

## **MINI PROJECT - 4**

### **(Castle of Shadows):**

#### **I. Problem Analysis and Statement:**

The problem is to implement a text-based adventure game with a storyline, character creation, room navigation, combat system, inventory management, and knowledge system. The game should allow players to explore a dark, abandoned castle, interact with its environment, and make choices that impact the game's outcome.

#### **II. Algorithm:**

1. **Game Initialization:** In this step, the game initializes the player's stats, inventory, knowledge, and game world. The player's stats include attributes such as health, strength, intelligence, dexterity, charisma, wealth, and experience. The inventory includes any items that the player has collected during the game. The knowledge includes any information that the player has learned about the game world. The game world includes the current state of the game world, such as the status of the village and the reputation of NPCs.
2. **Main Loop:** The main loop of the game is where the player is presented with a choice of actions, and the game processes their input and updates the game state accordingly. The loop continues until the game is over, which can happen if the player's health reaches zero or if the player achieves a certain goal.
3. **Character Creation:** In this step, the player is asked to create their character by entering their name and choosing a background. The background determines the player's starting stats and wealth.
4. **Room Navigation:** The game presents the player with a description of their current location, and allows them to navigate to other rooms by choosing a direction. Each room has its own description and set of choices.
5. **Combat System:** If the player encounters an enemy, the game enters the combat system. The combat system is turn-based, with the player and the enemy taking turns to attack, defend, or use items. The combat system continues until either the player or the enemy's health reaches zero.
6. **Inventory Management:** The player can view and manage their inventory at any time. The inventory includes any items that the player has

collected during the game, such as weapons, armor, potions, or keys. The player can use items from their inventory to overcome challenges or gain advantages.

7. Knowledge System: The knowledge system tracks any information that the player has learned about the game world. This information can be used to unlock new paths, solve puzzles, or make decisions.
8. Game Saving and Loading: The game allows the player to save and load their progress at any time. The game state is saved to a JSON file, which can be loaded later to resume the game.

The algorithm has a time complexity of  $O(n)$ , where  $n$  is the number of player choices. The space complexity is  $O(m)$ , where  $m$  is the number of game states. The algorithm is modular, with each component responsible for a specific aspect of the game. This makes the code more readable and maintainable.

### **III.Pseudo Code:**

```
DEFINE player1_stats AND player2_stats AS DICTIONARIES TO HOLD PLAYER  
INFORMATION
```

```
FUNCTION intro():
```

```
    PRINT "Welcome to the Multiplayer Adventure Game!"  
    CALL choose_characters()
```

```
FUNCTION choose_characters():
```

```
    PROMPT player1_stats['name'] TO ENTER PLAYER 1's NAME  
    PROMPT player2_stats['name'] TO ENTER PLAYER 2's NAME  
    CALL main_hall(player1_stats)
```

```
FUNCTION main_hall(player_stats):
```

```
    PRINT "You are in the main hall of a dark, abandoned castle."  
    PRINT "There are three doors: left, right, and straight ahead."  
    PROMPT player_stats['name'] TO CHOOSE A DOOR (left/right/straight)  
  
    IF choice IS 'left':
```

```

    CALL library(player_stats)
ELSE IF choice IS 'right':
    CALL armory(player_stats)
ELSE IF choice IS 'straight':
    CALL throne_room(player_stats)
ELSE:
    PRINT "Invalid choice, try again."
    CALL main_hall(player_stats)

FUNCTION library(player_stats):
    PRINT "You enter a dusty old library filled with ancient books."
    IF 'book_of_secrets' NOT IN player_stats['inventory']['items']:
        PRINT "There is a mysterious book on a pedestal."
        PROMPT player_stats['name'] TO TAKE THE BOOK (yes/no)

        IF choice IS 'yes':
            PRINT "You gain knowledge of the castle's secrets."
            ADD 'book_of_secrets' TO player_stats['inventory']['items']
        ELSE IF choice IS 'no':
            PRINT "You leave the book."
        ELSE:
            PRINT "Invalid choice, try again."
            CALL library(player_stats)
    ELSE:
        PRINT "You have already explored this room."

    CALL next_player(player_stats)

FUNCTION armory(player_stats):
    PRINT "You enter the armory. Weapons and armor line the walls."

```

IF 'sword' NOT IN player\_stats['inventory']['items']:

    PRINT "A sword catches your eye."

