Proyecto de Programación I. Mazerunners

Ignacio Miguel Rodríguez Pacheco C121

Mazerunners es un juego multijugador con temática de escape de un laberinto, desarrollado utilizando Windows Form. El objetivo del juego es recoger todas las esferas del dragón para que aparezca la salida del laberinto, y el jugador con mayor cantidad de esferas del dragón que llegue a la salida gana.

Instrucciones para jugar:

1. Ejecutar el proyecto utilizando el comando dotnet run en una terminal desde la carpeta del proyecto Mazerunners.

2. Hacer click en el botón "Jugar" para iniciar una partida.

3. Seleccionar los personajes para jugar.

4. Utilizar las teclas W (hacia arriba), S (hacia abajo), A (izquierda) y D (derecha) para moverse.

5. Utilizar la tecla P para atacar y la tecla O para usar una habilidad.

6. Si un jugador introduce otra tecla distinta a las mencionadas, realiza un movimiento no válido, ataca o usa su habilidad cuando los turnos sin atacar o el tiempo de enfriamiento son mayores que cero pierde su turno.

7. Si un jugador está paralizado (turnos sin moverse>0) se salta su turno.

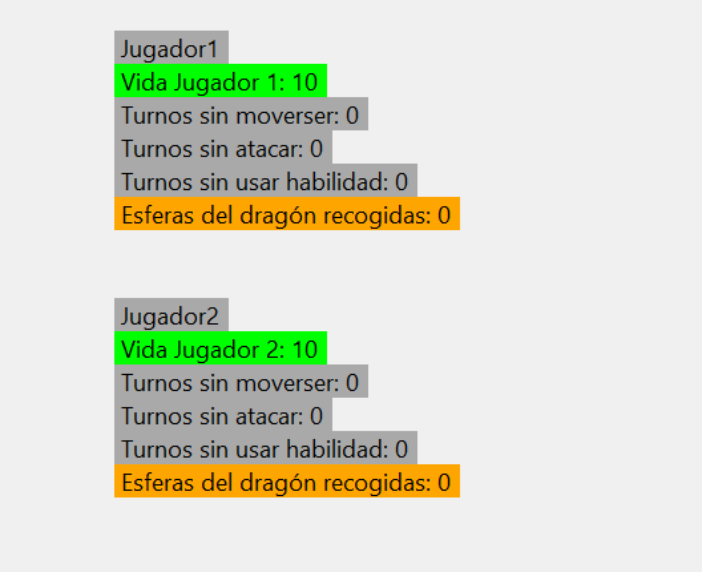
Personajes y habilidades:

- Goku: Genkidama (4 de daño al oponente)
- Gohan: Potencial Desatado (4 de daño al oponente)
- Vegeta: Garlick Canon (3 de daño al oponente y parálisis)
- Freezer: Habilidad similar a Vegeta con mayor tiempo de parálisis y menor daño sobre el oponente
- Piccollo: Regeneración (aumenta su vida)

- Cell: Absorción (reduce la vida del oponente y aumenta la suya)
- Krillin: Parálisis (impide el movimiento de los oponentes)
- Androide 18: Explosión (disminuye la vida del oponente y la suya propia)
- Trunks: Duplica su velocidad cada vez que se activa la habilidad
- Majin Buu: Aumenta su vida a 30 y aumenta en 2 su fuerza de ataque cada vez que se activa la habilidad

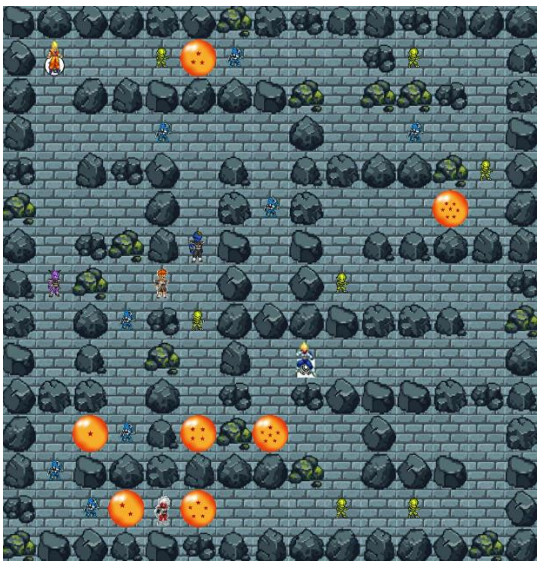
Al ser mucho más poderosas sus habilidades Trunks y Majin Boo cuentan con un tiempo de enfriamiento mucho mayor que el de los demás personajes (150 turnos).

