EBook Gratis

APRENDIZAJE pygame

Free unaffiliated eBook created from
Stack Overflow contributors.

#pygame

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: pygame

It is an unofficial and free pygame ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official pygame.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con pygame

Observaciones

Pygame es un envoltorio de Python para SDL , una biblioteca multiplataforma de C para controlar multimedia, escrito por Pete Shinners. Esto significa que, mediante pygame, puede escribir videojuegos u otras aplicaciones multimedia en Python que se ejecutarán sin modificaciones en cualquiera de las plataformas compatibles con SDL (Windows, Unix, Mac, beOS y otras).

Esta sección proporciona una descripción general de qué es pygame y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema grande dentro de pygame, y vincular a los temas relacionados. Dado que la Documentación para pygame es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

Versiones

Versión	==========>	Fecha de lanzamiento
Pygame 1.9.0	========>	1 de agosto de 2009
Pygame 1.8.1	========>	30 de julio de 2008
Pygame 1.8.0	========>	29 de marzo de 2008
Pygame 1.7.1	========>	16 de agosto de 2005
Pygame 1.6.2	========>	-
Pygame 1.6	=========>	23 de octubre de 2003
Pygame 1.5	=========>	30 de mayo de 2002
Pygame 1.4	=========>	30 de enero de 2002
Pygame 1.3	=========>	19 de diciembre de 2001
Pygame 1.2	=========>	Sep 4, 2001
Pygame 1.1	=========>	Jun 23, 2001
Pygame 1.0	=========>	5 de abril de 2001
Pygame 0.9	=========>	Feb 13, 2001
Pygame 0.5	=========>	6 de enero de 2001
Pygame 0.4	=========>	14 de diciembre de 2000
Pygame 0.3	=========>	Nov 20, 2000
Pygame 0.2	=========>	Nov 3, 2000
Pygame 0.1	=========>	Oct 28, 2000

Examples

Un simple 'juego'

Importar e inicializar

Cada módulo necesita ser importado y pygame no es una excepción. Aunque necesitamos llamar a la función `pygame.init()` para que todos los módulos importados en pygame se inicialicen correctamente. Si olvidamos esto, algunos módulos no funcionarán. La función también devuelve una tupla de todas las inicializaciones exitosas y fallidas (no generará un error si un módulo no se inicializa).

```
import pygame
successes, failures = pygame.init()
print("{0} successes and {1} failures".format(successes, failures))
```

Crear necesidades

También necesitamos crear una pantalla. Pygame ya ha creado una pantalla (oculta), por lo que todo lo que tenemos que hacer es configurar el modo de la pantalla (en este ejemplo solo configuramos la resolución). También es una buena idea crear un reloj para asegurarse de que nuestro programa se actualice a una velocidad fija (de lo contrario, funcionaría a una velocidad diferente según la velocidad de la computadora).

```
screen = pygame.display.set_mode((720, 480))  # Notice the tuple! It's not 2 arguments.
clock = pygame.time.Clock()
FPS = 60  # This variable will define how many frames we update per second.
```

Para un poco de legibilidad más adelante en nuestro código, crearemos dos constantes de color, que representan una tupla de Rojo, Verde y Azul (RGB). Los valores van desde 0 (sin luz) a 255 (luz completa).

```
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
```

En pygame usualmente usamos una *Superficie* para representar la apariencia de un objeto, y un *Rect* (rectángulo) para representar la posición de un objeto. Una *superficie* es como una hoja de papel en blanco que contiene colores o imágenes. Si está creando una clase, debe asignar *un* nombre a la *imagen de* los atributos y *rect,* ya que muchas funciones buscarán y usarán esos atributos. Estas clases se beneficiarían al heredar la clase `pygame.sprite.Sprite` por razones que puede leer aquí .

```
rect = pygame.Rect((0, 0), (32, 32))  # First tuple is position, second is size.
image = pygame.Surface((32, 32))  # The tuple represent size.
image.fill(WHITE)  # We fill our surface with a nice white color (by default black).
```

El bucle de juego

Ahora tenemos todo listo para nuestro bucle de juego. Este es un bucle que se ejecutará durante

todo el juego, donde manejamos eventos y actualizamos la pantalla y las posiciones de nuestros objetos.

Primero nos aseguraremos de que nuestro bucle se ejecute en un *FPS* determinado. Definimos el *FPS* y creamos nuestro reloj al comienzo del programa. El siguiente código asegurará que nuestro programa se duerma lo suficiente como para que nuestro bucle repita la cantidad que definimos como nuestro *FPS* . En este ejemplo, 60 veces por segundo.

```
clock.tick(FPS)
```

Entonces nos encargaremos de los eventos. Un evento es básicamente una acción del usuario, como mover el mouse o presionar una tecla. Pygame registrará todos estos eventos en una cola que obtendremos llamando a `pygame.event.get()` . Podemos iterar sobre esto y verificar si hay un evento que nos gustaría manejar. Los eventos tienen un atributo de *tipo* que podemos comparar con las constantes en el módulo de pygame para determinar qué tipo de evento es.

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:  # The user pressed the close button in the top corner of
the window.
        quit()
        # Close the program. Other methods like 'raise SystemExit' or 'sys.exit()'.
        # Calling 'pygame.quit()' won't close the program! It will just uninitialize the
modules.
```

También podemos verificar `if event.type == pygame.KEYDOWN` para ver si el usuario ha presionado una tecla hacia abajo. En ese caso, el evento tiene una *clave de* atributo que podemos verificar para ver qué clave representa.

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        quit()
    elif event.type == pygame.KEYDOWN:
        if event.key == pygame.K_w:
            rect.move_ip(0, -2)  # Changes the rect's position.
        elif event.key == pygame.K_s:
            rect.move_ip(0, 2)
        elif event.key == pygame.K_a:
            rect.move_ip(-2, 0)
        elif event.key == pygame.K_d:
            rect.move_ip(2, 0)
```

Ahora necesitamos mostrar nuestra imagen. Primero podríamos querer borrar nuestra pantalla de la representación anterior. Lo hacemos llenando toda la pantalla con negro (elimine el código para ver por qué queremos borrarlo). Entonces tenemos que *blit* nuestra *imagen* a la pantalla. Bitting esencialmente significa copiar la *imagen* a otra superficie (en nuestro caso, la pantalla). Por último *volteamos* o *actualizamos* la pantalla.

Cuando estamos trabajando, no estamos mostrando nada al usuario. Imagínelo como la computadora en un lado y el usuario en el otro. La computadora dibuja (se *funde*) en su lado de la pantalla, la *voltea* hacia el usuario y luego la repite.

```
screen.fill(BLACK)
screen.blit(image, rect)
pygame.display.update()  # Or 'pygame.display.flip()'.
```

Ahora tenemos un juego básico! Bastante aburrido, sí, ¡pero lo esencial está ahí! Combine esto con su conocimiento actual de Python y debería poder crear algo increíble.

Código completo

```
import pygame
successes, failures = pygame.init()
print("{0} successes and {1} failures".format(successes, failures))


screen = pygame.display.set_mode((720, 480))
clock = pygame.time.Clock()
FPS = 60  # Frames per second.

BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
# RED = (255, 0, 0), GREEN = (0, 255, 0), BLUE = (0, 0, 255).

rect = pygame.Rect((0, 0), (32, 32))
image = pygame.Surface((32, 32))
image .fill(WHITE)

while True:
    clock.tick(FPS)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            quit()
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_w:
                rect.move_ip(0, -2)
            elif event.key == pygame.K_s:
                rect.move_ip(0, 2)
            elif event.key == pygame.K_a:
                rect.move_ip(-2, 0)
            elif event.key == pygame.K_d:
                rect.move_ip(2, 0)

    screen.fill(BLACK)
    screen.blit(image, rect)
    pygame.display.update()  # Or pygame.display.flip()
```

Mecánica de juego ligeramente mejorada.

