**Instituto Tecnológico de Costa Rica**

**Ingeniería en Computadores**

**Taller de Programación**

**Grupo 2**

**Proyecto II | Documento Técnico Ejecutivo**

**Profesor: Milton Villegas Lemus**

**Ignacio Vargas Campos**

**2019053776**

**I Semestre, 2020**Introducción

Este proyecto consiste en crear un juego con temática espacial, en el que el juego en sí tiene la percepción de ser 3D. Debía incluir 2 modos de juego, el primero siendo destrucción de asteroides; en donde se deben de destruir o esquivar asteroides que se acercan hacia el jugador. El segundo siendo una prueba de maniobras, en donde se acercan aros hacia el jugador y se debe de intentar pasar a través de ellos. Además el juego debe incluir personajes para escoger y estos deben de poder configurar, borrar y agregar más si se desea.

Para la realización de este proyecto me apoyé principalmente en la librería Pygame (versión 1.9.6) para la creación del programa, ya que esta librería está orientada principalmente hacia la creación de juegos, en contraste con Tkinter que está más orientada hacia la creación de interfaces gráficas. De igual manera sí se usó la librería de Tkinter (versión 8.6) ya que el módulo after de esta fue clave para poder tener recursión infinita en el programa. También se usaron otras librerías estándar de Python como “os”, “random” y “platform”.

Conclusiones

1. Los objetos rect de Pygame tienen varios módulos para verificar si un punto está colisionando con ellos o si hay rects colisionando con otros rects.
2. Pygame genera eventos para cuando se tocan las teclas del teclado y se da click con el mouse, los cuales se pueden utilizar para manejar qué hacer cuando estos eventos suceden.
3. El módulo mouse de Pygame tiene módulos para detectar si los botones del mouse están siendo estripados.
4. El módulo mouse de Pygame tiene un módulo para obtener las coordenadas de dónde se encuentra el mouse pointer.
5. Es posible embeber una ventana de Pygame a otra ventana en el sistema a través del agregar la entrada ‘SDL\_WINDOWID’ en el diccionario de os.environ y asignarle el id de la ventana la que se quiere embeber. [2]
6. Al embeber una ventana de Pygame en Tkinter, se puede utilizar el módulo after de los widgets de Tkinter para tener recursión infinita en Pygame.
7. Para mantener un framerate estable con el módulo .after de Tkinter, esta llamada recursiva debe ponerse antes de las llamadas a funciones que hagan labores pesadas como pintar sobre la pantalla o muchos cálculos ya que estas pueden atrasar la llamada si ocurre al final de la función.
8. Si se desean crear o modificar assets, se puede utilizar la herramienta gratis y open source, Piskel. [3]
9. Usar nombres descriptivos y largos para las variables y funciones ayuda hace el código más legible a simple vista.
10. Usar comentarios para dividir las diferentes secciones del código lo hacen más fácil de navegar.
11. Para manejar las diferentes pantallas en el entorno de Pygame, se puede tener un loop infinito de recursividad que represente a cada pantalla y corra todas las funciones necesarias para su funcionamiento.
12. Usando archivos .txt es posible escribir en ellos y guardar información que se mantiene guardada aun cuando uno cierra y vuelve a abrir el programa.
13. 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.
14. 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.
15. Por la manera en la que Python maneja las listas, si una función le agrega un elemento a esa lista y el elemento es una lista. Se agrega una referencia a la lista agregada, entonces si la función agrega múltiples elementos realmente se están agregando referencias a la misma lista. Por lo tanto si se modifica una de las listas nuevas agregadas, se terminan modificando todas las que fueron agregadas por este método. [4]
16. Para crear una copia de una lista en vez de una referencia se puede llamar a la lista especificando el índice [:]. [4]
17. Para saber si se dio click en un frame específico. Se puede tener una variable booleana que se llame ”click” o algo parecido y cuando se detecte un evento de click entonces esta se cambia a True.
18. Para detectar si se le da click a un área rectangular específica de la pantalla se puede detectar un click y luego obtener el punto sobre el que está el mouse con el módulo mouse.get\_pos() y verificar si ese punto está sobre un objeto rect con el módulo rect.collidepoint(x,y).
19. Si uno escribe “color picker” en Google y le da buscar, puede accesar una herramienta de Google para escoger colores y le da el código hex, el código rgb con valores de 0-255 y otras maneras de representar el color.
20. En el sitio itch.io se pueden encontrar una amplia variedad de assets gratis para crear un juego.
21. Se puede detectar si un punto está dentro de un círculo calculando la distancia entre el punto y el centro del círculo y luego revisando si la distancia es menor al radio de dicho círculo.
22. Se puede detectar si un rectángulo o cualquier objeto está colisionando con un círculo calculando cuántos de sus puntos están dentro del círculo. Si todos están dentro entonces el objeto está contenido dentro del círculo y si solo algunos están adentro entonces está colisionando parcialmente.
23. Se puede crear un efecto 3d de que un objeto se está acercando si la imagen se pinta cada vez más grande y se mantiene centrada en el mismo lugar.
24. Se pueden crear “pseudo-clases” usando listas y definiendo que un “pseudo-objeto” objeto agregado a esa lista va a seguir un mismo formato de atributos.
25. Se puede crear una caja de entrada de texto en Pygame definiendo un rect al que si se le da click entonces se empiezan a detectar los eventos específicos de presionar teclas e ir agregando las letras de las teclas a una variable string que luego se use para escribir el texto en pantalla.
26. En Pygame se puede utilizar una parte específica de una imagen, pegando esta a un Surface con el método blit y especificando un área rectangular de la imagen que pegar.
27. Se puede transformar el tamaño de una imagen con el módulo transform de la librería. Pegando una imagen a un Surface y luego ajustando el tamaño de esta.
28. El método “chr()” de Python puede obtener un código ascii y retornar la tecla que representa.
29. Los eventos de teclas de Pygame tienen un componente Unicode y de código ascii cuando se estripan pero cuando se sueltan solo tienen el código ascii.
30. Para crear animaciones si se refresca el fondo de pantalla completo cada frame, ya que esto ralentiza mucho al programa. En cambio es posible crear una función que refresque pedazos especificados de la pantalla y así se ahorran recursos.

