Mini Tutorial de Monkiolivs

Beca Matías, Flores Federico, Moscheni Leandro

7 de Julio, 2013

Extracto

Un pequeño tutorial que explica como crear un juego (por sus autores arriba mencionados) en un framework de Python.

1 Introducción

Monkiolivs es un juego simple y entretenido basado en un framework de Python, que busca probar y mejorar la habilidad de los usuarios con el mouse.

2 Conociendo un poco el juego

2.1 Ambiente

El ambiente del juego consiste en un fondo verde dibujado con rombos (el tablero del juego), algunas bananas (dispuestas a lo largo y ancho del tablero, de forma aleatoria), un contador (que representa nuestro puntaje) y una barra situada en el borde inferior que se encargará de informarnos diferentes eventos (Ej.: Game Over, si nuestro personaje principal muere).

2.2 Personaje Principal

Nuestro protagonista es un agraciado mono, que posee la capacidad de moverse a través de todo el plano y comer bananas, para asi sumar puntos y ganar el juego; mientras evita entrar en contacto con las aceitunas que al comerlas, lo hará explotar, y así perder el juego.

Figure 1: Un mono amigable

Figure 2: Una aceituna no tan inocente

2.3 Enemigos

Los enemigos de nuestro mono son unas inocentes aceitunas a simple vista, pero que sin embargo resultan sumamente explosivas en cuanto el mono entra en contacto con ellas.

3 Objetivos

Los objetivos de este juego son básicamente dos:

- Que el mono coma todas las bananas, y de esta manera lograr el puntaje máximo
- Que el mono sobreviva a las aceitunas, para ello evitando el contacto con las mismas

4 ¿Cómo Jugar?

La forma de jugar a **Monkiolivs** es bastante sencilla.
Cuando el juego se ejecute, el jugador verá distribuidas (de forma aleatoria) las bananas y las aceitunas en el tablero. Con el mouse deberá controlar el movimiento del mono, que podrá ser hacia arriba, abajo, izquierda o derecha. Incluso puede moverlo en forma diagonal. Mientras las bananas permanecen

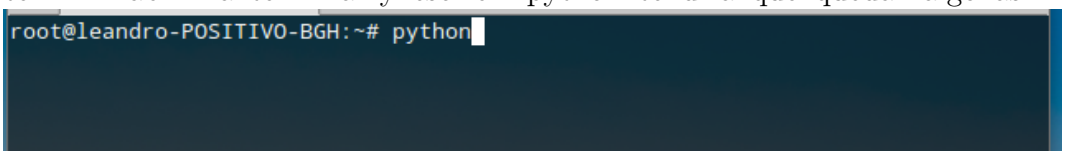
estáticas en el tablero, las aceitunas lo recorren de un lado a otro del mismo. Cada vez que el mono come una banana, el contador de nuestro puntaje se incrementa. Como ya se ha dicho antes, el mono debe evitar a toda costa el contacto con las aceitunas.

5 Tutorial

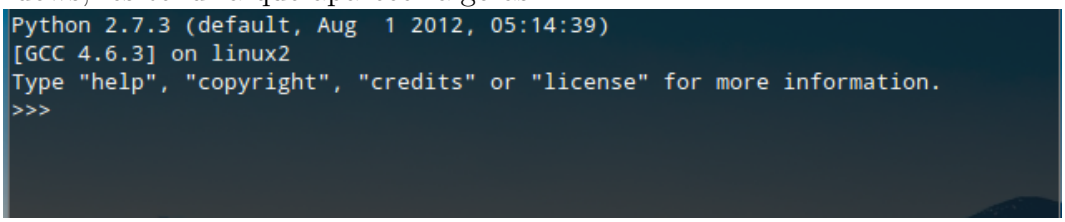
Pilas posee una gran variedad de actores y acciones predefinidas, pero además, tambien aporta sencillez a la hora de agregar nuevas cosas. En este turorial solo nos limitaremos a utilizar funciones ya predefinidas ya que es todo lo que necesitamos para desarrollar dicho juego.

