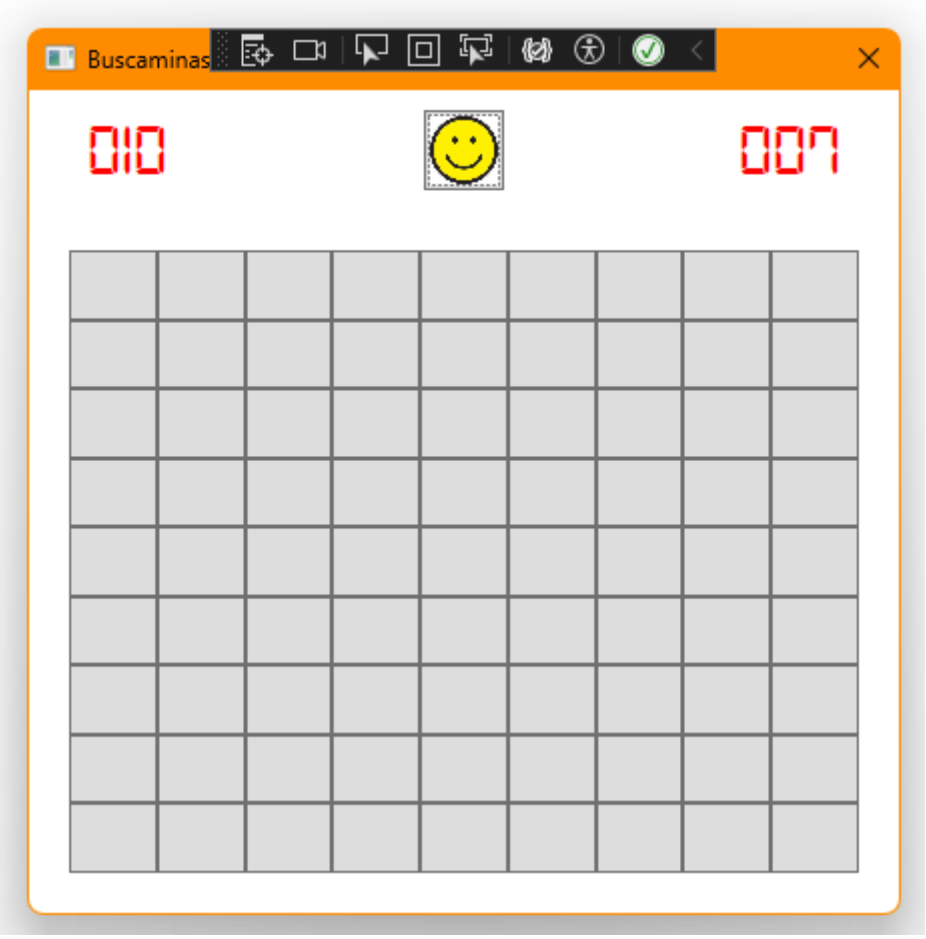
Buscaminas



Img.1 - Pantalla principal del Buscaminas al iniciar.

Índice

Índice de métodos (MainWindow.xaml.cs)

Métodos principales y auxiliares:

1. **public MainWindow():** Constructor de la clase MainWindow.
2. **private void Inicio():** Establece los valores iniciales de todos los atributos e inicia el cronómetro.
3. **private void CambiarIconoReinicio(String nombreImagen):** Cambia el icono del botón de reinicio a la imagen indicada.
4. **private void CrearTablero():** Crea el tablero de juego (matriz de botones).
5. **private void GenerarMinas():** Genera la matriz con las minas, insertándolas de manera aleatoria en el tablero.
6. **private int ContarMinasAdyacentes(int fila, int columna):** Cuenta las minas en las casillas vecinas y retorna el número.
7. **private void RevelarCasillasAdyacentes(int fila, int columna):** Revela las casillas adyacentes de manera recursiva.
8. **private Brush ObtenerColorDeNumero(int numero):** Obtiene el color según el número (devuelve un pincel).
9. **private void TerminarJuego(bool win):** Termina el juego y deshabilita todos los botones del tablero.
10. **private void RevelarMinas():** Revela todas las minas en el tablero.
11. **private void ActualizarMinasRestantes():** Actualiza las minas restantes en el contador.

Métodos de eventos:

12. **private void CotonReinicio_Click(object sender, RoutedEventArgs e):** Acción que se realiza al hacer clic en el botón de reinicio.
13. **private void Casilla_ClickDerecho(object sender, MouseButtonEventArgs e):** Acción que se realiza al hacer clic derecho sobre un botón (casilla) del tablero.
14. **private void Casilla_ClickIzquierdo(object sender, RoutedEventArgs e):** Acción que se realiza al hacer clic izquierdo sobre un botón (casilla) del tablero.
15. **private void Actualizar_Cronometro(object sender, EventArgs e):** Actualiza el contenido del cronómetro.

