Design Document

Version 1.0 – 2023.11.06

Created 2023.10.31

Trick or Trap

Philippe Filiatreault
Bonnie Ma
Matthew Dean Micallef
Clara Joy Melanie Tan

GitLab Repository:

https://mcsscm.utm.utoronto.ca/csc207_20239/group_39

Demo Video:

SECTION 1: PROJECT IDENTIFICATION

The purpose of this project is for the members of Group 39 to gain experience with accessible horror game development in a group setting, as well as to attempt to create an equitable horror game experience for a variety of players.

The game will incorporate core functionalities from Assignments 1 and 2 like room traversal, player inventory, game saving, and screen reader support. These features will be extended through the use of new and nested rooms, new objects, and a new inventory GUI. For a more complex game, the game will include additional features like cutscenes, NPCs, voiced dialogue, resource management, complex puzzles, new GUI elements, timed events, a point system, and more, all based on a new plot. The game will enhance code from Assignments 1 and 2 to improve the player experience with features such as skipping through text, adjustable text scrolling speed, and attaching new features to keyboard hotkeys so that the game can be played solely through the keyboard if desired.

SECTION 2: USER STORIES

Name	ID	Owner	Description	Acceptance Criteria	Implementation Details	Priority	Effort
General GUI	1.1	Matthew	As a developer of games using Assignment 2's codebase, I want a new/changed GUI for all new and improved features, so that it looks and feels different from Assignment 2's GUI.	Given that I am a developer who wants a fresh new GUI, I expect: - The removal of a static GUI. - Dynamic GUI that changes according to game situations and scenes. - Displays for inventory, settings, dialogue, room descriptions, and game log, all in designated windows or sections	Game display handler class responds to various player-induced event triggers (e.g. settings button, inventory button/keyboard hotkey) or intentional game triggers (e.g. finished a cutscene) and configures GUI appropriately.	1	3
Display Room Images	1.2	Bonnie	As a player with an interest in playing and enjoying the game, I want an image of the room I am currently in to be displayed prominently on the screen so that I can visualise where I am and use my surroundings to make optimal in-game decisions.	Given that I am a player who wants room displays, I expect: - That every room has an image. - The room image to consistently reflect the player's current room. - The room image is to be displayed in the middle of the game screen, in a consistent spot for every room.	Images will be pre-loaded into a file to be accessed by game files during runtime. Image display will be configured using the JavaFX library. Each room will be assigned its own image. The display manager class will handle which images to display and when, according to data from the player location class.	1	1
Display Room Text	1.3	Philippe	As a player with an interest in playing and enjoying the game, I want a written description of the room I am currently in so that I can better understand my surroundings and be aware of potential actions I can take.	Given that I am a player who wants room text to be displayed, I expect: - Each room to have its own description. - Each room has its own actionable choices. - The text to fit within the text pane	Room descriptions will be loaded in from pre-written files and assigned to room objects. Display manager class will display appropriate room description texts once the player enters a room, based on info from the player location class.	1	1
Display Objects	1.4	Clara Joy	As a player with an interest in playing and enjoying the game, I want room images to convey which objects in the room are interactable, so that I can clearly see my options and better inform my in-game decisions.	Given that I am a player who wants to see room objects, I expect: - The room image to highlight interactable objects within the current room. - That objects can be displayed as an icon in my inventory. - The room display changes according to object interactions.	Objects can be manually highlighted in the room images and then used by the game files appropriately. Upon user interaction with an object (e.g. pick up and remove from room) room image can be replaced with another version that's identical except with that object missing. Object icons can be loaded into a file and used by the game to display images for objects in inventory (just like room images).	1	3

