user clicks a button to see it.

5.2.7 Week 7

Over the week we worked on minor bug fixes and worked on our progress report going over what we have done over the term, what needs to be done, the problems we have run into, and the other various parts required in the report.

We will want to plan a meeting with the client to look over the project again this week.

5.2.8 Week 8

This week we were able to update the pendulum to have a thicker string as the client want and have the string display above the pendulum weight rather than on top of it. There has also been updates to the UI of the site so it is easier to read for the user. Another live update calculation for the second pendulum's height has been added. Lastly, more work has been done to reduce air resistance, but it is not perfect. In low angles energy is increasing.

We are meeting with the client next week for another progress update. More work on the repo will be done.

5.2.9 Week 9

This week we implemented a trail that is painted to the canvas and follows the path of the pendulums. We also implemented a modal that shows up when the simulation page is loaded to explain the scenario. Fixes to the exploratory mode were made as well as bug fixes.

Next week we want to begin work on the video and the end of term paper as well as work on bug fixes and small feature updates.

5.2.10 Week 10

Continued work on the repository addressing issues. Started working on the presentation and final paper for the class.

We are planning to meet Sunday to record the video. We are planning to meet with the client next Thursday for the final time this term.

5.3 Cameron's Weekly Blog Posts - Spring Term

5.3.1 Week 1

Worked on static values tables to be displayed. Fixed formatting issues. Met with client. Submitted model release forms.

Plans to meet with two other capstone groups to show final project to our client and each other week 6. Set up meeting with client on Fridays.

5.3.2 Week 2

We were able to implement two new graph types for users to utilize in the simulations (velocity and angle). There were also bug fixes and improvements to exploratory mode.

I plan to go to the next capstone meeting since I forgot to go to the last one. We will be sending the current version of the project to the client and meeting Next Monday. Revisions to the design doc will be made.

5.3.3 Week 3

Added a build all script and updated readme for graders. Added new graph options. Modified requirements and design document and got client to sign. Attended class to listen to guest speaker. Confirmed capstone plans for expo.

Meeting with client Monday. Waiting for feedback. Working on minor fixes throughout the week.

5.3.4 Week 4

We went to talk to Kirsten to critique the poster and have since finished said poster. We have not gotten a reply from Brian, so I nudged him with an email. We made updates to the coloring within the application. Still waiting on the final feedback.

Group meeting with the two other capstone groups. Bug fixes when they come up.

5.3.5 Week 5

Made revisions to poster and submitted it for printing. Attended three team Capstone meeting to demonstrate projects.

Made arrangements to meet with client next Friday.

5.3.6 Week 6

Started working on mouse constrains for exploratory. Met with client to go over expo plans and work that can still be done on the project.

Next week will try to implement mouse constraints before expo. Monitoring bugs.

5.4 Kelli's Weekly Blog Posts - Fall Term

5.4.1 Week 4

We compiled all of our problem statements together successfully and met our TA. We met our client for the second time, and he gave us more specific requirements about the project. He also gave us more specific requirements from the co-creator of the exercises we're incorporating our simulations into. We established a weekly meeting time.

We plan to create initial designs or mock ups for the simulations so the client can easily comment on it. It will be easier than planning through words what features or buttons there should be.

We should come up with a more definite time to show the client our design. But first we need to play with some of tools we can use to see what is generally feasible.

5.4.2 Week 5

Our team has a git hub. We are working on the tech review and the requirements document. The requirements document is straight forward since our client is pretty clear in what he wants. The technologies we needed for the tech review were hard to think of as a lot of the possible libraries encompasses many of the website's components.

5.4.3 Week 6

We finished the requirements document draft. We need to work on revising it so some of the statements are not so vague.

We scheduled a meeting with our client for next week Thursday and sent him a copy of our first draft so he could possibly look over it.

Now we are working on the Technology review. We are all discovering various ways to implement the web page. There are many different physics libraries/engines so we need to be able to pick the most convenient and easy to learn library. It needs to also be robust.

5.4.4 Week 7

We started looking at new technologies and thinking about implementation. We had another meeting with our client and asked for more clarifications.

We plan to meet with him again 2 weeks from now. By then we should have a basic mock-up of the simulations.

5.4.5 Week 8

We met up again to discuss the design document. We are a bit confused on how to split it up for our project because if we base it off the tech review it does not really cover all the aspects as nicely. Our group should find time to meet a little. I started working a little on the initial UI design.

We plan to meet the client again on Tuesday to talk more about the design.

5.4.6 Week 9

We created a mock-up of the UI and discussed it with our client on Tuesday. We began splitting up the components into the design document.

Our client gave suggestions for the UI and we will implement them into the mock up and give another verification. Since next week is dead week, we will mostly communicate with our client via email and then have a group client meeting on Friday. He said we are close to being done for the requirements.

I looked into trying to figure out the IDE one of our team members suggested to use for this project. It was easy to use, but it involves a scripting language instead of javascript. So it involves some learning into incorporating it into a web page.

5.5 Kelli's Weekly Blog Posts - Winter Term

5.5.1 Week 1

We had a meeting with our client. We as a team should find a set time to meet each week. We are being pretty open right now. We've just made some progress in starting animations.

5.5.2 Week 2

We met up on Tuesday to discuss how we should split up the work. We decided to go a different direction in implementation and use the Matter.js library instead of Defold. We split up the assignments by user interface, animation, and graphs. We decided last week to meet our client next week Friday to possibly also meet with Dr.Self. He expects at least one working simulation done with basic animation. We still must figure out how to work the servers to have it sync with Concept Warehouse.

There's a plan to meet on Friday to work on our pitches and see what we all did.

5.5.3 Week 3

We have been working on the early stages of the web development. We've created issues on git hub for easier tracking, so far I'm working on the UI, Cameron on the animation, and Sam on the graph. So far, we have 1 case, where there is a pendulum that just swings back and forth with a very barebones UI. We had a meeting with our client as he commented on our progress and mockups - what changes we had to make.

We've run into a couple of bugs, where the buttons weren't working correctly but we're addressing them pretty quickly.

We plan to meet in 2-3 weeks, but aim to have at least one case due by next week Friday as to give ourselves a deadline.

5.5.4 Week 4

We've been working on finishing up our first simulation. We have created issues on github. So far we aimed to finish our first two simulations by this week, and we're all working out some of the bugs. We're also fixing up some of the revisions our client suggested.

I decided to try Bulma instead of Bootstrap as a styling library as Bootstrap was not working out somehow.

We plan to meet with our client next week, hopefully with at least 2 simulations completed. We should also start working on exploratory mode since it seems the more complicated of them.

5.5.5 Week 5

We are finishing up the first simulation, and fixing bugs here and there. Case 1 is decent, there is just a few issues left to finish.

