ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

PROGRAMACIÓN AVANZADA

ASIGNATURA:	Programación Avanzada
PROFESOR:	Ing. Juan Pablo Zaldumbide
PERÍODO ACADÉMICO:	Feb. 2016 - Ago. 2016

Informe Final

TÍTULO:

Juego arcade

“SPACEFIGHTER “

ESTUDIANTES:

Mario Núñez
Oscar Pullay

FECHA DE ENTREGA: 11 de Agosto de 2016

CALIFICACIÓN OBTENIDA:

FIRMA DEL PROFESOR:

INDICE DE CONTENIDOS

1 OBJETIVO

1.1 Objetivo General:

- Ser una guía o tutorial de referencia para el desarrollo de juegos arcade con Python, utilizando la libreria pygame

1.2 Objetivos específicos:

- Documentar al lector sobre el lenguaje de programación Python.
- Documentar al lector sobre la librería multimedia pygame.
- Demostrar la sencillez del lenguaje de programación Python.
- Presentar el juego arcade como ejemplo practico

2 MARCO TEORICO

Antecedentes históricos

2.1 Python:

Python fue creado a principios de los 90 por Guido van Rossum en el Stichting Mathematisch Centrum8 en los Países Bajos como sucesor de un lenguaje llamado ABC y la necesidad del autor por un analizador de sintaxis sencillo y facil de aprender. Guido sigue siendo el principal autor de Python, aunque incluye multitud de contribuciones de otros.

Después de crear el interprete se comenzaron a agregar partes de código similares a las de un compilador o máquina virtual lo que permitió en un momento crear un interprete y lograr programas en ejecución bajo Python.

Python, a pesar de iniciar como el proyecto de una sola persona, ha llevado un diseño y lineamientos que lo han llevado a lo que hoy es. Entre los planes originales de diseño se encontraban:

- Similitud con la Shell de UNIX, lo que permite un ambiente amigable, muy similar a la terminal de UNIX, pero no un remplazo de la misma.

- Arquitectura Extensible. Es decir, una arquitectura en la que se puedan agregar cosas nuevas, no todo es perfecto en un principio y Python deberia ser capaz de cambiar de acuerdo a las necesidades de los usuarios. Guido

van Rossum describe como doloroso su trabajo con otros lenguajes como ABC al tratar de implementar nuevos tipos de datos.

- Una herramienta en lugar de muchas. Python deberia encajar bien en cualquier ambiente, ser de uso multiple, y por lo tanto ser un buen remplazo para muchas herramientas (y otros lenguajes de programación). Python sea altamente portable.

2.2 Pygame:

PyGame es un motor de juegos, conformado por un conjunto de librerías cuya finalidad es facilitar la tarea del programador a la hora de realizar un videojuego, pudiendo ser usado también para la realización de aplicaciones multimedia e interfaces gráficas.

El desarrollo de PyGame comenzó en el 2000 de la mano de Pete Shinners que acababa de conocer Python y SDL (Simple Directmedia Layer), así como un pequeño proyecto denominado PySDL que pretendía fusionar ambos, siendo la desparición de PySDL el motivo de asumir el proyecto de realizar un motor de juego que basado en Python y SDL.

PyGame permite la creación y desarrollo de videojuegos de una forma clara y sencilla, sin que esto nos impida obtener los resultados esperados puesto que puede combinarse con otras librerías, como PyOpenGl, para incluir graficos 3D.

Otra de las características que comparte con Python es la existencia de una completa documentación para facilitar la comprensión y utilización, así como una gran variedad de ejemplos.

Características de la librería

- Lenguaje de muy alto nivel lo que implica un código claro.
  Multiplataforma.
- Basado en SDL.
- Multiples usos, juegos interfaces graficas y multimedia.

Requisitos

Cualquier ordenador personal que se comercializa hoy en día cumple con los requisitos de hardware para poder instalar las librerías de desarrollo y Python es compatible con sistemas Unix y Windows por lo que para poder empezar a programar aparte de las librerias es unicamente necesario, que no imprescindible, un IDE de desarrollo.

En cuanto a IDEs hay una gran variedad donde elegir, empezado por los de pago:

Microsoft Visual Studio para el que existe una versión de Python llamada IronPython que se integra con este IDE permitiendo crear proyectos de python de forma nativa.

Descarga de Python: http://www.python.org/download/

Descarga de Pygame: http://www.pygame.org/download.shtml

3 HERRAMIENTAS UTILIZADAS:

3.1 Python:

Como lenguaje de programación, cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

3.2 Pygame:

Introducción general a la librería.

Como ya se comentó, Python es un lenguaje que admite múltiples paradigmas de programación: Estructurada, funcional, orientada a objetos y orientada a aspectos.

Python permite la combinación de todos esos paradigmas dentro de un mismo código, lo que le da una gran versatilidad.

PyGame está dividido en diversos módulos que agrupan múltiples funcionalidades comunes para desarrollar videojuegos o aplicaciones que usen gráficos en 2D. Al estar escrito en Python, tiene todas las ventajas del multiparadigma, por tanto, esta biblioteca puede ser usada utilizando cualquier estilo.

Internamente PyGame contiene funciones (de programación estructurada) y clases. El desarrollo habitual consiste en usar un paradigma orientado a objetos, sin embargo, no es extraño ver videojuegos desarrollados en programación estructurada usando esta biblioteca.

La estructura común suele ser, tras la carga e inicialización de componentes, un bucle infinito que transcurre hasta que la aplicación finaliza, donde en cada iteración se produce la evaluación de eventos, actualización de pantalla y actualización de componentes que estén en funcionamiento. Muy similar a cómo funciona OpenGL (de hecho, es habitual ver a ambas librerías trabajar juntas. El propio PyGame trae una opción para renderizar sobre una superficie OpenGL para potenciar el resultado).

Aunque el código principal de la aplicación suele estar en un módulo independiente y desarrollado siguiendo el paradigma de la programación estucturada, lo normal es desarrollar usando la programación orientada a objetos, dejando el código principal como una simple llamada a un método de una clase que se encargue de lanzar todos los sucesos del juego y controlarlos.

Finalmente, se puede ver en algunos casos el paradigma de programación funcional para resolver ciertos aspectos de la lógica del juego, gracias a la potencia que este paradigma aporta a la resolución de ciertos problemas. Normalmente el uso de la programación funcional suele ser ajeno a PyGame y no suele afectar de forma directa al uso de las librerías.

