Dungeon Escape

1. Project Overview

Dungeon Escape is an Action game where players must survive and escape from a dangerous dungeon filled with deadly traps, AI-controlled enemies, and randomized puzzles. Players must explore, collect items, upgrade their character, and find the exit before being defeated.

Mechanics:

- **Procedural Dungeon Generation**
- **Character Upgrades**
- **Statistical Tracking**
- **Enemy Tracking**

Features:

- **Player movement and combat mechanics**
- **Enemy pathfinding**
- **Random dungeon generation**
- **Item collection and power-ups**

2. Project Review

Relevant existing project: Hades

- Hades is an action game known for its fast-paced combat, enemy AI, and randomized dungeon layout. It provides a compelling progression system where players receive random boons from the gods, altering their playstyle. However, it lacks deep AI pathfinding, environmental puzzles, and detailed player statistics tracking.

Dungeon Escape Enhancements:

- **Enemy Pathfinding** – Enemies intelligently track the player instead of following preset patterns.
- **Traps and Puzzles** – Each dungeon run presents unique challenges, requiring problem-solving skills.
- **Item-Based Character Upgrades** – Players can customize their build through item enhancements instead of relying on random upgrades.
- **In-depth Statistical Analysis** – The game tracks detailed stats such as completion time, enemy kills, item usage, and survival rate, providing insights into player performance

3. Programming Development

3.1 Game Concept

Genre: Action / Roguelike / Puzzle

Objective: Survive a dungeon filled with traps and monsters, collect upgradeable items, and find the exit to escape.

Key Features:

- Procedural Generation – Every dungeon layout is unique.

- Enemy Pathfinding – Enemies intelligently chase and ambush players.

- **Randomized Traps & Puzzles –** Requires strategic thinking to progress.
- **Item Upgrades –** Enhance abilities with weapons, shields, and skills.
- **Stat Tracking –** Time spent, enemies defeated, items used, survival rate.

Unique Selling Points: High replayability, strategic depth, and challenging AI combat.

3.2 Object-Oriented Programming Implementation

- **Player (Represents the main character controlled by the player)**

  Attributes:
  - position (tuple) → Current coordinates in the dungeon
  - health (int) → Player's HP
  - inventory (list) → List of collected items
  - speed (float) → Movement speed

  Methods:
  - move(direction) → Updates position based on input
  - attack(target) → Attacks an enemy
  - use_item(item) → Uses an item from inventory
  - take_damage(amount) → Reduces health when hit

- **Enemy (Represents AI-controlled monsters in the dungeon)**

  Attributes:
  - position (tuple) → Current location
  - health (int) → Enemy HP
  - damage (int) → Attack strength
  - aggression_level (float) → Determines enemy behavior
  - pathfinding_algorithm (A* or similar) → Finds the best route

  Methods:
  - move_toward_player(player_position) → Moves toward the player using pathfinding
  - attack(player) → Damages the player when in range
  - take_damage(amount) → Reduces health when hit
  - patrol() → Moves randomly when not chasing

- **Dungeon (Handles map generation and dungeon structure)**
  Attributes:

    - grid (2D list) → Stores dungeon layout
    - rooms (list) → Stores room coordinates
    - traps (list) → Contains trap locations
    - exit_position (tuple) → Defines the exit

  Methods:

    - generate_map() → Creates a new dungeon layout
    - place_traps() → Randomly adds traps
    - is_walkable(position) → Checks if a position is accessible
    - find_exit() → Returns exit location

- **Item (Represents collectible and usable items)**

  Attributes:

    - name (str) → Item name
    - effect (str) → Effect type (e.g., heal, boost attack, shield)
    - value (int) → Strength of the effect
    - duration (int) → How long the effect lasts (if applicable)

  Methods:

    - apply_effect(player) → Applies the item's effect to the player
    - remove_effect(player) → Removes temporary effects when expired
    - display_info() → Shows item description

