Live de Python # 158

Roteiro

1. Display
O básico necessário

2. Image
Como trabalhar com imagens

3. Sprites
Fazendo as coisas ficarem legais

4. Eventos
Lendo inputs de mouse e teclado

5. Colisão
Fazendo sprites interagirem

6. Texto
Escrevendo coisas legais na tela

picpay.me/dunossauro

apoia.se/livedepython

PIX

Ajude o projeto

Ademar Peixoto, Alex Lima, Alexandre Harano, Alexandre Santos, Alexandre Souza, Alexandre Tsuno, Alysson Oliveira, Amaziles Carvalho, Anderson Araujo, Andre Rodrigues, André Rocha, Antonio Neto, Arnaldo Turque, Bianca Rosa, Bruno Oliveira, Caio Nascimento, Carlos Cardoso, Carlos Chiarelli, César Almeida, César Moreira, Davi Ramos, David Kwast, Diego Guimarães, Dilenon Delfino, Douglas Bastos, Edgard Sampaio, Elias Soares, Eugenio Mazzini, Everton Alves, Fabio Barros, Fabio Castro, Fabrício Coelho, Flavkaze Flavkaze, Franklin Silva, Fábio Serrão, Gabriel Simonetto, Gabriel Soares, Gabriela Santiago, Geandreson Costa, Guilherme Felitti, Guilherme Marson, Guilherme Ostrock, Haroldo Júnior, Henrique Machado, Hélio Neto, Isaac Ferreira, Israel Fabiano, Italo Silva, Jeison Sanches, Johnny Tardin, Jonatas Leon, Jorge Plautz, José Prado, Jovan Costa, João Lugão, João Schiavon, Juan Gutierrez, Jônatas Silva, Júlia Kastrup, Kaneson Alves, Leonardo Cruz, Leonardo Galani, Leonardo Mello, Lorena Ribeiro, Lucas Barros, Lucas Ferreira, Lucas Mello, Lucas Mendes, Lucas Teixeira, Lucas Valino, Luiz Lima, Maiquel Leonel, Maiquel Leonel, Marcela Campos, Marcelo Rodrigues, Maria Clara, Natan Cervinski, Nicolas Teodosio, Nilo Pereira, Otavio Carneiro, Patric Lacouth, Patricia Minamizawa, Patrick Gomes, Paulo Tadei, Pedro Andrade, Pedro Pereira, Peterson Santos, Reinaldo Silva, Rodrigo Campos, Rodrigo Ferreira, Rodrigo Vaccari, Ronaldo Silva, Rubens Gianfaldoni, Sandro Mio, Silvio Xm, Thiago Araujo, Thiago Borges, Thiago Bueno, Tyrone Damasceno, Valdir Junior, Victor Geraldo, Vinícius Bastos, Vinícius Ferreira, Wesley Mendes, Willian Lopes, Willian Lopes, Willian Rosa, Wilson Duarte

Obrigado você

py game

Uma apresentação inicial

Pygame

Pygame é uma biblioteca baseada em SDL (openGL) feita para criar jogos.

Ela é:

- Software Livre (LGPL)
- Portável
  - windows, linux, macOS, xxxBSD
- 20 anos de história (e bugs corrigidos)

pip install pygame

https://www.pygame.org/wiki/GettingStarted#Pygame%20Installation

Bora instalar?

python -m pygame.examples.aliens

Agora que você já instalou

Display

O básico necessário

Display

O display no pygame é responsável por controlar a janela (window) e a tela (screen) de maneira geral.

```
1   from pygame import display
2
3   superficie = display.set_mode()
4   display.set_caption('texto da janela')
```

Superfície

set_mode nos retorna a superfície do jogo. Porém, como usamos até agora, ele nos gerou uma window com a resolução da nossa tela.

```
1  from pygame import display
2
3  superficie = display.set_mode(
4    size=(largura, altura),
5    flags=0,    # Opções adicionais
6    depth=0,    # Opções de cores (0 adivinha o sistema de cores
7    display=0   # Diz em qual tela o jogo será aberto
8    vsync=0     # Habilita o vsync (sincronização vertical)
9  )
```

update

Toda vez que mudamos alguma configuração da superfície, devemos atualizar a mesma usando a função `update()`

```
1  from pygame import display
2
3  display.set_mode((800, 600))
4  display.set_mode((1280, 720))
5  display.update()
```

Image

Trabalhando com imagens

Trabalhando com imagens

Agora que sabemos o básico sobre a superfície, temos que colocar nossas imagens na tela. O pygame conta com dois módulos para nos ajudar com imagens:

- pygame.image
  - Faz o carregamento de imagens no disco
  - Salva imagens no disco
  - ...
- pygame.transform
  - Executa operações em imagens (bitmap)
  - Zoom, redimensionamento, rotação, ...

