**Informe del primer proyecto**

->Alumno: Adrián Xavier Brioso Jurado

->Grupo: C111

El proyecto consiste en un juego de laberintos. Los jugadores escogen entre varias fichas con las que luego buscarán salir de un laberinto generado aleatoriamente cada partida.

**Escenas del proyecto**

-Menú: Donde están las opciones iniciales desde las cuales se pueden abrir otras escenas.

-Settings: Donde el usuario puede regular el volumen y la calidad gráfica del juego.

-Créditos: Como su nombre indica, son los créditos del juego.

-Creation: Donde el usuario interactúa con un formulario en el que introduce algunos datos que luego se utilizarán para generar el laberinto.

-Game: Donde el usuario escoge con qué fichas jugará y donde verá el laberinto por el que se desplazará.

**Scripts principales del proyecto:**

->**Dialogue**: Almacena el nombre del personaje que se comunica con el usuario y las frases de un dialogo.

->**DialogueManager**: Vinculado a un Text para mostrar el nombre del personaje y las frases anteriormente mencionadas, a un animator para animar el panel de diálogo y a un GameObject que es el propio panel de diálogo. Permite visualizar los diálogos en la escena, controla la secuencia de frases y muestra un formulario al terminar el diálogo. Métodos importantes:

-StartDialogue(Dialogue dialogue): Inicia el diálogo.

-DisplayNextSentence(): Muestra la siguiente frase.

-ShowForm(): Muestra el formulario al finalizar el diálogo.

->**DialogueForm**: Vinculado a un InputField para ingresar el número de jugadores y nombres, a un Button para confirmar la selección y a un Text que muestra instrucciones al jugador. Interactua con el usuario para obtener entradas necesaria para generar el laberinto, valida los inputs y pasa los datos a GameData. Métodos importantes:

-ShowQuestion(): Muestra distintos textos para indicar al usuario que datos debe introducir.

-HandleSubmit(): El usuario confirma que cada input que introduce es el que desea.

->**ChipDescription**: Vinculado a un GameObject que es el prefab de la descripción y a un Text que muestra la descripción de la ficha. Muestra informaciones de la ficha al poner el cursor sobre ella.

->**VictoryUI**: Vinculado a un GameObject que es el panel de victoria y a un Text que muestra el mensaje de victoria. Muestra un mensaje de victoria cuando un jugador llega a la salida y carga la escena del menú principal.

->**Maze**: Vinculado a varios prefabs (wallPrefab, pathPrefab, trapPrefab) y a un Transform (mazeGrid) para generar visualmente el laberinto en la escena. Genera un laberinto basado en lo almacenado en GameData y lo visualiza en pantalla. Gestiona la colocación de trampas y la generación de caminos transitables. Métodos importantes:

-GenerateMazeFromGameData(): Genera el laberinto usando datos de GameData.

-MazeGen(int players, int chips): Crea la estructura del laberinto con entradas, salidas y trampas.

-CoordinatesRandomizer(): Define coordenadas aleatorias para la entrada y la salida.

-VisualizeMaze(Casilla[,]): Muestra el laberinto en la escena, coloreando las distintas casillas. Asigna efectos aleatorios a las trampas.

->**MazeController**: Vinculado a un GameObject y a un Transform para colocar las fichas en el laberinto. Controla la interacción con las casillas del laberinto. Métodos importantes:

-PlaceFichasInMaze(): Coloca las fichas de los jugadores en la entrada del laberinto. Vincula cada ficha con su parte lógica y su controlador.

-GetMazeEntrance(): Obtiene la casilla de entrada del laberinto.

-GetCasillasVecinas(Casilla): Devuelve las casillas vecinas transitables.

-GetCasillaAleatoriaValida(): Obtiene una casilla transitable aleatoria.

->**MazeGenerator**: Genera la estructura del laberinto en forma de matriz de enteros, donde el 0 representa las paredes y el 1 los caminos. Crea caminos transitables, asegura la conexión entre la entrada y la salida y evita que se generen bordes transitables. Métodos importantes:

-GenerateMaze((int, int), (int, int)): Genera el laberinto con una entrada y salida específicas.

-GeneratePath(int, int, (int, int)): Traza un camino desde la entrada hasta la salida.

-EnsureConnection((int, int)): Asegura que la salida esté conectada al -resto del laberinto.

->**ClassCasilla**: Define las propiedades de una casilla en el laberinto (posición, si es transitable, si es una trampa, si entrada o salida, fichas que contiene).

->**GameContext**: Controla el estado global del juego, almacena información como el laberinto, la lista de jugadores y el turno actual.

->**Player**: Clase abstracta que define a los jugadores con su nombre y las fichas que poseen.

->**CasillaTrampa**: Hereda de Casilla. Añade funcionalidades para las casillas trampa, como el tipo de efecto que aplican y cómo se activan. Métodos importantes:

-ActivarTrampa(FichaComponent): Aplica el efecto de la trampa a la ficha que la activa. Los efectos son los siguientes: MultiplicarCooldown, DividirVelocidad, RegresarEntrada, Teletransportar.

->**TrapManager**: Es el gestor de las trampas del laberinto. Esta vinculado a un GameObject que muestra mensajes de trampa y a un Text que describe el efecto de la trampa activada. Métodos importantes:

-RegistrarTrampa(): Añade las trampas a un diccionario con sus coordenadas.

