



Taller de Introducción a Pygame

2013

Video Juegos con Python

PyGroupUD

Proyecto Curricular de Ingeniería de Sistemas

Taller de Introducción a Pygame

Índice de contenido

Python, ¿Qué es?.....	3
Pygame, ¿Qué es?.....	3
Preparación del entorno.....	4
Primer ejercicio	5
Generando la ventana.....	5
Agregando imágenes.....	7
Agregando movimiento.....	10
Trabajo a realizar.....	13
Bibliografía consultada:.....	13



Python, ¿Qué es?

Python es un lenguaje de programación de muy alto nivel adecuado para implementar casi cualquier proyecto, tiene como características que es un lenguaje interpretado (no requiere que sus scripts se compilen previamente a la ejecución del programa), es un lenguaje de tipado dinámico (no requiere declaración explícita de tipos), es multiplataforma (funciona en bien en plataformas GNU/Linux, Windows, Mac) y es un lenguaje que tiene curva de aprendizaje muy rápida.


Python al ser un lenguaje de alto nivel permite una programación sin confusiones y el uso de un código más limpio al no requerir una gran cantidad de instrucciones para obtener funcionalidades sencillas.

Python soporta por lo menos parcialmente algo del paradigma de programación funcional y totalmente el paradigma orientado a objetos y la programación estructurada, esto le brinda al programador mayor cantidad de abstracciones base que facilitan el ejercicio de obtener soluciones mas rápidamente.

Pygame, ¿Qué es?

Pygame es una biblioteca multimedia multiplataforma que trabaja sobre las librerías SDL (Simple Direct Media Layer) que permite la creación de aplicaciones multimedia como videojuegos 2D de forma sencilla.

Pygame nos permite gestionar imágenes en formatos como PNG, BMP, PCX, TGA y otros; nos permite manejar sonido en formatos como MOD, OGG y MP3; nos



permite gestionar operaciones relacionadas con el gestor de ventanas; manejar eventos de aplicación y dispositivos de entrada; trabajar con temporizadores, sprites y sistema de colisiones y aunque es una biblioteca simple posee todo lo que se necesita para la creación de juegos 2D con python.

Preparación del entorno

Para poder trabajar con python y pygame debemos instalar ambos en nuestros equipos de así como un buen ide que nos facilite el trabajo, para el desarrollo de todos los ejemplos y evaluación de los ejercicios se usarán las siguientes versiones:

- Python 2.7.4
- Pygame 1.9.1

Como IDE se trabajará con Geany que ofrece un soporte básico a python, pero cualquier IDE puede ser utilizado (IDLE, Aptana, Ninja-IDE, etc).

Si trabajamos con Ubuntu – Linux la instalación de estas herramientas es bastante sencilla y basta con ejecutar la siguiente linea y se tendrá listo el entorno de trabajo.

```
sudo apt-get install python python-pygame geany
```

Si trabajamos sobre plataformas windows debemos descargar las herramientas y ejecutar sus instaladores para tener listo el entorno de trabajo necesario

Python: <http://www.python.org/download/>

Pygame: <http://www.pygame.org/download.shtml>

Geany: <http://www.geany.org/Download/Releases>

Primer ejercicio

Generando la ventana

Vamos a poner manos a la obra, para nuestro primer ejemplo trabajaremos una aplicación que genere el rebote de un balón en una pantalla de 400 x 640 con un fondo gráfico. Para este ejercicio necesitaremos dos imágenes en formato png una que sea de una bola del deporte de nuestra preferencia y background que nos parezca acorde a la bola que elijamos, se sugieren las dos siguientes imágenes:

- bola: <https://www.dropbox.com/s/uokpwbyihpndzul/basketball.png>
- fondo: <https://www.dropbox.com/s/jcj2czfkfx2g80f/cancha.png>

Lo primero que haremos será generar una ventana de dimensiones 400 x 640 haciendo uso de pygame para esto:

```
1. import sys, pygame
2.
3. size = width, height = 640, 400
4. screen = pygame.display.set_mode(size)
5.
6. def main():
7.     pygame.init()
8.     while 1:
9.         for event in pygame.event.get():
10.             if event.type == pygame.QUIT:
11.                 sys.exit()
12.
```

```
13.     if __name__ == '__main__':
14.         main()
```

Veamos el detalle de nuestro primer script:

En la primera línea:

```
import sys, pygame
```

Traemos a nuestro script los modulos disponibles en sys y pygame, del módulo sys usaremos la función `exit()` para cerrar el programa y del módulo pygame usaremos `display` para generar la ventana y `event` para manejar los eventos que ocurran sobre la misma.

