Software Design Document 软件设计文档

中国象棋对战软件
魏家栋 2017011445
2018年9月9日

简介

设计目标

该软件希望实现一个基于Qt的中国象棋对战图形界面，并基于Qt的network模块进行联机对战。要求实现

- 图形界面显示
- 网络数据传输
- 象棋规则实现

适用范围

适用于中国象棋的单机、联机对战。

文档综述

该文档首先对软件进行了一个概述，然后对软件的结构进行了详细分析。接下来介绍了软件的用户界面和测试用例。最后指出了软件的未来拓展方向。

安装、运行和使用

参见README。

版权

软件概述

中国象棋是起源于中国的一种二人对抗性棋类，在中国有着悠久的历史，玩法通俗易懂，技巧性强，在中国受众广大。该软件实现了中国象棋的图形界面，覆盖了中国象棋的所有规则，实现了联机对战功能，游戏性强。

软件结构

该软件分为5个模块：View-Scene-Controller-Model-Network，本质上采用了MVC（Model-View-Controller）设计模式。

View 模块

View模块负责软件图形化界面的整体布局和展示，包括象棋棋盘/棋子的展示，但不负责象棋棋盘/棋子的显示坐标，不负责棋盘/棋子数据的存储、计算和通讯。
主要包含类MainWindow和类CheeseView。MainWindow继承Qt的类QMainWindow，是程序的主界面，包括菜单栏、工具栏、状态栏，中间部分显示象棋棋盘/棋子，左侧提供认输按钮、倒计时数字、当前轮到哪方走子等，右侧提供缩放滑动条。CheeseView继承Qt的类QGraphicsView，负责显示CheeseScene存储的象棋棋盘/棋子的布局信息，并设定显示模式，如进行象棋棋盘/棋子的显示缩放等。

Scene 模块

Scene模块负责象棋棋盘/棋子的显示坐标，负责响应用户触发的事件，但不负责棋盘/棋子数据的存储、计算和通讯。
主要包含类CheeseScene。CheeseScene继承Qt的类QGraphicsScene，负责存储象棋棋盘/棋子的显示坐标信息，比如棋盘的左上角在（100，100）位置，棋盘的右下角在（900,1000）位置，象棋棋盘网格交点的逻辑坐标与显示坐标的对应，棋子显示得多大，等等。它还负责响应用户触发的事件，比如鼠标单击、键盘按键等。
将View模块和Scene模块分离，一是象棋棋盘/棋子的显示坐标本身较为繁多、复杂，单独列作模块有利于软件整体复杂度降低；二是可以为一个Scene配置多个View，比如实现棋盘的缩略图等功能。同时这一设计得益于Qt绘图采用的Graphics View框架，可以方便的继承Qt已有的类和对象，减少了代码量和软件逻辑复杂度。

Controller 模块

Controller模块负责将其它4个模块粘合在一起，负责它们之间的连接和通讯，尤其是Scene模块和Model模块之间的连接。
主要包含类CheeseController。CheeseController继承Qt的类QObject，充分利用Qt提供的信号槽（Signal-Slot）机制，实现各个模块之间的连接和通讯。比如，当用户点击棋子“拿起”棋子时，Scene模块感知到鼠标点击这一事件，通过Controller模块转发至Model模块，Model模块计算得到该棋子可走到的位置，通过Controller模块转发至Scene模块，Scene模块将这些位置的边框显示为红色。
单独列出Controller模块，一是有利于对软件整体结构和各个模块的作用的理解；二是Controller模块不依赖于任何模块，在替换某个模块时，只需要改动相应模块和Controller模块即可。

Model 模块

Model模块负责负责棋盘/棋子数据的存储和计算，但不负责如何显示。
主要包含类Cheese和类CheeseModel。Cheese定义了一个棋子，包括其颜色（红/黑）、种类（车马炮……）等信息。CheeseModel继承Qt的类QObject，存储了各个棋子在棋盘上的位置，定义了各种棋子的走棋规则，以及其它与象棋本身相关、与如何显示不相关的信息。
Model模块是整个软件的核心业务逻辑所在。

