

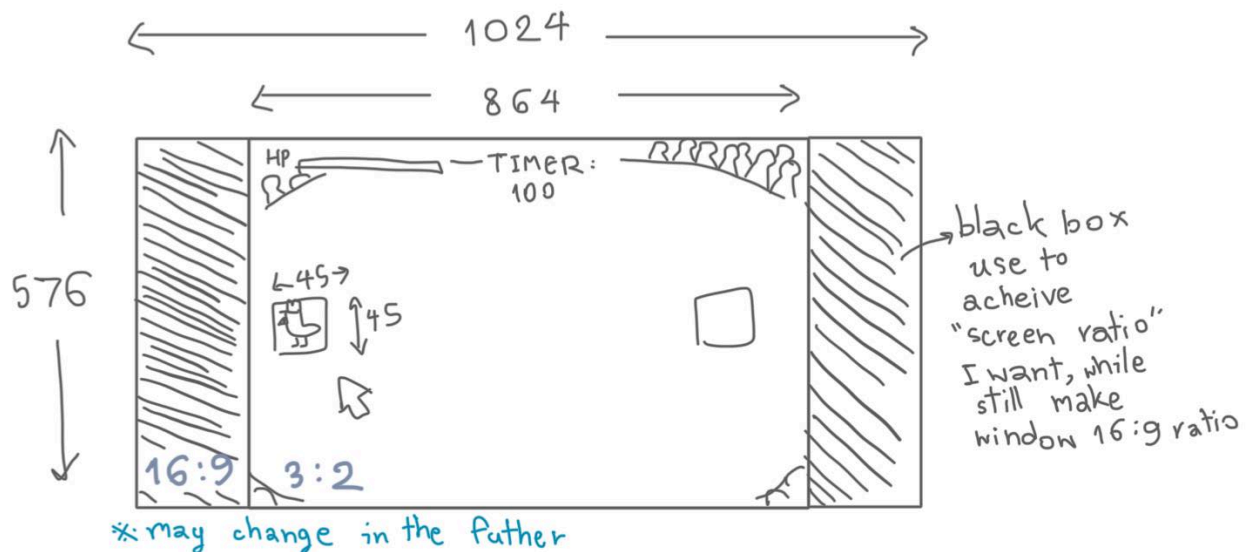
Project Proposal Name

Winner, Winner, Chicken Brawler!!!

1. Project Overview

Overview :

This project is a top-down melee boss-rush game featuring three unique bosses. The main objective is to complete the game with the highest possible score, which is determined by the time taken to finish the game and the percent of lives remaining across all three fights.



Concept/Story :

You are a duck who, one day, was suddenly taken and thrown into a chicken-fighting arena.

Mechanics (plan) :

Smooth movement, score board and database to contain score, dynamic AI behavior, and fully animated sprites.

2. Project Review

I'm not strictly improving upon an existing game, nor have I developed all the mechanics entirely on my own. Instead, I have selected the mechanics I want and researched guides to implement them.

NOTE: Many mechanics are still in development, so some methods or inspirations may change over time.

Example

- On key walk with pygame.sprite
https://youtu.be/OUOI6iCrmCk?si=99HG4txBAWM_Ea8V
- Utilize sprite sheet for animation
<https://youtu.be/ePiMYe7JpJo?si=M-xsOKlaaHSqmvVj>
- Speech bubble
https://youtu.be/Y_ghJY-sW3o?si=EvagURyQEe-pDQ7o
- AI behaviour
<https://youtu.be/wC9iu7cuQjI?si=hq4UJNmTYoS3hZwc>
- Melee attack hitbox
https://youtu.be/NVAXjTzqTyE?si=3DDBBATrwFEUZ8j_

3. Programming Development

3.1 Game Concept

Describe:

This is a top-down boss rush game featuring three unique boss fights. Players must utilize their moveset—including walking, rolling, and both light and heavy attacks—to defeat the bosses. The player can also move in up to eight directions.

