Instituto tecnológico de Costa Rica

Taller de programación

Grupo 1

Documentación Proyecto Space impact

Profesor: Jeff Schmidt Peralta

Estudiante: Harold Madriz Cerdas

2023

Introducción:

El juego Space Impact el cual es un juego para celular Nokia 3310/3330 que apareció en el año 2000. Luego de el primer juego salió para Nokia 3520/Nokia 5550 Space Impact 2. Por último, Space Impact Evolution salió para Nokia 7650.

El jugador se podía mover en los ejes Arriba/Abajo, Izquierda/Derecha.

Este juego consiste en una nave con una cantidad de balas o municiones ilimitadas. La nave puede cambiar de posición mediante a los comandos ‘w’, ‘a’, ‘s’, ‘d’, arriba, izquierda, abajo, derecha respectivamente. La trayectoria de las balas va de izquierda a derecha comenzando en la posición de la nave, la frecuencia de disparo tiene un límite establecido.

Los enemigos pueden disparar, al ser impactada la nave le resta un punto de vida de los 3 que tiene siendo 0 cuando se pierde en el juego.

Existen 3 dificultades, la primera se inicia al darle play al juego, la segunda cuando el puntaje es 10 y la tercera cuando el puntaje es 22.

Descripción del problema:

Este Proyecto debe hacerse en Python sin utilizar clases incluyendo el diseño y desarrollo de un videojuego llamado "Retro Space", el cual es un remake del clásico juego "Space Impact". En el juego, el usuario controla una nave espacial que dispara balas para destruir a los enemigos que aparecen en la pantalla. El número de balas es ilimitado y la nave puede moverse en diferentes direcciones utilizando el sistema de entrada de comandos de movimiento. Cada vez que la nave espacial golpea a un enemigo, se otorgan puntos según la cantidad de enemigos destruidos. Sin embargo, los enemigos también pueden disparar, y si la nave es golpeada, pierde salud. El juego se divide en tres niveles de dificultad. El jugador tiene 3 puntos de vida, si pierde todos los puntos de vida, pierde el juego. El juego debe tener una pantalla de inicio con un nombre, opciones y un botón de finalización, así como música de fondo ajustable. Desde esta pantalla, puedes elegir un nivel de dificultad y acceder a la pantalla del juego.

Análisis de Resultados:

Los resultados obtenidos en este proyecto no han alcanzado los objetivos establecidos. Aún así la compilación del código es correcta las funciones hechas funcionan correctamente.

La inserción de imágenes en las que se pueda ver sin un ‘fondo’ de algún color fue un inconveniente poco esperado, resolviéndolo creando una función para cargar imágenes a distintos ‘create_ image’ esto hace que se creen la imágenes correctamente encima de un canvas ya establecido como fondo.

```
'Cargar Imágenes'
def cargarimg(archivo):
    ruta = os.path.join('Archivos', archivo) #pone la ruta para cargar los archivos
    imagen = PhotoImage(file=ruta)  #hace que se sepa que la imagen está en la ruta
    return imagen
```

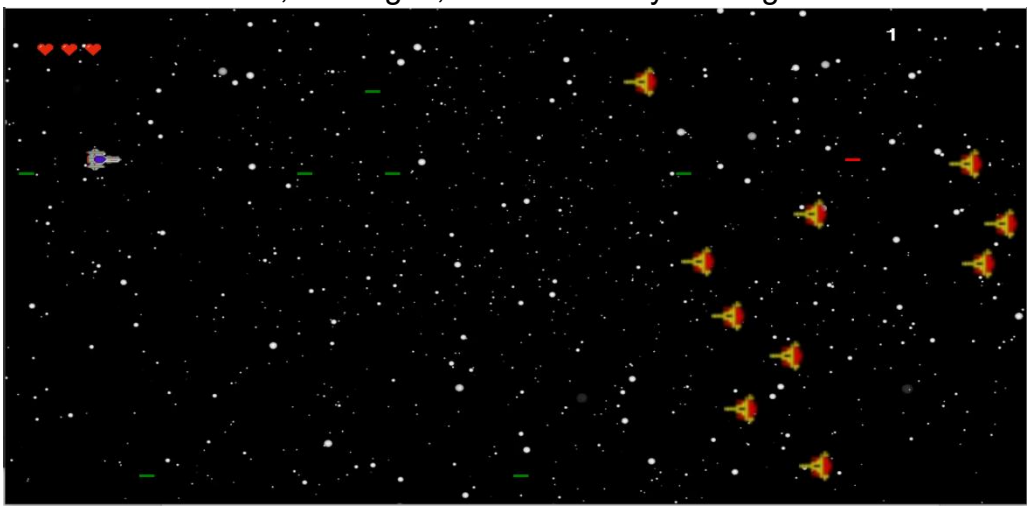
```
    'indicador de vida'
    corazonimg = cargarimg('vida.png')
```

El sonido se creó mediante la biblioteca ‘pygame’, creando funciones para los distintos sonidos del juego y llamándolas en sus respectivos lugares de reproducción.

