**Instituto Tecnológico de Costa Rica**

**Ingeniería en Computadores**

**Proyecto I: “BulletHead”**

**Elaborado por: Daniel Montoya Rivera**

**Profesor: Milton Villegas Lemus**

**Grupo 2**

**I Semestre**

**2020**

**Introducción**

El presente proyecto se basa en la creación de un videojuego cuyo objetivo es la eliminación y evasión de los diversos enemigos que aparecen en una interfaz gráfica durante cierto periodo de tiempo, de manera que replique o asemeje al videojuego “BulletHead”. Dicho proyecto utiliza módulos e instrucciones de software vistos en clase, como por ejemplo la recursividad, y también aquellos que fueron investigados mediante la documentación oficial de Python, foros en línea y demás. La versión de Python implementada en la programación del juego es Python 3.8.2

La temática del juego está basada en la franquicia “DOOM”, donde el jugador controla al personaje principal de esta, llamado “Doom Slayer” o “Doomguy” y tiene que luchar contra demonios que son enemigos representativos del mismo juego. El jugador cuenta con tres vidas y puede hacer que el personaje dispare proyectiles que se mueven en una trayectoria vertical para poder defenderse de los tres enemigos que aparecen de manera constante en el escenario del juego, y eliminarlos hasta que acabe el tiempo, a su vez debe evitar los ataques que cada enemigo realiza, además de evitar que haya una colisión entre ambos, ya sea esquivándolos o eliminándolos, de otra manera el jugador pierde ciertos puntos de vida. El juego se gana una vez el tiempo, de 40 segundos, llega a cero y se pierde cuando el jugador pierde todas sus vidas.

Los controles de movilidad y disparo del personaje se muestran en una interfaz que también enseña el objetivo del juego y las mecánicas de este. Dentro de esta misma interfaz se pueden editar los controles, lo cual permite adaptar la movilidad del personaje a las necesidades del usuario.

Para las funciones que requirieron repeticiones (como el movimiento, el temporizador, el chequeo constante de la vida del personaje y colisiones) se implementó recursión. Además, para dichas funciones, se utilizó threads o hilos para sus llamadas, esto con el propósito de llevar a cabo el funcionamiento paralelo del juego.

**Conclusiones**

Tras completar el videojuego, analizando su funcionamiento, en conjunto con los datos de las pruebas realizadas y los objetivos del proyecto, se concluye lo siguiente:

1. Se implementó el módulo “Principal” encargado de la creación de la interfaz gráfica.
2. Para limitar el tamaño de la interfaz se utilizó el módulo “Principal”
3. Para la creación de imágenes se utilizó el módulo “cargarImg2”
4. Se agregó el módulo “Menu” encargado de la creación del menú, el cual realiza las llamadas a otras funciones por medio de botones.
5. Se añadió el módulo “Help”, que incluye los controles del juego y el objetivo de este.
6. Para editar y validar los controles se utilizó el submódulo de “Help” llamado “CambioControles”.
7. Se colocó el módulo “Scores”, que muestra las puntuaciones de cada jugador
8. Se implementó el módulo “Credits”, el cual muestra información acerca del programa, su autor y la información acerca de los módulos utilizados.
9. Se añadió el botón “Exit Game”. Este botón permite salir del programa, no sin antes preguntarle al usuario por medio de un mensaje de confirmación si realmente desea salir
10. Para reproducir música en el juego se implementó el botón “Play Music”.
11. Se puso un botón de retorno al “Menu” en cada módulo que el Menu puede acceder, excepto la interfaz del juego.
12. Se agregó el módulo “Play”, que solicita al usuario el nombre y realiza la llamada al módulo de juego.
13. Para la validación del nombre ingresado se utilizó un submódulo de “Play” llamado “Play\_aux”
14. Se añadió el módulo “InterfazJuego” encargado de la realización de la lógica del juego. Este módulo integra las colisiones, los movimientos de enemigos y del personaje que uno opera, así como el registro de tiempo, puntuación y vida. Realiza llamadas a ciertos submódulos por medio de Threads.
15. Se implementó el submódulo de “InterfazJuego” llamado “MovDoom” encargado de las todas acciones de movimiento del personaje operado por el usuario.
16. Para las funciones de salto del personaje se implementó dentro del módulo “MovDoom” el módulo “MovDoomUp” y “MovDoomDown”

