Aula 5

Introdução

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

Utilizaremos a IDE Visual Studio Code durante o desenvolvimento deste jogo.

Para mais informações sobre a biblioteca **turtle**, acesse: https://docs.python.org/3/library/turtle.html

Criação da Tela Principal do Jogo

```
import turtle                           #1

#Criação da tela do jogo
jogo = turtle.Screen()                  #2
jogo.title("Falling Apples")            #3
jogo.bgcolor("light blue")              #4
jogo.setup(width=800, height=600)       #5

jogo.mainloop()                         #6
```

Código 1: Criação da tela principal do jogo

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

#1 : Primeiramente precisamos importar a biblioteca gráfica turtle para que seja possível utilizar as suas funções no nosso código.

#2 : Utilizando o turtle iremos criar a tela do nosso jogo.

#3 : Definimos um nome para a nossa tela. Seria o nome que aparecerá no topo da janela.

#4 : Definimos uma cor de fundo para a nossa tela. No exemplo iremos utilizar a cor “light blue” ou “azul claro”. Lembrando que o nome da cor deve ser escrito em inglês.

#5 : Definimos o tamanho da janela/tela do nosso jogo. Iremos usar as seguintes dimensões: 800 pixels de largura e 600 pixels de altura.

#6 : A função **mainloop()** mantém a tela em loop ou seja mantém ela aberta. Caso não seja adicionado esse comando, a janela do seu jogo será aberta e logo na sequência fechada, assim finalizando a execução do seu jogo.

Segue abaixo o resultado da execução do nosso código:

Tela 1: Tela principal do jogo

Criação do Jogador

```
import turtle

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

#Criação da classe de Elementos (Pai)
class Elemento(turtle.Turtle):                                          #1

   def __init__(self , formato, cor, posicao_x, posicao_y, velocidade):        #2
```

```
        turtle.Turtle.__init__(self)                                          #3
        self.speed(velocidade)                                                #4
        self.penup()                                                          #5
        self.shape(formato)                                                   #6
        self.color(cor)                                                       #7
        self.goto(posicao_x, posicao_y)                                       #8
        self.direction = "stop"                                               #9

#Criação da classe de Cesta (jogador)
class Cesta(Elemento):                                                        #10

    def __init__(self, formato, cor, posicao_x, posicao_y, velocidade):       #11
        Elemento.__init__(self, formato, cor, posicao_x, posicao_y, velocidade) #12
        self.pontuacao = 0                                                    #13
        self.vidas = 3                                                        #14

# Adicionar o jogador
cesta = Cesta("square","blue", 0, -250, 0)                                    #15

jogo.mainloop()
```

Código 2: Criação do jogador

#1 : Iremos criar uma classe que será responsável por todos os elementos que aparecerão no nosso jogo. Chamaremos ela de Elemento, e a mesma é filha da classe Turtle, ou seja, a classe Elemento herdará os atributos e métodos/funções da classe pai (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. Observação: o parâmetro self, faz referência ao próprio objeto.

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

#4 : Atribuímos o parâmetro velocidade ao atributo speed, o qual representa a velocidade que a animação do elemento se move na tela. São aceitos valores entre 0 a 10 sendo 0 o mais rápido e 10 o mais lento.

#5 : No turtle ao mover um elemento na tela é desenhado uma linha na mesma, para evitar isso é necessário adicionar o comando #5.

#6 : Atribuímos o parâmetro formato ao atributo shape, o qual representa a forma que o nosso elemento será “desenhado” na tela. Entre os valores aceitos estão: “arrow”, “turtle”, “circle”, “square”, “triangle” e “classic”. Na aula 06 iremos ensinar como definir uma imagem como um formato válido.

#7 : Atribuímos o parâmetro cor ao atributo color, o qual representa a cor que o nosso elemento será “desenhado” na tela.

#8 : O método goto está sendo utilizado aqui para posicionar o nosso elemento na tela. Para isso precisamos passar como parâmetros as coordenadas x e y que desejamos que nosso elemento seja “desenhado”. Lembrando que no turtle o centro da tela seria a posição (0, 0), assim no nosso exemplo os possíveis valores para x seriam entre -400 e 400 e para y seriam entre -300 e 300.

#9 : O atributo direction é um atributo que criamos que será responsável por armazenar para qual direção o nosso elemento se moverá na tela (principalmente o nosso jogador). Atribuímos o valor “stop” para representar que o elemento não irá se movimentar por enquanto.

#10 : Iremos criar uma classe que será responsável pelo nosso jogador. Chamaremos ela de Cesta, e a mesma é filha da classe Elemento e logo neta da classe Turtle, ou seja, a classe Cesta herdará os atributos e métodos/funções da classe pai (Elemento) e da classe avô (Turtle).

