Pixel Chomp

1. Project Overview

This project tries to create a Pac-Man game with Python and the Pygame. The game contains a classic arcade-style maze in which the player controls Pac-Man to eat dots and avoid ghosts. Pac-Man's game mechanics include power-ups that allow him to eat ghosts for a brief time, different difficulty levels, and score tracking.

2. Project Review

An analysis of existing Pac-Man implementations, including the current Python-based Pygame version, highlights several areas for improvement:

- Improved pathfinding and movement patterns for ghosts.
- Add player statistics collection for performance insights.
- Add timer for track time usage each round
- Three difficulty levels with different gameplay mechanics

3. Programming Development

3.1 Game Concept

The objective of the game is to guide Pac-Man through a maze, collecting all the dots while avoiding ghosts. Power pellets scattered throughout the maze temporarily allow Pac-Man to eat ghosts for bonus points. The player wins by clearing all the dots in the maze. And the player loses if Pac-Man loses all his lives after being caught by ghosts.

Game Modes

1. **Easy Mode:** Standard Pac-Man gameplay.
2. **Normal Mode**: Increased ghost speed and reduced power-up duration.
3. **Hard Mode**: More ghosts, higher speed, and fewer power-ups.

3.2 Object-Oriented Programming Implementation

1. **PacMan (The player-controlled character)**
    - **Attributes:**
        - **position:** Current location on the grid
        - **speed:** Movement speed
        - **state:** Normal or powered-up
        - **lives:** Remaining lives
    - **Methods:**
        - **move():** Updates player position
        - **eat_dot():** Add score, removes dot
        - **eat_ghost():** Add score if powered-up
        - **power_up():** Enables temporary ability to eat ghosts
2. **Ghost (Enemy chasing the player)**
    - **Attributes:**
        - **position:** Current location on the grid
        - **speed:** Movement speed
        - **direction:** Current moving direction
        - **AI_state:** chase/scatter/frightened
    - **Methods:**
        - **move():** Pathfinding logic
        - **change_state():** Switches AI mode
        - **respawn():** Resets position after being eaten
        - **check_collisions():** Interacts with Pac-Man and walls
3. **Maze (The game map and items on it)**
    - **Attributes:**
        - **layout:** Maze structure
        - **dots:** Positions of collectible dots
        - **power_pellets:** Positions of power-ups
    - **Methods:**
        - **load_maze():** Loads maze from file
        - **check_collision():** Detects wall interactions
        - **remove_dot():** Removes collected items
4. **GameController (Manages game logic and rules)**
    - **Attributes:**
        - **game_state:** Running, paused, game over
        - **score:** Current player score
        - **timer:** Tracks time for session and power-ups

        - **game_mode:** Current difficulty level
    - **Methods:**
        - **start_game():** Initializes and starts a new game
        - **update_game_state():** Updates game logic on each frame
        - **check_win_condition():** Detects win/loss
        - **set_difficulty():** Adjusts settings in each mode
5. **StatisticsManager (Collects and analyze player data)**
    - **Attributes:**
        - **player_data:** Stores all gameplay statistics
        - **session_timer:** Measures the duration of each play session
    - **Methods:**
        - **record_data():** Tracks gameplay
        - **save_to_file():** Outputs to CSV
        - **generate_report():** Calculates stats

3.3 Algorithms Involved

- **Pathfinding**: A* or BFS algorithm for ghost movement.
- **Event-driven mechanics**: Handling keypresses and game events dynamically.
- **Timer Tracking**: Tracks total playtime and session duration.

4. Statistical Data (Prop Stats)

4.1 Data Features

The game will track the following player metrics:

1. Number of power pallets collected
2. Number of dots collected
3. Number of ghosts eaten
4. Survival time per game session
5. High score history

3.2 Data Recording Method

Statistical data will be stored in a CSV file

3.3 Data Analysis Report

Basic Statistics – Measures player performance using:

- Mean – Average dots collected per game in different game modes.

- Median – Median of survival time.
- Standard Deviation – Shows consistency in performance.

