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 covers the basics of the project. It goes over the purpose of the document, as well as the audience for the document. It also goes over the scope of the project, identifying the product produced and its intended goal. Finally, it covers definitions for terms used in the document, as well as how the rest of the document is organized.
- Section 2 covers the specifications of the project. It begins with the perspective for how the final product will be used, and covers any constraints on the project. It then covers 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 covers the requirements of the project. It is in hierarchical format, with the most important requirements coming first.
- Section 4 contains 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 covers the current prototype, scenarios for how the software will be used, and includes screenshots of the current prototype.

1.1 Purpose

This software requirements specification document (SRS) is intended to describe PhysicsGame. This SRS document will describe what PhysicsGame is intended to be, how it 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

PhysicsGame is a Unity engine-based educational video game. PhysicsGame'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 player with the ability to interact with a realistic and directed (AKA 'guided' via the use of pop-up hints) physics environment, the player 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.

PhysicsGame 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

PhysicsGame – The Unity engine-based educational physics game which will be produced.

Player – Whomever is playing the game.

Vehicle – The object the player will move around.

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

SRS document - Software Requirements Specification document, this document.

Project - The process of development that will produce PhysicsGame and the SRS document.

AKA - Acronym for 'also known as.'

IEEE - Institute of Electrical and Electronics Engineers

GUI - Graphical User Interface, the buttons and text that a user sees and interacts with.

1.4 Organization

The remainder of this SRS document contains descriptions of PhysicsGame, describing what it 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 PhysicsGame explaining the purpose and features. Also included are sequence and use case diagrams explaining how the product will function. Finally, there are screenshots to show what the player will see upon entering PhysicsGame.

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

2 Overall Description

Summary of Section 2

Section 2.1 will be going over the intent of the project and the constraints that will need to be followed as PhysicsGame is developed. Section 2.2 is about the function that PhysicsGame will serve, this will then go over the primary function that the final product will serve, and show the reader a goal diagram. Section 2.3 contains details on the intended audience for the product. Section 2.4 goes over the constraints on the project. Section 2.5 is the final section and will be covering all of the things the team will assume of the end-users.

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

PhysicsGame is an education-based video game with a focus on physics concepts, meaning it will teach physics to the player through interactivity. The educational aspects of PhysicsGame, while important, cannot get in the way of the user having fun. Players must be able to easily understand how to play the game so they can focus on learning.

Hardware Constraints

PhysicsGame needs to use a mouse and keyboard for playing the game.
It must also run on older hardware and laptops to allow more players to be able to run and play PhysicsGame.

Software Constraints

PhysicsGame will be built on the unity engine, and must function when deployed on a Windows system.

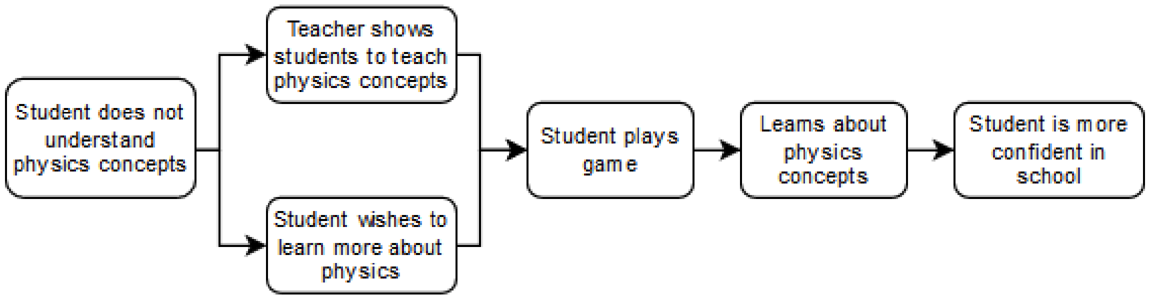
2.2 Product Functions

The product being produced will be a physics learning game called PhysicsGame. This game will feature multiple levels where the player is given a vehicle to control, with the goal of making it down a slope and passing a goal. Completing these levels will require the user to consider the different attachments available, what they will change about the vehicle, and pick the best one.

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.

While playing the game, hints will appear in each level. These hints will provide information on the level, including possible strategies for reaching the end goal as well as the basic physics concepts behind each level's design. It may be necessary for the player to grasp each concept prior to successfully completing the level. The player will then, through interacting with the game, be able to see for themselves how these concepts come into play.

Goal Diagram:


Student does not understand physics concepts
Teacher shows students to teach physics concepts
Student wishes to learn more about physics
Student plays game
Learns about physics concepts
Student is more confident in school

High-Level goal diagram.

2.3 User Characteristics

The targeted demographic is young students 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 focused on being an educational game, with a purpose of teaching the user about basic physics.

Hardware Limitations
- Developed to run on both old and new desktop computers, the final product must run on the older desktops that most schools use in computer labs.

Interface Limitations
- Must have a download link on the website,

- Must support proper version control with Github in order to be developed by multiple people.

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 player's hardware:

- Working computer
- Functioning Windows 7 (or newer) operating system.
- Working monitor, mouse and keyboard.
- The player's computer is capable of running basic video games for at least 15 minutes at a time.

Assumptions of the player:

- The player is able to play games for at least 15 minutes at a time.
- Player is a child, possibly a student, aged equivalent to those in US grades 4 through 6.
- Player has an interest in learning physics.
- Player is capable of operating a computer with a mouse and keyboard.
- Player has basic video game experience or ability.

2.6 Apportioning of Requirements

Some features are out of the scope of this project or are best left to a later, more complete version of PhysicsGame. This includes the final name of PhysicsGame, more advanced in-game graphics, and additional physics concepts. Another feature which will be left to a later version is more expansive levels and interactive hints, which would require the player to enter an answer to a physics-based question prior to continuing. A specific GUI for enabling and disabling vehicle attachments will be available in future versions when more levels are offered.

3 Specific Requirements

1. Must be interactive and responsive to player
    1. Provides feedback to the player
    2. Allows the player to interact with the game
2. Must be suitable for students from 4th-6th
    - 2.1 Concepts must be easy to understand
    - 2.2 Three concepts must be present:

