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 the players, teachers, parents, 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 contains 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 and gives the recommended system requirements to run AA along with the types of users we are trying to reach. Section 3 consists of the list of requirements that make up AA. These requirements outline clear functionalities and expectations the application must meet. Section 4 is made up of a variety of diagrams that illustrate different aspects of the application. Includes a use case, class and multiple sequence diagrams. Section 5 presents a 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. Finally, Section 6 is for listing any 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, learned in the 4th, 5th, 6th and 7th grades. 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 part of a bigger system as illustrated by Figure 1. 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.

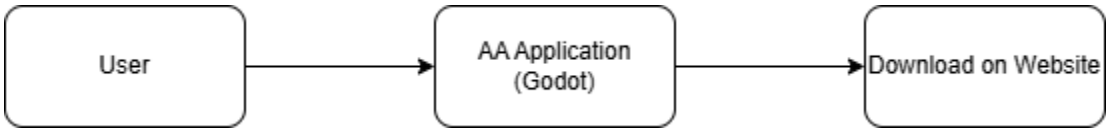

Figure 1: Pictorial Representation of Game System

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. This is all for the purpose of keeping enemies from taking the user's lives. 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.

As illustrated in Figure 2, one of the overall goals of AA is to provide 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.

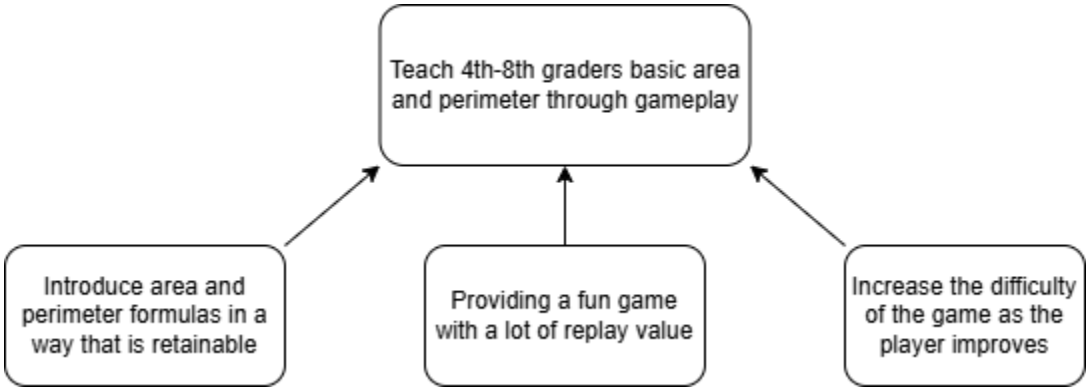

Figure 2: High Level Goal Diagram

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. It is also assumed that the user is proficient in basic addition, multiplication, division and the English language, and that the user is at or above the 4th grade reading level. Finally, the user is expected to be able to operate a computer, keyboard and mouse.

2.4 Constraints

The product utilizes mathematical concepts learned in the 4th to 7th grade of the Massachusetts Mathematical Curriculum Framework and must adhere to all standards set by it. The product mainly utilizes geometrical concepts, though the use of fundamental math skills is also explored. 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. Anything displayed in the game must be age appropriate for children aged 9 and up.

2.5 Assumptions and Dependencies

It is assumed that 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.

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.
    1.3. Some enemies will go in straight lines; other enemies will bounce from wall to wall.
    1.4. The player starts with five lives at the beginning of the game
    1.5. If an uninterrupted enemy reaches the bottom of your screen, a sound will play, and the player will lose a life.
        1.5.1. If all lives are lost, the game is over.
        1.5.2. Upon a game over, the game transitions to a “Game Over” screen
    1.6. 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.
    2.1. The shapes that will be encountered will be squares, rectangles, triangles, circles, half circles and quarter circles.
    2.2. The formulas that the player must solve will either be area or perimeter.
    2.3. The sprites of the enemies will distinctly represent the formula they need.
    2.4. Enemies that require an area to be solved will be displayed as filled in shapes with black lines.
    2.5. Enemies that require a perimeter to be solved will be displayed as empty shapes with colored lines.
3. The enemies will also have their information displayed in a sidebar.
    3.1. Enemies listed in the sidebar will be sorted by the order they appear.
    3.2. One enemy in the sidebar will always have their information highlighted.
    3.3. The enemy that is highlighted in the sidebar will have a target displayed over the enemy’s sprite
    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 the highlighted enemy’s 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 highlighted enemy will explode.
        4.2.1. The explosion will take down any other enemies within the blast radius.

4.2.2. The defeat of any enemy will increase the player's score and give the player money.

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.
      5.2.1. The player can only have up to ten of each powerup at a time.
      5.2.2. Powerups include a time freeze, score multiplier, and a fireball
         5.2.2.1. The time freeze power up halts the movement of the enemies and prevents any additional enemies from spawning temporarily. This effect will stay for five seconds.
         5.2.2.2. The score multiplier will double the score gained from defeating any enemy for ten seconds.
         5.2.2.3. The fireball automatically defeats the selected enemy.
   5.3. Upgrades are semi-permanent buffs that the player will have until they lose.
      5.3.1. Upgrades include more lives, larger explosions, and one that slows all enemies by set percentage.
      5.3.2. Each upgrade has levels. The higher the level number, the more effective the upgrade is. All upgrades start at level one and cap at level five.
      5.3.3. The more lives upgrade gives you five extra lives per level, capping at 25 max lives.
      5.3.4. The larger explosions upgrade increases the size of explosions triggered upon defeating an enemy to potentially hit more enemies.
      5.3.5. The slow upgrade causes all enemies to move slower by an increasing percentage per level.
6. Once the shop is closed, a new wave will start.
   6.1. The first set of waves will only have rectangles and squares.
   6.2. The second set of waves will only have triangles.
   6.3. The third set of waves will only have circles, half circles and quarter circles.
   6.4. The final set of waves will have a random mix of all of the enemies.
7. After two normal waves, a boss wave will start, featuring a large version of the designated shape of that assortment of waves.
   7.1. The boss will have a list of fifteen problems that the user must solve.
      7.1.1. The final boss on wave twelve will have twenty problems for the user to solve
   7.2. If the player fails to do this in time, the boss will take all the player's lives.
8. A tutorial will pop up at the start of the game.

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

9. All formulas for enemies can be accessed whenever the player wants.

9.1. The player must hit the button labelled ‘Formulas’ in the pause menu to do so.

10. The game will contain a high score/leaderboard system.

10.1. 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 if it is higher than the lowest score for that difficulty.

10.2. The player can put in their name when they start a game, and their name will show next to the score.

10.2.1. If no name was inputted, they will be listed as ‘Anonymous’ in game.

11. Music and sound effects will play in the background.

11.1. Music is paused when you pause the game.

11.2. There is music fanfare when the player wins a wave and a longer fanfare when they win the game.

11.3. There is ominous music for when the player loses.

11.4. There are sound effects when the player uses a power up.

11.5. There are sound effects when the player purchases an item or upgrade.

12. There will be an options menu for adjusting settings.

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

12.2. The menu will contain an option to disable the tutorial.

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

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

13.1. The main menu has a button labelled ‘Start’ which will show the difficulty options.

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

13.2.1. The four difficulties in order of easiest to hardest are ‘Easy’, ‘Medium’, ‘Hard’ and ‘Nightmare’.

13.2.2. The ‘Nightmare’ difficulty is locked upon the player starting the game for the first time.

13.2.3. The ‘Nightmare’ difficulty can be unlocked by winning on ‘Hard’ difficulty.

13.3. The menu has an input field for the player’s name.

13.4. The menu has a button labelled ‘Formulas’ to view all formulas relevant to the game.

13.5. The menu has a button labelled ‘High Scores’ to view the high scores list and leaderboards.

13.6. The menu has a button labelled ‘Options’ to view the options menu.

13.7. The menu has a button labelled ‘Quit’ to quit the game.

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

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

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

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

14.4. 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. 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. The player 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.