Tenga en cuenta que el programa verifica cuándo presionamos la tecla y no cuando presionamos la tecla hacia abajo. Para solucionar esto podríamos introducir una variable de *velocidad* . Podemos crear una clase de jugador para mantenerla más organizada. Para evitar el movimiento dependiente del cuadro (si cambiáramos el FPS a 30, los objetos se moverían a la mitad de la velocidad) introducimos el movimiento dependiente del tiempo al pasar el tiempo entre tics a

nuestros objetos móviles.

```
import pygame

successes, failures = pygame.init()
print("Initializing pygame: {0} successes and {1} failures.".format(successes, failures))

screen = pygame.display.set_mode((720, 480))
clock = pygame.time.Clock()
FPS = 60

BLACK = (0, 0, 0)
WHITE = (255, 255, 255)


class Player(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.Surface((32, 32))
        self.image.fill(WHITE)
        self.rect = self.image.get_rect()  # Get rect of some size as 'image'.
        self.velocity = [0, 0]

    def update(self):
        self.rect.move_ip(*self.velocity)


player = Player()
running = True
while running:
    dt = clock.tick(FPS) / 1000  # Returns milliseconds between each call to 'tick'. The
convert time to seconds.
    screen.fill(BLACK)  # Fill the screen with background color.

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_w:
                player.velocity[1] = -200 * dt  # 200 pixels per second
            elif event.key == pygame.K_s:
                player.velocity[1] = 200 * dt
            elif event.key == pygame.K_a:
                player.velocity[0] = -200 * dt
            elif event.key == pygame.K_d:
                player.velocity[0] = 200 * dt
        elif event.type == pygame.KEYUP:
            if event.key == pygame.K_w or event.key == pygame.K_s:
                player.velocity[1] = 0
            elif event.key == pygame.K_a or event.key == pygame.K_d:
                player.velocity[0] = 0

    player.update()

    screen.blit(player.image, player.rect)
    pygame.display.update()  # Or pygame.display.flip()

print("Exited the game loop. Game will quit...")
quit()  # Not actually necessary since the script will exit anyway.
```

Todavía hay muchas cosas que deben mejorarse sobre este código. Te recomiendo que leas el tutorial de pygame y esta charla de Richard Jones para más información.

Instalando pygame

En las ventanas

1. Vaya a http://www.lfd.uci.edu/~gohlke/pythonlibs/#pygame - un sitio no oficial que proporciona paquetes de Python de código abierto para la distribución oficial de CPython por *Christoph Gohlke* .
2. Descargue el archivo `.whl` .whl apropiado de acuerdo con su versión de python instalada. (El archivo se llama algo así como `pygame - <pygame version> - <python version> - win32.whl`)
3. correr

   ```
   pip install your-pygame-package.whl
   ```

   Dentro de su terminal, bash o consola.
   Nota: si no se encuentra `pip` en `PATH` intente ejecutar `python -m pip install your-pygame-package.whl`
4. Comprueba si puedes importar pygame como un módulo de python

   ```
   import pygame
   ```

   Si no recibe un error, ha instalado correctamente pygame en su computadora :)

En linux

1. Abre tu terminal y corre

   ```
   sudo apt-get install python-pygame
   ```

   Nota: Esto instalará pygame para python2
2. Intenta importar pygame dentro

   ```
   import pygame
   ```

   Si no recibe un error, ha instalado correctamente pygame en su sistema Linux :)

En macOS

Hay dos formas de instalarlo en mac:

Método 1

Ve a la página de descargas de Pygame y descarga el instalador de mac. Ejecútalo, y debería instalar Pygame en tu Mac.

Método 2

Instalar Homebrew :

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Luego usa Homebrew para instalar Python 2.7.12 y Pygame:

```
brew install python; brew install homebrew/python/pygame
```

Ahora ejecuta Python en tu terminal e intenta `import pygame` . Si no dice nada, se instala con éxito.

Importación de pygame y dibujo en una pantalla.

Empezando

Debes hacer lo siguiente para comenzar con Pygame:

```
import pygame
```

Esto abre una ventana de tamaño 640,480 y la almacena en una variable llamada pantalla.

Configuración de un nombre de ventana

La configuración de un nombre para la ventana de pygame requiere la siguiente sintaxis:

```
pygame.display.set_caption('Name')
```

Sobre la pantalla

- El punto (0,0) se encuentra en la esquina superior izquierda de la pantalla.
- Las coordenadas x aumentan de izquierda a derecha, las coordenadas y aumentan de arriba a abajo. Es decir, las coordenadas del lado derecho en el plano cartesiano son positivas y el lado izquierdo es negativo. Sin embargo, las coordenadas del lado superior en el plano cartesiano son negativas arriba y positivas en la parte inferior. (**Nota** : esto se considera si los puntos se toman del origen).

Actualizando la pantalla

Los cambios que realice en la pantalla, por ejemplo, llenándolos de color o dibujando en ella, ¡no

se muestran de inmediato!
Así que ¿cómo se hace?
Tienes que llamar a esta función:

```
pygame.display.update()
```

Colores

La coloración en pygame funciona en modo RGB.
El código para colorear es:

```
color_Name = (r,g,b)
```

- R significa rojo.
- G significa verde
- B significa azul.
- Los tres deben ser números enteros entre 0 y 255, con 255 más brillantes y 0 más oscuros

Dibujo

1. Dibujar lineas

   ```
   pygame.draw.lines(screen, color, closed, pointlist, thickness)
   ```

2. Dibujar rectángulo

   ```
   pygame.draw.rect(screen, color, (x,y,width,height), thickness)
   ```

3. Dibujar circulo

   ```
   pygame.draw.circle(screen, color, (x,y), radius, thickness)
   ```

Poner todo en un bucle

Para hacer un bucle usa el siguiente código:

```
running = True
while running:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      running = False
      pygame.quit()
```

Dibujando un rectángulo en la ventana de

pygame (código)

```
import pygame
background_colour = (255,255,255) # White color
(width, height) = (300, 200) # Screen size
color=(0,0,0) #For retangle
screen = pygame.display.set_mode((width, height)) #Setting Screen
pygame.display.set_caption('Drawing') #Window Name
screen.fill(background_colour)#Fills white to screen
pygame.draw.rect(screen, color, (100,50,30,40), 1) #Drawing the rectangle
pygame.display.update()

#Loop
running = True
while running:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      running = False
      pygame.quit()
```

Lea Empezando con pygame en línea: https://riptutorial.com/es/pygame/topic/3959/empezando-con-pygame

Capítulo 2: Añadiendo música de fondo y efectos de sonido.

Observaciones

Intente reproducir música en '.wav' en lugar de '.mp3'. En '.mp3' la música se retrasa.

Examples

Ejemplo para añadir música en pygame.

```
import pygame
file = 'some.mp3'
pygame.init()
pygame.mixer.init()
pygame.mixer.music.load(file)
pygame.mixer.music.play(-1) # If the loops is -1 then the music will repeat indefinitely.
```

Ejemplo para agregar lista de reproducción de música en pygame

```
import pygame
import time

pygame.mixer.init()
pygame.display.init()

screen = pygame.display.set_mode ( ( 420 , 240 ) )

playlist = list()
playlist.append ( "music3.mp3" )
playlist.append ( "music2.mp3" )
playlist.append ( "music1.mp3" )

pygame.mixer.music.load ( playlist.pop() )  # Get the first track from the playlist
pygame.mixer.music.queue ( playlist.pop() ) # Queue the 2nd song
pygame.mixer.music.set_endevent ( pygame.USEREVENT )    # Setup the end track event
pygame.mixer.music.play()           # Play the music

running = True
while running:
   for event in pygame.event.get():
      if event.type == pygame.USEREVENT:    # A track has ended
         if len ( playlist ) > 0:       # If there are more tracks in the queue...
            pygame.mixer.music.queue ( playlist.pop() ) # Q
```

Lea Añadiendo música de fondo y efectos de sonido. en línea:
https://riptutorial.com/es/pygame/topic/7419/anadiendo-musica-de-fondo-y-efectos-de-sonido-

Capítulo 3: Creando una ventana de pygame

Observaciones

Si desea tener otros colores como fondo, entonces nombre una nueva variable como `red = (255,0,0)` y cambie `display.fill(black)` a `display.fill(red)` . Puede crear colores almacenándolos en una variable y verificando sus valores RGB desde Internet.

