Desconocido

## 15 Crear al Sr. Hombre de Palo



En este capítulo, crearemos al personaje principal de Mr. Stick Man *Races for the Exit .* Esto requerirá la codificación más complicada que hemos hecho hasta ahora, porque el Sr. Hombre de Palo tiene que correr a izquierda y derecha, saltar, detenerse cuando choque con una plataforma y caer cuando se salga del borde de una plataforma. Utilizaremos eventos para las teclas de flecha izquierda y derecha para que el muñeco de palitos corra a izquierda y derecha, y haremos que salte cuando el jugador pulse la barra espaciadora.

### Inicialización de la figura de palo

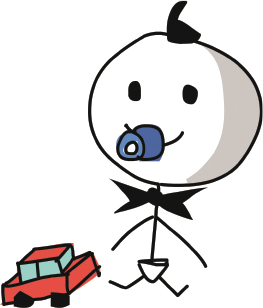
La función \_\_init\_\_ de nuestra nueva clase figura de palo se parecerá mucho a las de las demás clases de nuestro juego. Empezaremos dando un nombre a nuestra nueva clase - StickFigureSprite - y asignaremos esta clase a una clase padre, Sprite :

class StickFigureSprite(Sprite):  
 def \_\_init\_\_(self, game):  
 Sprite.\_\_init\_\_(self, game)

Este código es similar al de la clase PlatformSprite del [Capítulo 14](ch14.xhtml#ch14) , salvo que no utilizamos ningún parámetro adicional (aparte de self y game ). Esto se debe a que, a diferencia de la clase PlatformSprite , en el juego sólo se utilizará un objeto StickFigureSprite .

### Cargar las imágenes de los muñecos

Como tenemos muchos objetos plataforma en la pantalla, pasamos la imagen de la plataforma como parámetro de la función \_\_init\_\_ de PlatformSprite (algo así como decir: "Toma, PlatformSprite, utiliza esta imagen cuando te dibujes en la pantalla"). Pero como sólo hay un objeto figura de palo, no tiene sentido cargar la imagen fuera del sprite y luego pasarla como parámetro. La clase StickFigureSprite sabrá cómo cargar sus propias imágenes.



Las siguientes líneas de la función \_\_init\_\_ cargan las tres imágenes de la izquierda (que utilizaremos para animar la figura de palo que corre hacia la izquierda) y las tres imágenes de la derecha (utilizadas para animar la figura de palo que corre hacia la derecha). Necesitamos cargar estas imágenes ahora, porque no queremos cargarlas cada vez que mostremos la figura de palo en la pantalla (hacerlo llevaría demasiado tiempo y haría que nuestro juego funcionara lentamente):

class StickFigureSprite(Sprite):  
 def \_\_init\_\_(self, game):  
 Sprite.\_\_init\_\_(self, game)  
 ➊ self.images\_left = [  
 PhotoImage(file='figure-L1.gif'),  
 PhotoImage(file='figure-L2.gif'),  
 PhotoImage(file='figure-L3.gif')  
 ]  
 ➋ self.images\_right = [  
 PhotoImage(file='figure-R1.gif'),  
 PhotoImage(file='figure-R2.gif'),  
 PhotoImage(file='figure-R3.gif')  
 ]  
 ➌ self.image = game.canvas.create\_image(200, 470,  
 image=self.images\_left[0], anchor='nw')

Este código carga las tres imágenes de la izquierda, que utilizaremos para animar a la figura de palo que corre hacia la izquierda, y las tres imágenes de la derecha, que utilizaremos para animar a la figura de palo que corre hacia la derecha.

Creamos las variables de objeto images\_left ➊ y images \_right ➋ . Cada una contiene una lista de los objetos PhotoImage que creamos en el [Capítulo 10](ch10.xhtml#ch10) , que muestran la figura de palo mirando a izquierda y derecha.

Dibujamos la primera imagen ➌ con images\_left[0] utilizando la función create\_image del lienzo en la posición ( 200, 470 ), que sitúa la figura de palo en el centro de la pantalla del juego, en la parte inferior del lienzo. La función create\_image devuelve un número que identifica la imagen en el lienzo. Almacenamos este identificador en la variable objeto image para su uso posterior.

### Configurar variables

La siguiente parte de la función \_\_init\_\_ configura más variables que utilizaremos más adelante en este código:

self.image = game.canvas.create\_image(200, 470,   
 image=self.images\_left[0], anchor='nw')  
 ➊ self.x = -2  
 ➋ self.y = 0  
 self.current\_image = 0  
 self.current\_image\_add = 1  
 self.jump\_count = 0   
 self.last\_time = time.time()  
 self.coordinates = Coords()

Las variables de objeto x ➊ y y ➋ almacenarán la cantidad que añadiremos a las coordenadas horizontales ( *x1* y *x2* ) o verticales ( *y1* y *y2* ) de la figura de palo cuando se mueva por la pantalla.

