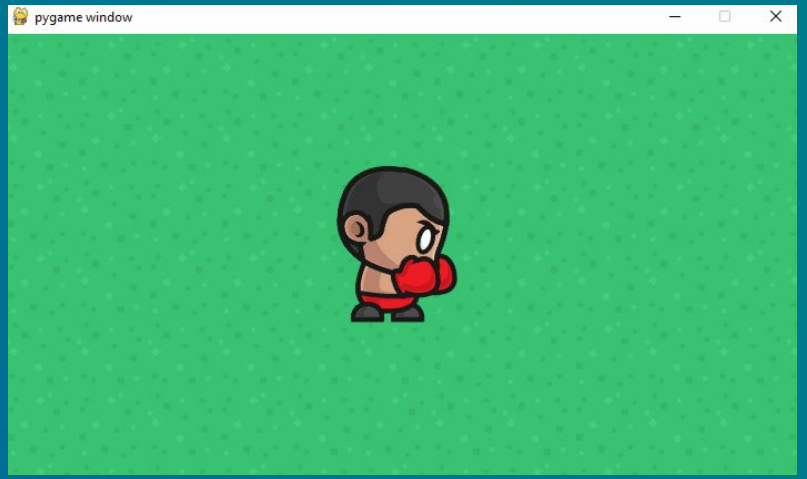
Facundo Falcone - Guevara Mariano - Manuel Samaniego

Biblioteca Pygame

Introducción

Pygame es un contenedor de Python para la biblioteca **SDL**, que significa Simple DirectMedia Layer.

SDL brinda acceso multiplataforma a los componentes de hardware multimedia subyacentes de su sistema, como sonido, video, mouse, teclado y joystick.



Facundo Falcone - Guevara Mariano - Manuel Samaniego

Características

Esta librería te permite:

- Levantar imágenes en formato PNG, BMP, JPG, entre otros.
- Usar sistemas de sonido con audio en formato WAV, OGG, MP3.
- Realizar operaciones relacionadas con el gestor de ventanas.
- Manejar eventos de la aplicación.
- Manejo de dispositivos de entrada tales como mouse y teclado.
- Crear temporizadores para realizar acciones.
- Manejar colisiones entre objetos.
- Manejar un sistema de Sprites para elementos del juego.

Ciclo de vida de un juego

Pensá en un juego como una película que se reproduce constantemente. Cada fotograma es una actualización del estado del juego. En Pygame, esto se logra con un bucle infinito que constantemente revisa si hay eventos (como pulsar una tecla), actualiza la posición de los objetos en pantalla y luego vuelve a dibujar todo.

Estructura básica de un juego en Pygame

1. **Inicialización:** Importamos Pygame, inicializamos todo y creamos la ventana principal.
2. **Bucle principal:** Este bucle se ejecuta mientras el juego esté activo.
3. **Gestionar eventos:** Revisamos si se han producido eventos, como pulsar una tecla o cerrar la ventana.
4. **Actualizar el juego:** Cambiamos las posiciones de los objetos, calculamos físicas, etc.
5. **Dibujar en pantalla:** Dibujamos todos los elementos en la pantalla.
6. **Actualizar la pantalla:** Mostramos lo que hemos dibujado.

Facundo Falcone - Guevara Mariano - Manuel Samaniego

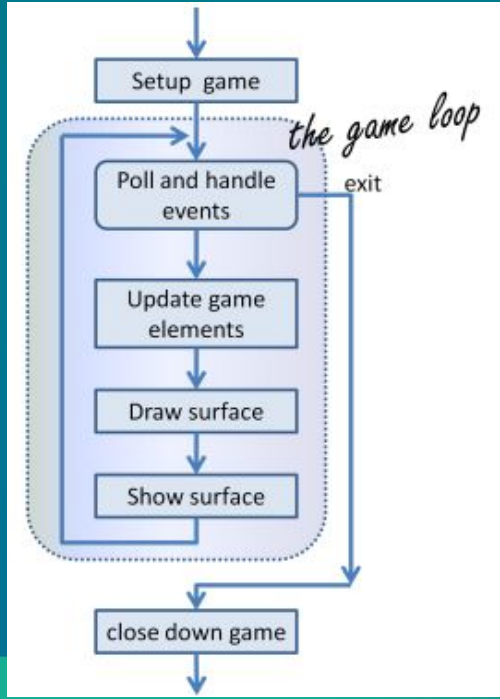
Ejemplo de código

En este ejemplo se importa la librería para luego inicializarla. Como siguiente paso se crea un objeto que se usará para la pantalla o display y luego se gestiona el bucle principal del juego en el cual se evalúan cada uno de los eventos para realizar una acción específica.

```python
import pygame

# Inicializamos Pygame
pygame.init()

# Creamos la ventana principal
pantalla = pygame.display.set_mode((800, 600))

# Bucle principal del juego
while True:
    # Gestionar eventos (por ejemplo, cerrar la ventana)
    for evento in pygame.event.get():
        if evento.type == pygame.QUIT:
            pygame.quit()
            quit()

    # Actualizar el juego (por ahora, nada)

    # Dibujar en pantalla (por ahora, rellenamos de un color)
    pantalla.fill((0, 255, 0))  # Verde

    # Actualizar la pantalla
    pygame.display.update()
```

