Diego Oliveira de Lemos
diegodol-@hotmail.com

O que é o Pygame?

- É uma biblioteca grátis em python para fazer aplicações multimídia, incluindo jogos.
- O Pygame é uma espécie de “game engine”.

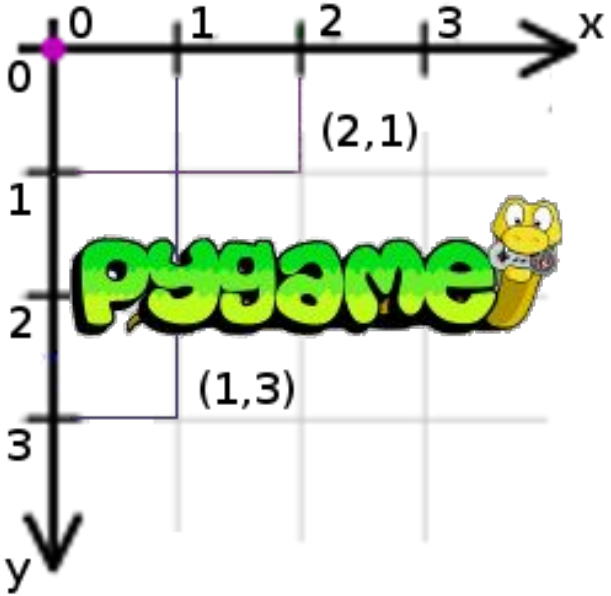
O que é uma biblioteca?

- É um conjunto de funções pré-escritas por outras pessoas que estão salvas em arquivos de programação.

O que é um jogo?

Conceitos importantes de jogos digitais:

1) Frame: um quadro por instante de tempo.
2) Event: qualquer ação que altera qualquer coisa no jogo.
3) Sprite: um objeto gráfico que pode ser manipulado como tal.
4) Pixel: é a unidade mínima de uma imagem.



Módulos do Pygame

Em python módulos são subdivisões de uma biblioteca para ajudar a organizar as funções por similaridade de categoria. Exemplo: **mouse, Rect, time, music, pygame, draw, event, font...**

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

1. Iniciando o Pygame

- Sempre no início do programa será necessário importar o Pygame para que se possa usar a biblioteca.
- Também é necessário inicializar o Pygame e finalizar após a conclusão da execução do jogo.
- Tudo do jogo ficará entre as funções init() e quit()

```
import pygame

pygame.init()

pygame.quit()
```

2. Configurando a tela

- Agora é necessário criar uma tela e mostrar para o usuário. A tela será guardada como uma variável para facilitar sua manipulação posteriormente.
- Na criação da tela, em **set_mode((640,480))**, deve-se passar como parâmetros o tamanho desejado da tela, (x,y).
- Note que as funções pertencem ao módulo display.

```
import pygame

pygame.init()

screen = pygame.display.set_mode((640,480)) #
pygame.display.set_caption( "Tutorial")#

pygame.quit()
```

3. Criando o laço do jogo

- O jogo tinha executado uma vez e logo em seguida encerrado. Agora é necessário achar uma forma de executar o jogo “para sempre”, ou até que o jogador queira encerrar. É necessário agora colocar um laço de repetição que faça isso, o while. O while vai ser controlado por uma variável que vai ser **True** até que o usuário faça alguma ação específica.
- Como cada ação dentro do jogo é um evento, cria-se um **for** para “tratar” todos os eventos que ocorrem a cada instante. Tudo que for dependente de eventos será colocado dentro desse for.
- A função **display.update()** atualiza a minha tela a cada execução de evento pois está dentro do for.

```
import pygame

pygame.init()

screen =
pygame.display.set_mode((640,480))
pygame.display.set_caption( "Tutorial")

jogo = True#
while jogo:#
    for evento in pygame.event.get(): #

        pygame.display.update() #

pygame.quit()
```

