**Instituto Tecnológico de Costa Rica**

**Ingeniería en Computadores**

**Taller de Programación**

**Grupo 2**

**Proyecto I | Documentación**

**Profesor: Milton Villegas Lemus**

**Ignacio Vargas Campos**

**2019053776**

**I Semestre, 2020**Introducción

Este proyecto consiste en recrear un clon del juego de Nitrome “Bullethead”, esto se tiene que realizar en Python apoyándose principalmente en la librería de Tkinter. Los elementos del juego que debía de incluir el clon realizado tenía que ser los componentes básicos el “gameplay” del juego original: controlar un personaje que se mueve hacia los lados, salta y dispara, 3 de los enemigos básicos del juego, el objetivo de tener que sobrevivir defendiéndose de dichos enemigos disparándoles, 3 vidas que se pierden al ser golpeado por los enemigos o sus disparos y solo 3 niveles de dificultad. También un menú del que se puede navegar al juego, a una pantalla de ayuda, una pantalla de puntajes o una pantalla de créditos. Además, debía incluir música y sonidos de acuerdo con la temática con la que uno replicara el juego.

En mi caso yo tomé la decisión para este proyecto de crear un clon lo más parecido al juego original posible, para acercarme lo más que pudiera utilicé todo el arte del juego, la música original del juego y traté de que las mecánicas de cómo todo funciona fuera lo más parecido a como son en el juego original.

