Preparatory Systems Software
CSCI 593
Fall 2024
Dr. Cavalcanti
Review or Training Project Assignment

Title

Creating Pac-Man Game using Python

Submitted by Group 13

Dinesh Seveti
Samuel Arne
Zak Groenewold

St. Cloud State University
Department of Computer Science

Date of Submission:11/24/2024

Table of Contents

Introduction

This project implements a simplified version of the classic Pac-Man game using Python and the pygame library. The game involves controlling Pac-Man to navigate through a maze, collect pellets, and avoid ghosts.

Features

- Interactive movement for Pac-Man using arrow keys.
- Four ghosts with randomized movements.
- Collision detection:
  - Pac-Man loses if caught by a ghost.
  - The game wins if all pellets are collected.
- Scoring system:
  - **10 points** per pellet.
  - **50 points** per large pellet.
- A fully rendered game board with walls, paths, pellets, and ghosts.

Game Logic

Pac-Man Movement

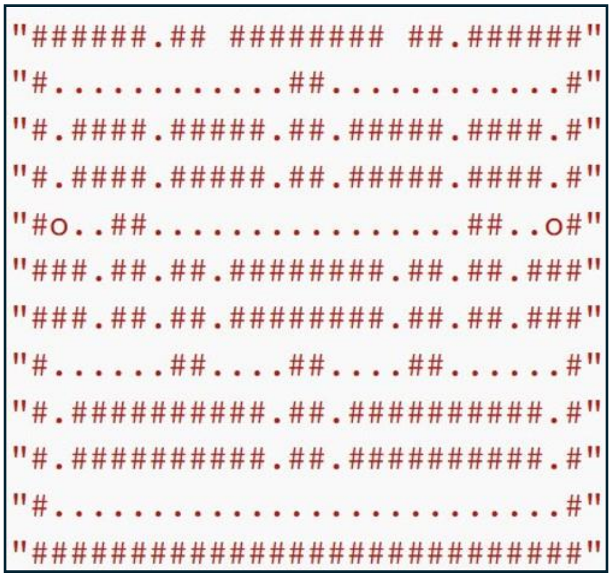
- Moves left, right, up, or down based on player input.
- Ensures movement is blocked by walls (# in the board grid).
- Collects pellets (. or o) and updates the score.

Ghost Movement

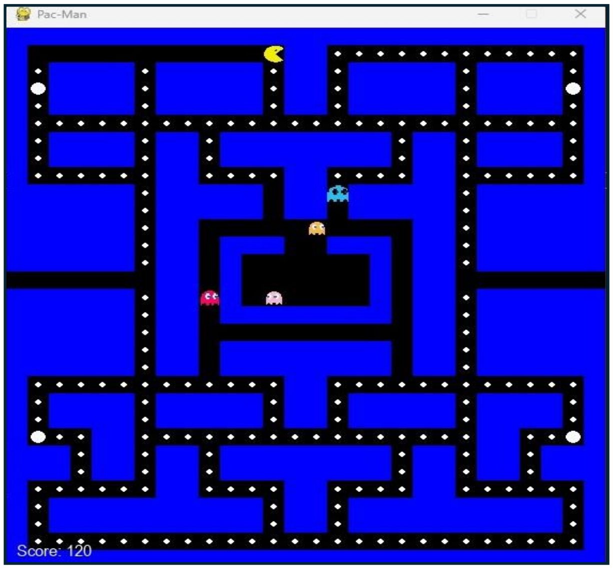
- Ghosts move in random directions.
- Avoid walls while navigating the maze.
- Positions are checked against Pac-Man's for collisions.

