Design Document

Version 1.1 – 2023.11.06

Created 2023.10.27

Sunset Abyss

Stephan Scilenyj

Simon Hunter

Shunqi Wang

Ryan Yao

GitLab Repository:

https://mcsscm.utm.utoronto.ca/csc207_20239/group_14

Section 1: Table of Contents

Section 2: Project Identification

- What are we doing?

- Why are we doing it?

Section 3: User Stories

- **Owner:** Stephan
  - Name: Sound Emitter
  - Name: KeyBinds

- **Owner:** Simon
  - Name: Mini-Map
  - Name: Highlight / Border over Button

- **Owner:** Shunqi
  - Name: Health System
  - Name: Battle System

- **Owner:** Ryan
  - Name: TTS (Text - To - Speech)
  - Name: ALT (Mouse to Keyboard Mode

Section 4: Software Design

- **User Story:** KeyBinds
  - Design Pattern: Observer Design

- **User Story:** Mini-Map
  - Design Pattern: Strategy Design

- **User Story:** Battle System
  - Design Pattern: Factory Design

- **User Story:** TTS (Text - To - Speech)
  - Design Pattern: Observer Design

Section 2: Project Identification

- We are building off of the foundation set in assignment 2. This being said, we are implementing new features that will improve the gameplay, sound, and accessibility to all players.

- We are creating a user-friendly experience that accommodates many accessibilities needs such as visual impairment, hearing impairment, and color blindness in order to appeal to a larger variety of audience.

Section 3: USER STORIES

Owner: Stephan Scilenyj

Sound Emitter

- **ID**: 1.1

- **Description:**

  As a developer, I want an easy way to play, kill, and manage sounds.

- **Acceptance Criteria:**
  - Sounds: Should be easily playable and killable.
  - Sounds: Multiple sounds should be able to play at the same time

Implementation details:

- SoundEmitter class:
  - Should be located in the AdventureGameView class.
  - Each played sound has a soundtype string identifier that allows it to be individually stopped.

Priority: 2

Effort: 1/10

Keybinds

- **ID:** 1.1

- **Description:**
  As a visually impaired player, I want to use static and easy-to-remember keyboard binds to play the game without needing to rely on the mouse or visual feedback.

- **Acceptance Criteria:**
  - Menus: Should be navigable using the keyboard only.
  - Rooms: Should be traversable using the keyboard only.
  - Keys: Should have an audible indication that an input was received and if the input has done something.
  - Keys: There should be an easy-to-access audible description of the available keys and what they do.
- **Implementation details:**
  - KeyDescriber class:
    - Stores string descriptions of keys/mode.
    - Plays a text-to-speech description of the requested keys/mode when describeKey is called.

  - KeyListener class:
    - Has key modes that allow switching between sets of keys.
    - Stores a map that maps modes to another map that maps keys to functions.
    - When a key is pressed, it checks for functions in the current mode then defaults to the null mode.
    - onAnyKeyPressed is called with a bool determining if the key was bound.
    - onKeyPressed is added as an EventListener in the onKeyPressed method in the provided scene.
  - Individual key classes that manage the keylistener for a particular scene:
    - GameViewKeyListener:
      - Sets some keybinds for the AdventureGameView and their functions:

        - ALT: Toggles alt mode
        - Tab: Reset node focus to the current scene's node
        - Space: Accept
        - Enter: Accept
        - Q: Drop menu
        - E: Pickup menu
        - WSAD/Arrow Keys: Used for movement between rooms
        - (ALT mode) WSAD/Arrow Keys: Used for menu navigation
        - H: Say possible keybinds using text to speech
        - P: Say current health
    - Creates a KeyDescriber and uses it to describe the keys when H is pressed.
    - Uses onAnyKeyPressed to make the SoundEmitter on the AdventureGameView class play an invalid button sound when a non-bound key is pressed.
- **Priority:** 2
- **Effort:** 5/10

Owner: Simon Hunter

Mini-Map

ID: 1.3

Description: As a player of the game, I want to clearly see the directions I can or cannot go to so that I can make informed decisions about my movements and navigate the game world effectively.

Acceptance Criteria:

- Given that I'm a visually impaired user,
- When I enter a room,
- Then I want a visual indicator of the directions I can go
- And selecting these buttons will move me into its directed room

Implementation Details

Overview

A Mini-Map will be implemented to provide visual indicators for navigation within the game. This will consist of a box with a black background and four directional objects that change in shape and color based on the room's state.

Class: MiniMap

- Implements image buttons for directions: North, South, East, West.
- Holds these buttons within a GridPane.
- Constructor to create buttons and set up the map based on the current room.
- updateMap() method to adjust the visibility of buttons corresponding to available passages.

Class: ImageButton

- Img: An image representing the button.
- View: An ImageView for displaying the image.
- Direction: A MoveDirection used to assign a direction to the button.
- Constructor to set the image and the rotation based on direction.

Class: MoveDirection (Interface)

- Interface for directional movement buttons (North, South, East, West).
- movePlayer() to move the player in the associated direction.