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.

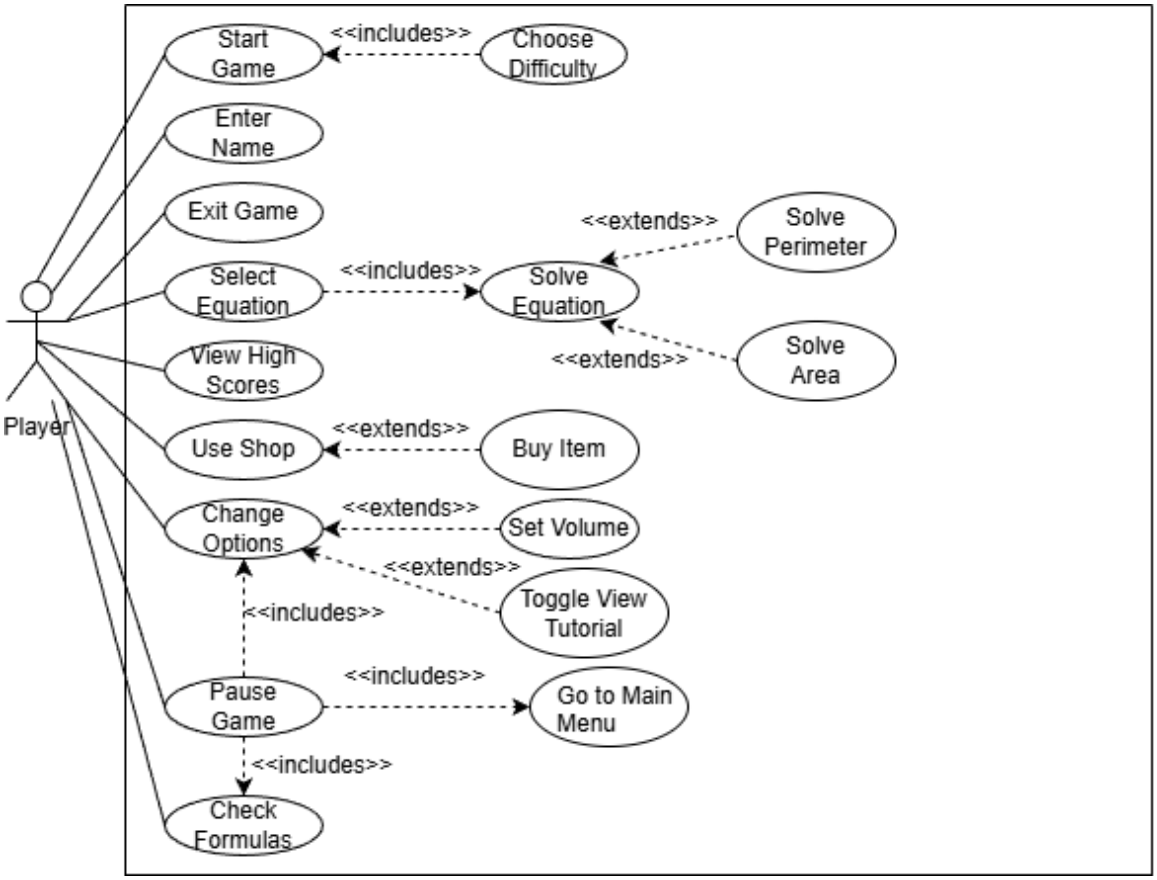

Figure 3: Use Case diagram

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 13.1
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 13.2
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:	Requirements 12, 13.6, 14.3
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 12.1
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 12.2
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 13.7
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 being either area or perimeter. If answer and solution match, the enemy explodes.
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:	The user calculates the answer to a perimeter problem and inputs their solution.

Type:	Tertiary
Includes:	N/A
Extends:	Solve Equation
Cross-refs:	Requirement 4
Uses cases:	N/A

Use Case Name:	Solve Area
Actors:	Player
Description:	The user calculates the answer to an area problem and inputs their solution.
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:	The highest scores attained in the save data 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 will change which scores are displayed as well.
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 13.5
Uses cases:	N/A

Use Case Name:	Use Shop
Actors:	Player

Description:	The player will periodically visit the shop during gameplay, in order to purchase power ups and upgrades. 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:	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:	Requirements 5.1, 5.2, 5.3
Uses cases:	Use Shop

Use Case Name:	Pause Game
Actors:	Player
Description:	The player can pause the game whenever they wish. This freezes any current in-game action and allows the player to do things like check formulas and change options.
Type:	Primary
Includes:	Check Formula, Go to Main Menu, Change Options
Extends:	N/A
Cross-refs:	Requirement 14
Uses cases:	Check Formula, Go to Main Menu, Change Options

Use Case Name:	Go to Main Menu
Actors:	Player
Description:	The player is taken 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 14.4
Uses cases:	N/A

Use Case Name:	Check Formulas
Actors:	Player
Description:	The player is shown all the formulas for perimeter and area for all shapes along with an example, and may switch between any of them at their leisure.
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirements 9, 13.4 and 14.2
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.

