# Software Requirements Specification (SRS)
# Project PhysicsGame


**Team:** **Group 5**
**Authors:** **Kyle Marcoux, Mit Bailey, Faycal Fedad, Jamison MacFarland, and Ian Sodersjerna**
**Customer:** **N/A**
**Instructor:** **Dr. James Daly**

# 1 Introduction

This document will provide all of the information about PhysicsGame. This covers the project context, audience description, requirements, modeling diagrams, and prototype information.

**Sections:**

- Section 1 will cover the basics of the project. It will go over the purpose of the document, as well as the audience for the document. It will also go over the scope of the project, identifying the product produced and its intended goal. Finally it will cover definitions for terms used in the document, as well as how the rest of the document is organized.
- Section 2 will cover the specifications of the project. It will start with the perspective for how the final product will be used, and cover any constraints on the project. It will then cover the function and goals of the final product's usage. Finally it covers the user base it is meant for, and explains the characteristics of the intended average user.
- Section 3 will cover the requirements of the project. It will be in hierarchical format, with the most important requirements coming first.
- Section 4 is multiple diagrams for how the software will work internally. It has a use-case diagram, a sequence diagram for every case, a class diagram, and a state diagram.
- Section 5 is the prototype section. This will cover the current prototype, scenarios for how the software will be used, and include screenshots of the current prototype.

## 1.1 Purpose

This software requirements specification document (SRS) is intended to describe the product, also referred to as PhysicsGame (working title), the project, or the game. This SRS document will describe what the product is intended to be, how the game will work, and what its intended functionality is, including requirements. Provided is a detailed set of requirements for the game explaining the purpose and features.

This document is intended for members of the development team to be used as a tool to establish and maintain a working, complete, and consistent understanding of the product, including how it is expected to perform and what its intended scope of functionality is. The document is also intended for the commissioning client, Dr. James Daly, for his review. This document should enable Dr. Daly to understand all the aforementioned items.

## 1.2 Scope

Software products which will be produced include the Unity engine-based educational video game PhysicsGame. The product's application includes the enhancement of understanding of physics concepts for use by students, educators, and parents in the home and school domains. The game is programmed in C# using the Unity game engine and is hosted on GitHub.

By providing the user with the ability to interact with a realistic and directed (AKA 'guided') physics environment, the user will develop an intuitive understanding of the causes and effects of such a system, as well as some of the basics of the underlying mathematics behind it. The system will be presented in a simplified and guided manner.

The product is not intended to teach complex nor advanced physics topics or concepts. PhysicsGame is also not intended to be challenging, but rather a guided, casual way to expand one's basic and beginner physics knowledge.

## 1.3 Definitions, acronyms, and abbreviations

*The product; The game, The project, PhysicsGame* – The Unity engine-based educational physics game which will be produced.

*User, player* – Whomever is playing the game.

*Vehicle* – The object the player will move around (may be as complex as a car, or as simple as a movable shape).

*Unity Engine* – A game engine in which video games are constructed; in this case, the game engine employed by PhysicsGame.

*SRS* - Software Requirements Specification document, this document.

*AKA* - Acronym for 'also known as.'

*IEEE* - Institute of Electrical and Electronics Engineers

## 1.4 Organization

The rest of this SRS document contains descriptions of the product, describing what the product is intended to be, how the game will work, and what its intended functionality is, including requirements. Provided is a detailed set of requirements for the game explaining the purpose and features. Also included are sequence and use case diagrams explaining how the product will function. Finally, screenshots have been added to show what the user will see upon entering the game.

The SRS document is structured in standard IEEE format. Below is the organizational structure.

1. Intro
    - 1.1 Purpose
    - 1.2 Scope
    - 1.3 Definitions
    - 1.4 Organization
2. Overall Description
    - 2.1 Product Perspective
    - 2.2 Product Functions
    - 2.3 User Characteristics
    - 2.4 Constraints
    - 2.5 Assumptions
3. Specific Requirements
4. Modeling Requirements
5. Prototype information

# 2 Overall Description

### Summary of Section 2

**2.1 Perspective**: The context for the project.

**2.2 Functions**: The goal of the project.

**2.3 User Characteristics**: The intended users of the product.

**2.4 Constraints**: Limitations on the project.

**2.5 Assumptions**: Things that are assumed for the project.

## 2.1 Product Perspective

PhysicsGame is intended to bridge the gap between formal in-school learning and intuitive understanding. The game can be deployed either in a classroom by an educator, at home by a parent, or by a student themselves in their spare time. All they require is a working computer setup. The game should allow the user to expand their knowledge of physics-based systems, incorporating some simple mathematics – which they may have learned in the classroom. Students will leave this game feeling more confident in physics.

Design Constraints:

- Education-based game, meaning it will focus on teaching physics to the user.
- While being focused on education, it cannot get in the way of the user having fun
- Users must be able to easily understand how to play the game so they can focus on learning.

Hardware Constraints:

- Supports mouse and keyboard
- Supports low-end desktop computers

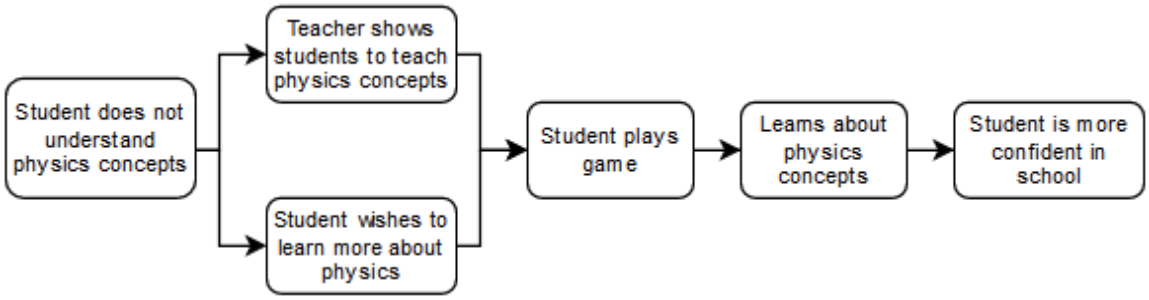
Software Constraints:

- Programmed and built using the Unity Engine
- Targeted for deployment on Windows machines.

## 2.2 Product Functions

The primary function of the game is to educate children grade 4 – 6 on basic physics concepts. It will do so using a game where the user tries to get a vehicle to a goal, using the game's lessons on basic physics concepts. The goal is to show users basic physics concepts, and have the game reinforce their learning by letting them put what they have learned into practice.

**Goal Diagram:**




## 2.3 User Characteristics

The targeted demographic is anyone with access to a computer capable of running the game and meeting the basic requirements assumed in section 2.4. No prior special physics expertise is expected nor required, however, the user is expected to understand basic mathematics, cause and effect, and have the educational attainment equivalent to a child aged in equivalence with that of a student attending American grades 4 - 6.

**Expectations of the user:**

- Child, possibly student, aged equivalent to those in US grades 4 through 6.
- Little to no understanding of physics except basic concepts and a general awareness.
- Basic computer operation abilities.

