Snake Game

Part 1: Importing and Initialization

- Import necessary libraries

- Start Pygame

- Set screen size and title

```
import pygame
import sys
import random

# Initialize Pygame
pygame.init()

# Set up the screen
WIDTH, HEIGHT = 800, 600
CELL_SIZE = 30
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Simple Snake Game")
```

Notes

- pygame.init() must always be called before using Pygame features.

Part 2: Colors and Object Setup

- Define color constants

- Initialize snake, direction, points, and bombs

```
# Colors
WHITE = (255, 255, 255)
GREEN = (0, 199, 100)
RED = (255, 0, 0)
BLACK = (0, 0, 0)

# Snake and point setup
```

Snake Game

```
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

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) ]

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) ]

eaten_amount = 0
score = 0
```

Notes

- pygame.Rect(x, y, width, height) creates rectangles.
- Points and bombs are placed randomly.

Part 3: Sounds and Images

- Load sound effects and image assets
- Prepare clock and font

```
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)
```

Notes

- Sound and images must be in the correct folder for the game to run.

Part 4: Movement Function

- move_snake() determines new head position

```
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
```

Notes

- Always move by CELL_SIZE to stay on the grid.

Part 5: Game Loop and Input

- Handle quitting and key inputs

```
# Game loop
while True:
    screen.fill(WHITE)
```

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

# Handle input
    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
```

Notes

- .get_pressed() tracks held-down keys.

Part 6: Game Over

- Define a game over function to end the game

```
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()
```

Notes

- Shows message for 2 seconds and exits.

Part 7: Move and Collide

Snake Game

- Move the snake and check for wall or self collision

```
# Move snake
    new_head = move_snake()


# Check collision with walls or self
    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)
```

Notes

- collidelist() returns -1 if no collision.

Part 8: Eating Points and Bombs

- Detect if snake eats a point
- Add a bomb every 3 points

```
    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(...)
                bombs.append(new_bomb)
            break
    if not eaten:
        snake.pop()

    for b in bombs:
        if new_head.colliderect(b):
            bomb_sound.play()
            game_over()
```

Notes

- Points and bombs reposition when eaten.

Part 9: Drawing and Display

- Render all elements and update the screen

```
# Draw snake and point
    score_text = font.render(f"Score {score}",True,BLACK)
    screen.blit(score_text,(0,0))
    for segment in snake:
        pygame.draw.rect(screen, GREEN, segment)
    for p in points:
        pygame.draw.rect(screen,RED,p)
    for b in bombs:
        screen.blit(bomb_image,b)

    pygame.display.update()
    clock.tick(10)
```

Notes

- tick(10) caps the game at 10 frames per second.