Software Requirements Specification

for

3D Maze Game

Prepared by Group 212

Tanmay Mathurvaishya (IIT2023117)
Aditya Sharma (IIT2023162)
Keshav Porwal (IIT2023211)
Eshant (IIT2023198)
Rahul Roy (IIT2023196)
Lakavath Peer Singh (IIT2023197)

April 2025

Contents

Revision History

Name	Date	Reason for Changes	Version
Tanmay	April 12, 2025	Implemented basic maze generation algorithm using depth-first search	0.1
Keshav	April 15, 2025	Integrated OpenGL for 3D rendering and tested initial wall rendering	0.2
Rahul	April 18, 2025	Added texture mapping to walls and floor using OpenGL shaders	0.3
Eshant	April 22, 2025	Developed collision detection for player navigation in the maze	0.4
Aditya	April 26, 2025	Refined lighting and camera controls for better visual experience	0.5
Peer Singh	April 29, 2025	Initial creation of SRS document	1.0

1 Introduction

1.1 Purpose

This document defines the software requirements for the 3D Maze Game developed by our group. It describes the system architecture, functionalities, and constraints in accordance with IEEE SRS standards.

1.2 Document Conventions

Requirements are labeled with identifiers such as REQ-1, REQ-2, etc. Headings follow IEEE SRS section numbering. Screenshots are provided where applicable.

1.3 Intended Audience and Reading Suggestions

This document is intended for developers, project managers, testers, and technical writers. Start with Sections 1–2 for overview, then refer to Section 3 for interfaces and Section 4 for system features.

1.4 Product Scope

The 3D Maze Game is a Python-based, first-person maze exploration game. It generates mazes procedurally and allows navigation with keyboard inputs using OpenGL and Pygame for rendering and controls.

1.5 References

- 3D Maze Game GitHub Repository
- Python Official Documentation
- Pygame Library Documentation

2 Overall Description

2.1 Product Perspective

This is a standalone application, not part of a larger system. It uses external libraries for rendering (PyOpenGL) and interaction (Pygame). Below is an architectural diagram placeholder:

2.2 Product Functions

- Generate 3D mazes using algorithms.
- Display real-time visualization of maze generation.

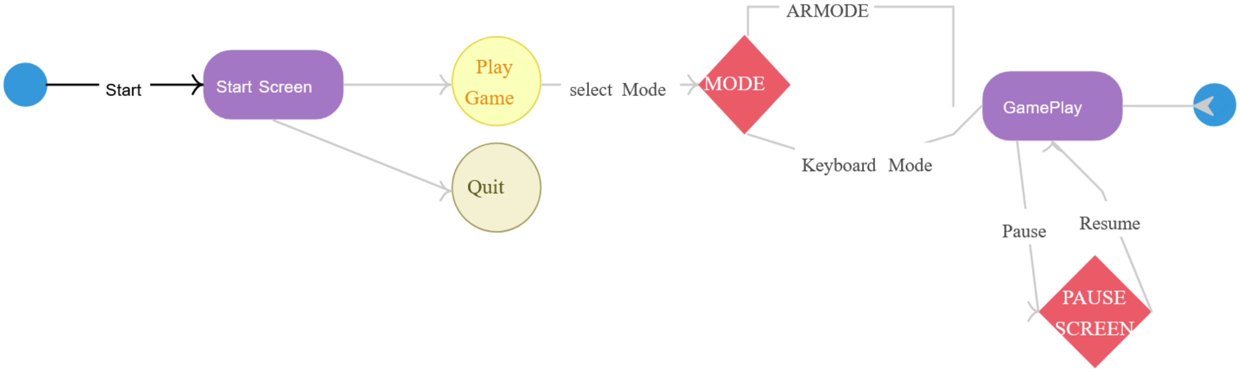

Figure 1: System Architecture

- Allow first-person movement.
- Add interactive elements (e.g., ghosts, teleportation).

2.3 User Classes and Characteristics

- **Players:** Basic familiarity with PC controls, casual gamers.
- **Developers:** Python programmers familiar with OpenGL.

2.4 Operating Environment

- OS: Windows/Linux/macOS
- Python 3.x
- Pygame
- PyOpenGL

2.5 Design and Implementation Constraints

- Must use Python 3.x.
- Use only open-source libraries.
- Cross-platform support.

2.6 User Documentation

- README.md in GitHub
- Inline comments in code
- Video demo (TBD)

2.7 Assumptions and Dependencies

- Dependencies are installed via `pip install -r requirements.txt`.
- No internet connection required during runtime.

3 External Interface Requirements

3.1 User Interfaces

- First-person 3D rendering window
- Keyboard controls: W/A/S/D for movement
- Escape key to exit

3.2 Hardware Interfaces

Keyboard, standard monitor, and optional GPU for better rendering.

