**Proyecto 1**

**David Jafteh Obando Blanco - 2024157494**

**Curso: Introducción a la programación**

**Carrera: Ingeniería en Computadores**

**Profesor: Leonardo Andrés Araya Martínez**

## 1. Introducción

Este trabajado es el primer proyecto del curso de Introducción a la programación. El proyecto pretende familiarizar al estudiante usando la recursividad y la implementación de interfaces gráficas.

## 2. Descripción del problema

Para este proyecto no se permite el uso de bucles, más que el principal del juego, por lo que todas las características y problemas deben implementarse y resolverse por recursividad.

Para este proyecto tampoco se permite el uso de la programación orientada a objetos. Esto quiere decir que no se permite que el estudiante cree clases u objetos, sin embargo, el aprovechamiento de los objetos predefinidos en pygame y tkinter está permitido, pues estos representan el mayor valor de estas herramientas.

Ya establecidas las restricciones. El proyecto consiste en programar una réplica del juego clásico “Bomberman”.

El juego debe tener por lo menos 3 niveles y tipos de enemigos, el jugador debe tener la capacidad de colocar bombas para explotar a los enemigos y los muros colocados estratégicamente, dentro de los cuales podrá encontrar una llave escondida, la cual podrá utilizar para abrir la puerta en el mapa y avanzar al siguiente nivel. El jugador también debe ser capaz de recolectar puntos repartidos alrededor del mapa para eleva su puntuación.

Además de las pantallas de juego, el juego debe lanzarse en una pantalla de inicio, la cual presenta el título del juego y los botones de “Juego”, “Ajustes”, “Mejores puntuaciones” “Información”.

En la pantalla de ajustes se le debe permitir al jugador cambiar su nombre, su aspecto, y activar o desactivar la música de fondo.

En la pantalla de mejores puntuaciones se deben presentar las mejores 5 puntuaciones obtenidas y mostrando quien las obtuvo.

Finalmente, en la pantalla de información se deben presentar detalles pertinentes sobre el proyecto, el autor, el docente, la institución, la carrera, el curso, el año y la versión.

