Design Document

Version 1 – 2023.12.05

Created 2023.10.31

Walter's World

1. Usha Sophea Janardhan
2. Aaron Davies
3. Franco Aparicio
4. Carson Lee

GitLab Repository:

https://mcsscm.utm.utoronto.ca/csc207_20239/group_27

Video Link:

https://youtu.be/eXWDIJoAY8I

SECTION 1: PROJECT IDENTIFICATION

The motivation behind this project is to foster inclusivity and engagement by catering to diverse player preferences and playstyles. By expanding the game's features, we aim to provide a more personalized experience for players, as well as further emphasizing the importance of accessibility. This extension to the adventure game will introduce new choices for players to take including multiple paths, player customization options, and diverse gameplay mechanics. We hope to enhance the existing functionality and encourage a richer, more interactive experience.

SECTION 2: USER STORIES

Name	ID	Owner	Description	Acceptance Criteria	Implementation Details	Priority	Effort
Completionist	1.1	Franco	As a user who is determined to fully complete every game I play, I want to know how many important aspects of the game I have completed and make sure that I never lose my progress.	As a completionist, I need a place to view a list of the important tasks I have completed and those I have yet to complete. I also need to be able to save my progress at any time and have the game save my progress when I achieve something important.	Add a set of achievements for each AdventureGame and notify the model when each one has been achieved. The model will contain a list of AdventureAchievement objects that will have a state attribute determining if it has been achieved or not.	1	5
Beginner-friendly	1.2	Aaron	I, Jacinda A., am not much of a gamer. I may take a while to become comfortable with the nature of the gameplay. I'd like to be able to referto the instructions from time to time if I forget them.	Given that I am a beginner, and relatively new to gaming in general, understanding the game will be difficult at first. With an instruction within arm's reach, I feel that I'll become more confident in navigating the game quickly.	Create a detailed instruction menu that can serve as a visual guide for players of all skill levels. Being able to easily access it will ensure players can spend more time enjoying the game instead of dealing with possible confusion and awkwardness.	1	3
Character Stats	1.3	Usha	As a dedicated player, I'd like to modulate my own gaming experience using "Stat Points"; they will allow the player to open and close opportunities through their own determination. I'd also like to have control over their distribution, so I can tailor my game	I need to have different "Character Classes" which influence the initial distribution of stat points. Character stats will determine the difficulty of the trolls encountered along the path.	The player class has a stats class. The aggregate value of stats is used to determine the strength of a troll that a player encounters.	1	5

			experience to a specific play style.				
Additional Trolls	1.4	Carson	As a dedicated player, I'd like to have variety to the trolls in while replaying the game.	The type of troll is determined arbitrarily upon encountering a troll. Trolls have varying types, they can modify player stats by a factor, initiate a minigame, or combat with the player.	Trolls are generated through a builder design pattern. The Troll selected is done arbitrarily through a random number generator. A text description will be required on each, which is made audible.	3	7
Visuals	1.5	Aaron	As a user who is visually impaired or has difficulty with reading certain fonts/colours, I would like to have a comfortable experience gaming without straining my eyes or taking away from the enjoyability of the game.	Within the settings menu or the side of the screen, there will be an accessibility button with extra options to change the visuals according to user needs.	The accessibility menu will be generated under the settings with a command design pattern.	3	5
Replayability/Player Choice	1.6	Usha	As an avid enjoyer of adventure games, I would like to be able to replay this game several times and be able to enjoy a different experience each time.	As a gamer who enjoys replaying games, I would like to be able to choose how I want to proceed through the game so that I may choose differently on different playthroughs.	Players may choose one of three player types that impact the strengths or weaknesses in their attacks on Trolls. Include different routes to the end goal with Trolls along the way.	1	6
Rewards	1.7	Carson	As a player who seeks to push the limits of games I play, I want my hard work to be rewarded whenever I pass a challenge.	After defeating a troll, I would like to have access to items that will improve my longevity or ability to	After trolls are defeated, grant players adventureObject rewards. We will add a special property to the	1	5

