Deflect

Chetphisuth Tongpa 6710545491

1. Project Overview

A 2d side view game where the player can only attack by deflecting the enemy's attacks.

2. Project Review

- The movement is similar to **mario,** it's a side scrolling game where the player can move left, right and jump with an addition of **double jumping** and **dashing forward** to dodge the enemy's attack.
- The main mechanic is that the player can deflect the enemy's attack and projectile, similar to **Genji from Overwatch.** This is the only way the player can harm the enemy.
- There will be 3 types of enemy, each with **different attack range, movement and attack move set.** Each individual enemy will also spawn with slightly different parameters (using randomness) to give them more life and unpredictability.
- The game loops like **Tetris** where there's not really an end point, the game just keeps going with increasing difficulty (using survival time to determine the difficulty) until the player dies.

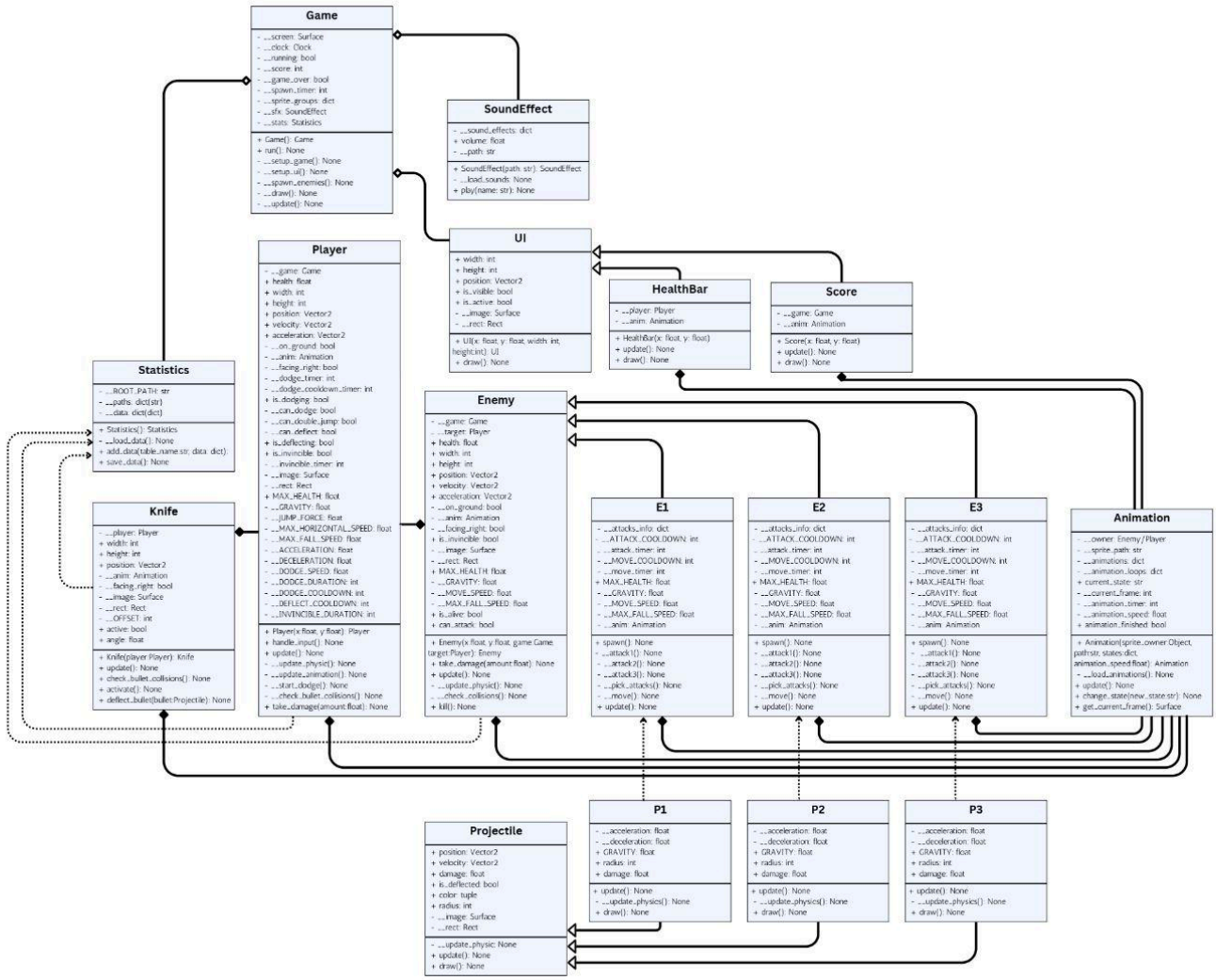
3. Programming Development

3.1 Game Concept

- A 2d side view game (like mario)
- The player can perform these movements: **Left** `(A)`, **Right** `(D)`, **Jump** `(SPACE/W)`, **Double Jump** `(SPACE/W)` **and Dodge**(`SHIFT`).
- **Dodging**(`SHIFT`)**:** Player dashes forward and for a split second, will be invulnerable to all the enemy's projectiles and attacks. It has a small cooldown before the player can dodge again and can only **dodge once if the player is in the air.**

- **Deflecting (`LEFT_MOUSE`):** Player can deflect enemy's projectiles and attacks, changing its ownership and damaging the enemy. It will have a small cooldown before the player can perform another deflect.
- The game will progress forward with a greater number of stronger enemies and the game will not stop until the player dies. Time and amount of dead enemies will determine the difficulty.