Una vez seleccionados los personajes se presiona el botón confirmar e inicia el juego. Al costado del laberinto hay varias etiquetas label para visualizar el estado de los jugadores:



El objetivo del juego es recoger todas las esferas del dragón para que aparezca la salida del laberinto, una vez esta aparezca el jugador con mayor cantidad de esferas del dragón que llegue a la salida gana y el juego se detiene mostrando un mensaje con el ganador, si el jugador que llega

no es el que más esferas del dragón tiene este volverá a su posición inicial, la salida del laberinto desaparece si luego de ser recogidas todas las esferas del dragón un jugador muere al caer en una trampa, ese jugador perderá una esfera del dragón que será generada en el mapa y deberá ser recogida para que aparezca la salida nuevamente.



Existen tres tipos de trampa:

Los Saibaiman te impiden moverte por dos turnos

Los soldados de freezer te quitan dos de vida

Cell Jr te transporta hacia alguna de las dos trampas anteriores, en caso de que no quedar trampas de los otros dos tipos te transporta a la posición inicial

Un jugador puede quitarle una esfera del dragón a otro si reduce su vida a cero, para lograr esto puede emplear su habilidad (si esta reduce vida) o puede atacarlo, para atacar a otro jugador este tiene que encontrar en una celda aledaña (arriba, izquierda, abajo, derecha)

Sobre el código, clases utilizadas con los métodos más importantes:

Celda: solo contiene un constructor donde se inicializan propiedades como el Valor de la celda (todas son cero por defecto, que representan un muro…arreglar…), si es trampa o es una posición clave (puntos de inicio y salida)

Menú: menú inicial, es un formulario con botones que permiten ejecutar las siguientes acciones, jugar, ver las instrucciones, ver las habilidades y salir

```
FormMenu.cs > ...
14      public partial class FormMenu : Form
            1 reference
26  >       private void btnJugar_Click(object sender, EventArgs e) ···
34
            1 reference
35  >       private void btnInstrucciones_Click(object sender, EventArgs e) ···
44
            1 reference
45  >       private void btnHabilidades_Click(object sender, EventArgs e) ···
49
            1 reference
50  >       private void btnSalir_Click(object sender, EventArgs e) ···
```

Laberinto: encargada de generar el laberinto (una matriz de celdas), contiene varios métodos:

Laberinto.cs > Laberinto

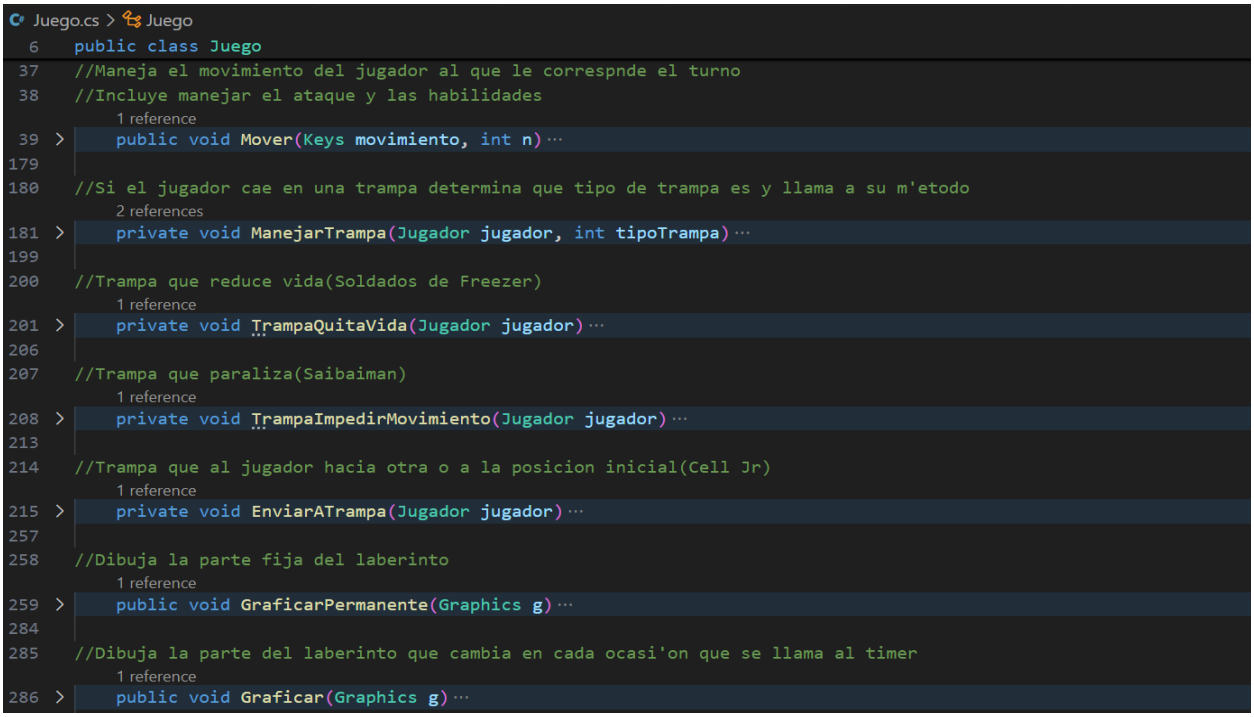
```
3   public class Laberinto
38  //Crea la matriz de Celdas llena de muros
        1 reference
39      private void GenerarLaberinto()···
63
64  //M'etodo para la generaci'on del laberinto usando un algoritmo de backtracking
        2 references
65      private void GenerarLaberinto(int x, int y)···
94  //Cambia el orden de las direcciones en cada llamada recursiva
        1 reference
95      private void Mezclar(int[] direcciones)···
105 //Determina si la posici'on que se est'a dentro de las dimensiones del laberinto
        1 reference
106     private bool PosionValida(int x, int y)···
110 //M'etodo para seleccionar una celda del laberinto
        31 references
111     public Celda GetCelda(int x, int y)···
115 //Genera las trampas
        1 reference
116     private void GenerarTrampas()···
158
159 // Método para verificar si hay trampas adyacentes
        1 reference
160     private bool HayTrampaAdyacente(int x, int y)···
168 //Selecciona las posiciones de inicio de los dos jugadores
        1 reference
169     private void SeleccionarCeldas()···
206 //Genera la salida cuando se recogen todas las esferas del drag'on
        1 reference
207     public void GenerarSalida(int posxjugador1, int posyjugador1, int posxjugador2, int posyjugador2)···
229
230 //M'etodo para poner las esferas del drag'on en el laberinto
        2 references
231     public void Poner1EsferasDelDragon(){···
260
261 // Método para verificar si dos celdas son adyacentes
        1 reference
262     private bool SonAdyacentes((int, int) celda1, (int, int) celda2)···
```

Jugador:

Jugador.cs > Jugador

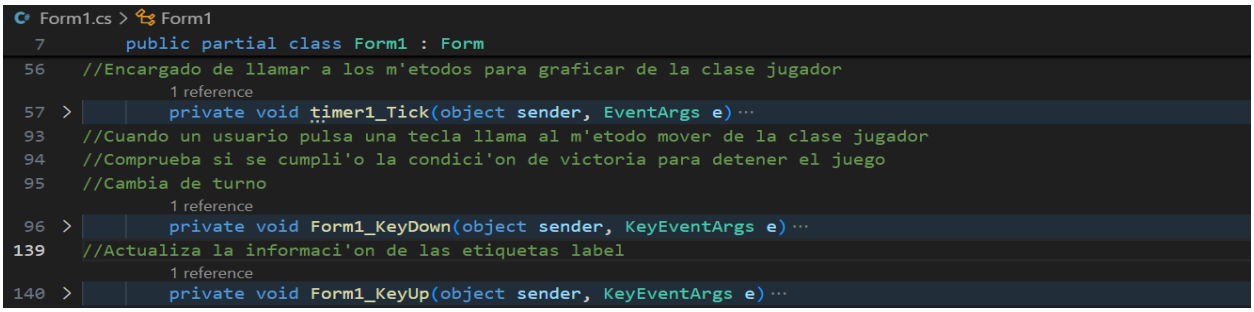
```
71  //Enlace de las im'agenes de los  personajes
        1 reference
72      private string DireccionImagen()···
86
87  //Posici'on inicial del jugador
        2 references
88      private void ObtenerPosicionInicialJ1(int jugadorNumero)···
98
99  //Ataque del jugador
        2 references
100     public void Atacar(Jugador jugador2){···
108
109 //Ver si el otro jugador se encuentra en una posici'on cercana
        1 reference
110     public bool Posicion_cercana(Jugador otroJugador)···
144
145 //Reduce la vida del jugador
        2 references
146     public void ReducirVida(int cantidad)···
151
152 //Gestiona el tiempo de enfriamiento, los  turnos sin atacar y sin moverse
153 //Devuelve al jugador a su posicion inicial si su vida llega a cero y verifica si muri'o por una trampa
154 //para quitarle una esfera del drag'on y gnerarla en el laberinto
        1 reference
155     public void ActualizarEstado(int n)···
188 //Turnos sin moverse
        2 references
189     public void NoMoversePorTurnos(int turnos)···
195 //Verifica si el jugador puede moverse
        2 references
196     public bool PuedeMoverse()···
201 //Muestra al jugador en el laberinto
        1 reference
202     public void Mostrar(Graphics g)···
```

