Facundo Falcone - Guevara Mariano - Manuel Samaniego
Biblioteca Pygame 4

Diccionarios

Representación de objetos mediante diccionarios

Podemos crear objetos para nuestro juego tales como nuestro personaje jugable, items, enemigos, trampas, etc.

Para ello, con lo que sabemos de Pygame nos crearemos un diccionario de nuestro personaje jugable donde especificaremos la imagen que usaremos como sprite, su rectángulo, posición y alguna otra característica que necesitemos

Crear personaje con diccionario

Por ejemplo, creamos un diccionario vacío para nuestro personaje y asignamos claves para los elementos que necesitamos tales como su superficie (la imagen cargada que usaremos), su rectángulo que usaremos para posibles colisiones el cual sacamos sus datos de su superficie, y la posición del rectángulo, opcionalmente podemos definir una clave de puntaje si así lo requerimos.

```
dict_personaje = {}
dict_personaje["surface"] = pygame.image.load("imagen_homero.png")
dict_personaje["surface"] = pygame.transform.scale( dict_personaje["surface"],(ancho,alto))
dict_personaje["rect_pos"] = pygame.Rect(x,y,200,200)
dict_personaje["rect"] = pygame.Rect((x+ancho/2)-10,y+90,40,20)
dict_personaje["score"] = 0
```

Actualizar rectángulo y posición

Podemos actualizar la posición de la superficie del rectángulo de nuestro personaje para generar el efecto de movimiento en la dirección que lo necesitemos sumando enteros en el atributo .x o .y de la siguiente manera:

```
dict_personaje["rect_pos"].x = dict_personaje["rect_pos"].x + 30 # pixeles
dict_personaje["rect"].x = dict_personaje["rect"].x + 30  # pixeles
```

En este caso, a nuestro personaje le aumentamos su coordenada X del rectángulo en 30 píxeles para que en algún momento de nuestro juego podamos dibujarlo 30 pixeles corrido hacia la izquierda y generar movimiento

Dibujar personaje

Para dibujarlo en nuestra ventana principal del juego, usaremos el método **blit()** de nuestro objeto ventana al cual le pasaremos como argumento la superficie de nuestro personaje y la posición del rectángulo de nuestro personaje.

```
ventana_ppal.blit(dict_personaje["surface"],dict_personaje["rect_pos"])
```

Conjuntamente en nuestro bucle principal podemos llamar el código de la siguiente manera:

Inicializar ventana principal y nuestro personaje

main_juego.py > ...

```python
import pygame
import colores

ANCHO_VENTANA = 800
ALTO_VENTANA = 500

pygame.init()
ventana_ppal = pygame.display.set_mode((ANCHO_VENTANA,ALTO_VENTANA))
pygame.display.set_caption("PYGAME HOMERO COME DONAS")

# TIMER
timer = pygame.USEREVENT + 0
pygame.time.set_timer(timer,100)

# player = personaje.crear(ANCHO_VENTANA/2,ALTO_VENTANA-200,200,200)

dict_personaje = {}
dict_personaje["surface"] = pygame.image.load("imagen_homero.png")
dict_personaje["surface"] = pygame.transform.scale( dict_personaje["surface"],(200,200))
dict_personaje["rect_pos"] = pygame.Rect(ANCHO_VENTANA/2, ALTO_VENTANA-200,200, 200)
dict_personaje["rect"] = pygame.Rect(
    (ANCHO_VENTANA/2 + 200/2)-10, ALTO_VENTANA-200+90, 40, 20)
```

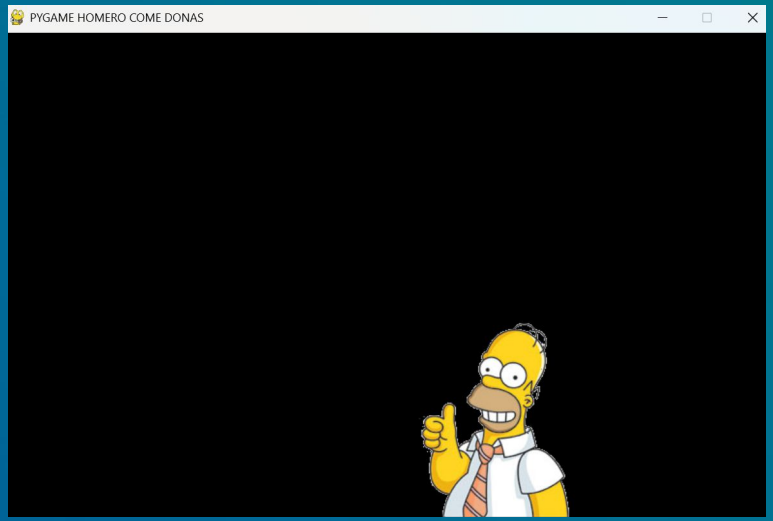
Dibujar nuestro personaje en la ventana

```
flag_run = True
while flag_run:
    lista_eventos = pygame.event.get()
    for evento in lista_eventos:
        if evento.type == pygame.QUIT:
            flag_run = False

    ventana_ppal.fill(colores.NEGRO)
    # Actualizamos la pantalla del personaje
    ventana_ppal.blit(dict_personaje["surface"],dict_personaje["rect_pos"])

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

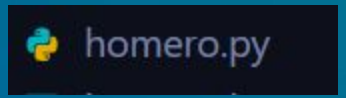
Resultado final



Mejoras en nuestro código

Todo esto se puede mejorar aplicando funciones que nos permitan crear el diccionario de nuestro personaje, actualizarlo en la ventana y dibujarlo, para evitar tener que repetir código y facilitarnos la tarea y prolijidad en nuestro trabajo.

Como primer paso lo ideal es crear un módulo para las funciones de nuestro personaje, en este caso lo llamaremos **homero.py**



Crear el diccionario de nuestro personaje

Respecto a crear el diccionario de nuestro personaje, podemos tener una función llamada **crear()** la cual reciba como parámetro las coordenadas X e Y, el ancho de la imagen redimensionada y el alto.