3.2 Object-Oriented Programming Implementation



Go to this link for higher resolution (sorry, it's just so big): https://drive.google.com/file/d/1SnORUcLbRCO0oj5oC7JTq08RsM8KGDCK/view?usp=sharing

`Player` – Represents the controllable character.
`Knife` – The weapon used by the player to deflect projectiles.
`Enemy` (Parent Class) – Base class for all enemies.
- `E1, E2, E3` (Child Classes of Enemy) – Different enemy types with unique attack and movement behaviors.
`Statistics` (Singleton Class) – Collects and stores game statistics.
`Projectile` (Parent Class) – projectiles that can be deflected.
- `P1, P2, P3` (Child Classes of Enemy) – Different projectile types for each enemy type. Each has different shapes, speed and other configurations.
`Animation` – Manages sprite animations for Player and Enemies. Load the images and play them as animation.
`SoundEffects` – Handles sound effects.
`UI` (Parent class) – Manages user interface elements like buttons, menus, scores, and health bars.
- `HealthBar, Score` – Shows player's status.
`Game` – Manages the game loop, object updates, and rendering.

3.3 Algorithms Involved

AI Behavior

The enemy movement and attack are handled using an **AI behavior algorithm** where each enemy type will move and attack differently, For example, a long range enemy will keep distance from the player while the close-combatters will try and get close to the player, some enemy will get more aggressive with lower hp, randomness with `random` library will also be used to make enemies a little bit more unpredictable.

Rule-Based Logic

The game mechanics and game flow are controlled through **rule-based logic**.

Event-Driven Mechanics

All of the player's control input will be handled using Pygame's event system.

4. Statistical Data (Prop Stats)

4.1 Data Features

- Player's position every second
- Player's position where damage is taken.
- Number of attacks and their type redirected in each deflect.
- Number of attacks and their type evaded in each dodge.
- Lifespan of each individual enemy.
- Total damage done to each enemy in each deflect.

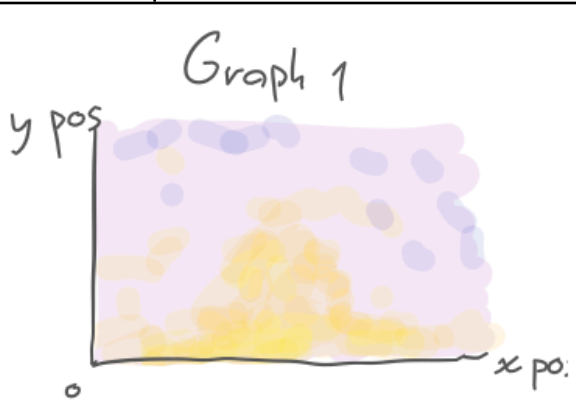
3.2 Data Recording Method

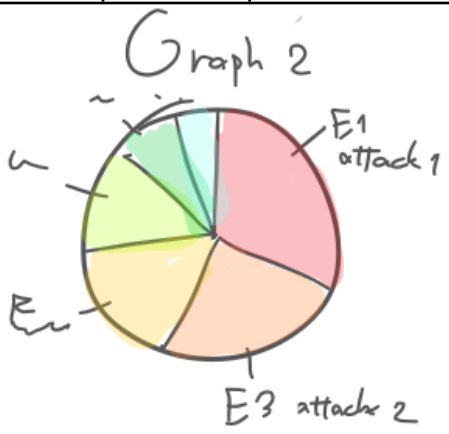
It will be stored in CSV files for simplicity and easier to export into Excel/Google sheet for analysis later.