Ciclo de vida del juego

Luego de que se configura el juego, Se manejan eventos y se actualizan los elementos del juego (por ejemplo: su posición). Luego de eso, se dibujan las superficies en pantalla y se muestran hasta que se ejecute el evento para cerrar el juego.



La Ventana Principal: Nuestro Lienzo

La ventana principal es el lienzo donde vamos a pintar nuestro juego. Es como un marco en el que encajamos todas las imágenes y elementos.

```
pantalla = pygame.display.set_mode((800, 600))
```

El método set_mode() recibe por parámetro una tupla con la dimensión en píxeles de la pantalla de nuestro juego, por ejemplo, si queremos que nuestra ventana sea de 800 de ancho X 600 de alt, utilizaremos:

```
dimension_pantalla = (800, 600)
set_mode(dimension_pantalla)
```

Configuraciones Adicionales

A nuestro display le podemos configurar unos datos extra para mejorar el aspecto de nuestra ventana del juego, algunos de ellos son:

Título: pygame.display.set_caption("Mi Juego")

Icono: pygame.display.set_icon(icono) (**donde ícono es una superficie de Pygame**)

Modo a pantalla completa: pygame.display.set_mode((800, 600), pygame.FULLSCREEN)

Ventana redimensionable: pygame.display.set_mode((800, 600), pygame.RESIZABLE)

Imágenes

El módulo de imagen contiene funciones para cargar y guardar imágenes, así como para **transferir superficies** a formatos que puedan utilizar otros paquetes.

Tenga en cuenta que no existe una clase Image; **una imagen se carga como un objeto Surface**. La clase Surface permite la manipulación (dibujar líneas, configurar píxeles, capturar regiones, etc.).

Para cargar una superficie, utilizamos lo siguiente:

- pygame.image.load
- imagen = pygame.image.load("01.png") Para cargar superficies.
- imagen = pygame.transform.scale(imagen,(ancho,alto)) Para escalar las superficies.

Imágenes

Algunos Módulos:

- pygame.image.load: Nos permite cargar imágenes.
- pygame.image.save: Guarda una imagen a archivo.
- pygame.image.get_sdl_image_version: Obtener el número de versión de la biblioteca SDL_Image que se está utilizando.
- pygame.image.get_extended: Pruebe si se pueden cargar formatos de imagen extendidos.
- pygame.transform.scale(imagen,(ancho,alto)) Nos permite escalar superficies.

Imágenes

Algunos formatos que podés cargar:

- BMP
- GIF (non-animated)
- JPEG
- LBM (and PBM, PGM, PPM)
- PCX
- PNG
- PNM
- SVG (limited support, using Nano SVG)
- TGA (uncompressed)
- TIFF
- WEBP
- XPM

Imágenes

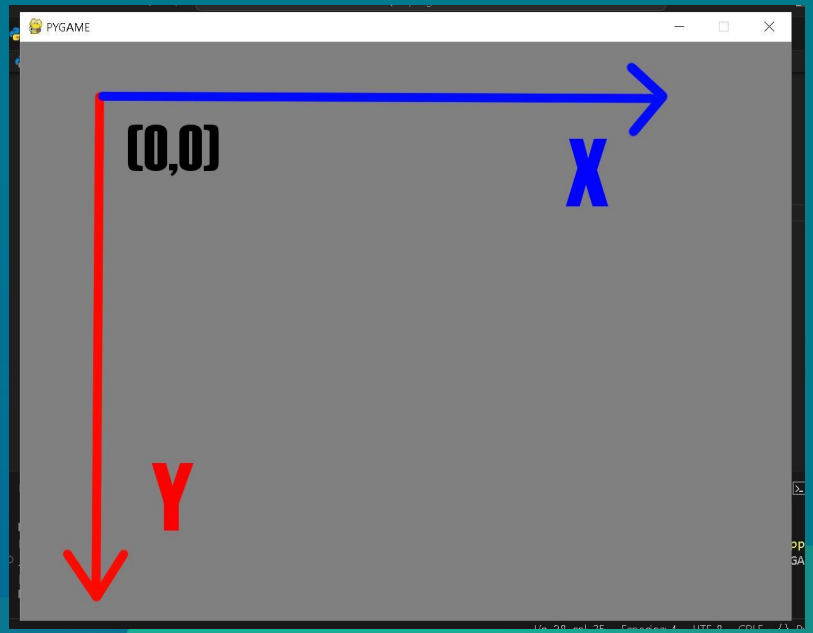
Solo se pueden guardar en los siguientes formatos:

- BMP
- JPEG
- PNG
- TGA

Posicionamiento

-La pantalla de un juego en pygame se representa como una cuadrícula donde cada punto tiene un valor, como un plano cartesiano.

