Universidade de São Paulo

EEL-USP

Depois do Big Bang

Apresentação

Lucas Campos Achcar

Módulos a ser estudado (Necessário para Programar o Pong)

- O Módulo Display
- O Módulo Rect
- O Módulo Surface
- O Módulo Draw
- O Módulo Image
- O Módulo Event, Mouse, Key
- O Módulo Font
- Programando o Pong

O módulo display

- O módulo display é o que gerencia toda a interface gráfica (janela e tela de exibição)
- O controle da janela diz respeito ao tamanho do ‘jogo’, titulo, ícone de exibição, sistema para tirar screenshot etc
- O controle da tela diz respeito a o que está sendo desenhado no buffer principal de imagem ou seja, a surface primária da tela (screen)

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

As funções principais do módulo display

- **pygame.display.set_mode() -> Surface**
  - set_mode(size=(0, 0), flags=0, depth=0, vsync=0) -> Surface

- A função set_mode cria uma surface de tela (exibição), além disso, é permitido passar alguns tipos de argumentos para melhorar o tipo de exibição na dependência do hardware.

- Propriedades:
  - **size=(largura, altura)** usado para definir o tamanho da tela
  - **flags=0** - Define as propriedades da janela
  - **depth=0** - Define das propriedades da cor (mapeamento de cor), se definido como 0 o pygame irá usar o melhor mapeamento de cor possível
  - **vsync=0** (sincroniza vertical) - força a GPU a manter a taxa de quadros igual ao da tela

Podem ser usadas mais de uma flag por vez, basta usar a condição | (pipe) no momento de chamar a função, exemplo: **pygame.DOUBLEBUF | pygame.FULLSCREEN**

Figura 1 - Flags disponíveis

```
pygame.FULLSCREEN    create a fullscreen display
pygame.DOUBLEBUF     recommended for HWSURFACE or OPENGL
pygame.HWSURFACE     hardware accelerated, only in FULLSCREEN
pygame.OPENGL        create an OpenGL-renderable display
pygame.RESIZABLE     display window should be sizeable
pygame.NOFRAME       display window will have no border or controls
```

As funções principais do módulo display

- **pygame.display.quit() -> None**
  - Finaliza todos os módulos abertos pelo pygame (usado quando deseja finalizar o programa)
- **pygame.display.flip() -> None**
  - Atualiza tudo que está no buffer (screen) para a tela (para mais detalhes consulte o documento do pygame)
- **pygame.display.update() -> None**
  - Faz a mesma coisa que o flip porém, otimizada (para mais detalhes consulte o documento do pygame)
  - **OBS:** Essa função não funciona com a flag pygame.OPENGL e deve ser utilizado a **pygame.display.flip()**
- **pygame.display.set_caption(title, icontitle=None) -> None**
  - Troca o texto da janela (parte superior da janela)
- **pygame.display.set_icon(Surface) -> None**
  - Troca o ícone da janela (parte superior da janela) - O argumento e uma surface
- **Existem outras funções do pygame.display que não será passado aqui por falta de tempo, porém na documentação do pygame (https://www.pygame.org/docs) você pode encontrar todas as funções e também como funciona cada uma.**

Código de Exemplo #01

```python
import pygame

#define o tamanho da tela
screenSize = (800, 600)
# cria a tela e salva a instância dessa tela em screen
screen = pygame.display.set_mode(screenSize, pygame.DOUBLEBUF | pygame.FULLSCREEN)
pygame.display.set_caption("Exemplo#01")

# cria uma instância do time.Clock() - vamos usar para limitar o fps
gameClock = pygame.time.Clock()

# cria uma variavel que verifica se o jogo ainda está rodando
gameRunning = True

# verifica a cada frame se o jogo está rodando
while gameRunning:
    # limita o FPS em 60 quadros por segundo
    gameClock.tick(60)

    # verifica os eventos que estão na pool de eventos
    for event in pygame.event.get():
        # verifica se o X (da janela) foi pressionado, se sim, finalziada o jogo (gameRunning = False)
        if(event.type == pygame.QUIT):
            gameRunning = False
        # verifica se uma tecla foi pressionada
        if(event.type == pygame.KEYDOWN):
            # verifica se a tecla é o ESC, se sim, finaliza o jogo (gameRunning = False)
            if(event.key == pygame.K_ESCAPE):
                gameRunning = False

    # depois que você definiu o que desenhar, faça a atualização da tela (chamamos essa parte de double-buffer)
    # o double-buffer evita flicks na tela
    pygame.display.update()

# finaliza todos os módulos que foram iniciados
pygame.quit()
```

As funções principais do módulo Rect

- O módulo Rect é responsável por agrupar a posição e o tamanho de qualquer objeto do jogo em uma única tupla com 4 parâmetros

  Veja abaixo 3 funções alternativas do Rect (Os três são equivalentes)

- **Rect(x, y, width, height)**
- **Rect((left, top), (width, height)) -> Rect**
- **Rect(object) -> Rect**
  - x representa a posição x do objeto na tela
  - y representa a posição y do objeto na tela
  - width representa a largura do objeto
  - height representa a altura do objeto
- **Rect.colliderect(Rect) -> bool**
  - Verifica se ocorreu uma colisão entre dois Rect, se sim, retorna true

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

As funções principais do módulo Surface

- O módulo Surface é responsável pela manipulação dos pixels na tela
- Essa função tem bastante coisa, portanto irei falar das principais (para mais detalhe consulte o manual)
- Primeiro, qualquer imagem no seu jogo é uma surface (inclusive o buffer principal ou seja, a tela principal), com isso, qualquer função que irei passar aqui é possível usar com uma imagem pré-carregada