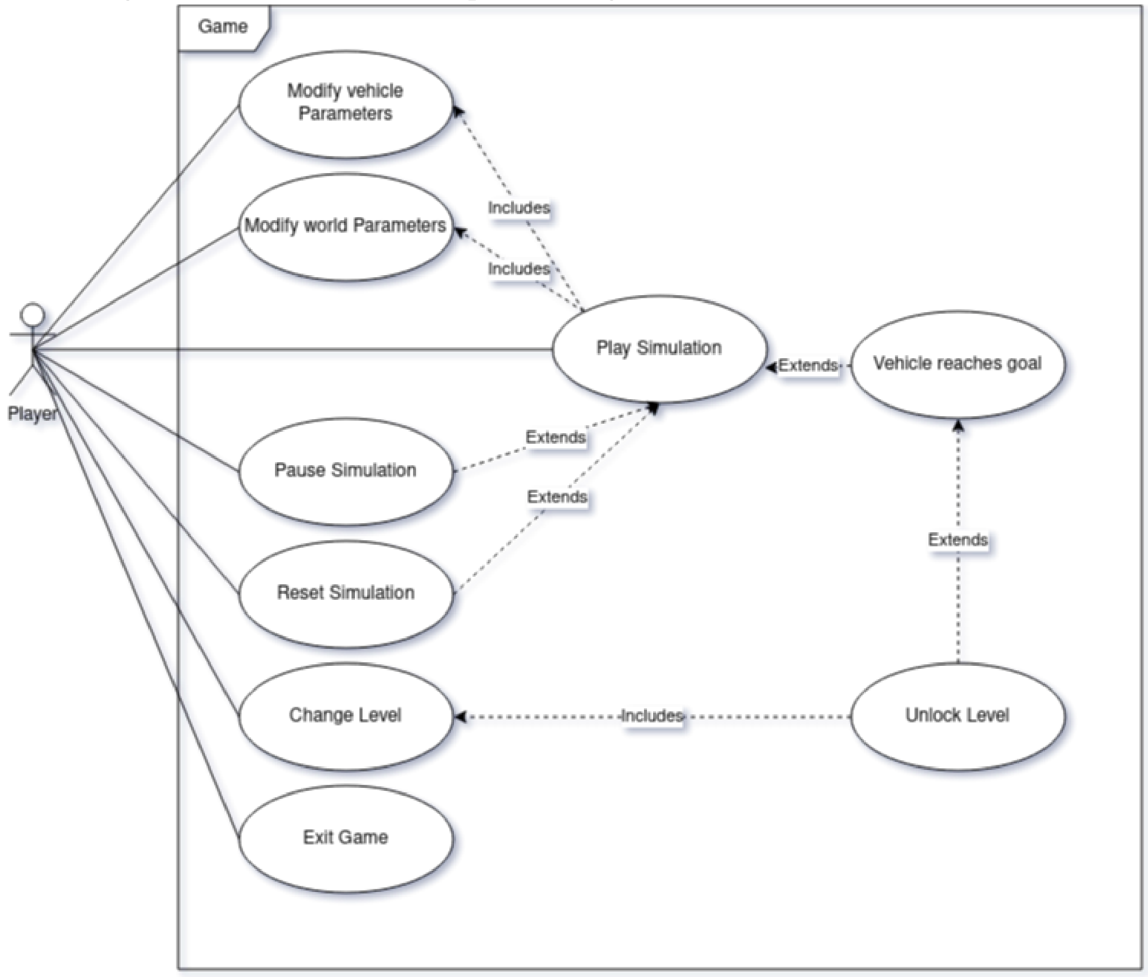
        1. Gravity
        2. Friction
        3. Acceleration
3. Must have an educational component
   3.1 Teach the user each concept they need to complete the level
   3.2 Pop-up lesson available for each concept
4. Must be constrained to a 2D world in Unity
5. Must have user-controlled vehicle
6. Must have variable parameters on simulation
   7.1 Must be able to modify vehicle
      7.1.1 Must have at least three modifications of some kind
7. Must have audio feedback
8. 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
9. Must have a menu or menus which enable the following 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
10. 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

This section contains the Use Case and Sequence Diagrams in section 4.1, followed by Class Diagrams in section 4.2, and State Diagram in section 4.3.

4.1 Use Case Diagrams

Below is the master Use Case Diagram, followed by a series of breakout tables describing each use case with its Sequence Diagram.



Master Use Case Diagram. Standard UML notation. Shows the interaction cases between the Player actor and the game itself, as well as pertinent cases relevant to the user experience which may not require direct player interaction.

4.1.1 Modify Vehicle Parameters

The Player uses the game GUI to add, remove, or edit the Vehicle attachments and attributes.

Use Case Name:	Modify Vehicle Parameters
Actors:	Player
Description:	Player uses GUI to add, remove, or change Vehicle attachments and attributes.
Type:	Function
Includes:	

Extends:	
Cross-refs:	
Uses cases:	

4.1.2 Modify World Parameters

The Player uses the game GUI to modify the physics parameters of the game Level, altering the motion of the Vehicle and its path through the Level and affecting its prospects for reaching the Goal.

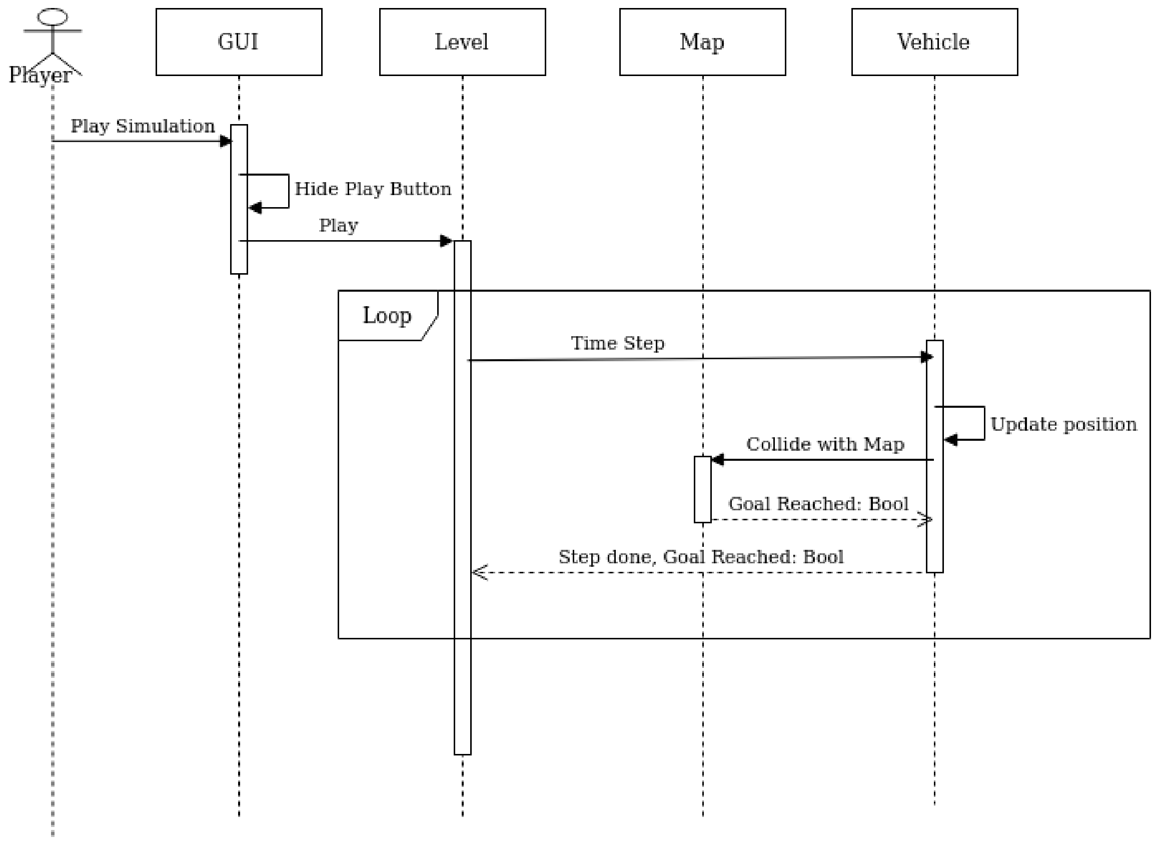
Use Case Name:	Modify World Parameters
Actors:	Player
Description:	Player uses GUI to change physics parameters within the game Level.
Type:	Function
Includes:	
Extends:	
Cross-refs:	
Uses cases:	

4.1.3 Play Simulation

The Player clicks the Play Button overlaid on a paused Level. This hides the Play Button, then causes the Level physics simulation to begin, enabling Vehicle movement and interaction with the Map. During the simulation, if the Map interaction indicates the Goal has been reached, the Vehicle Reaches Goal case occurs.

