<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).

Big Project

Session 2

Pacman movement in the maze

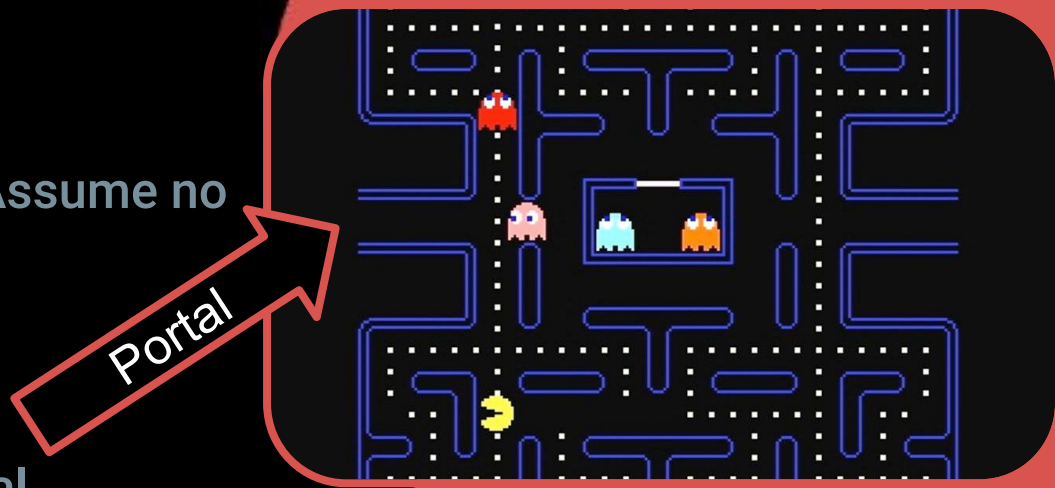
- Checking adjacent tiles for walls
- Colliding with a wall
- Moving in a straight line and at junctions
- Moving the rect (based on last week)
- Moving around the edge of the maze

wall_check(checking for wall left, right, up and down)

Function that takes centerx and centery of pacman and looks_ahead several pixels

```
def wall_check(x, y, look_ahead, can_enter_gate):
```

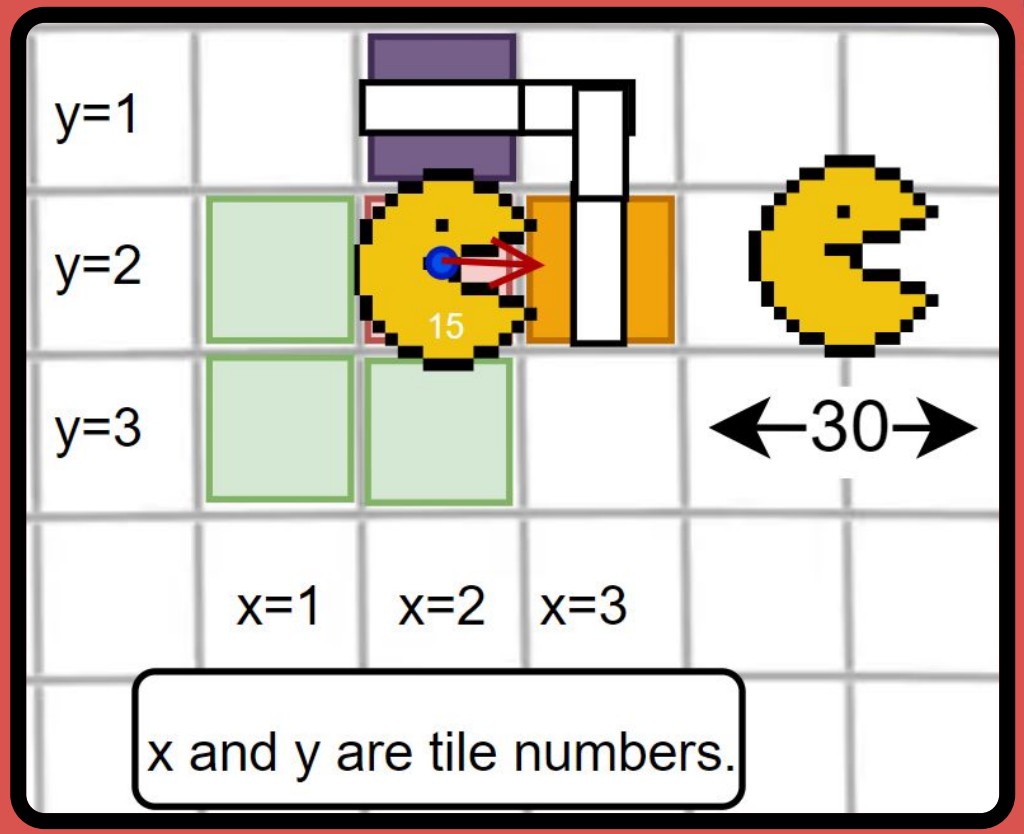
1. GET tile_x and tile_y of pacman is in using to_tile()
2. IF tile is within left and right limits of board
   a. SET walls to LIST of FALSE FALSE FALSE FALSE (Assume no walls)
   b. CHECK WALLS (later slide)
3. ELSE
   a. // We are going around the board through the portal
   b. SET walls to left and right is FALSE and up and down is TRUE
4. RETURN walls