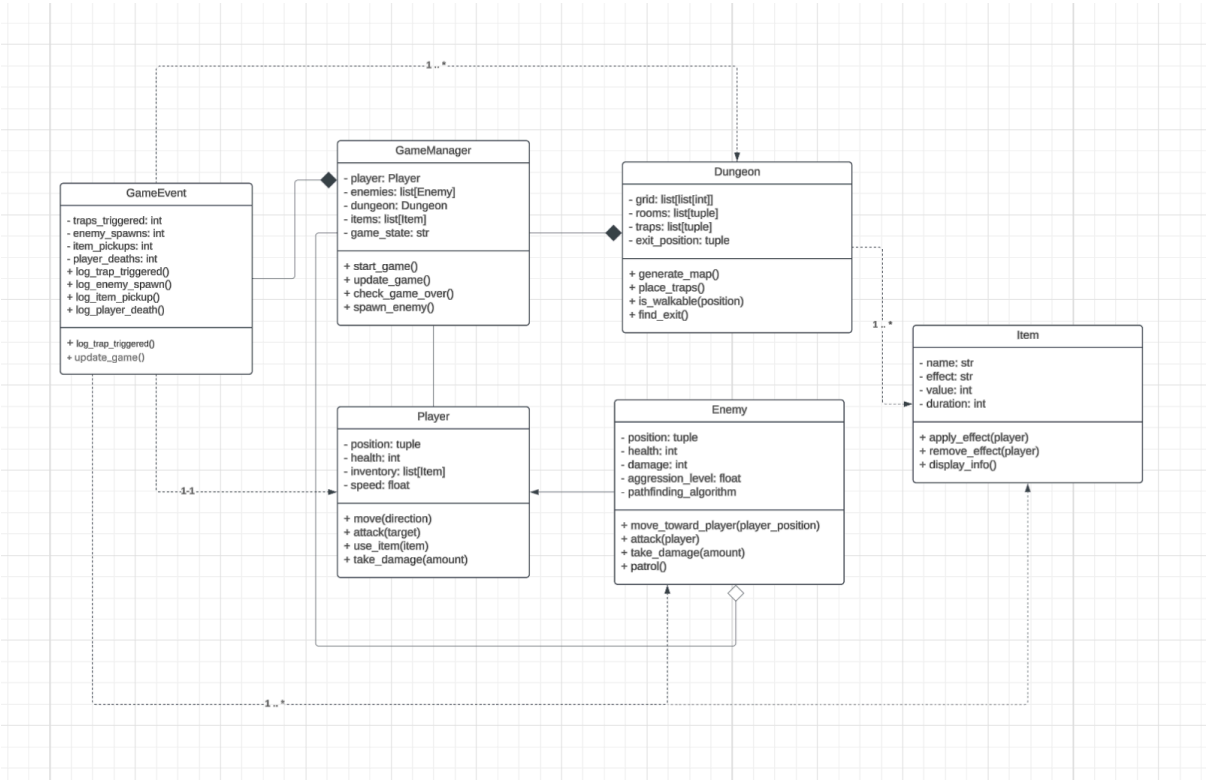
- **GameManager (Controls the game loop and overall mechanics)**

  Attributes:

    - player (Player) → Reference to the main character
    - enemies (list) → Stores all enemy objects
    - dungeon (Dungeon) → Manages dungeon layout
    - items (list) → Stores all available items
    - game_state (str) → Tracks current state (running, paused, game over)

  Methods:

    - start_game() → Initializes the game
    - update_game() → Runs game logic each frame
    - check_game_over() → Determines if the game should end
    - spawn_enemy() → Generates new enemies



3.3 Algorithms Involved

- **Enemy Pathfinding –** Enables enemies to chase players intelligently.
- **Combat System** – Rule-based logic for attack, defense, and damage calculation.
- **Item Drop System** – Weighted random selection for loot rarity balance.
- **Trap Activation** – Event-driven mechanics trigger traps when stepped on.
- **Stat Tracking** – Logs player actions for performance analysis.
- **Procedural Dungeon Generation** – Uses BSP (Binary Space Partitioning) to generate unique dungeon layouts.

4. Statistical Data (Prop Stats)

4.1 Data Features

- **Time Taken to Complete Levels** – Measures how long players spend in each dungeon.
- **Enemies Defeated –** Tracks the number of enemies eliminated.
- **Items Collected & Used –** Logs which items players pick up and use.
- **Survival Rate –** Calculates the percentage of successful vs. failed runs.
- **Trap Activations –** Records how often players trigger traps.

3.2 Data Recording Method

- **Format:** Data will be saved in a structured **CSV file** with columns for player ID, time taken, enemies defeated, items used, survival status, and trap activations.
- **Storage:** Each game session will append a new row to the CSV file.

3.3 Data Analysis Report

Statistical Measures:

- **Average & Median** – Calculate average time taken, enemies defeated, and survival rate.
- **Success Rate (%) –** Measure the percentage of completed runs.
- **Item Usage Frequency –** Identify the most and least used items.
- **Trap Activation Rate –** Analyze how often players trigger traps.

