**The Princess and the Dragon – Escape from the Castle**

****

**Team Members:**

**Maria Dubrovsky** – 323480533  
**Daniela Lavi** – 318608395

**Gameplay Video:**  
<https://youtube.com/shorts/WBco6Go-j-w?feature=share>

**Presentation Video:**  
<https://youtu.be/4AhVj6XVq-g>

**GitHub Repository:**  
<https://github.com/mashadu99/Princess-Maze>

**General Background:**

As part of the course requirements, we were asked to implement an advanced algorithmic solution within an interactive system. We chose to develop a maze adventure game called "Princess Maze", in which a princess must escape a haunted castle while being chased by a dragon.

The game features three levels, each introducing a new layer of complexity:

Level 1 – Basic movement and health system using hearts

Level 2 – Addition of ice tiles, which cause the player to slip unpredictably to preset locations

**L**evel 3 – Addition of mud tiles, which slow down the player's movement

The princess starts with three hearts, and each time the dragon catches her, she loses one. When all hearts are gone, the game ends. To successfully complete a level, the player must collect a key and reach the castle gate.

Each level is structured as a unique maze, gradually increasing in size and complexity — with more walls, loops, and dead ends. Strategic placement of ice, mud, hearts, and keys was essential to balance difficulty and ensure fair gameplay, giving the player enough room to strategize and adapt.

**Problem We Identified:**

The core challenge was to make the dragon’s behavior feel intelligent and reactive — not based on hardcoded paths. We needed a way for the dragon to dynamically adapt its movement to the terrain, the real-time changes, and the player’s decisions.

Another challenge was designing a maze that strikes the right balance: **difficult but fair** — ensuring that both the princess and the dragon had access to key areas of the map without getting trapped or breaking the game’s logic.

**Our Algorithmic Solution:**

We implemented the A\* (A-Star) pathfinding algorithm for the dragon, allowing it to calculate the shortest path to the princess at each turn. The algorithm also considered tile costs — for example, ice tiles were cheaper to cross than mud tiles — giving the dragon the ability to make weighted decisions about movement.

Additionally, during the maze design phase, we used Breadth-First Search (BFS) to verify that each level was solvable - ensuring that:

The princess could always reach the exit gate-

The dragon always had a path to the princess-

The key could be collected through a valid path-

**System Architecture:**

Built using Python and the Pygame library for interactive graphics and animations

-Three progressively challenging levels with unique maps

A centralized Game class that controls all game logic-

A separate A\* implementation in src/pathfinding/astar.py-

-Modular file structure with reusable components

Internal testing and documentation throughout development-

**Insights & Reflections:**

This project required us to think algorithmically, understanding how cost-based heuristics influence decision-making in real-time systems. Implementing A\* in a visual, interactive environment gave us a much deeper appreciation for how theoretical algorithms are brought to life in games and simulations.

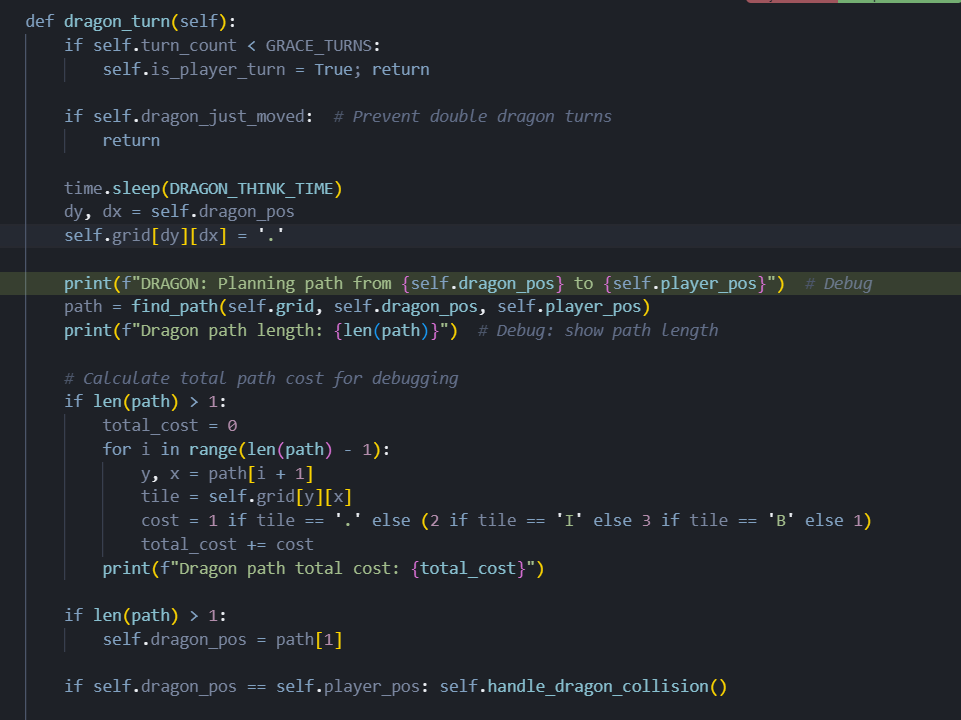
We also learned how to design fair and engaging challenges, making the gameplay both satisfying and beatable. Balancing fun and frustration required thoughtful level design, careful placement of obstacles, and tuning of AI behavior.

Ultimately, this experience gave us a better understanding of:

Integrating search algorithms into game logic--

Designing for both human and AI agents-

The importance of dynamic environments and stateful decision-making-



תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

