Project Report - Week 2
Draughts Lite

陈晟祺
2016010981
计算机科学与技术系 计 63 班

1 项目简介

本项目作为计算机系《程序设计基础》第二周的大作业编写，是一款 C/S 架构的国际跳棋游戏。本项目的核心算法得到了计 62 班周聿浩同学的点拨，在此对他表示由衷的感谢。项目中所用的图片与声音资源分别来自 `https://github.com/AndrewAndreev/chess` 与 `http://www.9game.cn/checkers10x10/1098252.html`，在此一并表示感谢。由于这些资源的使用未经作者许可，本项目仅限用于教学研究用途。

- 作为作业基本要求，实现的功能有；
  1. 网络对战功能
     - 一个可执行文件可兼当客户端与服务器二重角色，自由切换
     - 服务器可设定自身颜色、先手、监听端口，客户端根据指定 IP 与端口连接
  2. 完整的国际跳棋功能
     - 标准 10*10 棋盘，遵循底线升王、有子必吃、最长吃子等规则
     - 每次对方落子后自动计算并提示下一步可行动作，在棋盘上以标记
     - 当对局结束时自动判断输赢，支持中盘认输、求和
  3. 完善的游戏体验
     - 自动旋转棋盘：用户总是靠近颜色属于自己的棋子
     - 落子、吃子音效（可开关）及动画
     - 友好的当前状态提醒和错误提示
     - 基于格式字符串的自定义棋局，支持任意合法局面的生成

2 程序截图

下面是部分程序运行时的截图。

简要说明一下软件各个界面。图2是新建游戏时（作为服务器端）弹出的对话框，可选择自己的执棋颜色、当前局的先手方，指定监听端口。同时，能以特定的字符串格式（途中详细给出）指定开局后的局面；此处也会对字符串进行检查并指出存在的问题。

