Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores.

CE 1102- Taller de Programación

Estudiantes:

Andrea Fernández

Victor Cruz Jiménez

Profesor: Luis Barboza Artavia

Índice

[**Sección A: Tabla de contenidos** 3](#_Toc75206416)

[**Sección B: Partes del código importantes** 4](#_Toc75206417)

[**Resultados.** 9](#_Toc75206418)

[**Conclusiones.** 17](#_Toc75206419)

[**Recomendaciones** 17](#_Toc75206420)

# **Sección A: Tabla de contenidos**

Introducción.

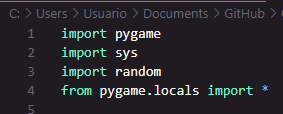
El presente documento se hablará acerca de todos los factores relacionados con el segundo proyecto programado de taller de programación. Se detallará el problema al que se le busca una solución, todos los pasos necesarios para resolverlo. Además, se mostrará las conclusiones y acerca de los principales conocimientos obtenidos al realizar este proyecto.

Descripción del problema.

En el presente proyecto, la empresa en donde trabajamos nos encomienda realizar un programa similar al del proyecto 1(Operation Moon Light), es decir, se busca realizar un programa en Python para probar el desempeño del producto realizado por la empresa, colocando un mayor numero de objetos en pantalla, para de este modo evaluar el desempeño en una situación más difícil. El juego es de tipo “Esquivar”, el jugador deberá esquivar a los elementos que aparezcan en pantalla con las flechas direccionales, los “enemigos” tendrán un movimiento aleatorio, aparecerán un numero especifico de enemigos en pantalla dependiendo del nivel (4 en el nivel 1, 8 en el nivel 2 y 12 en el nivel 3 ), se necesita a su vez una pestaña en la cual se muestren los mejores puntajes, la cual muestre en orden descendentes los mejores puntajes, ordenados por medio de Quicksort

# **Sección B: Partes del código importantes**

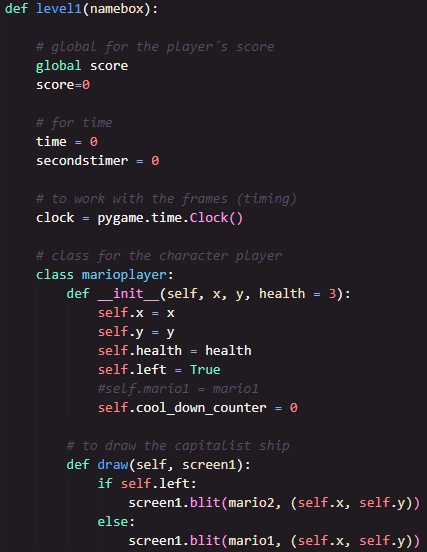
Para dar inicio a la siguiente sección, es importante aclarar que bibliotecas fueron necesarios para llevar a cabo de buena manera el proyecto, fue escogida la biblioteca de Pygame para levar a cabo el desarrollo del mismo. A su vez, es necesario importan unas cuantas bibliotecas mas para poder completar de buena manera el proyecto. Ver 1 figura



1 Figura

**Jugador**

Una vez importadas dichas bibliotecas, las cuales nos permitirán en caso de Pygame, crear una interfaz grafica y crear elementos con mayor comodidad a diferencia que en otras bibliotecas que nos permiten crear interfaces gráficas, la biblioteca random nos permite generar números random en los momentos que los necesitemos. Ahora bien, pasando a las partes del código que nos permitirán crear al jugador, en este caso se observara la clase marioplayer en el “level1”. Ver figura 2



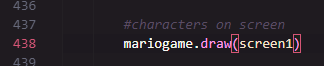
2 Figura

Este código lo que nos permitirá es crear una clase marioplayer la cual contendrá todos los valores necesarios para dibujar a nuestro personaje en pantalla y a su vez darle movimiento. Una vez establecidas las coordenadas en las que se creara el jugador



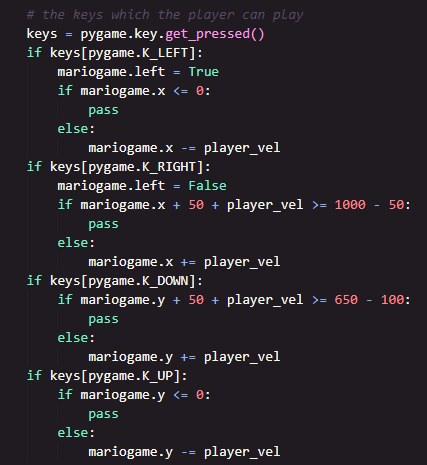
Figura

El código de la figura 3 es el que le dice al programa en que coordenadas se generara el jugador. Este código nos facilitara la generación del jugador. El código de la figura 4 es el encargado de cargar al jugador en pantalla, dicho código se encuentra en el bucle que crea la ventana.