Conclusiones

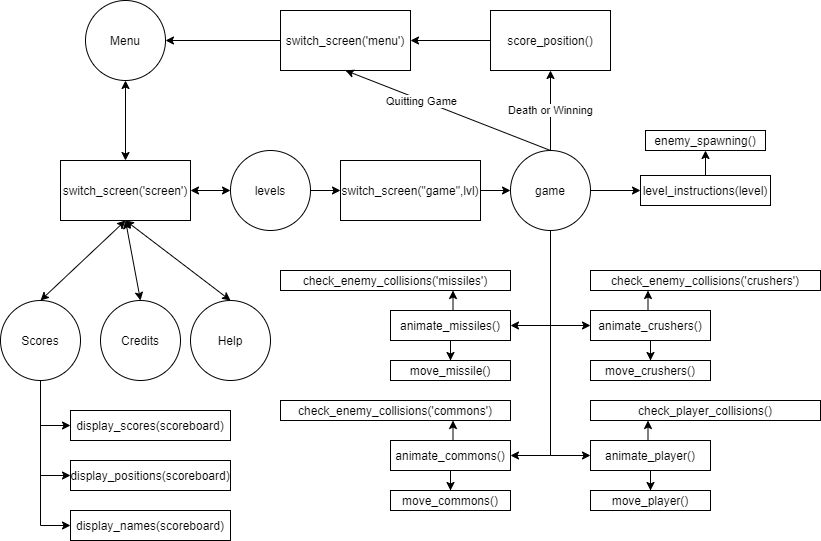
1. Es muy útil tener nombres descriptivos para las funciones y variables aunque sean largos.
2. Es buena idea crear una estructura para organizar el código de manera que todo sea fácil de encontrar
3. Para revisar si dos objetos están colisionando uno revisa primero si en el eje x el extremo derecho del primer objeto está en una posición mayor al extremo izquierdo del segundo objeto y luego que el extremo izquierdo del primer objeto está en una posición menor que el extremo derecho del segundo objeto. Luego se hace lo mismo pero verticalmente en el eje y.
4. Al menos en el contexto de cómo Tkinter detecta que se presiona una tecla, se pueden utilizar los eventos <KeyPress> y <KeyRelease> para detectar cuando se presiona una tecla y se mantiene presionada hasta que se suelte. [1]
5. Se puede utilizar un diccionario como un switch statement para Python, usando las llaves del diccionario para llamar sus valores y así evitarse una cadena de if statements. [3]
6. Una manera útil de diseñar un nivel es con la técnica previamente descrita tener una variable que cuenta los segundos y un diccionario que tiene índices de ‘segundo: instrucciones’, ósea que retorna las instrucciones de qué debería de suceder en cada segundo específico.
7. Para extraer sprites del juego original es útil usar Paint para cortar los screenshots que uno les tome y luego exportarlo a piskel para limpiarles el fondo y hacerlos pngs. [7]
8. Parte del proceso de extraer assets del juego original requiere tomar screenshots del juego y si uno le hace zoom a la ventana del juego en el browser entonces se pueden obtener assets con mejor calidad y resolución.
9. Utilizar funciones anónimas lambda es útil cuando se quiere asociar una llamada a una función con un botón, un after, un trace o algún otro objeto que haga llamada a funciones y se le ocupen pasar parámetros a la función.
10. Es bueno revisar cuando un enemigo, bala u objeto similar se sale de la pantalla para eliminarlo y que pare de gastar recursos.
11. Para animar el movimiento del jugador y de los enemigos uno lo que hace es moverlos una distancia corta muchas veces por segundo y parece que se mueven fluidamente.
12. Para la animación de los sprites del jugador o de los enemigos o de cualquier cosa uno lo que hace es cambiar la imagen del sprite muchas veces por segundo por la siguiente imagen del spritesheet.
13. Usando archivos .txt es posible escribir en ellos y guardar información que se mantiene guardada aún cuando uno cierra y vuelve a abrir el programa.
14. Cuando se va a leer un archivo txt se puede utilizar un try and except para manejar el caso en el que no se encuentra el archivo y en el except crear el archivo que se intentó leer con valores default.
15. Es posible escribir utilizando pngs de números y letras creando funciones que tomen un texto y dibujen las imágenes correspondientes a las letras en la pantalla.
16. Para poder escribir con imágenes se tiene que restringir el texto que escribe a solo los caracteres para los que uno tenga imágenes.
17. Es útil tener un canvas individual para cada pantalla e irlos intercambiando cuando uno se cambia de pantalla.
18. Cuando se cierra un canvas se pueden eliminar todos los objetos dibujados en él utilizando el método delete() y pasándole de parámetro “all”, por ejemplo: canvas.delete(‘all’). En vez de tener que borrar cada objeto individualmente. Aunque esto no borra los botones botones y para estos hay que utilizar el método destroy en cada uno individualmente.
19. El método de after se puede utilizar para crear recursión infinita con un tiempo definido entre cada repetición haciendo que una función se llame a sí misma utilizando este método. [2]
20. Es posible limitar de diferentes formas el texto que se ingresa en un entry widget usando una variable a la que se le asigne StringVar(), entonces uno le puede poner un trace que llame a una función que normalice el string a las condiciones que uno quiere (cortarle el tamaño, remover caracteres especiales, remover espacios o cualquier restricción que uno desee) y usando el método set del StringVar para reasignarle el string normalizado.
21. La librería de unidecode se puede utilizar para normalizar un string de caracteres especiales a sus caracteres sin acentos ni símbolos especiales [5]
22. Se pueden crear “pseudo-clases” usando listas y definiendo que un “pseudo-objeto” objeto agregado a esa lista va a seguir un formato, por ejemplo para los enemigos comunes todo se representaban con una lista que guardaba el objeto de la imagen e información como la posición, la velocidad con la que se está moviendo y otros datos.
23. Lo útil de tener todos los “pseudo-objetos” del mismo tipo en una lista es que para animar cada enemigo de un tipo se puede recorrer la lista fácilmente. Esto también aplica si se utilizan objetos de clases reales.
24. Es útil tener funciones que se encarguen de crear estos “pseudo-objetos” con valores iniciales en sus parámetros y los agreguen a su lista global respectiva.
25. La herramienta JFXR es una página web en la que se pueden crear sonidos de juego que suenen como tipo de arcade. [6]
26. Se puede simular un movimiento de parábola cuando el jugador salta acelerando su velocidad por un valor constante cada cierto tiempo hasta que vuelva a tocar el suelo, parecido a cómo funciona la gravedad en el mundo real.
27. Si se quiere detectar una tecla después de estripar un botón, se puede tener una variable global booleana que cambie de valor cuando se estripa el botón y que la función que se llama cuando se presione una tecla revise el estado de esta variable y solo actúe cuando es verdadera. [4]
28. Crear muchos threads diferentes en el programa puede pegarlo más que simplemente no usar threading del todo.
29. Una forma de apagar y prender los efectos de sonidos, que a diferencia de la música no están sonando siempre o están detenidos, es cambiar su volumen entre 0 y el volumen máximo.
30. Si se usan Tkinter y los módulos de mixer de pygame en conjunto es buena idea crear una función para cuando se sale el programa que cierre la ventana principal de Tkinter, la que se inicializó con Tk(), y que cierre mixer.