## 2.4 Constraints

**Regulatory Policies**

- Must be Educational

**Hardware Limitations**

- Developed for low-end Desktop Computers

**Interface Limitations**

- Works on website and with Github for version control

**Reliability requirements**

- Must be stable with little to no software errors

**PhysicsGame has no safety-critical properties.**

## 2.5 Assumptions and Dependencies

Assumptions of the user:

- Working computer
- Functioning Windows 7 (or newer) operating system.
- Working monitor, mouse and keyboard.
- The User's computer is capable of running basic video games for at least 15 minutes at a time.
- The User is able to play games for at least 15 minutes at a time.
- User is a child, possibly a student, aged equivalent to those in US grades 4 through 6.
- User has an interest in learning physics.
- User is capable of operating a computer with a mouse and keyboard.
- User has basic video game experience or ability.

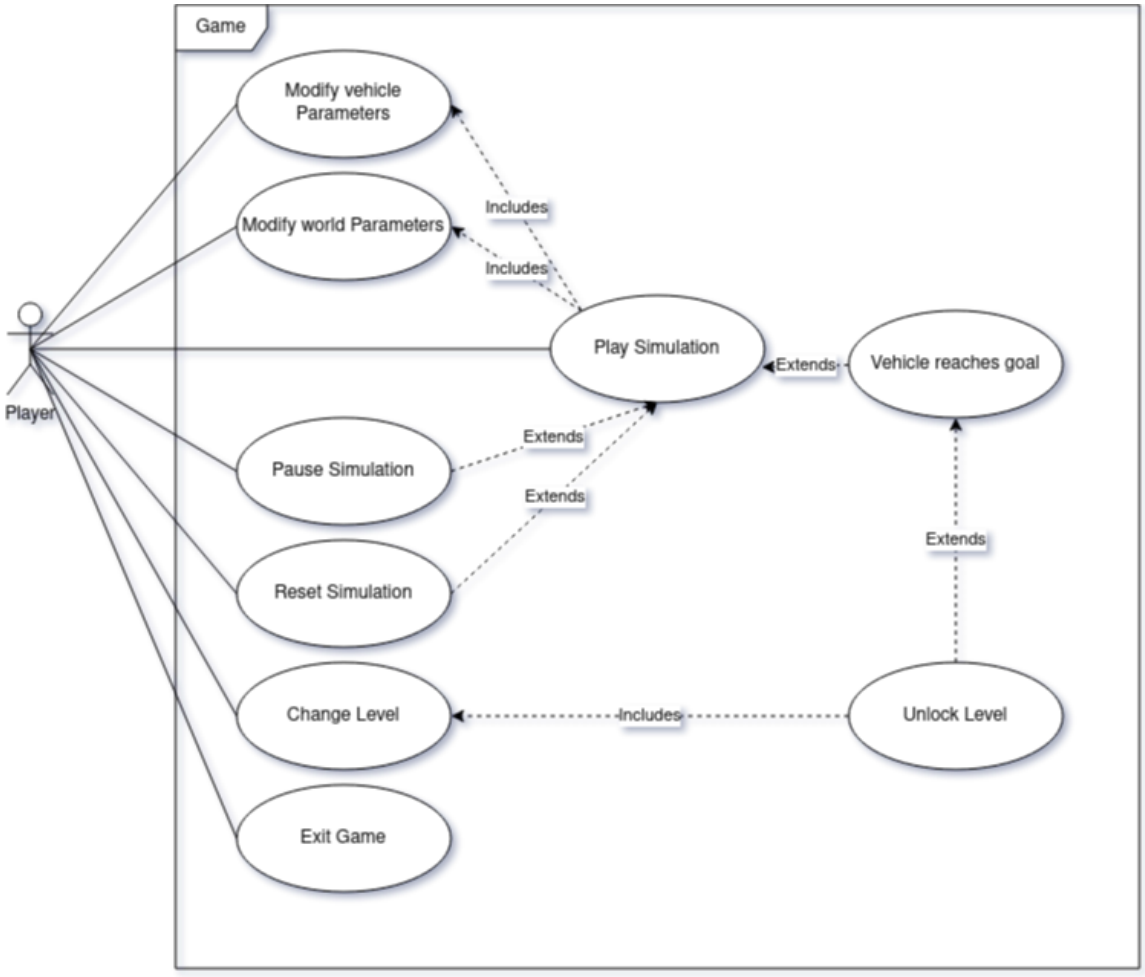
## 2.6 Apportioning of Requirements

**This project does not have any requirements that are beyond the scope of the project.**

# 3 Specific Requirements

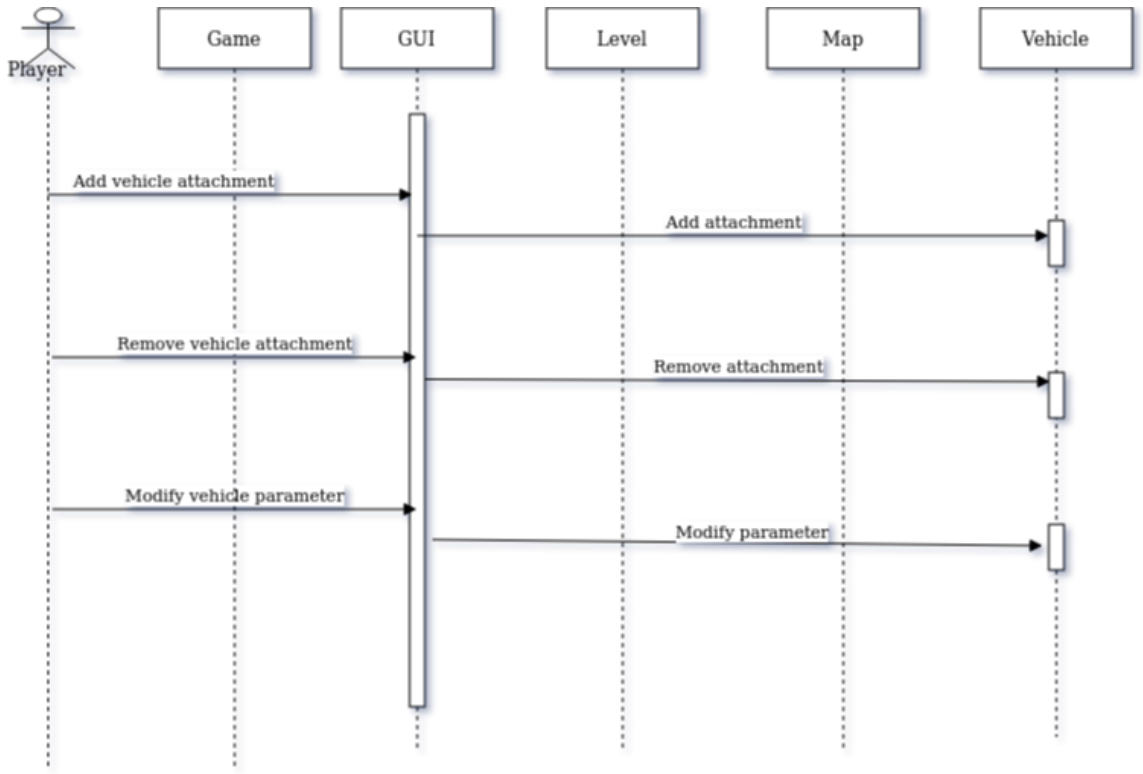
1. Must be interactive and responsive to user
2. Must be suitable for students from 4th-6th
    - 2.1 Easy to understand core concepts
        - 2.1.1 Gravity
        - 2.1.2 Friction
        - 2.1.3 Acceleration
3. Must have an educational component
    - 3.1 Teach the user each concept they need to complete the level
    - 3.2 Pop-up lesson available for each concept
4. Must be constrained to 2D physics model
5. Must be physics-oriented
6. Must have user-controlled vehicle
7. Must have variable parameters on simulation
    - 7.1 Must be able to tweak parameters of vehicle
    - 7.2 Must be able to tweak parameters of world
8. Must have responsive audio
9. Must have at least two levels
    - 9.1 Levels must have win conditions
    - 9.2 Levels must have failure conditions
    - 9.3 Must have difficulty get progressively harder with each level
    - 9.4 Must have unlockable tools and parts
10. Must have menu functionality
    - 10.1 Must be able to start level
    - 10.1 Must be able to pause level
    - 10.1 Must be able to reset level
    - 10.1 Must be able to quit game
    - 10.1 Must be able to change level
    - 10.1 Must be able to able to control volume
11. Must provide the physics lessons to understand the levels
    - 11.1 Must have tool tips
    - 11.2 Must have hints on failure condition

