Software Requirements Specification (SRS)
Leximon

Team: Group 2

Authors: Charlie Norton, Platon Supranovich, Samuel Stanley, Christopher Nguyen, Evan Sykowski

Customer: 4th to 5th grade English Language Arts teachers

Instructor: Dr. James Daly

Contents

1 Introduction

This Software Requirements Specification (SRS) describes Leximon, a top-down turn-based battle game where players use their knowledge of synonyms and antonyms to defeat computer-controlled enemies [12]. This document is divided into different sections and subsections which cover different information about the project. These sections introduce the project and the structure of this SRS, describe the purpose of the Leximon game, lay out its full requirements, provide diagrams representing key parts of the Leximon game, and give instructions on how to run the prototype.

1.1 Purpose

The purpose of this SRS is to provide a complete definition of the functional and non-functional requirements for Leximon. This is so that the development team has a clear blueprint for implementation and testing. This document serves as the primary reference for developers, testers, and stakeholders throughout the project lifecycle. For the customer, which are 4th and 5th grade English Language Arts teachers, the SRS communicates how Leximon will reinforce synonym and antonym recognition through engaging, self-contained battles. Battles that require no teacher grading or preparation, deliver immediate feedback to students. Battles that also motivate repeated practice in a game format that students already enjoy. By clearly delineating what will and will not be delivered in the prototype, this SRS also establishes realistic expectations. It also establishes facilitates future expansion of the product.

1.2 Scope

The result of this project will be a prototype of Leximon built in the Godot game engine containing a simple world with just enough functionality to engage with a full-featured turn-based battle system. Designing, testing, and improving the battle system is the main technical goal of this prototype.

Even in its prototype state Leximon will be playable by its 4th and 5th grade intended audience. Students will be able to engage in battles that remind them of what synonyms and antonyms are. Students will be presented with a list of words during battle and the effectiveness of their attack or defense will be determined by if they correctly identify a synonym or antonym. The objective of these battles will be to help students practice identifying synonyms and antonyms of words and to teach them new synonyms and antonyms.

The delivered product is a playable prototype that contains only one battle arena. It also contains a limited set of words/enemies. It will not include overworld exploration nor multiple levels. It will also not include persistent progress saving, account systems, multiplayer, mobile support, or teacher analytics/dashboard features.

Leximon addresses the common 4th-5th grade struggle of vocabulary retention by turning synonym/antonym practice into an engaging Pokémon-style battle, providing immediate feedback and repeated exposure in a format students already love.

1.3 Definitions, acronyms, and abbreviations

- The Game: The product described by this SRS.
- Game State: The data which determines the state of each component of The Game.
- The Computer: A device on which The Game runs.
- Player: The person who interacts with the game by providing input via The Computer.
- Type: A single-word label for a category of words which are synonyms.
- Action: A single word belonging to one or more Types.
- Dictionary: A collection of one or more Actions.
- HP: Health Points.
- Armor: A multiplier which affects how HP is changed.
- Character: An object in The Game which has a Type, a Dictionary, a maximum HP value, a current HP value, and an Armor value.
- Inventory: The collection of values owned by a Character.
- Player Character: The Character controlled by the Player.
- Enemy: A Character controlled by The Game which is hostile to The Player Character.
- Attack: An attempt by a Character to reduce another Character's HP.
- Defense: An attempt by a Character to increase their own Armor value.
- Turn: An opportunity for the Player Character or an Enemy to Attack or Defend while interacting.
- Round: A single loop of Turns, one for the Player Character and one for the Enemy.
- Battle: A sequence of Rounds.

1.4 Organization

The rest of this SRS will be structured as follows:

- **Section 2:** A description of The Game which includes its context, primary functions, user characteristics, constraints, dependencies, and future expansions.
- **Section 3:** An enumerated list of the complete requirements for The Game.
- **Section 4:** Use Case, Class, Sequence and State diagrams which describe The Game's behavior and requirements in visual form.
- **Section 5:** A description of the prototype, how to run it, and examples of using it.
- **Section 6:** A list of sources referenced or used in this SRS.
- **Section 7:** Point of contact.

2 Overall Description

This section will describe an overview of The Game. First the background, overall educational intention and the requirements to play the game will be detailed. Then the main functions and high-level educational goals will be described. After that the characteristics of the intended users of The Game are described. Then the constraints on the development of The Game are defined and then the assumptions and dependencies. Finally the requirements beyond the scope of the project are listed.

2.1 Project Perspective

One of the key linguistic concepts students learn near the end of elementary school is synonyms and antonyms. These constructs can greatly enhance students' ability to identify and define new vocabulary words. For many people it is easier to understand the meaning of a word if presented with a synonym or antonym than when presented with its definition. Synonyms are crucial for creating variation in writing and can often make the difference between boring and interesting literature.

Leximon is a game which presents students with an opportunity to practice identifying synonyms and antonyms in a fun and engaging way. The Game is intended to be introduced to students by teachers as a practice and learning tool which may be played in class or on students' own time. It is a standalone product which enhances existing education about synonyms and antonyms.

The Game requires a computer with a keyboard, mouse and monitor to be played. Currently Godot does not support compiling for web, so The Game must be downloaded as a program and run on The Computer [11]. The Computer must run the Windows operating system.

2.2 Project Functions

The primary function of The Game is to engage the Player in turn-based Battles where they practice their knowledge of synonyms and antonyms and learn new words by identifying synonyms and antonyms. Players will navigate a two-dimensional top-down world and interact with enemies they see to start Battles. In a Battle Players will be presented with a list of words and will be shown the Type of each Character. Players will select a word from the list and choose either to Attack or Defend. If they choose to Attack then the word they choose must be an antonym of the Enemy's Type to be effective. If they choose to Defend then the word they choose must be a Synonym of their own Player Character's Type to be effective. In this way, Players will test their knowledge of synonyms and antonyms.

While not in Battle Players may view their Dictionary containing all their words and the definitions of their words. Since a word may have multiple meanings for different contexts, multiple definitions corresponding with each relevant Type will be displayed. Players may also see the definitions of the visible words during a Battle. This teaches Players new words so that they may learn their synonyms and antonyms during Battle which, in turn, reinforces the definitions they learned.

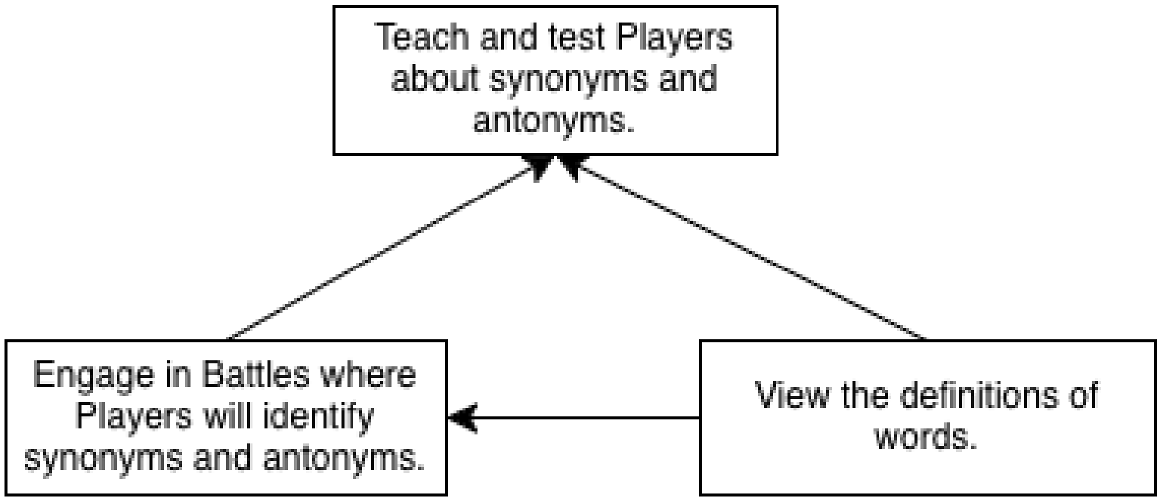

Teach and test Players about synonyms and antonyms.
Engage in Battles where Players will identify synonyms and antonyms.
View the definitions of words.

Figure 1: High Level Goals

2.3 User Characteristics

The intended users of The Game are students in grades 4 and 5, the grades where students learn about synonyms and antonyms [6]. Alternatively students in different grades who are learning about synonyms and antonyms are also intended users. The Game is intended for students who are already at least somewhat familiar with what synonyms and antonyms are but still need additional practice identifying them or need to practice learning new words using synonyms and antonyms.

2.4 Constraints

Technical Constraints: The development of The Game is constrained by the capabilities of the Godot game engine with C#. While the game engine is very permissive in the features which may be implemented, it only has the capability to compile C# projects for Windows, Linux and macOS, not for the web [11]. This means that the constraints of the Players' computers must be taken into account.

