

PYGAME

Conceptos Básicos

¿Qué es Pygame?

¡Abierta y gratuita!

- Una librería que permite trabajar con imágenes y sonido...

↓
¡Animadas!

- ... que gestiona su interacción...

↘ Sprites

- ... y que controla el Hardware.

↗ Bucle
de
Eventos

Esquema General

Importación

Inicialización

Ejecución

Importación

Contiene variables y constantes muy útiles

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

En particular
`sys.exit()`

 El módulo **sys** es importante para trabajar correctamente con el sistema operativo.

Inicialización

Tamaño de la
superficie

- `pygame.init()`
- `visor = pygame.display.set_mode((640, 480), 0, 32)`

Opciones

Profundidad
de color

 **`display.set_mode()`** crea la
superficie de visualización.

Inicialización

- `pygame.init()`
- `visor = pygame.display.set_mode((640, 480), 0, 32)`

¡Aquí es
donde se mostrarán las
animaciones!

📌 **visor** contiene un objeto **Surface** que puede ser una ventana o la pantalla completa.

Inicialización

Opciones

- `pygame.init()`
- `visor = pygame.display.set_mode((640, 480), 0, 32)`



FULLSCREEN Pantalla completa.



NOFRAME Sin bordes ni título.

Puedo poner varias con | de separador



OPENGL Para 3D.



HWSURFACE Acelerada por Hardware.



DOUBLEBUF Doble bufer.



RESIZABLE Tamaño variable.

Inicialización

Profundidad de
color (en bits)

- `pygame.init()`
- `visor = pygame.display.set_mode((640, 480), 0, 32)`



8 256 colores.



15 32 768 colores con transparencia.



16 65 536 colores.



24 16.7 millones de colores.



32 16.7 millones de colores con transparencia.

Ejecución

Responder a las
acciones del jugador

**Bucle
de
Eventos**

Mostrar
imágenes y sonido

Detectar las
incidencias en el juego

FPS
Fotogramas por Segundo

📌 Y debo hacerlo **una y otra vez**, continuamente,
durante todo el juego.

Bucle de Eventos

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

Mirar la lista de eventos

Responder al tipo de evento


from pygame.locals import *

 **Todo** lo que ocurre durante el juego debe estar **aquí**.

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


Aquí se
pondrán el resto de los
procesos...

... y eventos.

 Observa lo
que se está
haciendo...

- Haz sin parar:
- ...
- Mira cada evento pendiente:
- ...
- Si el tipo de evento es salir:
- sal de pygame
- sal del programa

Dibujado en Pantalla

 Para evitar **artefactos** se dibuja **offscreen** y luego se **vuelca** todo en pantalla.

Usando el método
`visor.blit()`

Usando el método
`pygame.display.update()`

¡Muchos fotogramas

producen una

animación...

... si son diferentes!

Frame
Fotograma

En realidad hay más métodos para dibujar **offscreen**...

 **blit()** para imágenes.

 **draw.line()** líneas

 **draw.circle()** círculos

 **draw.ellipse()** elipses

 **draw.rect()** rectángulos

Consulta la
documentación.

 ...

¿Cómo defino Colores y Posiciones?

📌 Colores con tuplas **RGB**.

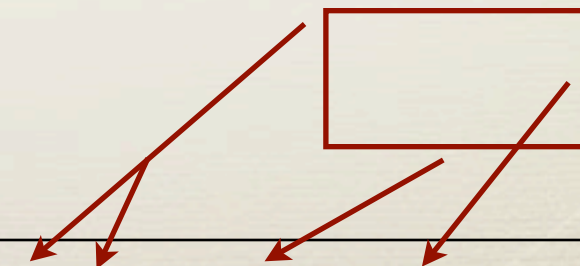
$r \quad g \quad b$
● $\text{colorVerde} = (0, 255, 0)$

📌 Posiciones con tuplas **2D**.

$x \quad y$
● $\text{punto} = (223, 327)$

📌 Y zonas con tuplas **RECT**.

● $\text{zona} = (0, 0, 300, 200)$

A diagram showing a rectangle with its bottom-left corner at the origin (0,0). Four red arrows point from the coordinates in the tuple (0, 0, 300, 200) to the corners of the rectangle: the first arrow points to the bottom-left corner, the second to the bottom-right corner, the third to the top-right corner, and the fourth to the top-left corner.

¿Y cómo dibujar **Texto** ?

 Primero, definimos el **tipo de letra**.

- `tipoLetra = pygame.font.SysFont('arial', 48)`

 Segundo, **generamos** el texto.

- `texto = tipoLetra.render('Hola', True, color1, color2)`

Suavizar

de texto

de fondo

 Y tercero, lo **dibujamos** offscreen.

- `visor.blit(texto, textRect)` ← lugar

¡Todo es un objeto!

- 📌 Siempre podemos acceder a sus **propiedades** o usar sus **métodos**.

Consulta la
Documentación.

● <code>xCentro = visor.get_rect().centerx</code>

● <code>visor.fill((0,0,0))</code>

¿Preguntas?