ASTRY Parto

1. Project Overview

"ASTRY Parto" is a game where you control a spaceship from a Top-Down View perspective. The spaceship constantly moves forward. There are two control buttons: one for rotating the spaceship clockwise (press it twice quickly to dash) and the other for shooting bullets. The goal of the game is to control your spaceship and shoot down other ships until you're the last one remaining.

2. Project Review

The reference game is Astro Party, which was played on a tablet when I was younger. My goal is to make it playable on a computer and add special skills, as well as create different level designs.

3. Programming Development

3.1 Game Concept

"Astry Parto" is a game where players control a spaceship in a Top-Down view. The spaceship constantly moves forward, and players can rotate and shoot to attack other ships. The game will include special skills, such as dashing to dodge or shooting special bullets with unique effects that vary depending on the difficulty level. The goal is to destroy other bots(players)' ships and be the last one remaining.

3.2 Object-Oriented Programming Implementation

1.Spaceship

Role: Manages the movement and rotation of the vehicle, shooting bullets, collecting power-ups, and taking damage.

Relation: May be linked to the **Bullet** class for shooting bullets and the **PowerUp** class for collecting power-ups.

2.Bullet

Role: Handles the movement of bullets shot from the spaceship and detects collisions.

Relation: Linked to **Spaceship** for shooting bullets and detecting collisions with **Asteroid** or **Spaceship**.

3.Asteroid

Role: Creates asteroids that appear in the game, move, detect collisions with the spaceship or bullets, and may contain power-ups.

Relation: Linked to **Spaceship** for collision detection and **PowerUp** for power-up collection.

4.GameManager

Role: Controls the start of the game, tracks the score, and manages win/loss states.

Relation: Linked to other classes like **Spaceship**, **Asteroid**, **PowerUp** to manage game flow.

5.PowerUp

Role: Manages special powers like laser, special bullets, and other boosts.

Relation: Linked to **Spaceship** for power-up collection.

6.Laser (if applicable)

Role: Manages the firing of more powerful lasers.

Relation: May be linked to **Bullet** or **Spaceship** for firing.

7.Bot

Role: Controls the AI-controlled spaceship, follows specific movement patterns, and interacts with the player's spaceship (e.g., shooting, dodging).

Relation: Linked to Spaceship for movement and interaction with bullets and power-ups.

8.AsteroidParticle

Role: Handles the behavior of small asteroid fragments that are created when asteroids are destroyed. These particles move and disappear after a short time.

Relation: GameManager

9.**SmokeParticle**

Role: Manages the visual effect of smoke produced when the spaceship or bot performs a dash move. This creates an atmospheric effect, simulating the exhaust or propulsion during a dash.

Relation: Linked to **Spaceship** and **Bot** for triggering smoke when they perform a dash move.

3.3 Algorithms Involved

Event-Driven Mechanism

The game listens for player inputs, like rotating the spaceship or shooting bullets, and responds immediately. For example, when a key is pressed, the spaceship will rotate or a bullet will be fired without delay.

Rule-Based Logic

The game ensures that player actions follow the rules, such as making sure the player can only shoot after the cooldown period or perform a dash only when the direction key is pressed twice quickly. This helps keep the gameplay fair and structured.

Randomization

Power-ups are randomly spawned whenever an **Obstacle** is destroyed. This means every time an obstacle is destroyed, the game will randomly select a power-up (like Laser or Triple Bullet) and place it in a random location, making each playthrough feel fresh and unpredictable.

Collision Detection

The game constantly checks if the **Spaceship** or **Bullets** collide with

obstacles. When a collision occurs, the game will trigger the appropriate response, such as destroying the obstacle or causing damage to the spaceship.

4. Statistical Data (Prop Stats)

4.1 Data Features

1. Accuracy *which measures how accurate the player is by calculating the ratio of successful hits to total bullets fired*
2.Power-ups Collected*: This tracks the number of power-ups the player collects during gameplay, helping us understand how often they utilize the game's power-ups and the effect they have on gameplay.*

3.Obstacles Destroyed*: This records how many obstacles (like asteroids) the player has destroyed, showing how effectively they are navigating and progressing through the game.*

4.Dash Usage*: This measures how frequently the player uses the dash ability, providing insights into how critical this movement strategy is to their survival and gameplay tactics.*

5.Time Played*: This tracks how much time the player spends in each game session, helping us understand how long players are engaged with the game and whether the session length correlates with other metrics like power-ups collected or obstacles destroyed.*

4.2 Data Recording Method

The data will be stored in the CSV file.

