BorderMazeLands

INSTRUCCIONES:

1. ***Abrir el enlace de GitHub y descargar el repositorio.***
2. ***Instalar y ejecutar Unity Hub, dentro de este instalar Unity Editor.***
3. ***Seleccionar "Abrir Proyecto" y elegir la carpeta del proyecto.***
4. ***En Unity, presionar el botón Play.***
5. ***Si desea solo jugar, puede solamente descargar la build adjunta en el repositorio de Github y ejecutar el .exe.***

SOBRE EL JUEGO:

BorderMazeLands es un juego multijugador por turnos de dos a cuatro jugadores, insipirado en Borderlands 2. El objetivo es ayudar a los BuscaCamaras que selecciones a encontrar la Cámara y salir del laberinto, ganará el jugador que encuentre una llave y abra La Cámara. Solo puedes escoger uno de los seis protagonistas. Debes buscar en la oscuridad, una llave, mientras sorteas las trampas del laberinto. Cada BuscaCamaras tiene una habilidad única que te ayudara, cada habilidad tiene un tiempo de enfriamiento, en el cual no podrás usarla. Cada jugador tiene además velocidad, que representa la cantidad de casillas que te puedes mover en cada turno, y vida, la cual representa la cantidad de daño que puedes soportar de las trampas…u otros BuscaCamaras.

HISTORIA:

"Solo una advertencia..."

No sé cómo terminaste aquí, pero si hay algo que he aprendido en Pandora, es que cuando una puerta misteriosa aparece de la nada, lo mejor es NO abrirla. Claro, nadie escucha.

Este laberinto no es normal. Cambia, siempre es diferente, juega contigo. Y en el centro hay una cámara cerrada con algo dentro. Algo que vale la pena... o algo que te matará por intentar conseguirlo. Necesitas una llave para abrirla, pero no creas que será fácil.

Axton, Maya, Salvador, Zer0, Gaige y Krieg ya están atrapados aquí, y créeme, si ellos están teniendo problemas, tú también los tendrás. Trampas, obstáculos, cosas que acechan en la oscuridad... Nada de esto es casualidad. Alguien diseñó este lugar y quiere vernos sufrir.

Así que si entras, más te vale estar listo. Corre, pelea, usa tu cabeza. Y si encuentras la llave... bueno, esperemos que lo que haya detrás de esa puerta valga la pena.

BuscaCamaras	*Habilidad*	*Velocidad*	*Tiempo de enfriamiento*	*Vida*
Zero	*Reduce en 5 puntos la vida de cualquier jugador en línea recta.*	*4*	*5*	*5*
Gaige	*Muestra las próximas 2 casillas hacia la Cámara si tienes una llave, de lo contrario muestra el camino a la llave más cercana.*	*4*	*3*	*5*
Axton	*Activa un escudo que lo vuelve inmune a las trampas durante 2 turnos.*	*4*	*3*	*5*
Krieg	*Explota las paredes adyacentes a él y las vuelve camino.*	*4*	*4*	*5*
Maya	*Congela a cualquier jugador a 4 casillas de distancia.*	*4*	*3*	*5*
Salvador	*Aumenta su velocidad y visión durante ese turno.*	*4*	*4*	*6*

En el laberinto se encuentran dispersas varias trampas que aplican un efecto al jugador que caiga en ellas. Cuidado, están todas ocultas, excepto las de teletransporte y las de visión…podrías usarlas a tu favor. El primer jugador que caiga en una trampa la revela, pero no anula su efecto…recuerda que puedes caer de nuevo.

