Choose Your Own Adventure Game In C++

By: William Newman

What is this game?

- A text-based adventure game built in C++ using object-oriented principles.
- Player navigates through a room by inspecting walls, collecting items, and interacting with objects.
- The game is time-sensitive, with a 3-minute countdown to escape.
- The project demonstrates concepts like inheritance, polymorphism, file handling, and more.

Overview of the Key Classes

- AdventureGame Class – The main controller that drives the game logic and user interactions.
- Inventory Class – Manages the player's inventory and handles item storage.
- Wall Class – Represents the walls in the room, with descriptions, items, and the ability to inspect or open doors.

AdventureGame Class Overview

Purpose: Manages the game loop, user input, and game logic.

Key Methods:

- start(): Starts the game and contains the main game loop.
- displayRoom(): Displays the current status of the game.
- showCurrentWall(): Displays the wall ASCII art based on the current wall.
- openLobby(), openStairwell(): Handle actions when the player chooses to open doors.
- getTimeRemaining(): Calculates the remaining time in the game.

Wall Class Overview

Purpose: Represents the walls in the room. Each wall can have an item, a description, and may have a door that can be unlocked with a key.
Key Methods:

- inspect(): Allows the player to examine the wall and find items.
- useItem(): Allows the player to use an item on the wall if it’s a door that requires a key.
- showAsciiArt(): Displays ASCII art based on the wall’s description.

Inventory Class Overview

Purpose: Manages the player's inventory, keeping track of items found throughout the game.

Key Methods:

- addItem(): Adds an item to the inventory.
- hasItem(): Checks if a certain item exists in the inventory.
- display(): Displays the current inventory to the player.
- saveToFile() and loadFromFile(): Save and load the inventory to/from a file.

User Interaction Flow

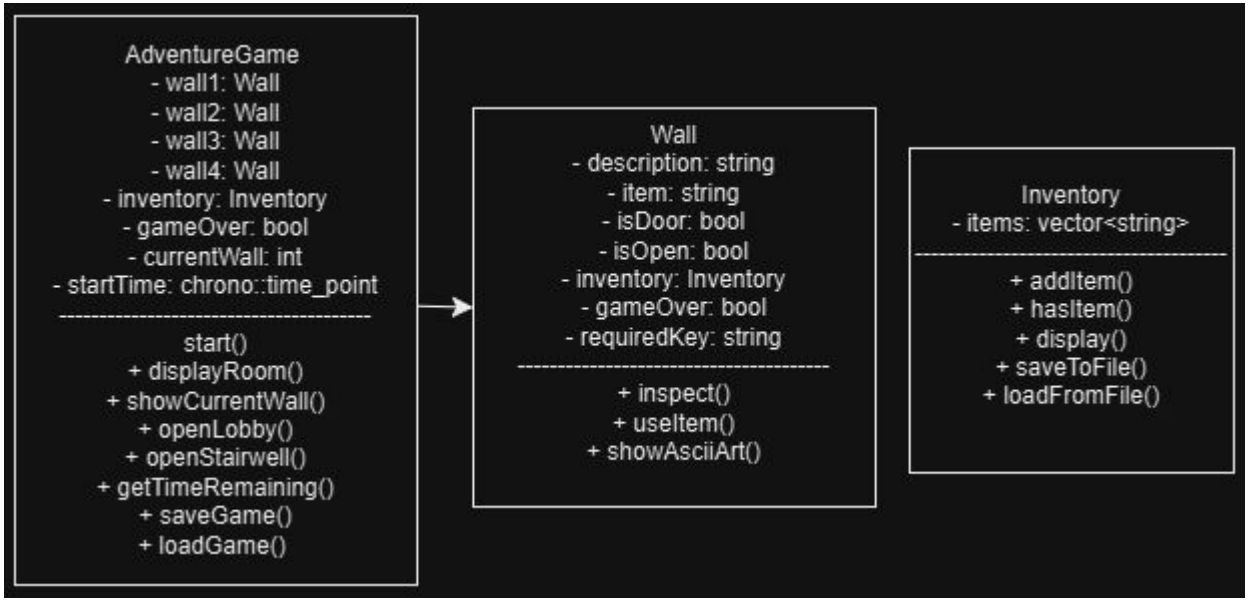
- Players can choose actions such as:
    - l: Turn left
    - r: Turn right
    - w: Inspect the current wall
    - i: View inventory
    - e: Open a door (if available)

- Interaction based on game state and the player's inventory (e.g., requiring keys to open doors).

Game Loop and Time Management

- Game Loop: Continuously prompts the player for actions and updates the game state until the game ends (either by escaping or running out of time).
- Time Management:
    - The game has a 3-minute countdown (represented as getTimeRemaining()).
    - If the time runs out, the game ends with a message: "Time's up! You didn't escape in time."

UML Diagram


AdventureGame
- wall1: Wall
- wall2: Wall
- wall3: Wall
- wall4: Wall
- inventory: Inventory
- gameOver: bool
- currentWall: int
- startTime: chrono::time_point
start()
+ displayRoom()
+ showCurrentWall()
+ openLobby()
+ openStairwell()
+ getTimeRemaining()
+ saveGame()
+ loadGame()
Wall
- description: string
- item: string
- isDoor: bool
- isOpen: bool
- inventory: Inventory
- gameOver: bool
- requiredKey: string
+ inspect()
+ useItem()
+ showAsciiArt()
Inventory
- items: vector<string>
+ addItem()
+ hasItem()
+ display()
+ saveToFile()
+ loadFromFile()

Conclusion

- The AdventureGame project is a great example of applying object-oriented design principles in C++.
- It uses multiple classes to structure the game, manage player interactions, and handle inventory and time.
- This project demonstrates key C++ skills such as classes, objects, inheritance, file handling, and more.