I started looking into exploratory mode. Previous libraries I thought would be ok didn't incorporate well into the project, so I used the noUiSlider library for range input. Also some of the css libraries (Bulma, Bootstrap) did not work as expected either. So far the site looks acceptable.

The client will meet with all the groups next week to see our current progress. We should have a demo by then.

5.5.6 Week 6

We finished the basic functionality of all the simulations and exploratory mode : simulation, graph, live update table, buttons, and a way to get input.

There are just bug fixes here and there. The main bug we are having is that the physics library we use is always accounting for air resistance in our model, and our client wants a perfect system. There is an option to remove air resistance in the physics library, but it is not working so we'd probably have to fix the bug ourselves. We also currently are counteracting it by apply a small force to the system to get the correct expected answer.

We plan to work on fixing the layout, fixing our live update table, and applying some nice to have features like different graphs to choose from.

5.5.7 Week 7

We have the basic functionality down, we are still working on some minor features: making sure the math is right is the most important. We are also doing more to polish the layout.

We are having more trouble than we anticipated with the air resistance for exploratory mode. It is the most obvious of the 6 stages that it is not a perfect physical system.

We plan to meet with our client next week to update them with our progress.

5.5.8 Week 8

We emailed our alpha version to our client -Dr.Self, who is not our main point of communication but is part of the project. We scheduled a meeting with out client for next Thursday. Our group is still having issues with the air resistance, but I found a hack-like way to reduce it at least for our pendulum case. It involves changing the library's source code. It may introduce more unforseen bugs if we incorporate it into our project, so we're holding off in incorporating it. Right now we're still working along on fixing minor bugs and adding some features - the layout for exploratory mode, the live update table.

5.5.9 Week 9

We met with the client this week to show him our small improvements. Our other client also got back to us for some feedback on the alpha. Our biggest issue is still the air resistance in exploratory mode. We explained our options to our client -

1) include a input range slider to control air resistance, and show the user that the air resistance cannot be 0

2) Try to incorporate a way to minimize it as much as possible - there is no right formula or way to do this, but we can reduce it a good amount by hardcoding a table of values. This option is more hack-like.

Our client also provided more feedback - minor issues like wording, layout. We also changed the display of two pendulums to hit so that they are inline with each other.

We plan to meet again during finals week.

5.5.10 Week 10

We've been working on putting the finishing touches on our project. We will probably be looking more into the air resistance issue next term too.

Right now we're trying to verify if the math is all correct since our client told us to make the weights collide inline with each other and now the calculations we initially had seems questionable. The method the engine uses causes the physics system to lose energy. We also started adding a help-modal and some extra information.

We plan to next week to finish up the progress video.

5.6 Kelli's Weekly Blog Posts -Spring Term

5.6.1 Week 1

We added finishing touches to our webpages. The last major thing we have to do is add a selection of graphs for exploratory mode. It's a nice-to-have instead of a requirement. We will email our client with more updates by early next week. There are some things that have been broken (the static graph and some word choices) with new features and we are working on fixing them before then.

5.6.2 Week 2

We finished adding the new graphs and the static tables our client asked for. In two weeks we will meet with our client again.There are still some options we are trying to choose from - whether or not we had the pivot point to be the same or not.

There's also very small bugs left to fix.

5.6.3 Week 3

We turned in our code freeze, and emailed our final questions to Dr.Self. We will have a meeting with Tom tomorrow to discuss our final parts of our project. We finished fixing small things from our last meeting, and finished fixing our design document.

5.6.4 Week 4

We finished up the poster and at the same time changed the color scheme of the simulations to be less saturated and bright. We also color coded the table.

We are still waiting on our final feedback from Dr.Self for some small details - which pivot point to use, what colors to change. We plan to have a group meeting next week with our client again.

5.6.5 Week 5

We submitted our poster for printing on time, and are currently making the final finishing touches on our project. We just had a group meeting with all of Milo's other capstone groups and they gave us more feedback on our project.

One of our problems is that one of our weights is not perfectly sticking together. Another thing is a professor wanted another case similar to case 4 of our simulations but with a coefficient of restitution of 0.7 so they bounce back. We also do not think we could get to the next spool simulation but we could try.

5.6.6 Week 6

We had a meeting with our client to talk about some of the final steps for our project. We are to comment our code for documentation which he said would be sufficient.

We added some fixes like the wrong height calculation for exploratory mode, and we added a new case which is similar to case4 but with e=0.7. We will also investigate some of the finishing details like adding tool tips to the buttons. Professor Self has given us some feedback, and we will investigate further. We are only left with nice-to-have features. We have also been looking into improving how the weights stick together in case2 and case3 since it isn't perfectly e=0.

We sent a snapshot of our code to Tom so he could sync it with the concept warehouse servers.

5.7 Samuel's Weekly Blog Posts - Fall Term

5.7.1 Week 4

Our group met with our client for the second time this week, discussing the information for the requirements document. We also met to finish up the problem statement. We have a good idea of what our project is, and plan on drawing some early mock-ups for the UI, as that is the next step to give our client something to look at.

5.7.2 Week 5

Our team met to discuss the tech review assignment. We were able to come up with enough technologies to analysis. No problems have come up yet. We are waiting until we have the requirements draft finished to meet with our client again.

5.7.3 Week 6

This week we finished the requirements document draft, as well as the tech review draft. We didn't run into any problems. The next goals are to finish the tech review final and start the design document.

5.7.4 Week 7

Tech reviews were completed and we had another meeting with the client concerning the requirements document. We didn't experience any problems this week. We plan on having some of the design document complete before the next meeting in two weeks.

5.7.5 Week 8

No real progress or problems. We are starting the design document this weekend.

5.7.6 Week 9

We met with our client, and presented some mock-ups of our UI layout. No problems presented themselves. We plan on finishing the design document over the weekend.

5.8 Samuel's Weekly Blog Posts - Winter Term

5.8.1 Week 1

We met with our client and showed a basic prototype of the project. We may change the software we are using, as we found a possible problem with the current one: Defold. We plan on meeting up to work with each other on Tuesday.

5.8.2 Week 2

Had a team meeting to designate tasks. We changed our mind on which of our techs to use, so we can accomplish stuff faster. We will meet Saturday to work on our poster.

5.8.3 Week 3

We presented a prototype to one of our client/experts for feedback. We ran into issues with using our chart library. We plan on improving our prototype into our first finished case by next Friday.

5.8.4 Week 4

We just worked on the project this week, with the goal of completing one scenario this week. We haven't run into any problems. We plan on making one more scenario next week and meet with our client.