## 3. Análisis de resultados

Demostracion en video

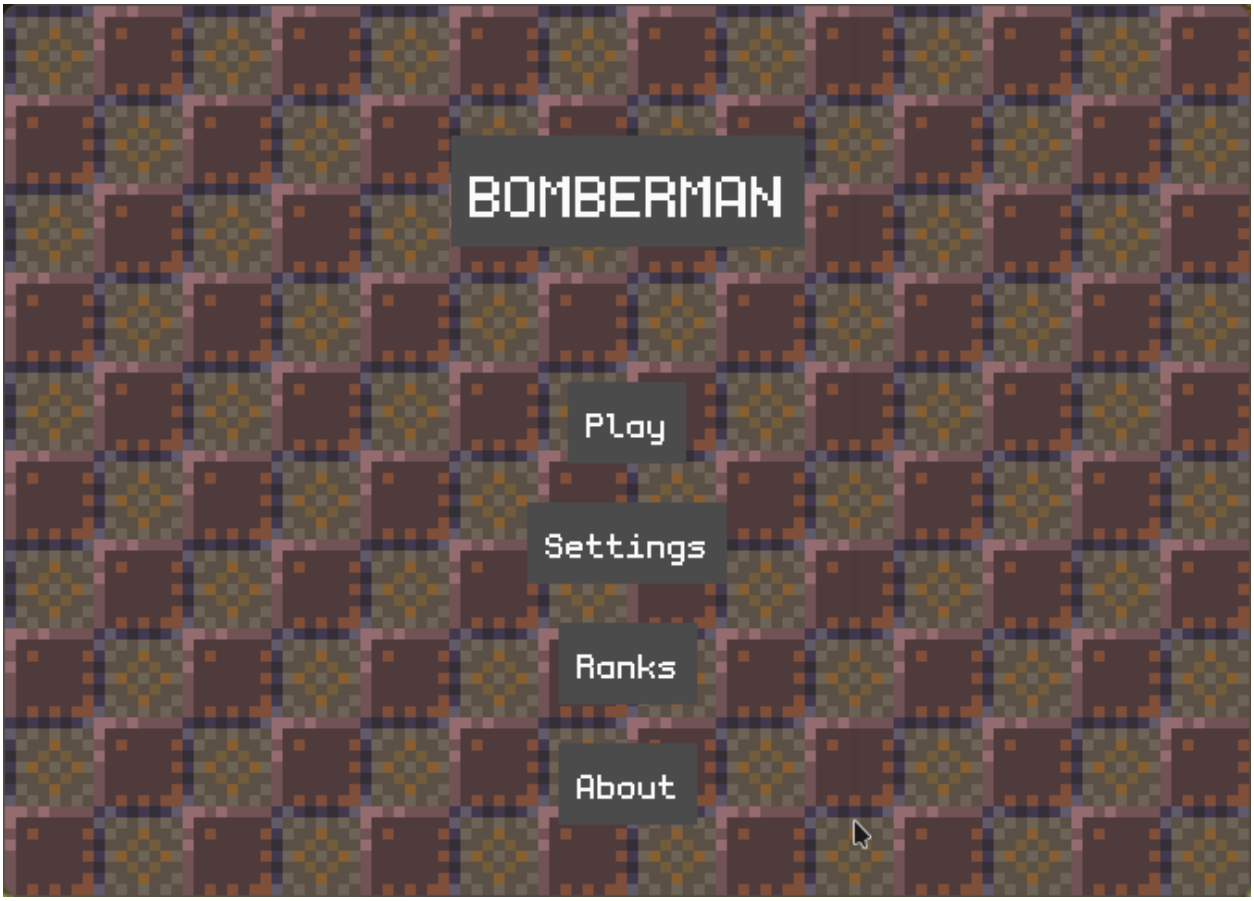


*Figura 1: Pantalla de inicio*

Se definen funciones para crear bloques de texto y botones interactivos, así como una función para generar fondos con una imagen como patrón. Estas funciones permiten una creación fácil y rápida de elementos visuales con texto, sin embargo, podrían generar problemas de rendimiento, pues generan rectángulos constantemente con cada iteración del bucle principal del juego

