Text based Escape game (Horror game)

Submitted By: ***Arman ali*** *160923733313*

Mohd Ibrahim ali *160923733314*

Mohd Sohail *160923733315*

Omer bin Zameer *160923733316*

Shaik Ahmed *160923733318*

Institution: *Lords Institute of Engineering and Technology*

Guide**:** Mohd Yakub

2. Table of Contents

---

3. Introduction

This report presents the development of a **text-based horror adventure game** using **Python** and **Tkinter** as the GUI (Graphical User Interface) framework. The game immerses players in an eerie journey where they must make critical decisions to navigate through a series of haunted locations, solve challenges, and survive. Sound effects, images, and interactive choices enhance the user experience, making the game engaging and thrilling.

---

4. Objectives

The main goals of this project are:

1. To create an interactive horror-themed game with a graphical interface.
2. To develop decision-based gameplay with multiple scenes and outcomes.
3. To implement inventory management and health tracking.
4. To integrate sound effects and visuals for an immersive experience.
5. To improve programming and problem-solving skills using Python.

---

5. Tools and Technologies Used

1. **Programming Language**: Python 3.10+
2. **GUI Framework**:
    - Tkinter (Standard GUI library for Python)
    - CustomTkinter (for a modern, customizable interface)
3. **Audio Library**: Pygame (for sound effects and background music)
4. **Image Library**: Pillow (PIL) for handling and resizing images

5. **Random Module**: To add randomness to game events
6. **IDE/Editor**: [e.g., PyCharm, Visual Studio Code]

---

6. Design and Features

The game follows a **scene-based design** where the player progresses by choosing options in each scene. Key features include:

1. **Main Menu**:
   - Welcome screen with buttons to start the game or view instructions.
2. **Scene-Based Gameplay**:
   - Locations include a graveyard, haunted house, abandoned mine, forest, and old mansion.
   - Each scene offers choices like searching for items, fighting enemies, or escaping.
3. **Inventory System**:
   - Players can collect items like a map, key, lantern, pickaxe, and a magical amulet, which help in progression or combat.
4. **Health Management**:
   - The player has 100 health points. Choices like fighting enemies can reduce health.
   - Losing all health results in "Game Over."
5. **Sound Effects and Music**:
   - Background music plays continuously to create a tense atmosphere.
   - Event-specific sounds (e.g., fighting, escaping, game over) enhance the immersion.
6. **Random Events**:
   - Randomized outcomes (e.g., escaping or encountering enemies) keep the game unpredictable and exciting.
7. **Endings**:
   - The game includes multiple endings such as victory, defeat, or being trapped, depending on player choices.

---

7. Code Implementation

The implementation follows a modular design for readability and reusability:

Code Structure:

- **Main Class**: `App(ct.CTk)` initializes the game window and manages all scenes.
- **Scene Management**: `show_scene()` dynamically switches between different game scenes.
- **GUI Elements**: Buttons and labels are created dynamically on a Tkinter canvas.
- **Sound Integration**: Using `pygame.mixer.Sound()` for sound effects and background music.

- **Inventory and Health**: Lists and variables manage the player's items and health points.

Key Code Snippets:

1. **Scene Transition**:

```
def show_scene(self, scene):
    self.clear_scene()
    if scene == "main_menu":
        self.show_main_menu()
    elif scene == "graveyard":
        self.show_graveyard()
```

2. **Inventory Management**:

```
def take_items_graveyard(self):
    self.inventory.extend(["key", "map"])
    self.show_scene("haunted_house")
```

3. **Health Reduction**:

```
def fight_beast(self):
    if "amulet" in self.inventory:
        self.show_scene("old_mansion")
    else:
        self.health -= 30
        if self.health <= 0:
            self.show_scene("game_over")
```

4. **Sound Playback**:

```
pygame.mixer.Sound("path_to_sound.mp3").play()
```

---

8. Testing and Results

The game was tested for the following aspects:

1. **Functionality**: All buttons, events, and scenes operate correctly.
2. **Inventory Management**: Items are stored and utilized as expected.
3. **Health System**: Health decreases properly during battles and events.
4. **Sound Effects**: Audio integrates seamlessly without delays.
5. **User Experience**: The gameplay is engaging, with responsive UI elements and immersive sounds.

Test Results: The game performs as intended, providing an enjoyable and challenging experience.

9. Challenges and Solutions

Challenges	Solutions
Integrating sound effects and music	Used `pygame.mixer` for efficient sound handling.
Managing dynamic scenes	Implemented modular scene functions and clearing methods.
Implementing inventory and health	Used lists for inventory and variables for health tracking.
Adding randomness for events	Used `random.randint` and `random.choice` for unpredictability.

10. Conclusion

The project successfully achieved its goals of developing an **interactive horror adventure game** using Python. The game combines an engaging storyline, sound effects, and visuals to create an immersive experience. It also demonstrates the effective use of Python's GUI capabilities and highlights the importance of modular design.

11. Future Improvements

1. Adding more scenes and story branches for replayability.
2. Incorporating animations for a more dynamic visual experience.
3. Implementing a save and load game feature.
4. Introducing a multiplayer or competitive mode.
5. Enhancing visuals with advanced libraries like `Pygame` or `Tkinter Canvas animations`.