FACULDADE ANHANGUERA DE GUARAPARI
ESTUDO DIRIGIDO II – 2024/2

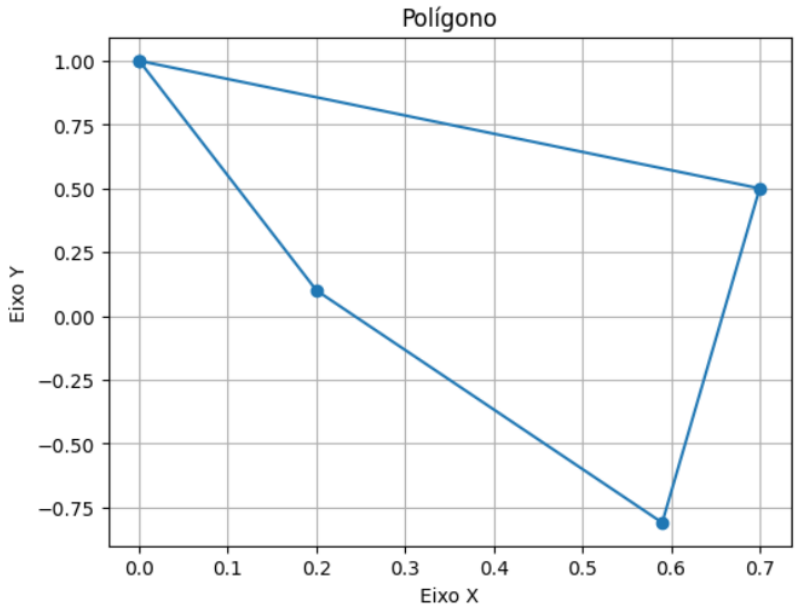
Nome:			
Professor(a):	Dr. Rodrigo Xavier de Almeida Leão	**Data:**	
Disciplina:	Computação Gráfica e Processamento de Imagens		
Curso:	Ciência da Computação	**Turma:**	

_____
Assinatura do Aluno(a)

1) Desenhe as figuras abaixo.

```python
import matplotlib.pyplot as plt
import numpy as np

# Função para desenhar um retângulo
def desenhar_retangulo(x, y, largura, altura, cor):
  plt.fill([x, x + largura, x + largura, x], [y, y, y + altura, y + altura], color=cor)

# Função para desenhar um losango
def desenhar_losango(x, y, largura, altura, cor):
  plt.fill([x, x + largura/2, x + largura, x + largura/2],
           [y + altura/2, y, y + altura/2, y + altura], color=cor)

# Função para desenhar um círculo
def desenhar_circulo(x, y, raio, cor):
  circle = plt.Circle((x, y), radius=raio, color=cor)
  plt.gca().add_patch(circle)

# Desenhar as formas
desenhar_retangulo(0, 0, 10, 15, 'green')
desenhar_losango(2, 3, 6, 8, 'yellow')
desenhar_circulo(5, 7, 2, 'blue')

# Configurar o gráfico
plt.axis('equal')
plt.axis('off')
plt.show()
```

2) Desenhe o polígono abaixo e aplique sobre ele uma rotação de 30° para a direita.

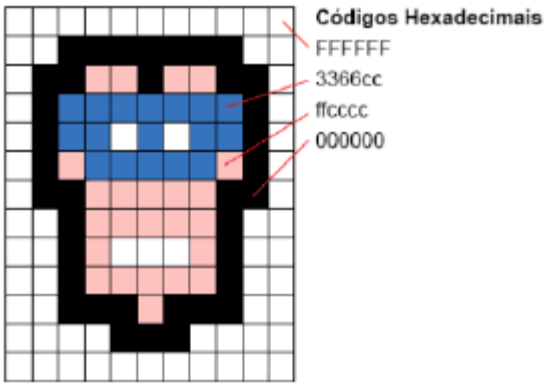



```python
import matplotlib.pyplot as plt
import numpy as np

def plot_polygon_with_lines(points):
  x_coords = [point[0] for point in points]
  y_coords = [point[1] for point in points]

  plt.plot(x_coords + [x_coords[0]], y_coords + [y_coords[0]], marker='o')
  plt.xlabel("Eixo X")
  plt.ylabel("Eixo Y")
  plt.title("Polígono")
  plt.grid(True)
  plt.show()

def apply_transformation(points, transformation_matrix):

  transformed_points = []
  for point in points:
    # Converter o ponto em um vetor coluna com coordenadas homogêneas (x, y, 1)
    if transformation_matrix.shape == (2, 2):
      homogeneous_point = np.array([point[0], point[1]])
      transformed_point = np.dot(transformation_matrix, homogeneous_point)
    elif transformation_matrix.shape == (3, 3):
      homogeneous_point = np.array([point[0], point[1], 1])
      transformed_point = np.dot(transformation_matrix, homogeneous_point)
      transformed_point = transformed_point[:2] / transformed_point[2]
    else:
      raise ValueError("A matriz de transformação deve ser 2x2 ou 3x3.")
    transformed_points.append((transformed_point[0], transformed_point[1]))
  return transformed_points
```

```
33
34 # Coordenadas dos vértices do pentágono
35 vertices_pentagono = [
36     (0, 1),
37     (0.2, 0.1),
38     (0.59, -0.81),
39
40     (0.7, 0.5)
41 ]
42
43 # Plotar o pentágono
44 plot_polygon_with_lines(vertices_pentagono)
45
46 # Exemplo de uso:
47 points = vertices_pentagono
48 # Matriz de transformação para rotação de 45 graus
49 angle = np.radians(45)
50 rotation_matrix = np.array([
51     [np.cos(angle), -np.sin(angle)],
52     [np.sin(angle), np.cos(angle)]  ])
53
54 transformed_points = apply_transformation(points, rotation_matrix)
55 plot_polygon_with_lines(transformed_points)
```