-HayTrampaEn(): Indica si hay una trampa en unas coordenadas.

->**CharacterCreator**: Vinculado a varios objetos de la escena como un Transform a donde van las fichas seleccionadas, una List de botones que representa las fichas disponibles, un GameObject que representa un panel donde están todos estos objetos, etc. Gestiona la creación de personajes y permite seleccionar fichas por los jugadores. Métodos importantes:

-OnFichaButtonClick(Button): Cada ficha es un botón que al oprimir llama a este método, que las envía a las fichas seleccionadas.

-CreateFichaByType(string): Crea una ficha específica según el tipo elegido.

-ConfirmSelection(): Confirma las fichas seleccionadas y pasa al siguiente jugador si aún no ha elegido. Al concluir inicia el juego.

-StartGame(): Inicia el juego cuando los jugadores hayan elegido sus fichas.

->**Ficha**: Clase abstracta que define las propiedades y comportamientos básicos de las fichas, como su velocidad, tiempo de recarga, habilidades y si pueden usarlas. Métodos importantes:

-Skill(): Método abstracto que define la habilidad especial de la ficha.

-ResetTurn(): Reinicia la cantidad de pasos y la confirmación de movimiento.

-ReducirCooldownDeFichas(): Reduce el cooldown de las fichas de un jugador.

->**FichaComponent**: Vinculado a un objeto para representar visualmente una ficha en el laberinto. Gestiona la inicialización de la ficha y su relación con una casilla. Métodos importantes:

-Initialize(Ficha, Casilla): Inicializa la ficha y la asocia con una casilla.

-OnFichaClick(): Permite al jugador seleccionar con que ficha moverse en su turno.

->**FichaController**: Vinculado a un objeto para gestionar el movimiento de las fichas en el laberinto. Permite a los jugadores mover sus fichas ingresando coordenadas en dos InputField. Métodos importantes:

-TryMove(): Intenta mover la ficha a la casilla que indiquen las coordenadas ingresadas.

-MoverFicha(Casilla): Mueve la ficha a una nueva casilla y detecta si son trampas o la salida. Activa las trampas y activa sus efectos.

-PuedeLlegarA(Casilla, Casilla, int): Verifica si la ficha puede llegar a la casilla destino con su velocidad actual.

-UsarHabilidad(): Ejecuta la habilidad única de la ficha de ser posible y termina el turno.

->**TurnManager**: Gestiona el sistema de turnos y la selección de fichas. Está vinculado a un GameObject donde se ingresan las coordenadas de la casilla a la que mover la ficha y un Text que muestra el jugador actual y la ronda. Métodos importantes:

-StartTurn(Player): Inicia el turno de un jugador. Activa la selección de fichas.

-EndTurn(): Reduce en 1 el cooldown de todas las fichas y avanza a la siguiente ronda.

-SeleccionarFicha(Ficha): Define con que ficha se va a mover el jugador y deshabilita las demás. Cambia el color de la ficha.

->**SkillButton**: Permite activar la habilidad de la ficha seleccionada. Esta vinculada a un Button con el que se activa la habilidad. Metodos importantes:

-HabilidadButtonClick(): Si hay una ficha seleccionada y esta puede usar su habilidad, llama a UsarHabilidad() de FichaController y activa la habilidad.

**Fichas**

-NormieChip:

Habilidad: Ninguna.

Stats base: Velocidad 3, Cooldown 2.

-CooldownChip:

Habilidad: Reduce en 2 el cooldown de todas las fichas del jugador.

Stats base: Velocidad 4, Cooldown 5.

-SpeedChip:

Habilidad: Duplica su velocidad.

Stats base: Velocidad 5, Cooldown 3.

-InvisibilityChip

Habilidad: Hace invisible al jugador actual.

Stats base: Velocidad 5, Cooldown 5.

-ShieldChip:

Habilidad: Aún no definida.

Stats base: Velocidad 2, Cooldown 5.

-TeleportChip:

Habilidad: Teletransporta la ficha a una casilla aleatoria.

Stats base: Velocidad 4, Cooldown 5.

-TrapChip:

Habilidad: Aún no definida.

Stats base: Velocidad 4, Cooldown 6.

**Tutorial**

El jugador comienza en la escena del menú. Aquí el jugador puede acceder a Settings, Créditos y Creation o salir del juego.

En Creation los jugadores se encontrarán con un mensaje. Para avanzar presionar Enter. Aparecerá un pergamino donde los jugadores deben indicar cuántos son, sus nombres y con cuántas fichas jugarán. Al confirmar todo esto, avanzarán a Game

Dentro de Game, el jugador verá el laberinto generado a la derecha. A la izquierda están las fichas disponibles. El jugador debe escoger las fichas que correspondan y confirmar su elección. Lo mismo para todos los jugadores.

Al confirmar todas las fichas, estas aparecerán en la entrada del laberinto. El primer jugador escogerá la ficha que desee y luego introducirá las coordenadas a las que desea moverla y lo mismo para el resto de jugadores. Si una ficha cae en una casilla rosada, activará una trampa con su correspondiente efecto. Si un jugador llega a la casilla verde, este ganará la partida