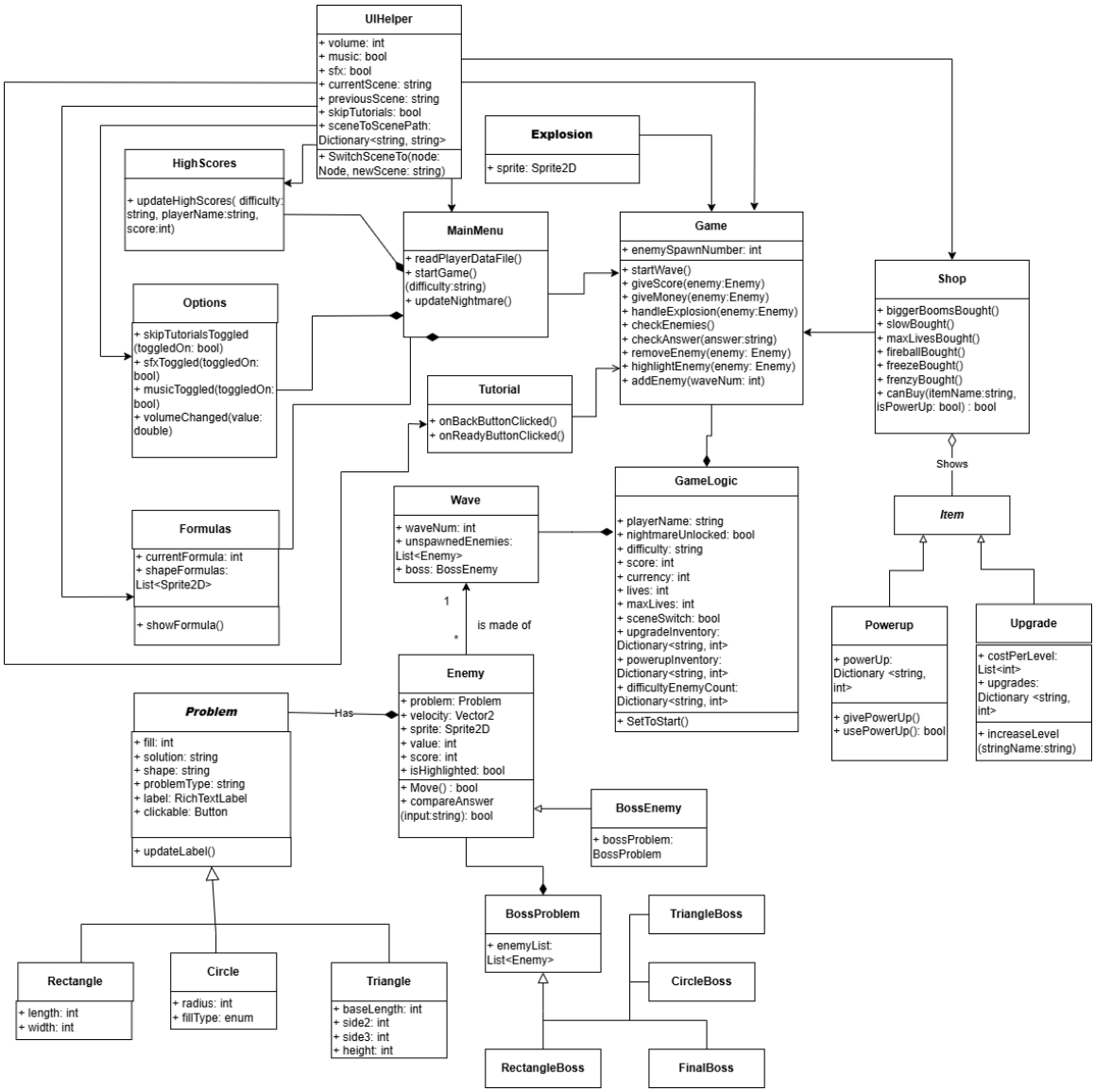

Figure 4: Class Diagram

4.2.1 Class Diagram Dictionary

Element Name	**Description**
GameLogic	GameLogic stores essential data for the game to function.
Attributes	
playerName: string	The name that the user inputs. If no name is entered, it will be “Anonymous” by default.

nightmareUnlocked: bool	Checks to see if the nightmare difficulty has been unlocked. It can only be unlocked by completing hard difficulty.
difficulty: string	The difficulty the user sets the game to. This setting determines how fast the enemies move, how many enemies spawn per wave, how much money is dropped from each enemy, and how much score is earned when an enemy is defeated. There are four difficulties; Easy, medium, hard, nightmare.
score: int	An accruement of points that increases as enemies are defeated. This will be displayed in the High Scores screen if the score is higher than the other scores in the high score file.
currency: int	The amount of money the user currently has. This is used for making purchases in the shop to enhance gameplay.
lives: int	The number of enemies that the player can allow to pass before they lose.
maxLives: int	The maximum number of lives the player can have.
sceneSwitch: bool	Determines whether the scene can switch or not.
upgradeInventory: Dictionary<string, int>	Holds the data regarding how many upgrades the user has purchased.
powerupInventory: Dictionary<string, int>	Holds the data regarding how many powerups the user currently has.
difficultyEnemyCount: Dictionary<string, int>	Holds the amount of enemies to be spawned per wave for each difficulty.
Operations	
setToStart()	Initializes the game so each important piece of data used for gameplay is set to a default state.
Relationships	Wave is stored in GameLogic Game is stored in GameLogic
UML Extensions	N/A

Element Name	Description
Game	Game handles all of the interactions between the objects and makes the game itself function properly.
Attributes	
enemySpawnNumber: int	The number of enemies that must spawn in the wave.
Operations	
startWave()	Spawns in enemies based off of the current wave data.
giveScore(enemy:Enemy)	Increases the user's score based on the enemy's value and difficulty.
giveMoney(enemy:Enemy)	Increases the amount of money the user has based on the enemy's value and difficulty. Unlike the score, this becomes lower as the difficulty increases.
handleExplosion(enemy: Enemy)	Creates an explosion at the given enemy's position, destroying each enemy within the explosion's radius and granting the player score and money based on each enemy.
checkEnemies()	Checks if there are still active enemies in the current wave.
checkAnswer(answer: string)	Compares the answer given to the solution of an enemy's problem. If they match, an explosion is created at the site.
removeEnemy(enemy: Enemy)	Deletes the given enemy.
highlightEnemy(enemy: Enemy)	Puts a target sprite on the given enemy and makes the corresponding problem in the problem list appear yellow rather than white.
addEnemy(waveNum: int)	Adds a random enemy based on the wave number. Only certain shapes are allowed in certain waves.
Relationships	Associates with Shop, Explosion, MainMenu, Tutorial, Enemy, Wave, and is a fundamental part of GameLogic.
UML Extensions	N/A

Element Name	Description

Tutorial	Represents the tutorial that the player sees at the beginning of the game.
Attributes	N/A
Operations	
onBackButtonClicked()	When the back button is clicked, it goes back to the MainMenu.
onReadyButtonClicked()	When the ready button is clicked, the scene changes to the Game scene.
Relationships	Associates with MainMenu and Game.
UML Extensions	N/A

Element Name	**Description**
High Scores	This class represents and displays the high score list.
Attributes	N/A
Operations	
updateHighScores(difficulty: string, playerName: string, score: int)	Manages high scores on the system.
Relationships	A fundamental component of MainMenu.
UML Extensions	N/A

Element Name	**Description**
OptionsMenu	This class represents and displays the options menu.
Attributes	N/A
Operations	
skipTutorials Toggled(toggledOn: bool)	Handles the setting of whether or not the tutorial that plays at the start of the game will be played.

sfxToggled (toggledOn: bool)	Handles the setting of whether or not the game sound effects will be played.
musicToggled (toggledOn: bool)	Handles the setting of whether or not the game music will be played.
volumeChanged (toggledOn: bool)	Handles the setting of changing the volume slider in settings
Relationships	Composition of Menu
UML Extensions	MainMenu

Element Name	**Description**
MainMenu	This class represents and displays the main menu.
Attributes	N/A
Operations	
readPlayerDataFile()	Reads the player_data.txt file stored in the user's AppData, setting values for settings, player name, and nightmare_unlocked.
updateName(newName:string)	Sets the player_data.txt file with the value of the player's name to store between runs of the app.
updateNightmare()	Sets the player_data.txt file with the value of whether nightmare is unlocked to store between runs of the app.
updateVolumeSlider(volume:int)	Sets the player_data.txt file with the value of the volume to store between runs of the app.
updateSettings(index:int, isToggled:bool)	Sets the player_data.txt file with the value isToggled, corresponding to the setting defined by index, between runs of the app.
hideLabel()	Removes the text telling the player to beat hard mode to unlock nightmare mode after three seconds of it being shown.

startGame(difficulty:string)	Stores the difficulty in GameLogic, and sets the scene to Tutorial or Game depending on if Skip Tutorials is toggled.
Relationships	The player interacts with UI from MainMenu MainMenu can set the scene to Tutorial, Options, Formulas, HighScores, or Game Win, GameOver, Formulas, Options, Game, Tutorial can set the scene to MainMenu
UML Extensions	N/A

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	N/A
Operations	
biggerBooms Bought()	The user purchases the upgrade “Bigger Booms”
slowBought()	The user purchases the upgrade “Bigger Booms”
maxLivesBought()	The user purchases the upgrade “Max Lives”
fireballBought()	The user purchases the upgrade “Fireball”
freezeBought()	The user purchases the upgrade “Freeze”
frenzyBought()	The user purchases the upgrade “Frenzy”
canBuy(itemName: string, isPowerUp: bool): bool	Checks to see if the player can afford to buy the powerup or upgrade by comparing the object to its price.
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	N/A
Operations	N/A
Relationships	Item is the parent class of PowerUp and Upgrade.
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	
costPerLevel: List<int>	A list of cost increases (not exact values) for each upgrade level.
upgrades: Dictionary <string, int>	Dictionary representation of upgrades by name and current price in the store. (Factors in the difference per level).
Operations	
increaseLevel(stringName: string)	Increases the level of an upgrade when purchased.
Relationships	Upgrade is an extension of item. Game logic 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
powerUp: Dictionary <string, int>	A dictionary representation of all different kinds of power up by name and price.

