Disparar a monos

Realizado por: **Fernando Salamero**

Versión de pilas-engine: **1.3 en adelante**

Pilas: Un primer Juego

disparar_a_monos.py

Entre los ejemplos que incorpora Pilas, tenemos una buena primera aproximación con el proyecto **disparar_a_monos.py**, pues muestra un esqueleto general de cara al jugador.

El juego consiste en un pequeño torreta que ha de disparar a los monos que se generan al azar en pantalla y que intentan llegar hasta él para destruirlo. El juego incluye un sencillo marcador de puntuación, un control de sonido, un sistema de bonus y avisos de texto en pantalla. Tanto sonidos como imágenes están incorporados en Pilas, así que no necesitamos ningún recurso añadido.

En cada paso que demos para llegar a nuestro objetivo, pondremos lo que sea nuevo o lo que modifiquemos con fondo verde. **¿Estamos preparados?**

¡Escribamos código!

paso1.py

El primer paso que vamos a dar es situar el escenario, es decir, elegir un fondo, preparar un marcador, poner un control de sonido:

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

import pilasengine

pilas = pilasengine.iniciar()

# Usar un fondo estándar
pilas.fondos.Pasto()

# Añadir un marcador
puntos = pilas.actores.Puntaje(x=230, y=200, color=pilas.colores
puntos.magnitud = 40

# Añadir el conmutador de Sonido
pilas.actores.Sonido()

# Arrancar el juego
pilas.ejecutar()
```

Lo primero, como ya sabemos, es importar pilas-engine e inicializarlo, en este caso creando la ventana por defecto de 640x480 pixeles.

A continuación, establecemos como fondo uno de los incorporados en el motor a través del módulo `pilas.fondos`. El elegido, `Pasto`, tiene una textura verde cercana a la selvática jungla donde viven los monos :-)...

Toca el turno del marcador. Entre los actores predefinidos en el módulo `pilas.actores` tenemos uno que se encarga precisamente de ello; `Puntaje`. Al crearlo y almacenarlo en la variable puntos para su uso posterior, aprovechamos a colocarlo en la posición adecuada, cerca de la esquina superior derecha (recuerda que el centro de la ventana tiene coordenadas `x=0` e

`y=0`). El color blanco que va a tener lo indicamos usando, nuevamente, uno de los predefinidos en otro módulo: `pilas.colores`.

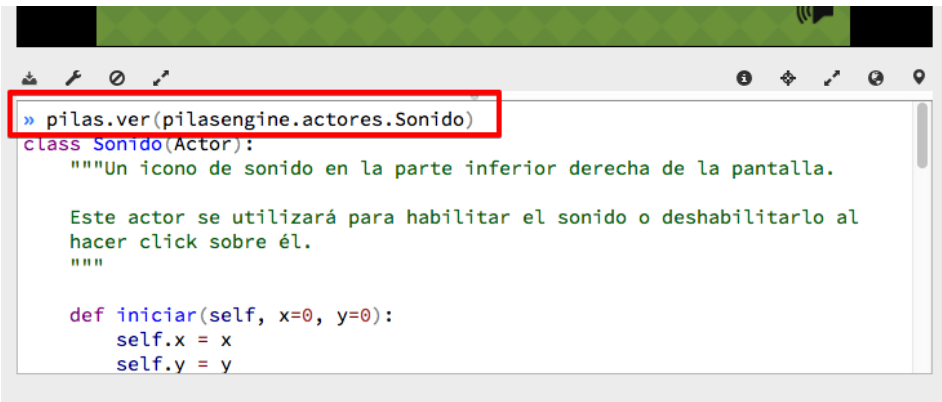
En realidad, la clase de actores `Puntaje` es una clase derivada de otra de Pilas, `Texto`, con lo que hereda sus características, atributos y métodos. Entre ellos, la propiedad `magnitud` indica el tamaño que tendrá el texto (del marcador, en este caso) y como queremos que se vea en grande, lo ponemos a un valor de `40`.

Por último, pasamos al sonido. Como puede verse en la captura de pantalla del juego, queremos poner un botón que permita activar/desactivar el sonido haciendo click sobre él. Nuevamente, Pilas contiene esta funcionalidad de serie...: El actor `Sonido`.

En el intérprete de Pilas puedes probar y escribir:

```
pilas.ver(pilasengine.actores.Sonido)
```

para que veas su código y como está implementado. Allí comprobarás como, efectivamente, el actor se crea directamente en la esquina inferior derecha. Además también incluye la modificación del icono para mostrar el estado del botón y un aviso de texto cuando éste se pulse. El sistema de avisos de texto de Pilas es muy elegante y poco intrusivo; aparece durante un breve periodo de tiempo en la parte inferior con un fondo semitransparente.



¡Ya estamos! Guarda el archivo con el nombre de **paso1.py** y arrástralo sobre la ventana para verlo en ejecución. Pulsa el botón pulsando su icono. ¿No está mal para tan pocas lineas, verdad?

paso2.py

Vamos a añadir ahora la torreta que va a manejar el jugador. Éste es el código:

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

import pilasengine

pilas = pilasengine.iniciar()

# Variables y Constantes
balas_simples = pilasengine.actores.Bala
monos = []


# Funciones
def mono_destruido():
    pass

# Usar un fondo estándar
pilas.fondos.Pasto()

# Añadir un marcador
puntos = pilas.actores.Puntaje(x=230, y=200, color=pilas.col
puntos.magnitud = 40

# Añadir el conmutador de Sonido
pilas.actores.Sonido()

# Añadir la torreta del jugador
torreta = pilas.actores.Torreta(municion_bala_simple=balas_s
                                enemigos=monos,
                                cuando_elimina_enemigo=mono_

# Arrancar el juego
pilas.ejecutar()
```