Trabalhando com imagens

A primeira coisa que precisamos entender é a usar o load. Para carregar a imagem dentro do pygame

```
1   from pygame.image import load
2
3   fundo = load('images/space.jpg')
```

Colocando a imagem no jogo

Todas as coisas foram carregadas agora.

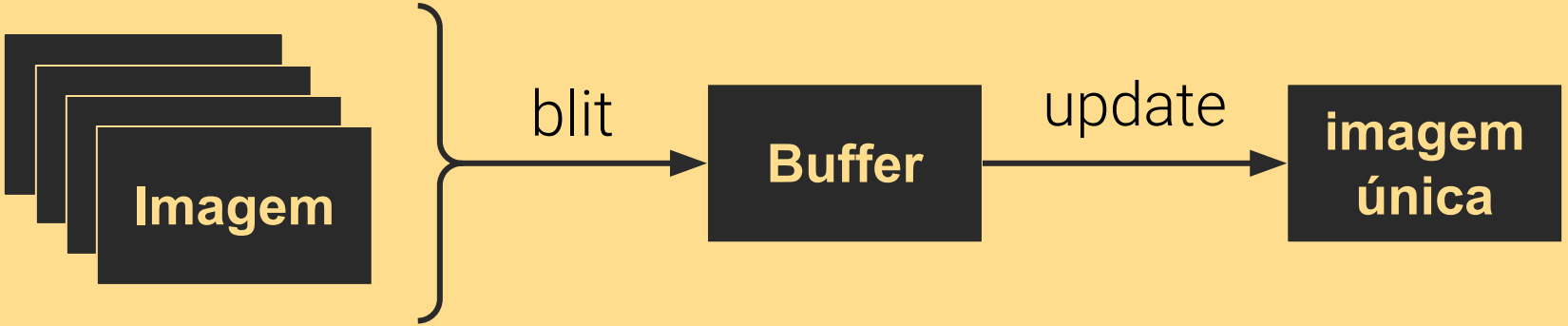
```
1  from pygame import display
2  from pygame.image import load
3
4  superficie = display.set_mode((1280, 720))
5  fundo = load('images/space.jpg')
6
7  display.update()
```

Bit blit

Agora que temos tudo carregado, precisamos entender o conceito de **bit blit**

Bit Blit

A operação de blit coloca uma imagem no buffer, mas o que esse buffer significa? O pygame faz uma junção de tudo que está no buffer para enviar ao display uma única imagem.