# 4 Modeling Requirements



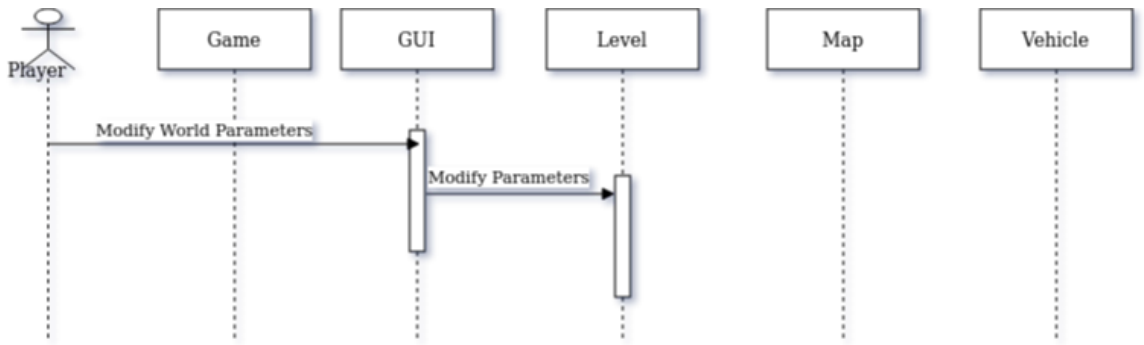

*Use Case Diagram. Standard UML notation.*

| Use Case Name: | Modify Vehicle Parameters |
|---|---|
| Actors: | Player |
| Description: | Player uses UI to add, remove, or change vehicle attachments and attributes. |
| Type: | Function |
| Includes: | |
| Extends: | |
| Cross-refs: | |
| Uses cases: | |



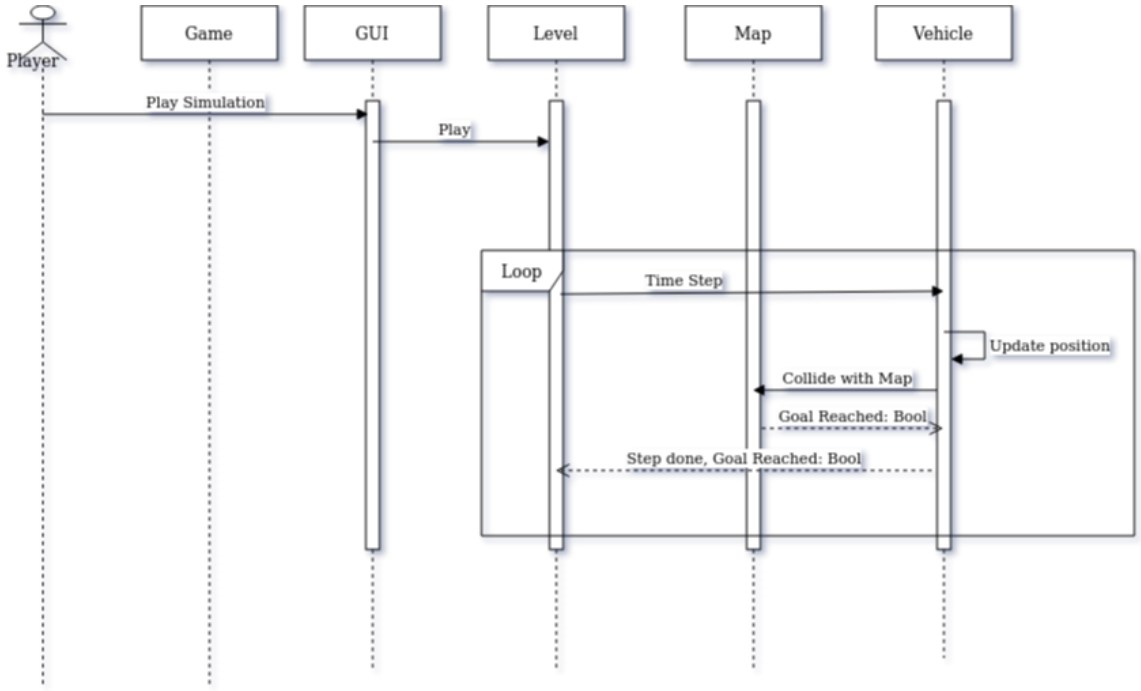

*Modify Vehicle Parameters sequence diagram.*

| Use Case Name: | Modify World Parameters |
|---|---|
| Actors: | Player |
| Description: | Player uses UI to change physics parameters within the in-game world. |
| Type: | Function |
| Includes: | |
| Extends: | |
| Cross-refs: | |
| Uses cases: | |