Features	**Why is it good to have this feature data? What can it be used for?**	**How will you obtain 50 values of this feature data?**	**Which variable (and which class) will you collect this from?**	**How will you display this feature data (via summarization statistics or via graph)?**
Time Spent in Dungeon	Helps understand player engagement and pacing. Can be used to adjust difficulty balance.	**The bot will play the game 50 times, with each session having randomized behaviors (e.g., fast/slow, extensive/minimal exploration) based on game logic.**	time_spent variable from **Player** class.	Histogram showing distribution of time spent.
Enemies Defeated	Measures player combat performance. Can be used to tweak enemy difficulty.	**The bot will randomly select a combat style (evasive/aggressive) for each session.**	enemies_defeated variable from **Player** class.	Bar chart comparing average kills per session
Items Collected	Indicates how well players explore and utilize items. Can be used to balance item	**The bot will randomly collect items based on its defined behavior (e.g., looting all points/selectively collecting only key**	items_collected variable from **Inventory** class.	Pie chart showing item type distribution.

	placements.	**items).**		
Traps Triggered	Helps analyze whether traps are too difficult or ineffective. Can be used to adjust trap mechanics.	**The bot will move randomly with a chance to trigger traps depending on its carefulness level.**	traps_triggered variable from **GameEvents** class.	Line graph showing trap activation trends over time.
Survival Rate	Measures how often players successfully escape. Can be used to balance difficulty.	**The bot will play until game completion based on its capabilities (HP, collected items) with randomized survival behaviors.**	survival_rate variable from **Player** class.	Percentage summary and success/failure ratio in a table
Distance Traveled	Helps understand how players navigate the dungeon. Can be used to refine level design.	**The bot will navigate the dungeon randomly, either exploring every room or heading straight to the objective.**	distance_traveled variable from **Player** class.	Box plot comparing distances traveled per session.

4.4 Statistical Data Revision

Table: Statistical Values

- **Feature Name:** Time Spent in Dungeon, Items Collected, Enemies Defeated, Survival Rate
- **Statistical Values:**
    - **Time Spent in Dungeon:** Mean, Median, Min, Max, Standard Deviation, Percentiles (25th, 50th, 75th)
    - **Items Collected:** Mean, Max, Percent Distribution by Item Type
    - **Enemies Defeated:** Mean, Median, Min, Max, Standard Deviation, Percentiles (25th, 50th, 75th)
    - **Survival Rate:** Mean, Median, Min, Max, Standard Deviation, Percentiles (25th, 50th, 75th)

Table for Statistical Values and Graph Selection:

Graph	Feature Name	Graph Objective	Graph Type	X-axis	Y-axis
1	Time Spent in Dungeon	Analyze the distribution of player session durations.	Histogram	Interval (minutes)	Number of players
2	Items Collected	Analysis type ratio of item	Pie Chart	Item category	Percent(%)
3	Enemies Defeated vs. Survival Rate	Relation between (Enemies Defeated vs. Survival Rate)	Scatter plot	Number of enemies killed	Survival rate(%)

Graph 1: Distribution (Histogram)

- **Axes:**
  - **x-axis:** Interval (minutes)
  - **y-axis:** Number of player
  - **Hand-Drawn Example:**

Graph 2: Proportion (Pie Chart)

- **Sections:**
  - Weapon (35%)
  - Potion (25%)
  - Key (20%)
  - Other (20%)
  - **Hand-Drawn Example:**

Graph 3: Relation (Scatter Plot)

- **Axes:**
  - **x-axis:** Number of enemies killed
  - **y-axis:** Survival rate (%)
  - **Hand-Drawn Example:**

Planing Submission

Weekly Goals

Date Range	Planned Tasks
26 Mar - 2 April	**Data collection setup, define statistical measures**
3 April - 9 April	**Implement data logging, start analysis**
10 April - 16 April	**Process data, generate initial tables and graphs**
17 April - 23 April	**Refine analysis, adjust data visualization**
24 April - 11 May	**Finalize report, optimize presentation for clarity**

Milestone Goals

- 50% Completion (April 16): Data collection, table and statistical calculations completed.
- 75% Completion (April 23): Graphs generated, analysis refined, and initial review.
- 100% Completion (May 11): Finalized report and full presentation ready.

4. Project Timeline

Week	Task
1 (10 March)	Proposal submission / Project initiation
2 (17 March)	Full proposal submission
3 (24 March)	Research & Refinement (Review and refine the proposal - Address any

	missing details - Incorporate feedback)
4 (31 March)	Planning & Structuring (Review and refine the proposal - Address any missing details - Incorporate feedback)
5 (7 April)	Final Edits & Review (Proofread for errors - Get feedback from peers or professors - Prepare final submission file)
6 (14 April)	Submission week (Draft)

5. Document version

Version: *1.0*
Date: *4 March 2025*

Date	Name	Description of Revision, Feedback, Comments
15/3	Phiranath	The idea is interesting, make sure that the path finding algorithm is efficient enough. The document is very detailed. Good Job!
15/3	Pattapon	Great job :)
28/3	Phiranath	The data seems good, don't manually collect them unless you really need to do it.