Como característica de Python, y como se ha comentado, es posible tener módulos desarrolados usando la programación estructurada y módulos de clases conviviendo entre si. Y en cada uno de ellos se puede usar el soporte que Python otorga a la programación funcional según sea conveniente.

4 MANUAL DE CREACION DEL JUEGO:

4.1 Spacefighter:

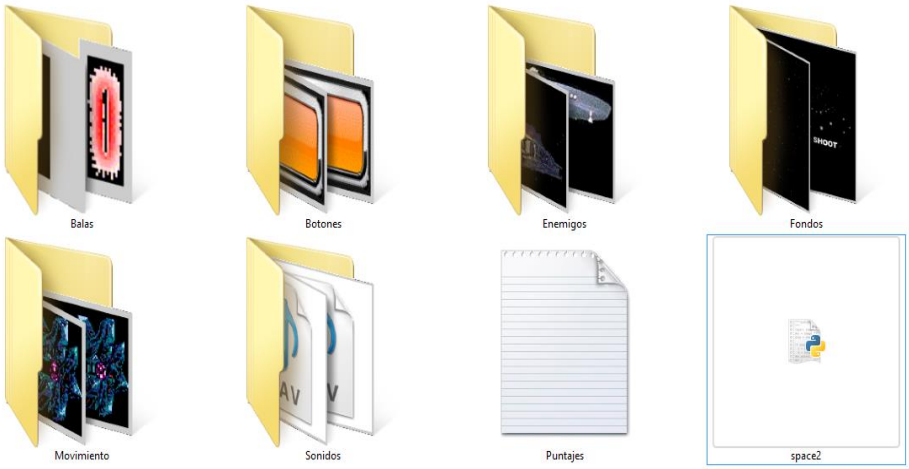
Vamos a realizar el juego paso a paso, un proyecto basado en python y pygame, se trata de **spacefighter.py**.

4.2 Introducción al juego:

El juego está inspirado en el clásico Space Invaders de las máquinas recreativas. Se trata de una nave espacial, dirigida con las flechas del cursor, que se enfrenta a una lluvia de enemigos que hay que destruir. El detalle importante es que al disparar a un enemigo, éste se destruye y aparecen otros que son más difíciles de destruir. Por otra parte, se trata de un mundo esférico, y si nosotros o las naves nos salimos por un extremo de la ventana, aparecemos por el extremo contrario.

Hay varios detalles más; un marcador, sonido, un contador de vidas y una pantalla de presentación de juego y otra de ayuda. Además, contaremos con varios niveles de dificultad, a medida que destruimos todas las naves de los enemigos. Todo ello nos permitirá aprender nuevas técnicas, empezando por la modularización. Y la organización en torno a escenas.

La modularización intenta identificar partes reutilizables de un programa y desacoplarlas en archivos diferentes. Por ejemplo, si el comportamiento de los enemigos lo implementamos por separado, más adelante podemos cambiar la forma en la que se crean, se desplazan o se destruyen sin interferir en nada más del videojuego. Además, permite que el código sea manejable, al estar divido en piezas más pequeñas y no un solo archivo de una longitud inabarcable. Por otra parte, el concepto de escena, es una manera más apropiada de dividir un videojuego en secciones diferentes sin que interfieran unas con otras y guardando el estado en el que se encuentran. En el caso que nos ocupa, tenemos tres escenas naturales; la de la pantalla de inicio, la de las instrucciones y la del propio juego. Éste es el esquema de archivos que vamos a construir y que vamos a ver paso a paso:



Organizarse el algo muy importante! Cuanto más claro dejemos el acceso a nuestros jugadores, más fácil será para ellos identificar qué deben hacer para jugar con nuestro videojuego. He visto demasiadas veces cómo, al abrir la carpeta de un juego, aparecen montonadas de archivos sin que quede claro cuál es el ejecutable o qué hay que hacer. Así que nosotros vamos a crear una carpeta en la que aparecerán tres cosas:

- Un archivo .py que será sobre el que hay que hacer doble click para ejecutar nuestro juego.
- Un archivo .txt (podría ser un pdf o similar) donde se grabaran los puntajes de las partidas jugadas.
- Una carpeta en donde irá ubicado todo el resto del material que necesita el juego, como la música, todas las imágenes y fondos.

4.3 Importación de librerías:

Se cargan las librerías habituales, incluyendo random para introducir aleatoriedad en el juego y os para usar direccionamiento de archivos independiente de la plataforma.

```
#declaracion de librerias
import pygame
from pygame.locals import *
from tkinter import *
from random import randint
from tkinter import *
```

4.4 Definición de constantes:

Como quiera que los tamaños de la ventana de juego se usan en varios lugares, es conveniente almacenarlos en variables. En realidad, al no modificarse, más que variables se habla de constantes; para distinguirlas visualmente cuando recorremos un código largo, un método habitual es usar un nombre en mayúsculas. Hemos definido dos:

- ANCHO: Anchura de la ventana de juego
- ALTO: Altura de la ventana de juego

```
#declaracion de variables locales
ancho = 700
alto = 600
```

4.5 Función cagar imagen:

La tarea repetitiva de cargar una imagen desde un archivo y convertirla al formato con el que trabaja PyGame la hemos empaquetado en esta función. La definición posee dos argumentos, uno obligatorio (el nombre del archivo donde está la imagen) y otro opcional que por defecto se establece como False. Si no utilizamos este segundo argumento cuando llamemos a la función, la imagen generada no tendrá transparencia. Si cuando la llamemos le pasamos True, se usará (fíjate en el código) como color transparente, a la hora de dibujar, el del pixel de la imagen de coordenadas (0,0).

Hay varias cosas que comentar:

- os.path.join() es una función del módulo os que permite usar direcciones de archivos en el disco duro sin pararse a pensar en qué sistema operativo se está. Fíjate que este programa tiene las imágenes y los sonidos en una subcarpeta llamada data.Así que el sonido de explosión explode1.wav se escribiría en MacOS X y en Linux

  data/explode1.wav

  mientras que en Windows sería

  data\explode1.wav

  Para no tener que poner ambas y encima tener que comprobar en qué sistema operativo se está ejecutando el juego, os.path.join() es genial, pues nos une el nombre de la carpeta y el archivo correspondientes de la forma adecuada automáticamente.

