Design Document

Version 1.1- 2023.11.03

Created 2023.10.20

Colossal Cave Adventure 2.0

Group 8: Daniel Sequeira, Dema Ismail, Eric Sun, Raiyan Rizwan

https://mcsscm.utm.utoronto.ca/csc207_20239/group_8

Section 1: Project Identification

The objective of this project is to improve upon the Colossal Cave adventure game. We aim to achieve this by enhancing the sensory depth of the game, making it more acoustically engaging and supportive of accessibility features for users with impaired vision. An enhanced graphical user interface will be constructed using JavaFX, where the user will be given more options to change display features like colour and sound features, such as adding auditory cues to make it easier and more entertaining and immersive to play the game. In addition, new interactive features will be added to make the game more challenging and add an additional layer of complexity to it. Design patterns will be employed to enhance the stability and maintainability of the project. The project will be managed using scrum/agile methodologies.

Section 2: User Stories

Name	ID	Owner	Description	Implementation Detail	Priority	Effort
Leaderboard	1.1	Raiyan	As a competitive adventure game user I want a leaderboard feature so that I can track my progress and compare my scores with friends.	As the game begins, a stopwatch feature will be implemented which will track the amount of time the player takes to complete the game. The time will be recorded on the leaderboard as the high score.	2	2
Companion Troll	1.2	Daniel	As an adventure game user, I want replayable elements in the game so that I have choices that affect the difficulty of my playthrough.	If a player is in a room with a Companion Troll there is a game that they can play where they have to convince the Troll to join them on their quest. If they successfully do that then the Troll follows them on their quest. As a result, they will find that the Companion Troll can allow them to enter more rooms. This will be implemented by instantiating a new AdventureObject in the player's inventory container object.	2	2
Tic Tac Toe Troll	1.3	Eric	As a user who craves for new and interesting challenges in any game, I want a troll who engages me in a game of Tic Tac Toe so that I can hon my ability to strategize.	In the event that the "Tic Tac Toe Troll" is in a room the user enters, the user is presented with a window with the 3x3 tic tac toe board. The user is prompted whether they or the troll should make the first move. The user and the troll would then place marks on empty spots on the board, in alternating order. For the user, this would be done using keyboard input. The player would win and be allowed to pass if they place three marks in a line or diagonally on the board.	3	2
Snake Game	1.4	Dema	As an arcade	If the player enters a room with a	4	2

Troll			enthusiast gaming user I want a troll that challenges me to play the classic snake game so that I can challenge my reflexes and relieve nostalgia of the past.	“Snake Game Troll”, the snake game is displayed using updatescene method. The user can utilize the arrow keys to collect food that when collected increases the length of the snake. This will be handled using an event-handling method. If the user hits the edge of the playing area or the snake wraps around itself the troll sends them back to a previous room. To win, the user must have a snake with a length of thirty. When the user wins the troll allows them to pass through. The Snake Game Troll implements a Troll interface and JavaFx will be utilized to implement the game window, snake, and food.		
Color Contrast	2.1	Dema	As a low vision screen user playing Colossal Cave Adventure 2.0, I want a game interface with high colour contrast so that I am able to differentiate and navigate the different interface elements like images, buttons, and text.	Add a combo box to represent the “colour options” button that provides a drop-down menu with colour schemes that cater to the different types of colour vision deficiencies so that when the user selects the colour scheme of their interest the background, images, text and buttons in the interface adjust using the updatescene and updateitem methods to the selected low vision friendly contrast scheme.	1	2
Magnification	2.2	Raiyan	As a low-vision user playing the game, I want to be able to change the font size of the text on the screen so that the text is more legible.	Add a combo box with a drop-down menu that includes varying font sizes that the user can change to.	2	2
Acoustic Immersion	2.3	Daniel	As a non-sighted user playing the	Room objects will have a container attribute that can contain	1	3

			game, I want to be able to have some sensory indication of the scene (atmosphere, background, mood) without the pictures.	different keywords that describe the scene. These will be processed if a Game folder has a "sounds.txt" file. Accordingly a roomAudio attribute will be given to the AdventureView class that holds an instance of a new RoomSounds class that will be updated in the updateScene() method. AdventureLoader will have an extra method added parseSounds() that populates the container attribute of every Room object that has sounds assigned to its room number.		
Button Sound Cue	2.4	Eric	As a visually impaired person interacting with the game, I want to be able to hear a description of which button I click when I hover on a button so that I can navigate the game more easily.	When the user hovers the mouse over a button, an audio recording of the button's description will begin to play.	2	3
Power Up feature	3.1	Eric	As an imaginative adventure game user I want to gain special abilities (like power-ups) in the game so that I can enhance my ability and be at an advantage during gameplay.	If a user defeats a troll a powerup is awarded, this is reflected by the "power up" button turning from grey to green when they next encounter a troll. When the user next encounters a troll they can use the power up to pass by the troll without needing to challenge them.	4	3
Sound Loading	3.2	Daniel	As a developer I want to change the way in which sound	Create a SoundManager Class that has a container attribute that holds Object instantiations of the loaded	2	3