Use Case Name:	Play Simulation
Actors:	Player
Description:	Player uses GUI to begin the physics simulation.
Type:	Function
Includes:	Modify Vehicle Parameters, Modify World Parameters

Extends:	
Cross-refs:	
Uses cases:	

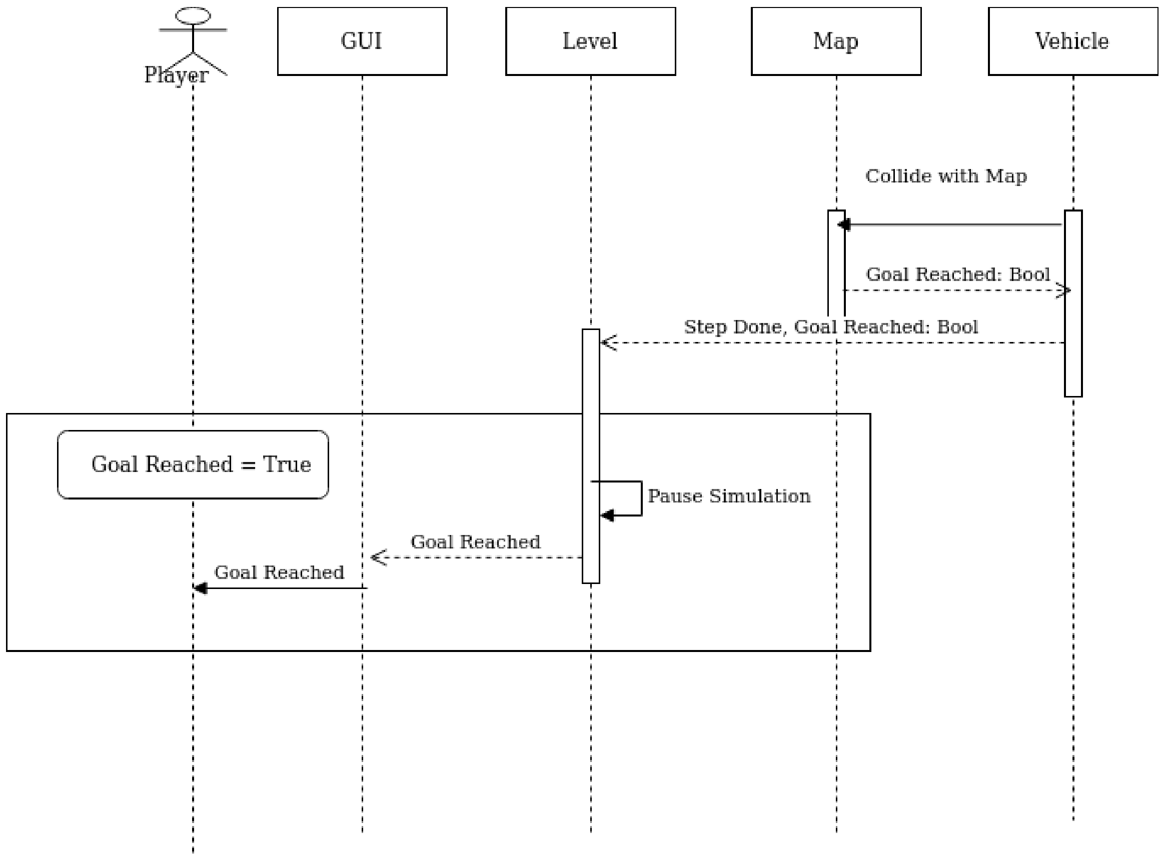


Play Simulation sequence diagram.

4.1.4 Vehicle Reaches Goal

The Level physics simulation has run and the Vehicle has successfully reached the goal condition. The simulation is paused, the Player is notified via the GUI that the goal has been met, and the next Level and/or Attachment is unlocked for the Player.

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:	

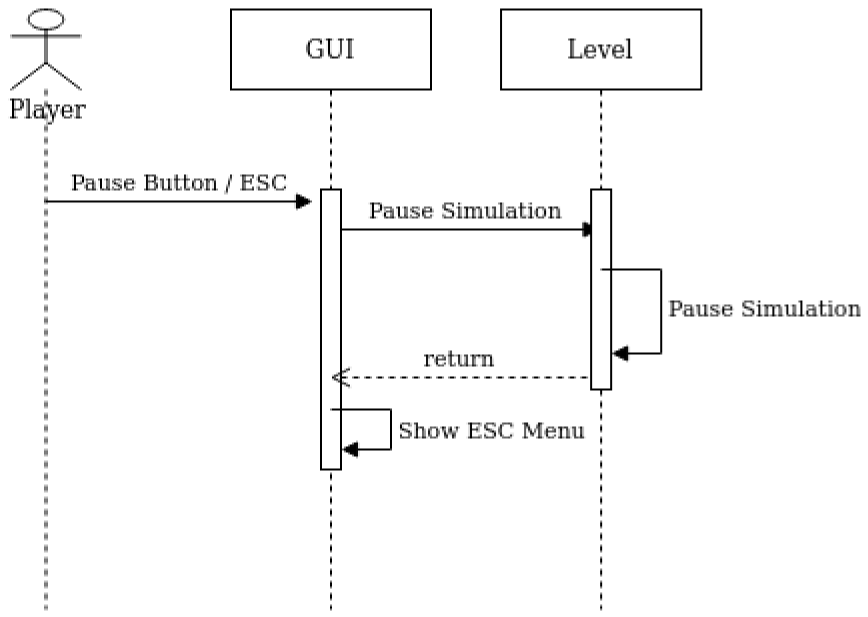


Vehicle Reaches Goal sequence diagram.

4.1.5 Pause Simulation

The Player pauses the simulation, using either the ESC key or the GUI Pause Button. In addition to pausing, this action also brings up the GUI ESC Menu, allowing the Player to Resume, Restart, go to the Main Menu, or Quit the Game.

Use Case Name:	Pause Simulation
Actors:	Player
Description:	Player pauses the physics simulation.
Type:	Function
Includes:	
Extends:	Play Simulation
Cross-refs:	
Uses cases:	

