AC 4

Equipe:

- André Lucas Fabbris de Toledo RA 1902777
-
-
-
-
-

```
import numpy as np

from IPython.display import HTML

from matplotlib.animation import FuncAnimation
import matplotlib.pyplot as plt
%matplotlib inline


def plot_objeto(objeto, new_figure=True, figsize=(6,6), args={}):
  x = objeto[:,0]
  y = objeto[:,1]

  if new_figure:
    plt.figure(figsize=figsize)

  plt.plot(x,y, **args);
```

Questão 1: $(1,5\ pontos)$

Quais as coordenadas de um triângulo equilátero com lado de 1 unidade que começa centrado no ponto $(-1, 0)$ após ser translatado 5 vezes com o vetor $[2, -1]$ e 4 vezes com o vetor $[4, 5]$?

Apresente uma simulação apresentando cada translação considerando as translações com o vetor $[2, -1]$ seguidas pelas translações com o vetor $[4, 5]$.

Apresente uma segunda simulação apresentando cada translação considerando as translações com o vetor $[2, -1]$ intercaladas pelas translações com o vetor $[4, 5]$.

Questão 2: $(1,5\ pontos)$

Qual o menor ângulo α que devemos rotacionar um pentágono regular em torno de seu centro para que cada vértice coincida com a posição do vértice adjacente?

Apresente uma simulação utilizando um polígono centrado no ponto $(5, 5)$ e lado de 1 unidade.

Questão 3: $(1,5\ pontos)$

Qual a área de um polígono com vértices em $\{(-2, -2), (-1, -1), (1, -1), (2, -2), (0, 5)\}$? Fazendo $s_x = 2$, qual deve ser o valor de s_y para termos um polígono com 6 vezes a área original? Mostre que o polígono após escalonado apresenta 6 vezes a área original.

Apresente uma simulação com esse polígono e o escalonamento.

```
objeto = np.array([[-2,-2],
                   [-1,-1],
                   [ 1,-1],
```

```
                    [ 2,-2],
                    [ 0, 5],
                    [-2,-2]])

objeto_homogeneas = np.hstack((objeto, np.ones((6,1))))
```

Questão 4: $(2,5\ pontos)$

Apresente uma simulação como animação de uma roda rodando sobre uma superfície horizontal. A roda deve apresentar ao menos 2 aros ortogonais que passam pelo centro da roda.

- $0,5\ pontos$ pela combinação da rotação com translação;
- $1,0\ ponto$ para a rodar dar 1 volta completa na animação;
- $1,0\ ponto$ se a roda não apresentar o efeito de derrapagem.

A animação deve ter de 5 à 10 segundos.

```
fig, ax = plt.subplots()

ax.set_aspect('equal')
ax.set_xlim((-1.2, 1.2))
ax.set_ylim((-1.2, 1.2))

reta = np.array([[-50,-1, 1],
                 [ 50,-1, 1]])

liner, = ax.plot([], [], lw=4)
linep, = ax.plot([], [], lw=2)
line1, = ax.plot([], [], lw=2)
line2, = ax.plot([], [], lw=3)
line3, = ax.plot([], [], lw=3)
line4, = ax.plot([], [], lw=2)
line5, = ax.plot([], [], lw=3)
line6, = ax.plot([], [], lw=3)
```

```
def init():

  x = reta[:, 0]
  y = reta[:, 1]

  liner.set_data(x, y)
  linep.set_data([], [])
  line1.set_data([], [])
  line2.set_data([], [])
  line3.set_data([], [])
  line4.set_data([], [])
  line5.set_data([], [])
  line6.set_data([], [])

  liner.set_color('black')
  linep.set_color('blue')
  line1.set_color('blue')
  line2.set_color('green')
  line3.set_color('green')
  line4.set_color('blue')
  line5.set_color('green')
  line6.set_color('green')

  return (liner, linep, line1, line2, line3, line4, line5, line6)


def animate1(i):

  d = i * (1/10)
  translacao = np.array([[1, 0, 0],
                         [0, 1, 0],
                         [d, 0, 1]])


  reta_i = np.matmul(reta, translacao)
  xr = reta_i[:, 0]
```

```
    yr = reta_i[:, 1]

    t = np.linspace(0, np.pi/2,50) + i*np.pi/100
    tp = np.linspace(0, np.pi*2) + i*np.pi/100


    xp = np.cos(tp)
    yp = np.sin(tp)
    x = np.cos(t)
    y = np.sin(t)
    x2 = -x
    y2 = -y

    liner.set_data(xr, yr)
    linep.set_data(xp, yp)

    line1.set_data(x, y)
    line2.set_data([0, x[0]], [0, y[0]])
    line3.set_data([0, x[-1]], [0, y[-1]])

    line4.set_data(x2, y2)
    line5.set_data([0, x2[0]], [0, y2[0]])
    line6.set_data([0, x2[-1]], [0, y2[-1]])




    return (liner, linep, line1, line2, line3, line4, line5, line6)



anim = FuncAnimation(fig, animate1, init_func=init,
                     frames=200, interval=50, blit=True)


HTML(anim.to_html5_video())
```

0:03 / 0:10

Questão 5: $(3,0\ pontos)$

Apresente uma simulação como animação de tema livre, com dois ou mais objetos e ao menos um objeto com movimentos. Os objetos e o movimento devem ter sentido semântico, ou seja, o significado tanto dos objetos quanto do movimento devem ser evidentes.

Objetos complexos, com duas ou mais partes, serão aceitos.

- $1,0\ ponto$ pelo uso de 2 ou mais transformações geométricas para o movimento;
- $1,0\ ponto$ pela naturalidade do movimento;
- $1,0\ ponto$ pela descrição dos objetivos e de como a animação foi feita no código.

```
fig2, ax = plt.subplots()

ax.set_aspect('equal')
ax.set_xlim((-1.2, 1.2))
ax.set_ylim((-1.2, 1.2))

linep, = ax.plot([], [], lw=2)
line1, = ax.plot([], [], lw=2)
```

```
    line4, = ax.plot([], [], lw=2)
    line2, = ax.plot([], [], lw=3)
    line3, = ax.plot([], [], lw=3)


    def init2():
      linep.set_data([], [])
      line1.set_data([], [])
      line2.set_data([], [])
      line3.set_data([], [])
      line4.set_data([], [])

      linep.set_color('black')
      line1.set_color('red')
      line2.set_color('green')
      line3.set_color('green')
      line4.set_color('red')

      return (line1, line2, line3, line4, linep)


    def radar(i):
      t = np.linspace(0,np.pi/2,50) + i*np.pi/100
      tp = np.linspace(0,np.pi*2,50) + i*np.pi/100

      xp = np.cos(tp)
      yp = np.sin(tp)
      x = np.cos(t)
      y = np.sin(t)
      linep.set_data(xp,yp)
      line1.set_data(x, y)
      line4.set_data(x/2, y/2)
      line2.set_data([0, x[0]], [0, y[0]])
      line3.set_data([0, x[-1]], [0, y[-1]])




      return (line1, line2, line3, line4, linep)
```

```
    return (line1, line2, line3, line4, line5)
```

```
anim2 = FuncAnimation(fig2, radar, init_func=init2,
                      frames=200, interval=50, blit=True)
```

```
HTML(anim2.to_html5_video())
```