Starry Night

1. Project Overview

Starry Night is a hybrid turn-based and bullet-hell dungeon adventure RPG, inspired by **Final Fantasy** and **Soul Knight**. Players control a hero on a quest to conquer dungeons, defeat enemies, gain experience, unlock abilities, and ultimately face a powerful final boss. The game features two distinct combat systems:

1. **Turn-based combat** (Final Fantasy style): Each turn, the player chooses an action — attack, defend, use an ability, or consume an item. Enemies respond with one of their unique movesets, ensuring each battle feels dynamic and strategic.
2. **Bullet-hell combat** (Soul Knight style): For weaker enemies, the player skips traditional turn-based combat and instead engages in fast-paced, action-based bullet-hell fights. This prevents repetitive battles while adding variety and excitement to encounters.

The combination of these two systems offers a unique gameplay loop, balancing strategy and reflex-based action.

2. Project Review

The idea draws inspiration from **Metaphor: ReFantazio**, which blends turn-based combat with action phases. In **Metaphor**, weaker enemies can be defeated through an action-based phase instead of going through full turn-based battles repeatedly. **Starry Night** builds upon this idea, offering two distinct combat modes — one for tactical, high-stakes battles and the other for faster-paced, action-oriented skirmishes against weaker enemies. This ensures pacing stays fresh, avoiding the fatigue of repeated turn-based encounters.

Key improvements include:

- **Unique enemy movesets** (2-4 per enemy) to ensure variety.
- **Exploration phase** between battles, including item collection and random events.

3. Programming Development

3.1 Game Concept

The player controls a hero exploring procedurally-generated dungeons. Enemies vary in strength, dictating which combat system is triggered:

- **Turn-based combat:** Tactical battles against stronger foes.
- **Bullet-hell combat:** Reflex-based fights for weaker enemies.

Key Features:

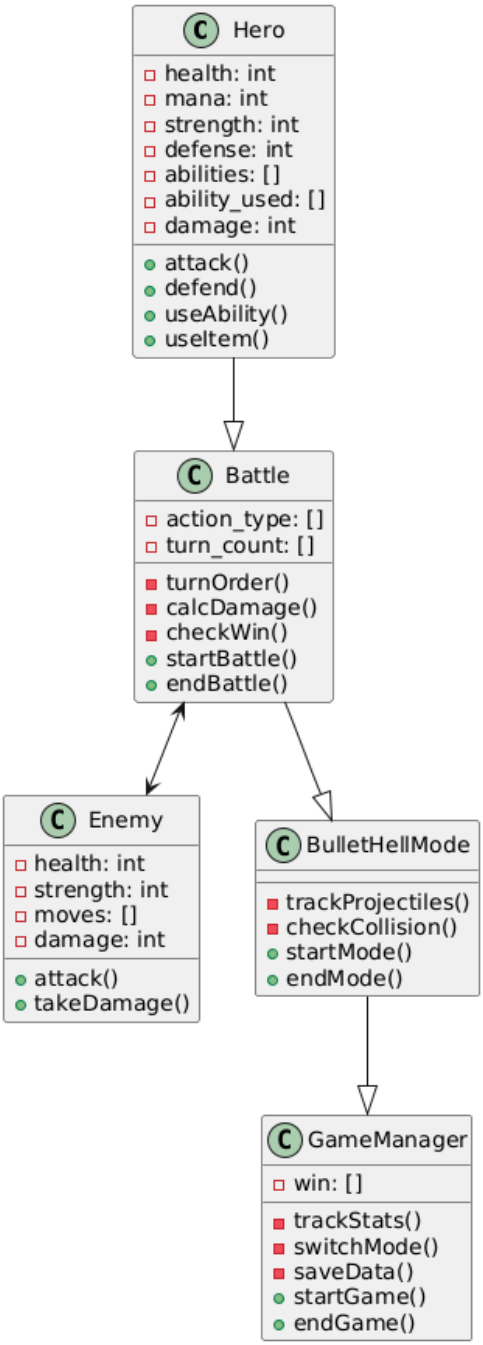
- **Dual combat mechanics:** Two distinct battle styles.
- **Hero progression:** Experience, leveling up, and unlocking abilities.
- **Enemy variety:** Each enemy has 2-4 unique moves to keep fights interesting.
- **Dungeon exploration:** Collect items and uncover secrets between battles.
- **Adaptive difficulty:** Combat becomes tougher as the player progresses.

3.2 Object-Oriented Programming Implementation

Hero	Represents the player-controlled character. Tracks stats like health, mana, strength, defense, and abilities. Handles attacks, defenses, abilities, and items.

Enemy	Represents enemy characters. Includes health, strength, and a randomizer that selects one of 2-4 moves per enemy.
Battle	Manages the turn-based combat system, turn order, win/lose checks, and logs. Supports transitioning to bullet-hell mode if needed.
BulletHellMode	Handles the action-based, bullet-hell combat system. Tracks player movement, enemy projectiles, and collision detection.
GameManager	Manages the main game loop, switches between exploration and combat modes, and tracks player progress and stats.

UML Diagram



3.3 Algorithms Involved

- **Turn Order Algorithm:** Determines who acts first based on a speed stat.
- **Damage Calculation:** Determines the damage based on the attacker and the defender.
- **Bullet-hell Collision Detection:** Tracks player movement and enemy projectile collision.
- **Victory/Defeat Logic:** Checks remaining health on both sides to determine outcomes.

4. Statistical Data (Prop Stats)

4.1 Data Features

1. **Player Actions:** Tracks frequency of attacking, defending, using abilities, and switching combat modes.
2. **Damage Dealt:** Records total and average damage per battle.
3. **Turns Taken:** Measures how efficiently the player wins turn-based battles.
4. **Win/Loss Ratio:** Evaluates overall player performance in both combat systems.

4.2 Data Recording Method

CSV File: Each battle generates a row of data saved to a CSV file, tracking actions, performance, and results.

4.3 Data Analysis Report

- **Statistical Measures:** Average damage, turn efficiency, bullet-hell accuracy, and win rate.

Feature	Why it is good to have this 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)?
Player Actions	Helps analyze player strategy trends (e.g., attack vs. defense choices).	Record each player's action choice per battle across 50 battles.	action_type variable in Battle class.	Bar chart to compare most used actions.
Damage Dealt	Useful for balancing game difficulty and player progression.	Track damage output across 50 battles.	damage variable in Hero and Enemy classes.	Line graph to show damage trends.
Turns Taken	Measures battle efficiency and pacing of turn-based combat.	Count the number of turns used in 50 battles.	turn_count variable in Battle class.	Summarization statistics (mean, min, max).
Win/Loss Ratio	Assesses overall player performance and difficulty curve.	Record the outcome (win/loss) of 50 battles.	win variable in GameManager class.	Pie chart to visualize win/loss distribution.
Most Used Ability	Identifies popular skills, helping with game balancing.	Track ability usage across 50 battles.	ability_used variable in Hero class.	Bar chart to compare ability popularity.

- **Visual Representations:**
  - **Bar chart:** Most used abilities.
  - **Line graph:** Damage output per battle.
  - **Pie chart:** Win/Loss breakdown.
- **Performance Insights:** Identifies patterns — e.g., over-reliance on one ability or excessive turns taken.

Feature Name	Graph Objective	Graph Type	X-axis	Y-axis
Player Actions	Identify popular player strategies(Atk, Buff Def)	Bar Chart	Actions	Frequency
Damage Dealt	Track damage done over battles	Line Graph	Battle Number	Damage Output
Win/Loss Ratio	Assess player performance trends	Pie Chart	Win/Loss Categories	Percentage

Milestone Goals

- **50% Completion by 16 April:**
  - Combat system (turn-based & bullet-hell) implemented
  - Data collection system logging player actions, damage, and battle results
  - Basic statistical calculations implemented
- **75% Completion by 23 April:**
  - Graph visualization (bar, line, and pie charts) functional in Tkinter
  - User interface refined for better readability
  - Gameplay balancing adjustments
- **100% Completion by 11 May:**
  - Final testing and debugging completed
  - Documentation and project report finalized
  - Proposal submission

5. Project Timeline

Week	Task
1 (10 March)	Proposal submission / Project initiation
2 (17 March)	Full proposal submission
3 (24 March)	
4 (31 March)	
5 (7 April)	
6 (14 April)	Submission week (Draft)

6. Document version

Version: 4.0
Date: 3 April 2025