Atividade 4: Teoria das Cores e Modelos de Cor 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 descreve a implementação de uma aplicação gráfica interativa desenvolvida em Python utilizando a biblioteca **Pygame**. A atividade tem como foco o estudo da teoria das cores e dos modelos de cor, sendo composta por três exercícios:

1. **Exercício 1:** Implementação de um gradiente RGB, onde o componente R varia de 0 a 255 e os componentes G e B permanecem constantes.
2. **Exercício 2:** Conversão de valores do modelo **RGB** para **HSV**. Diferentemente de abordagens fixas, este exercício permite que o usuário insira os valores de R, G e B via teclado, proporcionando uma interação dinâmica com a aplicação.
3. **Exercício 3:** Conversão de uma cor fixa (R=255, G=128, B=64) do modelo **RGB** para o modelo subtrativo **CMYK**.

O aplicativo possibilita a alternância entre os exercícios através das teclas 1, 2 e 3, e exibe os resultados das conversões em tempo real.

2 API Utilizada

A aplicação foi desenvolvida com a biblioteca **Pygame**, que oferece ferramentas para renderização gráfica, gerenciamento de eventos e manipulação de entradas via teclado. Para as conversões de cor, a biblioteca **colorsys** do Python foi empregada, facilitando a conversão entre os modelos RGB e HSV.

3 Trechos de Código

A seguir, são destacados os trechos de código referentes a cada exercício.

3.1 Exercício 1: Gradiente RGB

```
def draw_gradient():
    # Gradiente onde o componente R varia de 0 a 255, com G e B
       fixos em 128
    for x in range(WIDTH):
        r = int((x / WIDTH) * 255)
        color = (r, 128, 128)
        pygame.draw.line(screen, color, (x, 0), (x, HEIGHT))
```

3.2 Exercício 2: Conversão de RGB para HSV com Entrada de Teclado

Nesta parte, o usuário insere os valores de R, G e B através do teclado. Cada campo de entrada é atualizado conforme o usuário digita, e a conversão para HSV é realizada quando os três valores estiverem preenchidos.

```
def draw_rgb_to_hsv():
    global input_R, input_G, input_B
    % Exibe os campos de entrada para os valores de R, G e B
    label_R = font.render("Digite␣o␣valor␣de␣R␣(0-255):␣" +
       input_R, True, (255, 255, 255))
    label_G = font.render("Digite␣o␣valor␣de␣G␣(0-255):␣" +
       input_G, True, (255, 255, 255))
    label_B = font.render("Digite␣o␣valor␣de␣B␣(0-255):␣" +
       input_B, True, (255, 255, 255))
    screen.blit(label_R, (50, 100))
    screen.blit(label_G, (50, 150))
    screen.blit(label_B, (50, 200))

    % Realiza a convers o para HSV quando os tr s campos est o
        preenchidos
    if input_R != "" and input_G != "" and input_B != "":
        try:
            R = int(input_R)
            G = int(input_G)
            B = int(input_B)
            R = max(0, min(R, 255))
            G = max(0, min(G, 255))
            B = max(0, min(B, 255))
            r, g, b = R/255.0, G/255.0, B/255.0
            h, s, v = colorsys.rgb_to_hsv(r, g, b)
            h_deg = h * 360
            s_perc = s * 100
            v_perc = v * 100
            result_rgb = font.render(f"RGB:␣({R},␣{G},␣{B})",
               True, (255,255,255))
            result_hsv = font.render(f"HSV:␣({h_deg:.1f}  ,␣{
               s_perc:.1f}%,␣{v_perc:.1f}%)", True, (255,255,255)
               )
            screen.blit(result_rgb, (50, 300))
```

```
            screen.blit(result_hsv, (50, 350))
        except:
            error_text = font.render("Erro na convers o dos
               valores.", True, (255, 0, 0))
            screen.blit(error_text, (50, 300))
```