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

As funções principais do módulo Surface

- **blit(source, dest, area=None, special_flags=0) -> Rect**
  - Copia uma surface de um destino para uma origem, o retorno é um Rect contendo a área dos pixels afetados incluindo qualquer pixel que esteja fora do destino ou fora da área de corte definida pelo parâmetro ‘area’
- **copy() -> Surface**
  - Faz a cópia de uma surface ou seja, duplica a surface desejada
- **fill(color, rect=None, special_flags=0) -> Rect**
  - Preenche a surface com uma cor determinada (utilizado para limpar o buffer principal)
- **scroll(dx=0, dy=0) -> None**
  - Move a imagem dx pixels para direita ou dy pixels para baixo (dx e dy podem ser negativos invertendo o sentido do scroll)
- **get_size() -> (width, height)**
  - Função que retorna uma tupla contendo a largura e a altura respectivamente
- **get_height() -> height**
  - Função que retorna apenas a altura da imagem
- **get_width() -> width**
  - Função que retorna apenas a largura da imagem

O módulo draw

- O módulo draw é responsável por desenhar primitivas na tela

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

As funções principais do módulo draw

- **pygame.draw.rect(surface, color, rect) -> Rect**
  - Desenha um retângulo na tela, é necessário colocar onde você deseja desenhar o retângulo (surface), a cor principal e Rect define (x, y, largura, altura)
- **pygame.draw.polygon(surface, color, points) -> Rect**
  - Desenha um polígono baseado em pontos pré-definidos por você, points é uma lista de pontos e cada ponto é uma tupla (x, y)
- **pygame.draw.circle(surface, color, center, radius) -> Rect**
  - Desenha um círculo na tela, center é uma tupla (x, y) que representa o centro e radius é o raio do círculo
- **pygame.draw.ellipse(surface, color, rect) -> Rect**
  - Desenha uma elipse na tela, rect representa o tamanho de sua elipse, lembrando, a elipse possui um lado maior e outro menor que nada mais é o tamanho do seu rect (Rect(x, y, largura, altura)), sendo a = largura e b = altura
- **pygame.draw.arc(surface, color, rect, start_angle, stop_angle) - Rect**
  - Desenha um arco de circunferência, rect definine o retângulo que conterá o arco e o start_angle e stop_angle o ângulo inicial e final
- **pygame.draw.line(surface, color, start_pos, end_pos, width) - Rect**
  - Desenha uma linha, os argumentos principais são, start_pos e end_pos que define o ponto inicial e final (tupla, ou seja, (x, y)) e width que seria a largura da linha
- **Como mencionado, existem mais algumas comandos que são variações do que já foi demonstrado acima.**

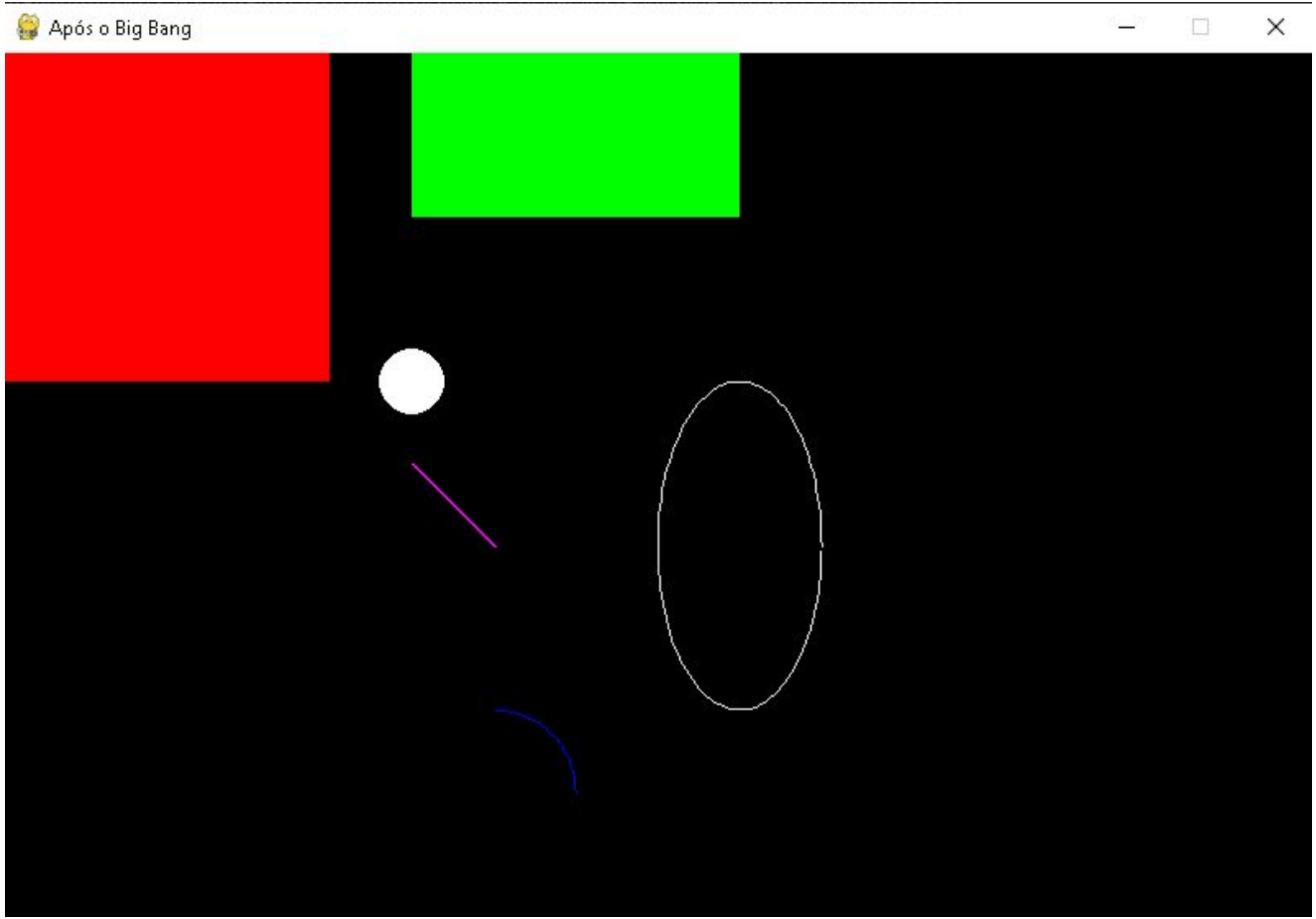
Código de Exemplo #02

```python
import pygame
import math

#define o tamanho da tela
screenSize = (800, 600)
# cria a tela e salva a instância dessa tela em screen
screen = pygame.display.set_mode(screenSize, pygame.DOUBLEBUF)
pygame.display.set_caption("Após o Big Bang")

# cria uma instância do time.Clock() - vamos usar para limitar o fps
gameClock = pygame.time.Clock()

# cria uma variavel que verifica se o jogo ainda está rodando
gameRunning = True

# verifica a cada frame se o jogo está rodando
while gameRunning:
    # limita o FPS em 60 quadros por segundo
    gameClock.tick(60)

    # verifica os eventos que estão na pool de eventos
    for event in pygame.event.get():
        # verifica se o X (da janela) foi pressionado, se sim, finalziada o jogo (gameRunning = False)
        if(event.type == pygame.QUIT):
            gameRunning = False
        # verifica se uma tecla foi pressionada
        if(event.type == pygame.KEYDOWN):
            # verifica se a tecla é o ESC, se sim, finaliza o jogo (gameRunning = False)
            if(event.key == pygame.K_ESCAPE):
                gameRunning = False

    # cria um retângulo vermelho na coordenada x = 0 e y = 0 com largura = 200 e altura = 200
    pygame.draw.rect(screen, (255, 0, 0), (0, 0, 200, 200))
    # cria um retângulo verde na coordenada x = 250 e y = 0 com largura = 200 e altura = 100
    pygame.draw.rect(screen, (0, 255, 0), (250, 0, 200, 100))
    # cria um circulo de raio 20 pixels na coordenada x = 250 e y = 200 com a cor branco
    pygame.draw.circle(screen, (255, 255, 255), (250, 200), 20)
```