Estamos usando el actor de Pilas `Torreta` que vamos a almacenar para un futuro uso en la variable torreta. En su creación, entre los argumentos que podemos pasarle, hemos usado los tres fundamentales:

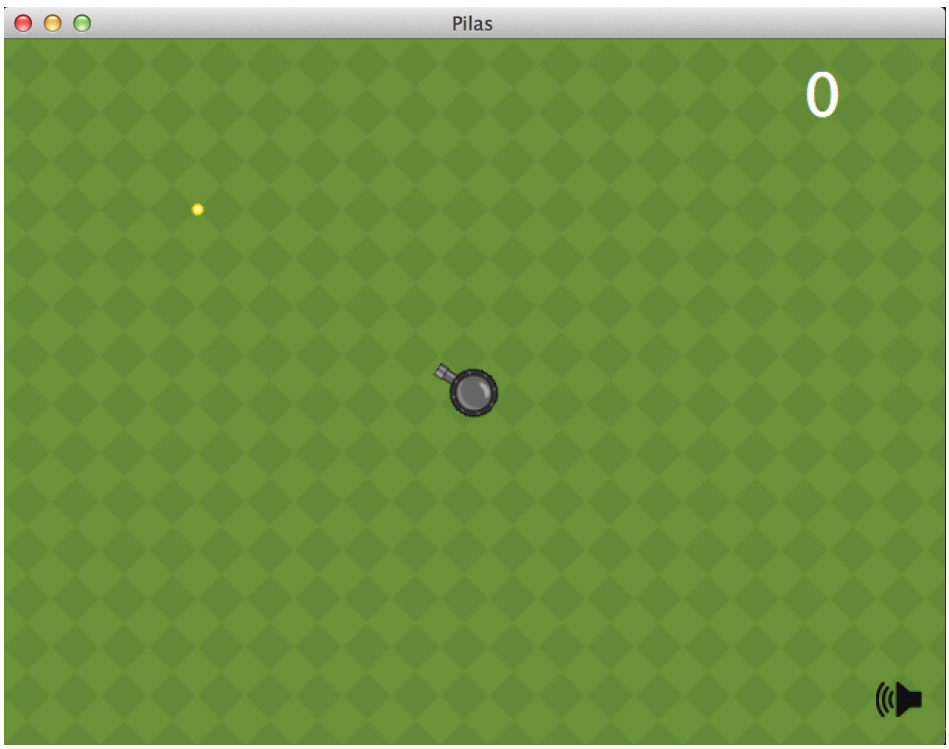
- En primer lugar hemos de indicarle cual es la munición que va a emplear con el argumento `municion_bala_simple` . Para ello, al principio del programa, hemos definido la variable balas_simples con el tipo de munición deseado, en este caso la clase correspondiente al actor `Bala` (que, por cierto, es la que usaría el propio Pilas por defecto):

```
balas_simples = pilasengine.actores.Bala
```

Observa que escribimos `pilasengine.actores.Bala` y no `pilasengine.actores.Bala()` o `pilas.actores.Bala()` ya que estamos indicando la clase de actor que vamos a usar y no lo estamos creando.

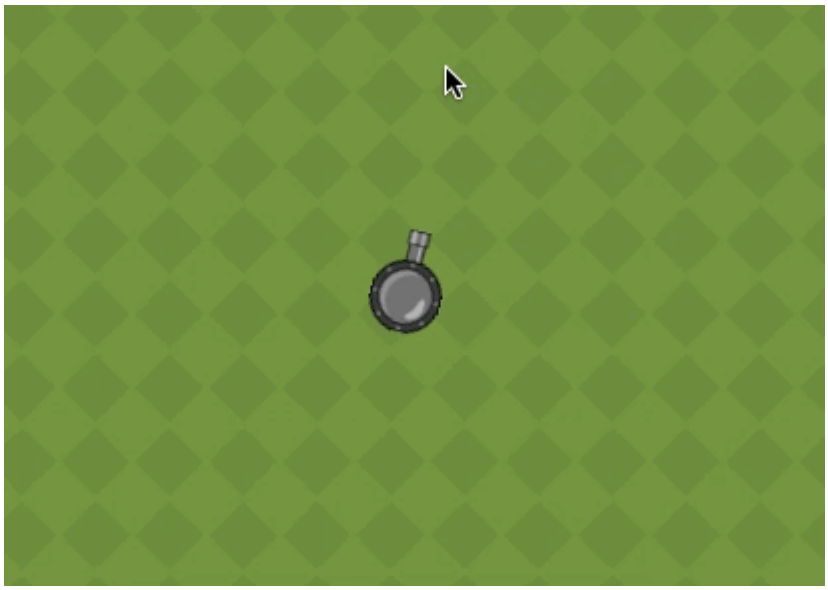
- Lo segundo que le pasamos, el argumento `enemigos`, es el grupo de enemigos a los que se va a disparar. Como en este punto del desarrollo no lo hemos decidido, hemos definido previamente la variable `monos` como una lista vacía.
- El tercer y último parámetro que le proporcionamos al constructor de la torreta, `cuando_elimina_enemigo`, ha de ser una función que se llamará cuando la munición que disparamos impacte con los enemigos. De momento no queremos concretar más detalles, así que hemos definido al principio del código una función `mono_destruido()` que no hace nada, usando `pass`:

```
def mono_destruido():
    pass
```



Observa, de nuevo, que en la creación de la torreta escribimos `mono_destruido` y no `mono_destruido()` ya que queremos pasarle la función que ha de usarse y no el resultado de ejecutarla.

Hecho. Guarda el código con el nombre **paso2.py** y ejecútalo. ¡La torreta responde!



CONTINUARÁ ...