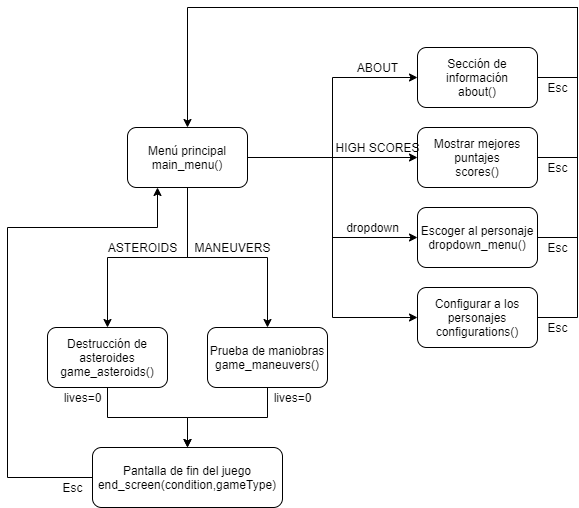
Recomendaciones

1. Se recomienda utilizar el módulo rect de Pygame para crear objetos rectangulares ya que este simplifica la detección de colisiones y es muy útil en general para guardar las propiedades de estos
2. Manejar los clicks del mouse en Pygame a través de los eventos que se generan automáticamente, ya que estos tienen varias propiedades que se pueden manejar y ser útiles sobre los clicks.
3. Se recomienda usar el módulo mouse de Pygame si se desea detectar si el mouse está siendo estripado en un dado momento, pero para detectar un solo click igual se recomienda más detectarlos a través de la cola de eventos.
4. Usar módulo mouse de Pygame para detectar la posición del mouse, especialmente útil para saber si se le está dando click a una parte específica de la pantalla.
5. Utilizar el módulo os y agregar en el diccionario environ de este una llave “SDL\_WINDOWID” con el id de alguna ventana en la que se quiera embeber una ventana SDL. [2]
6. Utilizar el módulo after de Tkinter para crear recursión infinita en el programa sin llenar la memoria.
7. Poner las llamadas recursivas en los loops del juego antes que todas las llamadas a funciones que pinten y dibujen sobre la pantalla, ya que estas toman tiempo y hacen que la velocidad a la que corra el programa sea inconsistente.
8. Utilizar Piskel para modificación y creación de sprites. Es una herramienta gratis, open source, fácil de usar, se puede encontrar en internet y utilizar desde el navegador o descargar la aplicación offline. [3]
9. Se recomienda ponerle los nombres más descriptivos posibles a las funciones y variables para aumentar legibilidad del código y disminuir la confusión al inspeccionarlo más tarde
10. También es recomendado utilizar comentarios de línea como una herramienta para crear particiones visuales en el código para que sea más fácil de navegar.
11. Se recomienda tener una función que represente a cada pantalla que corra en un loop infito de recursividad y que cada vez que sea corrida/llamada represente un frame y esta se encargue de hacer todas las llamadas a las funciones necesarias para el funcionamiento correcto de la pantalla específica.
12. Crear funciones que se encarguen específicamente de leer y guardar información en archivos .txt si se desea tener memoria que se guardad en el disco duro y no solo utilizar la memoria RAM temporal del programa. Esto es útil para cuando se desee retener información en el programa aun cuando se cierra y se vuelve a abrir.
13. Usar trys and excepts en las funciones que escriben y leen en archivos para manejar cualquier error posible a la hora de leer o escribir al disco duro, esto se va a evitar errores hasta de aspecto físico en el que el disco duro tuvo un comportamiento no esperado.
14. A la hora de leer información de un archivo .txt y guardarla en una estructura dentro del programa, revisar si efectivamente se leyó información, y que si no se leyó nada porque el archivo estaba vacío o aún no existía entonces inicializar datos automáticamente y guardarlos. Así se maneja el caso en el que se borró la información o se está utilizando el programa por primera vez en un nuevo dispositivo.
15. Tener funciones dedicadas al movimiento y animación de los elementos del juego que los muevan una distancia corta o les hagan cambios leves y que se llamen cada frame que corre el loop principal del juego o la pantalla en la que esté corriendo.