Examples

Creando la ventana de pygame

```
import pygame

background_colour = (255,255,255) # For the background color of your window
(width, height) = (300, 200) # Dimension of the window

screen = pygame.display.set_mode((width, height)) # Making of the screen
pygame.display.set_caption('Tutorial 1') # Name for the window
screen.fill(background_colour) #This syntax fills the background colour

pygame.display.flip()

running = True
while running:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      running = False
      pygame.quit()
```

Lea Creando una ventana de pygame en línea:
https://riptutorial.com/es/pygame/topic/6477/creando-una-ventana-de-pygame

Capítulo 4: Creando una ventana de pygame simple

Examples

El codigo completo

```
import pygame

pygame.init()

WIDTH = 300
HEIGHT = 200
SCREEN = pygame.display.set_mode((WIDTH, HEIGHT))

pygame.display.set_caption('My Game')

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)
YELLOW = (255, 255, 255)

SCREEN.fill(RED)
pygame.display.flip()

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

pygame.quit()
```

Importando e inicializando pygame.

Como hacemos con cualquier módulo en python, necesitamos importar pygame:

```
import pygame
```

Luego inicializamos todos los módulos de pygame importados:

```
pygame.init()
```

Esto se utiliza para inicializar todos los módulos de pygame. Sin esto los módulos no funcionarían.

Definiendo constantes

Entonces definimos algunas constantes aquí:

```
WIDTH = 300
HEIGHT = 200
SCREEN = pygame.display.set_mode((WIDTH, HEIGHT))
```

Las constantes `WIDTH` y `HEIGHT` se utilizan para crear una ventana, que tendría un ancho de 300 píxeles y una altura de 200 píxeles. La función utilizada en `SCREEN` , `pygame.display.set_mode((WIDTH, HEIGHT))` , establecerá el modo de la pantalla y devolverá un objeto Surface . Observe cómo los parámetros para esta función son las constantes `WIDTH` y `HEIGHT` definidas anteriormente.

Configurando el nombre de la ventana

Luego usamos esta función para cambiar el nombre de la ventana a Mi juego:

```
pygame.display.set_caption('My Game')
```

Definiendo colores

Luego definimos 6 colores que se pueden usar en nuestra ventana:

```
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)
YELLOW = (255, 255, 255)
```

Cuando definimos colores, colocamos 3 valores que oscilan entre 0 y 255. La clase pygame.Color normalmente toma este formato:

```
COLOUR = (r, g, b, a)
```

Cuando el parámetro r establece el valor rojo del color, el parámetro g establece el valor verde del color y el parámetro b establece el valor azul del color. El parámetro a establece el valor alfa del color.

Entonces damos este comando:

```
SCREEN.fill(RED)
```

Esta es una función pygame.Surface.fill que llena el objeto Surface, nuestra pantalla, con el color rojo.

Usando pygame.display.flip ()

Entonces usamos esta función

```
pygame.display.flip()
```

Básicamente, esto hace que todo lo que hemos dibujado en la pantalla Superficie se haga visible y actualice el contenido de toda la pantalla. Sin esta línea, el usuario no vería nada en su pantalla de pygame.

El bucle de juego

Las siguientes líneas son lo que se llama un "bucle de juego".

Para empezar, hacemos una variable y la hacemos verdadera:

```
is_running = True
```

Para que podamos comenzar nuestro bucle while:

```
while is_running:
```

que se ejecutará a lo largo de todo el juego.

En su forma más básica, pygame tiene "eventos" que toman las entradas del usuario, por ejemplo, al presionar un botón o al hacer clic con el mouse. Pygame maneja estos eventos a través de una cola de eventos. Podemos obtener estos eventos de la cola de eventos con este bucle for:

```
for event in pygame.event.get():
```

Que básicamente pasa por una lista de eventos, nuestra cola de eventos. Estas son las siguientes 2 líneas:

```
if event.type == pygame.QUIT:
    is_running = False
```

Esto lo hará para que cuando el usuario presione el botón de salir en la esquina superior, ocurra el evento con el tipo `pygame.QUIT` .

Esto finaliza el bucle while, ya que `is_running` ahora es `False` y la secuencia de comandos pasa a la línea final:

```
pygame.quit()
```

Lo que inicializa los módulos de pygame.

Lea Creando una ventana de pygame simple en línea:
https://riptutorial.com/es/pygame/topic/6597/creando-una-ventana-de-pygame-simple

Capítulo 5: Creando una ventana en pygame - pygame.display.set_mode ()

Sintaxis

- pygame.display.set_mode (resolución = (0,0), indicadores = 0, profundidad = 0) # Devuelve un pygame.Surface que representa la ventana en pantalla
- flags = pygame.FULLSCREEN | pygame.OPENGL # Las banderas se pueden combinar usando el "|" (O a lo largo del bit o "pipe") carácter.

Parámetros

parámetro	explicación
resolución	Un par de números que representan el ancho y el alto de la ventana.
banderas	opciones adicionales que cambian el tipo de ventana; consulte "Observaciones" para ver las banderas disponibles
profundidad	cantidad de bits utilizados para el color

Observaciones

- Los valores posibles para los argumentos de `flag` son:

bandera	descripción
pygame.FULLSCREEN	la ventana está en pantalla completa
pygame.RESIZABLE	la ventana es de tamaño variable
pygame.NOFRAME	La ventana no tiene borde ni controles.
pygame.DOUBLEBUF	usar doble buffer - recomendado para `HWSURFACE` o `OPENGL`
pygame.HWSURFACE	La ventana es acelerada por hardware, solo posible en combinación con `FULLSCREEN`
pygame.OPENGL	La ventana es ejecutable por OpenGL

Otras observaciones:

- Pygame actualmente solo puede manejar una única ventana a la vez. Al crear una segunda ventana llamando a `pygame.display.set_mode((x,y))` por segunda vez, se cerrará la primera

ventana.

- Cambiar el argumento de las `depths` casi nunca es necesario, pygame seleccionará el mejor por sí mismo. En caso de que se establezca una profundidad que no sea compatible con el sistema, pygame emulará esta profundidad, que puede ser muy lenta.
- Las cosas que se dibujan en la superficie que devuelve `pygame.display.set_mode()` no son visibles inmediatamente en la pantalla; la pantalla debe `pygame.display.update()` usando `pygame.display.update()` o `pygame.display.flip()` .

Examples

Crear una ventana de pygame

Esto crea una ventana en pantalla completa con un tamaño de 500x500 píxeles:

```
pygame.init()
screen = pygame.display.set_mode((500, 500), pygame.FULLSCREEN)
```

`screen` representa desde ahora en la ventana en pantalla; Es un objeto pygame.Surface. Todo lo que debería ser visible para el usuario debe dibujarse en él utilizando `screen.blit` .

Lea Creando una ventana en pygame - pygame.display.set_mode () en línea: https://riptutorial.com/es/pygame/topic/6442/creando-una-ventana-en-pygame---pygame-display-set-mode---

Capítulo 6: Dibujo en la pantalla

Examples

Dibujar formas, texto e imágenes en la pantalla con una pequeña animación.

