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 — 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.

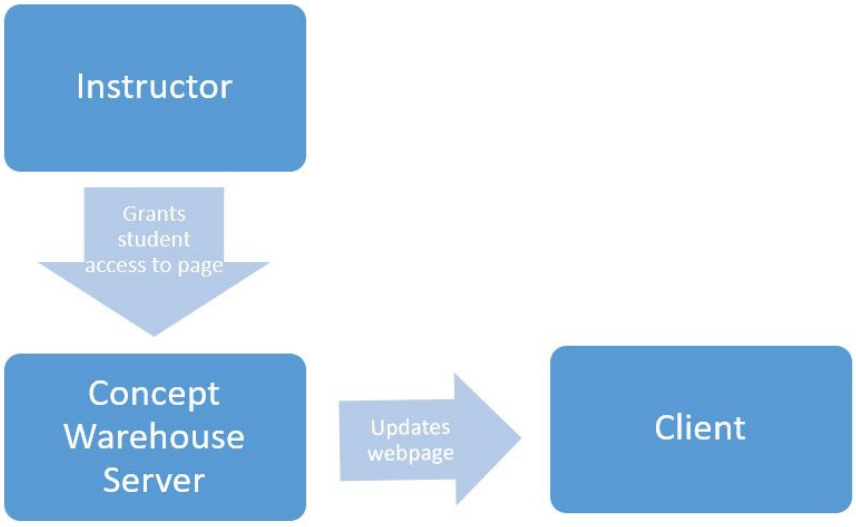

Fig. 1. Hosting Diagram

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.