#11 : 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.

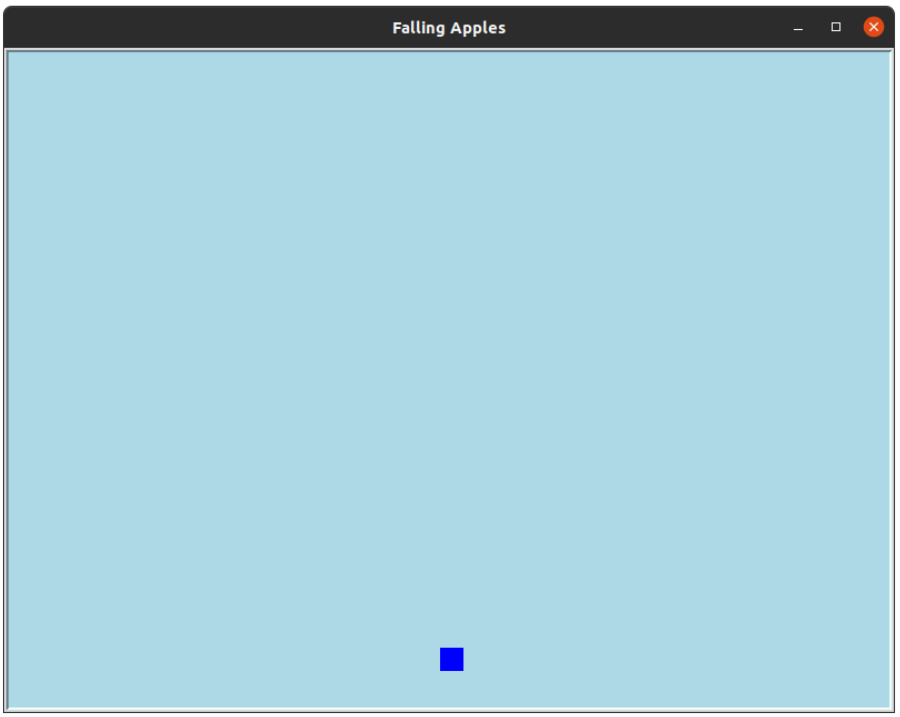
#12 : Chamamos o **__init__** da classe pai (Elemento).

#13 : Atribuímos 0 ao atributo pontuação, o qual representa a pontuação do jogador durante a execução do jogo.

#14 : Atribuímos 3 ao atributo vidas, o qual representa a quantidade de “vidas” que o jogador terá durante a execução do jogo.

#15 : Criamos o objeto cesta da classe Cesta, passando como formato “square” (quadrado), a cor “blue” (azul), a posição horizontal x como 0, a posição vertical y como -250 (pois gostaríamos que o jogador ficasse posicionado no final da tela) e a velocidade como 0.

Segue abaixo o resultado da execução do nosso código:



Tela 2: Criação do jogador

Movimentação do Jogador

```
import turtle

#Criação da tela do jogo
jogo = turtle.Screen()
jogo.title("Falling Apples")
jogo.bgcolor("light blue")
jogo.setup(width=800, height=600)
jogo.tracer(0)                                                          #1

#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"

#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):                                    #2
       self.direction = "right"                                         #3

   def movimenta_para_esquerda(self):                                   #4
       self.direction = "left"                                          #5

   def movimenta(self, distancia):                                      #6
       x = self.xcor()                                                  #7
       x += distancia                                                   #8
       if x < 390 and x > -390:                                         #9
           self.setx(x)                                                 #10
       self.direction = "stop"                                          #11

   def movimenta_cesta(self):                                           #12
       if self.direction == "right":                                    #13
           self.movimenta(20)                                           #14
       if self.direction == "left":                                     #15
           self.movimenta(-20)                                          #16

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

```
# Keyboard Binding
jogo.listen()                                                    #17
jogo.onkeypress(cesta.movimenta_para_esquerda, "Left")           #18
jogo.onkeypress(cesta.movimenta_para_direita, "Right")           #19