Código de Exemplo #02

```
38      # cria um arco de elipse de 0 até (PI / 2) com a = 100 e b = 100 (circuferência) na posição x = 250 e y = 400
39      pygame.draw.arc(screen, (0, 0, 255), (250, 400, 100, 100), 0, math.pi / 2)
40      # cria uma elipse branca (0 até 2*PI) com a = 100 e b = 200 na posição x = 250 e y = 400
41      pygame.draw.arc(screen, (255, 255, 255), (400, 200, 100, 200), 0, 2*math.pi)
42      # cria uma roxa com inicio no ponto (250, 250) e com ponto final em (300, 300) com largura 2 da linha
43      pygame.draw.line(screen, (255, 0, 255), (250, 250), (300, 300), 2)
44
45      # depois que você definiu o que desenhar, faça a atualização da tela (chamamos essa parte de double-buffer)
46      # o double-buffer evita flicks na tela
47      pygame.display.update()
48
49  # finaliza todos os módulos que foram iniciados
50  pygame.quit()
```

Código de Exemplo #02



O módulo image

- O módulo image é responsável por carregar as imagens do seu computador para a memória para manipulações das mesmas

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

As funções principais do módulo image

- **pygame.image.load_basic(file) -> Surface**
  - Faz o carregamento de uma imagem no formato BMP (file é o caminho do arquivo), o retorno é uma surface
- **pygame.image.load(filename) -> Surface**
  - Faz o carregamento de uma imagem porém, aceitando vários formatos (PNG, JPG, GIF (sem animação), BMP)
- **pygame.image.save(Surface, filename) -> None**
  - Salva uma imagem (surface) em um arquivo externo

