Universidade Federal de Ouro Preto - UFOP
Departamento de Ciência da Computação - DECOM

Relatório atividade 4

BCC327 - COMPUTACAO GRAFICA

Kayo Xavier Nascimento Cavalcante Leite - 21.2.4095

Professor: Rafael Alves Bonfim

Ouro Preto
30 de março de 2025

Sumário

Lista de Figuras

Lista de Códigos Fonte

1 Código e enuciado.

Na primeira atividade, foi requerido: Implementar um gradiente RGB onde R varia de 0 a 255 enquanto G e B permanecem constantes. Escreva um programa que receba um valor RGB como entrada e converta para HSV. Aplique uma cor R=255, G=128, B=64 em um modelo subtrativo e converta para CMYK.

Para tal, decidi utilizar a biblioteca pygame para gerar uma aplicação que satisfaça a demanda do gradiente. Como os outros exercícios dependem somente dos cálculos de conversão matemático, a implementação não necessitou de bibliotecas, apenas da linguagem. Foi escolhido fazer a representação visual da janela com gradientes.

O código da atividade se encontra anexado junto ao pdf com o relatório e, para executá-lo,o, necessário instalar o python e a biblioteca Pygame, para tal pode-se utilizar o seguinte comando :

Instalando bibliotecas

no linux:
sudo pip3 install pygame
para o windows:
pip install pygame

.

Após a instalação das bibliotecas, para rodar os códigos basta utilizar o terminal para entrar no diretório da atividade e usar o seguinte comando:

Rodando o exercício

python gradiente.py
python RGBtoCMYK,py
python RGBtHSV.py

O código se encontra todo comentado e formatado. A seguir a visualização do código completo da implementação do gradiente:

```
import pygame

pygame.init()

# Configura  es da tela: largura 256 (para representar de 0 a 255) e altura
    100
WIDTH, HEIGHT = 256, 256
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Gradiente RGB: R varia de 0 a 255")

# Valores constantes para G e B
G = 128
B = 128

running = True
clock = pygame.time.Clock()

print("Iniciando gradiente RGB...")

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Para cada coluna, desenha uma linha com a cor correspondente
    for x in range(WIDTH):
        R = x  # R varia de 0 a 255 conforme a posi   o x
```

```
        color = (R, G, B)
        pygame.draw.line(screen, color, (x, 0), (x, HEIGHT))

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

pygame.quit()
print("Programa de gradiente RGB finalizado.")
```

Código 1: Implementação do gradiente.

agora a implementação dos conversores:

```
def rgb_to_hsv():
    try:
        r = int(input("Digite o valor de R (0-255): "))
        g = int(input("Digite o valor de G (0-255): "))
        b = int(input("Digite o valor de B (0-255): "))
    except ValueError:
        print("Entrada inv lida! Por favor, insira n meros inteiros.")
        return None

    # Normaliza os valores para a faixa [0, 1]
    r_norm = r / 255.0
    g_norm = g / 255.0
    b_norm = b / 255.0
    print(f"Valores normalizados: r={r_norm:.2f}, g={g_norm:.2f}, b={b_norm:.2
        f}")

    # Calcula o m ximo, m nimo e a diferen a (delta)
    cmax = max(r_norm, g_norm, b_norm)
    cmin = min(r_norm, g_norm, b_norm)
    delta = cmax - cmin
    print(f"Cmax={cmax:.2f}, Cmin={cmin:.2f}, Delta={delta:.2f}")

    # Calcula o Hue (H)
    if delta == 0:
        h = 0
        print("Delta    0, definindo H = 0  ")
    elif cmax == r_norm:
        h = 60 * (((g_norm - b_norm) / delta) % 6)
        print("R    o valor m ximo, calculando H para o caso R")
    elif cmax == g_norm:
        h = 60 * (((b_norm - r_norm) / delta) + 2)
        print("G    o valor m ximo, calculando H para o caso G")
    elif cmax == b_norm:
        h = 60 * (((r_norm - g_norm) / delta) + 4)
        print("B    o valor m ximo, calculando H para o caso B")

    # Calcula a Satura  o (S)
    if cmax == 0:
        s = 0
        print("Cmax    0, definindo S = 0")
    else:
        s = delta / cmax
        print(f"Calculado S = {s:.2f}")

    # O Value (V)    o valor m ximo
    v = cmax
    print(f"Calculado V = {v:.2f}")

    print(f"\nRGB convertido para HSV:")
    print(f"H = {h:.2f}  ")
```

```
    print(f"S = {s*100:.2f}%")
    print(f"V = {v*100:.2f}%")
    return h, s, v

def main():
    rgb_to_hsv()


if __name__ == '__main__':
    main()
```

