


Desenvolvendo Jogos com pygame

Rafael Guterres Jeffman

2019

The background of the slide features several pixelated, 3D block-like figures in various colors (green, orange, yellow, pink, cyan) standing on a white surface. The figures are stylized, resembling characters from a video game. The central figure is a pink, blocky creature with a large head and a small body. Other figures are scattered in the background, some in focus and some blurred.

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!

The background features several pixelated, 3D block-like characters in various colors (orange, yellow, pink, teal) standing on a light gray surface. The characters are stylized, resembling a mix of animals and abstract shapes. The text "Por que Python?" is overlaid in the center in a bold, magenta font.

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.



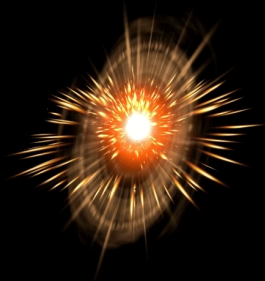
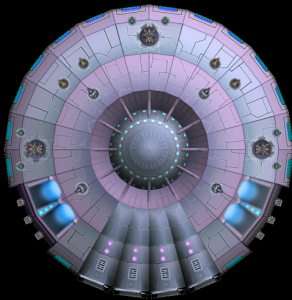
Por que
pygame?

pygame

- É multi-plataforma (SDL).
- É uma biblioteca de componentes.
- Retira *as paradas chatas* da programação de jogos.
- Não é um engine de jogos, afinal, queremos programar.



Genesis



Every saga has a beginning!

Durante um teste de rotina, a nave Genesis é transportada 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...

Modelo de Aplicação pygame

```
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()
```

O mínimo que voce precisa saber...

```
pygame.init()
```

```
width, height = size = (800, 600)
```

```
flags = pygame.FULLSCREEN | pygame.HWSURFACE | pygame.DOUBLEBUF
```

```
screen = pygame.display.set_mode(size, flags)
```

Desenhando na tela

```
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 universo a estrelar...

```
from random import randint, choice

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

List Comprehension

- É uma construção de Python que permite o processamento de uma lista de elementos.

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

Operador Ternário

- O operador ternário seleciona um entre dois valores, dada uma condição.
- Vai muito bem com *guacamole* e *list comprehension*.

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


Desenhando a lista de estrelas

- Os *list comprehension* estão entre as estruturas mais eficientes para processar listas.

```
[python.draw.circle(screen, color, (x, y), mag)  
  for x, y, _, 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.
- É possível utilizar eventos para animar *sprites*.
- pygame oferece diversos plugins que podem ser utilizados.
- Obviamente, existe um plugin para GIF animado (que não funciona).



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

```
def handle_event(event):  
    if event.type in [pygame.KEYDOWN]:  
        keydown(event)  
  
def keydown(event):  
    for handler in keydown_handlers:  
        handler(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_keydown((pygame.K_UP, pygame.K_DOWN,  
                  pygame.K_LEFT, pygame.K_RIGHT), move)  
game.on_keyup((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*...

A series of pixelated, 3D block-like figures in various colors (green, orange, yellow, red, pink, cyan) are arranged in a line, receding into the background. The figures are constructed from simple rectangular blocks, giving them a retro, digital appearance. The central figure is a pink one, and the text is overlaid on it.

E agora?
Pra onde vou?

The background features several pixelated, 3D block-like figures in various colors (brown, yellow, pink, teal) arranged in a row, slightly out of focus. The figures are composed of small cubes and have simple rectangular cutouts for eyes and mouths.

<https://python.org>

<https://pygame.org>

Próximos Passos

The background of the slide features several pixelated, 3D block-like characters. In the foreground, a large pink character is prominent. Behind it and to the sides are other characters in shades of orange, yellow, and cyan. They are all rendered in a low-poly, voxel-like style.

- 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.

The background of the slide features several pixelated, 3D blocky characters in various colors (orange, yellow, pink, teal) standing on a light gray surface. The characters are rendered in a low-poly, voxel-like style. The text "E para criar jogos, né?" is overlaid in the center in a bold, magenta font.

E para criar jogos, né?



Na verdade...

Esse nunca foi o objetivo...

The background of the slide features several pixelated, 3D block-like figures in various colors (green, orange, yellow, red, pink, cyan) standing on a white surface. The figures are stylized, resembling characters from a retro video game. The central figure is a pink one, and it is the largest and most prominent. Other figures are scattered in the background, some slightly out of focus.

Mas nem uma demonstração?

The background of the slide features several pixelated, 3D block-like models of animals. In the foreground, a pink pig-like creature is prominent. Behind it and to the sides are other similar creatures in orange, yellow, green, and light blue. They are all rendered in a low-poly, pixelated style with visible block edges and some hollowed-out sections.

<https://github.com/rafasgj/genesis>

The background features several pixelated, 3D block-like characters. In the foreground, a large pink character is prominent. Behind it, there are other characters in orange, yellow, and light blue, all rendered in a low-poly, pixelated style. The scene is set against a light, neutral background.

Muito Obrigado!

`mailto:rafasgj@gmail.com`
`https://rafaeljeffman.com/tchelinux`