### Crear la clase Nave

Después de elegir una imagen para la nave, necesitamos mostrarla en la pantalla. Para utilizar nuestra nave, crearemos un nuevo módulo ship que contendrá la clase Ship. Esta clase gestionará la mayor parte del comportamiento de la nave del jugador:

**ship.py**

import pygame  
  
class Ship:  
 """A class to manage the ship."""  
  
 def \_\_init\_\_(self, ai\_game):  
 """Initialize the ship and set its starting position."""  
❶ self.screen = ai\_game.screen  
❷ self.screen\_rect = ai\_game.screen.get\_rect()  
  
 # Load the ship image and get its rect.  
❸ self.image = pygame.image.load('images/ship.bmp')  
 self.rect = self.image.get\_rect()  
  
 # Start each new ship at the bottom center of the screen.  
❹ self.rect.midbottom = self.screen\_rect.midbottom  
  
❺ def blitme(self):  
 """Draw the ship at its current location."""  
 self.screen.blit(self.image, self.rect)

Pygame es eficiente porque te permite tratar todos los elementos del juego como rectángulos (*rects*), aunque no tengan exactamente la forma de rectángulos. Tratar un elemento como un rectángulo es eficiente porque los rectángulos son formas geométricas simples. Cuando Pygame necesita averiguar si dos elementos del juego han colisionado, por ejemplo, puede hacerlo más rápidamente si trata cada objeto como un rectángulo. Este enfoque suele funcionar lo suficientemente bien como para que nadie que juegue a se dé cuenta de que no estamos trabajando con la forma exacta de cada elemento del juego. En esta clase trataremos la nave y la pantalla como rectángulos.

Importamos el módulo pygame antes de definir la clase. El método \_\_init\_\_() de Ship toma dos parámetros: la referencia self y una referencia a la instancia actual de la clase AlienInvasion. Esto dará a Ship acceso a todos los recursos del juego definidos en AlienInvasion. A continuación, asignamos la pantalla a un atributo de Ship ❶, para que podamos acceder a ella fácilmente en todos los métodos de esta clase. Accedemos al atributo rect de la pantalla mediante el método get\_rect() y lo asignamos a self.screen\_rect ❷. Esto nos permite colocar la nave en el lugar correcto de la pantalla.

Para cargar la imagen, llamamos a pygame.image.load() ❸ y le damos la ubicación de la imagen de nuestro barco. Esta función devuelve una superficie que representa el barco, que asignamos a self.image. Cuando la imagen está cargada, llamamos a get\_rect() para acceder al atributo rect de la superficie del barco, de modo que podamos utilizarla posteriormente para colocar el barco.

Cuando trabajes con un objeto rect, puedes utilizar las coordenadas *x*- y *y*- de los bordes superior, inferior, izquierdo y derecho del rectángulo, así como el centro, para colocar el objeto. Puedes fijar cualquiera de estos valores para establecer la posición actual del rect. Cuando estés centrando un elemento del juego, trabaja con los atributos center, centerx o centery de un rect. Cuando estés trabajando en un borde de la pantalla, trabaja con los atributos top, bottom, left o right. También hay atributos que combinan estas propiedades, como midbottom, midtop, midleft, y midright. Cuando estés ajustando la colocación horizontal o vertical de rect, puedes utilizar simplemente los atributos x y y, que son las coordenadas *x*- y *y*- de su esquina superior izquierda. Estos atributos te evitan tener que hacer cálculos que antes tenían que hacer manualmente los desarrolladores de juegos, y los utilizarás a menudo.

[anterior](c12_11.html)[Subtema 12 de 43: (Ver todo)](c12.html)[siguiente](c12_13.html)