RANDOM DUNGEON

1er Proyecto de Programación

Descripción breve

Videojuego de consola multijugador por turnos desarrollado en C# sobre la idea de implementar un sistema de generación dinámica del mapa y personajes con habilidades propias.

Abner Alexandro Abreu Tamayo
Grupo C-121 de Ciencias de la Computación

Descripción

Random Dungeon se crea bajo la premisa de ofrecer un videojuego multijugador con mapas aleatorios y partidas variadas. En cada partida los jugadores, que pueden ser 2 o 4, construyen su personaje seleccionando entre varias clases con habilidades únicas que les ofrecen estrategias únicas para alcanzar la victoria. Estos personajes son colocados en el mapa previamente generado y tienen como objetivo alcanzar la habitación central lo más rápido posible enfrentándose a dos desafíos principales: descifrar los complejos caminos del laberinto y las diversas trampas que en ellos se esconden, haciendo que en cada movimiento se pueda cambiar completamente tu situación.

¿Cómo ejecutar el proyecto?

- Descargar el contenido del repositorio desde Random Dungeon
- Abrir la ubicación del proyecto desde la terminal o algún editor de texto/IDE
- Ejecutar el comando dotnet restore en la terminal para descargar archivos necesarios
- Ejecutar el comando dotnet run desde la terminal o la acción equivalente en el editor

Requisitos:

- DotNet 8
- Librería Spectre.Console (usar dotnet restore)
- Pueden existir problemas de compatibilidad con terminales antiguas (recomendado PowerShell 10 o algún equivalente o superior)

¿Cómo jugar?

1. Inicio de la Partida:

- Seleccionar el número de jugadores (2 o 4)

- Selecciona el tamaño del Mapa:

  - Pequeño (15 x 15): Para partidas rápidas [recomendado para 2 jugadores]

  - Mediano (25 x 25): Una experiencia equilibrada, ideal para cualquier partida

  - Grande (35 x 35): Para partidas más prolongadas [recomendado para 4 jugadores]

2. Creación de personajes:

- Cada Jugador crea el personaje con que jugará la partida:

  - Nombre: Entre 3 y 10 caracteres, no se permiten nombres repetidos

  - Clase: Escoger una de las clases, cada una con su propia habilidad

3. Partida:

- Al principio de cada turno regenera 3 de energía y 1 de maná

- El Jugador Activo se visualiza en verde en el mapa

- Cada Jugador puede:

  - Moverse: Se traslada por el mapa siempre y cuando le quede energía

  - Estado: Permite comprobar el estado del personaje y algunas estadísticas

  - Usar Habilidad: El personaje activa su habilidad siempre y cuando cuente con maná suficiente

  - Terminar Turno: Pasa al turno del siguiente jugador

4. Condición de Victoria:

Gana el primer jugador en llegar a la habitación del centro del mapa

Nota: El mapa está plagado de trampas invisibles... hasta que caes en ellas ;)

Implementación

Mapa:

Generado de manera que se logre la mayor aleatoriedad posible de acuerdo al tamaño seleccionado antes de empezar la partida, dando una mayor variedad de mapas:

- Genera una matriz cuadrada de nxn inicializando cada casilla como paredes
- Selecciona una de las cuatro direcciones aleatoriamente a partir de una posición inicial y empieza a "cavar" caminos [GenerateMaze]
- En cada iteración decide aleatoriamente (con un 10% de probabilidad) si crear un ciclo (bucles dentro del mapa) en una dirección aleatoria [CreateCycles]
- Una vez generado el laberinto en bruto coloca trampas en cada casilla caminable aleatoriamente con cierta probabilidad de acuerdo a la cercanía a la habitación central atendiendo a tres niveles de cercanía de manera que la dificultad incremente según se acerca la habitación del centro (probabilidad del 10%, 15% y 25% según la posición) [SetTraps]
- Se coloca una habitación en el centro del laberinto (con una dimensión de 3x3 celdas) [SetCenterRoom]
- Después de generado el mapa se colocan los jugadores (en función de su número) en las esquinas [UpdatePlayersPosition]

Generación del Laberinto [GenerateMaze]:

1- Toma una tupla (de dos enteros x, y) como posición inicial y la marca como camino
2- Crea una lista de direcciones con las cuatro direcciones posibles y la ordena aleatoriamente, obteniendo un laberinto con una estructura más interesante
3- Por cada una de las posibles direcciones comprueba si es está dentro de las posiciones válidas (dentro del laberinto) y si en ella se encuentra una pared
   - Si es una posición válida hace un llamado recursivo al propio método pasándole esta nueva posición como posición inicial (vuelve a empezar, pero con parámetros diferentes)
   - Si no es una posición válida pasa a la siguiente dirección y vuelve a comprobar
4- En cada llamado recursivo decide si incluir un ciclo al laberinto (caminos en “círculo“) empleando otro método [CreateCycles]

Creando Ciclos [CreateCycles]:

1- Toma una posición y en cada una de las cuatro direcciones posibles comprueba si se encuentra dentro de los límites del laberinto y si hay un camino

