

Universidade Federal de Ouro Preto - UFOP
Instituto de Ciências Exatas e Biológicas - ICEB
Departamento de Computação - DECOM
Ciência da Computação

Atividade 4 - Teoria das cores

BCC327 - Computação Gráfica

Matheus Peixoto Ribeiro Vieira - 22.1.4104
Professor: Rafael Alves Bonfim de Queiroz

Ouro Preto
5 de dezembro de 2024

Sumário

1	Introdução	1
1.1	Especificações da máquina	1
2	Exercício 1	1
3	Exercício 2	2
4	Exercício 3	2
5	Conclusão	2

Lista de Figuras

1	Tela Preta	1
2	Gradiente	2
3	RGB para HSV	3
4	RGB para CMYK	4

1 Introdução

Para este trabalho, é solicitado a criação de três programas diferentes que manipulam as cores.

A fim de realizar as atividades propostas, para a parte de exibição de imagens, a biblioteca matplotlib, do python, foi utilizada.

1.1 Especificações da máquina

A máquina onde o desenvolvimento foi realizado possui a seguinte configuração:

- Processador: Intel Core i5-9300H
- Memória RAM: 16Gb.
- Sistema Operacional: WSL 2.0 com Ubuntu 22.04.5 LTS

2 Exercício 1

Para a primeira atividade é solicitado a implementação de um gradiente RGB com os valores de G e B fixos e uma variação no R.

Assim, foi criado um vetor de numpy com valores zero e proporções 256 x 256 x 3 [2], ou seja, 256 pixels de altura e largura e três canais de cores (R, G e B), como pode ser visto na figura 1.

```
1 img = np.zeros((256, 256, 3))
2 plt.imshow(img)
3 plt.axis('off')
4 plt.show

✓ 1.1s

<function matplotlib.pyplot.show(close=None, block=None)>
```

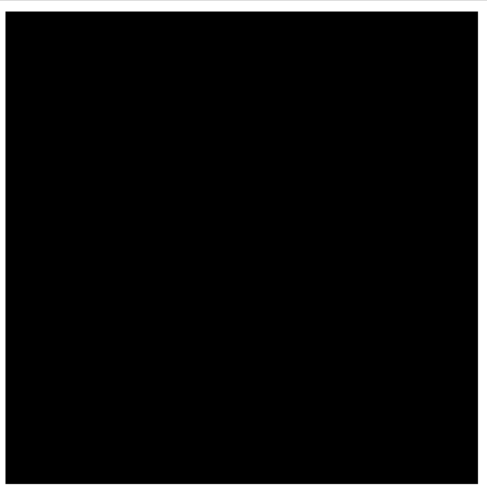


Figura 1: Tela Preta

Em seguida, para realizar o gradiente, dois espaços lineares foram gerados com o numpy [1], um com valores variando de 0 a 1 e outro de 1 a 0, ambos com 128 itens e sendo concatenados ao final. Gerando um gradiente onde a imagem começa com a cor preta, vai para o vermelho e retorna para o preto, como pode ser visto na figura 2.

Vale ressaltar que os valores no intervalo [0 a 255] são equivalentes aos valores [0 a 1], todavia normalizados.