Figura

Ahora una vez que tengamos establecidas las coordenadas del jugador, ahora será necesario llevar a cabo el movimiento del mismo, el código encargado del movimiento se puede ver en la figura 5

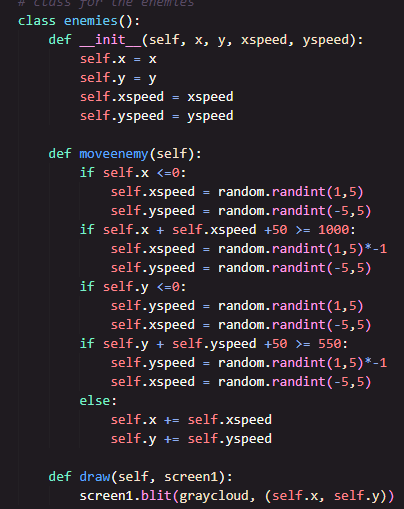


Figura

En palabras simples, el código presente en la figura 5 nos permite detectar el momento en el que se presiona una tecla de las flechas direccionales (UP, DOWN, LEFT, RIGHT). Una vez se presiona una de las teclas direccionales el programa detecta que flecha es y mueve al jugador 50 pixeles hacia la dirección establecida.

**Enemigos.**

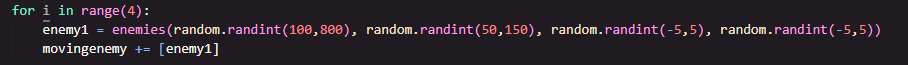
Una vez explicada la funcionalidad del jugador, es necesario explicar la funcionalidad de los enemigos, como se menciono anteriormente, los mismos se van a generar en diferentes cantidades dependiendo del nivel en el que se encuentre el jugador. El código de como se genera el enemigo se vera en la figura 6



Figura

El funcionamiento de este código en sí, es el generar una velocidad en x aleatoria para cada enemigo, y una velocidad en y aleatoria para cada enemigo. Compara su posición actual y hace un calculo para que en el momento que choque el borde de la ventana, convierta su velocidad con un \*-1, para que la velocidad se invierta. Posterior a esto los dibuja en el escenario a los enemigos.

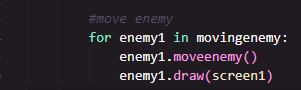
Este código nos genera una cantidad indefinida de enemigos, por lo que es necesario crear un ciclo for el cual va a contener la cantidad de enemigos que se quieren generar. Ver figura 7



Figura

Este código genera valores aleatorios en los cuales se van a generar los enemigos , y velocidades aleatorias para el movimiento del mismo. Despues de generar los valores anteriores, los enemigos se guardan en una lista vacía llamada movingenemy, la cual va a ser la encargada de guardar los datos y evitar que estos se pierdan.

Una vez establecidos los códigos de generación y movimiento de los enemigos, solo falta ejecutarlos, los mismos se ejecutan por medio de las siguientes líneas de código. Ver figura 8

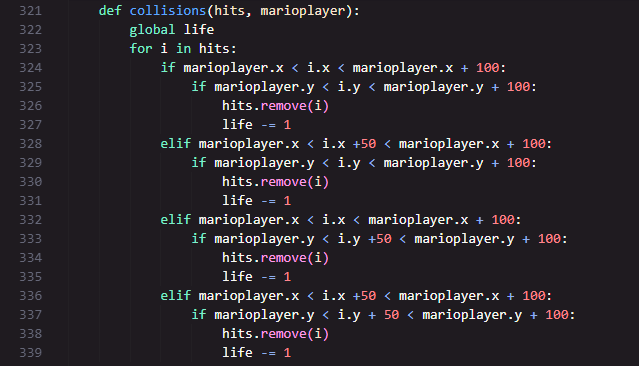


Figura

Dichas líneas de código presentes en la figura 8, se tienen que colocar en el bucle que genera los elementos de la pantalla.

