1 UP 25000

2 UP 003200

PACMAN

Name-Harsh Rajput
Panel-B-CSE
Roll No-39
PRN-1032222215

CREDITS

Introduction

- We delve into the creation of a Python-based implementation of the classic arcade game, Pacman.
- Originating in the 1980s, Pacman has remained a beloved icon in gaming history, captivating players with its simple yet addictive gameplay.
- I will also discuss the significance of creating Pacman in Python, both as an educational endeavor and as a demonstration of Python's versatility in game development.

Tools Used

- **Python:** Chosen for its simplicity, versatility, and extensive game development library support.
- **Pygame:** Primary framework for graphics rendering, input handling, and audio playback.
- **Visual Studio Code (VS Code):** Main IDE for code editing, debugging, and version control integration.
- **Object-Oriented Programming (OOP):** Structured game code for organization, modularity, and ease of maintenance.

Libraries used

- **Math Module:** Offers various mathematical functions and constants for efficient calculations within the Pacman game.
- **Copy Module:** Supports creating copies of objects, useful for maintaining immutability and preventing unintended side effects.
- **Pygame:** Popular library for game development, providing graphics rendering, event handling, and sound playback functionalities.
- **Board Module:** Custom module for game board-related operations such as maze generation and layout.
- **Ghost Images:** Visual representations of iconic Pacman ghosts, enhancing the game's graphical presentation and character.

Game Architecture

- **Components:**
  - Pacman, ghosts, pellets, and maze elements.
- **Game Loop:**
  - Controls game flow, updating and rendering the game state.
- **Input Handling:**
  -Captures user input and translates it into actions.
- **Rendering:**
  - Draws graphics and visual elements on the screen.
- **State Management:**
  - Tracks game state and manages transitions.
- **Event Handling:**
  -Manages events such as collisions and power-up activations.

Implementation Highlights

- **Player Movement:**

-Controls Pacman's navigation through the maze.
- Translates user input into movement commands.
- Handles collisions with maze walls and game elements.

- **Collision Detection:**

- Detects when Pac Man collides with maze walls, pellets, power-ups, and ghosts.
- Triggers appropriate actions based on collisions, such as stopping Pacman's movement or removing pellets.

- **Game State Management:**

- Tracks Pacman's position, score, lives, and maze state.
- Manages transitions between different game states, ensuring smooth gameplay progression.

Gameplay Mechanics

Gameplay mechanics entail the rules and interactions governing Pacman:

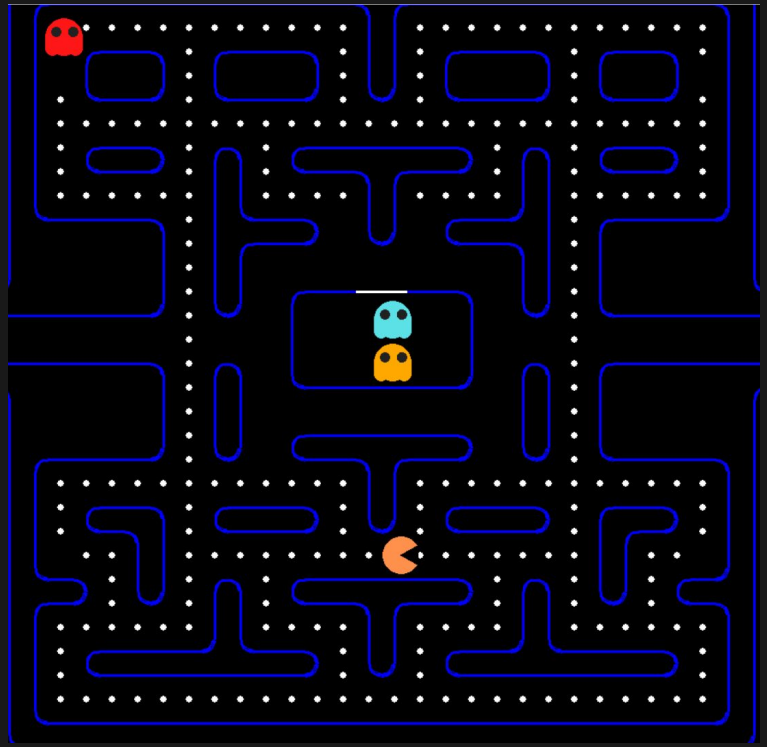
- **Pacman Movement:**
  - Player control of Pacman's navigation.
  - Movement restricted by maze walls.