5.8.5 Week 6

We finished our Alpha version and meet with our client this week. We feel confident we can complete the project on time. We only have to fix some bugs and formatting.

5.8.6 Week 7

Not much progress since the last client meeting. Will be working over the weekend.

5.8.7 Week 8

We worked on minor things and sent our alpha to one of the clients to get some feedback. We plan on fixing anything they find is problematic.

5.8.8 Week 9

We met with the client and discussed the final things we need to work on to complete the project. We plan on working on some of these things over the next week.

5.8.9 Week 10

Not a lot of progress this week. We will be working on the final video and report over the weekend.

5.9 Samuel's Weekly Blog Posts - Spring Term

5.9.1 Week 1

We are meeting with the client on Friday, and finishing up our list of issues. We plan on working on documentation next week.

5.9.2 Week 2

We sent an email to client for some verification. We are planning on meeting with our client Monday.

5.9.3 Week 3

We submitted document changes this week and we plan on meeting our client on Monday.

5.9.4 Week 4

We met with our client and turned in our poster. We will be having a multi-group meeting before expo, with the other groups that work with our client.

5.9.5 Week 5

We met with our client, and sent off our poster. Now we are just getting ready for expo.

5.9.6 Week 6

We met with our client and got info on the expo in class. We are ready for expo, and will work on some tweaks and documentation the following week.

We Make Physics Fun!!!

Tasked with transforming University worksheets into **web-based physical simulations**, our team was able to utilize Java Script to make the end product.

Why Bother Doing This?

- Universities often lack resources to visually show physical and mechanical interactions in the classroom
- Students grasp a better understanding of difficult concepts by learning through **interactive environments**

Concept Warehouse

- Educational website focusing on creating interactive learning opportunities for classrooms.
- Uses our 2D pendulum simulations, along with others, to replace in-class experiments.
- Allows teachers to see the knowledge retention of their students, through online quizzes.

Oregon State University

INTERACTIVE 2D MECHANICAL SIMULATIONS

Supporting **inquiry based learning** in Mechanical Engineering

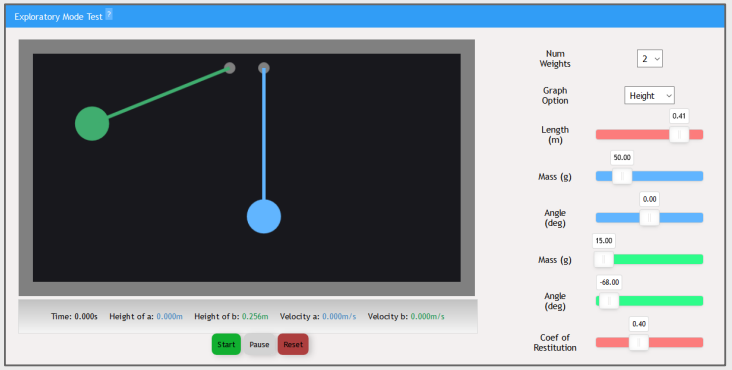


Figure 1: Exploratory Mode – Sandbox version of pendulum simulation

Key Features:

- Animations based on **actual physics** of a pendulum
- **Modifiable parameters** from user
- Real-time, measurement feedback.
- Real-time, graphical feedback.

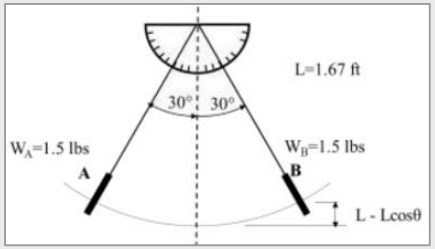


Figure 2: Given physics problem (Paper Version)

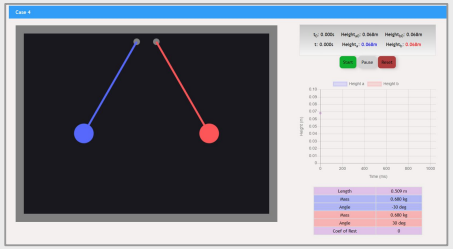


Figure 3: Conversion to a digital simulation (Online Version)

IMPLEMENTATION

The project utilizes different Java Script libraries for ease of completion

Matter.js – open source 2D physics engine for webpages

- Quick way to create physics animations

Gulp.js - toolkit for automating build tasks

Chart.js – JavaScript library to create automated interactive charts or graphs

- Prebuilt responsive graphs to easily integrate charts into webpage

HTML/CSS/Java Script – base used to create webpages

TEAM MEMBERS

Cameron Friel

- frielc@oregonstate.edu

Samuel Wilson

- wilsosam@oregonstate.edu

Kelli Ann Ulep

- ulepk@oregonstate.edu

PROJECT SPONSOR

- OSU School of Chemical, Biological, and Environmental Engineering
- Milo Koretsky –milo.koretsky@oregonstate.edu
- Tom Esktedt – Tom.Ekstedt@oregonstate.edu
- This work supports the National Science Foundation grant DUE 1821439. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

NSF

AIChE Education Division CONCEPT WAREHOUSE

7 PROJECT DOCUMENTATION

7.1 Running the Program on Concept Warehouse

The webpages are hosted on Concept Warehouse at https://jimi.cbee.oregonstate.edu/concept_warehouse/. To view the simulation one just needs an internet connection and a web browser. Concept Warehouse works in that each simulation is shown in order from Case 1-5 then exploratory mode. To view the simulation, students first predict what is going to happen in the simulation, given the initial state of the pendulum, and then they explain their answer. Students then activate the simulation and then they can press start and view the animation. A live graph update (synced with the simulation) is shown after starting, and live measurements of the height are also shown. Interactions included for the normal mode include the Start, Pause, and Reset button.

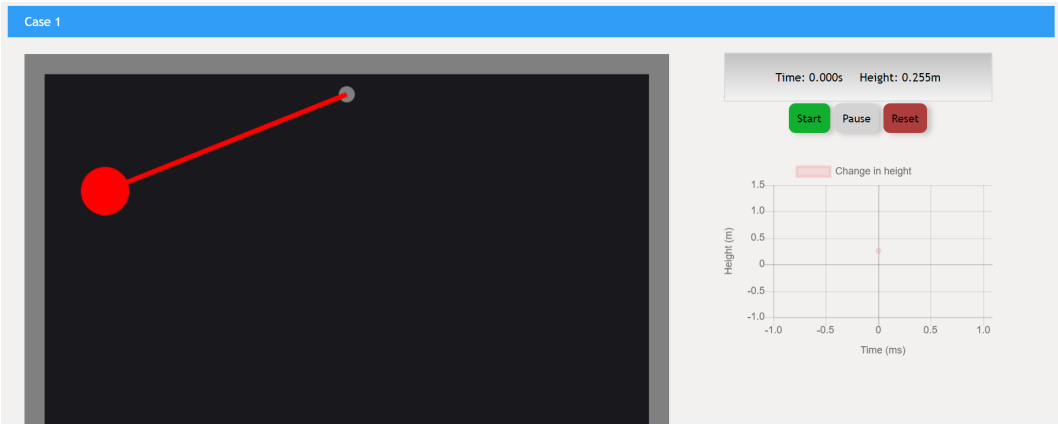


Fig. 1. Case 1 - initial screen

In Exploratory Mode, users adjust the initial state of the animation using the range sliders on the right - adjusting the number of weights, the graph to be displayed (height/angle/velocity), length, mass, starting angle, and the coefficient of restitution. Then users press start, and the live graph replaces the input sliders.

