Universidad San Carlos de Guatemala
Centro Universitario de Occidente
Ingeniería en Ciencias y Sistemas
Laboratorio de Estructura de Datos

PRIMER PROYECTO

Objetivo General

Elaborar algoritmos eficaces y eficientes para la resolución del problema de búsqueda del tesoro dentro de un entorno tridimensional, utilizando estructuras de datos avanzadas.

Objetivos Específicos

1. Utilizar la notación BigO para evaluar la eficiencia de los algoritmos implementados en el sistema.
2. Manejar de manera competente el uso de matrices ortogonales, listas enlazadas y árboles binarios para la gestión y organización de datos dentro del juego.
3. Implementar y comparar diferentes algoritmos de ordenación en la clasificación de datos como enemigos, pistas y puntuaciones.
4. Aplicar conceptos recibidos durante la clase y laboratorios en la implementación del juego, incluyendo estructuras de datos y métodos de ordenación.
5. Manipular de manera eficiente la memoria en el lenguaje de programación c/c++ para el desarrollo de este proyecto.

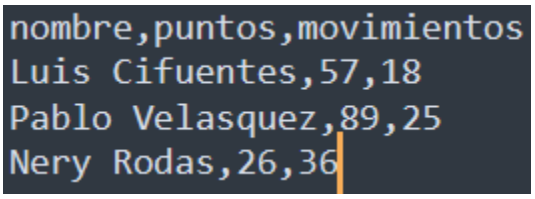
Descripción

Se le solicita desarrollar un juego basado en un tablero tridimensional, representado mediante matrices ortogonales. Un jugador explora este tablero en busca de un tesoro oculto mientras enfrenta desafíos como enemigos, trampas y pistas que indican la cercanía del objetivo. Se implementarán estructuras de datos como árboles binarios para gestionar la información de los enemigos y su ubicación, además de registrar los movimientos del jugador.

Especificaciones del Juego

1. **Tablero Tridimensional**
    - Se representará con matrices ortogonales.
    - Debe permitir dimensiones mínimas de 2x2x2 y ser escalable.
    - Cada nodo del tablero podrá contener diferentes elementos: jugador, tesoro, enemigos, trampas o pistas.
2. **Posicionamiento Aleatorio**
    - El tesoro se colocará aleatoriamente en uno de los nodos.
    - Los enemigos, trampas y pócimas se distribuirán en los nodos restantes de manera aleatoria.
    - Algunas casillas contendrán pistas sobre la ubicación del tesoro.
    - Si el nodo está lejos del tesoro, mostrará "Frío".
    - Si el nodo está adyacente a los nodos adyacentes del tesoro, mostrará “Tibio”.
    - Si el nodo es adyacente al tesoro, mostrará "Caliente".
3. **Árbol Binario para Registro de Enemigo**
    - Cada enemigo se almacenará en un árbol binario de búsqueda.
    - La ubicación de los enemigos se representará como un número concatenado a partir de sus coordenadas (Ejemplo: un enemigo en (1,5,3) será representado como 153).
    - El árbol se ordenará según el nivel (tablero) de los enemigos:
        - Enemigos de nivel bajo estarán en los nodos inferiores del árbol.
        - Enemigos de nivel alto estarán en la raíz o cerca de ella.
4. **Árbol Binario para Registro de Trampas**
    - Cada trampa se almacenará en un árbol binario de búsqueda.
    - La ubicación de las trampas se representará como un número concatenado a partir de sus coordenadas invertidas (Ejemplo: una trampa en (1,5,3) será representado como 351).
    - El árbol se ordenará según el nivel (tablero) de los trampas:
        - Trampas de nivel bajo estarán en la raíz o cerca de ella.
        - Trampas de nivel alto estarán en los nodos inferiores del árbol.

5. **Registro de Movimientos del Jugador**
    - Se llevará un seguimiento de cada movimiento realizado por el jugador.
    - Se registrará si el jugador entra en una casilla con un enemigo o una trampa, y el efecto que esto tenga en su vida.
    - Se registrará cuándo y cómo el jugador encuentra el tesoro.
6. **Historial de Partidas y Ordenación de Resultados**
    - Se almacenarán los datos de cada partida: jugador, puntuación, tiempo empleado y movimientos realizados.
    - Los resultados se ordenarán por:
        - Puntuación.
        - Nombre del jugador.
    - Se mostrará una tabla de clasificación con los mejores desempeños.
7. **Mecánicas del Juego**
    - El jugador podrá moverse en las tres dimensiones del tablero (arriba, abajo, izquierda, derecha, adelante y atrás).
    - Si el jugador encuentra un enemigo, perderá vida según el nivel del enemigo.
    - Si el jugador cae en una trampa, también perderá vida.
    - Si la vida del jugador llega a cero, pierde el juego.
    - El jugador puede encontrar pócimas para recuperar vida.
    - Si el jugador encuentra el tesoro, gana la partida.
8. **Carga CSV**
    - Se debe cargar un csv con los puntajes de los jugadores anteriores teniendo la siguiente estructura:



9. **Reportes**

Cuando el jugador encuentre el tesoro o pierda, se generará un reporte con:

- Nombre del jugador, tiempo total, movimientos y puntuación.

- Ubicación del tesoro y trayectoria del jugador.
- Pistas encontradas y su distancia al tesoro.
- Enemigos enfrentados y trampas activadas.
- Tabla de posiciones con los mejores jugadores.
- Gráfico de los árboles de enemigos y trampas.

Consideraciones

- Documentación Técnica:
  - Se debe incluir la complejidad de cada algoritmo utilizado en el proyecto.
  - La notación BigO debe justificarse adecuadamente para cada implementación.
- Estructuras de Datos Implementadas desde Cero:
  - Los árboles, listas enlazadas y matrices ortogonales deben ser implementados por los estudiantes desde cero.
  - No se permite el uso de arreglos para representar el tablero.
- Prohibición de Librerías de Estructuras de Datos:
  - No está permitido utilizar librerías o herramientas externas que proporcionen estructuras de datos como listas enlazadas, árboles binarios o colas de prioridad.
- Optimización y Análisis de Algoritmos:
  - Se deben comparar diferentes métodos de ordenación para determinar cuál es el más adecuado para cada caso.
  - Se debe analizar el impacto de las estructuras de datos utilizadas en la eficiencia del juego.

Importante

- Lenguaje a utilizar C/C++.
- Es obligatoria la programación orientada a objetos.
- El programa debe ejecutarse en consola.
- Las malas prácticas de desarrollo serán penalizadas.
- Las copias obtendrán nota de cero y se notificará a coordinación.
- Se requiere la creación obligatoria de un archivo Makefile, para obtener el compilado del programa.

- El uso de inteligencia artificial será penalizado hasta un 90%.
- Este proyecto es obligatorio para tener derecho a realizar el siguiente proyecto, así mismo debe obtener una nota mayor a 0 puntos.

Entrega

La fecha de entrega es el día 8 de Abril del 2025. Los componentes se entregarán en un repositorio de Git a través Classroom, siendo estos:

- Código fuente.
- Documentación técnica.
- Manual de Usuario.

Nota

El repositorio debe ser privado, sin embargo debe de incluir los siguientes usuarios como colaboradores: **LuisCif-20, David15Barrera, daaaniel6** para la calificación correspondiente, estos deben ser agregados antes de la fecha y hora de entrega.