    PROMPT player\_stats['name'] TO TAKE THE SWORD (yes/no)

IF choice IS 'yes':

    PRINT "You take the sword."

    ADD 'sword' TO player\_stats['inventory']['items']

ELSE IF choice IS 'no':

    PRINT "You leave the sword."

ELSE:

    PRINT "Invalid choice, try again."

    CALL armory(player\_stats)

ELSE:

    PRINT "You have already taken the sword."

CALL next\_player(player\_stats)

FUNCTION throne\_room(player\_stats):

    PRINT "You enter the grand throne room. An imposing figure sits on the throne."

    PRINT "It's the Dark King, and he doesn't look pleased."

    PROMPT player\_stats['name'] TO FIGHT OR TALK TO THE DARK KING (fight/talk)

IF choice IS 'fight':

    CALL combat(player\_stats, DARK KING STATS)

ELSE IF choice IS 'talk':

    IF 'book\_of\_secrets' IN player\_stats['inventory']['items']:

        PRINT "Using the knowledge, you negotiate with the Dark King."

        PRINT "The Dark King leaves peacefully. You win!"

    ELSE:

        PRINT "The Dark King attacks you."

```

    CALL combat(player_stats, DARK KING STATS)
ELSE:
    PRINT "Invalid choice, try again."
    CALL throne_room(player_stats)

FUNCTION combat(player_stats, enemy):
    PRINT "You encounter the enemy!"
    WHILE player_stats['health'] > 0 AND enemy['health'] > 0:
        PRINT PLAYER AND ENEMY HEALTH
        PROMPT player_stats['name'] TO CHOOSE ACTION (attack/defend/use item)

        IF action IS 'attack':
            DEAL DAMAGE TO enemy['health']
        ELSE IF action IS 'defend':
            INCREASE player_stats['health']
        ELSE IF action IS 'use item':
            PROMPT FOR ITEM TO USE
            CALL use_item(player_stats, ITEM NAME)
        ELSE:
            PRINT "Invalid action, try again."

    IF enemy['health'] > 0:
        CALL enemy_attack(player_stats, enemy)
    ELSE:
        PRINT "Enemy defeated!"
        IF enemy IS DARK KING:
            PRINT "Congratulations! You saved the kingdom!"
        BREAK

FUNCTION enemy_attack(player_stats, enemy):

```

```
DEAL DAMAGE TO player_stats['health']  
IF player_stats['health'] <= 0:  
    PRINT "You have been defeated. Game Over."
```

```
FUNCTION use_item(player_stats, item_name):  
    IF item_name IN player_stats['inventory']['items']:  
        PRINT "You use the item."  
        APPLY ITEM EFFECTS  
        IF ITEM IS SINGLE USE:  
            REMOVE ITEM FROM INVENTORY  
    ELSE:  
        PRINT "Item not available."
```

```
FUNCTION next_player(current_player_stats):  
    IF current_player_stats IS player1_stats:  
        CALL main_hall(player2_stats)  
    ELSE:  
        CALL main_hall(player1_stats)
```

```
CALL intro()
```

#### **IV. Analysis:**

The game's design uses a modular approach, breaking down the game into smaller components that interact with each other. The use of functions and conditional statements makes the code more readable and maintainable.

The game's algorithm has a time complexity of  $O(n)$ , where  $n$  is the number of player choices. The space complexity is  $O(m)$ , where  $m$  is the number of game states.

The game's design can be improved by adding more features, such as:

- More rooms and areas to explore
- More enemies and combat mechanics

- More items and inventory management options
- More knowledge and puzzle mechanics
- A more engaging storyline and characters
- Better error handling and input validation
- More polished user interface and presentation

Overall, the game's design provides a solid foundation for a text-based adventure game, and can be expanded upon to create a more immersive and challenging experience for players.

