Manual de Desarrollo

Proyecto Gwent - Harry Potter Edition

"Oscuros y difíciles tiempos nos aguardan. Pronto todos tendremos que decidir entre lo que es correcto y lo que es fácil."

- Albus Dumbledore

Rachel Mojena González

C122

Introduccion:

En el juego de cartas Gwent ambientado en el mundo de Harry Potter, los jugadores asumen el papel de líderes de dos poderosas casas de Hogwarts: Gryffindor, liderada por Harry Potter, y Slytherin, liderada por Lord Voldemort. La batalla entre estas dos casas icónicas determinará el destino de la escuela de magia y hechicería más famosa del mundo.

Cada jugador posee de un mazo con cartas que representan personajes, criaturas y hechizos del universo de Harry Potter. Las cartas están divididas en diferentes categorías, cada una con sus propias habilidades y estrategias únicas.

Con una combinación de estrategia, planificación y un poco de magia, los jugadores de Gwent: Harry Potter Edition pueden sumergirse en un emocionante enfrentamiento entre el bien y el mal, mientras experimentan la emoción y la intriga del universo mágico de Harry Potter.

El juego se encuentra desarrollado con tecnología .NET 7.0 utilizando Unity como interfaz grafica, lenguaje C#.

Funcionalidad del Juego:

La implementacion backend en Unity esta dada por una serie de scripts con distinta funcionalidad que están interralacionados. A continuación se detallara sobre cada uno de los utilizados.

Script Card: define la clase de objetos que representan las cartas en el juego.

1. Propiedades de la clase:

- points: Un entero que representa los puntos de poder de la carta.

- house: Un enum que indica la facción a la que pertenece la carta: Gryffindor o Slytherin.

- unidad: Un enum que describe el tipo de carta: Oro, Plata, Clima, Aumento, Despeje o Señuelo.

- campo: Un array de enums que enumera los posibles lugares donde puede colocarse la carta en el campo de juego, como Cuerpo a Cuerpo, Asedio, Distancia, etc.

- name: Un string que representa el nombre de la carta.

- hability: Un enum que indica la habilidad especial asociada con la carta.

- description_skills: Una cadena de texto que describe la habilidad especial de la carta.

2. Enums:

- Leaderfaction: Facciones de los líderes, Gryffindor y Slytherin.

- Kindofcard: Enumera los tipos de cartas disponibles, como Oro, Plata, Clima, Aumento, Despeje y Señuelo.

- Place: Enumera los posibles lugares en el campo de juego donde se puede colocar la carta.

- Superpower: Enumera las habilidades especiales asociadas con las cartas, como agregar aumento, agregar clima, limpiar una carta del oponente, robar una carta, etc.

3. Descripción de habilidades:

- Cada habilidad especial tiene una descripción asociada que explica lo que hace.

- Las habilidades incluyen agregar aumento/clima, limpiar cartas, robar cartas, afectar el campo con clima/aumento, eliminar unidades de clima, colocar señuelos, y cartas que no son afectadas por habilidades especiales.

Script hogwartsdeck: controla la funcionalidad del mazo de cartas en el juego.

1. Variables públicas: El script define varias listas públicas para almacenar objetos del juego, como las cartas del mazo (Deck), las posiciones de las cartas (Pos), las cartas en la mano del jugador (Hand), los puntos de las cartas (Points), y varias listas de posiciones específicas para diferentes tipos de cartas (PosCuerpoacuerpo, PosDistancia, PosAsedio, PosAumento, PosClima, PosDespeje), y una lista para el cementerio (Cementerio).

2. Método Start(): Este método se llama al inicio del juego. Barajea el mazo de cartas llamando al método Shuffle() y luego roba 10 cartas del mazo llamando al método rob().

3. Método Update(): Este método se llama en cada frame. Se encarga de actualizar la visualización de las cartas en la mano del jugador llamando al método muestracarta() y verificar si hay cartas en las diferentes posiciones llamando al método verificards().

4. Método rob(int count): Este método permite al jugador robar cartas del mazo. Toma un parámetro count que determina cuántas cartas se roban del mazo. Las cartas se agregan a la lista Hand y se eliminan del mazo Deck.

5. Método Shuffle(): Este método barajea el mazo de cartas. Genera un número aleatorio para seleccionar una carta del mazo y la agrega a una nueva lista barajeada, luego la elimina del mazo original. Repite este proceso hasta que se hayan barajeado todas las cartas del mazo.

6. Método muestracarta(): Este método actualiza la visualización de las cartas en la mano del jugador. Asigna la textura de las cartas a las imágenes en blanco que representan la mano del jugador.

7. Método verificards(): Este método verifica si hay cartas en las diferentes posiciones del juego, como las posiciones de cartas cuerpo a cuerpo, a distancia, de asedio, de aumento y del clima. Ajusta la escala de las imágenes en función de si hay cartas en esas posiciones o no.

