### Crear una ventana Pygame y responder a las entradas del usuario

Crearemos una ventana Pygame vacía creando una clase que represente el juego. En tu editor de texto, crea un nuevo archivo y guárdalo como *alien\_invasion.py*; luego introduce lo siguiente:

**alien\_invasion.py**

import sys  
  
import pygame  
  
class AlienInvasion:  
 """Overall class to manage game assets and behavior."""  
  
 def \_\_init\_\_(self):  
 """Initialize the game, and create game resources."""  
❶ pygame.init()  
  
❷ self.screen = pygame.display.set\_mode((1200, 800))  
 pygame.display.set\_caption("Alien Invasion")  
  
 def run\_game(self):  
 """Start the main loop for the game."""  
❸ while True:  
 # Watch for keyboard and mouse events.  
❹ for event in pygame.event.get():  
❺ if event.type == pygame.QUIT:  
 sys.exit()  
  
 # Make the most recently drawn screen visible.  
❻ pygame.display.flip()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # Make a game instance, and run the game.  
 ai = AlienInvasion()  
 ai.run\_game()

En primer lugar, importamos los módulos sys y pygame. El módulo pygame contiene la funcionalidad que necesitamos para hacer un juego. Utilizaremos las herramientas del módulo sys para salir del juego cuando el jugador lo abandone.

*Alien Invasion* comienza como una clase llamada AlienInvasion. En el método \_\_init\_\_(), la función pygame.init() inicializa los ajustes de fondo que Pygame necesita para funcionar correctamente ❶. Luego llamamos a pygame.display.set\_mode() para crear una ventana de visualización ❷, en la que dibujaremos todos los elementos gráficos del juego. El argumento (1200, 800) es una tupla que define las dimensiones de la ventana del juego, que será de 1.200 píxeles de ancho por 800 píxeles de alto. (Puedes ajustar estos valores en función del tamaño de tu pantalla.) Asignamos esta ventana de visualización al atributo self.screen, para que esté disponible en todos los métodos de la clase.

El objeto que asignamos a self.screen se llama superficie. Una *surface* en Pygame es una parte de la pantalla donde se puede mostrar un elemento del juego. Cada elemento del juego, como un alienígena o una nave, es su propia superficie. La superficie devuelta por display.set\_mode() representa toda la ventana del juego. Cuando activemos el bucle de animación del juego, esta superficie se redibujará en cada pasada por el bucle, para que pueda actualizarse con cualquier cambio provocado por la entrada del usuario.

El juego se controla mediante el método run\_game(). Este método contiene un bucle while ❸ que se ejecuta continuamente. El bucle while contiene un bucle de eventos y código que gestiona las actualizaciones de la pantalla. Un *event* es una acción que el usuario realiza mientras juega, como pulsar una tecla o mover el ratón. Para que nuestro programa responda a los eventos, escribimos un bucle *event loop* a *listen* para los eventos y realizar las tareas adecuadas en función del tipo de eventos que se produzcan. El bucle for ❹ anidado dentro del bucle while es un bucle de eventos.

Para acceder a los eventos que Pygame detecta, utilizaremos la función pygame.event.get(). Esta función devuelve una lista de eventos que han tenido lugar desde la última vez que se llamó a esta función. Cualquier evento de teclado o ratón hará que se ejecute este bucle for. Dentro del bucle, escribiremos una serie de sentencias if para detectar y responder a eventos específicos. Por ejemplo, cuando el jugador haga clic en el botón de cierre de la ventana del juego, se detectará un evento pygame.QUIT y llamaremos a sys.exit() para salir del juego ❺.

La llamada a pygame.display.flip() ❻ indica a Pygame que haga visible la pantalla dibujada más recientemente. En este caso, simplemente dibuja una pantalla vacía en cada pasada por el bucle while, borrando la pantalla anterior para que sólo sea visible la nueva. Cuando movemos los elementos del juego, pygame.display.flip() actualiza continuamente la pantalla para mostrar las nuevas posiciones de los elementos del juego y ocultar las antiguas, creando la ilusión de un movimiento suave.

Al final del archivo, creamos una instancia del juego y llamamos a run\_game(). Colocamos run\_game() en un bloque if que sólo se ejecuta si se llama directamente al archivo. Cuando ejecutes este archivo *alien\_invasion.py*, deberías ver una ventana vacía de Pygame.

[anterior](c12_5.html)[Subtema 6 de 43: (Ver todo)](c12.html)[siguiente](c12_7.html)