Este programa dibujará algunas formas en la pantalla, dibuja "¡Hola mundo!" en el centro de la pantalla y deja que una imagen vaya a cada esquina de la ventana. Puede usar cada imagen que desee, pero **tendrá que colocar el archivo de imagen en el mismo directorio que su programa.**

el código completo:

```
import pygame, sys
from pygame.locals import *

pygame.init()

FPS = 30 #frames per second setting
fpsClock = pygame.time.Clock()

#set up the window
screen = pygame.display.set_mode((400, 300), 0, 32)
pygame.display.set_caption('animation')

#set up the colors
white = (255, 255, 255)
black = (  0,   0,   0)
green = (0, 255, 0)
blue = (0, 0, 180)
red   = (255,   0,   0)

image  = pygame.image.load('image.png')
imagex = 360
imagey = 260
direction = 'left'

# text setting
font_obj = pygame.font.Font('freesansbold.ttf', 32)
text_surface_obj = font_obj.render('Hello World!', True, GREEN, BLUE)
text_rect_obj = text_surface_obj.get_rect()
text_rectObj.center = (200, 150)

while True: # the main game loop
    screen.fill(WHITE)

    # draw a green polygon onto the surface
    pygame.draw.polygon(screen, green, ((146, 0), (291, 106), (236, 277), (56, 277), (0,
106)))

    # draw some blue lines onto the surface
    pygame.draw.line(screen, blue, (60, 60), (120, 60), 4)
    pygame.draw.line(screen, blue, (120, 60), (60, 120))
```

```
    pygame.draw.line(screen, blue, (60, 120), (120, 120), 4)

    # draw a blue circle onto the surface
    pygame.draw.circle(screen, blue, (300, 50), 20, 0)

    # draw a red ellipse onto the surface
    pygame.draw.ellipse(screen, red, (100, 150, 40,80), 1)

    # draw a red rectangle onto the surface
    pygame.draw.rect(screen,red, (200, 150, 100, 50))

    # draw the text onto the surface
    screen.blit(text_surface_obj, text_rect_obj)


    #the animation of the image
    if direction == 'right':
        imagex += 5
        if imagex == 360:
            direction = 'down'
    elif direction == 'down':
        imagey += 5
        if imagey == 260:
            direction = 'left'
    elif direction == 'left':
        imagex -= 5
        if imagex == 20:
            direction = 'up'
    elif direction == 'up':
        imagey -= 5
        if imagey == 20:
            direction = 'right'
    screen.blit(image, (imagex, imagey))

    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()

    pygame.display.update()
    fpsClock.tick(FPS)
```

dibujando el fondo blanco:

```
screen.fill(white)
```

dibujando el polígono:

En esta función, define la superficie de visualización, el color y la posición de cada esquina del polígono, puede hacerlo en el sentido de las agujas del reloj y en el sentido contrario a las agujas del reloj.

```
pygame.draw.polygon(screen, green, ((146, 0), (291, 106), (236, 277), (56, 277), (0, 106)))
```

dibujando las líneas:

Aquí se define la superficie de visualización, el color, el primer y último punto y el ancho de la línea.

```
pygame.draw.line(screen, blue, (60, 60), (120, 60), 4)
pygame.draw.line(screen, blue, (120, 60), (60, 120))
pygame.draw.line(screen, blue, (60, 120), (120, 120), 4)
```

dibujando el circulo

En esta función, se define la superficie de visualización, el color, la posición, el radio y el ancho del círculo (0 da un círculo plano).

```
pygame.draw.circle(screen, blue, (300, 50), 20, 0)
```

dibujando la elipse:

En esta función se define la superficie de visualización, el color, la posición, el tamaño horizontal, el tamaño vertical y el ancho de la elipse.

```
pygame.draw.ellipse(screen, red, (100, 150, 40,80), 1)
```

dibujando el rectángulo:

En esta función, se define la superficie de visualización, el color, la posición y el tamaño vertical y horizontal del rectángulo.

```
pygame.draw.rect(screen,red, (200, 150, 100, 50))
```

definiendo el texto:

Primero define el tipo y el tamaño de su texto, uso una fuente básica que obtiene con pygame.

```
font_obj = pygame.font.Font('freesansbold.ttf', 32)
```

Luego define el texto real, si lo quiere en negrita o no (Verdadero / Falso), el color del texto y, si desea marcar su texto, un color de la marca.

```
text_surface_obj = font_obj.render('Hello World!', True, green, blue)
```

Si quieres marcar tu texto o quieres definir el centro de tu texto, debes decirle a pygame que con esta función:

```
text_rect_obj = text_surface_obj.get_rect()
```

Y después de eso, puedes definir el centro de tu texto con esta función:

```
text_rect_obj.center = (200, 150)
```

dibujando el texto:

Si marcó su texto o definió el centro, debe dibujar el texto de esta manera:

```
screen.blit(text_surface_obj, text_rectObj)
```

De lo contrario, dibuja su texto, pero necesita definir la posición, por lo que lo hace de esta manera:

```
DISPLAYSURF.blit(textSurfaceObj, (100,50))
```

definiendo la imagen:

Aquí define la imagen que desea utilizar, la posición de inicio (coordenadas x e y) y la dirección de la imagen.

```
image  = pygame.image.load('image.png')
imagex = 360
imagey = 260
direction = 'left'
```

animando la imagen:

Aquí verifica la dirección de la imagen, si llegó a una esquina, si es así cambia la dirección, si no, muévala 5 píxeles en la misma dirección y dibuje la imagen nuevamente. Eso es lo que hacemos con esta parte del código:

```
if direction == 'right':
    imagex += 5
    if imagex == 360:
        direction = 'down'
elif direction == 'down':
    imagey += 5
    if imagey == 260:
        direction = 'left'
elif direction == 'left':
```

```
    imagex -= 5
    if imagex == 20:
        direction = 'up'
elif direction == 'up':
    imagey -= 5
    if imagey == 20:
        direction = 'right'
screen.blit(image, (imagex, imagey))
```

nota: mi imagen es de 20 por 20 píxeles, usé `if imagex == 360` **y** `if imagey == 260:` **porque mi imagen está a 20 píxeles desde el borde, si su imagen tiene un tamaño diferente, tendrá que cambiar los números .**

comprobando si abandonas el programa:

Aquí comprobamos si cerraste la ventana de tu programa.

```
for event in pygame.event.get():
    if event.type == QUIT:
        pygame.quit()
        sys.exit()
```

actualizando la pantalla:

Aquí le dice a pygame que actualice la pantalla para que todo lo que ha dibujado aparezca en la pantalla.

```
pygame.display.update()
```

definiendo los cuadros por segundo:

Aquí le dices a Pygame que duerma lo suficiente para que se respete la configuración de cuadros por segundo.

```
fpsClock.tick(FPS)
```

Lea Dibujo en la pantalla en línea: https://riptutorial.com/es/pygame/topic/6287/dibujo-en-la-pantalla

Capítulo 7: Dibujo en la pantalla

Sintaxis

- pygame.draw.rect (Superficie, color, Rect, ancho = 0)
- pygame.draw.polygon (Superficie, color, lista de puntos, ancho = 0)
- pygame.draw.circle (Superficie, color, pos, radio, ancho = 0)
- pygame.draw.ellipse (Superficie, color, Rect, ancho = 0)
- pygame.draw.arc (Superficie, color, Rect, start_angle, stop_angle, width = 1)
- pygame.draw.line (Surface, color, start_pos, end_pos, width = 1)
- pygame.draw.lines (Superficie, color, cerrado, lista de puntos, ancho = 1)
- pygame.draw.aaline (Surface, color, startpos, endpos, blend = 1)
- pygame.draw.aalines (Surface, color, closed, pointlist, blend = 1)

Parámetros

Parámetros	Detalles
Superficie	La superficie para dibujar la forma.
color	Una secuencia de 3 o 4 enteros que representan rojo, verde y azul (y alfa), cada valor oscila entre 0-255.
Rect	Un área rectangular donde se dibujará la forma.
anchura	El ancho de las líneas. La forma se rellenará si ancho = 0.
lista de puntos	Una lista de una cantidad arbitraria de puntos / vértices, en píxeles (x, y).
pos	La posición del centro del círculo, en píxeles (x, y).
radio	El radio de los círculos en píxeles.
cerrado	Si es verdadero, se dibujará una línea entre el último y el primer punto, cerrando la forma.
mezcla = 1	Si es verdadero, los tonos se mezclarán con los tonos de píxeles existentes en lugar de sobrescribirlos.
ángulo inicial	El ángulo inicial del arco, en radianes.
stop_angle	El ángulo final del arco, en radianes.
start_pos	La posición inicial de la línea, en píxeles.
end_pos	La posición final de la línea, en píxeles.