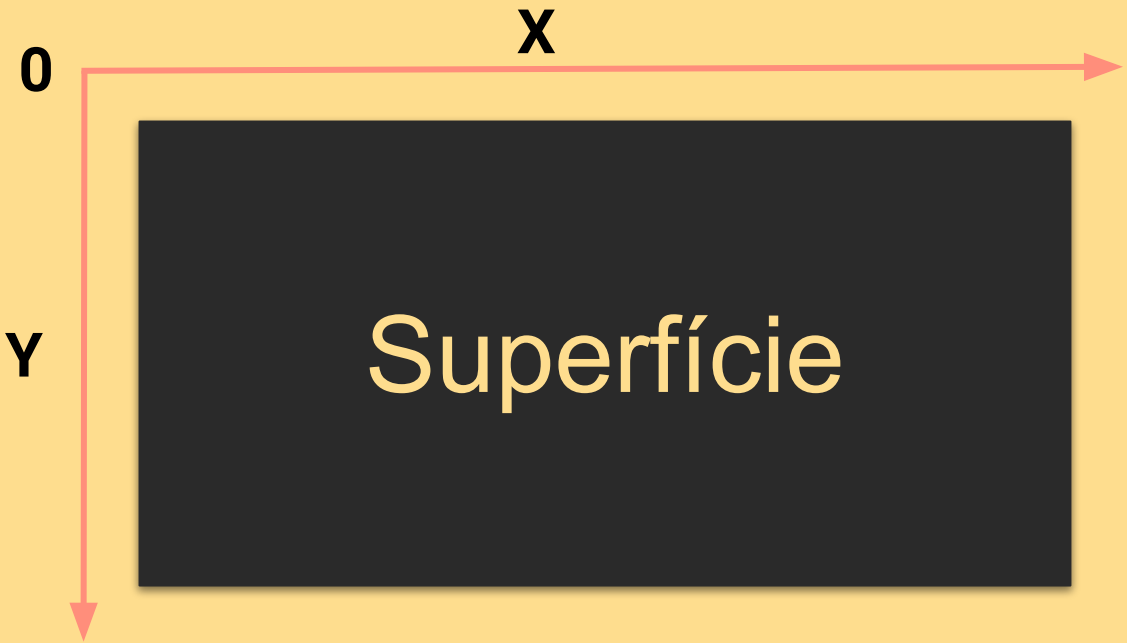


Como usar o blit?

Blit é uma função da superfície. Ele recebe dois argumentos, o que vai ser "blitado" e as coordenadas onde isso vai aparecer.

```
1   from pygame import display
2   from pygame.image import load
3
4   superficie = display.set_mode((1280, 720))
5   fundo = load('images/space.jpg')
6
7   superficie.blit(
8     fundo,  # Imagem
9     (0, 0)  # Posição
10  )
11
12  display.update()
```

Sobre a posição



Tudo junto, agora

```python
from pygame import display
from pygame.image import load

superficie = display.set_mode((1280, 720))
fundo = load('images/space.jpg')

superficie.blit(
  fundo,  # Imagem
  (0, 0)  # Posição
)

display.update()
```

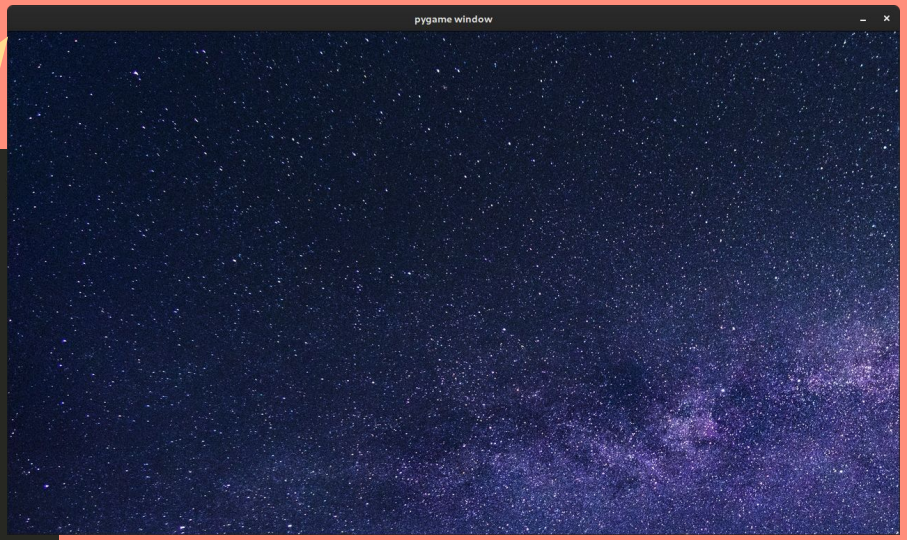
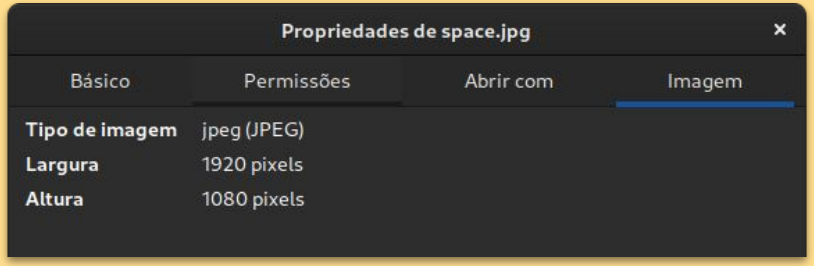
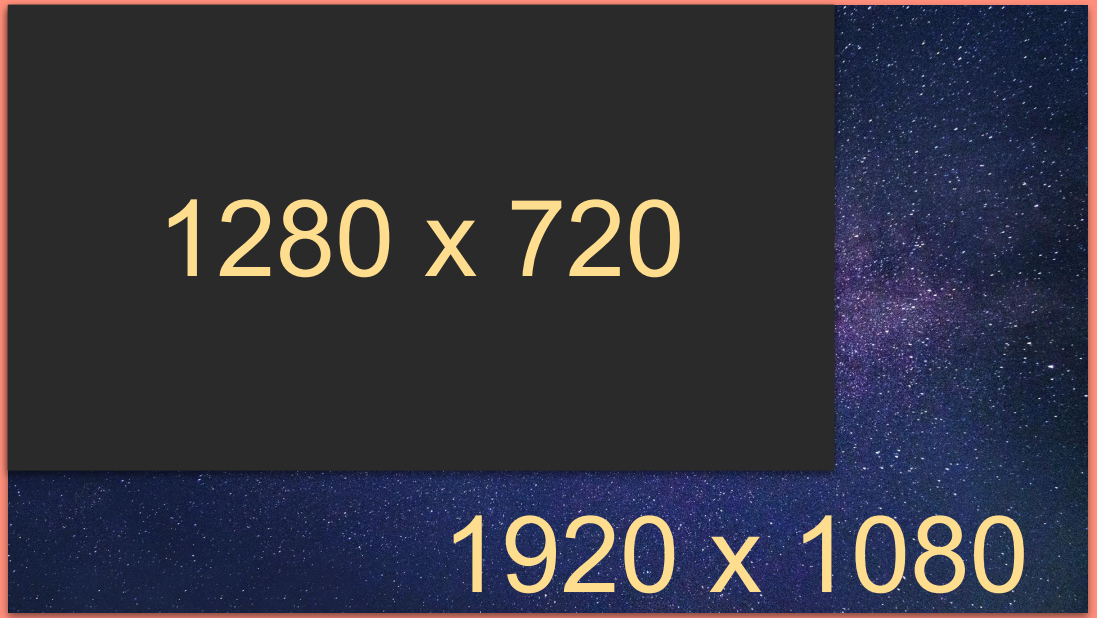


