Taller de sonidos y texto con Pygame

2013
VídeoJuegos con Python
PyGroupUD
Proyecto Curricular de Ingeniería de Sistemas

Taller de sonidos y texto con Pygame

Índice de contenido

Introducción

En esta entrega trabajaremos con el manejo de sonidos y la renderización de texto con pygame. Los recursos usados para este ejercicio son 6 imágenes, una de un fondo, una del enemigo de nuestro héroe (la chichilla) y cuatro de nuestro héroe (el mamut), se sugieren los siguientes links:

- chichilla: https://www.dropbox.com/s/qx3fnqnf4v3k2e8/chinchilla.png
- fondo: https://www.dropbox.com/s/9ydy1wrh6mysiis/fondo.jpg
- héroe: https://www.dropbox.com/s/zomt7b5s6ioryeg/Mammooth.png, https://www.dropbox.com/s/5528gmvo82lnq5m/Mammoth_Back.png, https://www.dropbox.com/s/0e5rpv0fuh2nakq/Mammoth_Happy.png y https://www.dropbox.com/s/1gh45hqd74pv8a5/Mammoth_Seated.png

Y un zip que contiene el sonido en formato wav que se cargara en el caso quil.py"e nuestro héroe colisione con una chinchilla:

- pierde_vida: https://www.dropbox.com/s/5qb4zq2s31swi3n/pierde_vida.wav.zip

Creando la utilería del juego

Para esta entrega crearemos un juego diferente al que veníamos trabajando, el juego consiste en un Mamut que tratara de cruzar un escenario el cual cruzan las chinchillas, la idea es que nuestro Mamut cruce sin colisionar sin las chichillas. Inicialmente crearemos un script "util.py" en el cual crearemos dos funciones, la primera encapsula la carga de imágenes y la segunda la carga de sonidos.

```
import pygame

def cargar_imagen(nombre, optimizar=False):
    imagen = pygame.image.load(nombre)
```

```
    if optimizar:
        return imagen.convert()
    else:
        return imagen.convert_alpha()

def cargar_sonido(nombre):
    return pygame.mixer.Sound(nombre)
```

De estas dos funciones la que define algo nuevo es cargar_sonido, en esta se utiliza el módulo mixer de pygame con el cual cargaremos el sonido en base al nombre del archivo de sonido.

```
pygame.mixer.Sound(nombre)
```

Definiendo nuestros enemigos

Veamos como queda la clase que define un enemigo (la chinchilla):

```
import pygame
from pygame.sprite import Sprite
from pygame import *
import util

class Villano(Sprite):
	def __init__(self,coord,vel):
		Sprite.__init__(self)
		self.image = util.cargar_imagen('imagenes/chinchilla.png')
		self.rect = self.image.get_rect()
		self.rect.move_ip(coord[0], coord[1])
		self.dir = "l"
		self.velocidad=vel

	def update(self):
		if self.dir == "l":
			self.rect.x -= self.velocidad
		elif self.dir == "r":
```

```
                self.rect.x += self.velocidad
            if self.rect.x<=0:
                self.dir="r"
            if self.rect.x>=608:
                self.dir="l"
```

En esta clase la imagen de la chinchilla es cargada usando la función cargar_imagen de la librería util de nuestro juego:

```
self.image = util.cargar_imagen('imagenes/chinchilla.png')
```

El comportamiento de este enemigo será moverse a lo ancho de nuestra ventana principal lo cual es implementado en el método update del villano, en el cual dependiendo de la dirección que tiene en un momento determinado se incrementa o disminuye su valor de posición en X:

```
if self.dir == "l":
      self.rect.x -= self.velocidad
elif self.dir == "r":
      self.rect.x += self.velocidad
```

Y en el caso que este alcance los límites de nuestra ventana cambia de dirección:

```
if self.rect.x<=0:
      self.dir="r"
if self.rect.x>=608:
      self.dir="l"
```

Creando la clase del héroe:

Veamos como está definida la clase héroe de nuestro juego, en está le daremos al mamut la posibilidad de cambiar de imagen cuando cambie de estado la información que este maneja, para esto cargaremos en una lista las cuatro imágenes que representaran los estados seran las siguientes:

Ilustración 1: mamut bajando en pantalla

Ilustración 2: mamut subiendo en pantalla

Ilustración 3: mamut colisionando

Ilustración 4: mamut sin puntos de vida

Esto queda definido en el método __init__ de nuestra clase:

```
def __init__(self):
	Sprite.__init__(self)
	self.puntos = 0
	self.vida = 100
	self.estado = "bajando"
	self.imagenes = [util.cargar_imagen('imagenes/Mammooth.png'),
			 util.cargar_imagen('imagenes/Mammoth_Happy.png'),
			 util.cargar_imagen('imagenes/Mammoth_Back.png'),
			 util.cargar_imagen('imagenes/Mammoth_Seated.png')]
		self.image = self.imagenes[0]
		self.rect = self.image.get_rect()
		self.rect.move_ip(200, 10)
```

