

# Sprites

Pygame

**pygame**



# ¿Qué es un Sprite?

- Un **sprite** es básicamente una representación gráfica de un objeto en un videojuego, como un personaje, un enemigo o cualquier otro elemento interactivo.

# Características Clave de la Clase Sprite

Cada sprite en Pygame generalmente tiene dos atributos esenciales:

**Image:** Un objeto de superficie (**pygame.Surface**) que almacena la imagen gráfica del sprite.

**Rect:** Un objeto rectángulo (**pygame.Rect**) que define la posición y el tamaño del sprite. Es crucial para el manejo de colisiones y el posicionamiento en pantalla.

- **update()**: Este método se utiliza para actualizar el estado del sprite. Por ejemplo, podrías cambiar su posición, actualizar animaciones, o cambiar su comportamiento. Este método se llama típicamente en cada fotograma del juego.
- **draw(surface)**: Aunque no es un método integrado de la clase Sprite, se utiliza para dibujar el sprite en una superficie. Normalmente, se pasa el atributo image del sprite a la función blit de una superficie

```
import pygame

class MiSprite(pygame.sprite.Sprite):

    def __init__(self, color, width, height):
        super().__init__()

        # Crea una imagen del bloque y rellénala con un color.
        self.image = pygame.Surface([width, height])
        self.image.fill(color)

        # Establece el rectángulo que tiene las dimensiones de la imagen.
        self.rect = self.image.get_rect()

    def update(self):
        # Actualiza la lógica del sprite
        pass
```

```
pygame.init()
screen = pygame.display.set_mode((largo, ancho))

# FPS
fpsClock = pygame.time.Clock()
pygame.display.set_caption (self.caption)

# Crear sprites y grupos
mi_sprite = MiSprite(rojo, 50, 50)
sprites = pygame.sprite.Group()
sprites.add(mi_sprite)
```

## pygame.Color

*pygame object for color representations*

Color(r, g, b) -> Color

Color(r, g, b, a=255) -> Color

Color(color\_value) -> Color

```
running = True

while running:

    fpsClock.tick(self.FPS)

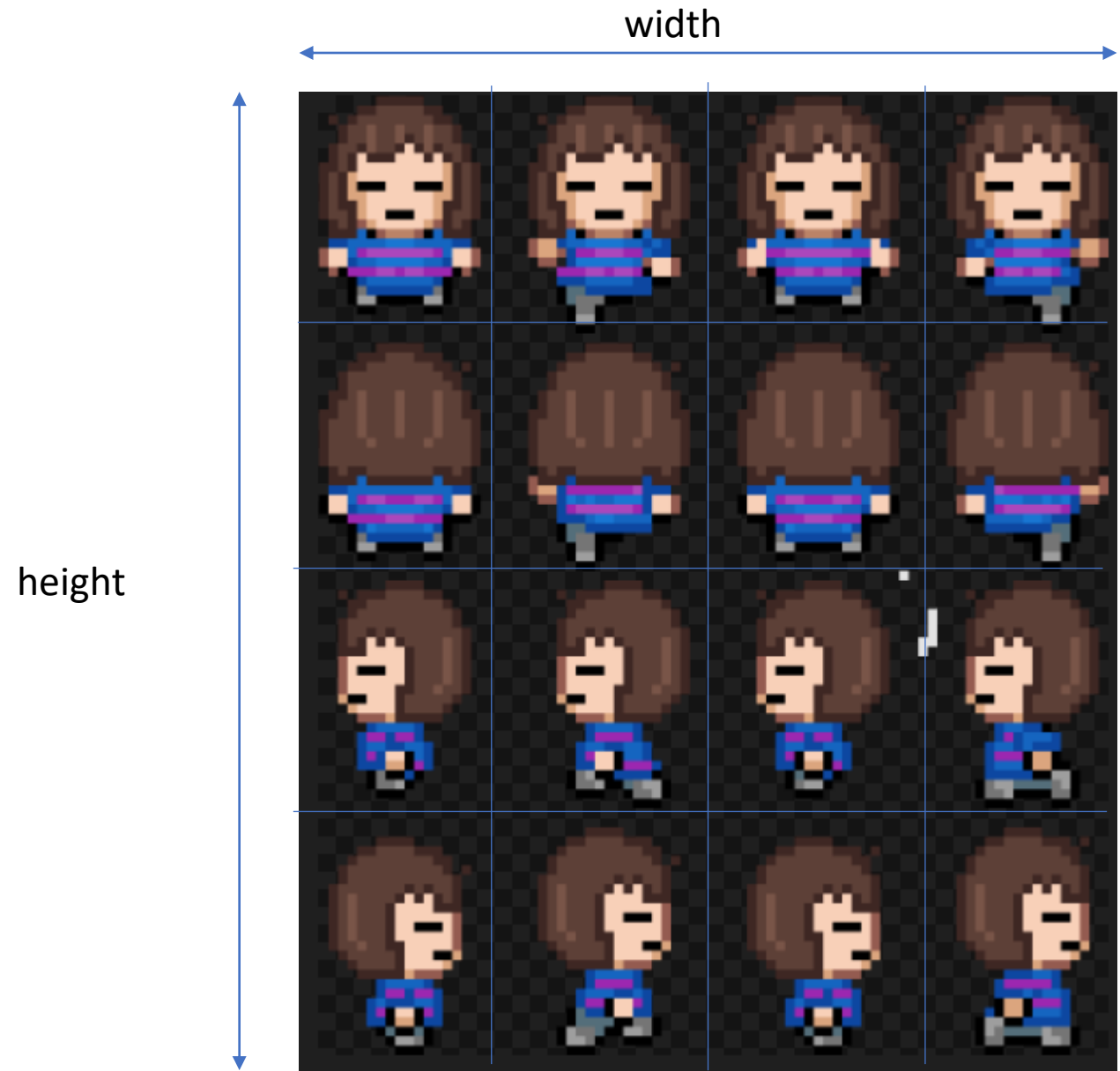
    # En el bucle del juego
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Actualizar sprites
    sprites.update()
    # Dibujar sprites
    screen.fill(negro)
    # Fondo
    sprites.draw(screen)
    # Update the full display Surface to the screen
    pygame.display.flip()

pygame.quit()
sys.exit()
```







