Proyecto final Paradigmas y técnicas de programación:

Alice's descent

Claudia García y Patricia Díaz

1.Introducción

Alice's Descent es una aventura que consiste en superar 3 niveles distintos. A través de Alicia, los jugadores se encuentran en distintos escenarios es los que tienen que superar las pruebas a las que se enfrenten. Los jugadores han de superar el nivel en el que se encuentran si quieren pasar al siguiente. En caso de fallar, el nivel en el que se encuentran se reinicia, listo para que el jugador lo vuelva a intentar.

Para desarrollar las mecánicas y la interacción en las distintas escenas de tu juego, hemos utilizado una muchas herramientas y componentes de Unity, tales como Rigidbody2D para la física, Collider2D para detectar colisiones, y scripts en C# para controlar la lógica de cada escena, como el movimiento de Alicia, la generación y comportamiento de obstáculos y cuerpos con los que Alicia puede interactuar en cada nivel, los límites de tiempo, y las condiciones de victoria o derrota

2. Desarrollo y funcionamiento del juego

El juego cuenta con 4 escenas distintas, la escena de MainMenu, la cual se muestra al comienzo, y se ha de presionar el botón de Start para comenzar. Además, cuenta con un botón de Exit en caso de que se quiera abandonar la partida.

Las otras tres escenas: Scene1, Scene2 y Scene3. Cada una de estas escenas representa un nivel del juego, con sus propios componentes y mecánicas.

A pesar de ser distintos niveles, existen ciertos componentes que son comunes a todos ellos:

LevelManager

- Gestiona la lógica de nivel, incluyendo el límite de tiempo para superar el nivel (como es el caso del nivel 1 y el nivel 2), inicio y fin del nivel, y la detección de condiciones de victoria o derrota.

- Están completamente ajustados a las necesidades y características de cada nivel.

- Inicializa los managers de puntuación y de vidas.

GameManager

- Actúa como el nexo central para todos los niveles, manejando cargas de escenas, estados de juego, y eventos globales.

- Mantiene referencias e inicializa ScoreManager, LifeManager, y UIManager.

- Escucha y maneja eventos importantes como completar un nivel, reiniciar el juego, y manejar la vida del jugador.

UIManager

- Administra todos los elementos de la interfaz de usuario como pantallas de Game Over y nivel completado.

- Actualiza la interfaz basada en eventos del juego, como cambios en las vidas del jugador o puntuaciones.

ScoreManager

- Lleva el registro de la puntuación actual y la puntuación más alta.

- Actualiza la puntuación basándose en acciones del jugador, como superar obstáculos o recolectar ítems.

LifeManager

- Gestiona las vidas del jugador, actualizando el conteo con base en los eventos del juego, como colisiones con obstáculos.

- Notifica al sistema cuando las vidas cambian, lo cual puede influir en la interfaz de usuario y en la lógica del juego.

EventManager

-Cada vez que ocurre algo importante en el juego, como un jugador que alcanza un nuevo nivel, es el encargado de mantener informadas a todas las partes del juego que necesitan saber sobre ese evento. Esto permite que diferentes componentes del juego interactúen entre sí sin tener que estar directamente conectados.

- Emplea principalmente el patrón **Observer**. En este patrón, los objetos (observadores) que están interesados en un evento particular se suscriben para recibir notificaciones del sujeto (el EventManager). Cuando se produce un evento, el EventManager notifica a todos los suscriptores registrados, permitiéndoles responder sin necesidad de interacciones directas entre ellos.

Alicia

- Alicia es el personaje principal que los jugadores controlan, y se manipula a través de las flechas del teclado.

- Utilizar el mismo personaje en todas las escenas es una implementación del patrón Singleton porque asegura que solo exista una instancia de Alicia en todo el juego, sin importar cuántas escenas se carguen o cambien. Es muy útil para mantener un estado consistente en el personaje del jugador a través de los distintos niveles del juego.

Estructuras Adicionales

-Además, todos los niveles cuentan con dos pantallas, la pantalla de GameOver cuando no se ha conseguido completar el nivel, y la pantalla de NextLevel, que te permite avanzar al siguiente nivel si lo has completado con éxito.

2.1 Escena 1: El Desafío del Sombrerero Loco

En la primera escena, Alicia ha de evitar una serie de obstáculos lanzados por Sombrerero Loco. El jugador debe controlar que Alicia salte para evitar los obstáculos que vienen en forma teteras y sombreros. Si Alicia logra esquivar los obstáculos durante el tiempo establecido, avanza al siguiente nivel; de lo contrario, el juego se reinicia en esta escena. Además, si Alicia no consigue esquivar un obstáculo, el nivel se reinicia.

Para este nivel hemos empleado un generador de obstáculos. Este sistema sigue el patrón Factory principalmente. A continuación, se describe el funcionamiento de los componentes y cómo se aplican los patrones de diseño:

Componentes Principales

- ObstacleSpawner: Este componente controla la creación de obstáculos en el juego. Es el encargado de decidir cuándo y dónde se crean los obstáculos, utilizando para ello spawnRate y un retraso inicial para que el jugador pueda estar preparado(startDelay). Utiliza dos fábricas específicas, una para cada tipo de obstáculo (sombreros y teteras), eligiendo cuál generar de manera aleatoria.

- Obstacle: Define el comportamiento de los obstáculos. Cada obstáculo se mueve hacia la izquierda a una velocidad constante y desaparece después de recorrer una distancia máxima.