**Colisión**

Una vez tenemos generados los enemigos, sus respectivos movimientos y al jugador con su respectivo movimiento, es necesario realizar la colisión entre el jugador y los enemigos. Para llevar a cabo la colisión de los mismos es necesario realizar el siguiente código. Ver figura 9



Figura

Explicando de manera resumida cómo funciona el código de la figura 9, este recibe 2 argumentos, hits y marioplayer, note que hits es equivalente a la lista que contiene a los enemigos. También es necesario llamar a la variable global de life, ya que esta nos funcionará para saber cuándo se detendrá el juego en caso de perder.

Una vez explicado lo anterior el funcionamiento del código se basa en tomar las coordenadas del jugador y en caso de ser menores a las coordenadas del enemigo y este a su vez es menor a las coordenadas del jugador +100, se removerá el enemigo de la lista y por lo tanto se borrará de la pantalla y a su vez se le resta -1 a la vida.

Para ejecutar el código anterior, se debe ingresar en el bucle que genera los elementos en la pantalla, se debe ingresar de la siguiente forma. Ver figura 10



10 Figura

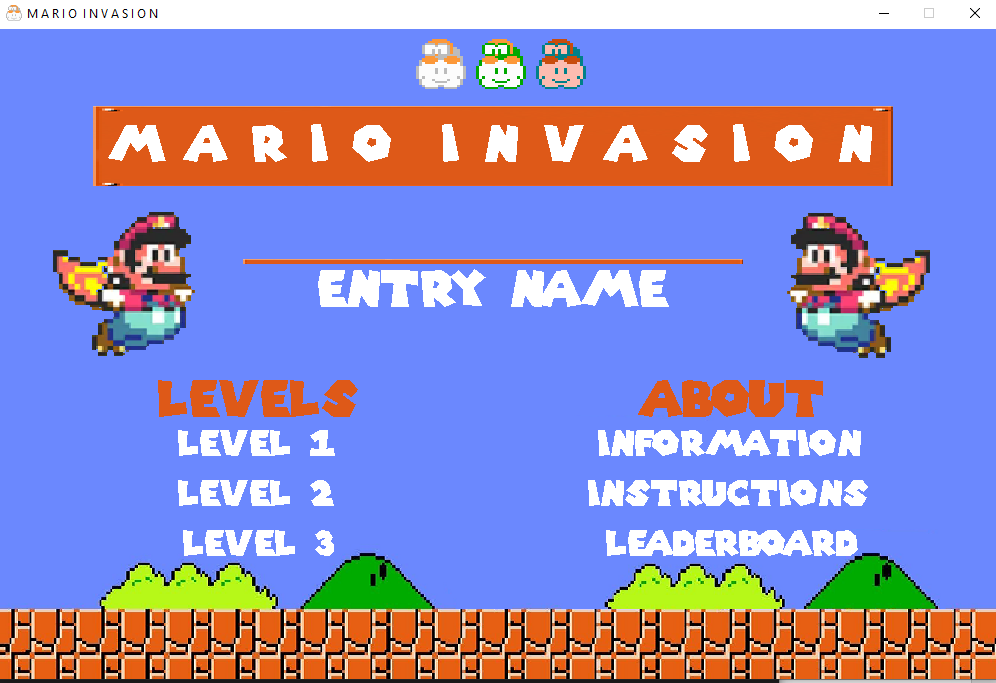
Todos los códigos anteriores se repiten tanto en el “leve1”, “level2” y en “level3”, cambiando únicamente la cantidad de enemigos que se genera en cada uno.

Quicksort

El Quicksort es una forma de ordenar listas de manera muy eficiente, por lo tanto, se utiliza el mismo para ordenar de mayor a menor los puntajes que obtienen los jugadores, es importante aclarar que únicamente se guardaran los primeros 7 puntajes.

