A mind’s Journey. Game documentation

Introduction

"A Mind's Journey" is a text-based adventure game that invites the player on a virtual exploration quest. The player navigates through different rooms, collects items, and uses these items to solve a final puzzle. The storyline begins with the start room, from which the player will navigate through the map, gathering the items that symbolize a person’s emotional and mental growth, to finally solve the puzzle at the end and win the game. There is a shortcut for the puzzle room. But it is necessary to collect the items in order to activate it. And so, the journey.

Game Commands

Players can interact with the game world through five specific commands: "look", "move", "get", "inventory", and "map".

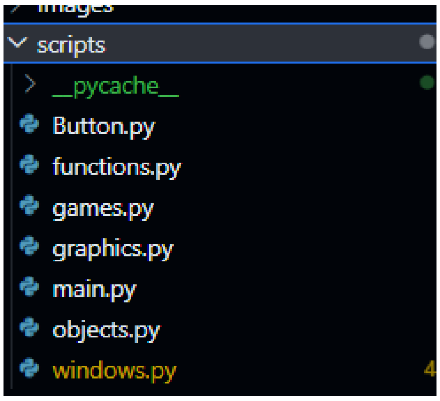
1. "look": Displays a description of the scenery in the chosen direction (north, south, east, west, up, down).
2. "move": Allows the player to navigate between rooms by specifying a direction.
3. "get": Enables the player to collect an item in a room by naming the item.
4. "inventory": Displays the items that the player has collected so far.
5. "map": Shows a window with an image of the map.

Goal and Score System

The player's goal is to collect all items and then solve the final puzzle, which requires pressing the item buttons in the correct order. The scoring system is based on the number of attempts made to solve the puzzle, with the lowest possible score (and thus the best) being 1.

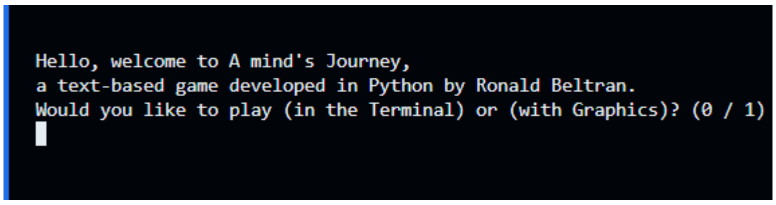
Design

The design of "A Mind's Journey" is based on a structure of modularized functions which encapsulate different functionalities of the game, making it easier to manage, debug and scale the project.



```
scripts
  __pycache__
  Button.py
  functions.py
  games.py
  graphics.py
  main.py
  objects.py
  windows.py
```

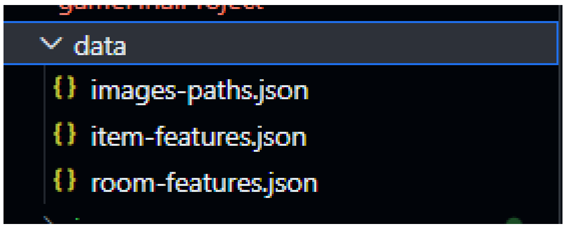
At the highest level, the `main` function acts as the entry point to the program. This function starts the user interaction and leads the player to choose between a terminal-based interface or a graphical interface. Based on the player's choice, either `run_game_in_terminal` or `run_game_with_graphics` is executed.



```
Hello, welcome to A mind's Journey,
a text-based game developed in Python by Ronald Beltran.
Would you like to play (in the Terminal) or (with Graphics)? (0 / 1)
```

The game data is stored in a JSON file and is accessed using the `get_data` function. This design allows the game to be easily extended with additional rooms, items, or attributes by simply modifying the JSON file.

The game logic is controlled via functions like `run_game_in_terminal` and `run_game_with_graphics`, which execute a loop until the game ends. These functions manage the game state, handle user input, update the display, and control game progression. The game state includes variables such as the player's current room, inventory, and score.



```
data
  images-paths.json
  item-features.json
  room-features.json
```

For the graphical version, the game utilizes the `graphics.py` module for window management, drawing shapes, and handling mouse interactions. The interface contains texts to guide the player, buttons for interaction, and an inventory bar where collected items are displayed.