- try y except son lo que se denomina gestores de excepciones. Si sabes que todo va a funcionar correctamente no es necesario nada más, pero un buen programador prevé los posibles errores que puedan surgir en la ejecución del programa y los trata adecuadamente. Imagínate que no están las imágenes donde deben estar. El programa

terminará con un error que a lo mejor nos es desconocido. Es allí donde entran try y except pues nos permiten detectar el error y hacer algo con ello. Después del try has de poner el código que deseas probar. Si se ejecuta sin problemas, el programa sigue con normalidad; pero si se produce un error, se salta a la instrucción que sigue al except. Allí se gestiona el tipo de error y se puede poner un mensaje, etc. Consulta la documentación de Python para más detalles. Una observación: en nuestro programa se usa la instrucción raise que lanza a su vez un error para indicar a Python que salga del programa (lógicamente, pues no se ha podido cargar la imagen que se quería). En cualquier caso, esto no es estrictamente necesario que aparezca en tus programas. Pero sí conveniente cuando se vuelven largos y complejos.

4.6 Función cargar sonido:

Esta función se encarga, en forma análoga a la función anterior, de convertir un archivo de sonido en un objeto Sound. De la misma forma que antes, si no comprobamos ningún error, la función quedaría muy sencilla; no obstante, vamos a hacer alguna comprobación para aprender alguna técnica nueva:

- Si hay algún problema con el sistema de sonido del ordenador, el programa no podrá ejecutarse. Antes de que nos dé algún error es mejor detectar este problema y hacer que no se reproduzca ningún sonido y que el resto del juego funcione sin problemas. La forma de hacerlo en el código es curiosa. Se define una nueva clase sinSonido que posee una función play(), como los sonidos. En la definición de esta función se usa simplemente la instrucción pass, es decir, no hacer nada. El objetivo es sencillo; cuando haya problemas con el sonido, se devuelve un objeto de este tipo y así cuando el juego invoque play() no se hará nada.

- Para comprobar si hay problemas con el sistema de sonido, dentro de un if (como es lógico) miramos si está disponible el módulo pygame.mixer (encargado de gestionar el sonido del juego) y también miramos si se ha inicializado correctamente con pygame.mixer.get_init(). En el caso de que alguno sea False, se devuelve la clase sinSonido del paso anterior.

- El resto es igual que previamente, pero ahora usando pygame.mixer.Sound() para cargar el archivo de sonido correspondiente.

```
sonido1 = pygame.mixer.Sound("Sonidos/music1.wav")
sonido1.play()
```

4.7 Clase nave espacial:

La clase nave espacial va a ser, por lo tanto, la que se encargue del sprite de la nave de nuestro protagonista. Como tal, es una clase derivada de pygame.sprite.Sprite.

Vamos a analizar su definición por partes:

- En __init__() se empieza por inicializar la clase de la que procede (algo, como ya vimos en el último tutorial, necesario). A continuación se definen sus atributos obligatorios, su image y su rect, este último usando el atajo de obtenerlo de la propia imagen gracias a la función get_rect(). La posición en la que se comenzará el sprite viene indicada por center, con lo que se usan las dimensiones de la ventana para situarlo. Finalmente, se definen las variables dx y dy para llevar el control de la velocidad de la nave.

- Pasemos a update(), quien controla cómo vamos a cambiar la posición de la nave en cada fotograma. La primera instrucción es la ya conocida con move_ip() que nos desplaza al sprite la cantidad indicada por dx y dy. Las siguientes líneas se encargan simplemente, de que la nave no sobrepase los límites de la zona por la que la dejamos mover; horizontalmente hasta los bordes de la ventana y verticalmente hasta la mitad. En tales casos, se fuerza a que la coordenada correspondiente tenga el valor límite y no lo supere.

```
#dibujo de la nave espacial
class naveEspacial(pygame.sprite.Sprite):
        def __init__(self):
                pygame.sprite.Sprite.__init__(self)
                self.ImagenNave = pygame.image.load("Fondos/image2.gif")
                self.ImagenExploscion = pygame.image.load("Fondos/explosion.gif")
                self.rect = self.ImagenNave.get_rect()
                #poscicion de la imgane en la pantalla de juego
                self.rect.centerx = ancho/2
                self.rect.centery = alto-30
                #disparos del jugador
                self.ListaDisparo = []
                self.vida= True
                self.velocidad= 20
                #sonido
                self.sonido2 = pygame.mixer.Sound("Sonidos/shoot.wav")
                self.sonido3 = pygame.mixer.Sound("Sonidos/explosion.wav")
```

4.8 Clase Enemigos:

El proceso de definición de esta clase es muy parecido a la anterior con algunos matices:

- La función __init__() usa ahora otro argumento más, posx, para permitir que las naves enemigas se creen de partida en lugares distintos. Eso lo podemos ver en la línea:

  self.rect.centerx = posx

  Que nos posiciona la nave de acuerdo a su valor (fíjate de paso que, en el eje vertical, la nave comienza en la coordenada 120, más arriba que la nave del protagonista). Por otra parte, la velocidad del sprite se genera al con la instrucción self.dx = random.randinty análogamente para dy. Ésa es la manera de conseguir que cada nave enemiga se mueva de forma distinta.

