



**Nombre:** Meylí Jiménez Velázquez

**Año:** Primero

**Grupo:** C-122

# Parte 1: ¿Cómo instalar el proyecto?

## I. Clonar este repositorio

## II. Instalar las dependencias

- a) Tener instalado **.NET 8**. Para verificar si tiene .NET 8 instalado, puede usar el siguiente comando en su terminal: `"dotnet --version"`.
- b) Instalar la librería **Spectre.Console**. Si no la tiene instalada, debe agregarla al proyecto con el siguiente comando:  
`"dotnet add package Spectre.Console --version 0.49.1"`.
- c) Instalar **.NET Resources** para habilitar el cambio de idioma. Use el siguiente comando para instalarlo:  
`"dotnet add package System.Resources.Extensions --version 9.0.0"`.

## II. Ejecutar el proyecto:

Una vez que las dependencias estén correctamente instaladas, podrá ejecutar el proyecto.

## IV. Otros detalles:

Asegúrese de tener instalada la extensión de C# en su IDE. Este es un juego de Consola.

# Parte 2: ¿Cómo jugar?

## I. En el menú principal, seleccione una de las siguientes opciones:

1. **Play**: Inicia una nueva partida.
2. **Change Language**: Selecciona entre español e inglés.
3. **Tokens**: Muestra la lista de fichas con una breve descripción, su habilidad, velocidad y tiempo de enfriamiento
4. **Exit**: Cierra el juego.

## II. Una vez que seleccione "**Jugar/Play**":

1. Escoja el tamaño del laberinto. Este debe ser al menos 7. Si ingresa un tamaño mayor a 15, el juego lo ajustará automáticamente a 15x15 (con `Math.Clamp`).
2. Antes de comenzar la partida se mostrará un menú con las fichas disponibles, la descripción de las habilidades de cada una, su velocidad y su tiempo de enfriamiento.
3. Cada jugador selecciona su ficha de entre las 6 disponibles:
  - a) *Nota: Se validará que el número elegido esté entre 1 y 6.*

## III. Durante la partida:

- i. En su turno, puede elegir si desea usar la habilidad especial de su ficha. Si el tiempo de enfriamiento no ha terminado, recibirá un mensaje indicando que no puede usarla.
- ii. Use las teclas de flecha para mover su ficha en el laberinto. Si intenta moverse hacia una pared, no se permitirá y se le informará que el movimiento no es válido. Su turno continuará hasta que realice un movimiento válido.
  -
- iii. El primer jugador en llegar a la salida (representada por una estrella) gana la partida. Tras ganar, puede decidir si desea jugar otra vez, conservando el idioma seleccionado previamente.

# Parte 3: Desglose de las mecánicas y características del juego

## Flujo del Juego

### I. Generación del Laberinto

1. La clase `MazeCreation` se encarga de inicializar el laberinto como un tablero de celdas cerradas `maze[i,j]=new Cell(false)`. Luego, a través del método `GenerateMaze`, se van abriendo las celdas adyacentes, formando pasillos. Este utiliza `recursive backtracking`, para cada dirección, salta dos celdas hacia esa dirección (para que se salte una pared). Si la celda en esa posición no está abierta, se abre la pared entre la celda actual y la celda destino (`maze[x + dx, y +`