Código de Exemplo #03

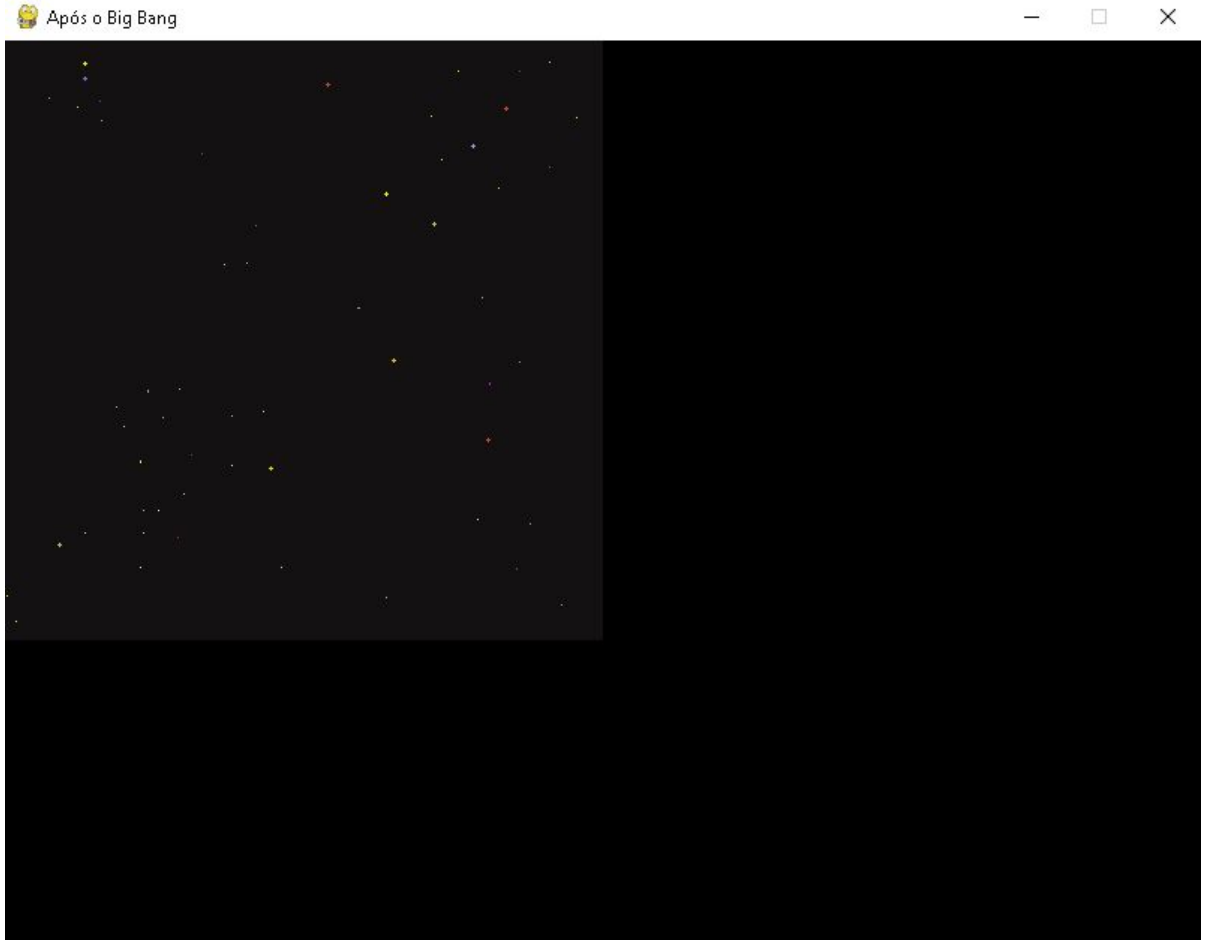
```python
import pygame
import os

#define o tamanho da tela
screenSize = (800, 600)
# cria a tela e salva a instância dessa tela em screen
screen = pygame.display.set_mode(screenSize, pygame.DOUBLEBUF)
pygame.display.set_caption("Após o Big Bang")

# cria uma instância do time.Clock() - vamos usar para limitar o fps
gameClock = pygame.time.Clock()

# cria uma variavel que verifica se o jogo ainda está rodando
gameRunning = True

"""
    o os.path.join está sendo usado pois dependendo do sistema operacional,
    a forma para acessar a pasta é diferente, assim, o os.path.join evita esse
    tipo de problema
"""

BG = pygame.image.load(os.path.join("assets", "background-black.png"))

# verifica a cada frame se o jogo está rodando
while gameRunning:
    # limita o FPS em 60 quadros por segundo
    gameClock.tick(60)

    # antes de fazer qualquer coisa, limpa a tela para o próximo frame
    screen.fill((0, 0, 0))

    # verifica os eventos que estão na pool de eventos
    for event in pygame.event.get():
        # verifica se o X (da janela) foi pressionado, se sim, finalziada o jogo (gameRunning = False)
        if(event.type == pygame.QUIT):
            gameRunning = False
        # verifica se uma tecla foi pressionada
        if(event.type == pygame.KEYDOWN):
```

Código de Exemplo #03

```
39              # verifica se a tecla é o ESC, se sim, finaliza o jogo (gameRunning = False)
40              if(event.key == pygame.K_ESCAPE):
41                  gameRunning = False
42
43      # faz uma copia da imagem do background para a tela (screen)
44      screen.blit(BG, (0, 0))
45
46      # depois que você definiu o que desenhar, faça a atualização da tela (chamamos essa parte de double-buffer)
47      # o double-buffer evita flicks na tela
48      pygame.display.update()
49
50  # finaliza todos os módulos que foram iniciados
51  pygame.quit()
```

OBS: A imagem **background-black.png** está dentro de uma pasta **assets**, e a pasta assets deverá estar na mesma pasta que contém o **arquivo .py**

Código de Exemplo #03



O módulo event, mouse, key

- O módulo event é responsável pelo evento de teclado, mouse, joystick porém como diz no manual do pygame, você pode ignorar a handle de event e acessar diretamente o teclado, mouse, joystick pelas funções pygame.key, pygame.mouse e pygame.joystick respectivamente deixando apenas eventos de tela para o módulo event

O módulo event

- **pygame.event.get(eventtype=None) -> Eventlist**
  - Pega uma mensagem na fila de eventos (queue event) e remove a mesma da fila

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

O módulo mouse

- **pygame.mouse.get_pressed() -> (button1, button2, ...)**
  - Retorna uma tupla contendo o status dos botões do mouse
- **pygame.mouse.get_pos() -> (x, y)**
  - Retorna a posição atual do mouse
- **pygame.mouse.get_rel() -> (x, y)**
  - Retorna a posição relativa do mouse, ponto final - ponto inicial que o mouse percorreu
- **pygame.mouse.get_rel() -> (x, y)**
  - Retorna a posição relativa do mouse, ponto final - ponto inicial que o mouse percorreu
- **pygame.mouse.set_visible(bool) -> bool**
  - Faz o mouse ficar invisível ou visível na tela

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

O módulo key

- **pygame.key.get_pressed() -> bools**
  - Retorna uma lista contendo o status de todas as teclas (True representa pressionado e False representa não pressionado)

Para quem quiser se aventurar um pouco mais nessa parte de eventos, o link abaixo é um exemplo de como faz para criar uma caixa de texto dentro do seu jogo

https://github.com/khalidtouch/XigmaLessons/blob/6d95e1961ce86a8258e398f00411b37ad5bc80bf/PYTHON/Music_Player_App/pygame/examples/textinput.py

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

O módulo font

- **pygame.font.init() -> None**
  - Inicializa o módulo da fonte
- **pygame.font.Font(filename, size) -> Font**
  - Função que recebe um arquivo de fonte (.ttf) e seu respectivo tamanho e cria um objeto Font para ser usado.
- **pygame.font.SysFont(name, size, bold=False, italic=False) -> Font**
  - Função que recebe o nome de uma fonte pré-definida no sistema (Ex: “Arial”), com seu respectivo tamanho (size), além disso pode ser definido bold e itálico que por padrão é False
- **Font.render(text, antialias, color, background=None) -> Surface**
  - Essa função pertence ao objeto instanciado Font e cria uma surface contendo o texto que você deseja. O parâmetro antialias se passado como True faz uma suavização nas letras, o parâmetro color diz qual a cor que você deseja para o texto e o background como o nome já fiz, se existe algum plano de fundo no texto (se não for passado nada, será transparente)

