Experimentando con Sonidos

El código

```
1  import pygame, sys, random
2  import pygame.locals as VARIABLES_GLOBALES_DEL_JUEGO
3  import pygame.event as EVENTOS_DEL_JUEGO
4  import pygame.time as TIEMPO_DEL_JUEGO
5
6  anchoVentana = 600
7  altoVentana = 650
8
9  pygame.init()
10 superficieDeDibujo = pygame.display.set_mode((anchoVentana, altoVentana))
11 pygame.display.set_caption('Sonidos')
12
13 botones = []
14 botonDetener = { "imagen" : pygame.image.load("recursos/iconos/detener.png"), "posicion" : (275, 585)}
15
16 posicionDelRaton = None
17 volumen = 1.0
```

Las líneas 1-14 deberían ser muy familiares: primero tenemos las declaraciones de importación en las líneas 1-5, luego establecemos las propiedades de las ventanas en las líneas 6-11, y finalmente creamos un par de variables para usarlas más tarde en las líneas 13-17. Si miras la línea 13, verás la variable botones. Cuando estemos listos para comenzar a crear los botones, añadiremos algunos diccionarios a esta lista para que podamos realizar un seguimiento de todos los botones de la interfaz de sonido a crearse. En la siguiente línea, tenemos el diccionario botonDetener. Cuando creamos este botón, se comportará como el resto de los botones de la interfaz, excepto que detendrá todos los sonidos actuales. Puesto que es único, el botón detener tiene su propia variable.

```
83  while True:
84
85    superficieDeDibujo.fill((255,255,255))
86
87    posicionDelRaton = pygame.mouse.get_pos()
88
89    for event in EVENTOS_DEL_JUEGO.get():
90
91      if event.type == pygame.KEYDOWN:
92
93        if event.key == pygame.K_ESCAPE:
94          salirDelJuego()
95
96      if event.type == VARIABLES_GLOBALES_DEL_JUEGO.QUIT:
97        salirDelJuego()
98
99      if event.type == pygame.MOUSEBUTTONUP:
100       manejarClick()
101
102   dibujarBotones()
103   verificarVolumen()
104   dibujarVolumen()
105
106   pygame.display.update()
```

En las líneas 83-106 tenemos un bucle familiar. Su aspecto es mucho menor que el de la última vez: es porque se ha dividido todo el código que podríamos poner en el ciclo principal en funciones independientes. Al tener funciones especializadas, podemos escribir un programa que funciona y se ve muy bien. Al igual que antes, nuestro bucle principal es responsable de limpiar la pantalla (línea 85); manejo de eventos de ratón, teclado y sistema (líneas 89-100); y llamar a funciones para dibujar en la superficie de dibujo (líneas 102-106).

El mezclador de Pygame

Si usamos sonidos en Pygame, utilizaremos su mezclador integrado. Se puede pensar en el mezclador como su equivalente en el mundo real: todos los sonidos del sistema pasan a través de él (o en nuestro caso, a través del juego). Cuando hay un sonido en el mezclador, se lo puede ajustar

de varias maneras, el volumen es una de ellas. Cuando el mezclado está terminado, el sonido pasa a través de una salida, los parlantes. Por lo tanto, antes de empezar a cargar o reproducir cualquier sonido, tenemos que inicializar el mezclador, al igual que tenemos que inicializar Pygame (la línea 19.)

El primer sonido

Se puede reproducir sonidos de dos maneras diferentes: de forma *directa o streaming*, es decir, el sonido se está reproduciendo mientras se está cargando, ó a través de un *objeto de sonido*, que carga el sonido, lo almacena en la memoria de la computadora, y luego lo reproduce. Primero se añadirá un sonido ambiente con audio de fondo de una granja. El audio de fondo se realiza en bucles sin ningún tipo de interacción del usuario, y el audio streaming puede configurarse para realizar algunos bucles.

```
19  pygame.mixer.init()
20  pygame.mixer.music.load('recursos/sonidos/OGG/granja.ogg')
21  pygame.mixer.music.play(-1)
```

Antes de poder reproducir música, debemos cargarla (**load**): en la línea 20 cargamos el archivo de audio granja.ogg. Esto carga el audio en el mezclador, pero no se reproducirá de inmediato. En la línea 21 llamamos pygame.mixer.music.**play**(**-1**), que comienza a reproducir el archivo de sonido. El número que pasamos es el número de veces que queremos que el sonido se repita (bucles) antes de que deje de tocar. Hemos pasado -1, lo que significa que se repetirá para siempre, o hasta que lo detengamos.

