Software Requirements Specification (SRS)
Project Aerial Areas

Team: Group 3
Authors: Benjamin Brookhart, Thomas Fitzpatrick, Bennett Porter, Omar Aly, Nathan Prentiss
Customer: Students in the 4th-8th grade
Instructor: Dr. Daly

1 Introduction

The Software Requirements Specification (SRS) document will provide the reader with all the information for *Aerial Areas*; a reinforcement learning game designed for kids looking to master basic area and perimeter concepts. This section talks about the purpose of the SRS, along with the scope of the project and notable terms for the reader to reference as they're mentioned later. The document features a project description, a list of requirements, software models, a prototype of the game, and references.

1.1 Purpose

The purpose of the SRS is to give the reader a clear explanation of the project's requirements and constraints. It will show the diagrams used to organize ideas during development and give a brief demonstration of how the game can be played. This document grants all parties involved, which may include shareholders, customers, players, and other interested parties; a more in-depth vision of what the project is meant to encompass.

1.2 Scope

The software product being produced is Aerial Areas. Aerial Areas is a 2D edutainment game developed using Godot and C#. Our game hopes to provide a fun way for kids to engage in math learning activities while progressing through a stimulating in-game experience that encourages them to keep completing problems and learning. This product will help players better conceptualize the area and perimeter of basic shapes. Pattern recognition, continuous repetition, and over time mastery of the formulas will sharpen the player's quick problem-solving skills.

1.3 Definitions, acronyms, and abbreviations

- SRS – Software Requirements Specification
- UI – User Interface
- Aerial Areas – The name of the game which this SRS is about
- AA – Short for Aerial Areas
- Player – The user who will be playing the game
- Geometroid – The alien enemies who the player is tasked with defeating
- Shape – Geometric figures, those being squares, rectangles, triangles, circles, half circles, and quarter circles
- Perimeter – The combined length of all sides of a shape, or in the case of a circle, the distance around it
- Side – A singular straight line on a shape
- Circumference – The perimeter of a circle
- Radius – Half of the length across a circle
- Area – The total space a shape takes up
- Power up – A temporary consumable the player can use to help them
- Upgrade – A permanent boost to the player's power and or abilities
- High Score – The best score the player has obtained whilst playing

- Problem – An equation asking for the area or perimeter of a shape that the player must solve
- Solution – The answer to the equation of a given problem, needed to defeat the associated enemy
- Problem List – A UI component which lists out all of the problems of all the enemies currently visible, and highlights the current problem selected
- Enemy – A threat to the player that has a problem the player must solve, whereupon they are defeated
- Boss – A difficult enemy that is harder and more challenging than usual
- Gold – Currency accumulated by the player by defeating enemies
- Shop – A place where the player can spend their gold for power ups and upgrades

1.4 Organization

The rest of the SRS contains information on what makes up the game Aerial Areas. This includes an introduction to what the game is and how it should function, the requirements the game must follow, and expectations regarding our intended audience. There are software models that help illustrate each functional part of the game and a prototype demo that explains how the game should be run.

Below is a summarization of the remaining sections of the SRS:

Section 2: Overall Description

An introduction to AA and its interfaces. Describes the major functionalities of the game and constraints that must be abided by throughout development and distribution. Gives the recommended system requirements to run AA and types of users we are trying to reach.

Section 3: Specific Requirements

The list of requirements that make up AA. These requirements outline clear functionalities and expectations the application must meet.

Section 4: Modeling Requirements

Variety of diagrams that illustrate different aspects of the application. Includes a use case, class and multiple sequence diagrams.

Section 5: Prototype

Demo that shows the reader how the game should be played and each menu's functionality. Includes a tutorial of an in-game scenario which has a falling enemy from the top of the screen and the player entering a solution to a math problem.

Section 6: References

Additional resources used in development.

2 Overall Description

This section will cover the general context of the game *Aerial Areas*, the project developed by our group. AA is an educational game that is meant to help students learn geometry through repetitive practice. This section explains the perspective of the project, what it is intended to do, the target audience, constraints of development, and the required hardware and dependencies that the users must have.

2.1 Product Perspective

Aerial Areas is an edutainment shoot ‘em up game that aims to reinforce area and perimeter concepts for late-elementary to middle school children. The app will engage players in progressively more efficient problem solving as they get further into the game or increase the difficulty. The player solves area and perimeter questions to destroy parts of the enemies or shapes to clear a wave.

This product is not part of a bigger system. It can be run on both desktop and laptop environments. The user interface will have a text field for players to enter the solutions to the questions for the level they are playing. There will also be interactive parts of the game such as a shop and upgrade screen for the player to click on. Either a mouse or trackpad will be required to click on these parts, and any form of keyboard is necessary to enter the solutions to given problems. Our game runs on Godot and uses OpenGL for rendering. AA will not require an internet connection to run once the game has been downloaded from the webpage.

2.2 Product Functions

AA provides the user with ample opportunities to interact with the environment. Users can click on enemies or their respective slot on the sidebar to get detailed information regarding what formula the enemies want the user to solve for. The player can also use the up and down arrow keys for the same function. The user can input the solution they come up with and automatically blast the enemy selected if their solution is correct. The user can also click on their stock of powerups to help them win or get a higher score. They can get more powerups by navigating the shop that will show up after every wave of enemies.

AA provides engaging gameplay that can be played repeatedly without getting tiresome. This heavily encourages practice through repetition. As users get more comfortable with the formulas they are presented, they will be met with harder formulas and more enemies, maintaining a level of challenge that would keep the user engaged. There would also be tutorials provided that include in-depth breakdowns of each of the formulas, making them easier to learn.

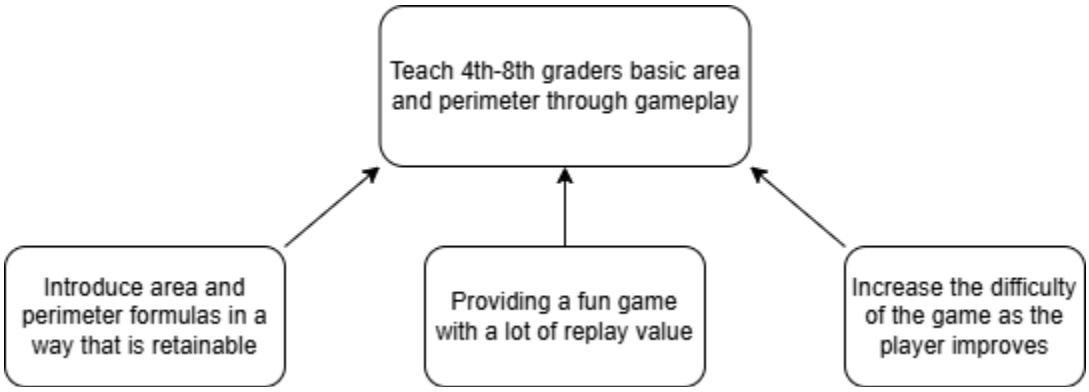

Figure 1: High Level Goal Diagram

2.3 User Characteristics

The intended audience for this game is kids in late elementary school or middle school. The user should have some sort of introductory learning experience with area and perimeter concepts for a better experience. Area and perimeter of sided shapes historically are introduced at the 4th grade level. Circumference and area of a circle is taught around grade 6 and probably not retained well in students until late middle school. The user should be able to take length and width values of a shape and plug these given values into the formulas provided by the game. While experience is recommended, it's not required because this game can also be used to introduce such concepts to interested individuals through its formula sheets and tutorials.

2.4 Constraints

The product utilizes mathematical concepts learned in the 4th to 8th grade of the Massachusetts Mathematical Curriculum Framework and must adhere to any and all standards set by it. The game must be operable by and safe for students in the 4th grade and above and should be simple to install and use even by students with limited computer experience.

2.5 Assumptions and Dependencies

Assumes the user has access to a standard desktop or laptop device. That device should have a modern operating system. A mouse or trackpad along with a keyboard will be required to enter inputs and interact with the game. The user must have a stable internet connection to find and download the game from the web. It is assumed that the player has at least been introduced to the terms shape, area, and perimeter, but they do not have to be proficient or even good. Proficiency in both addition and multiplication is implied. It is expected that the user is fluent in the English language and the United States customary units (inches, feet, etc.).

2.6 Apportioning of Requirements

The initial release of the game includes unique waves, multiple difficulties, a shop system, area and perimeter solving, and a way to track user high scores. A future release will include a webpage version of the game when Godot 4 and C# become compatible with HTML5 and other web browsers. Updated versions will bring potential bug fixes, new difficulties, a progression system, and in-game cosmetics for the player. A potential story mode has been discussed, which would revolve around the main character (the player) saving the Earth from the invading Geometroids from the sky.

3 Specific Requirements

1. The player is presented with a wave of enemies that take on various shapes.
    1.1. Enemies can spawn from random positions above the player.
    1.2. The enemies wait a few seconds before descending upon the player to attack.

Some enemies will go in straight lines; other enemies will bounce from wall to wall.

    1.3. The player starts with one life at the beginning of the game

If an uninterrupted enemy reaches the bottom of your screen, a sound will play, and the player will lose a life.

        1.3.1. If all lives are lost, the game is over.
            1.3.1.1. Transitions to a “Game Over” screen
    1.4. The player can pause the game by pressing a designated key or by selecting a specific button.
2. To combat the enemies, the player must solve the required formula for the given shapes with their respective side lengths given.

The shapes that will be encountered will be squares, rectangles, triangles, and circles.

The formulas that the player must solve will either be area or perimeter.

The sprites of the enemies will distinctly represent the formula they need.

Filled in shapes with black lines will ask for area.

Empty shapes with colored lines will ask for perimeter.

3. The enemies will also have their information displayed in a sidebar.
    3.1. Enemies will be sorted by the order they appear.
    3.2. One enemy will always have their information highlighted.
    3.3. The highlight will provide more information regarding that particular enemy.
    3.4. The player can change which enemy is highlighted by clicking on another enemy in the game, clicking on another segment in the sidebar, or pressing the up or down keys.
4. The player can input a number representing their guess for the answer to a formula.
    4.1. The player’s input can be seen underneath the sidebar containing the enemies.
    4.2. If the player’s input is correct, the target enemy will explode, taking down any other enemies within the blast radius.
    4.3. If any other enemies share the same answer, they will automatically explode as well.
    4.4. The defeat of an enemy will increase the player’s score and give the player gold.

