程序设计训练

Qt 军棋游戏设计文档

于雨琛　经 02-计 08

一.　游戏程序综述

本军棋游戏程序通过 C++语言结合 Qt 编写，主要涉及到了军棋的游戏规则判断、图形界面实现、实现 TCP 联机操作等三个重要部分，并深入使用了 Qt 中的信号槽机制。

程序编写语言为 C++，在程序中用到了 ClientDialog, HostDialog, Main, MyButton, MainWindow, PlayScene 六个类，运用面向对象的设计思想，将不同的功能以不同的类加以封装实现，增加了程序的可读性和复用性。程序编写中将服务器端与客户端分开实现，ClientDialog 从主界面中 Connect to Server 弹出窗口，实现网络传输与接收功能；HostDialog 与之类似，通过点击主菜单的 Create the Connection 触发。这两个类是面向对象的接口，在类中各自封装了一个 PlayScene 的对象，作为游戏界面，而在 PlayScene 中又封装有 MyButton 类的对象，继承自 QPushButton，作为棋子。

本程序实现了正确绘制棋盘棋子，正确判断走子吃子，正确判断胜负，正确连接、断开连接等功能。

二.　客户端、服务器端的工作流程

在整个游戏的流程中，首先由服务器端点击 Create the Connection 按钮，随后点击 Ok，此时服务器端运行 listen()开始监听是否有客户端的连接，随后客户端点击 Connect to Server 按钮，正确输入客户端窗口显示的 IP 地址，点击 Ok 开始连接（端口 port 设置为 10134），连接成功双方同时弹出棋盘界面，棋子均为未翻面状态，且此时棋子不可动。同时弹出界面后，由服务器端点击 Ready to Start 按钮，客户端会弹出弹窗“是否准备好开始？”，在客户端点击确定后双方同时开始倒计时，服务器端先手执子，游戏开始。

游戏过程中，每次一方进行了一步操作后（或者 20 秒超时），将向对方发送相应信号，对方接受到信号后对数据进行处理，更新棋盘，进行下一步操作。

三.　客户端与服务器端通信协议

通信协议以 QTcpSocket 为基础，基于行操作，一方进行一步操作后在 PlayScene 中发送一个信号，ClientDialog 或 HostDialog 根据不同信号发送相应数据，并同时在其中接收。

发送使用 QString 输入转 Utf8 格式，发送代码形如：

```
QString tmp = "Win";//其中 Win 处为传输的数据
m_tcp->write(tmp.toUtf8());
```

接受数据采用 QByteArray 接收，接收代码形如：

```
QByteArray dat = m_tcp->readAll();
```

发送的数据有如下几种：

1．Admit Defeat　　我方投降

2. OverTime　　我方超时
3. Win　　对方输了，我方获胜
4. lose　　对方赢了，我方失败
5. Dis_Connect　　断开连接
6. YouAreRed　　对方阵营为红
7. YouAreBlue　　对方阵营为蓝
8. Start　　询问对方是否开始
9. Agree_Start　　同意开始
10. 180 个字符的数据，每三位一断开，分别是每一个 Button 的是否翻面、Button 上的棋子颜色（红、蓝或空）、Button 上棋子是什么，共 60 个 Button。

由一方发送数据，另一端解析后发送对应信号进入相应 PlayScene 中更新棋盘或弹窗显示胜负。

四.　网络通信编程框架

第一步，服务器端在点击 Create the Connection 弹窗 Ok 后启动监听，代码如下：

```
m_s = new QTcpServer(this);
connect(&Okk, &QPushButton::clicked,[=](){
    m_s->listen(QHostAddress::Any, 10134);//10134 为默认端口 port
});
```

第二步，客户端在点击 Connect to Server 弹窗 Ok 后与服务器连接，代码如下：

```
connect(ui->pushButton_12, &QPushButton::clicked, [=](){
    QString ip;
    ip = ui->lineEdit->text();
    m_tcp->connectToHost(QHostAddress(ip), 10134);
});
```

第三步，服务器端检测连接成功，双方同时显示棋盘，游戏开始。部分代码如下：

```
connect(m_s, &QTcpServer::newConnection, this, [=](){//创建监听的服务器对象
    m_tcp = m_s->nextPendingConnection();
    this->hide();
    emit StartGame();
    Server_play.show();
    ……
}
```

第四步，双方传递数据，互相读取，并进行相应操作。部分代码如下：

```
connect(m_tcp, &QTcpSocket::readyRead, this, [=]()//  检测是否可以接收数据
{
    QByteArray dat = m_tcp->readAll();
    ……
}
```

五.　信号与槽机制设计

信号槽机制在本程序中普遍使用，其中网络通信部分尤其普遍。在双方对战时，鼠标

点击事件（鼠标点击按钮）通过信号槽机制进行，在点击之后进行：

```
//修改当下点击坐标位置
this->ButtonPressed(chessbutton[i]._row, chessbutton[i]._col);
//进行点击的事件操作（判断是否符合 MoveRule，EatRule）
this->mousePressEv();
//判断是否结束
this->Judge();
```

每一回合结束后，由 PlayScene 发送结束信号，ClientDialog 或 HostDialog 接收到信号后，再次进行一个信号槽机制，将需要传递的信息发送给对方，对方收到数据后通过信号槽机制将数据返回给己方的 PlayScene 中，更新棋盘。

```
//例：从 PlayScene 向 HostDialog 发送信号
connect(&Server_play, &PlayScene::ying,[=](){
    QString tmp = "Win";
    m_tcp->write(tmp.toUtf8());
});
//例：接收到对面传送过来的数据
else if(dat=="Win"){
    for (int i=1; i<=60; i++){
    Server_play.chessbutton[i].setAttribute(Qt::WA_TransparentForMouseEvents
    , true);
}
emit Server_play.Recvying();
}
```

六. GUI 界面设计

共设计了三个 GUI 界面，但最终游戏中除了 Connect to Server 的弹窗外其余均为纯代码写入，因为代码的 move()更加精准，位置更可控。

游戏中最重要的棋盘棋子均使用 QPixmap 进行绘制，QLabel, QPushButton, QDialog 等均有所使用，并使用 resize(), setFont(), setText(), move(), setFlat()等函数进行调整。