Juego:



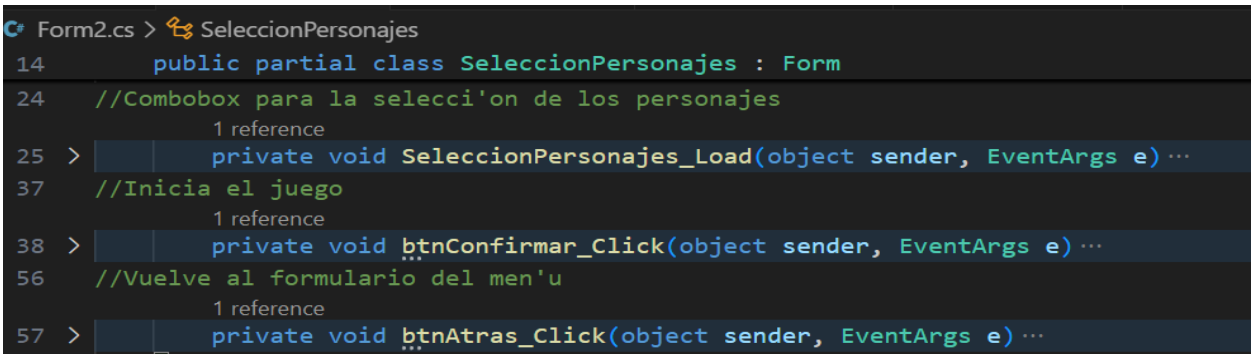
```
Juego.cs > Juego
  6   public class Juego
 37   //Maneja el movimiento del jugador al que le correspnde el turno
 38   //Incluye manejar el ataque y las habilidades
          1 reference
 39 >     public void Mover(Keys movimiento, int n)···
179
180   //Si el jugador cae en una trampa determina que tipo de trampa es y llama a su m'etodo
          2 references
181 >     private void ManejarTrampa(Jugador jugador, int tipoTrampa)···
199
200   //Trampa que reduce vida(Soldados de Freezer)
          1 reference
201 >     private void TrampaQuitaVida(Jugador jugador)···
206
207   //Trampa que paraliza(Saibaiman)
          1 reference
208 >     private void TrampaImpedirMovimiento(Jugador jugador)···
213
214   //Trampa que al jugador hacia otra o a la posicion inicial(Cell Jr)
          1 reference
215 >     private void EnviarATrampa(Jugador jugador)···
257
258   //Dibuja la parte fija del laberinto
          1 reference
259 >     public void GraficarPermanente(Graphics g)···
284
285   //Dibuja la parte del laberinto que cambia en cada ocasi'on que se llama al timer
          1 reference
286 >     public void Graficar(Graphics g)···
```

Form1: en este formulario se muestra el juego y se llaman a los métodos de la clase juego



```
Form1.cs > Form1
  7       public partial class Form1 : Form
 56   //Encargado de llamar a los m'etodos para graficar de la clase jugador
              1 reference
 57 >         private void timer1_Tick(object sender, EventArgs e)···
 93   //Cuando un usuario pulsa una tecla llama al m'etodo mover de la clase jugador
 94   //Comprueba si se cumpli'o la condici'on de victoria para detener el juego
 95   //Cambia de turno
              1 reference
 96 >         private void Form1_KeyDown(object sender, KeyEventArgs e)···
139   //Actualiza la informaci'on de las etiquetas label
              1 reference
140 >         private void Form1_KeyUp(object sender, KeyEventArgs e)···
```

SeleccionPersonajes:



```
Form2.cs > SeleccionPersonajes
 14       public partial class SeleccionPersonajes : Form
 24   //Combobox para la selecci'on de los personajes
              1 reference
 25 >         private void SeleccionPersonajes_Load(object sender, EventArgs e)···
 37   //Inicia el juego
              1 reference
 38 >         private void btnConfirmar_Click(object sender, EventArgs e)···
 56   //Vuelve al formulario del men'u
              1 reference
 57 >         private void btnAtras_Click(object sender, EventArgs e)···
```