Aula 7

Introdução

Nessa aula iremos dar continuidade na codificação do jogo “**Falling Apples**” em python utilizando a biblioteca gráfica **turtle**.

Essa aula será dividida em 3 partes: Criação do Placar, Adição de Imagens e Adição de Sons.

Criação do Placar

Primeiramente, iremos uma classe que será responsável pela apresentação de textos na tela do jogo. Essa nova classe deve ser criada após a classe Maca.

```
#Criação da classe de notificação
class Texto(Elemento):                                                          #1

   def __init__(self, formato, cor, posicao_x, posicao_y, velocidade, fonte):  #2
       Elemento.__init__(self, formato, cor, posicao_x, posicao_y, velocidade) #3
       self.hideturtle()                                                        #4
       self.fonte = fonte                                                       #5

   def escrever(self, texto):                                                   #6
       self.clear()                                                             #7
       self.write(texto, align="center", font = self.fonte)                     #8
```

Código 1: Criação da classe Texto

Para facilitar o entendimento iremos explicar linha a linha do nosso código:

#1 : Iremos criar uma classe que será responsável pelos textos apresentados na tela. Chamaremos ela de Texto, e a mesma é filha da classe Elemento e logo neta da classe Turtle, ou seja, a classe Texto herdará os atributos e métodos/funções da classe pai (Elemento) e da classe avô (Turtle).

#2 : Criaremos a função **__init__** essa é uma função padrão das classes do python e a mesma é a primeira função executada após a criação de um objeto dessa classe.Utilizando o turtle iremos criar a tela do nosso jogo.

#3 : Chamamos o **__init__** da classe pai (Elemento).

#4 : Ocultamos o turtle/ponteiro do objeto.

#5 : Atribuímos a fonte passada na criação do objeto para o objeto criado.

#6 : Criamos uma função escrever, que será responsável por “escrever” o objeto na tela. Lembrando que essa classe terá como objetos o placar e a mensagem de Game Over.

#7 : Limpa o texto.

#6 : Escreve o texto na tela.

Após a criação da classe Texto, iremos criar o objeto placar e ao mesmo tempo alteramos a condição de parada do loop principal do jogo:

```
placar = Texto("circle", "white", 0, 260, 0, ("Arial", 24, "normal"))
placar.escrever("Pontuação: {} - Vidas: {}".format(cesta.pontuacao, cesta.vidas))

#Loop principal do jogo
while cesta.vidas > 0:
```

Código 2: Criação do objeto placar e alteração da condição do loop principal do jogo

Criamos o placar como um objeto da classe Texto, a qual passamos o formato como círculo, a cor como branca, a posição horizontal x como 0, a posição vertical y como 260, a velocidade como 0 e a fonte como Arial com tamanho 24 e decoração normal.

Após isso chamamos a função escrever do objeto placar com o texto “Pontuação: {} - Vidas: {}”, sendo que os elementos {} serão substituídos pelos valores da pontuação do jogador e as vidas restantes do jogador.

E por fim alteramos a condição do loop principal do jogo para cesta.vidas > 0, ou seja, enquanto o jogador possuir vidas restantes continua o jogo.

Aproveitando que estamos alterando a condição do loop principal do jogo, também iremos criar o texto do game over após o loop principal do jogo.

```
   cesta.movimenta_cesta()


game_over = Texto("circle", "red", 0, 0, 0, ("Arial", 48, "bold"))
game_over.escrever("GAME OVER!")

jogo.mainloop()
```

Código 3: Criação do objeto game_over da classe Texto

Por fim, iremos alterar a função movimenta da classe Maca, para que a mesma contabilize a pontuação do jogador e as vidas restantes do mesmo.

```
  def movimenta(self, cesta):
      y = self.ycor()
      y -= self.speed()/5
      self.sety(y)
      if self.ycor() < -300:
          self.reposicionar_maca()
      if Elemento.colidiu(cesta, self):
          self.reposicionar_maca()
          if self.podre == False:                                                    #1
              cesta.pontuacao += 1                                                   #2
          else:                                                                      #3
              cesta.vidas -= 1                                                       #4
          placar.escrever("Pontuação: {} - Vidas: {}".format(cesta.pontuacao, cesta.vidas))    #5

#Criação da classe de notificação
```