- update() es muy parecido también pero ahora, al llegar al borde, se invierte el movimiento de la nave (ya conocemos la técnica que consiste en cambiar de signo

su velocidad). Lo que es nuevo es la parte del código que se encarga de generar al azar disparos. La forma de hacerlo es muy típica. Primero hay que evitar que se dispare sin parar, en plan ametralladora; para eso, generamos un número aleatorio entre 1 y 60 y solamente cuando sea 1 se procede a disparar (de lo que se encarga un if). Para crear el disparo, se utilizan dos objetos que se definen más tarde en el código:

```
#Clase enemigos
class EnemigoI(pygame.sprite.Sprite):
        def __init__(self,posx,posy,distancia,imagen1):
                pygame.sprite.Sprite.__init__(self)
                self.imageA = pygame.image.load(imagen1) #carga la imagen
                self.rect=self.imageA.get_rect() #posicion a la imagen

                self.ListaDisparo=[]
                self.velocidad=20 #velocidad para el enemigo
                self.rect.top=posy #posiciones
                self.rect.left=posx
                self.rangoDisparo=5 # numero de disparos que puede dispara el enemigo
                self.derecha=True
                self.contador=0
                #controla movimiento en la pantalla de juego
                self.maxdescenso=self.rect.top+40
                self.limiteDerecha=posx + distancia
                self.limiteIzquierda=posx - distancia
                self.activar=True #para ver si la nave se sigue moviendo o no

        def comportamiento(self): #actividad que realiza el enemigo
                if self.activar==True:
                        self.__movimientos()
                        self.__ataque()
        def __movimientos(self):#movimiento en un pequeño rango de tiempo
                if self.contador<3:
                        self.__movimientoLateral()
                else:
                        self.__descenso()
        def __descenso(self):
                if self.maxdescenso>600: #sis se va de la pantalla se borra y se vuelve a crear
                        self.maxdescenso=0
                        crear=len(ListaEnemigo)
                        for enemigo in ListaEnemigo:
                                ListaEnemigo.remove(enemigo)
                        cargarEnemigos(crear)
                else:
                        if self.maxdescenso==self.rect.top:
                                self.contador=0
                                self.maxdescenso=self.rect.top+40
                        else:
                                self.rect.top+=1
```

4.9 Clase proyectil:

Esta clase define los sprites que representan los disparos de la nave del jugador. Al ser un sprite sencillo su código también lo es. Fíjate que en su __init__() se emplea un argumento añadido para indicar, cuando se cree el objeto correspondiente, en qué posición hay que hacerlo (igual que vimos con las naves del enemigo). Por otra parte, en update(), que es donde implementamos el movimiento, se hace una comprobación con un bloque if. La primero es preguntarse si el disparo ha llegado al borde superior de la ventana (utilizando su rect.bottom; entonces valdría 0), en cuyo caso hay que eliminar el sprite del juego (para que no ocupe memoria y tiempo):

self.kill()

En efecto, como puedes imaginar la función miembro kill() que posee todo sprite lo elimina por completo. Si no se da esta circunstancia, simplemente hay que mover el disparo hacia arriba usando su rect.move_ip(). Para desplazarlo 4 pixeles, en cada fotograma, hacia arriba, debemos pasar 0 para el incremento en el eje horizontal y -4 para el vertical.

```
#dibujo del bala
class proyectil(pygame.sprite.Sprite):
        def __init__(self,posx,posy,ruta,personaje):
                pygame.sprite.Sprite.__init__(self)
                self.imageProyectil = pygame.image.load(ruta)
                self.rect=self.imageProyectil.get_rect()
                self.velocidadDisparo=5
                self.rect.top=posy
                self.rect.left=posx
                self.disparoPersonaje=personaje
        #movimiento de la vida dependiendo de quien dispare
        def trayectoria(self):
                if self.disparoPersonaje==False:
                        self.rect.top=self.rect.top + self.velocidadDisparo
                else:
                        self.rect.top=self.rect.top - self.velocidadDisparo
        #dibuja la bala en la pantalla de juego
        def dibujar(self,superficie):
                superficie.blit(self.imageProyectil, self.rect)
```

4.10 Clase proyectil enemigo:

Clase idéntica a la anterior con los matices de que representa a los disparos de los enemigos en lugar de los del jugador; ahora el movimiento es hacia abajo, con lo que el incremento en el movimiento vertical ha de ser de 4 y se comprueba si el disparo sale de la ventana por su borde inferior y no por el superior. De paso observa el detalle de que se emplea para posicionar el sprite su rect.midtop por que luego se ajustará su valor en función del rect.midbottom de la nave enemiga (justo al contrario que con el disparo y la nave del protagonista)

5 CUERPO PRINCIPAL DEL JUEGO

Muchas cosas ocurren en esta parte del código, aunque todas siguiendo el esquema habitual:

5.1 Inicialización:

A parte del habitual pygame.init() y de crear la Surface del juego (a la que hemos llamado visor) y ponerle título a la ventana, se ha usado random.seed() para garantizar que el juego no se repite cuando se arranca cada vez. En realidad, cuando se carga el módulo random el sistema ya lo realiza automáticamente. He dejado esta instrucción por que si le pasamos un valor numérico concreto, el sistema lo usa como semilla para generar números aleatorios (algo muy interesante si queremos que los diferentes niveles de un juego comiencen siempre de la misma forma, es tu elección). También se carga la imagen de fondo del juego y se sitúa en visor con la función blit() en las coordenadas (0,0), es decir, ocupando toda la ventana (pues tanto imagen como ventana tienen el mismo tamaño).

5.2 Definición de variables de útiles:

Antes de empezar con el bucle de animación del juego, definimos tres variables que nos serán de utilidad. La variable jugando nos indica que el juego está en activo. La usaremos en el bucle while del juego y bastará poner su valor a False, por lo tanto, para que se salga del bucle y termine el juego. intervaloEnemigos lo vamos a usar para indicar cada cuanto hay que crear un nuevo enemigo. Inicialmente está a 0 y se irá incrementando en cada fotograma. Cuando llegue a un determinado valor (200, como veremos luego), será el momento de crear un enemigo más, poner la variable a 0 y vuelta a empezar. Por último, reloj es un objeto tipo Clock que, como ya vimos en un tutorial anterior, nos permitirá ajustar automáticamente la velocidad de reproducción de la animación del juego.

5.3 Bucle del Juego:

Pasamos al bucle principal del juego en el que cada iteración representa un fotograma de la animación. De hecho, la primera línea y el uso de reloj nos indica que la velocidad de reproducción va a ser de 60 fotogramas por segundo. Lo siguiente es comprobar los eventos que hayan ocurrido y nos interesen. Aquí hemos de decidir si hay que terminar el juego, o bien por que el jugador haya hecho click para cerrar la ventana (evento de tipo QUIT) o bien que haya pulsado la tecla escape (evento de tipo KEYDOWN, tecla pulsada K_ESCAPE). Todo esto nos debería resultar familiar.

