### Refactorización de \_check\_events()

El método \_check\_events() aumentará su longitud a medida que sigamos desarrollando el juego, así que vamos a dividir \_check\_events() en dos métodos distintos: uno que se ocupe de los eventos de KEYDOWN y otro que se ocupe de los eventos de KEYUP:

**alien\_invasion.py**

def \_check\_events(self):  
 """Respond to keypresses and mouse events."""  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 sys.exit()  
 elif event.type == pygame.KEYDOWN:  
 self.\_check\_keydown\_events(event)  
 elif event.type == pygame.KEYUP:  
 self.\_check\_keyup\_events(event)  
  
 def \_check\_keydown\_events(self, event):  
 """Respond to keypresses."""  
 if event.key == pygame.K\_RIGHT:  
 self.ship.moving\_right = True  
 elif event.key == pygame.K\_LEFT:  
 self.ship.moving\_left = True  
  
 def \_check\_keyup\_events(self, event):  
 """Respond to key releases."""  
 if event.key == pygame.K\_RIGHT:  
 self.ship.moving\_right = False  
 elif event.key == pygame.K\_LEFT:  
 self.ship.moving\_left = False

Creamos dos nuevos métodos ayudantes: \_check\_keydown\_events() y \_check\_keyup\_events(). Cada uno necesita un parámetro self y un parámetro event. Los cuerpos de estos dos métodos están copiados de \_check\_events(), y hemos sustituido el código antiguo por llamadas a los nuevos métodos. El método \_check\_events() es más sencillo ahora con esta estructura de código más limpia, lo que facilitará el desarrollo de otras respuestas a la entrada del jugador.

[anterior](c12_24.html)[Subtema 25 de 43: (Ver todo)](c12.html)[siguiente](c12_26.html)