Display Icons	1.5	Matthew	As a player interested in playing and enjoying the game, I want objects in my inventory to be visually represented by icons so that I can quickly identify what they are and so that it is clear where to click to use/drop the item.	Given that I am a player who wants icons displayed, I expect: - Each icon to have its own image. - Each icon to be intractable. - Each object in my inventory is to be displayed as an icon.	Object images/icons can be pre-loaded into files prior to game startup to be used in-game. Button layout can be configured for the player to use/drop any item in inventory, similar to a2, except only when inventory is open/visible by choice of player	1	1
Respond to Keyboard	1.7	Phillippe	As a player with a keyboard, I want to be able to use it to click buttons, type input, and select GUI elements (“switch focus”) so that I can interact with the game in an easy and familiar manner.	Given that I am a player who uses a keyboard, I expect: - The text bar to accept my typed input accurately - The text bar to accept my typed input when the game requires typed input - The text bar to be visible and obvious - That the game can be fully playable using only a keyboard (bonus)	Will use JavaFX keyboard event handling and focus property to send input signals to relevant classes. Notable hotkeys may include “i” = open/close inventory, “enter/return” = select/mouse click, “tab” = switch focus, “s” = open/close settings	1	2
Cutscene GUI	1.8	Clara Joy	As a player who is interested in the plot when gaming, I want cutscenes so that I can experience backstory and exposition in a succinct manner.	Given that I want to be able to fully pay attention to the cut scene, I expect: - cutscenes to progress without any user input. - the user text bar to be gone/replaced during cutscenes. - for cutscenes to last less than 5 minutes. - For an automatic transition from cutscene to gameplay that lasts under 5 seconds	Can link cutscenes to scene objects, which are similar to room objects except they remove unnecessary parts of UI (e.g. text input bar, inventory button)	2	2
Cutscene Dialogue	1.9	Clara Joy	As a player who is interested in the story of the game, I want dialogue of the interacting characters so that I can have a more immersive experience.	Given that I want character dialogue in the game, I expect: - each character to have their own dialogue - dialogue to be related to the game's story	Use the implemented GUI for the characters to display the text for each character's dialogue. Create rooms for each dialogue.	2	1
New Room Images	1.10	Clara Joy	As a developer who is interested in the images of the game, I want new images to reflect the changes made to the rooms.txt file and the plot.	Given that I want new room images, I expect: - Each room to have its own image	Draw the new images and name them by their respective room number. Place the image files in the room-images directory of our game.	1	2

Dialogue GUI	2.1	Matthew	As a player who is interested in plot and game progression, I want character dialogue to be present in the game, so that the story and game mechanics can be explained in an organic way.	Given I want character dialogue to be displayed, I expect: - Each dialogue to correspond to the character's icon or the walkie-talkie. - Dialogue to be animated to create suspense.	Each scene containing dialogue can be structured as a queue to be played. Elements of the queue are dialogue objects that make calls to a dialogue method to character classes that contain the text to be displayed.	1	2
Summary	2.3	Bonnie	As a player who may forget some/many details as the game goes on, I want a summary feature, so that I can review everything that's happened when needed.	Given that I'm a player of the game who wants a summary tab, I expect: - Choices made populate the summary table as shorter descriptions (e.g. you chose to do x). - The table is expandable, with visibility being togglable - Only vertical scrolling with adjustable width, which adjusts the text length (for accessibility).	Add to the main GUI with a button that creates a pop-out window (like "save" and "load" from a2). One class to handle what goes in the summary table, fed by the "movePlayer" or "doAction" functions, and another for the GUI	2	3
Title screen	2.4	Philippe	As a player with an interest in playing and enjoying the game, I want to see a title screen for the game on launch, so that I can clearly see what game I am playing and can control how I start playing.	Given that I am a player who wants a title screen, I expect: - The name of the game to be displayed. - An option to start a new game. - An option to load a new game. - An option to access settings.	Can link the title screen to a scene, and have it play on launch, then force the player to first cutscene once the start button pressed - just like cutscene GUI, everything about title screen can be linked to scene object and displayed using display manager class	2	4
Load game	2.5	Philippe	As a player who can't play through the game in one sitting, I want to continue a game session I previously played so that I can continue from where I left off with no changes.	Given that I am a player who wants a load button, I expect: - A display of the three slots of saved games I can access. - A confirmation once a slot is chosen. - The current state of the game is changed to the saved state of the game.	This user story is to be implemented similarly to assignment 2. Game versions will be saved as serializable objects in a Saved folder. The GUI of the load game will be split into three slots, each with their name and current in-game time.	2	4
Save game	2.6	Philippe	As a player who can't play the entire game in one sitting, I want to be able to save my game easily so I can leave without losing progress.	Given that I am a player who wants a save button, I expect: - A display slots to save the current state of the game. - Be able to save the current game to a slot. - To give a name to the saved game.	This user story to be implemented similar to a2. Game versions will be saved as serializable objects in a Saved folder and loaded from there.	2	4

Character Headshots	2.10	Matthew	As a gamer who likes to be able to visual and personify characters in the games I play, I'd like to be able to see images of the characters when they speak.	Given that I am a player who likes to be able to visualize characters, I expect: - to be shown images of characters when they speak	Display a headshot of the character who is speaking when they have dialogue.	2	2
Animated Text	2.11	Clara Joy	As a player, I would like to see the text be animated when being displayed so that my game experience has more movement.	Given that I am a player who likes animated text, I expect: - The text to be animated when visiting a room. - The text to increase character by character.	Use a timeline with pauses to display the text so that players can use other aspects of the game while the text is being displayed.	2	3
Settings Menu	3.1	Matthew	As a player with distinct preferences or accessibility requirements, I would like to be able to toggle various settings of the game so that I can tailor the gameplay to suit myself.	Given that I am a player who wants a settings button, I expect: - The button to be accessible at all times, except cutscenes. - The option to change my accessibility settings. - The option to exit the game (optional: be prompted to save the game before exiting). - (Optional: A load button to load into a different saved file during the game.)	A window that opens when clicked, like the instructions tab in a2, and has options for text scroll speed toggle, and text colour/background colour. The observer pattern sends updates to relevant classes to enact changes.	1	3
Inventory GUI	3.2	Philippe	As a player who plays games for fun, I would like an inventory to keep track of the objects I've picked up so that I can easily keep track of the objects I have without spending too much energy.	Given that I am a player who wants an inventory, I expect: - My inventory to be accessible by toggling a button. - Ability to inspect objects and get a short description.	Use the iterator pattern to display all objects in inventory, window will pop up and replace the text pane when the player clicks the inventory button/presses the inventory hotkey	2	2
Text scrolling speed	3.4	Bonnie	As a player who reads slower/faster, I would like to be able to toggle the text-scrolling speed (e.g. the speed that the text in the text pane changes to a new block), so that I can read the game text at my own pace.	Given that I want to be given more/less time to read the text before it changes, I expect: - A setting in the settings menu that lets me change how many seconds text is displayed for (maybe just like 3 modes, slow, med, fast) - That I can change this speed at any point in the game (bonus) or maybe just once	Observer pattern that checks if settings manager class has been changed, and if so will send the updated settings information (e.g. integer speed for how many seconds each text block is displayed) to the displayer class	2	2

