### Crear una clase de botón

Como Pygame no tiene un método incorporado para crear botones, escribiremos una clase Button para crear un rectángulo relleno con una etiqueta. Puedes utilizar este código para crear cualquier botón en un juego. Aquí tienes la primera parte de la clase Button; guárdala como *button.py*:

**button.py**

import pygame.font  
  
class Button:  
 """A class to build buttons for the game."""  
  
❶ def \_\_init\_\_(self, ai\_game, msg):  
 """Initialize button attributes."""  
 self.screen = ai\_game.screen  
 self.screen\_rect = self.screen.get\_rect()  
  
 # Set the dimensions and properties of the button.  
❷ self.width, self.height = 200, 50  
 self.button\_color = (0, 135, 0)  
 self.text\_color = (255, 255, 255)  
❸ self.font = pygame.font.SysFont(None, 48)  
  
 # Build the button's rect object and center it.  
❹ self.rect = pygame.Rect(0, 0, self.width, self.height)  
 self.rect.center = self.screen\_rect.center  
  
 # The button message needs to be prepped only once.  
❺ self.\_prep\_msg(msg)

En primer lugar, importamos el módulo pygame.font, que permite a Pygame renderizar texto en la pantalla. El método \_\_init\_\_() toma los parámetros self, el objeto ai\_game, y msg, que contiene el texto del botón ❶. Establecemos las dimensiones del botón ❷, configuramos button\_color para colorear el objeto rect del botón en verde oscuro, y configuramos text\_color para renderizar el texto en blanco.

A continuación, preparamos un atributo font para renderizar el texto ❸. El argumento None indica a Pygame que utilice la fuente por defecto, y 48 especifica el tamaño del texto. Para centrar el botón en la pantalla, creamos un rect para el botón ❹ y establecemos su atributo center para que coincida con el de la pantalla.

Pygame trabaja con texto renderizando la cadena que quieres mostrar como una imagen. Por último, llamamos a \_prep\_msg() para que se encargue de esta renderización ❺.

Aquí tienes el código de \_prep\_msg():

**button.py**

def \_prep\_msg(self, msg):  
 """Turn msg into a rendered image and center text on the button."""  
❶ self.msg\_image = self.font.render(msg, True, self.text\_color,  
 self.button\_color)  
❷ self.msg\_image\_rect = self.msg\_image.get\_rect()  
 self.msg\_image\_rect.center = self.rect.center

El método \_prep\_msg() necesita un parámetro self y el texto a representar como imagen (msg). La llamada a font.render() convierte el texto almacenado en msg en una imagen, que luego almacenamos en self.msg\_image ❶. El método font.render() también toma un valor booleano para activar o desactivar el antialiasing (*antialiasing* hace que los bordes del texto sean más suaves). Los argumentos restantes son el color de fuente y el color de fondo especificados. Establecemos el antialiasing en True y el fondo del texto en el mismo color que el botón. (Si no incluyes un color de fondo, Pygame intentará renderizar la fuente con un fondo transparente).

Centramos la imagen del texto en el botón creando un rect a partir de la imagen y fijando su atributo center para que coincida con el del botón ❷.

Por último, creamos un método draw\_button() al que podemos llamar para mostrar el botón en pantalla:

**button.py**

def draw\_button(self):  
 """Draw blank button and then draw message."""  
 self.screen.fill(self.button\_color, self.rect)  
 self.screen.blit(self.msg\_image, self.msg\_image\_rect)

Llamamos a screen.fill() para dibujar la parte rectangular del botón. Luego llamamos a screen.blit() para dibujar la imagen del texto en la pantalla, pasándole una imagen y el objeto rect asociado a la imagen. Esto completa la clase Button.

[anterior](c14_2.html)[Subtema 3 de 29: (Ver todo)](c14.html)[siguiente](c14_4.html)