Data Structures and Algorithms-Lab

Quest of Shadows

Project Report

Section: B

Submitted by:

Anza Tamveel 2023-CS-87

Amna Atiq 2023-CS-83

Zainab Batool 2024-CS-89/2023-CS-652

Submitted to:

Prof. Nazeef-ul-Haq

15th December 2024

Department of Computer Science

University of Engineering and Technology, Lahore.

Table of Contents

Tables

Figures

1. Overview

1.1 Description

The Maze Game project is designed to demonstrate the practical application of Data Structures and Algorithms (DSA) in game development. This project focuses on generating dynamic mazes and solving them using efficient algorithms. The game challenges players to navigate through a randomly generated maze, offering an amazing game experience. Key concepts such as graph, trees, DFS, BFS, link list, stack, queue and pathfinding using traversing algorithms are integral to the project's design and implementation.

1.3 Motivation

The primary motivation for this project stems from the need to create an engaging way to learn and apply DSA concepts. Maze generation and solving are excellent case studies for exploring algorithms such as Depth-First Search (DFS), Breadth-First Search (BFS). These algorithms have real-world applications, and implementing them in a visual, interactive game makes the learning process more intuitive and enjoyable. Additionally, the project allows for creative problem-solving and challenges typical of game design.

1.4 Audience

This project is targeted at:

- Students and educators in computer science looking to learn or teach DSA concepts interactively.
- Game enthusiasts interested in understanding the technical aspects of maze generation and solving.
- Developers seeking a foundational framework for applications involving pathfinding.

1.5 Features

- A **Depth-First Search (DFS)** algorithm is used to create a randomized maze structure by connecting cells through a stack-based traversal. Each vertex is connected to its neighbors based on randomly shuffled directions, ensuring the maze is solvable.
- Visualization of algorithms like **BFS** and **DFS**.
- Interactive gameplay allowing users to manually solve the maze.
- Basic **AI** is integrated to **dynamically** identify and prioritize objects in the maze based on proximity. The system calculates the shortest path to the nearest enemy, then treasures, and finally the win door using **BFS**. This simulates intelligent decision-making by the player, ensuring an optimized traversal sequence.
- Different maze levels and increasing **complexity** as player moves further.

- User-friendly graphical interface with keyboard controls.
- User of game music to attract the player.

- DFS uses a **stack** (either explicitly or implicitly via recursion) to explore deeper into the maze, backtracking when dead ends are encountered.
- BFS employs a **queue** to explore nodes level by level, ensuring the shortest path between two points is found.
- Dynamic levels' generation.
- A **hash table** (dictionary in Python) is used to store and manage the positions of dynamic objects within the maze. Each key-value pair maps a node index to the type of object (e.g., enemy, treasure, or win), enabling efficient lookups and categorization.

1.6 Tools and Technologies

Table 1. tool and technology

Programming Language	Python
Libraries	Pygame for graphical interface, time, sys, random
Data structures	Stack, queue, graph, tree, link-list, hash table
Algorithms	DFS, BFS
Environment	Visual Studio Code

1.7 Class Details

1.7.1 Maze

- Represents the main maze structure using a graph where each cell is a vertex.
- Responsible for generating the maze using **DFS**, managing the adjacency list for graph representation, and dynamically finding paths to objects.

Key Methods:

- **generate_dfs_maze():** Creates the maze using DFS.
- **find_path():** Computes the shortest path between two points using BFS.
- **get_dynamic_objects_positions():** Prioritizes and finds paths to dynamic objects based on proximity.

1.7.2 Hurdles

- Manages dynamic objects in the maze, such as enemy, treasure, and win.
- Stores objects in a hashtable for efficient lookup.
- **Key Methods:**
  - **add_object():** Adds a new object to the maze.
  - **remove_object():** Removes an object after interaction.

1.7.3 Player

- Represents the player's current position in the maze.
- Handles player movement and updates the position dynamically.
- **Key Attributes:**
  - **x and y:** Player's current coordinates.

1.7.4 Bullet

- Represents a projectile fired by the player or enemies. The Bullet class handles movement, collision detection, and interactions with walls, the player, or enemies

Key Attributes:

- **x, y:** Current position of the bullet.
- **target_x, target_y:** Target position the bullet moves toward.
- **speed:** Speed of the bullet.
- **distance:** Distance to the target for movement calculation.
- **has_reached_target:** Indicates if the bullet has reached its destination.