New Methods for AdventureGameView

- initMiniMap(): Initializes the MiniMap in the game view and places it in the correct grid pane section.
- **Priority:** 2
- **Effort:** 4/10

Highlight / Border Over Button

ID: 1.4

Description: As a player of the game, I want to distinctly see the button I am hovering over with my mouse or have selected with my keyboard, so that I can easily identity my current selection and avoid making mistakes.

Acceptance Criteria:

- Given that I'm a hearing impaired user,
- When I hover over button with my mouse
- Or have the button selected with my keyboard,
- Then the button will illuminate a coloured border around the selected button
- And will make the button raise visually

Implementation Details

- Add a 2px wide border that is solid white over the button that the player is currently hovering over with mouse or has tabbed to.
- The buttons that are not selected will not be highlighted nor have a border at all. The button that has been selected will use the setStyle method in order to set the button properties.

Priority: 3

Effort: 4/10

Owner: Shunqi Wang

Health System

ID: 1.5

Description:
As a player who is preparing to enter into a battle or deciding whether to retreat from a battle, I want to see my current health clearly at the left top of the screen, so that I can make a wise decision of whether I should enter or continue the battle according to my health.

Acceptance Criteria:
Given that I am a player who is making decisions in a battle,
when I look at the top left of the screen,
then the screens display my health in the game clearly

Implementation Details:
Add an integer attribute health to the Player class to keep track with the player's current health.
Add a Vbox to the top left of the screen. This Vbox will display:
Your current health is: player.health
The Player class has a method called updateHealth which updates the player's health according to the input value and the method will also update the health displayed in the Vbox.

Priority: 3
Effort: 4/10

Health System ID: 2.2

Description:

As a developer, I want to add health potions into some rooms so when the player takes these health potions from the room to their inventory, they can drink the health potion(by simply clicking the health potion image) and recover some amount of health.

Acceptance Criteria:

Given that a player puts a health potion from room to his inventory,
when the player clicks the health potion in the inventory,
then the player's health will increase by a certain amount and the health potion will disappear from his inventory.

Implementation Details:

Implement the key event for clicking the key P, the event handler should call the java voice reader to speak out about the player’s current health.

Priority: 3
Effort: 2/10

Health System ID: 2.3

Description:

As a visually impaired player, I need the game to tell me my health in game by voice after I have pressed the corresponding keybind, so that I have the same information about my current health as a sighted person.

Acceptance Criteria:

Given that I am a blind player,
when I press the boardkey H,
then my health in the game will be displayed by a clear voice.

Implementation Details:

Implement the key event for clicking the key P, the event handler should call the java voice reader to speak out about the player's current health.

Priority: 2
Effort: 1/10

Fighting System

ID: 1.7

Description:

As a developer, I want to generate different kinds of enemies so that I can put enemies into some rooms for the player to fight.

Acceptance Criteria:

Given that the game starts,
the game should generate some enemies and randomly assign these enemies into a few rooms,
then each room can only exist 1 enemy at most.

Implementation Details:

Create an abstract class called enemy.

This abstract class implements an interface called attack since all the enemy should be able to perform attack

The enemy should have an..

- int attribute damageAmount,
- String attribute enemyDescription,
- An int attribute called enemyHealth.
- An Image attribute that contains an image to represent the enemy.
- A Room attribute called enemyRoom that stores which room is this enemy in.

Create and implement a few classes that inherit from the abstract class enemy.

Create a class called EnemyManager.

- Attributes: enemyList, which is a list containing all the enemies.
- Methods:
    - GenerateEnemy: This will only be called at the start of the game. randomly generate some enemies from the enemy classes we have implemented. Note the number of enemies must be smaller than the number of rooms and each room attribute of the enemies must be unique. Also, the enemy should not be assigned to the first room and the forced room.

Priority: 3

Effort: 9/10

Fighting System   ID: 1.3

Description:

As a developer, I want the room that has an enemy inside to be “blocked” to the player, so players have to defeat this enemy before they can actually visit this room.

Acceptance Criteria:

Given that the player enters into a room where an enemy exists,
when there are only two things the player can do: fight with the enemy or retreat.
then Set the room’s status to be “blocked” so the room's description will not be displayed.

Implementation Details:

- Modify the getDescription and the getImage methods in the Room class. If the Enemy attribute of the Room is not None, then the getDescription and the getImage methods of the room will return the description and the image of the enemy. Thus in the screen we will display the information of the enemy.

- Set the items in the room to be inaccessible to the player.

- After room view loading, if room.enemy is not empty, add two buttons to the screen: one button is called the “attack” and another button is called “retreat”.

Priority: 3
Effort: 2/10

Fighting System **ID:** 2.4

Description:

As a player entering into a room that has an enemy inside, I want the screen to display a detailed description and all information of that enemy, so I can know that I encounter a battle and the information of the enemy in battle.

Acceptance Criteria:

When the player enters a room with an enemy, the place that displays the description of the room will be replaced by the description of the battle, including the information of the enemy.