4. Trantar evento

- O primeiro evento tratado é o evento do tipo QUIT que encerrará o jogo caso o usuário clique no botão de fechar.
- A lógica é mudar a variável de controle do while para que se possa sair do while e encerrar o jogo.

```
. . .

jogo = True
while jogo:
    for evento in pygame.event.get():
        if (evento.type ==
pygame.QUIT):#
          jogo = False#
        pygame.display.update()

pygame.quit()
```

5. Criar cores

- Depois que a tela está configurada e conseguimos exibir de forma estável o jogo, podemos “desenhar e pintar” pixels sobre essa tela.
- Antes é preciso criar as cores. Cores em python são tuplas que possuem três parâmetros (RED, GREEN, BLUE).
- Tuplas são listas imutáveis.

```
import pygame

pygame.init()

white = (255,255,255)  #(R,G,B)
pink = (255,200,200) #
green = (0,255,0) #

screen = pygame.display.set_mode((640,480))
pygame.display.set_caption("Tutorial")

. . .
```

6. Pintar a tela

- Agora podemos pintar a tela de branco com a função **fill()**. Para isso, invocamos ela a partir da variável **screen** que criamos anteriormente.

```
. . .

jogo = True
while jogo:
    for evento in pygame.event.get():
        if (evento.type == pygame.QUIT):
          jogo = False
        screen.fill(white) #
        pygame.display.update()

pygame.quit()
```

7. Desenhando um retângulo

- Pygame tem funções específicas para desenhar formas geométricas. Para desenhar um retângulo fazemos o seguinte:

```
pygame.draw.rect(Surface, color, Rect, width )

# a superfície, a cor, o retângulo e a largura da linha
```

```
Quando se tem dúvida sobre como uma função se comporta ou como é
escrita deve-se consultar a documentação da biblioteca, nesse
caso o módulo draw do pygame.

rect(Surface, color, Rect, width=0) -> Rect
                  +
Rect(left, top, width, height) -> Rect
                  ||
rect(Surface, color, (left, top, width, height) , width=0) -> Rect
```

8. Desenhar

- #1 desenha um retângulo rosa, de tamanho 50x50 e que começa na posição 0,0 da tela.
- #2 desenha um retângulo verde, de tamanho 30x30 e que também começa na posição 0,0.

```
. . .

jogo = True
while jogo:
    for evento in pygame.event.get():
        if (evento.type == pygame.QUIT):
          jogo = False

        screen.fill(white)
        pygame.draw.rect(screen,pink,(0,0,50,50),0) #1
        pygame.draw.rect(screen,green,(0,0,30,30),1) #
        2

        pygame.display.update()

pygame.quit()
```

9. Biblioteca random

- Agora vamos combinar o Pygame com a biblioteca random (números aleatórios) para criar números aleatórios e assim poder criar retângulos de tamanho aleatório e em posição aleatória. Para isso é preciso importar a biblioteca no começo do código:

```
import pygame
import random#

pygame.init()

. . .
```

10. Entrada do teclado

- Para tratar os eventos do teclado usa-se o módulo **key**.
- No código ao lado, a variável **keys** vai receber um vetor da função **get_pressed()** onde cada posição do vetor representa um botão do teclado.
- Em cada posição desse vetor é armazenado o valor **True** ou **False** para indicar se aquele botão está pressionado.
- Agora para saber se um botão está pressionado basta “consultar” se o seu valor no vetor está em True ou False.

```
. . .
keys = pygame.key.get_pressed()

if (keys[pygame.K_1]==True):
. . .
```

11. Desenhando o retângulo aleatório

- Será definido como regra do nosso tutorial o seguinte: quando o usuário pressionar o botão 1 do teclado um retângulo aleatório deverá ser desenhado.
- Para tanto, devemos definir previamente os valores dos parâmetros com uma função aleatória, random.randint().

```
. . .
keys = pygame.key.get_pressed()

if (keys[pygame.K_1]==True):
    width = random.randint(0,50)
    height = random.randint(0,50)
    left = random.randint(0,400)
    top = random.randint(0,400)
    pygame.draw.rect(screen,green,(left,top,width,height),1)

. . .
```