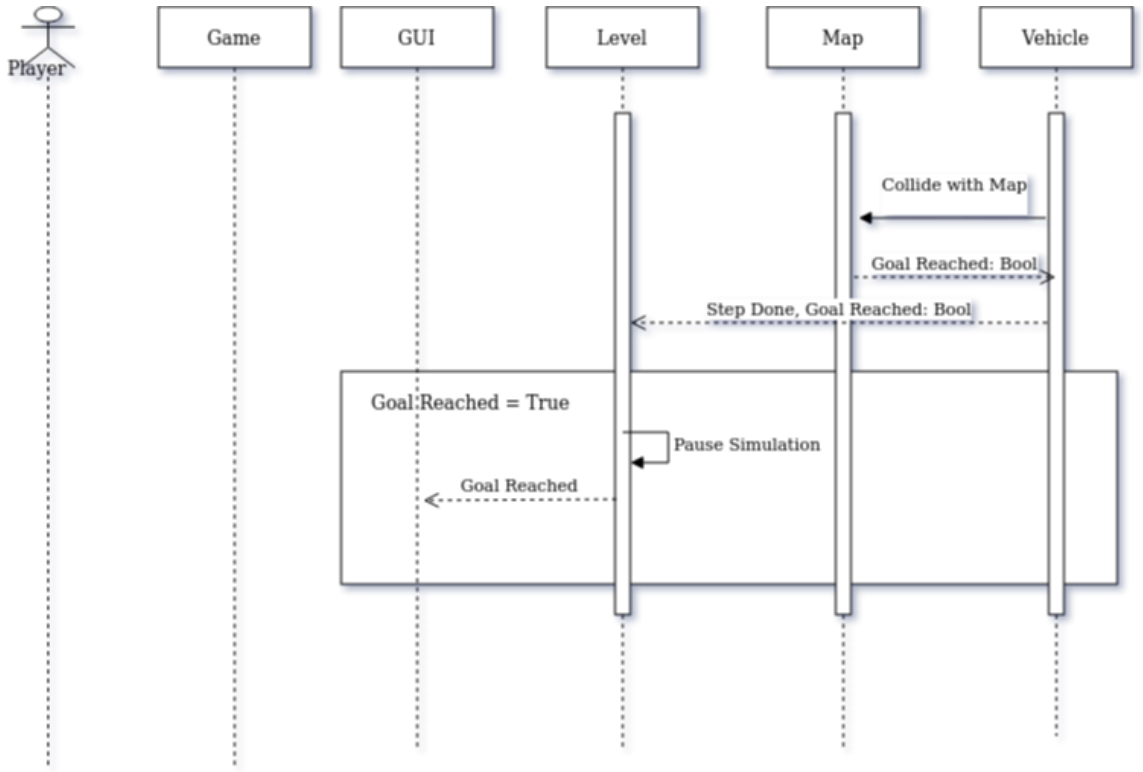

*Modify World Parameters sequence diagram.*

| | |
|---|---|
| Use Case Name: | Play Simulation |
| Actors: | Player |
| Description: | Player uses UI to begin the physics simulation. |
| Type: | Function |
| Includes: | Modify Vehicle Parameters, Modify World Parameters |
| Extends: | |
| Cross-refs: | |
| Uses cases: | |



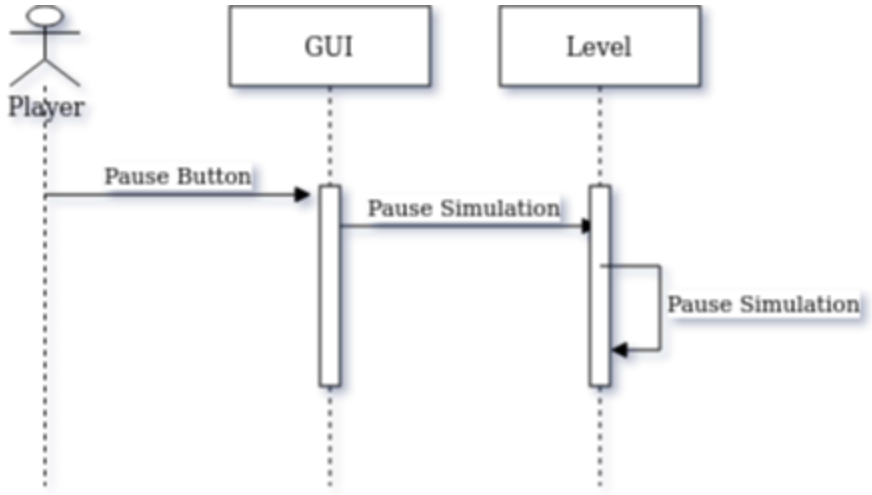

*Play Simulation sequence diagram.*

| | |
|---|---|
| Use Case Name: | Vehicle reaches goal |
| Actors: | |
| Description: | Player vehicle successfully fulfils the level's goal condition (reaches goal location). |
| Type: | Function |
| Includes: | |
| Extends: | Play Simulation |
| Cross-refs: | |
| Uses cases: | |



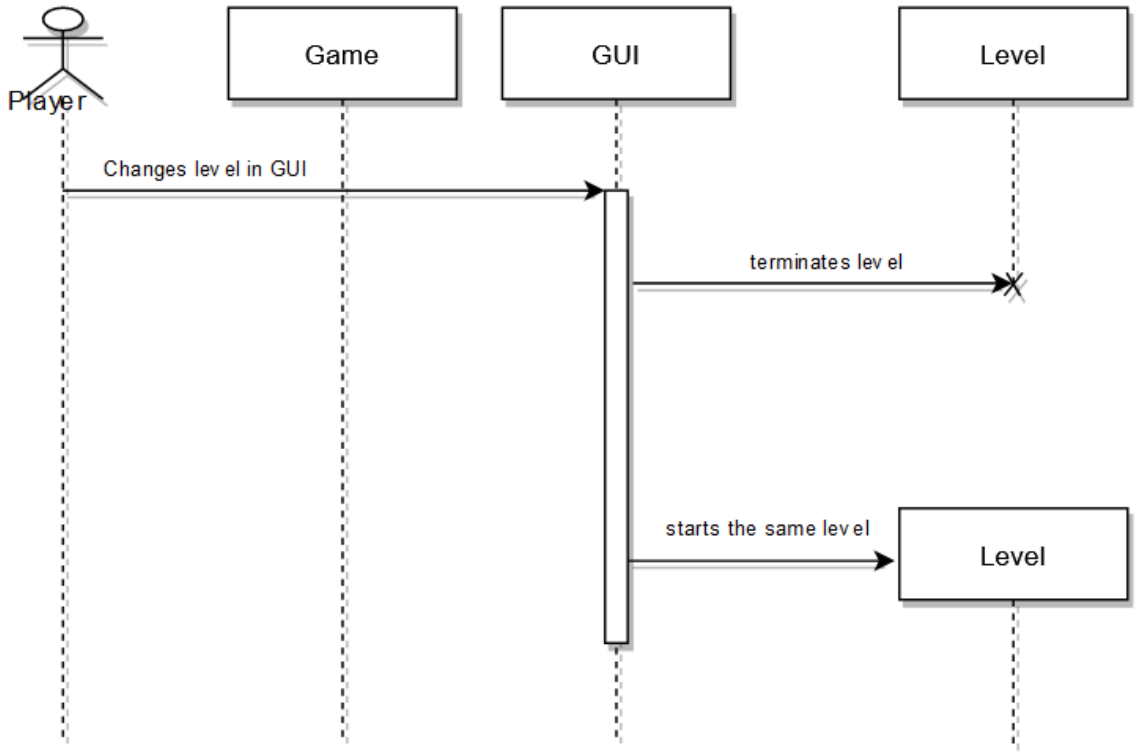

*Vehicle Reaches Goal sequence diagram.*

| | |
|---|---|
| Use Case Name: | Pause Simulation |
| Actors: | Player |
| Description: | Player uses UI to pause the physics simulation |
| Type: | Function |
| Includes: | |
| Extends: | Play Simulation |
| Cross-refs: | |
| Uses cases: | |