Operations	
usePowerUp():bool	This method uses a power up and activates its effect.
givePowerUp()	Gives a power up purchased to the player.
Relationships	Subclass of Item. Game logic has a number of powerups.
UML Extensions	Item

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.
unspawnedEnemies: List <Enemy>	A list of all the enemies in the current wave.
boss: BossEnemy	The wave's boss enemy, which is made up of
Operations	N/A
Relationships	Waves make up GameLogic, and enemies make up a Wave.
UML Extensions	N/A

Element Name	**Description**
Explosion	Object representation of the explosion sprite generated on the enemy when the player defeats it.
Attributes	
sprite: Sprite2D	PNG representation of the explosion on screen.
Operations	N/A
Relationships	N/A
UML Extensions	N/A

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.
score: int	The total score the player gains from defeating an enemy.
sprite: Sprite2D	PNG representation of an enemy on the screen.
isHighlighted: bool	A boolean that checks if the problem for the specified enemy is being solved for.
Operations	
move(): bool	Controls the enemy's movement down the world
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	N/A

Element Name	Description
BossEnemy	The subclass of Enemy that represents a boss enemy and its subparts.
Attributes	
bossProblem: BossProblem	The associated boss problem for the boss enemy.
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	
fill: int	Integer enum value for the type of circle (1: whole, 2: semi, 3: quarter).
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.
label: RichTextLabel	Label that holds the visual representation of the problem in the problem list.
clickable:Button	Button object that allows the user to click on problems in the problem list.
Operations	
updateLabel()	Updates the text highlighted in the specified problem label.
Relationships	Problem is the parent class for Rectangle, Triangle and Circle, 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 enemies which need to be solved to solve the BossProblem.
Attributes	

enemyList: List<Enemy>	The list of enemies associated with the boss problem.
Operations	N/A
Relationships	Contains a list of Problems
UML Extensions	N/A

Element Name	**Description**
RectangleBoss	The class representing the rectangle boss problem, which contains a list of rectangle enemies which need to be solved to solve the RectangleBoss.
Attributes	
Operations	N/A
Relationships	Subclass of BossProblem
UML Extensions	BossProblem

Element Name	**Description**
TriangleBoss	The class representing the triangle boss problem, which contains a list of triangle enemies which need to be solved to solve the TriangleBoss.
Attributes	
Operations	N/A
Relationships	Subclass of BossProblem
UML Extensions	BossProblem

Element Name	**Description**
CircleBoss	The class representing the circle boss problem, which contains a list of circle enemies which need to be solved to solve the CircleBoss.
Attributes	
Operations	N/A

Relationships	Subclass of BossProblem
UML Extensions	BossProblem

Element Name	**Description**
FinalBoss	The class representing the final boss problem, which contains a list of all types of enemies which need to be solved to solve the FinalBoss.
Attributes	
Operations	N/A
Relationships	Subclass of BossProblem
UML Extensions	BossProblem

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	
baseLength: 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.
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.
fillType: 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

Element Name	**Description**
UIHelper	A class that handles switching between scenes in Godot and also keeps track of audio level and whether audio and or tutorials are disabled.
Attributes	
volume: int	An integer that keeps track of the current volume level.
music: bool	A Boolean that tracks whether or not music is disabled.

sfx: bool	A Boolean that tracks whether or not sound effects are disabled.
currentScene: string	A string that signifies the current Godot scene.
previousScene: string	A string that signifies the previous Godot scene, used for returning from the pause menu.
skipTutorials: bool	A Boolean used to determine if the tutorial should be skipped when starting the game.
sceneToScenePath: Dictionary<string, string>	A Dictionary containing keys of strings which represent the names of scenes used in current_scene and previous_scene, and the values are the corresponding file paths for the scenes.
Operations	
SwitchSceneTo(node: Node, new_scene: string)	This function switches the scene from the current node to the scene represented by new_scene.
Relationships	Used to switch between all scenes and as such is called in Game, MainMenu, Shop, Tutorial, Formulas, HighScores and Options.
UML Extensions	N/A

Element Name	**Description**
Formulas	A class that handles the Formulas scene in Godot that displays formulas of the shapes.
Attributes	
currentFormula: int	Integer index for the displayed formula.
shapeFormulas: List<Sprite2D>	A list of all the formulas to be displayed in the Formulas scene.
Operations	

showFormula()	This function allows the different formulas to be displayed based on the user switching it by the arrow buttons. Only one can be displayed at a time.
Relationships	Can be accessed from the Main Menu or during the Game when it is paused.
UML Extensions	N/A

4.3 Sequence Diagrams

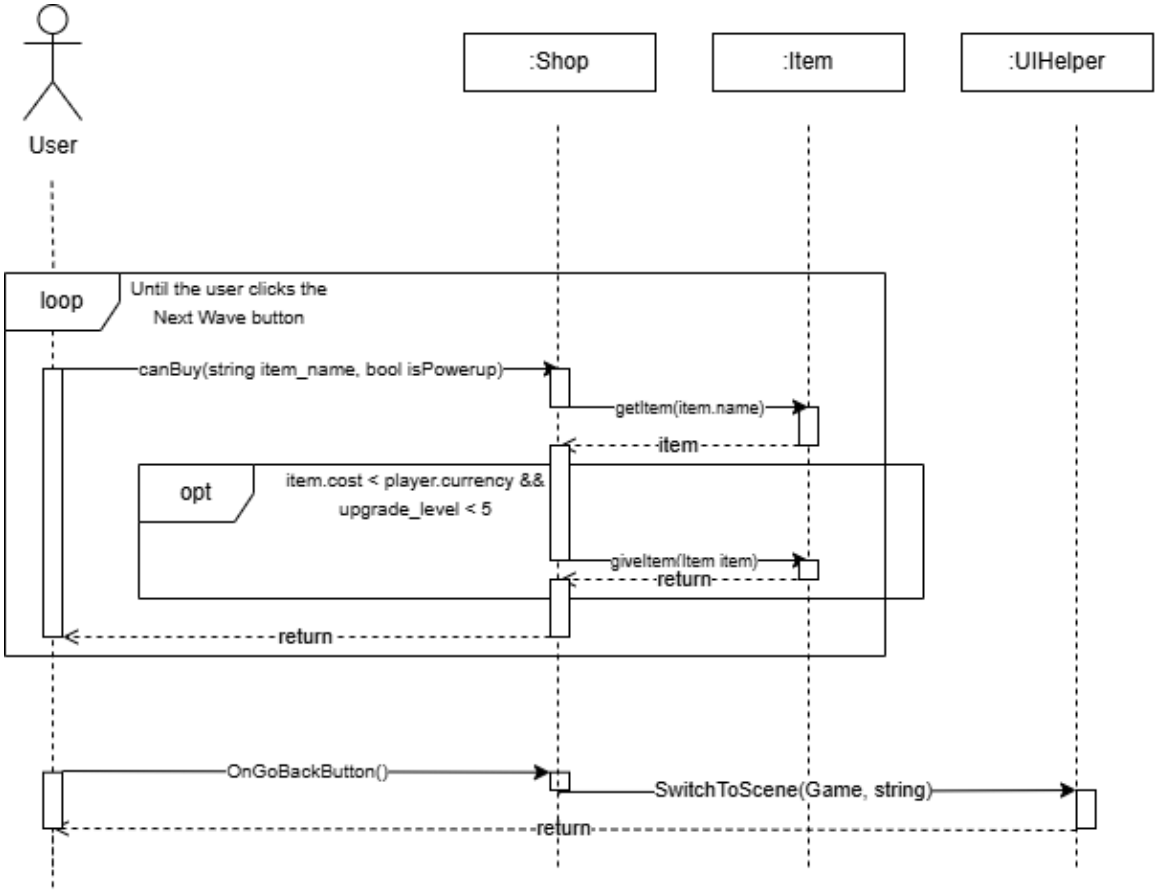

Figure 5: Interacting with the Shop

Figure 5 illustrates the process the user may take when entering the shop. They can click on Powerups and Upgrades to attempt to buy them whether they are able to or not. This calls the canBuy function which gives the item name and whether it is a Powerup or an Upgrade to the Shop Class. If the player has enough money, the item is updated in the corresponding inventory in the giveItem function. When the player wishes to leave the shop, they click the 'Next Wave' button, which calls OnGoBackButton(), which tells UIHelper to change the scene back to Game.