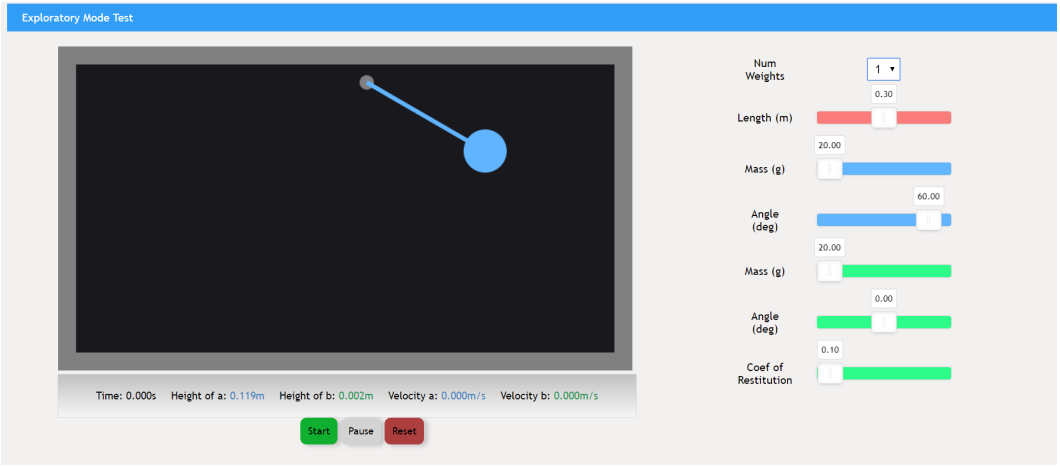


Fig. 2. Exploratory Mode - initial screen

7.2 Project Structure

7.2.1 Libraries

These JavaScript libraries are used in the webpage:

- Matter.js - Physics library used for the pendulum simulation
- Gulp.js - Used for automating build tasks
- Chart.js - Used for the graph
- noUiSlider - Used to create the range sliders in Exploratory mode

7.2.2 Modules

There are three basic modules in the project - State, Pendulum, Graph:

- The **State** module handles understanding the current state of the canvas, such as whether it is paused.
- The **Pendulum** module is in charge of keeping tack of the pendulum bodies and properties within the world.
- The **Graph** module handles the state of every graph generated in the project.

The sketch.js file in each folder uses these modules with a `require` statement and is the starting point for the webpage code.

7.3 Building the Webpage

This project uses certain packages to build by using Node as there are different modules used for this project. Make sure node is installed by typing `node --version` into the command line. After cloning the repository, run `npm install` on the command line to install dependencies.

7.3.1 Building With Gulp

This project utilizes Gulp to help automate build tasks. In order to install gulp enter

```
1    npm install -g gulp-cli
```

Once installed, to generate a build folder in the root of the project enter into the command line

```
1    gulp
```

This should generate folders case1-case5 and exploratory in the generated build folder.

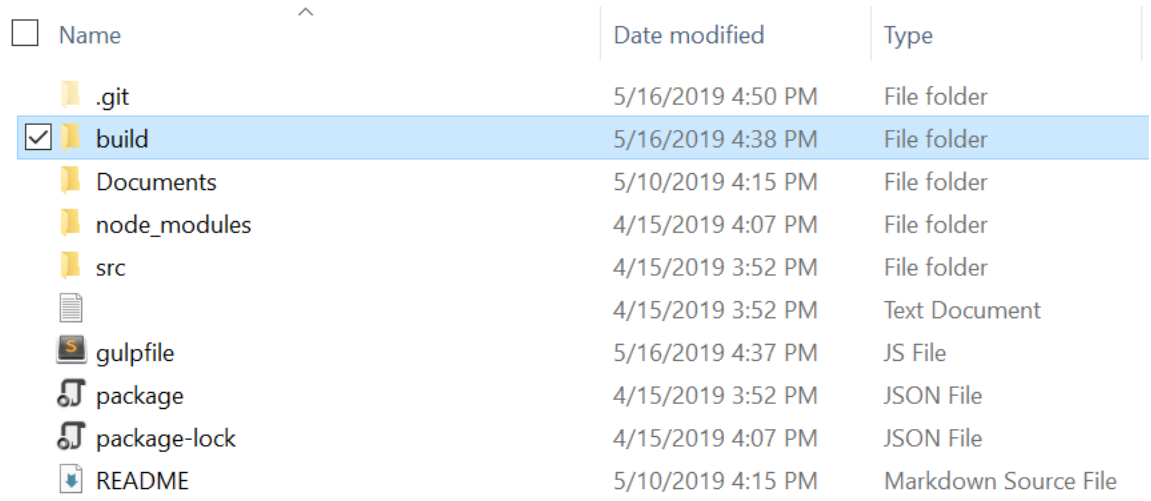


Fig. 3. Root Project Structure after gulp command

You can then navigate to one of the case folders in the generated build folder (i.e. **cd** `/build/case1`) and enter `start index.html` into the command line to view a specific case in the browser.

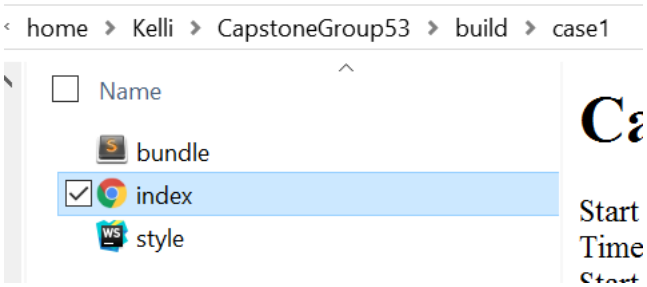


Fig. 4. Case folder in /build/case1

Alternatively, in order to build a single case into the build folder located in the root of the project, use the command:

```
gulp build --case [folder name]
```

For example: `gulp build --case case2`

7.3.2 Browserify

Alternatively, use browserify to manually generate the needed JavaScript file. Navigate to a specific case folder at /src/Pendulum/[case_folder_name] (i.e. /src/Pendulum/case2) and use the command

```
browserify sketch.js -o bundle.js.
```

After bundle.js is generated, use start index.html to start the webpage in the browser.

