75.41 - Algoritmos y Programación II

Cátedra Ing. Patricia Calvo - 1er cuatrimestre 2022

Trabajo Práctico 2: BATALLA CAMPAL V2.0

Objetivo

Generar una pieza de software que simule el funcionamiento del juego Batalla Campal en su versión Jugador contra Jugador.

Enunciado

Batalla campal es un juego de mesa para N jugadores en el que se introducen M soldados por cada jugador en un tablero de X por Y por Z con el fin de que los soldados sobrevivan hasta el final, En cada turno el jugador saca una carta del mazo y ataca una posición del tablero, eliminando si hay un soldado o armamento en ella y dejando la casilla inactiva, luego del disparo, el jugador puede optar por mover un soldado o armamento, ya sea horizontal, vertical o diagonal. Un soldado no se puede mover a una casilla inactiva y si se mueve a una casilla con un soldado contrario, se eliminan los 2 soldados. El juego termina cuando todos los jugadores menos uno se queda sin soldados, ganando el jugador con Soldados. Hay 6 tipos de cartas, 3 establecidas por el enunciado y 3 por el grupo. La carta se juega al final de cada turno. Las cartas establecidas por el TP son: misil, al atacar destruye 27 casilleros (3x3x3), un avión (si está en el aire puede disparar 2 veces adicionales por cada turno) y un barco (si está en el agua puede disparar un misil una vez por cada turno, adicional a los disparos del turno). El terreno tiene 3 casilleros tipos de casilleros, uno es tierra, otro es agua y otro es aire. El nivel 1 del tablero es tierra o agua, y el resto de los niveles son aire.

Interfaz de usuario

Toda la interfaz de usuario debe estar basada en texto, la entrada y salida y el estado del tablero tiene que mostrarse utilizando un Bitmap (ver librería) de una manera ideada por el grupo.

No es necesario que se limpie la pantalla, simplemente escribir el estado del tablero luego de cada jugada.

Cuestionario

Responder el siguiente Cuestionario:

1) ¿Qué es un svn?
2) ¿Qué es git?
3) ¿Qué es Github?
4) ¿Qué es un valgrind?

Normas de entrega

Trabajo práctico grupal: 6 personas. Se deberá elegir un nombre de Grupo.

Reglas generales: respetar el Apéndice B.

El tablero se debe implementar utilizando la clase Lista. Todos los objetos deben estar en memoria dinámica.

Se deberá subir un único archivo comprimido al campus, en un link que se habilitará para esta entrega. Este archivo deberá tener un nombre formado de la siguiente manera:

Nombre de Grupo-TP2.zip

Deberá contener los archivos fuentes (no los binarios), el informe del trabajo realizado, las respuestas al cuestionario, el manual del usuario y el manual del programador (Todo en el mismo PDF).

La fecha de entrega vence el día lunes 05/06/22 a las 23.59hs.

Se evaluará: funcionalidad, eficiencia, algoritmos utilizados, buenas prácticas de programación, modularización, documentación, gestión de memoria y estructuras de datos.

Apéndice A

1) Usar las siguientes convenciones para nombrar identificadores.
    a) Clases y structs: Los nombres de clases y structs siempre deben comenzar con la primera letra en mayúscula en cada palabra, deben ser simples y descriptivos. Se concatenan todas las palabras. Ejemplo: Coche, Vehiculo, CentralTelefonica.
    b) Métodos y funciones: Deben comenzar con letra minúscula, y si está compuesta por 2 o más palabras, la primera letra de la segunda palabra debe comenzar con mayúscula. De preferencia que sean verbos. Ejemplo: arrancarCoche(), sumar().
    c) Variables y objetos: las variables siguen la misma convención que los métodos. Por Ejemplo: alumno, padronElectoral.
    d) Constantes: Las variables constantes o finales, las cuales no cambian su valor durante todo el programa se deben escribir en mayúsculas, concatenadas por "_". Ejemplo: ANCHO, VACIO, COLOR_BASE.
2) El lenguaje utilizado es C++, esto quiere decir que se debe utilizar siempre C++ y no C, por lo tanto una forma de darse cuenta de esto es no incluir nada que tenga .h, por ejemplo #include <iostream> .
3) No usar sentencias 'using namespace' en los .h, solo en los .cpp. Por ejemplo, para referenciar el tipo string en el .h se pone std::string.
4) No usar 'and' y 'or', utilizar los operadores '&&' y '||' respectivamente.
5) Compilar en forma ANSI. Debe estar desarrollado en linux con eclipse y g++. Utilizamos el estándar C++98.
6) Chequear memoria antes de entregar. No tener accesos fuera de rango ni memoria colgada.
7) Si el trabajo práctico requiere archivos para procesar, entregar los archivos de prueba en la entrega del TP. Utilizar siempre rutas relativas y no absolutas.
8) Entregar el informe explicando el TP realizado, manual de usuario y manual del programador.

9) Comentar el código. Todos los tipos, métodos y funciones deberían tener sus comentarios en el .h que los declara.
10) Modularizar el código. No entregar 1 o 2 archivos, separar cada clase o struct con sus funcionalidades en un .h y .cpp
11) No inicializar valores dentro del struct o .h.
12) El tablero en el TP 1 tiene que ser un array dinámico.
13) Si cualquier estructura de control tiene 1 línea, utilizar {} siempre, por ejemplo:

```
for(int i = 0; i < 10; i++) {
      std::cout << i;
}
```