Collision Detection

- The game ends when Pac-Man's position matches any ghost's position.
- Winning condition met when all pellets are eaten.



```
"######.## ######## ##.######"
"#............##............#"
"#.####.#####.##.#####.####.#"
"#.####.#####.##.#####.####.#"
"#o..##................##..o#"
"###.##.##.########.##.##.###"
"###.##.##.########.##.##.###"
"#......##....##....##......#"
"#.##########.##.##########.#"
"#.##########.##.##########.#"
"#..........................#"
"############################"
```



Code Structure

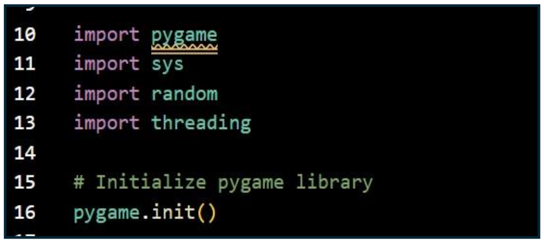
Core Components

- Game Board: 2D grid representing walls, paths, and pellets.
- Pac-Man: Controlled by the player using keyboard inputs.
- Ghosts: Move randomly within the maze.
- Score Tracking: Updates as Pac-Man collects pellets.

Main Functions

1. draw_board():
    - Renders the maze, walls, and pellets.
2. draw_pacman():
    - Displays Pac-Man at its current position.
3. draw_ghosts():
    - Displays ghosts at their respective positions.
4. move_pacman():
    - Updates Pac-Man's position based on input.
5. move_ghosts():
    - Randomly updates the positions of all ghosts.
6. check_collisions():
    - Checks if a ghost and Pac-Man occupy the same grid cell.
7. check_all_pellets_eaten():
    - Determines if the game is won.

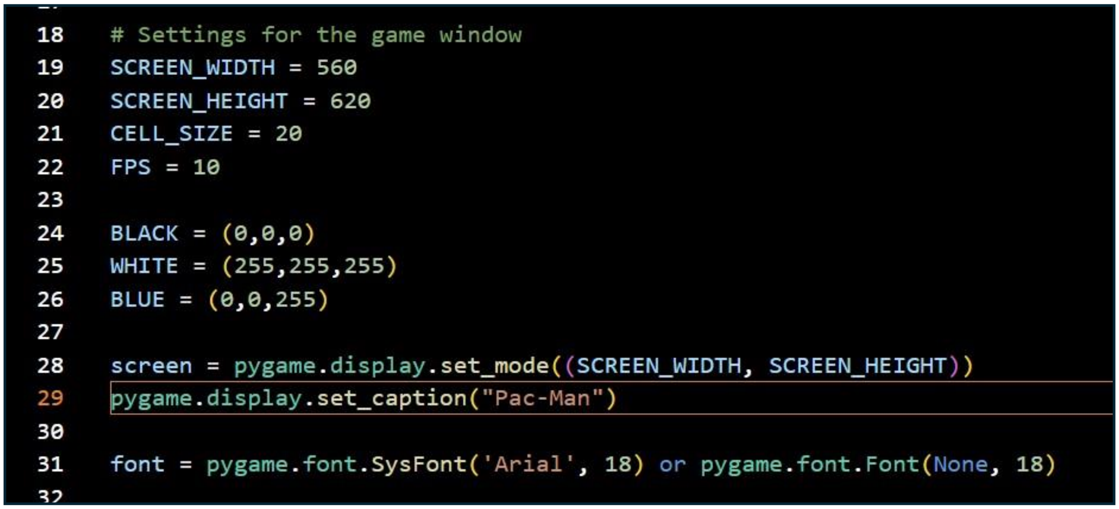
Library Import and Initialization



```
import pygame
import sys
import random
import threading

# Initialize pygame library
pygame.init()
```

pygame: Used for game development, providing tools for graphics, event handling, and input. sys: Allows the program to handle system-level operations (e.g., exiting the game). random: Used for random movement of ghosts. threading: Enables each ghost to move independently via multithreading.

