Luis Cruz Pereda

Ms. Kanemoto

12/5/2024

Final project Report

What my game is about, I made a text-based adventure game. Evolving the player into a series of rooms in a dungeon choosing to go left, right and back, however going back will be locked in player until they have access to enter. I wanted to have the player to have random encounters when exploring the rooms with the goal to reach the final boss, the Dark Lord defeat and end the game. The rooms will have four different encounters that are random actions, one room being a treasure room that finds gold and weapons/armor from equipment in my excel file. Another room being a healing pool where that player will get healed without using potions. Another room being a store where players can buy potions, armor, and weapons but only common. The last room will be a fighting encounter with three different types of creatures a lizard person, Man-bat, and a Goblin. During the fighting I gave the player four options to choose from: attack which will randomize how much attack power they will the enemy and player will hit for, a sneak where the attack will hit for random number but doubled, option to run which will success or fail if fail player will take damage. The last option player can choice from is using a potion to heal for 50 hp. I added one more encounter in the game how the players will not be able to access that encounter until they have 3 enemies defeated which will unlock the room with golem who asks a riddle if you answer correctly the player will be given a boss key and are able to go forward and fight the boss and win the game, however if players answer the riddle incorrectly will take damages and will have three tries to get it correctly if failed players will get kicked out of the room and the enemies defeated will reset back to 0 and will have to defeat 3 more enemies again. It is a short little game I would like to add more eventually and make it even more fun by adding puzzles, traps, more enemies and slowly adding more to weapons to make the game challenging and fun.

A computer screen with colorful text

Description automatically generatedAlgorithm 1: Enemy Encounter Algorithm

The purpose of This algorithm is handling the random enemy encounter when the player enters the room. This chooses which enemy the player could be fighting the Goblin, Lizard Person or the Man-Bat. Determines how much health and level the enemy has. Then initiates the combat.

1. Randomly selects an enemy from my list of creatures (Goblin, Lizard Person, Man-Bat)
2. Randomly chooses the level for the enemy from 1 to 3
3. Calculate the enemy’s health based on its level times by 5 and add that to 15.
4. Print out “you enter a dark room and a wild + enemies name + appear!”
5. Initiate battle between players and enemy.

A screen shot of a computer code

Description automatically generatedThis Algorithm goes to the next part for battle between player and enemy. Loops between the players turn with choices between attack, sneak, use potion and run and then the enemies turn to attack until one is defeated.

1. Prints out player name faces enemy name.
2. Prints out players’ actions Attack, Run, Sneak, Use potion.
3. Prints out players’ health.
4. Prints out enemies’ health.
5. Player choose action
   1. If player picks attack then randomize players attack
   2. Prints how much attack damage the player does
   3. Enemy takes damage
   4. Switches to enemies turn
   5. Randomize attack for the enemy between 1 to 11 plus 5
   6. Player take damages
   7. Returns to step 2
6. Player picks run
   1. If success player runs away turns to entrance
   2. If a failed player takes 10 hp of damage
      1. Switches to enemies turn
      2. Randomize attack for the enemy between 1 to 11 plus 5
      3. Player takes damages
      4. Returns to step 2
7. If player picks sneak
   1. See if sneak was done before if player haven’t
      1. then randomize players attack then times by 2
      2. Prints how much attack damage the player does
      3. Enemy takes damage
      4. Switches to enemies turn
      5. Randomize attack for the enemy between 1 to 11 plus 5
      6. Player takes damages
      7. Returns to step 2
   2. If players have done Sneak already
      1. Prints out the player already used sneak
      2. Enemy takes damage
      3. Switches to enemies turn
      4. Randomize attack for the enemy between 1 to 11 plus 5
      5. Player takes damages
      6. Returns to step 2
8. Player picks use potion
   1. Heals player plus 50 hp
   2. Switches to enemies turn
   3. Randomize attack for the enemy between 1 to 11 plus 5
   4. Player takes damages
   5. Returns to step 2
9. If player dies
   1. Prints out enemies’ name has defeated players’ name
   2. Ends game
10. If player wins
    1. Prints out player’s name has defeated enemies’ name
    2. Increases enemies’ defeated count by 1
    3. Returns to main entrance.

I was bad with Big O times I had to ask ChatGPT but it said Random number generation and enemy selection is constant time O(1).

The combat method involves iterating through turns until one either the player or enemy dies, soits time complexity can vary based on the number of turns, but it’s still considered O(n), where n is the number of turns in the fight.