				navigate Walter's World.	Room class to prohibit players from dropping items in particular rooms (those with Trolls), and limit players to getting rewards only upon first completion of the trolls.		
Exploit Protection	1.8	Carson	As a developer, I would like if the optimal gameplay is not repetitive. Therefore, I receiving multiple rewards from the same troll room should not be possible.	As a developer, I do not want players to be able to abuse the reward system. I would like Trolls to only drop their items once so that players do not abuse the reward feature.	A Troll's inventory will only be generated when the particular Troll has not yet been created.	1	6
Secret Rooms	1.9	Franco	As a player interested in exploration and discovery, I want the game to feature hidden paths and secret areas with some challenge or reward for exploring.	As a player interested in exploration, I would like at least 1 hidden room per game that can only be accessed one way – which changes on every new game. This room should have a challenge that guarantees good items after passing.	Exactly one "secret passage" will be required in each AdventureGame's room.txt file to specify a room hidden from the player. The passage name to access this room will be randomly generated and will be communicated to the player in pieces as they progress through different rooms. Once the player has all they pieces, they can enter them in any order in the normal command entry box to reach the secret room.	2	4

Population	1.12	Carson	As a developer, I would like to add many interactions to counteract the game becoming stale. Adding helpful NPCs gives the player an interaction with the game which is not positive rather than mostly negative.	As someone who enjoys games, It would be interesting if there were NPCs that have interactions that are unique from other mechanics from within the game.	There will be an NPC class. In contrast to trolls, they serve as helpful or neutral interactions with the player.	3	4
Personaliz e	1.13	Franco	I like when I can personalize items and things in order to make them mine. It creates a sense of ownership.	As someone who likes making their belongings unique and personal to them, I'd like to be able to demarcate them from just their default form.	Items in the player's inventory will be able to be altered through renaming.	4	1
Text-to-Speech	1.14	Carson	As a user who has visual impairments or disability, I would like alternative delivery options on my announcements on progressing in game.	As a user who does not benefit from visual components of a game, I would appreciate it if there was a text to speech feature in the game.	An online library will be used to help implement the text to speech feature in the game so that the user can hear instructions as they progress through the game.	5	4
Diversity	1.15	Usha	As a user who likes to have options when playing games and influence my game play based off of strategy, I would like to have choices that influence how the game goes for me.	I want versatility of the gameplay in the diversity of players I can choose from the starting screen.	The Decorator design pattern will be used to implement the Player Class such that users can pick from three types of players at the start of the game, with different special abilities that influence their	3	5

					advantages in gameplay.		
Settings Menu	1.16	Aaron	As an avid and frequent gamer, I want to be able to access and modify settings easily.	I want to be able to customize my experience and manage app functionalities according to my preferences.	Inside of the AdventureGameView, we will create a new button for settings, and a new View, SettingsView, containing buttons for different settings.		4

Acceptability Criteria:

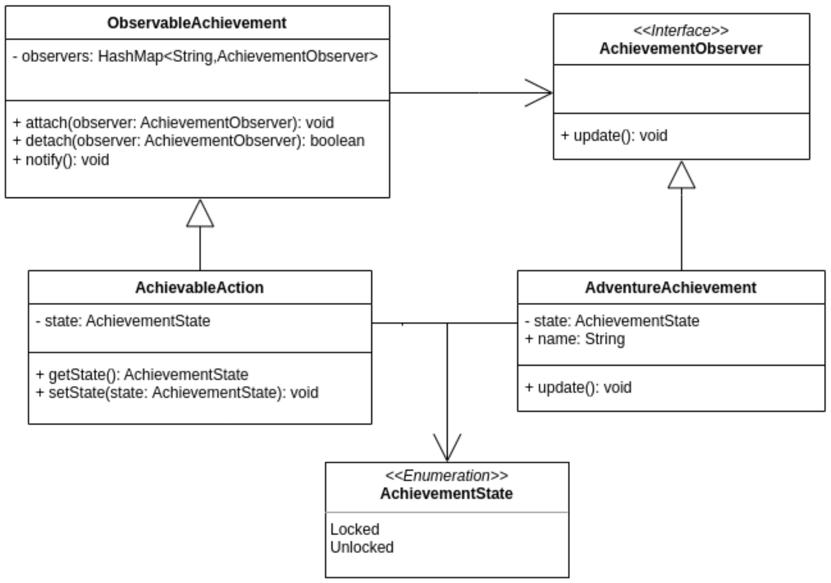
Name	ID	Description
Completionist	1.1	Require an achievements.txt file from each AdventureGame. Make an AdventureAchievement object with attributes that specify a name, description, tier, and image for each achievement. Make an achievements menu to allow players to view completed and missing achievements. Implement the observer design pattern to ensure achievements are rewarded accordingly. Automatically save the model when a player unlocks an achievement.
Beginner-friendly	1.2	Create an instruction/help menu. Have a button that lets you pull it up at any time. Make sure it's descriptive.
Character Stats	1.3	Make a Player object with stat attributes. Player classes have predetermined stats associated with them. Stats belong to the player class.
Additional Trolls	1.4	Upon moving into a troll passage, generate a new troll. Troll type is generated randomly. The troll can start an event, combat or game. The result will move you to a room or exit the game hp 0.
Visuals	1.5	Change certain visuals according to user needs: - Font sizes and types (serif and sans serif), background colours, The user can toggle accessibility settings using through a button in the screen.
Replayability/Player Choice	1.6	Create a PlayerSelectionView that includes the different player types to choose from. Create a StartView that includes the options of either starting a new game or loading an old one.
Rewards	1.7	Each troll drops an item that goes in inventory for player. The item grants a random amount of HP. Make the player unable to drop items or take items during a troll battle.

Exploit Protection	1.8	Edit the Troll builder to ensure that the Troll’s inventory is only populated with items if it has not been instantiated at a given location before. Only grant rewards upon the first victory against a troll.
Secret Rooms	1.9	Randomly generate a 5-character string to use to access the secret room. Select 5 random rooms (or the maximum number of non-special rooms) in the game. Display one (or more if necessary) character(s) of the string on the screen in each of the selected rooms.
Population	1.12	Create an NPC class. NPCs have dialogue, to share important information with the player. The NPCs can affect the Player’s stats or position.
Personalize	1.13	Creating a naming aspect. Having a button that lets you name items in your inventory. Ensure the names do not conflict with the names of other items.
Text-to-Speech	1.14	Use the FreeTTS java library to implement TTS. Have the TTS narrator say the long description upon entering a room for the first time. If the room is visited, the TTS will state the room name instead. The TTS will activate upon opening the instructions.
Diversity	1.15	Use the Decorator design pattern to implement different player types. The different player types will each have a unique specialMove that allows them advantages or disadvantages in the game, and thus can influence the choices the user makes.
Settings Menu	1.16	Add a new button to the AdventureGameView When you press, a new view, SettingsView should show up, containing buttons for all desired settings Ensure that every button in SettingsView executes its desired function.

SECTION 3: SOFTWARE DESIGN

Design Pattern #1: Observer Pattern

UML Diagram:


ObservableAchievement
- observers: HashMap<String,AchievementObserver>
+ attach(observer: AchievementObserver): void
+ detach(observer: AchievementObserver): boolean
+ notify(): void
<<Interface>>
AchievementObserver
+ update(): void
AchievableAction
- state: AchievementState
+ getState(): AchievementState
+ setState(state: AchievementState): void
AdventureAchievement
- state: AchievementState
+ name: String
+ update(): void
<<Enumeration>>
AchievementState
Locked
Unlocked