- Si es válido elimina la pared entre la posición inicial y la final convirtiéndola en un nuevo camino
- Si no es válido comprueba la siguiente dirección

Colocando Trampas [SetTraps]:

1- Recorre cada una de las posiciones del mapa y comprueba si en ella hay un camino y si hay espacio para una trampa [FreeSpaceForTrap] (comprueba que no haya otra trampa adyacente)
2- Entonces comprueba la posición:
- Si está en el sector interior (en un radio de 1/5 del tamaño del mapa partiendo del centro) coloca una trampa con un 25% de probabilidad en cada casilla
- Si está en el sector medio (entre un radio de 1/5 y 1/3 del tamaño del mapa) coloca una trampa con un 15% de probabilidad en cada casilla
- Si está en el sector exterior (más allá de 1/3 del mapa) coloca una trampa con un 10% de probabilidad en cada casilla

Colocando la Habitación Central [SetCenterRoom]:

- Calcula la casilla central del mapa y coloca una habitación de 3x3 a su alrededor (todos caminos)

Trampas [ActivateTrap]:

A lo largo de todo el tablero están colocadas trampas que son activadas cuando un personaje cae en una casilla con trampa, entonces selecciona aleatoriamente una de las siguientes:

Trampa de Drenado de Energía [EnergyDrainTrap]: al activar esta trampa el jugador pierde energía (-3 puntos) disminuyendo significativamente su capacidad de movimiento (si la energía original es menor que la perdida queda en negativo afectando el siguiente turno)

Trampa Anti-Maná [AntiManaTrap]: al activar esta trampa el jugador pierde maná (-3 puntos) impidiendo el uso de habilidades o aumentando el tiempo de enfriamiento (si el maná original es menor que el perdido queda en negativo aumentando el tiempo de enfriamiento)

Trampa de Cambio de Pasado [PastChange]: al activar esta trampa el jugador cambia otra a clase (y a su respectiva habilidad) diferente a la suya elegida al azar

Trampa de Re-Aparición de Trampas [TrapRespawnTrap]: al activar esta trampa las trampas de todo el mapa se vuelven a generar, cambiando su ubicación y agregando otras nuevas. Primero elimina todas las trampas existentes, entonces vuelve a generar las trampas y actualiza la posición de los jugadores en el mapa

Trampa de Teletransporte [TeleportTrap]: al activar esta trampa la posición del jugador cambia de acuerdo a la ubicación actual y la nueva posición será siempre una casilla transitable y nunca estará en la habitación central. La nueva función se determina de manera similar a la colocación de trampas [SetTraps] de manera que permanezca en el mismo sector que al activar la trampa.

Movimiento [ShowMovement]:

Cada personaje puede moverse en cuatro direcciones (arriba, abajo, izquierda y derecha) por las casillas transitables mientras tenga energía suficiente

1- Al seleccionar la opción Moverse el jugador usa las flechas del teclado para decidir la dirección de movimiento
2- Determina si el movimiento es válido (dentro de los límites del mapa y que sea hacia una casilla caminable)
3- Verifica si en el destino hay una trampa y, de haberla, la activa
4- Actualiza la posición del personaje y resta un punto de energía [Move]

Habilidades [UseHability]:

Cada jugador puede seleccionar una clase para su personaje al inicio de la partida (esta puede cambiar a lo largo de la partida) a la que se asocia la habilidad correspondiente que pueden activar siempre que tengan el maná requerido (3 puntos)

Guerrero: su habilidad Destructor de Muros [WallDestroyer] selecciona una pared adyacente al azar y la destruye (convirtiéndola en un camino), si se escoge una pared del borde del mapa la habilidad falla

Mago: su habilidad Intercambio [Swap] selecciona al azar un personaje de entre todos los jugadores e intercambia sus posiciones, si la posición del personaje escogido es igual a la suya la habilidad falla

Explorador: su habilidad Instinto [Instinc] chequea las casillas adyacentes y vuelve visibles las trampas, marcándolas en color rojo en el mapa

Invocador: su habilidad Invocar Goblin [Summon]chequea las casillas adyacentes al jugador y elimina las trampas sin revelar su ubicación o si había trampas

Viajero: su habilidad Brisa Refrescante [RefreshingBreeze]aumenta en dos puntos la energía del personaje, permitiéndole realizar dos movimientos extra

Creación del Personaje:

Antes de iniciar la partida cada jugador construye su personaje a su gusto:

- Inserta un nombre, es valido si no es nulo (null), contiene espacios en blanco (“ “) y contiene al menos 3 y no más de 10 caracteres
- Selecciona una clase [playerType] y se la otorga la habilidad asociada a esta

Condición de Victoria [RaceWinned]:

Un jugador cumple con la condición de victoria al alcanzar la habitación central

1- En cada movimiento realizado por el jugador se comprueba la Condición de Victoria comparando la posición del jugador con el interior de la habitación central
2- Si cumple la condición la partida termina y se muestran las estadísticas finales