

Pygame

Autor: Pablo Pizarro, [@ppizarro.com](https://ppizarro.com)

La librería Pygame permite crear aplicaciones 2D y 3D en Python al proveer al usuario de multitud de funciones para dibujar en pantalla, cargar sonidos o reconocer distintos tipos de input (teclado, ratón, controles externos).

El esquema de funcionamiento de cualquier aplicación en Python es típicamente ¹ el siguiente:

Código 1: Ejemplo en Python.

```
1 # Importación de librerías
2 import pygame
3 from pygame.locals import *
4
5 # Se inician módulos
6 pygame.init()
7 os.environ['SDL_VIDEO_CENTERED'] = '1' # Centra la ventana
8
9 ...
10
11 # Se cargan fuentes, sonidos e imágenes
12 sonido_muere = pygame.mixer.Sound('sounds/dead.wav') # Sonidos
13 fuente_juego = pygame.font.Font('font/8bit.ttf', 40) # Fuentes
14 imagen_mono = pygame.image.load('images/imagen1.png') # Imágenes
15 icono_juego = pygame.image.load('images/icon.png')
16
17 ...
18
19 # Se crea la pantalla
20 surface = pygame.display.set_mode((640, 480)) # Crea una ventana de 640 por 480 pixeles,
    retorna el CANVAS para dibujar en pantalla
21 pygame.display.set_caption('Titulo del juego') # Título de la ventana
22 pygame.display.set_icon(icono_juego) # Ícono de la ventana
23
24 ...
25
26 # Se crea el reloj del juego
27 clock = pygame.time.Clock()
28
29 ...
```

¹ En el sentido que, a pesar que pueden haber variaciones, a *Grosso modo* presenta el esquema presentado.

```

30
31 # Se crean los modelos
32 jugador1 = ModeloA(...)
33 jugador2 = ModeloB(...)
34 Nube = NubeAleatoria(...)
35
36 ...
37
38 # Bucle principal
39 while True:
40
41     # Setea el reloj
42     clock.tick(FPS)
43
44     # Busca eventos de aplicación
45     for event in pygame.event.get():
46         if event.type == QUIT: # Cierra la aplicación
47             exit()
48         ...
49
50     # Ve teclas pulsadas e interactúa con los modelos
51     keys = pygame.key.get_pressed()
52     if keys[K_w]:
53         jugador1.subir()
54         ...
55
56     # Los modelos interactúan entre sí, etc
57     jugador1.colisiona(jugador2)
58     ...
59
60     # Se pinta el fondo
61     surface.fill(COLOR_BLACK) # También se puede dibujar una imagen
62
63     # Se dibujan los modelos
64     jugador1.draw()
65     jugador2.draw()
66     ...
67
68     # Vuelca todo lo dibujado en pantalla
69     pygame.display.flip()

```

Imágenes

Las imágenes (archivos externos como **png**, **gif**) se cargan utilizando la instrucción `pygame.image.load(dir)`. Esta retorna un objeto del tipo **surface** (el cual conceptualmente es como un canvas).

Para dibujar todo lo que es imagen en pantalla se debe volcar el *surface* generado en el **surface** de la pantalla (es como pegar un canvas pequeño en uno más grande), esto se hace con la función **blit**, la cual toma por parámetro el surface de la imagen cargada y una posición (x,y) tal como se muestra a continuación:

Código 2: Pegar imágenes en la pantalla.

```
1 imagen_mono = pygame.image.load('images/imagen1.png')
2 surface.blit(imagen_mono, (30, 60)) # Recordar que surface era el objeto que retornaba al
    crear la ventana, ver línea 20 del código fuente 1
```

En este caso se dibuja la imagen cargada en pantalla en las coordenadas 30, 60 (30 píxeles horizontales y 60 píxeles verticales, considerando el (0,0) como la esquina superior izquierda).

Para ver la lista completa de funciones revisa este enlace: <https://www.pygame.org/docs/ref/image.html>

Líneas, polígonos, etc

Pygame ofrece la sub-librería **pygame.draw** para realizar una variedad de acciones. A continuación se presenta una lista de funciones más comunes:

1. Dibujar un polígono: `pygame.draw.polygon(surface, color, pointlist, width=0)`.
Pointlist es la lista de puntos del polígono, por ejemplo: [(10, 40), (60, 30), (40, 60) ...], **color** es el color del polígono y **width** es el ancho del borde del polígono, si `width=0` entonces el polígono tiene un relleno del color deseado.
2. Dibujar una línea: `pygame.draw.line(surface, color, [x1, y1], [x2, y2], width)`.
En este caso se dibuja una línea recta entre (x1, y1) y (x2, y2) de un color **color** y un ancho **width** (en píxeles).

Para ver la lista completa de funciones revisa este enlace: <https://www.pygame.org/docs/ref/draw.html>

NOTA: Los colores están definidos por una tupla de 3 números de la forma (R, G, B) en donde R indica el valor de la componente roja (0 no hay rojo, 255 es el máximo valor), G es la componente verde y B es la componente azul. Ejemplos:

Código 3: Colores en Pygame.

```
1 COLOR_BLACK = (0, 0, 0)
2 COLOR_WHITE = (255, 255, 255)
3 COLOR_GRAY = (150, 150, 150)
4 COLOR_RED = (255, 0, 0)
5 COLOR_PURPLE = (128, 0, 128)
```

Sonidos

Los sonidos se cargan con la instrucción `pygame.mixer.Sound(dir)`, en donde se retorna un objeto del tipo `Sound`. Cada objeto posee varias funciones, entre las cuales se encuentran:

Código 4: Sonidos en Pygame.

```
1 sonido_muere = pygame.mixer.Sound('sounds/dead.wav')
2 sonido_muere.play(0) # Sonido se reproduce 1 vez
3 sonido_muere.play(1) # Sonido se reproduce 2 veces
4 sonido_muere.play(n) # Sonido se reproduce n+1 veces
5 sonido_muere.play(-1) # Sonido se reproduce infinitas veces
6 sonido_muere.stop() # Sonido se detiene
7 sonido_muere.pause() # Sonido se pausa
```

Para ver la lista completa de funciones revisa este enlace: <https://www.pygame.org/docs/ref/mixer.html>

Fuentes

Para cargar una fuente se ejecuta la siguiente instrucción: `pygame.font.Font(dir, size)` en donde se carga la fuente en la ubicación **dir** con un tamaño de letra de **size** píxeles. Para crear un texto se debe generar su *surface* asociado, esto es, se debe renderizar una imagen de un texto con esa fuente para luego volcar ese *surface* con la función **blit** (ver subcapítulo de imágenes).

Para renderizar un texto se usa la función `font.render(text, antialias, color)` en donde **font** es la fuente cargada, **text** es el texto a renderizar, **antialias** indica si el texto a renderizar tiene *antialiasing* o no (por lo general se deja como 1) y **color** es el color del texto. Esta función retorna un *surface* el cual posteriormente se vuelca en pantalla con la función **blit**.

A continuación se muestra un ejemplo que escribe 'HOLA MUNDO' en el **surface** de la ventana en la posición (100, 500):

Código 5: Fuentes con Python.

```
1 # Se carga la fuente
2 fuente_juego = pygame.font.Font('font/8bit.ttf', 40)
3
4 # Se dibuja un texto
5 text = font.render('HOLA MUNDO', 1, COLOR_WHITE)
6
7 # Se vuelca el texto en pantalla
8 surface.blit(text, (100, 500))
```

Para ver la lista completa de funciones revisa este enlace: <https://www.pygame.org/docs/ref/font.html>