李墨珩 2017050025

程序设计第二周大作业说明文档

September 8, 2018

程序设计第二周大作业

网络象棋对战

一：网络通信部分

连接部分：基于 QTcpServer 和 QTcpSocket 建立套接字，initServer( ) 函数用于启用服务器的监听状态；startConnection( ) 函数用于向服务器发出连接请求。启用监听状态在获得请求连接的IP地址时，采用了对话框的形式，并加设软键盘，用来帮助用户输入IP地址，采用QSignalMapper将QPushButton发出的信号添加参数，以简化代码。在启用监听和开始连接时，都会向用户弹出窗口，用户可以选择是否继续等待，还是终止当前操作。

通信部分：采用QByteArray类来传递数据，确保数据的准确传输。发送信息时，先发送关键字，在发送有效信息。所以接收信息的函数 recvMessage( ) 要先对应关键字，对应不同关键字调用不同函数。

二：象棋对战部分

棋盘和棋子的绘制均在Widget类（继承自QWidget），重写了Qpaintevent()函数。

为了记录棋子信息，建立了如下数据：

```
int Chessman_Coordinate[9][10];//棋子的坐标就是数组下标换算而来，值代表了属性
int To_Move[9][10];//选中棋子后提示可移动的位置
int Clicked_X;//记录点击的x坐标，显示点击棋子,并在移动之后给对方提示
int Clicked_Y;//记录点击的y坐标，显示点击棋子
int Step_Number;//记录步数，判断哪一方移动棋子
```

为了记录棋局时间的参数，以实现超时功能，建立了如下数据：

```
int Step_Time_Number;//记录步时
int Sum_Time_Number;//记录局时
QLabel *Player1_Time;//显示玩家一的步时
QLabel *Player2_Time;//显示玩家二的步时
QLabel *Sum_Time;//显示对局时间
QTimer *Timer;//每一秒更新时间
```

游戏过程：

开始游戏，或从文件中读入残局时，首先设置 Chessman_Coordinate 和 To_Move 数组，定位棋子和每个棋子能够行走的位置；其次初始化各项参数，当所有参数初始化完毕时，调用 paintEvent( ) 函数绘制棋盘。

开始计时，QTimer 每次发出信号，主界面显示时间的部件都会更新，当某方时间超过规定上限时，直接导致游戏结束。

响应鼠标点击事件，重写了QMouseEvent( ) 函数，当现在不是我方下棋的时候，点击棋盘将会弹出对话框，提示不是我方落子时间。每当点击棋盘时，通过获取鼠标的坐标来计算点击的是哪一个棋子，并调用To_Move()函数来获取当前棋子可以移动到的位置，重新绘图。当产生有效的落子或吃子的动作时，调用Eat_Chessman()函数，对 Chessman_Coordinate 数组进行更新，并向另一端发出信号以同步棋盘。每当棋盘更新时，检测当前是否已经可以判断输赢，是否构成将军局面。若已经可以判断输赢，直接结束游戏，弹出对话框。若出现将军局面，播放声音。

将军局面的检测，检查黑方当前所有的子的可移动位置，若红方主将在可移动位置内，则构成将军局面。同样，也要检查红方的所有子的可移动位置，看黑方主将是否在内。

认输机制，当点击认输按钮时，直接结束游戏，向另一端发送信息，包含认输的关键字。

计时机制，当超出规定时间，直接结束游戏，向另一端发送信息，包含超时关键字。

保存当前棋局，遍历整个棋盘，获得当前所有棋子的位置信息，存到一个数组里，遍历结束后，整理棋子信息，写入文档，同时向另一端发送保存棋局的关键字，使另一端也进行保存操作。

读取残局，客户端读取某个残局，直接向服务器端发送关键字和读取的残局文件名，服务器端也打开残局文件，准备游戏。

函数的具体说明：

```
public:
    Widget(QWidget *parent = 0);
    ~Widget();
    bool Game_Is_Over;//判断游戏是否结束
    void  Initialize_Array();//初始化Chessman_Coordinate数组
    void Initialize_Move_Array();//初始化To_Move数组
    void paintEvent(QPaintEvent *);//绘图事件函数
    void mousePressEvent(QMouseEvent *event);//鼠标点击事件
    void To_Move_Chessman(int i, int j, int value);//传入待移动的值和坐标，确定棋子可移动位置
    void Eat_Chessman(int Eat_x, int Eat_y, int Eat_value, int Eaten_x, int Eaten_y, int Eaten_value);
    //移动或吃子函数，传入移动棋子前后的坐标和属性，悔棋数组也在此函数里面获取值
    void Is_Win();//判断对局是否结束
    bool Is_Checkmate();//判断当钱是否构成将军局面
    int To_Move_Checkmate[9][10]={0};
    void Check_Checkmate(int i,int j,int value);
    int Eat_xt=0; int Eat_yt=0; int Eat_valuet=0; int Eaten_xt=0; int Eaten_yt=0; int Eaten_valuet=0;

public slots:
    void Time_Out_Slots();//计时函数，时间间隔为1秒
    void Start_Game_Slots();//开始按钮槽函数
    void Regret_Game_Slots();//悔棋函数槽函数
    void Game_Help_Slots();//游戏规则槽函数
    void Exit_Game_Slots();//退出游戏槽函数
    void Read_Game_Slots();//读取残局槽函数
    void Save_Game_Slots();//保存棋局槽函数

signals:
    void move(int,int,int,int,int,int);//向mainwindow发出的信号，与另一端通信
    void startgame();//向mainwindow发出的信号，与另一端通信
    void readgame(QString);//向mainwindow发出的信号，与另一端通信
    void savegame();//向mainwindow发出的信号，与另一端通信
};
```