### Ajustar la velocidad de la nave

Actualmente, la nave se mueve un píxel por ciclo a través del bucle while, pero podemos tener un control más preciso de la velocidad de la nave añadiendo un atributo ship\_speed a la clase Settings. Utilizaremos este atributo para determinar la distancia que debe recorrer la nave en cada pasada por el bucle. Aquí tienes el nuevo atributo en *settings.py*:

**settings.py**

class Settings:  
 """A class to store all settings for Alien Invasion."""  
  
 def \_\_init\_\_(self):  
 --snip--  
  
 # Ship settings  
 self.ship\_speed = 1.5

Fijamos el valor inicial de ship\_speed en 1.5. Cuando la nave se mueve ahora, su posición se ajusta 1,5 píxeles (en lugar de 1 píxel) en cada pasada por el bucle.

Estamos utilizando un flotador para el ajuste de velocidad, para tener un control más preciso de la velocidad de la nave cuando aumentemos el ritmo del juego más adelante. Sin embargo, los atributos de rect como x sólo almacenan valores enteros, así que tenemos que hacer algunas modificaciones en Ship:

**ship.py**

class Ship:  
 """A class to manage the ship."""  
  
 def \_\_init\_\_(self, ai\_game):  
 """Initialize the ship and set its starting position."""  
 self.screen = ai\_game.screen  
❶ self.settings = ai\_game.settings  
 --snip--  
  
 # Start each new ship at the bottom center of the screen.  
 self.rect.midbottom = self.screen\_rect.midbottom  
  
 # Store a float for the ship's exact horizontal position.  
❷ self.x = float(self.rect.x)  
  
 # Movement flags; start with a ship that's not moving.  
 self.moving\_right = False  
 self.moving\_left = False  
  
 def update(self):  
 """Update the ship's position based on movement flags."""  
 # Update the ship's x value, not the rect.  
 if self.moving\_right:  
❸ self.x += self.settings.ship\_speed  
 if self.moving\_left:  
 self.x -= self.settings.ship\_speed  
  
 # Update rect object from self.x.  
❹ self.rect.x = self.x  
  
 def blitme(self):  
 --snip--

Creamos un atributo settings para Ship, de modo que podamos utilizarlo en update() ❶. Como estamos ajustando la posición de la nave en fracciones de píxel, necesitamos asignar la posición a una variable a la que se le pueda asignar un flotador. Puedes utilizar un flotador para ajustar un atributo de rect, pero rect sólo conservará la parte entera de ese valor. Para hacer un seguimiento preciso de la posición del barco, definimos un nuevo self.x ❷. Utilizamos la función float() para convertir el valor de self.rect.x en un flotador y asignamos este valor a self.x.

Ahora, cuando cambiamos la posición del barco en update(), el valor de self.x se ajusta en la cantidad almacenada en settings.ship\_speed ❸. Una vez actualizado self.x, utilizamos el nuevo valor para actualizar self.rect.x, que controla la posición del barco ❹. Sólo la parte entera de self.x se asignará a self.rect.x, pero eso está bien para mostrar la nave.

Ahora podemos cambiar el valor de ship\_speed, y cualquier valor mayor que 1 hará que la nave se mueva más rápido. Esto ayudará a que la nave responda con suficiente rapidez para derribar alienígenas, y nos permitirá cambiar el tempo del juego a medida que el jugador progresa en la partida.

[anterior](c12_22.html)[Subtema 23 de 43: (Ver todo)](c12.html)[siguiente](c12_24.html)