16. Se recomienda que si se va a usar el método .append o algo similar para agregarle un elemento de tipo lista a una lista en Python, llamar a la lista con el índice [:] para agregar una copia de esta y no una referencia a ella. Ya que si se quieren agregar varias listas a una lista y se agregan por referencia entonces pueden quedar vinculados y ser solo la misma referencia a una sola lista muchas veces [3].
17. Se recomienda tener una variable local en los loops de las pantallas que registre específicamente si se ha dado click en el frame específico que esté corriendo. La implementación recomendada es asignarla a False dentro del loop y luego revisar la cola de eventos y si se encuentra un click en esta cambiarla a True.
18. Utilizar el módulo mouse.get\_pos() para obtener las coordenadas del mouse en un momento dado y luego el módulo rect.collidepoint((x,y)) con las coordenadas dadas por mouse.get\_pos() para verificar si el mouse está sobre un objeto rectangular. Esto es especialmente útil para detectar si se le está dando click a un botón o un área de la pantalla específica.
19. Utilizar la herramienta de Google para escoger colores. Es gratis y fácil de usar, solo se debe de buscar las palabras “color picker” en Google y esta lo deja a uno buscar y escoger colores y le da el código del color en muchos formatos, incluyendo RGB con códigos decimales y hexadecimales.
20. Utilizar el sitio itch.io para buscar assets de juegos gratis y de uso libre.
21. Se recomienda detectar si un punto en el espacio está dentro de un círculo creando una función que calcule la distancia entre el punto, el centro del círculo y lo compare con el radio. Esto para simplificar le proceso de detectar colisiones con círculos.
22. Se recomienda crear una función que tome puntos, que representen los ejes de un objeto, verifique cuantos de estos están dentro del círculo para saber si el objeto es contenido por el círculo parcial, totalmente o no del todo.
23. Asignarles a los objetos una variable correspondiente a qué tan lejos están y hacerlos más pequeños de manera proporcional a esta variable para así crear un efecto 3d de distancia.
24. Estructurar al personaje y a los otros elementos del juego como asteroides y aros con clases. Alternativamente usar pseudo-clases.
25. Guardar todos los elementos de un mismo tipo, sean aros, asteroides o algún otro tipo de elemento. En listas donde se mantengan a todos los elementos activos en un momento dado. Así para animarlos o cualquier funcionamiento similar solo se debe recorrer la lista.
26. Si se desea tener una entrada de texto en Pygame se recomienda tener una variable string que se modifica dependiendo de las teclas estripadas que vengan en la cola de eventos y pintar esta variable en la pantalla, para así simular una entrada de texto.
27. Tener funciones específicas para agarrar partes específicas de un spreadsheet pintando las imágenes en superficies del tamaño que se desee cortar de la imagen y luego pintando la superficie en la pantalla principal.
28. Utilizar el módulo transform de la librería de Python para cambiar el tamaño de una superficie, esto se puede usar para cambiar el tamaño de las imágenes y es útil para crear el efecto 3d mencionado anteriormente.
29. Usar el método chr() con el que viene Python para convertir códigos ascii a su representación respectiva en string.
30. Crear una función que pinte el fondo de pantalla en un área específicada para así “borrar” áreas específicas de la pantalla y no volver a pintar la imagen de fondo en cada iteración de los loops, ya que pegar la imagen entera todas las veces ralentiza demasiado el programa.

Análisis de Resultados

Se logró la implementación completa de todas las especificaciones requeridas del juego. Se verificó que todas las partes del programa operan como se espera.

Diagrama de módulos:



Plan de pruebas:

Menú principal: main\_menu()

* Debe mostrar visualmente todos los elementos de la pantalla en su lugar correcto.
* Al mover el mouse encima de un botón el texto se debe de “prender”.
* Al darle click alguno de los elementos de la interfaz gráfica se debe de activar la pantalla escogida.

Destrucción de asteroides: game\_asteroids()

* La nave debe de moverse con las teclas de flechas.
* La nave debe inclinarse hacia arriba y hacia abajo cuando se mueve en las respectivas direcciones.
* Al estripar la tecla Z la nave debe disparar.
* Deben de aparecer asteroides, dependiendo de la dificultad escogida.
* Los asteroides se deben de acercar visualmente hacia el jugador.
* Si la nave está encima del asteroide cuando este se termina de acercar entonces le quita una vida al jugador y el asteroide desaparece.
* Si se esquiva el asteroide este desaparece y suma puntos.
* Si se le dispara al asteroide este desaparece y suma puntos.
* Si el jugador se queda sin vidas o sin combustible aparece la pantalla de fin de juego.
* Aparecen barriles de combustible aleatoriamente en la pantalla cada cierto tiempo.
* El jugador pierde combustible constantemente.

Prueba de maniobras: game\_maneuvers()

* La nave debe de moverse con las teclas de flechas.
* La nave debe inclinarse hacia arriba y hacia abajo cuando se mueve en las respectivas direcciones.
* Deben de aparecer aros, dependiendo de la dificultad escogida.
* Los aros se deben de acercar visualmente hacia el jugador.
* Si la nave está dentro del aro cuando este se termina de acercar entonces le suma puntos al jugador y este desaparece
* Si se esquiva el asteroide este desaparece y no suma puntos.
* Si cuando se termina de acercar el aro, el jugador está posicionado tocando el borde de este. Entonces el jugador pierde una vida, no se suman puntos y el aro desaparece.
* Si el jugador se queda sin vidas o sin combustible aparece la pantalla de fin de juego.
* Aparecen barriles de combustible aleatoriamente en la pantalla cada cierto tiempo.
* El jugador pierde combustible constantemente.

Pantalla de fin del juego: end\_screen(condition,gameType)

* Muestra si el nivel se terminó o si el jugador se murió.
* Muestra el puntaje del jugador
* Le informa al jugador si el puntaje obtenido con el personaje es un récord del personaje y se encuentra en el top 7 de puntajes.
* Si se da click en cualquier parte vuelve al menú principal.

Sección de información: about()

* Muestra el cuadro con la información encima del menú principal.
* Si se da click fuera del cuadro o se estripa la tecla escape vuelve al menú principal

Mostrar mejores puntajes: scores()

* Muestra los mejores 7 puntajes para los dos modos de juego.
* Los puntajes están en orden descendiente.
* Muestra los nombres e imágenes de los personajes a los cuales les corresponde cada puntaje.

Escoger al personaje: dropdown\_menu()

* Despliega el menú desplegable.
* Muestra todos los personajes disponibles, sin incluir al que está escogido en el momento dado.
* Al tener el mouse encima de una opción, esta se “prende”.
* Al dar click sobre algún personaje este se vuelve el nuevo personaje escogido y se refleja visualmente al volver al menú principal.
* Si hay más personajes de los que caben en la pantalla el menú desplegable se vuelve también desplazable. Entonces muestra una barra de desplazamiento y si se utiliza la rueda del mouse entonces se desplaza a través del menú.
* Al desplazar el menú se desplazan los personajes disponibles.
* Todas las funcionalidades mencionadas antes funcionan correctamente cuando el menú es desplazable.

Configurar a los personajes: configurations()

* Se despliegan 4 personajes en pantalla con toda su información correcta.
* Si se le da click a alguno de los campos modificables muestra todas las opciones disponibles y si se le da click a alguna opción el personaje que se estaba modificando demuestra el cambio visualmente.
* Si se le da click a las flechas de los lados se mueve de página.
* Si se le da click a la última tarjeta que solo tiene un “+”, se empieza el proceso de agregar un personaje nuevo.
* El personaje nuevo se muestra cuando se termina de agregar.
* Cuando hay más de 12 personajes aparece la opción de borrar los personajes.
* Al borrar el personaje este se deja de mostrar.
* Si se borra un personaje y vuelven a haber solo 12 personajes entonces desaparece la opción de borrar personaje.

Bitácora:

24/7/20:

Empieza la creación del proyecto. El problema se presentó de que se deseaba usar Pygame completamente para el juego, pero la única manera de realizar recursividad infinita que existe en Pygame y Tkinter es específicamente el módulo .after que se puede aplicar en los widgets de Tkinter. Esto implica que mínimo debo crear una ventana de Tkinter para poder tener recursividad, pero quiero crear mi juego en una ventana de Pygame. Al final descubrí un método de embeber una ventana de Pygame dentro de un widget de Tkinter y así logré mi objetivo.

29/7/20:

Se logró crear botones en Pygame. También se agregó una imagen de fondo al juego. Con la imagen de fondo se notó que agregarla a la pantalla en cada iteración recursiva hacía que el programa fuera muy lento por lo que se hizo que solo se agregue si la pantalla se está llamando por primera vez o si se está “reseteando” y esto arregló el problema.

Luego se creó una función que “borra” un área en la pantalla, para que si se desea por ejemplo cambiar de posición una imagen, cambiarle el color o algún otro cambio que requiera volver a dibujarla de manera o en una posición diferente, se pueda solo volver a pintar el fondo de pantalla encima de esa área en vez de volver a dibujar el fondo entero.

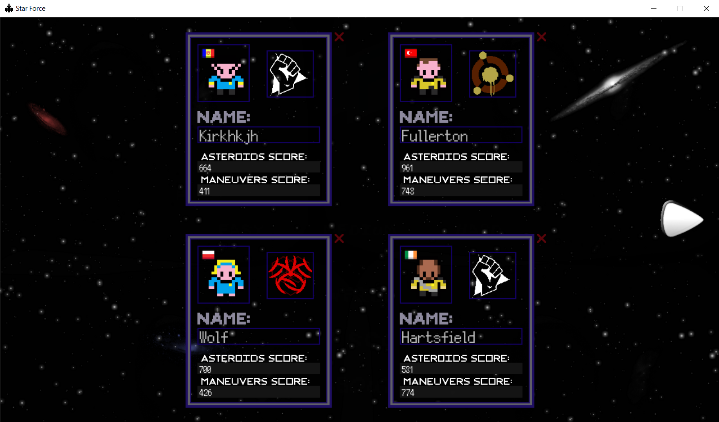
Por último se creó el esqueleto primitivo de los botones del menú, que solo serán texto al que si se le pone el mouse encima cambian de color y si se les da click cambian de pantalla.

