Nome: Felipe Braz Marques
Disciplina: BCC327 - Computação Gráfica
Professor: Rafael Alves Bonfim de Queiroz

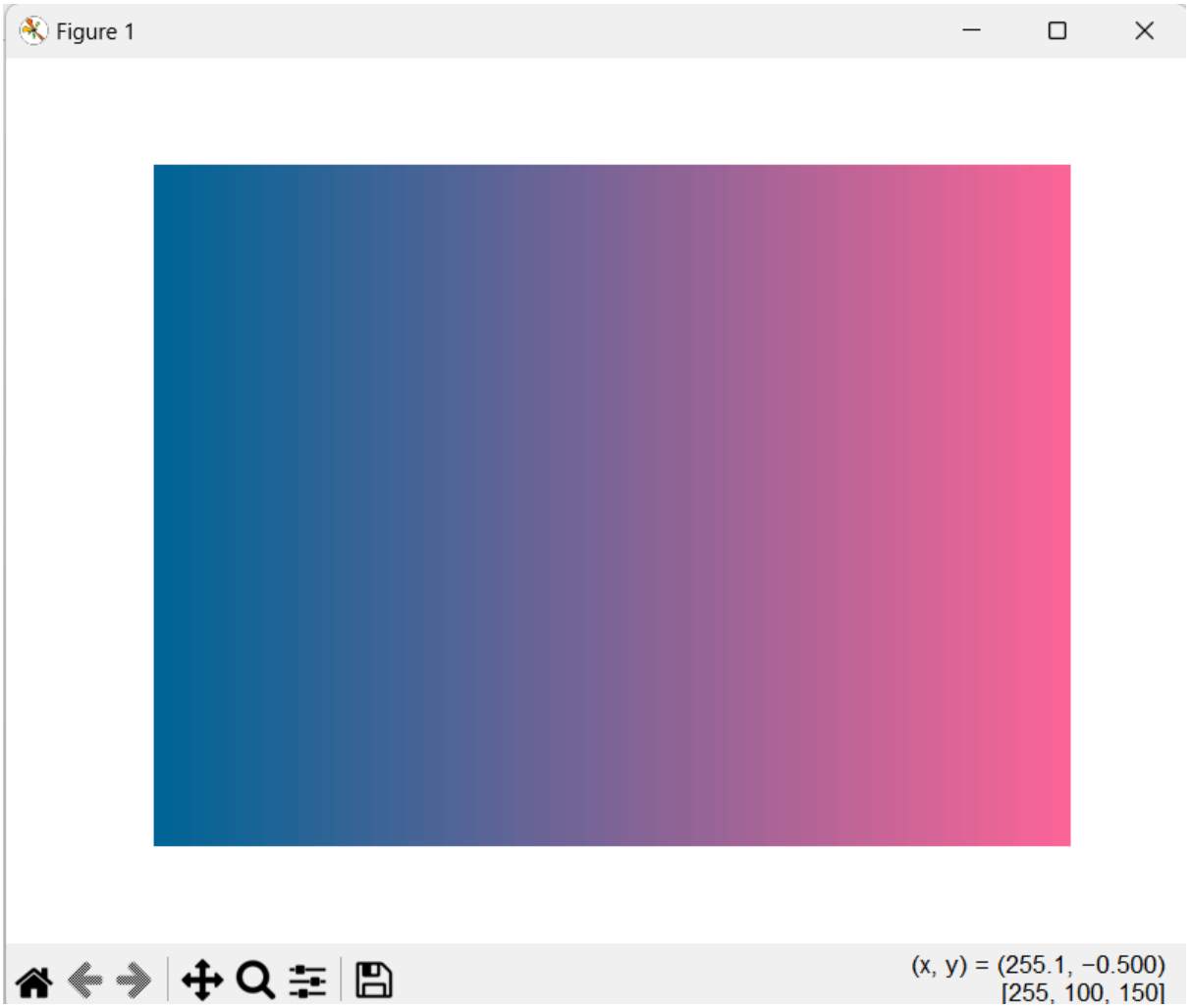
Descrição da atividade:

Exercício 1:

A atividade consistiu na implementação de um código para gerar e exibir um gradiente de cores utilizando a biblioteca matplotlib em Python. O objetivo era criar uma transição suave de cores variando o valor de vermelho (R) de 0 a 255, mantendo os valores de verde (G) e azul (B) constantes.

O código foi estruturado da seguinte forma:

1. Os valores constantes de verde (G = 100) e azul (B = 150) foram definidos.
2. Um laço foi utilizado para gerar o gradiente, variando o valor de vermelho (R) de 0 a 255. Para cada valor de R, uma cor foi criada no formato RGB e armazenada em uma lista.
3. A lista de cores foi então exibida como uma imagem, utilizando a função imshow da biblioteca matplotlib. A exibição foi feita de maneira a remover os eixos, proporcionando uma visualização limpa do gradiente.



Exercício 2:
A atividade envolveu a implementação de uma função para converter valores de cor no modelo RGB (Red, Green, Blue) para o modelo HSV (Hue, Saturation, Value) em Python. O código foi desenvolvido com o uso da biblioteca colorsys, que oferece a função rgb_to_hsv para a conversão.