Developmental Constraints: The development of The Game is also constrained by its developers and its timescale. Since this project is the first time several of its developers have used Godot or C#, learning both these tools presents a constraint on the complexity and quantity of the work which may be done on The Game. Additionally, all the developers are students and have a significant amount of tasks they must complete other than this project. Finally, this project is constrained by the amount of time given to complete it.

Legal Constraints: All assets and code used in The Game must adhere to U.S. copyright law and must provide attribution when applicable.

2.5 Assumptions and Dependencies

Assumptions: The Game assumes that users have access to computers with a keyboard, monitor and mouse running the Windows operating system. It also assumes that users' computers are powerful enough to render simple 2D graphics. Lastly, The Game assumes that users are at least somewhat familiar with what synonyms and antonyms are.

Dependencies: The Game depends on the user's computer supporting the Vulkan, Direct3D 12 or Metal rendering APIs, which are the default rendering drivers used by Godot [9].

2.6 Apportioning of Requirements

The current prototype focuses mainly on the Battle system, testing if the system is technically feasible, educational, and fun. In the future, the world outside of Battles may be expanded to include a story and main quest, Characters other than Enemies, different ways to obtain Actions and different worlds which may be visited. Battles may be expanded to include more types of Actions than just Attack and Defend, variations on the strength and quantity of Enemies, and an experience system to allow Players to level up. The ability to play and store multiple separate Game States may also be added.

3 Specific Requirements

1. Upon starting the game the player will be shown a Main Menu scene with the name of The Game and several options.

    (a) An option labeled "Start" will retrieve the saved Game State and apply it to The Game. If no saved Game State is available, a new Game State will be initialized and applied to The Game.

    (b) An option labeled "About" will open The Game's website in an external web browser.

    (c) An option labeled "Exit" will close The Game.

2. The Game will have a Hub World containing the Player Character and five Enemies.

    (a) The Hub World will be the scene shown to the Player by a new Game State.

    (b) The Player Character will be able to move freely around the Hub World.

    (c) Five Enemies will be spawned at random locations in the Hub World when the Player Character enters the Hub World for the first time.

    (d) Each time the Player Character enters the Hub World enemies will be spawned at random locations until there are five enemies in the Hub World.

3. While in the Hub World the Player will be able to open an Inventory Menu.

    (a) Actions from the Player Character's Dictionary will be displayed.

        i. Selecting an Action will display the Types it belongs to and the definitions of the Action corresponding to each Type.

    (b) The Player Character's Type will be displayed.

    (c) The Player Character's current and maximum HP will be displayed.

4. Characters will start with 100 current and maximum HP.

5. The Player Character interacting with an Enemy in the Hub World will start a Battle.

    (a) Starting a Battle will move the Player Character and the Enemy to the Battle Scene.

        i. The Player Character and the Enemy will be displayed facing each other.

        ii. The Type, HP, and Armor value of each Character will be displayed.

        iii. A list of five Actions randomly selected from the Player Character's Dictionary at the start of the Battle will be displayed.

(b) Each time it is the Player Character's turn the Player will select a displayed Action.

   i. Upon selecting an Action the Player will be prompted to choose whether to perform an Attack or Defense using the Action.

   ii. Attacks by the Player Character will deal a normal damage of 10 HP.

   iii. Defenses by the Player Character will normally increase their Armor value by 7% each time.

(c) The strength of the Player Character's Attacks will be determined by the Relation Engine.

   i. An Attack's strength will be unchanged if the Action belongs to a Type which is an antonym of the Enemy's Type.

   ii. Otherwise, an Attack's strength will be multiplied by 25% if the Action belongs to a Type which is a synonym of the Enemy's Type.

   iii. An Attack's strength will be multiplied by 0% if none of the Types the Action belongs to are antonyms or synonyms of the Enemy's type.

(d) The strength of the Player Character's Defenses will be determined by the Relation Engine.

   i. A Defense's strength will be unchanged if the Action belongs to a Type which is a synonym of the Player Character's Type.

   ii. Otherwise, a Defense's strength will be multiplied 25% if the Action belongs to a Type which is an antonym of the Player Character's Type.

   iii. A Defense's strength will be multiplied 0% if none of the Types the Action belongs to are antonyms or synonyms of the Player Character's type.

(e) Each time it is the Enemy's turn The Game will select Attack or Defense.

   i. Attacks by the Enemy will deal 7 HP of damage.

   ii. Defenses by the Enemy will increase their Armor value by 5% each time.

(f) All Attack values will be divided by the Armor value of the Character they affect.

(g) The Battle will end when one of the Characters runs out of HP.

   i. The Player Character running out of HP will be considered a Player Character Loss.

      A. The Enemy will be returned to their position in the Hub World before the Battle started.

      B. The Enemy's current HP will be reset to their maximum HP and their Armor value will be reset to 100%.

   ii. The Enemy running out of HP will be considered a Player Character Win.

      A. The Enemy will be removed from the Hub World.

B. The Player Character will receive one to three new Actions.

iii. The Player Character will be returned to their position in the Hub World before the Battle started.

iv. The Player Character's current HP will be reset to their maximum HP and their Armor value will be reset to 100%.

v. The Hub World will be shown to the Player.

vi. The current Game State will be saved.

6. While in the Hub World the Player will be able to save the current Game State.

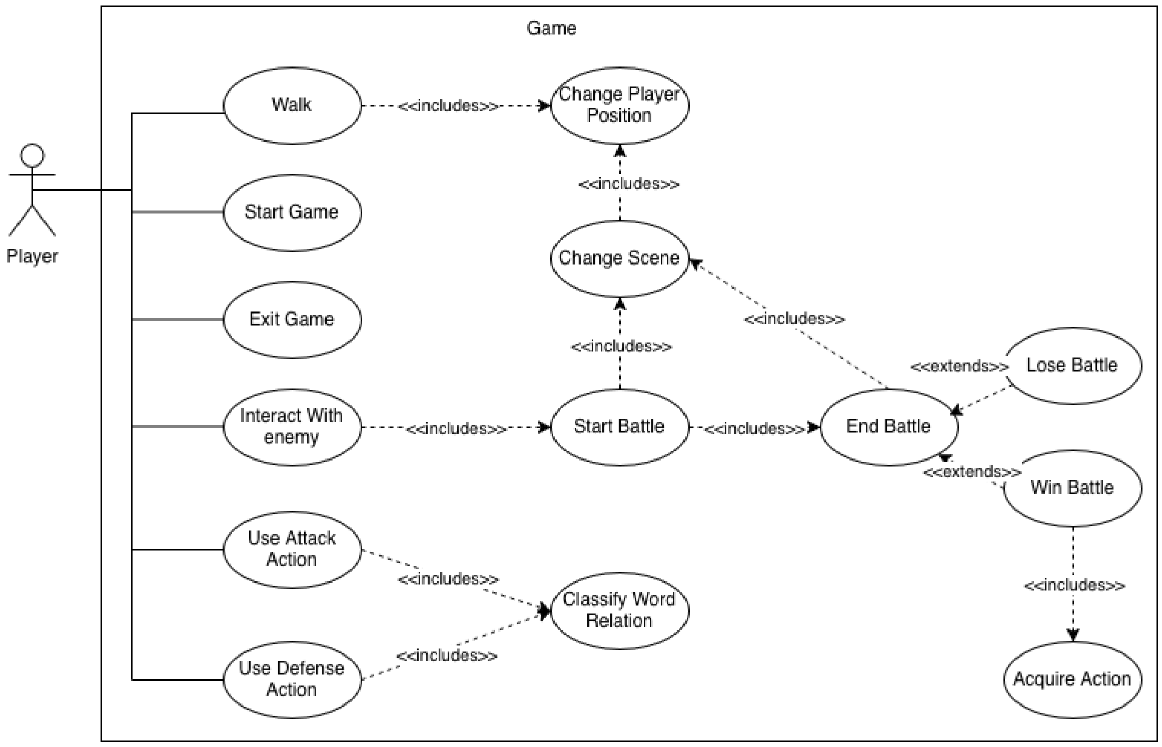
7. While in the Hub World the Player will be able to exit to the Main Menu.

   (a) Exiting to the Main Menu will save the current Game State.

4 Modeling Requirements

4.1 Use Case Diagrams

The following diagram shows the use cases for the Player, demonstrating how they interact with The Game and how different parts of The Game interact internally to perform concrete use cases.



Use Case Name	Walk
Actors	Player
Description	Move the Player's position in the Hub World. The Player may walk in any direction using the keyboard but may not walk past the edge of the Hub World.
Type	Primary
Includes	Change Player Position
Extends	N/A
Cross-refs	Requirement 2.a
Use Cases	N/A

Use Case Name	Change Player Position
Actors	System
Description	Change the coordinates of the player.
Type	Secondary
Includes	N/A
Extends	N/A
Cross-refs	Requirement 2.b & 5
Use Cases	Walk, Change Scene

Use Case Name	Start Game
Actors	Player
Description	The button in the Main Menu to retrieve the saved Game State and apply it to The Game. Initializes a new Game State if none exists.
Type	Primary
Includes	N/A
Extends	N/A
Cross-refs	Requirement 1.a
Use Cases	N/A

Use Case Name	Change Scene
Actors	System
Description	The game changes to a new environment.
Type	Secondary
Includes	Change Player Position
Extends	N/A
Cross-refs	Requirement 5.a, 5.g.v

Use Case Name	Exit Game
Actors	Player
Description	The button in the Main Menu to stop running The Game.
Type	Primary
Includes	N/A
Extends	N/A
Cross-refs	Requirement 1.c
Use Cases	N/A

Use Case Name	Start Battle
Actors	System
Description	Changes the scene to the battle scene and places the interacted with enemy in the opponent's slot.
Type	Secondary
Includes	Change Scene, End Battle
Extends	N/A
Use Cases	Interact With enemy

Use Case Name	Interact with Enemy
Actors	Player
Description	While in close proximity to an Enemy in the Hub World, a button may be pressed to interact with them, which starts a Battle.
Type	Primary
Includes	Start Battle
Extends	N/A
Cross-refs	Requirement 5

Use Case Name	Classify Word Relation
Actors	System
Description	Determines if the two words are synonyms, antonyms, or neither.
Type	Secondary
Includes	N/A
Extends	N/A
Cross-refs	Requirement 5.c
Use Cases	Use Attack Action, Use Defense Action

Use Case Name	Use Attack Action
Actors	Player
Description	While in a Battle the Player may use an Action to Attack. The Game will classify the word relation and apply the appropriate damage to the Enemy.
Type	Primary
Includes	Classify Word Relation
Extends	N/A
Cross-refs	Requirement 5.b.i
Use Cases	N/A

Use Case Name	Use Defense Action
Actors	Player
Description	While in a Battle the Player may use an Action to Defend. The Game will classify the word relation and apply the appropriate Armor value to the Player Character.
Type	Primary
Includes	Classify Word Relation
Extends	N/A
Cross-refs	Requirement 5.b.i
Use Cases	N/A

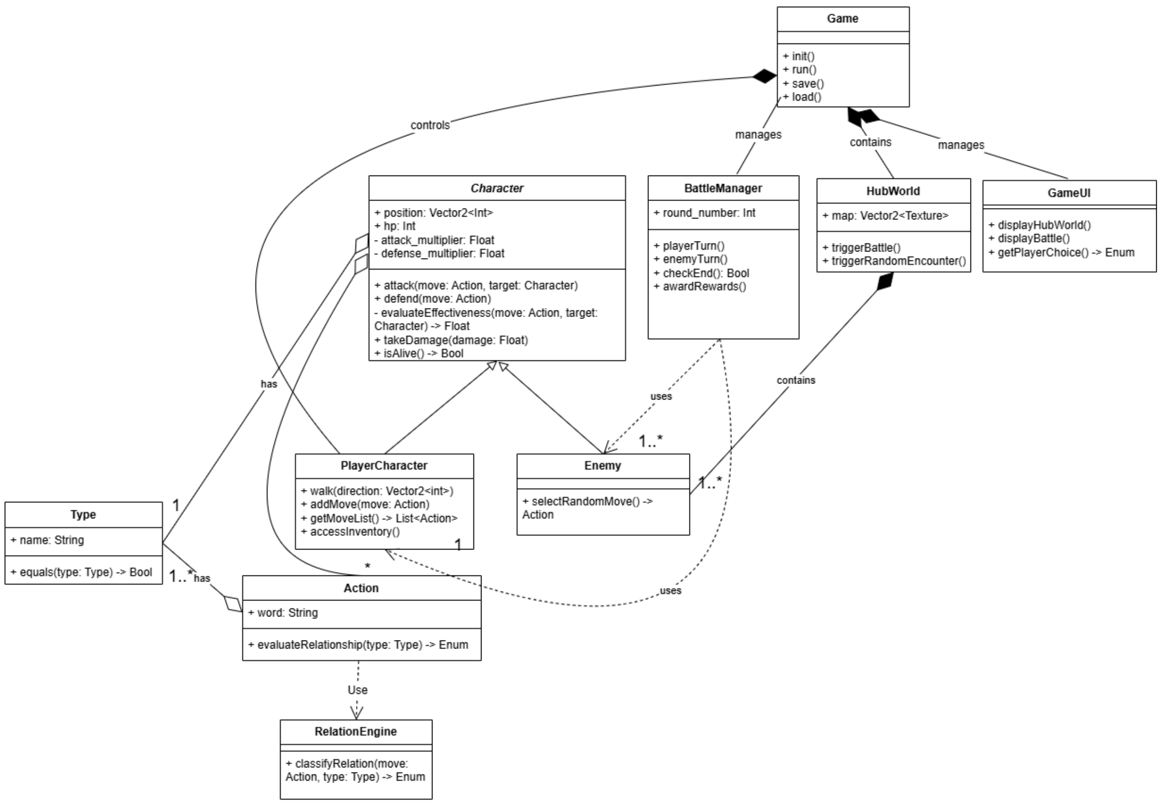
Use Case Name	End Battle
Actors	System
Description	End the current battle and return the player to the hub world. Restore the player's HP
Type	Secondary
Includes	Change Scene
Extends	Lose Battle, Win Battle
Cross-refs	Requirement 5.g
Use Cases	Start Battle

Use Case Name	Lose Battle
Actors	System
Description	Return the player and enemy to the hub world. Restore both of their HP
Type	Secondary
Includes	N/A
Extends	N/A
Cross-refs	Requirement 5.g.i
Use Cases	N/A

Use Case Name	Win Battle
Actors	System
Description	Return the player to the hub world. Restore their health. Add new Actions to the player's list of known moves.
Type	Secondary
Includes	Acquire Move
Extends	N/A
Cross-refs	Requirement 5.g.ii
Use Cases	N/A

Use Case Name	Acquire Action
Actors	System
Description	Add a new word to the player's known words that they can use in a battle
Type	Secondary
Includes	N/A
Extends	N/A
Cross-refs	Requirement 5.g.ii.B
Use Cases	Win Battle

4.2 Class Diagram


Game
+ init()
+ run()
+ save()
+ load()
controls
manages
contains
manages
Character
+ position: Vector2<Int>
+ hp: Int
- attack_multiplier: Float
- defense_multiplier: Float
+ attack(move: Action, target: Character)
+ defend(move: Action)
- evaluateEffectiveness(move: Action, target: Character) -> Float
+ takeDamage(damage: Float)
+ isAlive() -> Bool
BattleManager
+ round_number: Int
+ playerTurn()
+ enemyTurn()
+ checkEnd(): Bool
+ awardRewards()
HubWorld
+ map: Vector2<Texture>
+ triggerBattle()
+ triggerRandomEncounter()
GameUI
+ displayHubWorld()
+ displayBattle()
+ getPlayerChoice() -> Enum
uses
contains
has
1..*
1..*
PlayerCharacter
+ walk(direction: Vector2<int>)
+ addMove(move: Action)
+ getMoveList() -> List<Action>
+ accessInventory()
Enemy
+ selectRandomMove() -> Action
Type
+ name: String
+ equals(type: Type) -> Bool
1
1
*
1..*
has
uses
Action
+ word: String
+ evaluateRelationship(type: Type) -> Enum
Use
RelationEngine
+ classifyRelation(move: Action, type: Type) -> Enum

BattleManager

Element Name	Description
BattleManager	Manages the confrontation between the player and the enemy

Attributes		
	round_number: Int	Keeps track of the number of rounds in a battle.
Operations		
	playerTurn()	Provides the player to play their Action on their turn.
	enemyTurn()	Provides the player's opponent to play their Action on their turn.
	checkEnd(): Bool	Used to see whenever one of the Characters runs out of HP. If one of the Characters runs out of HP, a new sequence is progressed.
	awardRewards()	If the player wins the game, the player will have the health and armor filled up to the maximum and will receive from 1 to 3 new Actions.
Relationship	BattleManager has an association with the Game class. BattleManager uses PlayerCharacter and Enemy.	

Character

Element Name	Description
Character	Represents the entities that can attack, defend, and be defeated

Attributes	position:Vector2<Int>	Represents the X, Y position of the player on the map.
	hp: Int	Represents the health of the characters.
	attack_multiplier: Float	Multiplier for an attack boost, higher the multiplier means more damage done on opponent.
	defense_multiplier: Float	Multiplier for a defense boost, higher the multiplier means less damage done on character being attacked.
Operations		
	attack(move: Action, target: Character)	Character chooses an Action and targets the selected opponent in hopes of reducing the opponent's HP.
	defend(move: Action)	Character chooses an Action in hopes of limiting the damage dealt with by the opponent.
	EvaluateEffectiveness(move: Action, target: Character) ->Float	Used to find how effective an Action is towards an opponent based on the word selection
	takeDamage(damage: Float)	How much HP is reduced
	isAlive()-> Bool	Used to see if a character is alive or not (HP is greater than 0.
Relationship	Character class can contain PlayerCharacter, Enemy, Action, and Type.	