- dy].isOpen = true) y luego se llama a GenerateMaze recursivamente para seguir explorando desde esa celda.
2. Cuando el algoritmo llega a una celda que no tiene direcciones válidas o accesibles, vuelve a la celda anterior y prueba con otra dirección.
  3. El laberinto se termina de generar cuando no hay más celdas adyacentes que se puedan explorar, o sea, cuando no hay más celdas vecinas que estén dentro de los límites del laberinto y aún no se hayan abierto (`!maze[nx, ny].isOpen`)
  4. Después de completar la generación del laberinto, se asegura que todas las celdas sean alcanzables utilizando el método `IsMazeReachable`, que verifica si todos los espacios en el laberinto son accesibles desde la esquina superior izquierda (0,0), utilizando un algoritmo de propagación de distancias (Algoritmo de Lee).
    - a. Si una celda es una pared, su valor se establece como -10; si está abierta pero aún no se ha visitado, se establece como -1.
    - b. Después de completar la propagación de distancias, el método recorre toda la matriz `distances` y verifica si alguna celda tiene el valor -1 (lo que indica que esa celda es inalcanzable).
    - c. Si `IsMazeReachable()` devuelve false (es decir, el laberinto tiene celdas inalcanzables), se repite el proceso de generación del laberinto, regenerando las celdas y verificando de nuevo.
  5. Este ciclo continuará generando el laberinto hasta que `IsMazeReachable()` retorne true, lo que significa que todas las celdas del tablero son alcanzables
  6. El método `Shuffle`, aleatoriza las posibles direcciones, lo que garantiza que resulte en un laberinto único cada vez.
  7. Finalmente, se colocan trampas, casillas beneficiosas y un portal de teletransportación aleatorio.
-

### **Instalación de Trampas y Casillas Especiales:**

1. Las trampas se colocan en celdas abiertas aleatorias, con efectos específicos, que se verán más adelante. Estas trampas se colocan mediante el método `GeneratesTraps`.
  2. Además, se colocan casillas beneficiosas que ofrecen ventajas a los jugadores, lo cual se maneja a través del método `BenefititalTilesGeneration`.
- 

### **Movimiento de Jugadores:**

1. El movimiento de los jugadores se maneja en la clase `GameManager`, donde cada jugador puede moverse utilizando las teclas de dirección. El movimiento es validado a través del método `HandlesMovement`, que se asegura de que el jugador se mueva solo si la celda donde se moverá es válida (es decir, no es una pared ni está fuera del laberinto).
  2. Esto es posible con el método `IsValidMove`, que comprueba si la nueva posición del jugador está dentro de los límites del laberinto y si la celda es accesible.
  3. Si el jugador se mueve a una trampa, se activará el efecto de ésta.
- 

### **Teletransportación:**

1. La teletransportación se maneja a través de portales especiales. Si un jugador cae en una celda marcada como portal, su posición será aleatoriamente teletransportada a otro lugar válido dentro del laberinto, gestionado por el método `CheckTeleportation`.
-

## Fichas disponibles junto con sus habilidades

1. **Elf**: Copia de forma permanente la habilidad de la ficha del otro jugador.
  2. **Wizard**: Reduce la velocidad de la ficha del otro jugador.
  3. **Fairy**: Intercambia posiciones con la ficha del otro jugador.
  4. **Siren**: Salta el turno del otro jugador.
  5. **Abuela**: Incrementa su velocidad en 1.
  6. **Unicorn**: Cada vez que se usa la habilidad activa una aleatoria de entre las disponibles en el juego.
- 

## Trampas del juego

1. **Fuego**: Pierdes un turno intentando apagarlo.
  2. **Hoyo**: Te envía al origen del laberinto (coordenadas 0,0).
  3. **Abeja**: Te pica, reduciendo tu velocidad.
  4. **Murciélago**: Incrementa el tiempo de enfriamiento de tu habilidad.
- 

## Casillas beneficiosas

1. **Reina**: Incrementa la velocidad de tu ficha.
  2. **Rey**: Reduce tu tiempo de enfriamiento.
- 

## Portal invisible

Durante el juego, aparece una casilla invisible(no tiene un emoticono) que actúa como un portal de teletransportación. Si caes en esta casilla, serás transportado a una posición válida aleatoria dentro del laberinto. Una posición válida significa que:

- No es una pared.
  - No es la salida.
  - No es una trampa ni una casilla beneficiosa.
-

**Condición de Victoria:**

1. Un jugador gana el juego cuando llegue a la salida del laberinto, que se encuentra en una celda abierta en la última fila del tablero. El proceso de verificación de victoria se maneja en el método `win`, que compara la posición del jugador con la salida del laberinto.