Hint

- Variables x and y are the centerx and centery of pacman
- When tile_x is 0,1,2, .. TILES_WIDE-1, x is on board
- walls is a list of 4 booleans

CHECK WALLS



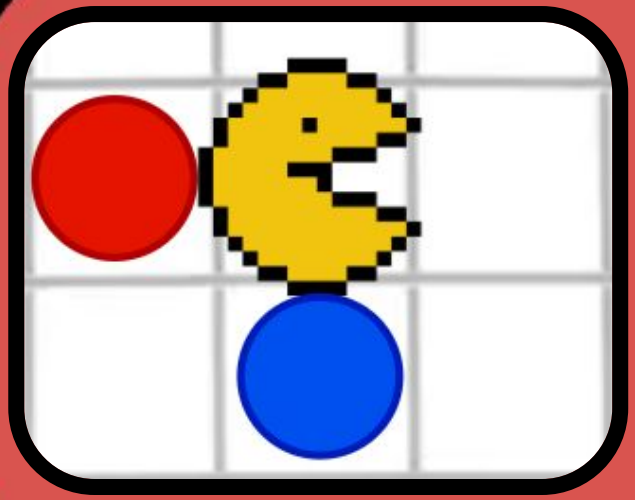
What value should look_ahead be?

(After SET walls to FALSE,FALSE,FALSE, FALSE)

1. GET tile number of tile look_ahead pixels Left, Right, Up and Down using to_tile()
2. IF can enter ghost house gate (gate=9)
   a. IF Left tile in board contains (3,..8)
   b. SET wall[0] to TRUE
   c. (Add contains conditions for Right - wall[1], Up - wall[2] Down - wall[3])
3. ELSE
   a. IF Left tile in board contains (3...9)
   b. SET wall[0] to TRUE
   c. (Add for Right - wall[1], Up - wall[2] Down - wall[3])

Hint:

1. Left_tile_x, _ = to_tile(x - look_ahead,y) returns tile x of the tile with Red dot. Red dot has same tile_y as pacman
2. to_tile(x, y + look_ahead) returns tile y of the tile below (Blue dot).
3. board[left_tile_x, this_tile_y] returns contents of tile to the left
4. board[left_tile_x, this_tile_y] > 2 returns true if Left tile contains 3,4,5,6,7,8,9,...



```
# 0 = empty black rectangle, 1 = dot, 2 = big dot, 3 = vertical line,
# 4 = horizontal line, 5 = top right, 6 = top left, 7 = bot left, 8 = bottom right
# 9 = ghost house gate
```