Network 模块

Network模块负责棋盘/棋子数据的网络传输。
主要包含类CheeseTcpConnection。CheeseTcpConnection继承Qt的类QObject，包括一个QTcpServer类型的成员、一个QTcpSocket类型的成员，负责单机/联机对战时两个程序之间的网络通信传输。
QTcpServer属于Server端，负责服务端的监听、接收连接请求，QTcpSocket属于Server端和Client端，负责发送、接收通讯消息。
该程序既可以作为Server端，也可以作为Client端。若选择新建游戏，则该程序作为Server端，首先监听端口，在有连接请求时进行接收，建立TCP socket；若选择加入游戏，则该程序作为Client端，首先向

Server端发出连接请求，成功后建立TCP socket。TCP socket一方发送消息后，另一方在readyRead信号驱动下接收消息。
Server端和Client端之间的通信消息采用一定格式。第一个字符为标识符，标识该消息的种类，主要有两种消息，一种是整个棋盘的消息，由一个代表棋盘的二维数组表示，另一种是走子消息，由走子的起始和结束坐标表示。它们都转化为字节数组后进行网络传输。

例子

通过几个例子理解各个模块的作用。

- 用户加载残局时，Model模块从文件中读取棋盘信息，比如红车在（3,3）逻辑坐标处等。一方面通过Controller模块转发至Scene模块，Scene模块将象棋棋盘的逻辑坐标转化为显示坐标，比如在（300,300）显示坐标处设置一个红车的图片，通过View模块展示给用户；另一方面通过Network模块，将该棋盘通过网络传输给对方，对方的Model模块从Network模块接收到棋盘信息，再用与上面类似的方法初始化该方的Scene模块和View模块。
- View模块向用户展示棋盘/棋子的图形化界面。当用户点击棋子“拿起”棋子时，Scene模块感知到鼠标点击这一事件，将鼠标点击的显示坐标转化为象棋棋盘的逻辑坐标，通过Controller模块转发至Model模块，Model模块使用象棋棋盘的逻辑坐标，计算得到该棋子可走到的位置，通过Controller模块转发至Scene模块，Scene模块将象棋棋盘的逻辑坐标转化为显示坐标，将这些位置的边框设置为红色，通过View模块展示给用户。
- 当用户通过View模块的展示点击位置“放下”棋子时，Scene模块感知到鼠标点击这一事件，将鼠标点击的显示坐标转化为象棋棋盘的逻辑坐标，通过Controller模块转发至Model模块，Model模块使用象棋棋盘的逻辑坐标，计算得到该位置是否是“拿起”的棋子可走到的位置（是否符合象棋规则）。若符合规则，更新棋盘模型信息进行走子，一方面通过Controller模块转发至Scene模块，Scene模块将象棋棋盘的逻辑坐标转化为显示坐标，更新该棋子的显示位置，通过View模块展示给用户；另一方面通过Network模块，将该走子通过网络传输给对方，对方的Model模块从Network模块接收到走子信息，再用与上面类似的方法更新该方的Scene模块和View模块。