31/7/20:

Hoy se terminó de crear la interfaz del menú principal, solo que los botones llaman a funciones que aún no hacen nada porque no se han creado esas pantallas. La imagen adjunta es el menú principal del juego. A futuro se planea agregar un menú desplegable para escoger al personaje con el que se va a jugar pero eso vendrá después de terminar de crear la pantalla de configurar los pilotos.

Además, se crearon el módulo que guarda los pilotos en un archivo .txt, el módulo que los lee y los guarda a una lista en el programa y un módulo que genera pilotos nuevos si el archivo no se ha creado aún. Se encontraron varios problemas en estos módulos tratando de crearlos, el primero fue el no saber cómo determinar si se había llegado al final del programa sin saltarse parte de la información. Se solucionó leyendo el primer carácter del nombre del piloto y sí retornaba vacío significa que se llegó al final, sino se termina de leer el nombre y se concatena con la primera letra ya leída. El resto de problemas fueron detalles triviales como que se olvidó poner return en las llamadas recursivas de la función.

Por último se obtuvieron todos los “assets” que se van a ocupar a futuro para todo lo que tenga que ver con los personajes.

1/8/20

Hoy se agregó la sección de configurar a los personajes en su completitud. Cuando se ingresa a esta sección del menú se muestran 4 de los personajes a la vez y se puede pasar de página para ver el resto de los personajes y si se llega al final de todos los personajes está la opción de agregar uno nuevo. Los mayores retos con esta sección fue hacer los módulos para modificar los atributos de los personajes, especialmente el nombre ya que se tuvo que crear una entrada de texto desde 0, conseguir todas las imágenes de personajes, que toda la información se desplegara correctamente y que detectara qué atributo de qué personaje desea uno modificar.

2/8/20

Se creó un “drop-down menu” en el menú principal, donde se puede escoger el personaje que se va a utilizar en el juego. Esto presentó muchas dificultades como lograr que mostrara correctamente todos los personajes, sin que mostrara el personaje que estuviera escogido en el momento. Luego se notó que con más de 14 personajes el menú se salía de la pantalla por lo que se agregó la capacidad de “scrollear” el menú si hay más de 14 personajes, esto también resultó muy complicado ya que dependiendo de qué tanto haya bajado en el menú hay que mostrar personajes diferentes y se dificultó mucho mostrar los personajes correctos y que el “scrolling” funcionara bien. La imagen adjunta es un ejemplo del menú desplegable funcionando.

Después de lograr esto se creó la pantalla de altos puntajes que no presentó ningún inconveniente y de igual manera la pantalla de “About” de información extra tampoco presentó un obstáculo.

3/8/20

Hoy se empezó la creación del juego como tal. Se empezó con la creación del movimiento de la nave espacial, para este se deseaba que la nave se inclinara verticalmente cuando se mueve hacia arriba abajo y se inclinara hacia los lados cuando se moviera horizontalmente. Se empezó trabajando en la inclinación vertical y luego se determinó que la inclinación hacia los lados haría varios cálculos, como de colisiones un poco muy complicados ya que habría que calcular colisiones con un rectángulo inclinado y no uno paralelo con los ejes de la pantalla.

Con la inclinación que sí se implementó, algunas dificultades fueron que en las imágenes utilizadas la nave entre más se aleje de su posición central viendo hacia al frente más bruscos son los cambios entre una imagen y la otra entonces se modeló la inclinación con una función exponencial para que no se sienta como que cada vez se inclina más rápido, sino que se inclina con una velocidad que se siente más natural.