3.3 Exercício 3: Conversão de RGB para CMYK

```
def draw_rgb_to_cmyk():
    % Convers o para CMYK da cor fixa (255, 128, 64)
    R, G, B = 255, 128, 64
    r, g, b = R/255.0, G/255.0, B/255.0
    K = 1 - max(r, g, b)
    if K == 1:
        C = M = Y = 0
    else:
        C = (1 - r - K) / (1 - K)
        M = (1 - g - K) / (1 - K)
        Y = (1 - b - K) / (1 - K)
    C_perc = C * 100
    M_perc = M * 100
    Y_perc = Y * 100
    K_perc = K * 100
    text_rgb = font.render(f"RGB: ({R}, {G}, {B})", True,
       (255,255,255))
    text_cmyk = font.render(f"CMYK: ({C_perc:.1f}%, {M_perc:.1f
       }%, {Y_perc:.1f}%, {K_perc:.1f}%)", True, (255,255,255))
    screen.blit(text_rgb, (50, HEIGHT//2 - 40))
    screen.blit(text_cmyk, (50, HEIGHT//2))
```

3.4 Loop Principal e Controle de Modos

No loop principal, a aplicação monitora os eventos de teclado para alternar entre os exercícios e, no caso do Exercício 2, processa a entrada numérica do usuário.

```
mode = 1  % 1: Gradiente, 2: RGB para HSV com entrada, 3: RGB
   para CMYK

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
                active_field = 0
                input_R = ""
```

```
                input_G = ""
                input_B = ""
            elif event.key == pygame.K_3:
                mode = 3
            % Processa a entrada para o Exerc cio 2
            if mode == 2:
                if event.key == pygame.K_BACKSPACE:
                    if active_field == 0 and len(input_R) > 0:
                        input_R = input_R[:-1]
                    elif active_field == 1 and len(input_G) > 0:
                        input_G = input_G[:-1]
                    elif active_field == 2 and len(input_B) > 0:
                        input_B = input_B[:-1]
                elif event.key == pygame.K_RETURN:
                    active_field = (active_field + 1) % 3
                elif event.unicode.isdigit():
                    if active_field == 0:
                        input_R += event.unicode
                    elif active_field == 1:
                        input_G += event.unicode
                    elif active_field == 2:
                        input_B += event.unicode

    screen.fill((0, 0, 0))
    if mode == 1:
        draw_gradient()
        text_mode = font.render("Exerc cio␣1:␣Gradiente␣RGB␣(R␣
           varia,␣G␣e␣B␣constantes)", True, (255,255,255))
        screen.blit(text_mode, (20, 20))
    elif mode == 2:
        draw_rgb_to_hsv()
        text_mode = font.render("Exerc cio␣2:␣Convers o␣RGB␣
           para␣HSV␣(Digite␣os␣valores)", True, (255,255,255))
        screen.blit(text_mode, (20, 20))
    elif mode == 3:
        draw_rgb_to_cmyk()
        text_mode = font.render("Exerc cio␣3:␣Convers o␣RGB␣
           para␣CMYK", True, (255,255,255))
        screen.blit(text_mode, (20, 20))

    instructions = font.render("Pressione␣1,␣2␣ou␣3␣para␣alternar
       ␣entre␣os␣exerc cios", True, (255,255,255))
    screen.blit(instructions, (20, HEIGHT - 40))

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

pygame.quit()
```

4 Instruções de Execução

1. Instale o Python e a biblioteca **Pygame** (por exemplo, `pip install pygame`).
2. Salve o código em um arquivo chamado `atividade4.py`.
3. Execute o script com o comando `python atividade4.py`.
4. Utilize as teclas 1, 2 e 3 para alternar entre os exercícios.
5. No Exercício 2, insira os valores de R, G e B via teclado. Use a tecla `Enter` para avançar para o próximo campo e `Backspace` para corrigir.

5 Funcionalidade e Resultados

A aplicação apresenta os seguintes resultados:

- **Exercício 1:** Um gradiente RGB é exibido, onde o valor de R varia linearmente de 0 a 255, enquanto G e B permanecem constantes em 128.
- **Exercício 2:** O usuário pode inserir valores para R, G e B por meio do teclado. Após preencher os três campos, a aplicação realiza a conversão para o modelo HSV e exibe os resultados na tela.
- **Exercício 3:** A aplicação converte a cor fixa (255, 128, 64) do modelo RGB para o modelo subtrativo CMYK e apresenta os valores convertidos em porcentagens.

Figure 1: Exercício 1: Gradiente RGB exibido na tela.