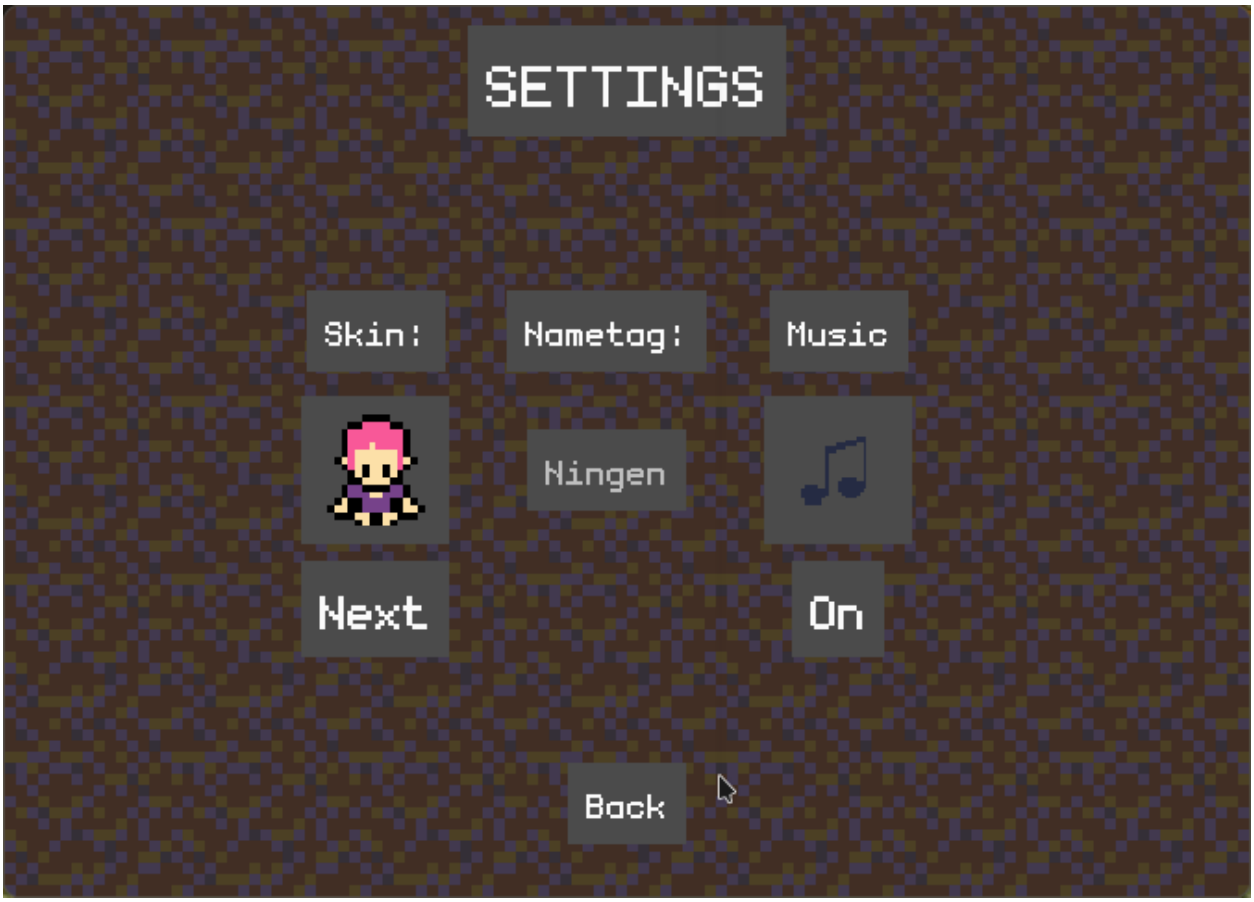


*Figura 2: Pantalla de ajustes*

Se define una función para crear bloques con imágenes y una función para crear campos de entrada de texto. Estas funciones permiten la introducción de elementos gráficos más complejos, pero padece de los mismos problemas que los bloques de texto.

*Figura 3: Pantalla de mejores puntuaciones*

Se define una función para acomodar la puntuación del jugador en la lista de mejores posiciones.


*Figura 4: Pantalla de información*.

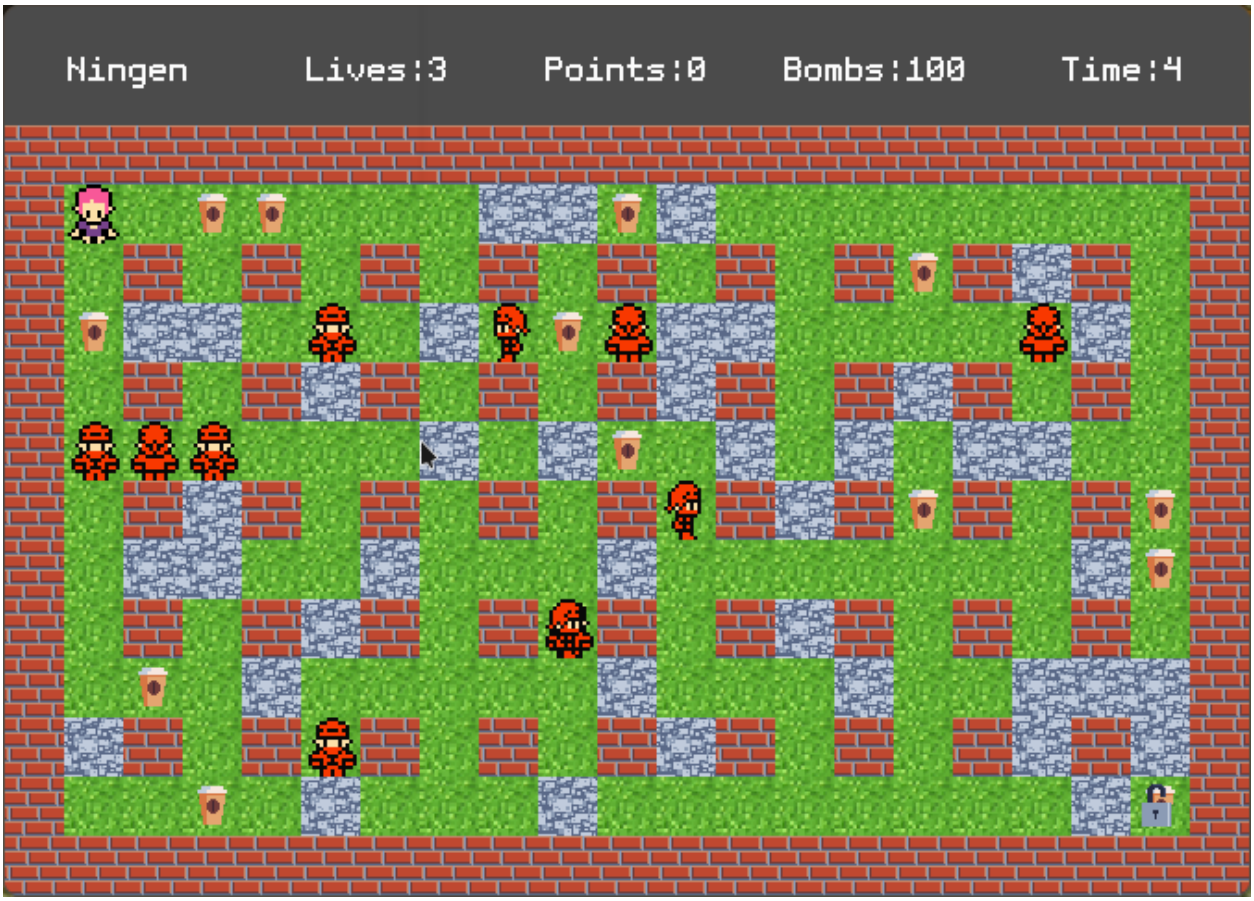

*Figura 5: Primer nivel*

Se crean funciones para dibujar el mapa preestablecido.

Se crean funciones para generar aleatoriamente y dibujar “scouts”, barreras destructibles, puntos (sprite de café), una llave escondida y una puerta (sprite de candado).

Todas estas funciones guardan los elementos creados en listas correspondientes. Estas funciones no sufren de los mismos problemas que las funciones de bloques visuales, pues estas se actualizan únicamente al iniciar el nivel o para mover a los enemigos en el caso de la función que mueve enemigos.

Se crea una función que identifica colisiones entre un rectángulo y cualquier otro dentro de la lista proporcionada. De esta manera se implementan las colisiones.

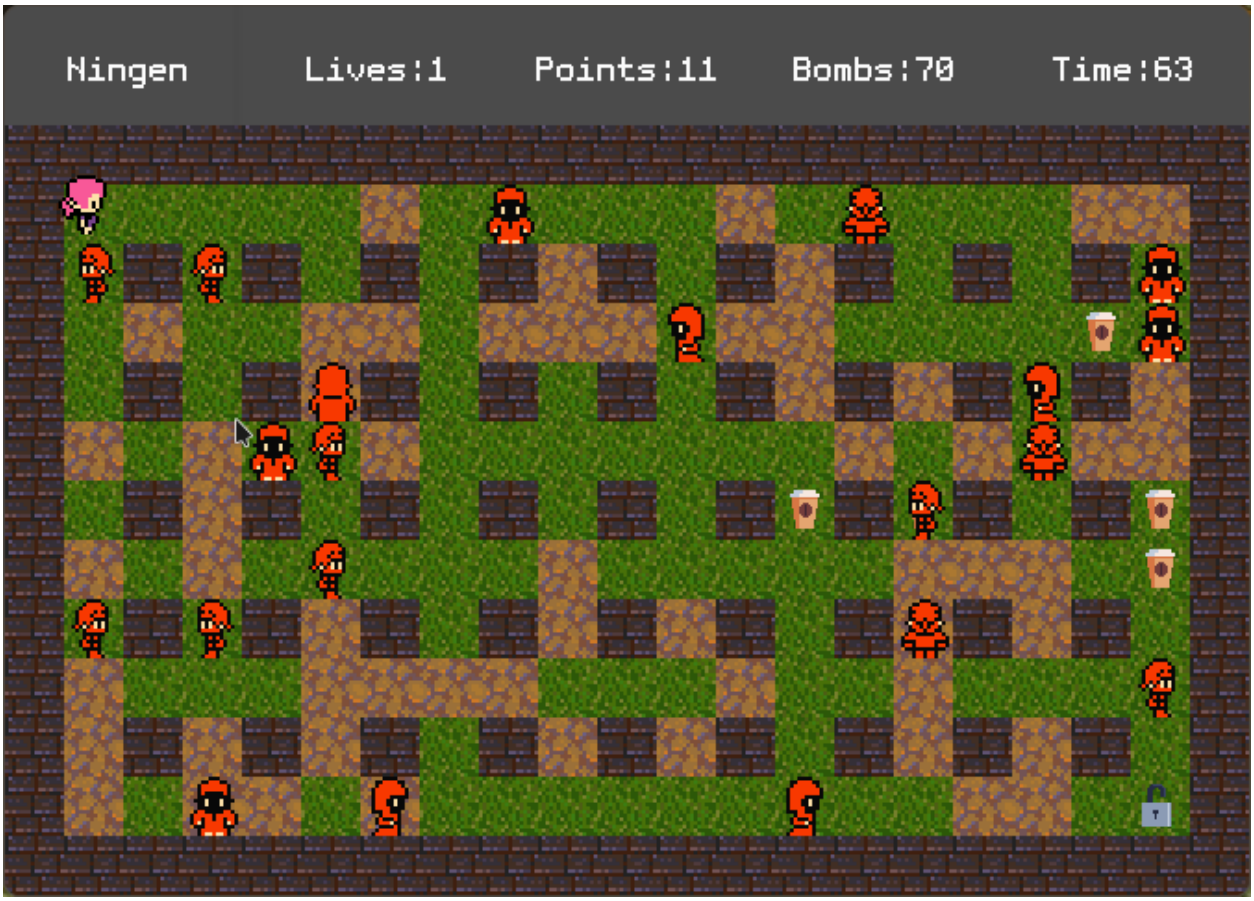

*Figura 6: Segundo nivel*

Se crean funciones que crean aleatoriamente, dibujan y mueven en el mapa a los enemigos tipos “conjurer”, los cuales pueden traspasar muros destructibles. Estas funciones siguen el mismo principio que las funciones para los “scout” y deberían tener un rendimiento y funcionalidad idénticos.