#Loop principal do jogo
while True:                                                      #20

   jogo.update()                                                 #21

   #Movimentar o jogador
   cesta.movimenta_cesta()                                       #22

jogo.mainloop()
```

Código 3: Movimentação do jogador

#1 : Será necessário adicionar esse comando para que não apresente uma animação (animação do turtle) e para que não ocorra delay ao atualizar a tela do jogo.

#2 e #3 : Criaremos a função *movimenta_para_direita* na classe Cesta, onde a mesma apenas atribui o valor “right” para o atributo direction, ou seja, ao chamar essa função o jogador passará a ter a sua direção (direction) como direita (right).

#4 e #5 : Criaremos a função *movimenta_para_esquerda* na classe Cesta, onde a mesma apenas atribui o valor “left” para o atributo direction, ou seja, ao chamar essa função o jogador passará a ter a sua direção (direction) como esquerda (left).

#6 : Criaremos a função movimenta que será responsável por mover o jogador a uma certa distância, a qual será passada por parâmetro. Ou seja, caso a distância seja positiva o jogador se moverá para a direita, caso contrário o jogador se moverá para a esquerda.

#7 : Declaramos x a qual receberá a posição horizontal do jogador pelo resultado da função xcor().

#8 : Adicionamos a distância a posição horizontal do jogador (x) para gerar a nova posição do jogador.

#9 : Verificamos se a nova posição horizontal x do jogador está dentro dos limites da tela do jogo. Lembrando que os limites horizontais da tela são -400 e 400, porém não queremos que o nosso jogador “suma” da tela e também que não fique “colado” no final da mesma, assim ao invés de verificar com os valores limites, verificaremos com os valores -390 e 390.

#10 : Como a nova posição está dentro dos limites da tela, atribuímos a mesma ao nosso jogador, assim movimentando ele na tela.

#11 : Atribuímos “stop” para a direção do nosso jogador, assim informando que ele já se movimentou como deveria.

#12 : Criaremos a função *movimenta_cesta* que será responsável por definir a distância e direção que o jogador deve se mover na tela.

#13 : Verifica se a direção do jogador é direita (right), ou seja, verifica se o jogador deve ou não se mover para a direita.

#14 : Chama a função *movimenta* passando como parâmetro a distância 20, ou seja, o jogador irá se mover 20 pixels para a direita.

#15 : Verifica se a direção do jogador é direita (left), ou seja, verifica se o jogador deve ou não se mover para a esquerda.

#16 : Chama a função *movimenta* passando como parâmetro a distância -20, ou seja, o jogador irá se mover 20 pixels para a esquerda.

#17 : Com a função listen() iremos passar a receber eventos do teclado, ou seja, agora poderemos verificar se determinada tecla do teclado foi pressionada.

#18 : Definimos que no evento da tecla Left (seta da esquerda do teclado) for pressionada, o jogo deve executar a função *movimenta_para_esquerda* do jogador. Observação: a seta para esquerda do teclado é definida pelo marcador Left.

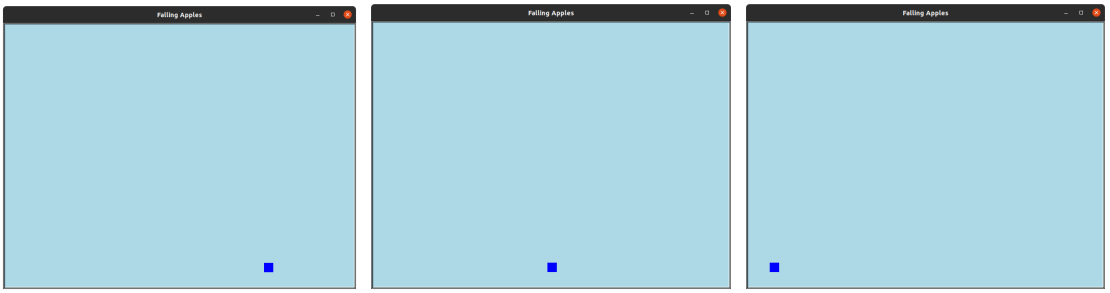
#19 : Definimos que no evento da tecla Right (seta da direita do teclado) for pressionada, o jogo deve executar a função *movimenta_para_direita* do jogador. Observação: a seta para direita do teclado é definida pelo marcador Right.

#20 : Criaremos o loop principal do nosso jogo, ou seja, enquanto a condição do loop não for verdadeira o jogo continuará a sua execução. No momento a condição que informamos é **True**, ou seja, a condição sempre será verdadeira, deixando assim o jogo em loop infinito.

#21 : A função update() é responsável por atualizar a tela do jogo, ela é necessária pois utilizamos comando **#1**. O tracer e o update estão sendo utilizados pois foram reportados em alguns sistemas operacionais alguns problemas com a atualização da tela pelo turtle. Adicionando as duas funções esses problemas não ocorrem.

#22 : Chamaremos a função *movimenta_cesta* do jogador, a qual moverá o jogador de acordo com a direction do mesmo.

Vale ressaltar que como os **#21** e **#22** estão dentro do loop principal do jogo eles serão executados a cada iteração do loop.



Tela 3: Movimentação do jogador