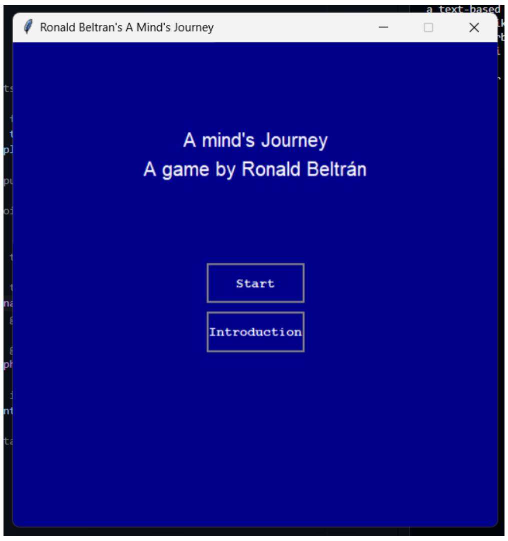
The puzzle portion of the game is designed as a function `puzzle_with_graphs` that creates a new window with items buttons for the player to interact with. The order of these items is checked when the player hits the enter button. If the correct order of items is selected, the player proceeds to the `final_window` function displaying the completion message and player's score.

In summary, "A Mind's Journey" is designed in a structured, modular fashion to allow for efficient code management and potential expansion of the game. The design incorporates both terminal-based and graphical interactions, offering a versatile player experience.

Windows

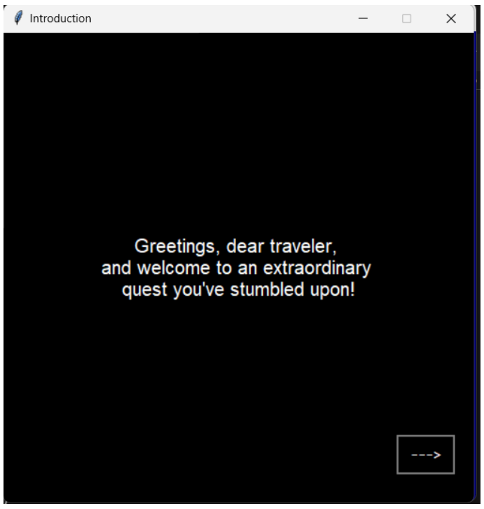
Menu Window

The menu window is the first window that players encounter when they start the game. It serves as the gateway to the gaming experience. The window asks players whether they prefer to play the game or read the introduction and storyline of the game.



Introduction Window

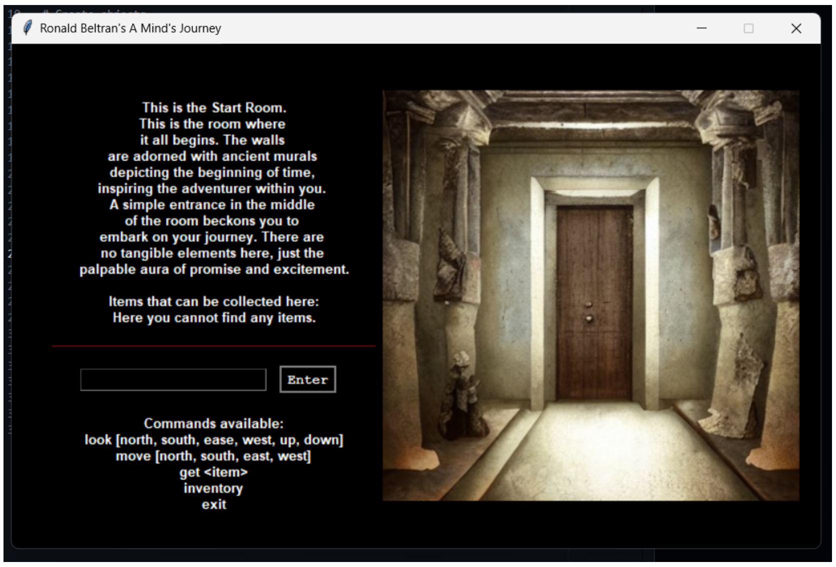
This window explains the premise of the game, the controls, and the game's objectives, providing essential context before the journey begins.



Greetings, dear traveler,
and welcome to an extraordinary
quest you've stumbled upon!

Interface Window

The interface window serves as the main window throughout the game when the player chooses to play with graphics. It dynamically updates to reflect the current room description, the player's inventory, and available directions for movement. It is where the player interacts with the game, inputting commands and receiving responses.



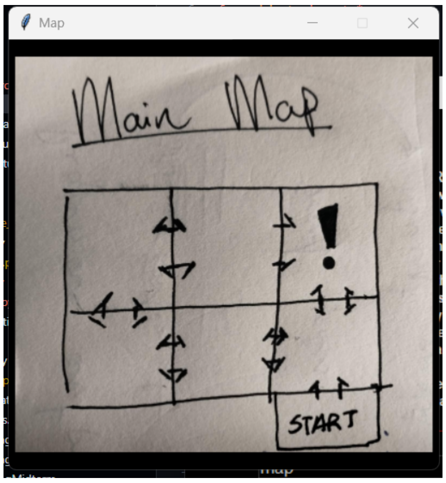
This is the Start Room.
This is the room where
it all begins. The walls
are adorned with ancient murals
depicting the beginning of time,
inspiring the adventurer within you.
A simple entrance in the middle
of the room beckons you to
embark on your journey. There are
no tangible elements here, just the
palpable aura of promise and excitement.