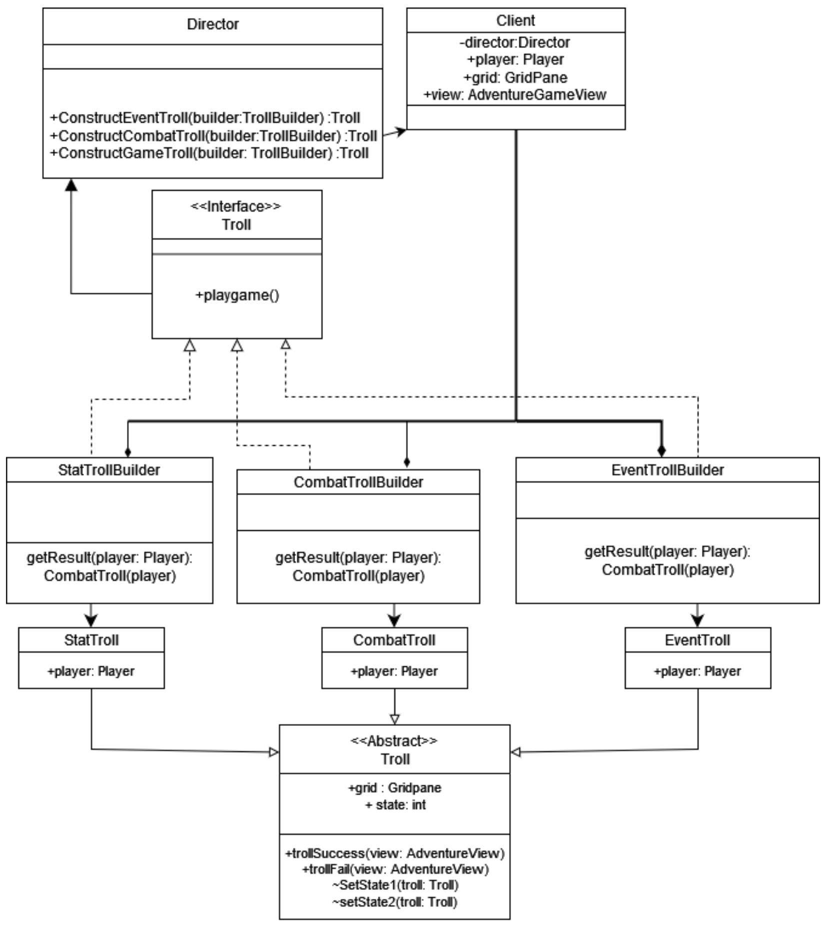
Overview: This pattern was used to implement the achievement functionality described in the user story with ID 1.1.

Implementation Details: The UML diagram outlines these main components:

- The *ObservableAchievement* abstract class, which includes three methods: attach, detach, and notify.
- The *AchievementObserver* interface, which includes an update method.
- The *AchievableAction*, which will have an *AchievementState* that it can report and set.
- The *AdventureAchievement*, which will have an *AchievementState* that will reflect the *AchievableAction* it observes.

The *notify* method of the *ObservableAchievement* will call *AchievementObserver.update*() on every *AchievementObserver* in its HashMap of observers. *AchievableAction.setState()* will modify the state of the *AchievableAction* and call *AchievementObserver.update(). AdventureAchievement.update()* implements this action by getting the state of the *AchievableAction* it observes, and updating its own state accordingly. It then triggers GUI updates to show the user.

