Software Requirements Specification (SRS)
Grammar Grable

Team: Group 4
Authors: Darin Oti Abankwa, Nick Stearns, Andrew Brandao, Jarrett Williams, Andrew Clark
Customer: Massachusetts Department of Education
Instructor: Dr. James Daly

Table Of Contents

1 Introduction

The following sections and subsections have information about the video game prototype Grammar Grable. This prototype is an educational computer game that aims to teach students from the grades of 4th-6th grade on vocab, definitions, and punctuation. Following this section will be subsections with the overarching description of this prototype. After that, Grammar Grable will be broken down into the many different design aspects of it: the specific requirements for it, followed by the modeling requirements and the abstract concept of the entire design. Pictures of this prototype will be shown following this section to show the general gameplay design of the game that connects the different design aspects. There will also be a resources section at the end to give credit to the multitude of resources we have used.

1.1 Purpose

The Software Requirements Specification (SRS) document provides a complete description of all function, specification, and constraints for the software system Grammar Grable. The primary purpose of this document is to define what and why of the system, not the implementation. It acts as the core agreement between the development team and stakeholders, showing a clear and shared understanding of the project's objectives and expected outcomes.

The intended audience for this document is as follows:

- Customers and Stakeholders: This includes the clients, and Prof. Daly. This document allows them to ensure that the requirements have been accrued.
- Development Team: The software engineers, designers, and architects will use this document as the source for designing. It is their guide for what to build.

1.2 Scope

Grammar Grable is an edutainment based game designed to increase student engagement and incentivise learning of multiple grammar concepts, including vocab, punctuation, and definitions. Students struggle with maintaining and studying their vocab and punctuation due to a lack of engagement with just the definitions and the words. This game will tie these concepts to a fun puzzle escape room esque game where they must find the correct key item with the right answer to unlock and complete the door blocking them while avoiding the teacher trying to catch them for slacking. This will allow students to engage both with fun and learning at the same time.

1.3 Definitions, acronyms, and abbreviations

Below is a list of definitions and acronyms that are used in our game that are not inherently understandable without a proper introduction.

Grammar Grable: The title of the game that this document describes

Key/item: interactable items that represent words that players will use to fill in a statement to attempt and unlock a door

Player: The person playing the game

Room: A subsection of the map that the player can walk around contained within 4 walls

School: The location in which the player can walk around containing all rooms

Teacher: The enemy that the player must avoid by walking awaying

Obstacles: Objects that a player cannot move through and must go around

Question: A Statement that will have a blank space that a key can fulfill

Doors: A closed pathway that needs a key to become passable

Popup: Text that appears when player grabs an item or interacts with door

DetectionCone: The implementation of how an enemy in the level detects a player, it is a hidden cone that moves with the enemy for detecting the player.

1.4 Organization

The remainder of this document comprises 6 additional sections. Section 2 provides a detailed description of the product, encompassing its prospective functions, user characteristics, constraints, assumptions, and allocated requirements. Section 3 presents a comprehensive list of all specific requirements that the software must fulfill. Section 4 offers a technical overview of the modeling requirements that the software must satisfy, which includes a use case diagram, a class diagram, two sequence diagrams, and a state diagram. Section 5 outlines the prototypes of the software that will be delivered prior to the final product. Section 6 contains a compilation of references, while Section 7 identifies the point of contact.

2 Overall Description

Our game Grammar Grable is designed as an educational game involving various vocabulary and definition based questions. The difficulty of these questions range from 4th-6th grade. It is a two-dimensional level-based exploration and puzzle game.

The following section will explain the context of the project and these following subsections: the product's context and functions, a description for the specific user characteristics for the ideal users of this product, the limitations and restrictions of the project, the assumptions and dependencies of the project.

2.1 Product Perspective

Grammar Grable is an educational exploration and puzzle game that is a standalone application with the goal of educating elementary and early middle school students on vocabulary and definitions. This game requires no integration with any other larger system. This game can be played on a student's personal computer or while in the school's computer lab on each individual computer. The game will not interact with any external source and can be run on most computers without issue.

Students playing Grammar Grable will learn about a variety of different vocabulary and definitions based on the difficulty they select. This game will help them prepare for the vocabulary exams and quizzes they might receive as well as reading comprehension as they will understand more words as they read.

2.2 Product Functions

The product is an edutainment game. To play the game the user will select a specific difficulty. Specific buttons will be displayed to make this selection. The user can select easy, medium, or hard for a range of difficulties. After they make their selection they can select the Play button to start the game.

Once the game itself is started, a player-controlled character can move about the level. The objective is to find specific items that can be used to unlock each room's door. Each of these items will either have a word or a definition that will be displayed to the user. These items can then be used on the locked door in the level by selecting the correct one from the player's inventory with their mouse.

Below is a high-level diagram for Grammar Grable. Represented in this diagram is the ultimate goal of what we are trying to achieve in this game. It is derived from the 3 key features we are adding to the game. The first being randomly selected questions and answers for the key items to unlock the doors, the second being the enemy the player must avoid throughout the level, and the third being the difficulty selection for the questions.

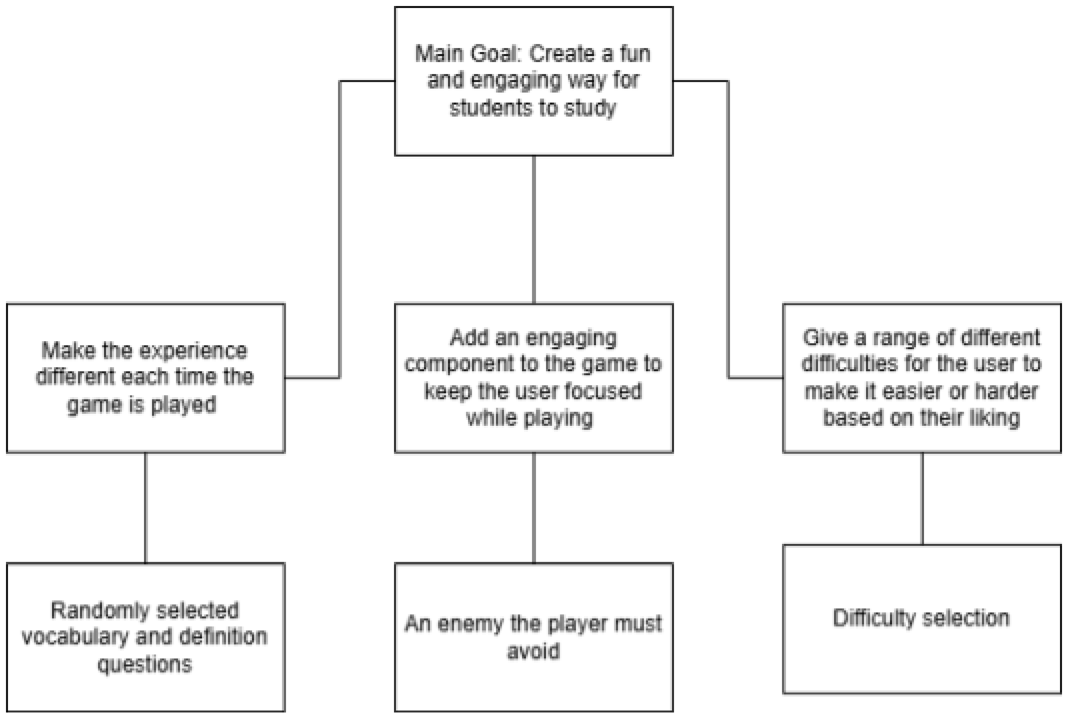

Main Goal: Create a fun and engaging way for students to study
Make the experience different each time the game is played
Add an engaging component to the game to keep the user focused while playing
Give a range of different difficulties for the user to make it easier or harder based on their liking
Randomly selected vocabulary and definition questions
An enemy the player must avoid
Difficulty selection

Figure 1: Breakdown Of Game

2.3 User Characteristics

The user is expected to have basic knowledge of a computer and how to use a keyboard and mouse to be able to play. The target audience for Grammar Grable is 4th to 6th grade students. They are expected to have learned enough basic vocabulary in order to understand the definitions and be able to associate them with their words. These vocabulary questions will assist them with their understanding for general reading vocabulary, mathematics vocabulary, and science vocabulary as listed for the 4th to 6th level in the Massachusetts Department of Education Curriculum.

