TEAM VIRUS ARCADE HUB: PROJECT REPORT

Course: Computational Thinking and Programming (2025CSET100)
Project Title: Team Virus Arcade Hub (Python Game Menu)
Batch: 33 | **Team Number:** 03
Team Name: Team Virus
Date: November 2025

Team Members -----

1. **Sarthak** (Team Lead)
2. **Aman**
3. **Hunny**
4. **Srijan**
5. **Rahul**

1. Executive Summary (Abstract)

The "Team Virus Arcade Hub" is a modular, unified game application developed using **Python** and the **Pygame** library. The project's core objective was to solve the problem of fragmented game files by integrating four classic arcade titles—Snake, Pong, Space Invaders, and Flappy Bird—into a single, cohesive **Graphical User Interface (GUI)**. The central "Matrix-style" menu allows for seamless navigation. This solution effectively demonstrates key computational thinking concepts, including event-driven programming, collision detection (AABB), and object-oriented logic within a single executable environment, providing a superior and efficient end-user experience.

2. Introduction -----

2.1 Background & Motivation

Retro arcade games serve as a strong foundation for understanding game physics and logic. However, managing multiple standalone Python scripts is inefficient for users. Our goal was to package these experiences into a single, polished application to revive the nostalgia of retro gaming while improving usability.

2.2 Problem Statement

Beginner Python game projects typically exist as separate scripts, resulting in a poor user experience where different files must be manually opened and run. A critical lack of a unified, central interface to navigate these programs was identified.

2.3 Proposed Solution

We developed a "Master Hub" (Main Menu) that serves as the parent control loop. It uses Pygame's surface rendering for a graphical menu. Control is passed to the specific game function upon selection and then gracefully returned to the Main Menu upon "Game Over" or exit, preventing unexpected application crashes.

3. System Analysis and Design -----

3.1 Tech Stack

- **Language:** Python 3.x
- **Library:** `pygame` (for graphics rendering, event handling, and input management).
- **Standard Modules:** `sys` (System exit), `random` (RNG for enemy/food placement), `math` (Calculations).

3.2 Architecture

The project utilizes a modular functional approach structured into five primary blocks, ensuring clear separation of concerns:

1. **`main_menu()`**: The application entry point, handling the "Matrix Rain" visual effect and button navigation.
2. **`game_snake()`**: Manages grid-based movement, tail growth, and self-collision logic.
3. **`game_pong()`**: Implements physics-based ball reflection and simple AI opponent logic.
4. **`game_space_invaders()`**: Handles projectile motion and list-based enemy management and collision.
5. **`game_flappy_bird()`**: Manages gravity simulation and dynamic, scrolling pipe generation.

3.3 Program Flowchart Logic

- **Start:** Initialize Pygame and global constants (Screen Width, Colors, FPS).
- **Menu Loop:** Wait for mouse click events on specific button coordinates.
- **Game Loop:** If a game is selected, enter its specific `while True` loop.
- **Return:** Pressing ESC breaks the game loop via a return statement, falling back to the main Menu loop.

4. Implementation Details (Game Logic) -----

4.1 Main Menu & GUI

- **Visuals:** A "Matrix Digital Rain" effect, implemented with randomized falling characters, provides a cohesive "hacker/retro" aesthetic.
- **Interactivity:** A custom `draw_button()` function handles hover effects (Dark Green to Neon Green transition) and detects mouse clicks using `collidepoint()`.

4.2 Snake Game

- **Logic:** The snake is represented as a list of coordinates `[(x,y), (x,y)...]`. Movement involves inserting a new "head" and popping the "tail," unless food is consumed.
- **Collision:** Checks if the head coordinate overlaps with the wall boundaries or any coordinate present in the snake body list.

4.3 Pong

- **Physics:** Employs AABB (Axis-Aligned Bounding Box) collision detection.
- **Ball Movement:** Ball velocity has `dx` and `dy` components. Hitting a paddle (detected via `colliderect`) reverses the `x`-velocity.
- **AI:** The opponent paddle automatically tracks the ball's Y-position with a slight, intentional speed delay to ensure the opponent is beatable.

