Introdução ao Pygame

Criando o jogo da Cobrinha I

Python para Todos

CEFET-MG

Conteúdo

Criando a tela

- Os valores do tamanho de largura e altura de nossa tela
- Criar uma variável para o nosso clock
- Definir o nosso loop principal do jogo
  - Todo jogo é um loop infinito que atualiza as informações na tela a cada momento. Cada movimentação ou interação nos controles e personagens será registrado, alterado e atualizado na tela.
  - Além disso, dentro loop que terá nossa função de parada, isto é, quando o jogo for fechado

Criando a tela

```
import pygame as pg

# SETUP
pg.init()

largura, altura = 720, 720
tela = pg.display.set_mode((largura, altura))
clock = pg.time.Clock()
```

Criando a tela

```
# LOOP PRINCIPAL
while True:

  clock.tick(60)

  for event in pg.event.get():
    if event.type == pg.QUIT:
      pg.quit()

  pg.display.update()
```

Criando a tela

Conteúdo

Adicionando a cobra

Agora, devemos criar a nossa cobra. Para isso, devemos:

- Definir o tamanho dos quadrados que serão usados para a cobra (e, posteriormente, para a maçã)
- Criar um vetor de duas dimensões para definir a posição da cobra a cada ciclo
- Desenhar a cobra na tela (inicialmente, ela será apenas um quadrado)

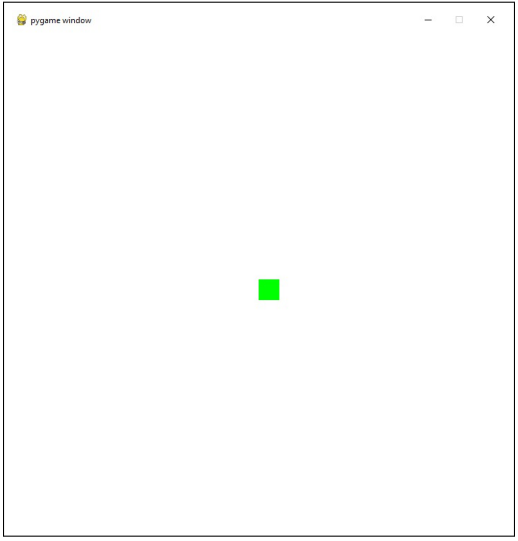
Adicionando a cobra

```
# SETUP
#...

tam = 30

cobra_pos = pg.Vector2(
  tela.get_width()/2,
  tela.get_height()/2
)
```

Adicionando a cobra

```
# LOOP PRINCIPAL
while True:
  #...

  tela.fill('white')

  cobra = pg.draw.rect(
    tela,
    'green',
    (cobra_pos.x, cobra_pos.y, tam, tam)
  )
```

Adicionando a cobra



Conteúdo

Adicionando movimentação na cobra

Precisamos fazer a nossa cobra se movimentar no mapa.

- Precisamos aumentar ou diminuir a coordenada referente ao eixo x caso a cobra se movimente para direita ou esquerda
- Precisamos aumentar ou diminuir a coordenada referente ao eixo y caso a cobra se movimente para baixo ou para cima

Adicionando movimentação na cobra

```
# SETUP
#...

velocidade = 5
```

Adicionando movimentação na cobra

```
# LOOP PRINCIPAL
  #...

  key = pg.key.get_pressed()
  if key[pg.K_w]:
    cobra_pos.y -= velocidade
  if key[pg.K_s]:
    cobra_pos.y += velocidade
  if key[pg.K_a]:
    cobra_pos.x -= velocidade
  if key[pg.K_d]:
    cobra_pos.x += velocidade
```

Conteúdo

Criando a maçã e sua colisão

Agora, precisamos criar a maçã e fazer com que, toda vez que a cobra encostar nela, ela desaparecer e aparecer em outro lugar

- Precisamos criar um vetor que define o movimento da maçã, desenhar a maçã na tela e criar o evento de colisão para isso
- Toda vez que encostar na maçã, um valor aleatório (limitado pelo tamanho da tela) será colocado na posição da maçã

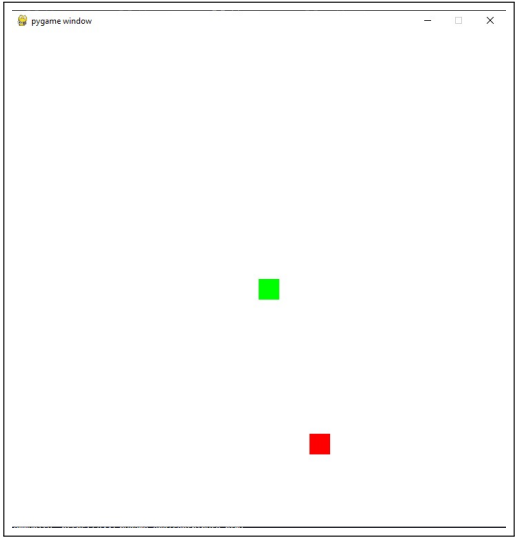
Criando a maçã e sua colisão

```
# SETUP
#...

import random as rd

maca_pos = pg.Vector2(
    rd.randint(tam, largura-tam),
    rd.randint(tam, altura-tam)
)
```

