Maze Game

Devangam Kishan teja

April 28, 2024

1 Introduction

The Maze Game depends on functions and logical operations to function. In this report, we'll look into the different functions and logical details used in the game.

2 Classses

The code for the Maze Game employs classes to organize and manage different elements within the game. These classes define essential components such as cells in the maze, the labyrinth itself, buttons for user interaction, and the game's main functionality.

2.1 Labyrinth

The Labyrinth class is responsible for generating the maze using Prim's algorithm and managing the maze's overall structure. It contains methods for maze generation, adding red cells (obstacles) to the maze, drawing the maze on the screen with restricted view, and handling various game functionalities such as level selection and scoring. These classes encapsulate the game's core logic, allowing for modular and organized code structure.

2.2 Cell

The Cell class represents individual cells within the maze, each containing information about whether it has been visited and the walls surrounding it. This class facilitates the creation and manipulation of maze cells during maze generation.

2.3 Button

The Button class in the code for the Maze Game serves as a fundamental element for user interaction. This class enables the creation of clickable buttons that trigger specific actions when activated by the user, such as starting the game, accessing help or high scores, or returning to the main menu.

3 Windows

windows are graphical interfaces that facilitate user interaction and display essential information throughout the gameplay experience. These windows serve as entry points to various functionalities and provide users with options to navigate the game, access help, view high scores, select game levels, and handle game endings.

3.1 Opening Window

This window appears when the game is launched and presents the main menu options to the player. It includes buttons for starting the game, accessing help, viewing high scores, and quitting the game.

3.2 Level Selection Window

This window allows the player to choose from different game levels with varying difficulties. It offers buttons representing each level, enabling the player to select their preferred level of challenge.

3.3 Help Window

The Help window provides guidance and instructions to the player on how to play the game effectively. It displays information about movement controls, gameplay objectives, scoring mechanics, and other essential tips.

3.4 High Scores Window

This window displays the top high scores achieved by players in previous game sessions. It offers a leaderboard-style view of the highest-scoring players, motivating users to strive for better performance.

3.5 Ending window

The Ending Screen appears when the game session concludes, either due to victory or defeat. It displays relevant information such as the player's score, option to return to the main menu, and the ability to quit the game.

4 Main function

This document provides a description of the main function in the Python script. The main function controls the flow of the program and orchestrates the gameplay loop of a maze game.

4.1 Function Overview

The main function is responsible for managing various aspects of the game, including player movement, collision detection, victory conditions, and game over scenarios.

4.2 Function Execution

The function begins by calling the `opening_window` function to prompt the player to select a game level. If the player chooses a level, the function initializes game parameters such as player position, timer duration, and maze layout based on the selected level.

4.3 Game Loop

The main function enters a game loop where it continuously checks for player input and updates the game state accordingly. It handles player movement using keyboard input and detects collisions between the player character and maze walls.

4.4 Victory and Game Over

If the player reaches the goal within the time limit, the function sets the victory flag to true and records the completion time. If the player runs out of lives or time, the function triggers the game over condition and ends the game.

5 Conclusion

In conclusion, the Maze Game employs a sophisticated combination of functions, logical operations, and graphical interfaces to deliver an engaging and immersive gaming experience. By leveraging object-oriented programming principles and intuitive user interface design, the game offers players a challenging yet enjoyable journey through intricate mazes and strategic gameplay.

References

1. `https://www.pygame.org/tags/maze`