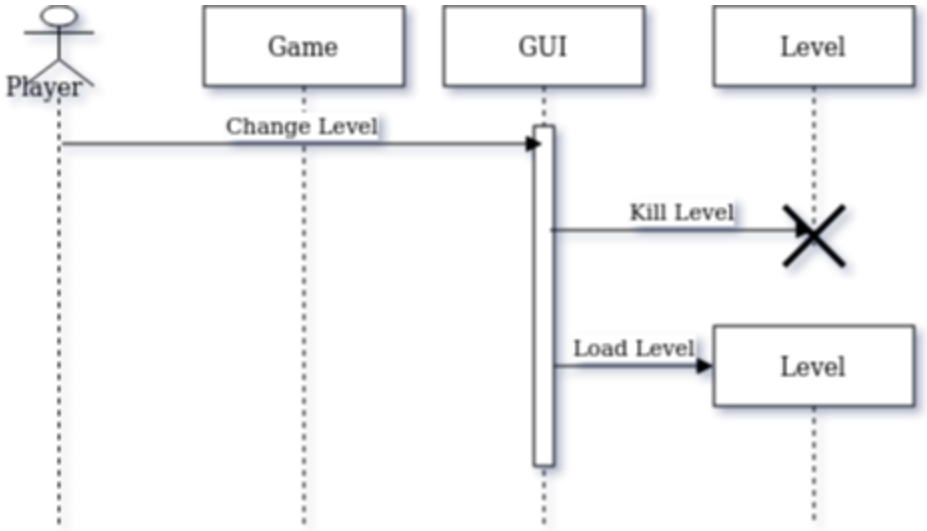

*Pause Simulation sequence diagram.*

| Use Case Name: | Reset Simulation |
|---|---|
| Actors: | Player |
| Description: | Player uses UI to reset the physics simulation, moving the vehicle back to the start location and removing progress made since the last start. |
| Type: | Function |
| Includes: | |
| Extends: | Play Simulation |
| Cross-refs: | |
| Uses cases: | |



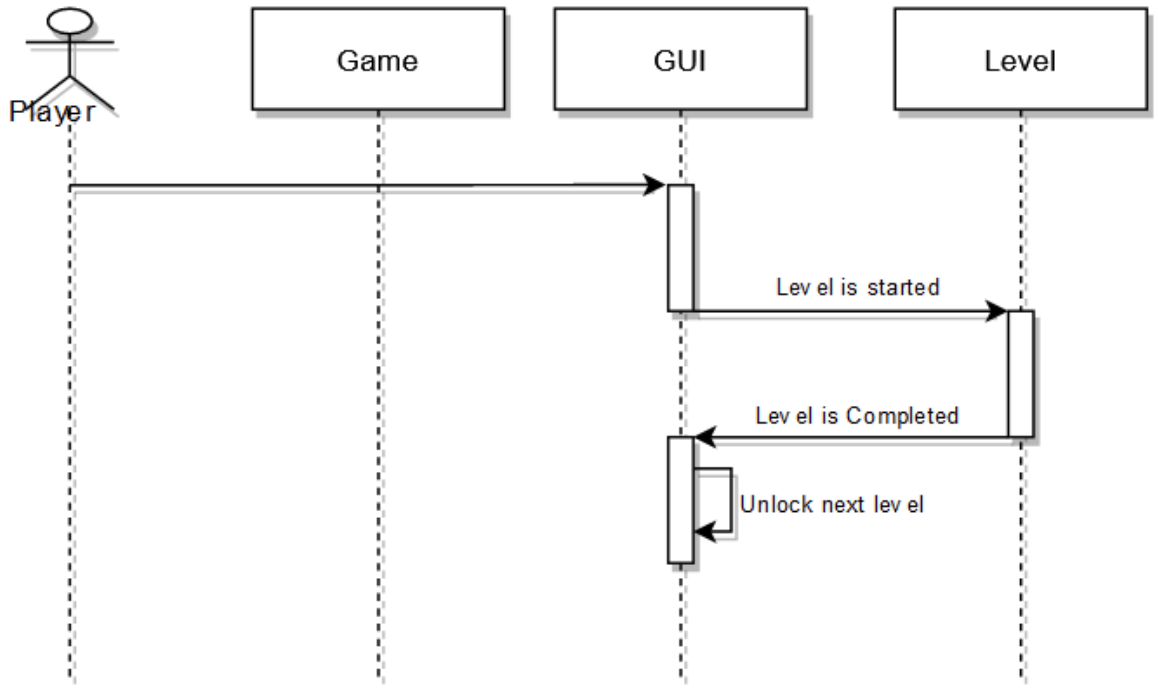

*Reset Simulation sequence diagram.*

| | |
|---|---|
| Use Case Name: | Change Level |
| Actors: | Player |
| Description: | Player uses the UI to change the current game level. |
| Type: | Function |
| Includes: | |
| Extends: | |
| Cross-refs: | |
| Uses cases: | |