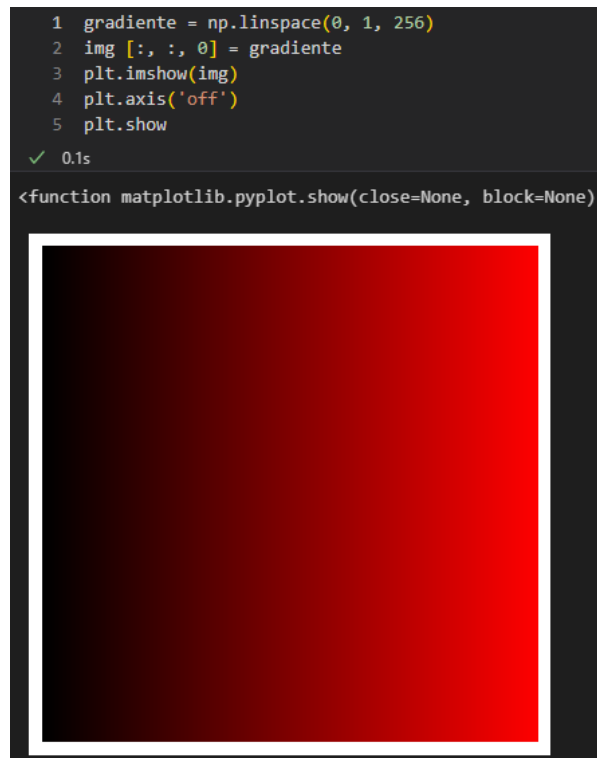


Figura 2: Gradiente

3 Exercício 2

Para esta atividade, foi criado um código que converte uma tripla de valores correspondente ao RGB em valores HSV, sendo inicialmente verificados se cada um dos valores estão nos limites do intervalo $[0, 255]$.

Os cálculos realizados são feitos seguindo as fórmulas apresentadas em sala de aula pelo professor, sendo que, no final, os valores de S e V são multiplicados por 100, sendo que todos do HSV são arredondados para representarem inteiros, como pode ser visto na figura 3, onde convertemos o valor $\text{RGB} = (55, 75, 250)$ para o HSV, gerando os valores $\text{HSV} = (234, 78, 98)$.

4 Exercício 3

No último exercício, é solicitado a conversão dos valores $\text{RGB} = (255, 128, 64)$ para o modelo CMYK. Dessa forma, foi criada uma função para realizar tal transformação.

Para evitar divisões por zero no cálculo de K em um situação onde os valores de R, G e B são zero, é retornado, de imediato, a quadrupla $(0, 0, 0, 1)$. caso contrário, dividimos os valores RGB por 255 para normalizá-los e os cálculos de conversão são feitos seguindo o que foi mostrado em sala de aula, como pode ser observado na figura 4.

5 Conclusão

Com o final do trabalho, os conceitos de transformações de cores entre diferentes modelos foram devidamente explorados e fixados.

```

1 def rgb_to_hsv(RGB):
2     r, g, b = RGB
3
4     if not (0 < r < 255 and 0 < g < 255 and 0 < b < 255):
5         raise ValueError("Canal fora dos limites [0, 255]")
6
7     maximo = max(r, g, b)
8     minimo = min(r, g, b)
9     delta = maximo - minimo
10
11     H = 0
12     if delta != 0:
13         if maximo == r:
14             H = 60 * (g-b)/delta
15         elif maximo == g:
16             H = 60 * (2 + (b-r)/delta)
17         else:
18             H = 60 * (4 + (r-g)/delta)
19
20     S = 0
21     if maximo != 0:
22         S = delta / maximo
23
24     V = maximo / 255
25
26     H = round(H)
27     S = round(S * 100)
28     V = round(V * 100)
29
30     return H, S, V
31
32 RGB = (55, 75, 250)
33 print(rgb_to_hsv(RGB))
34
✓ 0.0s
(234, 78, 98)

```

Figura 3: RGB para HSV

```

1 def rgb_to_cmyk( RGB ):
2     r, g, b = RGB
3
4     if not (0 <= r <= 255 and 0 <= g <= 255 and 0 <= b <= 255):
5         raise ValueError("Canal fora dos limites [0, 255]")
6
7     if r == 0 and g == 0 and b == 0: return 0, 0, 0, 1
8
9     r, g, b = r / 255., g / 255., b / 255.
10
11     k = 1 - max(r, g, b)
12     c = (1 - r - k) / (1 - k)
13     m = (1 - g - k) / (1 - k)
14     y = (1 - b - k) / (1 - k)
15
16     k = round(k * 100)
17     c = round(c * 100)
18     m = round(m * 100)
19     y = round(y * 100)
20
21     return c, m, y, k
22
23 RGB = (255, 128, 64)
24 print(rgb_to_cmyk(RGB))

```

✓ 0.0s

(0, 50, 75, 0)

Figura 4: RGB para CMYK

Referências

- [1] NumPy. `numpy.linspace`. <https://numpy.org/doc/2.1/reference/generated/numpy.linspace.html>. [Online; acessado em 04-Dezembro-2024].
- [2] NumPy. `numpy.zeros`. <https://numpy.org/doc/2.1/reference/generated/numpy.zeros.html>. [Online; acessado em 04-Dezembro-2024].