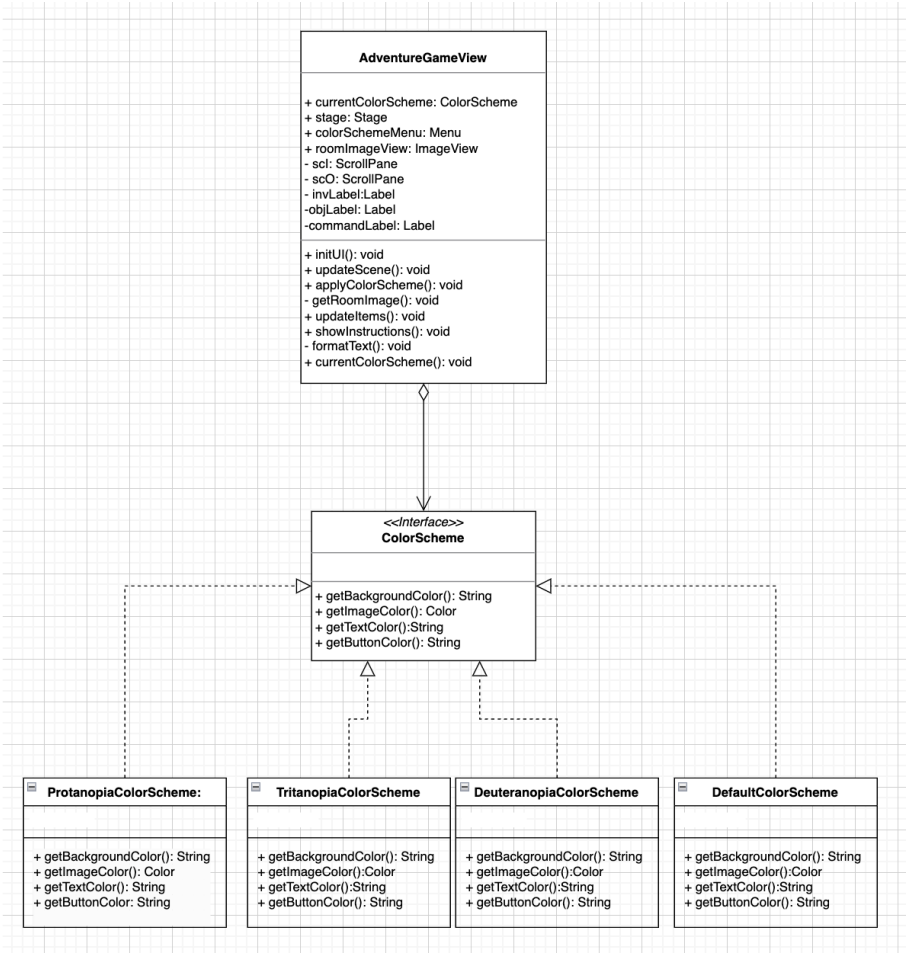
			files are loaded and played so that the game runs faster and more optimally.	sound files. This way when a certain sound needs to be played, it does not need to be loaded directly from memory again.		