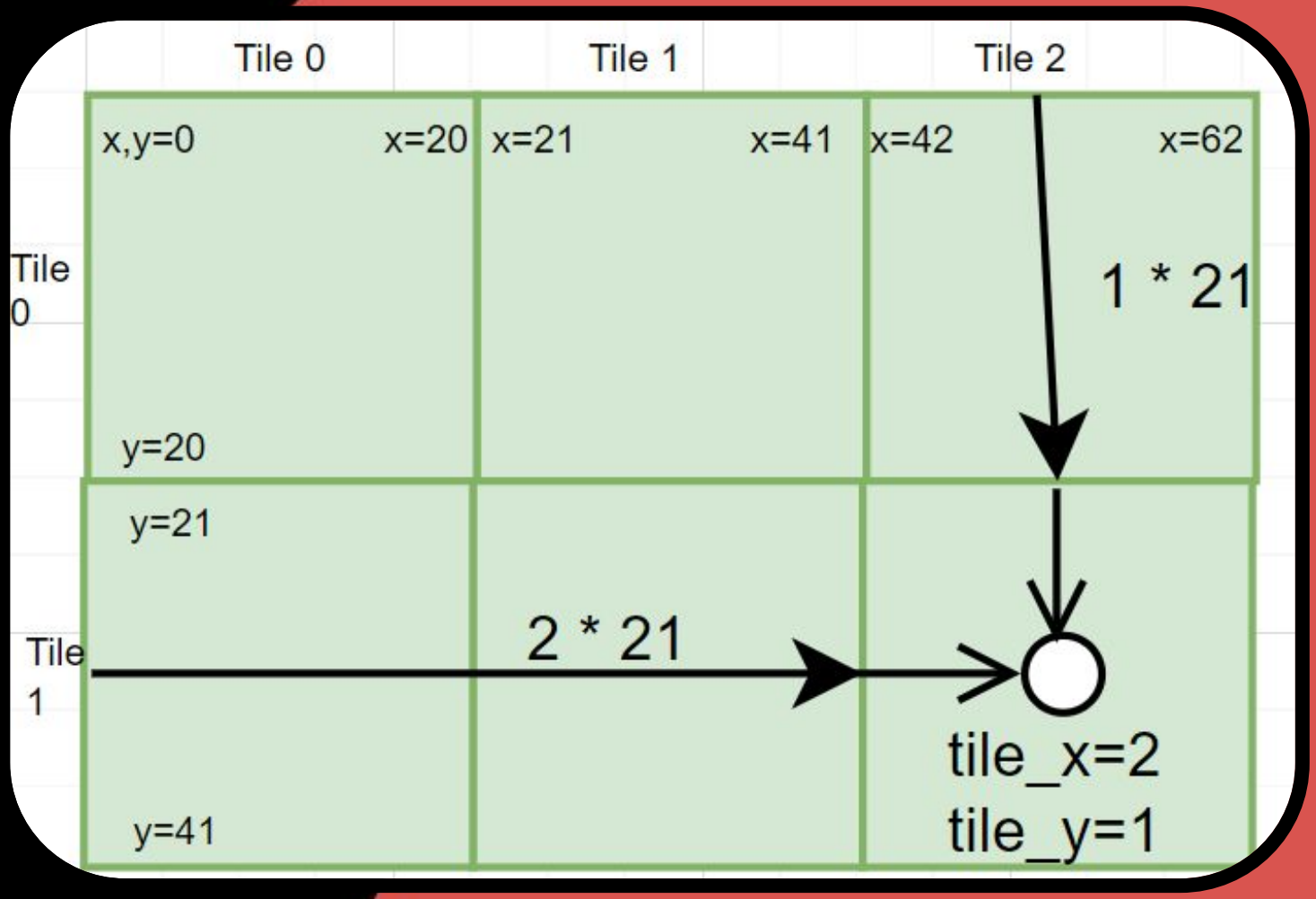
Player class > Move

1. GET keys pressed
2. GET walls
3. IF current direction has wall in same direction
    a. GET this_tile_x, this_tile_y pacman is in.
    b. // Set pacman's center to centerpoint of this tile
    c. SET centerx of pacman to x pixel coordinate of the tile. = TILE_WIDTH * this_tile_x + (width of a tile // 2). Integer division is //
    d. SET centery of pacman to y pixel coordinate of tile pacman is in.
4. SET horizontal = True if moving Left or Right
5. SET vertical = True if moving Up or Down
6. SET Lower= 7 upper = 13

Hint

If pacman is in the tile with white dot.

The centerx of the tile is found by multiplying tile number (tile 2) by width of each tile then add half.



Hint

pygame.key.get_pressed() returns dictionary of keycode to (True/False) values

Get walls with wall_check(pass in centerx and centery of pacman, look_ahead=15 and pacman cannot enter the gate)

Horizontal = (self.direction == ???) or (self.direction == ???)

Move

1. GET keys pressed
2. GET walls
3. IF direction currently facing has a wall
    a. GET TILE (this_tile_x, this_tile_y) pacman is in
    b. SET centerx of pacman to pixel coordinate of this_tile_x PLUS HALF the width of a tile
    c. SET century of pacman to pixel coordinate of this_tile_y PLUS HALF the height of a tile
4. SET horizontal = True if moving Left or Right
5. SET vertical = True if moving Up or Down
6. SET Lower= 7 upper = 13
7. **IF horizontal // Set direction (later slide)**
8. **IF vertical // Set direction (later slide)**
9. // MOVE rect in direction IF no wall in that direction (later slide)
10. IF outside board

