### Creación de la clase Bala

Ahora crea un archivo *bullet.py* para almacenar nuestra clase Bullet. Aquí tienes la primera parte de *bullet.py*:

**bullet.py**

import pygame  
from pygame.sprite import Sprite  
  
class Bullet(Sprite):  
 """A class to manage bullets fired from the ship."""  
  
 def \_\_init\_\_(self, ai\_game):  
 """Create a bullet object at the ship's current position."""  
 super().\_\_init\_\_()  
 self.screen = ai\_game.screen  
 self.settings = ai\_game.settings  
 self.color = self.settings.bullet\_color  
  
 # Create a bullet rect at (0, 0) and then set correct position.  
❶ self.rect = pygame.Rect(0, 0, self.settings.bullet\_width,  
 self.settings.bullet\_height)  
❷ self.rect.midtop = ai\_game.ship.rect.midtop  
  
 # Store the bullet's position as a float.  
❸ self.y = float(self.rect.y)

La clase Bullet hereda de Sprite, que importamos del módulo pygame.sprite. Cuando utilizas sprites, puedes agrupar elementos relacionados en tu juego y actuar sobre todos los elementos agrupados a la vez. Para crear una instancia de bala, \_\_init\_\_() necesita la instancia actual de AlienInvasion, y llamamos a super() para heredar correctamente de Sprite. También establecemos atributos para los objetos pantalla y ajustes, y para el color de la bala.

A continuación creamos el atributo rect ❶ de la bala . La bala no se basa en una imagen, así que tenemos que construir una rect desde cero utilizando la clase pygame.Rect(). Esta clase necesita las coordenadas *x*- y *y*- de la esquina superior izquierda del rect, y la anchura y altura del rect. Inicializamos el rect en (0, 0), pero lo moveremos a la ubicación correcta en la línea siguiente, porque la posición de la bala depende de la posición del barco. Obtenemos la anchura y la altura de la bala a partir de los valores almacenados en self.settings.

Ajustamos el atributo midtop de la bala para que coincida con el atributo midtop ❷ del barco . Esto hará que la bala emerja de la parte superior de la nave, haciendo que parezca que la bala se dispara desde la nave. Utilizamos un flotador para la coordenada *y* de la bala, de modo que podamos realizar ajustes finos en la velocidad de la bala ❸.

Aquí tienes la segunda parte de *bullet.py*, update() y draw\_bullet():

**bullet.py**

def update(self):  
 """Move the bullet up the screen."""  
 # Update the exact position of the bullet.  
❶ self.y -= self.settings.bullet\_speed  
 # Update the rect position.  
❷ self.rect.y = self.y  
  
 def draw\_bullet(self):  
 """Draw the bullet to the screen."""  
❸ pygame.draw.rect(self.screen, self.color, self.rect)

El método update() gestiona la posición de la bala. Cuando se dispara una bala, se desplaza hacia arriba en la pantalla, lo que corresponde a un valor de coordenada *y* decreciente. Para actualizar la posición, restamos la cantidad almacenada en settings.bullet\_speed de self.y ❶. A continuación, utilizamos el valor de self.y para fijar el valor de self.rect.y ❷.

El ajuste de bullet\_speed nos permite aumentar la velocidad de las balas a medida que avanza el juego o según sea necesario para perfeccionar su comportamiento. Una vez disparada una bala, nunca cambiamos el valor de su coordenada *x*, por lo que viajará verticalmente en línea recta aunque la nave se mueva.

Cuando queremos dibujar una bala, llamamos a draw\_bullet(). La función draw.rect() rellena la parte de la pantalla definida por la rect de la bala con el color almacenado en self.color ❸.

[anterior](c12_35.html)[Subtema 36 de 43: (Ver todo)](c12.html)[siguiente](c12_37.html)