用户界面

菜单栏中点击“Create”选择“New”项，或者按快捷键Ctrl+N，可以新建一个象棋开局。
菜单栏中点击“Create”选择“Pieces”项，可以新建一个象棋残局。弹出对话框，选择残局文件，进行象棋残局的读入。
新建象棋开局/残局后，弹出对话框，选择游戏模式 - “单人模式”、“单机对战模式”和“联机对战模式”。
若选择“单机对战模式”或“联机对战模式”，弹出对话框，显示本机IP地址，要求用户输入端口号，进行监听。在监听过程（30s）内可以取消监听。
菜单栏中点击“Join”选择“Join”项，或者按快捷键Ctrl+J，可以加入一个单机/联机对战象棋棋局。弹出对话框，要求用户输入要加入的象棋棋局的IP地址和端口号，进行连接。在连接过程（5s）内可以取消连接。
若选择“单人模式”，或者选择“单机对战模式”或“联机对战模式”且已经成功连接，开始象棋游戏。轮到用户走棋时，单击己方棋子将棋子“拿起”，棋盘上用红色边框显示该棋子可走到的位置，在合适的位置再单击将棋子“放下”。游戏规则与象棋游戏规则相同，但值得注意的是，“对帅/将”并不会被阻止，发生“对帅/将”时，己方“帅/将”可以吃掉对方“帅/将”。将军时会发出声音提示将军，但值得注意的是，软件并不会提示或阻止用户“送将军”的行为。一方的“帅/将”被对方吃掉，或者一方认输，或者一方倒计时结束时，对方赢得游戏胜利，游戏结束。
界面右侧为象棋棋盘，棋盘靠下的棋子颜色为用户所代表的棋子颜色。象棋棋盘右侧为一个滑动条，可以调整象棋棋盘的显示大小。象棋棋盘左侧提供“认输”按钮，用户按下后即可认输。左侧还显示当前步数剩余时间，倒计时60s，到时后当前轮走的用户自动认输。左侧还显示当前轮到哪个颜色走棋。
游戏进行过程中，菜单栏中点击“Save”选择“Save”项，或者按快捷键Ctrl+S，可以保存游戏残局。弹出对

话框，选择残局保存的位置和文件名。
菜单栏中点击“Exit”选择“Exit”项，或者按快捷键Ctrl+Q，可以退出游戏。若当前正在进行游戏，则会自动认输。

测试用例

利用残局样例文件，对该软件进行了如下测试：

- 连接方面
  - 尝试连接并等待 - 正常
  - 等待连接时可以取消连接 - 正常
  - 输入IP地址和端口号，成功连接 - 正常
- 联机游戏方面
  - 绘制象棋棋盘、棋子 - 正常
  - 正确走子 - 正常
    - 走子基本规则 - 正常
    - 别马腿 - 正常
    - 对帅/将 - 正常
  - 两端棋盘同步 - 正常
  - 将军音效 - 正常
  - 判断游戏胜负，并弹窗告知双方 - 正常
  - 单步走子超时判负 - 正常
  - 认输功能 - 正常
- 残局方面
  - 残局正确加载并能进行对战 - 正常
  - 对战过程中能保存残局 - 正常

未来拓展

由于大作业时间紧迫（只有一周），加上该软件工程量较大，有许多扩展没有来得及实现。

- 界面的进一步美化。希望增加功能：可跟随窗口大小自动调整大小的象棋棋盘；主界面各个显示元素布局的调整和美化；将对话框显示信息改为直接在主界面显示；增加帮助和关于菜单项；等等。
- 记录走棋功能、悔棋功能。目前的软件结构下，较容易添加记录走棋的功能，只需在每一次走棋时进行记录。记录走棋功能实现后，悔棋功能也较为容易实现。
- 代码的重构。有些代码归属的模块有误，比如当前步数剩余时间的数据存储在View模块中，但应该放在Model模块中。有些代码是冗余的，比如Cheese类包含的信息冗余；比如为棋子的显示创建了类CheesePixmap，继承Qt的类QGraphicsPixmapItem，但实际上并不需要这一继承，只需要利用Qt提供的函数修改QGraphicsPixmapItem的属性即可；比如一方判定胜负后，通过网络将胜负的信息传输给另一方，但实际上这没有必要，只需要像正常一样传输走子信息，让另一方自行判断胜负即可。有些代码是难看的，比如象棋规则判定函数、各种数据结构和字节数组之间的转换函数等，充满了switch-case语句，大量代码重复，不利于日后维护和修改。
- 数据结构的优化。目前Model模块存储棋盘/棋子数据、Scene模块存储棋盘/棋子显示坐标，都用二维数组存储了整个逻辑棋盘，在对应位置存放该位置棋子的信息。这样做可能有些浪费内存资源。或许其它数据结构能够在不太损失查询速度的情况下，降低内存占用。
- 网络传输协议的优化。目前初始化棋盘时，传输的是二维数组代表的整个逻辑棋盘，占用了较多的网络资源。或许将棋子的位置信息与棋子本身信息打包传输，而非传输整个棋盘，能够节省网络资源。