Desconocido

### Colisión horizontal de sprites

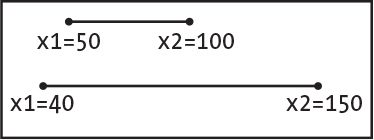
En primer lugar, crearemos la función within\_x para determinar si un conjunto de *coordenadas* *x* ( *x1* y *x2* ) se ha cruzado con otro conjunto de *coordenadas x* ( de nuevo, *x1* y *x2* ). Añade lo siguiente directamente debajo de la clase Coords :

class Coords:  
 def \_\_init\_\_(self, x1=0, y1=0, x2=0, y2=0):  
 self.x1 = x1  
 self.y1 = y1  
 self.x2 = x2  
 self.y2 = y2  
  
def within\_x(co1, co2):  
 ➊ if co1.x1 > co2.x1 and co1.x1 < co2.x2:  
 return True  
 ➋ elif co1.x2 > co2.x1 and co1.x2 < co2.x2:  
 return True  
 elif co2.x1 > co1.x1 and co2.x1 < co1.x2:  
 return True  
 elif co2.x2 > co1.x1 and co2.x2 < co1.x2:  
 return True  
 else:  
 return False

La función within\_x toma los parámetros co1 y co2 , ambos objetos Coords . Primero comprobamos si la posición más a la izquierda del primer objeto de coordenadas ( co1.x1 ) está entre la posición más a la izquierda ( co2.x1 ) y la posición más a la derecha ( co2.x2 ) del segundo objeto de coordenadas ➊ . Si es así, devolvemos True .



Echemos un vistazo a dos líneas con *coordenadas x* superpuestas para entender cómo funciona esto. Cada línea de la [Figura 14-4](ch14.xhtml#ch14fig04) comienza en x1 y termina en x2 .



*Figura 14-4: Coordenadas horizontales (* x) superpuestas

La primera línea de este diagrama ( co1 ) comienza en la posición de píxel 50 ( x1 ) y termina en 100 ( x2 ). La segunda línea ( co2 ) comienza en la posición 40 y termina en 150. En este caso, como la posición x1 de la primera línea está entre las posiciones x1 y x2 de la segunda línea, la afirmación if de la función sería verdadera para estos dos conjuntos de coordenadas.

Con la primera sentencia elif ➋ , vemos si la posición más a la derecha de la primera línea ( co1.x2 ) está entre la posición más a la izquierda ( co2.x1 ) y la posición más a la derecha ( co2.x2 ) de la segunda. Si es así, devolvemos True . Las dos siguientes sentencias elif hacen casi lo mismo: comprueban las posiciones más a la izquierda y más a la derecha de la segunda línea ( co2 ) con la primera ( co1 ).

Si ninguna de las sentencias if coincide, llegamos a else y devolvemos False . Esto equivale a decir: "No, los dos objetos de coordenadas no se cruzan horizontalmente".

Para ver un ejemplo del funcionamiento de la función, vuelve a mirar la [Figura 14-4 .](ch14.xhtml#ch14fig04) Las posiciones x1 y x2 del primer objeto de coordenadas son 50 y 100, y las posiciones x1 y x2 del segundo objeto de coordenadas son 40 y 150. Esto es lo que ocurre cuando llamamos a la función within\_x que hemos creado:

>>> c1 = Coords(50, 50, 100, 100)  
>>> c2 = Coords(40, 40, 150, 150)  
>>> print(within\_x(c1, c2))  
True

La función devuelve True . Este es el primer paso para determinar si un sprite ha chocado con otro. Por ejemplo, cuando creemos una clase para el Sr. Hombre de Palo y para las plataformas, podremos saber si sus *coordenadas x* se han cruzado.

No es una buena práctica tener muchas sentencias if o elif que devuelvan el mismo valor. Para resolver este problema, podemos acortar la función within\_x rodeando cada una de sus condiciones con paréntesis, separados por la palabra clave or . Para obtener una función algo más ordenada y con menos líneas de código, puedes cambiar la función  para que tenga este aspecto:

def within\_x(co1, co2):  
 if (co1.x1 > co2.x1 and co1.x1 < co2.x2) \  
 or (co1.x2 > co2.x1 and co1.x2 < co2.x2) \  
 or (co2.x1 > co1.x1 and co2.x1 < co1.x2) \  
 or (co2.x2 > co1.x1 and co2.x2 < co1.x2):  
 return True  
 else:  
 return False

Para extender la sentencia if a lo largo de varias líneas, de modo que no acabemos con una línea realmente larga que contenga todas las condiciones, utilizamos una barra invertida (\), como se muestra arriba.

[anterior](ch14_7.html)[Subtema 8 de 16: (Ver todo)](ch14.html)[siguiente](ch14_9.html)