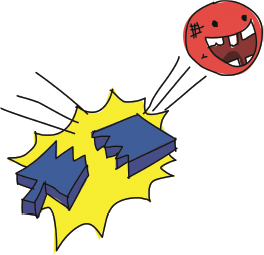
Desconocido

### Averiguar cuándo la pelota golpea la pala

En este punto de nuestro código, la pelota no golpeará la pala. De hecho, la pelota volará directamente a través de la paleta. La pelota necesita saber cuándo ha golpeado la paleta, igual que necesita saber cuándo ha golpeado una pared.



Podríamos resolver este problema añadiendo código a la función draw (donde tenemos el código que comprueba si hay paredes), pero es mejor idea mover este código a nuevas funciones para dividir las cosas en trozos más pequeños. Si ponemos demasiado código en un solo lugar (dentro de una función, por ejemplo), podemos hacer que el código sea mucho más difícil de entender. Hagamos los cambios necesarios.

En primer lugar, cambiamos la función \_\_init\_\_ de la bola para que podamos pasar el objeto paddle como parámetro:

class Ball:  
 ➊ def \_\_init\_\_(self, canvas, paddle, color):  
 self.canvas = canvas  
 ➋ self.paddle = paddle  
 self.id = canvas.create\_oval(10, 10, 25, 25, fill=color)  
 self.canvas.move(self.id, 245, 100)  
 starts = [-3, -2, -1, 1, 2, 3]  
 self.x = random.choice(starts)  
 self.y = -3  
 self.canvas\_height = self.canvas.winfo\_height()   
 self.canvas\_width = self.canvas.winfo\_width()

Observa que cambiamos los parámetros de \_\_init\_\_ para incluir la paleta ➊ . A continuación, asignamos el parámetro de la paleta a la variable de objeto paddle ➋ .

Una vez guardado el objeto paddle , tenemos que cambiar el código donde creamos el objeto ball . Este cambio se encuentra al final de nuestro programa, justo antes del bucle principal while :

paddle = Paddle(canvas, 'blue')  
ball = Ball(canvas, paddle, 'red')  
  
while True:  
 ball.draw()  
 paddle.draw()  
 tk.update\_idletasks()  
 tk.update()  
 time.sleep(0.01)

Para ver si la pelota ha golpeado la paleta, necesitamos un código un poco más complicado que el código añadido anteriormente para comprobar si hay paredes. Llamaremos a esta función hit\_paddle y la invocaremos en la función draw de la clase Ball , donde comprobamos si la pelota ha tocado el fondo de la pantalla:

def draw(self):  
 self.canvas.move(self.id, self.x, self.y)  
 pos = self.canvas.coords(self.id)  
 if pos[1] <= 0 or pos[3] >= self.canvas\_height:  
 self.y = self.y \* -1  
 ➊ if self.hit\_paddle(pos) == True:  
 ➋ self.y = self.y \* -1  
 if pos[0] <= 0 or pos[2] >= self.canvas\_width:  
 self.x = self.x \* -1

En el código recién añadido, si hit\_paddle devuelve True ➊ , cambiamos la dirección de la bola ajustando la variable de objeto y a su valor multiplicado por -1 ➋ (lo mismo que cuando toca la parte superior o inferior de la tela). Con este código, básicamente estamos diciendo: "Si la pelota ( self ) ha golpeado la pala, invertimos su dirección vertical".

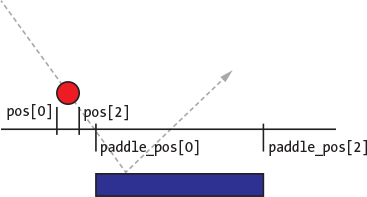
Podríamos combinar las comprobaciones de la parte superior, inferior y de la paleta en una sola declaración if -pero es más fácil para los nuevos programadores leer este código si las mantenemos separadas.

No intentes ejecutar el juego todavía; aún tenemos que crear la función hit\_paddle . Hagámoslo ahora. Añade la función hit\_paddle justo antes de la función draw en la clase Ball :

def hit\_paddle(self, pos):  
 paddle\_pos = self.canvas.coords(self.paddle.id)  
 if pos[2] >= paddle\_pos[0] and pos[0] <= paddle\_pos[2]:  
 if pos[3] >= paddle\_pos[1] and pos[3] <= paddle\_pos[3]:  
 return True  
 return False

En primer lugar, definimos la función con el parámetro pos . Este parámetro contiene las coordenadas actuales de la pelota. Después, obtenemos las coordenadas de la paleta y las almacenamos en la variable paddle\_pos .

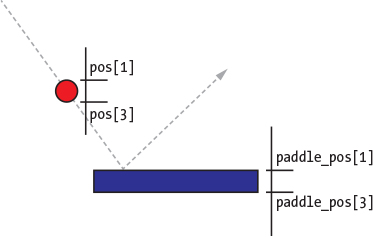
A continuación, tenemos la primera parte de nuestra sentencia if-then , que dice: "Si el lado derecho de la pelota es mayor que el lado izquierdo de la paleta, y el lado izquierdo de la pelota es menor que el lado derecho de la paleta. . ." Aquí, pos[2] contiene la *coordenada* x del lado derecho de la pelota, y pos[0] contiene la *coordenada* x de su lado izquierdo. La variable paddle\_pos[0] contiene la *coordenada* x del lado izquierdo de la paleta, y paddle\_pos[2] contiene la *coordenada* x del lado derecho. [La Figura 12-2](ch12.xhtml#ch12fig02) muestra el aspecto de estas coordenadas cuando la pelota está a punto de golpear la paleta.



*Figura 12-2: Pelota a punto de golpear la pala - mostrando coordenadas horizontales*

La pelota está cayendo hacia la paleta, pero en este caso, ves que el lado derecho de la pelota ( pos[2] ) aún no ha cruzado el lado izquierdo de la paleta (eso es paddle\_pos[0] ).

A continuación, vemos si la parte inferior de la bola ( pos[3] ) está entre la parte superior ( paddle\_pos[1] ) y la parte inferior ( paddle\_pos[3] ) de la pala. En la [Figura 12-3](ch12.xhtml#ch12fig03) , puedes ver que la parte inferior de la pelota ( pos[3] ) aún no ha golpeado la parte superior de la pala ( paddle\_pos[1] ).



Figura*12-3: Pelota a punto de golpear la pala - mostrando coordenadas verticales*

Así, basándose en la posición actual de la pelota, la función hit\_paddle devolvería False .

Nota

*¿Por qué necesitamos ver si la parte inferior de la pelota está entre la parte superior e inferior de la pala? ¿Por qué no ver simplemente si la parte inferior de la bola ha tocado la parte superior de la paleta? Porque cada vez que movemos la pelota por el lienzo, nos movemos a saltos de 3 píxeles. Si nos limitáramos a comprobar si la pelota ha llegado a la parte superior de la paleta ( pos[1] ), podríamos haber saltado más allá de esa posición. En ese caso, la bola seguiría viajando y atravesaría la paleta sin detenerse.*

[anterior](ch12_3.html)[Subtema 4 de 7: (Ver todo)](ch12.html)[siguiente](ch12_5.html)