Abstract

Text-based games, or Multi-User Dungeon ("MUD") games date back to as early as the 1970s when the first of their kind were developed as side projects in universities on large systems due to the lack of personal computers at the time. Their popularity grew with the early Internet and expanded outside of the university environment. They were interactive, fairly simple to design, and implemented many of the same features and interactions of other popular role-playing games of the time such as Dungeons and Dragons. These games are still enjoyed today by loyal fans of the genre with some of the very first MUD games still being played to this day.

Background

The idea for choosing this game as our project came from desire to implement a game of some sort. Not wanting to choose something that would be overly difficult to implement, but still challenging enough to learn from, the idea was proposed to implement the game as a text-based adventure that follows a fictitious undergraduate Computer Science student at Baylor. The scope would be niche but could be expanded to include the entire campus with quirky humor and familiar scenes from the daily life of a general Baylor student. We decided the scope could be as big or as small as we saw fit, with characters ranging from other students to the President of the University herself. We immediately started bouncing ideas off one another and soon after realized we had found our project.

Requirements

- REQ 001: Font color and background color must not clash.
- REQ 002: Keyboard and mouse for input.
- REQ 003: Non-touchscreen monitor for display.
- REQ 004: Text entry must be exclusive to standard ASCII excluding non-printable characters.
- REQ 005: Java Runtime Environment 8 to run the .jar file.

Analysis, Design and Implementation

Due to the programming language restrictions and that we were learning Java during the duration of the project, we used the tools and knowledge available to us to our full benefit. The bulk of our code design was implementing the project in Java Swing since we needed an interactive GUI for the player to use. The game could be as small or massive as we liked but we needed to find ways to make it fun enough to keep players coming back while being simple enough that anyone could play the game without much instruction. The player will initially start the game at the main menu where they can choose to start a new game or load into a previously saved state. They then proceed to the main map where they are free to travel to various buildings on Baylor's campus and may even encounter a few non-interactive encounters. To give the game depth, we decided to make a battle system where the player could "fight" common foes a computer science student might come across such as a difficult assignment or test. The battle system would be random, giving the player choices of tasks they could perform in a turn-based environment against their enemies. The player can choose to retreat from the fight or stand their ground and attempt to dispose of their enemy. The player wins the game by progressing through the scripted story without losing all of their health. However, if they do lose all their health, they are sent back to the menu where they can choose to load a saved game, start a new game, or quit. At any time on the main map screen, the player can save their game state for later. This is all done through Swing menus, tables, and text fields.

Evaluation and Testing

Testing was implemented through both Black Box and White Box testing methods. Initially, we tested and fixed errors as we found them while coding the modules. This style of testing fit more into Grey Box than White or Black Box methods. We found errors just by playing the game ourselves and then more errors while sifting through each other's lines of code. Team Coffee Beanz gave us a great report of bugs and errors they found while inspecting our code that was invaluable to fixing mistakes that we had not previously found. Their Black Box testing gave us motivation to look deeper into our code and find even more errors. For our strictly White Box testing we used both SpotBugs and JUnit to clean up last minute bugs in our implementation such as closing file streams and write errors in our load function.

Conclusion

After completing this project, we feel like we got a great, hands-on practical way to implement what we learned in an actual project. We believe this is a great approach to instill what is learned in the academic portion of the classroom in a fulfilling project. And while challenging to implement code-wise, we also found it a challenge to be creative enough to give the player a good experience with our game. We could see this project continuing far beyond the scope of this class, as it has nearly limitless possibilities to be expanded and made into a game that includes all students, faculty, and anyone familiar with the school in general. The scope could be expanded outside of the campus into Waco, and from their become quite massive. We enjoyed making the game, working together, and the sense of accomplishment we gained from making it.

Iteration 3 - Use Cases

UC 1: Player attacks an enemy

Name: Player Attacks

Scope: user goal
Primary Actor: Player

Stakeholders and Interests:

-Player: Would like to attack an enemy on the map

Preconditions:

-Battle is initiated, system properly loads the system

Postconditions:

-Battle system is loaded.

Main Success Scenario:

1) The player walks around in the map until a random encounter or scripted battle occurs.
2) The system loads the battle system and passes the player and enemy info to the battle system. 3) Battle system loads the battle with the proper stats passed to it from the system.

Extensions:

Special Requirements:

-Screen is readable
-Random encounters not too frequent or sparse.

Technology and Tech Variations List:

-Keyboard for input of commands by player

Frequency of Occurrence:

-Frequent, as the player will battle often for experience.

