Game-a-thon

27th June 2024 - 4th July 2024
Language: Java

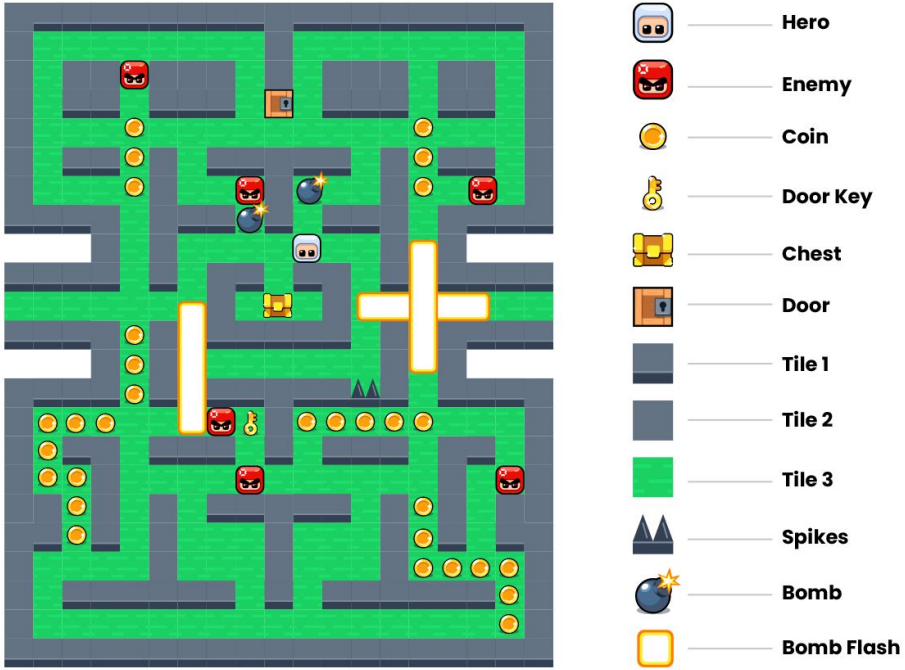
Mystery Maze

Introduction

Welcome to the exciting world of "Mystery Maze," a puzzle-adventure game that challenges players with tricky mazes, strategic thinking, and problem-solving. In "Mystery Maze," players explore a labyrinth full of hidden treasures, valuable coins, and clues to solve a final mystery. Adding to the thrill, a roaming AI enemy sets traps and chases the player. Players must use bombs to break certain walls or defeat the AI, while also collecting coins to increase their score. "Mystery Maze" combines exploration, strategy, and quick reflexes, offering a thrilling gaming experience.

Problem Statement

The goal of this hackathon is to create a working version of "Mystery Maze" in 8 days using Java. Participants must make sure the game is fun, engaging, and works well within the given time.



Reference Games: https://g.co/kgs/cJc2m2n, https://www.retrogames.cz/play_085-NES.php

Milestones (100 Points)

	Milestones	Section	Points
1	Maze Generation Algorithm	Main	25
2	Player Controls and Movement	Main	25
3	Graphics and User Interface	Main	25
4	Enemy AI	Main	25
5	Timer and Scoring System	Bonus	15
6	Treasures & Coins	Bonus	15
7	Bomb Mechanic	Bonus	15
8	Testing & Polishing	Bonus	15

Complete at least 50% of the main milestones to qualify for the reward.

Features

1. Maze Generation:
- Implement a basic maze generation algorithm, such as depth-first search or recursive backtracking.
- Ensure the paths lead to dead ends, exit doors, or hidden treasures.
- Introduce spike obstacles that the player must avoid.
- Randomly generate the maze each time for enhanced replayability.

2. Player Controls and Movement:
- Allow the player character to move left, right, up, and down using touch controls, WASD keys, or arrow keys.
- Implement player movement and collision detection with maze walls.
- Enable the spacebar or a double-tap key to drop a bomb.

3. Graphics and User Interface:
- Use the provided graphics for the maze, player, AI agent, bombs, treasures, and coins.
- Create a basic user interface featuring a start screen, an in-game HUD (displaying timer and score), and an end screen.
- Developers can use their custom assets instead of the UI assets provided by us.

4. Enemy AI Agent:
- Introduce an AI-controlled enemy that roams the maze.
- If the enemy touches the player, the player will die.
- Patrolling: The AI agent will randomly patrols the maze until it spots the player.
- Chasing: When the player enters the agent's line of sight (within a certain range), the agent starts chasing the player.
- Lost Sight: If the player escapes the agent's line of sight for a set period, the agent returns to patrolling.

Features

5. Timer and Scoring System:
- Implement a timer to track maze completion time.
- Develop a scoring system based on time and collected items (treasures and coins).
- Display the status of remaining bombs.

6. Treasures & Coins:
- Each coin awards 10 points to the player.
- A treasure box, placed randomly in the maze, awards 100 points upon collection.
- Randomly distribute treasures and coins throughout the maze.
- Implement logic for collecting treasures and coins.

7. Bomb Mechanic:
- Once dropped, a bomb will explode after 2 seconds.
- The bomb will detonate upon contact, killing both the player and enemy agents within its range
- Develop bomb countdown and explosion mechanics.
- Ensure bombs can destroy certain walls, such as spiked obstacles.

8. Testing & Polishing:
- Polish the game by adding sound effects and minor visual improvements.
- Include a bomb flash visual effect.
- Test the game for bugs and fix any issues.
- Handle unexpected crashes.
- Optimize code for reduced processing and memory usage.
- Feel free to unleash your creativity by adding new features or enhancing the gameplay to make it more engaging.

Language and resources

Core Language:
- Only Java is permitted for writing game logic.

Third-Party Libraries:
- Any Java-related libraries are allowed.

Frameworks:
- Any Java-related frameworks are permitted, such as LibGDX, JavaFX, or JMonkey.

Level Editors:
- Level editors are allowed for designing and testing maze layouts, such as Hyperlap2D, Tiled level editor etc..

Playable Build:
- Acceptable formats include .exe, .apk, and executable .jar files.

Version Control:
- Use GitHub for version control.

Online resources:
- Use of AI tools like ChatGPT or Gemini is allowed
- Use of code from other GitHub repositories or Stack Overflow is permitted.

Assessment

1. Submission:
- Create a private github project repository and add “gmonks” as a collaborator.
- Push separate commits for every completed milestones.
- Submit a 5 minute (max) video recording of the gameplay demo and its features.
- GForm will be floated for final submission on the last day, i.e., July 4.

4. Eligibility:
- Complete at least 50% of the main milestones to qualify for the reward.
- Top 5 performers will get the chance to present their project, maximum 2 candidates will be selected.

2. Presentation:
- Candidates will present their game to the judges.
- Judges may ask candidates to explain the logic or do some changes with the code.

3. Tie breaker:
- This round will only be done if more that 2 candidates become eligible for the reward.
- Java theory-related questions may be asked.
- Live Java coding tests using LeetCode questions will be conducted.

5. Reward:
- Selected candidates will be offered a Game Developer position at our Mumbai office.
- On successful joining, the candidates will be eligible to get 50,000/- as a joining bonus.
- CTC will be between 6L-9L based on candidates final assessment.