Pygame

RODRIGO LYRA

Pygame

Pygame é um conjunto de módulos Python projetados para escrever videogames. Isso permite criar jogos e programas multimídia com todos os recursos na linguagem Python.

```
pip install pygame
```

A pygame, além do import, precisa ter seus módulos inicializados com a função .init().

```
import pygame
pygame.init()
```

Pygame

A tela do jogo é resgatada pela função display.set_mode(), recebendo uma lista ou tupla de dois elementos, sendo as dimensões da tela.

```
screen = pygame.display.set_mode([500, 500])
```

Para manter a tela ativa, tudo precisa estar dentro de um laço de repetição, o gameloop:

```
running = True
while running:
```

Pygame

O Pygame trabalha com eventos, que podem ser acessados por event.get(). Um desses eventos, por exemplo, é o clique no botão de fechar da janela.

```
for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
```

Pygame

A cor base da tela de fundo pode ser modificada, com a função fill na variável da tela, ela espera uma tupla de 3 elementos, sendo o R, G e B. O exemplo abaixo pinta a janela de branco.

```
screen.fill((255, 255, 255))
```

Podemos também desenhar objetos na tela, utilizando funções do módulo draw. O código abaixo desenha um círculo.

```
pygame.draw.circle(screen, (0, 0, 255), (250, 250), 75)
```

Pygame

A tela não é atualizada a cada comando, ela é atualizada toda de uma vez quando o comando .flip é chamada

```
pygame.display.flip()
```

Lembramos sempre de, fora do loop, chamar a função quit para finalizar a execução da Pygame.

```
pygame.quit()
```

Pygame – Fazer um jogo

Vamos fazer um jogo:

- O objetivo do jogo é evitar obstáculos;
- O jogador inicia no lado esquerdo da tela;
- Os obstáculos entram aleatoriamente a partir da direita e movem-se para a esquerda em uma linha reta;
- O jogador pode se mover para a esquerda, direita, para cima ou para baixo para evitar os obstáculos;
- O jogador não pode sair da tela;
- O jogo termina quando o jogador é atingido por um obstáculo ou quando o usuário fecha a janela;

Pygame – Inicializar a tela

Criamos a tela:

```
tela_largura = 800
tela_altura = 600

tela = pygame.display.set_mode((tela_largura, tela_altura))
```

Pygame – GameLoop

Precisamos sempre criar um gameloop, todo jogo trabalha com ele, o jogo é algo dinâmico que é sempre atualizado, ele cuida de alguns aspectos básicos:

- Processa a entrada do usuário
- Atualiza o estado de todos os objetos do jogo
- Atualiza o vídeo e áudio
- Mantém a velocidade do jogo

Cada ciclo do ciclo do jogo é chamado de frame, e quanto mais rápido você puder fazer as coisas a cada ciclo, mais rápido o seu jogo será executado.

Pygame – Controle de Entrada

Teclas, movimentos do mouse e até movimentos do joystick são algumas das maneiras pelas quais um usuário pode fornecer informações para o jogo.

Todas as entradas do usuário resultam em um evento. Os eventos podem ocorrer a qualquer momento e geralmente (mas nem sempre) se originam fora do programa. Todos os eventos no pygame são colocados na fila de eventos, que podem ser acessados e manipulados. Lidar com eventos é chamado de manipulação, e o código para isso é chamado de manipulador de eventos.

Pygame – Controle de Entrada

Todo evento no pygame possui um tipo de evento associado. Para o nosso caso, os tipos de eventos nos quais a gente vai se concentrar são as teclas pressionadas e o fechamento da janela. Os eventos de pressionamento de tecla têm o tipo de evento KEYDOWN e o evento de fechamento da janela tem o tipo QUIT. Diferentes tipos de eventos também podem ter outros dados associados a eles. Por exemplo, o tipo de evento KEYDOWN também possui uma variável chamada key para indicar qual tecla foi pressionada.