Ahora viene una decisión importante. Cuando miramos las teclas que ha pulsado el jugador para gestionar el juego, lo podemos hacer de dos formas. Una es mirando en la cola de eventos (como acabamos de hacer) y otra es llamando a la función

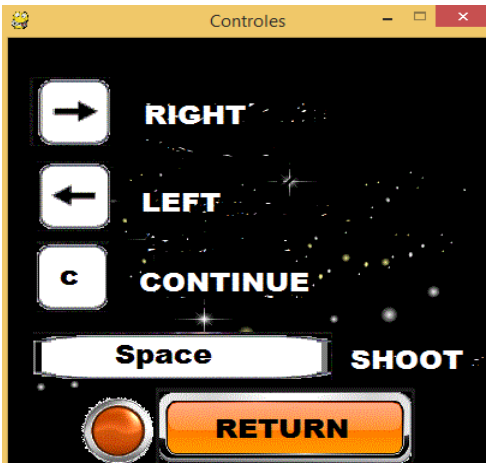
```
pygame.key.get_pressed()
```

y almacenando el resultado en una variable. El resultado de lo anterior es un diccionario en el que para cada tecla nos devuelve el valor True o False según esté pulsada o no.

¿Qué método usar? Cada uno tiene sus ventajas y susinconvenientes. El matiz está en que con la función pygame.key.get_pressed() averiguamos si una tecla está pulsada mientras que con la cola de eventos lo averiguamos en el momento en el que se ha pulsado. La diferencia es fundamental. En el movimiento de la nave, nos basta con saber si está pulsada la tecla correspondiente y mover la nave mientras dure la pulsación, pero en los disparos nos interesa el momento mismo en el que se produce dicha pulsación, pues le corresponde un único disparo y entonces necesitamos el método de la cola de eventos (en caso contrario, mientras estuviera la tecla pulsada tendríamos una ráfaga de disparos y el juego sería muy fácil). Dicho esto, podemos entender que se mire, en el evento KEYDOWN si se ha pulsado la tecla espacio (K_SPACE). En caso afirmativo se crea el disparo.

5.4 Ventana de ayuda y controles:

Vamos a empezar por la escena más sencilla, la que muestra la pantalla de ayuda. Lo que queremos mostrar tiene el siguiente aspecto:



Esta pantalla nos muestra los controles básicos de nuestro juego, movimiento de la nave botón para disparar y el botón para continuar nuestra partida una vez hayamos perdido.

5.5 Menú principal:

Ésta es la pantalla principal del juego, el menú inicial:

¿En qué consiste esta pantalla?

- Se muestran las opciones, que se iluminan al paso del ratón y dan lugar al resto de las escenas.
- El título del juego aparece por la parte superior hasta que alcanza su posición final.
- Se muestra un fondo estrellado, difuminado en su parte inferior para realzar el texto de las opciones del menú.