Design Pattern #2: Builder Pattern


Director
+ConstructEventTroll(builder:TrollBuilder) :Troll
+ConstructCombatTroll(builder:TrollBuilder) :Troll
+ConstructGameTroll(builder: TrollBuilder) :Troll
Client
-director:Director
+player: Player
+grid: GridPane
+view: AdventureGameView
<<Interface>>
Troll
+playgame()
StatTrollBuilder
getResult(player: Player): CombatTroll(player)
CombatTrollBuilder
getResult(player: Player): CombatTroll(player)
EventTrollBuilder
getResult(player: Player): CombatTroll(player)
StatTroll
+player: Player
CombatTroll
+player: Player
EventTroll
+player: Player
<<Abstract>>
Troll
+grid : Gridpane
+ state: int
+trollSuccess(view: AdventureView)
+trollFail(view: AdventureView)
~SetState1(troll: Troll)
~setState2(troll: Troll)

Overview: This pattern will be used used to implement the Additional Trolls described in 1.4

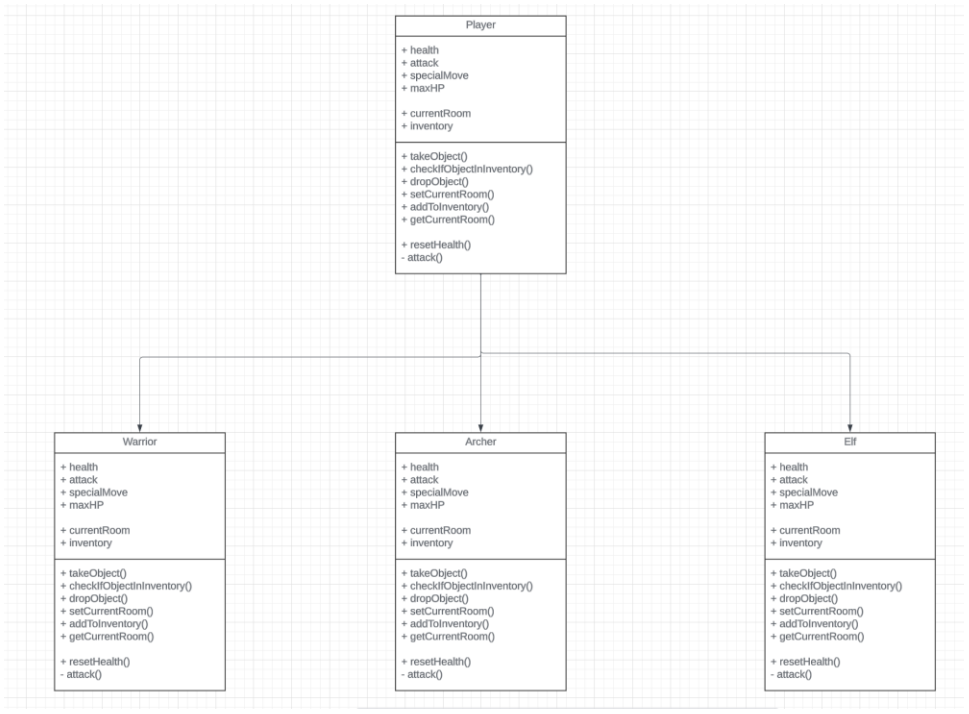
Overview: The client generates a director (capable of constructing all trolls) and a random number. Using the random number, the director calls a specific method which builds a troll, which is then returned to the client class.

Implementation Details: The UML outlines these components:

- The Director Class, containing the methods ConstructEventTroll, ConstructCombatTroll and ConstructGameTroll.
- Troll has common attributes state and grid, state is used to show AdventureGame where to move the player after a troll's event. The gridpane creates an area for the troll to generate the graphics that they need to display their game, combat or event. (jfx nodes)
- All trolls have a player class attatched. They can be used for combat, or their stats may be modified.
- The getResult method in the builder classes returns a child of the troll class to the client class.

When this code runs: builder created, random integer generated, builder method (returning troll) is called based on which range random number is between. In the builder class, the correct troll child is instantiated, containing attributes to display a game, event or combat to the user.

Design Pattern #3: Strategy Pattern



Overview:
The Player class implements the Decorator design pattern to achieve versatility and enjoyment in gameplay by allowing users to pick their choice of Player type between three options: Elf, Warrior, and Archer. Their choice also boosts the replayability and engagement of the game as each player type has their own special skill or specialMove that sets them apart.

