<InfPALS/>

Big Project

Session 1

Project Overview - Pac-Man in Pygame

- Session 1
  - Game loop
  - Drawing the board
  - Drawing the player
  - Player movement
- Session 2
  - Wall collisions
- Session 3
  - Implementing ghosts

Starter Files

GitHub

Understanding the Starter Files

- There are some numbers and functions given:
  - Constants:
    - Define fixed values like screen size, tile dimensions, and game settings.
  - Board:
    - Represents the maze layout using a 2D array (list of lists) with values indicating tiles (empty space, pellets, etc.).
  - Player class:
    - Encapsulates player properties (position, direction, speed, score) and methods for movement, drawing, and interaction with the board.
  - to_tile(x, y):
    - Converts a screen position (x, y) to its corresponding tile coordinates within the board array.
  - draw_board(screen, show_powerup):
    - Renders the maze layout on the screen based on the board data, potentially considering power-up visibility.
  - main():
    - The main loop that keeps the game running, handling user input, updating objects, rendering visuals, and controlling the frame rate.

Game loop

- At the bottom of the main function, implement the game loop:
- 1. WHILE the game is running
    - a. CHECK for any events (quitting the game)
    - b. UPDATE the positions and actions of all objects in the game
    - c. CLEAR the screen
    - d. DRAW the game elements (board, player, ghosts, score, etc.)
    - e. DISPLAY the updated screen
    - f. CALCULATE the time difference since the last frame
    - g. CONTROL the frame rate (e.g., aim for 60 frames per second)

Game loop – Hint

- **pygame.event.get():** Retrieves a list of events that have occurred (like key presses).
- **pygame.quit():** Quits the game.
- **player.move():** Updates the player's position based on user input.
- **player.actions(board):** Handles player interactions with the board (pellets, power-ups).
- **screen.fill():** Fills the screen with a background colour.
- **draw_board(screen, show_powerup):** Renders the maze layout on the screen. (This might be provided)
- **player.draw(screen):** Draws the player sprite on the screen.
- **pygame.display.update():** Updates the display to show the rendered graphics.
- **CLOCK.tick(fps):** Limits the game loop to a certain number of frames per second.

Draw the player

- In Player.draw()
    - 1. Draw the player's image onto the screen at its current position

Draw the player – Hint

- **screen.blit(image, rect):** Draws an image (player sprite) onto the screen at a specified position (rect).

Player movement

> In Player.move():

```
○ 1. Get user input (arrow keys)
○ 2. SET currentDirection = player's current direction
○ 3. IF horizontal movement is currently happening (left or right)
    ■ a. IF left key pressed AND possible to move left (check buffer zone around center)
        ● i. SET currentDirection = Left
    ■ b. IF right key pressed AND possible to move right (check buffer zone around center)
        ● i. SET currentDirection = Right
    ■ c. IF at a junction (check center coordinates within a buffer zone)
        ● i. IF up key pressed A. SET currentDirection = Up
        ● ii. IF down key pressed A. SET currentDirection = Down
○ 4. ELSE (vertical movement is currently happening)
    ■ a. IF at a junction (check center coordinates within a buffer zone)
        ● i. IF left key pressed A. SET currentDirection = Left
        ● ii. IF right key pressed A. SET currentDirection = Right
    ■ b. IF up key pressed AND possible to move up (check buffer zone around center)
        ● i. SET currentDirection = Up
    ■ c. IF down key pressed AND possible to move down (check buffer zone around center)
        ● i. SET currentDirection = Down
```

Player movement – Continued

- In Player.move():
    - 5. UPDATE player's direction based on currentDirection
    - 6. IF currentDirection is Left
        - a. DECREASE player's X position by speed
    - ELSE IF currentDirection is Right
        - a. INCREASE player's X position by speed
    - ELSE IF currentDirection is Up
        - a. DECREASE player's Y position by speed
    - ELSE IF currentDirection is Down
        - a. INCREASE player's Y position by speed
    - 7. Handle wrapping around the maze (check if going off one side, appear on the other)

Player movement – Hint

- **player.direction:** Stores the player's current direction (left, right, up, down).
- **player.speed:** Defines the speed at which the player moves per frame.
- **player.rect:** Represents the player's position and size as a rectangle.
- **pygame.key.get_pressed():** Returns a dictionary indicating which keys are currently pressed.

Player actions

> In Player.actions():

```
1. IF player is on a pellet
    a. INCREASE player's score by pellet value
    b. REMOVE pellet from the maze
2. IF power pellet eaten
    a. SET player to powered-up state
    b. START power-up timer
ELSE IF powered-up state is active
    a. DECREASE power-up timer
    b. IF timer runs out
        i. SET player to normal state
```

Player actions – Hint

- **board[y][x]:** Accesses the value (empty space, pellet, power-up) at a specific tile on the board (y = row, x = column).
- **player.score:** Stores the player's current score.
- **player.powered_up (boolean):** Indicates if the player is currently in a powered-up state.
- **time.time():** Gets the current system time in seconds (used for power-up timers).