If the player’s answer does not match the highlighted enemy’s formula, the player will receive a penalty that disables their inputs for a few seconds.

5. After all enemies are defeated, a shop will be displayed.

5.1. The player can use the gold they have accrued from defeating enemies to buy powerups or upgrades.

5.2. Powerups are one-time use boosts to give the player extra power for a short duration.

5.3. The player can only have up to three of each powerup at a time.

5.3.1. Powerups include a time freeze, score multiplier, and extra blast radius on explosions.

The time freeze power up halts the movement of the enemies. Their positions will stay for a short period of time.

The score multiplier will multiply the current score at the end of the wave.

5.3.1.1. The extra blast radius power up would increase the size of the explosion for a single wave

5.4. Upgrades are semi-permanent buffs that the player will have until they lose.

5.4.1. Upgrades include more lives, different explosions, among others

Each upgrade has levels. The higher the level number, the more effective the upgrade is. It caps at level 3.

5.4.1.1. More lives upgrade gives you one extra life at level 1 and increases by 1 with each level increase

6. Once the shop is closed, a new wave will start.

6.1. Waves toward the start of the game will only have rectangles and squares.

6.2. Waves toward the middle will have mostly triangles with rectangles sprinkled in.

6.3. Waves later on will have an emphasis on circles, that slowly become an even mix of all of the shapes.

After a set number of waves, a boss wave will start, featuring a large version of the most common shape of that assortment of waves, covered in armor.

6.3.1. The boss will have a list of problems that represent the many pieces of armor.

6.3.2. After the armor is removed, the boss will require the sum of all of the armor pieces' answers.

6.3.3. If the player succeeds, the game will transition to a "Win" screen

If the player fails to do this in time, the boss will take all the player's lives.

7. A tutorial will pop up at the start of the game, and when a new type of enemy is introduced

The first tutorial will explain the controls and the objective of the game.

Each subsequent tutorial will explain the data required for the shape (i.e. length and width).

Subsequent tutorials will explain how to solve for the area and perimeter.

7.1. Subsequent tutorials will show the differences between the enemies that require area to be solved and the enemies that require the perimeter to be solved.

8. All tutorials can be accessed whenever the player wants.

8.1. The player must hit a designated button in the pause menu to do so.

Will contain a high score/leaderboard system.

Every time the player loses or clears every wave the game has to offer, their final score is recorded and displayed in a separate screen on the main menu.

8.2. The player can put in their name when they start a game, and their name will show next to the score. If no name was inputted, they would be named 'Anonymous' in game

9. Music plays in the background

9.1. Different music plays for different parts of the game: Main menu versus Wave Start.

9.2. Music is paused when you pause the game.

Music fanfare when the player wins a wave and a longer fanfare when they win the game.

9.3. Sad music for when the player loses.

10. There will be an options menu for adjusting settings

10.1. The menu will contain a slider to adjust volume settings.

10.2. The menu will contain an option to disable any previously viewed tutorials

10.3. The menu will contain options to disable music and or sound effects

11. There will be a main menu upon the game opening

11.1. The main menu has a start button which will show the difficulty options.

The menu displays four different options for difficulty and will start the game on that difficulty when any are selected.

11.2. The menu has an input field for the player's name

11.3. The menu contains a button to view all formulas relevant to the game

11.4. The menu contains a button to view the high scores list and leaderboards

11.5. The menu has a button to view the options menu

11.6. The menu has a button to quit the game

12. The game contains a pause menu to be used during gameplay

12.1. The pause menu has an option to resume the game

12.2. The pause menu has an option to view all formulas

12.3. The pause menu has an option to view all tutorials

12.4. The pause menu has an option to open the options menu

12.5. The pause menu has an option to return to the main menu

4 Modeling Requirements

4.1 Use Case Diagram

This case diagram shows how the actor, being the player, interacts with the game. They can perform fundamental game functions like starting, pausing, and closing the game. There is an options menu the player can use to adjust settings for volume and whether or not they want pop-up tutorials. The player can enter their name before starting the game. It will be shown on their high score screen, which the user can access as well.

Each “use case” or user function in the diagram is represented in an oval. There are dotted lines with includes or extends. If a case is included, it implies it’s shared with another case; extends means it’s a separate case of the original.

In game, the player can select an equation, which is an enemy on the screen. That will then bring you to the associated problem where the user can enter a solution for perimeter or area. After completion of a level, the user will be able to open the shop that provides the option to buy helpful upgrades and powerups. The formula reference sheets will be available at all stages of the game for the user to access for reference.

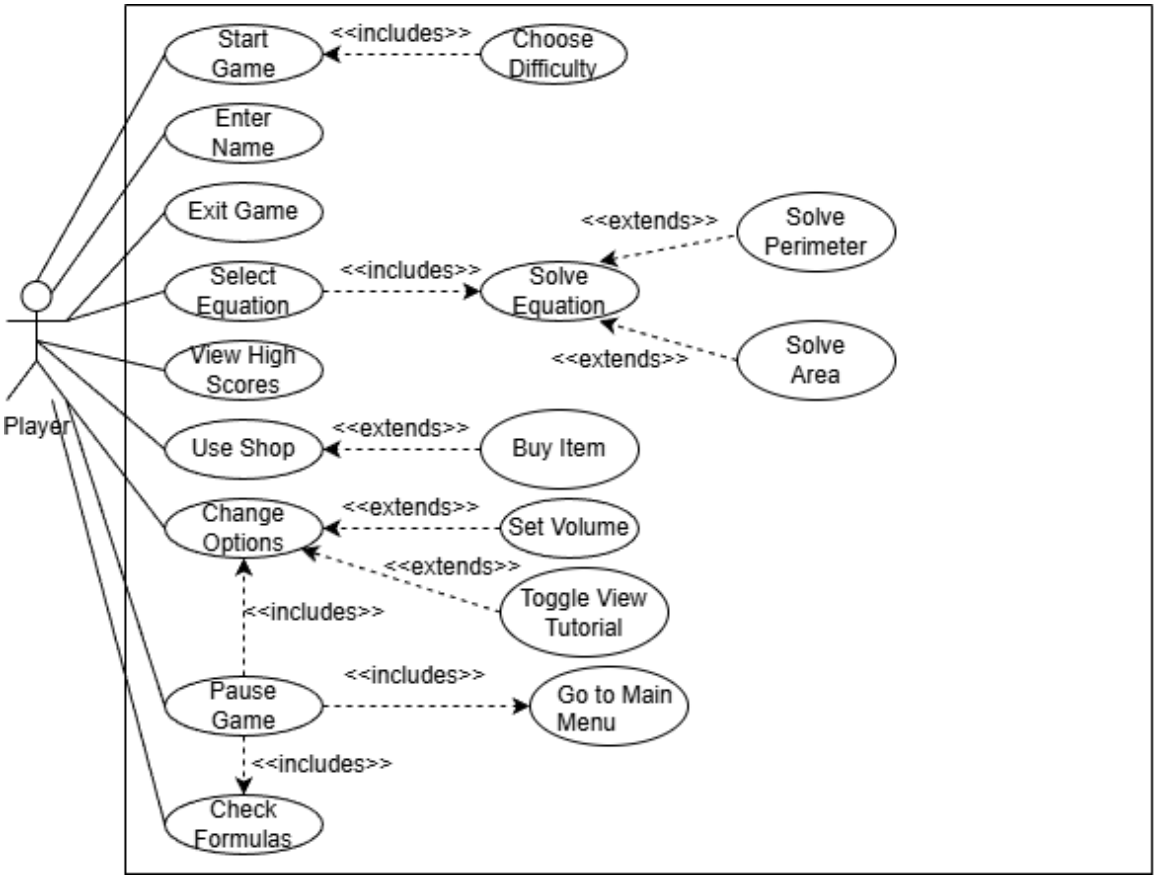

Figure 2: Use Case diagram

4.1.1 Use Case Data Dictionary

Use Case Name:	Start Game
Actors:	Player
Description:	Upon pressing the Start button, the menu will display a list of difficulties that the player can choose from. Pressing one of those difficulties will launch the game.
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 12.1
Uses cases:	Change Difficulty

Use Case Name:	Change Difficulty
Actors:	Player
Description:	The player selects the difficulty they wish to play by clicking the corresponding button, which will then begin the game.
Type:	Secondary
Includes:	Start Game
Extends:	N/A
Cross-refs:	Requirement 12.2
Uses cases:	N/A

Use Case Name:	Change Options
Actors:	Player
Description:	Clicking on the Options button will send the user to a new screen displaying multiple settings that can be changed.
Type:	Primary
Includes:	N/A
Extends:	N/A

Cross-refs:	Requirement 11
Uses cases:	Set Volume, Toggle View Tutorial

Use Case Name:	Set Volume
Actors:	Player
Description:	The player uses the volume slider to adjust the volume of all audio.
Type:	Secondary
Includes:	N/A
Extends:	Change Options
Cross-refs:	Requirement 11.1
Uses cases:	N/A

Use Case Name:	Toggle View Tutorial
Actors:	Player
Description:	The player selects the checkbox in the options menu to disable or reenable any tutorials they have previously seen being shown again.
Type:	Secondary
Includes:	N/A
Extends:	Change Options
Cross-refs:	Requirement 11.2
Uses cases:	N/A

Use Case Name:	Exit Game
Actors:	Player
Description:	The player selects the “Quit Game” button, which then closes the game.
Type:	Primary
Includes:	N/A
Extends:	N/A

Cross-refs:	Requirement 12.7
Uses cases:	N/A

Use Case Name:	Select Equation
Actors:	Player
Description:	During the game, the user can click on any of the segments in the sidebar or an enemy on the screen. The up and down keys can also be used to select adjacent segments. Doing so will highlight the corresponding segment and provide more information about the corresponding formula. This will also be the formula that the player will provide an input for.
Type:	Primary
Includes:	Solve Equation
Extends:	N/A
Cross-refs:	Requirement 3.4
Uses cases:	N/A

Use Case Name:	Solve Equation
Actors:	Player
Description:	The user can input their answer to the solution. Their solution will be compared to the answer that the enemies carry internally, which is calculated based on the type of equation the enemies want. That would be either area or perimeter.
Type:	Secondary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 4
Uses cases:	Select Equation, Solve Perimeter, Solve Area