Implementation Details:

Some features of this story are covered in the implementation of the last story(story with ID 1.3).

- After room view loading(after the code segment `updateScene(output); in line290 in a2, adventureGameViewClass`), if room.enemy is not empty,create a text bar in the screen and this text bar will display:
- Enemy’s Health: Room.getEnemy.getEnemyHealth().
- This text bar is always required to be updated since the enemy’s health keeps changing during the fight.

- After room view loading, if room.enemy is not empty, create another text bar that will describe the battle information. (Ex: enemy performed an attack, this attack causes a damage of 10 and your health decreases to 50, now it’s your turn).

- After these have been done, initialize a fightManger instance and use this instance to call the method player Turn inside the fightManger class. (This class will be implemented in next story)

Priority: 3

Effort: 1/10

Fighting System **ID:** 3.4.1

Description:

As a developer, when the player enters a room that has an enemy, I want the player to enter into a turn based game with the enemy so the battle can start.

Acceptance Criteria:

When a player enters a room with an enemy,
Then the game will enter into a turn based fighting state, and the first turn will always give to the player.

Implementation Details:

Create a class called FightManager.

- This class takes two attributes, an Enemy instance and a player instance to initialize. When a fight starts, we will initialize an FightManager instance and pass the enemy in the room and the player in the game as the constructor method’s parameter.

Priority: 3
Effort: 1/10

Fighting System   ID: 3.4.2

Description:

As a developer, when the player is staying in a room that has an enemy, if it's the enemy's turn, I want the enemy to perform an attack on the player then the turn changes to the player so the fighting can continue.

Acceptance Criteria:

Given that a player is staying in a room with the enemy
When it's the enemy's turn,
Then the player will be attacked by the enemy and afterwards change to the player's turn.

Implementation Details:

- Add a method called enemyTurn to the class FightManager.

Inside the method enemyTurn():

- use the attack interface to perform an enemy's attack action on the player.
- Print out the information of this attack in the text bar on screen.
- Determine if the player's health is above 0 or below than 0
- If it's below 0, then print a noticeable message on the screen telling the player you are dead, then end the whole game.
- If it's above 0, then call the method playerTurn which will be implemented in the next story.

Priority: 3
Effort: 3/10

Fighting System **ID:** 3.4.3

Description:

As a player, when I am staying in a room that has an enemy and it’s my turn, I want the game to show the options that I can do by the buttons on the screen, these options should include attacking the enemy or retreat, so I can make decisions simply by clicking the buttons on the screen.

Acceptance Criteria:

Given that a player is staying in a room with the enemy
When it's the player's turn,
Then the player will see two buttons displayed on screen: attack button and the retreat button.

When Player presses the retreat button
Then it ends the battle and sets him to the last room he stayed.

When Player presses the attack button
Then it will perform one attack on the enemy and change the turn to the enemy’s turn.

Implementation Details:

- Add a method called playerTurn to the FightManager, this method will implement the event handle of the two buttons: attack button and the retreat button.

- Inside the method playerTurn:
    - Implement the event handler of the retreat button so after it’s pressed,
        - After a few seconds, remove the buttons and the text bar we added for battle.
        - The game will load the last room the player stays at, then we end the method.
    - Implement the event handler of the attack button so after it’s pressed, decreases the enemy’s health by the amount of player’s damage.

    - After the attack button is pressed, determine if the enemy’s health is above or below than 0.
    - If it’s above than 0, then call the method enemyTurn so the game can continue.
    - If it’s below 0, then print a message in the text bar that you defeat the enemy.
        - After a few seconds, remove the buttons and the text bar we added for battle.
        - Afterwards, remove the enemy in the room. (Using player.getCurrRoom.setEnemy = None). Then reload the current room and now this room should be accessible to the player.

Priority: 3

Effort: 5/10

Fighting System **ID:** 4.5

Description:

As a visually impaired person, I want a way to play this game using only keyboards, or I can play another alternative accessible game to defeat the enemy, so that I can go through the whole process of the game as a sighted person

Acceptance Criteria:

Given that I am a visually impaired person,
when I enter a room with an enemy,
then the game should use a clear voice to tell me the description of this battle.

And when it’s my turn,
then the game should use voice to tell me what actions I can do,
and the corresponding kinbinds of these actions.

Implementation Details:

Use key events to substitute the buttons.

Priority: 3
Effort: 8/10

Owner: Ryan

TTS (Text - To - Speech) Menu Options

ID: 2.7

Description:

- As a visually impaired player/auditory learner, I want to listen to the options available in the menu when the mouse hovers over the button or when the buttons highlighted change via keybind option.

Acceptance Criteria:

- Given the game is controlled via the mouse,
- When the mouse pointer is hovering over a highlighted button,
- Then there must be a sound emitting what is written on the button.

- When the duration of the speech is longer than the duration of the mouse hovering over the button,
- Then the speech will only stay up to the duration of the latter.