Recomendaciones

1. Usar nombres lo más descriptivos posible para funciones y variables es recomendable para que así el código sea fácil de leer y entender y ayuda a depender menos en comentarios internos.
2. Dividir el código en secciones claramente diferenciadas puede ahorrar mucho tiempo cuando uno está tratando de encontrar algo y ayuda a entender aún mejor el código.
3. Una forma muy sencilla de revisar si dos objetos están colisionando es utilizando 2 ifs. El primero revisa que los objetos se estén tocando en el eje x revisando la posición de sus extremos y el segundo revisa los extremos verticales y que los objetos se estén tocando en el eje y.
4. Al menos en el contexto de cómo Tkinter detecta que se presiona una tecla, es buena idea tener una lista global donde se agrega una tecla cuando se presiona y se quita cuando se deja de presionar para controlar el movimiento del jugador porque así se detecta cuando una tecla se mantiene presionada en vez de detectar como si se presionara muchas veces, además así se puede presionar más de una tecla a la vez. [1]
5. Si se desea una funcionalidad parecida a un switch case en C, en vez de usar una cadena de ifs/elifs que puede volverse muy largo, es mejor crear un diccionario en que las llaves sean los diferentes casos y retornan el contenido de la llave con una llamada. [3]
6. Si se desean diseñar niveles manualmente se recomienda llevar el tiempo que ha pasado en el nivel y usar un diccionario que las llaves sean el segundo en el que se desea que ocurra algo específico. Luego se puede usar el método get de los diccionarios para saber qué hacer cada segundo y asignar instrucciones vacías cuando la entrada no existe en el diccionario. Esto permite una forma fácil de leer y entender lo que sucede en el nivel y modificarlo.
7. Para extraer sprites del juego original es útil usar Paint para cortar los screenshots que uno les tome y luego exportarlo a piskel para limpiarles el fondo y hacerlos pngs. [7]
8. Utilizar Paint y Piskel para modificación y creación de sprites. Ambas son herramientas gratis y fáciles de usar, la primera viene con Windows y la segunda se puede encontrar en internet. [7]
9. Si se van a extraer los assets del juego original, hacerle zoom a la ventana del browser en la que se esté jugando el jugo para que los screenshots vengan con la mayor resolución y calidad posible. Además el juego corre más lento y se vuelve más fácil tomarle screenshot a los frames que a uno le falten de la animación de un sprite específico.
10. Si se ocupan pasar funciones por referencia con parámetros, entonces usar funciones anónimas lambda ya que si se le pasan parámetros la función va a correr cuando se pasa por referencia.
11. Crear funciones que se encarguen de mover los diferentes objetos en la pantalla, moviéndolos un poquito y que se llame muchas veces por segundo para simular un movimiento fluido.
12. Agregar a las funciones de movimiento una parte que revise que si el objeto que se está moviendo se sale de la pantalla entonces su vida se reduzca a 0 para que sea eliminado.
13. Crear funciones que se encarguen específicamente de la animación de los sprites de los objetos que tengan animación, estas se van a encargar de cambiar la imagen del objeto muchas veces por segundo. Separar está función de la animación de movimiento en el espacio es útil para que se puedan animar a velocidades diferentes.
14. Crear funciones que guarden y lean archivos .txt si se desea tener memoria que se guarde en el disco duro y no solo utilizar la memoria RAM del programa. Esto es útil para cualquier información que uno quiera retener aún cuando se cierra el programa.
15. Utilizar trys and excepts cuando se va a intentar leer o escribir a un archivo para manejar errores posibles que se puedan dar a la hora de leer o escribir al disco duro, esto va a evitar que el programa tire error si ocurre algo o si el archivo que se intenta leer aún no existe.
16. Si se desea escribir texto usando imágenes de letras y números, es recomiendo que todas las imágenes tengan el mismo tamaño en pixeles para no tener que crear reglas específicas para manejar el tamaño de cada letra diferente.
17. Recomiendo asegurarse de que la entrada que se le da a la función que escribe texto con imágenes solo contenga caracteres para los que uno tenga imagen y limitar estos caracteres lo más posible para requerir la menor cantidad de imágenes posible. Por ejemplo no permitir acentos ni caracteres especiales y que todo esté en mayúsculas.
18. Recomiendo no utilizar un solo canvas sino usar varios dependiendo de la pantalla para no tener que eliminar absolutamente todos los widgets creados cuando se cambia de pantalla sino que siguen existiendo pero uno cambia el canvas solamente.
19. Recomiendo utilizar el método delete(‘all’) cuando se quieren eliminar todas las imágenes y textos dibujados en un canvas ya que este los elimina todos en vez de usar delete para cada objeto individualmete. También tratar de que los widgets creados sean lo más constantes posibles y no se ocupen eliminar ya que estos hay que removerlos individualmente a cada uno.
20. Utilizar el método de los canvas after para crear animar objetos y para repetir cualquier acción que requiera recursión infinita con un intervalo de tiempo definido entre llamadas. [2]
21. Si se quiere restringir el texto que alguien puede ingresar en un entry widget se le puede asignar una variable inicializada con StringVar(), luego utilizar el método trace de estas variables para registrar cuando se le hacen cambios a esta variable (por ejemplo escribiendo en el widget) y modificarla, cortándola, quitándole caracteres específicos o lo que uno desee.
22. Utilizar el módulo unidecode de la librería con el mismo nombre para remover los caracteres especiales de un string y convertir letras con acentos a su equivalente sin acentuar. Se puede utilizar si se desea restringir un entry widget para que solo acepte entrada de caracteres alfanuméricos regulares. [5]
23. Estructurar a los enemigos y al personaje con clases.
24. Guardar a todos los enemigos u objetos similares del mismo tipo en una lista global a la que se agregan cuando se crean y se eliminan de la lista cuando se remueven porque se mueren o condiciones similares.
25. Tener funciones dedicadas a inicializar estos objetos con valores iniciales y agregarlos a las listas globales.
26. Hacer que las funciones de animación, colisiones o cualquier otro tipo que se encargue de modificar estos objetos recorra las listas globales para afectar los cambios en todos los objetos.
27. Utilizar la herramienta JFXR para crear sonidos personalizados para los diferentes efectos de sonidos en el juego si es un juego con temática de arcade o retro. [6]
28. Referenciar las leyes de la física de la vida real para simular ciertas cosas como la gravedad.
29. No estar creando una gran cantidad de threads únicos ya que esto puede pegar al programa.
30. Para “apagar” un sonido que solo suena a veces, se puede dejar que siga sonando de manera normal pero cambiar su volumen a 0 y así uno se evita tener que poner condiciones que revisen si el sonido está prendido o apagado.