Use Case Name:	Solve Perimeter
Actors:	Player
Description:	For enemies with perimeter as their designated formula, the perimeter would automatically be calculated internally and will be compared to the player’s input to see if they match. If they do

	match, the enemy explodes. If they do not match, the player cannot enter another guess for an allotted amount of time.
Type:	Tertiary
Includes:	N/A
Extends:	Solve Equation
Cross-refs:	Requirement 4
Uses cases:	N/A

Use Case Name:	Solve Area
Actors:	Player
Description:	For enemies with area as their designated formula, the area will automatically be calculated internally and will be compared to the player's input to see if they match. If they do match, the enemy explodes. If they do not match, the player cannot enter another guess for an allotted amount of time.
Type:	Tertiary
Includes:	N/A
Extends:	Solve Equation
Cross-refs:	Requirement 4
Uses cases:	N/A

Use Case Name:	View High Scores
Actors:	Player
Description:	From the main menu, when the user presses the High Scores button, the highest scores attained on that save will be displayed. They will be organized by the difficulty, and only scores corresponding to the selected difficulty will be displayed. Clicking on the difficulties displayed would change which scores are displayed as well.
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 12.5

Uses cases:	N/A

Use Case Name:	Use Shop
Actors:	Player
Description:	The player will periodically visit the shop during gameplay, which provides multiple buttons for players to press. The player can exit the shop whenever they like.
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 5
Uses cases:	Buy Item

Use Case Name:	Buy Item
Actors:	Player
Description:	If the player has enough currency, which is acquired through defeating enemies, the player can click on any upgrades or powerups they can afford to add the purchase to their inventory. Upgrades are permanent; powerups are single-use consumables.
Type:	Secondary
Includes:	N/A
Extends:	Use Shop
Cross-refs:	Requirement 5.1
Uses cases:	Use Shop

Use Case Name:	Pause Game
Actors:	Player
Description:	The player presses the escape key on their keyboard or clicks on the on-screen pause button to pause the game. This freezes any current in-game action.
Type:	Primary
Includes:	Check Formula, Go to Main Menu, Change Options

Extends:	N/A
Cross-refs:	Requirements 1.5, 13
Uses cases:	Check Formula, Go to Main Menu, Change Options

Use Case Name:	Go to Main Menu
Actors:	Player
Description:	The player selects the “Go to Main Menu” button, which takes them out of the current game screen back to the main menu. Progress is not saved, and the user cannot go back.
Type:	Secondary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 13.5
Uses cases:	N/A

Use Case Name:	Check Formulas
Actors:	Player
Description:	The game opens a menu that shows the player all the formulas for perimeter and area. Use the arrow buttons on the screen to switch between formulas of different shapes. For example, one sheet will show the formulas for a rectangle, along with an example for each.
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirements 12.4 and 13.2
Uses cases:	N/A

4.2 Class Diagram

The class diagram shows all the object classes that make up Aerial Areas. Our game logic handles gameplay aspects. For example, the user entering an answer, what that means for how the world is going to update, the game startup itself. Our user interface deals with changing menus, such as the user traversing through the main menu, or pausing the game. The wave consists of several enemy objects, which have an

associated problem, and problems are broken down by shapes. The player has upgrades and powerups that can be bought from the shop object. These objects made up the entirety of our game's functionality.

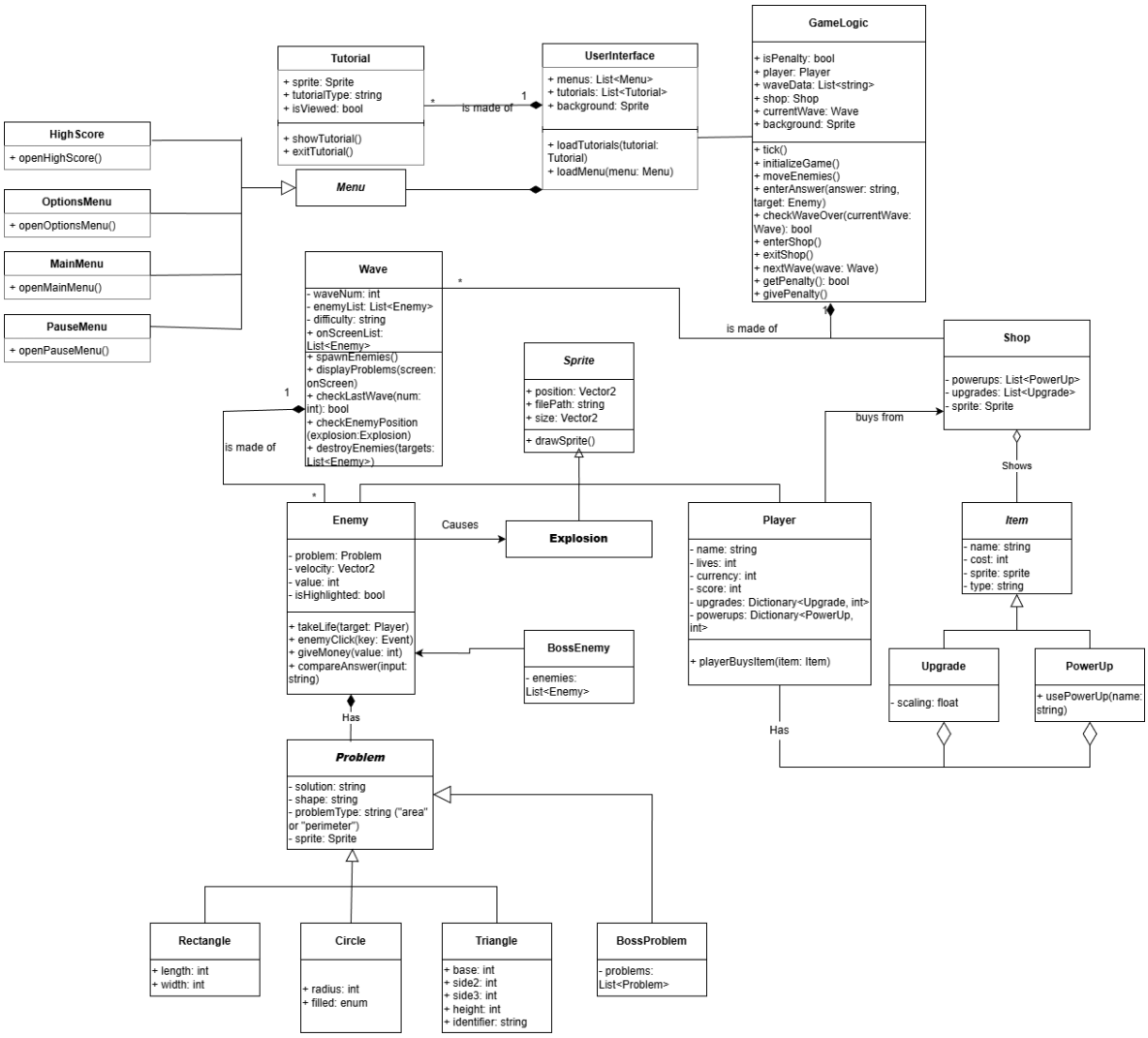

Figure 3: Class diagram

4.2.1 Class Diagram Dictionary

Element Name	Description
GameLogic	GameLogic takes care of all the gameplay elements in the background, such as spawning waves of enemies, moving the enemies, processing inputs, as well as many other functions.
Attributes	

IsPenalty: bool	If the player has a penalty currently active, the value is true
player: Player	An object to represent what the user has earned or can use during the game.
waveData: List<string>	A list of strings read from a given text file. Each line of the text file has each of the enemies for their respective wave.
shop: Shop	The shop that will be displayed after each wave.
currentWave: Wave	The wave contains data processed from waveData, and carries all the information needed for the current wave in game.
background: Sprite	A big sprite in the back that makes the game look cooler.
Operations	
tick(): void	Processes everything that happens in each frame.
initializeGame()	Sets all elements of the game itself to its default state
moveEnemies()	Handles the enemies' movement
enterAnswer(answer: string, target: Enemy)	During the game, an enemy will always be highlighted, either by default or by the user's inputs. That enemy will be the target. This function will pass the answer that was inputted to the enemy.
checkWaveOver(currentWave: Wave): bool	Checks to see if the current wave has enemies left.
enterShop()	Displays the shop and allows the user to interact with it.
exitShop()	Hides the shop, disables the user from interacting with it, and allows the game to continue.
nextWave(wave: Wave)	Processes the information in the given wave.
getPenalty(): bool	Checks whether or not a penalty is active. If so, it returns true.
givePenalty()	Gives the player a penalty and changes all related fields.
Relationships	It is made of a Shop, multiple Waves and has a UserInterface.

UML Extensions	N/A

Element Name	**Description**
UserInterface	UserInterface handles the changing of menu screens and background sprites. This includes the user navigating the main menu and in game changes such as the user pausing the game.
Attributes	
menus: List<Menu>	A list of all potential menus the user can switch to. This includes the pause menu, main menu, options menu and high score menu.
tutorials: List <Tutorial>	A list of all formula sheets to help the player learn how to solve area and perimeter formulas. For example, there will be a rectangle sheet that shows the area and perimeter formulas, along with an example for each.
background: Sprite	A sprite attribute that holds the sprite for the current menu.
Operations	
loadTutorials(tutorial: Tutorial)	Displays the specified formula sheet on screen. Includes area and perimeter formulas for rectangles, triangles and circles.
loadMenus(menu: Menu)	Displays the specified menu on screen. Includes the main, pause, options and high score menus.
Relationships	UserInterface is related to GameLogic Every tutorial and menu has a UserInterface
UML Extensions	N/A

Element Name	**Description**
Tutorial	Represents all tutorials the player will encounter throughout the game, including formulas.
Attributes	
sprite: Sprite	The sprite which will display the tutorial to the player.
tutorialType: string	A string storing the type of the tutorial, such as a formula, enemy, or controls tutorial.

isViewed: bool	A check if the tutorial has been previously seen, used to determine if it should be disabled when the option to disable viewed tutorials is selected in options.
Operations	
showTutorial()	This function displays the tutorial to the player.
exitTutorial()	This function exits the tutorial once the player is done viewing it and clicks the X button to leave.
Relationships	All tutorials are contained and managed within the UserInterface.
UML Extensions	N/A

Element Name	**Description**
Menu	Menu defines all of the different types of menus as one object
Attributes	N/A
Operations	N/A
Relationships	Menu is a parent class for HighScore, OptionsMenu, MainMenu, PauseMenu. The user interface is made up of a menu.
UML Extensions	N/A