A computer screen shot of a program code

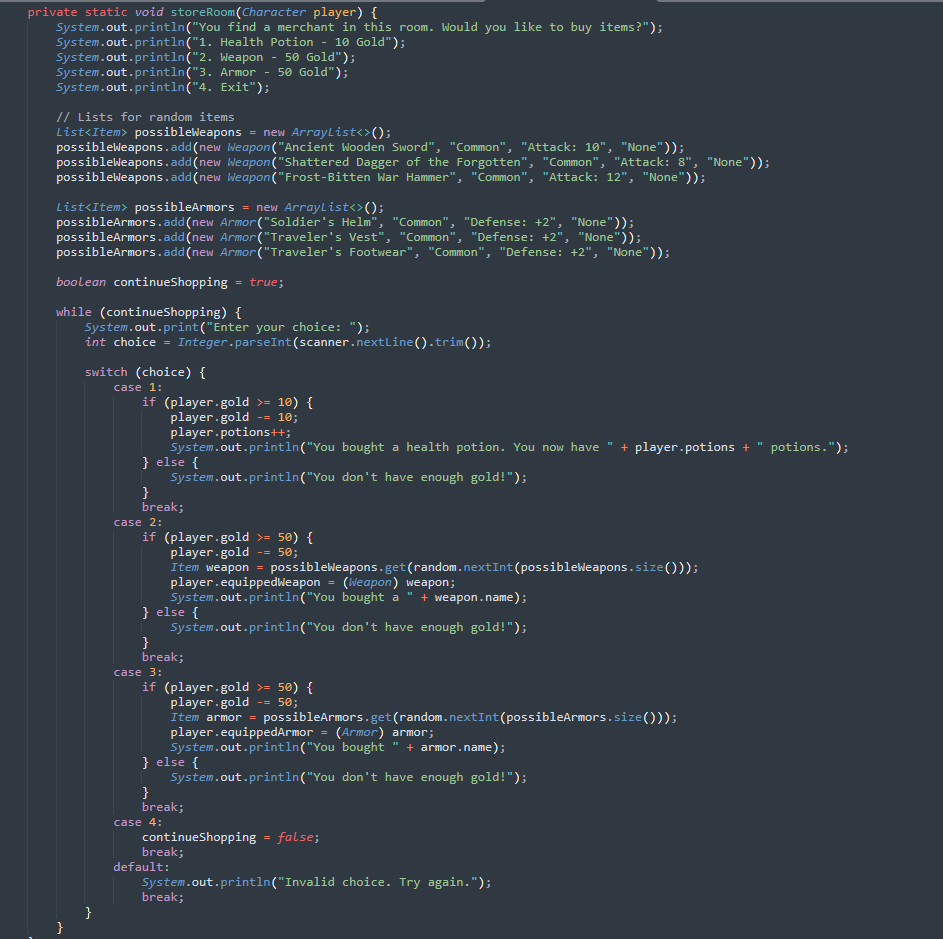
Description automatically generatedAlgorithm 2: Riddle answering for the Golem Encounter

The purpose of this algorithm has the player solve a riddle that is given by the Golem to acquire the boss key if player answers correctly, if player is incorrect, they take 10 health of damages each incorrect answer. If player fail three times the player is removed from the room and the enemies’ defeated counter is reset and player will need to defeat three more enemies to return to the room.

1. Player enters room
2. Prints out “A Golem stands before you, blocking your path.”
3. Prints out “To pass, you must solve its riddle.”
4. Randomly selects a riddle from the list of riddles I provided.
5. While the attempt is 3 or under( if attempt is over 3 go to step 12)
6. Prints out the selected riddle
7. Prints out “your answer: ”.
8. Player type in answer
9. Program checks if answer is correct.
10. If answer is correct
    1. Player is given boss key
    2. Returns to main entrances
    3. Prints out “You have the Boss Key. You can also go forward to face the Dark Lord.".
11. If the answer is incorrect
    1. Attempt is increased
    2. Player takes 10 damages subtracted 5 by the player defense taking 5 damage in total
    3. Prints out players new health
    4. Returns to step 5
12. If attempt is over 3
    1. Prints out “You failed to solve the riddle after 3 attempts.”
    2. enemiesDefeated is changed back to 0
    3. players’ are return to main entrance and can’t return until enemiesDefeated reaches back to 3.

The times complexity of this algorithm is O(1) since the number of attempts is fixed at 3, and the number of riddles is small. It does not depend on the size of the game world or other variables.

Algorithm 3: Item Shop Algorithm

The Purpose of this algorithm allows the player to interact with a merchant to purchase items, which are healing potions, weapons that are common type of rarities, or armor.