- When controlled via keybinds,
- Then the mouse pointer must vanish and one of the buttons must be highlighted to indicate which button the player is on.

- When the duration of the speech is longer than the duration of the highlighted button,
- Then the speech must be cut off as soon as the highlighted button changes

Implementation Details:

- TextToSpeechManager Class:
  - The main controller for functionality, like managing its engine, speech playback and also methods for button hover or keybinds

- Button Class:
    - The class that will create the buttons for the main menu, and also contain methods that will store attributes for the button’s text and handling mouse hover or keybind events
- KeybindManager Class:
    - This class will specifically handle key press events and trigger appropriate actions like TTS
- Menu Class:
    - Represents the game menu as a whole. Stores collection of buttons and handles interactions with the menu
- TTSProvider Interface:
    - The interface that defines methods for TTS Operations, and also classes that wrap TTS Library or API will implement this interface.
- FreeTTS Class implements TTSProvider:
    - The class that will contain the synthesizer for the TTS Library.
- Accessibility Class:
    - Allows users to customize TTS settings

- **Priority:** 2
- **Effort:** 4/10

ALT (Button to Mouse mode)

ID: 1.9

Description:

- As a player who is scrolling through the menu options, I want to have both keybinds and the mouse available to me so that I can have the freedom of choosing the options either way(via keybinds up, down, left, right or the mouse hover).

Acceptance Criteria:

- Given the player is in the main menu,
- When the mouse mode is activated
- Then the mouse pointer should be at the location it was at before entering the game.

- When no buttons are highlighted,
- And the mouse pointer is still on screen,
- Then alt is not necessary for mouse control.
- And it is directly linked to the TTS feature, it should emit a sound that says the text of the hovered button.

- Given the mouse pointer,
- When one of the valid keybinds is pressed,
- Then the mouse pointer should disappear,
- And the highlighted button should change in the direction of which keybind is pressed.

- When no buttons are highlighted,
- Then only the first button will be highlighted.
- Given the keybind option,
- When a long pressing alt
- Then the mouse pointer to reappear on whichever position it was on before the pointer disappeared.

Implementation Details:

- InputMode ENUM
    - Will contain two values, MOUSE and KEYBIND that will represent the current input mode

- Class AltManager
    - Contains attributes currentInputMode and Point mousePosition
    - Void toggleInputMode: switch whatever mode it is on rn
    - Void switchTokeybindMode: switch from mouse to keybind
    - Void switchToMouseMode: switch from keybind to mouse
    - Void handleKeybeingPressed(String key): if current mode is keybind, then move to the button in the direction of the key, else switch to keybind mode then do the same thing
    - Void handleLongPressAlt: if current mode is keybind switch back to mouse

- Class Menu
    - Contains attributes altManager and list of buttons
    - Menu method(AltManager AltManager): initializes menu with the altmanager
    - Void toggleInputMode: calls the toggleInputMode from altManager

- **Priority:** 3
- **Effort:** 4/10

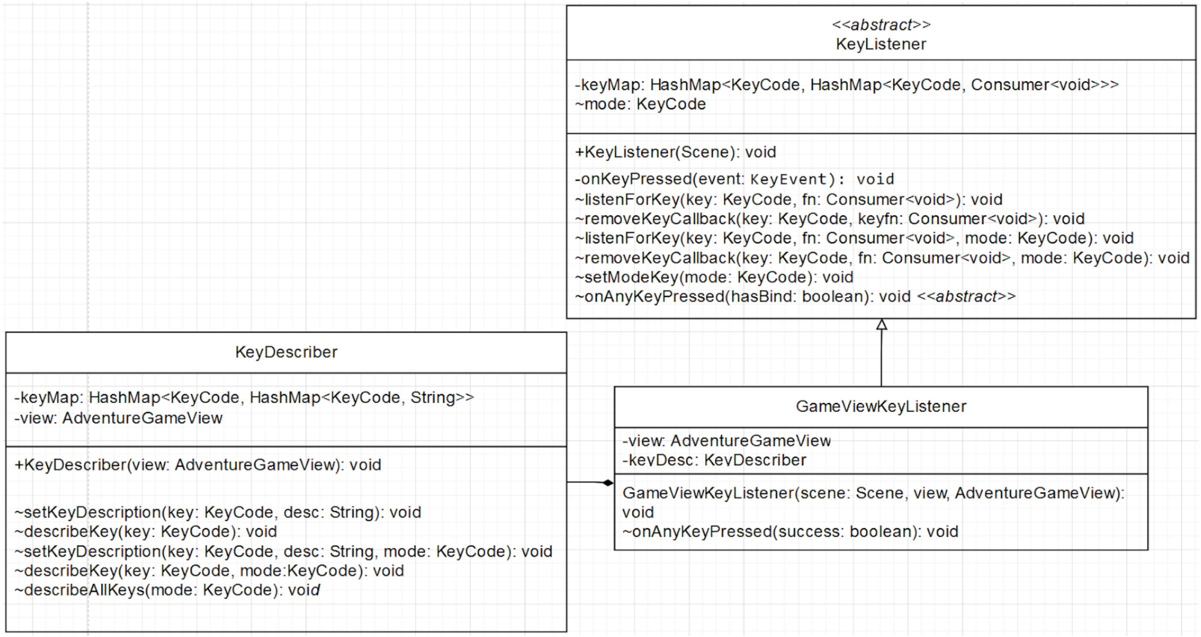
Section 4: SOFTWARE DESIGN

User Story: KeyBinds

Design Pattern #1: **Observer Pattern**

Overview: This pattern will be used to implement keybind functionality.