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

Exemplo #04

```python
import pygame
import os

#define o tamanho da tela
screenSize = (800, 600)
# cria a tela e salva a instância dessa tela em screen
screen = pygame.display.set_mode(screenSize, pygame.DOUBLEBUF)
pygame.display.set_caption("Exemplo #04")

# cria uma instância do time.Clock() - vamos usar para limitar o fps
gameClock = pygame.time.Clock()

# cria uma variavel que verifica se o jogo ainda está rodando
gameRunning = True
FPS = 60

# carrega a imagem da nave
Ship = pygame.image.load(os.path.join("assets", "ship.png"))

x = (screenSize[0] / 2) - (Ship.get_width() / 2)
y = (screenSize[1] / 2) - (Ship.get_height() / 2)

# velocidade será 5 vezes 1 pixel por segundo
velocity = (1 / FPS) * 5

# inicializa o módulo da fonte
pygame.font.init()
# cria uma fonte do tipo Font
font = None
font = pygame.font.SysFont("Arial", 17)

# exibe uma mensagem que não foi possível inicializar a fonte caso ocorra alguma falha
if(font == None):
    print('Não foi possível criar a fonte')


# verifica a cada frame se o jogo está rodando
```

Exemplo #04

```
38  while gameRunning:
39      # limita o FPS em 60 quadros por segundo (deltaTime é o tempo que levou do frame anterior até agora)
40      # OBS: A soma deltaTime em 60 frames será 1 segundo
41      deltaTime = gameClock.tick(FPS)
42
43      # antes de fazer qualquer coisa, limpa a tela para o próximo frame
44      screen.fill((0, 0, 0))
45
46      # verifica os eventos que estão na pool de eventos
47      for event in pygame.event.get():
48          # verifica se o X (da janela) foi pressionado, se sim, finalziada o jogo (gameRunning = False)
49          if(event.type == pygame.QUIT):
50              gameRunning = False
51          # verifica se uma tecla foi pressionada
52          if(event.type == pygame.KEYDOWN):
53              # verifica se a tecla é o ESC, se sim, finaliza o jogo (gameRunning = False)
54              if(event.key == pygame.K_ESCAPE):
55                  gameRunning = False
56
57      keys = pygame.key.get_pressed()
58
59      # OBS: o eixo x cresce da esquerda para direita e o eixo y de cima para baixo, sendo o ponto (0, 0) o canto superior esquerdo
60
61      # movimenta a nave para cima
62      if(keys[pygame.K_w] or keys[pygame.K_UP]):
63          y = y - deltaTime * velocity
64
```

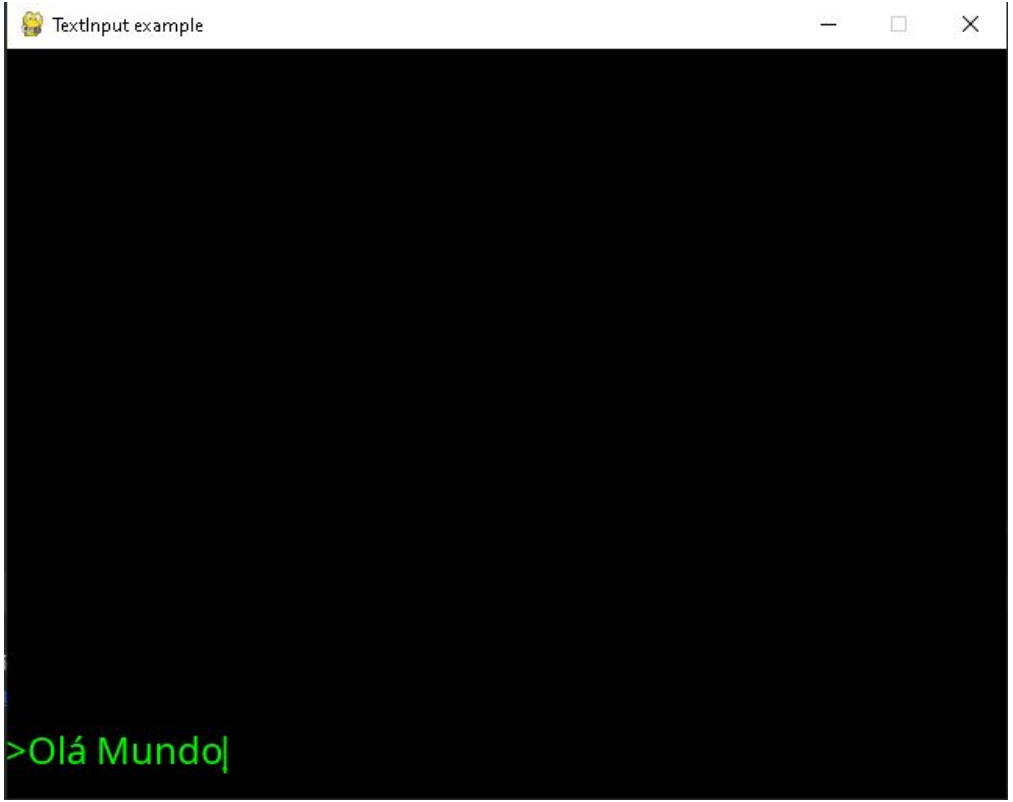
Exemplo #04

```
63          y = y - deltaTime * velocity
64
65      # movimenta a nave para baixo
66      if(keys[pygame.K_s] or keys[pygame.K_DOWN]):
67          y = y + deltaTime * velocity
68
69      # movimenta a nave para a direita
70      if(keys[pygame.K_d] or keys[pygame.K_RIGHT]):
71          x = x + deltaTime * velocity
72
73      # movimenta a nave para a esquerda
74      if(keys[pygame.K_a] or keys[pygame.K_LEFT]):
75          x = x - deltaTime * velocity
76
77      screen.blit(Ship, (x, y))
78
79      # cria um texto contendo a coordenada X e Y da cor branca usando a técnica de antialias
80      textX = font.render(f'X: {x:.2f}', True, (255, 255, 255))
81      textY = font.render(f'Y: {y:.2f}', True, (255, 255, 255))
82
83      # exibe o primeiro texto (textX) na posição x = 10 e y = 10
84      screen.blit(textX, (10, 10))
85
86      # exibe o segundo texto (textY) na posição x = 10 e y = 10 + a altura do textY
87      screen.blit(textY, (10, 10 + textX.get_height()))
88
89
90      howMove = font.render(f'Utilize as teclas WSAD ou as setas do teclado para movimentar a nave', True, (255, 0, 0))
91
92      screen.blit(howMove, (10, screenSize[1] - howMove.get_height() - 10))
93
94
95      # depois que você definiu o que desenhar, faça a atualização da tela (chamamos essa parte de double-buffer)
96      # o double-buffer evita flicks na tela
97      pygame.display.update()
98
99  # finaliza todos os módulos que foram iniciados
100 pygame.quit()
```