Examples

Dibujando con el módulo de dibujo.

Pygame tiene un módulo, `pygame.draw` , que contiene funciones que pueden dibujar formas directamente en una superficie.

Función	Descripción
pygame.draw.rect	dibujar una forma de rectángulo
pygame.draw.polygon	Dibuja una forma con cualquier número de lados.
pygame.draw.circle	dibujar un círculo alrededor de un punto
pygame.draw.ellipse	dibujar una forma redonda dentro de un rectángulo
pygame.draw.arc	dibujar una sección parcial de una elipse
pygame.draw.line	dibujar un segmento de línea recta
pygame.draw.lines	dibujar múltiples segmentos de línea contiguos
pygame.draw.aaline	dibujar líneas finas antialias
pygame.draw.aalines	dibujar una secuencia conectada de lineas antialias

Cómo utilizar el módulo

Para usar el módulo, primero debe importar e inicializar pygame correctamente y configurar un modo para la pantalla. Es conveniente definir de antemano las constantes de color, haciendo que su código sea más legible y hermoso. Todas las funciones toman una superficie para dibujar, un color y un argumento de posición que es un Recto de pygame o una secuencia de entero / flotante de 2 elementos (el `pygame.draw.circle` solo tomará enteros debido a razones indefinidas).

Ejemplo

El código a continuación mostrará todas las diferentes funciones, cómo se usan y cómo se ven. Inicializaremos pygame y definiremos algunas constantes antes de los ejemplos.

```
import pygame
from math import pi
pygame.init()

screen = pygame.display.set_mode((100, 100))
WHITE = pygame.Color(255, 255, 255)
RED = pygame.Color(255, 0, 0)
```

El color negro es el color predeterminado de la superficie y representa la parte de la superficie sobre la que no se ha dibujado. Los parámetros de cada función se explican a continuación en **Parámetros** .

Rect

```
size = (50, 50)

rect_border = pygame.Surface(size)  # Create a Surface to draw on.
pygame.draw.rect(rect_border, RED, rect_border.get_rect(), 10)  # Draw on it.

rect_filled = pygame.Surface(size)
pygame.draw.rect(rect_filled, RED, rect_filled.get_rect())
```

Polígono

```
size = (50, 50)
points = [(25, 0), (50, 25), (25, 50), (0, 25)]  # The corner points of the polygon.

polygon = pygame.Surface(size)
pygame.draw.polygon(polygon, RED, points, 10)

polygon_filled = pygame.Surface(size)
pygame.draw.polygon(polygon_filled, RED, points)
```

Circulo

```
size = (50, 50)
radius = 25

circle = pygame.Surface(size)
pygame.draw.circle(circle, RED, (radius, radius), radius, 10)  # Position is the center of the
circle.

circle_filled = pygame.Surface(size)
pygame.draw.circle(circle_filled, RED, (radius, radius), radius)
```

Los agujeros son una consecuencia desafortunada del algoritmo de dibujo de pygame.

Elipse

```
size = (50, 25)  # Minimize it's height so it doesn't look like a circle.

ellipse = pygame.Surface(size)
pygame.draw.ellipse(ellipse, RED, ellipse.get_rect(), 5)

ellipse_filled = pygame.Surface(size)
pygame.draw.ellipse(ellipse_filled, RED, ellipse.get_rect())
```

Los agujeros son una consecuencia desafortunada del algoritmo de dibujo de pygame.

Arco

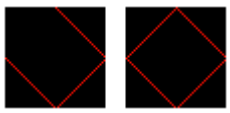
```
size = (50, 50)

arc = pygame.Surface(size)
pygame.draw.arc(arc, RED, arc.get_rect(), 0, pi)  # 0 to pi is 180° creating a half circle.
```

Línea

```
size = (50, 50)

line = pygame.Surface(size)
pygame.draw.line(line, RED, (0, 0), (50, 50))  # Start at topleft and ends at bottomright.
```

Líneas

```
size = (50, 50)
points = [(25, 0), (50, 25), (25, 50), (0, 25)]

lines = pygame.Surface(size)
pygame.draw.lines(lines, RED, False, points)

lines_closed = pygame.Surface(size)
pygame.draw.lines(lines_closed, RED, True, points)
```

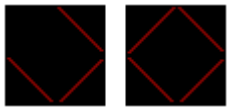


Línea de antialias

```
size = (50, 50)

antialiased_line = pygame.Surface(size)
pygame.draw.aaline(antialiased_line, RED, (0, 0), (50, 50))
```

Líneas suavizadas

```
size = (50, 50)
points = [(25, 0), (50, 25), (25, 50), (0, 25)]

antialiased_lines = pygame.Surface(size)
pygame.draw.aalines(antialiased_lines, RED, False, points)

antialiased_lines_closed = pygame.Surface(size)
pygame.draw.aalines(antialiased_lines_closed, RED, True, points)
```



Pruébalo

Para probarlo por ti mismo: copie uno de los fragmentos de código de arriba y el código de abajo en un archivo vacío, cambie la *imagen* del nombre al nombre de la superficie que desea mezclar y experimentar.

```
import pygame
from math import pi
pygame.init()

screen = pygame.display.set_mode((100, 100))
WHITE = pygame.Color(255, 255, 255)
RED = pygame.Color(255, 0, 0)

# But code snippet here

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            quit()

    screen.blit(image, (25, 25))
    pygame.display.update()
```

Superficies

En pygame, usualmente utilizas Superficies para representar la apariencia de los objetos, y Rectángulos para representar sus posiciones. Una superficie es como una hoja de papel en blanco que contiene colores o imágenes. Hay dos formas de crear una superficie: en blanco desde cero o cargando una imagen.

Crear una superficie

Para crear una superficie, necesita un tamaño mínimo, que es una secuencia entera de ancho y alto de 2 elementos, que representa el tamaño en píxeles.

También puede pasar argumentos adicionales al crear una Superficie para controlar la profundidad de bits, máscaras y funciones adicionales como alfa por píxel y / o crear la imagen en la memoria de video. Esto está fuera del alcance de este ejemplo sin embargo.

```
size = width, height = (32, 32)
empty_surface = pygame.Surface(size)
```

Puede usar el módulo `pygame.draw` para dibujar formas en la superficie, o rellénelo con un color llamando al método de `fill(color)` la superficie `fill(color)` . El *color del* argumento es una secuencia de enteros de 3 o 4 elementos o un objeto `pygame.Color` .

Cargar una imagen

La mayoría de las veces, le gustaría usar sus propias imágenes en un juego (llamadas sprites). Crear una superficie con tu imagen es tan fácil como:

```
my_image = pygame.image.load(path_to_image)
```

El camino a la imagen puede ser relativo o absoluto. Para mejorar el rendimiento, generalmente es aconsejable convertir su imagen al mismo formato de píxeles que la pantalla. Esto se puede hacer llamando al método de Surface `convert()` , así:

```
my_image = pygame.image.load(path_to_image).convert()
```

Si su imagen contiene transparencias (valores alfa) simplemente llame al método `convert_alpha()` lugar:

```
my_image = pygame.image.load(path_to_image).convert_alpha()
```

Blitting

Las superficies deben estar marcadas en la pantalla para poder mostrarlas. Fundir esencialmente significa copiar píxeles de una superficie a otra (la pantalla también es una superficie). También debe pasar la posición de la superficie, que debe ser una secuencia de enteros de 2 elementos o un objeto Rect. El tope de la superficie se colocará en la posición.

```
screen.blit(my_image, (0, 0))
pygame.display.update()  # or pygame.display.flip()
```

Es posible hacer blit a otras superficies que no sean la pantalla. Para mostrar lo que se ha borrado en la pantalla, debe llamar a `pygame.display.update()` o `pygame.display.flip()` .