# # Obtener la imagen de los sprites de los personajes

```
self.sheets =  
pygame.image.load(filename).convert_alpha()
```

Obtener un rectángulo

```
self.rect = self.sheets.get_rect()
```

## pygame.Rect

*pygame object for storing rectangular coordinates*

`Rect(left, top, width, height) -> Rect`

`Rect((left, top), (width, height)) -> Rect`

`Rect(object) -> Rect`

- ¿Cuántas filas y columnas está dividido mi Sprite?
- 4x4
- ¿Cuánto mide de ancho y de alto?
  - `self.rect.width`
  - `self.rect.height`
- ¿Cuánto mide de ancho y de alto cada celda?



CREAR 4 ARRAYS PARA CADA IMAGEN

# Cada fila es una dirección

- FILA 1 DOWN
- FILA 2 UP
- FILA 3 LEFT
- FILA 4 RIGHT



Esta clase representa la imagen de todas las posiciones del personaje

```
class SpreatSheets:  
    def __init__(self, filename, rows, cols):
```

Con este bucle se recorre la imagen y seleccionamos cada imagen que la añadimos a una lista por fila

```
for row in range(0, rows):  
    for col in range(0, cols):  
        animation = pygame.Rect (w*col, h*row, w, h)
```

## **pygame.Rect**

*pygame object for storing rectangular coordinates*

`Rect(left, top, width, height) -> Rect`

`Rect((left, top), (width, height)) -> Rect`

`Rect(object) -> Rect`



Cada fila se añade con append a una lista y cada lista tendrá 4 imágenes del personaje en esa dirección que son las imágenes de una fila de cada columna

```
for row in range(0, rows):
    for col in range(0, cols):
        animation = pygame.Rect (ancho*col,  alto*row, ancho, alto)
        if (row == 0):
            self.animation_down.append(self.sheets.subsurface(animation))

        if (row == 1):
            self.animation_up.append(self.sheets.subsurface(animation))

        if (row == 2):
            self.animation_left.append(self.sheets.subsurface(animation))

        if (row == 3):
            self.animation_right.append(self.sheets.subsurface(animation))
```

# La clase personaje hereda de Sprite

```
'''
```

```
La idea de esta clase es la del personaje y llama al sreatSheet que contiene  
las imágenes asociadas al personaje en diferentes arrays dependiendo de si  
va hacia la derecha y tenemos una lista, a la izuiqerda y tenemos otra lista  
arriba y abajo
```

```
'''
```

```
class AnimationCharacter (pygame.sprite.Sprite):
```

# Método update de Character

```
def update (self):  
  
    if self.index >= 3:  
        self.index = 0  
    print (self.index)  
  
    if self.direction == "RIGHT":  
        self.image = self.animationRIGHT[self.index]  
    if self.direction == "LEFT":  
        self.image = self.animationLEFT[self.index]  
    if self.direction == "UP":  
        self.image = self.animationUP[self.index]  
    if self.direction == "DOWN":  
        self.image = self.animationDOWN[self.index]
```

Esta clase se llama tantas veces por segundo como refrescos de pantalla es decir fps. Por tanto en cada fps mostramos una de las imágenes del personaje.

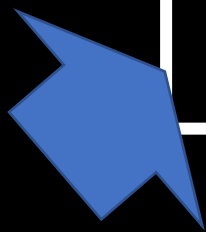
Self.index es un contador que va de 0 a 3 para recorrer cada lista del personaje, os acordáis que cada dirección era 4 imágenes.

Lo que miro es la dirección según el cursor para mostrar la imagen de cada lista.

# Obtener las pulsaciones si va a la izquierda le pongo dirección izquierda

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

# Mover el rectángulo
if keys[pygame.K_LEFT]:
    self.rect.x -= self.velocity
    self.direction = "LEFT"
if keys[pygame.K_RIGHT]:
    self.rect.x += self.velocity
    self.direction = "RIGHT"
if keys[pygame.K_UP]:
    self.rect.y -= self.velocity
    self.direction = "UP"
if keys[pygame.K_DOWN]:
    self.rect.y += self.velocity
    self.direction = "DOWN"
```



```
if self.direction == "RIGHT":
    self.image = self.animationRIGHT[self.index]
if self.direction == "LEFT":
    self.image = self.animationLEFT[self.index]
if self.direction == "UP":
    self.image = self.animationUP[self.index]
if self.direction == "DOWN":
    self.image = self.animationDOWN[self.index]
```

Cada vez que pulso le cambio la dirección y le asigno valor a la variable direction que luego cuando recorro el array para poner la imagen dependiendo de la dirección llamo a la lista que guarda las 4 imágenes de esa dirección, eso es en la diapositiva anterior

# Crear nuevos sprites a partir de los tuyos