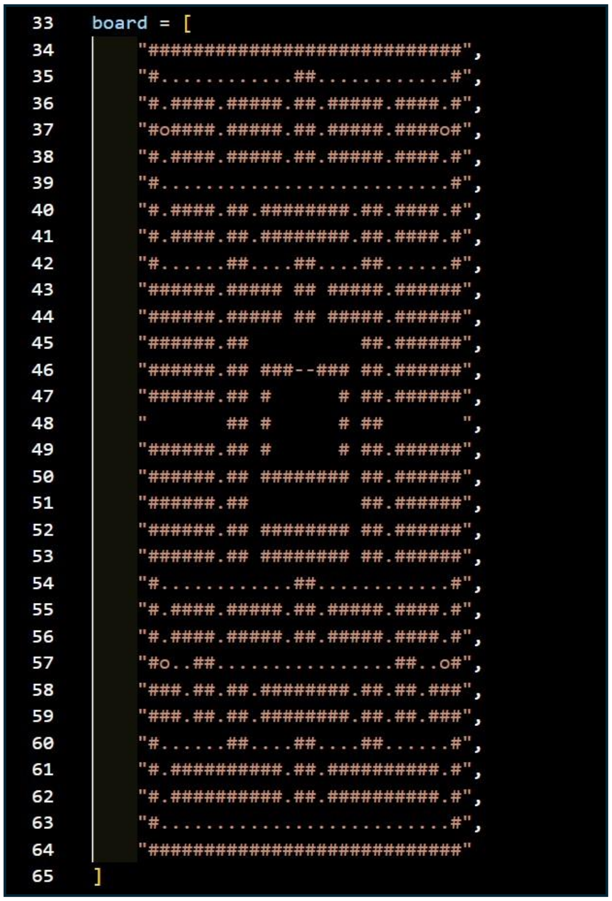
Screen Setup



```python
# Settings for the game window
SCREEN_WIDTH = 560
SCREEN_HEIGHT = 620
CELL_SIZE = 20
FPS = 10

BLACK = (0,0,0)
WHITE = (255,255,255)
BLUE = (0,0,255)

screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
pygame.display.set_caption("Pac-Man")

font = pygame.font.SysFont('Arial', 18) or pygame.font.Font(None, 18)
```

SCREEN_WIDTH and SCREEN_HEIGHT: Define the size of the game window. CELL_SIZE: Each grid cell is 20x20 pixels. FPS: Controls the game's speed (frames per second). RGB color definitions for rendering elements. Creates a game window and sets its title.

Game Board

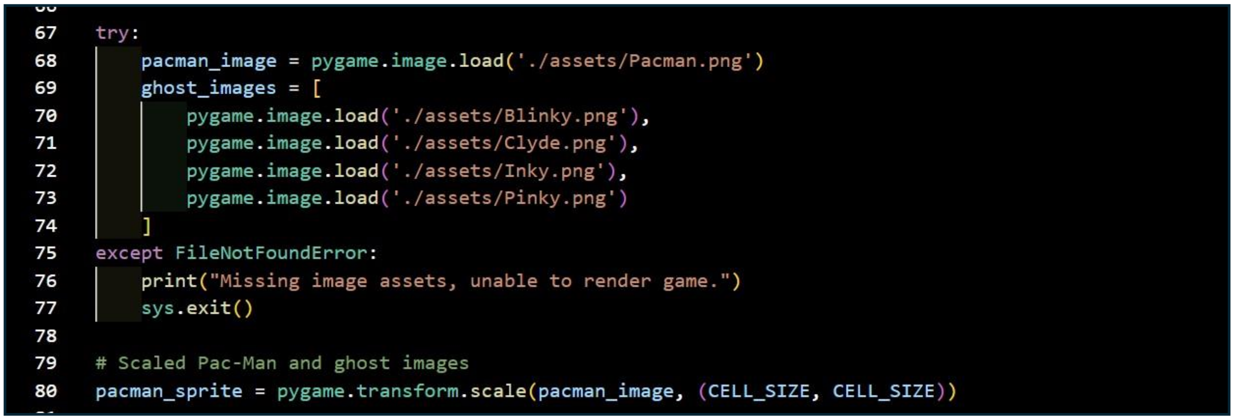


```python
board = [
        "############################",
        "#............##............#",
        "#.####.#####.##.#####.####.#",
        "#o####.#####.##.#####.####o#",
        "#.####.#####.##.#####.####.#",
        "#..........................#",
        "#.####.##.########.##.####.#",
        "#.####.##.########.##.####.#",
        "#......##....##....##......#",
        "######.##### ## #####.######",
        "######.##### ## #####.######",
        "######.##          ##.######",
        "######.## ###--### ##.######",
        "######.## #      # ##.######",
        "       ## #      # ##       ",
        "######.## #      # ##.######",
        "######.## ######## ##.######",
        "######.##          ##.######",
        "######.## ######## ##.######",
        "######.## ######## ##.######",
        "#............##............#",
        "#.####.#####.##.#####.####.#",
        "#.####.#####.##.#####.####.#",
        "#o..##................##..o#",
        "###.##.##.########.##.##.###",
        "###.##.##.########.##.##.###",
        "#......##....##....##......#",
        "#.##########.##.##########.#",
        "#.##########.##.##########.#",
        "#..........................#",
        "############################"
]
```