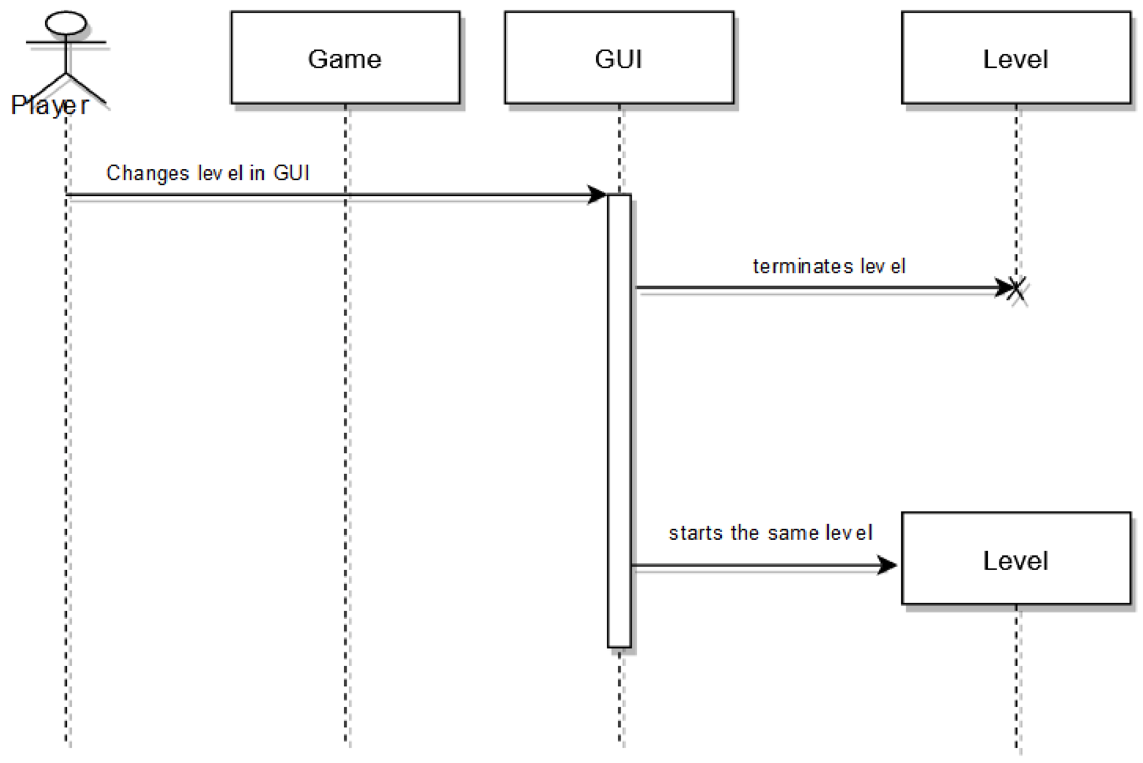


Pause Simulation sequence diagram.

4.1.6 Reset Simulation

The Level simulation is reset, moving the vehicle back to the start location and removing any progress made since the last start. This is accomplished by terminating the existing Level, and spawning a fresh copy of it. This Case may be activated automatically when the Game determines Vehicle Does Not Reach Goal, or manually by the Player selecting to Reset Level through the UI.

Use Case Name:	Reset Simulation
Actors:	Player
Description:	Physics simulation is reset, 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:	

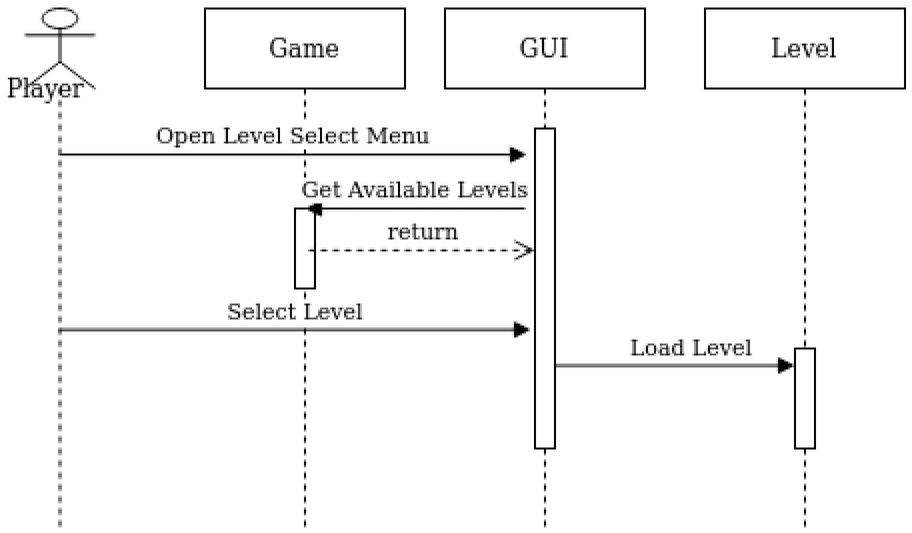


Reset Simulation sequence diagram.

4.1.7 Change Level

The Player starts at the Main Menu, and opens the Level Select Menu. The Player then selects a Level to load from the available Levels.

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

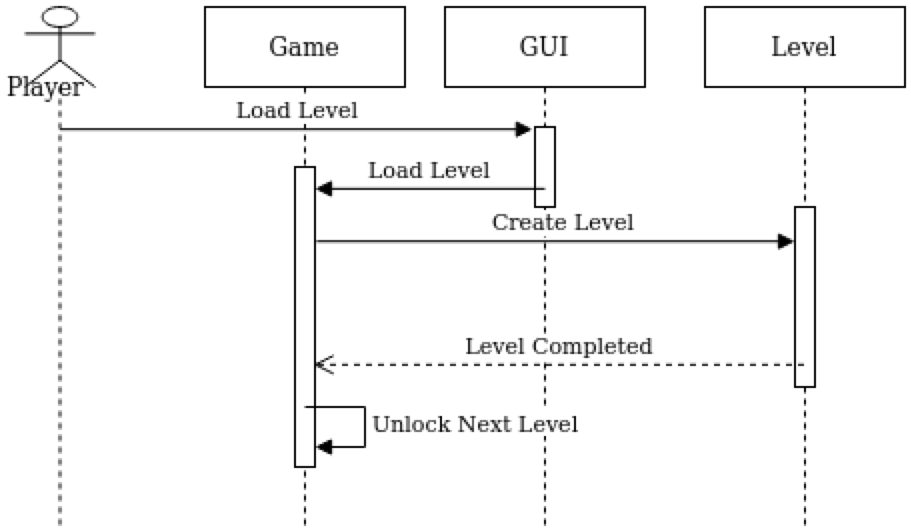


Change Level sequence diagram.

4.1.8 Unlock Level

This Case covers the beginning and completion of a Level, while abstracting away the actual details inside the Level. The Player uses the GUI to begin a Level, then eventually, if the Player completes the Goal, the Level returns, prompting the Game to unlock the next Level.

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:	

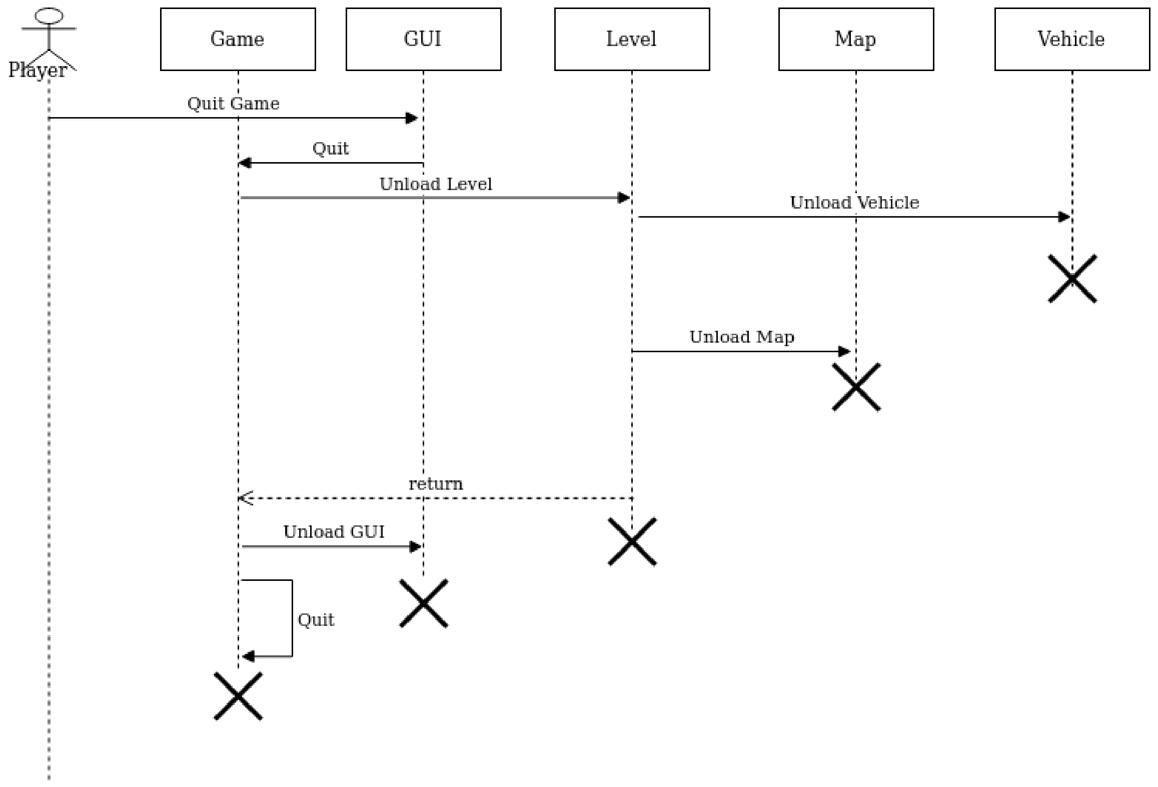


Unlock Level sequence diagram.

