PET computação

Introdução ao Pygame

Caio Paes, Demontiê Junior, Tales Pimentel

UFCG

SUPRA OMNES LUX LUCES

UFCG CEEI Departamento de Sistemas e Computação

Eventos do *mouse*

- pygame.mouse.get_pos()
  - Retorna uma tupla (x, y) referente à posição
- pygame.mouse.get_pressed()
  - Retorna uma tupla (button1, button2, button3)
  - Cada campo da tupla é um *boolean*
  - Ex.: pygame.mouse.get_pressed()[0]

    → Indica se o botão esquerdo foi pressionado

Eventos do teclado

- Podemos capturar as teclas pressionadas de duas formas:
  - Usando o pygame.event.get():

```
# Capturando eventos do taclado a partir
#do pygame.event.get()
for event in pygame.event.get():
    if event.type == KEYDOWN: # Ou KEYUP
        if event.key == K_ESCAPE:
            ...
```

Eventos do teclado

- Ou usando o pygame.key.get_pressed(), que retorna uma lista com *booleans*:

```
# Capturando eventos do taclado
# a partir do  pygame.key.get_pressed()

pressed_keys = pygame.key.get_pressed()

if pressed_keys[K_ESCAPE]:
    ...
```

- As constantes (K_ESCAPE, K_f, etc) representam o índice referente a cada tecla nessas listas

Algumas teclas

Constante	Nome da tecla
K_ESCAPE	“Esc”
K_RETURN	*Enter*
K_BACKSPACE	*Backspace*
K_SPACE	Barra de espaço
K_a, K_b, ..., K_z	Teclas de A a Z
K_LEFT	Seta para esquerda
K_RIGHT	Seta para direita
K_UP	Seta para cima
K_DOWN	Seta para baixo
K_0, K_1, ..., K_9	Teclas de 1 a 9
K_LSHIFT, K_RSHIFT	*Shift* esquerdo e direito
K_LCTRL, K_RCTRL	“ctrl” esquerdo e direito
K_LALT, K_RALT	“Alt” esquerdo e direito

Onde encontrar?

- Documentação do Pygame
  - Mouse: http://www.pygame.org/docs/ref/mouse.html
  - Teclado: http://www.pygame.org/docs/ref/key.html

Principais Módulos

- Image – Manipulação de imagens do sistema
- Mixer.Sound – Sons simples, efeitos sonoros
- Mixer.Music – Player de músicas
- Sprite – Objetos de colisão, ex: personagens
- Time – Manipulação do tempo no jogo
- Font – Criar textos e renderizar em imagens

Image

- Funções para lidar com as imagens externas ao Pygame.
- Pygame.image.load(filename)
  - Carrega uma imagem do sistema para o jogo
    - Formatos suportados: BMP, TGA, GIF (não animado), JPEG, PNG, TGA, dentre outras

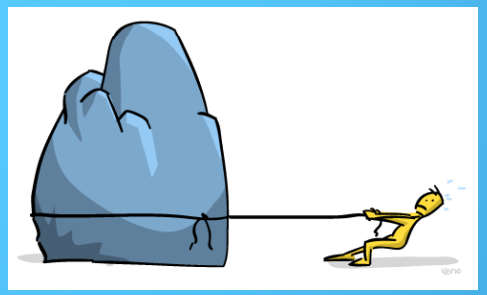
```
# Carrega uma imagem para o Pygame.
# OBS: Carrega, NÃO blita ela na tela!
img_pedra = pygame.image.load("images"+os.sep
                              \\+"pedra.gif").convert()

img_boneco = pygame.image.load("images"+os.sep+"objetos"
                           \\+ "garrafa.png").convert_alpha()
```

Image

- Para que os códigos anteriores funcionem corretamente, veja a organização dos arquivos:

```
programa.py
images/
        pedra.gif
        objetos/
                garrafa.png
```



Image

- Pygame.image.save(Surface, filename)
  - Salva uma imagem carregada no Pygame (Surface) como um arquivo de imagem no sistema.
  - Formatos suportados:
    - BMP, TGA, PNG, JPEG

```
# Salvando uma imagem no sistema
# Neste pontos existe uma imagem na variável img_desenho
Pygame.image.save(img_desenho, “images”+os.sep+”desenhos”
                                        \\+os.sep+”desenho1.jpg”)
```

Mixer

- É possível manter vários sons tocando ao mesmo tempo
- Parar um ou todos os sons de uma vez
- pygame.mixer.stop()
  - Para a execução de todos os sons
- Pygame.mixer.unpause()
  - Recomeça a execução dos sons parados

Sound

- pygame.mixer.Sound(filename)
  - Retorna um objeto Sound, que pode ser executado e parado quando você quiser
  - Formatos suportados: OGG e WAV (Descompressado)

```
# Carrega uma som de colisão para o jogo.
som_colisao = pygame.mixer.Sound("sounds"+os.sep+"toc1.wav")
…
if acontece a colisao:
    som_colisao.play(1) # inteiro é o numero de repetições

som_colisao.set_volume(0.7) # 0 <= volume  <= 1.0
```