En las líneas 3 y 4:

```
size = width, height = 640, 400
screen = pygame.display.set_mode(size)
```

Definimos una tupla de valores almacenada en la variable `size` que tiene los valores de 640 y 400 y creamos una ventana haciendo uso de `pygame.display.set_mode(size)` la cual podremos trabajar mediante el identificador `screen`.

Entre las líneas 8 y 13 definimos una función principal `main()` que se encarga de inicializar pygame mediante `pygame.init()` y maneja el ciclo de ejecución de nuestro juego, este ciclo hace una captura de los eventos que ocurran en nuestra ventana principal en el ciclo `for` de la línea 11 en el que captura todos los eventos que ocurran mediante `pygame.event.get()` y verifica si alguno de estos `event` es igual al evento `pygame.QUIT` y ejecuta la función `sys.exit()`.

```
def main():
    pygame.init()
    while 1:
```

```
for event in pygame.event.get():  
    if event.type == pygame.QUIT:  
        sys.exit()
```

Por último, la línea 20 verifica que si la ejecución que hace python del script obedece a la ejecución del script que contiene esta definición ejecute la función *main()*, en otras palabras verifica que si este escript es el que está ejecutando y si no ha sido importado por otro script, esto porque cuando desde python un script es ejecutado directamente este corre en el spacename *__main__* pero cuando es importado este es ejecutado en su propio namespace.

Al ejecutar el script obtenemos el siguiente resultado:

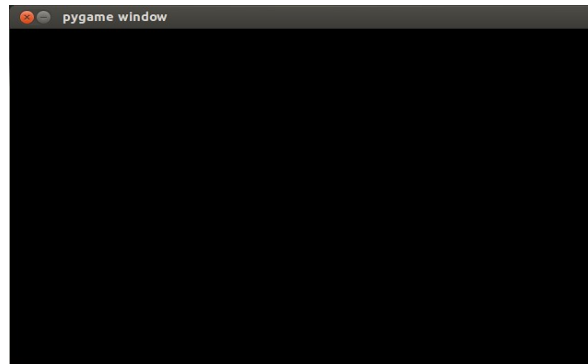


Ilustración 1: ventana simple

Agregando imágenes

Ahora pongamos las imágenes en el juego, para esto modificaremos nuestro script de manera que cargue las imágenes y luego actualice nuestra ventana con las misma. Usaremos la imagen de la cancha como fondo de la aplicación y la bola como elemento que se moverá en la misma.

```
1. import sys, pygame
```

```

2.
3.     size = width, height = 640, 400
4.
5.     screen = pygame.display.set_mode(size)
6.
7.     def main():
8.         pygame.init()
9.         ball = pygame.image.load("imagenes/basketball.png")
10.        ballrect = ball.get_rect()
11.        ballrect.left = (width/2)-(ballrect.width/2)
12.        ballrect.top = (height/2)-(ballrect.height/2)
13.
14.        court = pygame.image.load("imagenes/cancha.png")
15.        courtrect = court.get_rect()
16.        while 1:
17.            for event in pygame.event.get():
18.                if event.type == pygame.QUIT:
19.                    sys.exit()
20.
21.            screen.blit(court, courtrect)
22.            screen.blit(ball, ballrect)
23.
24.            pygame.display.update()
25.
26.     if __name__ == '__main__':
27.         main()

```

En esta versión del script se agregaron de la línea 9 a la 15 la carga de las dos imágenes y la inicialización de las áreas rectangulares para manejarlas, en este punto el área rectangular que mas nos interesa es el de la bola que será la que se mueva luego porque la del fondo se mantendrá estática.

```

ball = pygame.image.load("imagenes/basketball.png")
ballrect = ball.get_rect()
ballrect.left = (width/2)-(ballrect.width/2)

```



```
ballrect.top = (height/2)-(ballrect.height/2)

court = pygame.image.load("imagenes/cancha.png")
courtrect = court.get_rect()
```

La variable *ballrect* en las líneas 11 y 12 se modifica para obtener el punto central donde cargar la bola en la ventana.