4.1.9 Exit Game

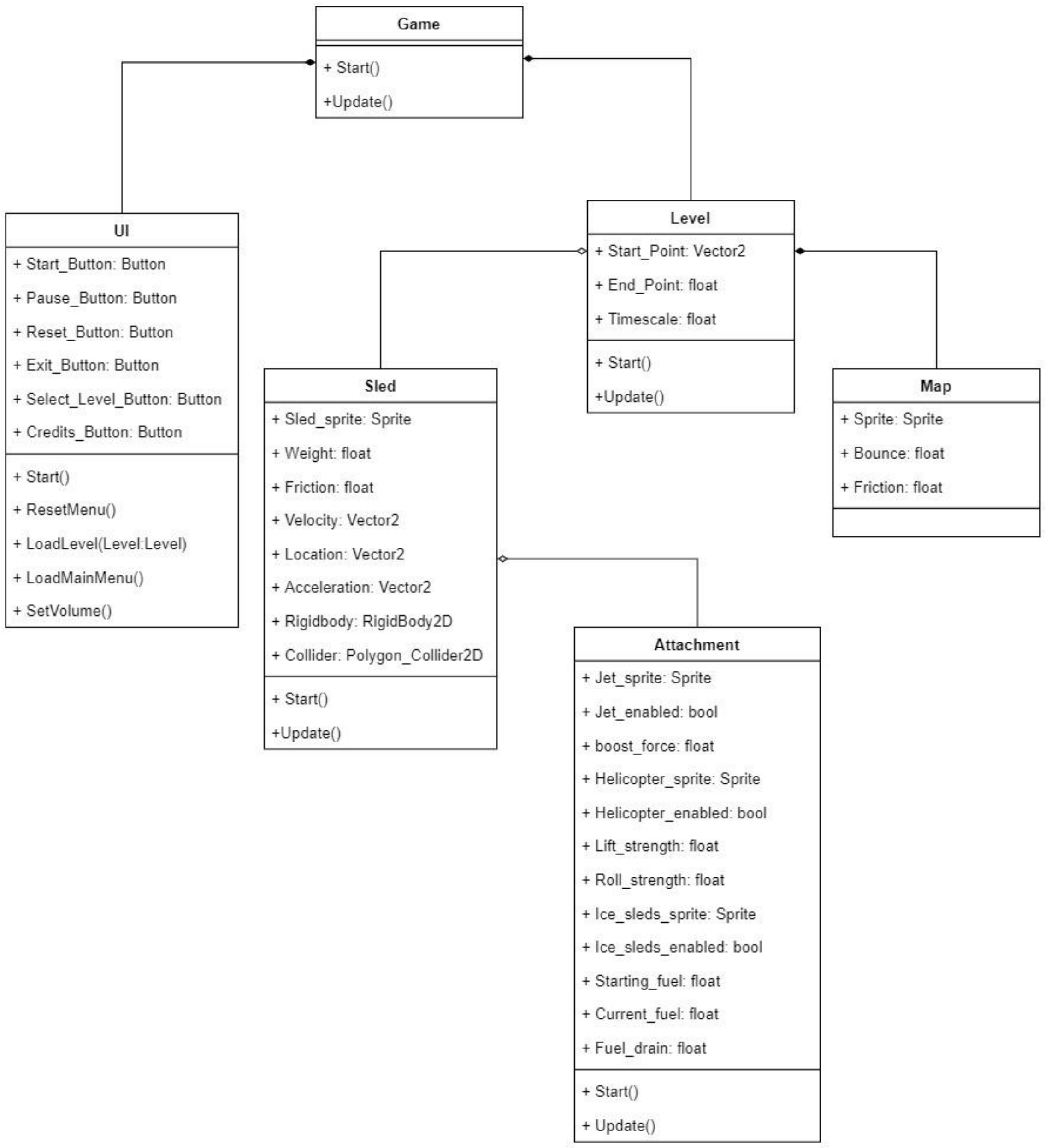
The Player uses the GUI (either the Main Menu Quit button, or the ESC Menu Quit button) to quit the game.

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



Exit Game sequence diagram.

4.2 Class Diagrams



4.2.1 Game

Class Name:	Game		
Description:	The class that represents Unity's game engine		
Extends:			
Attributes:			
Operations:	Update()		A Unity function called once per frame, used by unity
	Start()		A Unity function called before the first frame updates, used by unity

4.2.2 UI

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.
	Credits_button	Button	Button to load the game credits.
Operations:	Start()		A Unity function called before the first frame updates, used to setup GUI elements
	ResetMenu()		Reset menu states
	LoadLevel()		Load a level

	LoadMainMenu()		Load the main menu
	SetVolume()		Set the game volume

4.2.3 Level

Class Name:	Level		
Description:	The container that represents level in a game and contains all necessary elements for a level		
Extends:			
Attributes:	Timescale	float	Current timescale of the game.
	Start_Point	Vector2	Location the vehicle starts.
	End_Point	Vector2	Location the vehicle must get to to end the Level.
	Timescale	Float	Timescale game is currently running at.
Operations:	Update()		A Unity function called once per frame
	Start()		A Unity function called before the first frame updates

4.2.4 Map

Class Name:	Map		
Description:	Represents a level, can have different terrain drawn with a sprite shape renderer that can have different attributes such as friction and slope.		
Extends:			
Attributes:	Friction	Float	The frictional coefficient of the map on the vehicle.
	Sprite	Sprite_shape_renderer	Renderer for the level sprite
	Bounce	Float	A value that represents how bouncy the map is.
Operations:	Update()		A Unity function called once per frame