Como aprendiste en el [Capítulo 11](ch11.xhtml#ch11) , para animar algo con el módulo tkinter , añadimos valores a la posición x o y del objeto para moverlo por el lienzo. Ajustando x a -2 y y a 0, restamos 2 a la posición x más adelante en el código y no añadimos nada a la posición vertical, para hacer que la figura de palo corra hacia la izquierda.

Nota

*Recuerda que un número x negativo significa moverse a la izquierda en el lienzo, y un número x positivo significa moverse a la derecha.* Un número  *y negativo significa moverse hacia arriba, y un número y positivo significa moverse hacia abajo.*

A continuación, creamos la variable de objeto current\_image para almacenar la posición índice de la imagen tal y como se muestra actualmente en la pantalla. Nuestra lista de imágenes orientadas a la izquierda, images\_left , contiene *figura-L1.gif* , *figura-L2.gif* , y *figura-L3.gif .* Ésas son las posiciones índice 0, 1 y 2.

La variable current\_image\_add contendrá el número que sumaremos a la posición índice almacenada en current\_image para obtener la siguiente posición índice. Por ejemplo, si se muestra la imagen en la posición índice 0, sumamos 1 para obtener la siguiente imagen en la posición índice 1, y volvemos a sumar 1 para obtener la imagen final de la lista en la posición índice 2. (Verás cómo utilizamos esta variable para la animación en el capítulo siguiente).

La variable jump\_count es un contador que utilizaremos mientras salta la figura de palo. La variable last\_time registrará la última vez que cambiamos la imagen al animar nuestra figura de palo. Almacenamos la hora actual utilizando la función time del módulo time .

Por último, establecemos la variable objeto coordinates en un objeto de la clase Coords , sin parámetros de inicialización ( x1 , y1 , x2 , y y2 son todos 0). A diferencia de lo que ocurre con las plataformas, las coordenadas de la figura de palo cambiarán, por lo que estableceremos estos valores más adelante.

### Vinculación a las teclas

En la parte final de la función \_\_init\_\_ , las funciones bind vinculan una tecla a algo de nuestro código que debe ejecutarse cuando se pulse la tecla:

self.jump\_count = 0   
 self.last\_time = time.time()  
 self.coordinates = Coords()  
 game.canvas.bind\_all('<KeyPress-Left>', self.turn\_left)  
 game.canvas.bind\_all('<KeyPress-Right>', self.turn\_right)  
 game.canvas.bind\_all('<space>', self.jump)

Vinculamos <KeyPress-Left> a la función turn\_left , <KeyPress-Right> a la función turn\_right y <space> a la función jump . Ahora tenemos que crear esas funciones para que la figura de palo se mueva.

### Girar la figura de palo a izquierda y derecha

Las funciones turn\_left y turn\_right se aseguran de que la figura de palo no salta, y luego establecen el valor de la variable de objeto x para moverla a izquierda y derecha. (Nuestro juego no nos permite cambiar su dirección en el aire).



game.canvas.bind\_all('<KeyPress-Left>', self.turn\_left)  
 game.canvas.bind\_all('<KeyPress-Right>', self.turn\_right)  
 game.canvas.bind\_all('<space>', self.jump)  
  
 def turn\_left(self, evt):  
 if self.y == 0:  
 ➊ self.x = -2  
  
 def turn\_right(self, evt):  
 if self.y == 0:  
 ➋ self.x = 2

Python llama a la función turn\_left cuando el jugador pulsa la tecla de flecha izquierda, y pasa como parámetro un objeto con información sobre lo que ha hecho el jugador. Este objeto se llama *objeto de evento* , y le damos el nombre de parámetro evt .

Nota

*El objeto evento no es importante para nuestros propósitos, pero necesitamos incluirlo como parámetro de nuestras funciones o recibiremos un error, porque Python espera que esté ahí. El objeto evento contiene cosas como las* *posiciones* x *e* y *del ratón (para un evento de ratón), un código que identifica una tecla concreta que se ha pulsado (para eventos de teclado) y otra información. Para este juego, ninguna de esa información es útil, así que podemos ignorarla sin problemas.*

Para saber si la figura de palo está saltando, comprobamos el valor de la variable de objeto y . Si el valor no es 0, la figura de palo está saltando. En este código, si el valor de y es 0, ponemos x a -2 para correr a la izquierda ➊ o lo ponemos a 2 para correr a la derecha ➋ . Usamos -2 y 2, porque establecer el valor en -1 o 1 no haría que la figura de palo se moviera por la pantalla lo suficientemente rápido.