Botones

```
72  botones.append({ "imagen" : pygame.image.load("recursos/iconos/oveja.png"), "posicion" : (25, 25), "sonido" : pygame
73  botones.append({ "imagen" : pygame.image.load("recursos/iconos/gallo.png"), "posicion" : (225, 25), "sonido" : pygam
74  botones.append({ "imagen" : pygame.image.load("recursos/iconos/cerdo.png"), "posicion" : (425, 25), "sonido" : pygam
75  botones.append({ "imagen" : pygame.image.load("recursos/iconos/raton.png"), "posicion" : (25, 225), "sonido" : pygam
76  botones.append({ "imagen" : pygame.image.load("recursos/iconos/caballo.png"), "posicion" : (225, 225), "sonido" : py
77  botones.append({ "imagen" : pygame.image.load("recursos/iconos/perro.png"), "posicion" : (425, 225), "sonido" : pyga
78  botones.append({ "imagen" : pygame.image.load("recursos/iconos/vaca.png"), "posicion" : (25, 425), "sonido" : pygame
79  botones.append({ "imagen" : pygame.image.load("recursos/iconos/gallina.png"), "posicion" : (225, 425), "sonido" : py
80  botones.append({ "imagen" : pygame.image.load("recursos/iconos/gato.png"), "posicion" : (425, 425), "sonido" : pygam
```

Los botones están incluidos en la carpeta *recursos/iconos/*. Cada botón tiene una silueta de un animal y hará el sonido que hace este animal al apretarlo con el ratón, para ello vamos a usar listas y diccionarios. Si observa las líneas 71-80, verá que cada línea crea un nuevo diccionario para cada animal. Cada diccionario tiene tres *claves*. El primero es la **imagen**, que cargará la imagen para el botón con pygame.image.**load**(). Esto ahorra tiempo cuando tenemos que dibujar algo muchas veces, y como la imagen nunca cambia, tiene sentido tenerla allí. La siguiente clave es **posición:** contiene las coordenadas X e Y de donde se dibujarán los botones. La última clave es **sonido**, que carga un sonido. Aquí estamos cargando los sonidos como *objetos*, lo que significa que son esencialmente independientes en términos de cómo funcionan. Con el audio de fondo que cargamos antes, pasamos los datos directamente a través del mezclador y lo reproducimos a través de este último. Un objeto de sonido, tiene funciones que nos permiten controlar el audio. Por ejemplo, podríamos llamar a sound.**play**() para reproducir el sonido, o podríamos llamar a sound.stop(), pero sólo se aplicaría al sonido en el que se llama esas funciones: si tuviéramos dos sonidos reproduciendose al mismo tiempo y detuvíeramos sólo uno, el otro seguía reproduciéndose.

dibujarBotones()

En el bucle principal en la línea 102, se llama a la función **dibujarBotones()**, que se encuentra en las líneas 23-28. En la línea 25, el bucle **for** navega a través de la lista de botones inicializada en la línea 13. Para cada diccionario que encuentre en la lista, dibujará un botón sobre la superficie,

utilizando las propiedades que encuentra y el proceso llamado blitting. Esto ocurre en la línea 26. El **blitting** es esencialmente una manera elegante de decir 'pegar', tomamos los píxeles de la superficie y luego los cambiamo para que sean iguales a la imagen que estamos añadiendo. Esto significa que se pierde todo lo que estaba debajo de la zona que está siendo pegada. Algo parecido a recortar fotos de un periódico y pegarlas en un pedazo de papel.

manejarClick()

En las líneas 89-100 tenemos código que maneja eventos del juego. En las líneas 99-100 se encuentra el evento **MOUSEBUTTONUP**. ¿Por qué utilizamos MOUSEBUTTONUP y no MOUSECLICK?, para que un botón del ratón suba, tiene que haber bajado primero, lo que significa que el usuario debe haber hecho clic. Si se ha hecho clic en el ratón, llamamos a la función **manejarClick()**, que está en las líneas 38-55. Al igual que cuando dibujamos los botones, vamos a trabajar a través de la lista de botones para averiguar dónde están en la superficie. Si el ratón hace clic en un botón, se repruducirá su sonido, de lo contrario no sucede nada.

```
38 def manejarClick():
39
40   global posicionDelRaton, volumen
41
42   for boton in botones:
43
44     tamanioDelBoton = boton['imagen'].get_rect().size
45     posicionDelBoton = boton['posicion']
46
47     if posicionDelRaton[0] > posicionDelBoton[0] and posicionDelRaton[0] < posicionDelBoton[0] + tama
48
49       if posicionDelRaton[1] > posicionDelBoton[1] and posicionDelRaton[1] < posicionDelBoton[1] + ta
50         boton['sonido'].set_volume(volumen)
51         boton['sonido'].play()
```