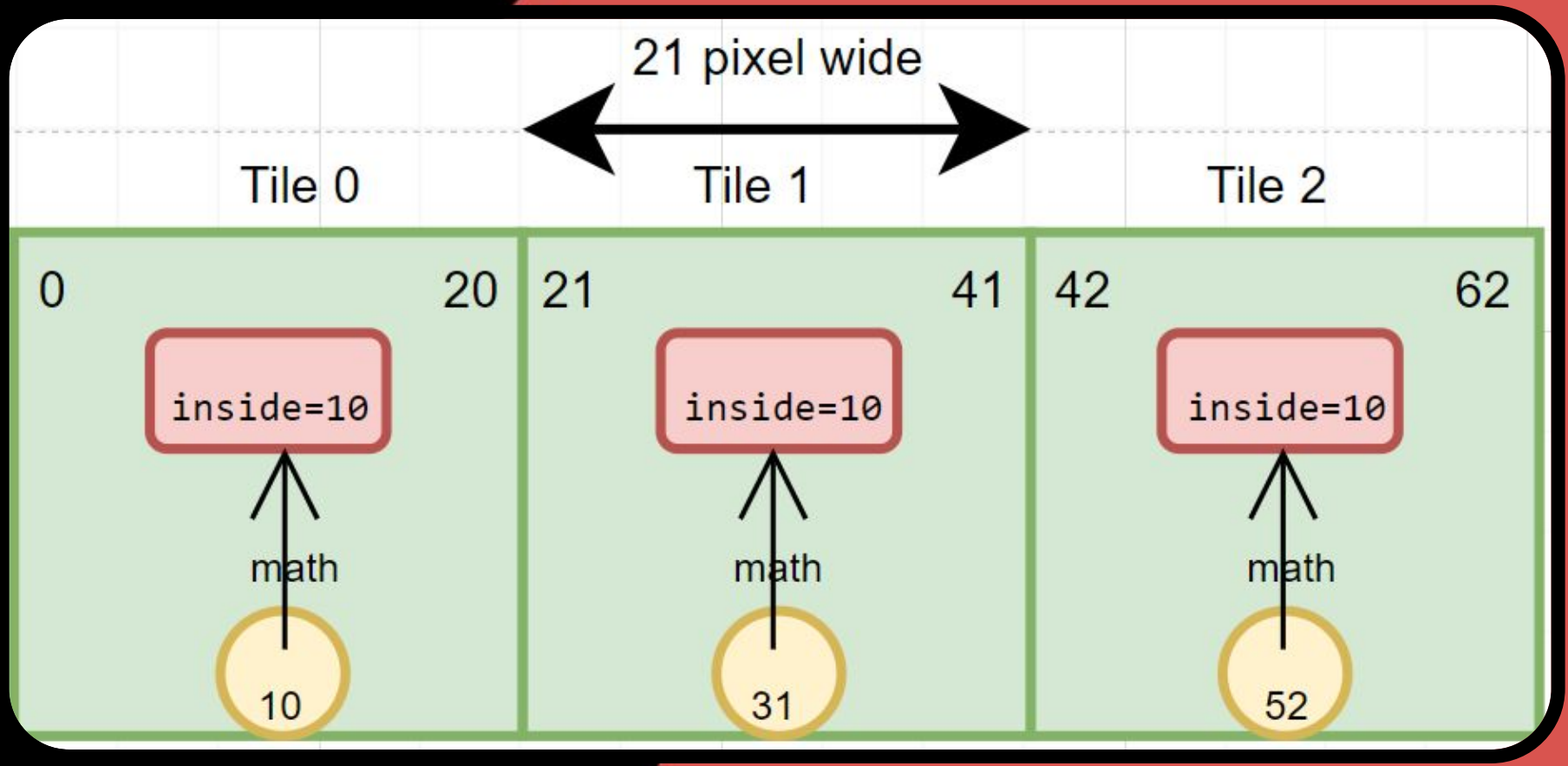
IF HORIZONTAL - set direction

1. (IF HORIZONTAL)
2. IF PRESS A and NO WALL LEFT //Player requests to go left and no wall left then lets go left
    a. SET direction to LEFT
3. IF PRESS D and NO WALL RIGHT
    a. SET direction TO (...)
4. // When turning at a junction, you must be roughly in the center of a tile to turn.
5. IF lower <= SOME_MATH(centerx, TILE_WIDTH) <= upper