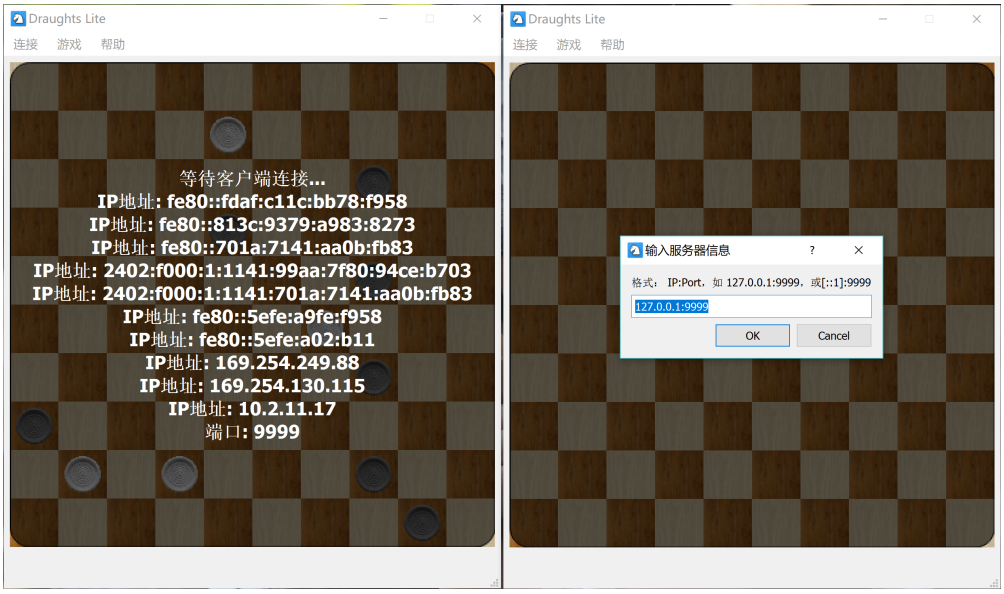

图 1: 服务器-客户端连接

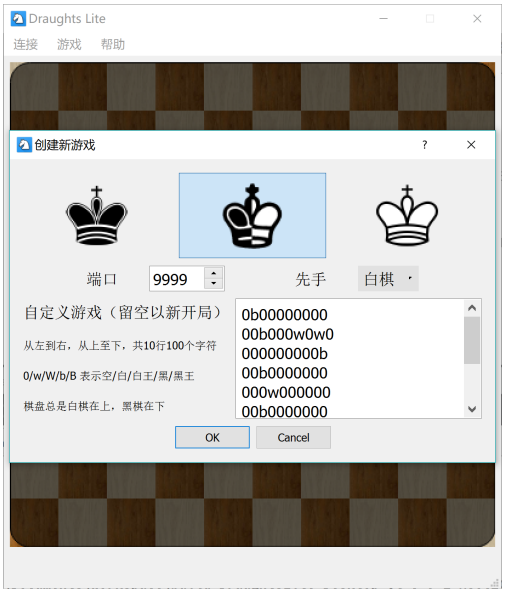

图 2: 新建游戏

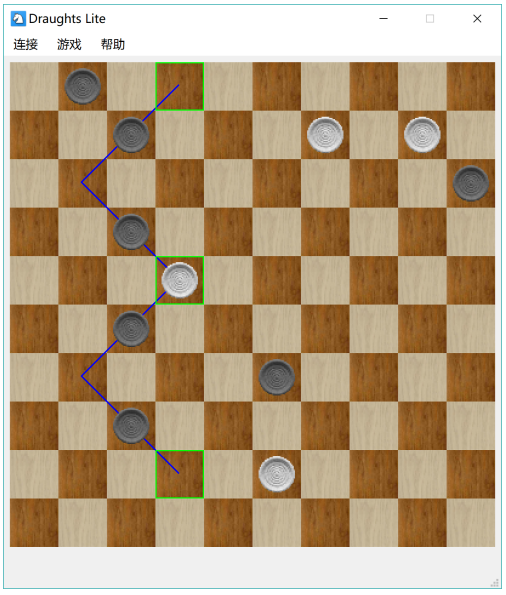

图 3: 游戏进行中界面

确认后即进入图1的状态，此时服务端（左）显示自身所有的可连接 IP（不仅是可路由 IPv4 地址但不包括本地环回地址）与监听端口，另一个程序作为客户端（右）中可进行连接（同样支持 IPv4 与 IPv6 两种格式）。

连接成功后即进入如图3的游戏界面，先手方可立刻行棋，后手需等待对方。在作为行棋方时程序会提示所有可选择的棋子（黄色边框），当选中一个棋子，其边框变为绿色，其所有可能的目的地也被加上绿色边框，同时路径用蓝色线条绘出。选择一个目的地后，棋子沿路径移动到该位置，吃掉途径的棋子，并配以相应的音效。

当某一方移动后对方不再有任何可能的移动，则程序自动判定游戏结束，并告知输赢情况。此外，游戏进行中任何一方都可以主动认输、求和（需征得对方同意）以立刻结束游戏。若游戏中遇到网络断开等情况，游戏将重新变为不可用状态，需要创建新游戏方可重新开始。

3 模块详述

下面简要介绍一下项目各个文件的功能或作用。

main.cpp 程序主入口，初始化并显示 `MainWindow` 类。

mainwindow.{cpp, h, ui} `MainWindow` 类，游戏主窗口。初始化时连接菜单到各个信号和槽，并初始化 `BoardWidget` 和其 `Controller`。