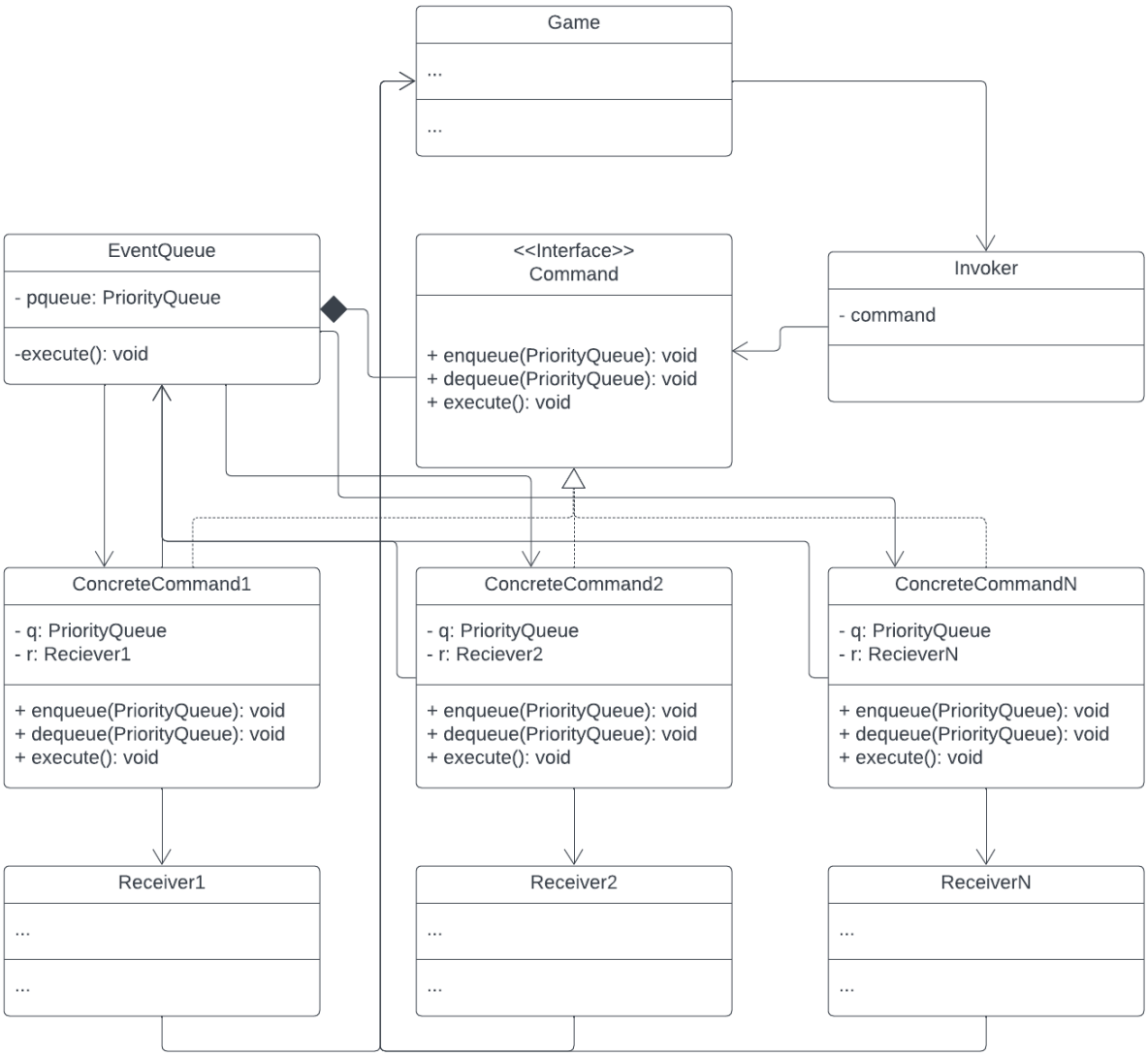
				at the beginning (maybe easier) - That my preferred settings stay put throughout entire rest of playthrough - That this text-scrolling speed has no effect on any part of the game besides the player comfort			
Keyboard controller	3.6	Matthew	As a user who doesn't have a touchpad or mouse, I would like to be able to play using only a keyboard, so that I can still experience the game to the fullest.	Given that I am a user without a touchpad, I expect: - certain keys can be used in substitution of mouse clicking - there should be a legend in the settings/instructions that will tell me what keys do what.	Use of JavaFX library to handle focus switching using the keyboard (e.g. tab key) and enter key to handle select, keyboard event handling input sent to relevant classes	3	2
Riddles	4.1	Matthew	As a player who enjoys logical thinking puzzles, I would like for there to be riddles and/or thinking puzzles so that I can progress through the game in an enjoyable manner.	Given that I am a player who enjoys logical thinking, I expect: - riddles that include answers to pick from - there should be consequences/rewards based on whether I properly figure out the riddle	Just like in assignments 1 and 2 where players can access rooms if they hold the correct key, the same concept is applied through riddles. The Player will have a Map attribute where the key is the clue name and the value is a boolean, true if the clue was completed, otherwise false. Users will have to associate clues together to finalise other Map keys to gain access to locked locations.	2	2
Skippable Text	4.5	Clara Joy	As a developer who must test the game many times, I would like to have the option to skip through the story text of my choosing, so that I don't have to waste time reading text I have already tested before.	Given that I am a developer who wants the ability to fast skip through text, I expect: - all the text of the current dialogue or description to be displayed immediately when the space key is pressed	Create a key event when a button is pressed, check if the key is space and check if there is a current dialogue. Toggle the text animation off and insert the whole text string into the text box to be displayed.	4	3
Audio replay	4.6	Matthew	As someone who has a harder time hearing, I sometimes miss audio; as such, I would like there to be a button that allows me to easily replay the latest audio	Given that I am a player who wants a button to repeat the most recent audio, I expect: - A button that repeats the most recent audio (alt) - The button to be labelled in the settings/instructions	Create a queue containing all recent audios, adding to the queue once the audio is played. Get the top value of the queue using a key event to listen to user button clicks	3	3

SECTION 3: SOFTWARE DESIGN

Design Pattern #1: Command Pattern

Overview: This pattern will be used to implement scheduled events (i.e. scheduling commands in a queue to activate background events such as dialogue or clock tolls)

UML Diagram:



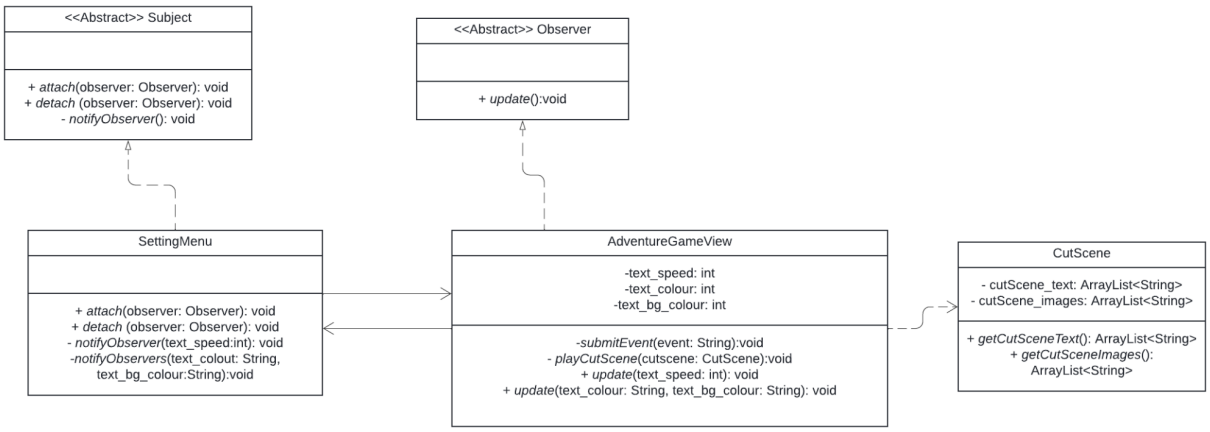
Implementation Details: The UML diagram outlines these main components:

- The *Command* interface, which includes three methods: execute, enqueue and dequeue
- A priority queue which will queue and execute events based on the time
- An invoker which will act to move the commands from the game to the queue
- The game which will invoke commands based on user input
- Various receivers which are defined roughly as the objects affected by the given command

The game instantiates commands and sends them through the invoker to be queued. In the command objects,

specifications about the receivers and necessary parameters are given. When the time comes, the command at the front of the queue will execute, dequeuing and executing as defined in the concrete command class. This execution will relate to the corresponding receiver. This design pattern will be used to schedule events for execution based on "time" determined by how many movements the player has made. As in, when a player moves, this will cause the "time" to increase. Certain events will be linked with "commands" (as in concrete commands) triggered by having reached a certain "time". For example, based on time we can have doors open or lock, clocks can toll and the player could be contacted via the "radio".

Design Pattern #2: Observer Pattern
Overview: This pattern will be used to fulfil load, save, text scrolling speed, and adjustable colour user stories.
UML Diagram: The UML diagram outlines these main components:

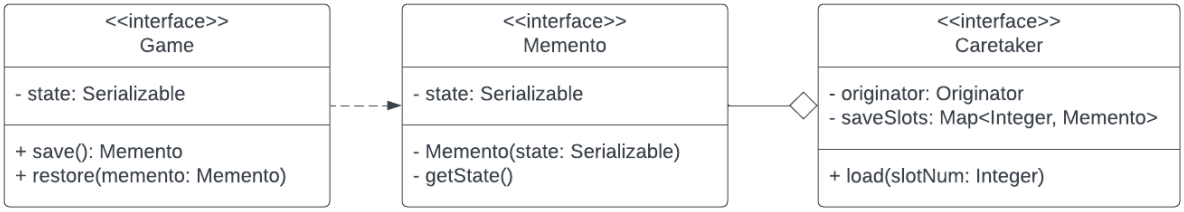


Implementation Details: The UML diagram outlines these main components:

- The *Observer* interface, which includes the abstract method *update()*.
- The *Subject* interface, which includes the abstract methods *attach(), detach(),* and *notifyObservers()*.
- The *AdventureGameView* class from a2, which handles displaying text and images on the GUI, and observes necessary changes from the *SettingMenu* class.
- The *SettingMenu* class, which serves as the observed object for the *AdventureGameView* class, informs *AdventureGameView* so that it can realise the changed settings.