Open Issues:

UC 2: Player battles an enemy

Name: Player Battle
Scope: user goal
Primary Actor: Player

Stakeholders and Interests:

-Player: Would like a quick battle, easy to navigate menus that display simple

Preconditions:

-Battle is initiated, system properly loads the system

Postconditions:

information effectively.

-Player exits the battle in victory or defeat, depending on which one a screen is shown.

Main Success Scenario:

1) The player selects an option from the battle menu.
2) If Attack, the system asks the player which enemy to attack.
3) The player selects the enemy they wish to attack.
4) The system then calculates damage based on the attack of the player minus the defense of the enemy.
5) The system deducts from the enemy health accordingly, and updates the screen.
6) This ends the player's turn.
7) After the user's turn, the enemy system selects the player to be attacked, and the system calculates damage.
8) The system then updates the screen.
9) This repeats until either the player flees, the enemy is defeated, or the player is defeated.
10)The system then exits the battle, and returns the player to the map screen.

Extensions:

a) If Flee is chosen, the system exits the battle and returns the user to the map.
b) If Abilities, the system updates the menu displayed and shows the user's abilities on the menu.

1) The user then selects which ability they wish to use.

2) If applicable, the system asks for a target to attack, and the user selects which enemy to attack with that ability.

3) The system performs the ability and if necessary, changes necessary data, then updates the screen.

4) The user's turn is now over.
c) If the user loses the battle, the system exits the battle and displays a game over screen

Special Requirements:

-Screen is readable
-Battle is fast, no hiccups in graphics

Technology and Tech Variations List:

-Keyboard for input of commands by player

Frequency of Occurrence:

-Frequent, as the player will battle often for experience.

Open Issues:

UC 3: Player travels on the map

Name: Player Traverses Map

Scope: user goal
Primary Actor: Player

Stakeholders and Interests:

-Player: Would like to easily traverse the map and enter the buildings quickly.

Preconditions:

-Map menu is loaded.

Postconditions:

-Player changes location.

Main Success Scenario:

1) The system displays the list of possible buildings the player can enter (like Cashion, Moody, etc.). 2) The system asks the player which building they would like to enter.
3) The player uses the input box on the map menu to enter their selection.
4) The system updates the player's location and displays the list of rooms the player can enter, if

any are available to the player.
5) The system asks where the player would like to go, and also offers the option to leave the

building.

Extensions:

1. a) The player wants to load a menu on the map menu. 1) The system loads the proper menu.
2. b) The player wishes to enter a building that does not exist.
   1) The system informs the player that the building does not exist and prompts again.
3. c) The player gives misspelled input.

1) The system informs the player that the input was not correct and prompts again.

Special Requirements:

-Screen is readable
-Random encounters not too frequent or sparse.

Technology and Tech Variations List:

-Keyboard for input of commands by player

Frequency of Occurrence:

-Frequent, as the player will need to traverse the map in order to go places.

Open Issues:

UC 4: Player launches a new game

Name: Launch New Game

Scope: Text-based RPG

Level: user goal

Primary Actor: Player

Stakeholders and Interests:

- Player: Wants to be able to start a new game from the beginning, including starting a new game when other save states exist.

Preconditions: Player is at the start menu at the main screen.

Success Guarantee: The game starts from the beginning with new character data.

Main Success Scenario:

1. Player boots into the opening start menu.

2. Player chooses New Game from options.

3. Player creates a new character and enters bio information.

4. Player is asked to verify they wish to continue with the information they entered and start a new game.

5. Verify the player's hard drive has enough space for a save state.

6. New character is saved to a save state, labeled by the character's name, and game starts from the beginning.

Extensions:

a. At any time, the player decides to cancel the new game creation: 1. Player chooses the cancel option.
2. Player is returned to the Start menu.

Special Requirements:

- Font color and background color must not clash. - Keyboard and mouse for input.

- Non-touchscreen monitor for display.

- Text entry must be exclusive to standard ASCII excluding non-printable characters.

- Hard drive space available for save states of 1GB recommended.

Technology and Data Variations List:

a. Player bio information fields entered with a keyboard.

b. Info fields can be selected with a mouse or with the keyboard.

Frequency of Occurrence: Can be created one at a time until either HD space is maxed or at user's discretion.

Miscellaneous:

- Explore idea of integrating a standard USB controller and an onscreen keyboard for inputting data.

- What character customization would we like to integrate.