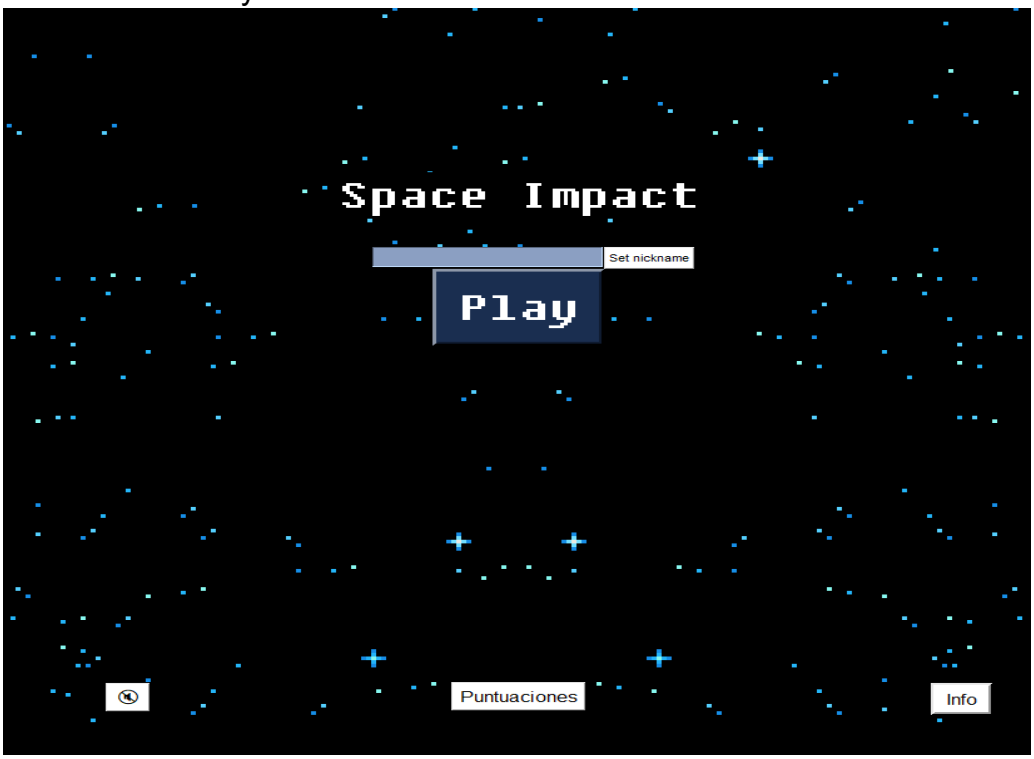
```
'canción'
pygame.init()   #inicia pygame
muted = False   #variable para mutear
volumen_actual = 0.1    #volumen
def music():    #función para canción principal
    pygame.mixer.init()        #inicializa el mezclador de sonido de Pygame
    pygame.mixer.music.load('Archivos/menusong.mp3')  #ruta
    pygame.mixer.music.set_volume(volumen_actual)  #volumen
    pygame.mixer.music.play()   #play

music() #hace que se inicie music al correr
```

El correcto movimiento de todos los objetos creados en el juego se hizo mediante un bucle llamado loop y varios ‘.after’ para la repetición de movimiento haciendo que se vea de manera fluida.

Se muestra en pantalla la cantidad de vidas de la nave y la puntuación obtenida.
Se muestra la nave, enemigos, balas aliadas y enemigas



La ventana principal tiene distintos widgets para la correcta ejecución de el juego, estas como: el botón de play, mutear música e información. Esto logrado con diferentes Label y Button.



Bitácora:

01/04/2023: Se recibe el documento con las instrucciones del proyecto

04/04/2023: Se crea la ventana principal y algún indicio de widgets

09/04/2023: Se añaden todos los widgets a la ventana principal

13/04/2023: Se añade fondo a la ventana principal y se crean la función de juego

17/04/2023: Creación de la nave aliada con todos sus movimientos

20/04/2023: Dificultad fácil terminada con todos sus enemigos

25/04/2023: Dificultad media terminada y añadida la información a al botón de info en la ventana principal.

29/04/2023: Terminada la dificultada difícil y se agrega la función de la edición del puntaje en pantalla.

Dificultades encontradas:

Durante el desarrollo se encontró con la dificultad de como bajar el sonido de una música con un botón en pantalla, esto se resolvió creando una variable de volumen global y creando una función que ponga este volumen en 0, haciendo así que se ponga en mute la canción principal.

Segundo problema encontrado fue la creación de una segunda ventana para el juego mientras la primera se minimiza, resolviéndolo fácilmente creando una función que llama la ventana de el juego y minimiza la ventana principal, además se agregó que al cerrar la ventana de el juego vuelva la ventana principal con otra función.

Una dificultad mayor fue el movimiento de la nave, ya que este se movía de manera que no se veía fluido y no avanzaba si se mantenía la tecla presionada. Se logró resolver creando un .after para llamar una y otra vez el movimiento.

Los .after crearon un problema que hacía que no parara nunca el movimiento, se resolvió haciendo una variable y 4 funciones de stop, la variable al ser False hace que pare el .after haciendo así que pare la nave.

Al agregar distintas naves enemigas con sus respectivas balas hacía que el movimiento de la nave fuera poco asertivo por la interrupción de varios .after entre sí, resolviéndolo con un bucle llamado loop que hace que se ejecuten correctamente.

Al llamar a las dificultades en loop hacía que se crearan enemigos en exceso, se pudo resolver haciendo una variable para limitar la cantidad de enemigos creados en cada dificultad.

Conclusiones:

Los .after son una alternativa útil a los hilos ya que estos son bastante más complejos.

Los objetos PhotoImage se usan correctamente cuando es necesario abrir un png sin fondo en una ventana.

El uso de print para comprobar si algunos If o else se están ejecutando correctamente es bastante eficiente.

Las variables globales para la resolución de bucles infinitos es una buena manera de arreglar estos.

El movimiento de objetos con teclas se hace con .bind para detectar la tecla y ejecutar la función requerida.

La biblioteca random funciona si se necesita crear una función que ocurra en algunas ocasione aleatorias.

Una correcta documentación de código facilita la edición de este en el futuro.

Fuentes:

How do I use the .create_image in Tkinter in Python

https://stackoverflow.com/a/65292619

Python Tkinter, destroy toplevel after function

https://stackoverflow.com/a/47307404