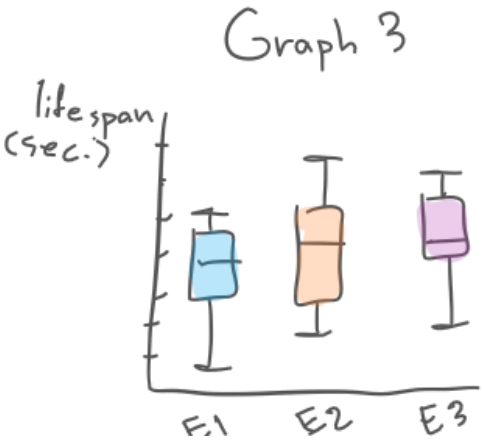
3.3 Data Analysis Report

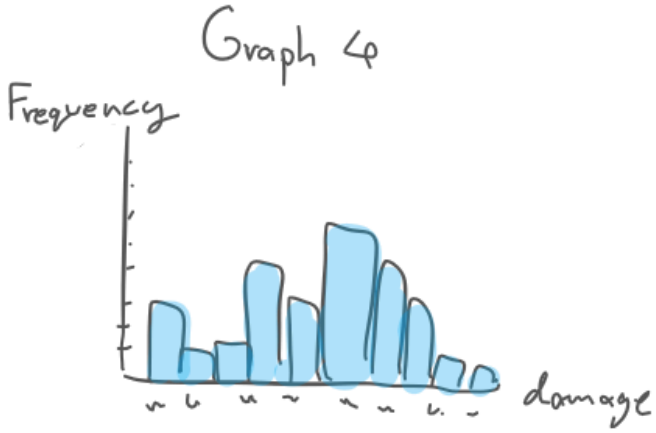
Data Feature	Why is it good to have this data? what can it be used for?	How will you obtain 50 values of this feature data?	Which variable and which class will you collect this from?	How will you display this feature data?
Player's position	Help analyze movement patterns, identify areas of the map that are frequently visited, and understand player behavior. This data can be used to optimize enemies and improve gameplay balance.	Record them every 3 seconds or so.	Collect from the `position` attribute in the `Player` class.	Use a heatmap overlay on the game map
Player's position where damage is taken, with the type of the attack	To balance enemy attacks and see if some of them are too hard to deal with.	Player takes damage multiple times in one game, playing 5-8 games would do it.	Collect from the `position` attribute in the `Player` class when the `take_damage` method is called.	Use a heatmap overlay on the game map + pie chart to view the type of attack.
Number of attacks and their type redirected in each deflect.	To see if some attacks are too easy or too hard to deflect and balance the deflect mechanic.	Player uses deflect a lot, playing 2-3 games would do it.	Collect from the `deflect_bullet` method in the `Knife` class	Histogram for number attack + pie chart for type of attacks
Total damage done to enemies in each deflect	To see if the player dealt damage as expected and use it to balance the enemies and the deflect mechanic.	Player uses deflect a lot, playing 2-3 games would do it.	Collect from the `deflect_bullet` method in the `Knife` class, give each deflected projectile a timestamp tag and tracks if they hit any enemy.	Use Histogram to see
Number of attacks and their type evaded in each dodge.	To see if some attacks are too easy or too hard to dodge and balance the dodge mechanic.	Player uses dodge a lot, playing 2-3 games would do it.	Check player's collision with projectiles when `is_dodging` is True.	Histogram for number attack + pie chart for type of attacks
Lifespan of each individual enemy.	To see the enemy that might be too hard or too easy to kill.	An average player should be able to kill 50 enemies within 4-6 games..	Timestamp enemy's spawn and death using the `clock` attribute in `Game` class	Histogram for the lifespan of each enemy.

	Feature Name	Objective	Graph type/ Table	Graph/Table Info
Table 1	Number of attacks and their type evaded in each dodge.	To see how powerful the dodge mechanic is and if the player can use it as intended.	Table	**Statistical values:** - minimum ,maximum, average and SD of damage evaded.
Graph 1	Player position	To see how players often position themselves, do mostly they stay still in the middle? Do they jump a lot? Do they just go everywhere?	Heatmap	**X-axis:** Player's x position **Y-axis:** Player's y position
Graph 2	Damage income type	Some attacks are meant to be harder to deal with so I want to see if I balance them appropriately.	Pie chart	**Each slice:** Damage income type, (which attack was used from which enemy?)
Graph 3	Lifespan of enemies	To see if the health of the enemy is balanced and if it's too hard to hit.	Boxplot	**X-axis:** Type of enemy (E1 / E2 / E3) **Y-axis:** Lifespan (in second)
Graph 4	Total damage done in each deflected.	To see how powerful the deflect mechanic is and use it to balance the difficulty levels.	Histogram	**X-axis:** Range of damage amount. **Y-axis:** The frequency of that damage range.









4. Project Timeline

Week	Task
1 (10 March)	Proposal submission / Project initiation Change the proposal if some ideas are not feastable.
2 (17 March)	Full proposal submission
3 (24 March)	Enemy is implemented, Player can die.
4 (31 March)	Figured out all enemy types and attacks.
5 (7 April)	All enemy moves and attacks is finished and fully playable
6 (14 April)	Submission week (Draft) Data collection are fully implemented
7 (16 April)	Every component is finished and Codes are cleaned up
8 (23 April)	Balance, animation, UI, Sound effects are polished
9 (11 May)	Everything is finished

5. Document version

Version: `4`

Date: `30 March 2025`