Connect5 - An Ancient Chinese Board Game with network battle implementation, based on Qt

苏克
2014011402, 计45, nagizero@foxmail.com

2015 年 9 月 6 日

目录

I Introduction

本次的大作业要求实现网络五子棋程序，作为网络和多线程编程的练习作业。笔者本想学习QML从而实现更美观的界面，却因事务繁忙失败。之后一定会有相应的更新。

下面具体地说明程序的编写思路和模块化构造。

II System Overview

简单地说，游戏除了游戏主窗口类Game和用以打开不同模式的游戏窗口的类Launcher，大多数都是游戏逻辑和交互的模块化。下面按照具体的情况进行说明：

III Basic Data Structure

3.0.1 Map.h

枚举类Cell给出一个格子上可能的所有状态：白子、黑子、空。枚举类Player为相应的玩家：白，黑，无。两个枚举类都继承于int，所以可以通过强制类型转换进行棋子到玩家之间的正确转换。

Pos类存储一个二维向量（相当于一个坐标），给出x()和y()两个接口。

CellMatrix类作为“棋盘”的数据结构存在。重要的外部接口函数如下：

- bool move(const Player& p, const Pos& p)

  用于一次下子。返回下子成功或失败的结果。

  实际程序中，Game类的move接口通过调用这个函数来进行双方落子的操作。

- bool checkWin()

当前棋局是否已经分出胜负？如果有人胜利，返回true，否则返回false。

五子棋的规则决定了胜利方胜利时，胜利方一定唯一，而且最后的落子一定直接导致胜利——亦即最后的落子一定属于某个“五长连块”。因此只要记录并更新棋盘上最后一处落子点的位置，在它的一周八个方向进行探索计数，得到四个方向（水平、竖直、主副对角线方向）上同色联通长列的长度，判断其是否为5以上即可。

当然，如果想要节省空间，也可以做这样的设计：每次move()内成功落子之后，判断是否胜利。这样便不需要记录最后落子点。

但是本程序中因为有“悔棋”的需求，记录落子顺序（或最后落子点压栈）成为了必须。便没有进行上述设计。

重要的外部槽函数如下：

- void undo()

  悔棋。当落子点栈元素超过两个时，从中取出两个并消除其对应位置的棋子。

- void save()

  存档。根据当前的日期和时间自动生成文件名，将存档文件保存在相同目录内。

- bool load(QWidget* parent = 0)

  读档。弹出一个文件对话框，加载用户选择的存档文件。返回是否读档成功的信息。

CellMatrix类的信号如下：

- void endGame(const Player& p)

  当判定游戏结束时，发射该信号并带有胜利方信息。

  Game类会监听这个信号，并作出正确的反应。

3.0.2 NetBattleMsg.h

NetBattleMsg类是所有网络互传信息的封装类。简单地说，所有的信息都会被封装成一个该类的实例，并藉由该类提供的接口转为对应的Qstring类实例后，通过TCP协议发送。

同样地，该类也提供了将收到的QString实例解码为NetBattleMsg类实例的接口，以读取并处理信息。

Game类中，将接收信号与getMsg相连接。在其中，会对可能发生的粘包现象进行自拆包处理，之后再处理分解开来的一条条信息。

因为上述接口用法比较简单，这里只介绍此类的功能，不具体说明接口。

3.1 Game Logic & User Interface

3.1.1 Game.h

Game类继承于QDialog，是游戏的主要窗口。其左边是一个棋盘，右边是根据游戏模式不同而不同的功能按键。

由于游戏需求繁多，Game类拥有大量的功能函数。不过比较大的功能块全部解耦成了单独的模块，下面进行介绍：

- 棋盘绘制

  棋盘的绘制全权交给MapPaintEngine类负责，后者只要一个map类常量指针就可以把棋盘状态完整地绘制在窗口上。

- 用户交互

  游戏本身的用户交互仅限于鼠标在棋盘上的点击。本程序中，用户交互由基类UserInteraction的指针处理，后面会有详细的说明。

此外，Game类本身也负责进行网络信息的处理：包括QTcpServer和QTcpSocket的搭建和连接、网络信息的接收和处理、用户可以发出的网络相关的指令等。

下面介绍上述的两个重要的模块。

3.1.2 MapPaintEngine.h

MapPaintEngine类负责绘制棋盘。其只提供一个paint接口，传入舞台窗口指针、CellMattix类的指针、和一个QRect参数代表绘制区域，就可以进行绘制。

依然使用QPainter进行简单的绘制。为了方便玩家查看游戏情况，最后一个落子点会比其他妻子亮一些，以示特别。

3.1.3 UserInteraction.h

UserInteraction类是所有用户交互类的基类。其带有当前玩家的信息，可以发射如下的信号：

- void move(const Player& player, int x, int y, bool flag)

  move信号代表用户指定了一个操作，携带有该操作的基本信息：颜色、位置。flag变量用于用户拓展。

  在使用时，只要将move信号连接给Game类的move函数就可以进行正常的落子操作。

因为本程序中所有的用户交互都基于鼠标，因此提供了MouseObserver类，用以监听某个区域的鼠标事件并发射相应的信号。使用时，UserInteraction类（或其子类）自己按需要创建MouseObserver类对象，并将后者发射的点击信号进行妥当的处理，最后发射move信号，即满足使用要求。

随程序携带了两个UserInteraction类的子类：

- LocalMultiUserInteraction

  此子类用于本地对战的用户交互。要点在于每次落子之后自动更换当前玩家。达到分别落子的效果。

- NetBattleUserInteraction

  此子类用于网络对战。与上面唯一的区别在于不会自动更换落子玩家，而一定要根据存储的用户信息进行落子。

3.1.4 Countdown.h

Countdown类主要完成倒计时工作。其内部含有一个QTimer类对象，用来监听每一秒的时间变化；而它自己则根据自己的状态（on或者off）判断是否对“一秒”的信号进行处理。

Countdown类发射三个信号供使用者接受并处理：

- void timeChanged(int time)

  表示当前剩余时间变化了，并附带上变化之后的值。可以直接把这个信号连接到显示元件上，来实时更新当前的剩余秒数。

- void timeOut()

  当时间归零时，会发送这个信号（不影响上面timeChange信号的发送）。同时，当前剩余秒数自动刷新至最大值。

- void oneSecSignal()

  仅当计时器处于开状态时，每过一秒，在更新数据之前会发射这个信号。使用者可以接受这个来计算总时间。

Countdown类主要的接口函数如下：

- void start()

  开启计时器。注意，在开启之前会将剩余秒数更新到最大值。

- void pause()

  暂停计时器。

- void stop()

  停止计时器。所谓停止和暂停的区别是：停止之前会将剩余秒数更新到最大值。

- void resume()

  重启计时器：只单纯地将计时器状态修改为开启，不会影响之前的剩余秒数。

IV Human Interface Design

打开应用程序会进入一个“Launcher”界面，用户可以选择打开一个本地对战窗口或者网络对战窗口。

对战窗口中，左侧是棋盘，右侧会有如下的按钮或文本框：

建立主机连接主机连接状态：分别代表建立一个主机，按ip连接到一个主机，和当前的连接状态。

Battle!按键：连接成功后玩家需要各自点击Battle!按键才可以开始下棋。

后面会显示当前落子方的颜色。

下面一排按键是：保存游戏，读取游戏。分别代表保存当前游戏状态，或者从已有的游戏状态中读取一个。

下面是悔棋和退出。点击时，会向对手发送相应的请求，若对手允许了请求则会进行相应的处理。