## **V. Source Code :**

```
import random
```

```
import json
```

```
# Initial Game State
```

```
player_stats = {
```

```
    'health': 100,
```

```
    'strength': 10,
```

```
    'intelligence': 8,
```

```
    'dexterity': 8,
```

```
    'charisma': 8,
```

```
    'wealth': 50,
```

```
    'experience': 0
```

```
}
```

```
inventory = {
```

```
    'items': {
```

```
        'torch': {
```

```
            'description': 'A lit torch to see in the dark.',
```

```
            'usable': True
```

```
    }  
    },  
    'gold': 50  
}
```

```
knowledge = {  
    'castle_layout': False,  
    'book_of_secrets': False  
}
```

```
game_world = {  
    'village_status': 'peaceful',  
    'npc_reputation': {}  
}
```

# Game Functions

```
def intro():  
    print("Welcome to the Advanced Adventure Game!")  
    print("You find yourself in a dark forest, on the outskirts of an ancient  
castle.")  
    create_character()  
    main_hall()
```

```
def create_character():  
    print("\nCharacter Creation:")  
    name = input("Enter your character's name: ").strip()  
    print(f'Hello, {name}. Choose your background:')
```



```

background = input("Options: Noble, Thief, Scholar: ").strip().lower()

if background == 'noble':
    player_stats['charisma'] = 15
    player_stats['wealth'] = 100
elif background == 'thief':
    player_stats['dexterity'] = 15
    player_stats['wealth'] = 20
elif background == 'scholar':
    player_stats['intelligence'] = 15
    player_stats['wealth'] = 50
else:
    print("Invalid choice, defaulting to Scholar.")
    player_stats['intelligence'] = 15
    player_stats['wealth'] = 50

print(f'{name}, you have chosen the path of a {background.capitalize()}')

def main_hall():
    print("\nYou are in the main hall of a dark, abandoned castle.")
    print("There are three doors: left, right, and straight ahead.")
    choice = input("Which door do you choose? (left/right/straight)
").strip().lower()

    if choice == 'left':
        library()
    elif choice == 'right':
        armory()

```

```

elif choice == 'straight':
    throne_room()
else:
    print("Invalid choice, try again.")
    main_hall()

def library():
    print("\nYou enter a dusty old library filled with ancient books.")
    if not knowledge['book_of_secrets']:
        print("There is a mysterious book on a pedestal in the center of the room.")
        choice = input("Do you take the book? (yes/no) ").strip().lower()

        if choice == 'yes':
            print("The book glows and disappears! You've gained knowledge of the
castle's secrets.")
            knowledge['book_of_secrets'] = True
        elif choice == 'no':
            print("You leave the book and exit the library.")
        else:
            print("Invalid choice, try again.")
            library()
    else:
        print("You have already explored this room.")

    main_hall()

def armory():
    print("\nYou enter the armory. Weapons and armor line the walls.")

```

```

if 'sword' not in inventory['items']:
    print("A sword catches your eye.")
    choice = input("Do you take the sword? (yes/no) ").strip().lower()

    if choice == 'yes':
        print("You take the sword. It feels balanced and strong.")
        inventory['items']['sword'] = {
            'description': 'A sharp steel sword, perfect for combat.',
            'usable': True
        }
    elif choice == 'no':
        print("You leave the sword and exit the armory.")
    else:
        print("Invalid choice, try again.")
        armory()
else:
    print("You have already taken the sword.")

main_hall()

def throne_room():
    print("\nYou enter the grand throne room. An imposing figure sits on the throne.")

    print("It's the Dark King, and he doesn't look pleased to see you.")
    choice = input("Do you fight the Dark King or try to talk to him? (fight/talk) ").strip().lower()

    if choice == 'fight':

```

```

    combat({'name': 'Dark King', 'health': 100, 'strength': 20})
elif choice == 'talk':
    if knowledge['book_of_secrets']:
        print("Using the knowledge from the book, you negotiate with the Dark King.")
        print("He reveals his tragic backstory and decides to leave the castle peacefully.")
        print("Congratulations, you've won the game through diplomacy!")
    else:
        print("The Dark King is not interested in talking and attacks you.")
        combat({'name': 'Dark King', 'health': 100, 'strength': 20})
else:
    print("Invalid choice, try again.")
throne_room()

```

```

def combat(enemy):
    print(f"You encounter a {enemy['name']}!")
    while player_stats['health'] > 0 and enemy['health'] > 0:
        print(f"Player Health: {player_stats['health']}, {enemy['name']} Health: {enemy['health']}")
        action = input("Choose your action (attack/defend/use item): ").strip().lower()
        if action == 'attack':
            damage = player_stats['strength']
            enemy['health'] -= damage
            print(f"You deal {damage} damage to the {enemy['name']}")
        elif action == 'defend':
            print("You brace for the enemy's attack.")
            player_stats['health'] += 5 # Reduced damage or heal slightly