2.4 Constraints

The game must be understandable and educational for students in grades 4–6. It must follow all guidelines in the Common Core State Standards for English Language Arts. The game also needs to be safe for students in grade 4 and above and meet all requirements for that age rating, including any restrictions connected to it.

2.5 Assumptions and Dependencies

It is assumed that the user will use a modern computer with either macOS or Windows 11 as their operating system. The user's computer should have a functioning keyboard and mouse. To download the game the users must have internet access, however to play the game when downloaded and installed, no internet access is required. It is assumed that the computer or laptop has enough power, or a stable power connection. It is also assumed that the users will have at least basic English reading comprehension skills.

2.6 Apportioning of Requirements

There will be no level-select, or more rooms and locations outside of the basic map. This game is designed only for a prototype meaning there will only be the preset levels created in order for there be enough gameplay to demonstrate the game's concept. For future improvements, the map can be expanded, and a level-select may be implemented, however for the basis of a prototype, only basic functionality is implemented.

3 Specific Requirements

1. The game shall provide a comprehensive menu system.
    - 1.1. The game shall display a Start Menu upon launch.
    - 1.2. The Start Menu shall provide options to start a new game and access other menus.
        - 1.2.1. The Start Menu shall give the player the ability to select the difficulty level (e.g., Easy, Medium, Hard).
    - 1.3. The game shall have a Controls Menu that displays the input mappings to the player.
    - 1.4. The game shall have a Pause Menu, accessible during gameplay.
        - 1.4.1. The Pause Menu shall provide options to resume, quit, and view the Controls Menu
    - 1.5. The game shall display a score to the user during gameplay.
        - 1.5.1. Upon completing the game, an End Game menu shall display the score the player managed to achieve.
        - 1.5.2. The game shall have a High Score menu that displays the saved users previous best five runs.
2. The game shall feature a player character that can be controlled to navigate the environment.
    - 2.1. The player character's movement shall be constrained by the environment.
        - 2.1.1. The character shall not be able to move through walls.
        - 2.1.2. The character shall not be able to move through locked doors.
        - 2.1.3. The character shall not be able to move through objects.
    - 2.2. The player character shall be able to move between rooms or levels.
3. The core gameplay shall involve solving grammar based puzzles to progress through multiple levels while avoiding enemies.
    - 3.1. The game shall consist of multiple levels.
    - 3.2. Each level shall contain key items that are solutions to grammar problems.
        - 3.2.1. Key items shall be visually distinct and placed throughout the level.
        - 3.2.2. Key item types shall include, but are not limited to: Nouns, Verbs, Adjectives, Adverbs, Punctuation Marks, and Correctly Spelled Words.
    - 3.3. The game shall contain interactable door objects.
        - 3.3.1. Door objects shall have a locked and unlocked state.
            - 3.3.1.1. Locked door objects shall present grammar questions to the player upon interaction.
    - 3.4. To solve a problem, the player must find and use the correct key item.
4. The game shall feature enemies that can detect the player.
    - 4.1. Enemies shall pursue detected players and deal damage on contact.
    - 4.2. The enemies must have a visual cue for detection.
5. The grammar questions shall be designed to match the English curriculum for students in grades 4 through 6.

5.1. The grammar questions shall be randomly selected from pools of questions separated by difficulty (eg. Easy, Medium, Hard).

5.1.1. Question difficulty shall correspond with the player's selected difficulty on the Main Menu.

5.2. All questions shall have a clear and defined answer.

6. The game shall provide clear visual feedback to the player based upon user actions.

6.1. Descriptive text shall appear when the player interacts with key objects, including locked doors, key items, and non-player characters.

6.2. The game shall provide clear visual or audio feedback upon successful or failed puzzle attempts.

4 Modeling Requirements

Below, there are a series of diagrams that show the internal works of how our game functions. The following section contains a use case diagram, a class diagram, two sequence diagrams, and a state diagram to show the products functionality.

4.1 Use Case Diagram

The use case diagram describes the user's interactions with the game where the primary objective is to open a door and advance to the next level. When trying to open a door, a question will always be asked and the user can either give a correct or incorrect answer. If the player answers correctly, they progress to the next level. If on the final level the player wins the game. Incorrect answers result in a punishment for the player.

Answering the question correctly requires having added the correct item to the player inventory, making adding items a significant subgoal. The player also has the option to pause the game, during which they can view the controls, change the difficulty, or resume play. The player always has the option to exit the game.

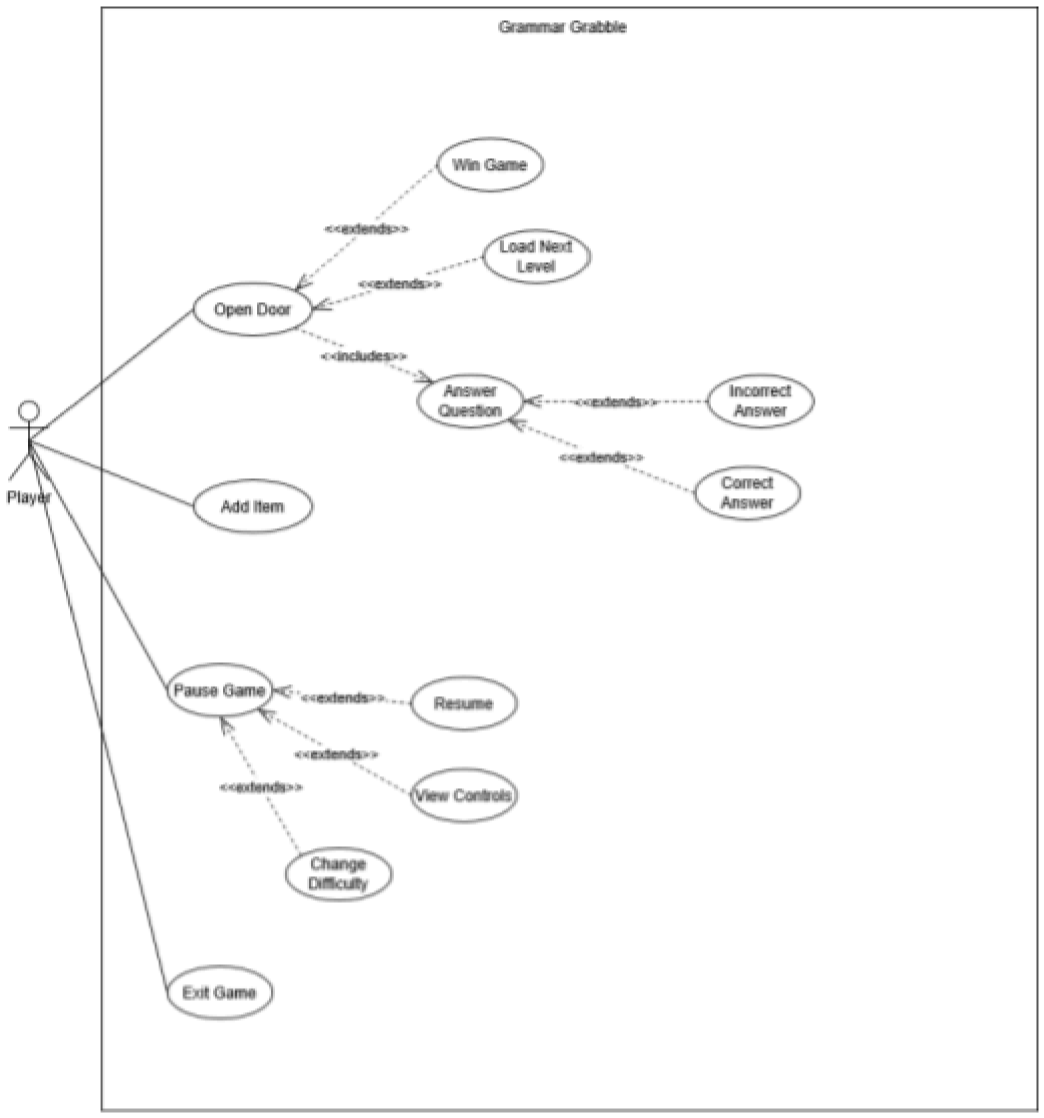


Figure 2: Player Interaction

Use Case Name:	Open Door
Actors:	Player
Description:	Open the door by correctly answering a question.
Type:	Primary and essential
Includes:	Answer Question
Extends:	None
Uses cases:	Opening doors is required to advance to the next level, completing all levels is required to win the game.