Imagem não está completa :(

Olhando as propriedades da imagem, podemos ver que ela é um pouco maior que a nossa superfície. Então vamos ter que trabalhar um pouco nessa imagem antes



Entendendo o problema



Escalonando a imagem

```python
from pygame import display
from pygame.image import load
from pygame.transform import scale

tamanho_da_tela = (1280, 720)
superficie = display.set_mode(tamanho_da_tela)

fundo = scale(
  load('images/space.jpg'),  # Imagem carregada
  tamanho_da_tela    # Tupla do tamanho escalonado
)

superficie.blit(
  fundo,  # Imagem
  (0, 0)  # Posição
)

display.update()
```

Outra operações de transformação

- transform.rotate(image, rotação)
  - Rotaciona a imagem em x graus
- transform.rotozoom(imagem, rotação, escala_de_zoom)
  - Rotaciona a imagem em x graus e redimenciona em x vezes
- transform.scale2x(imagem)
  - redimenciona a imagem em 2x
- ...

https://www.pygame.org/docs/ref/transform.html

Fazendo as coisas ficarem mais legais

Sprites

Sprites

Sprites, diferentes de imagens "paradas" são imagens que podem se movimentar e alternar entre imagens. É ideal para criação de personagens, inimigos e etc...

O pygame nos fornece uma classe que podemos estender, para criar as interações de um sprinte.

A classe Sprite

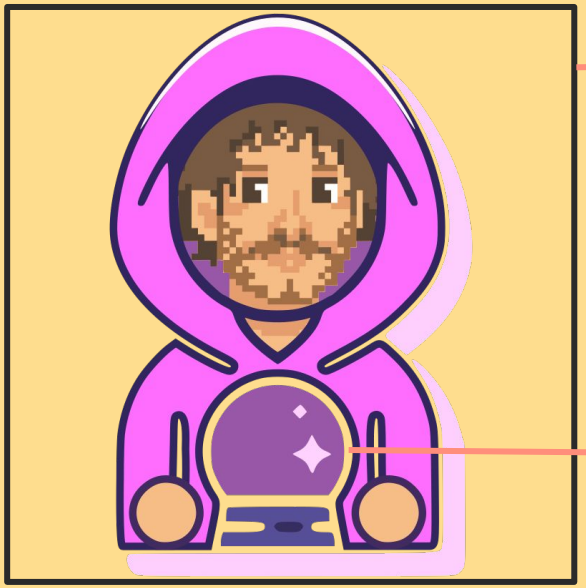
```python
from pygame.sprite import Sprite

class MeuSprite(Sprite):
  def __init__(self, torradas):
    super().__init__()  # Herança

    self.image = load('')              # Necessário
    self.rect = self.image.get_rect()  # Necessário

  def update(self):
    self.kill()  # Destroi o sprite
```