Element Name	**Description**
HighScore	This class represents and displays the high score list.
Attributes	N/A
Operations	
openHighScore()	Displays the high score list to the player until they exit the menu.
Relationships	Subclass of Menu, is contained in and managed by the UserInterface.
UML Extensions	Menu

Element Name	Description
OptionsMenu	This class represents and displays the options menu.
Attributes	N/A
Operations	
openOptionsMenu()	Displays the options menu to the player and handles any changes made to settings by the player, such as volume adjustment or disabling view tutorials until the player clicks the button to leave.
Relationships	Subclass of Menu, is contained in and managed by the UserInterface
UML Extensions	Menu

Element Name	Description
MainMenu	This class represents and displays the main menu.
Attributes	N/A
Operations	
openMainMenu()	Displays the main menu screen for the game and handles the changing of screens from that menu, such as starting and exiting the game, opening the settings, and checking high scores.
Relationships	Subclass of Menu, is contained in and managed by the UserInterface
UML Extensions	Menu

Element Name	Description
PauseMenu	This class represents and displays the pause menu.
Attributes	N/A
Operations	
openPauseMenu()	Displays the pause menu to the player and handles any interactions done within in it such as viewing formulas and

	tutorials, opening the options menu or quitting to the main menu until the player exits the menu by clicking the exit button.
Relationships	Subclass of Menu, is contained in and managed by the UserInterface
UML Extensions	Menu

Element Name	**Description**
Shop	Represents the shop object that the player buys items from in between waves. The shop has a list of power ups and upgrades with associated stocks and prices.
Attributes	
powerups: List<PowerUp>	List of all power ups the shop has to offer.
upgrades: List<Upgrade>	List of all upgrades the shop has to offer.
sprite:Sprite	The sprite that displays the shop to the user
Operations	N/A
Relationships	Shop is an aggregation of Item The shop contains upgrades and powerups. The player buys from the Shop Shop is a part of GameLogic
UML Extensions	N/A

Element Name	**Description**
Item	An abstract class that represents items the player can buy in the shop.
Attributes	
name: string	A string representing the name of the specific item.
cost: int	An integer representing how much it gold the player needs to purchase the item.
sprite: Sprite	The sprite representing the item.

type: string	A string representing which subclass the item is a part of.
Operations	N/A
Relationships	Item is the parent class of PowerUp and Upgrade, and both Shop and Player contains lists and dictionaries respectively of them both.
UML Extensions	N/A

Element Name	**Description**
Upgrade	An upgrade is an item the player can purchase from the shop to increase their attributes.
Attributes	
scaling: float	The multiplier the upgrade gives to the player.
Operations	N/A
Relationships	Upgrade is an extension of item. The player has a number of upgrades.
UML Extensions	Item

Element Name	**Description**
PowerUp	This class represents power ups the player can purchase to provide temporary power boost or ability.
Attributes	N/A
Operations	
usePowerUp(name: string)	This method uses a power up and activates its effect.
Relationships	Subclass of Item, the Player and Shop have a dictionary and list respectively of them.
UML Extensions	Item

Element Name	**Description**

Player	The object associated with the user. The player has all of the information associated with the user's current run.
Attributes	
name: string	The name of the player.
lives: int	The number of lives the player has remaining.
currency: int	The amount of money the player has.
score: int	The total score the player has on the current run.
upgrades: Dictionary<Upgrade, int>	The current level of all upgrades the player has.
powerups: Dictionary<PowerUp, int>	The current amount of all powerups the player has.
Operations	
playerBuysItem(item: Item)	The player is responsible for purchasing items from the shop.
Relationships	Player has a Sprite. Player buys from the Shop. Players have a number of Upgrade(s) and PowerUp(s). Players are related to Enemy.
UML Extensions	Sprite

Element Name	**Description**
Wave	Waves are lists of enemies that the user must clear. One wave will be active at a time, and when all waves are finished, the user has won.
Attributes	
waveNum: int	The number of the current wave.
enemyList: List <Enemy>	A list of all the enemies in the current wave.
difficulty: string	There will be four difficulties: Easy, medium, hard and nightmare. Nightmare difficulty will be locked behind completion of hard difficulty. The number of enemies, the

	speed of the enemies, and buying power of the player will all scale accordingly.
onScreenList: List <Enemy>	Not all the enemies can be on screen at once, or it may be overwhelming to the user. This is a list of all of the enemies that appear on screen in the current wave, but there will be enemies stored in the enemyList that are not on screen.
Operations	
spawnEnemies()	Puts the enemies from enemyList into a state that allows the user to interact with them.
displayProblems (screen: onScreen)	This takes the problems from each enemy that is on the screen and displays them in the sidebar.
checkLastWave(): bool	Checks to see if the wave is the final one, so instead of continuing to the next wave, the game finishes.
checkEnemyPosition (explosion: Explosion)	The given explosion has its position compared to all the on-screen enemies. Any enemies within the blast radius are destroyed.
destroyEnemies (targets: List <Enemies>)	Deletes the enemies from the given list.
Relationships	Waves make up GameLogic, and enemies make up a Wave.
UML Extensions	N/A

Element Name	**Description**
Sprite	This class handles visual depictions for classes that need to be displayed on the screen, such as Enemy, Player and Explosion.
Attributes	
Position: Vector2	A Vector2 that holds the position of the sprite.
filePath: string	A string containing the file path that corresponds to the image being displayed by the sprite.
size: Vector2	A Vector2 that holds the size dimensions of the sprite.
Operations	

drawSprite()	A function that draws the image in the file path to the position with the given size.
Relationships	Parent class of Player, Explosion and Enemy, also used in GameLogic for the background and by Item and Tutorial.
UML Extensions	N/A

Element Name	**Description**
Explosion	Object representation of the explosion sprite generated on the enemy when the player defeats it.
Attributes	N/A
Operations	N/A
Relationships	Explosion is an extension of Sprite
UML Extensions	Sprite

Element Name	**Description**
Enemy	The enemy is a shape or Geometroid on the screen the player must defeat. Each enemy has an associated problem and moves towards the bottom of the screen to harm the player.
Attributes	
problem: Problem	The math problem that is associated with the selected enemy. Each enemy has an area or perimeter question that must be solved in order to destroy it.
velocity: Vector2	The speed that the enemy is falling towards the player.
value: int	The total gold the player gains from defeating the enemy.
isHighlighted: bool	A boolean that checks if the problem for the specified enemy is being solved for.
Operations	
takeLife(target: Player)	If the enemy makes it to the player's location, the player is charged a life from their life count by the enemy.
enemyClick(key: Event)	If the enemy is clicked on, it will become the highlighted enemy and the associated flag will be marked.
giveMoney(value: int)	Drops gold to the user upon death of the enemy.

compareAnswer(input: string)	Compares the answer entered by the user to the actual answer for the question.
Relationships	A number of enemies make up a wave An Enemy causes an Explosion on death An Enemy has a Problem Enemy is the parent class of BossEnemy
UML Extensions	Sprite

Element Name	**Description**
BossEnemy	The subclass of Enemy that represents a boss enemy and its subparts.
Attributes	
enemies: List<Enemy>	The list of enemies that make up the boss enemy and must be solved to attempt to solve the boss problem.
Operations	N/A
Relationships	Contains a list of Enemy objects, is also a subclass of Enemy
UML Extensions	Enemy

Element Name	**Description**
Problem	An area or perimeter math question. A problem can be for a rectangle, triangle, or circle. Each problem has an answer.
Attributes	
solution: string	A string containing the solution to the problem.
shape: string	A string distinguishing the shape the problem is about.
problemType: string(“area” or “perimeter”)	A string distinguishing whether the problem is an area or perimeter problem.
sprite: Sprite	The sprite that represents the problem, displayed in the problem list.
Operations	N/A

Relationships	Problem is the parent class for Rectangle, Triangle, Circle and BossProblem, BossProblem has a list of Problems, and every Enemy has an associated problem.
UML Extensions	N/A

Element Name	**Description**
BossProblem	The class representing a boss problem, which contains a list of regular problems which need to be solved to solve the BossProblem.
Attributes	
problems: List<Problem>	The list of problems that must be solved before the boss problem can be solved.
Operations	N/A
Relationships	Contains a list of Problems, is a subclass of Problem
UML Extensions	Problem

Element Name	**Description**
Rectangle	A class that represents a problem that uses a rectangle.
Attributes	
length: int	An integer for the rectangle's length.
width: int	An integer for the rectangle's width.
Operations	N/A
Relationships	Subclass of Problem.
UML Extensions	Problem

Element Name	**Description**
Triangle	A class that represents a problem that uses a triangle.

Attributes	
base: int	An integer representing the length of the base of the triangle.
side2: int	An integer representing the length of the second side of the triangle.
side3: int	An integer representing the length of the third side of the triangle.
height: int	An integer representing the height of the triangle.
identifier: string	A string representing the type of triangle used in the problem, such as equilateral or isosceles.
Operations	N/A
Relationships	Subclass of Problem
UML Extensions	Problem

Element Name	**Description**
Circle	A class that represents a problem that uses a circle.
Attributes	
radius: int	An integer representing the radius of the circle.
filled: enum	An enumeration that represents if the circle is a full circle, half circle or quarter circle.
Operations	N/A
Relationships	Subclass of Problem
UML Extensions	Problem