Acessamos a lista de todos os eventos ativos na fila chamando pygame.event.get. Em seguida, percorremos esta lista, inspecionando cada tipo de evento e responde de acordo.

Pygame – Controle de Entrada

O código abaixo verifica se o botão de fechar da janela foi pressionado ou a tecla Esc foi pressionada.

```
running = True

while running:
  for event in pygame.event.get():
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    running = False
            elif event.type == pygame.QUIT:

                running = False
```

Pygame – Controle de Entrada

A tela é um objeto do tipo surface, e podemos preencher com uma cor e desenhar nele, mas também podemos criar um objeto a parte.

```
imagem = pygame.Surface((50, 50))
imagem.fill((0, 255, 0))
```

Pygame – Controle de Entrada

Podemos adicionar uma surface em outra a partir do comando blit()

```
tela.blit(imagem, (tela_largura/2, tela_altura/2))
```

Lembrando que para atualizar os conteúdos da tela, precisamos utilizar o comando flip:

```
pygame.display.flip()
```

Pygame – Capturar teclas

Além da maneira de captar todos os eventos vista anteriormente, existe uma forma de retornar diretamente um dicionários com as teclas pressionadas:

```
teclas_pressionadas = pygame.key.get_pressed()
```

Com isso podemos, por exemplo, modificar a posição da imagem de fundo a cada loop:

```
if teclas_pressionadas[pygame.K_UP]:
            posicao_j_h -= 1
        if teclas_pressionadas[pygame.K_DOWN]:
            posicao_j_h += 1
        if teclas_pressionadas[pygame.K_LEFT]:
            posicao_j_w -= 1
        if teclas_pressionadas[pygame.K_RIGHT]:
            posicao_j_w += 1
```

Pygame – Inimigos

Precisamos de inimigos, para isso utilizaremos a biblioteca do python para gerar valores aleatórios:

```
import random
```

Com isso podemos criar inimigos em posições aleatórias, com velocidades aleatórias:

```
if len(inimigos) < 10:
        inimigos.append([0, random.randint(0,
tela_altura), random.randint(1, 5)])
```

Pygame – Inimigos

Precisamos também atualizar estes inimigos, o desenho, a posição e destruir caso tenham saído da tela:

```
for inimigo in inimigos:
        inimigo[0] += inimigo[2] / 10
        imagemInimigo = pygame.Surface([10, 5])
        imagemInimigo.fill((255, 255, 255))
        tela.blit(imagemInimigo, (inimigo[0], inimigo[1]))

        if inimigo[0] > tela_largura:
            inimigos.remove(inimigo)
```

Pygame – Detectar colisões

Nem sempre a detecção de colisões é uma matemática trivial, mas o pygame facilita isso, dentro da classe Sprite, como ainda não vimos orientação a objeto em python, faremos uma pequena gambiarra aqui:

```
spJ = pygame.sprite.Sprite()
spJ.rect = pygame.Rect(posicao_j_w, posicao_j_h, jogador.get_width(), jogador.get_height())
```

E colocaremos isso dentro de cada laço do teste dos inimigos:

```
spI = pygame.sprite.Sprite()
spI.rect = pygame.Rect(inimigo[0], inimigo[1], 10, 5)
 if pygame.sprite.collide_rect(spJ, spI):
            print("O jogo acabou")
```

Pygame – Imagens

Podemos também carregar imagens para dentro do pygame, e adicionar diretamente para uma surface, vamos criar o fundo:

```
fundo = pygame.image.load("fundoTop.jpg").convert()
```

Vamos desenhar esse fundo diretamente :

```
tela.blit(fundo, (0, 0))
```

Pygame – Imagens

Podemos também carregar imagens para dentro do pygame, e adicionar diretamente para uma surface, vamos criar o fundo:

```
jogador = pygame.image.load("aviaoTop.png").convert()
jogador.set_colorkey((0,0,0))
```