3) Plote a matriz de pixels abaixo.

FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF
FFFFFF	FFFFFF	000000	000000	000000	000000	000000	000000	000000	FFFFFF	FFFFFF
FFFFFF	000000	000000	FFCCCC	FFCCCC	000000	FFCCCC	FFCCCC	000000	000000	FFFFFF
FFFFFF	000000	3366CC	3366CC	3366CC	3366CC	3366CC	3366CC	3366CC	000000	FFFFFF
FFFFFF	000000	3366CC	3366CC	FFFFFF	3366CC	FFFFFF	3366CC	3366CC	000000	FFFFFF
FFFFFF	000000	FFCCCC	3366CC	3366CC	3366CC	3366CC	3366CC	FFCCCC	000000	FFFFFF
FFFFFF	000000	000000	FFCCCC	FFCCCC	FFCCCC	FFCCCC	FFCCCC	000000	000000	FFFFFF
FFFFFF	FFFFFF	000000	FFCCCC	FFCCCC	FFCCCC	FFCCCC	FFCCCC	000000	FFFFFF	FFFFFF
FFFFFF	FFFFFF	000000	FFCCCC	FFFFFF	FFFFFF	FFFFFF	FFCCCC	000000	FFFFFF	FFFFFF
FFFFFF	FFFFFF	000000	FFCCCC	FFCCCC	FFCCCC	FFCCCC	FFCCCC	000000	FFFFFF	FFFFFF
FFFFFF	FFFFFF	000000	000000	000000	FFCCCC	000000	000000	000000	FFFFFF	FFFFFF
FFFFFF	FFFFFF	FFFFFF	FFFFFF	000000	000000	000000	FFFFFF	FFFFFF	FFFFFF	FFFFFF
FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF	FFFFFF



```python
# prompt: criar uma matriz 4x4 preenchida com valores da escala RGB e mostrar um mapa de cores desta matriz

import matplotlib.pyplot as plt
import numpy as np

# Criar uma matriz 4x4 com valores RGB
matrix_rgb = np.array([
    [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0)],
    [(255, 0, 255), (0, 255, 255), (255, 255, 255), (0, 0, 0)],
    [(128, 0, 0), (0, 128, 0), (0, 0, 128), (128, 128, 0)],
    [(128, 0, 128), (0, 128, 128), (128, 128, 128), (64, 64, 64)]
])

# Normalizar os valores RGB para o intervalo [0, 1]
matrix_rgb_normalized = matrix_rgb / 255.0

# Mostrar o mapa de cores da matriz
plt.imshow(matrix_rgb_normalized)
plt.colorbar()
plt.title("Mapa de Cores da Matriz RGB")
plt.show()
```

4) Aplique as curvas de Bézier sobre os conjuntos de pontos de controle a seguir para desenhar um olho. Desenhe um círculo no centro do olho.

```python
import numpy as np
import matplotlib.pyplot as plt

def bezier_curve(P0, P1, P2, P3, t):
    """Calcula um ponto na curva de Bézier cúbica para um valor de t (0 <= t <= 1)"""
    # Convertendo os pontos de controle para arrays NumPy
    P0 = np.array(P0)
    P1 = np.array(P1)
    P2 = np.array(P2)
    P3 = np.array(P3)
    return (1-t)**3 * P0 + 3*(1-t)**2 * t * P1 + 3*(1-t) * t**2 * P2 + t**3 * P3

def plot_bezier(P0, P1, P2, P3, color='black'):
    """Plota uma curva de Bézier cúbica usando os pontos de controle"""
    t_values = np.linspace(0, 1, 100)
    points = [bezier_curve(P0, P1, P2, P3, t) for t in t_values]
    x, y = zip(*points)
    plt.plot(x, y, color=color)
```

```python
left_eye = [
    [(-2, 2), (-1.5, 2.5), (-0.5, 2.5), (0, 2)],    # Pálpebra superior
    [(-2, 2), (-1.5, 1.8), (-0.5, 1.8), (0, 2)]     # Pálpebra inferior
```

```python
for curve in left_eye:
    plot_bezier(*curve)
```

plt.show()