Introdução ao Pygame

Criando o Brick Breaker II

Python para Todos

CEFET-MG

Conteúdo

Organização do código

É possível separar a definição de classes e a criação de objetos em arquivos diferentes no Python. Iremos elaborar os módulos:

`brick_breaker.py`

`jogador.py`

`bola.py`

Conteúdo

Estruturando os módulos

Iniciaremos os módulos com a importação da biblioteca Pygame:

```
import pygame as pg
```

Em seguida, moveremos as linhas de códigos já elaborados no `brick_breaker.py`

Finalizando com uma função de retorno para cada um dos módulos.

Movendo códigos para os seus módulos

O módulo `jogador.py` irá conter:

```
class Jogador:

  def __init__(self, cor, tamanho_barra, velocidade):
    pass

  def cria_jogador(self, altura):
    pass

  def move(self, x, largura):
    pass
```

Movendo códigos para os seus módulos

O módulo `bola.py` irá conter:

```
class Bola:

  def __init__(self,
               tamanho_bola,
               cor,
               velocidade_x,
               velocidade_y
             ):
    pass

  def cria_bola(self, largura, altura):
    pass
```

Conteúdo

Ajustando o módulo principal

Vamos adaptar o código desenvolvido até o momento para estabelecer comunicação com os módulos já criados. Devemos:

- Importar os módulos criados
- Apontar variáveis para o método criado

Ajustando o módulo principal

```
# SETUP
from jogador import Jogador
from bola import Bola
```

Conteúdo

Posição da bola

Utilizaremos a função `random.randint()`, que tem o papel de gerar um número inteiro aleatório, dentro de um intervalo específico. Esse número está sendo usado para definir a posição da bola.

Posição da bola - bola.py

```
# SETUP
import random

# FUNÇÃO
# def cria_bola(self, largura, altura):
  self.posicao = (
    random.randint(0, largura-self.tamanho_bola),
    altura*3/4
  )
```

Posição da bola - bola.py

```
def __init__(self,
             tamanho_bola,
             cor,
             velocidade_x,
             velocidade_y
           ):
  self.velocidade_x = velocidade_x
  self.velocidade_y = velocidade_y
  self.velocidade = [ velocidade_x, -velocidade_y]
  self.contador = 0
```

Movimento da bola - bola.py

```
def move(self):
    self.bola.x += self.velocidade[0]
    self.bola.y += self.velocidade[1]
```

Ajustando o módulo principal

```
# SETUP

bola = Bola(15, "white", 1, 1)
```

Ajustando o módulo principal

```
# LOOP
# while True:
    bola.move()
```

Conteúdo

Colisão da bola com a parede

Fazer uma função que verifica se a bola colidiu com as bordas da tela (paredes laterais e superior) e, ao detectar colisão, inverte a direção do movimento da bola, simulando o rebote.

- Inverte a direção horizontal se bater nas laterais.
- Inverte a direção vertical se bater no topo da tela.

Colisão da bola com a parede - bola.py

```
def confere_colisao_parede(self, largura, altura):
  if self.bola.x <= self.tamanho_bola :
        self.velocidade[0] = self.velocidade_x
```

Colisão da bola com a parede - bola.py

```
elif self.bola.x >= largura-self.tamanho_bola:
    self.velocidade[0] = -self.velocidade_x

if  self.bola.y <= self.tamanho_bola:
    self.velocidade[1] =self.velocidade_y
```

Colisão da bola com a parede - bola.py

```
def move(self, largura, altura):
  #...
  self.confere_colisao_parede(largura, altura)
```

Ajustando o módulo principal

```
# LOOP
# while True:
  #...
  bola.move(largura, altura)
```

Conteúdo

Colisão da bola com o jogador

Faremos uma função que verifica se a bola colidiu com o jogador e, caso isso aconteça, aplica regras de rebote e aceleração para tornar o jogo mais dinâmico.

- Detecta colisão entre a bola e o jogador.
- Aumenta a velocidade da bola a cada 3 colisões, até um limite.
- Inverte a direção vertical da bola para simular o rebote.

Colisão da bola com o jogador - bola.py

```
def confere_colisao_jogador(self, jogador):
   if jogador.jogador.collidepoint(self.bola.x,
                                    self.bola.y
                                   ):
      self.contador += 1

      if(self.contador == 3 and
         self.velocidade_y < 50
       ):
         self.contador = 0
         self.velocidade_y += 1
         self.velocidade_x += 1
```

Colisão da bola com o jogador - bola.py

```
            if(self.velocidade[0] > 0):
                self.velocidade[0] = self.velocidade_x

            elif(self.velocidade[0] < 0):
        self.velocidade[0] =-self.velocidade_x

self.velocidade[1] = -self.velocidade_y
```

Colisão da bola com o jogador - bola.py

```
def move(self, jogador, largura, altura):
  #...
  self.confere_colisao_jogador(jogador)
```

Conteúdo

Criando o bloco de colisão

Uma das atividades que precisamos implementar é a criação dos blocos de colisão do jogo. Para isso, será necessário definir:

- Posição
- Tamanho
- Característica
- Comportamento

Criando o módulo Bloco - bloco.py

```
# SETUP
import pygame as pg
import random
```

Criando o módulo Bloco - bloco.py

```
class Bloco:
    def __init__(self,  cor):
        self.cor = cor
        self.bloco = None
    def cria_bloco(self,
                   posicao_x,
                   posicao_y,
                   largura,
                   altura):
        self.bloco = pg.Rect(posicao_x,
                             posicao_y,
                             largura,
                             altura)
```

Criando o bloco de colisão - brick_breaker.py

```
# SETUP
bloco=Bloco("green")
bloco.cria_bloco(100, 100, 50, 20)
```

Criando o bloco de colisão - brick_breaker.py

```
# LOOP
# while True:
    desenha_elementos(tela, jogador, bola, bloco)
```

Criando o bloco de colisão - brick_breaker.py

```
def desenha_elementos(tela, jogador, bola, bloco):
  #...
  pg.draw.rect(tela, bloco.cor, bloco.bloco)
```