UML Diagram:



<<abstract>>
KeyListener
-keyMap: HashMap<KeyCode, HashMap<KeyCode, Consumer<void>>>
~mode: KeyCode
+KeyListener(Scene): void
-onKeyPressed(event: KeyEvent): void
~listenForKey(key: KeyCode, fn: Consumer<void>): void
~removeKeyCallback(key: KeyCode, keyfn: Consumer<void>): void
~listenForKey(key: KeyCode, fn: Consumer<void>, mode: KeyCode): void
~removeKeyCallback(key: KeyCode, fn: Consumer<void>, mode: KeyCode): void
~setModeKey(mode: KeyCode): void
~onAnyKeyPressed(hasBind: boolean): void <<abstract>>
KeyDescriber
-keyMap: HashMap<KeyCode, HashMap<KeyCode, String>>
-view: AdventureGameView
+KeyDescriber(view: AdventureGameView): void
~setKeyDescription(key: KeyCode, desc: String): void
~describeKey(key: KeyCode): void
~setKeyDescription(key: KeyCode, desc: String, mode: KeyCode): void
~describeKey(key: KeyCode, mode:KeyCode): void
~describeAllKeys(mode: KeyCode): void
GameViewKeyListener
-view: AdventureGameView
-keyDesc: KeyDescriber
GameViewKeyListener(scene: Scene, view, AdventureGameView): void
~onAnyKeyPressed(success: boolean): void

Implementation Details: The UML diagram outlines these main components:

- The abstract KeyListener class, which will contain all needed functionality for mapping a key to a function and managing different key modes.
- The GameViewKeyListener class, which will manage keybinds for the GameView scene and implement the related functions and accessibility articulation.
- The *KeyDescriber, which will manage storing descriptions of what a keybind does and articulating them using text to speech technology.*

The onKeyPressed method of KeyListener will be used to handle keyevents and call the correct function from the keyMap attribute if applicable. It will also toggle/switch modes if a mode key is pressed.

The ListenForKey and RemoveKeyCallback methods of KeyListener will add and remove keybinds from the keyMap attribute respectively. If an invalid mode is given an Exception will be thrown. If a key set as a mode is given as a key an Exception will also be thrown.

The setModeKey method of KeyListener will allow setting a key as a mode toggle. This is done by mapping the mode in the keyMap attribute to a new hashmap to store keys and functions.

The onAnyKeyPressed method of GameViewKeyListener will play a default invalid sound using the soundemitter stored in the view attribute if the pressed button does not have a key bound to it.

The setKeyDescription method of KeyDescriber will add a key description to the keyMap attribute.