Performance Trends – Tracks player improvement over multiple sessions by analyzing:

- Score progression over time
- Survival time trends
- Ghost encounters

Visualization

- Line Graphs → score progression over sessions
- Bar Charts → comparing performance in different game modes
- Pie Charts → player vs. ghost ratio

4. Project Timeline

Week	Task
1 (10 March)	Proposal submission / Project initiation
2 (17 March)	Full proposal submission
3 (24 March)	Initial game prototype development
4 (31 March)	Implementation of AI
5 (7 April)	Implementing difficulty modes and statistical analysis integration
6 (14 April)	Submission week (Draft)

5. Document version

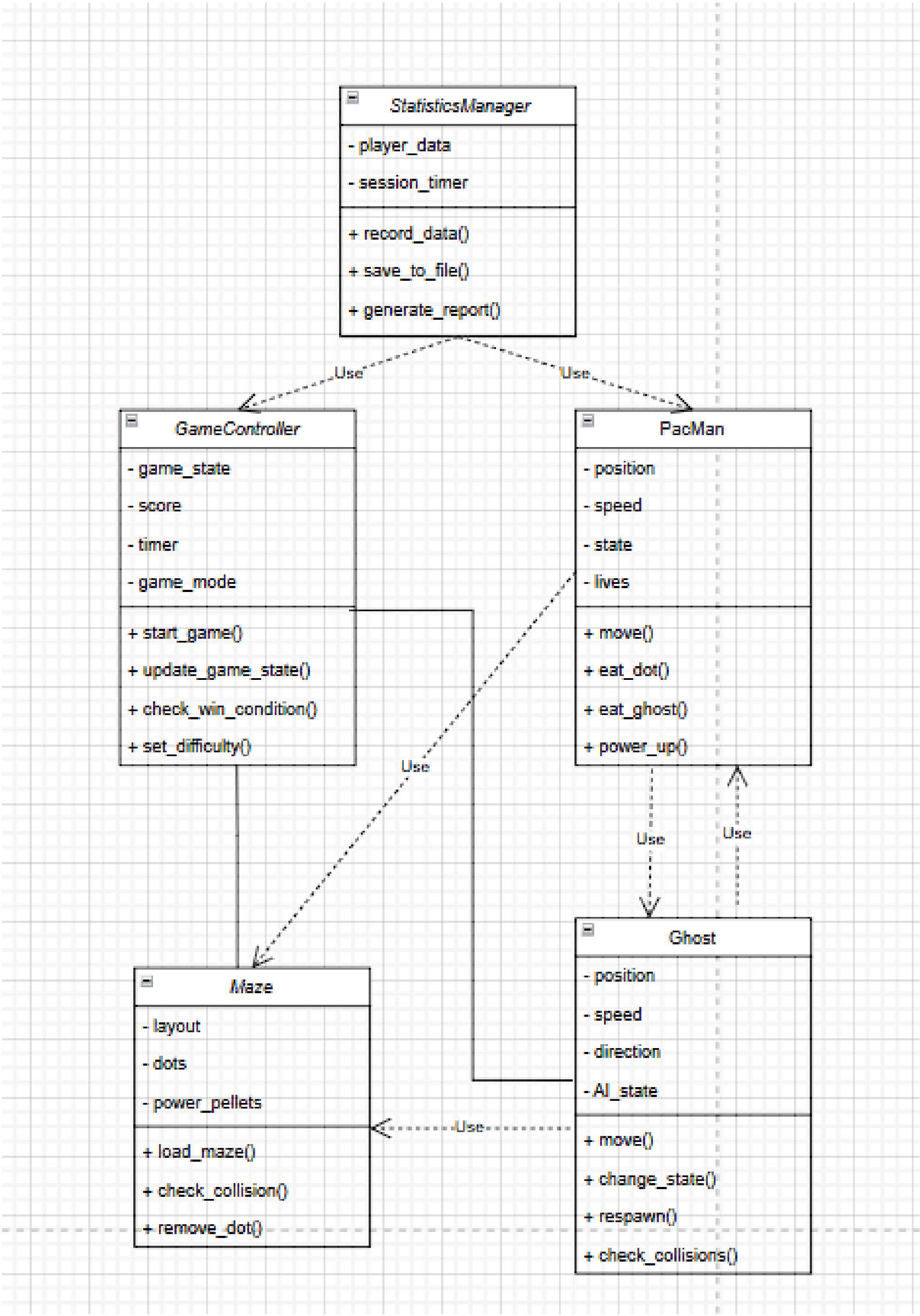
Version: *4.0*
Date: *31 March 2025*

Week	Goal	Milestone
1 (10 March)	Proposal submission / Project initiation	Proposal is submitted, project initiation complete
2 (17 March)	Full proposal submission	Full proposal is submitted and approved
3 (3 April - 9 April)	Initial game prototype development	Basic game prototype with core mechanics is developed
4 (10 April - 16 April)	Implementation of AI	Ghost AI implemented and functional (pathfinding, chase)
5 (17 April - 23 April)	Implementing difficulty modes and statistical analysis integration. Start final game testing.	Multiple difficulty modes implemented, statistical tracking integrated
6 (24 April - 11 May)	Final polishing, bug fixing, and documentation. Ensure final presentation and report submission.	Draft submission completed, final game polishing and testing done.

	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 (via summarization statistics or via graph)
~~Number of power pellets collected~~	~~Helps analyze how often players use power-ups and how they survive.~~	~~Track power pellet collection every time it collected~~	~~PacMan.power_pellets_collected (PacMan Class)~~	~~Bar Chart (Comparing power pellet collection across difficulty modes)~~
Number of dots collected	Measures player efficiency in clearing the maze and tracks performance.	Record the number of dots Pac-Man has eaten at 10 different timestamps per game	PacMan.dots_collected (PacMan Class)	Bar Chart (Comparing power pellet collection across difficulty modes)
Number of ghosts eaten	Indicates player's risk-taking behavior and effectiveness in using power-ups.	Track every ghost eaten at 10 different timestamps per game	PacMan.ghosts_eaten (PacMan Class)	Line Graph (Tracking ghost-eating trend over sessions)