UC 5: Player loads a saved game

Name: Load Game
Scope: Text-based RPG
Level: user goal
Primary Actor: Player **Stakeholders and Interests:**

- Player: Wants to be able to load a game and pick up the game right where they left off, with the same stats, name, and goals completed.

Preconditions: Player is at the pause menu, or main menu
Success Guarantee: The game loads player stats and completed goals from a file.
Main Success Scenario:
1. Player opens the pause menu or main menu.
2. Player Load Game from options.
3. Game loads file and initializes the stats and completed goals with the ones from the file 4. Game loads the new player and world to fit what was saved.
Extensions:
a. At any time, the player decides to cancel the new game creation:

1. Player chooses the cancel option.

2. Player is returned to the Start menu.

Special Requirements:

- Font color and background color must not clash.
- Keyboard and mouse for input.
- Non-touchscreen monitor for display.
- Text entry must be exclusive to standard ASCII excluding non-printable characters.
Technology and Data Variations List:

a. Player info loaded from a save file.
Frequency of Occurrence: Loaded every time the player wishes to load their save game.
Miscellaneous:
- Explore idea of integrating a standard USB controller and an onscreen keyboard for selection of save

UC 6: Player opens skills/inventory menu

Name: Open Menu
Scope: Text-based RPG

Level: user goal
Primary Actor: Player

Stakeholders and Interests:

- Player: Wants to be able to open a menu and interact with it.

Preconditions: Player is on the map, or in a battle, or paused.

Success Guarantee: The game opens the required menu.
Main Success Scenario:

1. Player wants to open a menu.
2. Player clicks on the menu button.
3. Game loads the menu that the player selected (pause, items, save, etc).

Extensions:

a) Player wants to close menu.
1) Player clicks the close menu button.

Special Requirements:

- Font color and background color must not clash.
- Keyboard and mouse for input.
- Non-touchscreen monitor for display.
- Text entry must be exclusive to standard ASCII excluding non-printable characters.

Technology and Data Variations List:

- Keyboard and mouse to provide player input.
- Screen to display the menu on.
Frequency of Occurrence: A menu could be loaded at any time.

Miscellaneous:

UC 7: Player saves game

Scope: Baylor RPG Game

Level: Player Goal
Primary Actor: Player

Stakeholders and Interests:

-Player: Wants to save the progress that has been made within the system, so the data can

be retrieved for future use.
Preconditions: Player has created a game and has made progress.
Postconditions: Game data is saved to the system and can be retrieved for future use.

Main Success Scenario:

1. Player chooses to save the game progress.
2. The system saves the game to the system memory.
3. The system confirms the data has been saved.
4. The system displays the menu.
5. Player chooses the next action.

Extensions:

Special Requirements:

-Menu is easy to read.

-Screen display is colorful.

Technology and Tech Variations List:
-Keyboard/Mouse to control Player, select menu items.

Frequency of Occurrence: Could be once for each session of play.

Open Issues:

UC 8: Player deletes game

Name: Player Deletes Game

Scope: Baylor RPG Game

Level: Player Goal
Primary Actor: Player

Stakeholders and Interests:

-Player: Wants to delete a previous game progress.

Preconditions: Player has game data within the system.

Postconditions: Game data is deleted from system memory.

Main Success Scenario:

1. Player chooses to delete a game.
2. The system erases the game data.
3. The system confirms the game has been erased.
4. The system displays the menu.
5. Player chooses the next action.

Extensions:

Special Requirements:

-Menu is easy to read.

-Screen display is colorful.

Technology and Tech Variations List:
-Keyboard/Mouse to control Player, select menu items.

Frequency of Occurrence:

Could be once for each session of play.

Open Issues:

UC 9: Player adds item to inventory

Name: Player Adds Item to Inventory

Scope: Baylor RPG Game
Level: Player Goal
Primary Actor: Player

Stakeholders and Interests:

-Player: Wants to be able to add an item to inventory for future.
Preconditions: Player has created and made progress within the game. Player has encountered an item within the game.
Postconditions: The item is added to the Player inventory.
Main Success Scenario:

1. Player encounters a game item and wishes to add it to the inventory for future use.
2. Player chooses to add the item to inventory.
3. The system adds the item to inventory.
4. The system confirms the item is added to the inventory.

Extensions:
Special Requirements:

-Menu is easy to read.

-Screen display is colorful.

Technology and Tech Variations List:
-Keyboard/Mouse to control Player, select menu items.

Frequency of Occurrence: Could be fairly frequent.

Open Issues:

-Can the player have duplicate items in inventory?