4.3 Sequence Diagrams

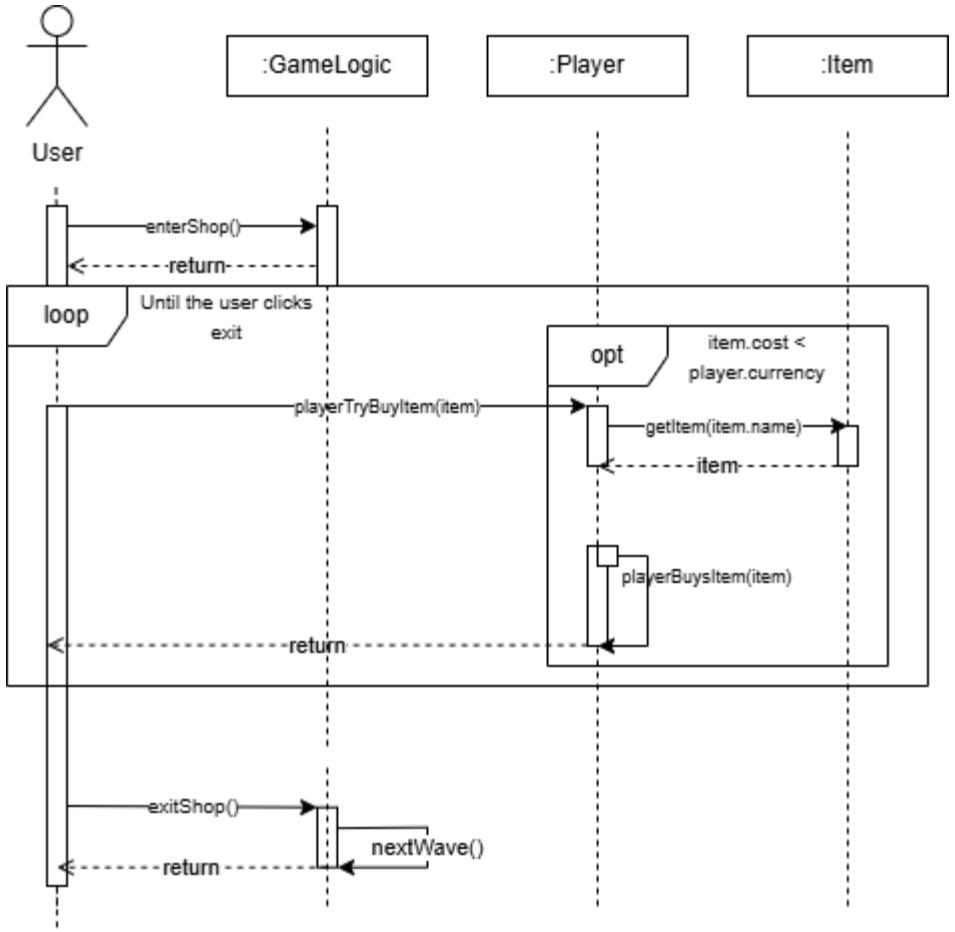

Figure 4: Interacting with the Shop

Figure 4 illustrates the process the user may take when entering the shop. Entering the shop is an event that is periodically triggered by the GameLogic. After the GameLogic sets up the shop, the user is granted control through exclusively mouse clicks. Essentially, the user can try buying items for as long as they desire, but the purchase may only succeed if the user has enough currency. When the user decides to leave the shop, the GameLogic calls a new wave, and the actual game continues.

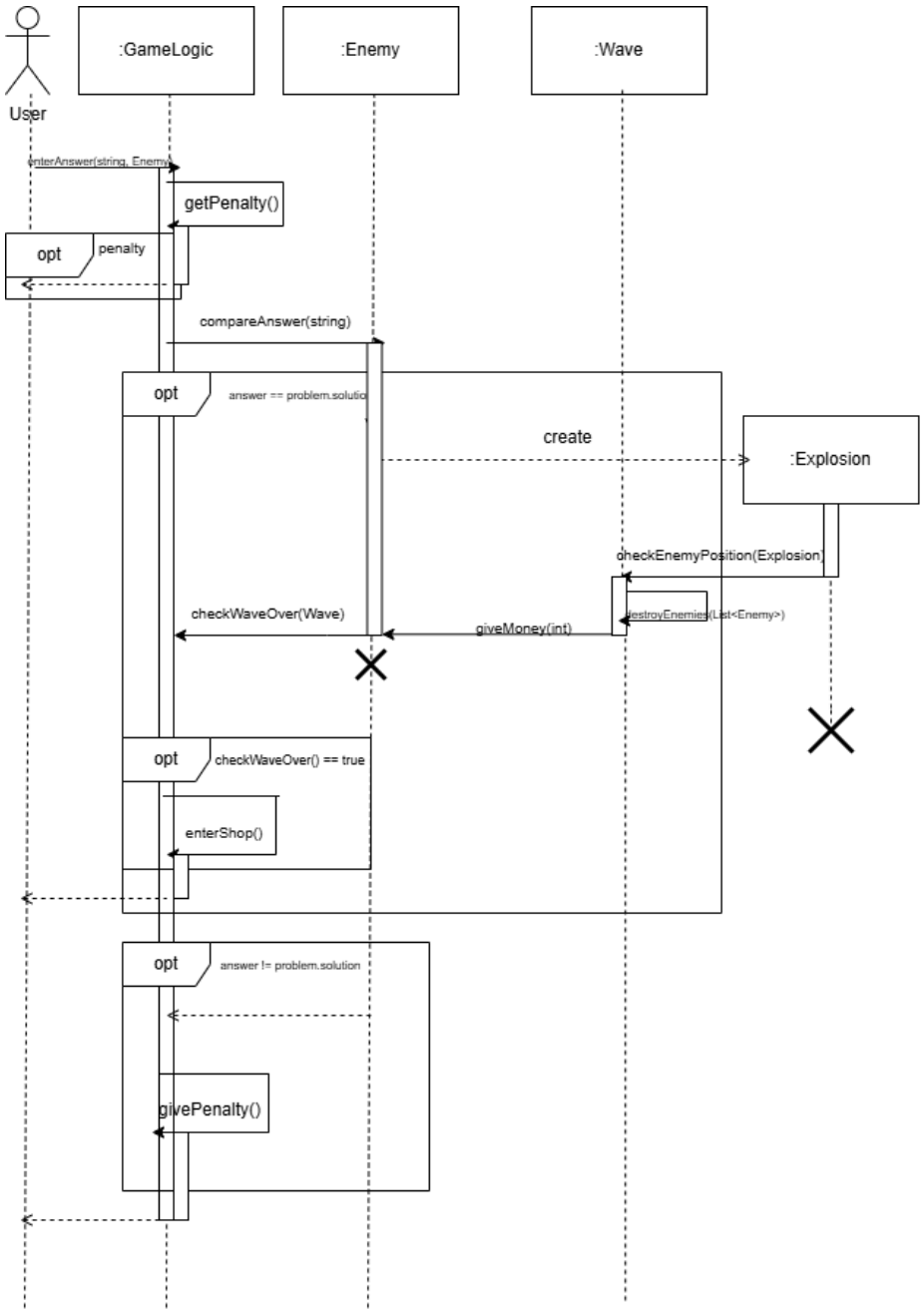

Figure 5: Inputting an Answer

Figure 5 depicts the user attempting to input a solution to a problem. First, the GameLogic checks to see if the player has a penalty applied for a previous wrong answer, if so they cannot submit an answer and it returns. The selected enemy will then check to see if the answer is correct, and if so it creates an explosion, which then has its position checked by the wave so the wave can then find which enemies to destroy. The selected enemies give the player gold before their objects along with the explosion are destroyed. The GameLogic checks if the wave is over and if so, opens the shop, otherwise it returns

control to the player. If the answer the player gave was wrong, the GameLogic gives the player a penalty and returns.

4.4 State Diagram

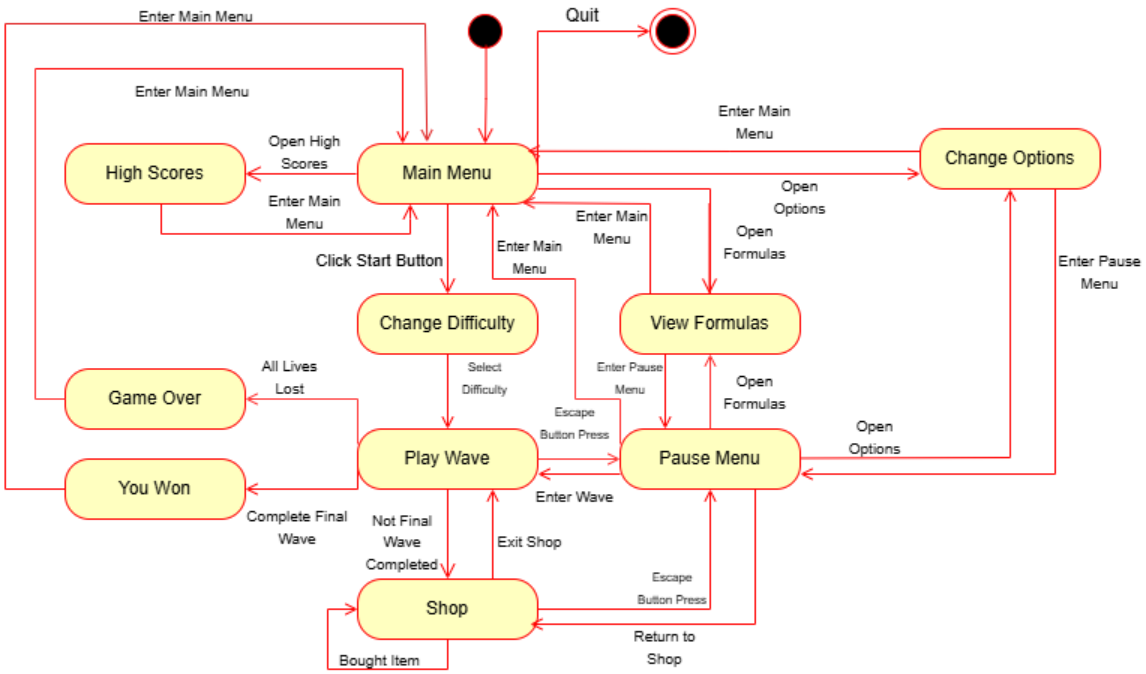

Figure 6: State Diagram

Figure 6 demonstrates the general flow of the game. The user begins in the Main Menu, which provides an assortment of options. High Scores, Change Options, and View Formulas are all simple screens that simply return to the Main Menu when the user is finished. The Main Menu also contains the only way to quit the game. Clicking the start button on the Main Menu will display a list of difficulties on the same screen that the user must press to begin the game. Once the difficulty is selected, the user starts the game, represented by Play Wave. Once a wave is completed, the user will go to the Shop, or the victory screen if the wave completed is the final one. At any moment during a wave or a visit to the shop, the user can access the Pause Menu, which will stop the game and allow the user to access the options menu, formulas menu, and the Main Menu. Once the user is done making changes to their options or viewing formulas, the user will be able to continue the wave. If the user loses too many lives during the wave, the user is sent to a Game Over screen. From both the victory screen represented by You Won, and the Game Over screen, the user will be guided back to the Main Menu.

5 Prototype

The user will be met with the main menu when the program is started. From there, they can go to the options menu through the options button. In this menu, they can adjust the volume slider (defaulting at the highest value of 100) and can toggle the music and sound effects (both defaulted to being on). From the main menu, they can also view local high scores through the high scores button. In the high scores menu, they can click the buttons with difficulties to view the high scores for that difficulty. The user can also view formulas from the main menu or from the pause menu. In the formulas screen, they can view each of the relevant formulas.