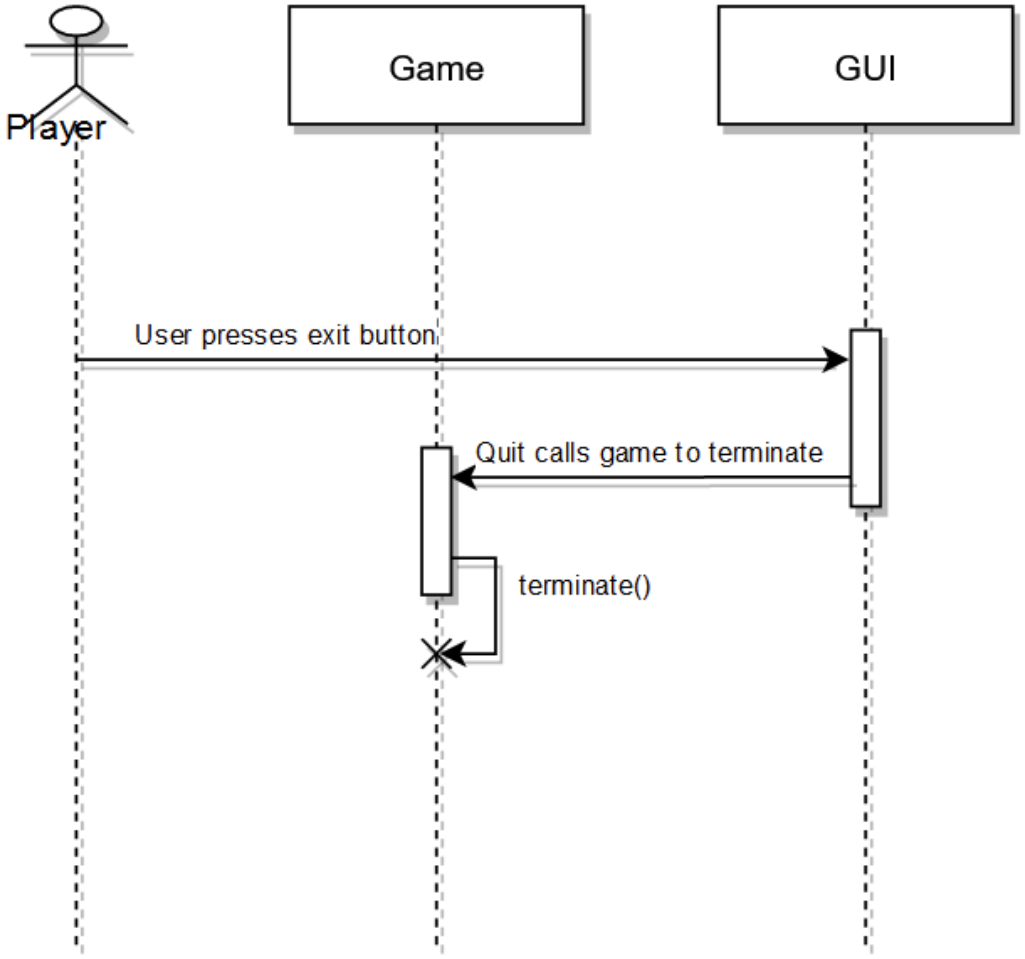

*Change Level sequence diagram.*

| Use Case Name: | Unlock Level |
|---|---|
| Actors: | |
| Description: | Player has unlocked a new level. |
| Type: | Function |
| Includes: | Change Level |
| Extends: | Vehicle Reaches Goal |
| Cross-refs: | |
| Uses cases: | |

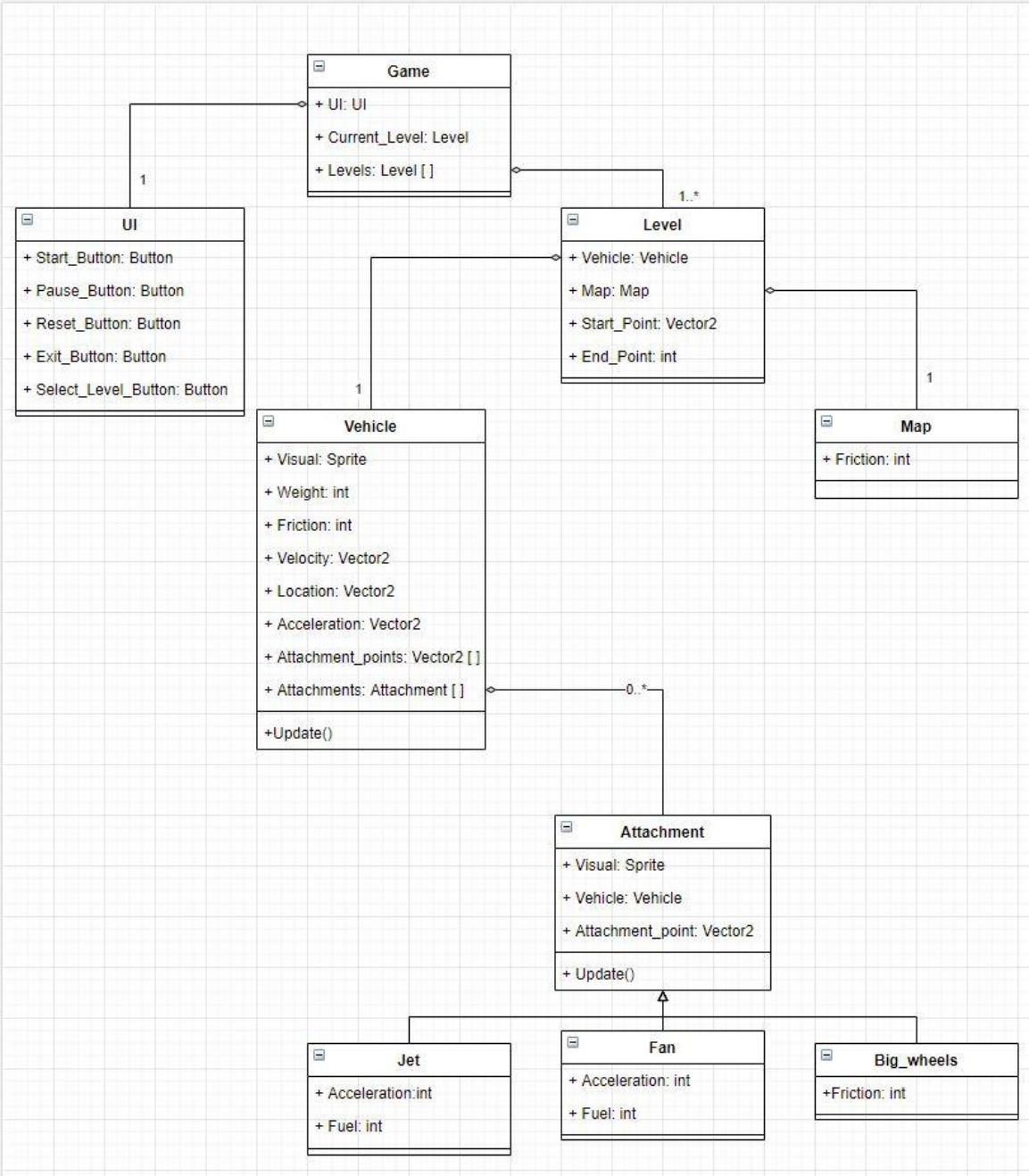



*Unlock Level sequence diagram.*

| Use Case Name: | Exit Game |
|---|---|
| Actors: | Player |
| Description: | Player uses the UI to exit the game, stopping all running processes. |
| Type: | Function |
| Includes: | |
| Extends: | |
| Cross-refs: | |
| Uses cases: | |