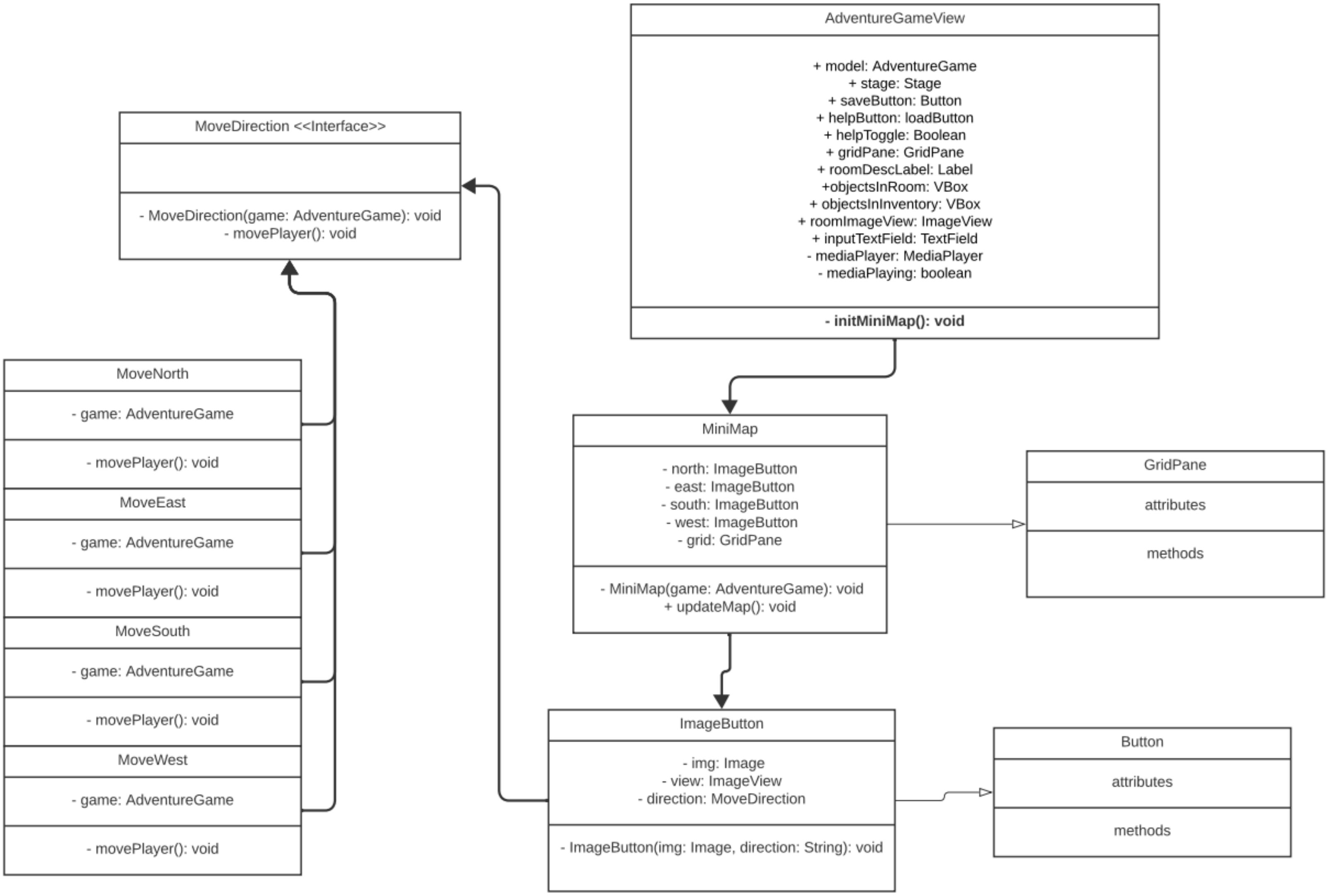
The describeKey and describeAllKeys methods of KeyDescriber will use text to speech accessed through the view attribute to articulate the text stored for the given key or all keys respectively.

User Story: Mini - Map

Design Pattern #2: Strategy Pattern

Overview: This UML class design will implement the mini-map when the game initializes.

UML Diagram:



Implementation Details:

Class: MiniMap

The MiniMap class is responsible for providing a visual representation of the player's current location and the available directions they can move in within the game. The class is depicted in the UML diagram and will interact with AdventureGame, ImageButton, and GridPane classes.

Attributes:

- north, east, south, west: ImageButton objects for each direction.
- grid: GridPane to hold the direction buttons.

Constructor:

- The constructor will take in an AdventureGame object in order to access the passageTable for the given room the player is currently in.

Methods:

- updateMap(): This method will update the visibility of each ImageButton depending on the available rooms the player can travel too.

Class: ImageButton

The ImageButton class is a custom button that contains both an image and a movement direction.

Attributes:

- img: The Image object representing the buttons arrow
- view: An ImageView that is used to display img.
- direction: A MoveDirection that indicates which direction this button is associated with. This will also alter the buttons rotation.

Constructor:

- ImageButton(Image img, String direction): img will be assigned to the buttons Imageview. A string called direction will be passed in and will alter decide the buttons rotation. For example, North will rotate the button 0 degrees, while South rotate it 180 degrees.

Interface: MoveDirection

This interface declares methods that must be implemented by classes representing directional movement.

Methods:

- movePlayer(): will move the player in a given direction based on the class it is implemented into.

New Methods in AdventureGameView

initMiniMap(): This method will be responsible for initializing the MiniMap instance within the AdventureGameView. It will create a new instance of MiniMap in the games primary Gridpane and place it accordingly. The AdventureGame will also be passed into the MiniMap in order to give it information about the rooms in the game.

Additional Implementation Details:

For MoveDirection Interface Implementers:

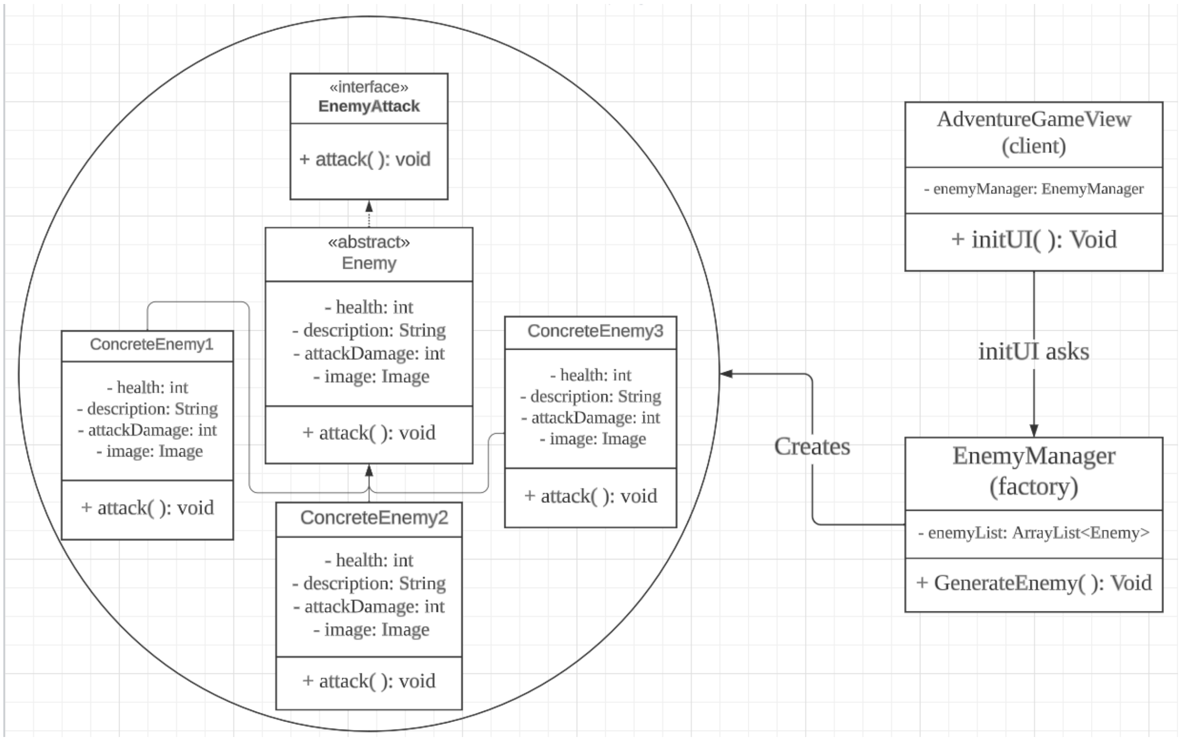
- Each directional class (MoveNorth, MoveEast, MoveSouth, MoveWest) that implements MoveDirection will contain the specific logic to update the player's position in the game world when the movePlayer method is called.

User Story: Battle System

Design pattern #3: Factory Pattern

Overview: This pattern will be used to generate enemies when the game starts.

UML diagram:



Implementation details:

Create an abstract class called enemy.

- This abstract class implements an interface called attack since all the enemy should be able to perform attack

The enemy should have an int attribute damageAmount,

- String attribute enemyDescription,
- An int attribute called enemyHealth.
- An Image attribute that contains an image to represent the enemy.
- A Room attribute called enemyRoom that stores which room is this enemy in.

Create and implement a few classes that inherit from the abstract class enemy.

- Create a class called EnemyManager.

Attributes:

- enemyList, which is a list containing all the enemies.

Methods:

- GenerateEnemy: This will only be called at the start of the game. randomly generate some enemies from the enemy classes we have implemented. Note the number of enemies must be smaller than the number of rooms and each room attribute of the enemies must be unique. Also, the enemy should not be assigned to the first room and the forced room.

- In addition, the Room class in the game has to add another attribute Enemy which will be assigned in the method GenerateEnemy.