3.3 Software Interfaces

- Python 3.x
- Pygame
- PyOpenGL

3.4 Communications Interfaces

None required — game is offline.

4 System Features

4.1 Maze Generation

4.1.1 Description and Priority

Generates a maze using algorithms such as DFS. Priority: High.

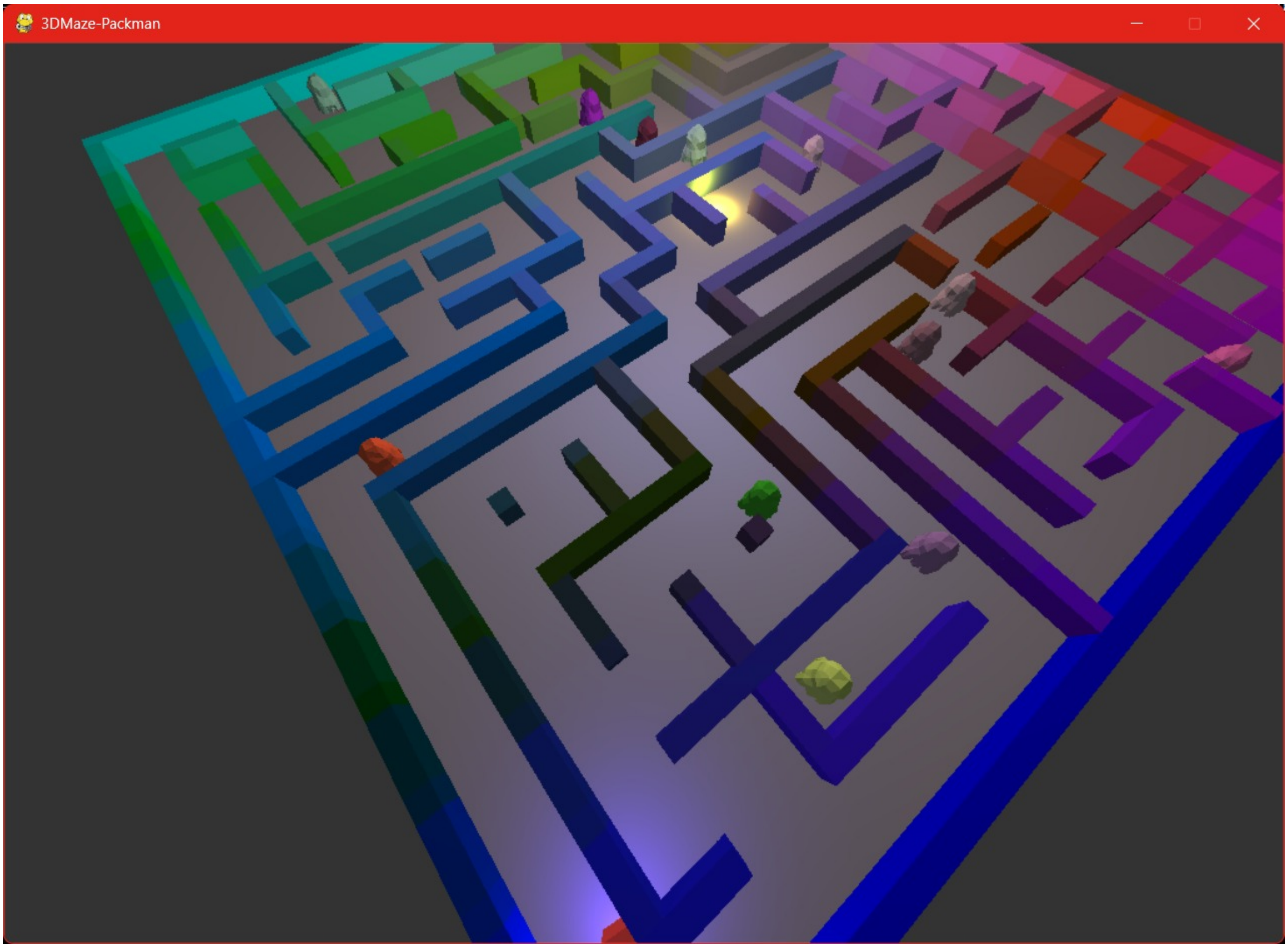

Figure 2: First-Person Maze Navigation

4.1.2 Stimulus/Response

Game start → algorithm runs → maze rendered in 3D.

4.1.3 Functional Requirements

- **REQ-1**: Maze must be generated each time the game starts.
- **REQ-2**: Visualization must update in real time.

4.2 First-Person Navigation

4.2.1 Description and Priority

Allows user to explore the maze in first-person. Priority: High.

4.2.2 Stimulus/Response

W/A/S/D keys → player moves in 3D space.

4.2.3 Functional Requirements

- **REQ-3**: Support for forward, backward, strafe left/right.
- **REQ-4**: Collision detection with walls.