Key Methods:

- **update(delta_time):** Updates the bullet's position towards the target.
- **draw(screen):** Draws the bullet on the screen if it's active.
- **check_collision_with_wall(game):** Checks if the bullet collides with walls.
- **check_collision_with_player(player, game):** Checks if the bullet hits the player.
- **check_collision_with_enemy(enemies):** Checks if the bullet hits an enemy and updates their hit count.

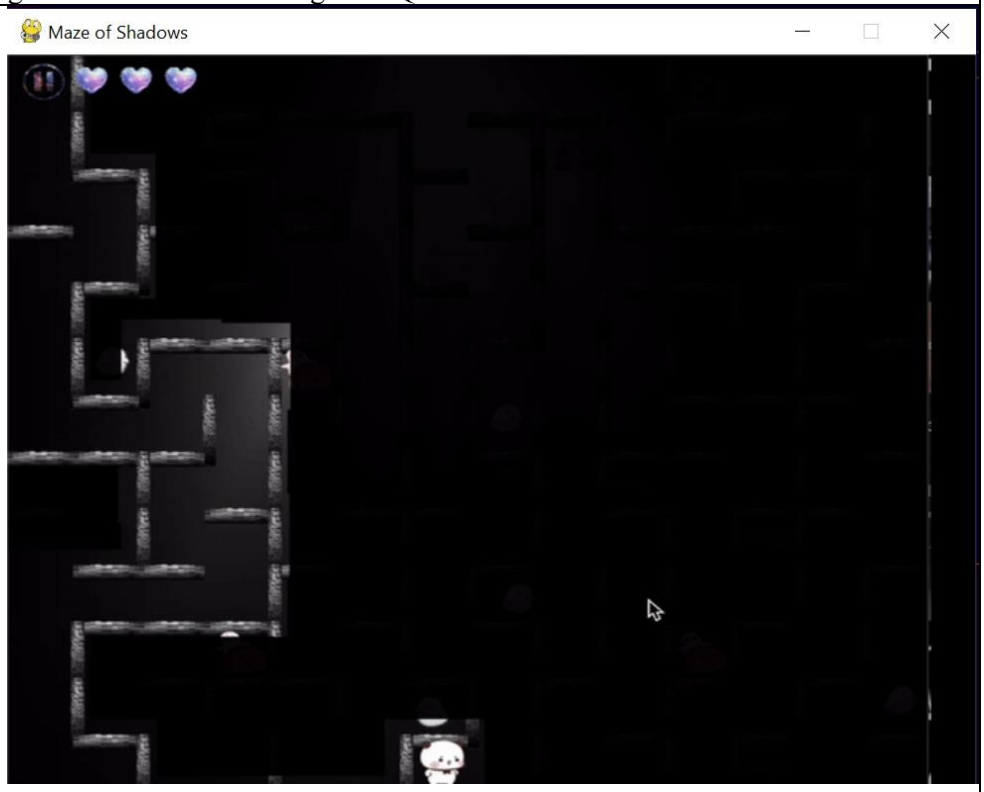
1.7.5 Game

- Manages the overall gameplay loop and integrates the maze, player, and hurdles.
- Implements logic for handling user input, updating the game state, and detecting win conditions.

2 Use Cases

2.1 Home Screen

Table 2. home screen

USE CASE	1
NAME	Home Screen
ACTOR	User
DESCRIPTION	It displays the home screen. Game starts and here we see the animation of the game and the name if the game “Quest of Shadows”.
FIGURE	 *Figure 1. home animation*

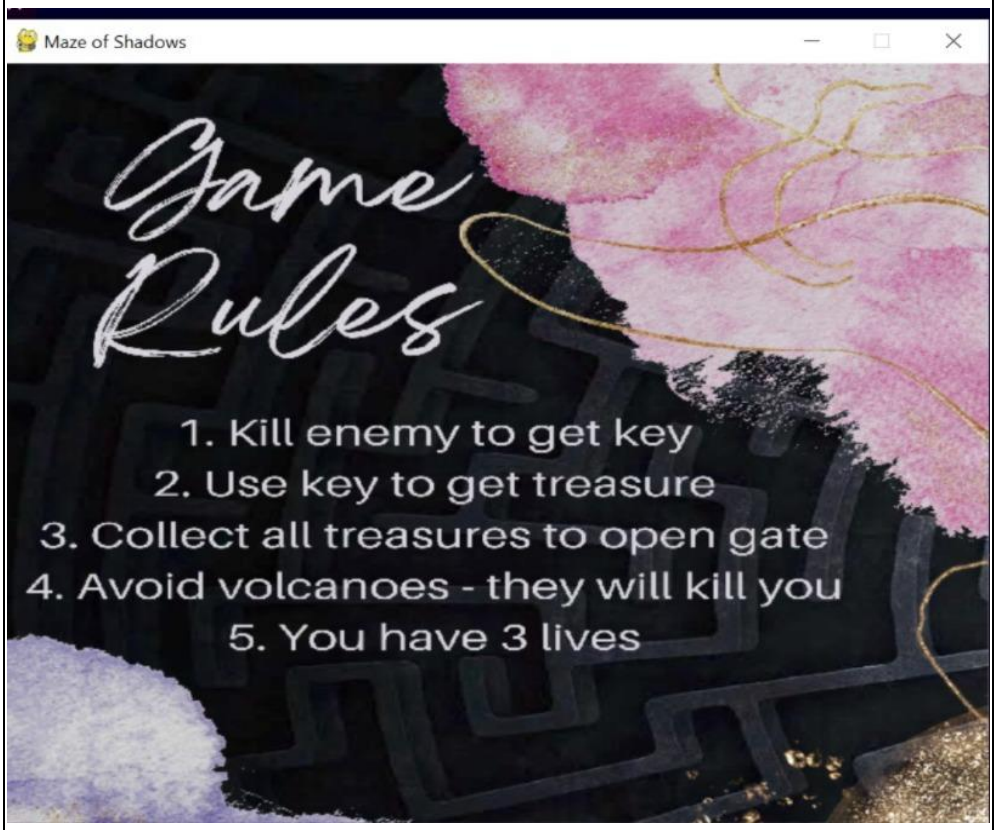
2.2 Menu of the game

Table 3 menu of the game

USE CASE	2
NAME	Menu
ACTOR	User
DESCRIPTION	It displays the menu which shows the title of the game, play button, rules and quit option. User can select any choice to proceed.
FIGURE	 *Figure 2. menu*

2.3 Rules Of The Game

Table 4 rules of the game

USE CASE	3
NAME	Rules
ACTOR	User
DESCRIPTION	It displays the rules of the game. If the player do not know the game rules, he can easily excess the option “rules”.
FIGURE	 *Figure 3. rules*

2.4 Levels Of The Game

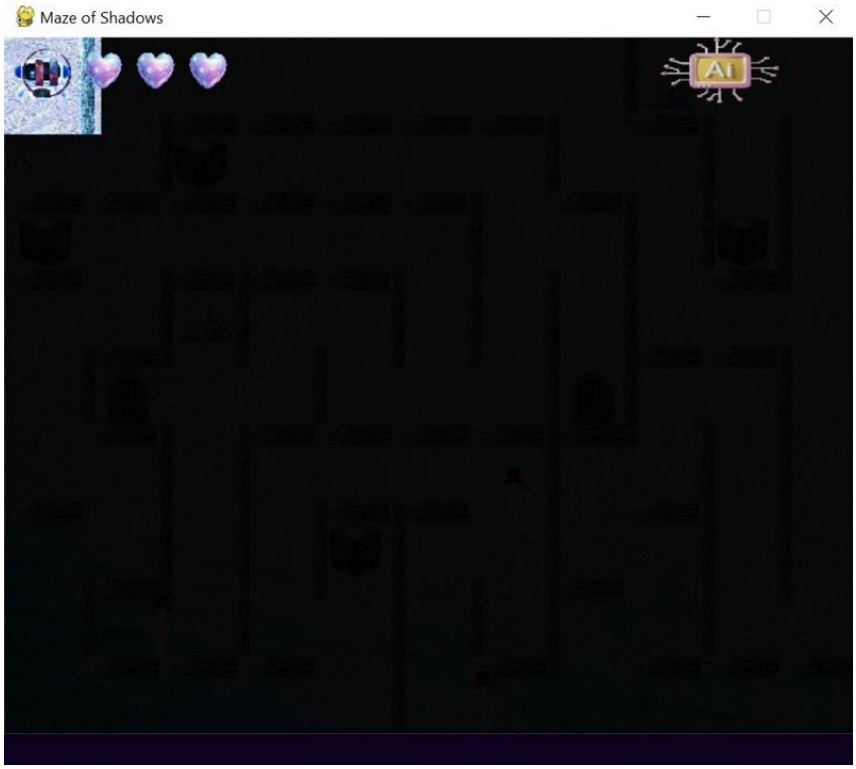
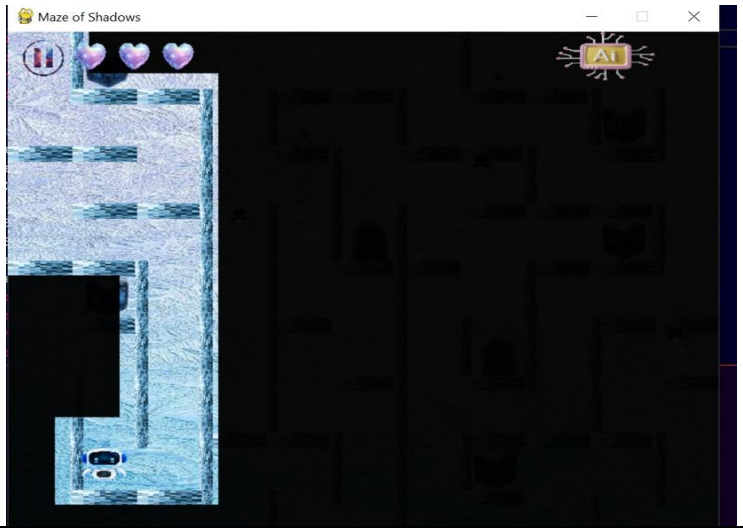
Table 5 levels of the game

USE CASE	4
NAME	Levels
ACTOR	User
DESCRIPTION	This shows the game levels. It also shows if the level is completed, locked of unlocked.
FIGURE	

Figure 4 levels

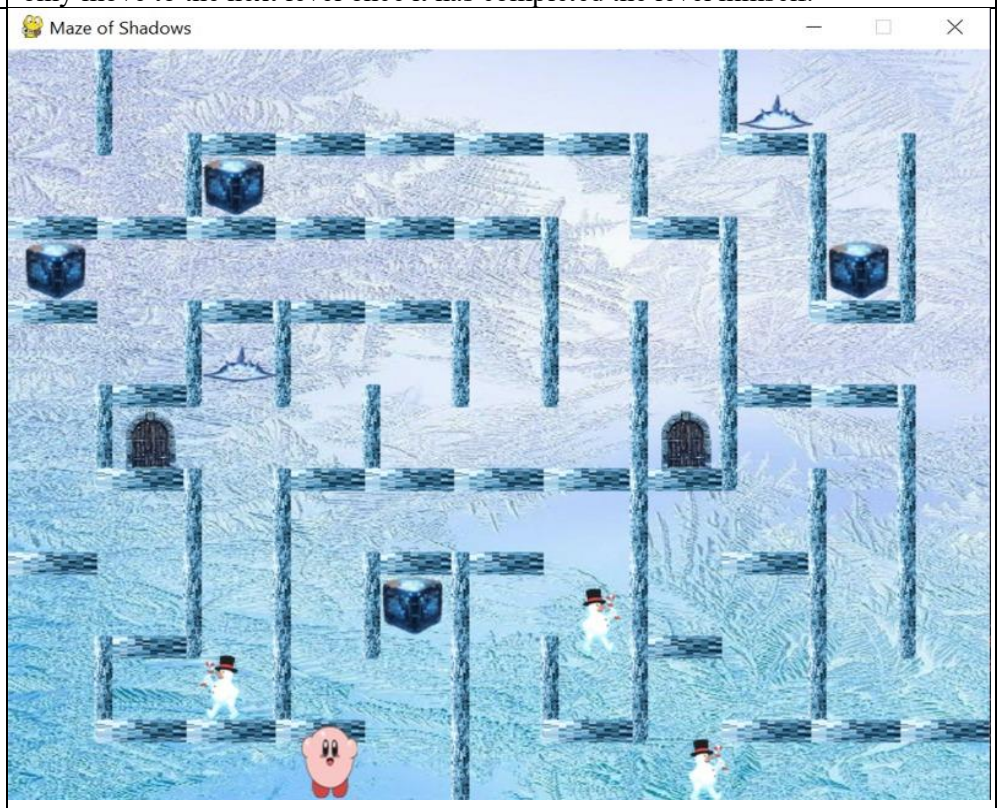
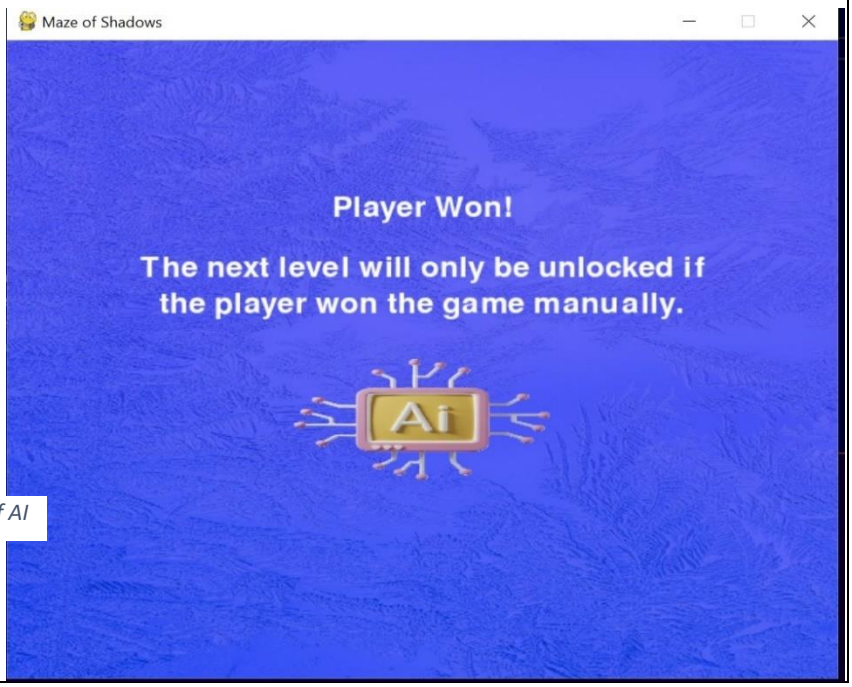
2.5 Level 1

