Qt大作业（Kriegspiel）设计文档

2020010909 智00 刘益枫

1. 设计背景：本游戏旨在运用Qt和网络编程相关技术，实现联机军棋（翻棋）游戏。
2. 设计思路：将行棋部分单独列为一个源文件，处理内部逻辑，而将阵营切换、信息处理、网络连接等功能合并到主窗口文件下进行处理。

1.Chess.h和Chess.cpp文件：

1）定义棋子（Chess）和阵营（Side）两个类:

Chess类用来存储每个棋子的死活（bool live）、值（int value）、名字（std::string photo）、是否被翻开（bool known）、位置（int pos[2]）等信息，在构造函数中，根据每个棋子的值确定所属图片的后缀（“xxx.png”）。约定司令、军长、师长、旅长、团长、营长、连长、排长、工兵、地雷、炸弹、军旗的值分别为10、9、8、7、6、5、4、3、2、1、0、-1。

Side类存储每一阵营所属棋子的信息，包括：

所属棋子（Chess \* sub[25]）；

该方地雷剩余数量（int dileinum）；

军旗是否存活（bool flagLive）。

在构造函数中，对25个所属棋子的值进行了规定。

另外还有函数（void）initializeSide（bool col）对阵营进行初始化。

此外，Chess和Side两个类各有指针指向对方。

2）通过对一般棋子、工兵、地雷、炸弹、军旗五种棋子的枚举，实现了检查单个棋子是否可移动（walkable）和某阵营所有棋子是否可移动（check\_walkable）的功能，能够检查某棋子能否被响应，以及能够检查能否移动到指定位置（canGet），能够对棋盘进行初始化操作（setIntPos），以及根据网络服务器传递的信息同步网络双方的棋盘（syninit、setbase）。检查某棋子移动到指定位置时，对在铁路上移动的工兵用深度优先搜索技术搜索。

（bool）Chess：：walkable（）函数：检查该棋子是否能走，通过该函数判定全阵营是否存在可走棋子，以及第一次选择是否有效。首先，地雷、军旗不能移动。其次，只要上下左右、左上、右上、左下、右下8个方向相邻格子存在能行区域即可行棋。斜线方向有公路当且仅当两端点至少有一个为行营。除中心的两处较为特殊之外，其余点只要不超出界外，上下左右均有铁路或公路。若目的地为空，即可行棋，若目的地不为空且为己方子则不行，若为对方子则判定双方子的特性比较。

（bool）Chess：：canGet（int x2, int y2）：检查该棋子是否能移动到目的地。若起讫点不全为铁路上的点，即搜索8个方向上的相邻点是否为终点和是否可走。若全为铁路上的点，则必然通过铁路到达。对一般子采取简单向前搜索方法搜索终点，对工兵，通过search和searching函数采用DFS技术动态搜索。

（void） syninit(Side sid1, Side sid2, int ord, int num0, int num1, int num2, int num3, int num4)：将套接字对方信息同步到本方。

（void） setbase()：初始化行营、铁路等信息。

（void） setIntPos(Side sid1, Side sid2)：初始化棋盘，随机生成棋子位置。

（void） judge(int x, int y, int x\_other, int y\_other)：棋子移动后处理棋子相关信息。

（bool） check(int x, int y, bool stat)：判定格子是否有子、棋子是否存活，棋子是否为我方棋子，棋子是否能移动从而判定能否被选中。

（bool） check\_walkable(Side sides)：通过判定每一个棋子的可移动性从而判定该阵营是否无棋可走。

2.main.cpp、mainwindow.h和mainwindow.cpp文件：

1）UI显示：

利用指针数组初始化60个棋子可能的位置坐标(QLabel\* chess60[12][5])；

利用标签和LCD屏实时显示棋盘、准备信息、显示当前玩家及阵营颜色、双方超时次数等(void reprint()、void setlabel（）、void setbutton（）、void showcolor（）)；

