[Año]

AJEDREZ 3D

Manual Técnico

Introducción

En este documento se presentara de cómo dar el mantenimiento que se le puede dar a la aplicación del juego de ajedrez 3D para su fácil comprensión y para que al programador que esté haciendo el mantenimiento le sea más fácil de cómo está programado el juego y entienda estas funciones. Para el desarrollo de esta aplicación se utilizo el lenguaje c++ y el ide QtCreator.

Para el desarrollo de esta aplicación se necesito de estructuras para la elaboración de clases con el objetivo de realizar las funciones necesarias.

Las siguientes estructuras se usaron para tener mejor manejo de memoria. Se trabajo con memoria dinámica, y otras estructuras para el desarrollo de las estructuras principales:

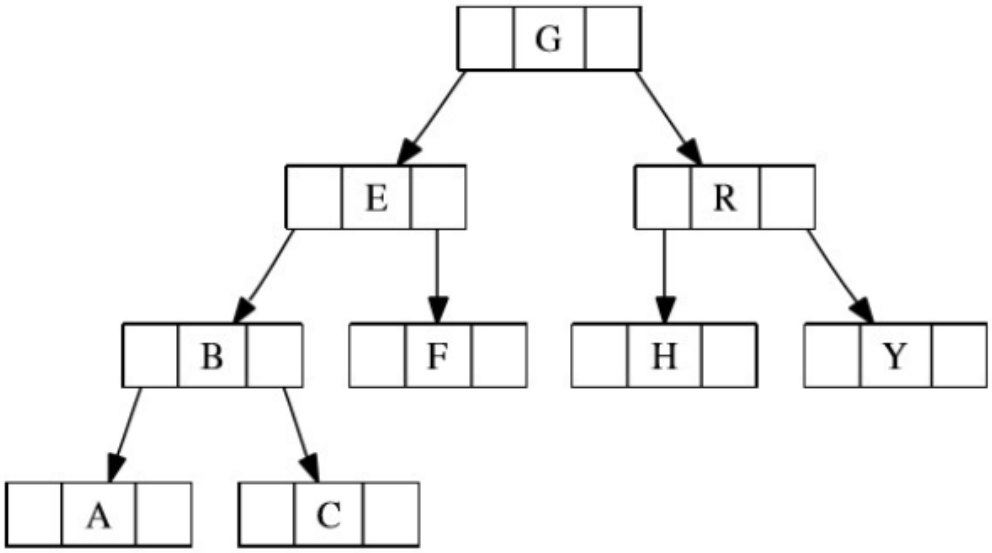
1. Árbol binario de búsqueda.
2. Lista doblemente enlazada.
3. Nodo_Binario.
4. NodoEncabezado.
5. Nodo.

Las siguientes clases se usaron la instrucción struct para el manejo de la tabla, funciones del juego y manejo de las piezas:

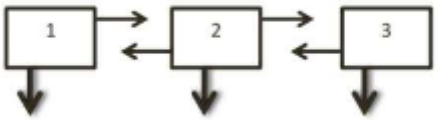
1. Constante.
2. Logica.
3. Ortogonal.
4. Pieza.

Mediante estas estructuras antes mencionadas se realizo el juego de ajedrez 3D.

Árbol Binario de búsqueda: Esta estructura almacena a los jugadores que se cargaron en el juego de menor a mayor y tiene métodos de recorridos que calculan la altura, ramas y número de hojas que tiene el árbol.



Lista doblemente enlazada: Esta lista guarda información de los encabezados de la matriz que se utiliza para el tablero del juego contiene métodos de inserción, eliminación y búsqueda del dato, principalmente se usa el atributo id de la lista. Y tiene enlaces de los nodos accesos a la matriz ortogonal.



Nodo_Binario: Esta clase es el que se encarga de guardar información del jugador con sus datos como el nombre, partidas ganadas y partidas perdidas.

```
struct Nodo_Binario{
    Nodo_Binario * izq;
    Nodo_Binario * der;

    string nombre;
    int p_ganadas;
    int p_perdidas;

    Nodo_Binario(string nombre_, int pg, int pp):izq(0), der(0), nombre(nombre_), p_ganadas(pg), p_perdidas(pp){}

    ~Nodo_Binario(){}
};
```