Table 6 level 1

USE CASE	5
NAME	Level 1
ACTOR	User
DESCRIPTION	This screen shows the description of level 1. Player has 3 lives. An AI button on the upper right corner shows the means that player can use assistance to win this level. A pause button on the upper left corner presents various options. The screen is completely black. It means player can not see through maze. Areas where player will go will become light, rest of the maze will remain dark.
FIGURE	 *Figure 5 level 1*  *Figure 6 level 1*

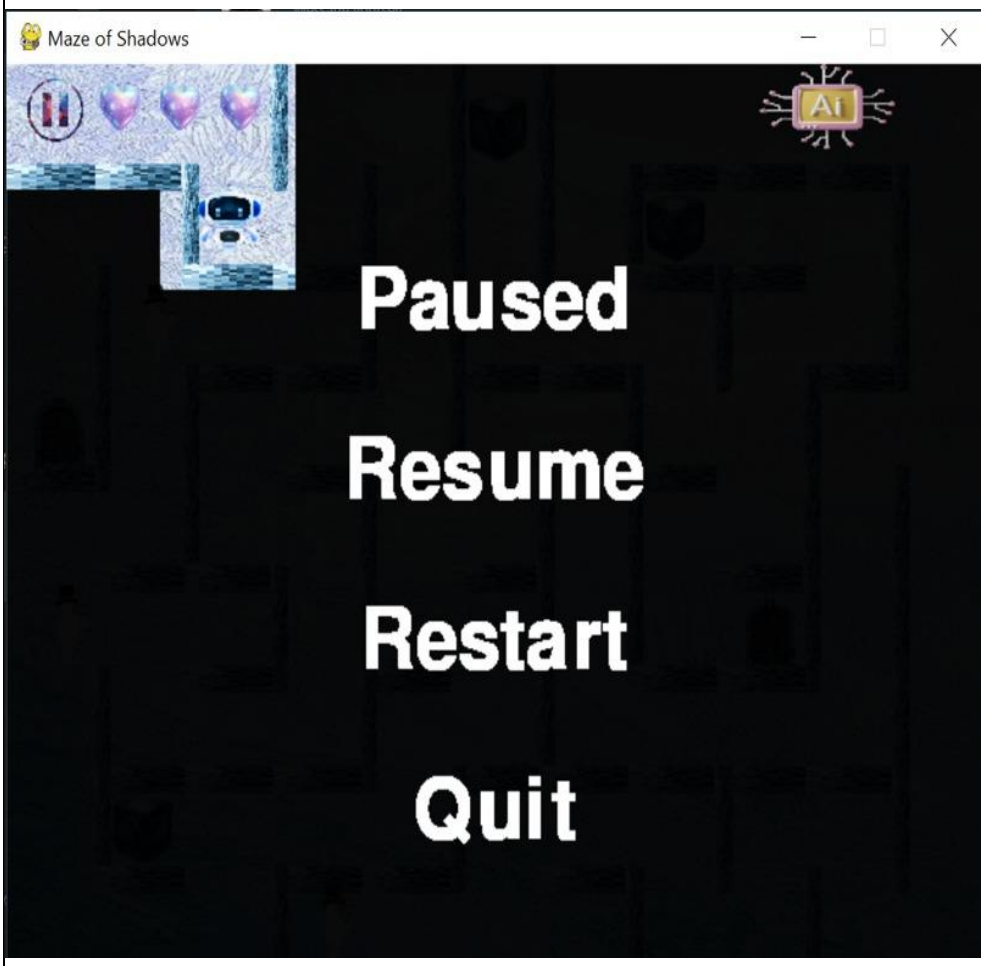
2.6 AI Assistance

Table 7 AI assistance

USE CASE	6
NAME	Use of AI
ACTOR	User