En las líneas 21 y 22 se pintan en el *screen* mediante el método *blit()* las imágenes de la cancha y de la bola, el primer parámetro del método es la imagen y el segundo es el área rectangular desde donde se pintan.

```
screen.blit(court, courtrect)
screen.blit(ball, ballpos)
```

Y por último en la línea 24 se actualiza la ventana para que repinte lo indicado en las líneas 21 y 22.

```
pygame.display.update()
```

El resultado obtenido al ejecutar el script es el siguiente:



Ilustración 2: carga de imágenes

Agregando movimiento

Ahora vamos a agregar movimiento a nuestra bola, de tal manera que parezca que rebota en la cancha, para esto vamos a hacer que se mueva el área rectangular de la bola y controlaremos que no suba mas de $\frac{3}{4}$ del área de la ventana y no sobrepase el límite inferior de la misma.

```
1.  import sys, pygame
2.
3.  size = width, height = 640, 400
4.  speed = [0, 2]
5.  screen = pygame.display.set_mode(size)
6.
7.  def main():
8.      pygame.init()
9.      ball = pygame.image.load("imagenes/basketball.png")
10.     ballrect = ball.get_rect()
11.     ballrect.left = (width/2)-(ballrect.width/2)
12.     ballrect.top = (height/2)-(ballrect.height/2)
13.
14.     court = pygame.image.load("imagenes/cancha.png")
15.     courtrect = court.get_rect()
16.     while 1:
17.         for event in pygame.event.get():
18.             if event.type == pygame.QUIT:
19.                 sys.exit()
20.
21.         ballrect = ballrect.move(speed)
22.
23.         if ballrect.top < height/3 or ballrect.bottom > height:
24.             speed[1] = -speed[1]
25.
26.         screen.blit(court, courtrect)
27.         screen.blit(ball, ballrect)
```

```
28.  
29.         pygame.display.update()  
30.  
31.     if __name__ == '__main__':  
32.         main()
```

Para lograr esto creamos una lista de dos elementos que representarán la velocidad de la bola en x y en y esta variable se creó en la línea 4 del script.

```
speed = [0, 2]
```

El componente con índice 0 de la lista representa la velocidad en x en la cual la bola no tendrá movimiento y el valor en el índice 1 de la lista representa la velocidad en y para la cual la bola sí tiene movimiento.

En la línea 21 se modifica el área rectangular asociada a la imagen de la bola mediante el método *move()* asociada a la misma.

```
ballrect = ballrect.move(speed)
```

Y entre las líneas 23 y 24 se controla que la bola no sobrepase los límites definidos para su movimiento que son $\frac{3}{4}$ de la parte baja de la pantalla y el tope inferior de la misma, verificando estos valores del área rectangular asociada con los parámetros *top* y *left* y si se alcanzan o superan se invierte la velocidad de la bola en el componente y.

```
if ballrect.top < height/3 or ballrect.bottom > height:  
    speed[1] = -speed[1]
```

El resultado obtenido al ejecutar el script es el siguiente:



Ilustración 3: movimiento 1

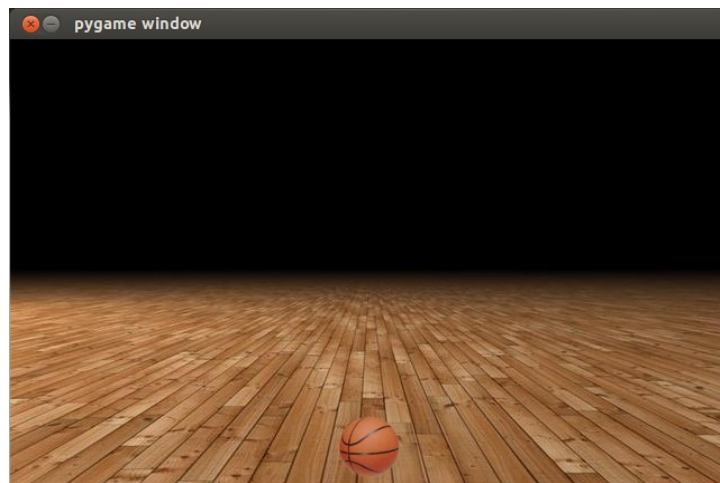


Ilustración 4: movimiento 2

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. Modifica el script final para que la bola rebote desde el tope de la ventana hasta el límite inferior de la misma. Guarda el script como taller01_01.py
2. Agrega movimiento en el eje x a la bola de tal manera que la bola se desplace por toda la ventana con movimientos diagonales, la bola no debe superar los bordes de la ventana. Guarda el script como taller01_02.py.

Bibliografía consultada:

“Pygame documentación”.

<<http://www.pygame.org/docs/index.html>>

(01 Sep 2013)

“Python documentation”.

<<http://www.python.org/doc/>>

(01 Sep 2013)