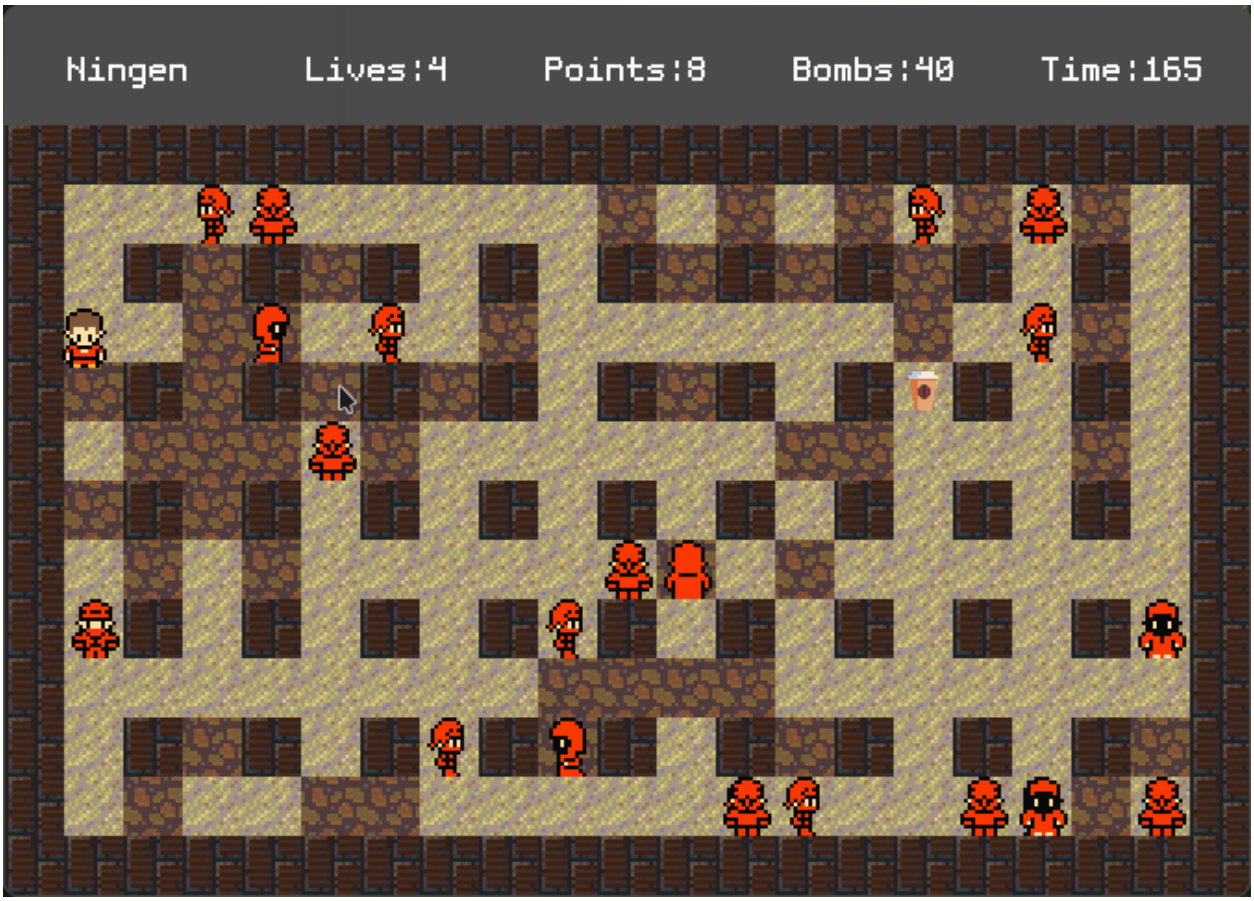

*Figura 7: Nivel 3*

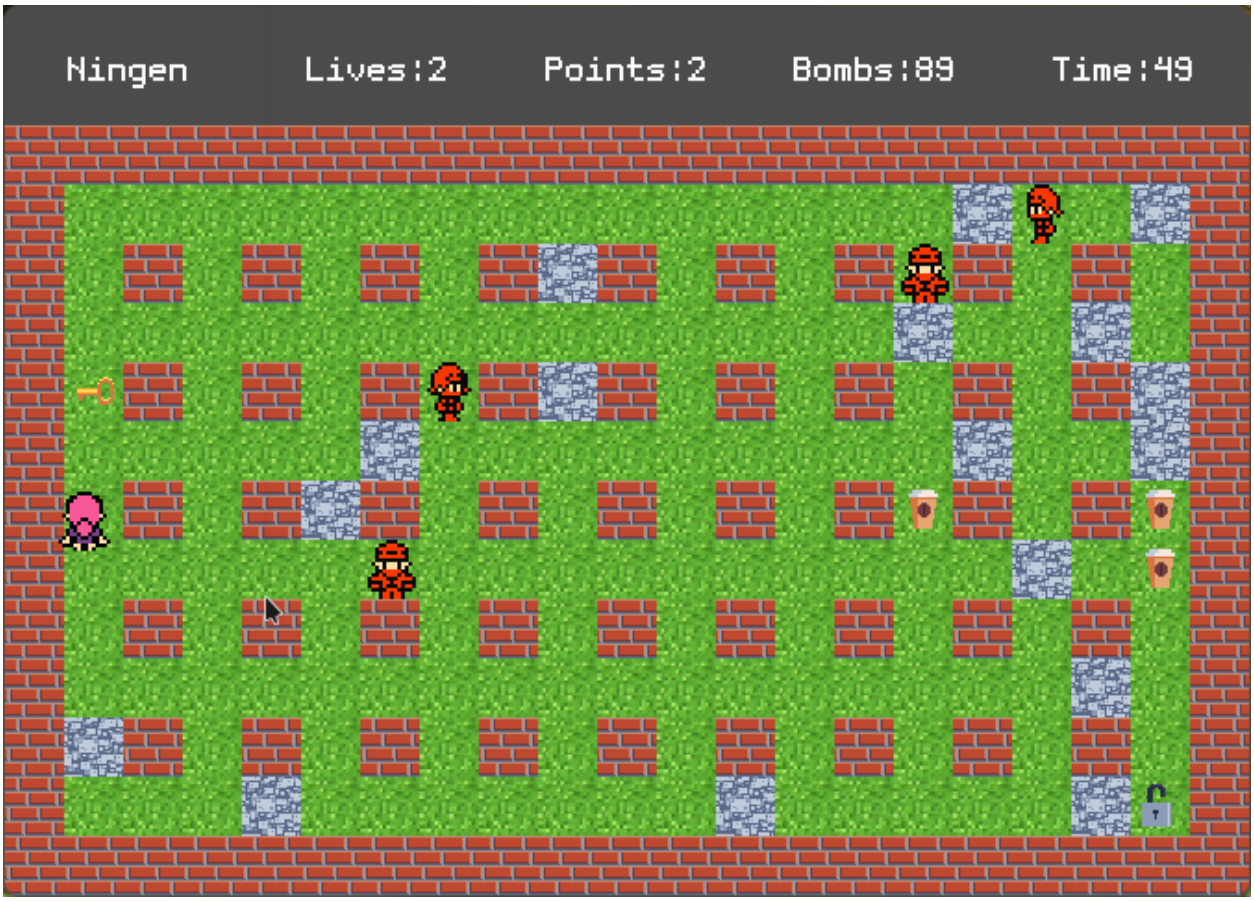

*Figura 8: Llave, barra superior y puntos*

Se crea una función que esconde una llave en un bloque aleatorio.

Se crean funciones que generan y dibujan los puntos.

Se crea una función que crea y dibuja la barra superior con todos los datos del juego recibidos. A diferencia de las otras funciones, esta podría generar problemas de rendimiento, pues sufre del mismo problema de la creación constante de rectángulos.

*Figura 9: Pantalla de derrota*

*Figura 10: Pantalla de victoria*

## 4. Dificultades encontradas

Unas de las implementaciones con mayor complicación fueron la animación de la bomba y el ordenamiento de las mejores posiciones.

Para la animación de la bomba, se establece el estado inicial de la animación al momento de eliminar la bomba tras haber pasado el tiempo de detonación. Este estado encadena la ejecución de un bloque que cada 200 milisegundos actualiza el estado de la explosión, cambiando los valores de sus variables y su imagen, permitiendo la observación de una explosión animada.

Para ordenar la puntuación del jugador en los “ranks”, se crea la función rank_points, la cual recibe la lista con los “ranks” actuales, la puntuación actual del jugador y su nombre.

Esta función mide el tamaño de la lista de “ranks” dada y la proporciona a su función auxiliar, la cual itera por cada elemento de la lista, comparando el valor anterior con la

puntuación actual y actualizando la posición de los “ranks” en caso de que la puntuación actual del jugador sea superior.

## 5. Bitácora de actividades

Sábado 6 de abril:

Se define la función “text_block”, la cual recibe un texto, tamaño, coordenadas, color de fondo, color del texto y un margen, y devuelve una tupla con la superficie y el rectángulo del bloque de texto resultante

Se define la función “button”, la cual recibe un texto, un tamaño, coordenadas, color de fondo, color del texto y un margen, y devuelve una tupla con la superficie, el rectángulo y valor booleano del botón resultante

Se define la fusión “img_load” la cual recibe un nombre de archivo, un directorio y un tamaño, y devuelve la imagen con el tamaño indicado

Se define la función “play_song” la cual recibe la dirección de un archivo de audio y lo reproduce indefinidamente

Se crea la pantalla de inicio, con música de fondo, el título del juego y los botones “Play”, “Settings”, “Rank”, y “About”, los cuales emiten un efecto de sonido al ser presionados

Duración de la sesión: Aproximadamente 3 horas.

Lunes 8 de abril:

Se define el tilemap en una matriz

Se define la función recursiva draw_tile_map, la cual recibe una matriz y dibuja el mapa correspondiente, guardando el rectángulo de cada bloque en una lista y devolviendo la lista final

Se define la función recursiva collideblock, la cual recibe un rectángulo y una lista de rectángulos, y comprueba si el rectángulo colisiona con alguno en la lista, devolviendo True o False

Se comienza la pantalla de juego. Se dibuja el mapa y el jugador. Se programa el movimiento del jugador considerando las colisiones con el mapa y los bloques

Duración de la sesión: Aproximadamente 2 horas.

Sábado 13 de abril:

Se define la función “img_block” la cual recibe una imagen, un tamaño, coordenadas, color de fondo y un margen, y devuelve la superficie y el rectángulo del bloque generado.

Se define la funcion “text_input” la cual recibe texto, tamaño, coordenadas, el estado booleano de recibiendo input o no, color de fondo, color de texto y un margen, y devuelve un bloque con el texto recibido, el cual se oscurece si se pone el botón sobre él, y se oscurece aún más si el bloque es presionado, activando el estado de input.

Se define la función “draw_bg” la cual recibe una imagen, un tamaño de azulejo y una superficie, y llena la superficie con un patrón de la imagen dada con el tamaño dado.

Se implementa la pantalla de configuración añadiendo ajustes de skin, nombre del jugador y la habilidad de activar o desactivar la música de fondo.

Duración de la sesión: Aproximadamente 4 horas.

Domingo 14 de abril

Se define la función “draw_top_bar” la cual recibe altura, ancho, cantidad de vidas del jugador, nombre del jugador, tiempo transcurrido y una superficie, y dibuja en la superficie una barra superior con los datos correspondientes.

Se define la función “add_barriers” la cual recibe un mapa de azulejos, una superficie y una probabilidad, genera aleatoriamente los rectángulos de los bloques destructibles y los devuelve en una lista.

Se define la función “draw_barriers” la cual recibe la lista de bloques destructibles y una superficie, y dibuja todos los bloques en la lista en la superficie dada.

Se consideran los bloques destructibles en la implementación previa de las colisiones.

Se define la función “load_skin” la cual recibe una skin y actualiza las variables globales de todos los sprites del jugador para la skin seleccionada.

Se define la función “add_bombs” la cual recibe una lista de bombas (inicializada como lista vacia) y una superficie, y añade una bomba (rect) a la lista con la posición actual del jugador.

Se define la función “draw_bombs” la cual recibe la lista de bombas y una superficie, y dibuja cada bomba en la superficie dada.

Se implementa la capacidad del jugador para colocar bombas.

Duración de la sesión: Aproximadamente 5 horas.

Viernes 19 de abril:

Se implementa la desaparición de las bombas al haber transcurrido un tiempo definido.

Se implementa la animación de explosión al desaparecer las bombas.

Se define la función “add_key” la cual recibe la lista de bloques destructibles y aleatoriamente asocia una llave a uno de los bloques destructibles, devolviendo el índice del bloque seleccionado en la lista de bloques, el rectángulo de la llave, un valor falso de llave encontrada.

Se define la función “destroy_barriers” la cual recibe el rectángulo de una explosión, la lista de barreras, el índice de la llave y el valor de llave encontrada. La función elimina los bloques destructibles que colisionen con el rectángulo de la explosión y actualiza el valor de llave encontrada.

