Desenvolvendo Jogos com pygame

Rafael Guterres Jeffman

2019

Por que jogos?

Desenvolver Jogos

- É divertido.
- Tu sempre quis fazer.
- Foi a primeira coisa que tu fez com algo que parecia um computador.
- Não precisa ser difícil.
- Não é fácil.
- Tu quer mostrar pra todo mundo que tu consegue desenvolver um jogo.

É muito divertido!

Por que Python?

Por que Python?

- É divertido.
- Permite que a preocupação seja o problema.
- Faz com que tu aprenda uma linguagem que está sendo muito utilizada.

Por que pygame?

pygame

- É multi-plataforma (SDL).
- Retira as parada chata da programação de jogos.
- Não é um engine de jogos, e queremos programar.
- É uma biblioteca de componentes.

Every saga has a beginning!

Durante um teste de rotina, a nave Genesis é trasportada através de um *wormhole* para o quadrante *gamma* da galáxia, e precisa sobreviver à *Guerra do Infinito*.

O que era só um dia de testes virou uma luta pela sobrevivência.

Na vida nada se cria...

- Gradius/Nemesis
- Farscape
- Star Trek: Voyager
- Um filme muito, muito ruim...

Hello World

```
import pygame
pygame.init()
# inicia tela
screen = pygame.display.set_mode((320,200))
pygame.display.set_caption("Hello World!")
# loop principal
running = True
while running:
    # trata eventos
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
    # atualiza objetos
    # desenha objetos
    # pygame usa double buffer!
    pygame.display.update()
```

A janela da aplicação

- Aplicações pygame podem usar o modo janela ou *fullscreen*.
- No modo *fullscreen*, o tamanho da janela é a sua resolução.

```
width, height = size = (800, 600)
flags = pygame.FULLSCREEN | pygame.HWSURFACE | pygame.DOUBLEBUF
screen = pygame.display.set_mode(size, flags)
```

Desenhando na tela

- A estrutura criada pela função `display.set_mode` é uma superfície, que utilizamos para desenhar na tela.
- Esta estrutura pode ser utilizada com o módulo `pygame.draw`.

```
python.draw.circle(screen, red, (100,100), 50)
python.draw.polygon(screen, white, point_list)
python.draw.rect(screen, white, (x, y, rect_w, rect_h))
screen.blit(image, (x, y))
```

Um campo de estrelas

- Todo jogo de plataforma que se preze usa o efeito de *paralaxe*.
- Um campo de estrelas com três planos pode ser criado com círculos que se movem com velocidades diferentes.
- *List comprehensions* e *tuplas* são muito úteis para isso.

Um campo de estrelas - Criação

```
def create_star(x):
    y = randint(0, height)
    speed = choice([1, 2, 3])
    magnitude = choice([1, 2, 3])
    color = (coice(100, 200, 250),) * 3
    return (x, y, speed, magnitude, color)

stars = [create_star(randint(0, width)) for star in range(count)]
```

Um campo de estrelas - Movimentação

```
stars = [[x - speed, y, speed, mag, color]
         if x - speed > 0
         else create_star(width)
         for x, y, speed, mag, color in stars]
```

Sprites

- Sprites são imagens 2D, mas nos jogos, eles tem movimento.
- O uso de sprites facilita a definição dos objetos móveis do jogo.
- Sprites, normalmente, tem suporte a transparência.

Sprites com animação

- pygame suporta imagens GIF, mas sem animação.
- pygame oferece diversos plugins que podem ser utilizados.
- Obviamente, existe um plugin para GIF animado.
- Mais óbvio ainda... ele estava abandonado e não funcionava...

Tratamento de Eventos

- pygame oferece um sistema de eventos por *polling*.
- Para criar um *engine* com um *loop* genérico, é preciso permitir que o código cliente seja chamado de volta.
- Um mecanismo desses permite que funções cliente sejam chamadas para eventos do pygame.
- *E para felicidade geral da nação...*
  funções são cidadãs de primeira ordem*!*

O loop de eventos

```
# loop genérico, em Game.run()
while self.running:
    # handle events
    for event in pygame.event.get():
        handle_event(event)
```

Respondendo a eventos de teclado

```
# my code
def move(event):
    """Move player with directional keys."""
    keys = pygame.key.get_pressed()
    dx, dy = 0, 0
    dy = -1 if keys[pygame.K_UP] else 0
    dy = dy + 1 if keys[pygame.K_DOWN] else dy
    dx = -1 if keys[pygame.K_LEFT] else 0
    dx = dx + 1 if keys[pygame.K_RIGHT] else dx
    player.move = (dx * config.speed, dy * config.speed)

# Configuring the game object
game.on_key((pygame.K_UP, pygame.K_DOWN,
             pygame.K_LEFT, pygame.K_RIGHT), move)
```

E o que mais falta fazer?

Um jogo tem tanta coisa...

- Tratamento de colisões.
- Comportamento de NPCs.
- Trocas de fazes.
- Cenários.

Sem audio?

pygame Mixer!

- pygame tem um mixer que, sem configuração, suporta 8 canais de audio.
- Suporte a loops de áudio já embutido.
- Suporte a diversos formatos de áudio.
- Ogg Vorbis é a melhor opção.
- E a internet está cheia de loops *royalty free*...

Quão produtivo é o pygame?

Você tem 24h para...

- Aprender a usar o pygame.
- Escrever uma demonstração.
- Criar os *slides* de uma palestra.
- Apresentar a palastra.

Quão produtivo é o pygame?

E ainda por cima...

- Ministrar três horas de aula.
- Dirigir de Porto Alegre a Pelotas (4h).

- Passear com os cachorros!

E DEU CERTO!

E agora?
Pra onde vou?

https://python.org

https://pygame.org

Próximos Passos

- Desenvolver um engine para criação de jogos 2D!
- Para ensinar programação orientada a objetos com Python.
- Para ensinar *design* de jogos.
- Para ensinar criação de roteiros de jogos.

E para criar jogos, né?

Na verdade...

Esse nunca foi o objetivo...

Mas nem uma demonstração?

https://github.com/rafasgj/genesis

Muito Obrigado!

mailto:rafasgj@gmail.com

https://slides.tchelinux.org