-La esquina superior izquierda de la pantalla tiene valor (0,0)
. X: representa la posición horizontal
. Y: representa la posición vertical

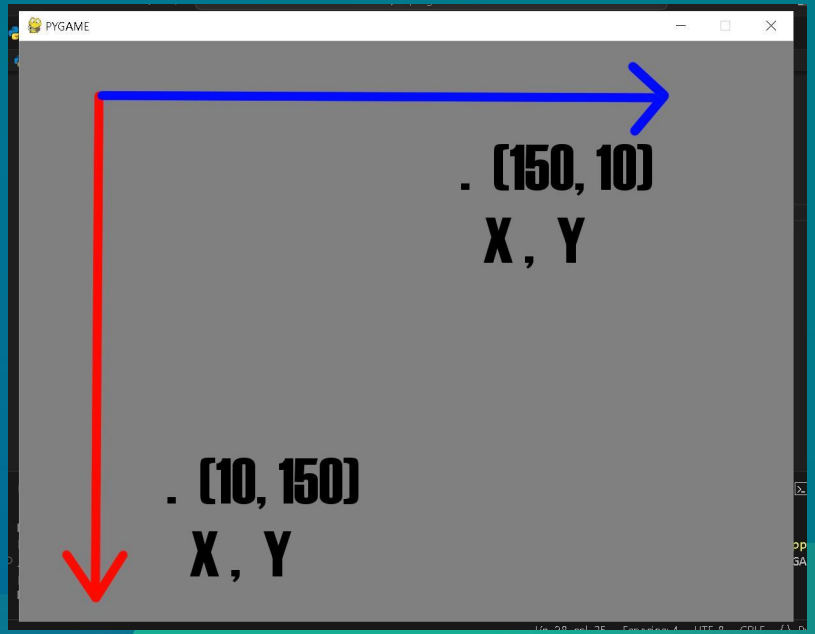


Posicionamiento

-A medida que aumentas el valor de X, te desplazas hacia la derecha

-A medida que aumentas el valor de Y, te desplazas hacia abajo

-Cada punto en pantalla es una intersección entre valores de los ejes (X,Y)



Figuras Geométricas

Sobre una superficie se pueden dibujar diferentes formas (círculos, rectángulos, elipses, líneas). Estas figuras son superficies en sí y para crearlas tendremos que tener en cuenta ciertos argumentos:

Para crear un Rectángulo:

```
color_verde = (0, 255, 0)
coordenadas = (30, 30, 60, 60)
rectangulo = pg.draw.rect(screen, color_verde, coordenadas)
```

Figuras Geométricas

El método **.rect** de pygame nos permite setear una superficie de rectángulo. El primer argumento será la superficie de display de nuestro juego, allí es donde lo dibujaremos, como segundo argumento pondremos el color de nuestro rectángulo, el color tiene que estar expresado como una tupla con formato numérico tal como si se tratase de un RGB (de hecho los números representan eso mismo) siendo, por ejemplo:
ROJO = (255, 0, 0) | VERDE = (0, 255, 0) | AZUL = (0, 0, 255)
Como siguiente argumento pondremos las coordenadas donde queremos que se dibuje dicho rectángulo, debe estar expresado como una tupla de 4 elementos que representan las coordenadas, ejemplo:

coordenadas = (30, 30, 100, 60) siendo:
coordenadas = (EJE_X, EJE_Y, ANCHO_RECTANGULO, ALTO_RECTANGULO)
El alto y ancho está expresado en píxeles.

Figuras Geométricas

En el caso del círculo, es similar a lo que hicimos con el rectángulo.
Sus argumentos serán:

- La pantalla donde lo dibujaremos,
- El color de la figura (en RGB),
- Las coordenadas del centro del círculo,
- El radio del círculo.

```
color_azul = (0, 0, 255)
coordenada_centro = (250, 250)
radio = 75
circulo = pg.draw.circle(screen, color_azul, coordenada_centro, radio)
```

Figuras Geométricas

Así se verán las dos figuras geométricas creadas: