EIE
Escuela de
Ingeniería Eléctrica

UNIVERSIDAD DE COSTA RICA

Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

IE-0117 Programación bajo plataformas abiertas

Proyecto Python - Juego de Buscaminas

Profesora: Carolina Trejos

Bryan Cortés Carné: C22422

Adrián Angulo Carné: C00465

Danny Solórzano Carné: C27667

Fecha: 5 de diciembre del 2023

Link del repositorio de GitHub

Juego de Buscaminas

1 Introducción

Los juegos de computadora han sido una parte muy importante del desarrollo y evolución de la tecnología informática. Entre ellos, el Buscaminas destaca como un clásico que ha entretenido a jugadores de todas las edades desde su lanzamiento. El objetivo principal del presente proyecto es hacer una implementación del Buscaminas en Python. Además, se resaltarán aspectos clave como el diseño de clases, la lógica del juego y la gestión de una interfaz gráfica.

El Buscaminas es conocido por ser un juego con una gran simplicidad y complejidad estratégica al mismo tiempo. La mecánica básica consiste en lo siguiente. Se tiene una tablero en el que hay una serie de bombas colocadas aleatoriamente en varias casillas. El usuario debe revelar poco a poco las casillas evitando revelar una casilla que contenga una bomba. La forma que tiene para hacerlo es utilizando el número de bombas adyacentes a una casilla, el cual aparece encima de todas las casillas sin bomba que han sido reveladas. La versión implementada en Python no solo busca replicar esta experiencia clásica, sino también proporcionar una visión práctica de los principios de Programación Orientada a Objetos y la manipulación de eventos en una interfaz gráfica.

Además, se destacarán los desafíos enfrentados y las soluciones implementadas para lograr una mejor experiencia de juego. En resumen, la idea principal es implementar el juego por medio de Programación Orientada a Objetos, según lo cual, se van a realizar diferentes clases para generar el tablero, las bombas, la interfaz, manejar eventos por medio de métodos, utilizar la aleatoriedad para generar diferentes juegos cada vez, entre muchas otras funciones.

2 Descripción del Programa

Este programa es una básicamente una implementación del juego de Buscaminas con una interfaz gráfica utilizando principalmente las funciones que brinda la biblioteca `pygame` de Python. A continuación, se presenta un diagrama que explica el funcionamiento básico del código:

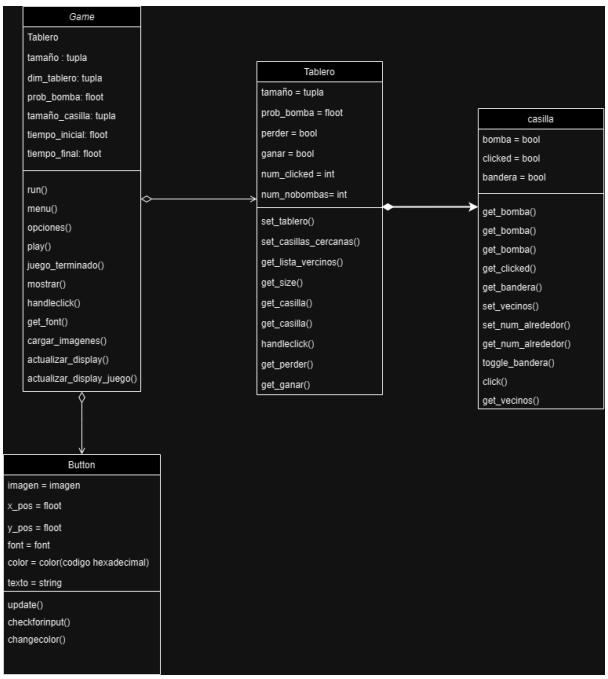

Figure 1: Diagrama del funcionamiento del código.

La idea principal es que el usuario pueda ingresar al menú principal, seleccionar la configuración del juego y generar el tablero con el cual va a jugar. Luego, empezará a revelar casillas hasta que toque una bomba o termine de revelar todas la casillas y gane. En ambos casos se mostrará un mensaje en la pantalla. Como se puede observar en el diagrama anterior, el programa consta de cinco módulos que se describen a continuación:

2.1 `casilla.py`

El archivo `casilla.py` contiene la definición de la clase `Casilla`. Esta clase representa una casilla en el tablero del Buscaminas. Cada instancia de `Casilla` tiene atributos como la presencia de una bomba, si la casilla ha sido clicada, si tiene una bandera, el número de casillas vecinas con bombas, etc. Además, proporciona métodos para interactuar con las casillas, como hacer clic, colocar banderas, etc.

2.2 `boton.py`

El archivo `boton.py` define la clase `Button`. Esta clase representa un botón en la interfaz gráfica del juego. Los botones pueden tener imágenes, texto y colores que cambian al pasar el mouse sobre ellos. La clase `Button` proporciona métodos para actualizar y verificar la interacción del usuario con el botón.

2.3 `tablero.py`

En `tablero.py`, se encuentra la implementación de la clase `Tablero`. Esta clase se encarga de crear y gestionar el tablero del juego. Puede generar un tablero con bombas distribuidas aleatoriamente, establecer las casillas cercanas para cada casilla y manejar las interacciones del jugador, como hacer clic en una casilla.

2.4 `game.py`

El archivo `game.py` contiene la clase `Game`, que actúa como el controlador principal del juego. Esta clase utiliza instancias de las clases anteriores para gestionar la lógica del juego, la interfaz gráfica, y la interacción con el usuario. Proporciona métodos para ejecutar el juego, manejar eventos y mostrar mensajes de juego terminado.

2.5 `main.py`

Crea una instancia de la clase `Game` y utiliza sus métodos para iniciar el juego.

3 Bibliotecas utilizadas

3.1 `pygame`

Sobre la biblioteca:

- Es una biblioteca diseñada para facilitar el desarrollo de videojuegos y aplicaciones multimedia en Python.
- Proporciona un conjunto de módulos para gestionar gráficos, sonido, eventos del sistema y entrada del usuario.
- Permite la creación de ventanas, manipulación de imágenes, reproducción de sonido y detección de eventos, como clics de ratón y pulsaciones de teclas.

En el programa:

- En el juego, el módulo `pygame` se utiliza para crear la interfaz gráfica, gestionar eventos del usuario y actualizar la pantalla.
- Carga imágenes desde archivos para representar las diferentes casillas del juego y otros elementos gráficos.

Funciones utilizadas:

- `pygame.init()`: Inicializa el motor de Pygame. Se llama al principio del programa.
- `pygame.image.load()`: Carga una imagen desde un archivo.
- `pygame.transform.scale()`: Cambia el tamaño de una imagen.
- `pygame.event.get()`: Obtiene todos los eventos en la cola de eventos, como clics de ratón y pulsaciones de teclas.
- `pygame.mouse.get_pos()`: Obtiene la posición actual del ratón.
- `pygame.mouse.get_pressed()`: Obtiene el estado de los botones del ratón.
- `pygame.display.set_mode()`: Crea una ventana de visualización.
- `pygame.display.set_caption()`: Establece el título de la ventana.

3.2 `os`

Sobre la biblioteca:

- El módulo `os` proporciona funciones para interactuar con el sistema operativo subyacente.
- Permite realizar operaciones en el sistema de archivos, como navegar por directorios, crear rutas de archivos y manipular archivos y directorios.

En el programa:

- En el juego, el módulo `os` se utiliza para construir rutas de archivos al cargar imágenes.
- Ayuda a crear rutas completas a las imágenes almacenadas en el directorio `images`.

Funciones utilizadas:

- `os.path.join()`: Combina partes de rutas para formar una ruta completa.
- `os.listdir()`: Devuelve una lista de nombres de archivos en un directorio.

3.3 `random`

Sobre la biblioteca:

- El módulo `random` proporciona funciones para trabajar con números aleatorios.
- Permite generar valores aleatorios, mezclar secuencias y realizar otras operaciones basadas en azar.

En el programa:

- En el juego, el módulo `random` se utiliza para introducir aleatoriedad en la distribución de bombas en el tablero.
- En particular, se utiliza para decidir si una casilla específica contendrá una bomba o no, basándose en una probabilidad.

Funciones utilizadas:

- `random()`: Genera un número decimal aleatorio en el rango [0.0, 1.0).

Estas bibliotecas ofrecen funcionalidades clave para el desarrollo del juego, desde la creación de la interfaz gráfica y la manipulación de archivos hasta la introducción de elementos aleatorios para una experiencia de juego dinámica. En resumen, en el programa se usan desde la inicialización de la interfaz gráfica y la carga de imágenes hasta la generación de números aleatorios para la colocación de bombas en el tablero.

4 Funcionamiento detallado del código

4.1 Clase `Casilla`

La clase `Casilla` representa una casilla en el juego de buscaminas. A continuación, se detalla su funcionamiento:

Atributos

- `bomba` (`bool`): Indica si la casilla contiene una bomba.
- `clicked` (`bool`): Indica si la casilla ha sido seleccionada por el jugador.
- `bandera` (`bool`): Indica si se ha colocado una bandera en la casilla.
- `vecinos` (`list`): Lista de casillas vecinas a la actual.
- `num_alrededor` (`int`): Número de bombas alrededor de la casilla.

Métodos

- `get_bomba()`: Retorna el valor de la propiedad `bomba`.
- `get_clicked()`: Retorna el valor de la propiedad `clicked`.
- `get_bandera()`: Retorna el valor de la propiedad `bandera`.
- `set_vecinos(vecinos)`: Establece la lista de casillas vecinas y actualiza `num_alrededor`.
- `set_num_alrededor()`: Calcula el número de bombas alrededor de la casilla.
- `get_num_alrededor()`: Retorna el valor de la propiedad `num_alrededor`.
- `toggle_bandera()`: Cambia el estado de la propiedad `bandera`.
- `click()`: Marca la casilla como seleccionada por el jugador.
- `get_vecinos()`: Retorna la lista de casillas vecinas.

Inicialización (`__init__`)

El método `__init__` se llama al crear una nueva instancia de la clase. Recibe un parámetro `bomba`, que indica si la casilla contiene una bomba. Inicializa los atributos `bomba`, `clicked` y `bandera` con los valores proporcionados y establece los demás atributos en `None`.

Establecer Vecinos (`set_vecinos`)

El método `set_vecinos` toma una lista de casillas `vecinos` como parámetro. Establece la propiedad `vecinos` con esta lista y luego llama al método `set_num_alrededor` para actualizar el número de bombas alrededor de la casilla.

Calcular Número de Bombas Alrededor (`set_num_alrededor`)

Este método recorre la lista de casillas vecinas y cuenta cuántas de ellas tienen una bomba. El resultado se almacena en la propiedad `num_alrededor`.

4.2 Clase `Button`

La clase `Button` representa un botón en una interfaz gráfica. A continuación, se detalla su funcionamiento:

Atributos

- `image`: Superficie de la imagen del botón.
- `x_pos`: Posición horizontal del botón.
- `y_pos`: Posición vertical del botón.

- `font`: Fuente utilizada para el texto del botón.
- `base_color`: Color del texto cuando no se pasa el mouse sobre el botón.
- `hovering_color`: Color del texto cuando se pasa el mouse sobre el botón.
- `text_input`: Texto que se muestra en el botón.
- `text`: Superficie de texto renderizada con la fuente y el color actual.
- `rect`: Rectángulo que delimita el área del botón.
- `text_rect`: Rectángulo que delimita el área del texto en el botón.

Métodos

- `__init__(image, pos, text_input, font, base_color, hovering_color)`: Inicializa una instancia de `Button`.
- `update(screen)`: Actualiza la representación visual del botón en la pantalla.
- `checkForInput(position)`: Verifica si la posición dada está dentro del área del botón.
- `changeColor(position)`: Cambia el color del texto del botón si el mouse está sobre él.

Inicialización (`__init__`)

El método `__init__` se llama al crear una nueva instancia de la clase `Button`. Recibe varios parámetros, como la imagen del botón, la posición, el texto, la fuente y los colores. Inicializa los atributos correspondientes y crea rectángulos para el botón y su texto.

Actualización (`update`)

El método `update` actualiza la representación visual del botón en la pantalla. Si hay una imagen, la coloca en el área del botón. Luego, coloca el texto en la posición adecuada.

Verificación de Entrada (`checkForInput`)

El método `checkForInput` verifica si una posición dada está dentro del área del botón. Retorna `True` si la posición está dentro y `False` en caso contrario.