Trampas	***Efecto***
Daño	***Reduce de 2 a 3 puntos tu vida***
Lentitud	***Reduce tu velocidad en 2 puntos durante 2 turnos***
Enfriamiento	***Reinicia el tiempo de enfriamiento de tu habilidad***
Congelación	***Congela de 1 a 2 turnos***
Teletransporte	***Te teletransporta a una casilla aleatoria del mapa***
Visión	***Aumenta tu visión durante 1 turno***

DETALLES DE LA IMPLEMENTACIÓN:

Entre las clases más importantes se encuentran:

MazeManager:

La clase MazeManager incluye todo lo relacionado con el laberinto, genera primeramente el laberinto usando el Algoritmo de Prim. Este algoritmo garantiza que todas las casillas sean alcanzables desde cualquier posición y además se genera aleatoriamente. En esta clase se encuentran funciones encargadas de generar los personajes, trampas y llaves. Además de funciones auxiliares como comprobar la distancia entre dos casillas usando BFS, y saber todas las casillas alcanzables (con su velocidad) por el jugador.

Piece:

Esta clase representa las fichas de los jugadores. Esta contiene todas las propiedades que tiene una ficha, como su vida, velocidad, tiempo de enfriamiento y el rol de la ficha, es decir el "Team" que las identifica; además de variables que representan si están afectados o no por una trampa, y variables importantes como la casilla donde se encuentra "Position", y la llave que tiene "key".

Team:

Esta clase representa los distintos BuscaCamaras, la he usado para crear archivos de tipo Team en la carpeta de assets, y desde ahí comodamente modifico el nombre, y las propiedades de cada BuscaCamaras, que representan un Team único.

Tile:

Esta clase contiene un constructor, el cual recibe la posición en el laberinto donde se instancia la casilla y si es camino o no. Contiene variables como piece, trap, key que representan los diferentes objetos que puede estar en una casilla.

Trap:

Esta clase contiene un constructor, el cual solo recibe la casilla donde se instancia la trampa y el tipo de esta. La función "Activate()" es utilizada desde la clase PieceManager para activar la trampa si un jugador cae en ella, este método llama , según el tipo de trampa, a la función que representa el efecto especifico de la trampa.

PieceManager:

Esta clase maneja funciones importantes relacionadas con las fichas, como "SelectPiece()" para seleccionar la ficha que se quiere mover, y la propia funcion "MovePiece()", ademas de funciones esenciales para el desarrollo de la partida como verificar la vida y el tiempo de enfriamiento de cada ficha.

Turn_Manager:

Esta clase maneja funciones importantes como iniciar y finalizar los turnos. Contiene el indice del jugador actual y funciones auxiliares como actualizar la camara y la luz de la escena hacia el proximo jugador cuando se finaliza el turno.Además de verificar en cada jugada la condición de victoria

HudManager:

Esta clase se encarga de aportar una experiencia mas user friendly, aportando a la partida un menú de pausa, una tabla con las estadísticas de tu ficha y un pequeño tutorial. Además de mostrar la pantalla de victoria al finalizar la partida.

Desafíos enfrentados:

-El primer problema fue generar correctamente el laberinto en la escena de Unity, aunque crearas una matriz de tipo Casilla o entero, me encontré con problemas para instanciarla en la escena y saber la posición en escena de cada casilla. Al final, existía una manera cómoda de hacerlo, la cual es usar las coordenadas de la escena e instanciar la casilla i,j en la posición i,j de la escena, esto genera el laberinto "al revés" pero no afecta a niveles prácticos.

-Cuando logré generar el laberinto y crear mis primeras fichas, me encontré con el problema de…dar click en la ficha, después en la casilla a la que me quería mover, y moverme sin que explote Unity. El problema de esto es que tuve que aprender cómo funcionan los colliders en Unity, ya que estos son los que detectan el click en pantalla.

-Los problemas con el movimiento no acaban, mi juego al dar click sobre una ficha, calcula las casillas validas en dependencia de la velocidad de la ficha, y esto lo hace haciendo un bfs y devolviendo las casillas con una distancia menor o igual a la velocidad. ¿Qué ocurre? Que cometí un error en el algoritmo, mi matriz con las distancias de cada casilla, es una matriz de enteros, comienza con -1 en los obstáculos y 0 en los caminos; el bucle busca constantemente busca la casilla sin visitar que se encoló primero…el problema está en que mi única condición para determinar si esa casilla estaba sin visitar, era preguntar si era diferente de -1, esto está mal, diferentes a -1 son todas las casillas que no son obstáculos, por tanto nunca paraba el bucle.