Name	ID	Acceptance Criteria
Leaderboard	1.1	Given I am a user playing this game, I want to have an option to enter my name so that it is kept track off and saved in the game. Also I want my name along with my score, i.e. the amount of time taken to complete the game to be shown after I win the game. Subsequently, I want my score to be saved and shown when other players complete the game so that those other players can compete with my score and I with theirs.
Companion Troll	1.2	Given I am a user who navigates to a room with a Companion Troll, I am able to clearly understand the scenario being presented, I can choose to exit the interaction if I want, I can understand how to interact with the Companion Troll, I can gain the Companion Troll as an inventory object if I win the game, I can lose the game and not be able to pick up the Companion Troll. Given I am a user that has obtained a Companion Troll, I can decide to part with the Companion Troll by entering a command in the text prompt or clicking on a button in their inventory, After a Companion Troll is left in a room they can be picked up again by me, the player, into my inventory to rejoin my quest.
Tic Tac Toe Troll	1.3	Given I am a user in this adventure game, and a tic tac toe troll is blocking my path. I am presented with a 3x3 grid on the screen and a text input box. The box prompts whether I want to go first or if I want the troll to go first and explains the game rules. Either way, when it is my turn to play, I can type the coordinates of the box I want to fill, and the associated box in the grid would be filled with a symbol. The troll would then fill another box with a different symbol. If my input does not make sense, I am prompted to reenter the input. If I win by placing three straight or diagonal symbols in line on the grid, a celebratory message is displayed. If the game is tied or I lose, I would return to the previous room.
Snake Game Troll	1.4	Given I am an arcade enthusiast gaming user, when I wanna pass through a certain passageway a troll blocks my way, and then I expect the troll to challenge me to play the snake game to pass. I suddenly see the screen displaying a game grid, a snake, and food. When I press the arrow keys I expect the snake to move in the corresponding direction on the game grid. If I move the snake onto a cell with food, then the snake's length increases by one, and a new food item appears on a random grid cell. When I move the snake near the game boundary, the snake's head collides with the wall, and a game-over message is displayed. Also, when the snake has a body length greater than one, and the snake's head collides with any part of its body the game-over message is displayed. In either case, the troll laughs at me and sends me back to the previous room. I can then re-enter the troll room and challenge the troll again. When my snake’s length reaches thirty, the troll then

		allows me to pass to a new room.
Contrast	2.1	Given I am a low-vision screen user, and I come across features in an interface with low colour contrast, I then expect to navigate to the “Colour Options” button. When I click the button, a drop-down menu appears with a list of colour schemes: Regular Deuteranopia, Protanopia, and Tritanopia. When I select the colour scheme I prefer, the background, images, text and button options change to match the colour scheme I selected. Suddenly, the interface features have high contrast, easing the game navigation process.
Button Sound Cue	2.4	Given I am a user with vision loss, and I come across a button that I cannot easily read, I expect to be able to navigate to the button and hover on the button and immediately hear a description of the button.
Magnification	2.2	Given I am a low-vision user, and I come across text that I can not read clearly, I expect to be able to change the font size to my comfort.
Acoustic Immersion	2.3	Given I am a non-sighted user playing the game, whenever I enter a new room, I expect music appropriate to the current room to be played. I then hear sound cues that describe the image being displayed in the current room. If there is no “sounds.txt” file the game is still playable without any issues.
Power-Up feature	3.1	Given I am an imaginative adventure game user and I come across a troll and defeat them, I expect to gain a power-up so that when I click the power-up button when I next encounter a troll, I pass the troll without having to undergo any challenges.
Sound Loading	3.2	Given I am a developer when the game is initiated I expect that individual sound files, are only ever loaded once so that no references exist to the file locations of the sound files outside of the SoundManager class.

Section 3: Software Design

Design Pattern #1: Strategy
Overview: This pattern will be used to implement the color contrast functionality since we have multiple color schemes for the color contrast task which can be chosen at runtime.
UML Diagram:

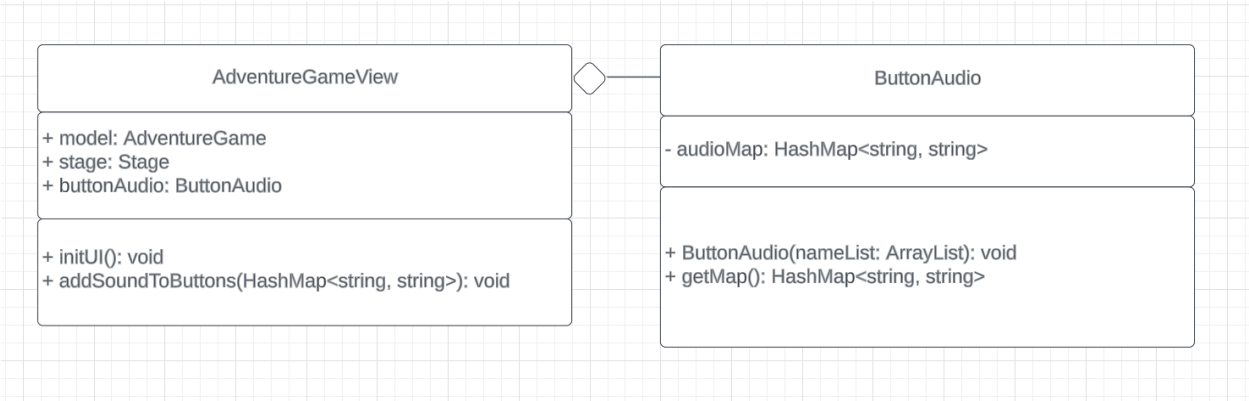


Implementation Details: The UML diagram includes the following components:

- The *ColorScheme* interface, which includes four public methods: getBackgroundColor, getImageColor and getTextColor, and getButtonColor.
- *DefaultColorScheme* is a concrete class that implements the *ColorScheme* interface. It is the default black-and-white color of the background, green buttons, images and white text.
- *DeuteranopiaColorScheme* is a concrete class that implements the *ColorScheme* interface. It sets the button, background, image and text to colors that support inability to visualize green light.
- *ProtanopiaColorScheme* is a concrete class that implements the *ColorScheme* interface. It sets the background, button, image and text to colors that support red light vision deficiency.

- *TritanopiaColorScheme* is a concrete class that implements the *ColorScheme* interface. It sets the background, button, image and text to colors that support blue light vision deficiency.
- The *AdventureGameView* class which is the context class that utilizes the strategy object reference when executing applyColorScheme() to update the scene, room images, text instructions, and the items in the UI to the desired color using the intiUI initializing method and the changeColorScheme helper that update the scene and items using the updateScene, getRoomImage, showinstructions, formatText and updateItems methods to the needed color scheme.

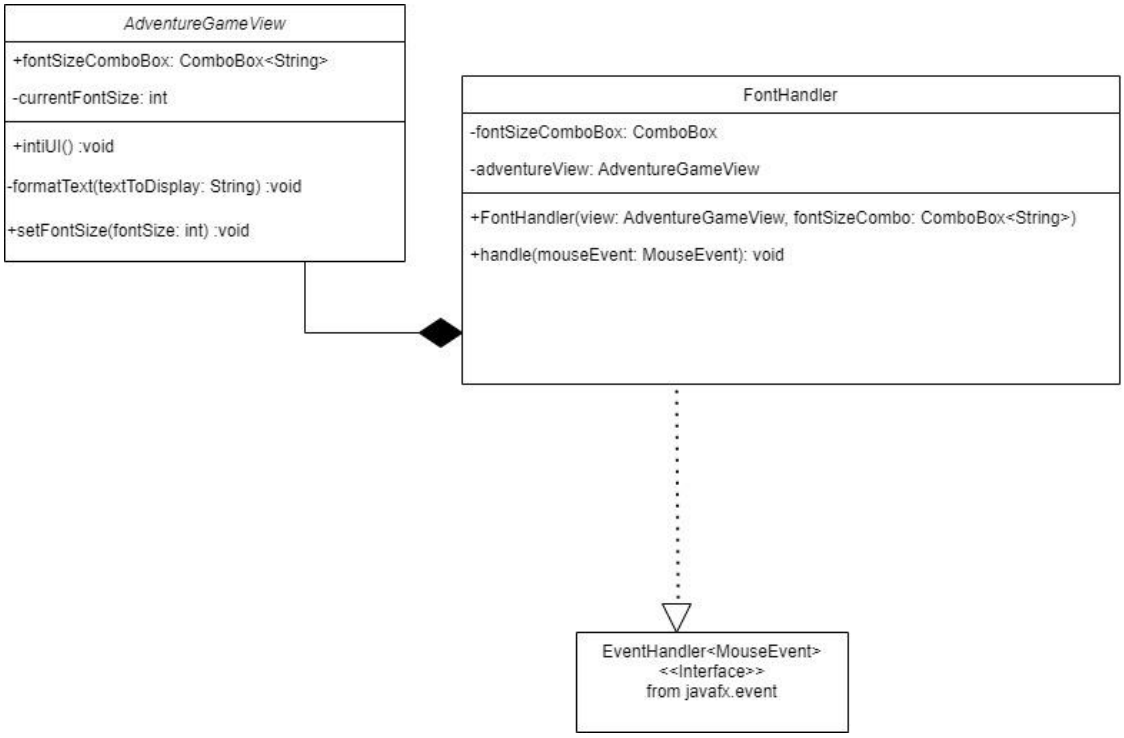
Design Pattern #2: Decorator
Overview: This pattern will be used to implement the audio cues functionality
UML Diagram:



Implementation Details:

- The ButtonAudio class contains two new methods.
- In the ButtonAudio class, the constructor function takes in an ArrayList of the names of the buttons, and searches for files containing an audio recording of their description, and then adds the information into the audioMap HashMap attribute.
- The audioMap attribute in ButtonAudio class contains the name of each button paired with the string name of the file location of the audio recording.
- The getMap attribute returns the audioMap.
- The addSoundToButtons attribute takes as argument the audioMap, and adds the audio recording to each button so that if the cursor is hovering over the button its description would play through a mouseEvent handler.

