Atividade 5: Fundamentos de Programação Gráfica com Pygame

Discente: Nicolas Expedito Lana Mendes
Matrícula: 22.1.4028
Docente: Rafael Alves Bonfim de Queiroz
Universidade Federal de Ouro Preto
Departamento de Computação
BCC327 - Computação Gráfica

Fevereiro de 2025

1 Introdução

Este relatório apresenta uma aplicação gráfica desenvolvida em Python utilizando a biblioteca **Pygame**. A atividade é dividida em dois exercícios que abordam fundamentos de programação gráfica:

1. **Exercício 1:** Criação de um triângulo 2D com gradiente de cores. Cada vértice do triângulo possui uma cor distinta (vermelho, verde e azul), e as cores dos fragmentos são interpoladas para criar um gradiente suave.

2. **Exercício 2:** Implementação de um quadrado 2D que pode ser movimentado na tela através das teclas de seta. O quadrado permanece visível enquanto se desloca.

O usuário pode alternar entre os exercícios pressionando as teclas `1` e `2`.

2 API Utilizada

A aplicação foi implementada com a biblioteca **Pygame**, que oferece suporte à renderização 2D, gerenciamento de eventos e manipulação de superfícies. Estes recursos possibilitam a criação e a interação com objetos gráficos em tempo real.

3 Código da Implementação

A seguir, são apresentados os trechos de código que compõem a aplicação.

3.1 Exercício 1: Triângulo com Gradiente de Cores

O triângulo é desenhado utilizando uma função que calcula as coordenadas baricêntricas para interpolar as cores dos seus vértices. Cada vértice possui uma cor distinta:

```
def draw_gradient_triangle(surface, vertices, colors):
    (x0,y0), (x1,y1), (x2,y2) = vertices
    min_x = int(min(x0, x1, x2))
    max_x = int(max(x0, x1, x2))
    min_y = int(min(y0, y1, y2))
    max_y = int(max(y0, y1, y2))
    denom = ((y1 - y2)*(x0 - x2) + (x2 - x1)*(y0 - y2))
    if denom == 0:
        return
    for y in range(min_y, max_y+1):
        for x in range(min_x, max_x+1):
            alpha = ((y1 - y2)*(x - x2) + (x2 - x1)*(y - y2)) /
               denom
            beta = ((y2 - y0)*(x - x2) + (x0 - x2)*(y - y2)) /
               denom
            gamma = 1 - alpha - beta
            if alpha >= 0 and beta >= 0 and gamma >= 0:
                r = int(alpha*colors[0][0] + beta*colors[1][0] +
                   gamma*colors[2][0])
                g = int(alpha*colors[0][1] + beta*colors[1][1] +
                   gamma*colors[2][1])
                b = int(alpha*colors[0][2] + beta*colors[1][2] +
                   gamma*colors[2][2])
                surface.set_at((x,y), (r, g, b))
```

3.2 Exercício 2: Quadrado Móvel

O quadrado é desenhado no centro da tela e sua posição é atualizada conforme o usuário pressiona as teclas de seta:

```
# Vari veis do quadrado
square_size = 50
square_x = WIDTH // 2 - square_size // 2
square_y = HEIGHT // 2 - square_size // 2
square_speed = 5

# No loop principal:
if mode == 2:
    if keys[pygame.K_LEFT]:
        square_x -= square_speed
    if keys[pygame.K_RIGHT]:
        square_x += square_speed
    if keys[pygame.K_UP]:
        square_y -= square_speed
    if keys[pygame.K_DOWN]:
        square_y += square_speed
    square_x = max(0, min(WIDTH - square_size, square_x))
    square_y = max(0, min(HEIGHT - square_size, square_y))
    pygame.draw.rect(screen, (0,255,255), (square_x, square_y,
       square_size, square_size))
```