6 CODIGO FUENTE

```
#declaracion de librerias
import pygame
from pygame.locals import *
from tkinter import *
from random import randint
from tkinter import *
#declaracion de variables locales
ancho = 700
alto = 600
ListaEnemigo=[] #variable que gurdara a los enemigos
# cambio de los botones al pasar el mouse
class Boton(pygame.sprite.Sprite):
        def __init__(self,imagen1,imagen2,x=200,y=200):
                self.imagen_normal=imagen1 #imagen normal
                self.imagen_seleccion=imagen2 #imagen de cambio
                self.imagen_actual=self.imagen_normal
                self.rect=self.imagen_actual.get_rect() #posicion
                self.rect.left,self.rect.top=(x,y)
        def update(self,pantalla,cursor): #cambia de imagen del boton cuando el mouse pasa por encima
                if cursor.colliderect(self.rect): #si colisiona
                        self.imagen_actual=self.imagen_seleccion #cambia
                else:
                        self.imagen_actual=self.imagen_normal
                pantalla.blit(self.imagen_actual,self.rect)#si no se mantiene
#Mouse en la pantalla recoge la posicion
class Cursor(pygame.Rect):
        def __init__(self):
                pygame.Rect.__init__(self,0,0,1,1)
        def update(self):
                self.left,self.top=pygame.mouse.get_pos()
#crea una pantalla que muestra los controles de juego
def controles():
        salir = False
        pygame.init()
        screen = pygame.display.set_mode((400, 390))
        pygame.display.set_caption("Controles")
        ImagenFondo = pygame.image.load("Fondos/control.gif")
        botonModel1=pygame.image.load("Botones/btn1.gif")
        botonModel2=pygame.image.load("Botones/btn5.gif")
        boton1=Boton(botonModel1,botonModel2,50,320) #llama a la clase boton
        cursor1=Cursor() #activa el curso del mouse en la pantalla
        while salir!=True: #loop principal
                for event in pygame.event.get(): #recoore los eventos
                        if event.type == QUIT: #cuando se cierra la pantalla
                                salir = True
                        if event.type==pygame.MOUSEBUTTONDOWN: #accion con el teclado
                                if cursor1.colliderect(boton1.rect): #colision con el boton
                                        menuJuego()
                                        salir=True
```

```
                screen.blit(ImagenFondo,(0,0)) #pone el fondo de pantalla
                cursor1.update() #actualiza el boton
                boton1.update(screen,cursor1)#dibuja el boton en la pantalla
                pygame.display.update()
#creacion y grbacion en el txt
def creartxt():
        archi=open('Puntajes.txt','w')
        archi.close()
def grabartxt(l): #guarda los datos en el archivo
        archi=open('Puntajes.txt','a')
        archi.write(l+"\n")
        archi.close()
#muestra el contenido de un txt en pantalla(con listbox)
def puntajes():

        root = Tk()
        root.title('SCORES')
        archi=open('Puntajes.txt','r')
        linea=archi.read()
        li = linea.split("\n")
        listb = Listbox(root,width=40, height=10)
        for item in li:
                listb.insert(len(li),item)
        listb.pack()
        root.geometry('300x210+450+70')
        root.mainloop()
        """linea=archi.readline()
        while linea!="":
                print (linea)
                linea=archi.readline()
        archi.close()"""

        """linea=archi.read()
        cadena = linea
        var = ''
        for h in cadena:
                if cadena[h]=="":
                        var ="Puntaje "+h+": " + var
                        print(var)"""
#programacion para salir del programa
def salirJuego():
    import sys
    sys.exit(0)
#dibujo de la nave espacial
class naveEspacial(pygame.sprite.Sprite):
        def __init__(self):
                pygame.sprite.Sprite.__init__(self)
                self.ImagenNave = pygame.image.load("Fondos/image2.gif")
                self.ImagenExploscion = pygame.image.load("Fondos/explosion.gif")
```

```
                self.rect = self.ImagenNave.get_rect()
                #poscicion de la imgane en la pantalla de juego
                self.rect.centerx = ancho/2
                self.rect.centery = alto-30
                #disparos del jugador
                self.ListaDisparo = []
                self.vida= True
                self.velocidad= 20
                #sonido
                self.sonido2 = pygame.mixer.Sound("Sonidos/shoot.wav")
                self.sonido3 = pygame.mixer.Sound("Sonidos/explosion.wav")
        #movimiento de la nave controlando los extremos
        def movimientoDerecha(self):
                self.rect.right+=self.velocidad
                self.__movimiento()
        def movimientoIzquierda(self):
                self.rect.left-=self.velocidad
                self.__movimiento()
        def __movimiento(self):
                if self.vida==True:
                        if self.rect.left <=0:
                                self.rect.left=0
                        elif self.rect.right>700:
                                self.rect.right=700
        #disparo del jugador y le agrega a la lista de disparo
        def dispara(self,x,y):
                MiProyectil  = proyectil(x,y,"Balas/image3.gif",True)
                self.ListaDisparo.append(MiProyectil)
                self.sonido2.play()
        #dibuja el jugador en la pantalla de juego
        def dibujar(self,superficie):
                superficie.blit(self.ImagenNave,self.rect)
        #cambia de imagen cuando colisiona con la nave enemiga
        def cambiar(self):
                self.sonido3.play()
                self.ImagenNave=self.ImagenExploscion
#dibujo del bala
class proyectil(pygame.sprite.Sprite):
        def __init__(self,posx,posy,ruta,personaje):
                pygame.sprite.Sprite.__init__(self)
                self.imageProyectil = pygame.image.load(ruta)
                self.rect=self.imageProyectil.get_rect()
                self.velocidadDisparo=5
                self.rect.top=posy
                self.rect.left=posx
                self.disparoPersonaje=personaje
        #movimiento de la vida dependiendo de quien dispare
        def trayectoria(self):
                if self.disparoPersonaje==False:
```

```
                        self.rect.top=self.rect.top + self.velocidadDisparo
                else:
                        self.rect.top=self.rect.top - self.velocidadDisparo
        #dibuja la bala en la pantalla de juego
        def dibujar(self,superficie):
                superficie.blit(self.imageProyectil, self.rect)
#Clase enemigos
class EnemigoI(pygame.sprite.Sprite):
        def __init__(self,posx,posy,distancia,imagen1):
                pygame.sprite.Sprite.__init__(self)
                self.imageA = pygame.image.load(imagen1) #carga la imagen
                self.rect=self.imageA.get_rect() #posicion a la imagen

                self.ListaDisparo=[]
                self.velocidad=20 #velocidad para el enemigo
                self.rect.top=posy #posiciones
                self.rect.left=posx
                self.rangoDisparo=5 # numero de disparos que puede dispara el enemigo
                self.derecha=True
                self.contador=0
                #controla movimiento en la pantalla de juego
                self.maxdescenso=self.rect.top+40
                self.limiteDerecha=posx + distancia
                self.limiteIzquierda=posx - distancia
                self.activar=True #para ver si la nave se sigue moviendo o no

        def comportamiento(self): #actividad que realiza el enemigo
                if self.activar==True:
                        self.__movimientos()
                        self.__ataque()
        def __movimientos(self):#movimiento en un pequeño rango de tiempo
                if self.contador<3:
                        self.__movimientoLateral()
                else:
                        self.__descenso()
        def __descenso(self):
                if self.maxdescenso>600: #sis se va de la pantalla se borra y se vuelve a crear
                        self.maxdescenso=0
                        crear=len(ListaEnemigo)
                        for enemigo in ListaEnemigo:
                                ListaEnemigo.remove(enemigo)
                        cargarEnemigos(crear)
                else:
                        if self.maxdescenso==self.rect.top:
                                self.contador=0
                                self.maxdescenso=self.rect.top+40
                        else:
                                self.rect.top+=1
        #se mueve hasta el limite establecido en los lados
        def   movimientoLateral(self):
```

```
                if self.derecha == True:
                        self.rect.left=self.rect.left+self.velocidad
                        if self.rect.left > self.limiteDerecha:
                                self.derecha=False
                                self.contador+=1
                else:
                        self.rect.left = self.rect.left - self.velocidad
                        if self.rect.left < self.limiteIzquierda:
                                self.derecha =True


        def dibujar(self,superficie): # dibuja el enemigo en la pantalla
                self.ImagenInvasor=self.imageA
                superficie.blit(self.ImagenInvasor, self.rect)
        def __ataque(self):

                #controla el numero de balas que disparara el enemigo
                if (randint(0,100)<self.rangoDisparo):
                        self.__disparo()

        def __disparo(self): #dibuja los disparos
                x,y = self.rect.center
                MiProyectil = proyectil(x,y,"Balas/bullet.gif",False)
                self.ListaDisparo.append(MiProyectil)
#creacion de enemigos
def cargarEnemigos(crearenemigos):
        for i in range(crearenemigos):
                enemigo=EnemigoI(randint(50,285),randint(-100,300),randint(50,300),"Movimiento/mov2.gif")
                ListaEnemigo.append(enemigo)
#carga los enemigos especiales
def cargarEnemigoEspecial(nivel):
        if nivel==1:
                enemigo=EnemigoI(270,-10,210,"Enemigos/enemigo1.gif")
                ListaEnemigo.append(enemigo)
        if nivel==2:
                enemigo=EnemigoI(270,-10,210,"Enemigos/enemigo2.gif")
                ListaEnemigo.append(enemigo)
        if nivel==3:
                enemigo=EnemigoI(270,-10,210,"Enemigos/enemigo3.gif")
                ListaEnemigo.append(enemigo)
#pantalla de juego
def PantallaJuego():
        pygame.init()
        pantalla = pygame.display.set_mode([700,600])
        salir = False
        pygame.display.set_caption(" juego")
        ImagenFondo = pygame.image.load("Fondos/Stars.gif")
        sonido1 = pygame.mixer.Sound("Sonidos/music1.wav")
        sonido1.play()
        fuente1 = pygame.font.SysFont("Arial",24,True,False     )
```

```
info = fuente1.render("",0,(255,255,255))
jugador = naveEspacial()
enJuego=True
enJuego2=True
conE=True
reloj = pygame.time.Clock()
crearenemigos=1
enemigoespecial=0
enemigoVida=0
Nivel=1
cargarEnemigos(crearenemigos)
segundosint=0
puntaje=0
score = ""
while salir != True:#Loop principal
        try :

                reloj.tick(60)
                score = fuente1.render("SCORE: "+str(puntaje),0,(255,255,255))
                for event in pygame.event.get(): #recoore todo los eventos
                        if event.type == pygame.QUIT: #cuando se cierra la ventana
                                salir = True
                        if enJuego == True: # para que despues no pueda realizar ninguna actividad
                                if event.type == pygame.KEYDOWN: # movimeinto con el teclado
                                        if event.key == pygame.K_LEFT:
                                                jugador.movimientoIzquierda()
                                        if event.key == pygame.K_RIGHT:
                                                jugador.movimientoDerecha()
                                        if event.key == pygame.K_SPACE:
                                                x,y=jugador.rect.center
                                                jugador.dispara(x,y)
                        if enJuego == False:
                                sonido1.stop()
                                if event.type == pygame.KEYDOWN:
                                        if event.key ==pygame.K_c:
                                                grabartxt(str(puntaje))
                                                menuJuego()
                                                salir=True

                pantalla.blit(ImagenFondo,(0,0))

                #niveles
                if Nivel==1: #impresion
                        info = fuente1.render("LEVEL 1",0,(255,255,255))
                        pantalla.blit(info,(350,300))
                if Nivel==1:
                        if enJuego2 == False :
                                #creacion de enemigos cuando ya elimino a todos
                                if len(ListaEnemigo)==0 and crearenemigos<=2 and conE==True:
                                        crearenemigos+=1
```

```
                        cargarEnemigos(crearenemigos)
                        enJuego2=True
                #carga el enemigo especial cuando ya acabo con los enemigos principales
                if crearenemigos==3 and len(ListaEnemigo)==0 and conE==True:
                        cargarEnemigoEspecial(Nivel)
                        crearenemigos+=1
                #cuando ya elimina al enemigo especial
                if crearenemigos==4 and len(ListaEnemigo)!=0 and conE==True and enemigoVida==5:
                        enJuego=False
                        enemigo.activar=False
                        #borra todo (disparos de enemigo y jugador y la lista de enemigos)
                        #reinicia todos los valores
                        for enemigo in ListaEnemigo:
                                ListaEnemigo.remove(enemigo)
                        Nivel=2
                        crearenemigos=0
                        enJuego=True
                        enemigo.activar=True
                        enemigoVida=1
                        conE=True
        else:
                info = fuente1.render("GAME OVER",0,(255,255,255))

        if Nivel==2:#impresion
                info = fuente1.render("LEVEL 2",0,(255,255,255))
                pantalla.blit(info,(350,300))

        if Nivel==2:
                if enJuego2 == False :
                        #creacion de enemigos cuando ya elimino a todos
                        if len(ListaEnemigo)==0 and crearenemigos<=3 and conE==True:
                                crearenemigos+=1
                                cargarEnemigos(crearenemigos)
                                enJuego2=True
                #carga el enemigo especial cuando ya acabo con los enemigos principales
                if crearenemigos==4 and len(ListaEnemigo)==0 and conE==True:
                        cargarEnemigoEspecial(Nivel)
                        crearenemigos+=1
                #cuando ya elimina al enemigo especial
                if crearenemigos==5 and len(ListaEnemigo)!=0 and conE==True and enemigoVida==8:
                        enJuego=False
                        enemigo.activar=False
                        #borra todo (disparos de enemigo y jugador y la lista de enemigos)
                        #reinicia todos los valores
                        for enemigo in ListaEnemigo:
                                ListaEnemigo.remove(enemigo)
                        Nivel=3
                        crearenemigos=0
                        enJuego=True
                        enemigo.activar=True
```

```
                enemigoVida=1
                conE=True
else:
        info = fuente1.render("GAME OVER",0,(255,255,255))

if Nivel==3:
        info = fuente1.render("LEVEL 3",0,(255,255,255))
        pantalla.blit(info,(350,300))

if Nivel==3:#impresion
        info = fuente1.render("GAME OVER",0,(255,255,255))
        if enJuego2 == False :
                #creacion de enemigos cuando ya elimino a todos
                if len(ListaEnemigo)==0 and crearenemigos<=4 and conE==True:
                        crearenemigos+=1
                        cargarEnemigos(crearenemigos)
                        enJuego2=True
        #carga el enemigo especial cuando ya acabo con los enemigos principales
        if crearenemigos==5 and len(ListaEnemigo)==0 and conE==True:
                cargarEnemigoEspecial(Nivel)
                crearenemigos+=1
        #cuando ya elimina al enemigo especial
        if crearenemigos==6 and len(ListaEnemigo)!=0 and conE==True and enemigoVida==11:
                enJuego=False
                enemigo.activar=False
                for enemigo in ListaEnemigo:
                        ListaEnemigo.remove(enemigo)

                Nivel=4
if Nivel==4:
        info = fuente1.render("COMPLETED GAME!!",0,(255,255,255))
        sonido1.stop()

jugador.dibujar(pantalla)
#bala de jugador contra enemigo
if len (jugador.ListaDisparo)>0:
        for x in jugador.ListaDisparo:
                x.dibujar(pantalla)
                x.trayectoria()
                if x.rect.top<10:
                        jugador.ListaDisparo.remove(x)
                else:
                        for enemigo in ListaEnemigo:
                                if x.rect.colliderect(enemigo.rect):
                                        jugador.ListaDisparo.remove(x)
                                        if Nivel==1:
                                                if crearenemigos==4 and len(ListaEnemigo)!=0 and conE==True:
                                                        enemigoVida+=1
                                                        puntaje+=10
                                                        ListaEnemigo.remove(enemigo)
```

```
                                                #print("vidaenemigo:"+str(enemigoVida))
                                                cargarEnemigoEspecial(Nivel)
                                        else:
                                                ListaEnemigo.remove(enemigo)
                                                enJuego2=False
                                                puntaje+=10
                                                score = fuente1.render("SCORE: "+str(puntaje),0,(255,255,255))

                                if Nivel==2:
                                        if crearenemigos==5 and len(ListaEnemigo)!=0 and conE==True:
                                                enemigoVida+=1
                                                puntaje+=10
                                                ListaEnemigo.remove(enemigo)
                                                #print("vidaenemigo:"+str(enemigoVida))
                                                cargarEnemigoEspecial(Nivel)
                                        else:
                                                ListaEnemigo.remove(enemigo)
                                                enJuego2=False
                                                puntaje+=10
                                                print(puntaje)
                                if Nivel==3:
                                        if crearenemigos==6 and len(ListaEnemigo)!=0 and conE==True:
                                                enemigoVida+=1
                                                puntaje+=10
                                                ListaEnemigo.remove(enemigo)
                                                #print("vidaenemigo:"+str(enemigoVida))
                                                cargarEnemigoEspecial(Nivel)
                                        else:
                                                ListaEnemigo.remove(enemigo)
                                                enJuego2=False
                                                puntaje+=10
                                                print(puntaje)

pantalla.blit(score,(50,550))
#choque de enemigo con el jugador
if len(ListaEnemigo)>0:
        for enemigo in ListaEnemigo:
                enemigo.comportamiento()
                enemigo.dibujar(pantalla)
                if enemigo.rect.colliderect(jugador.rect):
                        sonido1.stop()

        if len (enemigo.ListaDisparo)>0:
                for x in enemigo.ListaDisparo:
                        x.dibujar(pantalla)
                        x.trayectoria()
                        if x.rect.colliderect(jugador.rect):
                                #bala de enemigo contra el jugador
                                jugador.cambiar()
                                for enemigo in ListaEnemigo:
```

```
                                                enemigo.activar=False
                                        for enemigo in ListaEnemigo:
                                                ListaEnemigo.remove(enemigo)
                                        for disparo in jugador.ListaDisparo:
                                                jugador.ListaDisparo.remove(disparo)
                                        for disparo in enemigo.ListaDisparo:
                                                enemigo.ListaDisparo.remove(disparo)
                                        enJuego=False
                                        conE=False
                                        sonido1.stop()
                                if x.rect.top >700:
                                        enemigo.ListaDisparo.remove(x)
                                else:
                                        #bala contra bala
                                        for disparo in jugador.ListaDisparo:
                                                if x.rect.colliderect(disparo.rect):
                                                        jugador.ListaDisparo.remove(disparo)
                                                        enemigo.ListaDisparo.remove(x)
                        if enJuego==True:
                                segundosint = pygame.time.get_ticks()/1000
                                segundos = str(segundosint)
                                contador1 = fuente1.render("TIME: "+segundos,0,(255,255,255))
                                pantalla.blit(contador1,(550,550))


                        if enJuego==False or conE==False:
                                pantalla.blit(info,(330,330))
                                pantalla.blit(contador1,(450,30))
                                pantalla.blit(score,(450,80))



                                pygame.display.update()
                        except ValueError:
                                print("Oops! No era válido. Intente nuevamente...")
#menu de juego
def menuJuego():
        salir = False
        pygame.init()
        screen = pygame.display.set_mode((400, 380))
        pygame.display.set_caption("MENU")
        ImagenFondo = pygame.image.load("Fondos/image.gif")
        botonModel1=pygame.image.load("Botones/btn1.gif")
        botonModel2=pygame.image.load("Botones/btn1.1.gif")
        botonModel3=pygame.image.load("Botones/btn4.gif")
        botonModel4=pygame.image.load("Botones/btn3.gif")
        botonModel5=pygame.image.load("Botones/btn2.gif")
        boton1=Boton(botonModel1,botonModel2,50,20)
        boton2=Boton(botonModel1,botonModel3,50,100)
        boton3=Boton(botonModel1,botonModel4,50,180)
```

```
        boton4=Boton(botonModel1,botonModel5,50,260)
        cursor1=Cursor()
        while salir !=True:
                for event in pygame.event.get():
                        if event.type == QUIT:
                                salir = True
                        if event.type==pygame.MOUSEBUTTONDOWN:
                                if cursor1.colliderect(boton1.rect):
                                        PantallaJuego()
                                        salir=True
                                if cursor1.colliderect(boton4.rect):
                                        salirJuego()
                                if cursor1.colliderect(boton3.rect):
                                        puntajes()
                                if cursor1.colliderect(boton2.rect):
                                        controles()

                screen.blit(ImagenFondo,(0,0))
                cursor1.update()
                boton1.update(screen,cursor1)
                boton2.update(screen,cursor1)
                boton3.update(screen,cursor1)
                boton4.update(screen,cursor1)
                pygame.display.update()
#llama a los metodos iniciales

creartxt()
menuJuego()
```