    a. IF PRESS W and NO WALL UP
        i. SET direction to UP
    b. IF PRESS S and NO WALL (...)
        i. SET direction TO (...)

Hint (do not proceed to answer on next slide)

The math

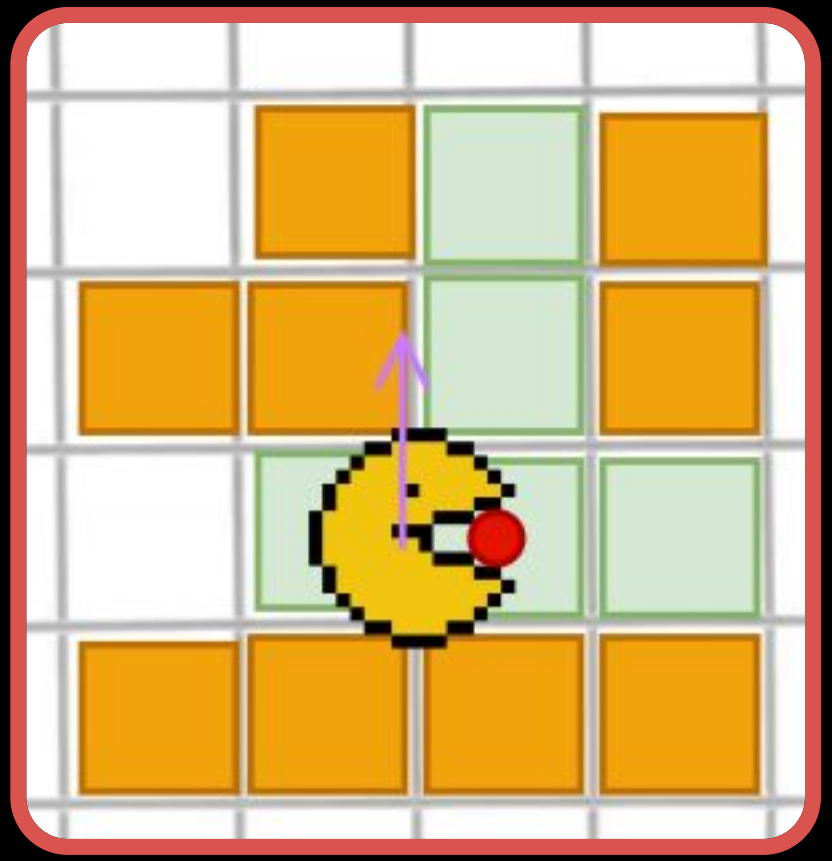
If pacman's position inside a side must be near the center of 10. So between 7 and 13. How do we convert pacman's position (yellow) to the "inside" position