1. Players choose left or right
2. Random encounter found storeroom
3. Prints out “You find a merchant in this room. Would you like to buy items?”
4. Prints out “1. Health Potion - 10 Gold”
5. Prints out “2. Weapon - 50 Gold”
6. Prints out “3. Armor - 50 Gold”
7. Prints out “4. Exit”
8. While player are in the store
9. Prints out “Enter your choice: ”
10. Player pick from 1 to 4
11. If players’ pick 1 to buy potions
    1. Checks if player has 10 gold or more
    2. If player has 10 gold or more
       1. Take 10 gold form player
       2. Increase the number of potions by 1
       3. Prints out the players bought potion and shows how many potions the player has in total
       4. Returns to step 8
    3. If player has less than 10 gold
       1. Prints out player doesn’t have enough gold!
       2. Return to step 8
12. If player picks 2 to buy Weapon
    1. Checks if player has 50 gold or more
    2. If player has 50 gold or more
       1. Take 50 gold from player
       2. Randomize and randomly picks one of the weapons form the list<Item>
       3. Equips weapon to player
       4. Prints out the name of the weapon that the player bought
       5. Returns to step 8
    3. If player has less than 50 gold
       1. Prints out player doesn’t have enough gold!
       2. Returns to step 8
13. If player picks 3 to buy Armor
    1. Checks if player has 50 gold or more
    2. If player has 50 gold or more
       1. Randomize and randomly picks one of the armors form the list<item>
       2. Equips Armor to player
       3. Prints out the name of the armor that the player bought
       4. Returns to step 8
14. If player picks 4 to exit
    1. Player exits storeroom and returns to main entrance.

The storeRoom handles a simple purchasing system. The player can choose between health potions, weapons, and armor, and each item has a price. The player can also choose to exit the storeRoom

The player is only allowed to purchase an item if they have enough gold. After each purchase, the player's inventory is updated, and the cost is deducted from their gold.

The time complexity of the storeroom algorithm is 0(1) for the decision-making process (since it only handles a fixed list of items and prices), and O(1) for checking the player’s gold.

Data Structures:

ArrayList for Items: The ArrayList is used to store a list of items I have put in the game, I used to store 3 weapons and 3 armors in the store room to sell at the store and when the player goes to the treasureRoom it will randomly pick weapons form my list of weapons in my excel file I made. I used ArrayList to help make the choices for weapons and armor randomly easier and more efficient for both the storeRoom and the treasureRoom.

Character Class for Player/Enemy: A class Character is used to represent both the player and enemies. It stores all necessary attributes like health, attack power, and defense. Using this class structure allows for easy modification and addition of attributes for characters and enemies.

String Arrays for Riddles: A String[][] is used to store the riddles and answer for the Golem encounters. This structure worked great for looking up the riddles when in the room a lot easier and efficiently.

Opportunity Encountered:

When making the game, I first started with 6 random encounters which were the storeRoom, room for healing, TreasureRoom, EnemyEncounter, GolemRoom, and BossRoom. I liked them however I didn’t like how you encountered the boss, and it wouldn’t end my game, it kept looping. So, I thought what if I moved the BossRoom out of the random encounter and had them locked away until you found the key. I always had the Golem having riddles. I really liked that idea but didn’t give player chances if you got it wrong you fought the golem, So I decided I wanted to have it be riddles and lock the boss key in the room. I just didn’t like the golem to be a random encounter. I liked the idea of having him be in room locked until you get enough enemies defeated like a mini boss without battle. I felt like this help game had a goal to beat the boss and a puzzle to solve not just battle and it made me want to add more to it like traps and more puzzles are different in different rooms and getting to learn all this like an old dream come true, always want to make a game.

Error Encounter:

I thing I made a mistake on is not plan how much I wanted to do in this game once I really started working on it I got really involved in everything and I wanted to add different weapons with rarities system so I made a excel file that I started late and I had trouble getting to work I ran into some issues adding to the player or the other I wanted to add was Golem being locked in the room I didn’t know why the counter for enemies being defeated was never going pass 1. So I added and inventory system to the character which helped me keep track of not just weapons and armor of the character but also the health of the player how much potions they had, how much gold which helped me out on other problem when adding the weapons and armor to the treasure I accidently deleted the room giving the player gold and didn’t catch it for awhile until I added the store to have weapons and armor for sell and realizing I never had money. The other in my inventory I got to see what weapon and armor the player was given. Having the player be able to use potions outside of battle it was great. The last thing I added was the enemiesDeafeated counter which helped me realize that the counter would go back to 0 each time it would exit a different encounter room. It helped me resolve some issues. I still have more to do than I want to do but that exciting part of working on this I get to learn new things which I can’t wait to have more free time to work on this more and take I learn and make it 100 times better.

Future Enhancements:

**Weapons and armor upgrade**: I want to add status to my weapons and armor which it would increase the players attack and defense and started in my excel file but I realized when it might be to high of an increase especially when test defense for player and enemies having no damages taken because the defense was way to high, and then I want to add special effects to epic and legendary weapons.

**More Room types**: I want to do more puzzles that are different and not just riddles and add trapRooms to make it more changeling.

**Expand Rooms and mini-Bosses**: I want to have mini bosses where if players defeat them, you enter a new set of rooms to increase the level of the players and enemies each time it gets harder, and the player will need that higher-level equipment to survive.

**Story in the beginning and random events:** I want to introduce some character you can find with random events of a room that maybe can help you down the road.

**Class and level system**: I want to make different classes for the character you can pick and level that can increase with XP from fight.