

Tecnológico de Costa Rica

Taller de programación

CE - 1102

Documentación escrita

Del primer proyecto

Profesor:

Milton Villegas Lemus

Autor:

Dylan Garbanzo Fallas

Carnet: 2021057775

I Semestre Año 2021

Tabla de contenidos

[Introducción 3](#_Toc72772583)

[Conclusiones 4](#_Toc72772584)

[Recomendaciones 6](#_Toc72772585)

[Diagrama de módulos: 8](#_Toc72772586)

[Plan de Pruebas de los módulos: 8](#_Toc72772587)

[Bitácora: 9](#_Toc72772588)

[Literatura o fuentes bibliográficas: 10](#_Toc72772589)

# Introducción

El proyecto consistió en la realización de un videojuego estilo “Shoot ‘em up”, similar al famoso juego Space Invaders. El juego consiste en 3 niveles, cada uno con un jefe con diferentes mecánicas, dónde el jugador tiene que lograr destruir los jefes de cada pantalla.

Las bibliotecas utilizadas en las programaciones del juego son: **Tkinter**: es la bilbioteca utilizada para todo lo relacionado con la parte gráfica del videojuego, ya sea la pantalla principal, las pantallas de los jefes, y las animaciones de movimiento de cada nave y su respectivo proyectil. **PIL:** esta biblioteca permite cargar cualquier tipo de imagen, es necesaria ya que Tkinter no es compatible con algunos tipos de archivos de imágenes, y por ende no las carga en la interfaz gráfica. **Os:** su función es acceder a carpetas externas para obtener los recursos necesarios tanto gráficamente como auditivamente. **Threading:** esta biblioteca nos permite realizar operaciones paralelas en algunas partes del videojuego. **Time:** mediante esta biblioteca se implementó el time.sleep(), para utilizarlo cuando el videojuego lo requería. **VLC**: esta biblioteca es la que se encarga de todo lo relacionado con la parte auditiva del videojuego. **RANDOM:** esta biblioteca nos permite generar números aleatorios necesarios para ciertas mecánicas dentro del videojuego.

El videojuego se realizó mediante el uso de módulos, los módulos principales del videojuego son: **pantalla\_principal()**: este módulo es el que nos permite navegar a cualquier pantalla dentro del videojuego, ya sea alguna pantalla específica del jefe, la pantalla de ranking, o la pantalla de información complementaria. **Pantalla\_boss1():**  en este se encuentra el primer jefe. **Pantalla\_boss2():**  en este se encuentra el segundo jefe. **Pantalla\_boss3():**  en este se encuentra el tercer jefe. **Pantalla\_ranking():** en este se encuentra el top 5 mejores puntaje obtenidos, el cual se actualiza cada vez que termina una partida. **Pantalla\_about():** en este se encuentra toda la información complementaria. El resto de módulos están divididos según su funcionalidad, ya sea para cargar y reproducir sonido o imágenes, o mostrar los resultados.

Se llegó a un grado de completitud de entre 87% y 92%.