	Start()		A Unity function called before the first frame updates

4.2.5 Sled

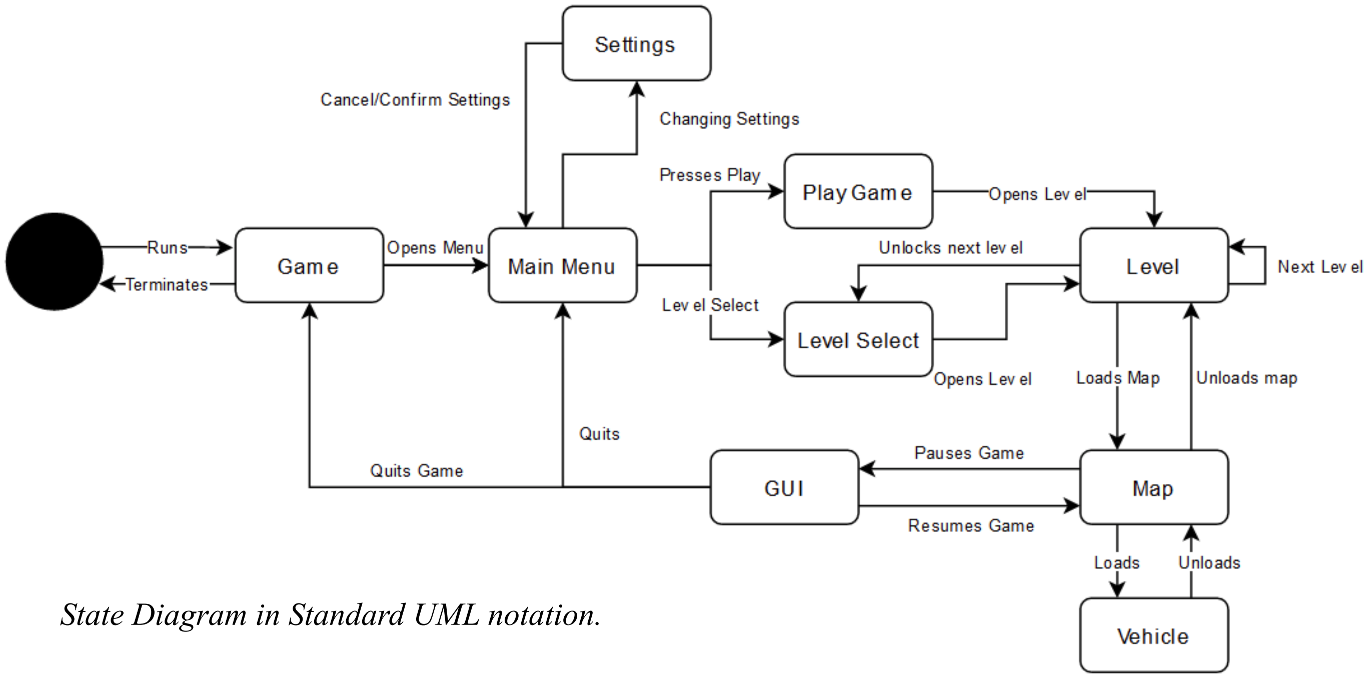
Class Name:	Sled		
Description:	The Sled class represents the controllable vehicle that the user uses to complete a level		
Extends:			
Attributes:	SledSprite	Sprite	Visual representation of the vehicle.
	Weight	Float	Weight of the vehicle.
	Friction	Float	Frictional coefficient of the vehicle.
	Velocity	Vector2	Current velocity of the vehicle.
	Location	Vector2	Current Location of the vehicle
	Acceleration	Vector2	Current Acceleration of the vehicle.
	Rigidbody	Rigidbody2D	Unity builtin for applying physics to game object.
	Collider	Polygon_Collider2D	Unity builtin for giving a sprite a collider.
Operations:	Update()		A Unity function called once per frame, used to calculate velocity, location and acceleration using, weight, friction, and attachments.
	Start()		Unity function called before the first frame update, used to set up sled.

4.2.6 Attachment

Class Name:	Attachment		
Description:	The abstract class that represents an attachment that changes the properties of a vehicle		
Extends:			
Attributes:	Jet_sprite	Sprite	Visual representation the jet attachment
	Jet_enabled	Bool	Determines if the jet is enabled and able to be seen.
	Boost_force	Float	Determines boosting force for the jet
	Helicopter_sprite	Sprite	Visual representation the helicopter attachment
	Helicopter_enabled	Bool	Determines if the helicopter is enabled and able to be seen.
	Lift_strength	Float	Determines lift force for the helicopter
	Roll_strength	Float	Determines roll force for the helicopter
	Ice_sleds_sprite	Sprite	Visual representation the Ice Sleds attachment
	Ice_sleds_enabled	Bool	Determines if the Ice Sleds is enabled and able to be seen.
	Starting_fuel	Float	Determines the starting fuel for the helicopter and jet.
	Current_fuel	Float	Determines the current fuel for the helicopter and jet.
	Fuel_drain	Float	Determines the fuel drain for the helicopter and jet.
Operations:	Update()		A Unity function called once per frame, used by inheriting classes to update parameters.

	Start()		Unity function called before the first frame update, used to set up attachments.

4.3 State Diagram



State Diagram in Standard UML notation.

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, involving 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 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

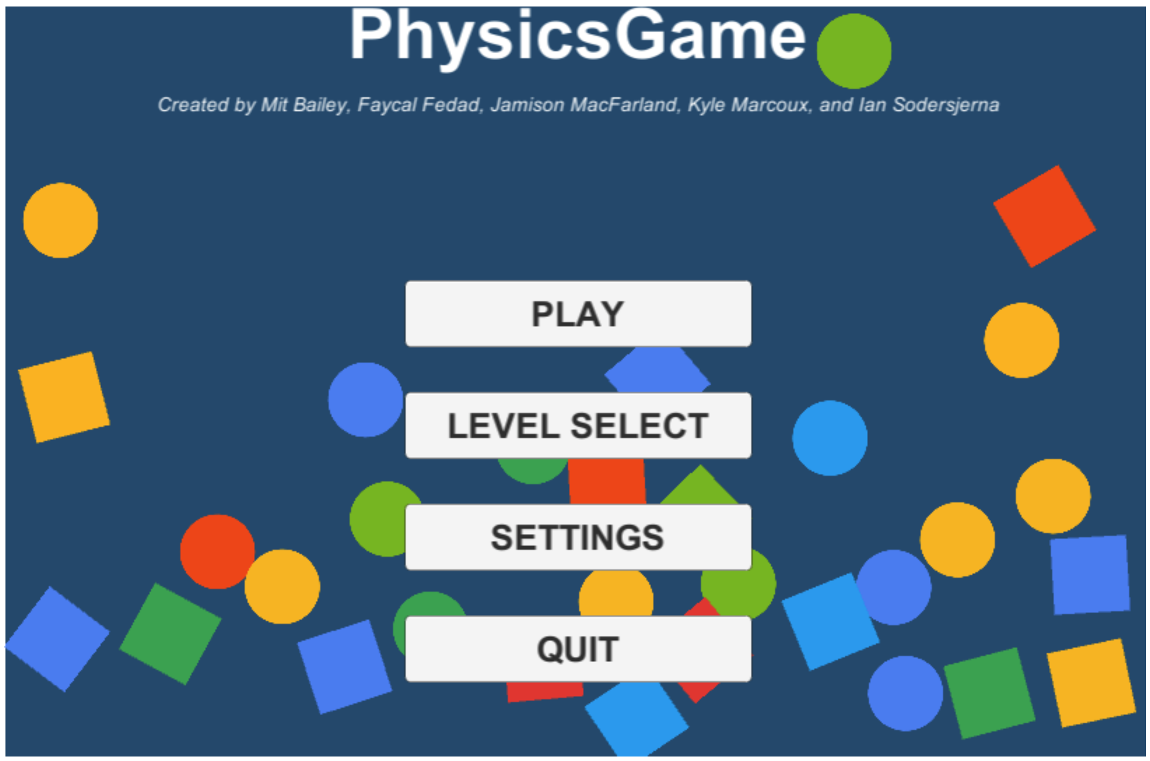
Example Scenario 1:

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.