8 Recommended Technical Resources

The sites having the documentation for each of the libraries are helpful in understanding the project.

1) Matter.js - http://brm.io/matter-js/docs/
2) Matter.js introduction - https://blog.alexandergottlieb.com/matter-js-the-missing-tutorial-70aafc06b167
3) Chart.js - https://www.chartjs.org/docs/latest/
4) noUiSlider - https://refreshless.com/nouislider/

The demos on the Matter.js site are also very helpful in demonstrating the feature set that the library offers.

9 Conclusions and Reflections

9.1 Reflection - Kelli Ulep

1) **What technical information did you learn?** - I gained more experience in web development - using different JavaScript libraries/modules and having them work together - specifically matter.js and noUiSlider and having them interact. This object oriented style in JavaScript made the code more reusable and organized. This specific project also involved using modules in JavaScript with Node which is something I have not used past simple implementations for classes. I have not used any of these libraries before, and have not experienced building a web page from scratch to be used by people.
2) **What non-technical information did you learn?** - I gained more experience in creating software for a client rather than for a class or just being assigned a project for an internship. It is different since there is more freedom in designing the user interface keeping in mind that actual users will be using this webpage. Also since we were building it without existing project code, we had more freedom to pick our libraries we wanted to work with. Researching each library in depth and testing a simple part of it with the code, before fully integrating it into a project, is easier than just trying out one library and trying to make it work.
3) **What have you learned about project work?** - Be clear about the requirements from the beginning. Clients and people working with you can have a different interpretation from you, so you must make sure to communicate.
4) **What have you learned about project management?** - Having a version control that everyone can use is important. Having a set project flow is better for catching errors and keeping track of the tasks needing to be done - especially when most of the project is self directed. Staying organized with tasks, file sharing, and documents from the beginning helps in the end.
5) **What have you learned about working in teams?** - It is helpful to set team deadlines and make sure everyone is on the same page. Establishing from the beginning our process for submitting work and adding to the code was helpful in the long run. Being well versed in Git or some other type of version control is necessary when working in teams.
6) **If you could do it all over, what would you do differently?** Doing it over, I would probably experiment with more JavaScript libraries or just use JQuery in the animation. Using a physics engine was definitely easier for exploratory mode as it easily integrated with the other libraries. However, we did not forsee the air resistance problem in which the system lost energy with each swing. There are many implementations for the animation of a single pendulum online, but implementing the physics of two pendulums colliding from scratch required more complex formulas we could have investigated more. I also would have started to look into more of the simulations once we have mostly finished the pendulum one.

9.2 Reflection - Cameron Friel

1) **What technical information did you learn?** I gained a lot more experience in JavaScript, as this was the primary language used in the project. Specifically, I got a lot more used to ES6 JavaScript as they have now introduced modules and classes, so that you can modulate your projects. This was very helpful for us because we were able to create modules to avoid duplicating a lot of code over the different simulations. In order to bundle these files we decided to use Browserify, which uses the Common.js paradigm for including files. I also learned about Gulp.js, a toolkit for automating developments tasks. This library helped us quickly build our projects to update our changes among other things. I also got to mess around with my first physics engine getting to learn how the library worked and how all the different components went together. Lastly, I enhanced my knowledge of more making responsive web pages.
2) **What non-technical information did you learn?** Taking on this project got me more involved in a long term team setting, as this project spanned over the course of three whole terms. I also got to learn to be a lot more creative as the client gave a lot of freedom when it came to the design of the project.
3) **What have you learned about project work?** I learned that you must make sure that your expectations align with the client so that the product that comes out aligns with each party. I also learned that clear outlines go a long way.
4) **What have you learned about project management?** The single most beneficial tool for project management has been an active GitHub. Creating pull request with descriptions keeps a good history of the design decisions and features built into the project. Having bi weekly meetings helped keep everyone on track and aware of their responsibilities. It also provided idea generation sessions.
5) **What have you learned about working in teams?** I have learned that keeping good documentation can save a lot of time in the future. It will also benefit team members and other people affiliated with the project better understand how the project works. Also establishing deadlines helped keep everyone in the project on top of things.
6) **If you could do it all over, what would you do differently?** If I could start this project all over again I probably would not start out with a physics engine. The team I and thought that using a physics engine would help cut development time down by a lot since we would not be reinventing the wheel, however, due to bugs within the engine the simulations developed were not as accurate as we had hoped for. The major long standing issue is the air resistance calculation in the simulation. The client wished for a simulation with no energy lost so that in our pendulum cases the pendulum would move forever, but this ended up not being the case despite our best efforts to fix the issue. Instead, I would propose that we create a specific system for the use of pendulums. This would mean that the project is not modular for future simulations, but what would be displayed would be as accurate as possible.

9.3 Reflection - Samuel Wilson

1) **What technical information did you learn?** During this project, I learned about the technical process of creating physics simulations, such as the math behind real world physics, and how physics engines work. I also worked exclusively with Javascript to implement our project, which allowed me to better understand the capabilities of web-based applications.

2) **What non-technical information did you learn?** I learned more about the process of creating something for a client. Its important to be able to describe what is happening during the projects creation, to the client, in clear language. This allows them to make decisions easier, instead of being unsure of what the problem really is.
3) **What have you learned about project work?** Setting daily goals is important to stay productive. If given the chance to put a tasks off until tomorrow, I will. This goes together with team goals, where assigning low-level tasks becomes important, to keep people productive.
4) **What have you learned about project management?** When starting off on a project, it is important to create some form of design documentation to use as a reference while working on the project. During the project, timelines and task lists budget the time needed to complete everything that is outlined in the design document.
5) **What have you learned about working in teams?** Working in teams requires a constant stream of communication to be effective. Micro-goals have to be set and assigned to ensure that projects get done in a timely manner. Expectations are part of the communication, which also works to keep people on time.
6) **If you could do it all over, what would you do differently?** One of the main problems we ran into was imperfections in the physics engine we selected. While our client was ultimately satisfied with our end product, it is clear that if we had found the limitations of the engine beforehand, we would have changed our development plan. Specifically, we would have designed a custom engine to handle the physics of pendulums, despite it taking a longer time.