The game board is a grid where:

- #: Wall.
- .: Small pellet.
- o: Large pellet.
- : Walkable space.

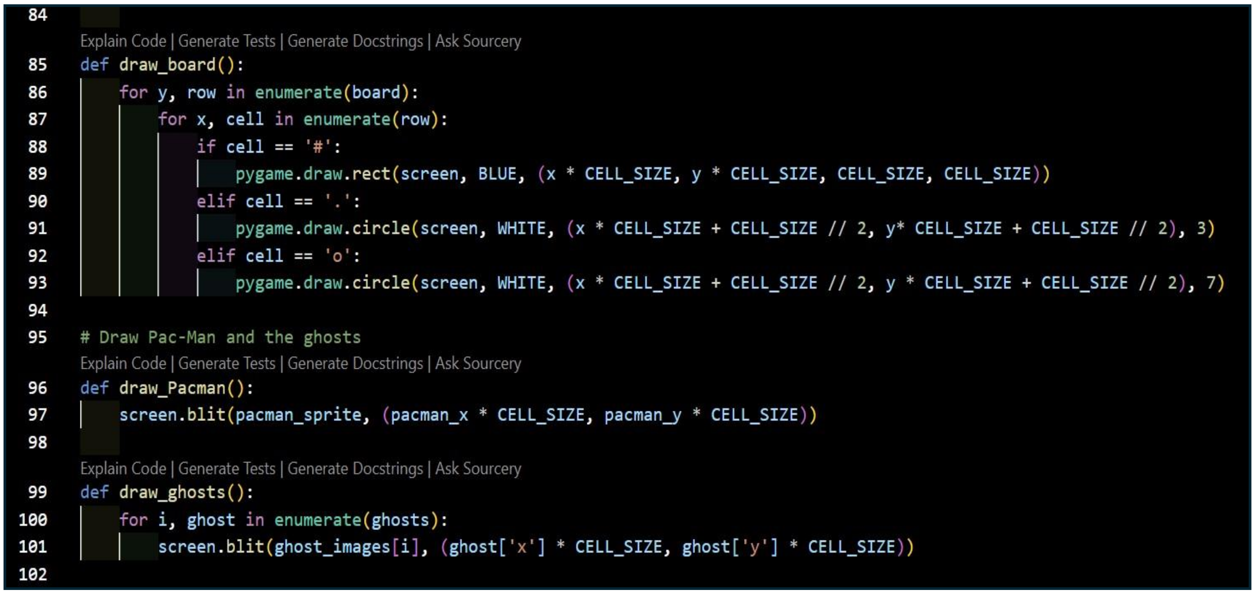
Loading and Scaling Assets



```python
try:
    pacman_image = pygame.image.load('./assets/Pacman.png')
    ghost_images = [
        pygame.image.load('./assets/Blinky.png'),
        pygame.image.load('./assets/Clyde.png'),
        pygame.image.load('./assets/Inky.png'),
        pygame.image.load('./assets/Pinky.png')
    ]
except FileNotFoundError:
    print("Missing image assets, unable to render game.")
    sys.exit()

# Scaled Pac-Man and ghost images
pacman_sprite = pygame.transform.scale(pacman_image, (CELL_SIZE, CELL_SIZE))
```

Loads images for Pac-Man and ghosts. If the images are missing, the program exists with an error. Scales the images to fit within the grid cells (20x20 pixels)

