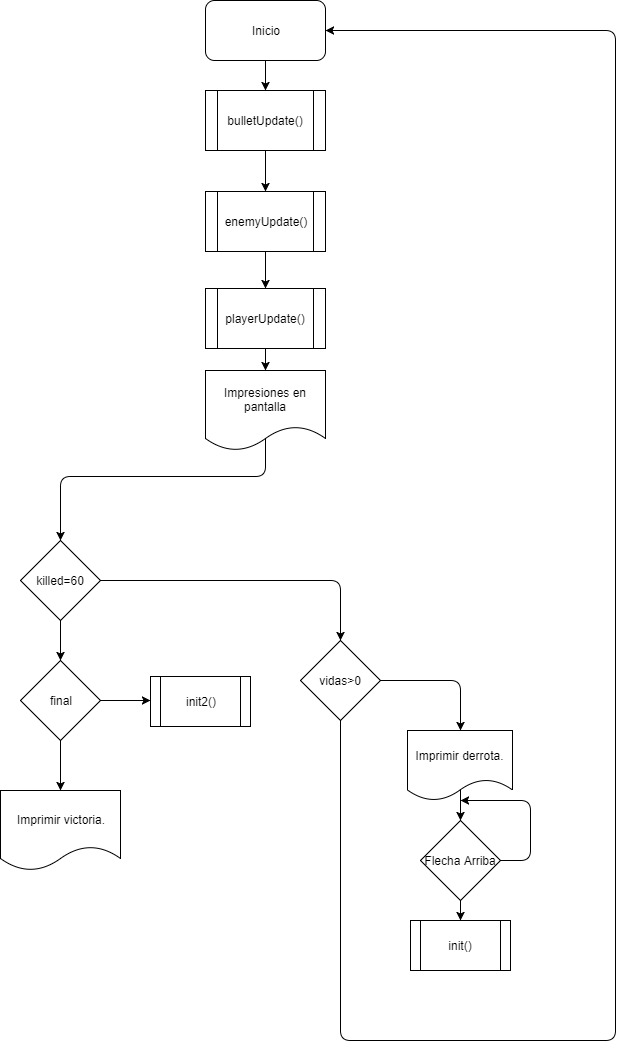
***Documentación Técnica:***

*Funcionamiento general.*

Para el desarrollo de la aplicación se utilizó programación procedural. El funcionamiento del juego por lo tanto se desarrolla dentro de una función principal, en el que se van actualizando mediante el llamado a funciones, el estado y posición del jugador, los enemigos y las balas. A su vez se debe volver a dibujar a los mismos en sus nuevas posiciones. Por otra parte en esta función principal también se realizan pruebas lógicas que comprueban si se presionaron las teclas de pausa o salir. Además, existen otras pruebas que determinan si el jugador ha perdido, ha ganado, o todavía está jugando. Una vez que el jugador pierde, es decir, sus vidas son 0, se le pregunta si desea volver a jugar, lo cual vuelve a llamar a la función *init()* la cual vuelve a iniciar en el primer nivel. En caso de que el jugador gane, es decir que haya matado 60 enemigos, se le informa por una impresión por pantalla que si presiona un botón, pasará al segundo nivel, para realizar esto, se llama a la función *init2()* la cual es muy similar a *init()* ya que solo se realizan cambios a las variables de los *sprites*, y la frecuencia de disparos enemigos. Otra variable que cambia es *final*, Si el jugador ya se encuentra en el segundo nivel, la misma es *True*, por lo tanto si se determina que (*killed = 60*)seimprimirá un mensaje por pantalla, indicando que el jugador ha ganado el juego. En cambio si las vidas son 0, el jugador deberá comenzar desde el primer nivel.

En resumen, *init*() e *init2*() se encargan simplemente de definiciones de variables, y carga de recursos, mientras que *run*() es la función en la cual se van ejecutando todas las instrucciones y llamados a procedimientos que derivan en el funcionamiento, a continuación se muestra un diagrama de flujo de cómo funciona *run*() y sus llamados a *init*() e *init2*().