# Conclusiones

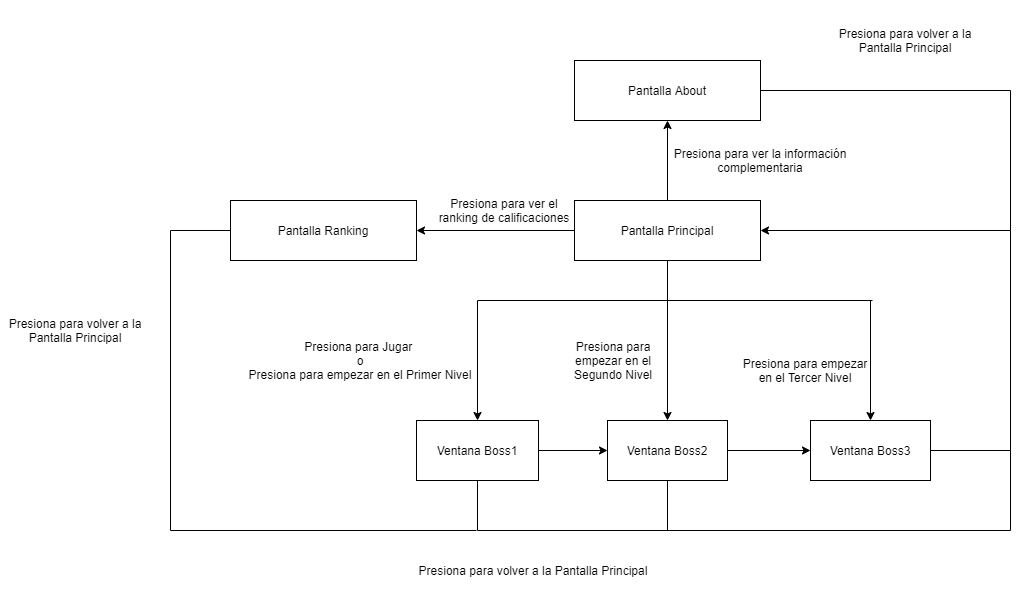
* Tkinter no es buena cuando queremos cargar todo tipo de imágenes, ya que algunos formatos no son compatibles con la biblioteca.
* Tkinter no es muy buena para realizar movimientos de animaciones continuas de objetos, ya que puede llegar a sobrecargar el programa.
* Los Canvas y los Labels son excelentes herramientas para hacer interfases gráficas simples. Los Canvas son buenos para imágenes y los Labels para texto (aunque también pueden cargar imágenes).
* Utilizar canvas como ventanas principales, en vez de crear ventanas secundarias, va a sobrecargar los recursos del sistema.
* La función. coords() no es muy buena cuando se quiere trabajar con coordenadas, puede llegar a ser confusa.
* Es importante tener en cuenta como trabajan los ejes de simetría en Tkinter cuando se requiere realizar algún tipo de animación, ya que no son iguales a los ejes de simetría que se ve en matemáticas o física.
* La herramienta de YouTube es bastante útil cuando no se logra el funcionamiento del programa.
* Agregar varias imágenes y además utilizar hilos de fondo puede llegar a sobrecargar los recursos utilizados y llevar al mal funcionamiento del programa.
* Trabajar con HexColors permitirá tener una variedad más amplia de tonalidades y así tener diferentes colores en los trabajos.
* Los archivos .txt externos sirven bastante cuando se busca tener una base de datos externa que permita estar cargando, guardando y sobrescribiendo los datos.
* Tener datos de diferentes extensiones dentro del .txt dificulta la manipulación de datos.
* Tener múltiples módulos de movimientos, puede llegar a ser confuso o puede llegar a dificultar el proceso de animación de imágenes.
* Es necesario utilizar .seek() cuando se trabaja con documentos externos y se utiliza recursividad, ya que .readline() va a hacer que siempre comience en la línea 1.
* Tratar de realizar comparaciones dentro de un archivo .txt va a ser demasiado confuso.
* La función path facilita la manipulación de imágenes externas o sonidos y la implementación de estos dentro del Tkinter.
* La función time es muy útil cuando se quiere realizar intervalos de tiempo entre animaciones, pero si se usa fuera de un Thread, va a afectar toda la ventana, y no solamente el elemento deseado.
* La función VLC facilita la reproducción de sonidos dentro de Tkinter, ya sea simultaneo o selectivo de sonidos.
* Utilizar múltiples FLAGS permite detener diferentes Threads en diferentes intervalos de tiempo.
* Tener múltiples funciones de imágenes permite tener un mayor orden en las carpetas externas del programa
* Tener múltiples funciones de música permite ya sea la reproducción simultanea o selectiva de música o efectos especiales.
* Dejar espacios entre módulos facilita la navegación por el programa.
* No es necesario utilizar Threads cuando se busca el movimiento repetitivo de algún Canva, es posible realizarlo con recursividad.
* Una buena documentación dentro del programa, facilita la navegación dentro del mismo.
* Utilizar un ambiente de programación diferente al IDLE, va a facilitar la realización de programas extensos.
* Utilizar lenguaje explicito en la creación de variables facilita la manipulación de elementos dentro de la ventana.
* Se requiere invertir el máximo tiempo posible en este estilo de programaciones
* Tener un conocimiento previo del posible comportamiento de los datos, permite avanzar con mayor velocidad.
* Ocultar líneas de código que ya funcionan facilita el movimiento a lo largo del programa, ya que se tendría que mover una menor cantidad de espacios.
* Utilizar .bind permite la detección de alguna tecla, ya sea al presionarla o al soltarla.
* Utilizar return cuando se quiere ir a otra pantalla, no solamente llamar la función.