Items that can be collected here:
Here you cannot find any items.

Commands available:
look [north, south, ease, west, up, down]
move [north, south, east, west]
get <item>
inventory
exit

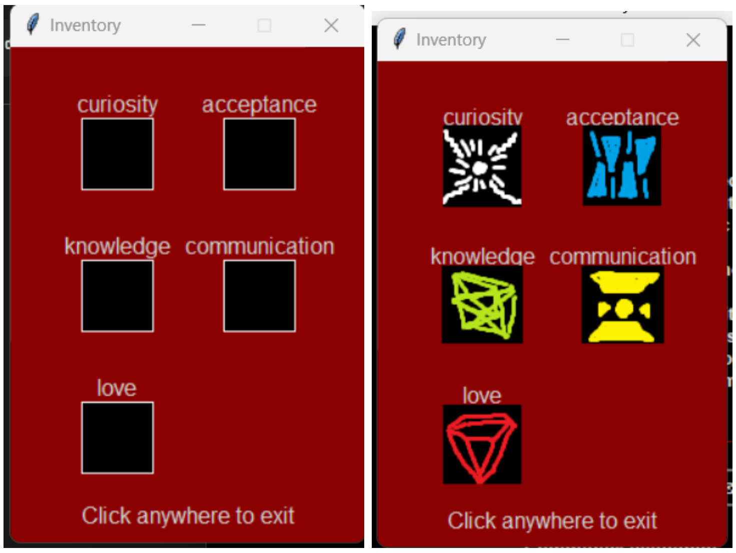
Map Window

The map window can be invoked by the player using the 'map' command. It provides a graphical representation of the game layout.



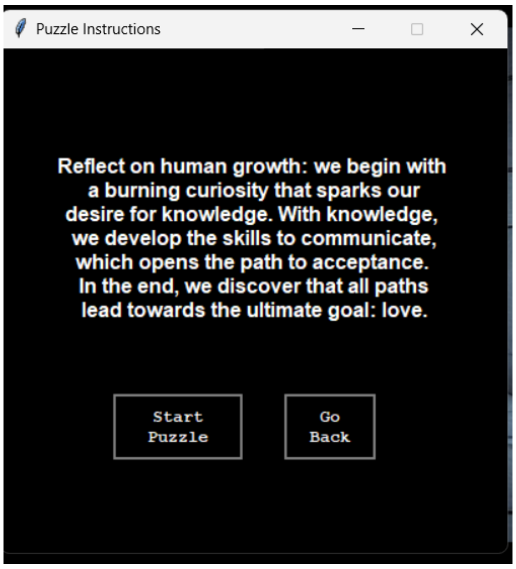
Inventory Window

The inventory window is a special window that can be accessed by using the 'inventory' command. It displays the items that the player has collected so far, representing each item with a square and its name. Once the player has collected an item, an image of that item appears in the corresponding square.



Pre-puzzle Window

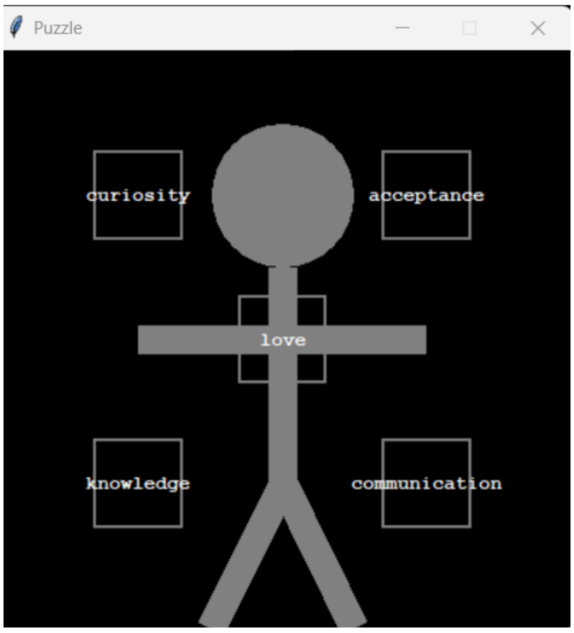
The window displays an informative hint to help the player solve the puzzle. Alongside the hint, it features two buttons - one to start the puzzle and the other to return to the previous room, giving the player a choice on how to proceed. By providing a clear hint and straightforward options, this window serves as a vital transition point, helping the player shift from exploration to problem-solving.