1. Para las funciones de disparo del personaje se añadió dentro de “MovDoom” el módulo llamado “DisparoDoom”.
2. Para el movimiento recursivo del disparo se agregó dentro de “DisparoDoom” el módulo “RecursionMovBullet”, en este mismo se evalúan las colisiones.
3. Se colocó dentro de “InterfazJuego” el módulo “Timer”, encargado del conteo de los 40 segundos. Este es llamado por medio de un Thread.
4. Se implementó dentro del módulo “InterfazJuego” la función llamada “TimerVidaYPuntuacion”, encargada de mostrar y actualizar el tiempo restante, la vida del personaje, puntuación y de finalizar el juego una vez se acaba el tiempo o la cantidad de vidas del jugador.
5. Se añadió el submódulo de “InterfazJuego” llamado “GeneraEnemigos”, encargado de generar al enemigo común.
6. Para las funciones de movimiento del enemigo común se implementó dentro de “InterfazJuego” el módulo llamado “Enemigo”, este es llamado por “GeneraEnemigos”
7. Dentro de “InterfazJuego” se añadió “RecursiveMovDer” que se encarga de hacer que el enemigo común se mueva hacia la derecha hasta alcanzar el límite del escenario.
8. Se colocó dentro de “Enemigo” el submódulo “RecursiveShotDemonDown” cuya función es disparar un proyectil.
9. Se implementó el submódulo de “InterfazJuego” llamado “RecursiveMovIzq” el cual tiene la labor de hacer que el enemigo común se mueva hacia la izquierda hasta alcanzar el límite del escenario.
10. Para realizar el movimiento del enemigo común hacia abajo se agregó el submódulo de “InterfazJuego” llamado
11. La desaparición del enemigo se colocó en “RecursiveMovDown”, “RecursiveMovDer” y “RecursiveMovIzq”
12. Se añadió el submódulo de “InterfazJuego” llamado “GeneraMisiles”, encargado de generar al misil en caída libre.
13. Para el patrón de movimiento del misil en caída libre se implementó dentro de “InterfazJuego” el módulo llamado “Enemigo2”, este es llamado por “GeneraMisiles”
14. Se agregó el submódulo de “InterfazJuego” llamado “RecursionMovMisil”, encargado del movimiento hacia abajo del misil hasta el límite del escenario.
15. Las colisiones del misil se implementaron en “RecursionMovMisil”.
16. Se añadió el submódulo de “InterfazJuego” llamado “GeneraAplasta”, encargado de generar el enemigo aplastador.
17. Para el patrón de movimiento del enemigo aplastador se implementó dentro de “InterfazJuego” el módulo llamado “Enemigo3”, este es llamado por “GeneraMisiles”.
18. Se agregó dentro de “InterfazJuego” los módulos llamados “RecursionMovAplastaAbajo”, “RecursionMovAplastaDer”, “RecursionMovAplastaIzq” quienes se encargan del movimiento de enemigo aplastador.
19. Para las colisiones del enemigo aplastador con el disparo del jugaror, se trabajó en el módulo “RecursionMovAplastaAbajo” y la colisión del mismo con el jugador se utilizó el módulo “RecursionMovAplastaIzq” y “RecursionMovAplastaDer”
20. Para el disparo del enemigo aplastador se añadió a “InterfazJuego” el modulo “DisparoRecursivoAplasta” que genera el disparo basado en la posición en que se encuentra el enemigo aplastador.
21. Se implementó el módulo de “InterfazJuego” llamado “RecusionBalaAplasta” que se encarga del movimiento del proyectil hacia el lado opuesto del escenario.
22. La variación de la puntuación y vidas del jugador se trabajó en conjunto con las colisiones.
23. Se implementa el módulo llamado “GameOver” el cual es llamado por “TimerVidaYPuntuacion”.
24. Para mostrar si el jugador ganó o perdió, se trabajó con el módulo “GameOver”, a su vez el módulo añade a un archivo la puntuación y nombre del jugador mediante un submódulo llamado “setPuntuacion”.
25. Para un mejor manejo de las colisiones entre los distintos objetos del juego, los controles, la vida y las puntuaciones, se trabajó con variables globales.
26. La variación de movimiento de los objetos del juego se implementó con recursión alterando sus coordenadas en el Canvas.
27. Las teclas de movimiento del personaje manejado por el usuario se implementaron mediante un “KeyRelease”, esto para evitar que se generase un efecto de rayo láser cuando se dispara.