Con el fin de comprobar si se ha pulsado o no un botón, necesitamos saber tres cosas: 1) la posición de cada botón, 2) el tamaño de ese botón y 3) donde estaba el ratón cuando se hizo clic. Si las coordenadas del ratón son mayores que las coordenadas X e Y de la imagen del botón, pero menores que las coordenadas X-Y más el ancho y la altura de la imagen (líneas 47 y 49), podemos estar seguros de que se ha hecho clic en el botón y por lo tanto reproducir el sonido (líneas 50 y 51); de lo contrario, el ratón estaba fuera del botón. Los botones son círculos, pero comprobamos el clic dentro de un cuadrado que rodea el botón. Lo hacemos porque el resultado es casi exactamente el mismo y el código más simple que para comprobar un círculo o una forma irregular. Este es un objeto de sonido, no un streaming de sonido, cuando se hace un clic en él, se reproduce a través del mezclador. Tenga en cuenta que el mezclador no tiene control sobre ese sonido específico.

```
53     if posicionDelRaton[0] > botonDetener['posicion'][0] and posicionDelRaton[0] < botonDetener['posi
54       if posicionDelRaton[1] > botonDetener['posicion'][1] and posicionDelRaton[1] < botonDetener['po
55         pygame.mixer.stop()
```

En la línea 14 tenemos un diccionario llamado **botonDetener**; A diferencia del resto de los botones, no tiene sonido, solo una imagen y un elemento de posición. Debajo de todo el código utilizado para manejar los botones de sonido, tenemos código que sólo se ocupa del botón detener: en la línea 28 dibujamos el botón detener, justo después de haber dibujado todos los otros, y en las líneas 53-55 verificamos específicamente si se ha pulsado dicho botón.

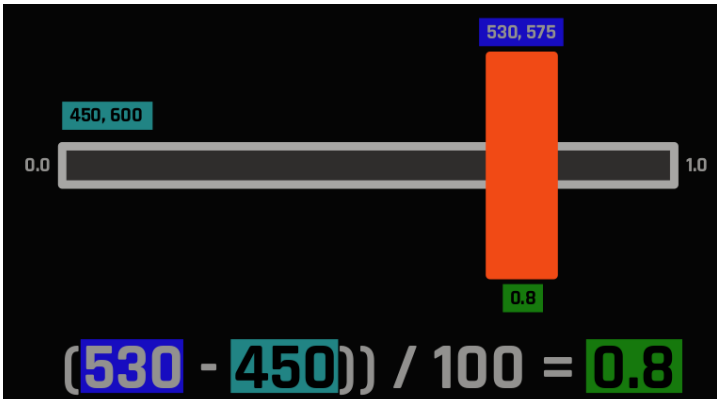
dibujarVolumen() y verificarVolumen()

Cada uno de nuestros objetos de sonido tiene una función **set_volume()** que permiten pasar un valor entre 0,0 y 1,0. 0,0 es silencio, mientras que 1,0 es volumen completo. Si se pasa un valor mayor que 1.0, se convertirá en 1.0, y si se pasa un valor menor que 0.0, se convertirá en 0.0.

```
30  def dibujarVolumen():
31
32    pygame.draw.rect(superficieDeDibujo, (229, 229, 229), (450, 610, 100, 5))
33
34    posicionDeVolumen = (100 / 100) * (volumen * 100)
35
36    pygame.draw.rect(superficieDeDibujo, (204, 204, 204), (450 + posicionDeVolumen, 600, 10, 25))
```

Para comenzar, necesitamos hacer un control deslizante de volumen. En las líneas 30-36 dibujamos dos rectángulos. El primer rectángulo representa el rango de 0,0 a 1,0, y el segundo rectángulo es un indicador del volumen actual. Cuando se inicia el juego, el volumen (en la línea 17 del código) se establece en 1,0, por lo que el control deslizante esta completamente a la derecha.

```
57  def verificarVolumen():
58
59    global posicionDelRaton, volumen
60
61    if pygame.mouse.get_pressed()[0] == True:
62
63      if posicionDelRaton[1] > 600 and posicionDelRaton[1] < 625:
64        if posicionDelRaton[0] > 450 and posicionDelRaton[0] < 550:
65          volumen = float((posicionDelRaton[0] - 450)) / 100
```



Para cambiar las cosas. Justo antes de llamar a **dibujarVolumen()** en la línea 104, llamamos **verificarVolumen()** en la línea 103. Aquí se verifica la posición actual del ratón y si el botón izquierdo del ratón se ha presionado o no (línea 61). Si es así, el usuario ha arrastrado el indicador al nivel en el que quieren que esté el sonido. Entonces, cuando se llama a la función **dibujarVolumen()**, el indicador se dibujará en la posición correcta. Ahora, al hacer clic en un sonido, se ajustará al nivel que se ha elegido con la función **set_volume()** en el objeto de sonido, que se puede ver en la línea 50.

Asignación: Escribe el código usando lo aprendido para activar los sonidos de los animales usando las teclas 1-9 del teclado.

Observaciones: MP3 es un formato muy popular para reproducir música y sonidos, la desventaja es que es una tecnología patentada. Como tal, Pygame y otras bibliotecas populares no soportan MP3. Por lo tanto, vamos a utilizar OGG, un formato de sonido abierto. Cualquier archivo puede convertirse de MP3 a OGG con infinidad de programas pagados y libres.