Mechanic:


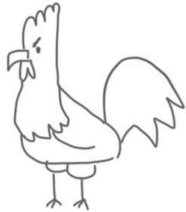


- A persistent database that retains data even after the program is closed.
- Smooth movement utilizing acceleration rather than rough position updates using integers.
- Dynamic AI behavior for engaging enemy combat.
- Fully animated sprites, including idle, attack, and death animations.

Objectives:

The goal is to complete the game by defeating all three bosses as quickly as possible while maintaining as much remaining health as possible. The final score is determined based on these factors and will be displayed on the scoreboard.

Key-Selling Point :

- Fully animated sprites for an immersive visual experience.
- Dynamic enemy AI behavior for fun and exciting gameplay.

<p>player</p>  <p>→ cap</p> <p>#very cool duck</p>	<p>boss1</p>  <ul style="list-style-type: none">- has the same move set to player- will always move the shortest path to be in his hitzone	<p>boss 2</p>  <ul style="list-style-type: none">- projectile user- always go far from player in certain before attack	 <ul style="list-style-type: none">- He is wizard that still don't want to loss bet- can attack the whole area- can change attribute of player make us faster, slower- Has magic ✨
---	---	---	---

3.2 Object-Oriented Programming Implementation

Config : contains configuration information that is not specific to any particular class. This module is referenced in the main game loop

EventHandler : acts as the central handler for majority event-related actions, including the handling of KEYUP and KEYDOWN events. This class manages event processing, allowing specific classes to call key events (ex. keyup, keydown) without embedding event-handling logic directly within the main game loop.

Player : defines the attributes and behaviors specific to the player character.

Enemy : serves as the parent class for all enemy-related entities, containing core attributes such as health and other relevant statistics.

Boss_Factory : acts as the central entity for creating various boss types. This class is responsible for instantiating specific boss entities, potentially including unique or "weird" boss designs (ex. bosses with HUGE sizes)

Boss1/ Boss2/ Boss3 : These classes define the behavior, health, and distinctive characteristics of each boss type.

Sprite_handle : serves as the central utility for managing and loading sprites for all game entities. This class handles the retrieval and management of sprites

GUI : operate GUI related functions. this module is responsible for rendering menus, buttons, and on-screen text

Information_record : stores detailed records of player progress and export to CSV file