4.4 Space Invaders

- **Data Structures:** Enemies and player bullets are stored in separate lists of `pygame.Rect` objects.
- **Logic:** Nested loops manage collision detection. A collision (`bullet.colliderect(enemy)`) results in both objects being removed from their respective lists.
- **Movement:** Enemies move horizontally and shift down when they reach the screen edge.

4.5 Flappy Bird

- **Gravity:** A `bird_vel` variable increases each frame (simulating gravity) and is set to a negative value (jump strength) when the Space key is pressed.
- **Pipes:** Generated dynamically and stored in a list. Off-screen pipes are automatically removed to optimize memory usage.

5. Team Contributions

Team Member	Role	Primary Contribution
Sarthak	Team Lead	Main Menu GUI, Code Integration, Debugging, "Matrix" Effect Implementation.
Aman	Developer	Space Invaders logic (Enemy Movement, Bullet Lists, Collision).
Hunny	Developer	Snake Game algorithms (Tail Growth, Grid Movement, Self-Collision).
Srijan	Developer	Pong physics (Ball Reflection, AABB, AI Paddle Logic).
Rahul	Developer	Flappy Bird gravity simulation and dynamic pipe generation.

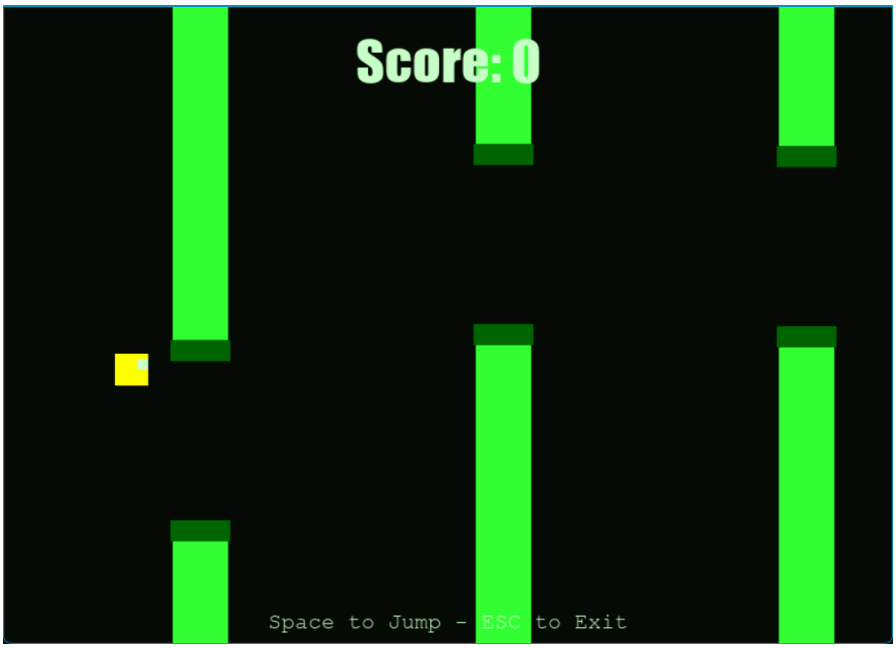
6. Results and Output

- **Unified Experience:** The project successfully launches all 4 distinct games from a central hub without operational errors.
- **Performance:** The application runs smoothly and consistently at **60 FPS** on standard lab machines.
- **Visuals:** A consistent "Neon Green" terminal theme is maintained across the menu and all four games.

Screenshots

- **Figure 1: The Main Menu Hub**
  - 

- **Figure 2: In-Game Screen**



7. Future Scope

- **Database Integration:** Implement SQL connectivity to save high scores permanently for all games.
- **Audio:** Add 8-bit sound effects (SFX) for key game actions, such as shooting and

jumping.
- **Multiplayer:** Implement LAN sockets to enable two players to play Pong or Space Invaders across different network computers.

8. References

1. Pygame Documentation: www.pygame.org/docs
2. Python Standard Library Documentation (Random, Math).
3. *Computational Thinking* Course Materials (Bennett University).

-----**Signature:** (Team Virus)