PlayerCharacter

Element Name	Description
PlayerCharacter	Represents the character the user uses to play the game.

Operations		
	walk(direction: Vector2<int>)	Indicates the direction the player is moving (up, down, left, right).
	addMove(move: Action)	This occurs whenever the player acquires more Actions.
	getMoveList()->List<Action>	Lists all the Actions the player have
	accessInventory()	Allows the player to view the dictionary, the player's type, and the player's current and maximum HP.
Relationship	The PlayerCharacter class is a part of the Character class. PlayerCharacter controls the game and BattleManager uses PlayerCharacter.	

Action

Element Name	Description
Action	A single word belonging to one or more Types.

Attributes		
	word: String	What the word is called.
Operations		
	evaluateRelationship(type: Type) -> Enum	Will check to see how related the type of words are.
Relationship	Character can have Action. Action has Type. RelationEngine Uses Action.	

Enemy

Element Name	Description
HubWorld	The opponent the player is attempting to defeat.

Operations		
	selectRandomMove()	The enemy will randomly select a move (attack, defend, heal) to beat the player.
Relationship	Enemy contains HubWorld, is used by BattleManager, and is a part of the Character class.	

Game

Element Name	Description
Game	The basic structure of the game.

Operations		
	init()	Init creates a new game.
	run()	Run actually starts the game.
	save()	Saves the game so the next time a user runs it they can continue with the progress they already accumulated.
	load()	Loads the game so players can access previous progress.
Relationship		The game controls PlayerCharacter, manages BattleManager, contains HubWorld, and manages GameUI.

GameUI

Element Name	Description
GameUI	The user interface the player can interact with.

Operations		
	displayHubWorld()	Will take the player to the HubWorld.
	displayBattle()	Will show the battle interaction between the player and the Enemy.
	getPlayerChoice()->Enum	Will get the result of player choice (attack, defend, heal).
Relationship	GameUI is managed by the game class	

HubWorld

Element Name	Description
HubWorld	The place where the player starts the game and moves around before battling an enemy.

Attributes		
	Map: Vector2<Texture>	A map is represented with a texture of the different kinds of tile in the map.
Operations		
	triggerBattle()	A battle will start with the player and the enemy.
	triggerRandomEncounter()	The enemies will randomly be spawned across the map.
Relationship	The Game class contains HubWorld. HubWorld contains Enemy.	

RelationEngine

Element Name	Description
RelationEngine	Determines what category the Action belongs to and how it’s related with other Actions.

Operations		
	+ classifyRelation(move: Action, type: Type) -> Enum	Determines the relationship between the word and its category.
Relationship	It is used by the Action class.	

Type

Element Name	Description
Type	A single-word label for a category of words which are synonyms.

Attributes		
	name: String	The category of the word.
Operations		
	equals(type: Type) → Bool	Returns true if the type equals the type provided in the parameter. Return false otherwise.
Relationship	The Game class contains HubWorld. HubWorld contains Enemy.	

4.3 Sequence Diagrams

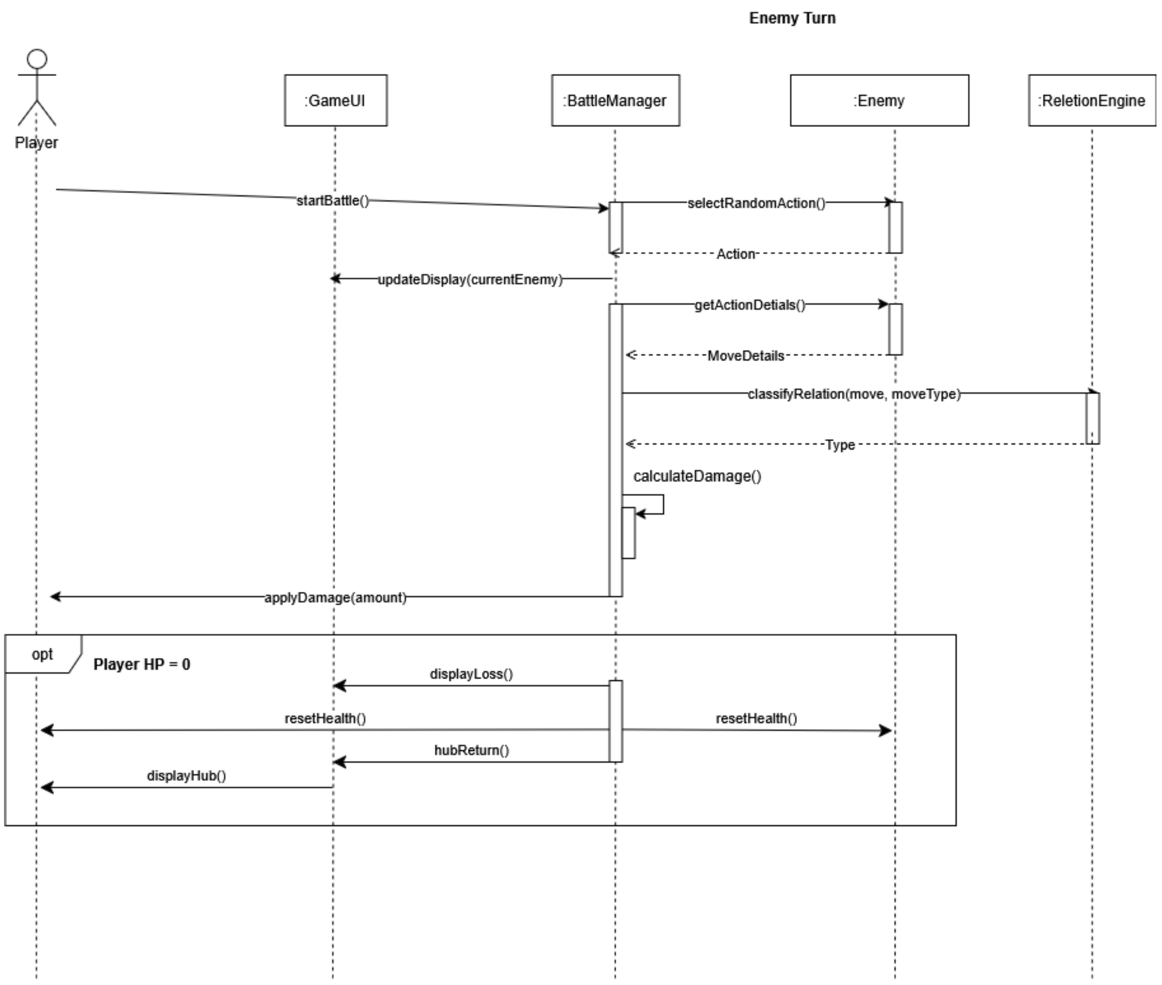


Figure 2: This sequence diagram shows the process of an enemy's turn during a battle. First the player enters a battle by interacting with an enemy. Then a random action for the enemy is selected. This move is compared against the player to determine the damage to apply. This damage is then apply to the player. IF the player's HP reaches 0, the game ends with the player showing a loss and both combatants being returned to the hub world with their HP reset.