The *SettingMenu* class will record changes made by the user when the setting menu is open on the GUI. Changes will then be sent to the observers (*AdventureGameView*) registered via the *attach* method, which stores all Observers in an ArrayList, via the appropriate *notifyObservers* method. This will then allow *AdventureGameView's* relevant *update* function to store the new value for future use. Every subsequent call to *submitEvent* (used for displaying room text, as per Assignment 2) and *playCutscene* (which will play cutscene text) will apply the updated versions of text_speed, text_colour, or text_bg_colour when rendering the GUI. In

Design Pattern #3: Memento Pattern
Overview: This pattern will be used to implement the save and load functionality.
UML Diagram:



Implementation Details: The UML diagram outlines these main components:

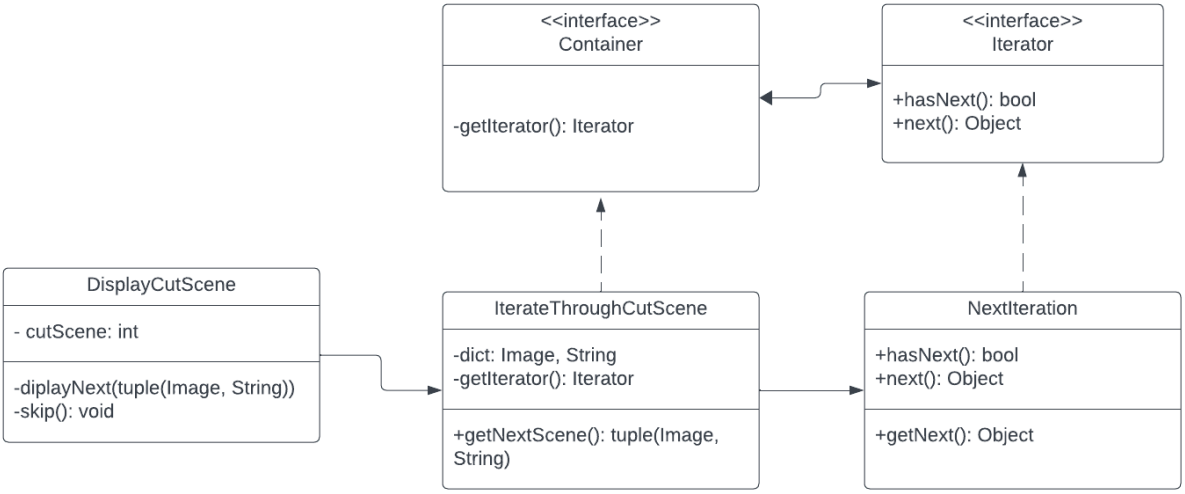
- The *Game* class, which is the originator or the target to be saved and loaded. Includes attribute *state* and the methods *save* and *restore*.
- A *Memento* class, whose objects would contain a saved state of the *Game* class.
- A *Caretaker* class, which holds three *Memento* objects and can change the current state of the *Game* to a saved one.

The *Game* class is the Originator, which requests a *Memento* object to capture its current state. The Originator then sends the *Memento* object to the *Caretaker* class to store it to be retrieved by the player at some later time. When the player needs to load a saved state, the Originator needs to restore its state to a previous point, being one of the three slots in the map attribute, by requesting the corresponding *Memento* from the *Caretaker* using the load method and the file number to be loaded.

Design Pattern #4: Iterator Pattern

Overview: This pattern is used to iterate through the cut scenes so that they can be displayed

UML Diagram:



Implementation Details:

The UML diagram outlines these main components:

- The *Iterator* interface is implemented by *NextIteration*. The *NextIteration* inherits both attributes of Iterator which includes the method *next*() which returns the next object in the iteration, and *hasNext*() which returns a boolean for if there is a following object. *NextIteration* has the method getNext() which returns the next object in the iteration. In correlation to our game, the getNext() will return to the next scene.

- The *Container* interface is implemented by *IterateThroughCutScene*. The *IterateThroughCutScene* has a dictionary that allows for knowing which image and string match up for each scene depending on the scene.
- *IterateThroughCutScene* also has the public method *getNextScene*() which returns a tuple of the following scene's image and text.

DisplayCutScene has the attribute cutScene which keeps track of which cut scene the game is on (the cut scene is not the same as the scene; there are multiple scenes per cutscene). This class also displays the following scene to the user. *DisplayCutScene* also has the skip method which will force a skip to the next scene.