Exemplo #04

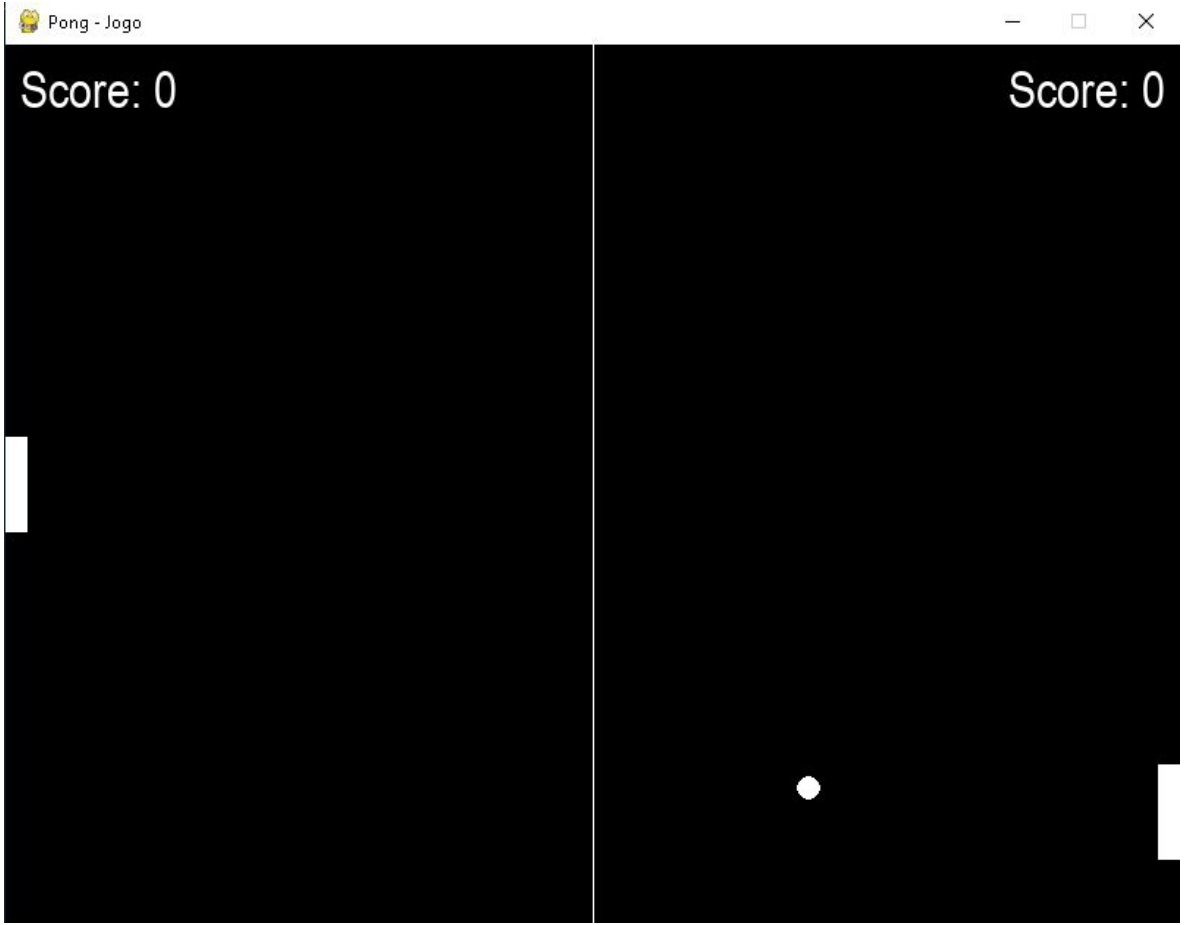

Exemplo #04
X: 395.33
Y: 170.33
Utilize as teclas WSAD ou as setas do teclado para movimentar a nave

Exemplo TextInput (Exemplo do github)

