Desconocido

### Ponerlo todo junto: Nuestro código final de detección de colisiones

Una vez que hemos determinado si un conjunto de *coordenadas x* se ha cruzado con otro, y hemos hecho lo mismo con *las coordenadas y* , podemos escribir funciones para ver si un sprite ha chocado con otro sprite y por qué lado. Haremos esto con las funciones collided\_left , collided\_right , collided\_top , y collided\_bottom .

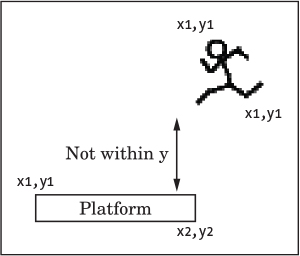
#### La función colisionado\_izquierdo

Añade el siguiente código para la función collided\_left debajo de las dos funciones within que acabamos de crear:

def collided\_left(co1, co2):  
 if within\_y(co1, co2):  
 if co1.x1 >= co2.x1 and co1.x1 <= co2.x2:  
 return True  
 return False

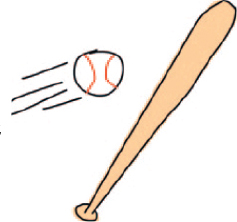
Esta función nos dice si el lado izquierdo (el valor x1 ) de un primer objeto de coordenadas ha chocado con otro objeto de coordenadas.

La función toma dos parámetros: co1 (el primer objeto de coordenadas) y co2 (el segundo objeto de coordenadas). Comprobamos si los dos objetos de coordenadas se han cruzado verticalmente, utilizando la función within\_y . Al fin y al cabo, no tiene sentido comprobar si el Sr. Hombre de Palo ha chocado contra una plataforma si está flotando muy por encima de ella (como en [la Figura 14-5](ch14.xhtml#ch14fig05) ).



*Figura 14-5:* El Sr. Hombre de Palo*por encima* de la plataforma

A continuación, comprobamos si el valor de la posición más a la izquierda del primer objeto de coordenadas ( co1.x1 ) ha chocado con la posición x2 del segundo objeto de coordenadas ( co2.x2 ). Si es así, debe ser menor o igual que la posición x2 . También comprobamos que no ha sobrepasado la posición x1 . Si la ha sobrepasado, devolvemos True . Si ninguna de las afirmaciones de if es cierta, devolvemos False .



#### La función derecha\_colisionada

La función collided\_right se parece mucho a collided\_left :

def collided\_right(co1, co2):  
 if within\_y(co1, co2):  
 if co1.x2 >= co2.x1 and co1.x2 <= co2.x2:  
 return True  
 return False

Al igual que con collided\_left , comprobamos si las *coordenadas y* se han cruzado entre sí, utilizando la función within\_y . A continuación, comprobamos si el valor x2 del primer objeto de coordenadas se encuentra entre las posiciones x1 y x2 del segundo objeto de coordenadas, y devolvemos True si es así. En caso contrario, devolvemos False .

#### La función collided\_top

La función collided\_top es muy similar a las dos funciones que acabamos de añadir:

def collided\_top(co1, co2):  
 if within\_x(co1, co2):  
 if co1.y1 >= co2.y1 and co1.y1 <= co2.y2:  
 return True  
 return False

Esta vez, comprobamos si las coordenadas se han cruzado horizontalmente, utilizando la función within\_x . A continuación, comprobamos si la posición más alta de la primera coordenada ( co1.y1 ) se ha cruzado con la posición y2 de la segunda coordenada, pero no con su posición y1  . Si es así, devolvemos True (la parte superior de la primera coordenada ha chocado con la segunda coordenada).

#### La función colided\_bottom

Nuestra última función, collided\_bottom , es un poco diferente:

def collided\_bottom(y, co1, co2):  
 if within\_x(co1, co2):  
 y\_calc = co1.y2 + y  
 ➊ if y\_calc >= co2.y1 and y\_calc <= co2.y2:  
 return True  
 return False

Esta función toma un parámetro adicional, y , un valor que añadimos a la posición y de la primera coordenada. Nuestra sentencia if comprueba si las coordenadas se han cruzado horizontalmente (como hicimos con collided\_top ). A continuación, añadimos el valor del parámetro y a la posición y2 de la primera coordenada, y almacenamos el resultado en la variable y\_calc . Si el nuevo valor calculado se encuentra entre los valores y1 y y2 de la segunda coordenada ➊ , devolvemos True porque la parte inferior de la coordenada co1 ha tocado la parte superior de la coordenada co2 . Sin embargo, si ninguna de las afirmaciones if es cierta, devolvemos False .

Necesitamos el parámetro adicional y porque el Sr. Hombre de Palo podría caerse de una plataforma. A diferencia de las otras funciones collided , necesitamos poder comprobar *si* colisionaría en la parte inferior, en lugar de si ya lo ha hecho. Si se cae de una plataforma y sigue flotando en el aire, nuestro juego no será muy realista; por eso, mientras camina, comprobamos si ha colisionado con algo a la izquierda o a la derecha. Cuando comprobamos debajo de él, vemos si colisionaría con la plataforma; si no, ¡tiene que caer estrepitosamente!

[anterior](ch14_9.html)[Subtema 10 de 16: (Ver todo)](ch14.html)[siguiente](ch14_11.html)