利用QMessageBox弹出警告窗口；

利用菜单栏显示创建客户端、连接客户端、投降（成功、失败）弹窗，且能开始游戏。

2）网络连接：

能够将输入框中的QString转化为字符串数组判定输入的IP地址是否合法(test.setAddress)；

能够识取本地IP地址并成功进行侦听(createreturn)；

能够与客户端通过TCP协议建立连接，在长时间未侦听到信号后能够弹出失败窗口(connectreturn)；

建立连接后将readyRead（）信号关联到readData（）槽函数里；

能够利用套接字技术传输数据并且成功处理数据，用于侦听初始化（“start”、“prepared”、“sucinit”、“sucsyn”、“toinit”、“init”）、翻棋（“roll”）、行棋（“move”）、投降（“surrender”）、胜利（“win”）、失败（“lose”）、超时（“overtime”）、退出（“quit”）、断开连接（“disconnect”、“checkdis”、“yesdis”）等信息，以及同步两者时间（“time”）；

能够断开连接，断开连接时，通过不断交换disconnect（）信号，使得服务器获取主动权，通知客户端断开连接后，等待客户端回应，然后断开连接，关闭服务器；断开连接后，依然能建立新的连接并重新开始。

3）监听器：

能够监听鼠标点击，判断鼠标点击位置的合法性，并根据当前棋局进行情况调用相应函数处理(mousePressEvent())。

4）游戏推进部分：

应用程序启动后，初始化界面，设置60个格子label，设置棋盘信息。

建立连接后，双方点击“start”按钮进行开始游戏确认，然后由服务器端进行初始化，随机确定顺序，并将初始化内容通过两次交换信号同步到客户端（void casting（））。根据初始化的顺序，双方依次进行翻棋、行棋操作，并判定游戏是否结束。

翻棋操作：由监听器监听到合法的鼠标点击的位置后，转化为12×5网格的坐标，再判断该处是否有未翻开的棋子，之后进行翻棋操作并将操作通知给另一端。

行棋操作：由监听器监听到合法的鼠标点击的位置后，转化为12×5网格的坐标，再判断该处是否有已翻开的本方阵营的棋子，之后监听第二次合法操作，判断棋子是否能走到对应位置，成功后进行行棋操作并将操作通知给另一方。（注：本程序允许小棋向大棋处“送死”）

判定游戏是否结束：一方开始回合前，检测双方军棋是否被摧毁（前提是地雷被清除完毕），检测双方是否超时3次，若游戏结束，能够根据结束情况进行弹窗并通知另一端游戏情况。

投降操作：能够判定步数是否达到20步，若未达到，弹出投降失败弹窗，若达到，弹出失败弹窗并通知对方。

时间处理：(timestarted)建立QTimer记录时间，每100毫秒检测是否操作，并通知对方同步时间显示。若超时，更改超时次数显示并通知对方，将操作权移交给对方。

投降操作：当我方回合投降时，发送信息告知对方己方已投降；当对方回合我方投降时，发送信息告知对方己方已投降，但数据会阻塞至缓冲区里，readyRead（）信号被阻塞，这时对方会定时检查缓冲区是否有新数据，从而及时得知投降事宜，并弹窗告知。

中途应用程序关闭：重载closeEvent函数，当本方回合，本方直接关闭应用程序时，发送quit信息告知对方己方应用窗口已关闭；当对方回合，本方直接关闭应用程序时，发送quit信息告知对方己方应用窗口已关闭，但数据会阻塞至缓冲区里，readyRead（）信号被阻塞，这时对方也会定时检查缓冲区时候否新数据，从而及时得知己方退出事宜，并弹窗告知。

重新开始：当游戏结束时，能够显示提示界面，重新开始，并更新棋盘。

3.其他文件：能够显示投降失败弹窗、建立服务器界面、服务器连接输入界面。实现数字软键盘输入。

三、游戏截图：

电脑的屏幕截图

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

图形用户界面, 应用程序

描述已自动生成

表格

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成图形用户界面

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成图形用户界面, 文本, 应用程序

描述已自动生成