7 RECOMENDACIONES

- Profundizar nuestros conocimientos en este lenguaje tan flexible.
- Dedicar más tiempo a la programación para perfeccionarnos en un futuro profesionalmente.
- La mayor parte de la documentación de esta librería pygame se la encuantra en ingles poco a poco aparecen nuevas páginas que tratan de de acercar a los usuarios latinos.

8 CONCLUSIONES

- Lenguaje agradable y de fácil aprendizaje.
- Lenguaje en auge y muy extendido.
- Al ser de código abierto existe mucha documentación.
- La última versión de python no es totalmente compatible con anteriores versiones.

9 BIBLIOGRAFIA

[1] Abelson, H., Sussman, G. J. and Sussman, J.: Structure and Interpretation of Computer Programs, Segunda edici´on, 1996, MIT Press/McGraw-Hill.

[2] Allen Downey, Jeff Elkner and Chris Meyers: How to Think Like a Computer Scientist: Learning with Python. Green Tea Press. ISBN: 0971677506. http://www.ibiblio.org/obp/thinkCSpy/dist/thinkCSpy.pdf.

[3] Alan Gauld: Learn to Program Using Python: A Tutorial for Hobbyists, Self- Starters, and All Who Want to Learn the Art of Computer Programming. Addison-Wesley. ISBN: 0201709384.

[4] LAMP: The Open Source Web Platform. http://www.onlamp.com.