Transparencia

Hay tipos 3 de transparencia admitidos en pygame: colorkeys, alfas de superficie y alfas de píxel.

Colorkeys

Hace que un color sea completamente transparente, o más exactamente, haciendo que un color simplemente no se vea. Si tiene una imagen con un rectángulo negro en su interior, puede establecer una combinación de colores para evitar que el color negro se vea borroso.

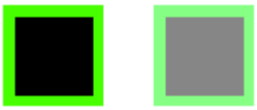
```
BLACK = (0, 0, 0)
my_image.set_colorkey(BLACK)  # Black colors will not be blit.
```

Una superficie solo puede tener un colorkey. Configurar otro colorkey sobrescribirá el anterior. Colorkeys no puede tener valores alfa diferentes, solo puede hacer que un color no sea visible.

Alfas de superficie

Hace que toda la superficie sea transparente por un valor alfa. Con este método, puede tener diferentes valores alfa, pero afectará a toda la superficie.

```
my_image.set_alpha(100)  # 0 is fully transparent and 255 fully opaque.
```



Alfa por píxel

Hace que cada píxel de la superficie sea transparente por un valor alfa individual. Esto te da la mayor libertad y flexibilidad, pero también es el método más lento. Este método también requiere que la Superficie se cree como una Superficie alfa por píxel, y los argumentos de color deben contener un cuarto entero alfa.

```
size = width, height = (32, 32)
my_image = pygame.Surface(size, pygame.SRCALPHA)  # Creates an empty per-pixel alpha Surface.
```

La superficie ahora dibujará transparencia si el color contiene el cuarto valor alfa.

```
BLUE = (0, 0, 255, 255)
pygame.draw.rect(my_image, BLUE, my_image.get_rect(), 10)
```

A diferencia de las otras Superficies, este color predeterminado de la Superficie no será negro sino transparente. Es por eso que el rectángulo negro en el medio desaparece.

Combina colorkey y superficie alfa

Los alfas de Colorkey y Superficie pueden combinarse, pero no el alfa por píxel. Esto puede ser útil si no desea el rendimiento más lento de una superficie por píxel.

```
purple_image.set_colorkey(BLACK)
purple_image.set_alpha(50)
```

Codigo completo

Copie esto en un archivo vacío y ejecútelo. Presione las teclas 1, 2, 3 o 4 para que aparezcan las imágenes. Presione 2, 3 o 4 varias veces para hacerlos más opacos.

```
import pygame
pygame.init()

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255, 50)  # This color contains an extra integer. It's the alpha value.
PURPLE = (255, 0, 255)

screen = pygame.display.set_mode((200, 325))
screen.fill(WHITE)  # Make the background white. Remember that the screen is a Surface!
clock = pygame.time.Clock()

size = (50, 50)
red_image = pygame.Surface(size)
green_image = pygame.Surface(size)
blue_image = pygame.Surface(size, pygame.SRCALPHA)  # Contains a flag telling pygame that the
Surface is per-pixel alpha
purple_image = pygame.Surface(size)

red_image.set_colorkey(BLACK)
green_image.set_alpha(50)
# For the 'blue_image' it's the alpha value of the color that's been drawn to each pixel that
```

```
determines transparency.
purple_image.set_colorkey(BLACK)
purple_image.set_alpha(50)

pygame.draw.rect(red_image, RED, red_image.get_rect(), 10)
pygame.draw.rect(green_image, GREEN, green_image.get_rect(), 10)
pygame.draw.rect(blue_image, BLUE, blue_image.get_rect(), 10)
pygame.draw.rect(purple_image, PURPLE, purple_image.get_rect(), 10)

while True:
    clock.tick(60)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            quit()
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_1:
                screen.blit(red_image, (75, 25))
            elif event.key == pygame.K_2:
                screen.blit(green_image, (75, 100))
            elif event.key == pygame.K_3:
                screen.blit(blue_image, (75, 175))
            elif event.key == pygame.K_4:
                screen.blit(purple_image, (75, 250))

    pygame.display.update()
```

Lea Dibujo en la pantalla en línea: https://riptutorial.com/es/pygame/topic/7079/dibujo-en-la-pantalla

Capítulo 8: Lo escencial

Examples

Dibujo y una animación básica.

Este programa dibuja algunas formas y ' *hola mundo!* 'y dejar que una imagen vaya a cada esquina de la ventana.

el código completo:

```
import pygame,sys
from pygame.locals import *

pygame.init()

FPS = 30 #frames per second setting
fpsClock = pygame.time.Clock()

#set up the window
screen = pygame.display.set_mode((500,400), 0, 32)
pygame.display.set_caption('drawing')

#set up the colors
black = (  0,   0,   0)
white = (255, 255, 255)
red   = (255,   0,   0)
green = (  0, 255,   0)
blue  = (  0,   0, 255)

imageImg  = pygame.image.load('baddie.png')
imagex = 320
imagey = 220
direction = 'left'

fontObj = pygame.font.Font('freesansbold.ttf', 32)
text = fontObj.render('Hello World!', True, green, blue)
rect = text.get_rect()
rect.center = (200, 150)

# the main game loop
while True:
    screen.fill(white)

    # draw a green polygon onto the surface
    pygame.draw.polygon(screen, green, ((146, 0), (291, 106), (236, 277), (56, 277), (0,
106)))

    # draw some blue lines onto the surface
    pygame.draw.line(screen, blue, (60, 60), (120, 60), 4)
    pygame.draw.line(screen, blue, (120, 60), (60, 120))
    pygame.draw.line(screen, blue, (60, 120), (120, 120), 4)

    # draw a blue circle onto the surface
```

```
    pygame.draw.circle(screen, blue, (300, 50), 100, 0)

    # draw a red ellipse onto the surface
    pygame.draw.ellipse(screen, red, (300, 250, 80,80), 1)

    # draw a red rectangle onto the surface
    pygame.draw.rect(screen,red, (200, 150, 100, 50))

    # draw the text onto the surface
    screen.blit(text, rect)

    if direction == 'right':
        imagex += 5
        if imagex == 320:
            direction = 'down'
    elif direction == 'down':
        imagey += 5
        if imagey == 220:
            direction = 'left'
    elif direction == 'left':
        imagex -= 5
        if imagex == 20:
           direction = 'up'
    elif direction == 'up':
        imagey -= 5
        if imagey == 20:
           direction = 'right'
    screen.blit(imageImg, (imagex, imagey))

for event in pygame.event.get():
    if event.type == QUIT:
        pygame.quit()
        sys.exit()

pygame.display.update()
fpsClock.tick(FPS)
```

configurando pygame y la ventana:

```
import pygame,sys
from pygame.locals import *

pygame.init()

#set up the window
screen = pygame.display.set_mode((500,400), 0, 32)
pygame.display.set_caption('drawing')
```

dibujando el fondo blanco:

En esta función se define el color del fondo.

```
screen.fill(white)
```

dibujando el polígono verde:

Aquí define la superficie de visualización, el color y la posición de cada esquina del polígono (coordenadas x e y), puede hacerlo en sentido horario y antihorario.

```
pygame.draw.polygon(screen, green, ((146, 0), (291, 106), (236, 277), (56, 277), (0, 106)))
```

dibujando las lineas azules:

En esta función, se define la superficie de visualización, el color, el primer y último punto y el ancho de la línea (si no da un ancho, es solo 1).

```
pygame.draw.line(screen, blue, (60, 60), (120, 60), 4)
pygame.draw.line(screen, blue, (120, 60), (60, 120))
pygame.draw.line(screen, blue, (60, 120), (120, 120), 4)
```

dibujando el circulo azul:

En esta función, define la superficie de visualización, el color, la posición, el radio y el ancho del círculo (si da un 0 para el ancho, es un círculo plano).

```
pygame.draw.circle(screen, blue, (300, 50), 100, 0)
```

dibujando la elipse:

En esta función usted define la superficie de visualización, el color, la posición, el tamaño horizontal y el tamaño vertical y el ancho.

```
pygame.draw.ellipse(screen, red, (300, 250, 80,80), 1)
```

dibujando el rectángulo:

En esta función, se define la superficie de visualización, el color, la posición y el tamaño horizontal y vertical.

```
pygame.draw.rect(screen,red, (200, 150, 100, 50))
```

definiendo el texto:

Primero, defina el tipo y el tamaño de su texto con esta función:

```
fontObj = pygame.font.Font('freesansbold.ttf', 32)
```

A continuación, defina el texto real, si el texto está en negrita, el color y, si lo desea, el color de la marca. Puedes hacer eso con esta función:

```
text = fontObj.render('Hello World!', True, green, blue)
```

Si quieres marcar tu texto, debes decirle a pygame que con esta función:

```
rect = text.get_rect()
```

Y si desea definir la posición del centro del texto, puede hacerlo con esta función:

```
rect.center = (200, 150)
```

dibujando el texto:

Si ha definido una marca y / o el centro:

```
screen.blit(text, rect)
```

De lo contrario, debe definir la posición del texto, así que dibuje el texto de esta manera:

```
screen.blit(text, (100,50))
```

definiendo la imagen:

Aquí usted define qué imagen desea usar (si lo hace de esta manera, el archivo de imagen debe estar en el mismo directorio que el archivo de su programa), la posición de inicio (xey) y la dirección de la imagen.

```
image  = pygame.image.load('image.png')
baddiex = 320
baddiey = 220
direction = 'left'
```

animando la imagen:

Con esta parte del código verificamos la dirección de la imagen, si llegó a una esquina, si es así, cambiamos la dirección, si no, dibujamos la imagen 5 píxeles más en la misma dirección.

```
if direction == 'right':
    imagex += 5
    if imagex == 360:
        direction = 'down'
elif direction == 'down':
    imagey += 5
    if imagey == 260:
        direction = 'left'
elif direction == 'left':
    imagex -= 5
    if imagex == 20:
       direction = 'up'
elif direction == 'up':
    imagey -= 5
    if imagey == 20:
       direction = 'right'
screen.blit(imageImg, (imagex, imagey))
```

nota: mi imagen es de 20x20 píxeles, la uso `if imagex == 360:` y `if imagey == 260:` porque mi imagen está a 20 píxeles desde el borde de la ventana, al igual que las otras 2 esquinas. Si su imagen tiene un tamaño diferente, probablemente tendrá que cambiar esos números.

comprobando para dejar de fumar:

Aquí verificamos si cerró la ventana de pygame, y si es así, cierre la ventana, si no escribe esto en algún lugar de su programa, probablemente no podrá cerrar la ventana.

```
for event in pygame.event.get():
    if event.type == QUIT:
        pygame.quit()
        sys.exit()
```

actualizando la pantalla:

Con esta función usted actualiza la pantalla para que todo lo que ha dibujado se haga visible.

```
pygame.display.update()
```

Ajuste de FPS:

Con esta función le dices a Pygame que duerma lo suficiente para que se respete tu configuración de FPS.

```
fpsClock.tick(FPS)
```

Usando con PIL

Cuando tienes que usar tanto PIL como Pygame porque faltan funcionalidades en ambos, necesitas una forma de convertir entre Pygame Surfaces y PIL Images, preferiblemente sin escribirlas en el disco.

Para eso puede usar las funciones "tostring" y "fromstring" que se proporcionan en ambas bibliotecas.

Conversión de PIL a Pygame:

```
strFormat = 'RGBA'
raw_str = image.tostring("raw", strFormat)
surface = pygame.image.fromstring(raw_str, image.size, strFormat)
```

Conversión de Pygame a PIL:

```
strFormat = 'RGBA'
raw_str = pygame.image.tostring(surface, strFormat, False)
image = Image.frombytes(strFormat, surface.get_size(), raw_str)
```

Lea Lo escencial en línea: https://riptutorial.com/es/pygame/topic/4196/lo-escencial

Capítulo 9: Manejo de eventos

Examples

Bucle de eventos

Pygame registrará todos los eventos del usuario en una cola de eventos que puede recibirse con el código `pygame.event.get()` . Cada elemento de esta cola es un objeto de `Event` y todos tendrán el `type` atributo, que es un número entero que representa qué tipo de evento es. En el módulo de pygame hay constantes enteras predefinidas que representan el tipo. Excepto por este atributo, los eventos tienen atributos diferentes.

Nombre constante	Atributos
DEJAR	ninguna
Evento activo	ganancia, estado
Tecla de abajo	Unicode, clave, mod
TECLA ARRIBA	clave, mod
MOUSEMOCION	pos, rel, botones
MOUSEBUTTONUP	pos, botón
MOUSEBUTTONDOWN	pos, botón
JOYAXISMOTION	alegría, eje, valor
JOYBALLMOTION	alegría, pelota, rel
Alegria	alegría, sombrero, valor
JOYBUTTONUP	alegría botón
JOYBUTTONDOWN	alegría botón
VIDEORESIZAR	tamaño, w, h
Videoexpuesto	ninguna
USEREVENT	código

Ejemplo

Para manejar nuestros eventos, simplemente recorramos la cola, verificamos qué tipo es (con la

ayuda de las constantes predefinidas en el módulo de pygame) y luego realizamos alguna acción. Este código verificará si el usuario ha presionado el botón de cerrar en la esquina superior de la pantalla y, si es así, finalizar el programa.

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        # Close the program any way you want, or troll users who want to close your program.
        raise SystemExit
```

ATENCIÓN : ¡Debes llamar a la cola de eventos regularmente cuando usas pygame! Además de obtener los eventos disponibles, llamar a la cola de eventos también es la forma en que Pygame interactúa con el sistema operativo internamente. Si la cola de eventos no se llama regularmente, su sistema operativo asumirá que su programa ya no funciona correctamente y posiblemente hará que parezca que el programa se bloqueó (en Windows la ventana se vuelve blanca). Si no quieres hacer nada con los eventos, debes llamar a `pygame.event.pump()` cada ciclo del juego para hacer que pygame procese los eventos internamente.

Eventos del teclado

Hay dos tipos de eventos clave en pygame: `KEYDOWN` y `KEYUP` . Estos eventos tienen una `key` atributo que es un número entero que representa una clave en el teclado. El módulo pygame tiene constantes enteras predefinidas que representan todas las claves comunes. Las constantes se nombran con una `K` mayúscula, un guión bajo y el nombre de la clave. Por ejemplo, `<-` se llama `K_BACKSPACE` , `a` se llama `K_a` y `F4` se denomina `K_F4` .

Ejemplo

Este código verificará si el usuario ha presionado `w` , `a` , `s` o `d` .

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:  # Usually wise to be able to close your program.
        raise SystemExit
    elif event.type == pygame.KEYDOWN:
        if event.key == pygame.K_w:
            print("Player moved up!")
        elif event.key == pygame.K_a:
            print("Player moved left!")
        elif event.key == pygame.K_s:
            print("Player moved down!")
        elif event.key == pygame.K_d:
            print("Player moved right!")
```

Modificadores

No hay una constante entera para mayúsculas. En su lugar, los eventos clave tienen otro atributo llamado `mod` , que son los modificadores (`shift` , `ctrl` , `alt` etc.) que se presionan simultáneamente como la tecla. El atributo `mod` es un número entero que representa el modificador

que se presiona. El valor entero de cada modificador se almacena en el módulo pygame con el nombre de `KMOD_` y su nombre. Por ejemplo, el `desplazamiento a la izquierda` se llama `KMOD_LSHIFT` , la `pestaña` se llama `KMOD_TAB` y `Ctrl` se llama `KMOD_CTRL` .