Main : Manage the main loop, be the main core of the program

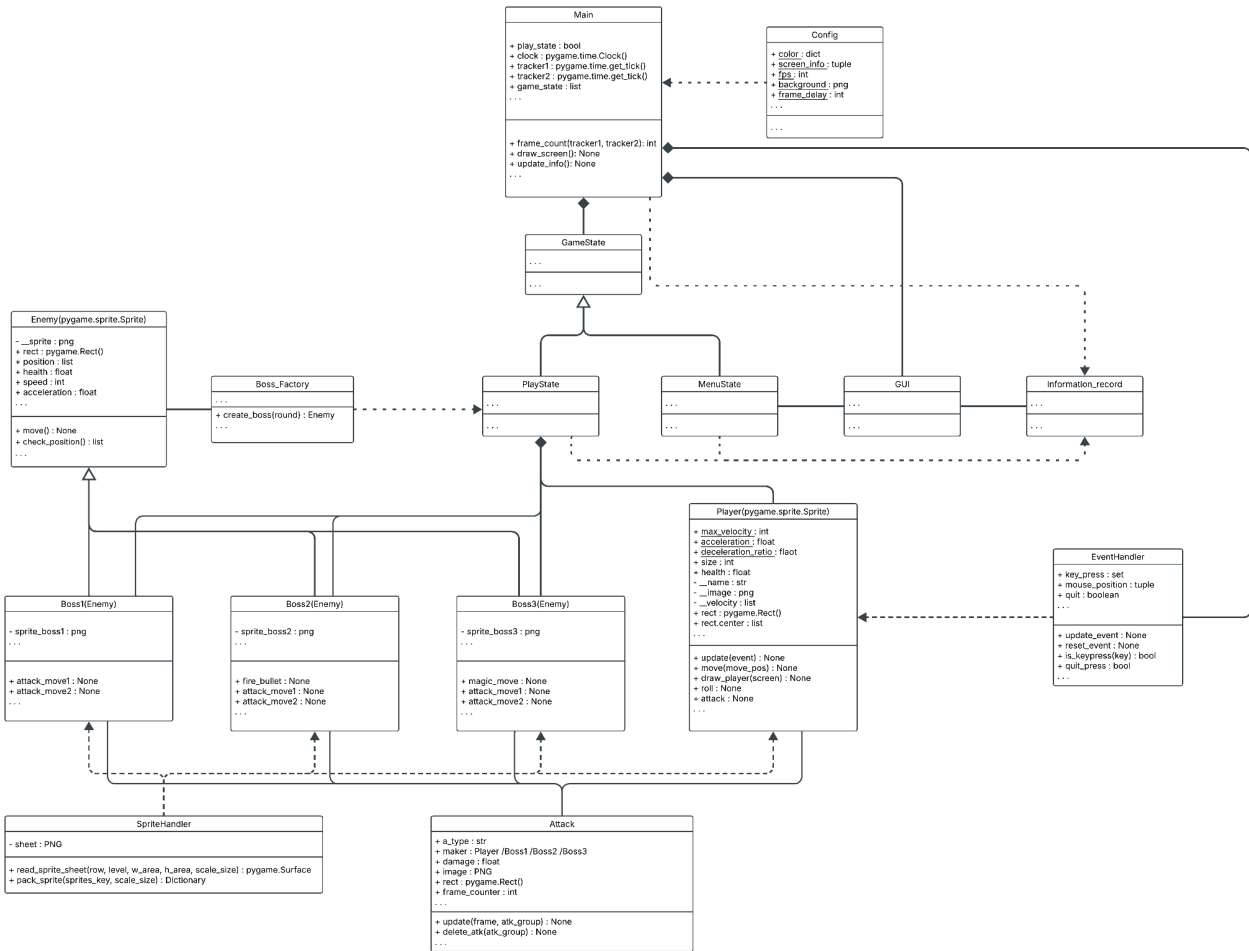
GameState : be the parent class more game state class provide some simple function and attribute

PlayState : Call to running the real gameplay

MenuState : Call to open the menu, score board etc.

Attack : defines the attributes and behaviors specific to the attack combat example type of attack is it "melee" or "range"

UML Link : <https://lucid.app/lucidspark>



3.3 Algorithms Involved

Note: Implementation is still in progress, but there are plans for its development.

Example

Boss1 : The AI will always seek the shortest path to reach its target hit zone, using calculations such as the Pythagorean theorem or basic point-to-point distance measurement.

Boss2 : also find the nearest paths but will always check when that shortest path is too short it will start running to find the safe location to fire

Boss3 : This boss does not walk but has the ability to teleport. He will continuously attack the player with a full-field attack. He remains in the same position as long as no player is within his circle radius. Once a player enters a specified radius, a countdown timer begins. When the timer reaches zero, the boss will teleport again.

4. Statistical Data (Prop Stats)

4.1 Data Features

1. The number of bosses the player successfully defeats in each run.
2. The total time taken to complete the run for the player who wins.
3. The amount of health remaining for the player who wins.
4. Time between each damage event (player or boss health reduce)
5. How the player's score changes when replaying the game.

