### El método \_actualizar\_pantalla()

Para simplificar aún más run\_game(), trasladaremos el código para actualizar la pantalla a un método independiente llamado \_update\_screen():

**alien\_invasion.py**

def run\_game(self):  
 """Start the main loop for the game."""  
 while True:  
 self.\_check\_events()  
 self.\_update\_screen()  
 self.clock.tick(60)  
  
 def \_check\_events(self):  
 --snip--  
  
 def \_update\_screen(self):  
 """Update images on the screen, and flip to the new screen."""  
 self.screen.fill(self.settings.bg\_color)  
 self.ship.blitme()  
  
 pygame.display.flip()

Trasladamos el código que dibuja el fondo y la nave y voltea la pantalla a \_update\_screen(). Ahora el cuerpo del bucle principal en run\_game() es mucho más sencillo. Es fácil ver que buscamos nuevos eventos, actualizamos la pantalla y hacemos tictac en el reloj en cada pasada por el bucle.

Si ya has construido varios juegos, probablemente empezarás dividiendo tu código en métodos como éstos. Pero si nunca has abordado un proyecto como éste, es probable que al principio no sepas exactamente cómo estructurar tu código. Este enfoque te da una idea de un proceso de desarrollo realista: empiezas escribiendo tu código de la forma más sencilla posible, y luego lo refactorizas a medida que tu proyecto se hace más complejo.

Ahora que hemos reestructurado el código para que sea más fácil añadir cosas, ¡podemos trabajar en los aspectos dinámicos del juego!

[anterior](c12_16.html)[Subtema 17 de 43: (Ver todo)](c12.html)[siguiente](c12_18.html)