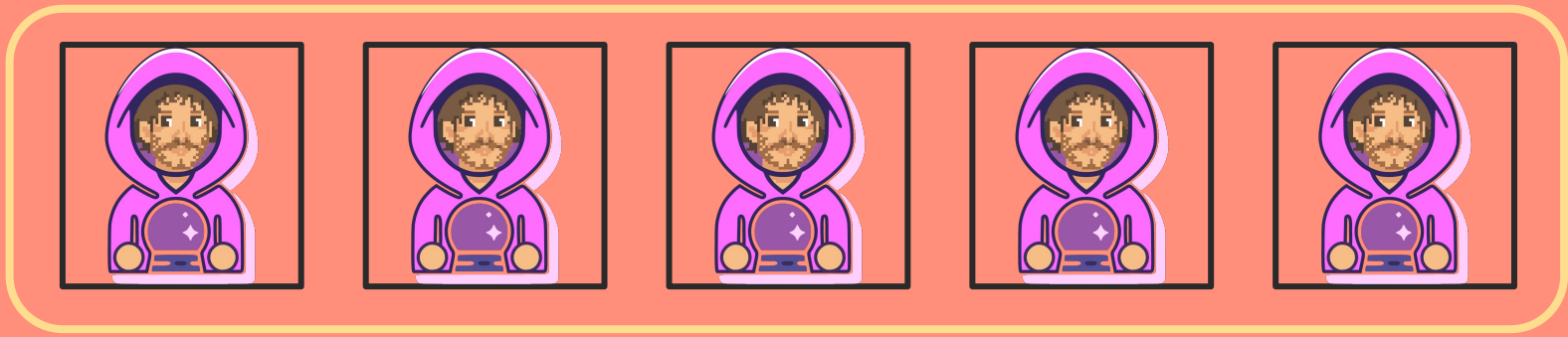
Como funciona um sprite?



Retangulo

Imagem

Grupos de sprites



Os grupos são formas de unir sprites e auxiliar na atualização de todo o grupo chamando o `update()` do grupo todo.

Mesmo que seu grupo tenha somente um sprite, o pygame recomenda que você use um grupo

Criando grupos

As classes de grupos são super simples de utilizar, é só importar e adicionar no grupo os sprites que fazem parte deles:

```
1  from pygame.sprite import Group, GroupSingle
2
3  grupo_de_inimigos = Group()
4  grupo_de_inimigos.add(Inimigo())
5  grupo_de_inimigos.add(Inimigo())
6
7  grupo_do_heroi = GroupSingle()  # Grupo de um sprite
8  grupo_do_heroi.add(Heroi())
```

Grupos e o loop de eventos

Agora que temos o grupo de sprites definidos, podemos fazer os grupos serem desenhados `draw()` e fazer o `update()` de todos e maneira uniforme.

Com isso, todos os sprites terão interação.

```
while True:
    # Desenhando o Rect de cada sprite
    grupo_de_inimigos.draw(superficie)
    grupo_do_heroi.draw(superficie)

    # Chamando o update de cada sprite
    grupo_de_inimigos.update()
    grupo_do_heroi.update()

    display.update()
```

E o jogo, cadê?

blah blah blah

Lendo inputs de mouse e teclado

Eventos

Eventos

Eventos são "acontecimentos" de quando interagimos com o jogo. Seja usando o mouse ou o teclado (voz, quem sabe???)

Precisamos notificar nossos sprites quando algo acontece. Por exemplo, quando apertar para direita, o sprite de fato deve ir para direita.

Eventos

O pygame tem 3 modos de capturar essa interação.

- key
  - captura teclas pressionadas
- mouse
  - captura posição e clicks
- event
  - Mouse, teclado e mais coisas

Vamos começar

O evento mais simples que podemos fazer é o evento de sair do jogo. Reparou que até fechamos tudo no terminal?

Vamos começar a popular nosso loop de jogo e entender um pouco mais sobre ele

O boilerplate

Todo código para o pygame tem mais ou menos essa estrutura inicial:

- Captura de eventos
- desenhos dos sprites
- update dos grupos
- update do display

```
import pygame
from pygame.locals import QUIT
from pygame import event

pygame.init()

while True:
  # Loop do jogo

  # Eventos
  for evento in event.get():
    if evento.type == QUIT
        pygame.quit()
    # outros eventos

  # Display
  superficie.blit(fundo, (0, 0))

  grupos.draw()
  grupos.update()

  display.update()
```

Entenendo event.get()

O método get() nos retorna uma lista de `Event`s.

Eventos se comportam como dicionários e tem acesso fácil as chaves.

O atributo universal de todos os eventos é `type`. E type é representado por um número.

Felizmente o pygame pensou nisso e dentro de pygame.locals podemos encontrar todos os tipos de eventos que podem ser associados

Eventos de teclado

Os eventos de teclado, são caracterizados como `KEYUP` e `KEYDOWN`, ambos códigos obtidos em pygame.locals.

Nos eventos de teclado, temos o atributo `.key` de evento que nos retorna o código da tecla digitada. Os atalhos também são encontrados em pygame.locals

Exemplificando

Estamos observando 3 eventos no momento.

QUIT, KEYUP e KEYDOWN

Caso a tecla de espaço seja pressionada teremos KEYDOWN de K_SPACE

Caso a tecla de espaço seja solta teremos KEYUP de K_SPACE

```
1 import pygame
2 from pygame.locals import (
3   QUIT, KEYUP, KEYDOWN, K_SPACE
4 )
5 from pygame import event
6
7
8 while True:
9   for evento in event.get():
10    if evento.type == QUIT:
11      pygame.quit()
12
13    if evento.type == KEYUP:
14      if evento.key == K_SPACE:
15        print('espaço solto')
16
17    if evento.type == KEYDOWN:
18      if evento.key == K_SPACE:
19        print('espaço apertado')
```

Usando Key

Outra maneira de capturar eventos de teclado é usar **pygame.key.get_key_pressed()**

Isso retornará uma um dicionário com as teclas pressionadas.

Vamos aplicar isso ao nosso sprite

Comandos no update

Uma forma simples de movimentar nosso sprite.

```
1 class Dunofausto(Sprite):
2     def __init__(self, torradas):
3         super().__init__()
4         self.image = load('images/dunofausto_small.png')
5         self.rect = self.image.get_rect(center=centro)
6         self.speed = 2
7 
8     def update(self):
9         keys = pygame.key.get_pressed()
10 
11        if keys[pygame.K_LEFT]:
12            self.rect.x -= self.speed
13        if keys[pygame.K_RIGHT]:
14            self.rect.x += self.speed
15        if keys[pygame.K_UP]:
16            self.rect.y -= self.speed
17        if keys[pygame.K_DOWN]:
18            self.rect.y += self.speed
```

Fazendo sprites interagirem

Colisão

Colisão

Colisão é quando dois sprites se colidem ou se encostam.

Na maioria dos jogos, um projétil é disparado por um sprite, por exemplo, e quando ele encontra outro sprite alguma coisa acontece.

Os tipos de colisão

O pygame nos provê 3 formas de colisão

- spritecollide(sprite, grupo, dokill)
  - retorna o grupo que foi colidido pelo sprite
  - dokill, executa o kill em todos os colididos
- spritecollideany(sprite, grupo)
  - Retona qual sprite do grupo colidiu com o Sprite
- groupcollide(grupo1, grupo, dokill1, dokill2)
  - Retorna todas as colisões entre os dois grupos
  - dokill1: executa kill no sprite gupo1
  - dokill2: executa kill no sprite gupo2

Bora implementar

Tá cheio de teoria hoje, só por deus

Escrevendo coisas legais na tela

Texto

Trabalhando com texto

Para trabalhar com texto, seguimos o mesmo estilo das imagens. Porém, precisamos carregar a fonte antes. Após isso, precisamos renderizar a escrita com a fonte e fazer o bit blit

Exemplo de texto na tela

```python
from pygame import font

minha_fonte = font.SysFont('nomedafonte', tamanho_da_fonte)

meu_texto = minha_fonte.render(
  'Meu texto',
  antialias=True,        # Arredondar as fontes
  color=(255, 255, 255)  # Tupla RGB
)

superficie.blit(meu_text, posição)
```

BORA ACABAR ESSE JOGO

Bora?

picpay.me/dunossauro

apoia.se/livedepython

PIX

Ajude o projeto