Game of Mazes

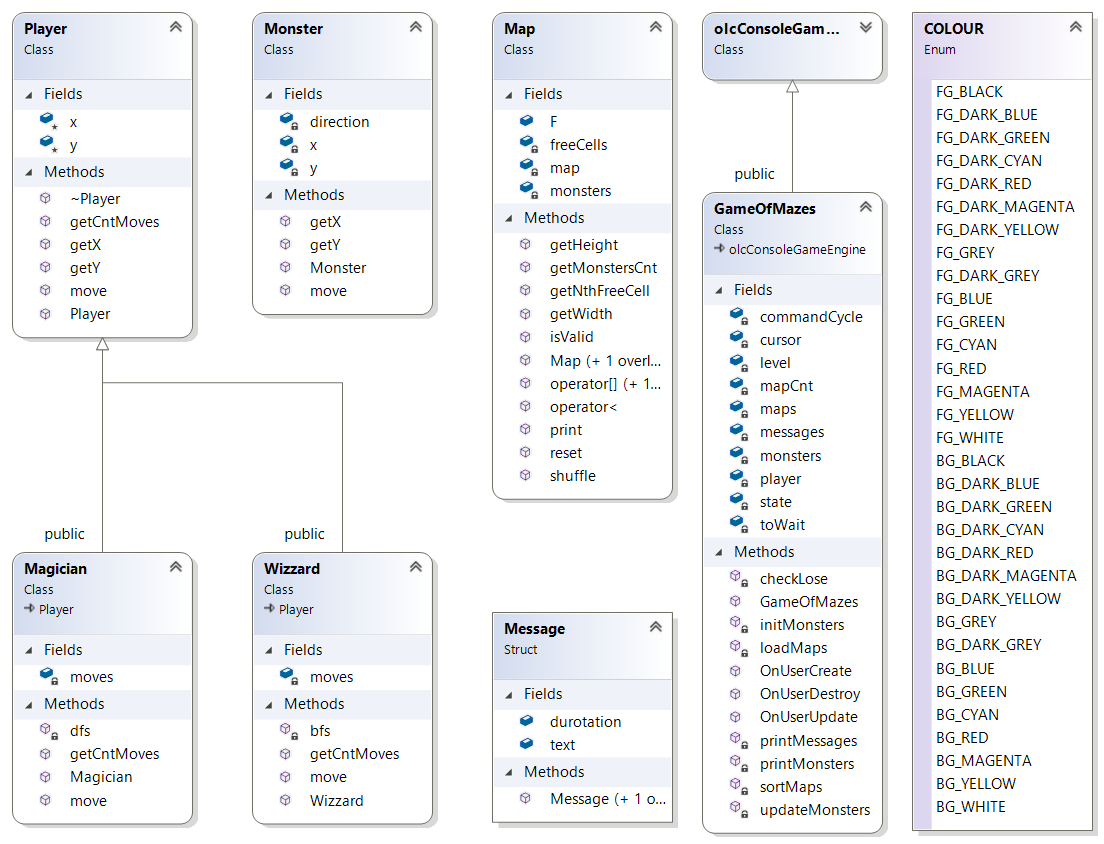
A brief description

Mazes... We see them all through our lives, from children to... programers. The concept is easy: you have to get from a point, called „starting point“ to another one, called „end point“ through some network of paths... and that‘s all. Even though the task is all clear and easy at first glace, the algorithms required for the computation of the problem are even easier – the well known DFS and BFS. Yay! The current project displays, in a fun and iteractive way, how DFS and BFS work.

Architecture

The architecture of the project is as simple as the task itself.

* We have a pure virtual multithreaded olcConsoleGameEngine class, which is being used for more user-friendly UI.
* We have a self-called GameOfMaze class, which inherits olcConsoleGameEngine and is the core class, connecting all others.
* We have a class Map, which stores the information for a single maze.
* We have a pure virtual Player class, whcih is being inherited by classes Magician and Wizzard - the representatives of the two traversal algorithms.
* We have a class Monster, representing the enemy of the user.



olcConsoleGameEngine

Data members (protected):

struct sKeyState

{

bool bPressed;

bool bReleased;

bool bHeld;

} m\_keys[256], m\_mouse[5];

int m\_mousePosX;

int m\_mousePosY;  
int m\_nScreenWidth;  
int m\_nScreenHeight;  
short m\_keyOldState[256] = { 0 };  
short m\_keyNewState[256] = { 0 };  
bool m\_mouseOldState[5] = { 0 };  
bool m\_mouseNewState[5] = { 0 };  
bool m\_bConsoleInFocus = true;

CHAR\_INFO \*m\_bufScreen; //the screen buffer  
std::wstring m\_sAppName; //the title of the console  
HANDLE m\_hOriginalConsole; //handle to the standart console  
CONSOLE\_SCREEN\_BUFFER\_INFO m\_OriginalConsoleInfo; //screen buffer to the standart console  
HANDLE m\_hConsole; //output handle to the current console  
HANDLE m\_hConsoleIn; //input handle to the current console  
SMALL\_RECT m\_rectWindow; //struct having left, top, right, bottom

Member functions (public):

int ConstructConsole(int width, int height, int fontw, int fonth);  
virtual void Draw(int x, int y, short c = 0x2588, short col = 0x000F);  
void Fill(int x1, int y1, int x2, int y2, short c = 0x2588, short col = 0x000F);  
void DrawString(int x, int y, std::wstring c, short col = 0x000F);  
int ScreenWidth();  
int ScreenHeight();  
sKeyState GetKey(int nKeyID);  
sKeyState GetMouse(int nMouseButtonID);  
int GetMouseX();

int GetMouseY();  
void Start(); //creates the game thread: thread(&olcConsoleGameEngine::GameThread, this);

// User MUST OVERRIDE THESE!!

virtual bool OnUserCreate() = 0;

virtual bool OnUserUpdate(float fElapsedTime) = 0;

// Optional for clean up

virtual bool OnUserDestroy();