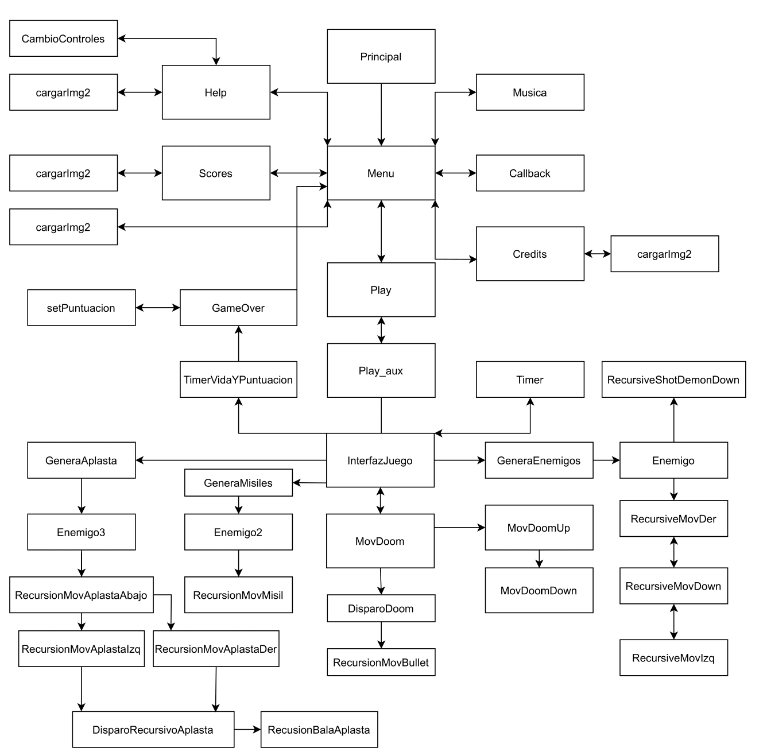
**Recomendaciones**

1. Utilizar la librería de pygame, ya que esta última permite el pausado de la misma y tiene más funciones en comparación a playsound.
2. Para la creación de imágenes, a menos que el formato de todas sea .png o .gif, se recomienda importar la librería Pillow de Alex Clark, ya que esta permite el uso de imágenes de diversos formatos
3. Se recomienda validar la edición de los controles del juego, para evitar fallos a la hora de jugar.
4. Cuando se presiona el botón Exit Game es recomendable mostrar un mensaje de confirmación en lugar de salir de inmediato, esto para evitar una salida accidental del juego
5. Al ingresar el nombre antes de jugar se recomienda limitar su número de caracteres, porque en caso de que el nombre fuese muy extenso, esto haría que, al mostrar el nombre en las puntuaciones, este se saliera del rango o esconda el resto de las puntuaciones
6. Al inicializar el juego se recomienda que la coordenada inicial del personaje con respecto a la coordenada inicial de los enemigos debe ser diferente para evitar que el juego reconozca ambas coordenadas como una colisión.
7. Se recomienda que una vez que ocurra una colisión entre la bala y un enemigo, o un objeto enemigo y el personaje, se borre tanto la imagen como la coordenada de la bala y el objeto enemigo, esto para evitar que se genere un doble, o mayor registro de colisión.
8. Se recomienda que cuando un objeto, ya sea enemigo o del personaje, llegue a su límite con respecto a su patrón de movimiento o escenario, el objeto debe ser eliminado con el fin de prevenir que el juego se vuelva más lento.
9. Cuando el tiempo o la vida del personaje llegan a 0, se deben eliminar las imágenes y coordenadas de todos los objetos del juego y se tienen que finalizar los threads u operaciones recursivas, esto para evitar fallos en el programa y para prevenir que el juego se pegue a causa de la acumulación excesiva de datos.
10. Para trabajar las colisiones de los objetos del juego, se recomienda no trabajar utilizando coordenadas porque se pueden generar registros de impacto doble o superior, o bien no se puede generar ningún registro de colisión.

.