Cambio de Color (`changeColor`)

El método `changeColor` cambia el color del texto del botón si el mouse está sobre él. Si la posición del mouse está dentro del área del botón, actualiza el color del texto al color de resaltado (`hovering_color`); de lo contrario, utiliza el color base (`base_color`).

4.3 Clase `Tablero`

La clase `Tablero` representa el tablero del juego de buscaminas. A continuación, se detalla su funcionamiento:

Atributos

- `tamaño` (`tuple`): Tupla que indica las dimensiones del tablero (filas, columnas).
- `prob_bomba` (`float`): Probabilidad de que una casilla contenga una bomba.
- `perder` (`bool`): Indica si el jugador ha perdido el juego.
- `ganar` (`bool`): Indica si el jugador ha ganado el juego.
- `num_clicked` (`int`): Número de casillas clicadas por el jugador.
- `num_nobombas` (`int`): Número de casillas sin bomba en el tablero.
- `tablero` (`list`): Lista bidimensional que representa el tablero de casillas.

Métodos

- `__init__(tamaño, prob_bomba)`: Inicializa una instancia de `Tablero`.
- `set_tablero()`: Crea el tablero y coloca las bombas según la probabilidad dada.
- `set_casillas_cercanas()`: Asigna las casillas vecinas a cada casilla en el tablero.
- `get_lista_vecinos(index)`: Retorna la lista de casillas vecinas a la casilla en la posición dada.
- `get_size()`: Retorna las dimensiones del tablero.
- `get_casilla(index)`: Retorna la casilla en la posición dada.
- `handleclick(casilla, bandera)`: Maneja la interacción del jugador con una casilla.
- `get_perder()`: Retorna si el jugador ha perdido el juego.
- `get_ganar()`: Retorna si el jugador ha ganado el juego.

Inicialización (`__init__`)

El método `__init__` se llama al crear una nueva instancia de la clase `Tablero`. Recibe dos parámetros, `tamaño` y `prob_bomba`, que representan las dimensiones del tablero y la probabilidad de que una casilla contenga una bomba. Inicializa los atributos y llama al método `set_tablero` para crear el tablero.

Configuración del Tablero (`set_tablero`)

El método `set_tablero` crea el tablero iterando sobre las filas y columnas, colocando bombas en cada casilla según la probabilidad dada. Lleva un conteo de casillas sin bomba (`num_nobombas`). Después, llama a `set_casillas_cercanas` para asignar las casillas vecinas.

Asignación de Casillas Cercanas (`set_casillas_cercanas`)

El método `set_casillas_cercanas` asigna las casillas vecinas a cada casilla en el tablero llamando al método `set_vecinos` de la clase `Casilla`.

Obtención de Casillas Vecinas (`get_lista_vecinos`)

El método `get_lista_vecinos` retorna la lista de casillas vecinas a la casilla en la posición dada. Itera sobre las posiciones vecinas y verifica si están dentro del rango del tablero.

Obtención de Dimensiones (`get_size`)

El método `get_size` retorna las dimensiones del tablero.

Obtención de Casilla (`get_casilla`)

El método `get_casilla` retorna la casilla en la posición dada.

Manejo de Clicks (`handleclick`)

El método `handleclick` maneja la interacción del jugador con una casilla. Actualiza el estado del juego según la interacción, marcando casillas clicadas, colocando banderas y verificando si se ha ganado o perdido.

Verificación de Pérdida (`get_perder`)

El método `get_perder` retorna si el jugador ha perdido el juego.

Verificación de Ganancia (`get_ganar`)

El método `get_ganar` retorna si el jugador ha ganado el juego.

4.4 Clase `Game`

La clase `Game` es la clase principal que gestiona el juego de buscaminas. A continuación, se presenta una explicación detallada de su funcionamiento.

Atributos

- `tablero`: Representa el tablero del juego. Se inicializa como `None` y se asigna más tarde con la función `Tablero()`.
- `tamaño`: Dimensiones de la ventana del juego.
- `size`: Dimensiones del tablero.
- `prob_bomba`: Probabilidad de que una casilla contenga una bomba.
- `tamaño_casilla`: Tamaño de cada casilla en píxeles.
- `screen`: Superficie de la ventana del juego. Se inicializa como `None` y se asigna más tarde con la función `actualizar_display()`.
- `images`: Diccionario que mapea nombres de imágenes a superficies.