When the user clicks the Start button, the difficulties will appear, and once they click on one of those difficulties, the game will start with that difficulty and the player's name. From here, they can enter text in the bottom left and submit the text as their answer with the Enter key. If the answer is correct (the only enemy in our prototype has the answer of 30) the enemy will display text that it has been defeated. The user can also click on the buttons representing their powerups, and text will show that powerups are being used. There is also text showing the player's name and difficulty. We added debug buttons that the user can click that won't be shown for the final product. First, we have a button that takes the user to the shop, where the user can click on buttons, and text shows that they bought the powerups and upgrades. Also, we have victory and defeat screens that show the user's score and wave if they did not win. The victory and defeat screens also allow the player to go back to the main menu.

5.1 How to Run Prototype

To be able to play the prototype, you must go to the website to download the game. Once you are on the website, scroll down to find a link that says, "Download The Game." Click the link and download a zip file. Once it is downloaded, extract the zip file and find the application's executable. Click on it and run it. The game can only be run on Windows 10 or 11.

5.2 Sample Scenarios

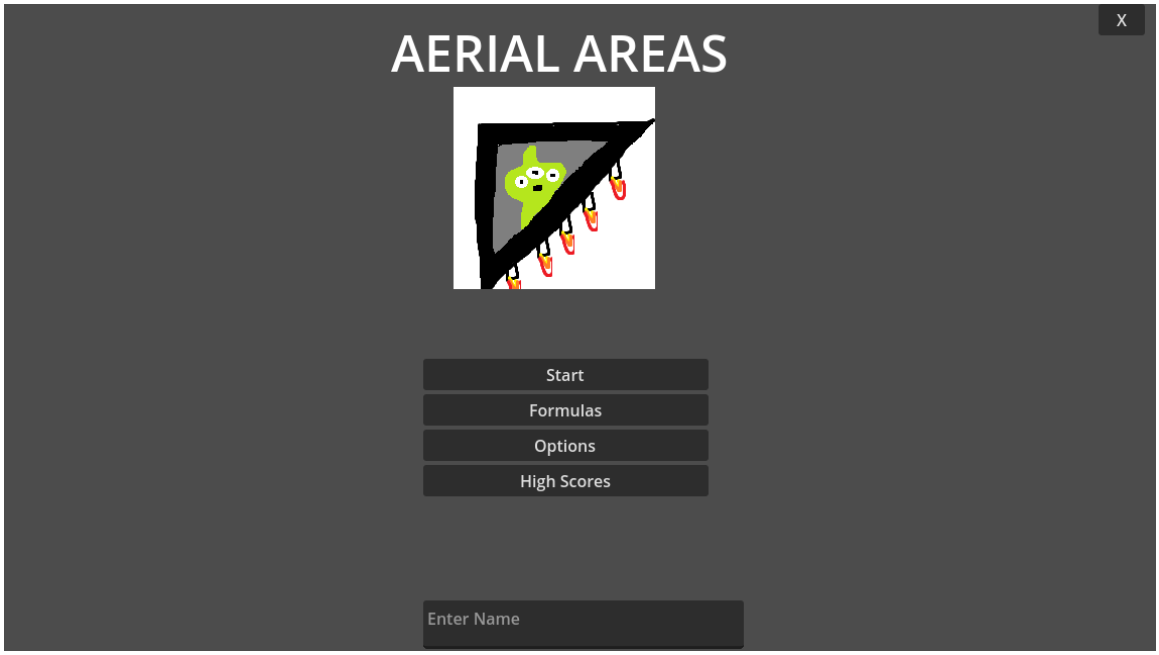

Figure 7: Main Menu

This is the main menu upon starting the program, where the user can start the game, view formulas change options, view high scores, enter their name, and exit the program

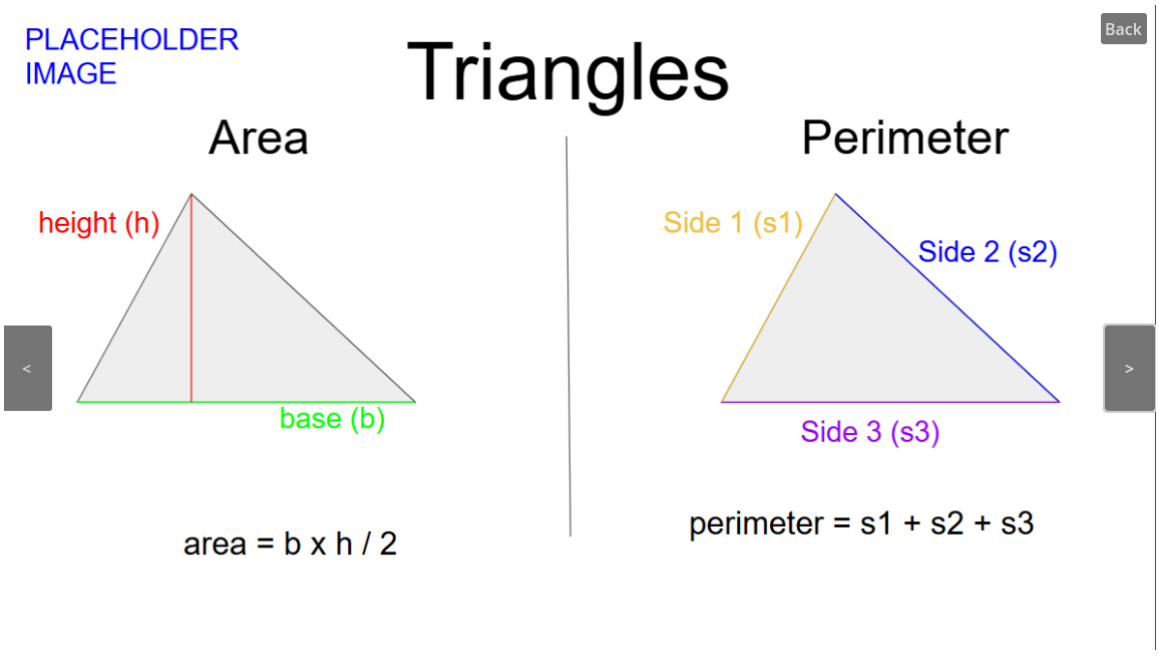

Figure 8: Formulas Page

The user can select the formulas button from the main menu or the pause menu and can use the left and right arrows to view each of the formulas. They can exit this menu and return to their previous menu with the back button.

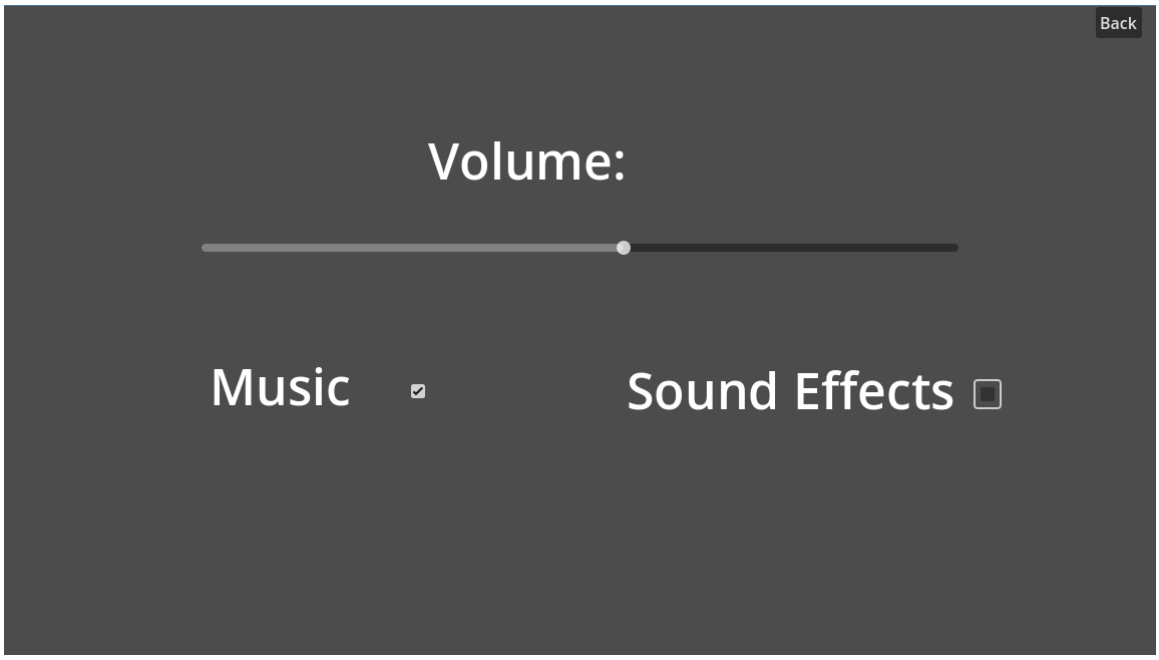

Figure 9: Options Page

The user can enter the options menu from the main menu or the pause menu. Here they can adjust the volume, and toggle music and sound effects. They go back to the previous menu with the back button.

