CoinDash

1. Project Overview

This project aims to develop a basic 2D platformer game using Python and Pygame. The game will involve a player-controlled character navigating through a single-level environment, avoiding obstacles, and reaching a goal. The mechanics will include jumping, running, and collecting simple items.

2. Project Review

An example of a 2D platformer is Super Mario Bros., where the player jumps over obstacles and reaches a goal. However, this project will focus on a minimalistic design with easy-to-implement mechanics such as basic movement, collision detection, and score tracking. The game will exclude advanced AI and complex level structures to ensure a straightforward development process.

3. Programming Development

3.1 Game Concept

The game will be a side-scrolling platformer where the player navigates through a single-level environment, avoiding obstacles and collecting coins to achieve the highest possible score. The player must jump over gaps, evade spikes, and avoid enemy objects while progressing. If they collide with an obstacle, fall into a gap, or are hit by an enemy, the game will be over. The game will track deaths based on their cause, such as falling or hitting an obstacle, to analyze difficulty. Since there is only one level, players must carefully time their movements and master the mechanics to successfully reach the goal.

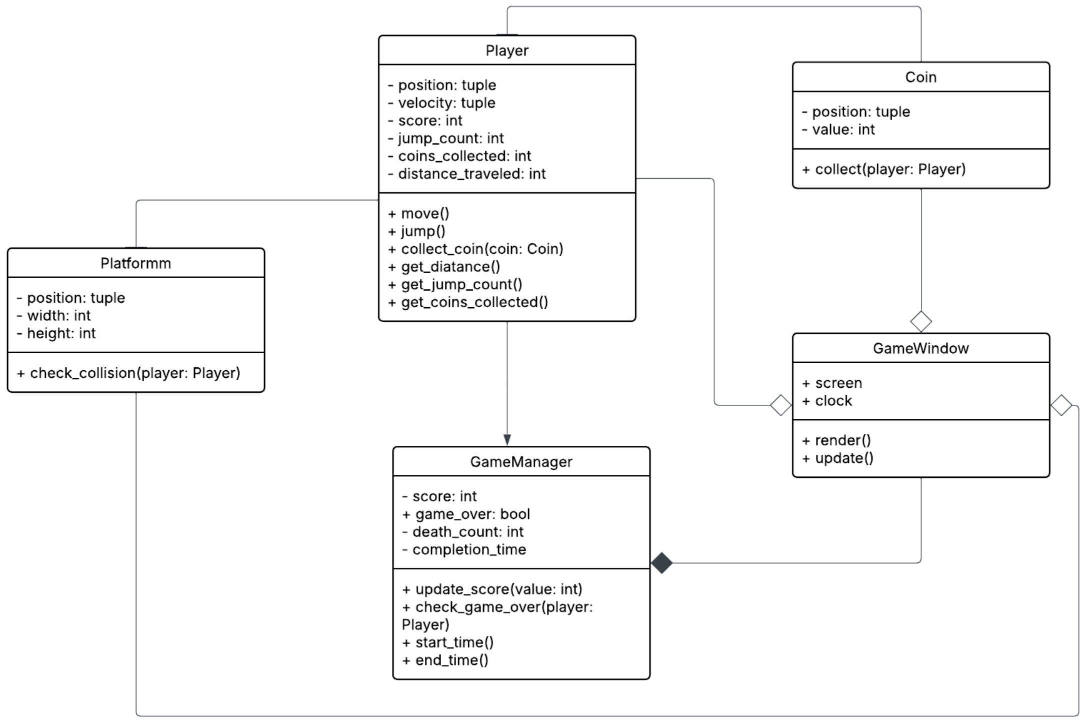
Key Features:

- character movement (jumping, running)
- Collision detection
- Collectibles (coins)
- A single-level environment
- Score tracking

<u>3.2 Object-Oriented Programming Implementation</u>

The game will implement the following five key classes:

1. **Player** – Handles movement and interactions.
    - Attributes: position, velocity, score, jump_count, coins_collected, distance_traveled
    - Methods: move(), jump(), collect_coin(),get_distance(), get_jump_count(), get_coins_collected
2. **Platform** – Represents solid surfaces the player can stand on.
    - Attributes: position, width, height
    - Methods: check_collision()
3. **Coin** – Represents collectible items.
    - Attributes: position, value
    - Methods: collect()
4. **GameManager** – Controls the game flow and score tracking.
    - Attributes: score, game_over, death_count, completion_time
    - Methods: update_score(), check_game_over(), start_timer(), end_timer()
5. **GameWindow** – Manages rendering and game loop.
    - Attributes: screen, clock
    - Methods: render(), update()



3.3 Algorithms Involved

- **Basic Physics**: Implementing gravity for smooth jumping.
- **Score Calculation**: Keeping track of collected coins.
- **Event Handling**: Responding to player input (jumping, movement).
- **Collision Detection**: Checking if the player lands on a platform or collects a coin.

4. Statistical Data (Prop Stats)

4.1 Data Features

	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 movement(distance traveled)	Measures how much the player moves, indicating engagement	Collect data every 10 seconds during gameplay.	Distance_traveled in Player class	Line graph showing how movement changes over time.
Coins collected	Determines effectiveness of level design	Count each time a player collects a coin	Coins_collected in Player class	Bar chart comparing number of coins collected per session
Completion Time	Helps analyze difficulty level	Track time from level start to finish	Completion_time in GameManager class	Histogram displaying frequency distribution of completion times

Deaths	Identifies difficult areas	Count deaths per session and record cause (falling, hitting obstacle)	Death_count in GameMange class	Pie Chart – Shows the proportion of each cause of death across all sessions.
Jump Frequency	Measures how often a player jumps, indicating skill and challenge	Count jump occurrences per session	Jump_count in Player class	Scatter plot showing the number of jumps per session..

Feature Name	**Statical Values**
Completion Time	Average, Min, Max, standard Deviation
Deaths	Frequency of each cause(falling, hitting obstacle)

	Feature Name	**Graph Objective**	**Graph Type**	**X-axis**	**Y-axis**
Graph 1	Player Movement (Distance Traveled)	Track player engagement	Line Graph	Time	Distance traveled
Graph 2	Jump Frequency	Analyze difficulty	Scatter Plot	Session Number	Number of jumps

		and skill level			
Graph 3	Coins Collected	Assess level design effectiveness	Bar Chart	Session Number	Coins collected per session
Graph 4	Deaths	Identify difficult areas	Pie Chart	Cause of Death	Frequency of Death
Graph 5	Completion Time	Analyze difficulty level	Histogram	Session Number	Completion time

4.2 Data Recording Method

The statistical data will be stored in a CSV file for easy analysis. Data will be collected at fixed intervals (every 10 seconds) and upon key events (e.g., collecting a coin, jumping, dying).

4.3 Data Analysis Report

The recorded data will be analyzed using statistics such as total counts and averages. The analysis will be presented through:

- Graphs showing player performance.
    - Line Graph: tracking player movement and jump frequency.
    - Bar chart: displaying coins collected per session for level design analysis.
    - Scatter Plot: Comparing jump frequency across multiple sessions to analyze difficulty.
    - Histogram: analyzing frequency distributions, such as time taken per session.

- Tables displaying collected data, including death causes and completion time statistics.

---

5. Project Timeline

Week	Task
1 (10 March)	Proposal submission / Project initiation
2 (17 March)	Full proposal submission
3 (24 March)	Initial game development (basic structure & setup)
4 (31 March)	Core mechanics implementation (movement, collision)
5 (7 April)	Additional features (collectibles, scoring system)
6 (14 April)	Submission week (Draft)
16 April (50%)	Basic game mechanics (movement, jumping, collision, scoring)
23 April (75%)	Death mechanics, data collection system, preliminary analysis
11 May (100%)	Full implementation, final analysis, and report submission

6. Document version

Version: 4.0
Date: 31 March 2025

Date	Name	Description of Revision, Feedback, Comments

15/3	Rattapoom	Very Good! But don’t forget to fill in the project timeline and de-italicize text.
16/3	Parima	The project is great in detail. Good Work.
29/3	Parima	Game Concept and Statistical Data should include more detailed information.
29/3	Rattapoom	