```
def crear(x,y,ancho,alto):
    dict_personaje = {}
    dict_personaje["surface"] = pygame.image.load("imagen_homero.png")
    dict_personaje["surface"] = pygame.transform.scale( dict_personaje["surface"],(ancho,alto))
    dict_personaje["rect_pos"] = pygame.Rect(x,y,200,200)
    dict_personaje["rect"] = pygame.Rect((x+ancho/2)-10,y+90,40,20)
    dict_personaje["score"] = 0

    return dict_personaje
```

Actualizar personaje

También podemos crear una función llamada **update()** que se encargue de actualizar la posición de nuestro personaje, la cual reciba como parámetro el diccionario del personaje y cuando va a incrementar o decrementar en X

```
def update(personaje,incremento_x):
    nueva_x = personaje["rect_pos"].x + incremento_x
    if(nueva_x > 0 and nueva_x < 600):
        personaje["rect_pos"].x = personaje["rect_pos"].x + incremento_x
        personaje["rect"].x = personaje["rect"].x + incremento_x
```

Actualizar pantalla

Finalmente podemos crear una función llamada **actualizar_pantalla()** la cual reciba como parámetro el diccionario del personaje y la superficie de la pantalla en la cual lo dibujara

```
def actualizar_pantalla(dict_personaje,ventana_ppal):
    ventana_ppal.blit(dict_personaje["surface"],dict_personaje["rect_pos"])
```

Programación I

Facundo Falcone - Guevara Mariano - Manuel Samaniego

Biblioteca Pygame 4

Nuestro modulo **homero.py** se verá de la siguiente manera

homero.py > ...

```python
import pygame

def crear(x,y,ancho,alto):
    dict_personaje = {}
    dict_personaje["surface"] = pygame.image.load("imagen_homero.png")
    dict_personaje["surface"] = pygame.transform.scale( dict_personaje["surface"],(ancho,alto))
    dict_personaje["rect_pos"] = pygame.Rect(x,y,200,200)
    dict_personaje["rect"] = pygame.Rect((x+ancho/2)-10,y+90,40,20)
    dict_personaje["score"] = 0

    return dict_personaje

def actualizar_pantalla(dict_personaje,ventana_ppal):
    ventana_ppal.blit(dict_personaje["surface"],dict_personaje["rect_pos"])


def update(personaje,incremento_x):
    nueva_x = personaje["rect_pos"].x + incremento_x
    if(nueva_x > 0 and nueva_x < 600):
        personaje["rect_pos"].x = personaje["rect_pos"].x + incremento_x
        personaje["rect"].x = personaje["rect"].x + incremento_x
```

Programación I

Facundo Falcone - Guevara Mariano - Manuel Samaniego

Biblioteca Pygame 4

main_juego.py > ...

```python
import pygame
import colores
import homero

ANCHO_VENTANA = 800
ALTO_VENTANA = 500

pygame.init()
ventana_ppal = pygame.display.set_mode((ANCHO_VENTANA,ALTO_VENTANA))
pygame.display.set_caption("PYGAME HOMERO COME DONAS")

# TIMER
timer = pygame.USEREVENT + 0
pygame.time.set_timer(timer,100)

dict_personaje = homero.crear(ANCHO_VENTANA/2,ALTO_VENTANA-200,200,200)

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

    ventana_ppal.fill(colores.NEGRO)
    # Actualizamos la pantalla del personaje
    homero.actualizar_pantalla(dict_personaje, ventana_ppal)

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

Con estas mejoras solo nos queda importar el módulo en nuestro main y llamar a las funciones creadas, quedando de la siguiente manera

Agregar eventos de teclado

Adicionalmente podemos agregar eventos de teclado para mover y actualizar nuestra imagen usando la funcion **update()** del modulo **homero.py** para que al presionar teclas izquierda y derecha, nuestro personaje se mueva en el eje X en esas direcciones

Programación I

Facundo Falcone - Guevara Mariano - Manuel Samaniego

Biblioteca Pygame 4

```
flag_run = True
while flag_run:
    lista_eventos = pygame.event.get()
    for evento in lista_eventos:
        if evento.type == pygame.QUIT:
            flag_run = False

    lista_teclas = pygame.key.get_pressed()

    if lista_teclas[pygame.K_LEFT] :
        pixeles_a_mover_en_x = -10
        homero.update(dict_personaje, pixeles_a_mover_en_x)

    if lista_teclas[pygame.K_RIGHT] :
        pixeles_a_mover_en_x = 10
        homero.update(dict_personaje, pixeles_a_mover_en_x)

    ventana_ppal.fill(colores.NEGRO)
    # Actualizamos la pantalla del personaje
    homero.actualizar_pantalla(dict_personaje, ventana_ppal)

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