DESCRIPTION	Player can use the AI button to travers the maze automatically. But the player can only move to the next level once it has completed the level himself.
FIGURE	 *Figure 8 use of AI*  *Figure 7 use of AI*

2.7 Use of pause button

Table 8 use of pause button

USE CASE	7
NAME	Pause button
ACTOR	User
DESCRIPTION	Once payer clicks the pause button on the upper left corner, various options are presented to the player such as resume, restart and quit. Player can select any of his choice.
FIGURE	 *Figure 9 pause button*

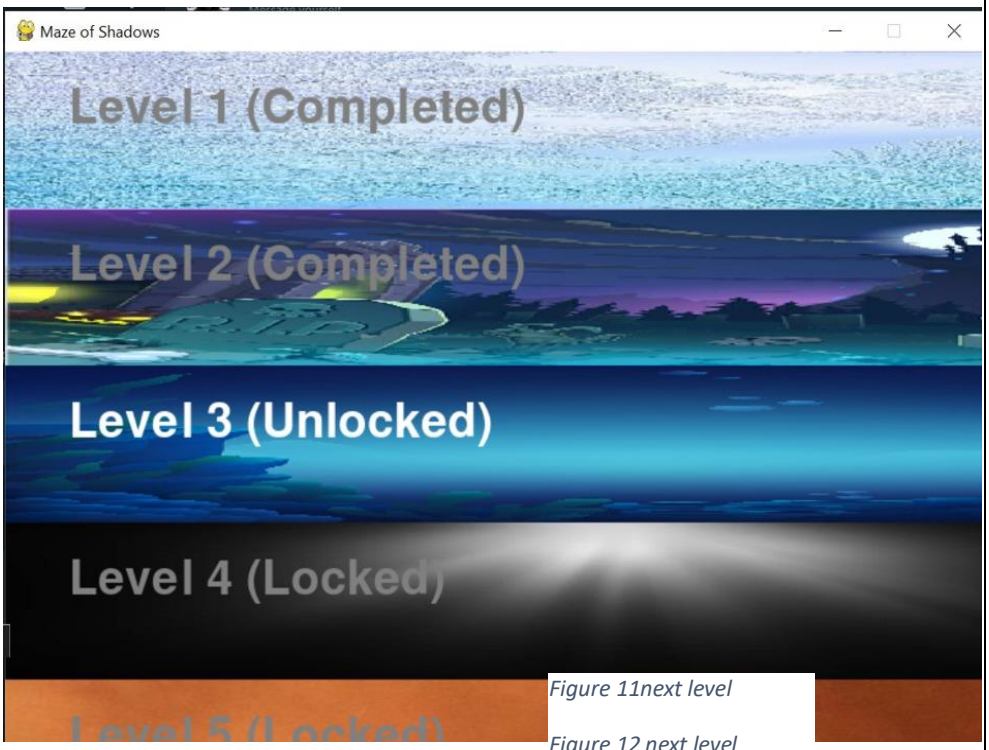
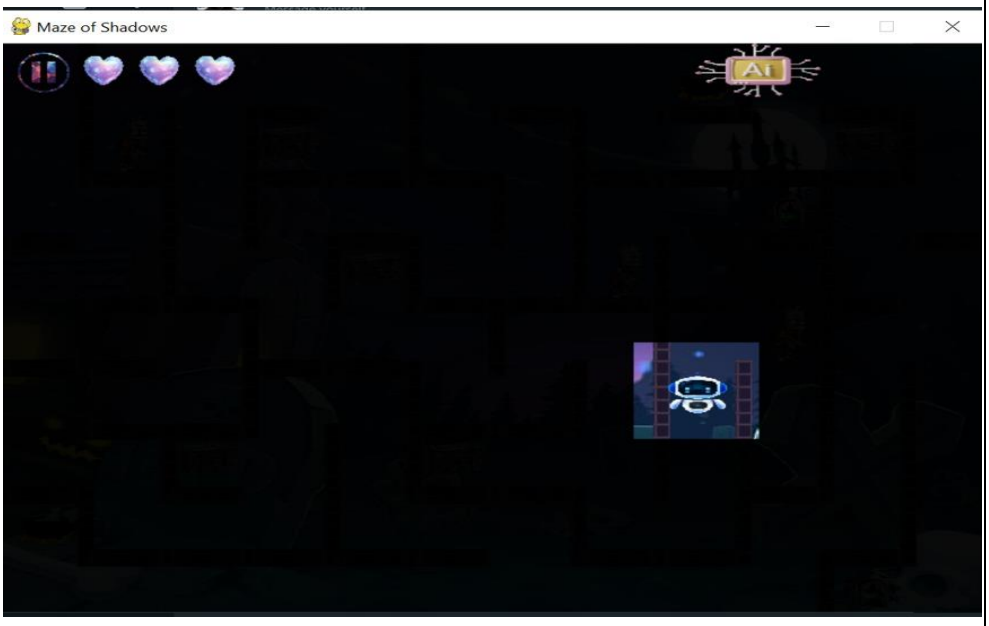
2.8 Life And Death