- HatFactory y TeapotFactory: Son implementaciones del patrón Factory, heredadas de una clase base ObstacleFactory. Cada fábrica está especializada en crear su tipo específico de obstáculo. Utilizar el Factory es muy útil en situaciones donde se crean varios objetos que comparten una interfaz común. En este caso, permite encapsular la lógica de creación de cada tipo de obstáculo, lo que sería muy útil en caso de querer añadir más tipos de obstáculos.

2.2 Escena 2: El Laberinto del Gato de Cheshire

En el segundo nivel, Alicia cuenta con 60 segundos para lograr escapar del laberinto en el que se encuentra. Sólo hay una única manera de ganar: alcanzar el corazón que se encuentra en la salida del laberinto. Si no logra alcanzarlo en menos de 60 segundos, el nivel se reinicia. Si consigue alcanzarlo, puede avanzar al siguiente y último nivel.

Componentes Principales:

- Tile Palette: Para este nivel se ha creado una nueva Tile Palette para poder construir nuestro laberinto. Se ha creado a partir de un Sprite con múltiples baldosas.

-Grid: Para crear el laberinto se ha creado un grid con dos Tilemaps: un backround, que consiste en un fondo de baldosas en la capa más lejana para poder tener un fondo, y un Ground, que consiste en las paredes del laberinto con las que Alicia puede interactuar, ya que Ground tiene un collider asignado.

-Heart: Este elemento se encuentra al final del laberinto, y al interactuar con Alicia (mediante un collider) manda la señal que consiste en la condición de victoria.

2.3 Escena 3: Escape de la Reina de Corazones

El nivel final consiste en una enarenación entre Alicia y un soldado de la Reina de Corazones, quien la persigue con inteligencia artificial. Para superar este nivel, Alicia tienen que alcanzar la llave final antes de ser atrapada por el soldado. Si el soldado alcanza a Alicia antes de que llegue a la llave, muere y el nivel se reinicia.

Componentes Principales:

- Tile Palette: Para este nivel se ha creado una nueva Tile Palette para poder construir nuestro laberinto. Se ha creado a partir de un Sprite con múltiples baldosas.

-Grid: Para crear el laberinto se ha creado un grid con dos Tilemaps: un backround, que consiste en un fondo de baldosas en la capa más lejana para poder tener un fondo, y un Ground, que consiste en las paredes del laberinto con las que Alicia puede interactuar, ya que ground tiene un collider asignado.

-CardSoldier: La función del soldado de la Reina de Corazones es perseguir a Alicia y alcanzarla antes de que ella consiga la llave para poder escapar. Para ello se usa el sistema de navegación NavMesh de Unity. Este sistema permite al soldado perseguir a Alicia a través del nivel, ajustando su ruta en tiempo real para evitar obstáculos y mantenerse en el camino más directo hacia ella.

-Key: Este objeto se encuentra en medio del grid, y, al igual que el corazón del nivel anterior, al interactuar (mediante un collider) con Alicia manda la condición de victoria.

3. Comparaciones con el modelo original

En general, nuestro proyecto coincide con nuestra idea original. Estos son los cambios realizados y las razones por las cuales los hemos hecho:

3.1 Cambios en el nivel 1

Este nivel ha sido prácticamente igual a nuestra idea original, salvo por un aspecto:

Nuestra idea original era incluir unos relojes mágicos que Alicia podía recolectar para parar la aparición de obstáculos durante un tiempo predeterminado. El problema que nos encontramos con esto era que el juego era muy aburrido durante ese tiempo, asi que optamos por no incluir estos relojes mágicos y doblar la cantidad de obstáculos que aparecían en pantalla. Para mantener la idea original de los relojes de alguna manera, incorporamos un contador en el LevelManager, de tal manera que si Alicia lograba esquivar los obstáculos durante un tiempo predeterminado, superaba el nivel.

3.2 Cambios en el nivel 2

Al igual que en el nivel anterior, este nivel es muy parecido a nuestra idea original, salvo por un par de cambios:

Nuestra idea original consistía en un laberinto formado por baldosas rosas y moradas, de tal manera que Alicia solo pudiera caminar y moverse a través de las baldosas rosas, y moría si tocaba las moradas. El problema que nos encontramos con esta idea es que el juego no era muy dinámico, ya que era muy difícil no tocar las baldosas y la partida se reiniciaba todo el rato.

Para solucionar este problema decidimos que Alicia no pasaba nada si Alicia tocaba las celdas de corazones (las celdas que originalmente iban a ser moradas), pero que solo podía moverse a través de las celdas rosas. Para complicar el juego, decidimos poner un límite de tiempo para que Alicia lograse salir del laberinto, y un objetivo que tiene que alcanzar para pasar al siguiente nivel: el corazón que se encuentra a la salida.

3.2 Cambios en el nivel 3

Este nivel es con diferencia el que más cambios ha sufrido se los tres, pues no tiene mucho que ver con nuestra idea original.

Nuestra idea original consistía en un participar en un juego de cartas contra la Reina de Corazones, en el cual tenías que hacer el máximo número de parejas con las cartas. Al final, decidimos que esta idea era muy simple y que no encajaba con el resto de niveles, así que

nos decidimos por hacer un nivel en el que Alicia tiene que huir de un soldado de la Reina, que está controlado con Inteligencia Artificial, y lograr alcanzar la llave para superar el nivel.