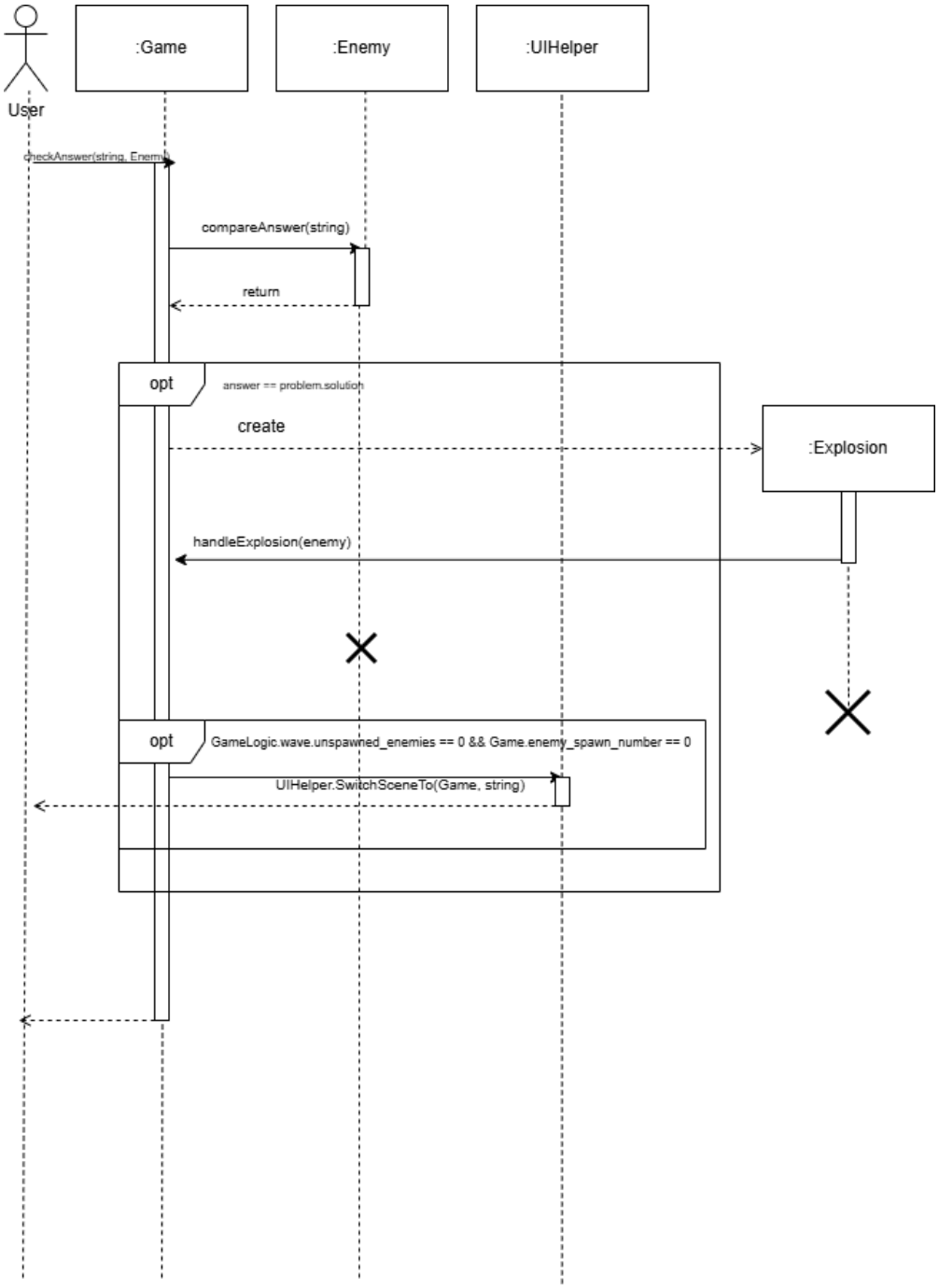

Figure 6: Inputting an Answer

Figure 6 depicts the user attempting to input a solution to a problem. The Game calls a function to check the inputted answer against the selected enemy. The selected enemy will then check to see if the answer is correct, and if so, the Game creates an explosion, which then handles it along with the enemy, destroying them both, along with any other enemies caught in the explosion, and then giving the player score and gold for all

enemies defeated. The Game checks if the wave is over and if so, enlists the UIHelper to switch the scene to the shop before returning control to the player. If the answer the player gave was wrong, the Game immediately returns control to the player.

4.4 State Diagram

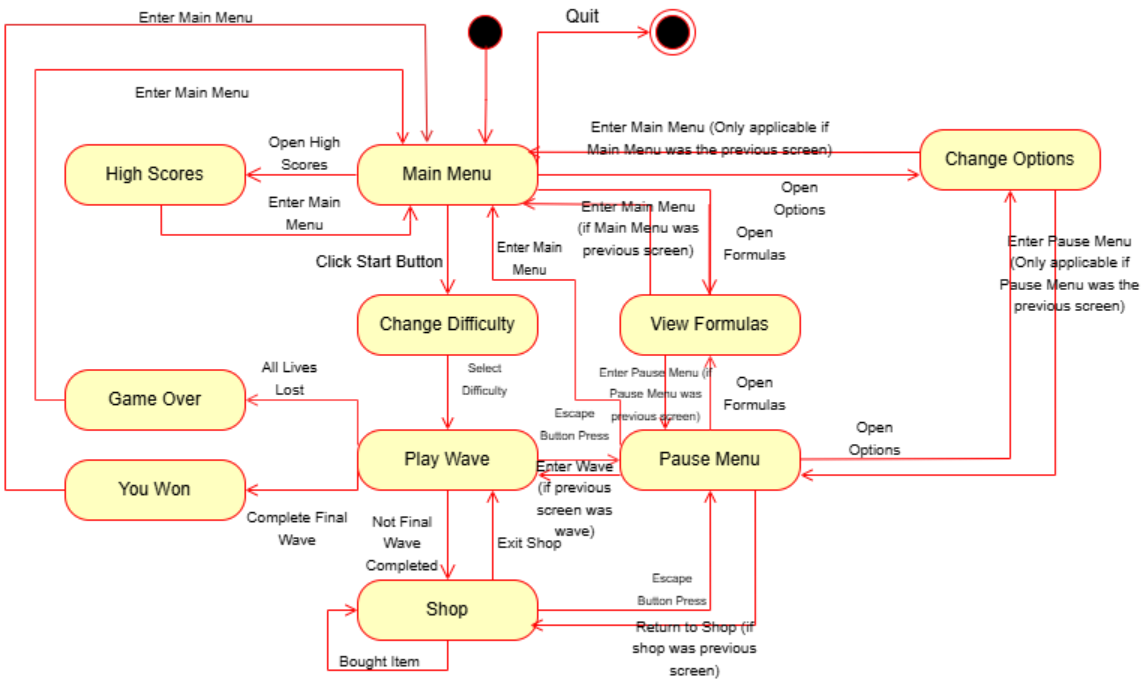

Figure 7: State Diagram

Figure 7 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 purpose of the prototype is to display the gameplay loop and overall functionality of the game Aerial Areas. The prototype contains problems designed to reinforce the skills the target audience has learned and demonstrates the various tools and resources the player has at their disposal to aid them, such as the tutorial, formulas, power ups and upgrades. The prototype aims to uphold the goals set out in Figure 2, and fulfill the requirements laid out in Section 3 to provide a fun and engaging way for children in the 4^{th} to 8^{th} grades to hone their skills with perimeters and areas.

5.1 How to Run Prototype

To be able to play the prototype, you must go to the website at "https://aerialareas.github.io/Aerial-Areas" 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 8: The Main Menu

In Figure 8, the user opens the game and enters their name as “Drake Maye”.