Table 9 life and death

USE CASE	8
NAME	Life and death
ACTOR	User
DESCRIPTION	All levels have different hurdles that player encounters. There are enemies that shoot at the player. Once the bullet strikes the player, it dies that ultimately reduces the lives of the player. Hurdles also include volcano or icebergs. If player fall into that, it dies. The ultimate goal is to find the key from enemy and unlock the doors to reach the next level.
FIGURE	 *Figure 10 life and death*

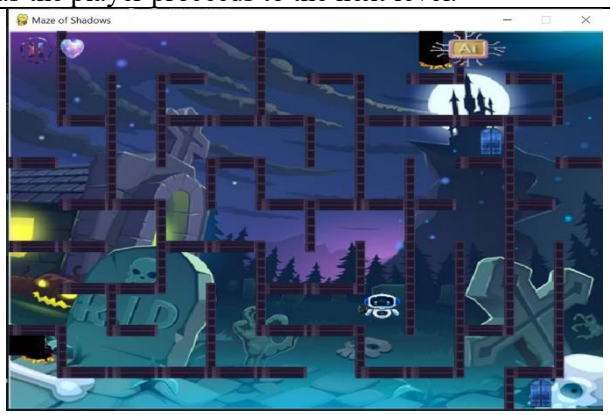
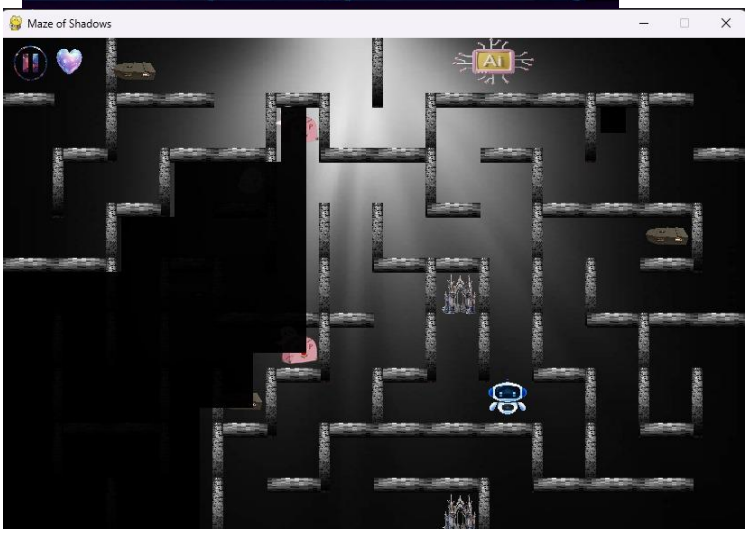
2.9 Next Level