# Recomendaciones

* Para cargar cualquier formato de imagen es mejor utilizar ImageTk de la biblioteca PIL ya que permitirá cargar imágenes de cualquier formato sin problemas.
* No utilizar los Labels para cargar imágenes ya que puede tirar errores o dificultar el proceso.
* Utilizar la función .bbox() facilita el trabajo con coordenadas o mover imágenes en un plano cartesiano.
* Dibujar el plano cartesiano que utiliza Tkinter, facilitará el proceso de mover imágenes a lo largo de la ventana, en especial si no se tiene mucha experiencia con la biblioteca.
* Hacer una investigación a fondo del funcionamiento de la biblioteca va a facilitar después el trabajo con la misma.
* Utilizar un diagrama de flujo antes de programar va a impedir el tener que escribir código innecesario.
* Obtener las coordenadas de los bordes de la ventana al inicio va a disminuirle el tiempo adivinando coordenadas cuando se implementan objetos al Canvas.
* Utilizar Google tanto como fuente de información para ayudar en el programa, o para conseguir los códigos HexColors para los elementos visuales.
* Utilizar HexColors para tener colores más específicos y con diferentes tonalidades.
* Hacer un módulo de movimiento base, el cual sea fácil de editar y adaptar, va a permitir ahorrar tiempo cuando se debe realizar varias animaciones continuas.
* Hacer un módulo base de ventanas, para ahorrar tiempo cuando se deba implementar una nueva ventana.
* Utilizar un módulo global el cual haga una misma función pero en diferentes ventanas, va a ahorrar líneas de códigos innecesarias.
* Tratar que los datos dentro del archivo .txt tengan la misma cantidad de dígitos, ya que va a facilitar la manipulación de estos cuando no se usan bibliotecas externas de manipulación de datos.
* Sacar los datos de un archivo.txt a una lista vacía dentro del programa, ya que así se va a facilitar la manipulación de estos datos, y luego solamente reescribirlos al documento.
* Utilizar un equipo con los componentes adecuados facilita o permite un correcto testeo del programa.
* Utilizar Visual Estudio Code o Pycharm va a facilitar programar códigos extensos, ya que sus extensa personalización nos permite evitar cansar nuestra vista más rápido, además de las herramientas preconstruidas que tiene que ayuda con el código.
* Dedicar espacios de relajación alejados a la computadora ayuda relajarse y posiblemente a idear la solución al programa.
* Ocultar líneas de código que ya funcionan así no tiene que moverse mucho para llegar a líneas de código que estén en números más altos
* Dedicarle un espacio de tiempo para realizar una buena documentación interna, para facilitar la ubicación del usuario dentro del programa.
* Utilizar threads va a facilitar la ejecución de módulos de forma paralela, si no se utiliza, puede llegar a consumir demasiados recursos, llevando así a errores en el programa.
* Utilizar módulos de lectura de archivos .txt para entender la navegación y lograr realizar una lectura y sobreescritura de datos en este tipo de archivos.
* Tener diferentes módulos para cargar imágenes, ya que así va a tener un mayor orden dentro de su programa.
* Probar continuamente el correcto funcionamiento de los módulos, ya que así, si surge un problema, lograr encontrarlo lo antes posible.
* Ser claro y directo con los nombres de las variables y de las funciones, esto facilitará su manipulación
* Dejar espacios entre módulos, para así poder ubicarse con mayor facilidad
* Si se van a utilizar pocas animaciones, mejor utilizar recursividad y no hilos, ya que los hilos ocupan una función extra para eliminar el hilo, en cambios la recursividad no
* Discutir errores con compañeros, así podrán solucionar el error y seguir adelante con el código.
* Siempre utilizar return() para así poder eliminar la pantalla anterior, sino se va a ejecutar en segundo plano consumiendo recursos
* Distribuir el tiempo lo mejor posible, para poder cumplir con las fechas establecidas.
* Utilizar contadores externos en caso de que time.sleep() congele todo el canva, aunque se tenga dentro de un hilo

Análisis de resultados:

## Diagrama de módulos:



## Plan de Pruebas de los módulos:

La ventana principal se probó ejecutando el programa, al ser la primera ventana, no se requería acceder a esta desde ningún botón externo.