*Resoluciones varias*

# *Barrera de defensa:*

La barrera de defensa se codifico utilizando arreglos, los cuales determinan mediante valores de “1” o “0”, si debe pintarse o no esa porción de la pantalla. Para pintar se utiliza varios ciclos *for each* que recorren todos los datos de este arreglo para pasarlos como rectas a una lista. Luego, se dibujan las barreras mediante un ciclo *for each* que va leyendo todas las rectas en esta lista y pinta la parte correspondiente de la pantalla. También, se controlan las colisiones entre las balas y las barreras en pos de poder borrar aquellas partes de la barrera que colisionen. Para controlar esto se utiliza *collidelist,* esta función devuelve -1 si no hubo colisiones, y si las hubo, devuelve el índice del primer rectángulo que haya colisionado. En base a esto, si el resultado es distinto de -1, se procede a analizar si hubo una colisión en cada rectángulo, y en caso de que así sea, se lo quita de la lista. Esta comprobación se realiza por separado para las balas de los enemigos, y para las balas del jugador.

# *Movimiento de los enemigos:*

Para el movimiento de los enemigos, en primer lugar, en base a una variable de velocidad, se va aumentando su posición en x, o reduciendo, dependiendo de la variable de dirección, la cual determina si los enemigos vuelan hacia la izquierda o la derecha. Una vez que llegan a los limites de x establecidos, (0 y 750), se llama a un procedimiento que se encarga de mover los enemigos hacia abajo, el cual mediante ciclos *for each* recorre la lista de enemigos y les aumenta el valor de su posición en y.

# *Disparos de los enemigos:*

Los disparos de los enemigos son generados al azar. Para lograr eso, se genera un numero al azar entre 0 y 1000, con la función *random.randint(),* y luego, si ese número es mayor a otro numero definido con anterioridad (en *init()* o *init2()*, dependiendo del nivel), se genera un disparo.

# *Movimiento y disparo del jugador:*

El movimiento del jugador se maneja mediante tres *if .*El primero, pregunta si se presiono la tecla derecha, y si el jugador esta antes del límite derecho determinado (800), si esto es verdadero, se aumenta el valor de su posición en x. El segundo, pregunta si se presiono la tecla izquierda, y si el jugador esta antes del límite izquierdo determinado (0), si esto es verdadero, se reduce el valor de su posición en x. Por último se pregunta si se presiono la barra espaciadora, si es así se genera un disparo que surge de la posición del jugador.

*Música y efectos de sonido:*En cuanto a la programación, el sonido no representa una dificultad, simplemente se cargan los sonidos, y se los reproducen.

La canción del juego fue compuesta, y producida en Fl Studio, con los plugins VSTi Digital Maths Chip Machine y Bit box. El efecto de sonido de los disparos fue descargado de la página freesound.org.

*Menú y pausa:*

En cuanto a la pausa del juego, se manejó de la siguiente manera. En la función principal del juego (run), se pregunta si en algún momento es presionada la tecla 'p', si es así, una variable global se pone en verdadero y entra en el procedimiento principal de la implementación de la pausa. En este, se carga y se actualiza la pantalla con un nuevo fondo, y se agregan dos botones, "Continuar " y "Salir". En este procedimiento hay un ciclo while, que tiene como condición de ejecución que esta variable global sea verdadera, y muestra la imagen, si se presiona continuar, la variable global se pone en falso y continua con la ejecución del juego, si se presiona cerrar, se ejecuta una instrucción que cierra la ventana del juego.

En cuanto al menú principal del juego, es el primer procedimiento del juego. El mismo se encarga de cargar y mostrar una imagen de fondo, donde se muestra un título "Los anillos de Saturno", mediante una función y se agregan tres botones "Jugar”, "Controles" y " Salir". Luego queda la pantalla congelada hasta que se presione alguno de los botones, al presionar algún botón llamamos al procedimiento botón(), en el que se encarga de preguntar si el mouse está posicionado sobre el botón, si es así se cambia de color, si no queda con el color por defecto. Además se encarga de averiguar si se presionó el mouse en la posición que se encuentra alguno de los botones mediante una serie de sentencias if, si es así se realiza una acción determinada y el programa sale del menú a ejecutar la misma.