Table 10 life and death

USE CASE	9
NAME	Next level
ACTOR	User
DESCRIPTION	Once the player completes the current level, the next level starts. The status of the level is also updated.
FIGURE	 *Figure 11next level* *Figure 12 next level* 

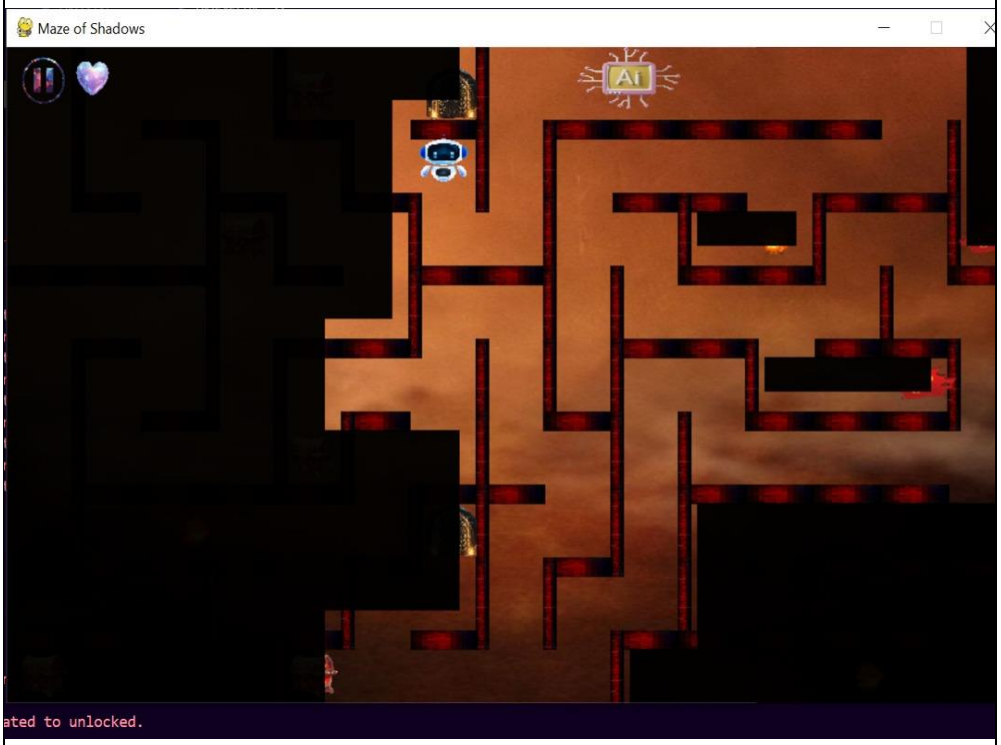
2.10 level 2, 3 ,4

Table 11 next level

USE CASE	10
NAME	Level 2, 3, 4
ACTOR	User
DESCRIPTION	One after another the player proceeds to the next level. The difficulty level increases as the player proceeds to the next level.
FIGURE	 *Figure 13 level 2*  *Figure 14 level 3*  *Figure 15 level 4*

2.11 final level

Table 12 final level

USE CASE	11
NAME	Final level
ACTOR	User
DESCRIPTION	After completing all 4 levels, the player unlocks the final and most difficult level. You are lucky if you complete this level without dying.
FIGURE	 *Figure 16 final level*

3 Game Details

Game is played by a single player. Player can use arrow keys to control the movements. Once the current level is completed, next level is automatically unlocked. One after another the player proceeds to the next level. The difficulty level increases as the player proceeds to the next level. After completing all 4 levels, the player unlocks the final and most difficult level. All levels have different hurdles that player encounters. There are enemies that shoot at the player. Once the bullet strikes the player, it dies that ultimately reduces the lives of the player. Hurdles also include volcano or icebergs. If player fall into that, it dies. The ultimate goal is to find the key from enemy and unlock the doors to reach the next level.

4 Conclusion

The maze game project implements all core concepts of DSA and is an excellent learning tool for understanding and applying DSA. This project highlights the efficiency of algorithms in interactive programming tasks.

5 Future Enhancements

- Implementing additional gameplay features such as checkpoints, timed challenges, and obstacles.
- Extending the game to support multiplayer modes.
- Introducing machine learning techniques to dynamically adjust maze difficulty based on player performance.
- Porting the game to mobile platforms for wider accessibility.