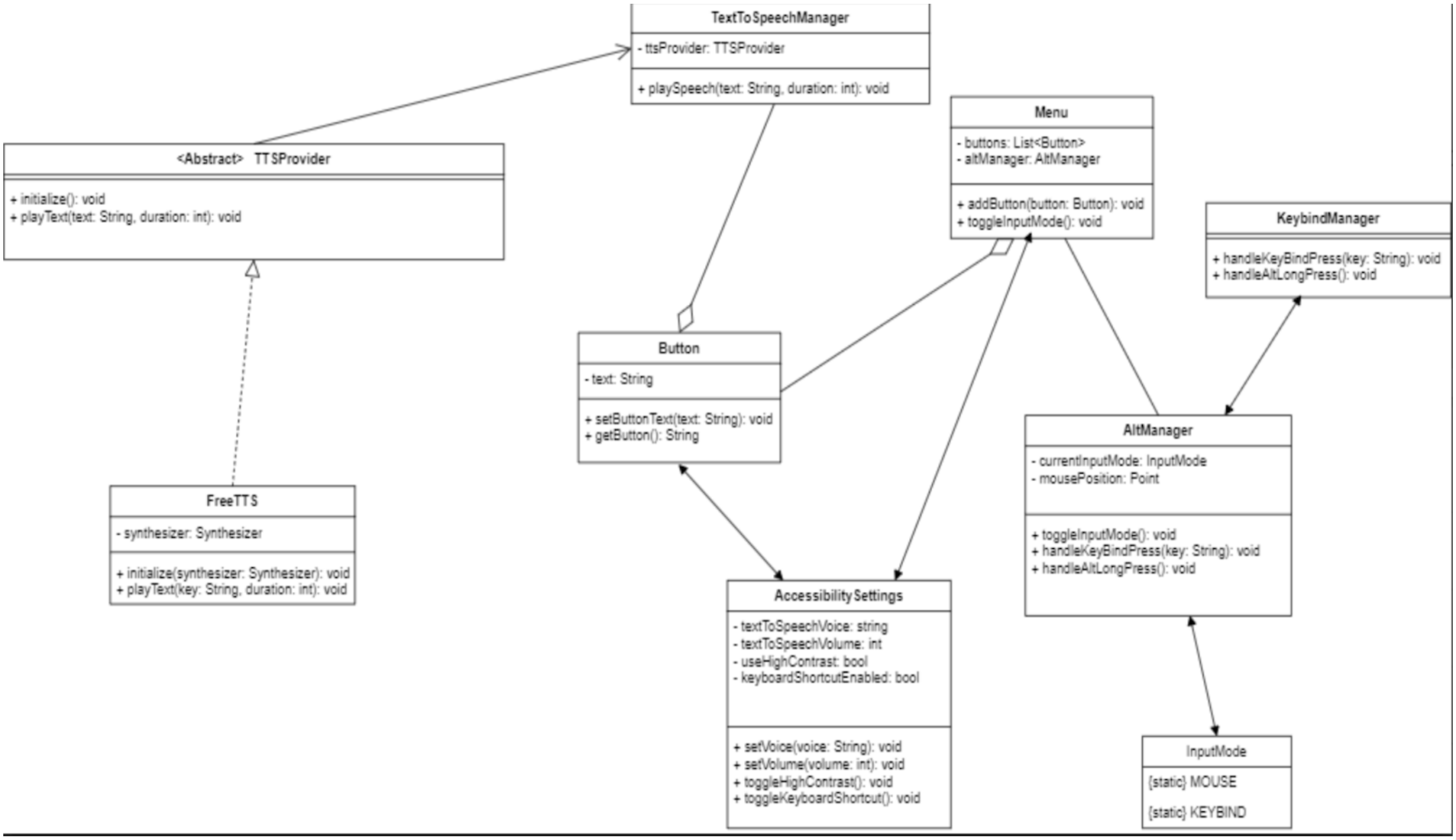
Add attribute EnemyManager to the main class AdventureGameView. During setting up the UI, initialize the EnemyManager instance and use the GenerateEnemy method to create enemies for the game.

User Story: TTS (Text - To - Speech)

Design pattern #4: Observer Pattern

Overview: This pattern will be used to generate a TTS system when the game loads.

UML diagram:


TextToSpeechManager
- ttsProvider: TTSProvider
+ playSpeech(text: String, duration: int): void
<Abstract> TTSProvider
+ initialize(): void
+ playText(text: String, duration: int): void
FreeTTS
- synthesizer: Synthesizer
+ initialize(synthesizer: Synthesizer): void
+ playText(key: String, duration: int): void
Menu
- buttons: List<Button>
- altManager: AltManager
+ addButton(button: Button): void
+ toggleInputMode(): void
KeybindManager
+ handleKeyBindPress(key: String): void
+ handleAltLongPress(): void
Button
- text: String
+ setButtonText(text: String): void
+ getButton(): String
AltManager
- currentInputMode: InputMode
- mousePosition: Point
+ toggleInputMode(): void
+ handleKeyBindPress(key: String): void
+ handleAltLongPress(): void
AccessibilitySettings
- textToSpeechVoice: string
- textToSpeechVolume: int
- useHighContrast: bool
- keyboardShortcutEnabled: bool
+ setVoice(voice: String): void
+ setVolume(volume: int): void
+ toggleHighContrast(): void
+ toggleKeyboardShortcut(): void
InputMode
{static} MOUSE
{static} KEYBIND

Implementation details:

The following UML contains the following:

- Abstract Class TTSProvider: It will allow the FreeTTS class to implement it with the synthesizer attribute, as well as playing the text in the button for a set duration.

- Class TextToSpeechManager: This class primarily manages the functionality of the TTS speaker, containing the attribute ttsProvider of object TTSProvider from the TTSProvider class, and playing the actual speech when playSpeech method is called.

- Class Button, which forms an aggregation relationship with the previous TextToSpeechManager class, containing the attribute text which indicates the text in the button, and methods setButtonText, taking in the text and placing the text in the button. The getButton method returns a string that indicates the text within the button.

- The Class Menu, which forms an aggregation relationship with the class button and AltManager, while forming an association with Accessibility Settings. It contains a list of Button objects as well as an AltManager object, with the method addButton adding the list of button/buttons to the menu and method toggleInputMode which will toggle the method of selecting options.

- Class AltManager, which controls the toggling between mouse pointer and keybinds via InputMode. The mousePosition attribute also allows the mouse pointer to appear/disappear wherever the last position of the mouse pointer was using the method handleAltLongPress.

- Class KeybindManager, which controls the main functionality of handling the toggling of keybinds and mouse pointer.

- Class AccessibilitySettings, which contains 4 attributes which are directly linked to their methods, allowing accessibility features like enabling high contrast mode or setting the voice/volume of the TTS feature.