Pygame

Autor: Pablo Pizarro, @ppizarror.com

La librería Pygame permite crear aplicaciones 2D y 3D en Python al proveer al usuario de multitud de funciones para dibujar en pantalla, cargar sonidos o reconocer distintos tipos de input (teclado, ratón, controles externos).

El esquema de funcionamiento de cualquier aplicación en Python es típicamente [1] el siguiente:

Código 1: Ejemplo en Python.

```python
# Importación de librerías
import pygame
from pygame.locals import *

# Se inician módulos
pygame.init()
os.environ['SDL_VIDEO_CENTERED'] = '1' # Centra la ventana

...

# Se cargan fuentes, sonidos e imágenes
sonido_muere = pygame.mixer.Sound('sounds/dead.wav')  # Sonidos
fuente_juego = pygame.font.Font('font/8bit.ttf', 40)  # Fuentes
imagen_mono = pygame.image.load('images/imagen1.png') # Imágenes
icono_juego = pygame.image.load('images/icon.png')

...

# Se crea la pantalla
surface = pygame.display.set_mode((640, 480)) # Crea una ventana de 640 por 480 pixeles, retorna el CANVAS para dibujar en pantalla
pygame.display.set_caption('Titulo del juego') # Título de la ventana
pygame.display.set_icon(icono_juego) # Ícono de la ventana

...

# Se crea el reloj del juego
clock = pygame.time.Clock()

...

```

[1] En el sentido que, a pesar que pueden haber variaciones, a *Grosso modo* presenta el esquema presentado.

```
# Se crean los modelos
jugador1 = ModeloA(....)
jugador2 = ModeloB(....)
Nube = NubeAleatoria(...)

...

# Bucle principal
while True:

# Setea el reloj
clock.tick(FPS)

# Busca eventos de aplicación
for event in pygame.event.get():
if event.type == QUIT: # Cierra la aplicación
exit()
...

# Ve teclas pulsadas e interactúa con los modelos
keys = pygame.key.get_pressed()
if keys[K_w]:
jugador1.subir()
...

# Los modelos interactúan entre sí, etc
jugador1.colisiona(jugador2)
...

# Se pinta el fondo
surface.fill(COLOR_BLACK) # También se puede dibujar una imagen

# Se dibujan los modelos
jugador1.draw()
jugador2.draw()
...

# Vuelca todo lo dibujado en pantalla
pygame.display.flip()
```

Imágenes

Las imágenes (archivos externos como **png**, **gif**) se cargan utilizando la instrucción *pygame.image.load(dir)*. Esta retorna un objeto del tipo **surface** (el cual conceptualmente es como un canvas).

Para dibujar todo lo que es imagen en pantalla se debe volcar el *surface* generado en el **surface** de la pantalla (es como pegar un canvas pequeño en uno más grande), esto se hace con la función **blit**, la cual toma por parámetro el surface de la imagen cargada y una posición `(x,y)` tal como se muestra a continuación:

Código 2: Pegar imágenes en la pantalla.

```
1 imagen_mono = pygame.image.load('images/imagen1.png')
2 surface.blit(imagen_mono, (30, 60)) # Recordar que surface era el objeto que retornaba al crear la
      ↪ ventana, ver línea 20 del código fuente 1
```

En este caso se dibuja la imagen cargada en pantalla en las coordenadas `30, 60` (30 píxeles horizontales y 60 píxeles verticales, considerando el `(0,0)` como la esquina superior izquierda).

Para ver la lista completa de funciones revisa este enlace: https://www.pygame.org/docs/ref/image.html

Líneas, polígonos, etc

Pygame ofrece la sub-librería **pygame.draw** para realizar una variedad de acciones. A continuación se presenta una lista de funciones más comunes:

1. Dibujar un polígono: *pygame.draw.polygon(surface, color, pointlist, width=0).*
   Pointlist es la lista de puntos del polígono, por ejemplo: `[(10, 40), (60, 30), (40, 60) ..]`, **color** es el color del polínono y **width** es el ancho del borde del polígono, si *width=0* entonces el polígono tiene un relleno del color deseado.

2. Dibujar una línea: *pygame.draw.line(surface, color, [x1, y1], [x2, y2], width).*
   En este caso se dibuja una línea recta entre `(x1, y1)` y `(x2, y2)` de un color **color** y un ancho **width** (en pixeles).

Para ver la lista completa de funciones revisa este enlace: https://www.pygame.org/docs/ref/draw.html

NOTA: Los colores están definidos por una tupla de 3 números de la forma `(R, G, B)` en donde `R` indica el valor de la componente roja (0 no hay rojo, 255 es el máximo valor), `G` es la componente verde y `B` es la componente azul. Ejemplos:

Código 3: Colores en Pygame.

```
1 COLOR_BLACK = (0, 0, 0)
2 COLOR_WHITE = (255, 255, 255)
3 COLOR_GRAY = (150, 150, 150)
4 COLOR_RED = (255, 0, 0)
5 COLOR_PURPLE = (128, 0, 128)
```

Sonidos

Los sonidos de cargan con la instrucción *pygame.mixer.Sound(dir)*, en donde se retorna un objeto del tipo Sound. Cada objeto posee varias funciones, entre las cuales se encuentran:

Código 4: Sonidos en Pygame.

```
sonido_muere = pygame.mixer.Sound('sounds/dead.wav')
sonido_muere.play(0)  # Sonido se reproduce 1 vez
sonido_muere.play(1)  # Sonido se reproduce 2 veces
sonido_muere.play(n)  # Sonido se reproduce n+1 veces
sonido_muere.play(-1) # Sonido se reproduce infinitas veces
sonido_muere.stop()   # Sonido se detiene
sonido_muere.pause()  # Sonido se pausa
```

Para ver la lista completa de funciones revisa este enlace: https://www.pygame.org/docs/ref/mixer.html

Fuentes

Para cargar una fuente se ejecuta la siguiente instrucción: *pygame.font.Font(dir, size)* en donde se carga la fuente en la ubicación **dir** con un tamaño de letra de **size** píxeles. Para crear un texto se debe generar su *surface* asociado, esto es, se debe renderizar una imagen de un texto con esa fuente para luego volcar ese *surface* con la función **blit** (ver subcapítulo de imágenes).

Para renderizar un texto se usa la función *font.render(text, antialias, color)* en donde **font** es la fuente cargada, **text** es el texto a renderizar, **antialias** indica si el texto a renderizar tiene *antialiasing* o no (por lo general se deja como 1) y **color** es el color del texto. Esta función retorna un *surface* el cual posteriormente se vuelca en pantalla con la función **blit**.

A continuación se muestra un ejemplo que escribe 'HOLA MUNDO' en el **surface** de la ventana en la posición `(100, 500)`:

Código 5: Fuentes con Python

```
# Se carga la fuente
fuente_juego = pygame.font.Font('font/8bit.ttf', 40)

# Se dibuja un texto
text = font.render('HOLA MUNDO', 1, COLOR_WHITE)

# Se vuelca el texto en pantalla
surface.blit(text, (100, 500))
```

Para ver la lista completa de funciones revisa este enlace: https://www.pygame.org/docs/ref/font.html