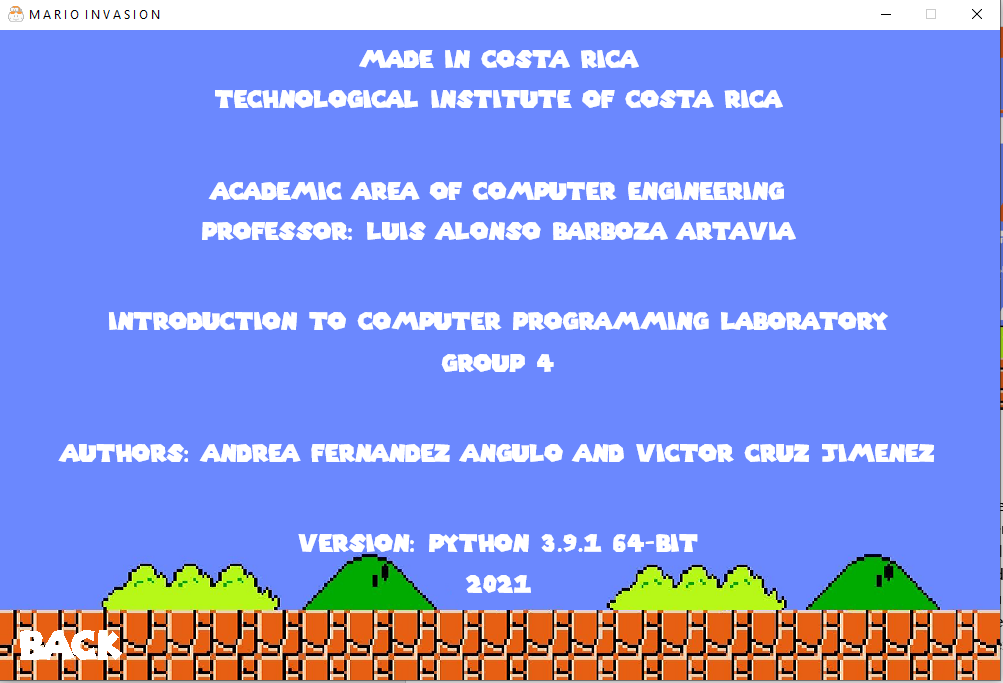
# **Resultados.**

Una vez elaborado el código base del juego se obtuvieron los siguientes resultados.



11 Figura

En la figura 11 se aprecia la ventana principal, en la cual se encuentran una entrada de texto, la cual sirve para ingresar el nombre del jugador. A su vez se encuentran botones de “information”, para conocer un poco mas del origen del producto y sus desarrolladores, “Instructions” en el cual al momento de oprimirlo se dirija a la ventana de instrucciones del juego, el botón del “leaderboard” el cual redirige a una ventana con los máximos puntajes obtenidos por los jugadores. También existen los botones “level 1”,” level 2”,” level 3” a los cuales solo se puede acceder cuando hay algún carácter ingresado en la entrada de texto.