Drawing Functions

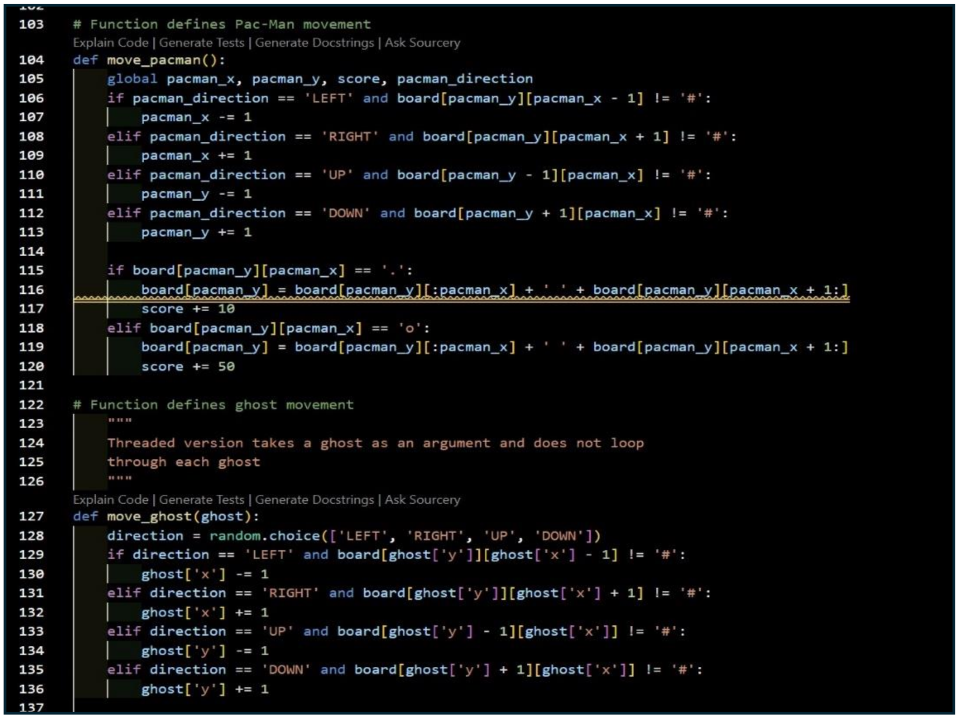

Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery
Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery
Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery

```python

def draw_board():
    for y, row in enumerate(board):
        for x, cell in enumerate(row):
            if cell == '#':
                pygame.draw.rect(screen, BLUE, (x * CELL_SIZE, y * CELL_SIZE, CELL_SIZE, CELL_SIZE))
            elif cell == '.':
                pygame.draw.circle(screen, WHITE, (x * CELL_SIZE + CELL_SIZE // 2, y* CELL_SIZE + CELL_SIZE // 2), 3)
            elif cell == 'o':
                pygame.draw.circle(screen, WHITE, (x * CELL_SIZE + CELL_SIZE // 2, y * CELL_SIZE + CELL_SIZE // 2), 7)

# Draw Pac-Man and the ghosts
def draw_Pacman():
    screen.blit(pacman_sprite, (pacman_x * CELL_SIZE, pacman_y * CELL_SIZE))

def draw_ghosts():
    for i, ghost in enumerate(ghosts):
        screen.blit(ghost_images[i], (ghost['x'] * CELL_SIZE, ghost['y'] * CELL_SIZE))

```

Iterates over the board grid and draws:

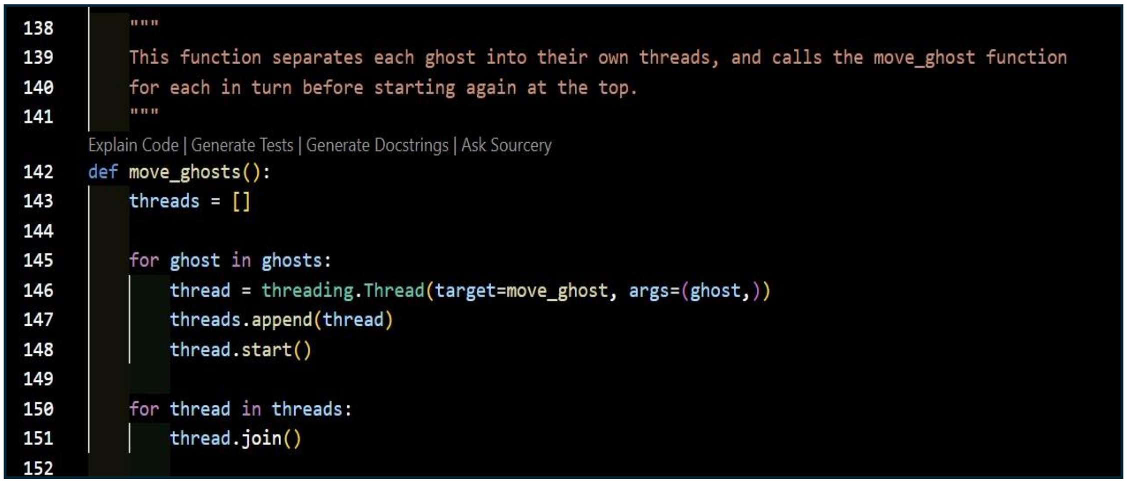
- Blue rectangles for walls (#).
- White circles for small pellets (.) and large pellets (o).
- Draws Pac-Man at its current position.
- Draws each ghost at its respective position.

Movement Functions



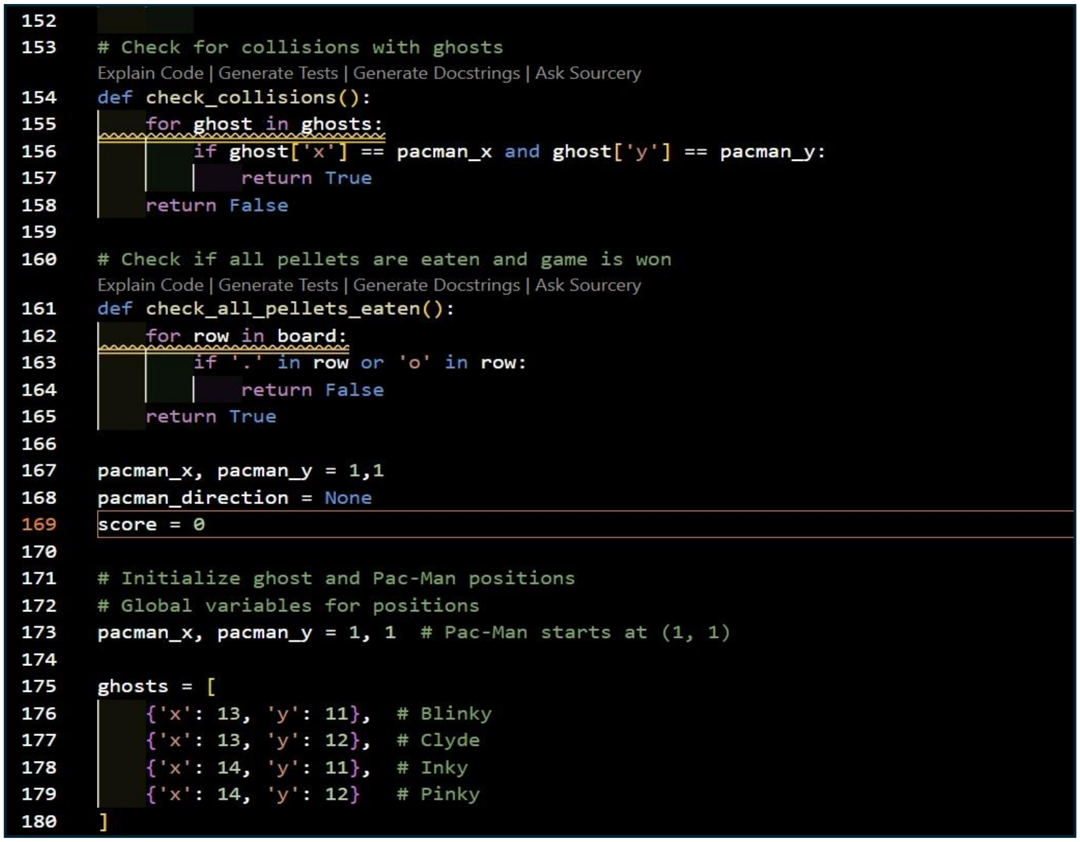
```
# Function defines Pac-Man movement
Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery
def move_pacman():
    global pacman_x, pacman_y, score, pacman_direction
    if pacman_direction == 'LEFT' and board[pacman_y][pacman_x - 1] != '#':
        pacman_x -= 1
    elif pacman_direction == 'RIGHT' and board[pacman_y][pacman_x + 1] != '#':
        pacman_x += 1
    elif pacman_direction == 'UP' and board[pacman_y - 1][pacman_x] != '#':
        pacman_y -= 1
    elif pacman_direction == 'DOWN' and board[pacman_y + 1][pacman_x] != '#':
        pacman_y += 1

    if board[pacman_y][pacman_x] == '.':
        board[pacman_y] = board[pacman_y][:pacman_x] + ' ' + board[pacman_y][pacman_x + 1:]
        score += 10
    elif board[pacman_y][pacman_x] == 'o':
        board[pacman_y] = board[pacman_y][:pacman_x] + ' ' + board[pacman_y][pacman_x + 1:]
        score += 50

# Function defines ghost movement
    """
    Threaded version takes a ghost as an argument and does not loop
    through each ghost
    """
Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery
def move_ghost(ghost):
    direction = random.choice(['LEFT', 'RIGHT', 'UP', 'DOWN'])
    if direction == 'LEFT' and board[ghost['y']][ghost['x'] - 1] != '#':
        ghost['x'] -= 1
    elif direction == 'RIGHT' and board[ghost['y']][ghost['x'] + 1] != '#':
        ghost['x'] += 1
    elif direction == 'UP' and board[ghost['y'] - 1][ghost['x']] != '#':
        ghost['y'] -= 1
    elif direction == 'DOWN' and board[ghost['y'] + 1][ghost['x']] != '#':
        ghost['y'] += 1
```