Ejemplo

Este código será comprobado si el usuario pulsa `una, desviación a la izquierda` + `a` + o `gorras a.`

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:  # It's still wise to be able to close your program.
        raise SystemExit
    elif event.type == pygame.KEYDOWN:
        if event.key == pygame.K_a:
            if event.mod == 0:  # No modifier.
                print("You pressed 'a'")
            elif event.mod == pygame.KMOD_LSHIFT or event.mod == pygame.KMOD_CAPS:
                print("You pressed 'A'")
            else:
                print("You pressed 'a' with another modifier than right shift or caps.")
```

Eventos del mouse

Hay tres tipos de eventos de mouse en pygame `MOUSEMOTION` , `MOUSEBUTTONDOWN` y `MOUSEBUTTONUP` . Pygame registrará estos eventos cuando se haya establecido un modo de visualización.

`MOUSEMOTION` se recibe cuando el usuario mueve su mouse en la pantalla. Tiene los `buttons` atributos, `pos` y `rel` .

- *`buttons`* es una tupla que representa si los botones del mouse (`left` , `mouse-wheel` , `right`) se presionan o no.
- *`pos`* es la posición absoluta (`x` , `y`) del cursor en píxeles.
- *`rel`* es la posición relativa a la posición anterior (`rel_x` , `rel_y`) en píxeles.

`MOUSEBUTTONDOWN` y **`MOUSEBUTTONUP`** se reciben cuando el usuario presiona o suelta un botón del mouse. Tienen el `button` atributos y `pos` .

- *`button`* es un número entero que representa el botón que se presiona. `1` para el botón izquierdo, 2 para la rueda del ratón y `3` para el botón derecho.
- *`pos`* es la posición absoluta del mouse (`x` , `y`) cuando el usuario presionó el botón del mouse.

Ejemplo

Aquí hay un breve ejemplo que utiliza algunos de los atributos de cada evento del mouse:

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:  # Close your program if the user wants to quit.
```

```
        raise SystemExit
    elif event.type == pygame.MOUSEMOTION:
        if event.rel[0] > 0:  # 'rel' is a tuple (x, y). 'rel[0]' is the x-value.
            print("You're moving the mouse to the right")
        elif event.rel[1] > 0:  # pygame start y=0 at the top of the display, so higher y-
values are further down.
            print("You're moving the mouse down")
    elif event.type == pygame.MOUSEBUTTONDOWN:
        if event.button == 1:
            print("You pressed the left mouse button")
        elif event.button == 3:
            print("You pressed the right mouse button")
    elif event.type == pygame.MOUSEBUTTONUP:
        print("You released the mouse button")
```

Como no hay constantes predefinidas para el atributo de botón del mouse en el módulo de pygame, aquí están los valores para cada uno:

Botón	Valor
Boton izquierdo del raton	1
Botón de la rueda del ratón	2
Botón derecho del mouse	3
Rueda del ratón desplazarse hacia arriba	4
La rueda del mouse se desplaza hacia abajo	5

Al desplazar el botón del mouse se generarán los eventos `pygame.MOUSEBUTTONDOWN` y `pygame.MOUSEBUTTONUP` .

Comprobación del estado

Es posible llamar a funciones desde el módulo `pygame.key` y `pygame.mouse` para recibir el estado de la tecla y el mouse. Sin embargo, no es la forma recomendada de procesar eventos en pygame, ya que existen algunos defectos:

- Recibirá los estados cuando se llame a la función, lo que significa que puede perder eventos entre las llamadas si el usuario está presionando los botones rápidamente.
- No se puede determinar el orden de los eventos.
- Aún debes llamar a una de las funciones de eventos de pygame para que pygame interactúe internamente con el sistema operativo, de lo contrario, advertirá que el programa no responde. Las funciones que puedes llamar son:
    - `pygame.event.get()` para obtener todos los eventos o tipos de eventos (pasando los tipos como un argumento) de la cola.
    - `pygame.event.poll()` para obtener un solo evento de la cola.

- `pygame.event.wait()` para esperar un solo evento de la cola.
- `pygame.event.clear()` para borrar todos los eventos en la cola.
- `pygame.event.pump()` para permitir que pygame maneje acciones internas (las funciones anteriores lo llaman implícitamente).

Eventos del teclado

El módulo de claves tiene una función `pygame.key.get_pressed()` que devuelve una lista del estado de todas las claves. La lista contiene `0` para todas las teclas que no se presionan y `1` para todas las teclas que se presionan. Su índice en la lista está definido por constantes en el módulo de pygame, todos con el prefijo `K_` y el nombre de la clave.

```
pygame.event.pump()  # Allow pygame to handle internal actions.
key = pygame.key.get_pressed()
if key[pygame.K_a]:
    print("You pressed 'a'")
if key[pygame.K_F1]:
    print("You pressed 'F1'")
if key[pygame.K_LSHIFT]:
    print("You pressed 'left shift'")
if key[pygame.K_q]:  # Press 'q' to exit the program
    quit()
```

Si desea verificar una sola pulsación de tecla en lugar de si la tecla se mantiene presionada, puede almacenar el estado anterior de todas las teclas en una variable temporal y verificar si el valor cambia:

```
pygame.event.pump()  # Allow pygame to handle internal actions.
key = pygame.key.get_pressed()
if key[pygame.K_q] and not previous_key[pygame.K_q]:
    print("You pressed 'q'")
if key[pygame.K_p] and not previous_key[pygame.K_p]:
    print("You pressed 'p'")
previous_key = key
```

La instrucción se evalúa como verdadera solo cuando se presiona la tecla actual y no se presiona la tecla anterior. Para verificar si el usuario soltó la clave, solo tiene que cambiar la palabra clave `not` (`if not key[pygame.K_q] and previous_key[pygame.K_q]`). Para que esto funcione correctamente, debes configurar la variable `previous_key = pygame.key.get_pressed()` antes del ciclo del juego, de lo contrario recibirás un `NameError` .

Eventos del mouse

El módulo del mouse tiene funciones que nos permiten verificar y establecer la posición del mouse, así como también verificar los botones que se presionan. La función `pygame.mouse.get_pressed()` devuelve una tupla de tupla que representa si los botones del mouse (izquierda, rueda del mouse, derecha) se presionan o no.

```
pygame.event.pump()  # Allow pygame to handle internal actions.
mouse_pos = pygame.mouse.get_pos()
mouse_buttons = pygame.mouse.get_pressed()
if mouse_pos[0] > 100:
    pygame.mouse.set_pos(10, mouse_pos[1])  # Reset the mouse's x-position to 10.
    print("YOU SHALL NOT PASS!")
if mouse_buttons[2]:
    print("I'm right, right?")
if mouse_buttons[0]:  # Press left mouse button to exit.
    print("Program left")
    quit()
```

Lea Manejo de eventos en línea: https://riptutorial.com/es/pygame/topic/5110/manejo-de-eventos

Creditos

S. No	Capítulos	Contributors
1	Empezando con pygame	Community, elegent, Inazuma, Nearoo, Ni., numbermaniac, svs, Ted Klein Bergman, White Shadow
2	Añadiendo música de fondo y efectos de sonido.	Hikaryu, White Shadow
3	Creando una ventana de pygame	Rishi Malhotra, White Shadow
4	Creando una ventana de pygame simple	ModoUnreal
5	Creando una ventana en pygame - pygame.display.set_mode ()	Nearoo
6	Dibujo en la pantalla	svs
7	Lo escencial	Ni., numbermaniac, svs, Ted Klein Bergman
8	Manejo de eventos	elegent, Mikhail V, Nearoo, Ted Klein Bergman