Use Case Name:	Answer Question

Actors:	Player
Description:	A question will be displayed which the player is required to correctly answer in order to unlock the door.
Type:	Primary and essential
Includes:	None
Extends:	None
Uses cases:	Correctly answering the question is required to open the door.

Use Case Name:	Add Item
Actors:	Player
Description:	Players can add items to their inventory by interacting with them.
Type:	Secondary and essential
Includes:	None
Extends:	None
Uses cases:	The correct item is needed to correctly answer the question.

Use Case Name:	Resume
Actors:	Player
Description:	The player can resume the game if the game is currently paused.
Type:	Secondary
Includes:	None
Extends:	Pause
Uses cases:	Returns a paused game to normal gameplay.

Use Case Name:	Correct Answer
Actors:	Player
Description:	If the player answers the question correctly, they will be granted the ability to open the door.
Type:	Secondary and essential
Includes:	None
Extends:	Answer Question
Uses cases:	Required to unlock a door.

Use Case Name:	Change Difficulty
Actors:	Player
Description:	Allows the player to change the difficulty of the game when they are in the pause menu.
Type:	Secondary
Includes:	None
Extends:	Pause
Uses cases:	Used to alter the game's difficulty.

Use Case Name:	Exit Game
Actors:	Player
Description:	Allows the player to exit the game.
Type:	Secondary
Includes:	None
Extends:	None
Uses cases:	The player can choose to exit the game at any time.

Use Case Name:	Pause
Actors:	Player
Description:	Allows the player to pause the game during gameplay.
Type:	Secondary
Includes:	None
Extends:	None
Uses cases:	To pause the game and allow for the difficulty to be changed or controls viewed.

Use Case Name:	View Controls
Actors:	Player
Description:	Allows the player to view the control scheme if they are in the pause menu.
Type:	Secondary
Includes:	None
Extends:	Pause
Uses cases:	To view the game's control scheme.

Use Case Name:	Win Game
Actors:	Player
Description:	The player has completed all levels and won the game.
Type:	Primary and essential
Includes:	None
Extends:	Open Door
Uses cases:	The game is won when the final door is opened.

Use Case Name:	Incorrect Answer
Actors:	Player
Description:	The player answers the question incorrectly.
Type:	Secondary
Includes:	None
Extends:	Answer Question
Uses cases:	Punishes the player if they give an incorrect answer.

Use Case Name:	Load Next Level
Actors:	Player
Description:	Loads the next level after the player has successfully opened a door.
Type:	Primary and essential
Includes:	None
Extends:	Open Door
Uses cases:	All levels need to be completed for the game to be won.

4.2 Class Diagram

This diagram details the relationship between the various classes within the game. The class diagram has seven major classes with one enumeration type and an interface class. The PlayerCharacter class houses attributes and methods tied to the playable character, such as the player's health, list of the player's inventory, and the character's RigidBody. PlayerCharacter's methods include functions that move the character, keep track of the player's health, interactions with interactable objects, and using and collecting Item objects. The Enemy class inherits from the Interactable interface and houses only one additional attribute, its ID. The Enemy class has methods that relate to moving, detecting the player, and following a detected player. The

DetectionCone class handles field of view detection between objects, and is used within the Enemy class’s detection method.

The main intractable objects in the game have their own classes: Door and Item. The Door class has attributes such as a boolean to track if it is locked or not and the question that is displayed when the door is interacted with. The Item class contains attributes regarding whether or not it is the key which correctly answers the question and the item's ID. The Item class's only method consists of removing the item from the player. PlayerCharacter’s store added Item objects in an inventory field. The Door and Item classes inherit from the Interactable interface.

QuestionManager and QuestionBank are central to the game’s question logic. The QuestionManager class is responsible for randomly selecting questions and assigning them to doors when levels load. It also evaluates if a player’s answer is correct or incorrect. QuestionManager also uses the Difficulty enumeration to store the player’s selected difficulty. QuestionManager contains a QuestionBank object, an instance of the QuestionBank class which is used to store pools of questions and answers as key-value pairs, separated by difficulty.