Script GameManager: se encarga de manejar la lógica del juego, incluyendo el conteo de puntos, el inicio de nuevas rondas, la determinación del ganador y la limpieza de la escena al finalizar una ronda.

2. Métodos `Start()` y `Update()`:

- En Start(), inicializa algunas variables y componentes necesarios para el juego.

- En Update(), se encarga de ejecutar la lógica del juego durante cada fotograma, como verificar si hay un ganador, actualizar los contadores de puntos y rondas, y manejar el cambio de turnos.

3. Métodos de Conteo:

- contador(): Calcula los puntos de cada jugador sumando los valores de las cartas en sus respectivos campos. También considera los efectos de las cartas de aumento y clima.

- new_round(): Controla el inicio de una nueva ronda, determinando el ganador de la ronda anterior y reiniciando los puntos y campos de juego.

4. Métodos de Limpieza:

- clean_scene(): Limpia los campos de juego, eliminando las cartas que estaban en juego.

- end_round(): Limpia todo el campo al final de una ronda.

5. Métodos de Determinación del Ganador:

- winner(): Determina si hay un ganador en función de las rondas ganadas por cada jugador. Si un jugador alcanza 2 rondas ganadas, se activa la imagen correspondiente al ganador.

6. Métodos de Utilidad:

- mayor_carta(): Determina la carta con mayor poder en una lista de cartas.

- Puntos_asociados(): Devuelve la lista de puntos asociados a una fila específica del campo.

Script menuoptions: proporciona la funcionalidad básica para iniciar el juego, regresar al menú principal y salir del juego.

1. Método Play_Game():

- Este método se llama cuando se presiona el botón para comenzar el juego.

- Utiliza SceneManager.LoadScene() para cargar la escena del juego, identificada por el índice 1 en el Build Settings.

2. Método Play_Menu():

- Se llama cuando se desea regresar al menú principal.

- Utiliza SceneManager.LoadScene() para cargar la escena del menú principal, identificada por el índice 0 en el Build Settings.

3. Método Quit_Game():

- Este método se llama cuando se presiona el botón para salir del juego.

- Imprime un mensaje de depuración "Quit" en la consola.

- Utiliza Application.Quit() para salir de la aplicación. Este método solo funciona en compilaciones específicas (como en el caso de una compilación ejecutable), pero no funciona en el editor Unity.

Script cardaction: es responsable de gestionar las interacciones y acciones relacionadas con las cartas del juego.

1. Variables:

- Declara variables para almacenar referencias a elementos visuales como la imagen de la carta, mazos de cartas, paneles de opciones, texto de descripción, etc.

- También contiene referencias a otros scripts como GameManager y objetos del juego como los líderes (Leader1 y Leader2).

2. Métodos Start() y Update():

- En Start(), inicializa varias variables y objetos necesarios para el funcionamiento del juego, como la imagen de la carta, los mazos, y los paneles.

- Update() llama al método muestrapuntos() para actualizar la visualización de los puntos de las cartas durante cada fotograma.

3. Métodos de Interacción:

- Opciones(): Activa el panel de opciones cuando se selecciona una carta durante el turno del jugador correspondiente. Muestra la imagen y descripción de la carta seleccionada.

- leader1() y leader2(): Muestran la habilidad del líder seleccionado en el texto de la descripción.

- muestrapuntos(): Actualiza los puntos de las cartas en el campo de juego visualmente.

4. Métodos de Efecto:

- searching_effect() y effect(): Estos métodos manejan los efectos especiales de algunas cartas, como robar cartas adicionales o aplicar efectos al campo de juego.

5. Métodos de Utilidad:

- campocarta(): Verifica si una carta tiene una propiedad específica en su campo.

- ListWithMayorElement(): Encuentra y devuelve la lista que contiene la mayor cantidad de elementos no nulos entre tres listas dadas.

- CountNonNullTextures(): Cuenta el número de elementos no nulos en una lista de imágenes.

6. Método Invocar():

El propósito principal de este método es permitir a los jugadores invocar cartas en diferentes campos (cuerpo a cuerpo, distancia, asedio, etc.) durante su turno en el juego.

Funcionamiento:

1. Verificación de turno y condiciones: El método verifica si es el turno del jugador y si aún no se ha confirmado que haya jugado en el turno actual.

2. Validación de carta: Verifica que haya una imagen de carta disponible y que la carta que se está invocando pertenezca al jugador que tiene el turno actual.

3. Selección del campo: Dependiendo del campo seleccionado (cuerpo a cuerpo, distancia, asedio, etc.), el método coloca la carta invocada en la posición correspondiente en el tablero de juego.