Survival time per game session	Shows how long players last, helping analyze difficulty balance. Useful for tracking improvements in player skill over multiple sessions.	Record survival time every 10 seconds	StatisticsManager.session_timer (StatisticsManager Class)	Median, Standard Deviation
High score per game	Helps track player progression and performance trends over time. Used to compare skill levels in different difficulty modes.	Log the current score at 10 different timestamps per game	GameController.score (GameController Class)	Line Graph (Score progression over multiple sessions)
Player vs. Ghost Ratio	Analyzes player aggression vs. caution, power-up effectiveness, and difficulty balancing.	Track every ghost eaten and pac man eaten by ghost at 10 different timestamps per game	PacMan.eat_ghost() and PacMan.live (PacMan Class)	Pie Chart (comparing eaten ghosts vs. times eaten by ghosts)

	Feature Name	Graph Objective	Graph Type	X-axis	Y-axis

Graph 1	Number of dots collected	Measure player efficiency in clearing the maze	Bar Chart	Difficulty Mode	Dots Collected
Graph 2	Number of ghosts eaten	Track risk-taking behavior and power-up effectiveness	Line Graph	Number of Ghosts Eaten	Session Number
Graph 3	Survival time per game session	Assess player endurance and difficulty balance	Median, SD	Session Number	Survival Time (Seconds)
Graph 4	High score per game	Track progression and compare skill levels	Line Graph	Session Number	Score
Graph 5	Player vs. Ghost Ratio	Show ratio of players eating ghosts vs. being eaten	Pie Chart	Number of Pacman Eating Ghosts vs. Number of Ghost Eaten	Percentage of Total Interactions


StatisticsManager
- player_data
- session_timer
+ record_data()
+ save_to_file()
+ generate_report()
Use
Use
GameController
- game_state
- score
- timer
- game_mode
+ start_game()
+ update_game_state()
+ check_win_condition()
+ set_difficulty()
PacMan
- position
- speed
- state
- lives
+ move()
+ eat_dot()
+ eat_ghost()
+ power_up()
Use
Use
Use
Ghost
- position
- speed
- direction
- AI_state
+ move()
+ change_state()
+ respawn()
+ check_collisions()
Maze
- layout
- dots
- power_pellets
+ load_maze()
+ check_collision()
+ remove_dot()
Use

Date	**Name**	**Description of Revision, Feedback, Comments**
13/3	Rattapoom	Very good! Very detailed description about how to do Data Analysis part 👍
16/3	Parima	Your proposal has great detail.
28/3	Rattapoom	For “Number of power pellets collected”, how are you planning to “Track power pellet collection every time it is collected”? Playing the game 50 times may not be a good idea since it may be tedious.