*Exit Game sequence diagram.*




| Class Name: | Game |
|---|---|
| Description: | The class that represents Unity's game engine |
| Extends: | |

| | | | |
|---|---|---|---|
| Attributes: | UI | UI | The Games main UI |
| | Current_Level | Level | The Currently selected level |
| | Levels | Level[ ] | The List of available levels |
| Operations: | | | |

| | | | |
|---|---|---|---|
| Class Name: | UI | | |
| Description: | The class that represents the User Interface | | |
| Extends: | | | |
| Attributes: | Start_button | Button | Button to start the game. |
| | Pause_button | Button | Button to pause the game. |
| | Reset_button | Button | Button to reset the game. |
| | Exit_button | Button | Button to exit the game. |
| | Select_level_button | Button | Button to select the game level. |
| Operations: | | | |

| | | | |
|---|---|---|---|
| Class Name: | Level | | |
| Description: | The container that represents level in a game and contains all necessary elements for a level | | |
| Extends: | | | |
| Attributes: | Vehicle | Vehicle | Player controlled vehicle. |
| | Map | Map | Map the vehicle will drive on. |
| | Start_Point | Vector2 | Location the vehicle starts |
| | End_Point | Vector2 | Location the vehicle must get to to end the Level |

| | | | |
|---|---|---|---|
| Operations: | | | |

| | | | |
|---|---|---|---|
| Class Name: | Map | | |
| Description: | Represents a level, can have different terrain with different attributes such as friction and slope. | | |
| Extends: | | | |
| Attributes: | Friction | int | The frictional coefficient of the map on the vehicle |
| Operations: | | | |

| | | | |
|---|---|---|---|
| Class Name: | Vehicle | | |
| Description: | The Vehicle class represents the controllable vehicle that the user uses to complete a level | | |
| Extends: | | | |
| Attributes: | Visual | Sprite | Visual representation of the vehicle. |
| | Weight | int | Weight of the vehicle. |
| | Friction | int | Frictional coefficient of the vehicle. |
| | Velocity | Vector2 | Current velocity of the vehicle. |
| | Location | Vector2 | Current Location of the vehicle |
| | Acceleration | Vector2 | Current Acceleration of the vehicle |
| | Attachment_points | Vector2 [ ] | Attachment points on the vehicle |
| | Attachments | Attachment [ ] | Attachments installed on the vehicle. |
| Operations: | Update() | | A Unity function called once per frame, used to calculate velocity, location and acceleration using, |

| | | | |
|---|---|---|---|
| | | | weight, friction, and attachments |

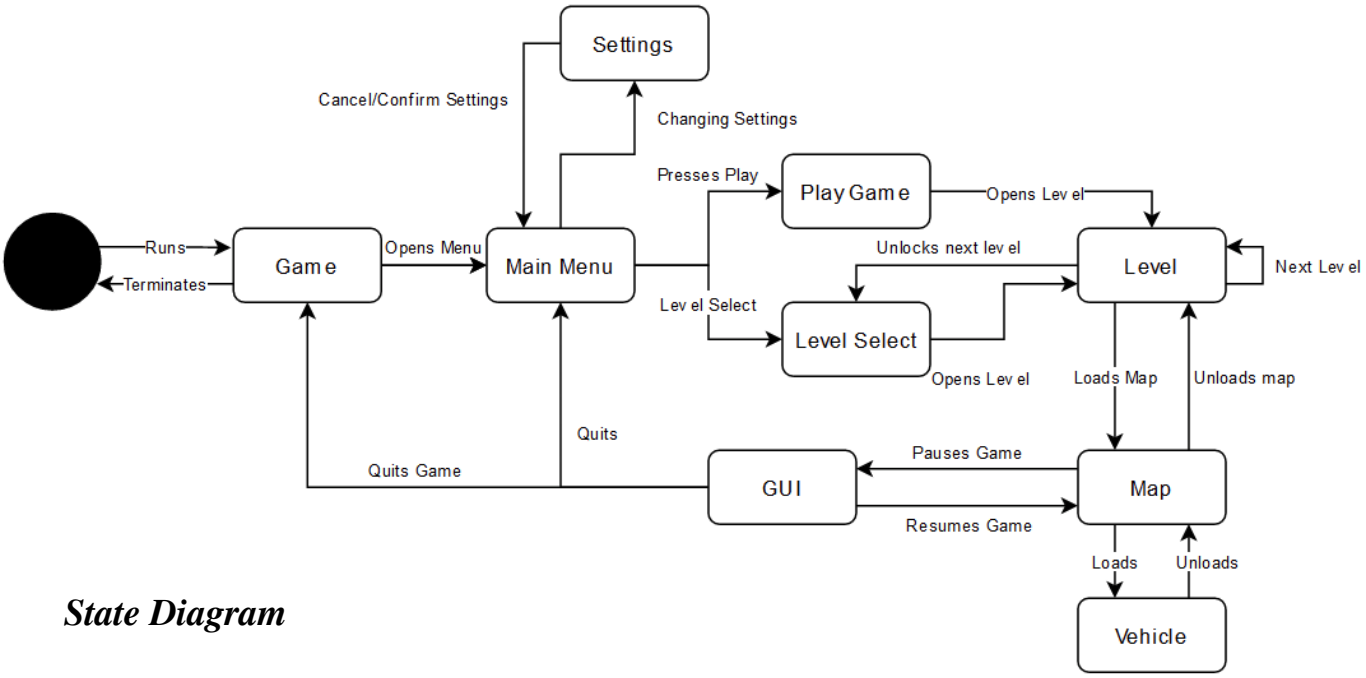
| Class Name: | Attachment | | |
|---|---|---|---|
| Description: | The abstract class that represents an attachment that changes the properties of a vehicle | | |
| Extends: | | | |
| Attributes: | Visual | Sprite | Visual representation of a attachment |
| | Vehicle | Vehicle | Vehicle the Attachment is attached to |
| | Attachment_point | Vector2 | The location of the attachment on the Vehicle |
| Operations: | Update() | | A Unity function called once per frame, used by inheriting classes to update parameters |

| Class Name: | Jet | | |
|---|---|---|---|
| Description: | | | |
| Extends: | Attachment | | |
| Attributes: | Acceleration | Vector2 | Acceleration applied by jet |
| | Fuel | int | Fuel level in percentage (0-100) |
| Operations: | Update() | | A Unity function called once per frame, used to update remaining fuel. |

| Class Name: | Fan | | |
|---|---|---|---|
| Description: | | | |
| Extends: | Attachment | | |
| Attributes: | Acceleration | Vector2 | Acceleration applied by fan |

| | | | |
|---|---|---|---|
| | Fuel | int | Fuel level in percentage (0-100) |
| Operations: | Update() | | A Unity function called once per frame, used to update remaining fuel. |

| | | | |
|---|---|---|---|
| Class Name: | Big_wheels | | |
| Description: | | | |
| Extends: | Attachment | | |
| Attributes: | Friction | int | Represents the frictional coefficient of the big wheels on a surface |
| Operations: | | | |




***State Diagram***

# 5 Prototype

The prototype will be presented as a standalone executable built targeting Windows. It will allow the user to open levels and adjust settings through a main menu. Once the player loads into the selected level, they will be able to play, using physics to solve levels. This will consist of demonstrating basic physics concepts in an interactive manner, building up to the driving of a vehicle to a designated goal, which will require the implementation of learned physics concepts to complete successfully. The prototype will therefore demonstrate the entire core infrastructure and implementation of PhysicsGame.

## 5.1 How to Run Prototype

The PhysicsGame prototype can be run by visiting the GitHub repository hosted at www.github.com/KyleM21/PhysicsGame and switching to the *release* branch. Then, once this branch is downloaded as a zip and extracted or, alternatively, cloned, the executable can then be run. PhysicsGame should then boot to the Main Menu.

These instructions are also available in the README.md files within the repository. Links to the repository are also available through the website.