```

```

elif action == 'use item':
    item = input("Which item do you want to use? ").strip().lower()
    use_item(item)
else:
    print("Invalid action, try again.")

if enemy['health'] > 0:
    enemy_attack(enemy)
else:
    print(f"You have defeated the {enemy['name']}!")
    if enemy['name'] == 'Dark King':
        print("Congratulations! You have saved the kingdom!")
    break

def enemy_attack(enemy):
    damage = enemy['strength']
    player_stats['health'] -= damage
    print(f"The {enemy['name']} attacks you for {damage} damage.")
    if player_stats['health'] <= 0:
        print("You have been defeated. Game Over.")

def use_item(item_name):
    if item_name in inventory['items'] and
inventory['items'][item_name]['usable']:
        print(f"You use the {item_name}.")
        if item_name == 'torch':
            print("You use the torch to light up the dark room.")
        elif item_name == 'potion':

```

```

        player_stats['health'] += 20
        print("You drink a health potion and restore 20 health.")
        del inventory['items']['potion']
    else:
        print(f"You can't use the {item_name} or it doesn't exist.")

def check_inventory():
    print("Inventory:")
    for item, details in inventory['items'].items():
        print(f"{item.capitalize()}: {details['description']}")
    print(f"Gold: {inventory['gold']}")

def save_game(filename="savegame.json"):
    game_state = {
        'player_stats': player_stats,
        'inventory': inventory,
        'knowledge': knowledge,
        'game_world': game_world
    }
    with open(filename, 'w') as file:
        json.dump(game_state, file)
    print("Game saved successfully.")

def load_game(filename="savegame.json"):
    global player_stats, inventory, knowledge, game_world
    try:
        with open(filename, 'r') as file:

```

```
game_state = json.load(file)

player_stats = game_state['player_stats']

inventory = game_state['inventory']

knowledge = game_state['knowledge']

game_world = game_state['game_world']

print("Game loaded successfully.")

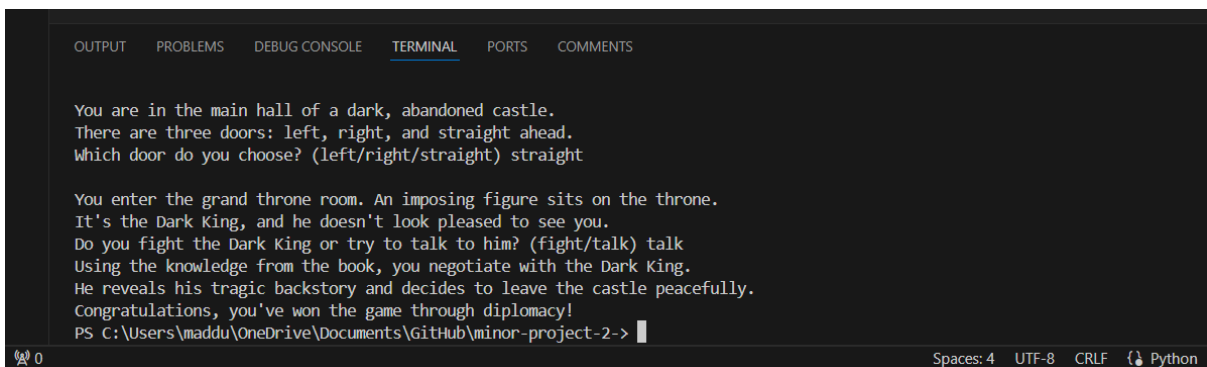
except FileNotFoundError:

    print("Save file not found.")
```

# Start Game

intro()

## **VI.Output:**



```
OUTPUT  PROBLEMS  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

You are in the main hall of a dark, abandoned castle.
There are three doors: left, right, and straight ahead.
Which door do you choose? (left/right/straight) straight

You enter the grand throne room. An imposing figure sits on the throne.
It's the Dark King, and he doesn't look pleased to see you.
Do you fight the Dark King or try to talk to him? (fight/talk) talk
Using the knowledge from the book, you negotiate with the Dark King.
He reveals his tragic backstory and decides to leave the castle peacefully.
Congratulations, you've won the game through diplomacy!
PS C:\Users\maddu\OneDrive\Documents\GitHub\minor-project-2-> |
```

Spaces: 4 UTF-8 CRLF Python