Y el comportamiento de nuestro héroe corresponde a la respuestas de eventos del teclado sobre las flechas K_UP, K_DOWN, K_LEFT y K_RIGHT ante los cuales el mamut modifica su posición en pantalla y la imagen en caso de cambiar de dirección.

```
def update(self):
	teclas = pygame.key.get_pressed()
	if self.vida > 0:
		if teclas[K_LEFT] and self.rect.x>=10:
			self.rect.x -= 10
		elif teclas[K_RIGHT] and self.rect.x<=640-self.rect.width:
			self.rect.x += 10
		if teclas[K_UP] and self.rect.y>=10:
			self.rect.y -= 10
			self.image = self.imagenes[2]
			if self.rect.y==0 and self.estado == "subiendo":
				self.puntos = self.puntos + 1
				self.estado = "bajando"
		elif teclas[K_DOWN] and self.rect.y<=480-self.rect.height:
			self.rect.y += 10
			self.image = self.imagenes[0]
			if self.rect.y==410 and self.estado == "bajando":
				self.puntos = self.puntos + 1
				self.estado = "subiendo"
```

```
        else:
                self.image = self.imagenes[3]
```

Resultando toda la clase “Heroe” de la siguiente manera:

```
class Heroe(Sprite):
        def __init__(self):
                Sprite.__init__(self)
                self.puntos = 0
                self.vida = 100
                self.estado = "bajando"
                self.imagenes = [util.cargar_imagen('imagenes/Mammooth.png'),
                                 util.cargar_imagen('imagenes/Mammoth_Happy.png'),
                                 util.cargar_imagen('imagenes/Mammoth_Back.png'),
                                 util.cargar_imagen('imagenes/Mammoth_Seated.png')]
                self.image = self.imagenes[0]
                self.rect = self.image.get_rect()
                self.rect.move_ip(200, 10)

        def update(self):
                teclas = pygame.key.get_pressed()
                if self.vida > 0:
                        if teclas[K_LEFT] and self.rect.x>=10:
                                self.rect.x -= 10
                        elif teclas[K_RIGHT] and self.rect.x<=640-self.rect.width:
                                self.rect.x += 10
                        if teclas[K_UP] and self.rect.y>=10:
                                self.rect.y -= 10
                                self.image = self.imagenes[2]
                                if self.rect.y==0 and self.estado == "subiendo":
                                        self.puntos = self.puntos + 1
                                        self.estado = "bajando"
                        elif teclas[K_DOWN] and self.rect.y<=480-self.rect.height:
                                self.rect.y += 10
                                self.image = self.imagenes[0]
                                if self.rect.y==410 and self.estado == "bajando":
                                        self.puntos = self.puntos + 1
                                        self.estado = "subiendo"
                else:
                        self.image = self.imagenes[3]
```

Armando el script principal

Ahora armemos el script principal, en este además de iniciar pygame, debemos iniciar el mixer que se encargara del soporte de sonidos para el juego.

```
pygame.mixer.init()
```

El otro manejo que hacemos de sonidos en este script es la carga del sonido usando la función cargar_sonido de “util.py”.

```
pierde_vida = util.cargar_sonido('sonidos/pierde_vida.wav')
```

Y la reproducción del sonido como tal se da en el ciclo que manejara las colisiones de los enemigos con el mamut.

```
pierde_vida.play()
```

Resultando nuestro script de la siguiente forma:

```
import sys, pygame, util
from pygame.locals import *
from heroe import *
from villano import *
from random import *

SCREEN_WIDTH = 640
SCREEN_HEIGHT = 480
ICON_SIZE = 32

def game():
    pygame.init()
    pygame.mixer.init()
    screen = pygame.display.set_mode( (SCREEN_WIDTH,SCREEN_HEIGHT) )
    pygame.display.set_caption( "El Cruce" )
    background_image = util.cargar_imagen('imagenes/fondo.jpg');
    pierde_vida = util.cargar_sonido('sonidos/pierde_vida.wav')
    pygame.mouse.set_visible( False )
```

```
    temporizador = pygame.time.Clock()
    heroe = Heroe()
    villano = [Villano((100,80),randint(1,10)),
               Villano((100,150),randint(1,10)),
               Villano((200,220),randint(1,10)),
               Villano((300,300),randint(1,10)),
               Villano((100,350),randint(1,10))]

    while True:
        fuente = pygame.font.Font(None,25)
        texto_puntos = fuente.render("Puntos: "+str(heroe.puntos),1,(0,0,0))
        texto_vida = fuente.render("Vida: "+str(heroe.vida),1,(0,0,0))

        heroe.update()
        for n in villano:
            n.update()

        for n in villano:
            if heroe.rect.colliderect(n.rect):
                heroe.image = heroe.imagenes[1]
                pierde_vida.play()
                if heroe.vida > 0:
                    heroe.vida=heroe.vida-1
                n.velocidad=randint(1,10)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()
        screen.blit(background_image, (0,0))
        screen.blit(heroe.image, heroe.rect)
        screen.blit(texto_vida,(400,450))
        screen.blit(texto_puntos,(100,450))
        for n in villano:
            screen.blit(n.image, n.rect)
        pygame.display.update()
        pygame.time.delay(10)


if __name__ == '__main__':
      game()
```