12 Figura. Ventana de información

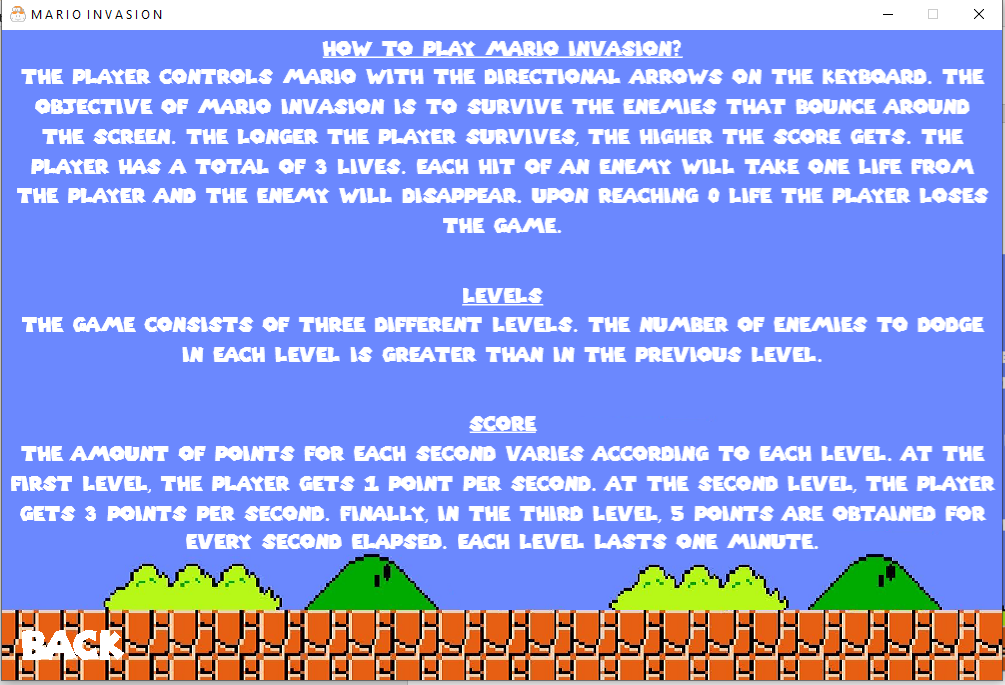
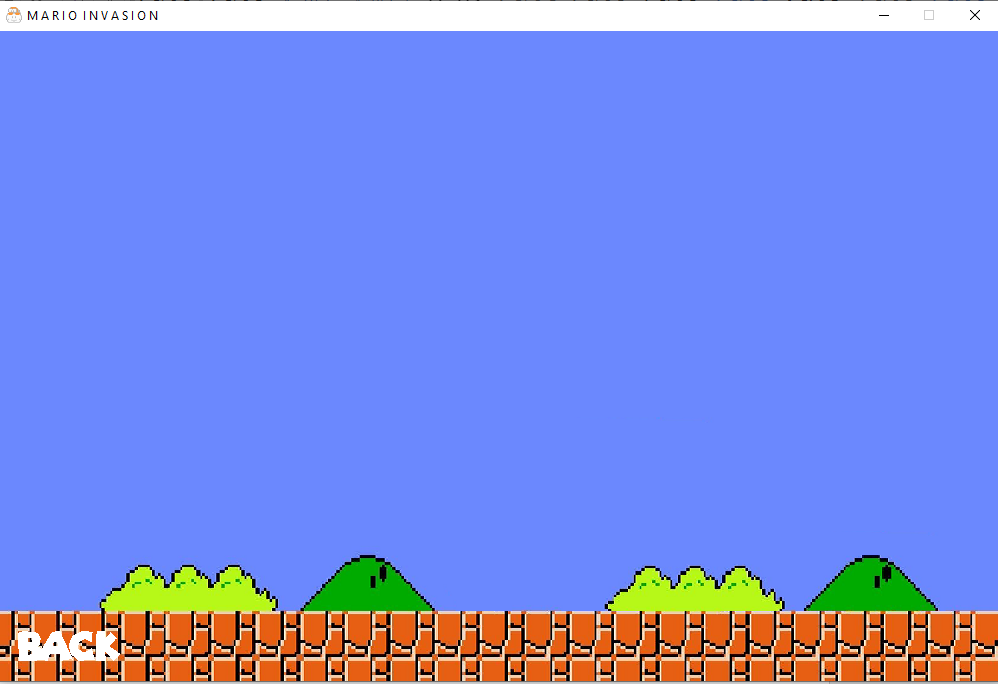


Figura. Ventana de instrucciones



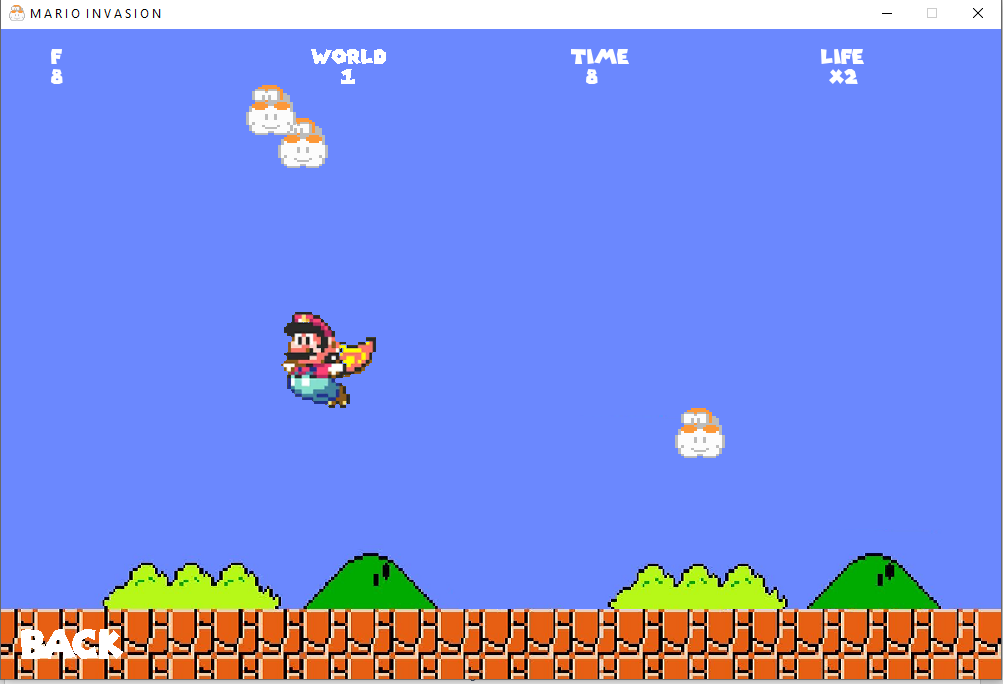
14 Figura. Ventana de mejores Puntuaciones