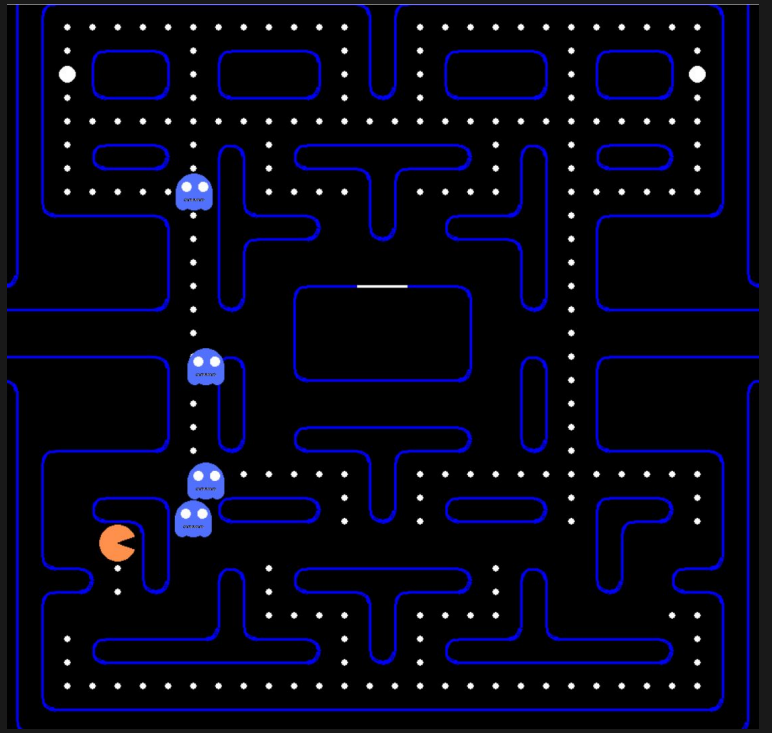
- **Pellets and Power-ups:**
  - Pacman collects pellets scattered in the maze.
  - Power-ups grant temporary invincibility.

- **Ghost Behavior:**
  - Ghosts roam with specific patterns and behaviors.
  - Collision results in Pacman losing a life.

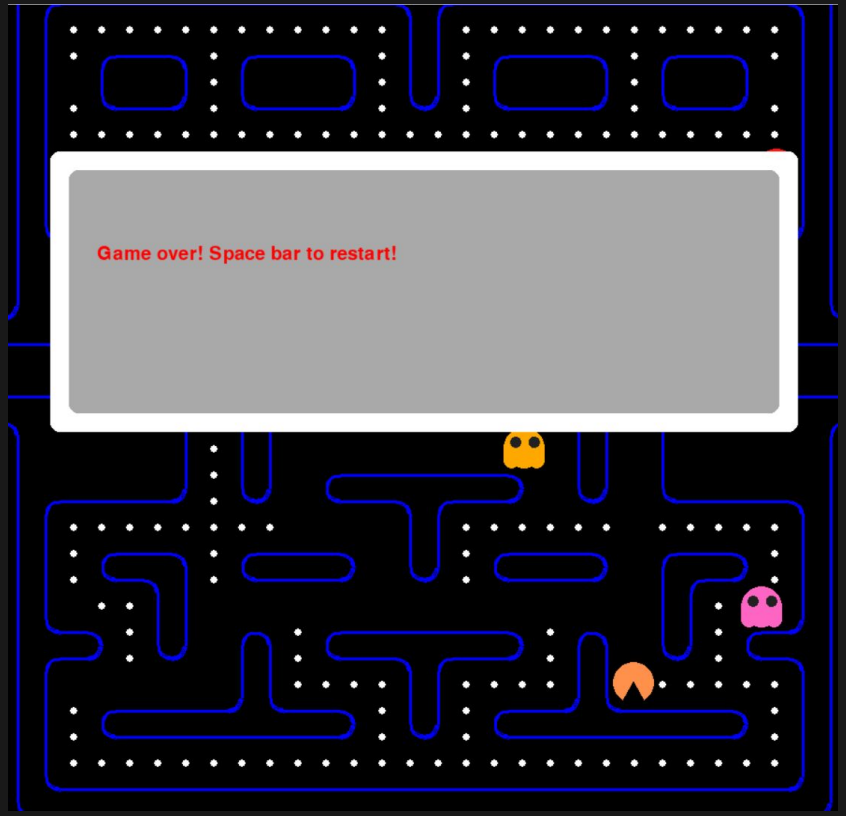
- **Lives:**
  - Pacman loses a life when colliding with a ghost or hazard.
  - Game concludes when all lives are lost.

Demonstration



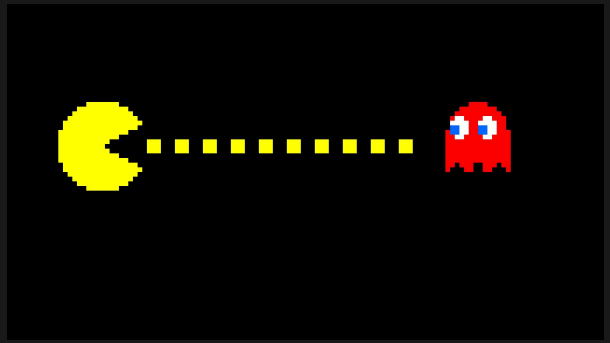


Demonstration



Conclusion

- Exciting journey: Developed Python-based Pacman game using Pygame and Python's flexibility.
- Capturing essence: Meticulously designed game with iconic maze layout and ghost movement patterns.
- Engaging gameplay: Crafted game to deliver enjoyable experience through thoughtful implementation.
- Sharing excitement: Looking forward to sharing game with others and exploring more game development projects.



THANK YOU