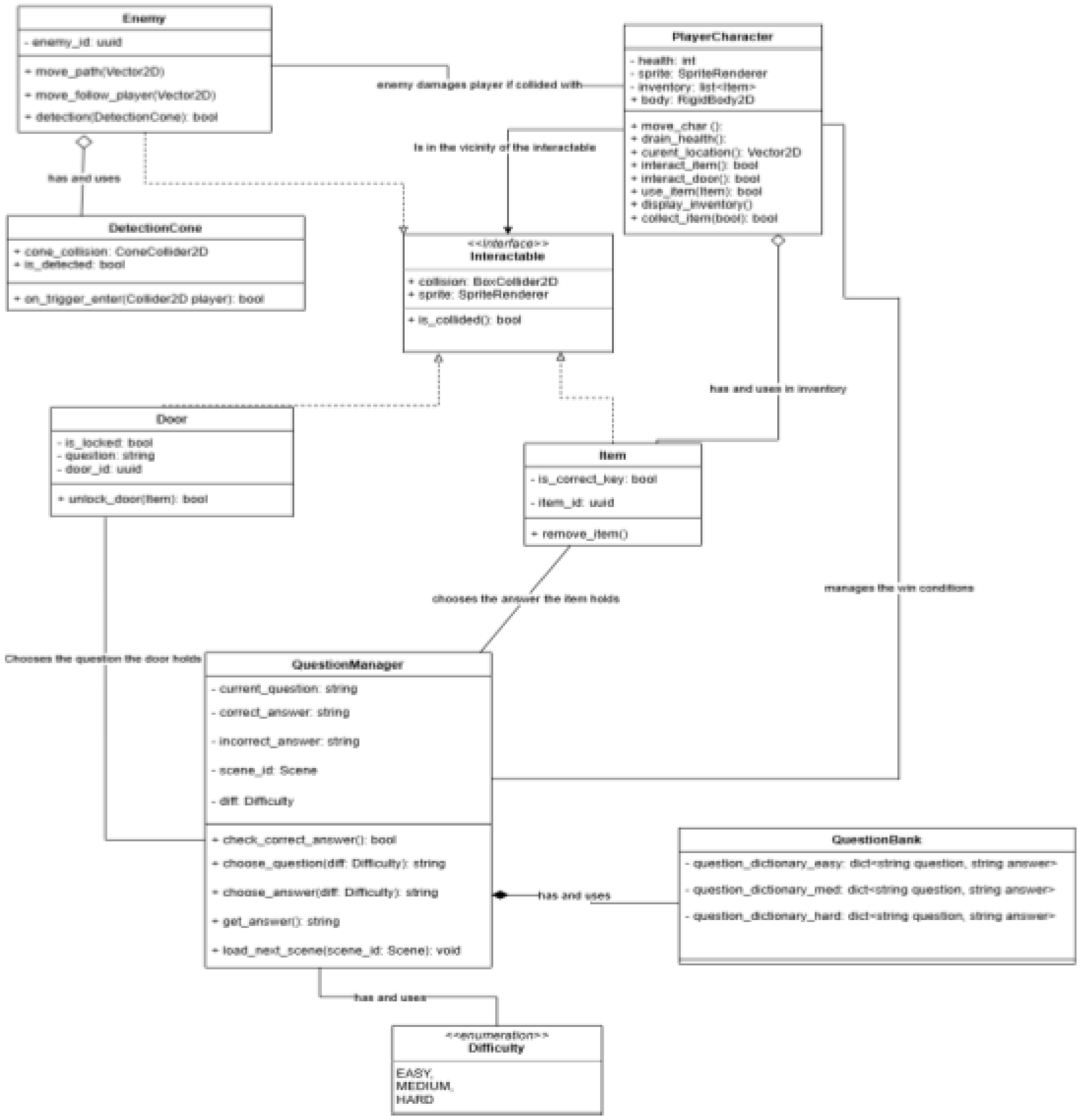

Enemy
- enemy_id: uuid
+ move_path(Vector2D)
+ move_follow_player(Vector2D)
+ detection(DetectionCone): bool
enemy damages player if collided with
PlayerCharacter
- health: int
- sprite: SpriteRenderer
- inventory: list<Item>
+ body: RigidBody2D
+ move_char ():
+ drain_health():
+ curent_location(): Vector2D
+ interact_item(): bool
+ interact_door(): bool
+ use_item(Item): bool
+ display_inventory()
+ collect_item(bool): bool
Is in the vicinity of the interactable
has and uses
DetectionCone
+ cone_collision: ConeCollider2D
+ is_detected: bool
+ on_trigger_enter(Collider2D player): bool
<<interface>>
Interactable
+ collision: BoxCollider2D
+ sprite: SpriteRenderer
+ is_collided(): bool
has and uses in inventory
Door
- is_locked: bool
- question: string
- door_id: uuid
+ unlock_door(Item): bool
Item
- is_correct_key: bool
- item_id: uuid
+ remove_item()
chooses the answer the item holds
manages the win conditions
Chooses the question the door holds
QuestionManager
- current_question: string
- correct_answer: string
- incorrect_answer: string
- scene_id: Scene
- diff: Difficulty
+ check_correct_answer(): bool
+ choose_question(diff: Difficulty): string
+ choose_answer(diff: Difficulty): string
+ get_answer(): string
+ load_next_scene(scene_id: Scene): void
has and uses
QuestionBank
- question_dictionary_easy: dict<string question, string answer>
- question_dictionary_med: dict<string question, string answer>
- question_dictionary_hard: dict<string question, string answer>
has and uses
<<enumeration>>
Difficulty
EASY,
MEDIUM,
HARD

Figure 3: Class Diagram

Class Data Dictionary

Class Name:	Interactable <<interface>>
Description:	An object or class that can interact with the enemy is derived from this class.
Attributes:	is_collided(): bool
Methods:	Collision: BoxCollider2D Sprite: SpriteRenderer
Relationships:	Enemy class is derived from Interactable. PlayerCharacter class is derived from Interactable. Door class is derived from Interactable. Item class is derived from Interactable.

Class Name:	DetectionCone
Description:	A Unity collision detector that is representative of vision to detect if the player is within the enemies field of view.
Attributes:	on_trigger_enter(Collider2D player): bool
Methods:	cone_collision: ConeCollider2D is_detected: bool
Relationships:	Enemy class uses and has DetectionCone class.

Class Name:	PlayerCharacter
Description:	Represents the player and their actions within the game world.
Attributes:	Health: int Sprite: SpriteRenderer inventory:list<Item> Body: RigidBody2D
Methods:	move_char() drain_health() current_location(): Vector2D interact_item(): bool interact_door(): bool use_item(Item): bool display_inventory() collect_item(bool): bool
Relationships:	Derives from Interactive class. Can have and use Item class.

	Can collide with the Enemy class. The QuestionManager class manages the win conditions.

Class Name:	Enemy
Description:	Searches for the player and attempts to collide with player to lower player health
Attributes:	Enemy_id: uuid

Methods:	move_path(Vector2D)
	move_follow_player(Vector2D)
	detection(DetectionCone): bool

Relationships:	Derived from Interactable class. Uses the DetectionCone class. Can collide with PlayerCharacter.

Class Name:	Item
Description:	Collectables that the player can use as answers to the question.
Attributes:	is_correct_key: bool item_id: uuid
Methods:	remove_item()
Relationships:	Derived from Interactable class. PlayerCharacter can collect Item objects into its inventory field. QuestionManger chooses which answer the item holds.

Class Name:	Door
Description:	A blocked pathway representative of a door which the player must interact with to use an item to answer a popup question.
Attributes:	is_locked: bool question: string door_id: uuid
Methods:	unlock_door(Item): bool
Relationships:	Derived from Interactable class. QuestionManager chooses the question the door holds.

Class Name:	QuestionManger
Description:	A class that directs the questions which appear as a popup on the door and the answers that appear as items.
Attributes:	current_question: string correct_answer: string incorrect_answer: string scene_id: Scene diff: Difficulty
Methods:	check_correct_answer(): bool choose_question(diff: Difficulty): string choose_answer(diff: Difficulty): string get_answer(): string load_next_scene(scene_id: Scene): void
Relationships:	QuestionManager provides the Door class with a question. QuestionManger gives items the answers they hold. QuestionManager manages the win conditions. QuestionManger uses the Difficulty class to choose an appropriate level of question. QuestionManager has and uses the QuestionBank class to select from a variety of potential questions.

Class Name:	QuestionBank
Description:	A storage class with all potential questions of varying difficulties.
Attributes:	question_dictionary_easy: dict<string question, string answer> question_dictionary_med: dict<string question, string answer> question_dictionary_hard: dict<string question, string answer>
Methods:	N/A
Relationships:	QuestionManger has and uses QuestionBank to select appropriate questions.

Class Name:	Difficulty
Description:	An enumeration class that represents the difficulty level selected.
Attributes:	EASY MEDIUM HARD
Methods:	N/A

Relationships:	QuestionManger uses Difficulty class to select appropriate level questions based on user selected difficulty.

4.3 Sequence Diagrams: 1

Sequence Diagram 1 depicts an element of core gameplay. When a level loads, the QuestionManager calls a method on itself to randomly select a question and assigns it to a door. Each question has an answer paired with it, this answer is assigned to an item in the level. A player then presses the interact with item button. If they are near an item, it is added to their inventory. The player then presses interact with the door, if the door is locked, the question is displayed. The player then chooses an item from their inventory as the answer. If the answer is correct, the door is unlocked and the next level is loaded. If not, normal play resumes.

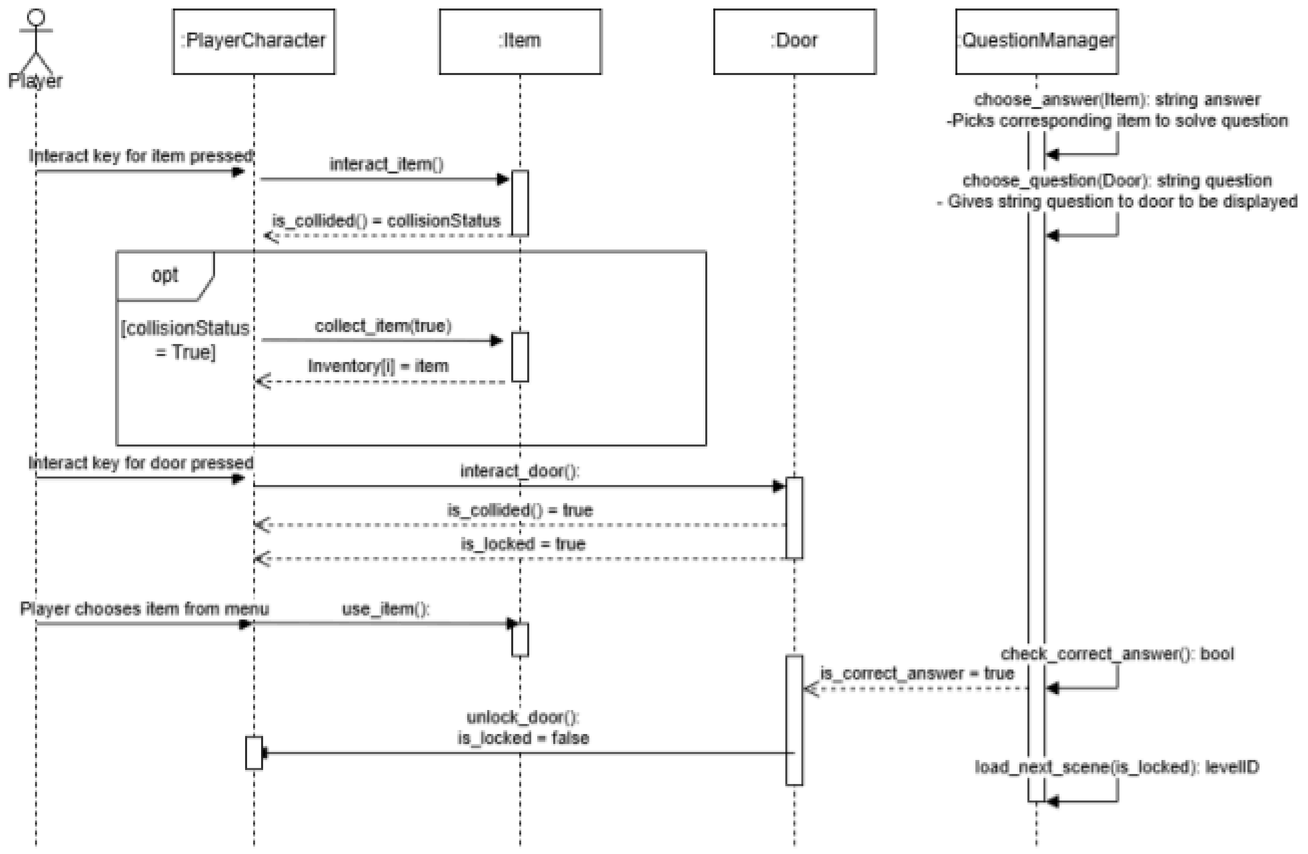

Player
:PlayerCharacter
:Item
:Door
:QuestionManager
choose_answer(Item): string answer
-Picks corresponding item to solve question
choose_question(Door): string question
- Gives string question to door to be displayed
Interact key for item pressed
interact_item()
is_collided() = collisionStatus
opt
[collisionStatus = True]
collect_item(true)
Inventory[i] = item
Interact key for door pressed
interact_door():
is_collided() = true
is_locked = true
Player chooses item from menu
use_item():
check_correct_answer(): bool
is_correct_answer = true
unlock_door():
is_locked = false
load_next_scene(is_locked): levelID

Figure 4: How Player Interacts With Item And Door

4.4 Sequence Diagrams: 2

Sequence Diagram 2 is another piece of core gameplay. An enemy in the level follows a certain set path throughout the level. As soon as the enemy detects the player within its Detection Cone, it then will start following the player. As soon as the player collides with the enemy, the player then loses health. If the player escapes the enemies vision, the enemy reverts to following its main path.

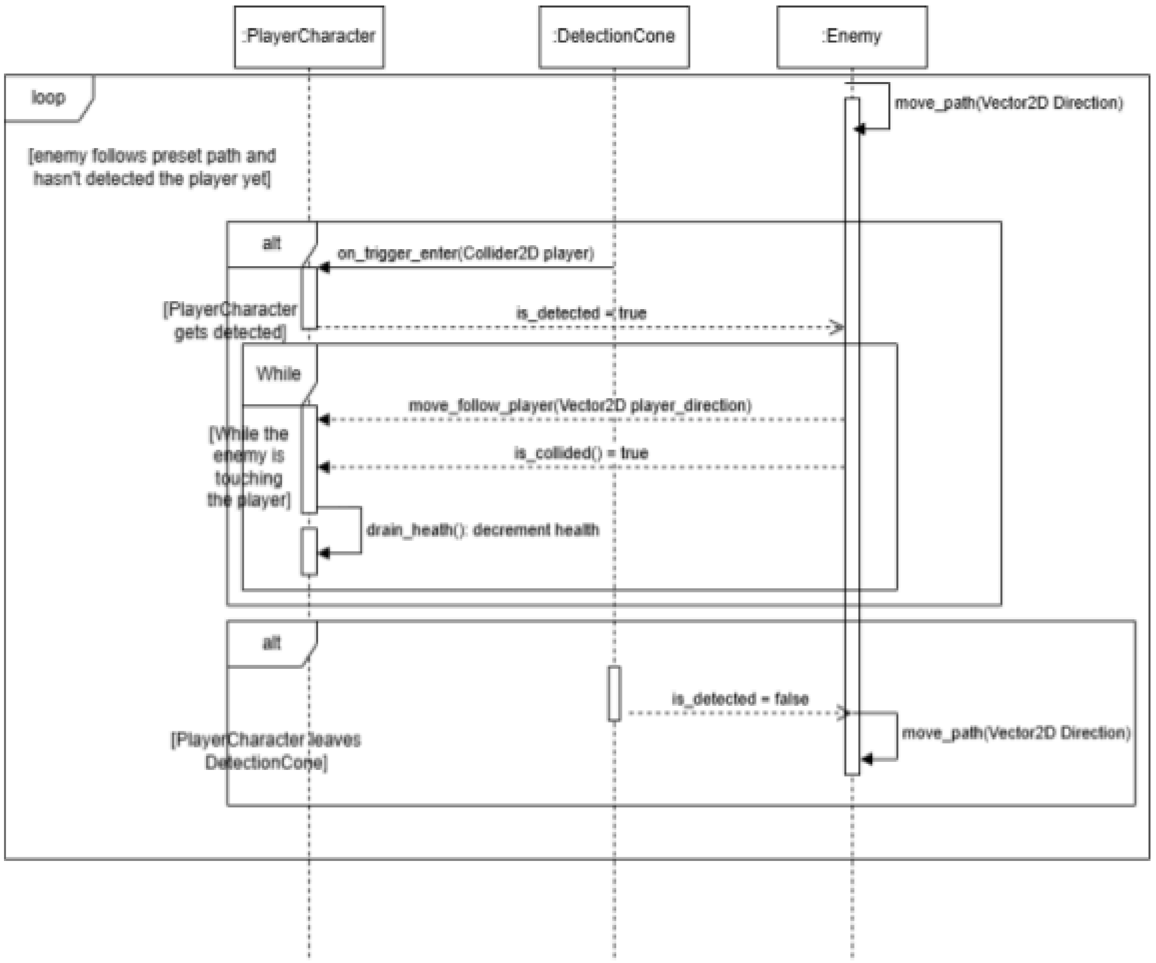

:PlayerCharacter
:DetectionCone
:Enemy
loop
move_path(Vector2D Direction)
[enemy follows preset path and hasn't detected the player yet]
alt
on_trigger_enter(Collider2D player)
[PlayerCharacter gets detected]
is_detected = true
While
move_follow_player(Vector2D player_direction)
[While the enemy is touching the player]
is_collided() = true
drain_heath(): decrement health
alt
is_detected = false
move_path(Vector2D Direction)
[PlayerCharacter leaves DetectionCone]

Figure 5: The sequence for the enemy detecting and attacking the player

4.5 State Diagram

The state diagram describes the control flow of the game. When the game is first started the player will be at the main menu. Here they have options to view and change the difficulty, view the controls, exit, and play the game. When playing the game a level will first be loaded, then the game will be in a normal gameplay state where the player can move about and collect items. When a player interacts with a door, a new state will be entered where a question is asked. An incorrect answer by the player leads back to normal gameplay, while a correct answer causes the next level to load. If the player has reached the last level, a correct answer will win the game and the player will return to the main menu.

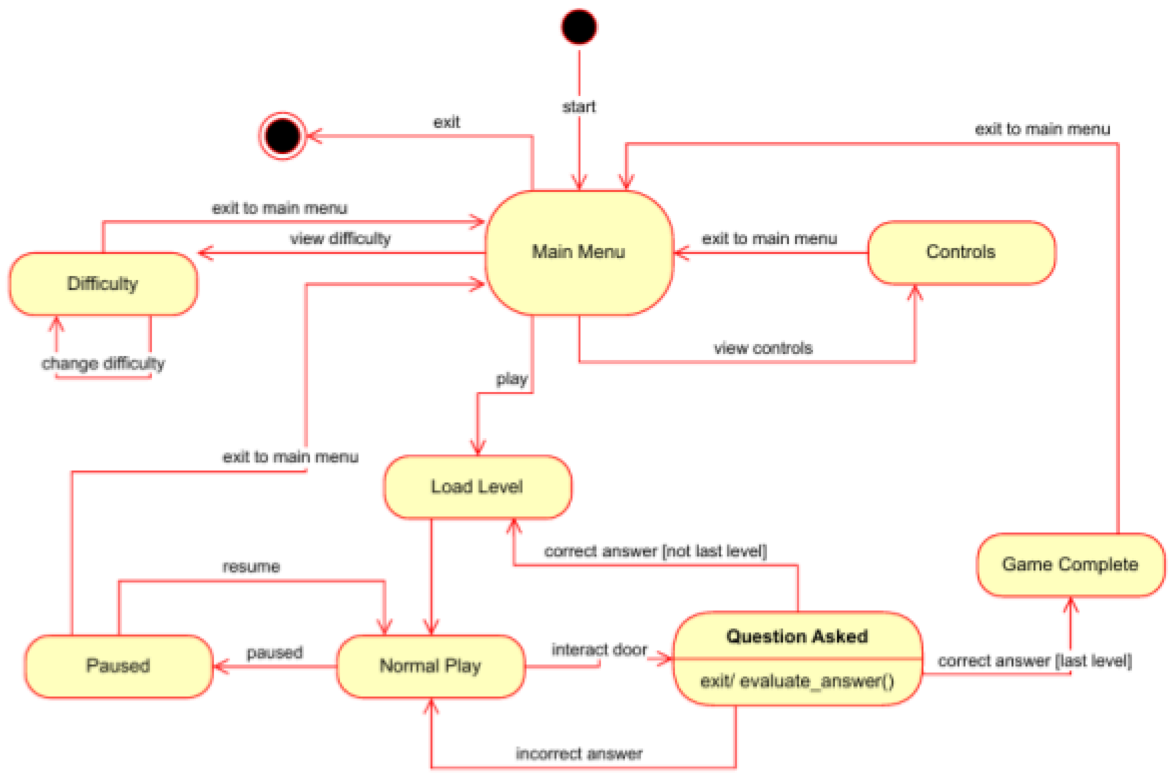


Figure 6: State Diagram

5 Prototype

In terms of system functionality, the prototype will be playable and will feature 3 different difficulty options for the player: easy, medium, and hard. Once selected the main game will initiate where the player will traverse around the level to find the various grammar clues to reach to the next level. Gameplay will display enemy navigation, hierarchical question system for checking if the plate acquired the correct answer, and level progression system to move on to the next level if grammar questions were answered successfully.

5.1 How to Run Prototype

The prototype v1 can be run through the Unity Editor in which you need both the Unity Hub and Specific Unity Editor version required to run the game. The Editor version is 6000.2.10f1 which must be installed before running in the editor. You must then open the project into the Unity Editor and click the play button at the top of the editor to run the game. Also make sure the game window in the Unity Editor is on Full HD (1080p).

5.2 Sample Scenarios

5.2.1 The main menu will allow the player to select the difficulty of the questions, view the controls, and start or stop the game by using the left-click on the mouse.

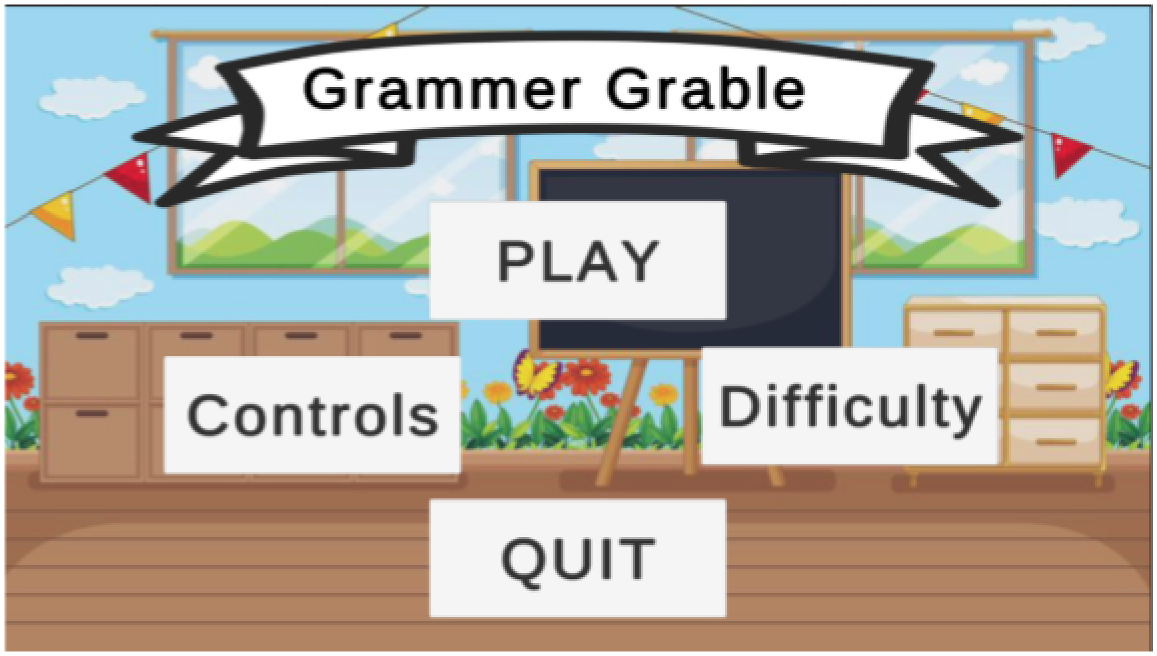


Figure 7: Main Menu

5.2.2 Difficulty selection menu with a back button to be able to return to the main menu still using the left click for all use on the menu.