## 5.2 Sample Scenarios

Example Scenario 1:

The user opens up the website and starts the game. They hit play and will start from level 1. Level 1 will consist of getting a vehicle to a goal, and the user will have to figure out how to make it happen. Upon completion of this level, the user will be allowed to move on to level 2. Completion of these levels will require the modification of the vehicle in some way. There will be lessons on the concepts used in each level available to the user.

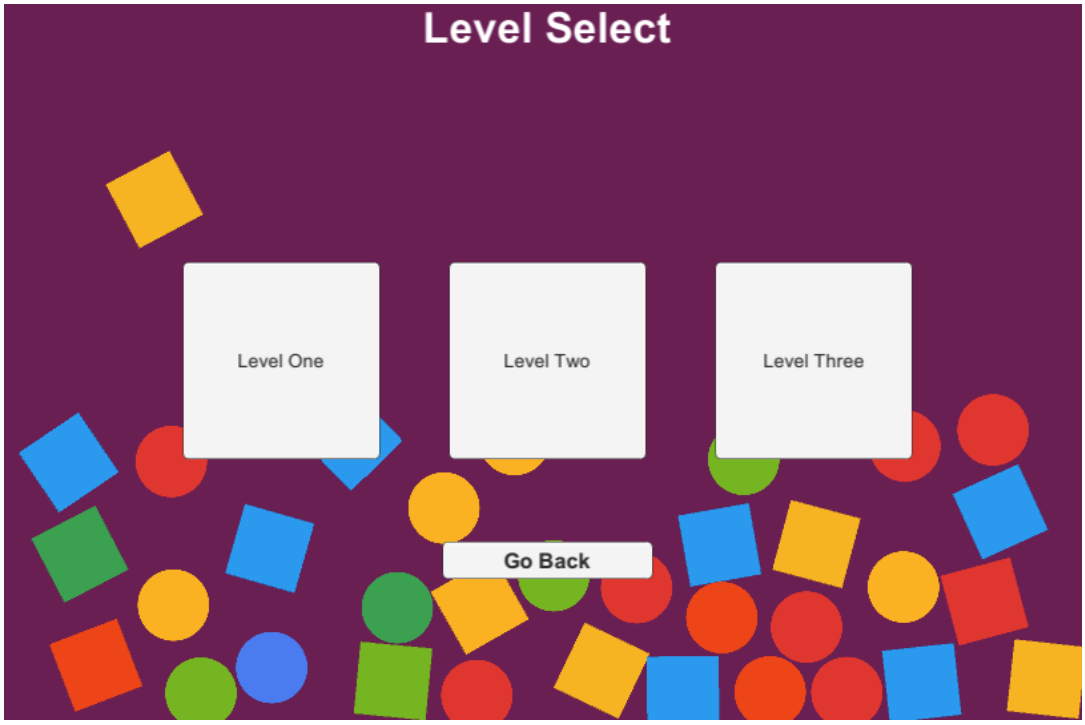
Example Scenario 2:

The user downloads and runs the game on their computer. They are greeted by the main menu, and decide to tweak some settings before beginning play. They click on the settings button which activates the settings menu. They then adjust the volume using a slider, and press the back button to return to the main menu. They then press Play to begin a new game. They are then presented with Level One, which involves driving a vehicle to a destination using physics concepts. They are guided by text which appears and disappears as they continue through the level. Once the vehicle has successfully reached its goal in the game world, the level completes and loads the next level, level two. This continues similarly until the player presses Escape and Quit to Desktop, which exits the game, or completes all levels, at which point they are returned to the Main Menu.

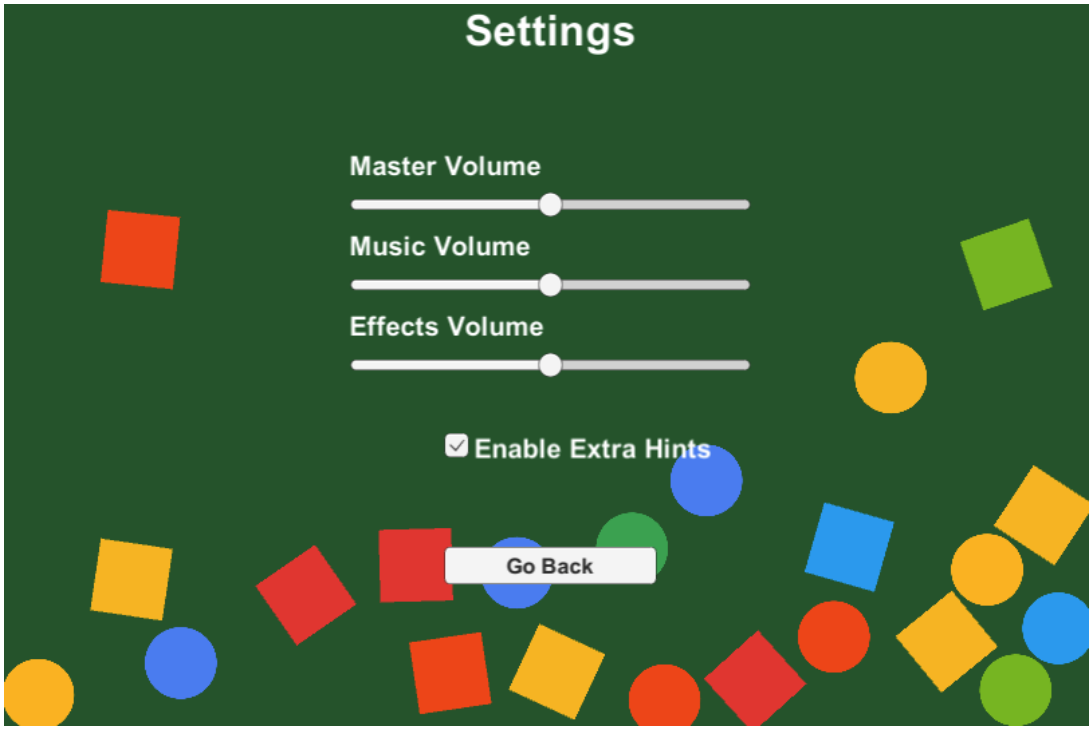
**On the next page you will see screenshots of the latest prototype.**


*This is the main menu, the user has four options: Play, Level Select, Settings, and Quit*


*This is two balls bouncing on a platform. This is purely for testing purposes and will not be included in the final build*




*This is the level selection menu, the user can pick the level they wish to play and the game will bring them to it.*




*This is the settings menu, the user can customize volume and enable extra help.*

## 6 References


[1] Project Website: https://kylem21.github.io/PhysicsGame/

[2] Project Repository: https://github.com/KyleM21/PhysicsGame

[3] Official IEEE SRS Document Template. IEEE Std 830-1998. 11/12/2021. IEEE Comp. Society. http://www.cse.msu.edu/~cse870/IEEEXplore-SRS-template.pdf


## 7 Point of Contact

For further information regarding this document and project, please contact **Prof. Daly** at University of Massachusetts Lowell (james_daly at uml.edu). All materials in this document have been sanitized for proprietary data. The students and the instructor gratefully acknowledge the participation of our industrial collaborators.