4. Efectos adicionales: Luego de invocar la carta en el campo seleccionado, se buscan y aplican efectos adicionales de la carta invocada.

5. Actualización de estado: Se actualiza el estado del juego y se confirma que el jugador ha jugado una carta en su turno.

6. Movimientos y eliminación de cartas: Se mueven las cartas correspondientes del jugador de su mano al campo de juego, se eliminan de la mano del jugador y se agregan al cementerio del mazo.

7. Gestión de eventos especiales: Se manejan eventos especiales como la carta "Señuelo".

Estructura:

- El método está dividido en secciones para cada tipo de campo (cuerpo a cuerpo, distancia, asedio, etc.), lo que facilita la comprensión y el mantenimiento del código.

- Se utilizan bucles for para recorrer las diferentes posiciones del campo y encontrar una posición vacía donde colocar la carta invocada.

- Se llama a la función campocarta() para validar si la carta pertenece al tipo de campo seleccionado.

Script Buttonaction: es responsable de manejar las acciones relacionadas con los botones en el juego.

1. Variables públicas: El script define variables públicas para referenciar objetos del juego, como un panel de opciones, un script de acciones de carta (cardaction) y el administrador del juego (GameManager).

2. Función `invocacion()`: Esta función se llama cuando se presiona un botón de invocación. Desactiva el botón de invocación, activa el panel de opciones y asigna la carta asociada al panel de opciones.

3. Funciones de invocación específicas: Hay varias funciones, como invocarCuerpoacuerpo(), invocarDistancia(), invocarAsedio() e invocarDespeje(), que se llaman cuando se presionan botones específicos en el panel de opciones. Cada función llama al método invocar() del script cardaction para invocar cartas relacionadas con el tipo de juego seleccionado.

En resumen, el script Buttonaction facilita la interacción del jugador con el juego, permitiéndole invocar diferentes tipos de cartas y controlar el flujo del juego al activar y desactivar paneles de opciones.

Script ChangeTurn: maneja el cambio de turno en el juego y controla la lógica asociada con la interfaz gráfica de usuario (GUI).

1. Variables públicas:

- GameManager: Una referencia al GameManager que controla la lógica principal del juego.

- GryffReverse: Una referencia al GameObject que representa el reverso de las cartas de Gryffindor.

- HandG: Una referencia al GameObject que representa la mano de cartas de Gryffindor.

- SlythReverse: Una referencia al GameObject que representa el reverso de las cartas de Slytherin.

- HandS: Una referencia al GameObject que representa la mano de cartas de Slytherin.

2. Variables de estado:

- Gryff_changeit: Un booleano que indica si Gryffindor ha realizado un cambio de cartas.

- Slyth_changeit: Un booleano que indica si Slytherin ha realizado un cambio de cartas.

3. Método `cambioturno()`:

- Este método se llama cuando se presiona el botón para cambiar de turno.

- Si el turno actual es de Gryffindor, cambia el turno a Slytherin y realiza las siguientes acciones:

  - Activa el reverso de las cartas de Gryffindor y muestra el reverso de las cartas de Slytherin.

  - Habilita el botón de cambio de cartas si Slytherin no ha realizado un cambio de cartas; de lo contrario, lo deshabilita.

- Si el turno actual es de Slytherin, hace lo contrario.

- Establece ConfirmaTurno en false, lo que permite que se pueda invocar una carta en el nuevo turno.

Script ChangeCards: maneja la lógica para permitir a los jugadores cambiar dos cartas aleatorias de su mano durante el juego.

1. Variables públicas:

- hogwartsdeck1: Una referencia al mazo de cartas de Gryffindor.

- hogwartsdeck2: Una referencia al mazo de cartas de Slytherin.

- cardaction: Una referencia al script cardaction que maneja las acciones relacionadas con las cartas.

- carta_mano: Una variable para almacenar el índice de la carta en la mano.

- GameObject: Una referencia al GameObject del botón mismo, que se desactiva después de un cambio de carta.

2. Método `Cambiar_carta()`:

- Este método se llama cuando se presiona el botón para cambiar cartas.

- Utiliza un objeto System.Random para generar números aleatorios.

- Elige dos índices aleatorios dentro del rango del tamaño de la mano de cartas de la facción actual.

- Si el turno es de Gryffindor, elimina las cartas seleccionadas de la mano de Gryffindor, roba dos cartas nuevas y desactiva el botón.

- Si el turno es de Slytherin, hace lo mismo pero con el mazo de Slytherin.

- Establece las variables Gryff_changeit o Slyth_changeit en true para confirmar que se realizó el cambio de cartas, lo que afectará la lógica de activación del botón en el siguiente turno.

"Sólo porque tienes el poder, no significa que tengas el derecho."

- Hermione Granger

"No tengas piedad, porque no recibirás ninguna."

- Bellatrix Lestrange