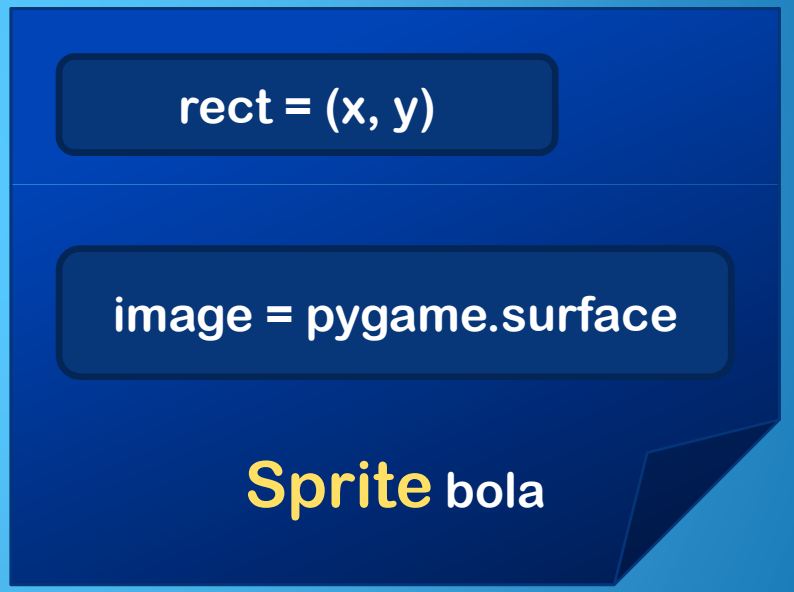
Music

- É interno ao módulo Mixer
- Executa músicas durante o jogo.
- Pode executar apenas uma música por vez.
- Formatos suportados:
  - MP3 e OGG

```
# Carrega uma música ambiente para o jogo.
musica =
pygame.mixer.music.load("musics"+os.sep+"jazz_and_blues1.mp3
")
musica.play(-1) # Toca a música infinitamente
musica.set_volume(0.9) # 0 <= volume <= 1.0
Musica.fadeout(3000)  # Diminui o volume de acordo com o
tempo em milisegundos
```

Sprite

- Módulo próprio para os objetos, personagens, e cenário do jogo
- Um Sprite, é a forma de representar um item do jogo. Possui uma posição (na tela) e uma imagem
- As fuções do módulo sprite lidam com objetos Sprite()
- Pygame.sprite.Sprite()
  - Sprite.rect
  - Sprite.image



Sprite

- Várias funcões do módulo sprite são para detecção de colisão.
- Pygame.sprite.collide_mask(sprite_a, sprite_b)
  - Retorna um bool indicando se as imagens dos dois sprites estão se sobrepondo (colidindo)

```
# Verifica se dois Sprites estão colidindo.
# Neste ponto devem existir dois sprites, bola e parede.
if pygame.sprite.collide_mask(bola, parede):
    som_colisao.play(1)
    # mude a direção da bola ou
    # faça o jogador perder uma vida e recomece o jogo
```

Sprite

- pygame.sprite.collide_rect(sprite_a, sprite_b)
  - Detecção de colisão entre dois sprites, usando rect (posicionamento e dimensões)
- Pygame.sprite.collide_circle(sprite_a, sprite_b)
  - Detecção de colisão usando áreas circulares
- Outras fuções do módulo sprite, permitem outros tipos de verificação (por grupos de sprites, por camada)

Time

- Módulo responsável pela informação sobre o tempo no jogo.
- Muito útil em jogos que possuem movimentos e física
  - Regula a movimentação dos objetos na tela
  - Evita diferença de execução entre máquinas diferentes

Time

```
# Cria um relógio
clock = pygame.time.Clock()
…
x , y = (10, 5)
vel_x, vel_y = 7, 3
…
While True:
    delta_tempo =
clock.tick()
    …
    x += vel_x * delta_tempo
    y += vel_y * delta_tempo

# S = S0 + v * t
```

- pygame.time.Clock()
  - Cria um relógio que serve para se obter a variação de tempo ao longo do programa
  - Clock.tick()
    - Retorna o tempo, em milisegundos, desde a última chamada deste método
    - Deve ser uma vez por loop

Font

- Permite renderizar TrueType Fonts (*.TTF) em imagens para o jogo
- Permite a utilização de fontes extras, especiais
- pygame.font.Font(filename)
  - Carrega uma fonte do sistema
  - Formato suportado: TTF

```
# Carrega uma fonte colocada na pasta pessoal game_fonts
fonte_jogo =
pygame.font.Font("game_fonts"+os.sep+"showgothic_card.ttf")
```

Font

- Font.render(texto, antialias, cor)
  - Renderiza (Interpreta a fonte e converte em uma imagem) o texto na cor desejada
  - O retorno é uma imagem do Pygame (Surface)

```
…
tela = pygame.display.set_mode((800, 600), 0, 32)
# Carrega uma fonte colocada na pasta pessoal game_fonts
fonte_jogo =
pygame.font.Font("game_fonts"+os.sep+"showgothic_card.ttf")

tela.blit(fonte_jogo.render("Que bom!", True, (255, 0, 0)),
                                                \\ (200,
150))
```


PET
computação
pygame
Dúvidas?
UFCG
SUPRA OMNES LUX LUCES
{caiocm, demontie, tales}@dsc.ufcg.edu.br
UFCG CEEI Departamento de Sistemas e Computação

Referências

- http://www.pygame.org/
- Beginning Game Development with Python and Pygame, Will McGugan – Apress 2007.