Análisis de Resultados

Diagrama de módulos:



Plan de pruebas:

**Colisiones:**

1. Correr el juego y observar que se comporte de manera esperada.
2. Corregir cualquier error que indique el Shell de Python.
3. Agregar un print a las listas que contienen a los enemigos y las balas para ver si se remueven correctamente al colisionar.
4. Hacer un print cuando se pasa cada etapa del checkeo de la colisión (si está colisionando en su posición X y en su posición Y).

**Pantalla de Scores**

1. Correr el juego para generar puntajes y ver que sí se guardan en el archivo .txt.
2. Corregir cualquier error que indique el Shell de Python.
3. Revisar que en la pantalla de scores salgan bien como se espera.
4. Hacerle print a la lista que contiene los datos del scoreboard y revisar que esté formateada como se espera.
5. Editar manualmente el .txt que contiene los puntajes y ver que los despliega correctamente de igual manera.

**Botones**

1. Correr el programa e intentar estriparlos y ver que funcionen como se espera.
2. Corregir cualquier error que indique el Shell de Python.
3. Agregar prints en las funciones que llaman para verificar que las están llamando correctamente.

**Movimiento y animaciones**

1. Correr el juego y observar que todo se comporte como se espera.
2. Corregir cualquier error que indique el Shell de Python.
3. Agregar prints a las listas de enemigos, del jugador y de las balas para revisar que sus datos están siendo modificados como se debe.

**Cambio de controles**

1. Corro el programa e intento cambiar los controles para asegurarme que los botones funcionan como se espera.
2. Corregir cualquier error que indique el Shell de Python.
3. Juego el juego para verificar que ahora el personaje se controla con los nuevos controles.
4. Le hago print a las variables que contienen la tecla asignada a cada control.

Literatura y Fuentes Consultadas

[1]Recuperado Julio(2020): <http://effbot.org/tkinterbook/><https://stackoverflow.com/questions/39606019/tkinter-using-two-keys-at-the-same-time>

[2]Recuperado Julio(2020): <https://runestone.academy/runestone/books/published/thinkcspy/GUIandEventDrivenProgramming/10_timer_events.html>

[3]Recuperado Julio(2020): <https://data-flair.training/blogs/python-switch-case/>

[4]Recuperado Julio(2020): <https://www.reddit.com/r/learnpython/comments/avlm8n/tkinter_button_that_waits_for_keypress_after/?sort=top>

[5]Recuperado Julio(2020): <https://pypi.org/project/Unidecode/>

[6]Recuperado Julio(2020): <https://jfxr.frozenfractal.com/>

[7]Recuperado Julio(2020): <https://www.piskelapp.com/>