5.1 Instalación

Lo primero que se debe hacer es instalar todo lo necesario para poder empezar a a jugar y divertirse con el framework. Para empezar hay que tener instalado python en una versión 2.6 o superior. Si estamos utilizando alguna distribución de Linux o MAC OSX seguramente ya venga instalado, con lo que alcanza con abrir la terminal y escribir "python" y dar enter. Al abrir la terminal y escribir python tendria que quedar algo asi:



En caso de tener Windows, nos tenemos que descargar el instalador, e instalarlo, como si de cualquier otro programa se tratase. Aqui les dejo el link: **http://python.org/download/** de donde pueden descargar el instalador, luego tienen que ir todos los programas, y buscar el directorio en donde tienen instalado python, y darle click al ejecutable.
Al ejecutar python, ya sea desde la terminal, o desde el acceso directo de Windows, les tendria que aparecer algo así:



Luego de haber instalado python, procederemos a la instalación de Pilas (el framework que usaremos para hacer los juegos). Para eso debemos entrar en el sitio web de pilas: **http://pilas-engine.com.ar/**, y descargaremos el instalador para nuestro sistema operativo. En caso de no contar con el instalador para nuestro Sistema operativo, podremos instalarlo desde el repositorio donde se encuentra el proyecto, los pasos necesarios para hacerlo están en la documentación oficial de pilas. Aqui les haremos una breve explicación de que deben hacer para instalarlo con Pypi una herramienta de python:

```
Primero escribimos esto en la terminal:

sudo apt-get install python-setuptools python-qt4 python-qt4-gl git-core
sudo apt-get install python-setuptools python-qt4-phonon build-essential
sudo apt-get install python-setuptools python-dev swig subversion

Luego ponemos los siguientes dos comandos:

sudo easy_install -U box2d
sudo easy_install -U pilas

Y listo!! ya estamos listos para usar pilas
```

5.2 Empezando a escribir codigo

Se debe crear un archivo con extensión ".py" para que pueda ser reconocido por la IDE de python que deseemos usar.

5.3 Actores

En pilas todo es un actor, ya sea el entorno o los personajes. Nuestros actores son un mono,varias aceitunas y bananas.

- El Mono: será un personaje manipulado por nosotros. Para crearlo lo haremos de la siguiente forma.

  El mono debe mover de acuerdo a los clicks que reciba tambien el mono debe explotar cuando entra en contacto con una aceituna (lo cual veremos mas adelante), por ahora solo le enseñamos a explotar y a seguir a los clicks.Podemos enseñarle a hacer esto con el siguiente codigo.

```
mono = pilas.actores.Mono()

# Con esto le enseñamos al mono a seguir a los clicks
mono.aprender(pilas.habilidades.SeguirClicks)

# Con esto le enseñamos al mono a explotar cuando se le mande el mensaje
# eliminar()
# Osea que cuando hagamos mono.elminar() BOOM! el mono explotara
mono.aprender(pilas.habilidades.PuedeExplotar)
```

- Las Bananas: las creamos de la misma manera que el mono, pero se necesitan varias por lo que haremos una lista de bananas y haremos que se ubiquen en posiciones aleatorias. Lo Podemos hacer de la siguiente manera:

```
# Ahora definiremos una lista de bananas, mas especificamente, definiremos una
# lista con 5 bananas en posiciones aleatorias.
bananas = pilas.atajos.fabricar(pilas.actores.Banana,5)
```

Una vez defindo esto nos queda "enseñarles a moverse", lo cual se logra escribiendo el siguiente codigo:

```
# Ahora definiremos una funcion, que dada una lista de bananas, las mueva de un
# punto a otro de la pantalla, a una velocidad determinada.
def moverBananas(b):
    # con esto le decimos a las bananas que se muevan en el eje x desde el punto
    # -250 al 250, con una velocidad de 5
    b.x [-250,250] , 5
```

- Las Aceitunas: los personajes malvados de nuestro juego los instanciaremos de la misma manera que las bananas y les daremos el mismo movimiento que a las bananas.

```
# Ahora definiremos una lista de aceitunas, mas especificamente, definiremos una
# lista con 5 bananas en posiciones aleatorias.
aceitunas = pilas.atajos.fabricar(pilas.actores.Aceituna,5)

# Ahora definiremos una funcion, que dada una lista de aceitunas, las mueva de
# un punto a otro de la pantalla, a una velocidad determinada.
def moverAceitunas(a):
    # con esto le decimos a las aceitunas que se muevan en el eje x desde el
    # punto -250 al 250, con una velocidad de 4
    a.x [-250,250] , 4
```