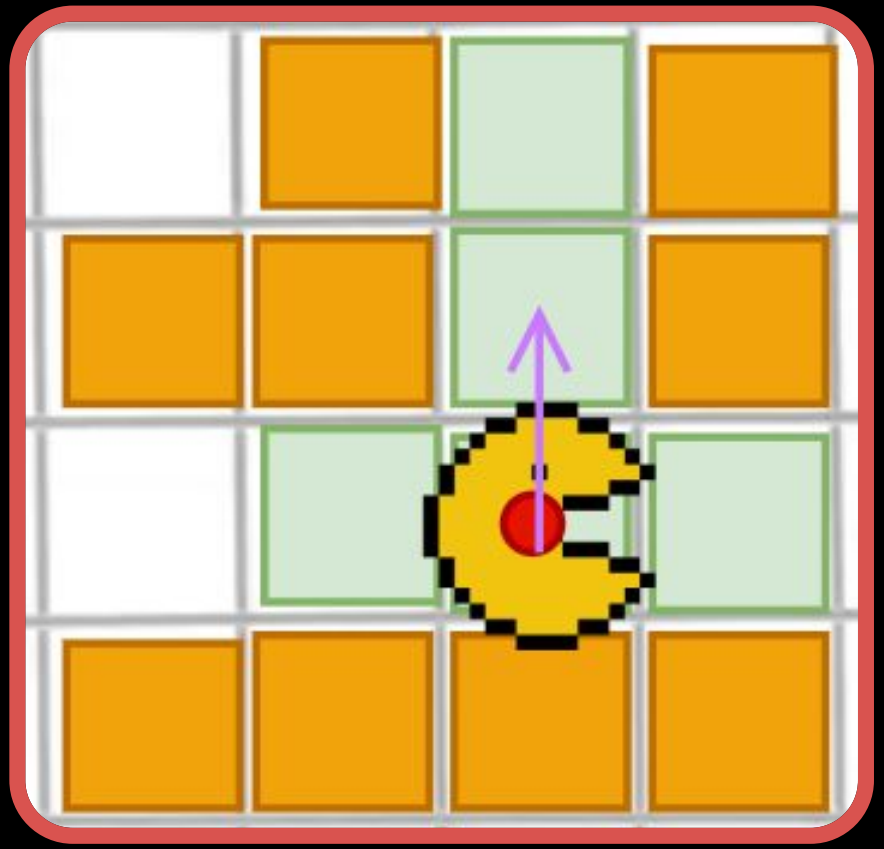
Answer to Math

```
if lower_bound <= (self.rect.centerx % TILE_WIDTH) <=
upper_bound:
```

