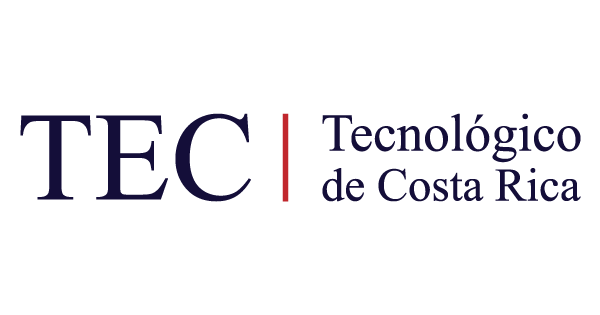
****

**Área Académica de Ingeniería en Computadores**

**Curso:** Taller de Programación

**Grupo:** 02

**Profesor:** Milton Villegas Lemus

**Proyecto 1**

**Alumno:**

García Downing Geovanny, 2020092224

**Fecha de entrega:** 24/05/2021

**I Semestre**

**2021**

1. **Introducción**

El siguiente proyecto aborda la creación de un juego programado en Python 3.9.2 de estilo *“Shoot them up”* dicho juego consta de tres niveles donde el jugador debe disparar al jefe para que este muera. Dichos jefes tienen distintos modos de ataque y hay bonus por completar a cierto tiempo y con cierta cantidad de vidas.

Para la creación de este juego se hizo uso de las siguientes bibliotecas. **Tkinter:** para la creación de la pantalla principal con Tk, así como las subventanas con toplevel. **Os:** específicamente path, para poder acceder al sistema y cagar archivos para las animaciones. **Time:** se usa para crear pausas que permitan la reacción del sistema mediante sleep, y los diferentes tiempos de ataque de los jefes así como el cronómetro se crearon mediante root.after. **Random:** se usó para los diferentes algoritmos que requerían de algún ataque random, como el del nivel 1 y el último nivel. **Pygame 2.0.1:** específicamente el módulo mixer para utilizar la función sound y music y trabajar los efectos de sonido del juego.

Así mismo, se crearon módulos que permitieron la jugabilidad. A continuación, se detallan los módulos principales y se listan sus submódulos. **Vnivel1:** módulo donde se desarrolla el nivel 1del juego, usa el módulo **SWAP** para realizar el ataque random. **Vnivel2:** módulo donde se desarrolla el nivel 2del juego, usa el módulo **shooting** para realizar el ataque de tres láseres por segundo, así como la función **laser\_attack** para verificar colisión entre laser-jugador, se apoya en función **collision**. **Vnivel3:** módulo donde se desarrolla el nivel 3del juego, usa los módulos del nivel 1 y 2 modificados. **info\_adicional:** función que presenta el about. **mejores\_puntajes:** función que presenta los mejores puntajes y los carga a través de la función **tablero**.

Por otro lado, los siguientes módulos son parte de la pantalla principal y son usados por los demás módulos de juego: **guardar\_archivo, leer\_archivo, cargar\_imagen, cargarMP3, reprod\_FX, detener\_cancion, reprod\_cancion, reproducir\_salida, limitador** (limita el texto del entry), **Vnivel1Check, Vnivel2Check, Vnivel3Check** (estas últimas tres verifican que el nombre no esté vacío antes de continuar con el nivel respectivo), **checkPoints, mensajePuntos, cronom** (sirve de timer en el juego), **reset\_time, pause\_time**.

Además, los módulos que se repiten en los diferentes niveles son: **end, back, close** (las últimas tres sirven de cierre en distintas situaciones), **checkBoss** (verifica que tanto el jefe como el jugador sigan con vida), **boss\_move** (generan el movimiento del jefe, cambia para los tres niveles, pero su función básica es esta), **collision** (verifica si hay colisión entre dos objetos, dadas las coordenadas, ancho y largo), **laser\_move** (movimiento del láser del jugador), **smooth\_move** (movimiento suave por parte del jugador), **create\_laser** (crea un láser cada vez que sea llamada), **move\_ship** (contiene las respuestas a cada presión de tecla disponible en el juego), **stop\_shoot** (controla las banderas o flags del juego).

Por último, la autopercepción en cuanto a la finalización del proyecto es de un 100% dado a que cumple todas las funciones.