Luego se le agregó la funcionalidad de disparar a la nave lo cual fue difícil porque para crear el efecto de que los disparos se alejan en la distancia hay que encogerlos pero entonces hay que borrar la imagen anterior y luego no se pueden pintar en el mismo lugar sino que hay que mover la coordenada x para que sigan centrados en el mismo punto. Esta es una característica que luego se aplica a todos los objetos que se alejan o se acercan en el juego. Además se tuvo problema con que si el disparo ocupaba un espacio cercano a la nave entonces no aparecía porque al borrar el espacio de la nave para updatearlo y animar la nave, se borraban los disparos. Por último también se tuvo problemas con que los disparos se quedaban pintados en la pantalla cuando ya debían de haber desaparecido completamente y era un error de cómo estaba programada la función recursiva que animaba los disparos.

Después de esto se crearon los asteroides y sus animaciones lo cual fue sencillo después de haber hecho los disparos pero igual se encontraron algunos inconvenientes triviales. Por ejemplo, a los objetos que tienen profundidad se les asigna una variable de “depth” y se había decidido que “depth”=0 significa que están en el mismo plano que la nave, pero esto se tuvo que cambiar a “depth”=1 porque sino habían partes de los módulos que intentaban dividir por 0.

Luego se programaron las colisiones asteroides-jugador y disparos-asteroides lo cual no presentó inconvenientes.

Por último se le asignó una variable de “fuel” a la nave que se resta conforme avanza el juego y se programaron los barriles de gasolina que aparecen conforme avanza el nivel para regenerar la energía de la nave, si la nave se queda sin energía se acaba el juego pero aún no se programa la pantalla de fin del juego.

4/8/20

Hoy se creó una barra en el borde superior de la pantalla que muestra el nivel de energía, el tiempo transcurrido, el puntaje que lleva el jugador, las vidas que tiene y el personaje que se tiene seleccionado.

También se diseñaron los niveles para las diferentes dificultades del juego de destrucción de asteroides.

Por último se creó la pantalla que sale cuando uno termina el juego. Esta avisa si se obtiene un puntaje entre los mejores 7, pero se tuvo dificultad porque lo mostraba aunque el personaje no alcanzará un nuevo mejor puntaje; esto creaba una discrepancia porque podía salir que era el puntaje #5 y cuando uno entraba a ver los puntajes realmente tenía el puesto #2. Esto se arregló haciendo que este mensaje solo se active si el personaje supera su propio record.

4/8/20

Se empezó creando los aros y haciendo que estos se vayan acercando hacia el jugador, luego se creó el modo de juego de prueba de maniobras, se diseñó el nivel y por último se programó la colision de los aros con el jugador. Todo esto fue muy fácil excepto por las colisiones porque se hizo de la misma manera que con los asteroides.

El problema que se tuvo con las colisiones es que al probar el juego. Parecía que el jugador podía pegar el ala con el borde del aro e igual iba a contar como que pasó el aro. Después de muchas pruebas se notó que realmente si estaba pasando dentro del aro en su máximo tamaño, pero el tamaño máximo del aro pasaba muy rápido entonces el jugador quedaba con la impresión de que el tamaño máximo del aro en realidad era la mitad del tamaño real del aro. Lo que se hizo para resolver este problema fue ralentizar la velocidad a la que el aro se acercaba hacia el jugador conforme más cerca estuviera en términos de profundidad. Con esto se concluye el juego.

6/8/20

Se empezó a trabajar en el documento técnico ejecutivo. Se escribieron las conclusiones obtenidas con la realización del proyecto

7/8/20

Se continuó trabajando en el documento técnico ejectuvo. Se escribieron las recomendaciones basadas en las conclusiones obtenidas.

8/8/20

Se finalizó el trabajo en el documento técnico ejecutivo y se escribió la documentación dentro del programa y el módulo mi\_auto\_doc().

Literatura y Fuentes Consultadas

[1]Recuperado Julio(2020): <https://www.pygame.org/docs/>

[2]Recuperado Julio(2020): <https://www.semicolonworld.com/question/59548/embedding-a-pygame-window-into-a-tkinter-or-wxpython-frame>

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

[4]Recuperado Agosto(2020): <https://stackoverflow.com/questions/5280799/list-append-changing-all-elements-to-the-appended-item>