Implementation Details:
The UML outlines these components:

- The *Player* decorator class, which includes previous attributes and methods already part of the existing Assignment 2, as well as 4 new attributes and 2 new methods:

Attributes:

- *health*: An integer representing the health points of the player.
- *attack*: An integer representing the damage caused by the player onto a troll when using *attack()*.
- *specialMove*: the special skill specific to each player type. It is only defined in the constructors of the child classes.
- *maxHP*: An integer representing the ceiling or maximum Health Points that can be possessed by a player.

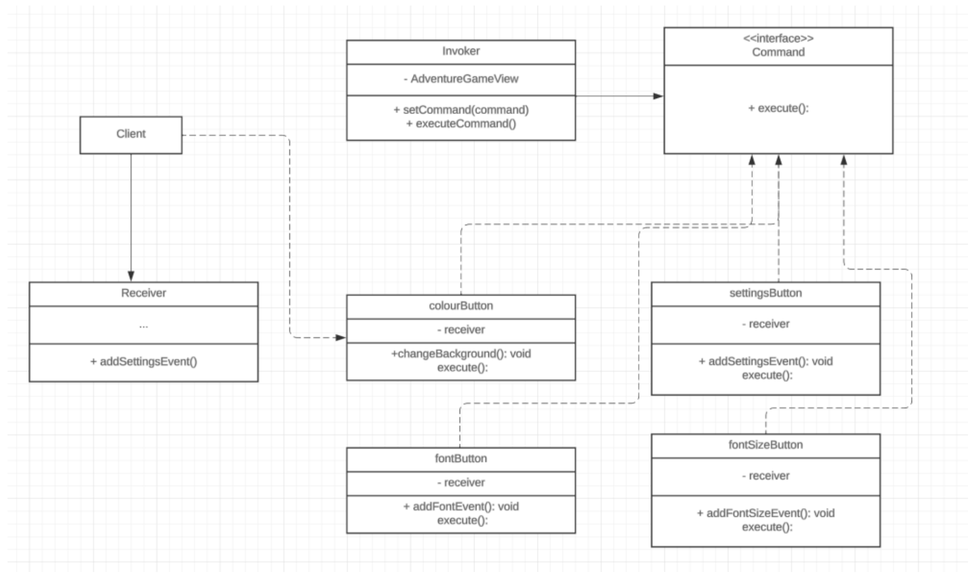
Methods:

    - *resetHealth()*: Used when starting a new game, it reverts the Player.health points back to 100%.
    - *attack()*: This is a blank method in Player that is implemented by each child class according to their player-troll interaction.
    -
- The three concrete child classes:
    - Warrior
        - For the Player type Warrior. The Warrior has a specialMove of Fists.
    - Elf
        - For the Player type Elf. The Elf has a specialMove of Magic.
    - Archer
        - For the Player type Archer. The Archer has a specialMove of Arrows.

In the game, users will be prompted to pick one of the three player types. Once they click on which player they would like, the relevant information is sent to *setUpGame*, where *makePlayer()* is called.
After the client calls *new Player()*, the according type of player out of the three is generated, and then its initializer implements their specialMove using their extended class constructor. The initializer will also make sure the correct values of health and attack are assigned or decorated to the Player. Furthermore, the player will be given a specific *attack()* method.

Design Pattern #4:

Overview: This Pattern will be used to implement the games settings / accessibility menu.



Implementation Details:

- The Command interface, which includes method execute
- The Invoker which initiates requests
- The receiver, which does the work of the commands
- The Client creates commands. It will toggle commands by pressing a button
- colourButton, settingsButton, fontButton, and fontSizeButton buttons implement different kinds of requests

The execute method in Invoker will take in a command. The executed actions allow the player to perform a specific action, such as changing the background color, changing the font, and changing the font size