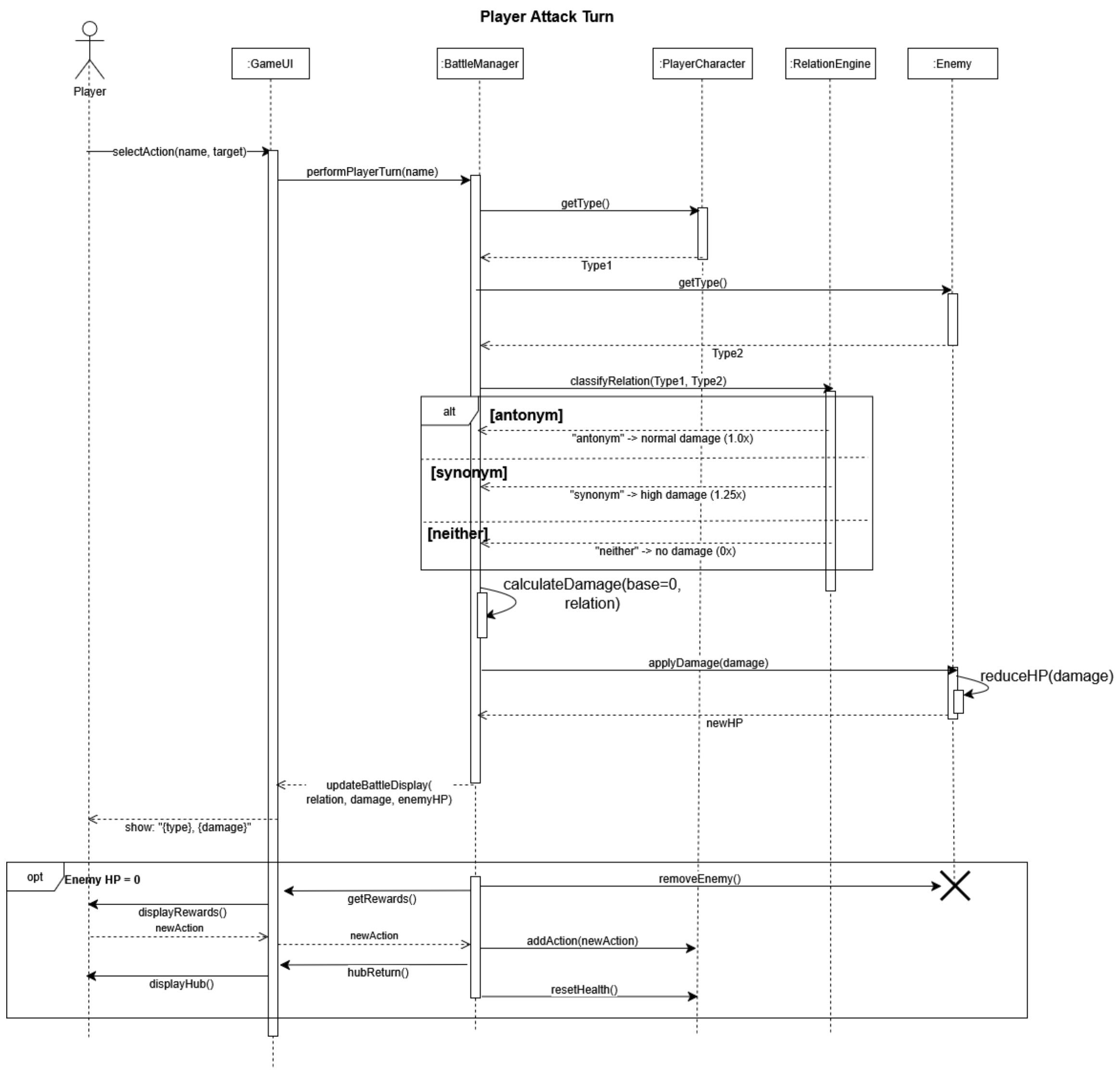
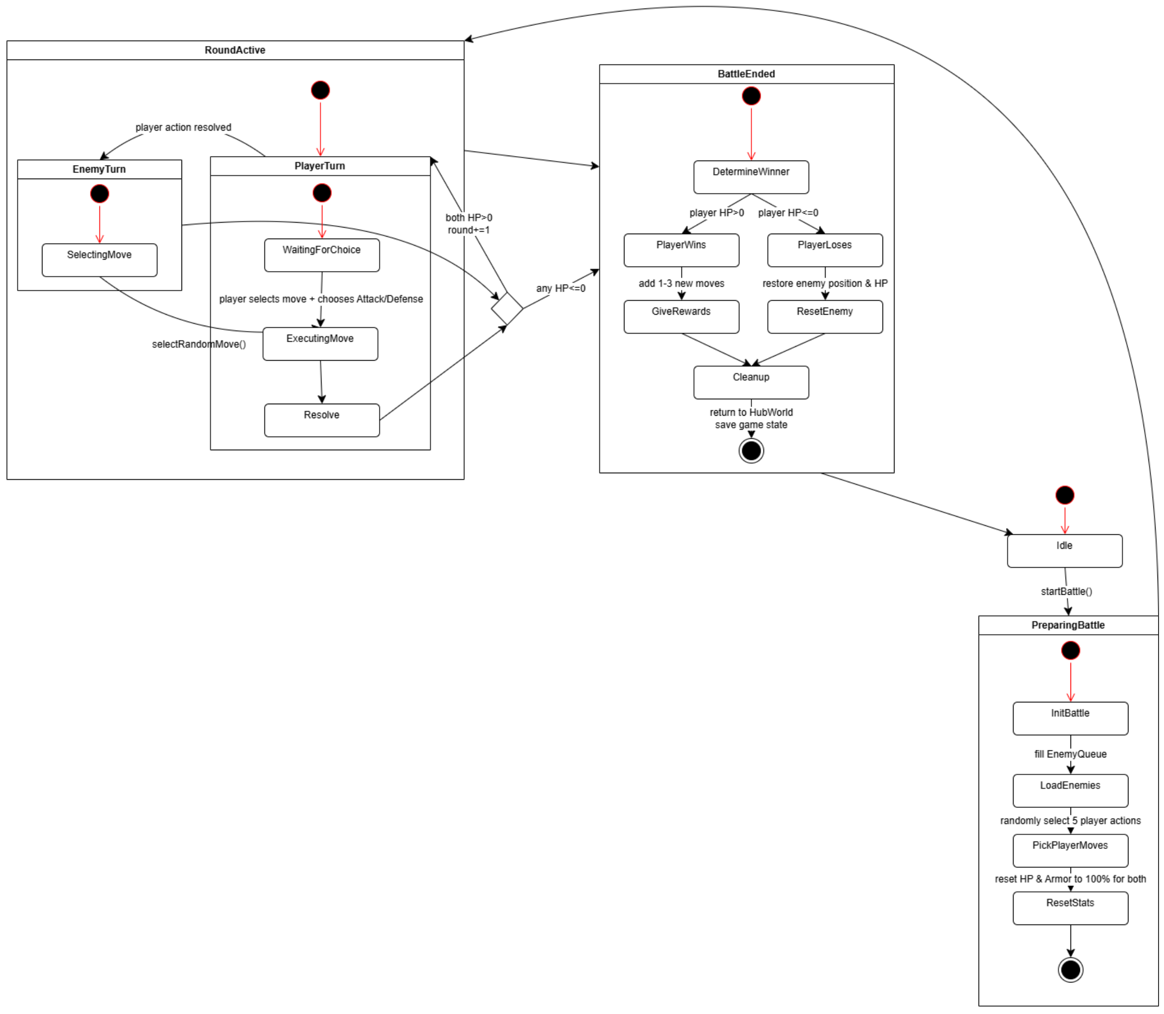
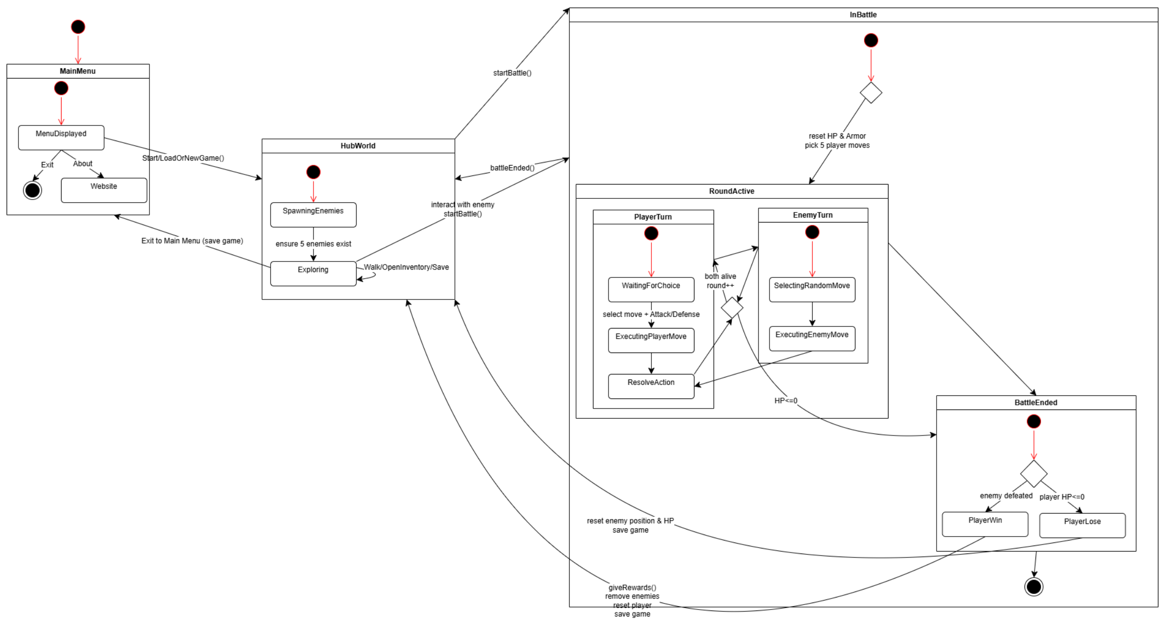


Figure 3: This sequence shows the the process the player goes through on their turn in a battle. First they select what action they want to use. Then the system determines the types of the player's move and the enemy. These are compared and a damage multiplier is derived from their relationship. Then the ddamage is applied to the enemy. After the damage is applied the UI is updated. Lastly, if the enemy's HP dropped to 0, the battle ends, and the player is granted new moves and returned to the hub world.

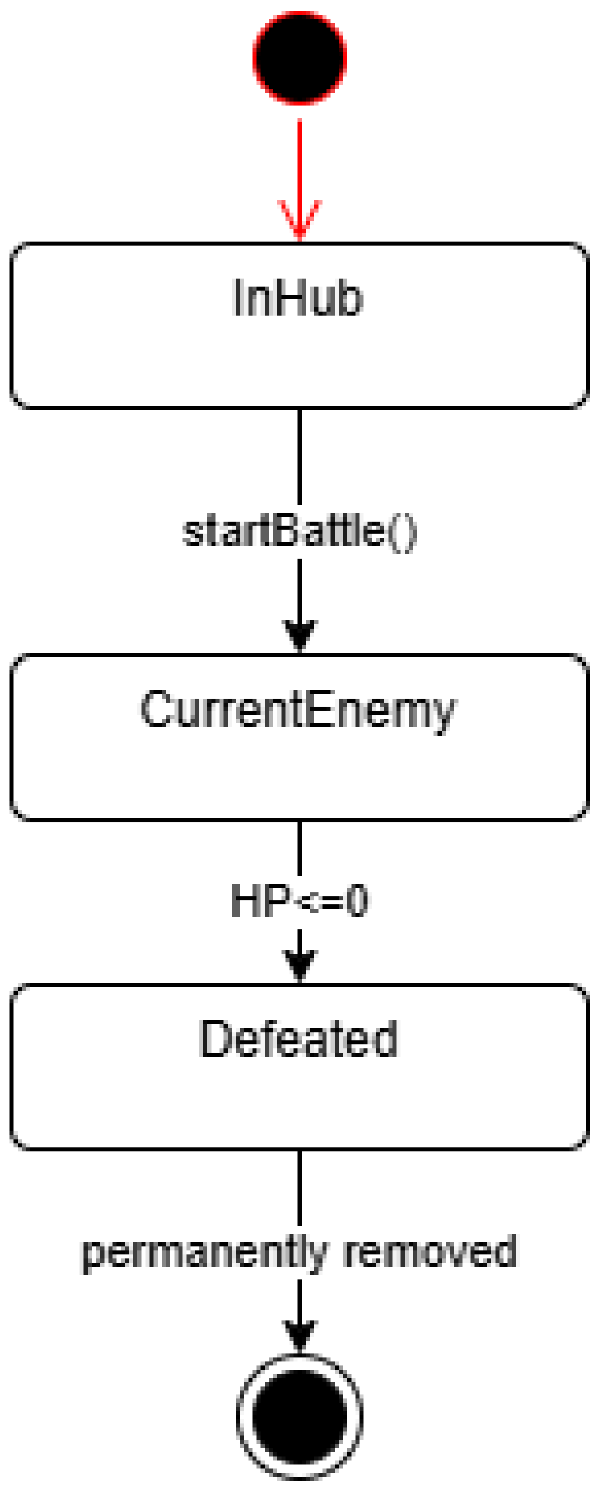
4.4 State Diagrams

BattleManager State Diagram





Enemy & EnemyQueue State Diagram



5 Prototype

The following sources were used in the production of the UI prototype [10, 14, 5, 8, 4, 1, 2, 3, 13, 7]

5.1 How to Run Prototype

To run the prototype you must be using a Windows computer. Download the most recent release from the project GitHub at `https://github.com/Leximon-SWE-2025/Leximon/releases`. Unzip the file and run the leximon.exe executable. If there is a popup that has the title "Windows protected your PC", click "More info" at the bottom of the explanation, and then click the "Run anyway" button at the bottom of the popup. If you want to move the executable, you must also have all the files in the zip in the same directory as the executable.

The controls for the game is a follows

Movement: Use the *WASD* or arrow keys to move.

Exit Game: Use the <escape> key to bring up the exit menu, and click the button to exit the game

View Inventory: Use the *f* key to open in Inventory

Interact with enemy: Use the *e* key when close to an enemy to battle them

Interact with UI: Use the mouse to place the cursor over a UI element and then click it with the left mouse button

5.2 Sample Scenarios

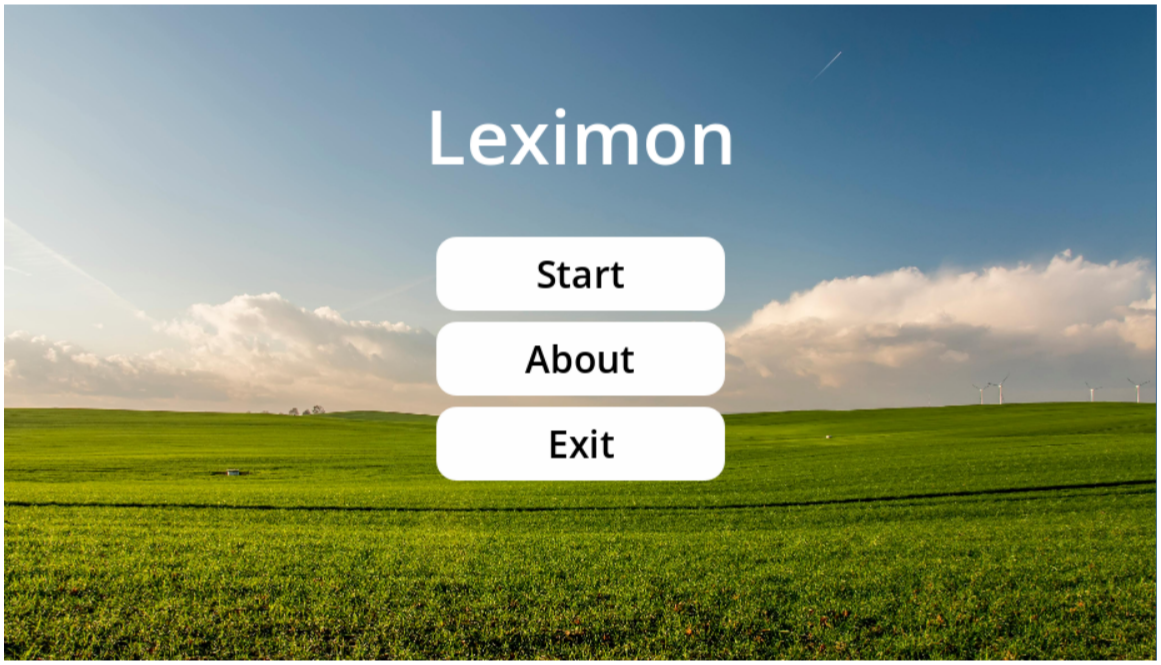


Figure 4: Main menu of the game. This is shown every time the game is opened. From here the player can start playing, quit the game, or go to the website to learn more about the team

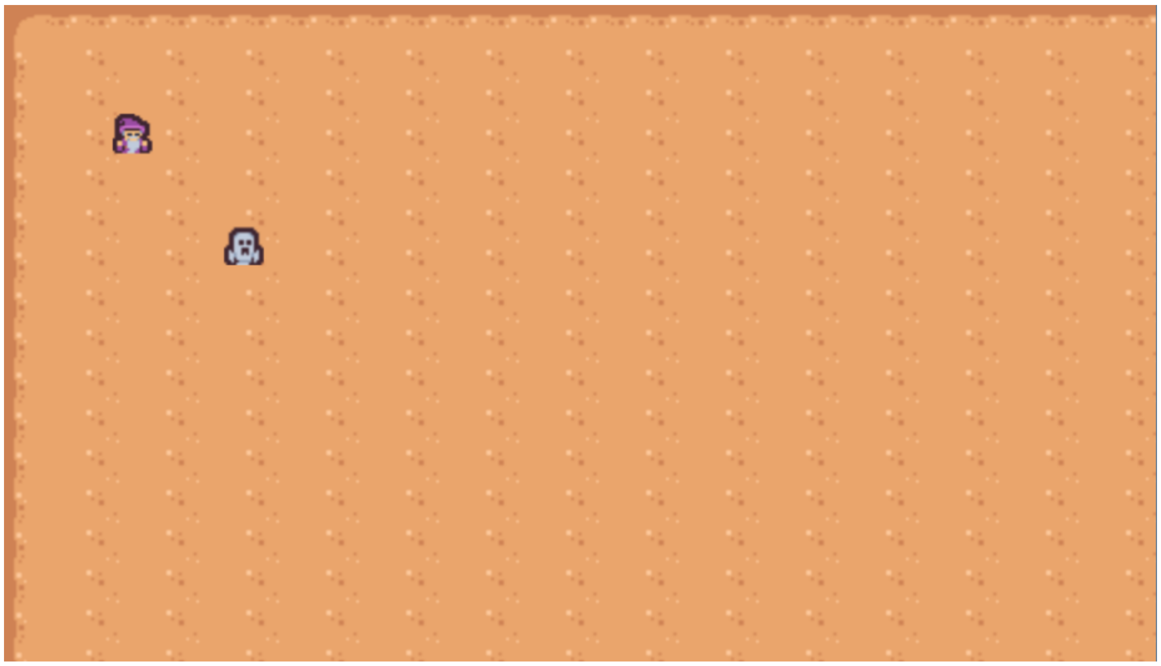


Figure 5: This is the main game world. The player can navigate using the arrow keys or *wasd*. The camera will follow them around

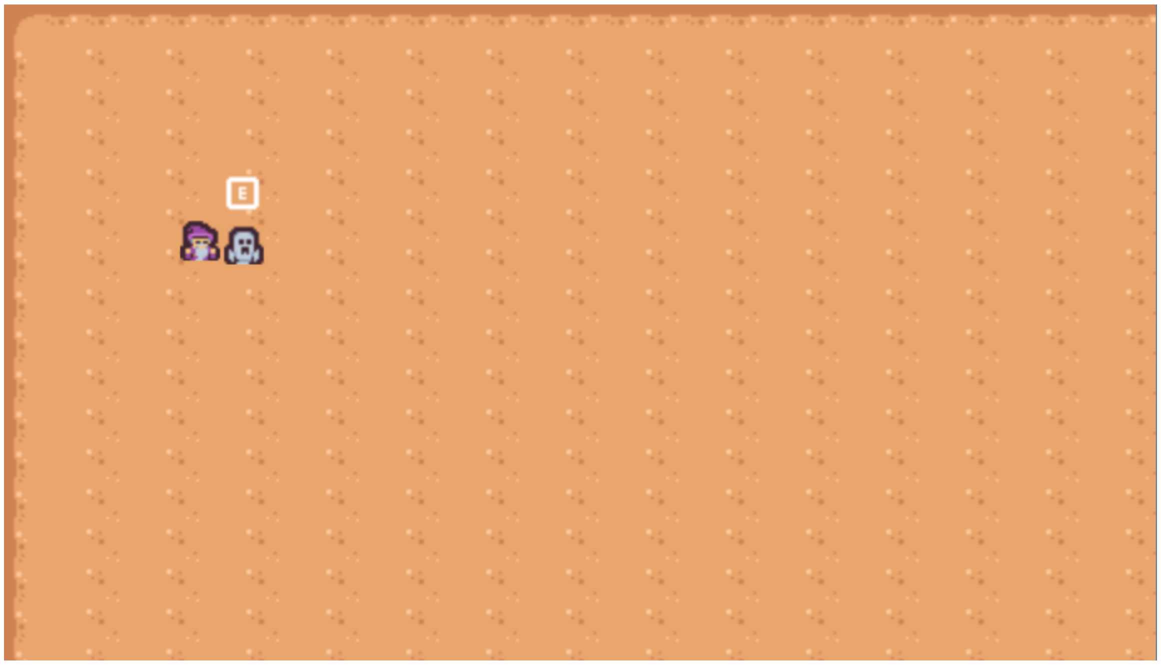


Figure 6: When the player approaches an enemy it will show an interaction prompt. The player can press the indicated key to enter a battle with the enemy

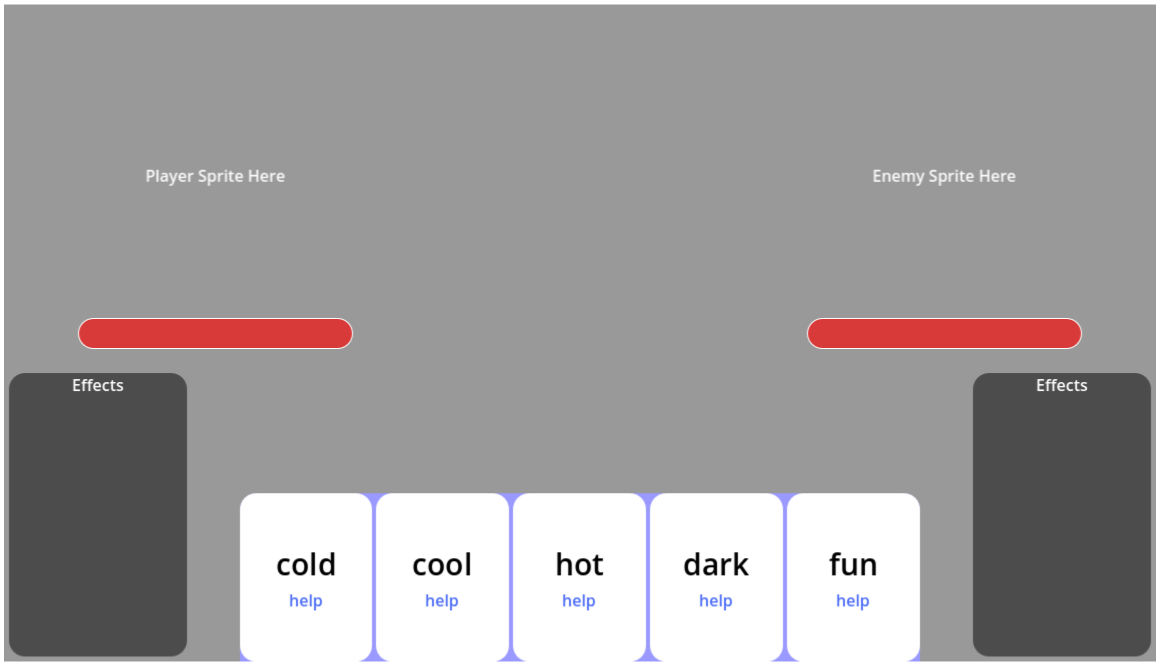


Figure 7: The battle UI. Click on a card to select it. Click on the help text to get the definition of the word on the card.

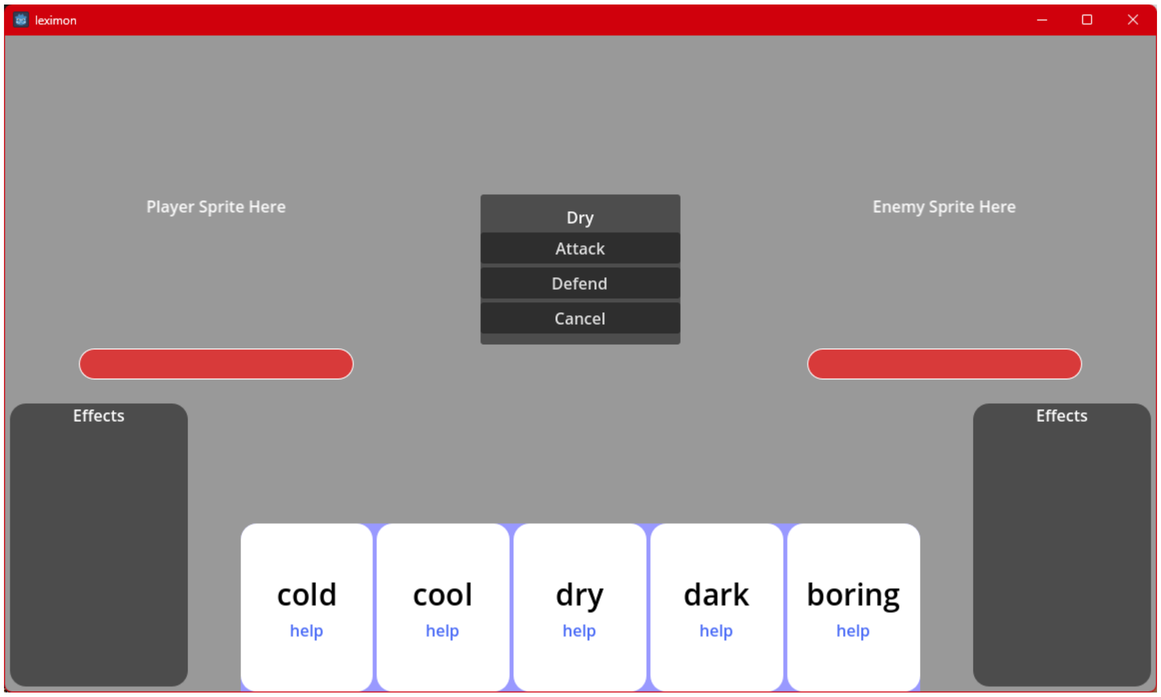


Figure 8: Select whether to use the word to attack or defend

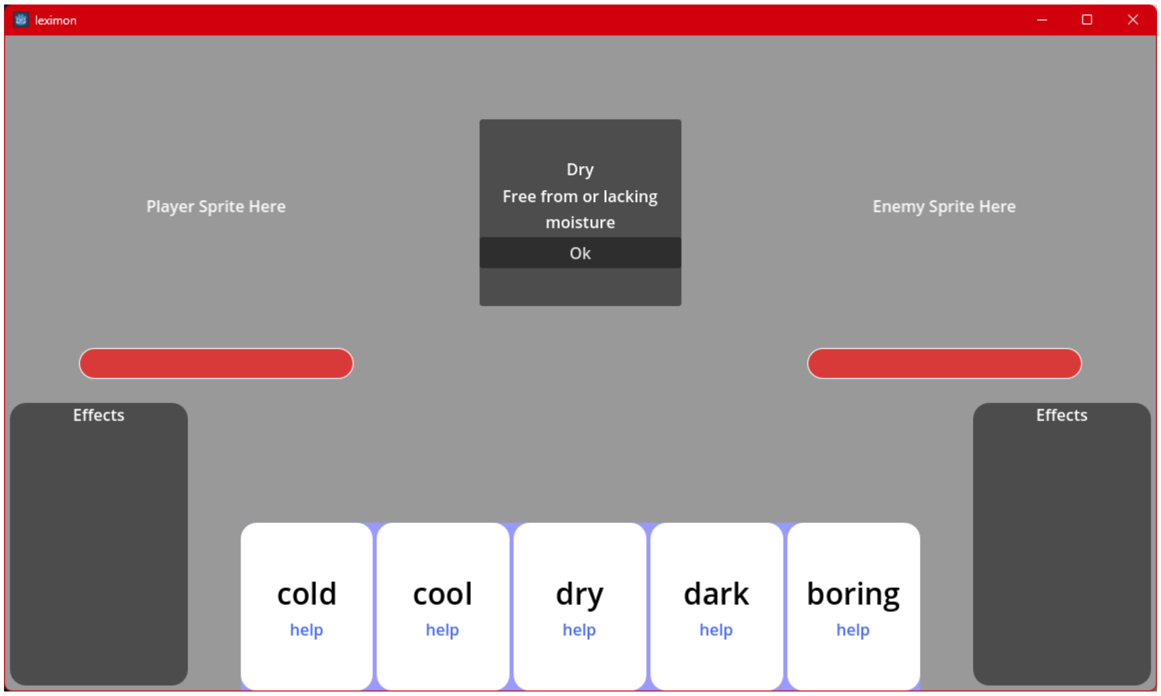
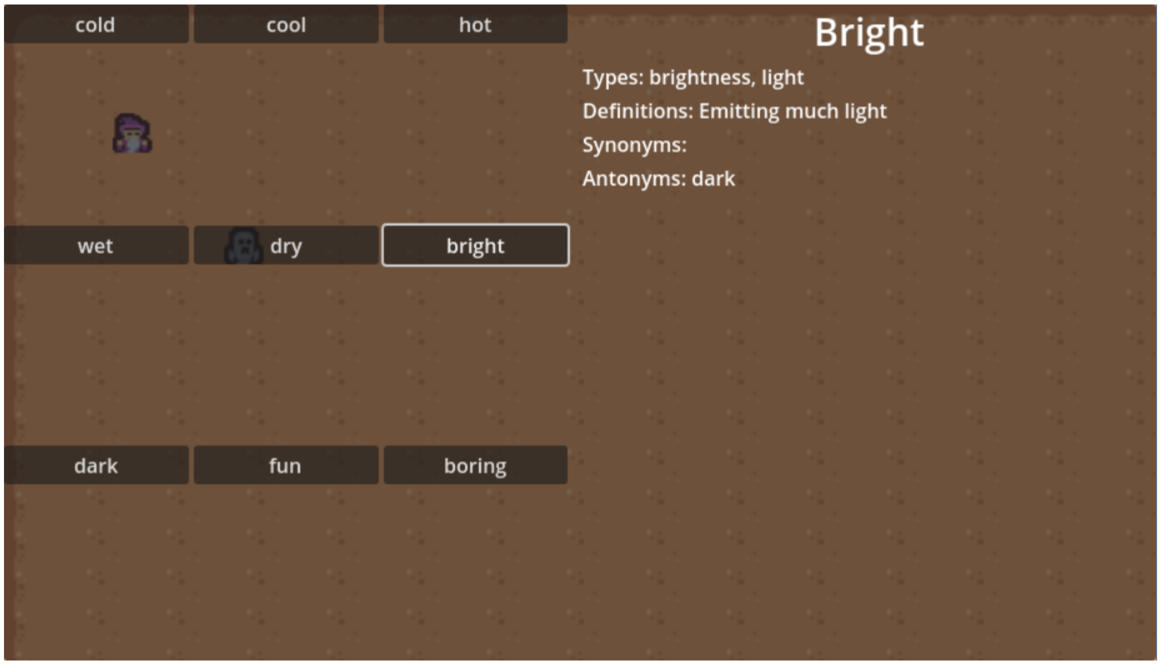


Figure 9: View the definition of the selected word



Bright

Types: brightness, light

Definitions: Emitting much light

Synonyms:

Antonyms: dark

Figure 10: View the list of words the player currently has. Click on one to view the definition, as well as the synonyms, antonyms, and game type

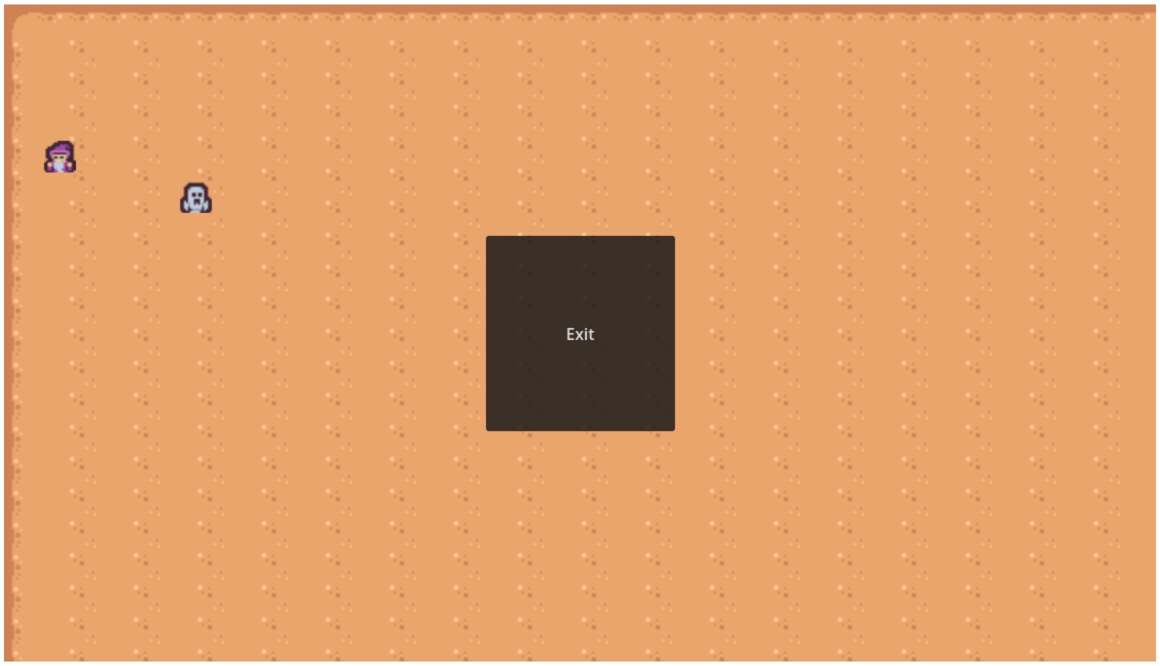


Figure 11: The exit conformation screen. Click the button to exit the game

6 References

[1] C# Docs. *How to read JSON as .NET objects (deserialize).* https://learn.microsoft.com/en-us/dotnet/standard/serialization/system-text-json/deserialization.

[2] C# Docs. *Records (C# Reference).* https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/record.

[3] C# Docs. *Using the StringBuilder Class in .NET.* https://learn.microsoft.com/en-us/dotnet/standard/base-types/stringbuilder.

[4] Godot Engine Documentation. *FileAccess.* https://docs.godotengine.org/en/stable/classes/class_fileaccess.html.

[5] Godot Engine Documentation. *Input examples - Godot Engine documentation.* https://docs.godotengine.org/en/stable/tutorials/inputs/input_examples.html.

[6] Massachusetts Department of Elementary and Secondary Education. *English Language Arts and Literacy Massachusetts Curriculum Framework.* https://www.doe.mass.edu/frameworks/ela/2017-06.pdf. 2017.

[7] kingzonas. *How do I fix error: "No export template found at the expected path: [path]"?* https://forum.godotengine.org/t/how-do-i-fix-error-no-export-template-found-at-the-expected-path-path/1982. 2023.

[8] Kron. *How to Open Web Pages from Godot.* https://www.youtube.com/watch?v=3oWiAF_UbEA. 2021.

[9] *Overview of Renderers.* https://docs.godotengine.org/en/stable/tutorials/rendering/renderers.html.

[10] *Quick way to create a list of values in C#?* https://stackoverflow.com/questions/723211/quick-way-to-create-a-list-of-values-in-c. 2009.

[11] Raul Santos. *Current state of C# platform support in Godot 4.2.* Tech. rep. The Godot Foundation, 2024. URL: https://godotengine.org/article/platform-state-in-csharp-for-godot-4-2/#web.

[12] Platon Supranovich et al. *Leximon.* https://leximon-swe-2025.github.io/Website/. 2025.

[13] u/_P0PCAT_012345. *Grid Container with Scroll.* https://www.reddit.com/r/godot/comments/mhnpgl/grid_container_with_scroll/.

[14] u/Gulferamus. *BoxContainers seem to squish child nodes to nothing.* https://www.reddit.com/r/godot/comments/1ag5agj/boxcontainers_seem_to_squish_child_nodes_to/.

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.