4.3 Data Analysis Report

Dash vs Time Played

*Use a **scatter plot** or **line plot** to show the relationship between dash usage and time played.*

Bullet Fire vs Accuracy

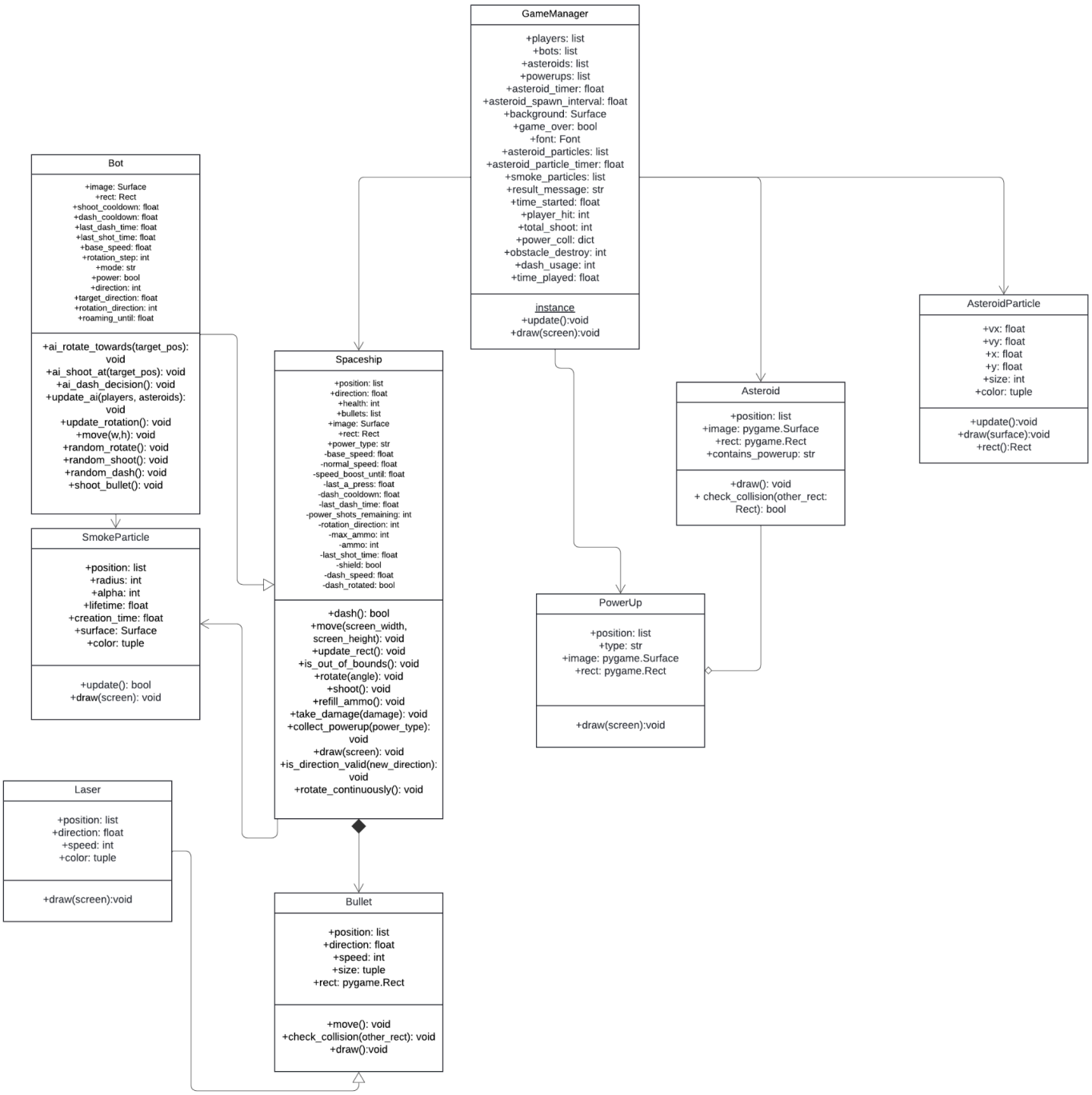
*Use a **scatter plot** to display the relationship between bullets fired and accuracy.*

Power-up Frequency

*Use a **bar chart** to show the frequency of each power-up collected.*

4. Project Timeline

Week	Task
1 (10 March)	Proposal submission / Project initiation
2 (17 March)	
3 (24 March)	Full proposal summit
4 (31 March)	Finish the base code of all features
5 (7 April)	Implement data tracking system
6 (14 April)	Submission week (Draft)



5. Document version

Version: 5.0

Date: 11 May 2025

Date	Name	Description of Revision, Feedback, Comments
15/3	Rattapoom	Very good!
16/3	Parima	Game Concept needs more information.
29/3	Rattapoom	You'd have to complete the game fast so that you'd have time for beta testers to play. Otherwise, you could collect something like "Bullets fired per minute", "Bullets hit per minute" etc. since it won't require you (or beta testers, or both) to play the game 50 times.
29/3	Parima	Overall, the proposal is well-structured. For the statistical data, I recommend collecting data during gameplay at regular intervals, such as every 30 seconds. This will make data collection easier and allow you to represent time-series data, tracking player progress over time.