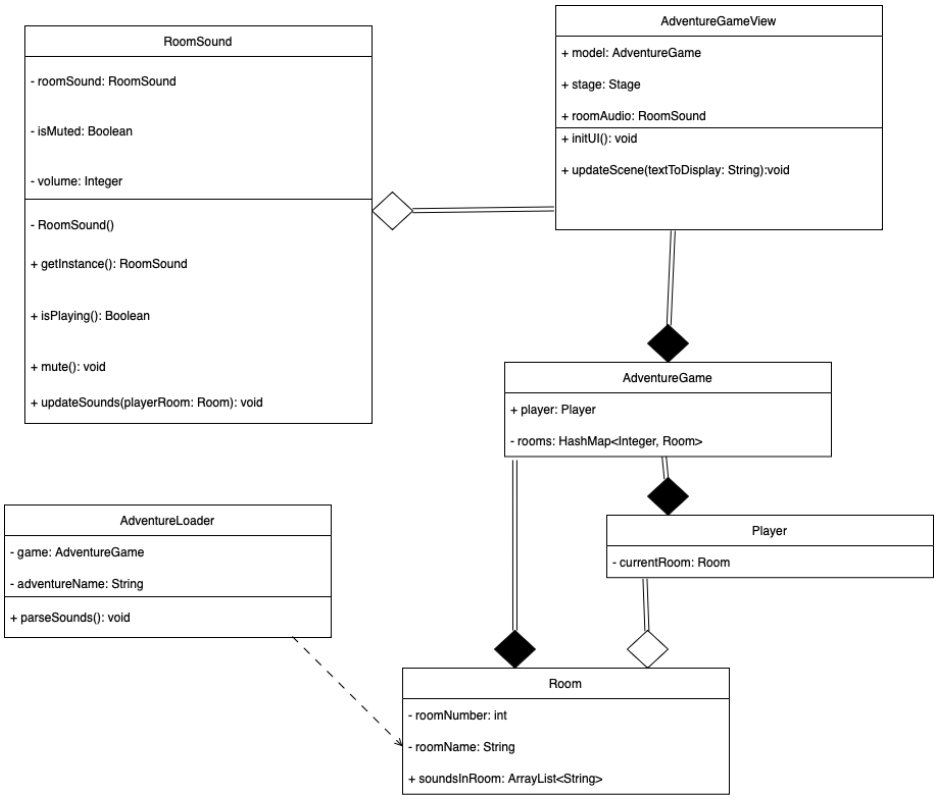
Design Pattern #3: Behavioral Design Pattern (Observer Pattern)
Overview: This pattern will be used to magnify the text within the game based on the User's selection between plausible font sizes from a dropdown menu.
UML Diagram:



Implementation Details:

- The eventHandler for the ComboBox is located in the AdventureGameView.java file and directly calls on the setFontSize method, which eventually changes the font sizes of the text within the specified nodes in the gridpane.
- On the other hand, the FontHandler.java file (implements the <EventHandler> interface provided by Javafx)handles the mouse event for when the font size from the drop-down menu is clicked. It then gets the selected font size and calls the setFontSize method in AdventureGameView.java.

Design Pattern #4: Singleton
Overview: This pattern will be used to create a separate class that performs an operation (processing the appropriate sound effects) while not interfering with the existing code.
UML Diagram:



Implementation Details:

- Room objects will have a container attribute that has distinct strings called soundsInRoom. Each string is a sound file name
- In the AdventureLoader class, a new method parseSounds() will populate the soundsInRoom attribute of each appropriate instantiated Room object if the “sounds.txt” exists.
- AdventureView will have a roomAudio attribute that holds an instance of RoomSounds
- RoomSounds will be updated in the updateScene() method of AdventureView
- RoomSounds will be responsible for processing what sounds are playing depending on which room the player is in.
- sounds.txt files must follow the format:
    Sounds (All caps)
    Room number
    4 dashes (“----”)

Example:

HAUNTED SAD RAIN
2

VICTORY TRUMPETS CHEER
3