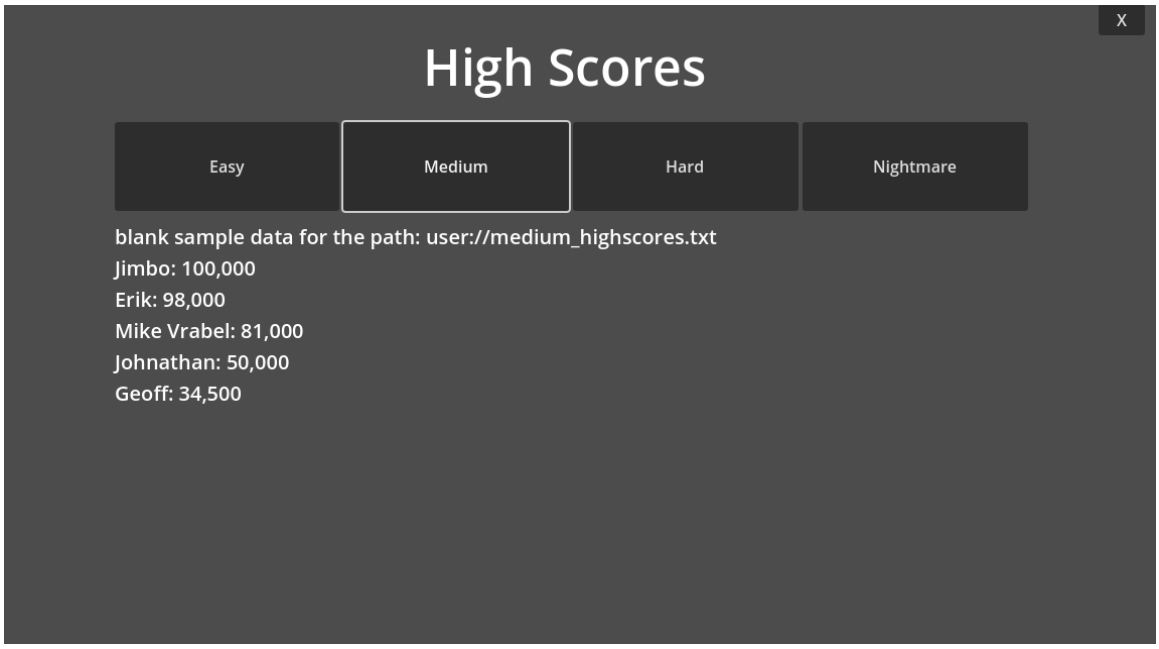

Figure 10: High Scores Page

The user can view high scores stored in text files locally from the High Scores button in the main menu. They can click on each difficulty button to view high scores for that difficulty, as scores for each difficulty are stored separately. They can click the ‘X’ button to return to the main menu.

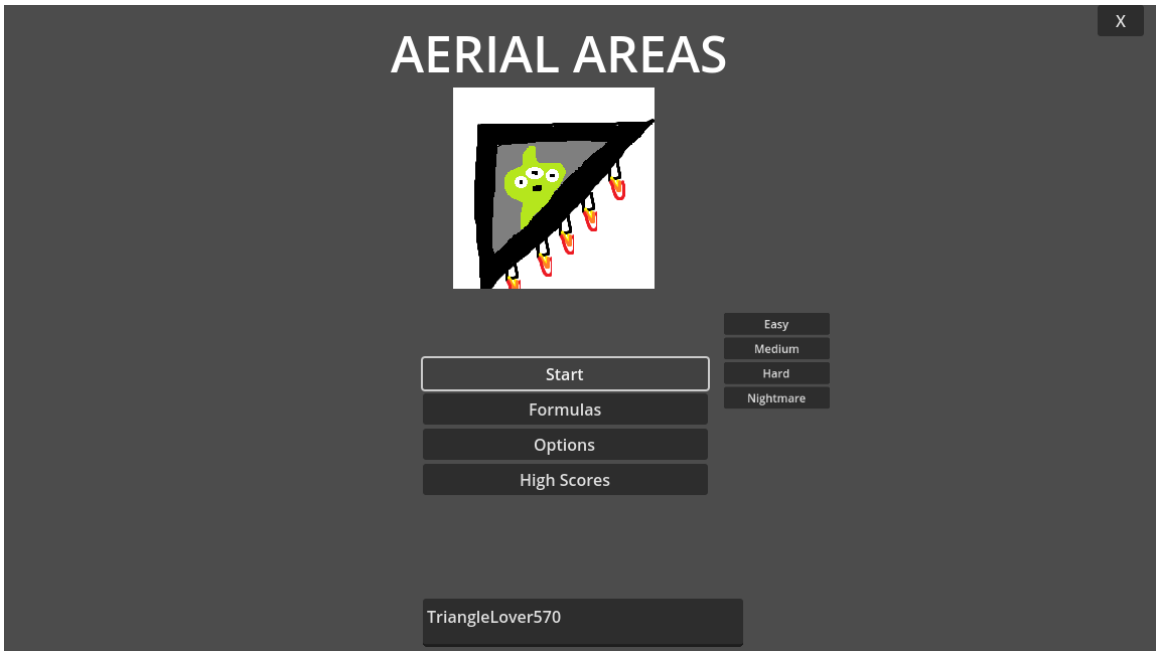

Figure 11: Start Pressed and Entering Name

The user types in their name and clicks start which opens the selection for difficulties. Once they select one of these difficulties, the game will begin.

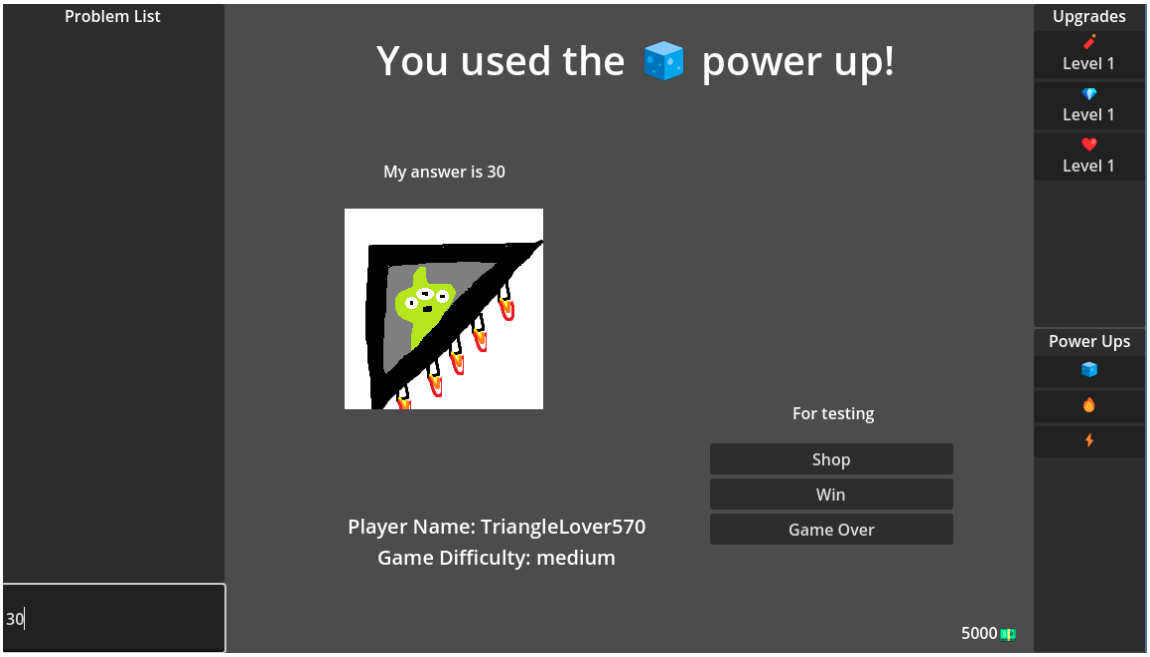

Figure 12: Entering Answer and Using Powerup

The user clicked on the medium difficulty button. Then, the user types in their answer in the bottom left and will press the Enter key to submit. They can also click on powerups which, for now, sets debug text.

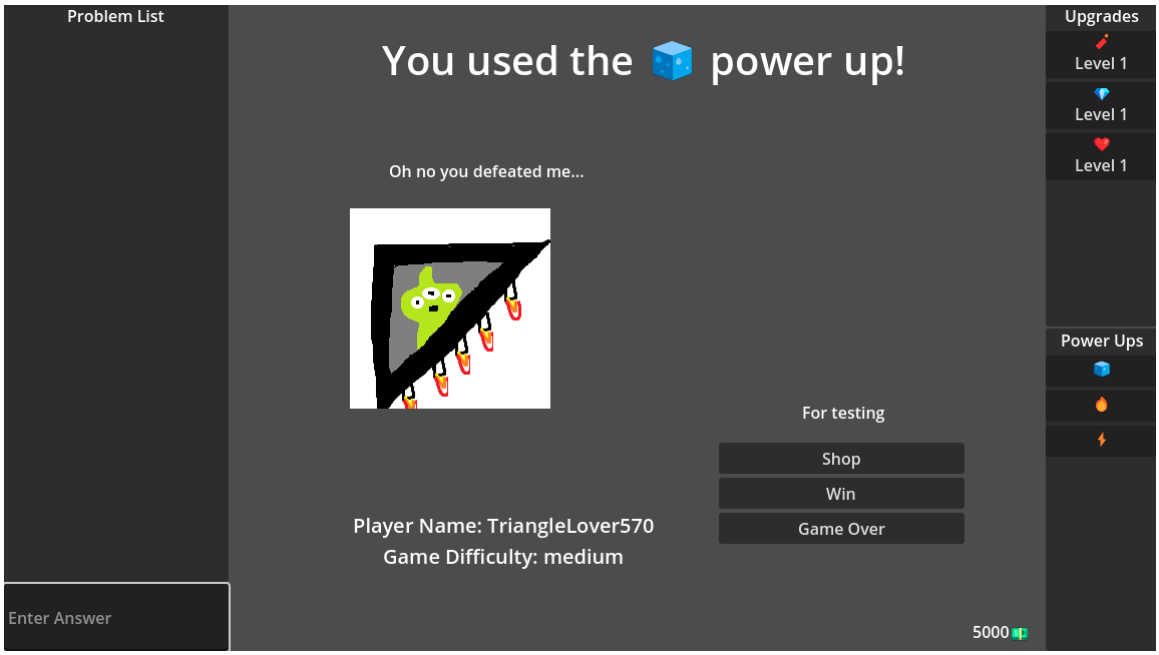

Figure 13: Defeated Enemy

The user pressed enter which ‘defeated’ the enemy and cleared the text edit field.

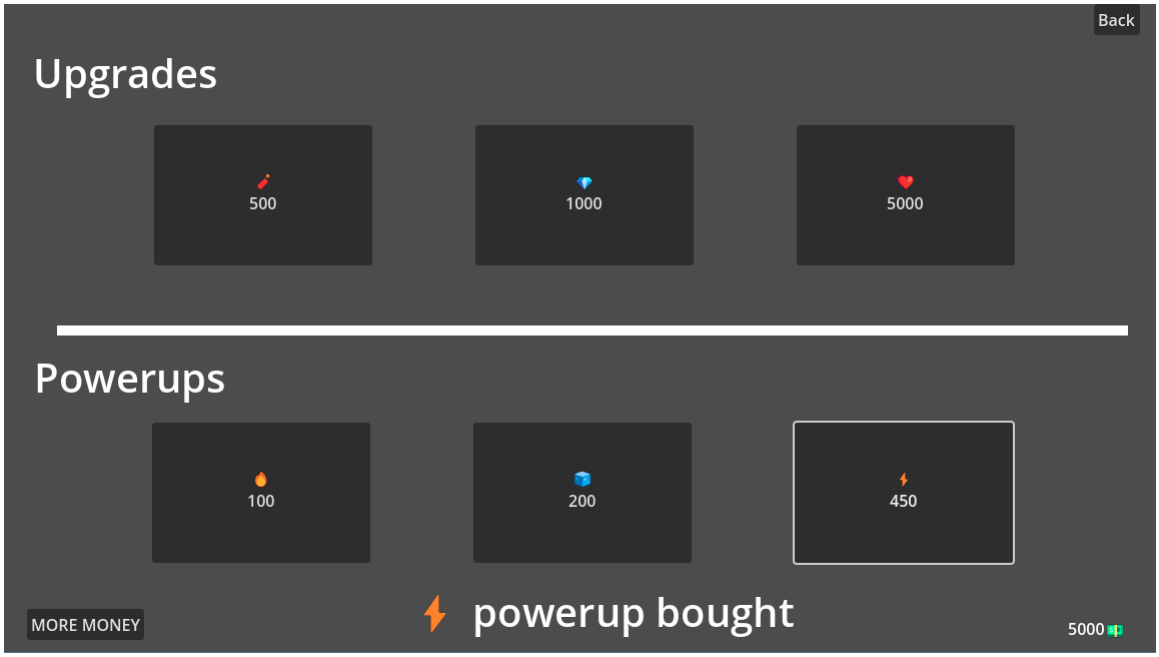

Figure 14: The Shop

The user clicked the Shop button then bought a powerup. This set debug text. The user will normally be able to enter the shop between waves of enemies once implemented. The back button takes them back to the game screen.

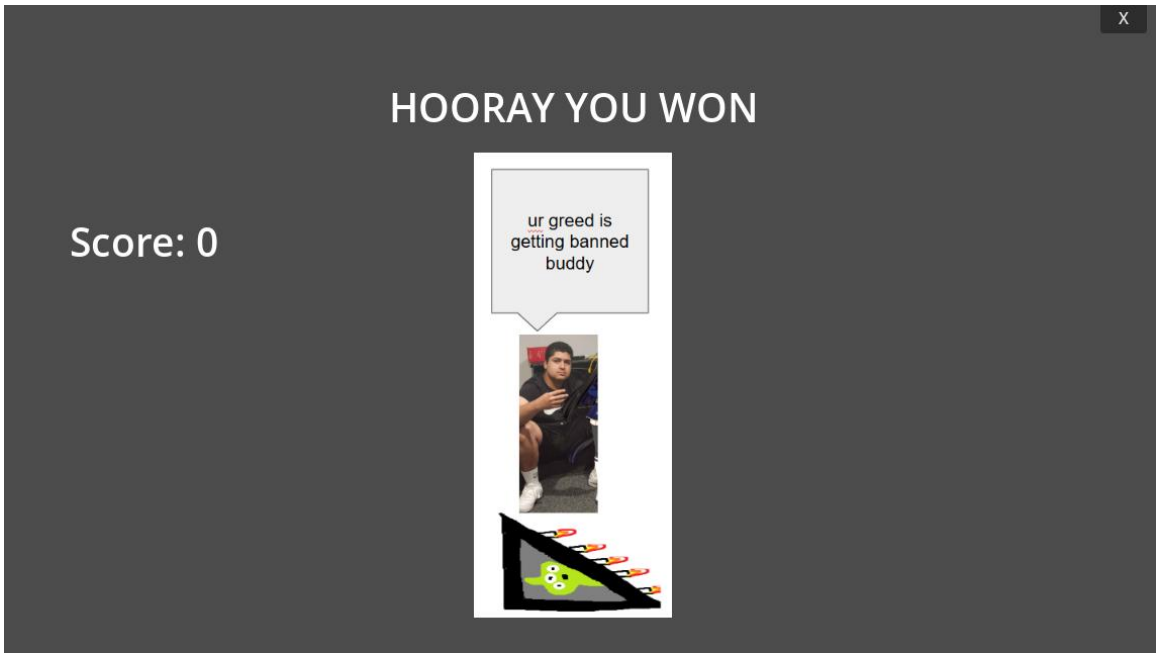

Figure 15: The Victory Screen

From the game screen, the user clicks the ‘Win’ button which takes them to the victory screen. From here, the user can see their score and press the X button to go to the main menu.

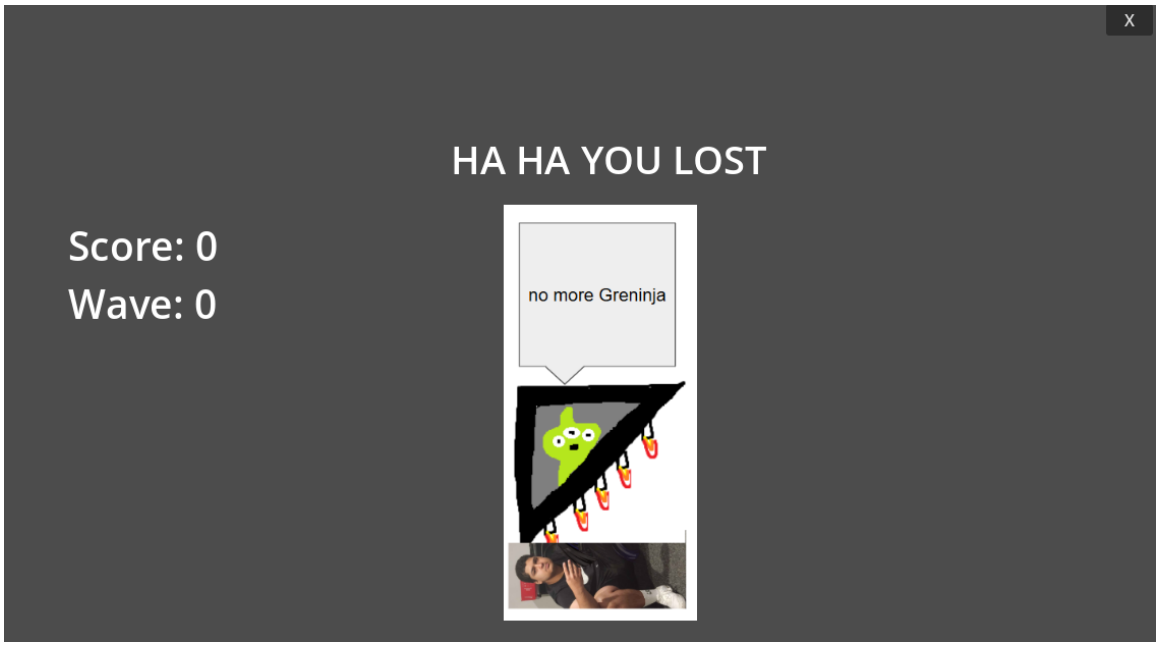

Figure 16: The Defeat Screen

If instead in the game screen the user clicks the ‘Game Over’ button that will take them to this screen, that shows their score and the wave they lost on. They can also press the X button to go to the main menu.

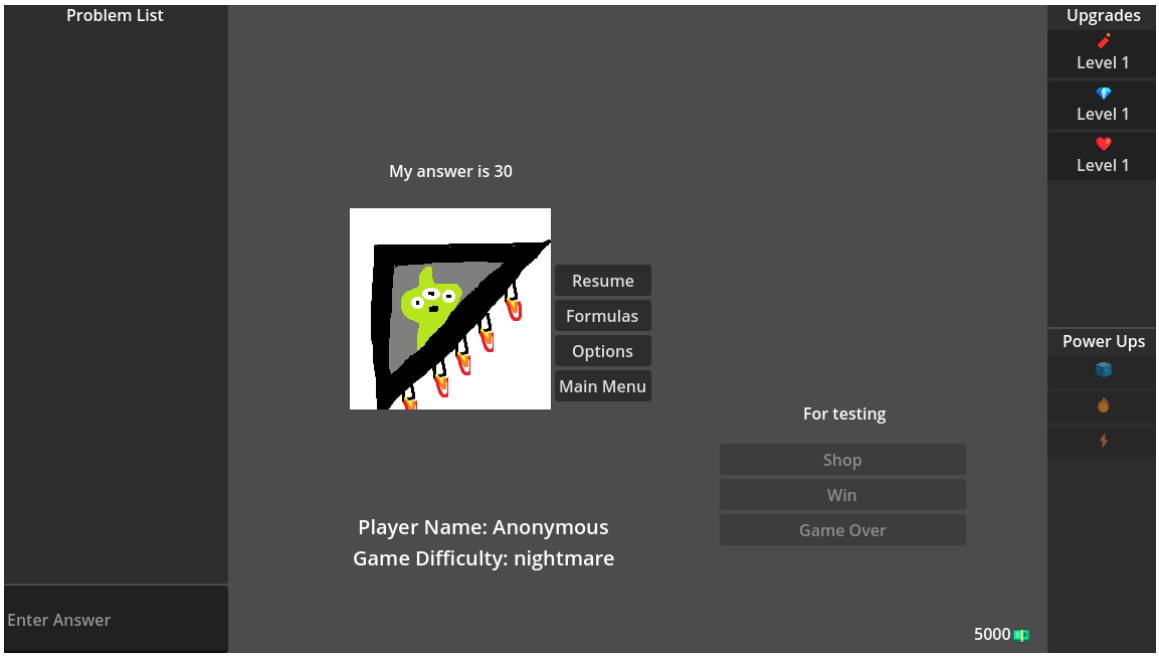

Figure 17: The Pause Menu

From the in-game screen or the shop screen, the user can press the Esc key and bring up the pause menu. They can resume the game, go to the main menu, or view the formulas or options screens shown in Figure 8 and Figure 9.

6 References

[1] Draw.io, "Diagram Software and Flowchart Maker," *drawio.com*, 2024. https://www.drawio.com/

[2] Godot Engine, "Godot Engine – Free and open source 2D and 3D game engine," *Godotengine.org*, 2019. https://godotengine.org/

[3] Massachusetts Department of Elementary and Secondary Education, "Grades Pre-Kindergarten to 12 Massachusetts Curriculum Framework - 2017," 2017. Available: https://www.doe.mass.edu/frameworks/math/2017-06.pdf

[4] O. Aly, B. Brookhart, T. Fitzpatrick, B. Porter, and N. Prentiss, Aerial Areas, https://aerialareas.github.io/Aerial-Areas (accessed Nov. 19, 2025).

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.