第二周大作业设计报告

计 82 尤艺霖 2018011324

1. 题目背景

要求使用 Qt 开发一个国际象棋对战，支持一个不完整的国际象棋规则和网络联机，以及对国际象棋残局的读写。

2. 设计思路

首先将项目分割成控制、行棋规则、网络三个模块。每个模块功能相对集中，便于发挥面向对象的思想。网络模块主要负责服务端和客户端的数据交流；行棋规则模块主要控制合法下棋；控制模块负责综合两者，并提供用户交互的界面。

3. 文件结构

-castlingcheck.h/cpp
包含 CastlingCheck 类，是负责检查王车易位合法性的模块
-chessgame.h/cpp
包含 ChessGame 类，负责行棋规则的整体控制，以及残局的读写
-clientdialog.h/cpp/ui
包含 ClientDialog 类，负责控制模块的客户端交互界面
-hostdialog.h/cpp/ui
包含 HostDialog 类，负责控制模块的服务端交互界面
-mainwindow.h/cpp/ui
包含 MainWindow 类，是控制模块的主要功能实现的地方
-networkmodule.h/cpp
包含 NetworkModule 类，负责网络通信
-promotedialog.h/cpp
包含 PromoteDialog 类，负责行棋规则中的升变
-main.cpp
项目的启动文件
-res.qrc
Qt 资源文件，其中 board 前缀下存放两个文本文件，其中一个是开局状态，另一个是测试样例；piece 前缀下存放透明背景的棋子图片

4. 实现细节

1. **行棋规则模块细节-快速开发**
   行棋规则这一带我采取了一定的复制黏贴代码的方法。对于车的移动，我用了四个循环来实现四个方向，终止条件是碰到边界或者其他棋子；而对于象的移动也是相似的四个循环，只需要复制黏贴并修改每个循环的方向即可；王后的移动更是简单，只需要将车和象的移动接在一起即可。同时考虑到王车易位需要判断对方威胁范围的逻辑，因而需要走子的代码段，又可以进行复制黏贴。这里通过一定的复用，大大加快了开发速度。
2. **行棋规则模块细节-升变**
   升变的主要问题是时间的控制，我的选择是对升变框单独计时，在弹出升变窗口的

时候暂停主界面的计时，然后如果升变窗口超时直接判负。这样的设计回避了一些两边同时计时可能出现的问题。

3. **网络模块细节-封装、通信协议和网络通信编程框架**

   我将 QTcpServer 和 QTcpSocket 重新包装成了一个适合自己使用的 NetworkModule 库，支持建立服务器、建立客户端、单对单通信。这样实际上是自己重新封装了轮子，可以在其他地方更方便的使用，比如第二周考试的第三题，我就直接使用了这个 NetworkModule 来完成通信部分。

   为了处理拆包和黏包的问题，NetworkModule 中对发送的数据做了一定的包装，并规定了两端的通信模式，某种程度上可以称之为通信协议。这一块使用了 Stackoverflow 上对于 QDataStream 的应用，首先向 QDataStream 写入一个 quint16 代表数据长度，然后写入需要传输的数据，这里设计成了只能传输 QString。在读取的时候使用一个变量控制待读入的长度，如果 bytesAvailable 足够的话才进行读取，否则等到下一次有数据写入。这样有效的防止了黏包。

4. **控制模块细节-交互界面**

   根据要求，使用了文本输入框，在服务端弹窗中显示本机地址，在客户端弹窗中允许用文本框输入。判断输入是否合法的任务交给了 NetworkModule。获取本机地址的时候用的是 QHostInfo，比 QNetworkInterface 更加靠谱。

5. **控制模块细节-工作流程**

   在双方建立连接之后在两侧建立棋盘，并读入初始状态，白方开始倒计时。随后触发 mouseEvent，控制模块找到点击的是哪个方格，并发送给行棋逻辑模块，行棋模块处理点击事件，如果已经完成了这一步棋，则进行胜负判定，并返回对应的信号。然后控制模块将这一步所作的事情编码，传送给网络模块，网络模块完成通信。另一端的网络模块收到消息之后进行处理，如果游戏未结束，则开始倒计时，并允许 mouseEvent 将信息送到行棋模块。如此循环就完成了一局国际象棋。

5. 总结

这个项目满足了大作业的所有基础需求，并实现了王车易位的规则。同时符合面向对象的思想，具备优秀的扩展性。缺点在于有大量复制黏贴的代码段，虽然提升了开发速度，但是影响了观感。