UC 10: Player enters player data

Name: Enter Player Data

Scope: Text-based RPG

Level: user goal
Primary Actor: Player

Stakeholders and Interests:

- Player: Wants to be able to enter their information and have the game store it.

Preconditions: Player starts a new game.
Success Guarantee: The game begins with the new player data.
Main Success Scenario:

1. Player opens a new game.
2. Game asks for the Player's name.
3. Player enters desired name.
4. Game begins a new game with the player's name initialized, as well as initial stats.

Extensions:

1. Player does not enter a name.
2. Game asks the player to please enter a name.

Special Requirements:

- Font color and background color must not clash.
- Keyboard and mouse for input.
- Non-touchscreen monitor for display.
- Text entry must be exclusive to standard ASCII excluding non-printable characters. **Technology and Data Variations List:**

- Keyboard and mouse to provide player input.
- Screen to display the menu on.

Frequency of Occurrence: Occurs only in a new game. **Miscellaneous:**

UC 11: Player uses item from inventory

Name: Use item from inventory

Scope: Text-based RPG Game (Baylor Edition)

Level: user goal
Primary Actor: Player

Stakeholders and Interests:

- Player: wants to use an item from the inventory. Wants easy-to-use interface. Wants game to save progress.

Preconditions: the item is available in the inventory.
Success Guarantee (or Postconditions): Player was able to use the item. Inventory is updated.

Main Success Scenario (or Basic Flow):

1. Player chooses option to access Inventory.
2. System loads Inventory.
3. Player chooses desired item from Inventory.
4. System updates state of Inventory.
5. Player uses the item form Inventory.

Extensions (or Alternate Flows):

*a. At any time, the desired item may not be in Inventory:

1. Player chooses option to access Inventory.
2. System loads Inventory Menu.
3. Player realizes item is not in Inventory.
    1. If Player chooses another item from Inventory
    2. Inventory state is updated

4. Player exits Inventory Menu.

Special Requirements:

- Screen is readable.
- Inventory Menu is easy to read.

Technology and Data Variation List:

- Keyboard to control Player and mouse to select Menu and Inventory items.

Frequency of Occurrence:

- Fairly frequent.

Open Issues:

UC 12: Player enters random encounter

Name: Encounter
Scope: Text-based RPG

Level: user goal
Primary Actor: Player

Stakeholders and Interests:

- Player: Wants to seed a random encounter with a NPC to give the game more depth.

Preconditions: Player is on the map.
Success Guarantee: The game seeds a random encounter with a scripted dialog while the player is traveling to different locations on the map.
Main Success Scenario:
1. Player travels to a location on the map.
2. Game seeds a random number that corresponds to a value in a file and prints that text to the screen.
3. Afterwards, player can still travel to their location.
4. Game resumes its processes.
Extensions:
Special Requirements:
- Font color and background color must not clash.
- Keyboard and mouse for input.
- Non-touchscreen monitor for display.
- Text entry must be exclusive to standard ASCII excluding non-printable characters.
Technology and Data Variations List:

Frequency of Occurrence: 1:15 chance depending on the value generated by the random integer function.

Miscellaneous:

UC 13: Player pauses game

Name: Pause Game
Scope: Text-based RPG

Level: user goal
Primary Actor: Player

Stakeholders and Interests:

- Player: Wants to be able to pause and unpause the game at will, letting them come back to the game or access menus.

Preconditions: Player is on the map.
Success Guarantee: The game stops running and loads the pause menu.
Main Success Scenario:
1. Player clicks to open the pause menu.
2. Game pauses its running and loads the pause menu.
3. After player does what they wish in the pause menu, they unpause.
4. Game resumes its processes.
Extensions:
Special Requirements:
- Font color and background color must not clash.
- Keyboard and mouse for input.
- Non-touchscreen monitor for display.
- Text entry must be exclusive to standard ASCII excluding non-printable characters.
Technology and Data Variations List:
Frequency of Occurrence: Pause menu and game can be paused when the player is on the map.
Miscellaneous:

UC 14: Player completes a stage

Use Case UC3: Player completes a level

Scope: Text-based RPG Game (Baylor Edition)

Level: user goal
Primary Actor: Player
Stakeholders and Interests:

- Player: wants to finish the current level and start the a new one. Wants easy-to-use interface. Wants game to save progress.

- Enemy: wants to destroy Player.

- Battle: is initiated.

Preconditions: Player has started a level.
Success Guarantee (or Postconditions): Player has successfully completed a level. Game progress is saved.