Código 4: Adição da contabilização da pontuação e das vidas restantes do jogador

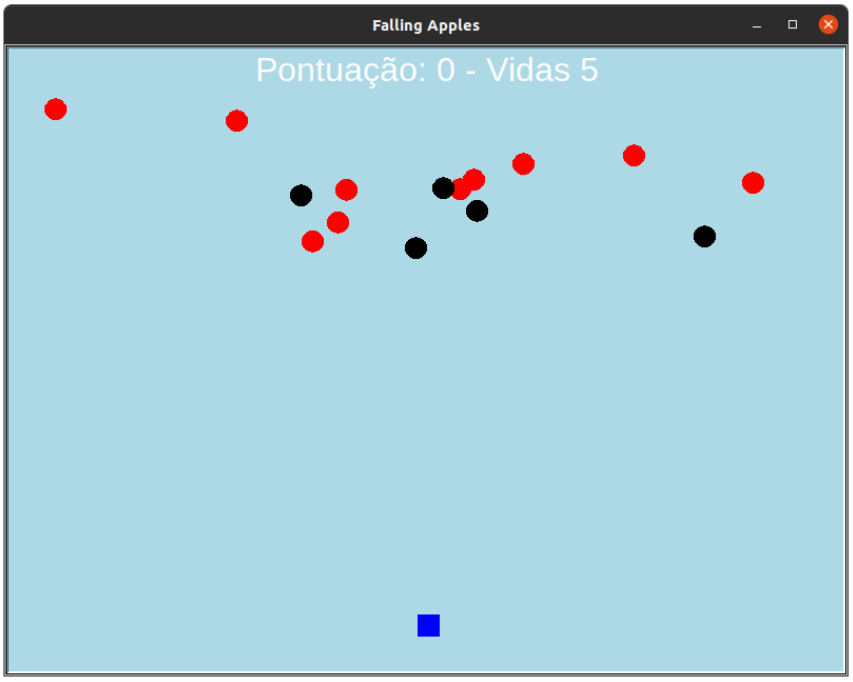
#1 : Verifica se a maçã não é uma maçã podre.

#2 : Se a maçã não for uma maçã podre soma 1 na pontuação do jogador

#3 e #4 : Senão, ou seja, se a maçã for uma maçã podre, subtrai uma vida das vidas restantes do jogador.

#5 : Atualiza o objeto placar, chamando a função escrever do mesmo.

Segue abaixo o resultado da execução do nosso código:



Tela 1: Apresentação do placar durante o jogo

Tela 2: Apresentação da mensagem de game over ao final do jogo

Adição de Imagens

Para adicionarmos imagens aos nossos elementos (jogador/maçã/maçã podre) do jogo, primeiramente precisamos registrar a imagem como um formato, assim sendo possível utilizar esse objeto pelo atributo shape do turtle. Para registrar uma imagem basta utilizar o comando **register_shape**. No código abaixo, atribuímos uma imagem de background para a tela do nosso jogo pelo comando **bgpic** (apple_trees.gif) e registramos 3 imagens (apple.gif, bad_apple.gif e basket.gif).

```
#Criação da tela do jogo
jogo = turtle.Screen()
jogo.title("Falling Apples")
jogo.bgcolor("light blue")
jogo.bgpic("apple_trees.gif")
jogo.setup(width=800, height=600)
jogo.tracer(0)
```

```
jogo.register_shape("apple.gif")
jogo.register_shape("bad_apple.gif")
jogo.register_shape("basket.gif")

#Criação da classe de Elementos (Pai)
class Elemento(turtle.Turtle):
```

Código 5: Atribuição da imagem de background do jogo e registrando as imagens do jogador, maçã e maçã podre

Para atribuirmos a imagem “basket.gif” como o formato do jogador, basta alterar o formato “square” pelo “basket.gif”. Como o mesmo já foi registrado a alteração é simples, conforme apresentado abaixo:

```
#Adicionar o jogador
cesta = Cesta("basket.gif", "blue", 0, -250, 0)
```

Código 6: Atribuindo a imagem “basket.gif” como o formato do jogador

Para atribuirmos a imagem “apple.gif” como o formato da maçã, basta alterar o formato “circle” pelo “apple.gif”, conforme apresentado abaixo:

```
for _ in range(numero_de_macas):
   macas.append(Maca("apple.gif", "red", random.randint(-380, 380),
random.randint(200, 300), random.randint(1, 4), False, "apple.wav"))
```