4.3 Interactive Elements

4.3.1 Description and Priority

Includes ghosts and teleporters. Priority: Medium.

4.3.2 Functional Requirements

- **REQ-5**: Ghosts must follow player path logic.
- **REQ-6**: Teleporters move player to a new location.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

- Maintain at least 30 FPS.
- Maze generation ¡ 3 seconds.

5.2 Safety Requirements

- Ensure that the game does not cause any physical harm or discomfort to users.
- Implement measures to prevent motion sickness, such as adjustable camera sensitivity.

5.3 Security Requirements

- Protect the game from unauthorized modifications by verifying the integrity of game files.
- Ensure that user data, if any, is stored securely and complies with data protection regulations.

5.4 Software Quality Attributes

- **Maintainability:** Modular Python code.
- **Portability:** Cross-platform compatible.
- **Usability:** Simple control scheme.

5.5 Business Rules

- The game is open-source and free to use under the MIT license.
- Contributions to the project must adhere to the project's coding standards and guidelines.

6 Other Requirements

- The application should run smoothly on both Windows and Linux platforms.
- Source code must be modular, well-documented, and maintainable.
- External libraries used (e.g., PyGame, NumPy) must be listed along with their versions.
- All assets (e.g., textures, models, sounds) must either be original or properly licensed for use.
- A short gameplay video (1–2 minutes) demonstrating the main features is recommended for better evaluation.

A Glossary

- **Pygame:** A Python library for writing video games.
- **OpenGL:** A standard API for rendering 3D graphics.

B Analysis Models

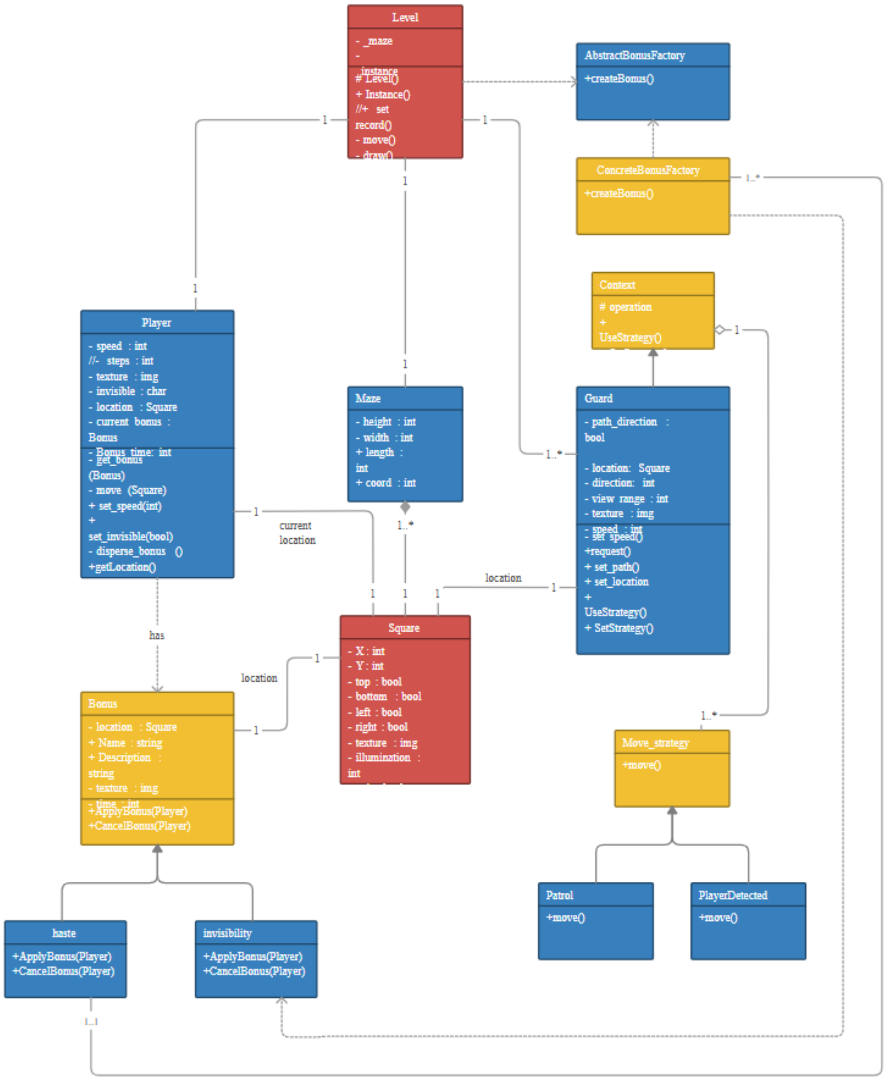

Figure 3: Class Diagram of the 3D Maze Game