### Creación del método \_update\_bullets()

Queremos mantener la clase AlienInvasion razonablemente bien organizada, así que ahora que hemos escrito y comprobado el código de gestión de las balas, podemos trasladarlo a un método aparte. Crearemos un nuevo método llamado \_update\_bullets() y lo añadiremos justo antes de \_update\_screen():

**alien\_invasion.py**

def \_update\_bullets(self):  
 """Update position of bullets and get rid of old bullets."""  
 # Update bullet positions.  
 self.bullets.update()  
  
 # Get rid of bullets that have disappeared.  
 for bullet in self.bullets.copy():  
 if bullet.rect.bottom <= 0:  
 self.bullets.remove(bullet)

El código de \_update\_bullets() está cortado y pegado de run\_game(); lo único que hemos hecho aquí es aclarar los comentarios.

El bucle while de run\_game() vuelve a parecer sencillo:

**alien\_invasion.py**

while True:  
 self.\_check\_events()  
 self.ship.update()  
 self.\_update\_bullets()  
 self.\_update\_screen()  
 self.clock.tick(60)

Ahora nuestro bucle principal sólo contiene un código mínimo, por lo que podemos leer rápidamente los nombres de los métodos y comprender lo que ocurre en el juego. El bucle principal comprueba la entrada del jugador, y luego actualiza la posición de la nave y las balas que se hayan disparado. A continuación, utilizamos las posiciones actualizadas para dibujar una nueva pantalla y marcar el reloj al final de cada pasada por el bucle.

Ejecuta *alien\_invasion.py* una vez más, y asegúrate de que aún puedes disparar balas sin errores.

[anterior](c12_40.html)[Subtema 41 de 43: (Ver todo)](c12.html)[siguiente](c12_42.html)