Se implementa la puerta, eliminando el ultimo bloque de la lista de bloques destructibles y reemplazándolo por la puerta.

Se implementa la funcionalidad de la llave, dibujándola en caso haber sido encontrada y eliminándola en caso de haber colisionado con el jugador.

Se implementa la progresión de niveles junto a la funcionalidad de la puerta, modificando el escenario en caso de que el jugador tenga la llave y este colisione con la puerta.

Duración de la sesión: Aproximadamente 4 horas.

Sábado 20 de abril:

Se implementa una pantalla de introducción a cada nivel.

Se modifican los fondos de las pantallas de inicio, configuración e introducción.

Se definen las funciones add_scouts y add_conjurers, las cuales reciben un mapa de baldosas, una lista de bloques y una probabilidad de aparición, y devuelven una lista de enemigos de los tipos scout y conjurer respectivamente.

Se definen las funciones draw_scouts y draw_conjureres, las cuales reciben una lista de enemigos, una lista de imágenes y una superficie, y dibuja cada enemigo en la superficie con su imagen respectiva.

Se implementa la generación aleatoria de enemigos y su aparición en el mapa

Duración de la sesión: Aproximadamente 3 horas

Domingo 21 de abril:

Se definen las funciones move_scouts y move_conjurers, las cuales reciben una velocidad de movimiento, una lista de enemigos del tipo respectivo y una lista de bloques, devuelven la lista de enemigos modificada para implementar un movimiento aleatorio, y modifican las listas de imágenes de los enemigos respectivos, asignándole a cada enemigo la imagen correspondiente a su último movimiento.

Se define la función destroy_enemies, la cual recibe el rectángulo de una explosión y una lista de enemigos, eliminando a los enemigos que se encuentren en el rango de la explosión.

Se implementa el movimiento aleatorio de los enemigos, así como su vulnerabilidad a las explosiones.

Se implementa la vulnerabilidad del jugador a las explosiones y al contacto con los enemigos.

Se define la función add_points, la cual recibe un mapa de baldosas y una probabilidad de aparición, y genera aleatoriamente puntos recolectables dispersos por el mapa.

Se define la función draw_point, la cual recibe una lista de puntos y una superficie, y dibuja en la superficie cada uno de los puntos.

Se define la función rank_points, la cual recibe una lista de puntuaciones, la puntuación actual y el nombre del jugador, y devuelve la lista de mejores puntuaciones actualizada.

Se implementa la característica de puntos recolactables alrededor del mapa.

Duración de la sesión: Aproximadamente 5 horas

Lunes 22 de abril (Dia de entrega):

Se implementan la pantalla de mejores puntuaciones y la pantalla de información.

Se implementan las pantallas de victoria y derrota.

Se actualiza la función draw_top_bar para implementar los valores de puntaje.

Se añaden efectos de sonido y música de victoria y derrota.

Duración de la sesión: Aproximadamente 2 horas

## 6. Estadísticas de tiempo

| Sesión de trabajo | Fase principal | Tiempo |
|---|---|---|
| 6 de abril | Pantalla de inicio | 3 horas |
| 8 de abril | Primer nivel | 2 horas |
| 13 de abril | Pantalla de ajustes | 4 horas |
| 14 de abril | Primer nivel | 5 horas |
| 19 de abril | Primer nivel y progresión (segundo y tercer nivel) | 4 horas |
| 20 de abril | Progresión y enemigos | 3 horas |
| 21 de abril | Movimiento de enemigos, sistema de vidas y puntos | 5 horas |
| 22 de abril | Interfaz gráfica y detalles | 2 horas |

## 7. Conclusiones

- Durante el desarrollo de este proyecto, hubo un amplio desarrollo de las capacidades de solución de problemas, entendiendo y aprovechamiento de la recursividad, así como el desarrollo de interfaces graficas con el uso de pygame.
- En general, el proyecto tuvo un desarrollo decente, sin embargo, el apartado de la optimización y la implementación del movimiento de los enemigos pudo haber sido mucho mejor.

## 8. Literatura o fuentes consultadas


- Se utiliza Codeium para realizar consultas y evaluaciones, así como recibir sugerencias

- Se consultan los videos de YouTube "Get Started in Pygame in 10 minutes!", "HOW TO MAKE A MENU SCREEN IN PYGAME!" Y "PyGame Level Editor Using Tilemaps in Python - Tutorial | Part 1 - Initial Setup and Background"
- Se consulta la Documentacion de Pygame