10 APPENDIX 1

This is a typical sketch file used to add a simulation to a web page. This is taken from the first simulation.

```
'use strict';

let State = require('../State.js');
let Pendulum = require('../Pendulum.js');
let Graph = require('../Graph.js');

const CANVAS_WIDTH = 800;
const CANVAS_HEIGHT = 600;

const PENDUMDULUM_HEIGHT_ID = 'pendulum-height';

const PTM = 634.773; // converts pixels to meters for calculations

let Engine = Matter.Engine;
let Render = Matter.Render;
let World = Matter.World;
let Bodies = Matter.Bodies;
let Body = Matter.Body;
let Constraint = Matter.Constraint;
let Events = Matter.Events;

let engine = Engine.create();

let pendulum = new Pendulum;

let render = Render.create({
    element: document.getElementById('canvas'),
    engine: engine,
    options: {
        width: CANVAS_WIDTH,
        height: CANVAS_HEIGHT,
        wireframes: false
    }
 });

let ctx = document.getElementById("chart").getContext('2d');

```

```
let plotInterval = null;

 let graphData = {
   datasets: [{
     label: 'Height',
     borderColor: 'rgba(255, 0, 0, 0.1)',
     backgroundColor: 'rgba(255, 0, 0, 0.1)',
     data: [{
       x: 0,
       y: 0.255
     }]
  }],
  xAxes: [{
    type: 'linear',
    position: 'bottom',
    ticks: {
      min: 0,
      max: 2000,
    },
    scaleLabel: {
      labelString: 'Time (ms)',
      display: true
    }
  }],
  yAxes: [{
    type: 'linear',
    position: 'left',
    ticks: {
      min: 0,
      max: 0.30,
    },
    scaleLabel: {
      labelString: 'Height (m)',
      display: true
    }
  }],
 };
```

```
77 /**
78   * Create world
79 */
80
81 createWorld(); // add bodies to canvas
82
83 Render.run(render); // allow for the rendering of frames of the world
84
85 renderLoop(); // renders frames to the canvas
86
87 Graph.createGraph(ctx, graphData); // add graph
88
89 /*
90   * Renders frames to send to the canvas
91 */
92
93 function renderLoop() {
94
95   if (State.getIsPausedFlag()) { // the world is paused
96     requestAnimationFrame(renderLoop); // render next frame
97   }
98   else {
99     Engine.update(engine, 1000 / 60); // update at 60 FPS
100    requestAnimationFrame(renderLoop); // render next frame
101  }
102 }
103
104 /*
105   * Sets up initial bodies of the world
106 */
107
108 function createWorld() {
109   World.add(engine.world, [
110     Bodies.rectangle(400, 0, 800, 50, { isStatic: true, render: {fillStyle: 'grey'} }),
111     Bodies.rectangle(400, 600, 800, 50, { isStatic: true, render: {fillStyle: 'grey'}
        }),
112     Bodies.rectangle(800, 300, 50, 600, { isStatic: true, render: {fillStyle: 'grey'}
        }),
113     Bodies.rectangle(0, 300, 50, 600, { isStatic: true, render: {fillStyle: 'grey'} })
```

```
114   ]);
115
116   pendulum.pendulumBody = Bodies.circle(100, 170, 30, { mass: 0.680389, frictionAir: 0,
          interia: Infinity, render: {fillStyle: '#FC5658'} }); // Light red
117
118   let protractor = Bodies.circle(400, 50, 10, { isStatic: true, render: {fillStyle: '
          grey'}});
119
120   World.add(engine.world, [pendulum.pendulumBody, protractor]);
121
122   pendulum.pendulumString = World.add(engine.world, Constraint.create({
123     bodyA: protractor,
124     bodyB: pendulum.pendulumBody,
125     length: 0,
126     render: {
127       strokeStyle: '#FC5658',
128       lineWidth: 6
129     }
130   }));
131
132   pendulum.pendulumStringLength = pendulum.calculateStringLength(protractor.position,
          pendulum.pendulumBody.position);
133 }
134
135 /**
136   * Sends a request to plot a coordinate every 100ms
137 */
138
139 function runPlotInterval() {
140   plotInterval = setInterval(function() {
141     Graph.addGraphData({ x: engine.timing.timestamp.toFixed(3), y: pendulum.
            pendulumHeight.toFixed(3) },0);
142   }, 100);
143 }
144
145 /**
146   * Stops the plot interval from running
147 */
148
```

```
149 function stopPlotInterval() {
150   clearInterval(plotInterval);
151 }
152
153 /*
154   * Pauses or unpauses the world from rendering
155 */
156 var pauseBtn = document.getElementById('pause-button');
157
158 pauseBtn.onclick = function() {
159   if (State.getSimulationRunning() === true) { // only allow pause and continue when the
        simulation is running
160     if (pauseBtn.value == "pause") {
161       pauseBtn.innerText = "cont.";
162       pauseBtn.value = "continue";
163       stopPlotInterval();
164     }
165     else {
166       pauseBtn.value = "pause";
167       pauseBtn.innerText = "Pause";
168       runPlotInterval();
169     }
170
171     State.setIsPausedFlag(!State.getIsPausedFlag());
172     State.onPause(render);
173   }
174 };
175
176 /*
177   * Starts running the simulation
178 */
179
180 document.getElementById('start-button').onclick = function() {
181   if (engine.timing.timestamp === 0) {
182     State.setIsPausedFlag(false);
183     State.onPause(render);
184     State.setSimulationRunning(true);
185     Body.applyForce(pendulum.pendulumBody, {x: pendulum.pendulumBody.position.x, y:
          pendulum.pendulumBody.position.y}, {x: 0.028, y: 0});
```

```
186     runPlotInterval();
187   }
188 };
189
190 /*
191   * Resets the world to its starting state
192 */
193
194 document.getElementById('reset-button').onclick = function() {
195   World.clear(engine.world);
196   createWorld();
197   engine.timing.timestamp = 0;
198   Graph.resetGraphData(graphData);
199   stopPlotInterval();
200   State.displayRunningTime(engine);
201   State.setSimulationRunning(false);
202   pendulum.pendulumAngle = pendulum.calculateAngle(pendulum.pendulumString.bodies[0].
        position, pendulum.pendulumBody.position);
203   pendulum.pendulumHeight = pendulum.calculatePenulumHeight(pendulum.
        pendulumStringLength / PTM, pendulum.pendulumAngle);
204   pendulum.displayPendulumHeight(PENDUMDULUM_HEIGHT_ID);
205
206   if (State.getIsPausedFlag() === false) {
207     State.setIsPausedFlag(true);
208     State.onPause(render);
209   }
210   else {
211     pauseBtn.value = "pause";
212     pauseBtn.innerText = "Pause" ;
213   }
214 };
215
216 // Updates UI before each update of the simulation
217 Events.on(engine, 'beforeUpdate', function(event) {
218   pendulum.pendulumAngle = pendulum.calculateAngle(pendulum.pendulumString.bodies[0].
        position, pendulum.pendulumBody.position);
219   pendulum.pendulumHeight = pendulum.calculatePenulumHeight(pendulum.
        pendulumStringLength / PTM, pendulum.pendulumAngle);
220   pendulum.displayPendulumHeight(PENDUMDULUM_HEIGHT_ID);
```

```
  State.displayRunningTime(engine);

  // Stop when speed is below 0.2
  if (pendulum.pendulumBody.speed <= 0.2 && pendulum.pendulumBody.speed !== 0){
    State.setIsPausedFlag(true);
    State.onPause(render);
    stopPlotInterval();
    State.setSimulationRunning(false);
  }
});

/**
  * Listens for whether the current browser tab is active or not
*/

document.addEventListener('visibilitychange', function() {
  if (!document.hidden) {
    if (State.getIsPausedFlag() === false) {
      runPlotInterval();
    }
  }
  else {
    stopPlotInterval();
  }
});
```