Updates Pac-Man's position based on the direction and ensures it doesn't pass through walls (#). If Pac-Man eats a pellet, the board updates to remove it, and the score increases. Randomly chooses a direction for each ghost and updates its position if no wall is in the way. Uses threads to move all ghosts simultaneously.



```
    """
    This function separates each ghost into their own threads, and calls the move_ghost function
    for each in turn before starting again at the top.
    """
Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery
def move_ghosts():
    threads = []

    for ghost in ghosts:
        thread = threading.Thread(target=move_ghost, args=(ghost,))
        threads.append(thread)
        thread.start()

    for thread in threads:
        thread.join()
```

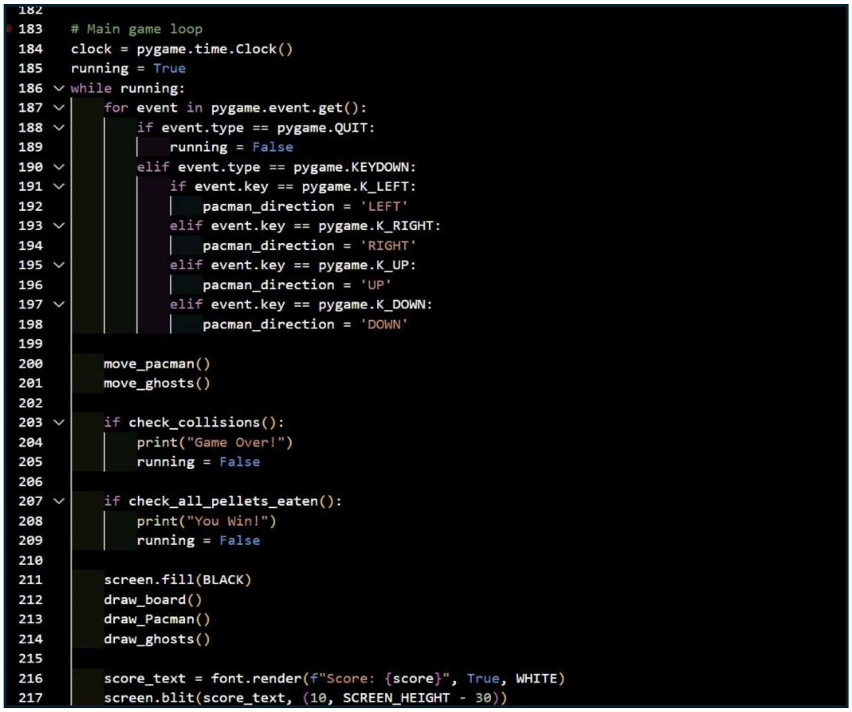
Collision Detection, Victory and Game Over


Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery
Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery

```python

# Check for collisions with ghosts
def check_collisions():
    for ghost in ghosts:
        if ghost['x'] == pacman_x and ghost['y'] == pacman_y:
            return True
    return False

# Check if all pellets are eaten and game is won
def check_all_pellets_eaten():
    for row in board:
        if '.' in row or 'o' in row:
            return False
    return True

pacman_x, pacman_y = 1,1
pacman_direction = None
score = 0

# Initialize ghost and Pac-Man positions
# Global variables for positions
pacman_x, pacman_y = 1, 1  # Pac-Man starts at (1, 1)

ghosts = [
    {'x': 13, 'y': 11},  # Blinky
    {'x': 13, 'y': 12},  # Clyde
    {'x': 14, 'y': 11},  # Inky
    {'x': 14, 'y': 12}   # Pinky
]
```

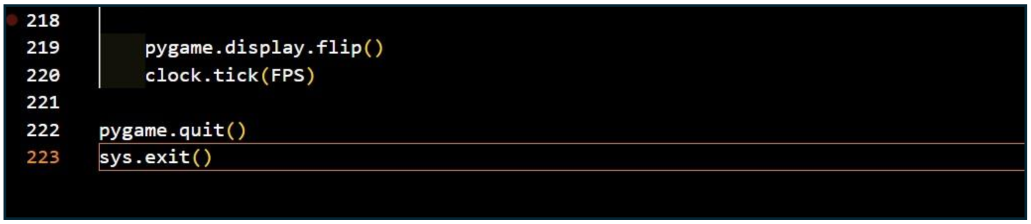
Checks if Pac-Man occupies the same cell as any ghost. Captures player inputs and updates Pac-Man's direction. Ends the game if Pac-Man collides with a ghost or all pellets are eaten.

Main Game Loop, Rendering and Frame Control



```python

# Main game loop
clock = pygame.time.Clock()
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                pacman_direction = 'LEFT'
            elif event.key == pygame.K_RIGHT:
                pacman_direction = 'RIGHT'
            elif event.key == pygame.K_UP:
                pacman_direction = 'UP'
            elif event.key == pygame.K_DOWN:
                pacman_direction = 'DOWN'

    move_pacman()
    move_ghosts()

    if check_collisions():
        print("Game Over!")
        running = False

    if check_all_pellets_eaten():
        print("You Win!")
        running = False

    screen.fill(BLACK)
    draw_board()
    draw_Pacman()
    draw_ghosts()

    score_text = font.render(f"Score: {score}", True, WHITE)
    screen.blit(score_text, (10, SCREEN_HEIGHT - 30))
```

Game End and Frame Control



```

    pygame.display.flip()
    clock.tick(FPS)

pygame.quit()
sys.exit()
```

Technologies Used

- Python: Programming language.
- Pygame: Game development library for rendering graphics and handling events.

Installation and Usage

- Prerequisites:
  - Python 3.x installed on your system.
  - Pygame library (pip install pygame).
- Steps to Run:
  - Clone the repository or copy the code.
  - Place required image assets (Pac-Man, ghosts) in an assets/ folder.
  - Run the script:

```
Python pacman.py
```

Future Improvements

- AI for Ghosts: Implement smarter ghost behavior using pathfinding algorithms.
- Levels: Add more complex mazes for higher levels.
- Power-Ups: Enable Pac-Man to eat ghosts temporarily after collecting a power pellet.

Conclusion

This Pac-Man game demonstrates fundamental programming concepts such as, Grid-based movement, Collision detection, Event handling, Game loop structure. The game leverages Pygame’s drawing and reding capabilities to create animations, sprite movements, and a visually engaging experience. The project is a fun and engaging way to learn Python and basic game development principles. It showcases essential game programming skills while leaving room for creativity and further exploration.