Veamos que el ciclo principal del juego se hace un manejo de texto en el cual en la primera línea se define una fuente por defecto para pygame, con un tamaño de 25 puntos, luegos se

renderizan dos textos, en los cuales se les especifica como primer parámetro la cadena de texto a renderizar, como segundo parámetro el booleano que determina el antialias para los bordes de la fuente y el tercero una tupla que expresa el color en formato (red, green, blue).

```
fuente = pygame.font.Font(None,25)
texto_puntos = fuente.render("Puntos: "+str(heroe.puntos),1,(0,0,0))
texto_vida = fuente.render("Vida: "+str(heroe.vida),1,(0,0,0))
```

esto crea un surface que luego es actualizado sobre la ventana como si tratara de una imagen mediante el método blit:

```
screen.blit(texto_vida,(400,450))
screen.blit(texto_puntos,(100,450))
```

el resultado de la ejecución de nuestro script es el siguiente:

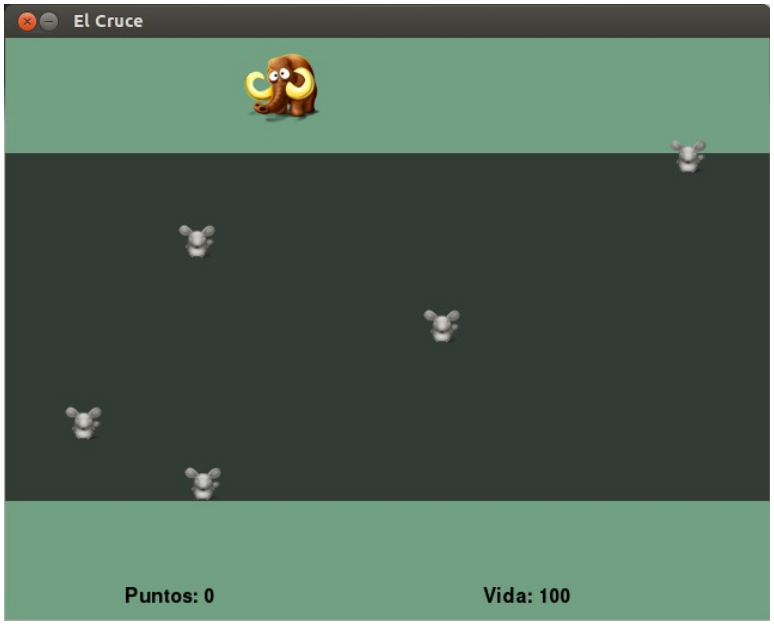

Ilustración 5: Inicio del juego

Ilustración 6: Mamut subiendo en el juego

Trabajo a realizar

Sigue las instrucciones de modificación del ejercicio indicadas en cada uno de los puntos siguientes y genera un zip que contenga los archivos solicitados (scripts de python) y los recursos (imágenes, sonidos, archivos de texto, etc) necesarios para su ejecución.

1. Observa que cuando se producen colisiones el mamut pierde puntos de vida durante el tiempo que dura la colisión, corrige este problema haciendo que mediante un estado en_colision del mamut solo se pierda un punto de vida cada ves que colisione. Guarda el script como taller04_01.py
2. Agrega bonos que permitan que el mamut recupere puntos de vida cuando los capture. Guarda el script como taller04_02.py
3. Agrega bonos de inmunidad que durante un periodo de tiempo de 10

segundos modifiquen el color del mamut a azul en el cual el mamut al colisionar con las chinchillas no pierda vidas, en lugar de esto la chinchilla con la que colisione debe desaparecer del juego durante el tiempo que dure la inmunidad y reaparecer cuando esta termine, el tiempo de inmunidad debe ser renderizado en pantalla. Guarda el script como taller04_03.py

Bibliografía consultada:

“Pygame documentatión”.
<http://www.pygame.org/docs/index.html>
(01 Sep 2013)

“Python documentation”.
<http://www.python.org/doc/>
(01 Sep 2013)