Para el resto de módulos principales (ventana ranking, about, boss1, boss2, boss3), ya que son ventanas, se probaron llamando la función que contiene las ventanas, utilizando botones presentes en la ventana principal.

El resto de los detalles ya sea animaciones, sonidos, entre otros, se probaron ejecutando los canvas que contienen los elementos a probar, accediento a estos mediante botones situados en la ventana principal.

El sistema de ranking se probó en un programa apartado del principal, utilizando varios números para que quedaran en diferentes posiciones.

# Bitácora:

|  |  |
| --- | --- |
| Fecha: | Bitácora |
| 4/5/21 | * Se consiguieron las imágenes necesarios para la realización del proyecto. * Empecé a realizar un diagrama de flujo para darme una idea del funcionamiento del juego. |
| 5/5/21 | * Realizar la pantalla principal, ranking y pantalla del primer jefe (sin animaciones) * Buscar música |
| 6/5/21 | * Se trabajó en el movimiento de las naves. |
| 7/5/21 | * Se logró el movimiento de la nave del personaje, se estableció los parámetros de movimiento de cada nave |
| 8/5/21 | * Avance en el movimiento de la nave jefe 1, pero aún no se logra su completo movimiento |
| 9/5/21 | * Se completa movimiento del jefe y del personaje al 100% |
| 10/5/21 | * Se agregan iconos y porcentaje de vida de ambas naves * Se limita el movimiento de la nave del personaje para que no salga de la pantalla * Se trabajó en el daño realizado si la nave impacta con el jugador o si falla * Se implementó la vida del jefe y del jugador |
| 11/5/21 | * Se trabajó en la pantalla del segundo jefe y su movimiento * Se completó el movimiento del segundo jefe y de la nave en la segunda pantalla |
| 12/5/21 | * Se realizó la pantalla about * Se completó el movimiento del tercer jefe |
| 15/5/21 | * Se completó el sistema de ranking, y puntaje |
| 16/5/21 | * Se completó el sistema de colisiones entre naves. |
| 19/5/21 | * Se mejoró visualmente los 3 niveles * Se implementó el registro del nombre del personaje * Se implementó una ventana extra la cual da los resultados de la partida (en caso de que el jugador quedara en el Top 5) * Se eliminaron los iconos |
| 20/5/21 | * Se terminó la pantalla de ranking, se trabajó nueva mente en el disparo de la nave (no se logró su funcionamiento) |
| 21/5/21 | * Se completó el sistema de disparo de la nave del jugador * Se trabajó en el sistema de ranking |
| 22/5/21 | * Se completó los disparos de los bosses * Se implementó el cronometro * Se arregló el sistema de puntajes * Se finalizó 90% del proyecto. |
| 23/5/21 | * Se arregló detalles * Se establecieron los detalles que faltaron por terminar * Se finalizó la parte de documentación del proyecto |

# Literatura o fuentes bibliográficas:

Codemy.com. (2020, May 14). *Move Canvas Shapes With Arrow Keys—Python Tkinter GUI Tutorial #69*. <https://www.youtube.com/watch?v=xifcE6xvnyg>

TokyoEdtech. (2020). *Python Game Coding: Introduction to Collision Detection*. <https://www.youtube.com/watch?v=bFURdsszto0>

Corey Schafer. (2016). *Python Tutorial: File Objects - Reading and Writing to Files*. <https://www.youtube.com/watch?v=Uh2ebFW8OYM&t=1081s>

*8.6. Methods on Canvas widgets*. (n.d.). Retrieved May 16, 2021, from <https://anzeljg.github.io/rin2/book2/2405/docs/tkinter/canvas-methods.html>

Magno Efren. (2021, March 31). *Cronometro en Python usando Tkinter—GUI*. <https://www.youtube.com/watch?v=26D17QgpIBE&t=320s>