6 Colisiones

Las colisiones nos sirven para saber que deben hacer los actores cuando entran en contacto con otros elementos del mundo. En Pilas todos los actores ya tienen un radio de colisión predefinido,pero si se quiere modificar se hace de la siguiente manera:

6.1 Colisionando actores

Ahora ya tenemos a los actores en la escena. Ahora debemos decirles que sucede cuando interactuan entre ellos. Ya que nuestro personaje es el mono solo debemos decirle lo que el debe hacer cuando entre en contacto con otros elementos del mundo

- Contacto con banana: El mono al comer una banana se alegra y aumenta su puntaje. Entonces en el juego cuando colisionen el mono con la banana,la banana debe desaparecer, el puntaje aumentar y el mono sonreir. Lo hacemos con el siguiente codigo:

```
#·Creamos·la·pizarra·del·puntaje
puntaje·=·pilas.actores.Puntaje()

#·Aumentamos·el·tamaño·del·puntaje·para·mejor·visibilidad
puntaje.escala·=·1.5·

#·Definimos·al·blanco·como·color·del·puntaje
puntaje.color·=·pilas.colores.Blanco·

#·Posicionamos·el·indicador·de·puntaje·en·el·extremo·superior·derecho·de·la
#·pantalla
puntaje.x·=·295
puntaje.y·=·180

#·Definimos·la·funcion·que·hara·comer·una·banana·a·un·mono,·y·asi·aumentar·su
#·puntaje·en·10.
def·comerBanana(m,b):
····b.eliminar()
····m.sonreir()
····puntaje.aumentar(10)

#·Con·esto·definimos·una·nueva·colision,·que·basicamente·dice·lo·siguiente:
#·si·un·mono·toca·a·una·banana,·efectua·la·funcion·comerBanana(),·que·lo·que
#·que·hace·es·eliminar·a·la·banana,·hace·sonreir·al·mono,·e·incrementa·el
#·puntaje·en·10
pilas.escena_actual().colisiones.agregar(mono,bananas,comerBanana())
```

- Contacto con aceitunas: El mono al entrar en contacto con las aceitunas deberia morir. En este juego somos un poco extremistas y hacemos que el mono explote, en consecuencia, el juego termine. Lo hacemos de la siguiente manera:

```
64 #·Definimos·la·funcion·que·hara·comer·una·aceituna·a·un·mono,·y·asi·producir
65 #·su·explosion,·y·fin·del·juego.
66 def·comerAceituna(m,a):
67 ····b.sonreir()
68 ····m.eliminar()
69 ····pilas.avisar("Game·Over")
70 
71 #·Con·esto·definimos·una·nueva·colision,·que·basicamente·dice·lo·siguiente:
72 #·si·un·mono·toca·a·una·aceituna,·efectua·la·funcion·comerAceitna(),·que·lo·que
73 #·que·hace·es·reventar·al·mono,·y·terminar·el·juego·con·la·partida·perdida.
74 pilas.escena_actual().colisiones.agregar(mono,aceitunas,comerAceituna())
```

Muy bien ya tenemos los métodos que se utilizarán en el juego, la forma en la que interactúan los actores al entrar en contacto(las colisiones), ahora solo nos queda darle movimiento automatizado a los personajes que no controlamos, para eso fijaremos las tareas que queremos que se ejecuten siempre, durante la ejecución del juego, que son básicamente, que las aceitunas y las bananas se muevan solas...
Lo Hacemos de la siguiente manera:

```
79 #·Aca·le·decimos·a·pilas,·que·cada·3·seg,·mueva·a·todas·las·bananas
80 pilas.mundo.agregar_tarea_siempre(3,moverBanana(),bananas)
81 
82 #·Aca·le·decimos·a·pilas,·que·cada·3·seg,·mueva·a·todas·las·aceitunas
83 pilas.mundo.agregar_tarea_siempre(3,moverAceituna(),aceitunas)
```

Note que en Pilas se debe agregar el metodo Pilas.Iniciar() la principio y pilas.Ejecutar() al final para que todo se pueda ejecutar.

7 Conclusión

Este juego es divertido, apto para toda la familia y, por sobre toda las cosas, no genera ningún instinto de violencia en el usuario.
Asimismo, invita a todos aquellos interesados en la programación (sobre todo la orientada a juegos), a conocer e interactuar con un framework open-source de fácil uso.