五子棋函数接口

这里是五子棋游戏的函数声明

文档版本 V1.0

最后更新于2019年12月22日

文档编著者 梦之翼bug小组

main类

```
//判断是否有winner出现
void checkWinner();

//电脑回调函数，p是电脑下棋的位置
void computerCallback(POSITION p);

//游戏初始化函数，创建必要的对象
void initNew();

//从配置文件获取配置参数并更新全局变量
void getConfig();

//将全局变量保存的配置文件
void saveConfig();

//释放对象
void freeNew();

//设置界面，应用设置的更改，hDlg为对话框的句柄
void applySetting(HWND hDlg);

//开始游戏
//mode 游戏的模式，有人机，人人，残局
//firstplayer 谁先走，电脑 或 player
void startGame(int mode,int firstPlayer);

//结束游戏
void endGame();

//悔棋
void takeBack();

//处理player下棋
//p是玩家落子位置
void procPlayerPlayer(POSITION p);

//处理电脑下棋
```

```
//p是电脑落子位置
void procPlayerComputer(POSITION p);

//检查玩家是否超时，
//若超时，着超时方输
int checkTimeout();

//保存局面到文件
void saveBoardToFile();
```

PLAYER类，基类

这是游戏角色的基类：

提取了游戏角色的基础属性 角色名称 角色ID 角色头像

实现通用方法 记录角色的思考时间

```
//构造函数，传入img路径，和MAP类的一个实例指针
PLAYER(PCWSTR imgPath,MAP* map);

//重置计时器
//total为超时时间，单位毫秒
void reset(DWORD total);

//开始计时
void startRecodingTime();

//停止计时
void endRecordingTime();

//返回角色名称
LPCWSTR getPlayerName();

//设置角色名称
void setPlayerName(LPCWSTR name);

//获取剩余时间
DWORD getLeftTime();

//获取角色头像
Gdiplus::Image* getPlayerPortrait();

//需要实现的虚函数，执行下棋操作
virtual void play(POSITION p) = 0;

//返回角色ID
int getPlayerInt();

//设置角色ID
void setPlayerInt(int playerInt);
```

COMPUTER类，基础PLAYER类

实现对游戏引擎（弈心）的操作，并由此来提供AI角色

```
//电脑的初始化函数，开局前调用
void beforeStart();

//轮到电脑落子
//p是玩家落子位置
void turn(POSITION p);

//电脑的思考状态
//true 为正在思考
bool isThinking();

//悔棋
//p为要悔棋的位置
void takeBack(POSITION p);

//设置电脑的难度
//1最垃圾
//2，3，4 难度加大，思考时间变久
void setLevel(int level);

//电脑载入残局数据
void loadHalf();
```

UI_BOARD类

画图类，实现游戏的所有画图操作。

```
//构造函数，
//rc界面的大小
UI_BOARD(Gdiplus::Rect& rc);

//绘制当前游戏图片
//hdc为绘图句柄
void draw(HDC hdc);

//绘制地图信息
//hdc绘图句柄
void drawMap(HDC hdc);

//绘制一个提示框
//p位置
//hdc绘图句柄
void drawTipCircle(HDC hdc,POSITION p);

//更新棋谱图片
```

```
void updateBoard();

//更新角色信息绘图
void updateInfo();

//单独绘制角色信息
void drawInfo(HDC hdc);

//设置绘图的玩家，
//p1，玩家1
//p2，玩家2
void setPlayer(class PLAYER* p1,class PLAYER* p2);

//个体UI类设置数据源
//map数据源
void setMap(class MAP* map);

//设置背景透明度
void setBoardTransparent(float alpha);
```

MAP类

游戏的地图数据类

主要用于处理游戏的所有数据

尝试实现游戏的数据处理与绘图分离

```
//初始化数据
void init();

//设置先行玩家
void setFirstPlayer(int player);

//获取先行玩家的ID
int getFirstPlayer();

//设置游戏模式
//mode 人人，人机，棋谱
void setMode(int mode);
//获取游戏模式
int getMode();

//落子
//p落子位置
bool putChess(POSITION p);

//悔棋，
//返回，悔棋的位置
POSITION takeBack();
```

```
//获取当前思考玩家的ID
int getCurPlayer();

//获取，棋谱某指定位置的落子情况
//x,y 为棋盘位置
//返回，落子情况
int boardIndex(int x, int y);

//获取某一步的落子位置
//index 第几步
POSITION moveIndex(int index);

//获取当前总的步长
int getSumSteps();

//获取，数据的长度
int getTotalIndex();

//获取最后一次落子的位置
POSITION getLastPos();

//是否出现获胜者
int hasWinner();

//棋盘某个位置是否为空
bool isEmpty(POSITION p);

//查看棋谱的下一步
bool next();
//查看棋谱的上一步
bool prev();

//从文件中载入棋盘数据
//filename要读取的文件名
bool loadBoardFromFile(const char* filename);
//将期棋盘数据保存到文件
//filename要保存的文件名
bool saveBoardToFile(const char* filename,int mode);
```

MUSIC类

```
//初始化，加载所有声音文件
void initMusic();

//关闭所有声音文件
void closeMusic();

//播放落子声音
void playPutchessMusic();

//播放背景音乐
```

```
    void playBkMusic();

    //停止播放背景音乐
    void stopBkMusic();

    //播放获胜音乐
    void playWinMusic();
    //播放失败音乐
    void playLoseMusic();

    //根据配置，更新播放音乐
    void updateBkMusic();
```