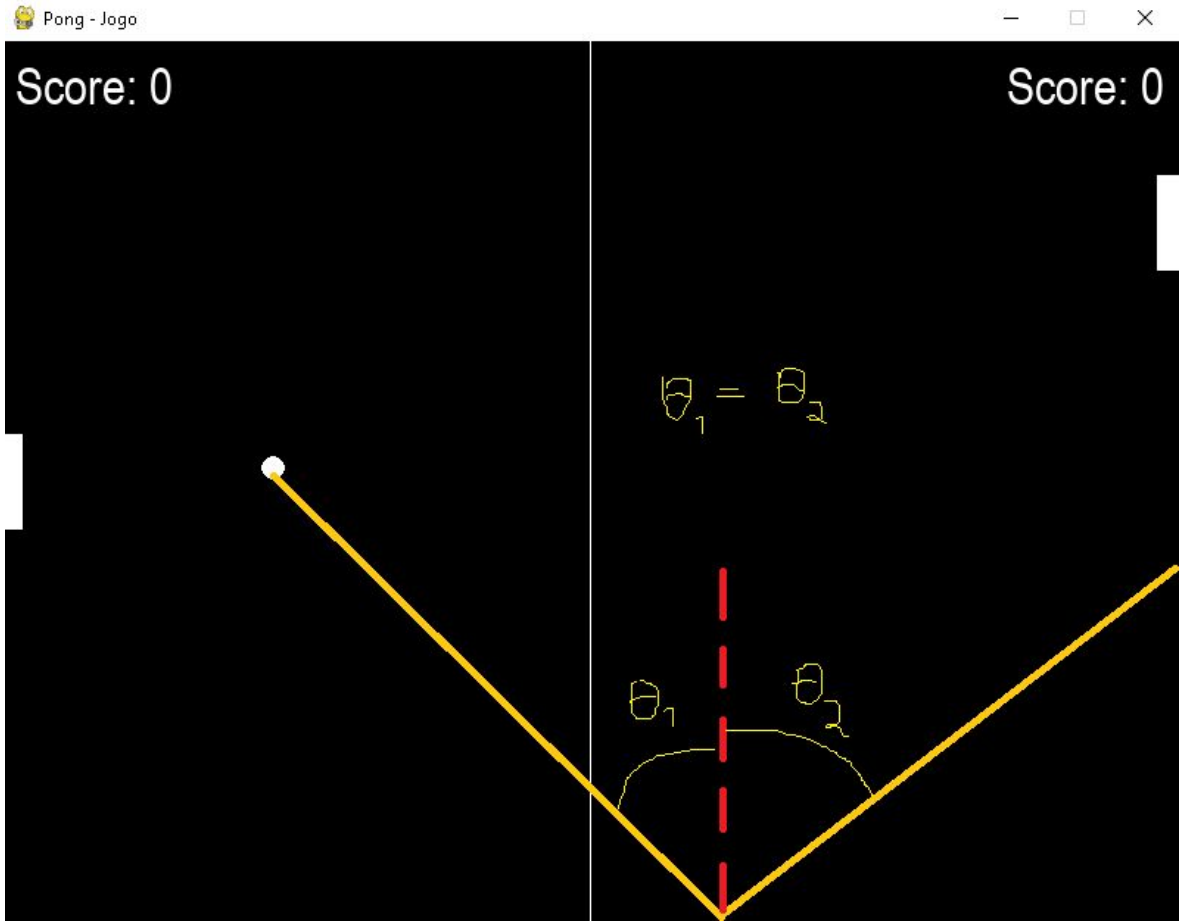


https://github.com/khalidtouch/XigmaLessons/blob/6d95e1961ce86a8258e398f00411b37ad5bc80bf/PYTHON/Music_Player_App/pygame/examples/textinput.py

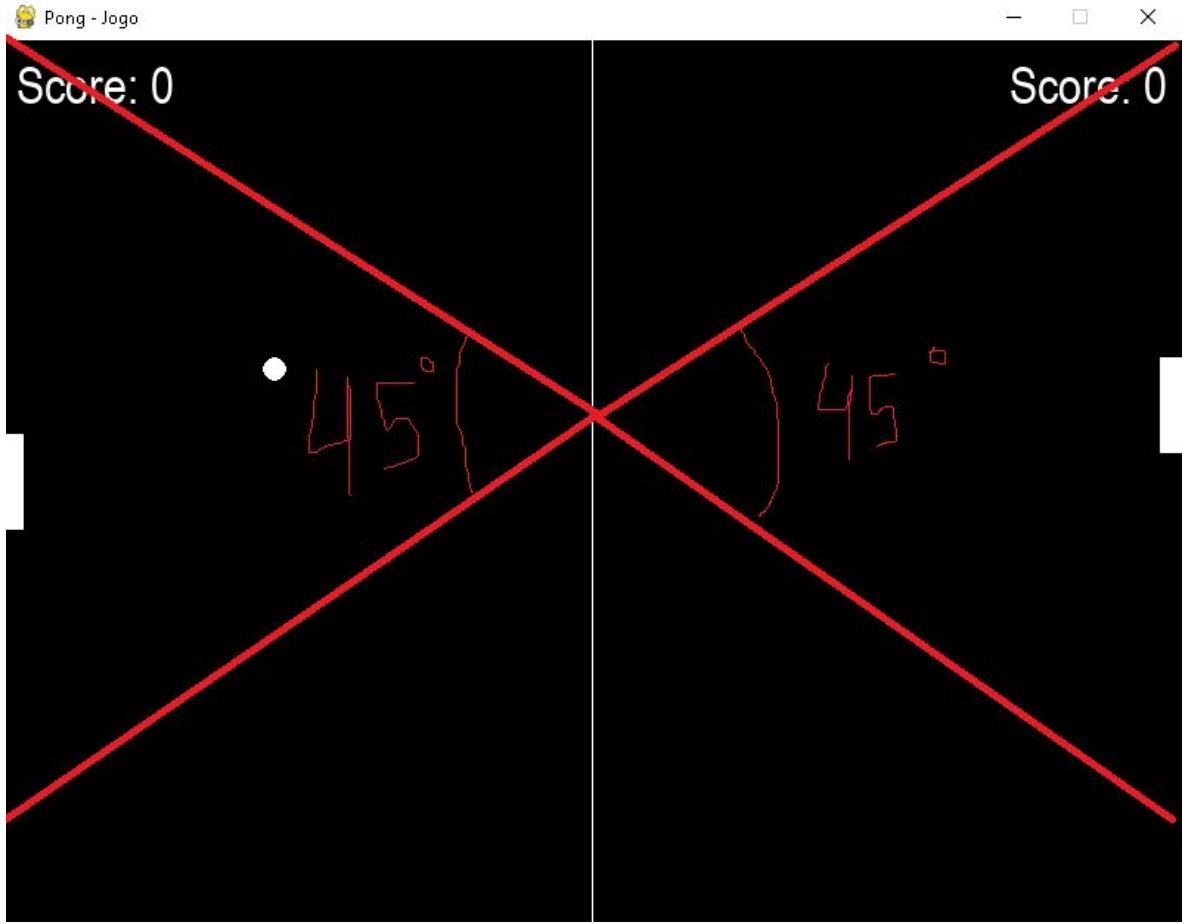
Vamos programar o Pong ??