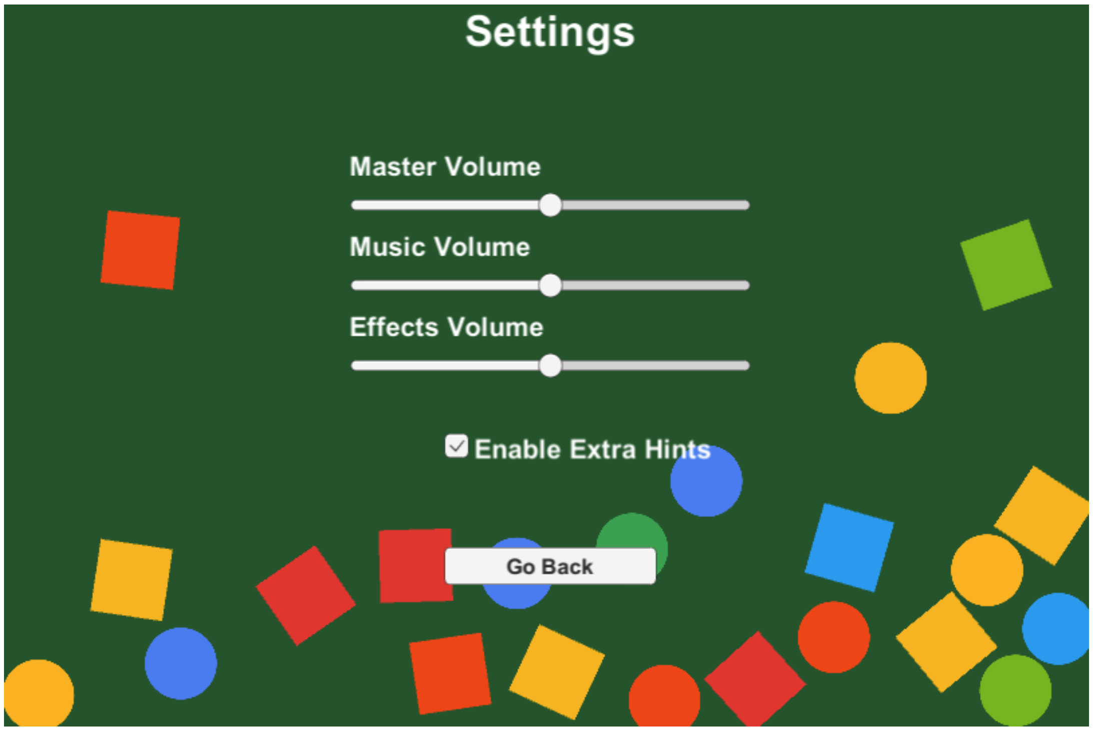
Example Scenario 2:

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.

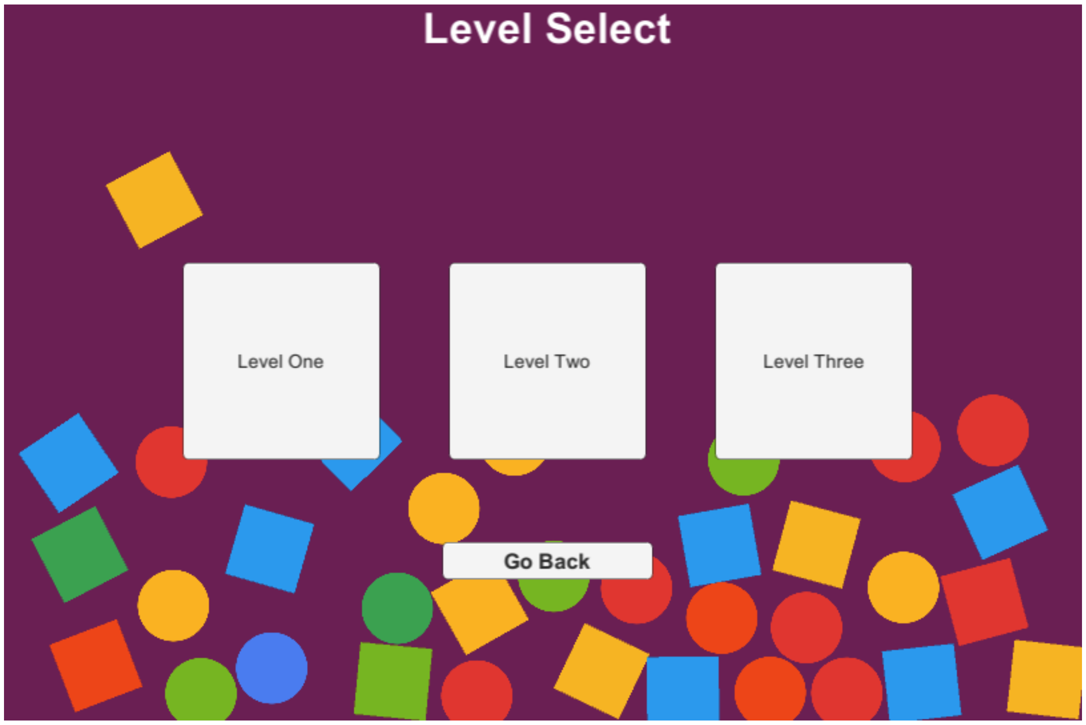
On the next page you will see screenshots of the latest prototype.


PhysicsGame
Created by Mit Bailey, Faycal Fedad, Jamison MacFarland, Kyle Marcoux, and Ian Sodersjerna
PLAY
LEVEL SELECT
SETTINGS
QUIT

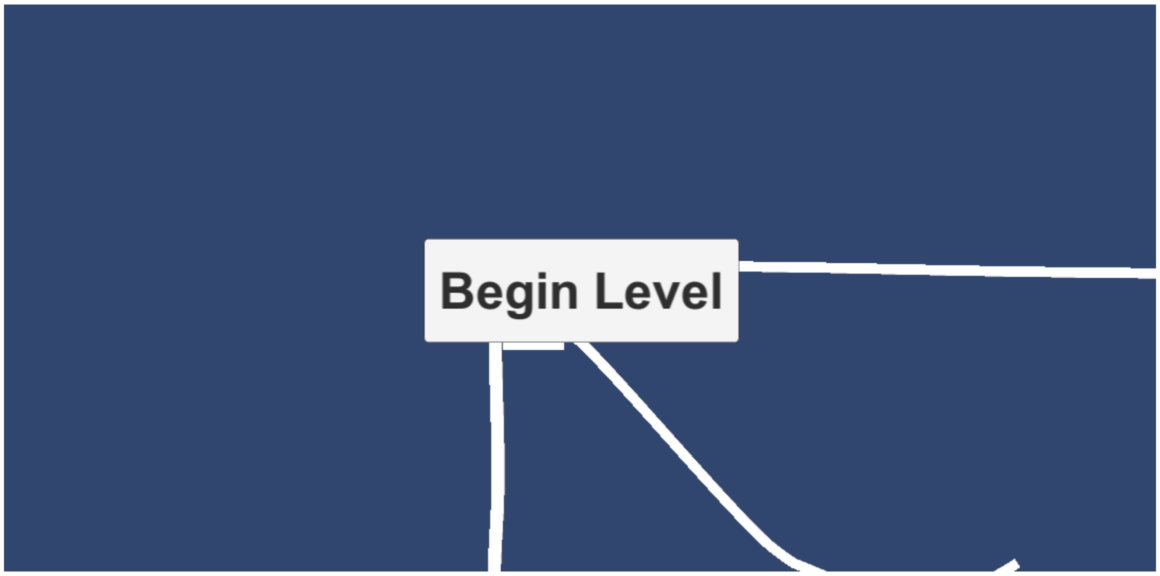
This is the main menu, the player has four options: Play, Level Select, Settings, and Quit