Métodos

Inicialización (`__init__`)

Este método se encarga de inicializar una instancia de la clase `Game`. Establece los atributos iniciales, como `tablero` en `None`, y calcula el `tamaño_casilla` dividiendo el tamaño de la ventana por las dimensiones del tablero.

Correr juego (`run`)

Este método inicia el juego de buscaminas. Primero, inicializa Pygame y luego muestra el menú principal llamando al método `menu()`.

Menú del juego (`menu`)

Este método muestra el menú principal del juego. Utiliza la posición del mouse para determinar las interacciones del jugador con los botones del menú.

Opciones del juego (`opciones`)

Este método muestra la pantalla de opciones, donde el jugador puede seleccionar niveles de dificultad y volver al menú principal.

Iniciar juego (`play`)

Este método inicia el juego. Gestiona eventos mientras se juega, mostrando el tablero y terminando el juego si es necesario.

Juego terminado (`juego_terminado`)

Este método muestra la pantalla de juego terminado, permitiendo al jugador volver a jugar o salir del juego después de ganar o perder.

Mostrar tablero (`mostrar`)

Este método muestra el tablero en la ventana del juego.

Clic del mouse (`handleclick`)

Este método maneja el clic del mouse en una casilla del tablero, actualizando el estado del tablero según la interacción del jugador.

Obtener fuente (`get_font`)

Este método retorna una fuente de Pygame con el tamaño dado.

Cargar imágenes `(cargar_imagenes)`

Este método carga las imágenes necesarias para el juego, escalándolas al tamaño de las casillas del tablero.

Imagen casilla `(get_imagen)`

Este método retorna la imagen correspondiente a una casilla del tablero.

Actualizar mensaje `(actualizar_display)`

Este método actualiza la superficie de la ventana del juego con el mensaje proporcionado.

Actualizar ventana `(actualizar_display_juego)`

Este método actualiza la superficie de la ventana del juego según las dimensiones del tablero.

5 Ejemplo de uso

Primero, debe instalar las dependencias necesarias, esto se puede hacer ingresando lo siguiente en la línea de comandos:

```
pip install pygame
```

Luego, debe ejecutar el módulo principal, puede hacer ingresando lo siguiente en la línea de comandos:

```
python3 main.py
```

Entonces, se mostrará el menú inicial que brinda las opciones de "Jugar", "Opciones" (dificultad) y "Quit" (salir). Se vería como la siguiente pantalla:

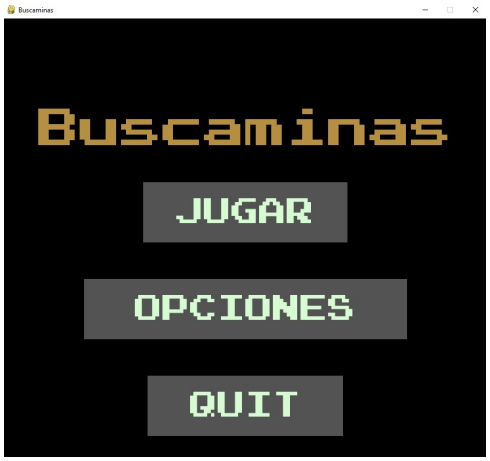

Figure 2: Menú principal.

Lo ideal es primero entrar al menú de opciones para ajustar la dificultad. Las dificultades se miden por la cantidad de casillas en el tablero, donde "fácil" es un tablero 9x9, "medio" es un tablero 10x10 y "difícil" es un tablero 12x12. En caso de no elegir una opción, la dificultad predeterminada es la "fácil". Los tres tableros se verían como los siguientes:

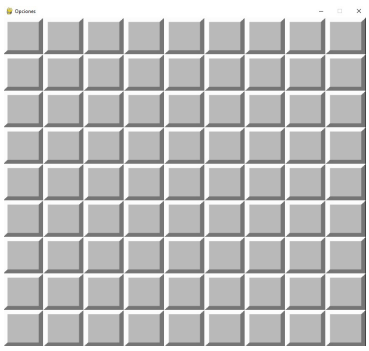

Figure 3: Tablero nivel fácil.