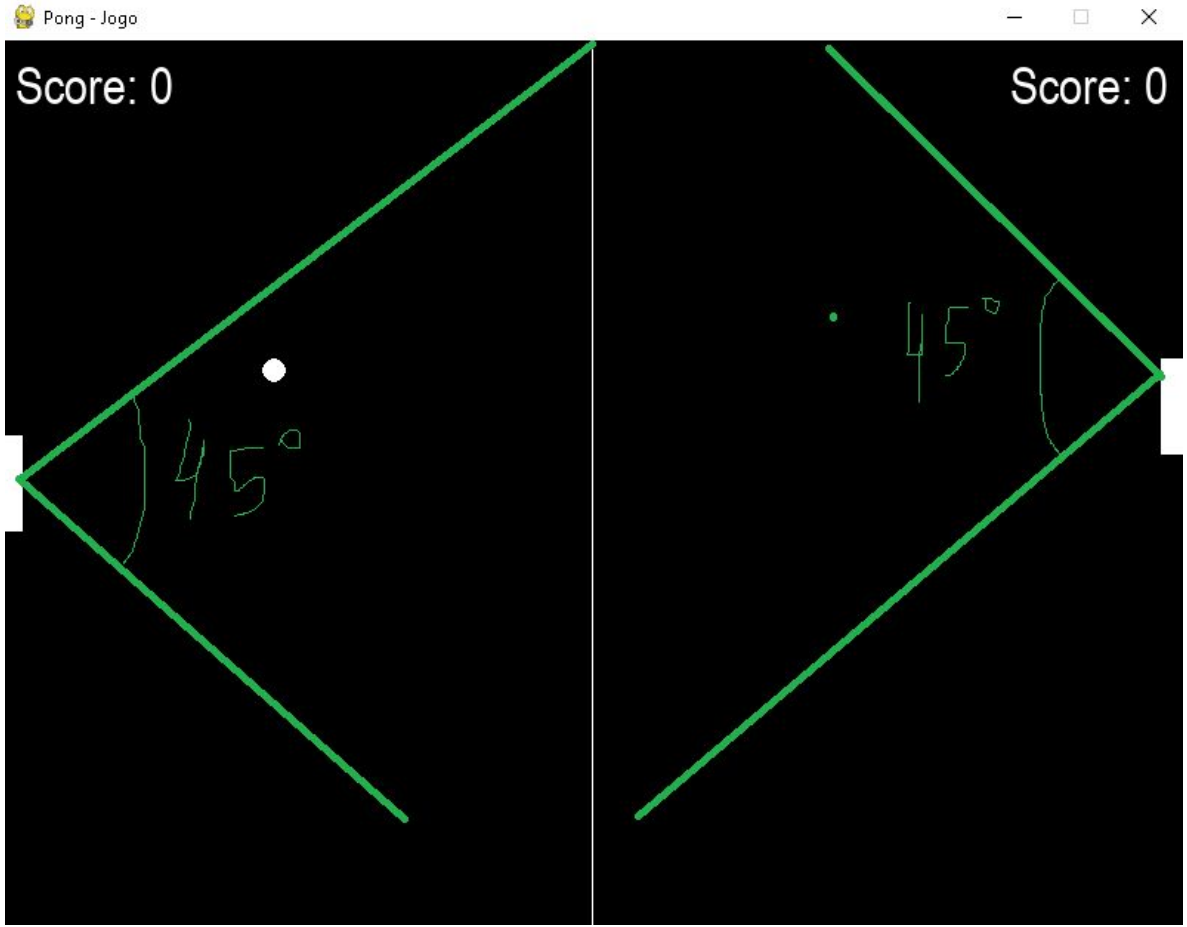
Vamos programar o Pong ?? - Algumas Regras



Vamos programar o Pong ?? - Algumas Regras



Vamos programar o Pong ?? - Algumas Regras



Roteiro de Programa

1. Criar uma estrutura (class) que terá dados comuns tanto ao jogador quanto a máquina
2. Instanciar essa estrutura (class), um para o jogador outro para a máquina
3. Iniciar as variáveis do jogador e da máquina com valores adequados
4. Desenhar o jogador e a máquina
5. Movimentar o jogador na tela (a máquina (adversário) vamos movimentar depois)
6. Limitar o jogador e a máquina na tela
7. Criar uma estrutura (class) que terá todos os dados da bola
8. Instanciar a bola
9. Desenhar a bola
10. Movimentar a bola segundo as regras estabelecidas anteriormente (cone de lançamento)
11. Limitar a bola na tela superiormente e inferiormente
12. Verificar se a bola saiu pelo lado direito, o lado da máquina (ponto do jogador)
13. Verificar se a bola saiu pelo lado esquerdo, o lado do jogador (ponto para a máquina)
14. Movimentando a máquina segundo uma regra (a máquina vai sempre se movimentar junto com a bola no eixo y)
15. Colocando placar do jogador e da máquina (Uso de Font)
16. Desenhando o campo, dando efeitos para a bola (criatividade e efeitos visuais), vamos nos limitar apenas a desenhar um traço no centro da tela (divisória do campo)
17. Upgrades (ai é com você)

Código Fonte dos Exemplos e do Pong

Código Fonte dos Exemplos
https://github.com/AchcarLucas/Source-Monitoria
Código Fonte do Jogo Pong
https://github.com/AchcarLucas/Pong