Python Fighting Game Development

A Pygame-based 2D Fighting Game Implementation

Djunice, Frederik, Tobias and Assil

December 2024

Project Overview

- Development of a 2D fighting game using Pygame
- Core features:
  - Two-player combat system
  - Physics-based movement
  - Character collision detection
  - Health system implementation
  - Sprite-based animation system
  - Background rendering

Program Structure

- Two main Python files:
  - `mainGame.py`: Game loop and initialization
  - `fighter.py`: Fighter class implementation
- Key Components:
  - Game Window: 1000x600 pixels
  - Frame Rate Control: 60 FPS
  - Sprite Sheet Management
  - Background Image System
  - Health Bar System
  - Character Movement System

Game Initialization Code

```
# Create Game Window
SCREEN_WIDTH = 1000
SCEEN_HEIGHT = 600
screen = pygame.display.set_mode((SCREEN_WIDTH, SCEEN_HEIGHT))
pygame.display.set_caption("Brawler")

# Set Framerate
clock = pygame.time.Clock()
FPS = 60

# Define Fighter variables
SAMURAI1_SIZE = 128
SAMURAI1_SCALE = 2.5
SAMURAI1_OFFSET = [20, 60]
SAMURAI1_DATA = [SAMURAI1_SIZE, SAMURAI1_SCALE, SAMURAI1_OFFSET]

# Load Background Image
bg_image = pygame.image.load("korea.jpg").convert_alpha()
```

Game Loop Implementation

```
# Game Loop
run = True
while run:
    clock.tick(FPS)
    # Draw background
    draw_bg()

    # Show health bars
    draw_health_bar(fighter_1.health, 20, 20)
    draw_health_bar(fighter_2.health, 580, 20)

    # Move and update fighters
    fighter_1.move(SCREEN_WIDTH, SCEEN_HEIGHT, screen, fighter_2)
    fighter_2.move(SCREEN_WIDTH, SCEEN_HEIGHT, screen, fighter_1)

    fighter_1.update()
    fighter_2.update()

    # Event handler
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
```

Fighter Class Core Implementation

```
1  class Fighter():
2      def __init__(self, x, y, flip, data, sprite_sheet,
3                   animation_steps, player_num):
4          self.player_num = player_num
5          self.size = data[0]
6          self.image_scale = data[1]
7          self.offest = data[2]
8          self.rect = pygame.Rect((x, y, 80, 180))
9          self.vel_y = 0
10         self.running = False
11         self.jump = False
12         self.attking = False
13         self.attack_type = 0
14         self.health = 100
15         self.alive = True
```

Movement System Implementation

```
def move(self, screen_width, screen_height, surface, target):
    SPEED = 10
    GRAVITY = 2
    dx = 0
    dy = 0

    # Get keypresses
    key = pygame.key.get_pressed()

    if not self.attking and self.alive:
        # Controls for Player 1
        if self.player_num == 1:
            if key[pygame.K_a]:
                dx = -SPEED
            if key[pygame.K_d]:
                dx = SPEED
            if key[pygame.K_w] and not self.jump:
                self.vel_y = -30
                self.jump = True

    # Apply Gravity
    self.vel_y += GRAVITY
    dy += self.vel_y
```

Combat System Implementation

```
1  def attack(self, surface, target):
2      if self.attack_cooldown == 0:
3          self.attking = True
4          attacking_rect = pygame.Rect(
5              self.rect.centerx - (2 * self.rect.width * self.flip),
6              self.rect.y,
7              2 * self.rect.width,
8              self.rect.height
9          )
10         if attacking_rect.colliderect(target.rect):
11             target.health -= 10
12             target.hit = True
13
14         pygame.draw.rect(surface, (0, 255, 0), attacking_rect)
```

Animation System

```
def update(self):
    # Update animation
    if self.health <= 0:
        self.health = 0
        self.alive = False
        self.update_action(6)  # Death animation
    elif self.hit:
        self.update_action(5)  # Hit animation
    elif self.attking:
        if self.attack_type == 1:
            self.update_action(3)  # Attack1 animation
        elif self.attack_type == 2:
            self.update_action(4)  # Attack2 animation
    elif self.jump:
        self.update_action(2)  # Jump animation
    elif self.running:
        self.update_action(1)  # Run animation
    else:
        self.update_action(0)  # Idle animation
```

Health System Implementation

```
def draw_health_bar(health, x, y):
    ratio = health / 100
    pygame.draw.rect(screen, WHITE, (x - 2, y - 2, 404, 34))
    pygame.draw.rect(screen, RED, (x, y, 400, 30))
    pygame.draw.rect(screen, YELLOW, (x, y, 400 * ratio, 30))

# Usage in game loop:
draw_health_bar(fighter_1.health, 20, 20)
draw_health_bar(fighter_2.health, 580, 20)
```

Sprite Animation Loading

```python
def load_images(self, sprite_sheet, animation_steps):
    animation_list = []
    for y, animation in enumerate(animation_steps):
        temp_img_list = []
        for x in range(animation):
            temp_img = sprite_sheet.subsurface(
                x * self.size,
                y * self.size,
                self.size,
                self.size
            )
            temp_img_list.append(pygame.transform.scale(
                temp_img,
                (self.size * self.image_scale,
                self.size * self.image_scale)
            ))
        animation_list.append(temp_img_list)
    return animation_list
```

Drawing System

```
1  def draw(self, surface):
2      img = pygame.transform.flip(self.image, self.flip, False)
3
4      if self.flip:
5          draw_x = self.rect.x - self.flip_offset[0]
6      else:
7          draw_x = self.rect.x - self.offest[0]
8
9      draw_y = self.rect.y - self.offest[1]
10
11     # Draw character
12     surface.blit(img, (draw_x, draw_y))
```

Live Demo

Game Demonstration Video

Click here to watch the demo video

Future Development Opportunities

Potential enhancements:

- Enhanced combat mechanics
- Additional character animations
- Expanded character roster
- Special moves system
- Sound effects and music
- Multiplayer networking
- Tournament mode
- Character customization

Thank You

Questions?