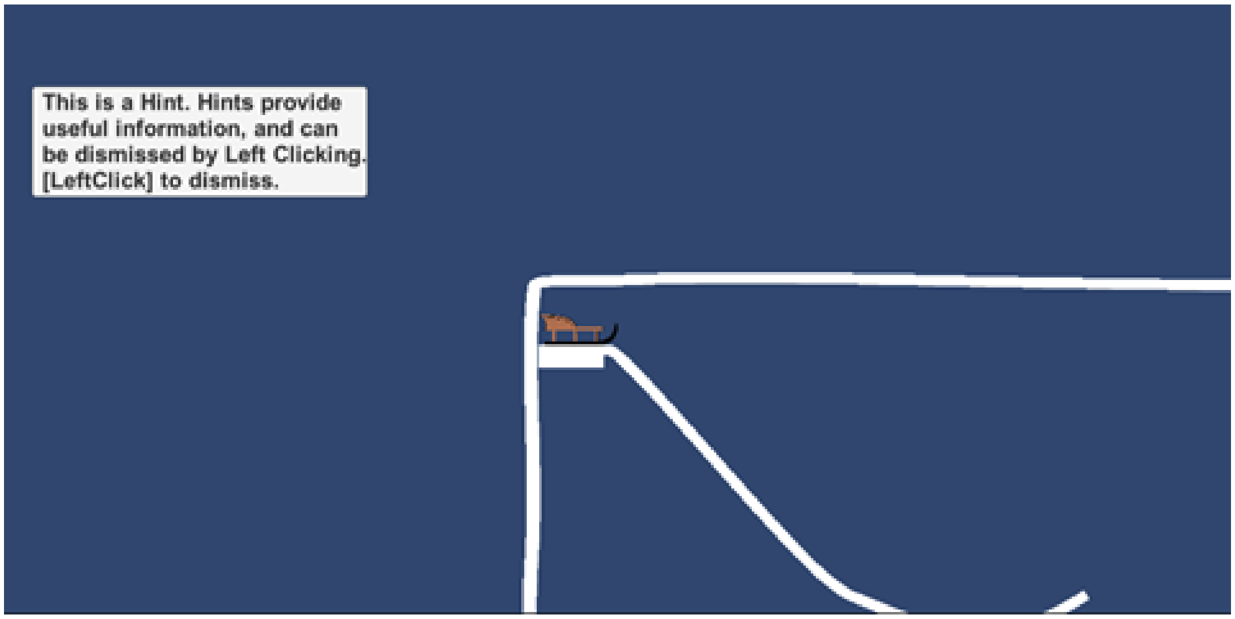
This is the settings menu, the player can customize volume and enable extra help.



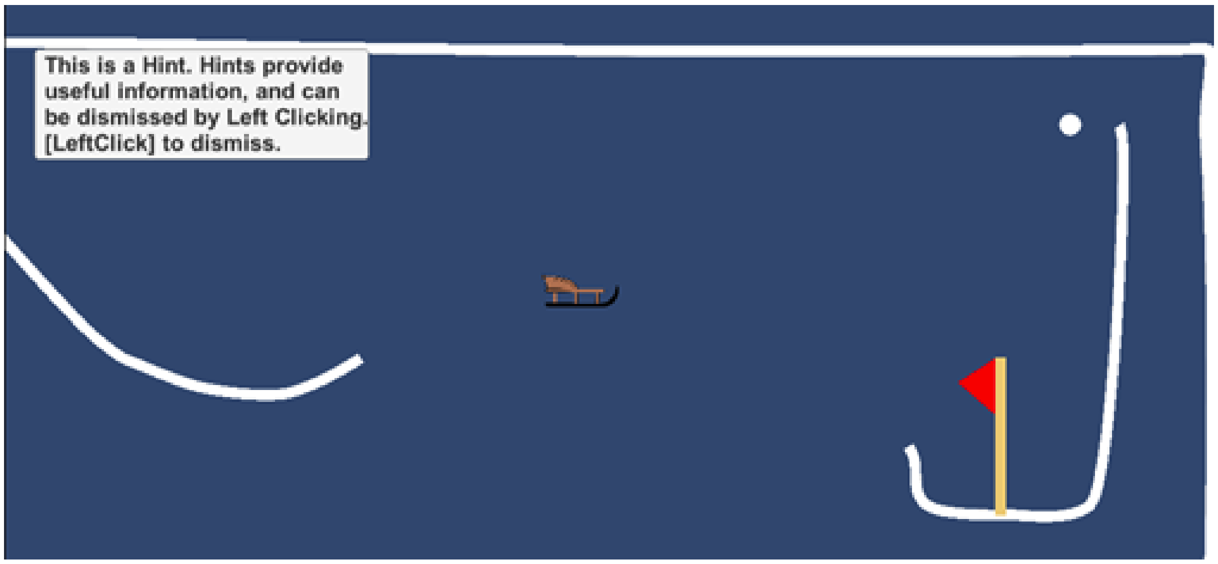
This is the level selection menu, the player can pick the level they wish to play and the game will bring them to it.



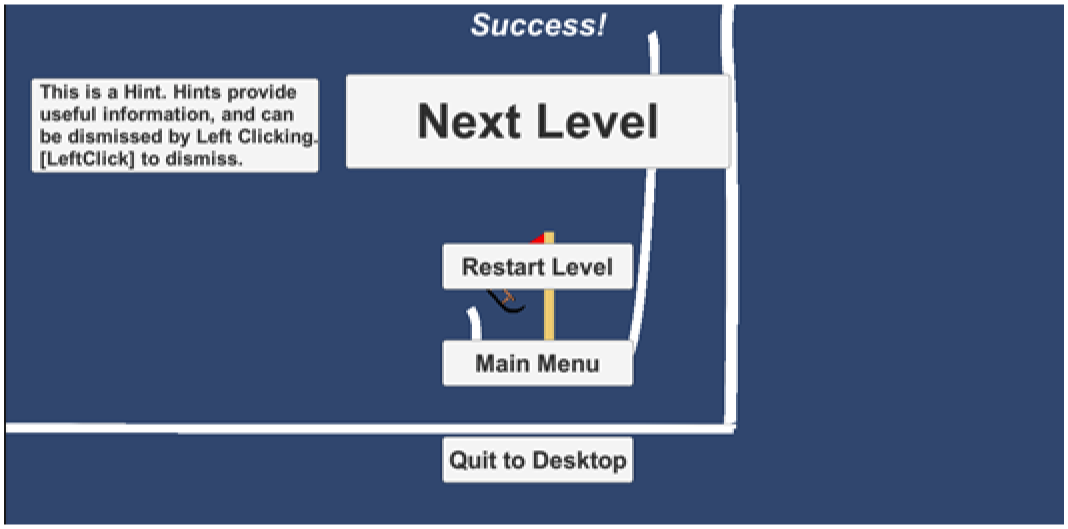
This is the opening screen of level one, behind the "Begin Level" button is the vehicle the player will be controlling.


This is a Hint. Hints provide useful information, and can be dismissed by Left Clicking. [LeftClick] to dismiss.

This is the first part of level one, the goal is on the other side of the level and the player will need to get the vehicle down the slope with enough speed to get to the goal.


This is a Hint. Hints provide useful information, and can be dismissed by Left Clicking. [LeftClick] to dismiss.

This is the player mid-air towards the goal of level one. In order to clear level one you need to know how to build speed using the slope. There are hints on the side of the screen to guide the player through the level.


Success!
This is a Hint. Hints provide useful information, and can be dismissed by Left Clicking. [LeftClick] to dismiss.
Next Level
Restart Level
Main Menu
Quit to Desktop

This is the ending screen of level one, giving the player choices on what to do next. It will appear after making it into the goal.

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.