En el nivel 1 se encuentran el jugador y 4 enemigos, según lo planteado en las diferentes reuniones de grupo. Estos acompañados a su vez en la parte superior de la pantalla con el puntaje obtenido, el tiempo transcurrido, el nivel y cuántas vidas le quedan al jugador. Ver figura 15



15 Figura. Nivel 1

Cuando el jugador impacta con un enemigo, el enemigo se desaparece y la vida disminuye en -1. Ver figura 16



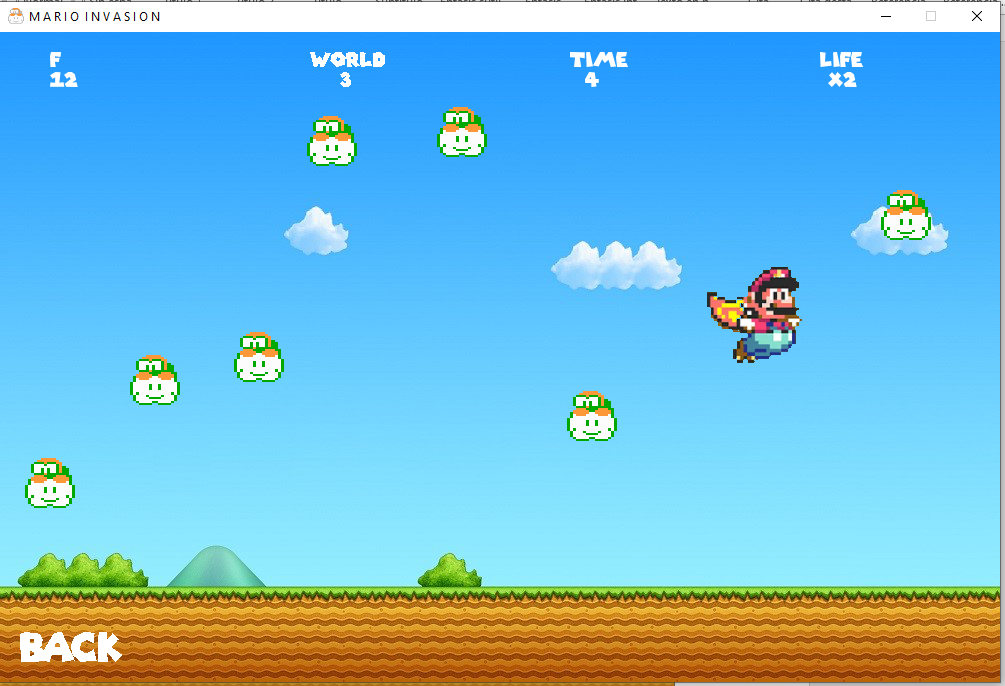
16 Figura

El nivel 2 cuenta con las mismas características del nivel 1, pero cuenta con un mayor numero de enemigos, 8 en total. Ver figura 17



17 Figura. Nivel 2

El nivel 2 cuenta con el mismo funcionamiento que el nivel 1. Al chocar con el enemigo, este se desaparece y la vida disminuye en -1. Ver figura 18