This is the State module that handles the current state of a simulation based on user feedback.

```
'use strict';

let State = (function() {
  let Render = Matter.Render;
  let _simulationRunning = false;
  let _isPausedFlag = true;

  return {
    /**
      * Updates the value of _isPausedFlag
      * @param {boolean} bool - the value to set the _isPausedFlag
    */

```

```
14    setIsPausedFlag: function(bool) {
15      _isPausedFlag = bool;
16    },
17
18    /**
19      * Returns whether the world is paused or not
20      * @returns {State} _isPausedFlag
21    */
22
23    getIsPausedFlag: function() {
24      return _isPausedFlag;
25    },
26
27    /**
28      * Updates the value of _simulationRunning
29      * @param {boolean} bool - the value to set _simulationRunning to
30    */
31
32    setSimulationRunning: function(bool) {
33      _simulationRunning = bool;
34    },
35
36    /**
37      * Returns whether the simulation is running or not
38      * @returns {bool} _simulationRunning
39    */
40
41    getSimulationRunning: function() {
42      return _simulationRunning;
43    },
44
45    /**
46      * Determines whether to continue rendering the world or not
47      * @param {Render} render - displays the world
48    */
49
50    onPause: function(render) {
51      if (_isPausedFlag) { // the world is paused
52        Render.stop(render);
```

```
      }
      else {
        Render.run(render);
      }
    },

    /**
      * Displays to the user the current running time of the simulation
    */

    displayRunningTime: function(engine) {
      let currentTime = (engine.timing.timestamp / 1000).toFixed(3);

      if (currentTime > 60) { // more than a minute has gone by
        let quotient = Math.floor(currentTime / 60); // minutes
        let remainder = (currentTime % 60).toFixed(3); // seconds

        document.getElementById('running-time').textContent = quotient + 'm ' +
            remainder + 's';
      }
      else {
        document.getElementById('running-time').textContent = currentTime + 's';
      }
    },
  };
})();

module.exports = State;
```

This is the graph module that handles the state of the graph of a simulation.

```
'use strict';

let Graph = (function() {
  let _graph = {};

  return {
    /**
      * Creates a new graph
      * @param {} reference - The reference to the html page
      * @param {} data - The graph data
```

```
11    */
12
13    createGraph: function(reference, data) {
14      let tempData = JSON.parse(JSON.stringify(data));
15
16      _graph = new Chart(reference, {
17      type: 'line',
18      data: {
19        datasets: tempData.datasets,
20      },
21      options: {
22        responsive: true,
23        maintainAspectRatio: false,
24        showLines: true,
25        scales: {
26          xAxes: tempData.xAxes,
27          yAxes: tempData.yAxes,
28        }
29      }
30    });
31  },
32
33  /**
34    * Adds points to the graph
35    * @param {} data - The graph data
36  */
37
38  addGraphData: function(data, dataset) {
39    _graph.data.datasets[dataset].data.push(data);
40
41    _graph.update();
42  },
43
44  /**
45    * Resets the chart data back to an empty state
46    * @param {} newData - The graph data
47  */
48
49  resetGraphData: function(data) {
```

```
    let tempData = JSON.parse(JSON.stringify(data));

    _graph.config.data = tempData;
    _graph.options.scales.xAxes[0].scaleLabel.labelString = tempData.xAxes[0].scaleLabel
        .labelString;
    _graph.options.scales.yAxes[0].scaleLabel.labelString = tempData.yAxes[0].scaleLabel
        .labelString;
    _graph.update();
  },
};
})();

module.exports = Graph;
```

