Snake Game Pt: 2

PART 1: Importing and Initialization

```
import pygame

import sys

import random


pygame.init()


WIDTH, HEIGHT = 800, 600

CELL_SIZE = 30

screen = pygame.display.set_mode((WIDTH, HEIGHT))

pygame.display.set_caption("Simple Snake Game")
```

Detailed Explanation for First-Time Coders

- `import pygame`: We're bringing in a special toolkit called Pygame, which makes it easier to build games. Without it, you'd have to build your own game engine — which is much harder.

- `import sys`: This toolkit helps us safely close or exit the game. Think of it like giving the computer a way to hit the "exit" button when needed.

- `import random`: This lets the computer pick random numbers — like rolling dice. We use this to place food or bombs randomly on the screen.

- `pygame.init()`: Before we can use any of the Pygame tools, we have to turn them on. This line activates everything in Pygame.
- `WIDTH, HEIGHT = 800, 600`: These two values define how big our game screen is: 800 pixels wide and 600 pixels tall.
- `CELL_SIZE = 30`: Each square in the game (the snake's body, food, or bombs) will be 30 pixels by 30 pixels. This makes sure everything fits nicely on a grid.
- `screen = pygame.display.set_mode(...)`: This creates the actual game window where everything will appear.
- `pygame.display.set_caption(...)`: This puts a title at the top of the window so it says "Simple Snake Game."

This part is like building the game board before you start playing.

PART 2: Colors and Object Setup

```
WHITE = (255, 255, 255)

GREEN = (0, 199, 100)

RED = (255, 0, 0)

BLACK = (0, 0, 0)
```

What's This?

- These are RGB colors. Computers show colors by mixing Red, Green, and Blue.
- `WHITE = (255, 255, 255)` means full red + full green + full blue → makes white.
- `GREEN = (0, 199, 100)` is a soft green color we'll use for the snake.

- `RED` and `BLACK` are used for text, bombs, and other parts.

These are like your colored markers. You're setting them up to use later in the drawing part.

Setting Up Game Objects

```
snake = [pygame.Rect(100, 100, CELL_SIZE, CELL_SIZE),
         pygame.Rect(100, 100, CELL_SIZE, CELL_SIZE),
         pygame.Rect(100, 100, CELL_SIZE, CELL_SIZE)]

direction = pygame.K_RIGHT
```

Snake Explained

- A snake is made of square segments. We're starting with 3, all in the same spot.
- `pygame.Rect(x, y, width, height)` creates a rectangle object in Pygame.
- `100, 100` is the top-left corner of the square.
- `CELL_SIZE, CELL_SIZE` makes it 30x30 pixels.

Later, when we move the snake, these segments will spread out and form the moving body.

Starting Direction

```
direction = pygame.K_RIGHT
```

- This tells the snake to start moving right across the screen.
- `pygame.K_RIGHT` is the value used in Pygame to represent the right arrow key.

Placing Food (Points)

```
points = [
    pygame.Rect(random.randint(0, WIDTH // CELL_SIZE - 1) * CELL_SIZE,
                random.randint(0, HEIGHT // CELL_SIZE - 1) *
CELL_SIZE,
                CELL_SIZE, CELL_SIZE),
    pygame.Rect(random.randint(0, WIDTH // CELL_SIZE - 1) * CELL_SIZE,
                random.randint(0, HEIGHT // CELL_SIZE - 1) *
CELL_SIZE,
                CELL_SIZE, CELL_SIZE)
]
```

- We're creating 2 food items and placing them in random spots.
- `random.randint(0, WIDTH // CELL_SIZE - 1)` gives us a random column or row.
- Multiplying by `CELL_SIZE` turns it into a pixel location so it aligns perfectly with the grid.

Food appears on the screen in a spot that fits perfectly with the snake's size.

Bombs

```
bombs = [pygame.Rect(random.randint(0, WIDTH // CELL_SIZE - 1) *
CELL_SIZE,
                     random.randint(0, HEIGHT // CELL_SIZE - 1) *
CELL_SIZE,
                     CELL_SIZE, CELL_SIZE)]
```

- We start the game with one bomb placed randomly on the screen.

Scoring

```
eaten_amount = 0

score = 0
```

- `eaten_amount` keeps track of how many food pieces the snake has eaten in total.
- `score` is your actual game score (shown to the player), increased by 1 each time you eat.

PART 3: Sounds and Images

```
eaten_sound = pygame.mixer.Sound("game2/coin.wav")

bomb_sound = pygame.mixer.Sound("game2/bomb.mp3")

die_sound = pygame.mixer.Sound("game2/die.mp3")


bomb_image = pygame.image.load("game2/bomb.png").convert_alpha()

bomb_image = pygame.transform.scale(bomb_image, (CELL_SIZE,
CELL_SIZE))


clock = pygame.time.Clock()

font = pygame.font.SysFont(None, 36)
```

Sounds

- These lines load sound files for eating food, hitting a bomb, and game over.
- `pygame.mixer.Sound(...)` brings the sound into the game.
- `.play()` (used later) plays the sound at the right moment.

Bomb Image

- `pygame.image.load(...)` loads a bomb picture file.
- `.convert_alpha()` keeps the image transparent where needed.
- `transform.scale(...)` shrinks the bomb image to the size of a cell.

Make sure the files are inside a folder called `game2` or the game will crash.

Clock and Font

- `clock` keeps the game running at a steady pace.
- `font` is used to draw text, like your score or “Game Over!”

PART 4: Movement Function

```
def move_snake():
    head = snake[0].copy()
    if direction == pygame.K_LEFT:
        head.x -= CELL_SIZE
    elif direction == pygame.K_RIGHT:
        head.x += CELL_SIZE
    elif direction == pygame.K_UP:
        head.y -= CELL_SIZE
    elif direction == pygame.K_DOWN:
        head.y += CELL_SIZE
    return head
```

Explanation

- `snake[0]` is the head of the snake.
- `.copy()` makes a duplicate so we can move it without breaking the original.
- Depending on the current direction, we change the x or y position.
- We move the head by exactly `CELL_SIZE` to stay on the grid.
- `return head` gives the new head position back to the game.

PART 5: Game Loop and Key Input

```
while True:
    screen.fill(WHITE)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()

    keys = pygame.key.get_pressed()
    for key in [pygame.K_LEFT, pygame.K_RIGHT, pygame.K_UP,
pygame.K_DOWN]:
        if keys[key]:
            direction = key
```

Game Loop

- `while True:` keeps repeating — this is the main loop of the game.
- `screen.fill(WHITE)` clears the screen each frame so we can draw fresh.

Input

- `pygame.event.get()` looks for things like the player clicking the X to close the window.
- `pygame.key.get_pressed()` checks which arrow keys are being held down.
- If a key is pressed, we update the snake's direction.

This is how you control the snake.

PART 6: Game Over

```
def game_over():
    die_sound.play()
    game_over_text = font.render("Game Over!", True, RED)
    screen.blit(game_over_text, (WIDTH // 2 - 80, HEIGHT // 2))
    pygame.display.update()
    pygame.time.wait(2000)
    pygame.quit()
    sys.exit()
```

Breakdown

- Plays the death sound.
- Displays "Game Over!" in red text at the center of the screen.
- Waits 2 seconds (2000 milliseconds).
- Shuts down the game and exits.

PART 7: Snake Movement and Wall Collisions

```
new_head = move_snake()

if (new_head.left < 0 or new_head.right > WIDTH or
    new_head.top < 0 or new_head.bottom > HEIGHT or
    new_head.collidelist(snake) != -1):
    game_over()

snake.insert(0, new_head)
```

Collision Detection

- `new_head` is where the snake is trying to go next.

- We check if:
    - It went off the screen
    - It hit its own body
- If yes, we call `game_over()`.
- Otherwise, we add the new head to the front of the snake to move it forward.

PART 8: Eating Points and Dropping Bombs

```
eaten = False
for p in points:
    if new_head.colliderect(p):
        p.x = random.randint(0, WIDTH // CELL_SIZE - 1) * CELL_SIZE
        p.y = random.randint(0, HEIGHT // CELL_SIZE - 1) * CELL_SIZE
        eaten_sound.play()
        score += 1
        eaten_amount += 1
        eaten = True
        if eaten_amount % 3 == 0:
            new_bomb = pygame.Rect(
                random.randint(0, WIDTH // CELL_SIZE - 1) * CELL_SIZE,
```

```
                random.randint(0, HEIGHT // CELL_SIZE - 1) *
CELL_SIZE,
                CELL_SIZE, CELL_SIZE
            )
            bombs.append(new_bomb)
```

Food Logic

- We loop through all the food items.
- If the snake touches one:
    - Move that food to a new location
    - Play the eating sound
    - Increase score and `eaten_amount`
- Every 3 food pieces eaten, we drop a new bomb in a random location.

The more you eat, the more dangerous the game becomes.