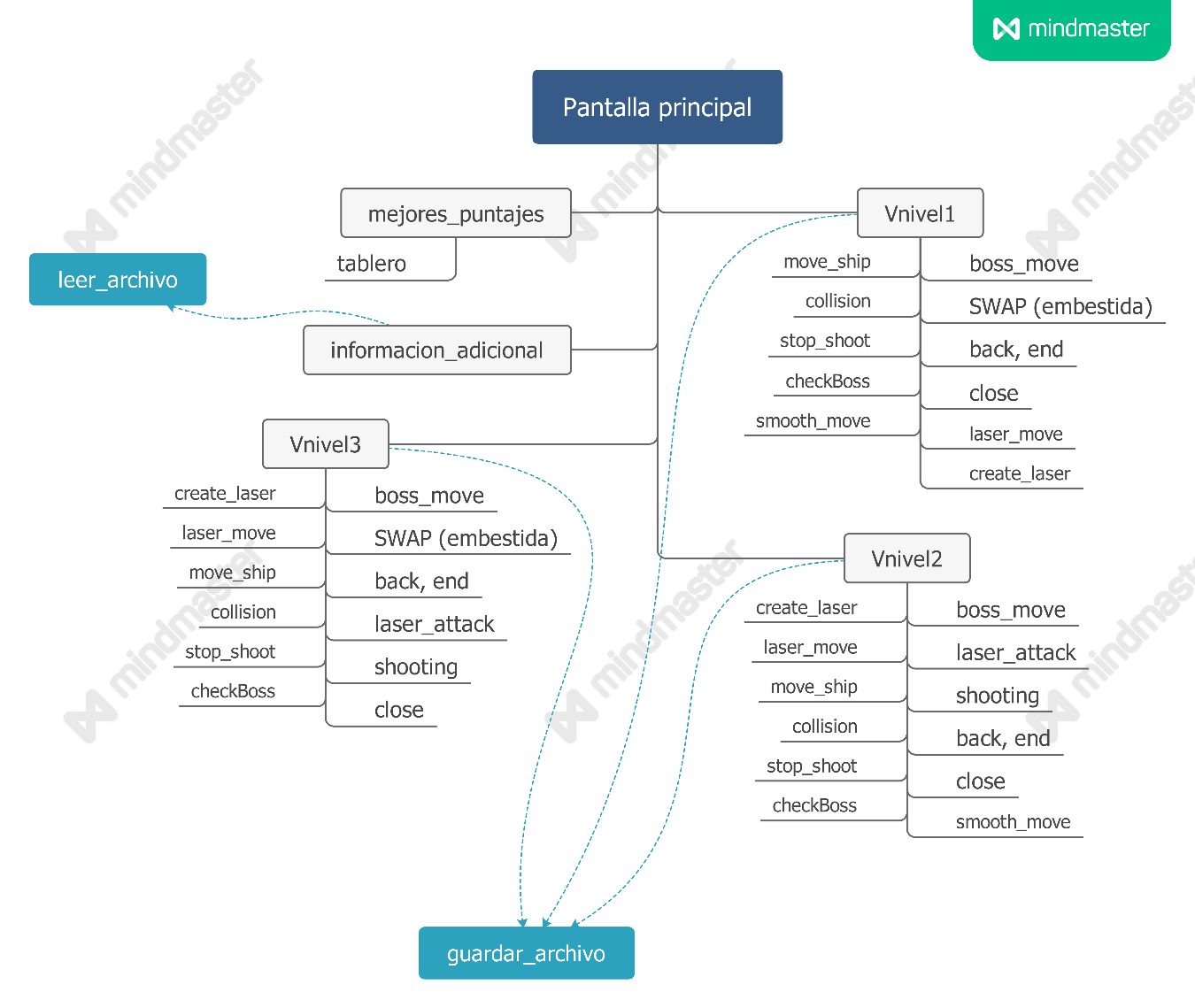
1. **Conclusiones**

En este apartado, se listan las diferentes conclusiones que se obtuvieron de este proyecto:

1. La utilización de flags o banderas permitió el detener funciones o reiniciarlas, lo cual fue de gran ayuda cuando había varias condiciones de inicialización de un ciclo por recursividad.
2. Se debe poner un límite en la recursividad del movimiento del láser, así cuando salga de pantalla no se siguen moviendo y no vuelven lento el juego.
3. Es importante llamar a las variables nonlocal y global especialmente cuando estas van a ser modificadas dentro de la función, sino se consideran como variables locales.
4. Para la modificación de imágenes se puede hacer uso de Photopea, herramienta online gratuita que se asemeja a Photoshop.
5. Si la condición de inicio de una función es un keypress, el evento va a enviar repetidas señales de que la tecla está siendo presionada y puede repercutir en la repetición múltiple de un ciclo recursivo.
6. La herramienta color picker de Google sirve para elegir los colores que se necesiten en el juego.
7. Al momento de trabajar directamente con un archivo, se pueden tener problemas si la última línea está en blanco, pues no responderá cortes por posiciones de caracteres.
8. Al crear un objeto se debe tener cuidado en posicionarlo en el canvas del nivel que se está trabajando, de otro modo el objeto nunca aparecerá.
9. Para trabajar colisiones, la utilización de rectángulos que bordeen los objetos resulta de gran utilidad.
10. Para la utilización de pygame.mixer se requiere inicializar mediante un .\_\_init\_\_
11. Si se utilizan variables globales para puntos, vida y vida jefe, estos se deben reiniciar.
12. Cuando True y False no son suficientes estados para una variable Flag, se puede utilizar algún otro valor que lo cambie de tipo o None.
13. La utilización de muchos assets puede crear confusión con los nombres al momento de programar, por ello estos deben tener nombres significativos.
14. El método after permite crear un delay sin “dormir” al programa como sí lo hace sleep.
15. Al momento de que cada láser verifica colisión con nave se repite la colisión tantas veces como el ciclo del movimiento del láser se dé estando en contacto ambos objetos.
16. La función de juego terminado debe llamarse una vez y bloquearse para que no haya múltiples mensajes de finalización.
17. Para las funciones de movimiento, un tiempo menor de repetición con movimientos más pequeños genera una animación más suave.
18. Se puede implementar una barra de vida de longitud x mediante la relación lineal matemática entre la variable de vida del jefe y la longitud x.
19. La utilización de print sirve para verificar si el programa está pasando a través de ciertos condicionales.
20. Si no se limita el texto de entrada de nombre, este puede salirse de la pantalla al momento de jugar un nivel.
21. Si se quiere disminuir la dificultad del juego para probar los módulos, se puede disminuir el tiempo de repetición (root.after).
22. Si no se controla el inicio de un ciclo de ataque, el jugador puede ser atacada apenas empieza el juego y sin dar tiempo de reacción.
23. Si se programa la aparición de un objeto pequeño que se desea ver, debajo de otro más grande, este no aparecerá.
24. Al trabajar con archivo se debe probar que las funciones de carga funcionen existiendo el archivo o sin existir.
25. Si no se documenta mientras se programa, se puede perder tiempo luego tratando de entender lo que está detrás de un proceso programado antes.
26. El llamado a módulos recursivos que siempre están corriendo, deben hacerse al final de cada módulo principal.
27. La velocidad de las balas del jefe tiene que ser lo suficientemente rápida para que el juego no sea tan fácil, pero suficientemente lenta para que el jugador pueda atacar.
28. La reproducción de muchos efectos de sonido al mismo tiempo puede incurrir en que algunos no se escuchen por la sobrecarga.
29. Para el disparo de balas de los jefes, debe haber un ciclo con delay dentro del ciclo con delay de dos segundos.
30. Se deben actualizar las variables que contienen las coordenadas de los objetos antes de realizar una acción y no después.
31. **Recomendaciones**

A continuación, se presenta una serie de recomendaciones que facilitaran el proceso de elaboración del código del juego:

1. Usar tantas banderas como sean necesarias para que el juego se detenga cuando tenga que y cumpla con todas las condiciones.
2. Usar un flag para evitar que un keypress repita muchas veces una instrucción o ciclo.
3. Usar un try except en caso de tener líneas en blanco en un archivo que se desea tratar.
4. Programar en el orden en que se desea que aparezcan los objetos.
5. La utilización de variables locales para puntajes, vida y jefes facilita el proceso de reiniciarlos al cambiar de nivel.
6. Crear assets con nombres que se puedan identificar dentro de una carpeta en medio de otros assets.
7. Usar el método after para casos como los ataques de jefes, de este modo no se detiene todo el programa.
8. Usar banderas para controlar las repeticiones de colisión de un mismo láser.
9. Usar bandera para bloquear la función de juego finalizado, de este modo, si se gana luego de un disparo, el impacto de un segundo disparo no llamará de nuevo a la función
10. La implementación de una barra de vida ayudaría a que el jugador esté más al tanto de la situación del juego.
11. Para probar la funcionalidad de los módulos de puntaje se pueden determinar los valores de vida y tiempo en cero.
12. Se recomienda agregar un delay de unos cuantos segundos para que inicie el ciclo de ataque, de este modo no se sorprende al jugador.
13. Tener presente el canvas en que se está trabajando para posicionar objetos en este.
14. Usar un try except para determinar con qué tipo de argumento se abre un archivo, así si este no existe se crea.
15. Se recomienda documentar por líneas tanto como se pueda y mientras se programa.
16. Se recomienda primero crear todas las funciones y luego llamarlas, en lugar de crear y llamar inmediatamente.
17. Se debe determinar una velocidad de ataque que entretenga al jugador, que no lo aburra ni que tampoco sea imposible.
18. Para las ráfagas se recomienda utilizar ciclos dentro del ciclo con el método root.after y no con sleep.
19. Cada vez que se va a utilizar una coordenada actualizarla mediante root.coords(object).
20. Usar anchor center en lugar de posiciones cardinales ayuda con el place de los objetos.
21. Se recomienda usar las funciones que se repiten como globales.
22. Se recomienda destruir subventanas en vez de ocultarlas. De este modo las variables se reinician solas y sólo hay máximo dos procesos al mismo tiempo.
23. Se recomienda usar en la pantalla principal, botones de colores similares al fondo de pantalla.
24. Se recomienda usar el movimiento de la nave del jugador, por recursión como se desarrolla en este proyecto.
25. Se recomienda que la ventana no sea maximizable, de este modo los objetos no se desordenan.
26. Hay condiciones que deben de ponerse en if por separados uno dentro de otro, en lugar de usar and, en el primer caso se tienden a omitir o permitir ciertas condiciones que no deben ser permitidas.
27. Se recomienda usar el formato de texto para evitar la concatenación de strings que puede ser tediosa.
28. Si se va a utilizar un solo archivo py se recomienda crear divisiones internas mediante comentarios que permitan la identificación ágil de los módulos.
29. Se recomienda detener el cronómetro antes de realizar el conteo de puntos, luego resetearlo.
30. Se recomienda separar el código en módulos que agrupen la mayor cantidad de funcionalidades, por ejemplo, por niveles y no por acciones.
31. **Análisis de Resultados**
    1. **Diagrama de módulos**



* 1. **Plan de pruebas**

Para la función de **check\_boss**, se probó muriendo, pero antes de morir dejar ir algunos laser que golpearan la nave, de este modo se aseguraba que la función check\_boss no se llamara después de muerto el jugador. Además, se bajó la velocidad de la nave para ganar más rápido y se comprobó que no se llamaba más de una vez.

En el caso de la función **check\_points** se corrió con un puntaje de prueba 10, con los minutos en 0, segundos en 0. De este modo se comprobó que se daba el bonus por tiempo. Así mismo, se probó ganando el juego con la disminución de vida del jugador desactivada (código se convirtió en comentario), de este modo se comprobó que se daba el bonus al no perder vidas.

En el caso de **laser\_move** se comprobó que dejara de mover el láser luego de cierta posición, mediante la colocación de un límite mayor a -100, esto permitió ver como el láser se detenía y aseguraba que se detuviera fuera de pantalla al poner ese límite en -100.

La función **shooting** se probó igual que la función anterior, pero el límite era menor a 900 para poderlo observar.

Las funciones **boss\_move** se probaron con valores de vida de jefe y nave igual a cero, lo que aseguró que se detuvieran al perder. Para el caso del nivel 3 se generó un print luego del número random generado para una eventual segunda embestida.

La función **SWAP** se probó mediante la colocación de un print del número random que se generaba.

La función **info\_adicional** se probó con texto estático antes de mostrar el puntaje y los nombres.

En el caso de Vnivel1, Vnivel2 y Vnive3 se probaron con valores de vida de nave y jefe igual a 0, y tiempo estático, esto para probar que el movimiento de la nave, láser y jefes, funcionara antes de continuar.

1. **Bitácora**

**Fecha:** 7/05/2020

**Objetivos alcanzados:**

* Se creó un esquema de trabajo, se programó la ventana principal, fondos, botones sin funciones a los diferentes niveles y el entry para el nombre del jugador.