Puzzle Instructions

Reflect on human growth: we begin with a burning curiosity that sparks our desire for knowledge. With knowledge, we develop the skills to communicate, which opens the path to acceptance. In the end, we discover that all paths lead towards the ultimate goal: love.

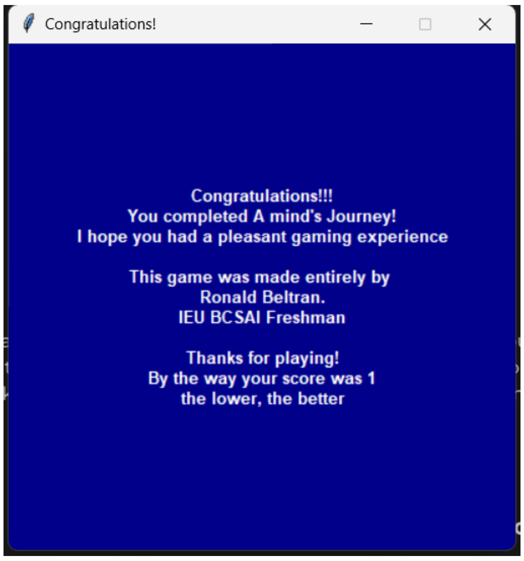
Puzzle Window

The puzzle window is enabled once the player has collected all the items. This window presents the central challenge of the game: a puzzle where the player has to press the buttons of the items in the correct sequence. In the graphical version of the game, these buttons are interactive, and the player can click on them.



Final Window

The final window appears after the player has successfully solved the puzzle. It displays a congratulatory message and reveals the player's final score. The window also thanks the player for their participation, bringing the game to a close.


Congratulations!

Congratulations!!!
You completed A mind's Journey!
I hope you had a pleasant gaming experience

This game was made entirely by
Ronald Beltran.
IEU BCSAI Freshman

Thanks for playing!
By the way your score was 1
the lower, the better

Pseudocode

1. Start the game and ask the player for their preferred mode (terminal or graphics).
2. Based on the selected mode, initialize the game environment.
3. Load the game data (room and item descriptions) using the 'get_data' function.
4. While the game is not over, listen for player commands and respond accordingly:
   a. If the command is "look", display the appropriate direction description.
   b. If the command is "move", change the current room.
   c. If the command is "get", add the item to the inventory if available in the room.
   d. If the command is "inventory", display the current inventory.
   e. If the command is "map", show the game map.
5. If all items are collected, enable the puzzle.
6. Implement a puzzle function based on the game mode:
   a. In terminal mode: Ask the player to enter the sequence of items, then check if the sequence is correct.
   b. In graphics mode: Wait for the player to press the item buttons in a specific order, then check if the sequence is correct.

Implementation

The implementation of the game "A Mind's Journey" entails writing Python 3.10 or higher, and scripts that translate the logical structure and rules outlined in

the pseudocode into a working program. Here's how the various sections of the pseudocode were implemented.
The libraries implemented are time and json, along with the modules inside.

1. Setting Up: The initial part of the game includes importing the necessary Python modules, defining classes and functions, and initializing the game state. For instance, the `Player` and `Button` classes are defined to represent the player and interactive buttons in the game. The `get_data` function is implemented to extract information from JSON files containing game data.

2. Main Function: The `main` function is implemented as the entry point of the program. This function prompts the user to select either the terminal or graphical version of the game and calls the appropriate function (`run_game_in_terminal` or `run_game_with_graphics`) based on the user's choice.

3. Game Loop: Both `run_game_in_terminal` and `run_game_with_graphics` functions implement a game loop, which continues running until the game is over. This loop involves processing user input, updating the game state, and redrawing the screen.

4. User Interaction: User commands are processed in the game loop. In the terminal version, user input is obtained using Python's `input` function. In the graphical version, the `graphics.py` module's functionality is used to handle mouse clicks on various elements.

5. Puzzle: The `puzzle_with_graphs` function implements the puzzle part of the game. It creates a new graphical window with item buttons, handles user interactions with these buttons, checks if the items are placed in the correct order, and accordingly updates the game state.

6. Game Conclusion: The `final_window` function is used to display a message upon successful completion of the game. It presents the player's score and concludes the game experience.

7. Graphics and Visual Elements: The graphical elements of the game, such as windows, texts, and buttons, are implemented using the `graphics.py` module. Image files for items are loaded and displayed using the `Image` class from the same module.

Throughout the implementation process, Python's object-oriented programming (OOP) principles are followed to encapsulate game elements into classes and functions, promoting code reusability and maintainability.