Código 7: Atribuindo a imagem “apple.gif” como o formato da maçã

Para atribuirmos a imagem “bad_apple.gif” como o formato da maçã podre, basta alterar o formato “circle” pelo “bad_apple.gif”, conforme apresentado abaixo:

```
for _ in range(numero_de_macas_podres):
   macas_podres.append(Maca("bad_apple.gif", "black",
random.randint(-380, 380), random.randint(200, 300), random.randint(1,
4), True, "smash.wav"))
```

Código 8: Atribuindo a imagem “bad_apple.gif” como o formato da maçã podre

Segue abaixo o resultado da execução do nosso código:

Tela 3: Jogo com imagem de background, imagens como formato para o jogador, maçãs e maçãs podres.

Adição de Sons

Para adicionarmos sons primeiramente temos que importar a biblioteca de execução dos arquivos de áudio.

```
import os              #1A
# import playsound     #1B
# import winsound      #1C
```

Código 9: Import da biblioteca de execução de arquivos de áudio

#1A : Biblioteca de execução de áudios para Linux e MAC.

#1B : Biblioteca de execução de áudios para diversos Sistemas Operacionais.

#1C : Biblioteca de execução de áudios para windows.

Após isso adicionamos a execução do áudio na verificação de colisão na função movimenta da classe Maca:

```
    def movimenta(self, cesta):
        y = self.ycor()
        y -= self.speed()/5
        self.sety(y)
        if self.ycor() < -300:
            self.reposicionar_maca()
        if Elemento.colidiu(cesta, self):
            self.reposicionar_maca()
            if self.podre == False:
                cesta.pontuacao += 1
            else:
                cesta.vidas -= 1
            os.system("aplay " + self.som + "&")                    #1A
            # os.system("afplay " + self.som + "&")                 #1B
            # playsound.playsound(self.som, block=False)            #1C
            # winsound.PlaySound(self.som, winsound.SND_FILENAME)   #1D
            # winsound.PlaySound(self.som, winsound.SND_ASYNC)      #1E
            placar.escrever("Pontuação: {} - Vidas:
{}".format(cesta.pontuacao, cesta.vidas))
```

Código 10: Adicionando os comandos de execução do arquivo de áudio caso detectado uma colisão

#1A : Execução do arquivo de áudio caso ocorreu uma colisão. Para Linux.

#1B : Execução do arquivo de áudio caso ocorreu uma colisão. Para MAC.

#1C : Execução do arquivo de áudio caso ocorreu uma colisão. Para diversos Sistemas Operacionais.

#1D : Execução do arquivo de áudio caso ocorreu uma colisão. Para Windows, o som é executado síncrono ao rodar o jogo.

#1E : Execução do arquivo de áudio caso ocorreu uma colisão. Para Windows, o som é executado assíncrono ao rodar o jogo.

E por fim, definimos os sons das maçãs e das maçãs podres na criação dos objetos da classe Maca:

```
for _ in range(numero_de_macas):
   macas.append(Maca("apple.gif", "red", random.randint(-380, 380),
random.randint(200, 300), random.randint(1, 4), False, "apple.wav"))
```

Código 11: Atribuindo o arquivo de áudio“apple.wav” como o som da maçã

```
for _ in range(numero_de_macas_podres):
   macas_podres.append(Maca("bad_apple.gif", "black",
random.randint(-380, 380), random.randint(200, 300), random.randint(1,
4), True, "smash.wav"))
```

Código 12: Atribuindo o arquivo de áudio“smash.wav” como o som da maçã podre

Código Completo:

Segue abaixo o código completo do jogo:

```
import turtle
import random
import os
# import playsound
# import winsound

numero_de_macas = 10
numero_de_macas_podres = 5


#Criação da tela do jogo
jogo = turtle.Screen()
jogo.title("Falling Apples")
jogo.bgcolor("light blue")
jogo.bgpic("apple_trees.gif")
jogo.setup(width=800, height=600)
jogo.tracer(0)

jogo.register_shape("apple.gif")
jogo.register_shape("bad_apple.gif")
jogo.register_shape("basket.gif")

#Criação da classe de Elementos (Pai)
class Elemento(turtle.Turtle):
```

```
    def __init__(self, formato, cor, posicao_x, posicao_y, velocidade):
        turtle.Turtle.__init__(self)
        self.speed(velocidade)
        self.penup()
        self.shape(formato)
        self.color(cor)
        self.goto(posicao_x, posicao_y)
        self.direction = "stop"

    def colidiu(elemento_0, elemento_1):
        if elemento_1.distance(elemento_0) < 40:
            return True
        else:
            return False

#Criação da classe de Cesta (jogador)
class Cesta(Elemento):

    def __init__(self, formato, cor, posicao_x, posicao_y, velocidade):
        Elemento.__init__(self, formato, cor, posicao_x, posicao_y,
velocidade)
        self.pontuacao = 0
        self.vidas = 3

    def movimenta_para_direita(self):
        self.direction = "right"

    def movimenta_para_esquerda(self):
        self.direction = "left"


    def movimenta(self, distancia):
        x = self.xcor() #Localizar a posição x do self
        x += distancia
        if x < 390 and x > -390:
            self.setx(x)
        self.direction = "stop"

    def movimenta_cesta(self):
        if self.direction == "right":
            self.movimenta(20)
        if self.direction == "left":
```

```
            self.movimenta(-20)

#Criacao da classe da Maçã
class Maca(Elemento):

    def __init__(self, formato, cor, posicao_x, posicao_y, velocidade,
podre, som):
        Elemento.__init__(self, formato, cor, posicao_x, posicao_y,
velocidade)
        self.podre = podre
        self.som = som

    def reposicionar_maca(self):
        x = random.randint(-380, 380)
        y = random.randint(200, 300)
        self.goto(x, y)

    def movimenta(self, cesta):
        y = self.ycor()
        y -= self.speed()/5
        self.sety(y)
        if self.ycor() < -300:
            self.reposicionar_maca()
        if Elemento.colidiu(cesta, self):
            self.reposicionar_maca()
            if self.podre == False:
                cesta.pontuacao += 1
            else:
                cesta.vidas -= 1
            os.system("aplay " + self.som + "&")
            # os.system("afplay " + self.som + "&")
            # playsound.playsound(self.som, block=False)
            # winsound.PlaySound(self.som, winsound.SND_FILENAME)
            # winsound.PlaySound(self.som, winsound.SND_ASYNC)
            placar.escrever("Pontuação: {} - Vidas:
{}".format(cesta.pontuacao, cesta.vidas))

#Criação da classe de notificação
class Texto(Elemento):

    def __init__(self, formato, cor, posicao_x, posicao_y, velocidade,
fonte):
        Elemento.__init__(self, formato, cor, posicao_x, posicao_y,
```

```
velocidade)
        self.hideturtle()
        self.fonte = fonte

    def escrever(self, texto):
        self.clear()
        self.write(texto, align="center", font = self.fonte)


#Adicionar o jogador
cesta = Cesta("basket.gif", "blue", 0, -250, 0)

#Criar maçãs
macas = []

for _ in range(numero_de_macas):
    macas.append(Maca("apple.gif", "red", random.randint(-380, 380),
random.randint(200, 300), random.randint(1, 4), False, "apple.wav"))

#Criar maçãs podres
macas_podres = []

for _ in range(numero_de_macas_podres):
    macas_podres.append(Maca("bad_apple.gif", "black",
random.randint(-380, 380), random.randint(200, 300), random.randint(1,
4), True, "smash.wav"))


#Vincular o teclado
jogo.listen()
jogo.onkeypress(cesta.movimenta_para_direita, "Right")
jogo.onkeypress(cesta.movimenta_para_esquerda, "Left")


placar = Texto("circle", "white", 0, 260, 0, ("Arial", 24, "normal"))
placar.escrever("Pontuação: {} - Vidas: {}".format(cesta.pontuacao,
cesta.vidas))

#Loop principal do jogo
while cesta.vidas > 0:

    jogo.update()
```

```
    #Movimentar as maçãs
    for maca in macas:
        maca.movimenta(cesta)

    #Movimentar as maçãs podres
    for maca_podre in macas_podres:
        maca_podre.movimenta(cesta)

    #Movimentar o jogador
    cesta.movimenta_cesta()


game_over = Texto("circle", "red", 0, 0, 0, ("Arial", 48, "bold"))
game_over.escrever("GAME OVER!")

jogo.mainloop()
```

Código 12: Código completo do jogo

Lembrando que os arquivos de imagem e de áudio devem estar presentes no mesmo diretório do arquivo de código-fonte do jogo.