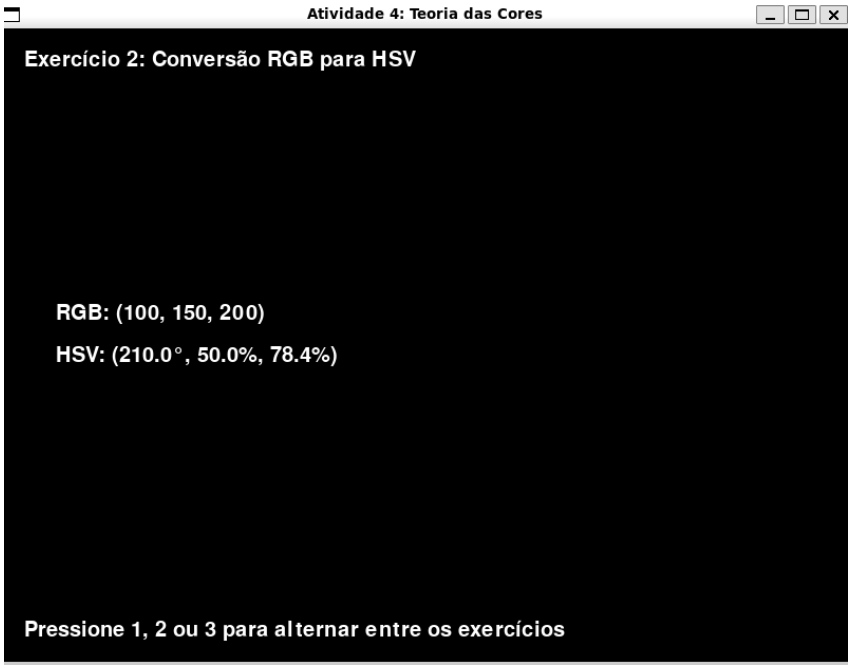

Figure 2: Exercício 2: Tela de entrada para valores de RGB, permitindo a conversão para HSV.

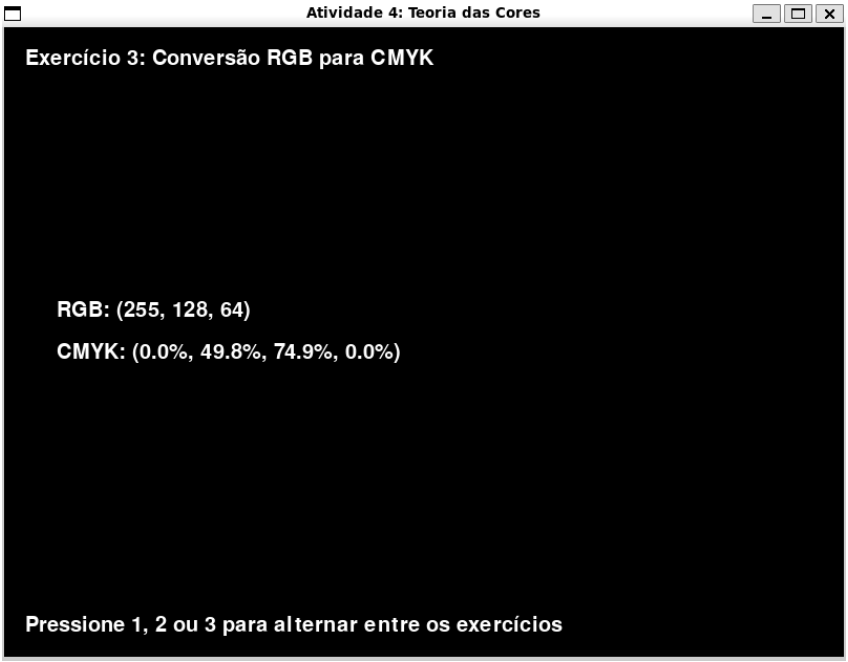

Figure 3: Exercício 3: Conversão da cor RGB para o modelo CMYK.

6 Conclusão

A atividade permitiu explorar os conceitos fundamentais da teoria das cores e dos modelos de cor, através da implementação de três exercícios interativos. Destaca-se:

- A criação de um gradiente RGB dinâmico (Exercício 1), que ilustra a variação contínua de um componente de cor.

- A implementação de um mecanismo de entrada via teclado para converter valores RGB para HSV (Exercício 2), promovendo a interatividade e permitindo ao usuário experimentar diferentes combinações de cores.
- A conversão de uma cor fixa do modelo RGB para CMYK (Exercício 3), demonstrando a aplicação dos modelos subtrativos de cor.

Esta implementação contribuiu para a compreensão prática dos modelos de cor e evidenciou a utilidade de ferramentas interativas na visualização de conceitos teóricos.