### Modificar los ajustes de velocidad

Primero reorganizaremos la clase Settings para agrupar los ajustes del juego en estáticos y dinámicos. También nos aseguraremos de que los ajustes que cambien durante el juego se restablezcan cuando empecemos una nueva partida. Este es el método \_\_init\_\_() para *settings.py*:

**settings.py**

def \_\_init\_\_(self):  
 """Initialize the game's static settings."""  
 # Screen settings  
 self.screen\_width = 1200  
 self.screen\_height = 800  
 self.bg\_color = (230, 230, 230)  
  
 # Ship settings  
 self.ship\_limit = 3  
  
 # Bullet settings  
 self.bullet\_width = 3  
 self.bullet\_height = 15  
 self.bullet\_color = 60, 60, 60  
 self.bullets\_allowed = 3  
  
 # Alien settings  
 self.fleet\_drop\_speed = 10  
  
 # How quickly the game speeds up  
❶ self.speedup\_scale = 1.1  
  
❷ self.initialize\_dynamic\_settings()

Seguimos inicializando los ajustes que permanecen constantes en el método \_\_init\_\_(). Añadimos un ajuste speedup\_scale ❶ para controlar la velocidad del juego: un valor de 2 duplicará la velocidad del juego cada vez que el jugador alcance un nuevo nivel; un valor de 1 mantendrá la velocidad constante. Un valor como 1.1 debería aumentar la velocidad lo suficiente como para que el juego sea desafiante pero no imposible. Por último, llamamos al método initialize\_dynamic\_settings() para inicializar los valores de los atributos que deben cambiar a lo largo del juego ❷.

Este es el código de initialize\_dynamic\_settings():

**settings.py**

def initialize\_dynamic\_settings(self):  
 """Initialize settings that change throughout the game."""  
 self.ship\_speed = 1.5  
 self.bullet\_speed = 2.5  
 self.alien\_speed = 1.0  
  
 # fleet\_direction of 1 represents right; -1 represents left.  
 self.fleet\_direction = 1

Este método establece los valores iniciales de las velocidades de la nave, la bala y el alienígena. Aumentaremos estas velocidades a medida que el jugador progrese en el juego y las restableceremos cada vez que el jugador comience una nueva partida. Incluimos fleet\_direction en este método para que los alienígenas se muevan siempre al principio de una nueva partida. No necesitamos aumentar el valor de fleet\_drop\_speed, porque cuando los alienígenas se muevan más rápido por la pantalla, también bajarán más rápido por ella.

Para aumentar las velocidades de la nave, las balas y los alienígenas cada vez que el jugador alcance un nuevo nivel, escribiremos un nuevo método llamado increase\_speed():

**settings.py**

def increase\_speed(self):  
 """Increase speed settings."""  
 self.ship\_speed \*= self.speedup\_scale  
 self.bullet\_speed \*= self.speedup\_scale  
 self.alien\_speed \*= self.speedup\_scale

Para aumentar la velocidad de estos elementos del juego, multiplicamos cada ajuste de velocidad por el valor de speedup\_scale.

Aumentamos el ritmo del juego llamando a increase\_speed() en \_check\_bullet\_alien\_collisions() cuando el último alienígena de una flota ha sido derribado:

**alien\_invasion.py**

def \_check\_bullet\_alien\_collisions(self):  
 --snip--  
 if not self.aliens:  
 # Destroy existing bullets and create new fleet.  
 self.bullets.empty()  
 self.\_create\_fleet()  
 self.settings.increase\_speed()

Cambiar los valores de los ajustes de velocidad ship\_speed, alien\_speed, y bullet\_speed ¡es suficiente para acelerar todo el juego!

[anterior](c14_10.html)[Subtema 11 de 29: (Ver todo)](c14.html)[siguiente](c14_12.html)