Índice de componentes (MainWindow.xaml)

1. **Window:** Ventana principal de la aplicación. Contiene el resto de elementos.

    1. **FontFamily (Window.Resources):** Fuente personalizada que hace referencia al recuso con “Digital-7”, que no es más que una fuente para simular efecto digital en los contadores.

2. **Grid:** Contenedor principal que contiene y organiza todos los elementos en filas.

    1. **RowDefinitions(Grid):** Define dos filas en el Grid, una con altura fija (60) que contendrá los dos contadores (Label) y el botón de reinicio (Button) y otra que ocupa el resto del espacio (*) que contendrá el tablero (UniformGrid).

3. **Label (lblCronometro):** Cronómetro digital que cuenta el tiempo.

4. **Label (lblMinas):** Contador de minas que informa de las minas restantes.

5. **Button(botonReinicio):** Botón de reinicio con un evento de clic asociado → botonReinicio_Click, que reinicia la partida.

    1. **Image (iconoReinicio):** Imagen dentro del botón de reinicio. Es la forma de establecer una imagen como fondo de un botón. Y se actualiza mediante eventos: TerminarJuego() e Inicio().

6. **UniformGrid (gameGrid):** Tablero de juego con una cuadrícula sin definir tamaño, lo que permite asignar un tamaño de forma dinámica (mediante código), que en este caso será 9x9. (Aunque no es accesible desde el cuadro de herramientas, forma parte de los componentes de WPF y se puede añadir mediante código).

Funcionamiento y diseño

La aplicación consta de los elementos descritos anteriormente en cuanto a funcionamiento y diseño. El funcionamiento es el del juego típico del Buscaminas con las reglas básicas del Buscaminas. Y en cuanto a diseño, se ha intentado alcanzar el mayor grado de similitud con el original.

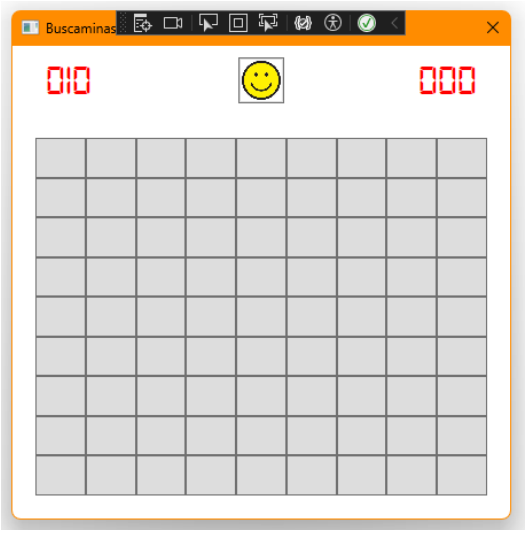
Funcionamiento

Resumen del juego

El jugador debe revelar todas las casillas sin pisar ninguna mina (ganar) o pisar una mina para que el juego termine (perder). El jugador puede colocar banderas donde considere que hay una mina (click derecho), puede revelar una casilla (click izquierdo) o puede reiniciar la partida cuando considere (botón de reinicio).

Lógica y funcionamiento interno

La aplicación arranca con el constructor de **MainWindow()** en el que se lanzan los métodos **InitializeComponent()**, que viene por defecto, e **Inicio()**, que dará valores iniciales a todas las variables y componentes del juego, como por ejemplo el cronómetro o el icono del botón de reinicio.



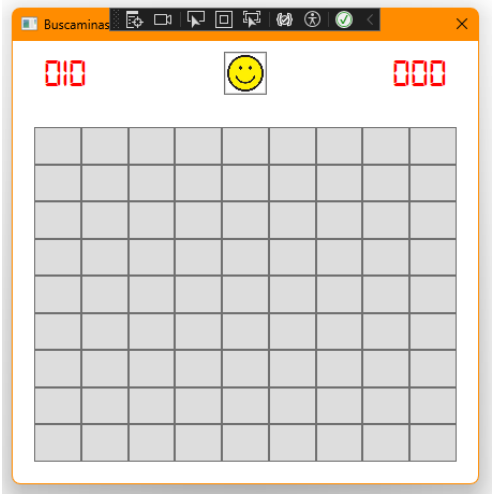
Img. 2 - Pantalla inicial. (Minas 010, Tiempo: 000)

Dentro de inicio se lanzan también los métodos **CrearTablero()**, que creará el tablero de juego (matriz de botones), y **GenerarMinas()**, que generará de forma aleatoria las minas y las insertará en una matriz de minas (bool [,]).
Básicamente tenemos dos matrices: una de botones que representa las casillas del tablero, y una de Booleanos, que representa las minas en dicho tablero.

Una vez iniciado el juego y con todos los componentes activos, los eventos **BotonReinicio_Click**, **Casilla_ClickDerecho()**, **Casilla_ClickIzquierdo()** y **Actualizar_Cronometro()**, serán los encargados de actualizar la UI según corresponda. Esto es:

- **BotonReinicio_Click()**: Este evento se acciona al pulsar el botón de reinicio (botonReinicio) y es el encargado de desactivar el cronómetro (realmente desactiva el DispatcherTimer que controla el cronómetro) y llamar de nuevo al método Inicio().

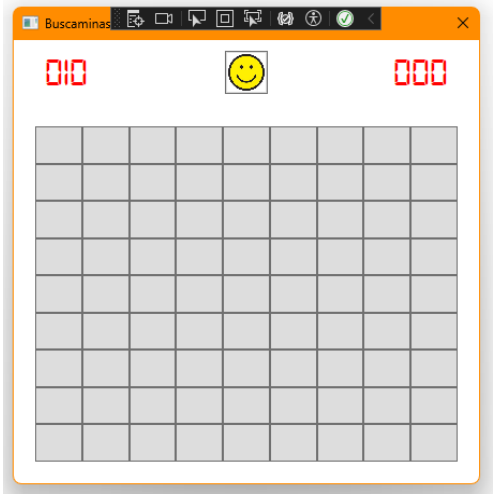
Img. 3 - Antes de pulsar reinicio.



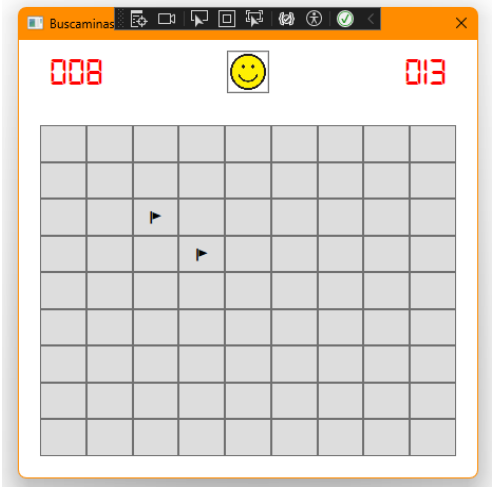
Img. 4 - Tras pulsar reinicio.

- **Casilla_ClickDerecho()**: El evento se acciona al hacer click derecho sobre una casilla del tablero. Permite al usuario poner y quitar banderas (⚑) para indicar que en la casilla hay una mina (suposición del usuario). También permite recoger banderas colocadas.

  Colocar bandera:

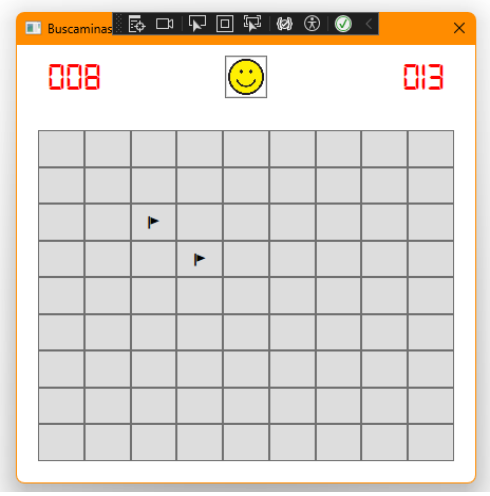


Img. 5 - Antes del click derecho.

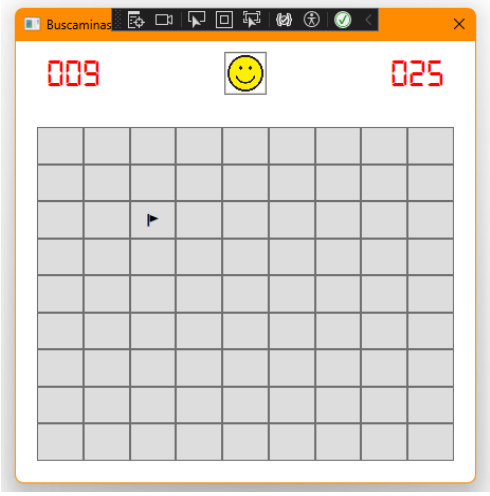


Img. 6 - Tras dos clicks derechos.