Criando a maçã e sua colisão

```
# LOOP PRINCIPAL
#...
  maca = pg.draw.rect(
    tela,
    "red",
    (maca_pos.x, maca_pos.y, tam, tam)
  )

  if maca.colliderect(cobra):
    maca_pos.x=rd.randint(tam, largura-tam)
    maca_pos.y=rd.randint(tam, altura-tam)
```

Criando a maçã e sua colisão



Conteúdo

Atualização da movimentação

Queremos que a cobra se movimente sozinho para uma direção com apenas um aperto no botão de movimento

- Para isso, devemos atualizar nosso movimento, colocando um vetor auxiliar para aceitar aumentar/diminuir ou o eixo x, ou o eixo y
- Depois disso, adicionamos esse valor no vetor responsável pelo movimento da cobra

Atualização da movimentação

```
# SETUP
#...

control_pos = pg.Vector2(velocidade,0)
```

Atualização da movimentação

```
# LOOP PRINCIPAL
#...
  key = pg.key.get_pressed()
  if key[pg.K_w]:
    control_pos.y = -velocidade
    control_pos.x = 0
  if key[pg.K_s]:
    control_pos.y = velocidade
    control_pos.x = 0
    #...
```

Atualização da movimentação

```
#...
  if key[pg.K_a]:
    control_pos.y = 0
    control_pos.x = -velocidade
  if key[pg.K_d]:
    control_pos.y = 0
    control_pos.x = velocidade

  cobra_pos.x += control_pos.x
  cobra_pos.y += control_pos.y
```

Conteúdo

Montando o corpo da cobra

Por fim, vamos montar o corpo da cobra.

- O corpo da cobra é composto pelas posições que a cabeça esteve anteriormente, isto é, na última atualização de tela.
- Como perdemos essas informações toda vez que a tela é atualizada, precisamos armazenar as coordenadas de cada "cabeça" em uma lista e, depois, criar uma lista de cabeças, montando, assim, o corpo.

Montando o corpo da cobra

- Criaremos uma lista do corpo, que guardará listas das coordenadas da cabeça da atualização anterior ($[x_1, y_1], [x_2, y_2], [x_3, y_3], \ldots$)
- Uma lista das coordenadas, que guarda a coordenada das cabeças ($[x, y]$)
- Um loop que irá acessar cada conjunto de coordenadas da lista do corpo e criar o corpo

Montando o corpo da cobra

```
# SETUP
#...

lista_cobra = []
```

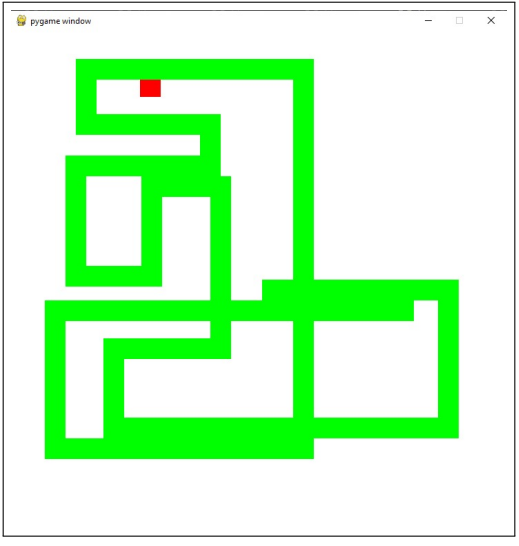
Montando o corpo da cobra

```
# LOOP PRINCIPAL
#...
    lista_cabeca = []
    lista_cabeca.append(cobra_pos.x)
    lista_cabeca.append(cobra_pos.y)
    lista_cobra.append(lista_cabeca)
```

Montando o corpo da cobra

```
#...

for pos in lista_cobra:
  pg.draw.rect(
    tela,
    'green',
    (pos[0], pos[1], tam, tam)
  )
```

Montando o corpo da cobra



Montando o corpo da cobra II

Como podemos ver, a cobra tem um corpo infinito. Devemos limitar o tamanho da cobra. Criaremos, então:

- Uma variável que define quantos “quadrados” a cobra terá de tamanho
- Uma condição que compara esse tamanho com o tamanho atual da lista do corpo. Caso seja maior, deletaremos da lista a primeira posição, dando a sensação de movimento.

Montando o corpo da cobra II

```
# SETUP
#...

tamanho_cobra = 15
```

Montando o corpo da cobra II

```
# LOOP PRINCIPAL
#...
  if len(lista_cobra) > tamanho_cobra:
    del lista_cobra[0]
```

Montando o corpo da cobra II