	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 (via summarization statistics or via graph)
01	Motivate players to finish the game as an progression	Collect it myself & Ask my acquaintance to test my game project	Boss_Factory: when new Boss got build boss_number will update	summarization statistics (mode)
02	Create a competitive environment where players try to improve their time or compete with others.	Collect it myself & Ask my acquaintance to test my game project	Information_record build their own get_tick when start the program	Graph (Boxplot)
03	If too many players finish with high health, the game may be too easy. We can adjust bosses to hit harder and faster for more challenge	Collect it myself & Ask my acquaintance to test my game project	Check player.health when the boss_number = 3	Graph (Histogram)

04	If fights take too long without damage, players might spend too much time running or chasing, it might make the game boring so we can adjust character speed to help that	Collect it myself & Ask my acquaintance to test my game project	At mainloop of the game make if-else statement that check when the player or boss loss health get_tick() and send data to Information_record	summarization statistics (Average)
05	If completion times don't improve, the game may rely too much on luck and may need to adjust enemy behaviour	Collect it myself & Ask my acquaintance to test my game project	See the values of feather2 and 3	Graph (Scatter plot)

4.1.1 Summarization statistics data

- A. (1.)The number of bosses the player successfully defeats in each run.
 (4.) Time between each damage event.
- B. Feature1: “mode” for all 50 tests the number of boss that got defeated
 Feature4: “average” time in second for each hit for all 50 tests

4.1.2 Graphs data

	Feature Name	Graph Objective	Graph type	X-axis	Y-axis
Graph1	The total time taken to complete the run for the player who wins.	To motivate the player by showing which quarter they are in and how well they compare to others.	Boxplot	*only 1 boxplot Test data (all 50 data)	Time player take to win the game (second)
Graph2	The amount of health remaining for the player who wins.	To test that the game not to easy the Y-axis not to high	Histogram	Test number	The amount of health player has left (MAX:100.0)

Graph3	How the player's score changes when replaying the game.	To analyze the graph's trend and test whether winning the game depends more on skill than luck.	Scatter plot	How many times the same person replay the game	Player score at the end of the game
--------	---	---	--------------	--	-------------------------------------

4.2 Data Recording Method

A CSV file ,and it will also be used for the scoreboard system. Since the main goal of the game is to achieve the highest score, the scoreboard and competitive aspect are essential features.

4.3 Data Analysis Report

Number 1-4.

Testing will likely involve statistical measures such as variance or mean absolute deviation (MAD) to evaluate performance. Present by tables
Number 5.

To assess improvement, the range will be used as the primary measure. Present by Charts

5. Project Timeline

5.1 First weekly plan

Week	Task
1 (10 March)	Proposal submission / Project initiation/Basic pygame behavior: open window, receive input
2 (17 March)	Full proposal submission/ acceleration & smooth movement
3 (24 March)	Attack combo and hitbox ,Walk & attack sprite animation of player
4 (31 March)	Enemy Behaviour
5 (7 April)	Enemy sprite and animation
6 (14 April)	Submission week (Draft) / finalize and add sound

5.2 Second weekly plan

Week	Task
26 March - 2 April	Test sprites change mechanic with substitute sprites, Change global main game loop to be in class format
3-9 April	Implement game state, Build Enemy class(Boss parent class), Combat mechanic
10-16 April	Boss factory class, Boss behaviour
17-23 April	game menu and GUI, add sound , Collects information function & collect some test data
24 April - 11 May	All player, enemy, map sprites and check for bug, build the graph and table

5.2.2 List 50% of the tasks expected to be completed by 16 April.

- Open pygame window
- Player class
- Movement mechanic
- Load Sprite pictures mechanic
- Main game class
- Game state
- Boss1 simple function
- Combat mechanic
- Boss factory class
- Boss2, Boss3 simple function
- All Boss Behaviour

5.2.3 List 75% of the tasks expected to be completed by 23 April.

- Game menu & GUI
- Add sound
- Build collect information function

5.2.4 remaining 25% expect to complete by 11 May.

- Finish draw all game sprites
- Collect the information
- Build the Graph and Table
- Fix the bug

6. Document version

Version: 4.0

Date: 31 March 2025

[Pannawit Mahacharoensiri](#)

[6610545855](#)

Date	Name	Description of Revision, Feedback, Comments
14/3	Pattapon	Good job. :)
15/3	Phiranath	The game idea is very interesting. All of the topics are covered. Good job!
29/3	Phiranath	Some of the class relationships are missing, for example, Config class.