**Análisis de resultados**

**Diagrama de Módulos**

****

**Plan de Pruebas**

Durante la ejecución del proyecto, se realizaron diversas pruebas que permitieran evaluar el funcionamiento adecuado del proyecto, así como la detección de errores tanto de sintaxis como propios de la lógica del programa, o bien para analizar el comportamiento de ciertas funciones. Entre estas pruebas se realizaron:

* Análisis de errores mostrados por el Shell de Python.
* Variaciones en los datos de entrada en las funciones.
* Aplicación del método de prueba y error
* Uso de la función print().
* Uso del debbuger.

Es importante señalar que las pruebas no solo se realizaron dentro de los módulos principales, sino que también se aplicaron a módulos secundarios, esto con el fin de identificar con mayor eficiencia el origen de una discrepancia dentro del programa.

Los datos obtenidos al final de cada prueba facilitaron no solo a la solución de ciertos errores, sino que también ayudaron a optimizar el funcionamiento del programa, por medio de la reducción o adición de funciones, variables globales y demás. Cabe destacar también que el uso del debbuger y de la función print ayudaron a mejorar y a hacer más precisas a ciertas funciones recursivas o las funciones llamadas por medio de hilos.

**Fuentes consultadas**

DelfStack. (25 de Junio de 2020). *How to Set Window Icon in Tkinter*. Obtenido de DelfStack: https://www.delftstack.com/howto/python-tkinter/how-to-set-window-icon-in-tkinter/

effbot. (s.f.). *The Tkinter Canvas Widget*. Obtenido de tkinterbook: https://effbot.org/tkinterbook/canvas.htm

programcreek. (s.f.). *Python tkMessageBox.askokcancel() Examples*. Obtenido de programcreek: https://www.programcreek.com/python/example/9949/tkMessageBox.askokcancel

Python. (s.f.). *random — Generate pseudo-random numbers*. Obtenido de Python 3.8.4 documentation: https://docs.python.org/3/library/random.html

Python. (s.f.). *tkinter — Python interface to Tcl/Tk*. Obtenido de Python3.8 Documentation: https://docs.python.org/3/library/tkinter.html

Python Tkinter Course. (s.f.). *Events and Binds*. Obtenido de Pyhon Course: https://www.python-course.eu/tkinter\_events\_binds.php#:~:text=In%20computing%20an%20event%20is,a%20key%20on%20the%20keyboard

stack overflow. (2009). *How to remove tab indent from several lines in IDLE?* Obtenido de stack overflow: https://stackoverflow.com/questions/790711/how-to-remove-tab-indent-from-several-lines-in-idle

stack overflow. (2012). *How to generate keyboard events in Python?* Obtenido de stack overflow: https://stackoverflow.com/questions/13564851/how-to-generate-keyboard-events-in-python

stack overflow. (2012). *How to use an image for the background in tkinter?* Obtenido de stack overflow: https://stackoverflow.com/questions/10158552/how-to-use-an-image-for-the-background-in-tkinter

stack overflow. (2014). *How do you call an instance of a class in Python?* Obtenido de stack overflow: https://stackoverflow.com/questions/24253761/how-do-you-call-an-instance-of-a-class-in-python

stack overflow. (2015). *Hide label when a button is clicked in Python*. Obtenido de stack overflow: https://stackoverflow.com/questions/27654494/hide-label-when-a-button-is-clicked-in-python

stack overflow. (2015). *Opening txt file in a label in Python*. Obtenido de stack overflow: https://stackoverflow.com/questions/33438952/opening-txt-file-in-a-label-in-python

tkinterbook. (s.f.). *Events and Bindings*. Obtenido de effbot: https://effbot.org/tkinterbook/tkinter-events-and-bindings.htm

tutorialspoint. (s.f.). *Python - Object Oriented*. Obtenido de tutorialspoint: https://www.tutorialspoint.com/python/python\_classes\_objects.htm

tutorialspoint. (s.f.). *Python - Tkinter Button*. Obtenido de tutorialspoint: https://www.tutorialspoint.com/python/tk\_button.htm

tutorialspoint. (s.f.). *Python - Tkinter Label*. Obtenido de tutorialspoint: https://www.tutorialspoint.com/python/tk\_label.htm

tutorialspoint. (s.f.). *Python 3 - Tkinter tkMessageBox*. Obtenido de tutorialspoint: https://www.tutorialspoint.com/python3/tk\_messagebox.htm

w3schools. (s.f.). *Python File Open*. Obtenido de w3schools: https://www.w3schools.com/python/python\_file\_handling.asp