**Fecha:** 10/05/2020

**Objetivos alcanzados:**

* Se crea el about con la información pedida y se añade el comando al botón.
* Se programa la subventana de puntaje, de modo que esta muestre como texto formateado los elementos de la lista de puntos y la lista de nombres. Además, se añade el comando al botón
* Se crean las subventanas del nivel 1, 2 y 3, vacías. Se cargan los fondos y la música de fondo. Todas con función back, close y protocolo de salida. A todas se les pone el botón de back.
* Se programa el movimiento de la nave en del nivel 1, se intentó con movimiento normal, pero luego se creó una función que permitiera un movimiento más suave, mediante dos banderas.
* Se programó la carga de láser y movimiento.
* Se programó el láser de la nave del jugador y su tecla respectiva.
* Se añaden los comandos a los botones.

**Fecha:** 14/05/2020

**Objetivos alcanzados:**

* Se carga el boss para cada nivel, estático.
* Se crean las variables de puntos y vidas. Así como los labels.
* Se crea el label de tiempo, estático

**Fecha:** 15/05/2020

**Objetivos alcanzados:**

* Se programó el cronómetro
* Se programó el movimiento del boss del nivel 1.
* Se programó la embestida del nivel 1.
* Se programó una barra de vida.
* Se programó la función que encontraba colisión.
* Se programaron colisiones entre nave y boss, laser y boss.
* Se programaron las funciones de modificación de puntaje y de modificado de puntaje en caso de ser nuevo record.
* Se programaron los bonus.
* Se programó función que daba mensaje final.

**Fecha:** 16/05/2020

**Objetivos alcanzados:**

* Se crea una función que revisa si el jefe y/o jugador siguen con vida.
* Se crea mensaje de finalización de juego.

**Fecha:** 17/05/2020

**Objetivos alcanzados:**

* Se migran las funciones que se repiten en los 3 niveles, al nivel 2 y 3. Así como variables y banderas.

**Fecha:** 20/05/2020

**Objetivos alcanzados:**

* Se programan los láseres del jefe del nivel 2.
* Se programa la colisión entre láser nave.
* Se modifica la función de movimiento, se elimina la embestida y se cambia de movimiento LR a random.
* Se crea una función que mueve los laser y otra que los dispara.
* Se migran las funciones de disparo y movimiento de láser al nivel 3 para ser utilizados.
* Se modifica la función de movimiento del nivel 3 para que después de una embestida haga otra o se teletransporte.
* Se modifica la función de embestida, ya no con números random múltiplos de tres, sino cada seis segundos.
* Se agrega un delay en la iniciación de los ciclos de disparo y ataque de los niveles, para que no ataquen de inmediato al iniciar el juego.
* Se agregan todos los efectos de sonido.
* Se modifican las colisiones de todos los niveles, para que cambien de color al colisionar.

**Fecha:** 21/05/2020

**Objetivos alcanzados:**

* Se crean las funciones de guardar y leer archivos.

**Fecha:** 23/05/2020

**Objetivos alcanzados:**

* Se realizan las pruebas finales para verificación de banderas y que todas las condiciones se cumplan.

**Fecha:** 24/05/2020

**Objetivos alcanzados:**

* Se da formato a la bitácora, plan de pruebas, conclusiones y recomendaciones.
* Se redacta la introducción.
* Se crea el diagrama de módulos.

1. **Fuentes consultadas**

*Cambiar coordenadas x,y de un item tkinter*. (2019, May 22). Stack Overflow En Español. <https://es.stackoverflow.com/questions/265683/cambiar-coordenadas-x-y-de-un-item-tkinter>

*Colisiones*. (n.d.). Ants. <https://ants.inf.um.es/staff/jlaguna/tp/tutoriales/colisiones/index.html>

Descottes, J. (n.d.). *Piskel - Free online sprite editor*. Piskel. <https://www.piskelapp.com/>

GeeksforGeeks. (2021, January 27). *Read a file line by line in Python*. <https://www.geeksforgeeks.org/read-a-file-line-by-line-in-python/>

Morales, J. (29, Marzo, 2021) *Taller de Tkinter*.

*Photopea | Online Photo Editor*. (n.d.). Photopea. <https://www.photopea.com/>

*Python Random randint() Method*. (n.d.). W3School. <https://www.w3schools.com/python/ref_random_randint.asp>