Descrição do código:

1. Entrada de dados: O código solicita ao usuário que insira os valores para as componentes RGB de uma cor, que variam de 0 a 255.
2. Função rgb_to_hsv:
    a. Os valores de RGB são normalizados para o intervalo [0, 1], dividindo por 255.
    b. Em seguida, a conversão de RGB para HSV é realizada usando a função rgb_to_hsv da biblioteca colorsys.
    c. O valor de brilho (V) é ajustado de volta para o intervalo [0, 255], multiplicando o valor de HSV por 100 para adequá-lo ao formato tradicional.
3. Saída: O código exibe o resultado da conversão, mostrando os valores de matiz (H), saturação (S) e valor (V), com duas casas decimais.

```
C:\Users\felip\OneDrive\Documentos\SextoPeriodo\Computação Gráfica\Atividades\Atv_4>python exe_2.py

Digite o valor de R (0-255): 255
Digite o valor de G (0-255): 128
Digite o valor de B (0-255): 64
HSV: H=20.10, S=74.90, V=100.00
```

Exercício 3:

A atividade consiste em um programa Python que converte valores de cor no modelo RGB (Red, Green, Blue) para o modelo CMYK (Cyan, Magenta, Yellow, Black). O código começa solicitando ao usuário os valores de RGB, que são normalizados para o intervalo [0, 1] antes de serem usados nas fórmulas para calcular os valores de CMYK.

A função principal, rgb_to_cmyk, calcula o valor de K (Black) como o mínimo de 1 menos os valores de R, G e B. Se K for igual a 1, significa que a cor é preta, e os valores de C, M e Y são definidos como 0. Caso contrário, os valores de C (Cyan), M (Magenta) e Y (Yellow) são calculados com base nos valores de RGB e K. Em seguida, todos os valores são convertidos para porcentagens e retornados.

Na função main, o código valida se os valores RGB fornecidos estão dentro do intervalo permitido (0 a 255) e, em seguida, chama a função rgb_to_cmyk para realizar a conversão. O resultado é exibido na tela no formato de porcentagens de C, M, Y e K.

Além disso, o código conta com um tratamento de erros para garantir que os valores inseridos sejam válidos inteiros dentro do intervalo correto. Caso contrário, uma mensagem

de erro é exibida. Esse programa é útil para quem trabalha com design gráfico ou impressão, onde é necessário realizar conversões entre esses modelos de cores.

```
C:\Users\felip\OneDrive\Documentos\SextoPeriodo\Computação Gráfica\Atividades\Atv_4>python exe_3.py

Enter the Red value (0-255): 255
Enter the Green value (0-255): 128
Enter the Blue value (0-255): 64
RGB(255, 128, 64) converts to CMYK:
C = 0%, M = 50%, Y = 75%, K = 0%
```

```python
# Exercício 1

import matplotlib.pyplot as plt

# Valores constantes para G e B
g = 100
b = 150

# Lista para armazenar as cores do gradiente
gradient = []

# Gerando o gradiente RGB
for r in range(256):  # R varia de 0 a 255
    gradient.append((r, g, b))

# Exibindo o gradiente como uma imagem
plt.imshow([gradient], aspect='auto')
plt.axis('off')  # Remove os eixos
plt.show()
```

```python
# Exercício 2

import colorsys

def rgb_to_hsv(r, g, b):
    # Normaliza os valores RGB para o intervalo [0, 1]
    r, g, b = r / 255.0, g / 255.0, b / 255.0

    # Converte RGB para HSV
    h, s, v = colorsys.rgb_to_hsv(r, g, b)

    # Converte os valores de volta para o intervalo [0, 255] para o valor V
    return (h * 360, s * 100, v * 100)

# Entrada de valores RGB
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): "))

# Converte RGB para HSV
hsv = rgb_to_hsv(r, g, b)

# Exibe o resultado
print(f"HSV: H={hsv[0]:.2f}, S={hsv[1]:.2f}, V={hsv[2]:.2f}")
```

```python
# Exercício 3

def rgb_to_cmyk(r, g, b):
    # Normalize RGB values to the range [0, 1]
    r, g, b = r / 255.0, g / 255.0, b / 255.0

    # Calculate K (Black)
    k = min(1 - r, 1 - g, 1 - b)

    # If K is 1, it means the color is black, so C, M, Y are all 0
    if k == 1:
        c = m = y = 0
    else:
        # Calculate C, M, and Y values
        c = (1 - r - k) / (1 - k)
        m = (1 - g - k) / (1 - k)
        y = (1 - b - k) / (1 - k)

    # Convert to percentage and return the result
    return round(c * 100), round(m * 100), round(y * 100), round(k * 100)

def main():
    # Get RGB values from the user
    try:
        r = int(input("Enter the Red value (0-255): "))
        g = int(input("Enter the Green value (0-255): "))
        b = int(input("Enter the Blue value (0-255): "))

        # Validate input range
        if not (0 <= r <= 255 and 0 <= g <= 255 and 0 <= b <= 255):
            print("Error: RGB values must be between 0 and 255.")
            return

        # Convert RGB to CMYK
        c, m, y, k = rgb_to_cmyk(r, g, b)

        # Display the result
        print(f"RGB({r}, {g}, {b}) converts to CMYK:")
        print(f"C = {c}%, M = {m}%, Y = {y}%, K = {k}%")

    except ValueError:
        print("Error: Please enter valid integer values for RGB.")

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