gui/ 程序图形界面的绘制和相关事件处理。

- **createdialog.{cpp,h,ui}** `CreateDialog` 类，创建新游戏对话框。用户可配置各个选项，在确认时将配置传给 `Controller` 并启动服务器监听。
- **boardwidget.{cpp,h}** `BoardWidget` 类，继承于 `QWidget` 类，负责棋盘的绘制与显示、点击判定、可行区域的绘制、棋子移动动画的播放、移动时对 `ChessBoard` 中数据的更新，负责棋子到达对方底线升变为王的处理。
- **waitingwidget.{cpp,h}** `WaitingWidget` 类，继承于 `QWidget` 类，处于等待状态时的棋盘半透明遮罩，显示连接信息与发生的异常信息。
- **soundplayer.{cpp,h}** `SoundPlayer` 类，维护音效与资源中音频文件的实际对应关系，统一管理音效的播放。

logic/ 与跳棋判定相关的核心逻辑及程序 GUI 部分的控制。

- **chesscell.{cpp,h}** `ChessCell` 类，定义棋盘每一格的数据结构，包含颜色、类型以及附着于其上（用于显示图片）的 `QLabel` 的指针。
- **move.{cpp,h}** `Move` 类，定义棋子一步移动的数据结构，其中存储起始坐标、终点坐标、中途吃子坐标（如果有）。
- **chessboard.{cpp,h}** `ChessBoard` 类，跳棋核心逻辑处理。维护棋盘上所有棋子的状态，并在对手每次移动完成后用深度优先搜索的方法计算当前所有棋子的所有可能移动（和跳吃）路径，以找出当前最长的（如果可能，则为可跳吃的）路径提供给 `BoardWidget`。感谢周聿浩同学对我编写此类的巨大帮助。

controller.{cpp,h} `Controller` 类，连接 GUI 与网络组件、主窗口的控制器中间件。将网络部分（`Network` 类）发来的信息解码后向各组件发送指令（开局、移动、认输、求和等），并将各组件反馈信息编码（均采用 JSON 格式）后发往网络部分；也负责将各组件播放音频请求统一处理并转发给 `SoundPlayer` 类。

utilities/ 程序的配置和数据结构定义。

gameconfig.{cpp,h} `GameConfig` 结构体，存储新建游戏的配置，包含先手颜色、当前颜色、开局棋盘信息（用户给定或默认生成）。

enumerates.h 定义程序中用到的各个枚举量，如棋子类型、颜色、操作类型。

network/network.{cpp,h} `Network` 类，处理网络通信相关事宜。负责 TCP 套接字的创建、连接与管理，与 `Controller` 之间以 `QByteArray` 的方式传递信息，并恰当处理通信中的异常情况。

common.h 附加于所有编译单元上的预编译头。主要用于给 MSVC 编译器提供指令，以改变可执行文件的运行编码为 UTF-8，避免乱码发生。

resources/* 程序中所用图片、声音资源，用于程序图标、棋盘背景、棋子、走棋音效等。

4 设计理念

在本项目的开发中，我秉持以下的理念：

- 尽量遵循 Google C++ Code Style，命名规范，风格完善
- 多使用 Qt 提供的基础设施，实现完美的跨平台运行
- 拒绝硬编码，将常量、字符串、资源等集中管理
- 坚持 OOP 思想，明确区分各功能模块职责，降低各部分的耦合

5 项目感想

作为第二个项目，比起上周来我熟练度提高了许多，也有了更多的经验，可以写出更优雅的代码。但这周作业引入的网络和多线程也带来了更大的不确定性和复杂度，对我的耐心和细心程度的考验也大大提高了。在过程中我遇到了不少问题（如经典的 TCP “粘包”、字节序不一致、状态不同步等），都通过多方查找成功找到了完善的解决方案。同时，这个游戏还涉及到略有难度的算法实现，也对我构成了不小的挑战。不过最终我还是顺利完成了这个项目，并补上了上周的部分遗憾，在 GUI 和网络的通信中使用了 MVC 架构，也感受到了经典优秀架构对编码带来的便利。和上周一样，我的收获依旧很大。希望能在接下来的最后一周中再接再厉。最后，仍然十分感谢老师以及助教的帮助。