NodoEncabezado: Esta clase almacena información como id del encabezado de la matriz por lo que se utiliza en la lista doblemente enlazada.

```
struct NodoEncabezado{
    NodoEncabezado * anterior; // arriba
    NodoEncabezado * siguiente; //abajo
    Nodo * acceso;

    int id;
    string nombre;

    NodoEncabezado(int id_):anterior(0), siguiente(0), acceso(0), id(id_){}

    ~NodoEncabezado(){}
};
```

Nodo: Esta clase es la que se utiliza para realizar la matriz por lo que tiene más atributos que los otros nodos y es la que más se utiliza en la aplicación.

```
struct Nodo{
    Nodo * anterior;
    Nodo * siguiente;
    Nodo * arriba;
    Nodo * abajo;

    Nodo * zarriba;
    Nodo * zabajo;

    bool existe;
    int x; // valor de [1-8]
    int y; // cod ascii [H-A][1-8]
    int z; // es el nivel max = 2 [0 1 2]

    Pieza * p;

    Nodo(int x_, int y_, int z_, bool existe_, Pieza * p_){
        anterior = 0; siguiente = 0; arriba = 0; abajo = 0;
        zarriba = 0; zabajo = 0; existe = existe_;
        x = x_;
        y = y_;
        z = z_;
        p = p_;
    }

    ~Nodo(){}

    void setPieza(Pieza * p_){
        p = p_;
    }
};
```

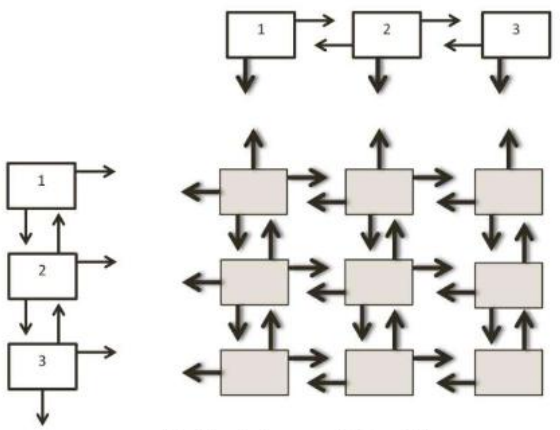
Constante: Esta es una estructura que tiene atributos constantes que ayudan el funcionamiento y legible entendimiento para el uso de las piezas en el tablero, en ella contiene atributos tales como:

- Tipo de pieza.
- Color de pieza.
- Dirección de la imagen de la pieza dependiendo el color.
- Representaciones de los encabezados en vertical.

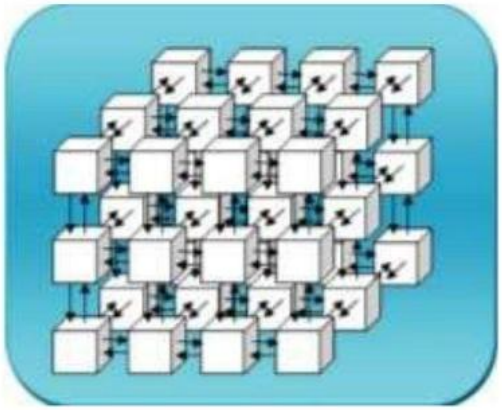
Pieza: Esta clase tiene la información de la pieza como el tipo, color y dirección para el funcionamiento de movilidad de cada pieza.

```
struct Pieza{
    int tipo;
    string color;
    int colors;
    string path;
    Constante cnt;

    Pieza(int tipo_, int color_);

    void asignarPath();

    string obtenerPieza();

    ~Pieza(){}
};
```

Ortogonal: Esta clase es una de las mas principales que está en la aplicación ya que conlleva métodos de inserción de un nodo en la matriz con su nivel que se va agregar, también tiene métodos como eliminar y brusquedad de nodo. En esta clase se manejan los encabezados en horizontal y vertical para el manejo de los niveles. Solo el nivel 0 tiene encabezados, los otros niveles parten del encabezado del nivel 0 para su inserción.



Matriz Ortogonal (Nivel 0)



Cubo Ortogonal

Esta imagen muestra la forma que se genera la matriz mediante la inserción. Además de estos métodos principales tiene otros que son de ayuda para la inserción como el de búsqueda de nodos accesos en cada nivel y métodos que verifican si existe un nodo en cualquier nivel. Así también métodos que hace que se enlacen los nodos en sus direcciones posibles.

Lógica: En esta clase genera los niveles que tiene esta aplicación, además tiene métodos de movimiento de cada pieza y la función de terminado de juego que siempre busca si la dama existe en cada jugador.