Alvaro Leonel Socorro Queiroz Mariano e Rodolpho Pivetta Sabino

Explicação da Implementação

Flappy Bird

Tópicos em Computação I

Ciência da Computação – UEMS

Inicialização de tela

Para a inicialização da tela, obtemos as configurações de vídeo do sistema operacional, de modo que geramos a tela de jogo em *full screen*. Para tal, utilizamos as seguintes funções:

```
1.   videoInfo  = pygame.display.Info()
2.   resolution = (videoInfo.current_w, videoInfo.current_h)
3.   screen = pygame.display.set_mode(resolution,
pygame.FULLSCREEN)
4.   pygame.display.set_caption("Flappy Bird 1.0")
```

Na linha 1, é obtida a informação de configuração de tela, com tamanho de altura e largura onde na terceira linha é setado a resolução em *full screen* e a linha 4 seta o nome da tela como "Flappy Bird 1.0".

Inicialização de variáveis

A inicialização de variáveis feitas na função principal (main) são:

```
1.   terrain = [resolution[0] * 2, int(resolution[1] * 0.3)]
2.   skye = [resolution[0] * 2, resolution[1] - terrain[1]]
3.   bird = Bird([50, 50], [100, 100], 5, 40)
4.   ground = Ground(terrain)
5.   background = Background(skye)
6.   options = menu.menu(screen, resolution, ground, background,
bird)
7.   pipe = Pipe(80, options[1], options[2], options[3], skye)
8.   score = scoreClass.Score(screen)
9.   clock = pygame.time.Clock()
10.  exit = options[0]
11.  done = False
12.  hit = False
13.  delayJump = 0
14.  jumped = False
```

De acordo com as variáveis apresentadas acima, são apresentadas as tarefas que cada uma realiza no jogo a seguir:

- terrain: seta a área de terreno do jogo;
- skye: área disponível para movimentação do pássaro;

- bird: cria o objeto pássaro;
- ground: cria o objeto terreno;
- background: cria o objeto fundo;
- options: opções do menu (*easy, medium, hard*);
- pipe: cria um objeto cano;
- score: cria o objeto score (número de pontuação que aparece na tela);
- clock: relógio do jogo;
- exit: controla o fim do jogo;
- done: controla o fim da partida;
- hit: controla colisão;
- delayJump: controla o tempo de *delay* entre cada pulo do pássaro;
- jumped: controla a última ação do pássaro (último pulo).

Criação de imagens e formas na tela

As funções que criam os objetos (funções __init__ de cada classe objeto) realizam as importações de imagens (disponibilizadas nas pastas da raiz do código), sons e fonte para escrita.

Objetos do jogo

Os objetos do jogo são:

- Pássaro (bird)
- Canos (pipes)
- Plano de fundo (background)
- Chão (ground)

Como o jogo foi feito com orientação à objetos, as classes foram criadas de acordo com os nomes supracitados (em inglês).

Movimentos do jogo (laço principal)

O laço principal do jogo é realizado pelo código a seguir:

```
1.   while not exit:
2.        if not done and not hit:
3.             screen.fill(0)
4.             for event in pygame.event.get():
5.                  if event.type == pygame.QUIT:
```

```
6.                        pygame.quit()
7.                        sys.exit()
8.
9.                  if event.type == pygame.KEYDOWN:
10.                       if event.key == pygame.K_UP and jumped ==
False:
11.                            Jumped = True
12.                            delayJump = pygame.time.get_ticks()
13.                            bird.jump()
14.                       if event.key == pygame.K_ESCAPE:
15.                            done = True
16.
17.                 if event.type == pygame.KEYUP:
18.                       if event.key == pygame.K_UP:
19.                            jumped = False
20.            if pygame.time.get_ticks() - delayJump > 200:
21.                 bird.fall()
22.
23.            ground.move(5)
24.            background.move(5)
25.            pipe.move(5)
26.
27.            background.draw(screen, 0)
28.            ground.draw(screen, resolution[1])
29.            pipe.draw(screen)
30.            bird.draw(screen)
31.            score.draw([resolution[0] / 2, 10])
32.
33.            pygame.display.flip()
34.
35.            hit = collision.check([resolution[0], resolution[1]
- int(resolution[1] * 0.3)], bird, pipe, score)
36.
37.            clock.tick(60)
38.
39.       else:
40.            if menu.gameOver(screen, resolution, score, ground,
background, bird):
41.                 options = menu.menu(screen, resolution, ground,
background, bird)
42.                 del(pipe)
43.                 pipe = Pipe(80, options[1], options[2],
options[3], skye)
44.                 done = options[0]
45.                 hit = options[0]
46.                 bird.setPosition([100,100])
47.                 score.resetScore()
48.            else:
49.                 exit = True
```

O laço acima realiza iterações enquanto o usuário não solicita a saída do jogo. Para cada

iteração, é capturada a tecla de seta para cima como pulo do pássaro, chamando a função de pulo do pássaro. As linhas de 23 a 25 movem os elementos (objetos) na tela, as linhas de 27 a 31 imprimem os elementos de acordo com cada objeto e a linha 35 verifica se houve a colisão do pássaro com teto, chão ou canos. As linhas de 40 a 49 executam caso o pássaro tenha colidido com algum objeto, onde caso o usuário deseja reiniciar o jogo, os canos são excluídos (linha 42), os canos são reconstruídos (linha 43) e as variáveis reiniciadas (linhas de 44 a 47). Caso o usuário não deseja reiniciar o jogo, a variável exit é setada como verdadeiro, saindo do laço principal e finalizando o jogo.

Tratamento de colisões

O tratamento de colisões do pássaro com o chão e o teto é feito pela função à seguir:

```
1.   def topBottom(bird, resolution):
2.        if bird.Position[1] < 0:
3.             return True
4.        elif bird.Position[1] + bird.Size[1] > resolution[1]:
5.             return True
6.        else:
7.             return False
```

Nesta função, é verificado se o objeto pássaro colidiu com o teto (linhas 2 e 3), se colidiu com o chão (linhas 4 e 5) ou se não colidiu (linhas 6 e 7).

Para o tratamento de colisões do objeto pássaro com os tubos, é utilizada a função à seguir:

```
1.   def birdPipes(bird, pipes, score):
2.        hit = False
3.        counter = 0
4.        for pipe in pipes.top:
5.             if bird.Position[0] + bird.Size[0] - tolerance >
pipe[1] and bird.Position[0] < pipe[1] + pipe[0].get_width() and
bird.Position[1] < pipe[0].get_height():
6.                  hit = True
7.             if bird.Position[0] > pipe[1] + pipe[0].get_width()
and not pipe[3]:
8.                  score.point()
9.                  pipes.setOverpast(counter)
10.            counter += 1
11.
12.       for pipe in pipes.bottom:
13.            if bird.Position[0] + bird.Size[0] - tolerance >
pipe[1] and bird.Position[0] < pipe[1] + pipe[0].get_width() and
bird.Position[1] + bird.Size[1] > pipe[2]:
14.                 hit = True
15.
16.       return hit
```

Nesta função, as linhas de código de 3 a 10 percorre todos os canos superiores verificando se

o pássaro ultrapassou completamente ou não um cano, somando um ponto ou retornando a colisão. O mesmo ocorre nas linhas de 12 a 14, porém com os canos inferiores.

É verificado se, de acordo com o tamanho do pássaro e o limite do cano os valores se encontram e ultrapassam, concluindo ou não a travessia do pássaro. Caso o pássaro ultrapasse o cano superior, é verificado se ele colidiu com o cano inferior, caso nenhum dos dois tratamentos é verdadeiro (ou seja, o pássaro não colidiu com nenhum dos canos), a função soma 1 ponto ao total de pontos e seta verdadeiro à variável hit que é retornada posteriormente. Caso tenha havido alguma colisão, o código não seta verdadeiro à variável hit, sendo assim é retornado o valor setado no inicio da função (linha 2) o valor falso.

Esta função é chamada em todas as iterações da função *main* (laço principal do jogo) onde é verificado o retorno desta, caso seja retornado o valor falso, na função principal é parado o laço do jogo e apresentada a tela de *Game Over*.