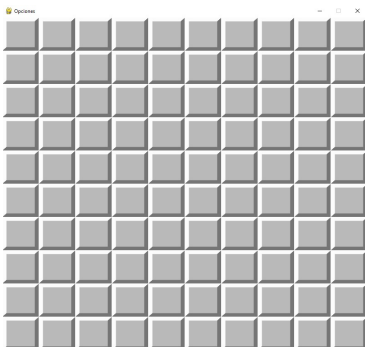

Figure 4: Tablero nivel medio.

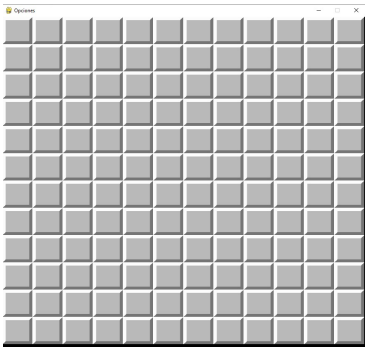

Figure 5: Tablero nivel difícil.

Ahora bien, una vez que haya elegido una dificultad (o haya elegido al inicio la opción de "Jugar" con el modo predeterminado como "fácil") se desplegará el tablero. Las instrucciones del juego son las siguientes:

- En cada paso deberá elegir una casilla para revelarla con clic izquierdo.
- Si no es una bomba, en la casilla se revelará un número.
- El número indica la cantidad de casillas adyacentes a la casilla seleccionada que tienen una bomba.
- Con esta información para cada casilla revelada deberá deducir en dónde están las bombas para evitar revelarlas.
- Si está seguro de que una casilla debe contener una bomba la puede marcar con una bandera con clic derecho.
- Si toca una bomba pierde el juego.
- Si logra revelar todas las casillas que no tienen bombas ganará el juego.

El inicio del juego se verá de la siguiente forma:

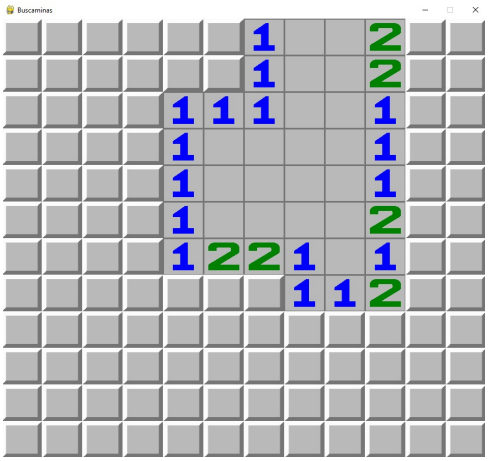

Figure 6: Inicio de una partida.

El siguiente es un ejemplo de cómo se vería el avance del juego con algunas banderas colocadas:

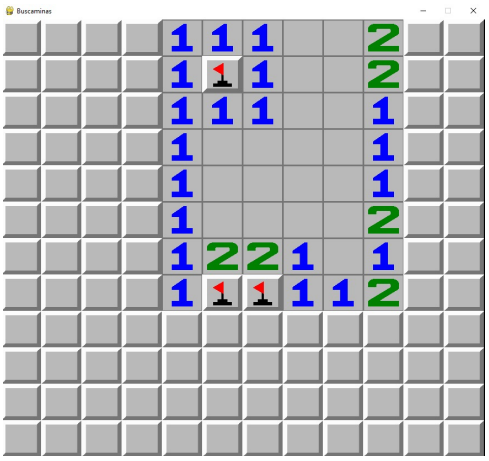

Figure 7: Avance de una partida.

Luego, si toca una bomba se verá un mensaje como el siguiente en pantalla:

Figure 8: El usuario ha perdido.

Si logra revelar todas las casillas que no contienen una bomba se verá el siguiente mensaje en pantalla:

Figure 9: El usuario ha ganado.

Una observación importante es que se incorporó una funcionalidad al juego según la cual cuando el usuario logra completar todo el tablero y ganar se le permite observar el tiempo que ha durado en completar el juego. Esto permite que el usuario pueda medir su habilidad y proponerse a superar su propio récord.

