INSTRUCCIONES:

1. ***Abrir el enlace que lo llevará a mi repositorio en GitHub donde se encuentra el juego.***
2. ***Descargar todos los archivos incluyendo las librerías spectre.console, naudio y la canción Slime.***
3. ***Descargar e instalar Visual Studio Code y .net.***
4. ***Abrir el proyecto en Visual Studio e ir al archivo Program a la función Main y sustituir la ruta de la canción puesta por la ruta de la canción en su pc.***
5. ***Abrir la terminal y escribir: dotnet run.***

SOBRE EL JUEGO:

Slimes in Maze es un juego para dos jugadores ligeramente inspirado en Slime Rancher. El objetivo es ayudar a los slimes que selecciones a salir del laberinto, ganará el jugador que saque todos sus slimes primero. Solo puedes ayudar a 2 Slimes de los 6 que puedes escoger. El jugador 1 deberá llevar sus slimes hasta la casilla en la que empieza el jugador 2 y el jugador 2 deberá llevar sus slimes hasta la casilla en la que empieza el jugador 1. Cada Slime tiene una habilidad única que puedes usar o no durante el turno, además cada habilidad tiene un determinado tiempo de enfriamiento, mientras una habilidad esté en tiempo de enfriamiento no se puede usar. Por ejemplo si su habilidad tiene tiempo de enfriamiento 3, tendras que jugar 3 veces con ese slime hasta poder volver a usar su habilidad. Cada slime

tiene una velocidad específica, la velocidad es la cantidad de pasos que tienes que moverte en el turno.

ID	*Nombre*	*Habilidad*	*Velocidad*	*Tiempo de enfriamiento*
1	*Fast Slime*	*Duplica su velocidad*	*15*	*2*
2	*Trap Slime*	*Las trampas no le afectan*	*14*	*3*
3	*Strong Slime*	*Destruye las paredes del laberinto*	*10*	*1*
4	*Begin Slime*	*Regresa un slime del oponente al inicio*	*12*	*5*
5	*Frozen Slime*	*Congela los slimes de oponente*	*18*	*4*
6	*Skill Slime*	*Impide que el rival use las habilidades de sus slimes*	*22*	*2*

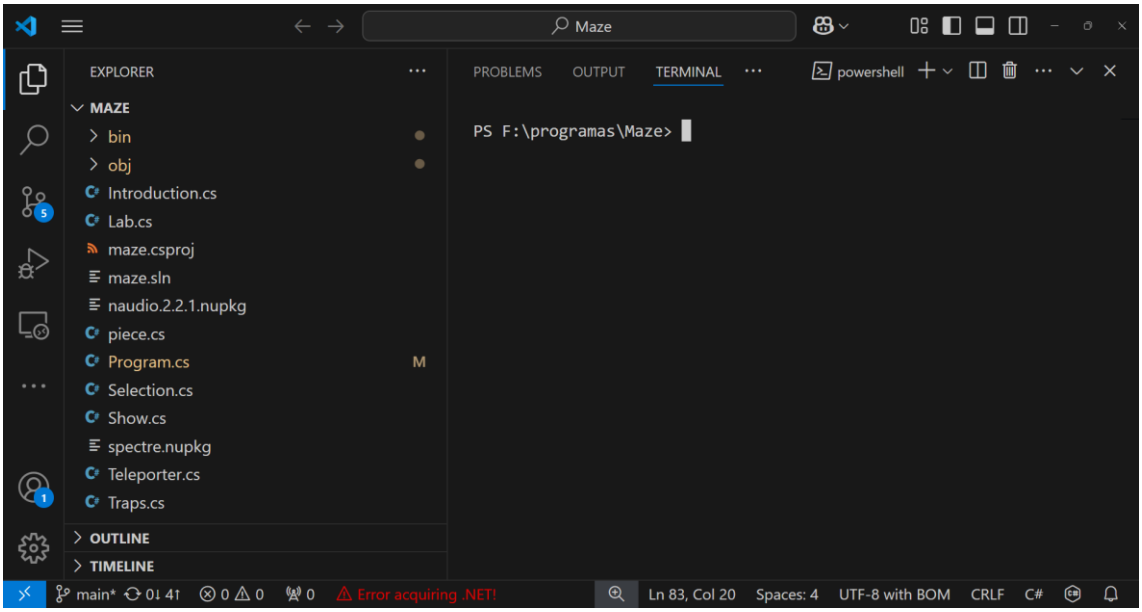
También por el laberinto se encuentran dispersas varias trampas que aplican una penalización al jugador que caiga en ellas.

Trampa	*Penalización*
1	*Regresa el Slime al inicio*
2	*Paraliza al Slime por ese turno*
3	*Divide entre 2 la velocidad del slime*

Además en el laberinto hay puertas slimes que teletransportan su slime al final haciendo más amena la partida. En cada turno deberá seleccionar que Slime desea usar y si desea o no usar su habilidad. Luego deberá mover su slime hasta que sus movimientos restantes sean 0.

SOBRE EL CÓDIGO:

Mi proyecto se divide en 8 archivos diferentes, en cada archivo se encuentra una clase con 1 o varias funciones diferentes.



LAB:

La clase Lab fue la segunda clase que hice, en esta clase genero el laberinto con los requisitos especificados, para ello utilizo un algoritmo recursivo muy similar al DFS (búsqueda en profundidad) el cual es usado para recorrer grafos. Mi algoritmo garantiza que todas las casillas sean alcanzables desde cualquier posición y además se genera aleatoriamente. En esta clase tengo 2 funciones, la primera se llama out y verifica que la posición a la que me quiero mover no se salga de rango y la segunda función llamada dfs es la que genera el laberinto.

INTRODUCTION:

Esta clase es la encargada de introducir al jugador. En la función Menu se muestra el nombre del juego y las opciones para que el jugador seleccione si desea jugar o no. En la función Begin se da una breve descripción del juego.

TRAPS:

En esta clase implementé 3 trampas diferentes. En la función Trap_1 está la primera trampa, si caes en esta tu ficha retrocede al inicio. En Trap_2 análogamente implementé la segunda trampa, la penalización de esta consiste en dejar sin movimientos restantes la ficha que pase por ahí y en Trap_3 está la trampa que reduce la velocidad a la mitad durante todo el juego. Cada trampa está diseñada para que solo funcione una sola vez, luego desaparece. Además las trampas aparecen en el tablero aleatoriamente pero nunca en las posiciones iniciales de las fichas de ambos jugadores. En total serían 12 trampas, 4 de cada tipo.

TELEPORTER:

En esta clase hay una sola función : Tele, en esta implementé las puertas slimes que teletransportan las fichas a su destino final. Al igual que en las trampas los coloco aleatoriamente por el tablero pero no en las posiciones iniciales de las fichas, y solo se pueden usar una sola vez. Genero un total de 3.

SELECTION:

Esta clase la hice para no cargar tanto mi clase principal. Aquí tengo varias funciones para cuando el jugador tiene que elegir entre varias opciones. En Options elige si desea jugar o salir. En Decide escoge que slime va a usar para la partida y en Choose escoge la ficha que quiere mover en ese turno.

SHOW:

En esta clase tengo 2 funciones, la primera es Maze y supuestamente debería mostrar el laberinto pero como uso un buffer (para evitar pantallazos) lo que hago es meter al buffer lo que quiero mostrar en en mi clase principal muestro el buffer. La función Slime lo que hace es mostrar el nombre del slime que corresponde al id que se le pasa. Esto ocurre cuando se le muestra al jugador sus slimes disponibles.

PIECE:

En esta clase todo lo que hago es con las fichas. En la función piece (el constructor) inicializo propiedades como nombre e ícono. En la función skill según la ficha que sea llamo otra función que está en esa misma clase en la que implementé la habilidad de esa ficha. En la función init le asigno a cada ficha su posición inicial, su posición actual y su posición final. Así verifico luego la condición de victoria (si la posición actual es igual a a final). En la función Move muevo la ficha por el laberinto teniendo en cuenta si caigo en una trampa o en un teletransportador y abajo están las funciones ya mencionadas de las habilidades.

PROGRAM:

Esta es mi clase principal. Aquí llamo a las funciones que están en las otras clases. Primero declaro algunas variables globales como el tamaño del laberinto. En la función MainMenu le asigno a cada ficha su nombre, su velocidad, su habilidad y su tiempo de enfriamiento. En MakeTable creo una tabla en la que le muestro al jugador todos los slimes y sus características. En MakeLab hago el laberinto, pasa de ser un laberinto de int a uno de string. En Main lo primero que hago es ponerle música al juego para eso usé la librería NAudio, aquí llamo a las demás funciones y creo el buffer para evitar los pantallazos de cuando se me actualiza el laberinto. También en esta función implementé varias cosas como avisarle al jugador si cayó en una trampa, si se teletransportó o si ganó ya la partida.

Para ponerle color a todo lo que muestro y crear la tabla uso spectre.console. Así hago la experiencia más user friendly. A medida que fui programando me fueron surgiendo varios problemas que poco a poco fui solucionando (ver en commits) pero lo que más problemas me ha dado ha sido el buffer.