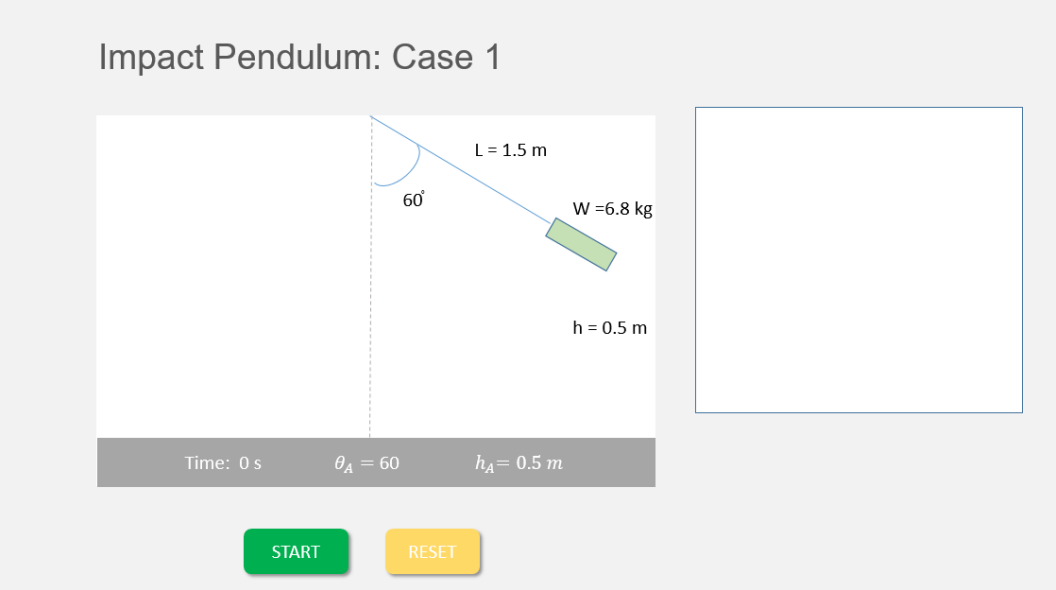

Fig. 2. Pendulum Case 1 - Before animation UI

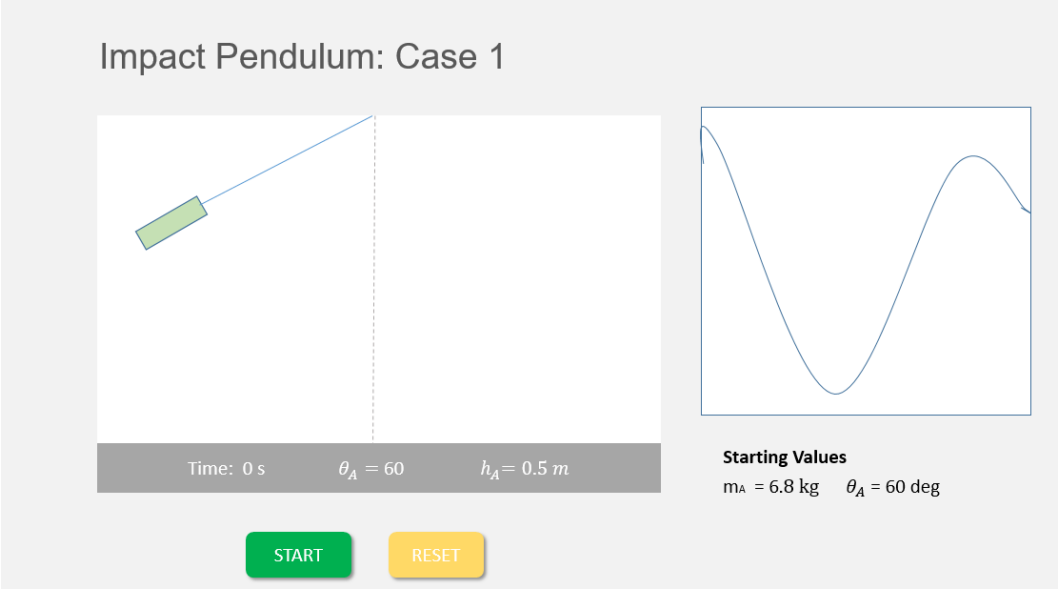

Fig. 3. Pendulum Case 1 - After animation completion UI

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.