6 Limitaciones del programa

Por ahora, solo se ha encontrado una limitante considerable en el programa, esto es que la primera casilla que el usuario selecciona podría ser una mina. Esto es un problema pequeño, ya que no representa un error propiamente del código. No obstante, para el usuario puede ser un poco frustrante que el juego termine sin haber empezado en realidad. Por tanto, se podría trabajar en alguna forma de evitar que el usuario pueda seleccionar una mina al inicio. Esto estaría relacionado propiamente con la forma en la que se generan las posiciones de las minas al iniciar el juego, por tanto, sería necesario realizar algunas modificaciones a los métodos que se encargan de esto.

7 Trabajo por realizar

Aunque se podría decir que la implementación de este código cumple con los requisitos básicos que debe tener un juego de Buscaminas, todavía se le pueden realizar ciertas modificaciones o agregados para mejorar el juego.

7.1 Registros de usuarios

Un ejemplo muy concreto de esto es que el juego no permite guardar datos de los usuarios. Así, una versión mucho más trabajada de este código podría registrar a diferentes usuarios. Todo esto con la intención de guardar sus récords, es decir, los menores tiempos en los que cada usuario ha logrado resolver el tablero.

7.2 Dificultad del tablero

Por otra parte, otra mejora que se le puede hacer al juego es extender la función de dificultad del tablero. Por ejemplo, se podría hacer que el usuario elija específicamente la cantidad de minas que tiene el tablero junto con la cantidad de casillas, es decir, el tamaño del tablero. Por el momento, solo se está implementando esta función con tres niveles de dificultad, lo cual no está nada mal, pero se podría extender para darle más libertad y opciones al usuario.

7.3 Desbloqueo de minas

En general, hay otras muchas modificaciones que se pueden realizar, todo esto con el fin de perfeccionar el trabajo. Una que se relaciona con el juego mismo cuando está en ejecución es la forma en la que se desbloquean las casillas. Después de utilizar el juego por un tiempo es fácil notar que la manera en la que se revelan las casillas es un poco incómoda, ya que en algunas ocasiones el usuario es consciente de que una casilla ya no tiene minas a su alrededor y aún así debe revelarlas una por una. Una opción que se podría brindar es agregar un botón que permita revelar todas las casillas adyacentes a una casilla una vez que esta tenga a su alrededor una cantidad de banderas equivalente al número de minas que la rodean.

Finalmente, también se podrían mencionar aspectos como la mejora de la apariencia de la interfaz gráfica, pero ese aspecto es mucho más subjetivo.

8 Conclusiones

La implementación del juego de Buscaminas en Python ha sido un proyecto integral que ha proporcionado una valiosa experiencia de aprendizaje en diversas áreas de la programación y el desarrollo de software. En primer lugar, la representación de las casillas, la lógica del juego y la interacción con el usuario se han encapsulado eficazmente en clases bien definidas. La Programación Orientada a Objetos ha sido esencial para organizar y estructurar el código de manera clara y modular. Cada instancia de la clase `Casilla` y `Button` actúa de forma independiente, lo que facilita el mantenimiento y la futura expansión del programa.

Se pueden resaltar algunos aspectos importantes, como los siguientes. La distribución aleatoria de bombas se abordó con éxito mediante el uso del módulo `random` de Python. La lógica de juego, que incluye el manejo de clics, la actualización del tablero y los mensajes de ganador y perdedor resultó ser mucho más difícil de implementar de lo que se esperaba. El juego a simple vista parece muy sencillo, pero se deben considerar demasiados detalles y aspectos cuando se quiere implementar. La interfaz gráfica resultó bastante atractiva e incluso "retro". La capacidad de cambiar la dificultad del juego y la visualización del tiempo de juego fueron detalles que mejoraron mucho el resultado final.

Este proyecto presentó desafíos importantes, como gestionar eventos de clic del mouse y manejar la presentación de mensajes de juego terminado. Sin embargo, se pudieron superar exitosamente y completar los objetivos propuestos. En conclusión, la implementación del juego de Buscaminas en Python ha sido una experiencia muy educativa y divertida que sirvió para consolidar algunos conocimientos y adquirir otros nuevos.