3.3 Loop Principal e Alternância entre Exercícios

O loop principal processa os eventos e permite alternar entre os exercícios:

```
mode = 1  # 1: Tri ngulo com gradiente, 2: Quadrado m vel

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_1:
                mode = 1
            elif event.key == pygame.K_2:
                mode = 2
    keys = pygame.key.get_pressed()
    screen.fill((0,0,0))
    if mode == 1:
        vertices = [(200,100), (100,300), (300,300)]
        colors = [(255,0,0), (0,255,0), (0,0,255)]
        draw_gradient_triangle(screen, vertices, colors)
        text = font.render("Exerc cio␣1:␣Tri ngulo␣com␣
           gradiente", True, (255,255,255))
        screen.blit(text, (20,20))
    elif mode == 2:
        if keys[pygame.K_LEFT]:
            square_x -= square_speed
        if keys[pygame.K_RIGHT]:
            square_x += square_speed
        if keys[pygame.K_UP]:
            square_y -= square_speed
        if keys[pygame.K_DOWN]:
            square_y += square_speed
        square_x = max(0, min(WIDTH - square_size, square_x))
        square_y = max(0, min(HEIGHT - square_size, square_y))
        pygame.draw.rect(screen, (0,255,255), (square_x, square_y
           , square_size, square_size))
        text = font.render("Exerc cio␣2:␣Quadrado␣m vel␣(use␣as
           ␣setas)", True, (255,255,255))
        screen.blit(text, (20,20))

    pygame.display.flip()
    clock.tick(60)
pygame.quit()
```

4 Instruções de Execução

Para executar a aplicação:

1. Instale o Python e a biblioteca **Pygame** (por exemplo, via `pip install pygame`).

2. Salve o código em um arquivo denominado `atividade5.py`.

3. Execute o script com o comando `python atividade5.py`.

4. Pressione a tecla `1` para visualizar o triângulo com gradiente (Exercício 1) e a tecla `2` para ativar o quadrado móvel (Exercício 2).

5. No Exercício 2, utilize as setas do teclado para mover o quadrado pela tela.

5 Funcionalidade e Resultados

A aplicação demonstrou os seguintes resultados:

- **Exercício 1:** Um triângulo 2D foi renderizado com um gradiente de cores, onde cada vértice possui uma cor distinta (vermelho, verde e azul). A interpolação das cores gerou um efeito de transição suave entre os vértices.

- **Exercício 2:** Um quadrado 2D, inicialmente centralizado, pode ser movimentado na tela utilizando as teclas de seta. O objeto permanece visível enquanto se desloca.

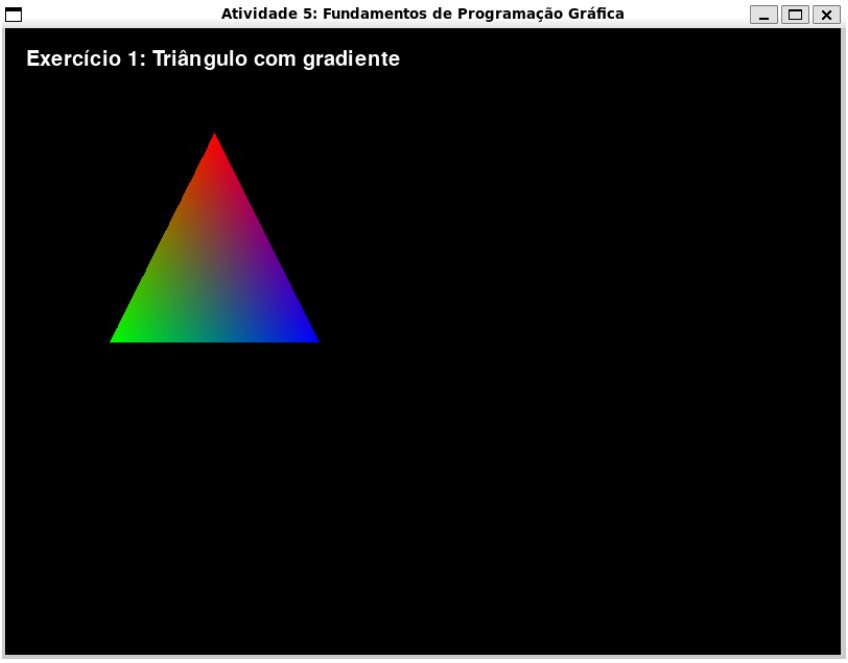

Figure 1: Exercício 1: Triângulo com gradiente de cores.

Figure 2: Exercício 2: Quadrado móvel, demonstrando a movimentação com as teclas de seta. Parte 1

Figure 3: Exercício 2: Quadrado móvel, demonstrando a movimentação com as teclas de seta. Parte 2

6 Conclusão

A atividade permitiu a aplicação dos fundamentos da programação gráfica através da criação de um triângulo com gradiente de cores e da implementação de um objeto interativo (quadrado) controlado pelo teclado. O Exercício 1 demonstrou a interpolação de

cores usando coordenadas baricêntricas, enquanto o Exercício 2 evidenciou a manipulação de eventos para movimentação de objetos na tela. Assim, a implementação consolidou conceitos essenciais da computação gráfica e a utilização prática dos recursos oferecidos pela biblioteca **Pygame**.