This is the pendulum module that handles the state of a pendulum within a simulation.

```
'use strict';

/**
  * @class Pendulum
 * The Pendulum module is in charge of keeping tack of the pendulum bodies and
    properties within the world.
*/
const DEG_TO_RAD = Math.PI / 180;

class Pendulum {


  constructor() {
    this._pendulumBody = null;
    this._pendulumAngle = null;
    this._pendulumHeight = null;
    this._pendulumString = null;
  }

  /**
    * Sets the body of the pendulum
  */

  set pendulumBody(body) {
    this._pendulumBody = body;
```

```
25  }
26
27  /**
28    * Returns the body of the pendulum
29    * @returns {Bodies} _pendulumBody
30  */
31
32  get pendulumBody() {
33    return this._pendulumBody;
34  }
35
36  /**
37    * Sets the body of the string
38    * @returns {Bodies} _pendulumString
39  */
40
41  set pendulumString(string) {
42    this._pendulumString = string;
43  }
44
45  /**
46    * Returns the body of the string
47    * @returns {Body} _pendulumString
48  */
49
50  get pendulumString() {
51    return this._pendulumString;
52  }
53
54  /**
55    * Calculates the arctangent of a line given two coordinates
56    * @param {Int} firstPoint - the first coordinate given as (x, y) as its datamembers
57    * @param {Int} secondPoint - the second coordinate given as (x, y) as its
        datamembers
58    * @returns {Int} angle - the angle of the line
59  */
60
61  calculateAngle(firstPoint, secondPoint) {
62    let angle = Math.round(Math.abs(Math.atan2(firstPoint.y - secondPoint.y, firstPoint.
```

```
      x - secondPoint.x) * 180 / Math.PI));
63
64    return Math.abs(angle - 90);
65  }
66
67  /**
68    * Calculates the length of the pendulum's string
69    * @param {Vector} firstPoint - the first coordinate given as (x, y) as its
          datamembers
70    * @param {Vector} secondPoint - the second coordinate given as (x, y) as its
          datamembers
71    * @returns {Int} length
72  */
73
74  calculateStringLength(firstPoint, secondPoint) {
75    return Math.hypot(firstPoint.x - secondPoint.x, firstPoint.y - secondPoint.y);
76  }
77
78  /**
79    * Calculates the height of the pendulum in meters
80    * @param {Int} length - the length of the pendulum
81    * @param {Int} angle - the angle of the pendulum
82    * @returns {Int} height - the height of the pendulum
83  */
84
85  calculatePenulumHeight(length, angle) {
86    return length * (1 - Math.cos(angle * DEG_TO_RAD));
87  }
88
89  /**
90    * Updates the angle of the pendulum to the user
91  */
92
93  displayPendulumAngle() {
94    document.getElementById('pendulum-angle').textContent = 'Angle: ' + this.
          pendulumAngle;
95  }
96
97  /**
```

```
98    * Updates velocity of angle
99    * @param {String} id
100   */
101
102   displayVelocity(id) {
103      document.getElementById(id).textContent = this.pendulumBody.speed.toFixed(3) + 'm/
          s';
104   }
105
106 /**
107   * Updates the height of the pendulum to the user
108   * @param {String} id - a DOM id
109 */
110
111 displayPendulumHeight(id) {
112   document.getElementById(id).textContent = this.pendulumHeight.toFixed(3) + 'm';
113 }
114
115 /**
116   * Updates the value of _pendulumAngle
117   * @param {Int} angle - the current angle of the pendulum
118 */
119
120 set pendulumAngle(angle) {
121   this._pendulumAngle = angle;
122 }
123
124 /**
125   * Returns the angle of the pendulum
126   * @returns {Int} _pendulumAngle
127 */
128
129 get pendulumAngle() {
130   return this._pendulumAngle;
131 }
132
133 /**
134   * Updates the value of _pendulumHeight
135   * @param {Int} height - the current height of the pendulum from the bottom of the
```

```
      canvas
136   */
137
138   set pendulumHeight(height) {
139     this._pendulumHeight = height;
140   }
141
142   /**
143     * Returns the height of the pendulum measured from the bottom of the canvas to the
          bottom of the pendulum
144     * @returns {Int} _pendulumHeight
145   */
146
147   get pendulumHeight() {
148     return this._pendulumHeight;
149   }
150
151   /**
152     * Returns the velocity of the pendulum
153     * @returns {Int}
154   */
155
156   get pendulumVelocity() {
157     return this._pendulumBody.speed;
158   }
159
160   /**
161     * Updates the value of the length of the string
162     * @param {Int} length - the length of the string
163   */
164
165   set pendulumStringLength(length) {
166     this.pendulumString.length = length;
167   }
168
169   /**
170     * Returns the length of the pendulum's string
171     * @returns {Int}
172   */
```

```
173
174   get pendulumStringLength() {
175     return this.pendulumString.length;
176   }
177
178   get mass() {
179     return this.pendulumBody.mass;
180   }
181 }
182
183 module.exports = Pendulum;
```

This is the gulp file used to automate build tasks in the project. In order to add a new folder to build just add the folder's name to the array named "folders".

```
1  let gulp = require('gulp');
2  let terser = require('gulp-terser');
3  let clean = require('gulp-clean');
4  let fs = require('fs');
5  let browserify = require('browserify');
6  let source = require('vinyl-source-stream');
7
8  let folders = ['case1', 'case2', 'case3', 'case4', 'case5', 'case6', 'exploratory'];
9
10 /**
11   * Minifies all js files in the build folder
12 */
13
14 gulp.task('minify', () => {
15   return new Promise((resolve, reject) => {
16     try {
17       for (let i = 0; i < folders.length; i++) {
18         fs.access(`./build/${folders[i]}`, (error) => {
19           if (error) {
20             console.log(`${folders[i]} folder does not exist. Skipping...`);
21           }
22           else {
23             gulp.src(`./build/${folders[i]}/*.js`).pipe(terser()).pipe(gulp.dest(`./
                 build/${folders[i]}`), {mode: 0777});
24           }
25         });
```

```
26        }
27
28        resolve();
29      }
30      catch (error) {
31        console.log('Script could not be run: ' + error);
32        reject();
33      }
34    });
35  });
36
37  /**
38    * Builds all pendulum cases to the build folder
39    */
40
41  gulp.task('build-all', () => {
42    return new Promise((resolve, reject) => {
43      try {
44        for (let i = 0; i < folders.length; i++) {
45          browserify(`./src/Pendulum/${folders[i]}/Sketch.js`).bundle().pipe(source('
              bundle.js')).pipe(gulp.dest(`./build/${folders[i]}`), {mode: 0777});
46          gulp.src(`./src/Pendulum/${folders[i]}/**/*.css`).pipe(gulp.dest(`./build/${
              folders[i]}`), {mode: 0777});
47          gulp.src(`./src/Pendulum/${folders[i]}/**/*.html`).pipe(gulp.dest(`./build/${
              folders[i]}`), {mode: 0777});
48        }
49
50        resolve();
51      }
52      catch (error) {
53        console.log('Script could not be run: ' + error);
54        reject();
55      }
56    });
57  });
58
59  /**
60    * Builds a single pendulum case to the build folder
61    * Takes a command line argument --case [case] where case is a folder name
```

```
62 */
63
64 gulp.task('build', () => {
65   return new Promise((resolve, reject) => {
66     try {
67       if (process.argv.indexOf('--case') !== -1) {
68         let folder = process.argv[process.argv.indexOf('--case') + 1];
69
70         browserify(`./src/Pendulum/${folder}/Sketch.js`).bundle().pipe(source('bundle.js
               ')).pipe(gulp.dest('./build'), {mode: 0777});
71         gulp.src(`./src/Pendulum/${folder}/**/*.css`).pipe(gulp.dest('./build'), {mode:
               0777});
72         gulp.src(`./src/Pendulum/${folder}/**/*.html`).pipe(gulp.dest('./build'), {mode:
                0777});
73       }
74       else {
75         console.log('Error: The folder to build was not specified. Enter the command
               gulp build --case [Your Case].');
76       }
77
78       resolve();
79     }
80     catch (error) {
81       console.log('Script could not be run: ' + error);
82       reject();
83     }
84   });
85 });
86
87 /**
88   * Cleans the build directory
89 */
90
91 gulp.task('clean', () => {
92   return new Promise((resolve, reject) => {
93     try {
94       fs.access('./build', (error) => {
95         if (error) {
96           console.log('Error: ./build does not exist.');
```

```
97          }
98          else {
99            gulp.src('./build', {read: false}).pipe(clean(), {mode: 0777});
100         }
101       });
102
103       resolve();
104     }
105     catch (error) {
106       console.log('Script could not be run: ' + error);
107       reject();
108     }
109   });
110 });
111
112 gulp.task('default', gulp.series('build-all'));
```