Pac-Man Game Code - Line by Line Explanation

Import Statements and Color Constants

```python
import random
from termcolor import colored
```

- `random`: Used for ghost movement - ghosts move in random directions
- `termcolor`: Library for adding colors to terminal output

```python
# Constants for colors
COLOR_WALL = "blue"
COLOR_GHOST = "red"
COLOR_PACMAN = "yellow"
COLOR_PILL = "grey"
COLOR_FINAL_WIN = "green"
COLOR_FINAL_LOSE = "red"
```

- Define color schemes for different game elements
- `COLOR_FINAL_WIN/LOSE`: Colors used when the game ends

ASCII Art for Game Elements

```python
ui_wall = [
    ".....",
    ".....",
    ".....",
    "....."
]
```

- 4-line ASCII art representing walls (dots create a solid block effect)

```python
ui_ghost = [
    " .-. ",
    "| OO| ",
    "|   | ",
    "'^^^' "
]
```

- Ghost representation with eyes (OO) and a wavy bottom

```python
ui_hero = [
    " .--. ",
    "/ _.-'",
    "\\  '-.",
    " '--' "
]
```

- Pac-Man representation (looks like a circular character)

```python
ui_empty = [
    "      ",
    "      ",
    "      ",
    "      "
]
```

- Empty space (6 spaces per line, 4 lines)

```python
ui_pill = [
    "      ",
    " .-.  ",
    " '-'  ",
    "      "
]
```

- Small pill that Pac-Man needs to collect

Map Creation Function

```python
def create_map():
    return [
        "|--------|",
        "|G..|..G.|",
        "|...PP...|",
        "|G....@|.|",
        "|...P...|.|",
        "|--------|"
    ]
```

- Creates the initial game map as a list of strings

- `|` and `-`: Walls
- `G`: Ghosts (3 total)
- `@`: Pac-Man starting position
- `P`: Pills to collect (3 total)
- `.`: Empty spaces

Map Display Function

```python
def print_map(game_map, final_color=None):
    for row in game_map:
        for piece in range(4):
            for point in row:
                color = None
```

- `final_color`: Optional parameter to override all colors (used for win/lose screen)
- Triple nested loop: rows → 4 lines per character → each character in the row

```python
                if point == 'G':
                    color = COLOR_GHOST if not final_color else final_color
                    print(colored(ui_ghost[piece], color), end='')
```

- If character is 'G', print the ghost ASCII art in red (or final_color)
- `piece` selects which of the 4 lines of ASCII art to print
- `end=''` prevents automatic newline

Similar logic follows for walls (`|`, `-`), Pac-Man (`@`), empty spaces (`.`), and pills (`P`).

```python
            print()
    print()
```

- After each set of 4 lines, print a newline
- Extra newline at the end for spacing

Ghost Movement Function

```python
```

```
def move_ghosts(game_map):
    game_finished = False
    ghosts = []
    for x, row in enumerate(game_map):
        for y, cell in enumerate(row):
            if cell == 'G':
                ghosts.append([x, y])
```

- Find all ghost positions and store them in a list
- `enumerate` gives both index and value

python

```
    for ghost in ghosts:
        old_x, old_y = ghost
        directions = [[old_x, old_y+1],[old_x+1, old_y],[old_x, old_y-1],[old_x-1, old_y]]
        nx, ny = random.choice(directions)
```

- For each ghost, define 4 possible moves: right, down, left, up
- Randomly choose one direction

python

```
        if 0 <= nx < len(game_map) and 0 <= ny < len(game_map[0]):
            target = game_map[nx][ny]
            if target == '@':
                game_finished = True
            elif target in ['.', 'P']:
                game_map[old_x] = game_map[old_x][:old_y] + "." + game_map[old_x][old_y+1:]
                game_map[nx] = game_map[nx][:ny] + "G" + game_map[nx][ny+1:]
```

- Check if new position is within map bounds
- If ghost hits Pac-Man (`@`), game ends
- If target is empty space or pill, move ghost there
- String slicing replaces characters: `[:old_y] + "." + [old_y+1:]` replaces the character at `old_y` with "."

Pac-Man Movement Function

python

```
def move_pacman(game_map, key):
    pacman_x, pacman_y = -1, -1
    for x, row in enumerate(game_map):
        for y, cell in enumerate(row):
            if cell == '@':
                pacman_x, pacman_y = x, y
```

- Find Pac-Man's current position

```python
nx, ny = pacman_x, pacman_y
if key == 'a': ny -= 1    # left
elif key == 'd': ny += 1  # right
elif key == 'w': nx -= 1  # up
elif key == 's': nx += 1  # down
else: return game_map, False, False  # invalid key
```

- Calculate new position based on key input (WASD controls)
- Return unchanged state for invalid keys

```python
if not (0 <= nx < len(game_map) and 0 <= ny < len(game_map[0])): return game_map, False, False
if game_map[nx][ny] in ['|','-']: return game_map, False, False
if game_map[nx][ny] == 'G': return game_map, True, False
```

- Boundary checking: can't move outside map
- Wall checking: can't move through walls
- Ghost collision: if Pac-Man hits ghost, game ends (lose)

```python
game_map[pacman_x] = game_map[pacman_x][:pacman_y] + "." + game_map[pacman_x][pacman_y+1:]
game_map[nx] = game_map[nx][:ny] + "@" + game_map[nx][ny+1:]
```

- Move Pac-Man: replace old position with empty space, new position with '@'

```python
total_pills = sum(row.count('P') for row in game_map)
if total_pills == 0: return game_map, True, True
```

- Count remaining pills across all rows
- If no pills left, player wins

```python
return game_map, False, False
```

- Return: updated map, game_finished=False, win=False (continue playing)

Main Game Loop

```python
```

```
def play_game():
    game_map = create_map()
    game_finished = False
    win = False
```

- Initialize game state

```
python

    while not game_finished:
        print_map(game_map)
        key = input("Enter move (WASD): ").lower()
        game_map, game_finished, win = move_pacman(game_map, key)
        if game_finished: break
        game_map, ghost_finished = move_ghosts(game_map)
        if ghost_finished:
            game_finished = True
            win = False
```

- Game loop: display map → get input → move Pac-Man → move ghosts
- Game ends if Pac-Man wins (all pills collected) or loses (hit by ghost)

```
python

    final_color = COLOR_FINAL_WIN if win else COLOR_FINAL_LOSE
    print_map(game_map, final_color)
    print("You win! :)" if win else "You lost! :/")
```

- Display final screen in green (win) or red (lose)
- Print win/lose message

```
python

# Start the game
play_game()
```

- Execute the game

Game Flow Summary

1. **Initialization**: Create 6x10 map with walls, 3 ghosts, 3 pills, and Pac-Man
2. **Player Turn**: Display map, get WASD input, move Pac-Man
3. **Ghost Turn**: Each ghost moves randomly to adjacent empty space or pill
4. **Win Condition**: Collect all 3 pills
5. **Lose Condition**: Pac-Man touches a ghost
6. **End**: Display final screen with appropriate color and message