Retirar bandera:



Img. 7 - Antes del click derecho.



Img. 8 - Tras el click derecho.

- **Casilla_ClickIzquierdo()**: El evento se acciona al hacer click izquierdo sobre una casilla del tablero. La mayor parte de la lógica del juego está contenida en este evento.

  Permite al usuario revelar la casilla pulsada para verificar si existe una mina o no. Además, verificar si hay minas cerca para indicarlo mediante un número con un color concreto: 1: azul, 2: verde, 3: rojo, nada en caso de que no haya minas en ninguna de las 8 casillas adyacentes, o una mina (💣) en caso de que la casilla contenga una mina (en realidad todo esto se verifica en la matriz de minas).

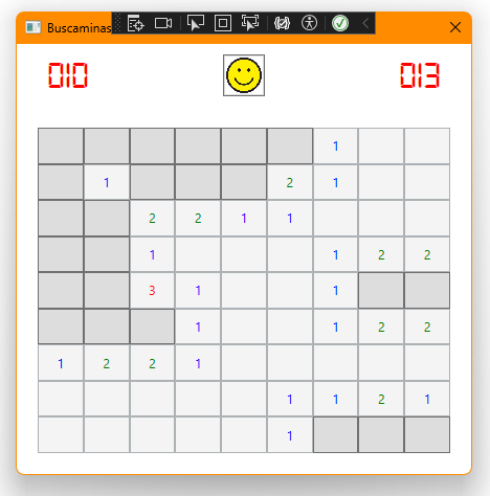
  También es el encargado de terminar el juego con el método TerminarJuego(bool win), que mostrará un mensaje en función de si se gana o se pierde. Para ello comprueba si se han revelado todas las casillas sin explotar minas (Filas * Columnas – NumMinas) o, por el contrario, se ha pisado una mina.

Img. 9 – Click izquierdo ok.

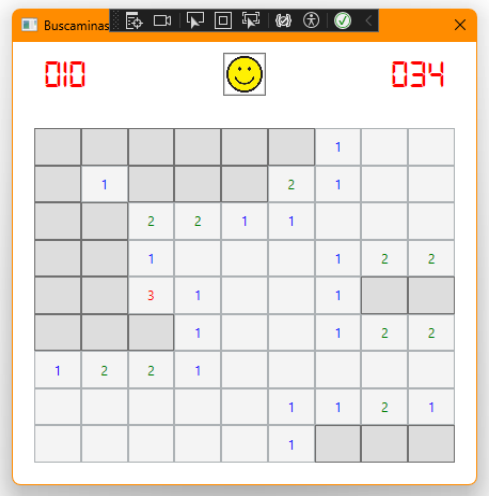
Img. 10 – Click izquierdo con mina.

- **Actualizar_Cronometro()**: El cronómetro (DispatcherTimer) llamará tras cada intervalo definido (1s en nuestro caso) al método que actualizará el cronómetro (lblCronometro) de la UI para simular un reloj digital (gracias a la fuente Digital-7 importada desde la carpeta Resources del proyecto.

  Paso de varios segundos con el juego en curso:

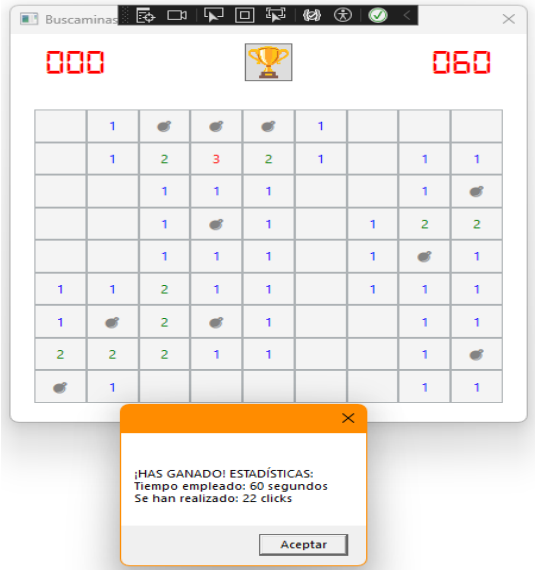


- *Img. 11 – Tiempo inicial.*

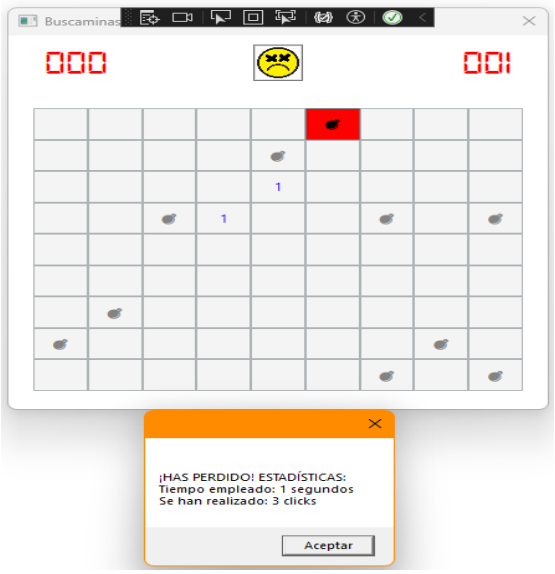


Img. 12 – Tiempo tras 11 segundos.

Ejemplos de victoria y derrota:



Img. 13 – Victoria y mensaje



Img. 15 – Derrota y mensaje.

Diseño

Se ha tratado de lograr que el juego se parezca lo máximo al original (se puede apreciar en las capturas de pantalla realizadas: img. 1-15). Se ha añadido la imagen de Smile (sonrisa) al botón de reinicio, que cambia a Dead (muerto) con la derrota y a Win (trofeo) con la victoria. El resto de diseño se explica en el apartado siguiente con las decisiones del mismo explicadas (o en apartados anteriores).

Decisiones

A lo largo del proyecto nos encontramos con diversas decisiones que tomar en cuanto a componentes seleccionados, lógica implementada o incluso el nombres de las variables.

En cuanto a la elección de los distintos componentes, el primero con el que nos encontramos es el Window.Resource → **FontFamily** x:Key="DigitalFont". El elemento es la forma de definir una fuente de texto externa (Digital-7 en este caso) y la decisión se toma en base al deseo de un aspecto "digital" en los dos contadores (Minas y Tiempo).

El resto de componentes son los componentes básicos empleados de manera habitual en las aplicaciones WPF:

Grid es el contenedor básico empleado normalmente para contener otros elementos y permite una organización en dos filas en este caso, dando un tamaño fijo (60) para la cabecera que contiene los contadores y el botón de reinicio, y un tamaño variable (*) para el tablero, lo que permite que el tablero pueda adquirir diferentes dimensiones (9x9, 15x15, etc.).

A su vez, **UniformGrid** permite distribuir el tablero en casillas uniformes sin establecer un tamaño determinado (o incluso establecer un tamaño prefijado). Era la opción más sencilla de implementar y que garantizaba la uniformidad del tablero.

Label para los contadores de Minas y Tiempo, ya que son contenedores de texto que no permiten al usuario interactuar directamente, pero sí mostrar sus contenido y actualizarlo fácilmente mediante código. Otra opción habría sido usar TextBlock, un clásico.

Button en el caso de botón de reinicio y **Button** para las casillas sin definirlos en la UI, pero sí definiéndolos en el código. Los botones permiten manejar de forma sencilla eventos como hacer click izquierdo o click dererecho. Además, era sencillo reutilizar parte del código de EventosBoton proporcionado en los ejemplos del tema 3 por Luis Fortich.

Para los nombres se ha optado por seguir la convención de C# y de las UI. Empleando PascalCase para métodos, clases, constantes y carpetas.

Así como separado por "_" en el caso de los métodos que son eventos. Dado que todas las variables (constantes o no) son privadas, se ha optado por omitir la convención "_nombre" por razones meramente estéticas y de extensión de código (un carácter más por cada variable podría hacer en total la longitud de otra posible variable).

Además, en cuanto a los componentes, se ha optado por establecer lblNombre, iconoNombre o botonNombre como criterio a la hora de nombrarlos para facilitar su reconocimiento.

Posibles mejoras

En cuanto a posibles mejoras, podría conectarse la aplicación con una base de datos para almacenar las puntuaciones y añadir un botón para acceder a dichas puntuaciones.

También podría añadirse un modo creativo o "tutorial" que permitiera deshabilitar el Game Over y pese a pisar una mina seguir jugando, o incluso añadir un sistema de vidas que permitiera fallar un número concreto de veces en función de la dificultad seleccionada, p.e. con el siguiente sistema (cada mina resta un corazón ❤):

- **Fácil** → 5 corazones ❤
- **Medio** → 4 corazones ❤
- **Difícil** → 2 corazones ❤
- **Extremo** → 1 corazón ❤ (Buscaminas clásico)

O, en mi caso, añadir un selector de dificultad para cambiar el tamaño del tablero y el número de minas.