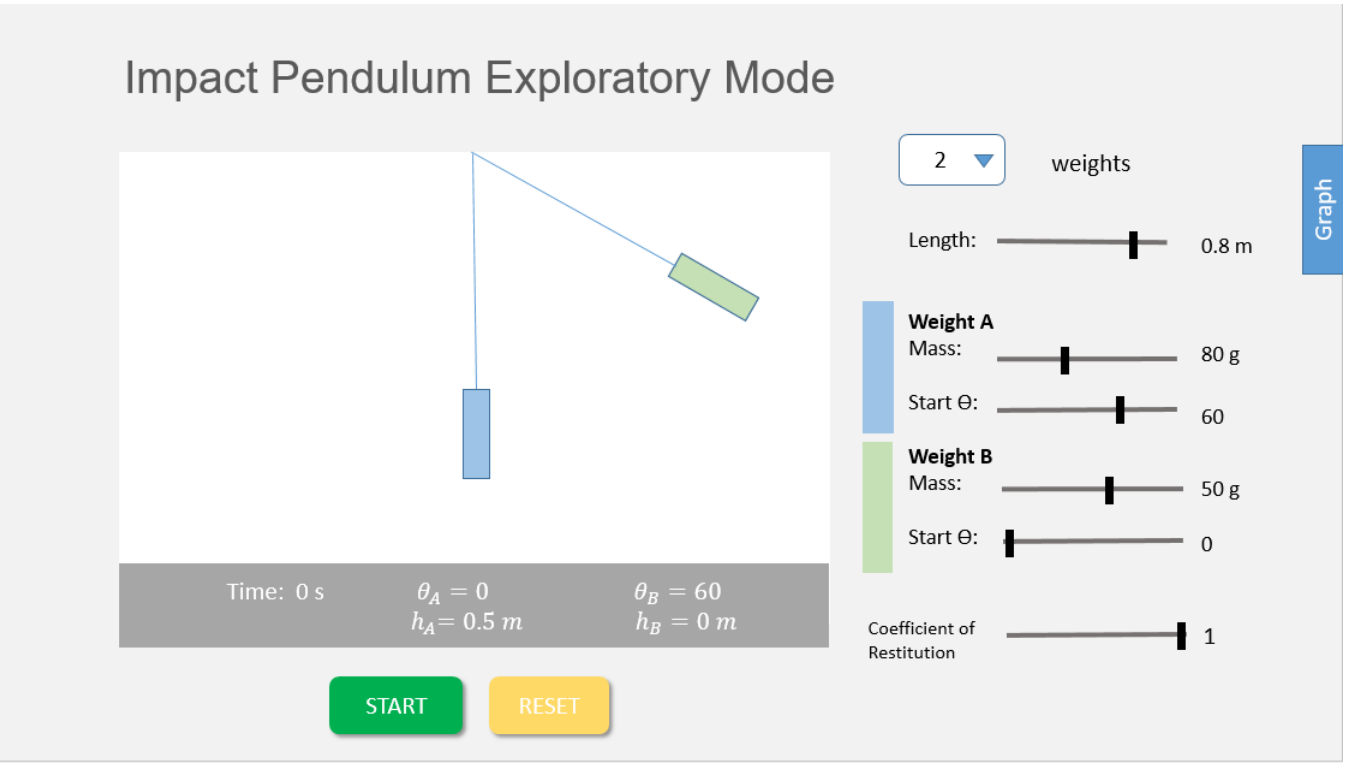

Fig. 4. Exploratory Mode Start Screen

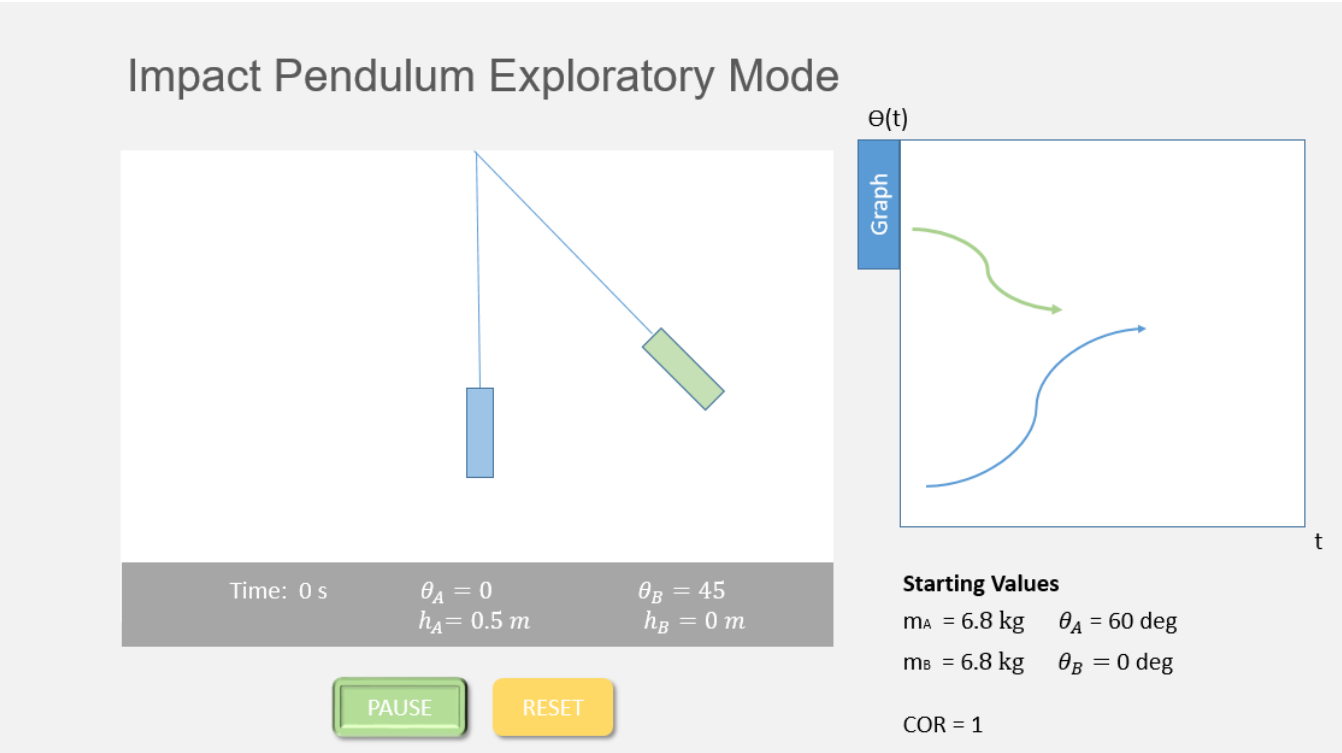

Fig. 5. Exploratory Mode Mid Animation

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.