Shortest Path in a 3D Maze using A* Search

Shreyansh Pathak
D24CSA006
IIT Jodhpur

November 14, 2024

Abstract

This project aims to find the shortest path within a 3D maze using the A* search algorithm, with a significant focus on rendering the maze environment in Unity. By integrating concepts from computer graphics, we render a visually engaging and interactive 3D maze where users can observe the calculated shortest path. This report discusses the computer graphics techniques, mathematical concepts, and Unity implementations that contribute to rendering and navigating the 3D maze.

Contents

Chapter 1

Introduction

The goal of this project is to develop a 3D maze in Unity and employ the A* search algorithm to find and visualize the shortest path between two points. Beyond pathfinding, the project focuses on rendering aspects, utilizing Unity's 3D graphics capabilities to create an immersive environment. The core components include maze generation, pathfinding visualization, and rendering techniques that enhance the visual fidelity and interactivity of the maze.

1.1 Problem Statement

Creating a realistic and interactive 3D maze involves rendering walls, floors, and path visualizations, which requires knowledge of computer graphics, game development, and algorithmic efficiency. This report emphasizes the rendering techniques and mathematical foundations applied in the project.

Chapter 2

Methodology

2.1 Maze Generation and Rendering

Maze generation is handled by the `MazeGenerator.cs` script, where we create a grid of cells. Each cell is represented by a `MazeCell` object, which can either be a wall or floor.

2.1.1 Grid-Based Layout

Each cell's position is calculated based on its grid coordinates (x, z):

$$\text{worldPosition} = (x \cdot \text{cellSize}, 0, z \cdot \text{cellSize})$$

This enables the creation of a structured grid, where walls are placed on the borders or randomly throughout the maze to create a challenging path layout.

2.1.2 Material Assignment

To differentiate walls from the path, the maze uses distinct materials:

- **WallMaterial**: Applied to cells marked as walls.
- **FloorMaterial**: Used for floor cells.
- **PathMaterial**: Highlights the shortest path found by the algorithm.

2.2 Pathfinding with A* Search

The A* algorithm is used to find the shortest path. It calculates a heuristic, h, which is the estimated cost from the current cell to the target cell. The total cost function, f, is defined as:

$$f = g + h$$

where g is the cost from the start to the current cell, and h is the heuristic cost to the target.

2.2.1 Heuristic Calculation

The heuristic, h, is computed using Manhattan distance:

$$h = |x_1 - x_2| + |z_1 - z_2|$$

This choice of heuristic suits a grid-based layout, providing efficient estimates without complex calculations.

Chapter 3

Core Computer Graphics Concepts

3.1 3D Object Instantiation

The maze is rendered by instantiating wall and floor prefabs at designated coordinates. Unity's `Instantiate` function allows the creation of 3D objects in the scene. The position of each wall or floor prefab is calculated based on its grid coordinates, ensuring uniform spacing.

3.2 Rendering Path Visualization

Path visualization enhances interactivity by rendering the shortest path using spheres or other indicators along the path found by the A* algorithm. The spheres are placed slightly above the floor level:

$$\text{pathMarker.position} = \text{cell.worldPosition} + (0, 0.5, 0)$$

This provides a clear, elevated view of the path without obstructing the maze.

3.3 Material Shading and Lighting

Materials in Unity define the appearance of objects, including their color, texture, and reflectiveness. By adjusting properties in the `PathMaterial`, we highlight the shortest path in a visually appealing manner. Lighting and shading are applied to simulate realistic interactions between light and surfaces, enhancing immersion.

3.4 Camera and Perspective Projection

The camera setup in Unity determines the view of the maze. Using a perspective projection, we create depth perception, making the 3D environment look more realistic. The transformation from world coordinates to screen coordinates follows this basic projection formula:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \\ z \end{pmatrix}$$

where (x, y, z) are the world coordinates and (x', y', z') are the projected screen coordinates.

Chapter 4

Mathematical Concepts

4.1 Pathfinding Costs and Distance Calculation

The A* algorithm's cost calculations involve both movement costs (g) and heuristic costs (h). In this project, we use a Manhattan heuristic for simplicity in grid-based movement. The movement cost is calculated as follows:

$$g = 10 \times \text{steps\_in\_same\_direction} + 14 \times \text{diagonal\_steps}$$

where moving diagonally between cells adds a slightly higher cost, helping prevent unrealistic path shortening.

4.2 Transformations and Rotations

To move the player smoothly through the maze, transformations and rotations are applied:

$$\text{newPosition} = \text{oldPosition} + \text{direction} \times \text{speed} \times \Delta t$$

This equation governs the player's movement through each frame, ensuring smooth navigation. The rotation speed further controls the orientation of the player based on user input.

Chapter 5

Results

5.1 Maze Generation and Rendering

The maze generation algorithm creates a grid with randomly placed walls, rendered with materials to distinguish walls from paths. Figures below show different perspectives of the 3D maze.

5.2 Pathfinding Visualization

The A* pathfinding algorithm successfully finds and highlights the shortest path, which is visualized by spheres along the route. The visualized path guides users through the maze, adding an interactive element.

Chapter 6

Conclusion

This project demonstrates the effective integration of pathfinding with 3D rendering techniques. By focusing on materials, lighting, and perspective in Unity, the project delivers an immersive maze experience. Future improvements could include more complex mazes and dynamic obstacles, expanding the project's application scope.

Chapter 7

References

- Unity Documentation: `https://docs.unity3d.com/`
- Computer Graphics: Principles and Practice, Foley et al.
- Artificial Intelligence: A Modern Approach, Russell and Norvig