Junction reasoning



Pacman should not go up until he is in the center of a junction



Pacman is at the center of a junction so can go up

IF VERTICAL - set direction

1. (IF VERTICAL)
2. // When turning at a junction, you must be roughly in the center of a tile to turn.
3. IF lower <= SOME_MATH(centery, TILE_HEIGHT) <= upper
    a. IF PRESS A and NO WALL LEFT
        i. SET direction to LEFT
    b. IF PRESS D and NO WALL RIGHT
        i. SET direction TO RIGHT
4. IF PRESS W and NO WALL UP
    a. SET direction to UP
5. IF PRESS S and NO WALL DOWN
    a. SET direction TO DOWN

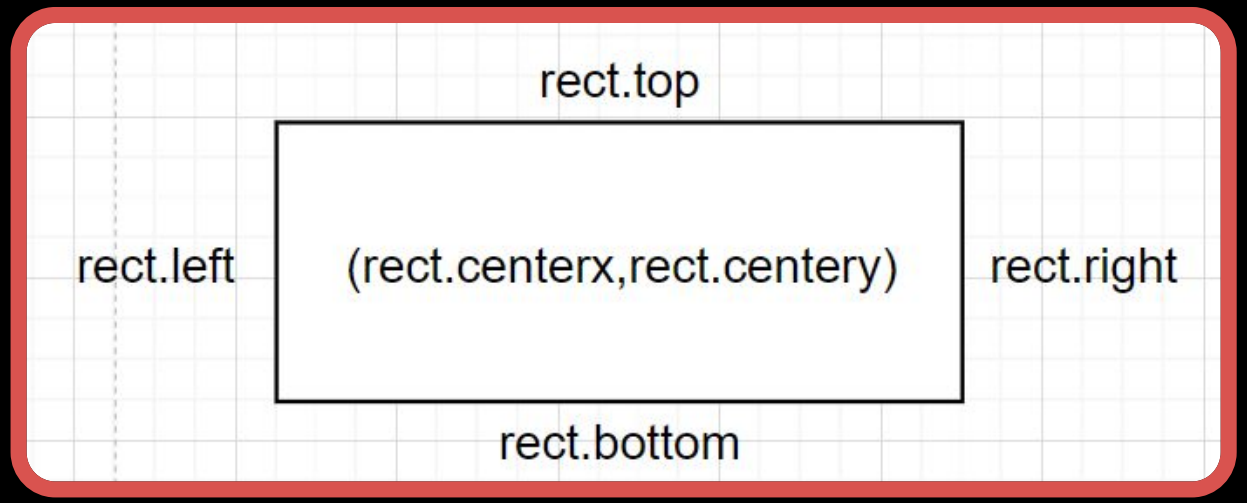
Move

1. GET keys pressed
2. GET walls
3. IF direction currently facing has a wall
    a. GET TILE (this_tile_x, this_tile_y) pacman is in
    b. SET centerx of pacman to pixel coordinate of this_tile_x PLUS HALF the width of a tile
    c. SET century of pacman to pixel coordinate of this_tile_y PLUS HALF the height of a tile
4. SET horizontal = True if moving Left or Right
5. SET vertical = True if moving Up or Down
6. SET Lower= 7 upper = 13
7. IF horizontal // Set direction (later slide)
8. IF vertical // Set direction (later slide)
9. **// MOVE rect in direction IF no wall in that direction (recall last week)**
10. **IF outside board**

Move and loop around the board via portal

```
// based on last week
IF direction is LEFT and no wall LEFT
        REDUCE centerx by speed
IF direction is RIGHT and no wall RIGHT
        INCREASE centerx by speed
(Write UP and DOWN for centery)
// looping around the board via portal
IF RIGHT EDGE of pacman < 0
        SET RIGHT EDGE of pacman to width of maze
IF LEFT EDGE of pacman > width of maze
        SET RIGHT EDGE of pacman to 0
```

Hint:



Pygame rect (explanation behind self.rect)