### Respuesta a las colisiones entre naves alienígenas

Ahora tenemos que averiguar qué ocurrirá exactamente cuando un alienígena colisione con la nave. En lugar de destruir la instancia ship y crear una nueva, contaremos cuántas veces ha sido golpeada la nave mediante las estadísticas de seguimiento del juego. Las estadísticas de seguimiento también serán útiles para la puntuación.

Escribamos una nueva clase, GameStats, para realizar el seguimiento de las estadísticas del juego, y guardémosla como *game\_stats.py*:

**game\_stats.py**

class GameStats:  
 """Track statistics for Alien Invasion."""  
  
 def \_\_init\_\_(self, ai\_game):  
 """Initialize statistics."""  
 self.settings = ai\_game.settings  
❶ self.reset\_stats()  
  
 def reset\_stats(self):  
 """Initialize statistics that can change during the game."""  
 self.ships\_left = self.settings.ship\_limit

Crearemos una instancia de GameStats para todo el tiempo que se ejecute *Alien Invasion*, pero necesitaremos restablecer algunas estadísticas cada vez que el jugador comience una nueva partida. Para ello, inicializaremos la mayoría de las estadísticas en el método reset\_stats(), en lugar de hacerlo directamente en \_\_init\_\_(). Llamaremos a este método desde \_\_init\_\_() para que las estadísticas se establezcan correctamente cuando se cree por primera vez la instancia GameStats ❶. Pero también podremos llamar a reset\_stats() cada vez que el jugador empiece una nueva partida. Ahora mismo sólo tenemos una estadística, ships\_left, cuyo valor cambiará a lo largo de la partida.

El número de naves con las que comienza el jugador debe almacenarse en *settings.py* como ship\_limit:

**settings.py**

# Ship settings  
 self.ship\_speed = 1.5  
 self.ship\_limit = 3

También tenemos que hacer algunos cambios en *alien\_invasion.py* para crear una instancia de GameStats. En primer lugar, actualizaremos las sentencias import de la parte superior del archivo:

**alien\_invasion.py**

import sys  
from time import sleep  
  
import pygame  
  
from settings import Settings  
from game\_stats import GameStats  
from ship import Ship  
--snip--

Importamos la función sleep() del módulo time de la biblioteca estándar de Python, para que podamos pausar el juego un momento cuando la nave sea alcanzada. También importamos GameStats.

Crearemos una instancia de GameStats en \_\_init\_\_():

**alien\_invasion.py**

def \_\_init\_\_(self):  
 --snip--  
 self.screen = pygame.display.set\_mode(  
 (self.settings.screen\_width, self.settings.screen\_height))  
 pygame.display.set\_caption("Alien Invasion")  
  
 # Create an instance to store game statistics.  
 self.stats = GameStats(self)  
  
 self.ship = Ship(self)  
 --snip--

Hacemos la instancia después de crear la ventana del juego, pero antes de definir otros elementos del juego, como la nave.

Cuando un alien impacte contra la nave, restaremos 1 al número de naves restantes, destruiremos todos los alienígenas y balas existentes, crearemos una nueva flota y recolocaremos la nave en el centro de la pantalla. También pausaremos el juego un momento para que el jugador pueda darse cuenta de la colisión y reagruparse antes de que aparezca una nueva flota.

Pongamos la mayor parte de este código en un nuevo método llamado \_ship\_hit(). Llamaremos a este método desde \_update\_aliens() cuando un alien impacte contra la nave:

**alien\_invasion.py**

def \_ship\_hit(self):  
 """Respond to the ship being hit by an alien."""  
 # Decrement ships\_left.  
❶ self.stats.ships\_left -= 1  
  
 # Get rid of any remaining bullets and aliens.  
❷ self.bullets.empty()  
 self.aliens.empty()  
  
 # Create a new fleet and center the ship.  
❸ self.\_create\_fleet()  
 self.ship.center\_ship()  
  
 # Pause.  
❹ sleep(0.5)

El nuevo método \_ship\_hit() coordina la respuesta cuando un alienígena choca contra una nave. Dentro de \_ship\_hit(), el número de naves restantes se reduce en 1 ❶, tras lo cual vaciamos los grupos bullets y aliens ❷.

A continuación, creamos una nueva flota y centramos la nave ❸.(Dentro de un momento añadiremos el método center\_ship() a Ship.) A continuación, añadimos una pausa después de que se hayan realizado las actualizaciones de todos los elementos del juego, pero antes de que se haya dibujado ningún cambio en la pantalla, para que el jugador pueda ver que su nave ha sido golpeada ❹. La llamada a sleep() detiene la ejecución del programa durante medio segundo, el tiempo suficiente para que el jugador vea que el alienígena ha golpeado la nave. Cuando finaliza la función sleep(), la ejecución del código pasa al método \_update\_screen(), que dibuja la nueva flota en la pantalla.

En \_update\_aliens(), sustituimos la llamada a print() por una llamada a \_ship\_hit() cuando un alienígena golpea la nave:

**alien\_invasion.py**

def \_update\_aliens(self):  
 --snip--  
 if pygame.sprite.spritecollideany(self.ship, self.aliens):  
 self.\_ship\_hit()

Este es el nuevo método center\_ship(), que pertenece a *ship.py*:

**ship.py**

def center\_ship(self):  
 """Center the ship on the screen."""  
 self.rect.midbottom = self.screen\_rect.midbottom  
 self.x = float(self.rect.x)

Centramos la nave del mismo modo que en \_\_init\_\_(). Después de centrarla, reseteamos el atributo self.x, que nos permite seguir la posición exacta de la nave.

[anterior](c13_26.html)[Subtema 27 de 33: (Ver todo)](c13.html)[siguiente](c13_28.html)