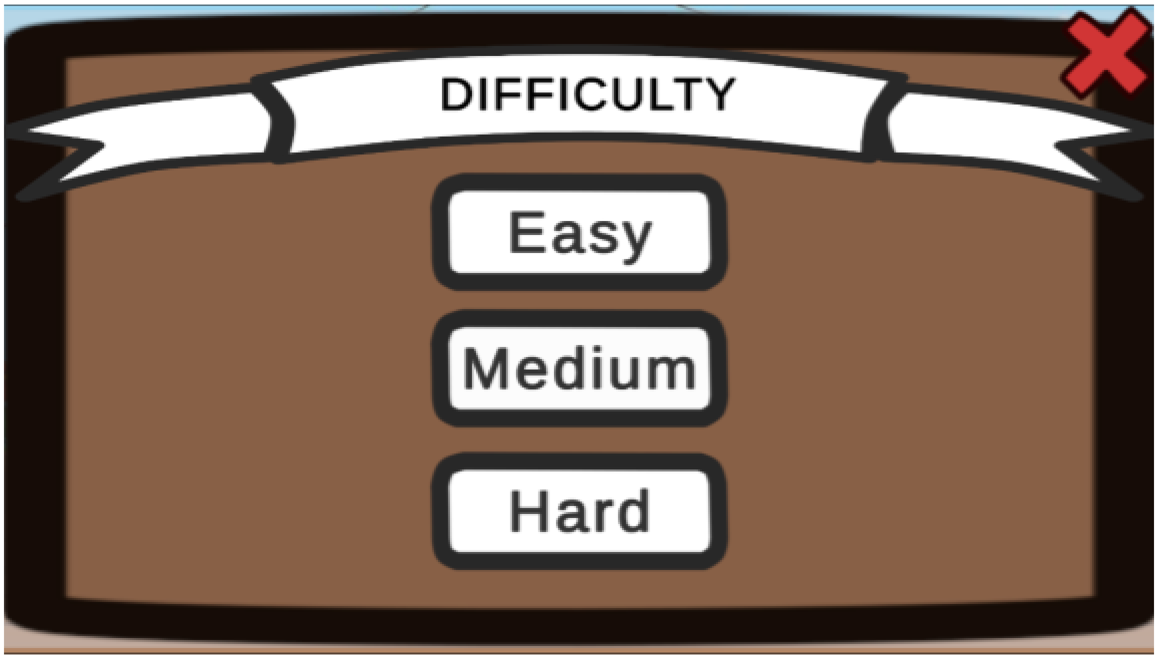


Figure 8: Difficulty Panel

5.2.3 Controls menu to display the controls to the user in order for them to understand how to play.

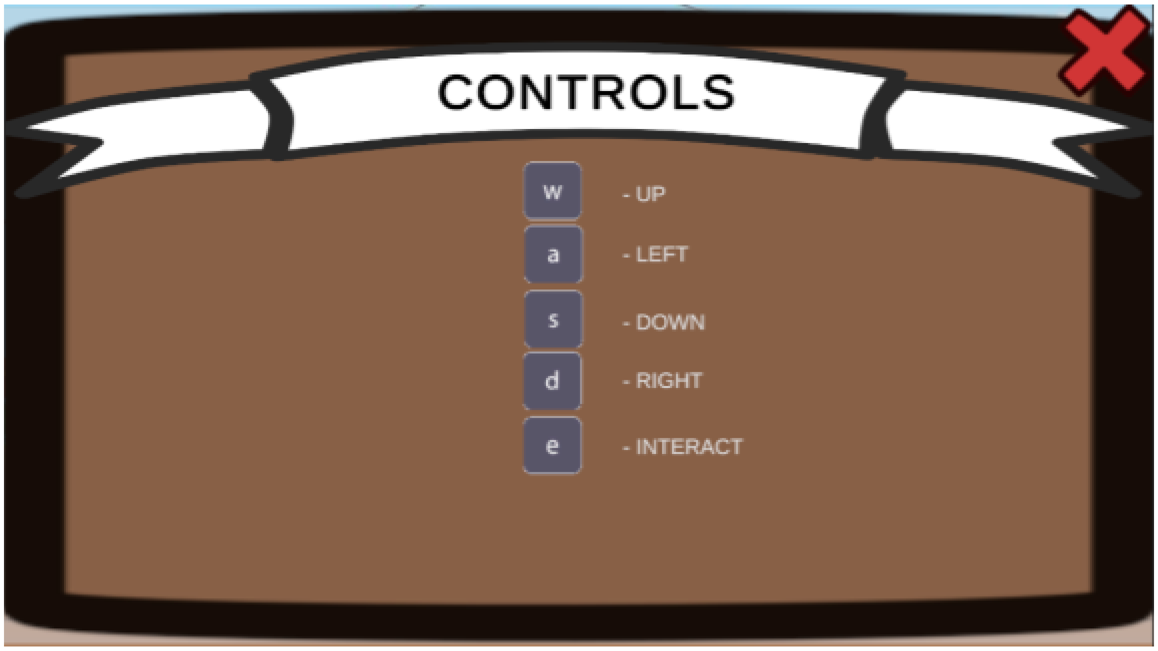


Figure 9: Control Panel

5.2.4 Below is the pause menu from in game by clicking the pause button in the bottom right corner of the screen to allow you to then either resume, go back to the main menu, or view the controls.

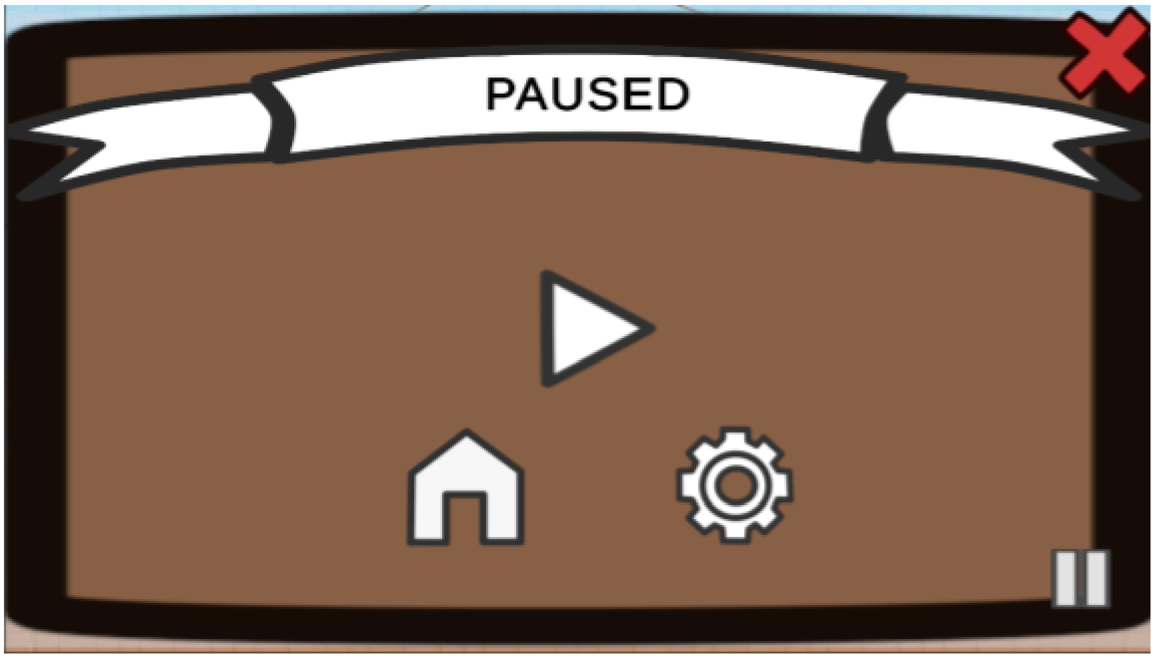


Figure 10: Pause Menu

5.2.5 After clicking on the Play button with the mouse, the player can interact with items by walking over them and pressing the interact key E on the keyboard.