Cuando la animación de tu muñeco funcione, prueba a cambiar este valor para ver qué diferencia hay.

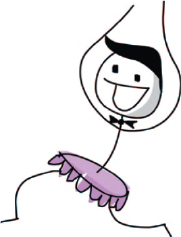
### Cómo hacer que la figura de palo salte

La función jump es muy similar a las funciones turn\_left y turn\_right :

def turn\_right(self, evt):  
 if self.y == 0:  
 self.x = 2  
  
 def jump(self, evt):  
 if self.y == 0:  
 self.y = -4  
 self.jump\_count = 0

Esta función vuelve a tomar un parámetro evt (el objeto evento), que podemos ignorar porque no necesitamos más información sobre el evento (igual que antes). Si se llama a esta función, sabemos que se ha pulsado la barra espaciadora.

Como queremos que nuestra figura de palo salte sólo si no está saltando ya, comprobamos si y es igual a 0. Si la figura de palo no está saltando, ponemos y a -4 (para moverla verticalmente por la pantalla), y ponemos jump\_count a 0. Utilizaremos jump\_count para asegurarnos de que la figura de palo no sigue saltando eternamente. En lugar de eso, dejaremos que salte durante un tiempo determinado y luego haremos que vuelva a bajar, como si la gravedad tirara de él. Añadiremos este código en el próximo capítulo.



### Lo que tenemos hasta ahora

Repasemos las definiciones de las clases y funciones de nuestro juego hasta ahora, y dónde deben estar en tu archivo.

En la parte superior de tu programa, deberías tener las declaraciones import , seguidas de las clases Game y Coords . La clase Game se utilizará para crear un objeto que será el controlador principal de nuestro juego, y los objetos de la clase Coords se utilizan para mantener las posiciones de las cosas en nuestro juego (como las plataformas y el Sr. Hombre de Palo):

from tkinter import \*  
import random  
import time  
  
class Game:  
 --snip--  
class Coords:  
 --snip--

A continuación, deberías tener las funciones within (que dicen si las coordenadas de un sprite están dentro de la misma área de otro sprite), la clase padre Sprite (que es la clase padre de todos los sprites de nuestro juego), la clase PlatformSprite y el principio de la clase StickFigureSprite . Utilizamos la clase PlatformSprite para crear objetos plataforma, sobre los que saltará nuestra figura de palo. También creamos un objeto de la clase StickFigureSprite , para representar al personaje principal de nuestro juego:

def within\_x(co1, co2):  
 --snip--  
def within\_y(co1, co2):  
 --snip--  
def collided\_left(co1, co2):  
 --snip--  
def collided\_right(co1, co2):  
 --snip--  
def collided\_top(co1, co2):  
 --snip--  
def collided\_bottom(y, co1, co2):  
 --snip--  
class Sprite:  
 --snip--  
class PlatformSprite(Sprite):  
 --snip--  
class StickFigureSprite(Sprite):  
 --snip--

Por último, al final de tu programa, deberías tener el código que crea todos los objetos de nuestro juego hasta ahora: el propio objeto del juego y las plataformas. La línea final es donde llamamos a la función mainloop :

g = Game()  
platform1 = PlatformSprite(g, PhotoImage(file='platform1.gif'),  
 0, 480, 100, 10)  
...  
g.sprites.append(platform1)  
...  
g.mainloop()

Si tu código parece un poco diferente, o tienes problemas para hacerlo funcionar, siempre puedes saltar al final del [Capítulo](ch16.xhtml#ch16) 16 para ver el listado completo de todo el juego.

### Lo que has aprendido

En este capítulo, empezamos a trabajar en la clase para nuestra figura de palo. Por el momento, si creáramos un objeto de esta clase, en realidad no haría mucho más que cargar las imágenes que necesita para animar la figura de palo y establecer unas cuantas variables de objeto que se utilizarán más adelante en el código. Esta clase contiene un par de funciones para cambiar los valores de esas variables de objeto en función de los eventos del teclado (cuando el jugador pulsa la flecha izquierda o derecha o la barra espaciadora).

En el próximo capítulo, terminaremos nuestro juego. Escribiremos las funciones para que la clase StickFigureSprite muestre y anime la figura de palo y la mueva por la pantalla. También añadiremos la salida (la puerta) a la que intenta llegar el Sr. Hombre de Palo.