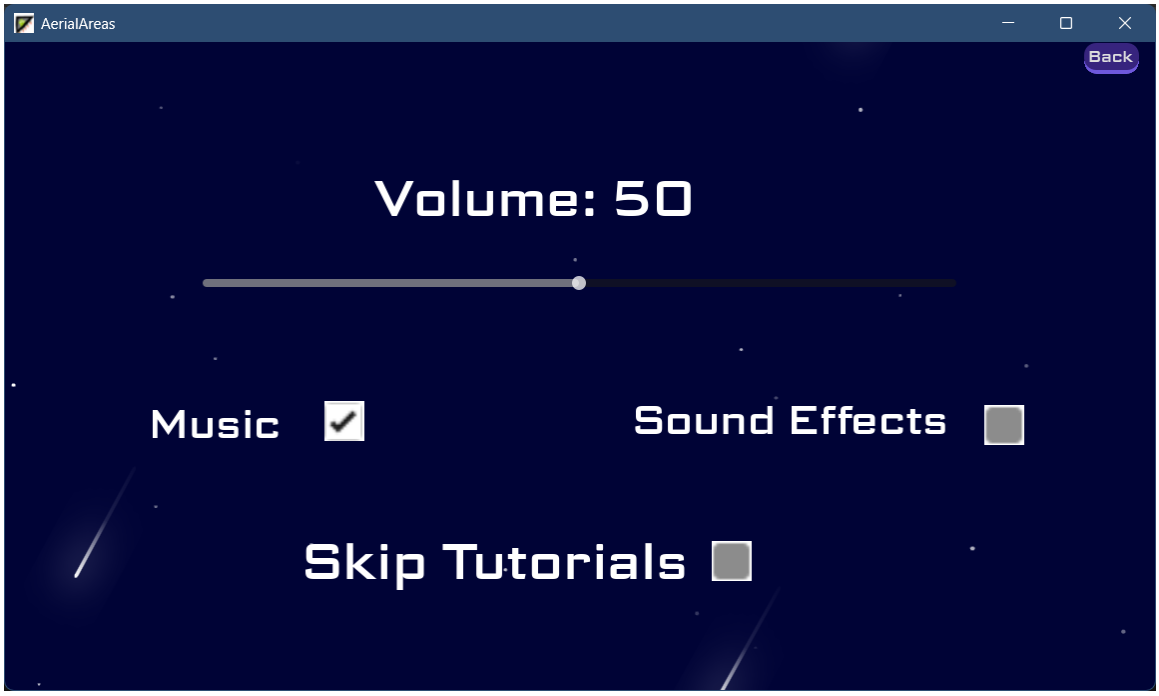

Figure 9: The Options Menu

In Figure 9, they then go to the options menu, modify the volume slider to 50, and toggle the sound effects off.

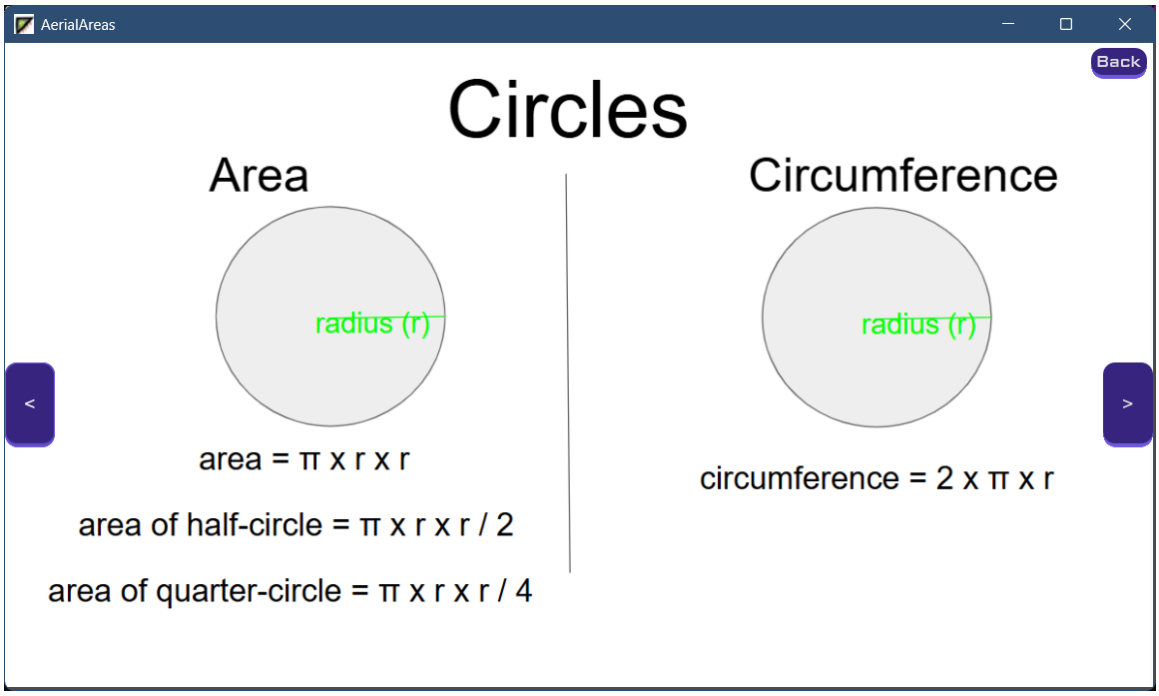

Figure 10: The Formulas Menu

In Figure 10, they return to the main menu, enter the Formulas menu, and navigate to the Circle formulas.

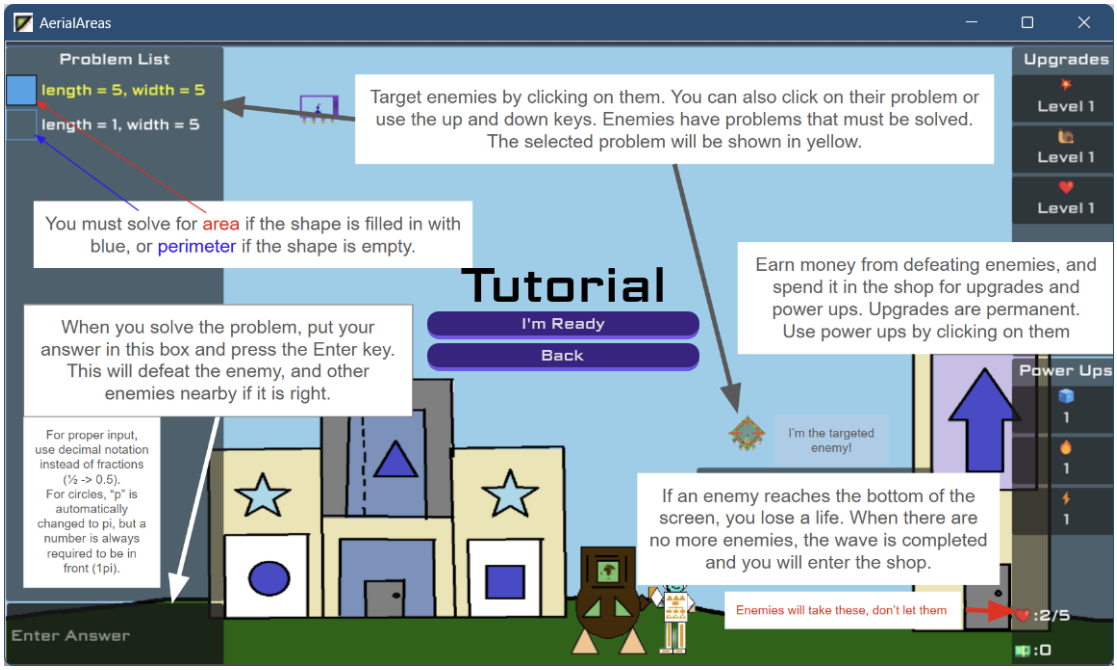

Figure 11: The Tutorial Screen

In Figure 11, after returning to the menu, the player clicks the Start button and selects the Easy difficulty, sending them to view the tutorial.

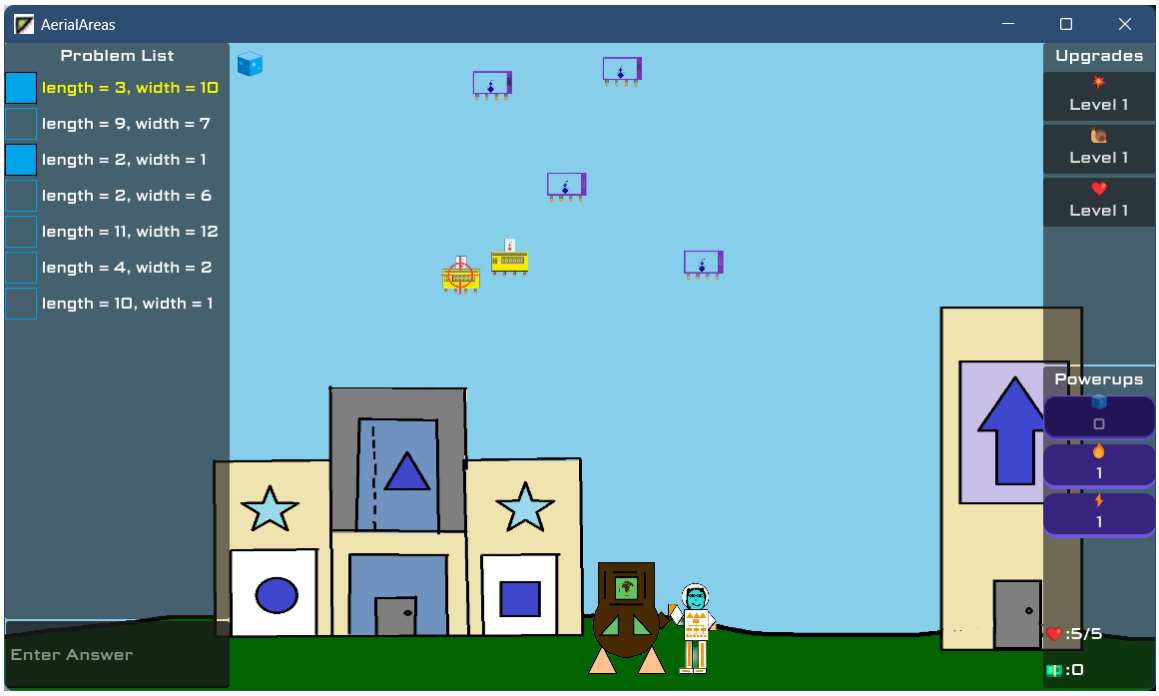

Figure 12: Normal Wave

In Figure 12, the player enters some answers and defeats some enemies, earning score and gold. They also use a freeze powerup, which is indicated by the top left of the screen. They select enemies by clicking on the sprites and by clicking on the enemies' corresponding problem.

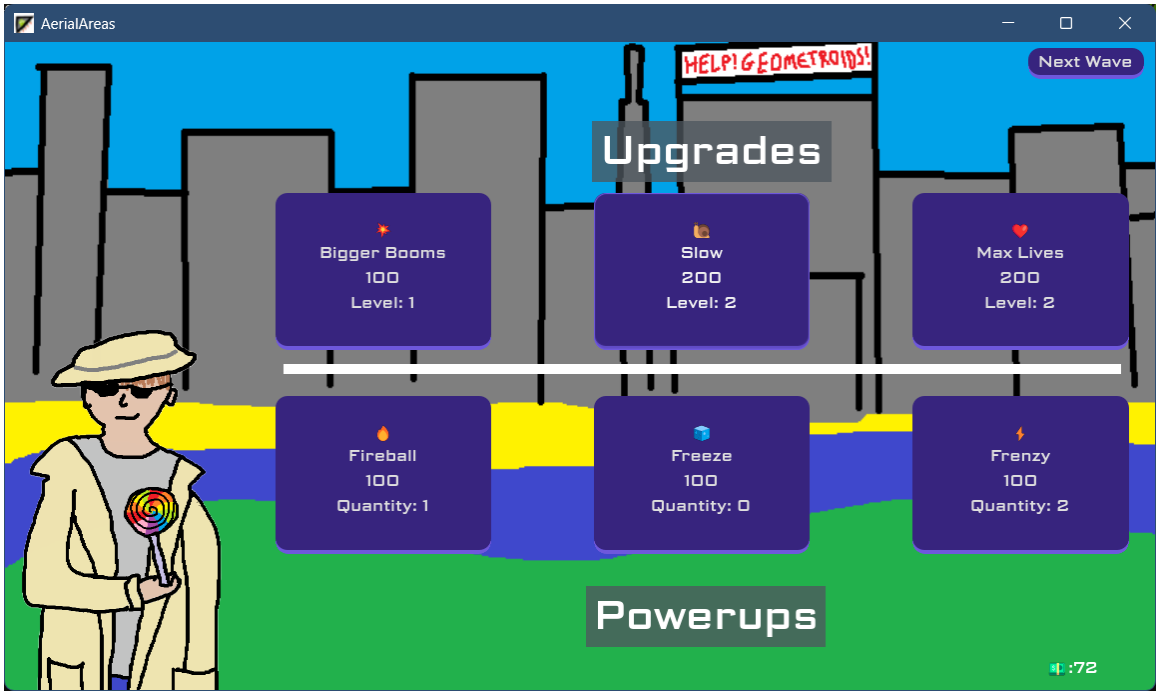

Figure 13: The Shop

In Figure 13, after completing the wave, the player enters the shop, buys a Max Lives upgrade, a Slow upgrade, and a Frenzy powerup.

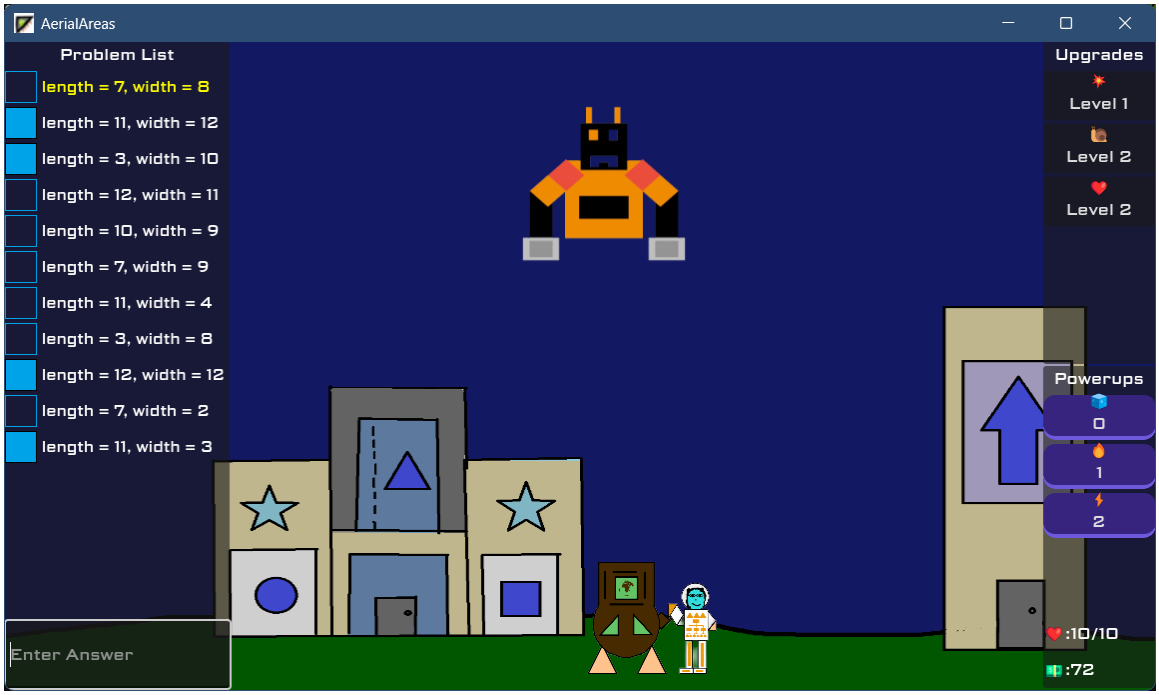

Figure 14: Boss Enemy

In Figure 14, the player has completed a second wave and moves on to the third, where they face a boss enemy, which has a list of problems to solve.

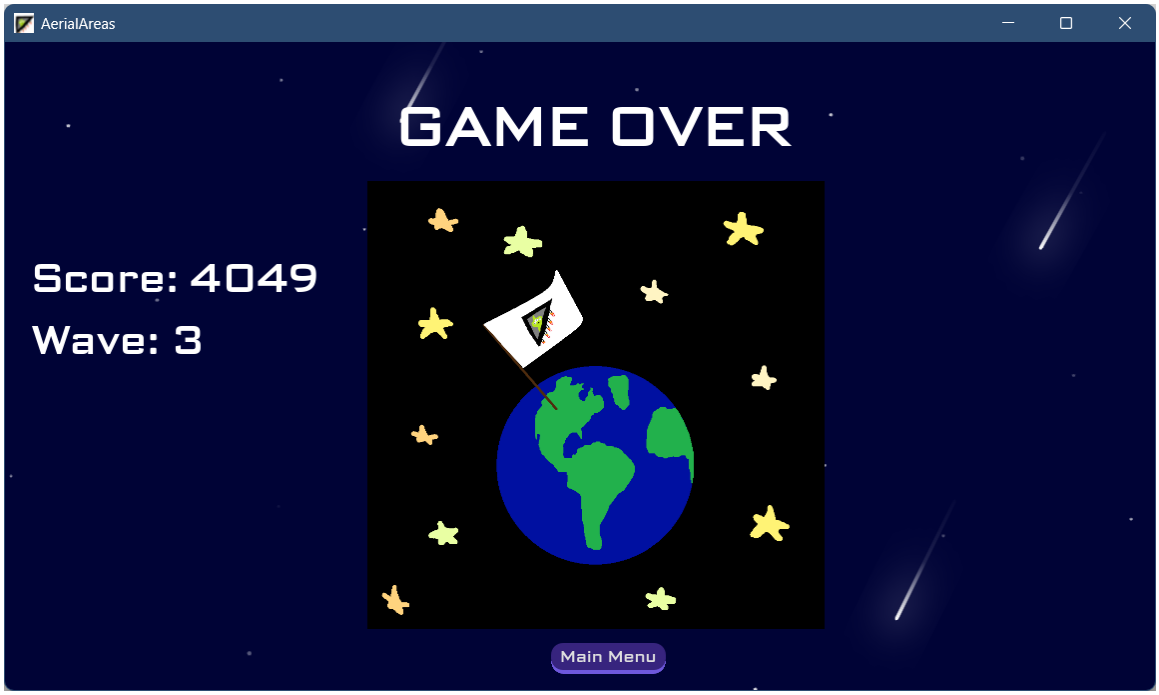

Figure 15: Game Over

In Figure 15, since the player did not solve the problems in time, they are met with the Game Over screen, indicating their score and the wave they made it to.

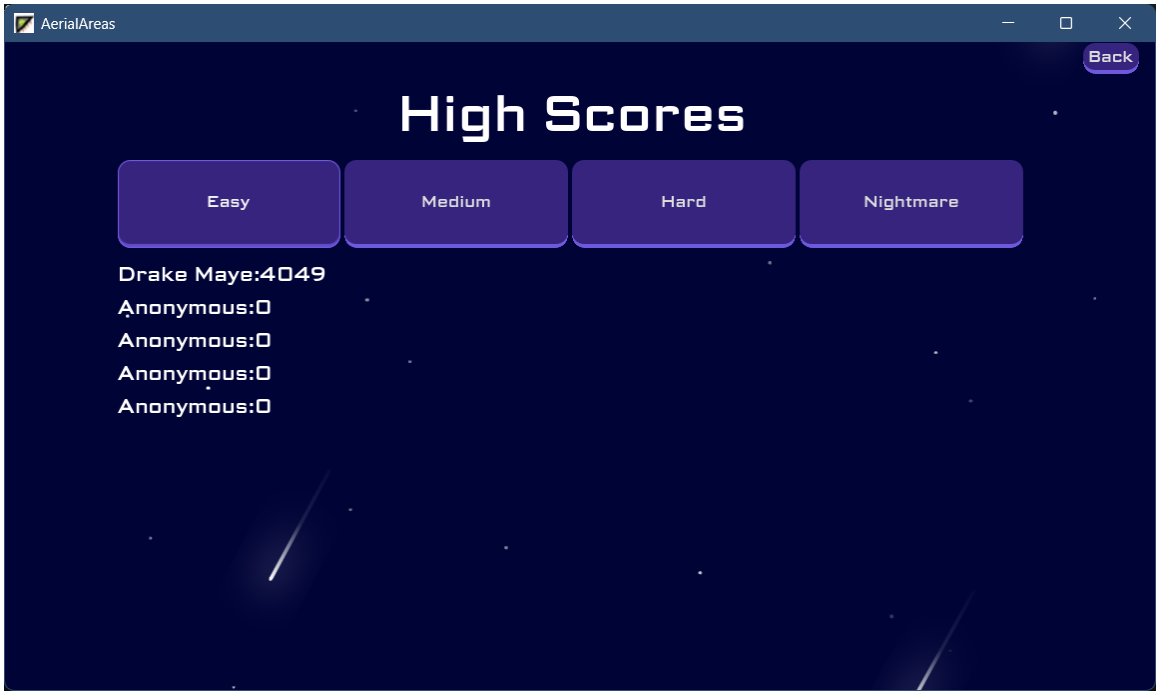

Figure 16: High Scores

In Figure 16, after returning to the main menu, they view the High Scores menu and check their score in the Easy difficulty tab.

6 References

[1] Apple, "Apple launches Freeform: a powerful new app designed for creative collaboration," Apple Newsroom, Dec. 13, 2022. [Online]. Available: https://www.apple.com/newsroom/2022/12/apple-launches-freeform-a-powerful-new-app-designed-for-creative-collaboration/

[2] "Convert PDF to PNG Online," PDF to PNG, https://pdf2png.com/ (accessed Dec. 6, 2025).

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

[4] "Free Background Remover," Picsart, https://picsart.com/background-remover/ (accessed Dec. 6, 2025).

[5] "Free Image Resizer," Picsart, https://picsart.com/resize-image/ (accessed Dec. 6, 2025).

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

[7] Kaleido, “Upload image,” remove.bg, https://www.remove.bg/upload (accessed Dec. 6, 2025).

[8] 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

[9] Microsoft. (2025). Microsoft Paint (Version 11.2504.17.0) [Software]. Microsoft [Online]. Available: www.microsoft.com

[10] O. Aly, B. Brookhart, T. Fitzpatrick, B. Porter, and N. Prentiss, Aerial Areas, https://aerialareas.github.io/Aerial-Areas (accessed Nov. 19, 2025).’

[11] “Song Maker,” Chrome Music Lab - Song Maker, https://musiclab.chromeexperiments.com/Song-Maker/ (accessed Dec. 6, 2025).

[12] “Starry Sky Generator – by the MagicPattern Design Toolbox,” MagicPattern, http://www.magicpattern.design/tools/starry-sky-generator (accessed Dec. 6, 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.