Código 2: Implementação do RGBtoHSV.py

```
def rgb_to_cmyk():
    try:
        r = int(input("Digite o valor de R (0-255): "))
        g = int(input("Digite o valor de G (0-255): "))
        b = int(input("Digite o valor de B (0-255): "))
    except ValueError:
        print("Por favor, insira n meros inteiros.")
        return
    print(f"Convertendo a cor RGB({r}, {g}, {b}) para o modelo CMYK...")
    # Normaliza os valores para a faixa [0, 1]
    r_norm = r / 255.0
    g_norm = g / 255.0
    b_norm = b / 255.0

    # Calcula o valor de K (preto)
    k = 1 - max(r_norm, g_norm, b_norm)
    if k == 1:
        c = m = y = 0
    else:
        c = (1 - r_norm - k) / (1 - k)
        m = (1 - g_norm - k) / (1 - k)
        y = (1 - b_norm - k) / (1 - k)
    print(f"CMYK: C={c:.2f}, M={m:.2f}, Y={y:.2f}, K={k:.2f}")
    return

def main():
    rgb_to_cmyk()


if __name__ == '__main__':
    main()
```

Código 3: Implementação do RGBtoCMYK.py

A seguir, a visualização da tela da aplicação com o gradiente

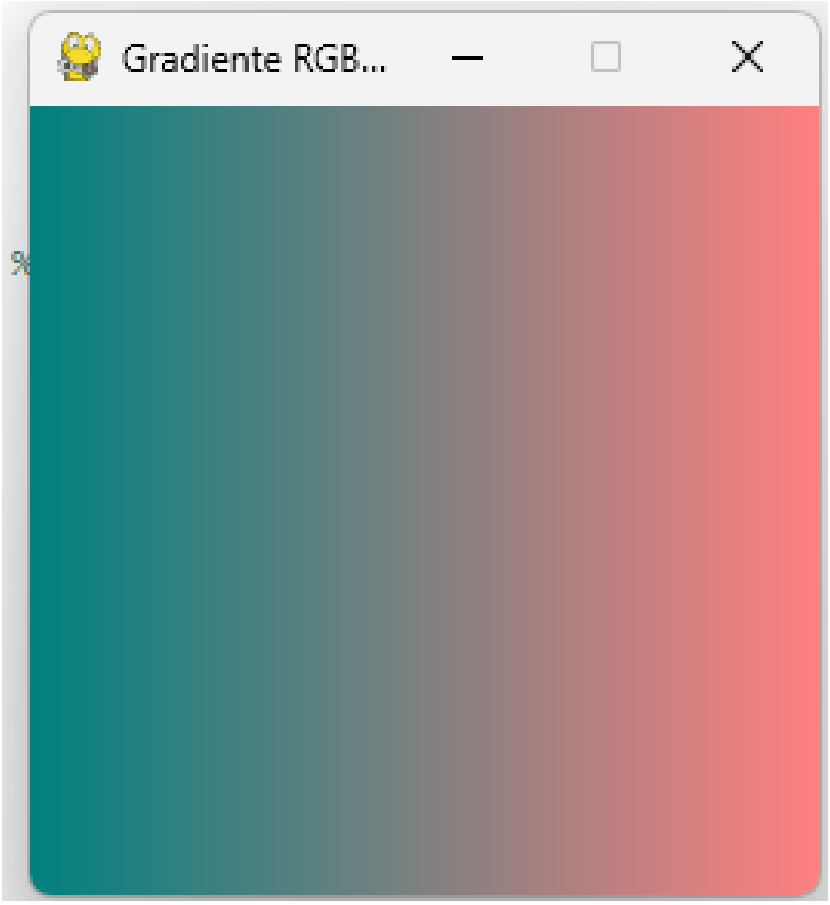

Figura 1: Visualização do gradiente.