Desconocido

### Crear una animación básica

Ya hemos visto cómo crear imágenes estáticas que no se mueven. Ahora nos centraremos en la creación de animaciones.

La animación no es necesariamente una especialidad del módulo tkinter , pero puede manejar lo básico. Por ejemplo, podemos crear un triángulo relleno y hacer que se mueva por la pantalla utilizando este código:

>>> import time  
>>> from tkinter import \*  
>>> tk = Tk()  
>>> canvas = Canvas(tk, width=400, height=200)  
>>> canvas.pack()  
>>> canvas.create\_polygon(10, 10, 10, 60, 50, 35)  
>>> for x in range(1, 61):  
 canvas.move(1, 5, 0)  
 tk.update()  
 time.sleep(0.05)

Cuando ejecutes este código, el triángulo empezará a moverse por la pantalla hasta el final de su recorrido, como en la [Figura 10-18 .](ch10.xhtml#ch10fig18)

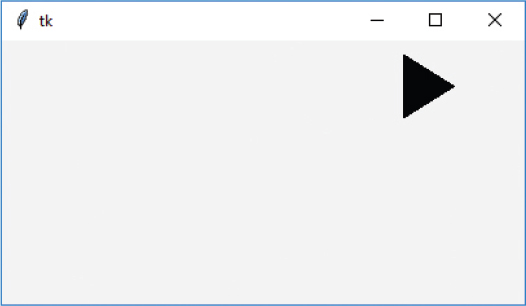


Figura 10-18*: Triángulo en* movimiento

Como antes, hemos utilizado las tres primeras líneas después de importar tkinter para realizar la configuración básica para visualizar un lienzo. Creamos el triángulo con la llamada a la función canvas.create\_polygon(10, 10, 10, 60, 50, 35) .

**NOTA**

*Cuando introduzcas esta línea, se imprimirá un número en la pantalla. Se trata de un identificador para el polígono. Podemos utilizarlo para referirnos a la forma más adelante, como se describe en el siguiente ejemplo.*

A continuación, creamos un sencillo bucle for para contar de 1 a 61, comenzando por for x in range(1, 61) .

El bloque de código dentro del bucle mueve el triángulo por la pantalla. La función canvas.move moverá cualquier objeto dibujado añadiendo valores a sus coordenadas x y y . Por ejemplo, con canvas.move(1, 5, 0) , movemos el objeto con ID 1 (el identificador del triángulo -ver la Nota anterior) 5 píxeles a lo ancho y 0 píxeles a lo bajo. Para volver a moverlo, podríamos utilizar la llamada a la función canvas.move(1, -5, 0) .



La función tk.update() obliga a tkinter a actualizar la pantalla (redibujarla). Si no utilizáramos update , tkinter esperaría a que terminara el bucle antes de mover el triángulo, lo que significa que lo verías saltar a la última posición, en lugar de moverse suavemente por el lienzo. La última línea del bucle, time.sleep(0.05) , indica a Python que espere una vigésima de segundo (0,05 segundos) antes de continuar.

Para hacer que el triángulo se mueva en diagonal por la pantalla, podemos modificar este código llamando a move(1, 5, 5) . Cierra el lienzo y crea un nuevo archivo ( **Archivo ▸ Nuevo Archivo** ) para el siguiente código:

import time  
from tkinter import \*  
tk = Tk()  
canvas = Canvas(tk, width=400, height=400)  
canvas.pack()  
canvas.create\_polygon(10, 10, 10, 60, 50, 35)  
for x in range(0, 60):  
 canvas.move(1, 5, 5)  
 tk.update()  
 time.sleep(0.05)

Este código difiere del original en dos aspectos:

* Hicimos la altura del lienzo 400, en lugar de 200, con canvas = Canvas(tk, width=400, height=400) .
* Añadimos 5 a las *coordenadas* *x* e *y* del triángulo con canvas.move(1, 5, 5) .

[La Figura 10-19](ch10.xhtml#ch10fig19) muestra la posición del triángulo al final del bucle, después de guardar el código y ejecutarlo.

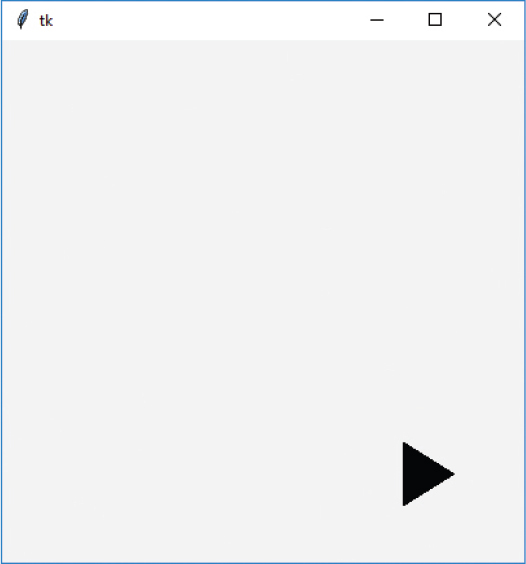


Figura 10-19*: El triángulo se ha desplazado a la parte inferior de la pantalla.*

Para mover el triángulo diagonalmente hacia arriba en la pantalla hasta su posición inicial, utiliza (-5, -5). Añade este código al final del archivo:

>>> for x in range(0, 60):  
 canvas.move(1, -5, -5)  
 tk.update()  
 time.sleep(0.05)

Después de ejecutar este código, el triángulo se moverá hacia atrás hasta donde empezó.

[anterior](ch10_12.html)[Subtema 13 de 17: (Ver todo)](ch10.html)[siguiente](ch10_14.html)