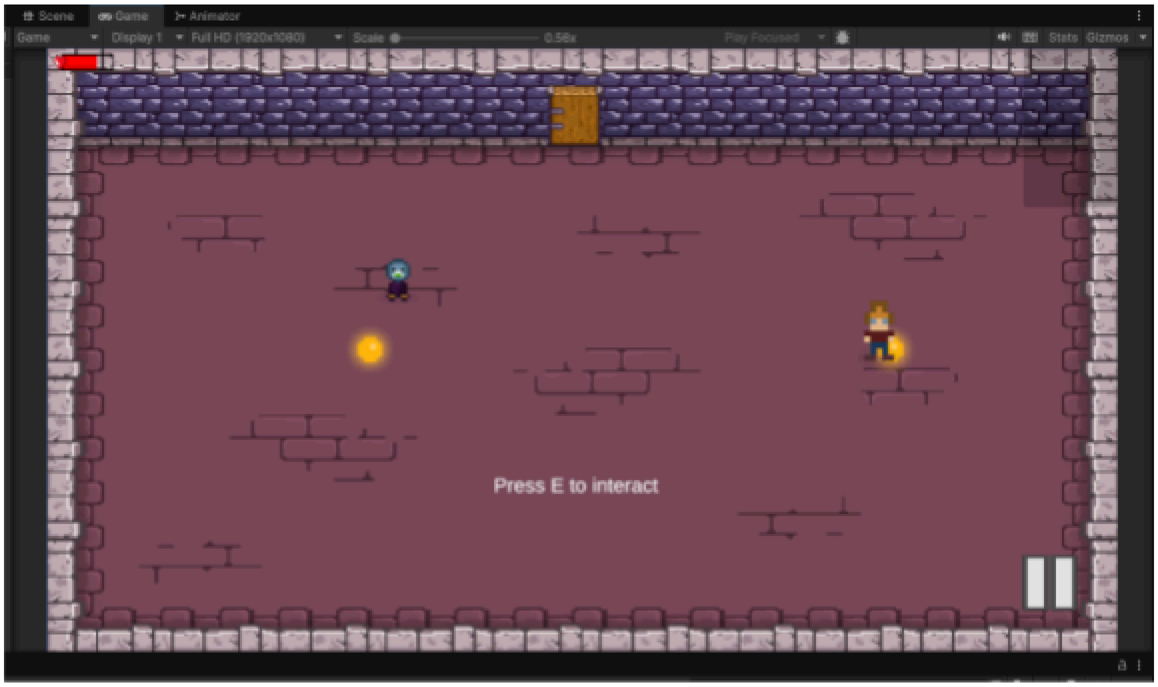
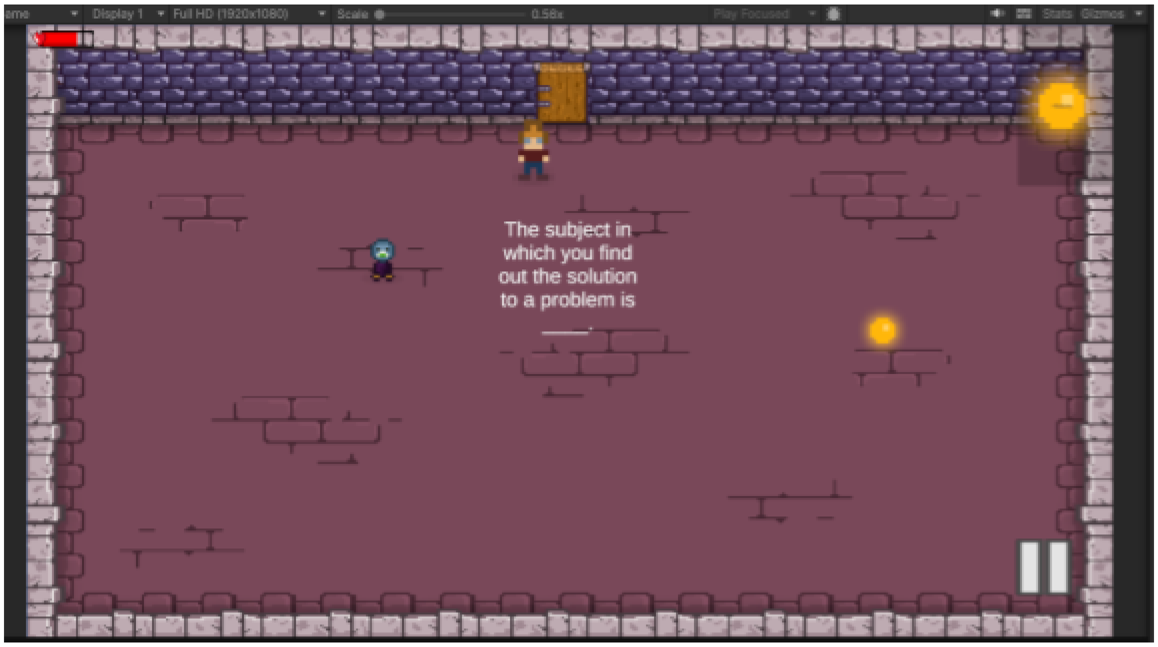


Figure 11: Player Interacting With Item

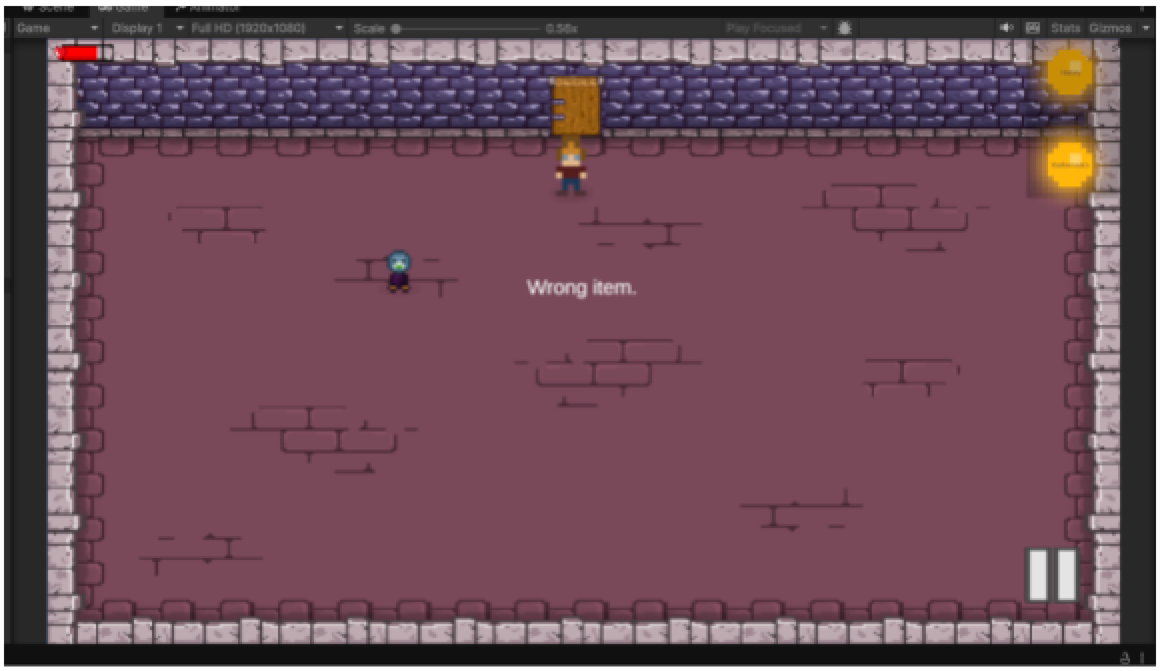
5.2.6 The player interacts with the locked door to start the question, the player must now select the items in their inventory in order to answer the question presented to them.



The subject in which you find out the solution to a problem is ____.

Figure 12: Player Interacting With Door

5.2.7 The player left-clicks on an item in the inventory that doesn’t answer the question. If correctly guessed, the player will move onto the next level. The player must do all of this while avoiding the enemy. If correctly chosen, it loads the next level.



Wrong item.

Figure 13: Player Using Item On Door

6 References

Website: Grammar Grable

[1] Unity Technologies, Unity Game Engine, Version Unity 6. [Online]. Available: https://unity.com/products/unity-engine

[2] Background: https://www.freepik.com/free-vector/empty-classroom-interior-with-chalkboard_16845438.htm#fromView=search&page=1&position=0&uuid=12e28d48-307e-47c3-892d-463516ef9af0&query=school+Cartoon+background

[3] Pencil: https://pngtree.com/element/down?id=NDM4NjAxNg==&type=1&time=1762981899&token=ZGY5ZDlhNDdlZDRmNmFhMzhhYmYxYWRjNDA5Mjc1ZmM=&t=0

[4] Keyboard: https://pngtree.com/freepng/full-keyboard-key_6569504.html

[5] Prototype: https://verzatiledev.itch.io/ui-prototype-base/download/eyJleHBpcmVzIjoxNzYzNDAzMzkwLCJpZCI6MjEyODQyNn0%3d.JygM3qBIq%2bZsiKnnMa9hvdGkOuQ%3d

[6] Our Game: https://andrewbrandao88.github.io/GrammarGrable/

[7] Free Asset pack used: https://assetstore.unity.com/packages/2d/free-2d-mega-pack-177430

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.