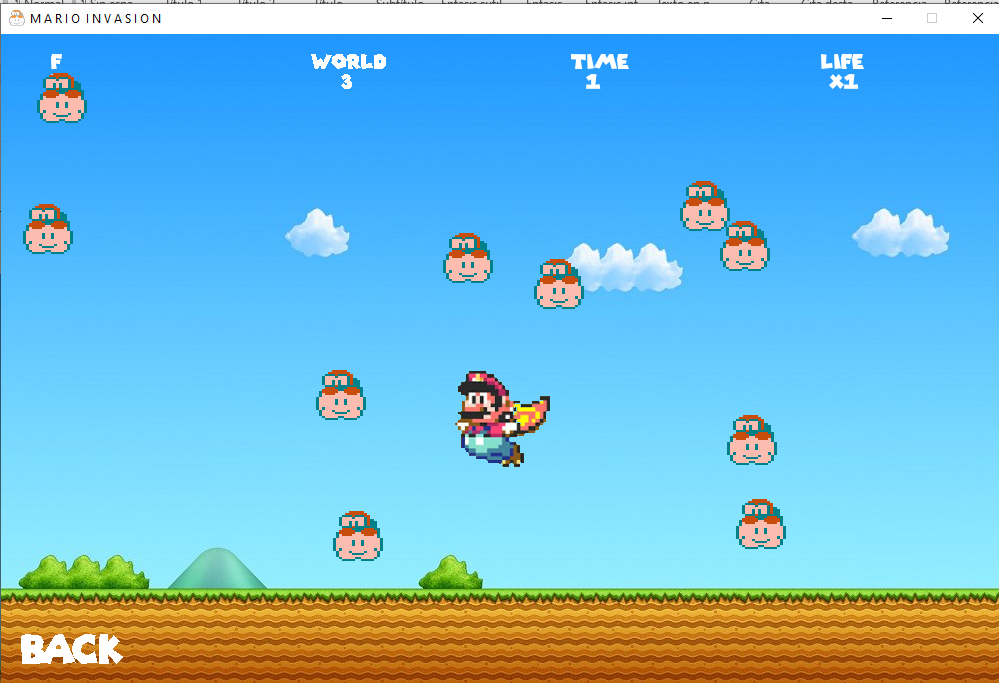


18 Figura

El nivel 3 es igual que los niveles anteriores, pero con un mayor numero de enemigos. Ver figura 19 y figura 20.



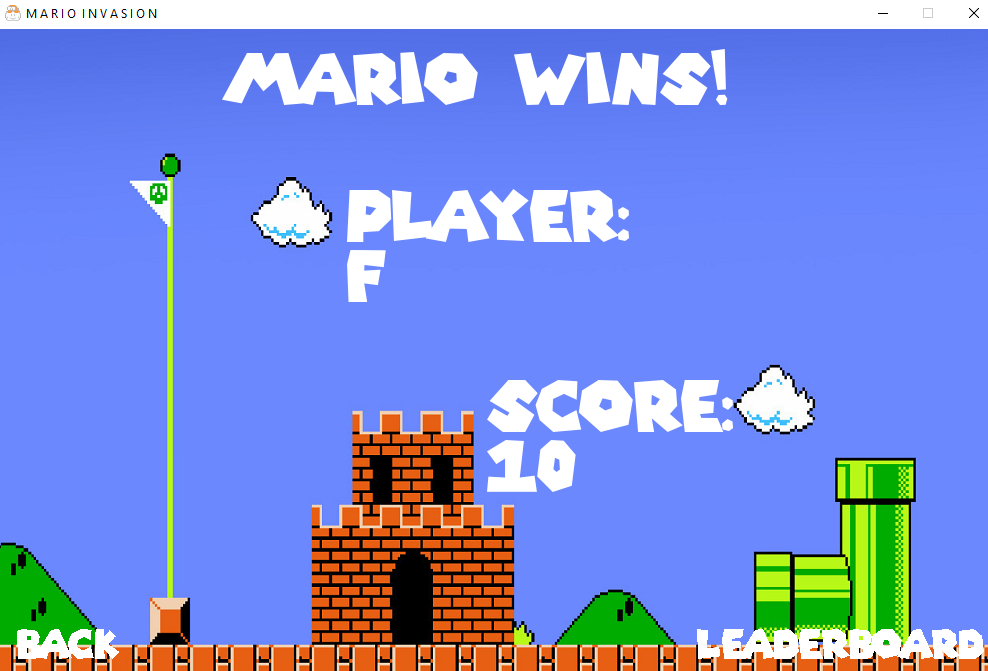
19 Figura. Nivel 3



Figura

En caso de llegar a 0 de vida en cualquier momento del juego, el jugador será enviado a la pantalla final del juego. Ver figura 21

En caso de completar el nivel 3 sin morir el jugador será enviado a la ventana final del juego, en el que aparecerá su nombre y la puntuación obtenida. Ver figura 22



22 Figura

Aquí se podrá regresar a al menú principal presionando el botón back o al leaderboard el cual contendrá las mejores puntuaciones.

# **Conclusiones.**

# **Recomendaciones**