Main Success Scenario (or Basic Flow):

7. Player clicks to start a new game.
8. System loads the level.
9. Player starts the new level.
10. PlayerfacesanEnemy.
11. Player defeats the Enemy.

Player repeats steps 4-5 until all Enemies are defeated.

12. Player reaches end of level.
13. System presents message regarding end of level.
14. Player move on to next level.

Extensions (or Alternate Flows):

*a. At any time, Player may restart level: 1. System loads the level.
2. Player starts the level again.
3. Player faces an Enemy.

4. Player defeats the Enemy.

Player repeats steps 4-5 until all Enemies are defeated.

5. Player reaches end of level.
6. System presents message regarding end of level. 7. Player move on to next level.

*b. At any time, Player may pause game:

1. System pauses game.
2. System loads Menu for Player.
3. If Player does not wish to take further actions, then game is renewed.
4. Player finds Enemy.

5. Player defeats the Enemy.

Player repeats steps 4-5 until all Enemies are defeated. 6. Player reaches end of level.

7. 8.

*c. At 1.

2. 3. 4. 5. 6. *7. 8. 9.*

System presents message regarding end of level. Player move on to next level.

any time, Player may quit game:
Player chooses option to pause game
System pauses game.
System loads Menu for Player.
Player selects the quit game option.
System asks Player if he/she wishes to quit game. Player selects "Yes".
System quits game.
System shows Main Menu to Player.
Player chooses a new option from Menu.

*d. At any time, Player may be defeated by Enemy:

9. Player battles Enemy.
10. Enemy defeats Player.
11. Player loses a life.
12. System updates the Player's lives.
13. System shows message regarding defeat.
14. *System loads Main Menu for Player.*
15. *Player chooses a new option from Menu.*

Special Requirements:

- Screen is readable.
- Menu easy to read.

Technology and Data Variation List:

- Keyboard to control Player and mouse to select Menu and Inventory items.

Frequency of Occurrence:

- Fairly frequent, though the Player may get stuck in a level.

Open Issues:

UC 15: Player gains a life

Name: Gain life

Scope: Text-based RPG Game (Baylor Edition)

Level: user goal
Primary Actor: Player
Stakeholders and Interests:

- Player: wants to play a game that is fast and fun. Wants easy-to-use interface. Wants game to save progress.

- Enemy: wants to defeat Player.

Preconditions: Player must be alive.
Success Guarantee (or Postconditions): Player gains one live. Lives list is updated.

Main Success Scenario (or Basic Flow):

1. Player clicks to start a new game.
2. System loads the level.
3. Player starts the new level.
4. Player finds a live.
5. Player gets live.
6. Lives list is updated.

Extensions (or Alternate Flows):

Special Requirements:

- Screen is readable.
- Inventory Menu is easy to read.

Technology and Data Variation List:

- Keyboard to control Player and mouse to select Menu and Inventory items.

Frequency of Occurrence:

- Not that frequent.

Open Issues:

UC 16: Player dies

Name: Player Dies
Scope: Text-based RPG

Level: user goal
Primary Actor: Player

Stakeholders and Interests:

- Player: Wants to be able to have the possibility of losing the game.

Preconditions: Player health reduced to 0.
Success Guarantee: The game stops running and loads the load menu.
Main Success Scenario:
1. Player enters a battle.
2. Player's health is reduced to zero as a result of a fight.
3. After player dies the load menu is presented and player chooses to load a saved game.
4. Game resumes its processes.
Extensions:
Special Requirements:
- Font color and background color must not clash.
- Keyboard and mouse for input.
- Non-touchscreen monitor for display.
- Text entry must be exclusive to standard ASCII excluding non-printable characters.
Technology and Data Variations List:
Frequency of Occurrence: Anytime the player's health is reduced to zero as a result of losing a battle.
Miscellaneous:

UC 17: Player stats increase

Name: Stats Increase
Scope: Text-based RPG

Level: user goal
Primary Actor: Player

Stakeholders and Interests:

- Player: Wants to be able to increase their stats as they progress through the game.

Preconditions: Player is on the map.
Success Guarantee: The player's stats increase.

Main Success Scenario:
1. Player accomplishes a goal such as winning a battle.
2. Game increases player stats.
4. Game resumes its processes.

Extensions:

Special Requirements:
- Font color and background color must not clash.
- Keyboard and mouse for input.
- Non-touchscreen monitor for display.
- Text entry must be exclusive to standard ASCII excluding non-printable characters.

Technology and Data Variations List:

Frequency of Occurrence: Random depending on the player's ability to win battles and other goals.

Miscellaneous: