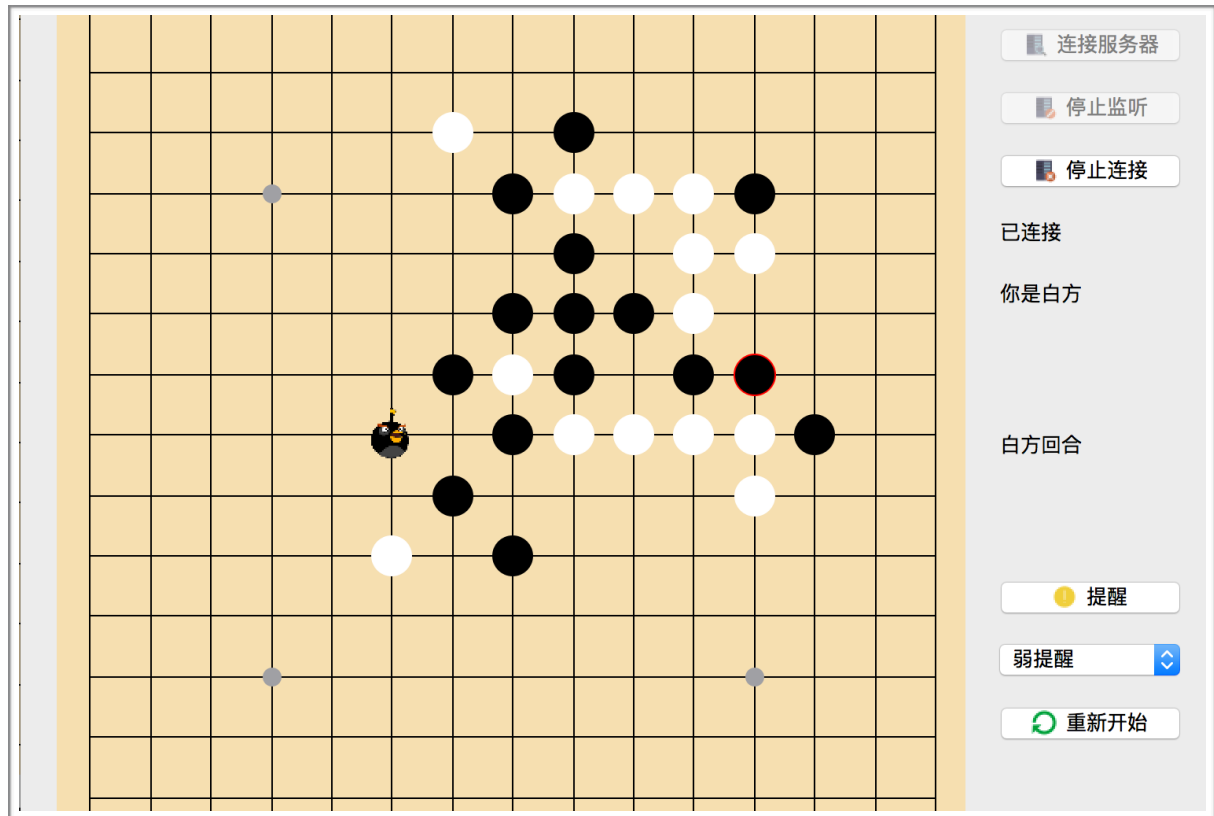


SmartGobang

Software Design Document



张钰晖

计55，计算机系，清华大学

September 1st

2016 夏季学期

1 INTRODUCTION

1.1 Purpose

本篇文档是为游戏SmartGobang编写的设计文档，SmartGobang是清华大学2015-2016学年夏季学期程序设计训练课程的大作业，本文档将具体讨论该游戏的设计思路与代码实现，并给出该游戏的使用方法，通过阅读此文档，读者将具体了解该游戏的实现算法和使用方法，并且了解部分Qt软件设计思路。

1.2 Scope

SmartGobang是一个小巧的网络联机五子棋对战游戏，游戏的目标是通过网络TCP通信协议实现网络五子棋对战功能，实时同步数据，判断胜负，同时，提供危险提醒功能，针对一些危险局势判断并提醒用户，下面简述了SmartGobang软件设计的原始要求：

1. 能够创建网络主机。在界面上添加功能按钮，显示创建主机对话框，对话框显示主机的**IP**，还有取消按钮（在连接过程中可以随时结束）
2. 游戏客户端能够加入主机。添加软键盘，用来输入主机**IP**
3. 棋子坐标的网络传输。传递报文，表示最新落子的坐标，另一端接收报文，并且刷新棋盘
4. 界面标记游戏状态。显示当前落子方，如果哪方赢了，弹出对话框提示
5. 在用户点击提醒按钮时候，可以判断出对方如果在某个位置下一子，你可能出现如下情况的位置：再落一子，出现两个无阻挡的连续**3**子的情况；再落一子，出现**1**个无阻挡的连续**3**子和**1**个有单侧阻挡的连续**4**子的情况。这种情况，要以炸弹图标给出危险位置

1.3 Overview

本文档将首先讨论该游戏的系统设计，具体介绍数据部分的管理和处理方法。接着，本文档将讨论游戏的各种实用接口以及游戏的使用方法。最后，会对该应用进行简单的总结和鸣谢。

2 SYSTEM OVERVIEW

SmartGobang游戏是基于Qt 5.7 (Clang++) 进行开发的游戏，开发平台基于OS X 10.11.6。SmartGobang游戏是一款图形用户程序，提供了原始要求中的所有的功能，并且较为良好的完成了用户交互部分，还提供了更多的功能。比如用户可以选择提醒的强弱，来增大或减小游戏难度。

3 SYSTEM ARCHITECTURE

3.1 Architecture Design

应用主要分为设计部分、算法部分和数据部分。设计部分主要指GUI设计，本应用同时采用源码设计和Qt Designer方式。其中MainWindow类和一些继承重构后的类采用源码设计方式构建，而用户交互的类则由Qt Designer进行实现。算法部分主要是指大量的基于信号-槽机制的开发，和判胜与提示功能，同样采用了上述两种方式。数据部分主要是指棋盘棋子的存储以及通信信息的传输，后文会具体讨论这一部分的实现。

3.2 Decomposition Description

此部分将给出各个类的作用，对几个关键的类进行简单的介绍，分为如下几个部分——Widget、Data和MainWindow。如果需要具体查看每个类的具体使用方法和代码实现，可以查看后文介绍接口的部分。

3.2.1 Widget

Widget部分由如下类组成。

- ChessBoard
- CreateDialog
- ConnectDialog

ChessBoard

继承了Qt提供的QWidget，内含一个二维的ChessLabel*向量（下文数据类会介绍这个类）。该类负责管理整个棋盘，提供了创建棋盘、重置棋盘、判胜、警告、设置与删除炸弹、发送数据、根据接受的内容更新棋盘信息等功能。

CreateDialog

继承了Qt提供的QDialog，在主界面点击创建服务器后会弹出该对话框，默认显示本机IPV4地址，点击确认后会创建连接。该类是Qt Designer Form，可以再源码中直观的看到ui文件。

ConnectDialog

继承了Qt提供的QDialog，在主界面点击连接服务器后会弹出该对话框，会允许用户通过键盘或使用该类创建出的小键盘输入服务器的IP地址，点击确认后会进行连接。该类是Qt Designer Form，可以再源码中直观的看到ui文件。

3.2.2 Data

Data部分由如下类组成。

· ChessLabel

ChessLabel

继承了Qt提供的QLabel，每一个对象负责显示一个格子，内含了游戏所需要的数据，包括指针如本机棋子颜色、是否已经连接、当前下棋方颜色，以及变量如该点对应颜色、该点是否为最后一个棋子、以及一些变量用于正确初始化格子和美化UI等。该类重载了paintEvent(QPaintEvent *event)和mousePressEvent(QMouseEvent* event)，用于处理鼠标点击事件并画图，ChessBoard类就是ChessLabel对象的集合体，正如宏观与微观的关系。

3.2.3 MainWindow

MainWindow部分由如下类组成。

- MainWindow
- main

MainWindow

继承了Qt提供的QMainWindow，是应用程序的主体窗口，同时是应用的中心。为了使布局优美便于管理，本类采用纯代码的方式实现，包含了棋盘ChessBoard以及右侧各种功能按钮QPushButton、QComboBox以及连接状态、所执棋子、游戏状态的QLabel。同时该类负责数据的交互以及数据的传输，该类内含QTcpServer与QTcpSocket两个private变量，负责根据主机与客机来借助网络传输数据。同时重载了closeEvent(QCloseEvent* event),禁止在连接状态下关闭窗口，导致另一端崩溃。

main

应用程序的入口。

4 DESIGN

4.1 Data Description

本章将具体讨论此应用程序的数据处理工作。

首先是棋盘的正确执行。MainWindow类有三个本地变量，为m_who, m_connected, m_current,分别对应着自己方棋子的颜色、网络是否连接正常、以及现在该下方棋子的颜色，这三个变量通过指针传给ChessBoard，ChessBoard内含一个二维向量，存放着ChessLabel，在构造时，这三个指针再传入ChessLabel，ChessLabel不仅保存着该格子对应的棋盘信息，也保存着这三个指针，每次用户点击某个格子触发MouseEvent时，ChessLabel会根据是否已连接、是否该本方下棋以及本方的颜色正确的修改数据结构，相应用户事件，同时重绘该格子。

其次是数据的传输，考虑到数据的简单性以及规模较小，数据的传输采取极其精简的数据压缩方式，每一次下棋，都会发送数据，同时对方下棋都会接收数据。每一次接收与发送，都对应着一个两个Byte的char*，其中char[0]表示row, char[1]表示column, 其中若char[0] = DISCONNECT_CODE(一个宏，特定的数字),代表断开连接，char[0]=RESET_CODE（同上）代表重新开始，但由于棋盘行列从（0,0)开始编号，

所以若char[0] = 0会导致发送异常，所以增加了偏移量DET（宏），解决这个问题。数据的发送与接收都在MainWindow进行，同时将数据传给ChessBoard更新棋盘状态。

最后是一些增加用户体验的数据，如每个ChessLabel中增加了变量负责线条的绘制，特殊点（如中心点等的定位）、以及最后位置坐标提示用户上一步棋子落点，感兴趣的读者可以自行参考源代码。

4.2 Net Description

本章将具体讨论此应用程序的客户端、服务器端的工作流程，二者通信协议，以及网络通信编程框架。

首先整个游戏的通信协议为TCP通信协议，TCP中建立连接需要形成五元组，即连接协议、主机地址、主机端口、客机地址、客机端口。为此，需要在工程.pro文件增加QT += network，同时include <QTcpServer>和<QTcpSocket>两个头文件。

MainWindow内含了两个private数据，QTcpServer* m_listenSocket和QTcpSocket* m_readWriteSocket，两个指针均会在构造时被初始化，从而每个客户端既可以成为主机又可以成为客机。正如其名字所暗示的那样，如果用户选择本机作为服务器，那么将会执行m_listenSocket->listen(QHostAddress::Any, 8888),即开始监听所有地址来自8888端口的连接，当其收到newConnection()信号，表明有人连接，将会跳转到槽acceptConnction(),此时将m_readWriteSocket赋值为m_listenSocket->nextPendingConnection(),建立五元组，形成连接，同时改变数据，开始游戏。若用户点击停止监听，会执行m_listenSocket->close()，中止等待连接。

如果用户选择本机作为客机，那么点击连接按钮并输入服务器IP后，会执行m_readWriteSocket->connectToHost(address, 8888),即尝试连接地址为address端口号为8888的服务器，然后若其state变为ConnectingState，表明正在连接，这是会给出2秒的连接时间waitForConnected(2000)，若状态变为ConnectedState，表明连接成功，开始游戏，否则提示服务器无响应，若state仍为UnconnectedState表明连接失败，需要检查IP。

不论作为主机或客机，两者连接后，m_readWriteSocket都处于ConnectedState状态，这是建立connect(m_readWriteSocket, SIGNAL(readyRead()), this, SLOT(receiveMessage())),每次收到消息时，都会立即读取消息，并处理后继续向ChessBoard传下去，修改棋盘状

态，而每次更改棋盘状态时，通过之前的连接都会触发槽sendMessage(),将处理好的信息通过m_readWriteSocket传出去。实现棋盘实时同步。

连接状态中，若点击停止连接按钮，会先通过发送特定的断开连接信号到主机/客机，然后发送方执行m_readWriteSocket->disconnectFromHost(),接收方收到断开连接信后后执行同样指令，断开连接，并重置一些信息。（重新开始按钮与之类似，发送特定的指令实现）

这也是网络编程的基本框架，服务器端通过QTcpServer开始监听，连接到来后建立QTcpSocket进行数据传输，客机端通过QTcpSocket直接尝试连接主机，连接后进行数据传输。

4.3 Algorithm Description

本章将具体讨论此应用程序的判胜以及危险提醒功能的算法。

判胜功能由ChessBoard提供，接口为bool isEnd(int current)。游戏的胜利只可能是在最后落下的棋子处触发，因此需要记录最后落下棋子的位置，并且传入落下棋子的颜色，通过检查最后落下的棋子水平方向、竖直方向、以及两个斜方向上当前颜色的棋子是否大于等于了5个，即可宣告游戏是否结束。胜利方即为落下最后棋子的一方。该算法的时间复杂度为 $O(1)$ ，因为只需要常数次执行，常数最大也不到100，因此该算法时间复杂度很低。在每落下新的一子时均会执行该函数，这就保证了不会造成引用的阻塞。

危险提醒功能是本游戏的亮点，该功能由ChessBoard提供，接口为QList<QPoint> isRisky()。并且用户可以选择强提醒或弱提醒，强提醒下会显示所有有威胁的位置，会很早的提醒用户作出判断。弱提醒下只会显示每下一子形成两个活三，或一个活三一个堵四，或两个活四的情形，只会提醒用户必输的局面，及时挽回局势。算法的原理是遍历棋盘每个位置，假设该出为敌方棋子，然后检查其四个方向是否会形成活三或堵四，然后更新计数器，若该点计数器变化大于等于2，则代表敌方下在该点会出现两个活三、两个堵四、或者一个活三与一个活四，此时敌方下一步下在该点用户必输，记录该点位置并设置炸弹，为弱提醒模式；若该点计数器变化大于等于1，代表这一点只是有危险，还不至于输，此时为强提醒模式，提前让用户预警。此算法的复杂度为 $O(n^2)$ ，n为棋盘大小，n=15，算法效率仍然足够高，保证点击提醒按钮时不会导致阻塞。

5 HUMAN INTERFACE DESIGN

5.1 Overview of User Interface

由于应用是一个桌面游戏，用户接口都是以GUI实现。但是，源码中包含有一些可以被后续开发者所使用的类，记作Library类，这里会具体介绍一些笔者认为有这样价值的类和及其接口，同时会介绍GUI的使用方法。

5.1.1 Library

对于源文件中各个类的接口，我们将以源码的形式来展示这些接口，由于篇幅有限，这里只详细介绍几个笔者认为有介绍价值的类。

CreateDialog

```
class CreateDialog : public QDialog
{
    Q_OBJECT

public:
    explicit CreateDialog(QWidget *parent = 0);
    ~CreateDialog();
    QString getHostIP();
    QString getInputIP();

private:
    Ui::CreateDialog *ui;
};
```

CreateDialog负责用户创建服务器，在这里有一个可以复用的接口，可以获得主机IPV4地址，该接口为QString getHostIP(),实现如下。

```
QString CreateDialog::getHostIP() {
    QString localHostName = QHostInfo::localHostName();
    QHostInfo info = QHostInfo::fromName(localHostName);
    foreach(QHostAddress address, info.addresses()) {
        if(address.protocol() == QAbstractSocket::IPv4Protocol) {
            return address.toString();
        }
    }
}
```


首先判断获取主机名，然后通过主机名得到QHostInfo，然后遍历其中的address，若其为IPV4地址，则取出并返回。

ConnectDialog

```
class ConnectDialog : public QDialog
{
    Q_OBJECT

public:
    explicit ConnectDialog(QWidget *parent = 0);
    ~ConnectDialog();
    void createKeyboard();
    QString getInputIP();

private:
    Ui::ConnectDialog *ui;
};
```

ConnectDialog负责用户连接服务器，其中值得借鉴的是其中的创建数字小键盘并正确连接槽与信号的函数createKeyBoard()。

```
void ConnectDialog::createKeyboard() {
    for (int i = 1; i <= 12; i++) {
        QPushButton* button = new QPushButton;
        .....
        connect(button, &QPushButton::clicked, [=] () {
            switch (i) {
                case 10:
                    ui->ipEdit->setText("");
                    break;

                case 11:
                    ui->ipEdit->setText(ui->ipEdit->text() + "0");
                    break;

                case 12:
                    ui->ipEdit->setText(ui->ipEdit->text() + ".");
                    break;

                default:
                    ui->ipEdit->setText(ui->ipEdit->text() +
QString::number(i));
                    break;
            }
        });
    }
}
```

在连接时，用到了lambda表达式，使得连接代码显得极为干净简洁，省去了复杂的SignalMapper操作。其中lambda表达式可以理解为匿名函数，熟练掌握lambda表达式有助于程序写的简介而紧凑、便于修改、易于理解。

MainWindow

MainWindow是一个ChessBoard、PushButton、Label等部件的集合体。MainWindow类为纯代码类，没有使用设计者模式。代码实现UI的流程基本为：

1. new出所需要的组件
2. 对new出的组件设置QHBoxLayout、QVBoxLayout、QGridLayout
3. 再对第二步的Layout设置整体Layout
4. 创建QGroupBox，应用第三步整体Layout

通过这四步不断整合，控制UI界面，提高美观性。

由于篇幅原因，其余类不再细谈，感兴趣的读者可以阅读相关代码。

5.1.2 GUI

游戏的GUI有设计较为简洁，有许多功能，下面将一一介绍其使用方法。

棋盘界面

左侧为棋盘界面，当正确完成链接并轮到自己回合时，点击相应格子会下对应棋子，并传输/同步数据。红色圆圈标记的棋子为最后落下的棋子。

创建服务器

点击后会弹出对话框显示主机IP，按确认创建。创建后不可以连接、断开连接（对应按钮不可用），并可以取消监听。

连接服务器

点击后会弹出对话框显示数字小键盘并允许用户输入主机IP，按确认连接

停止监听

若为创建服务器模式，点击后会取消等待用户连接。

停止连接

若为连接状态，点击后双方会断开连接。

提醒

分为强提醒和弱提醒，强提醒会提醒任何有危险的位置，弱提醒只会提醒必输的位置，点击后会在对应位置设置炸弹图标。下棋后炸弹图标消失。

重新开始

点击后游戏会重新开始，棋盘重置。

退出

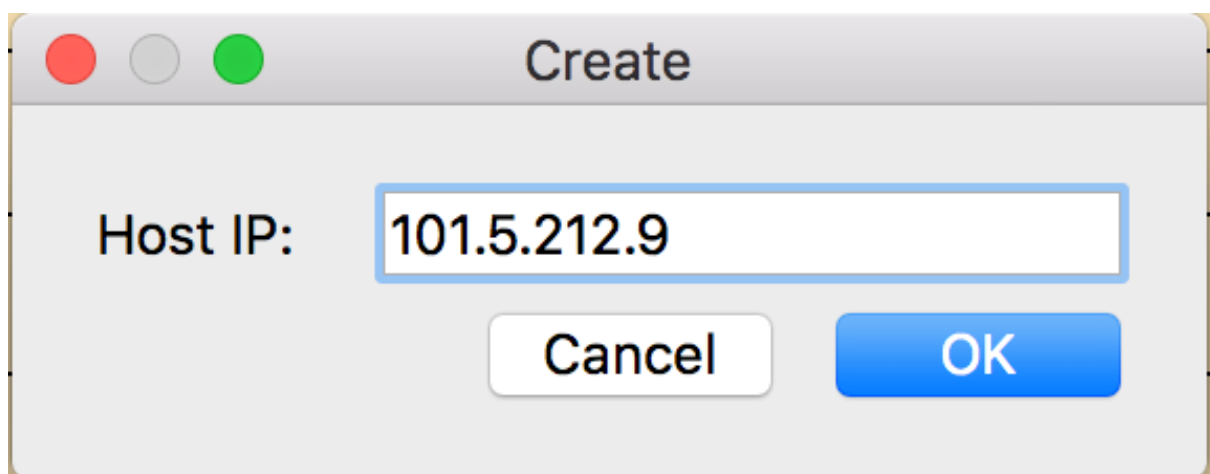
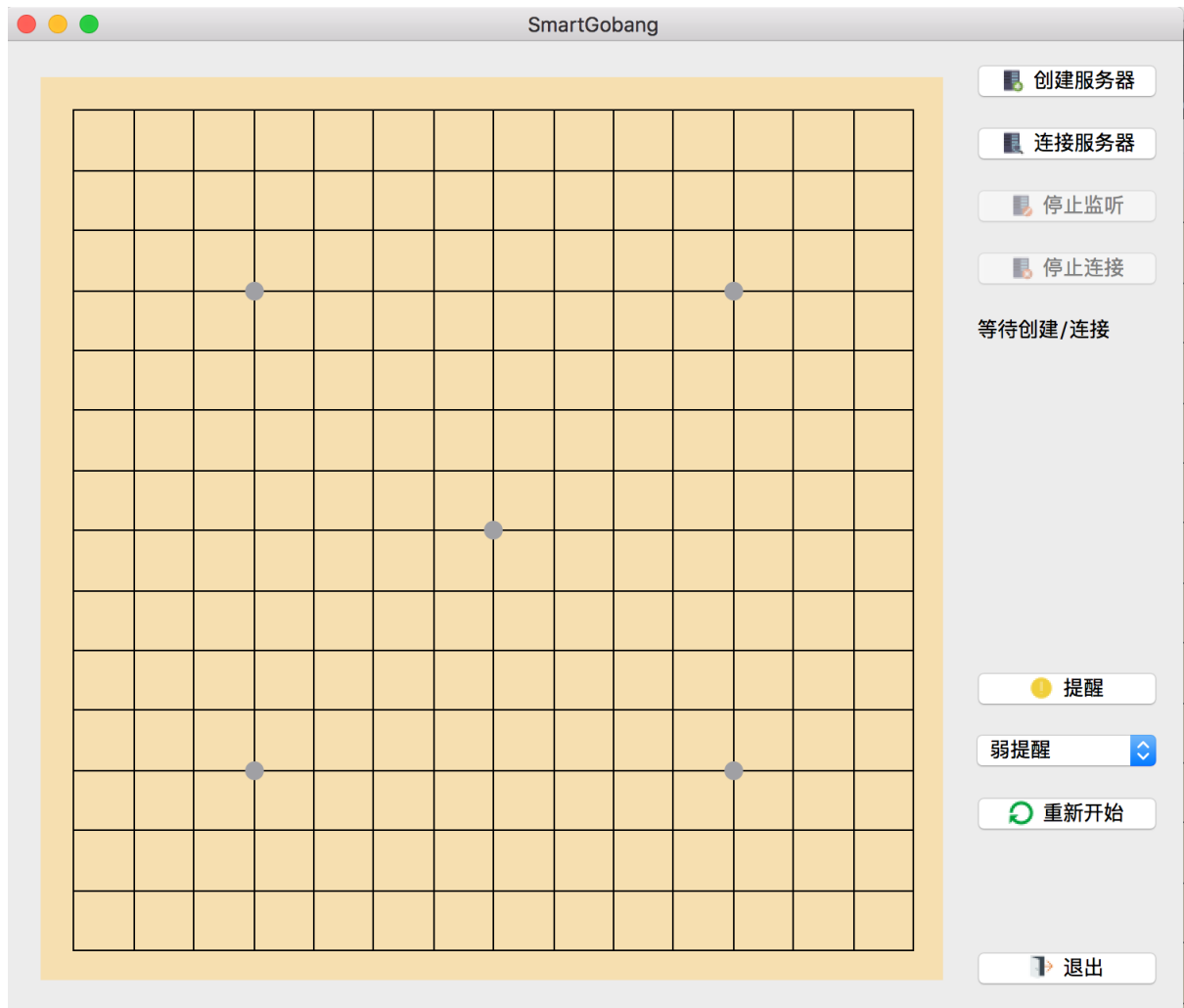
若为非连接状态，点击后退出游戏。

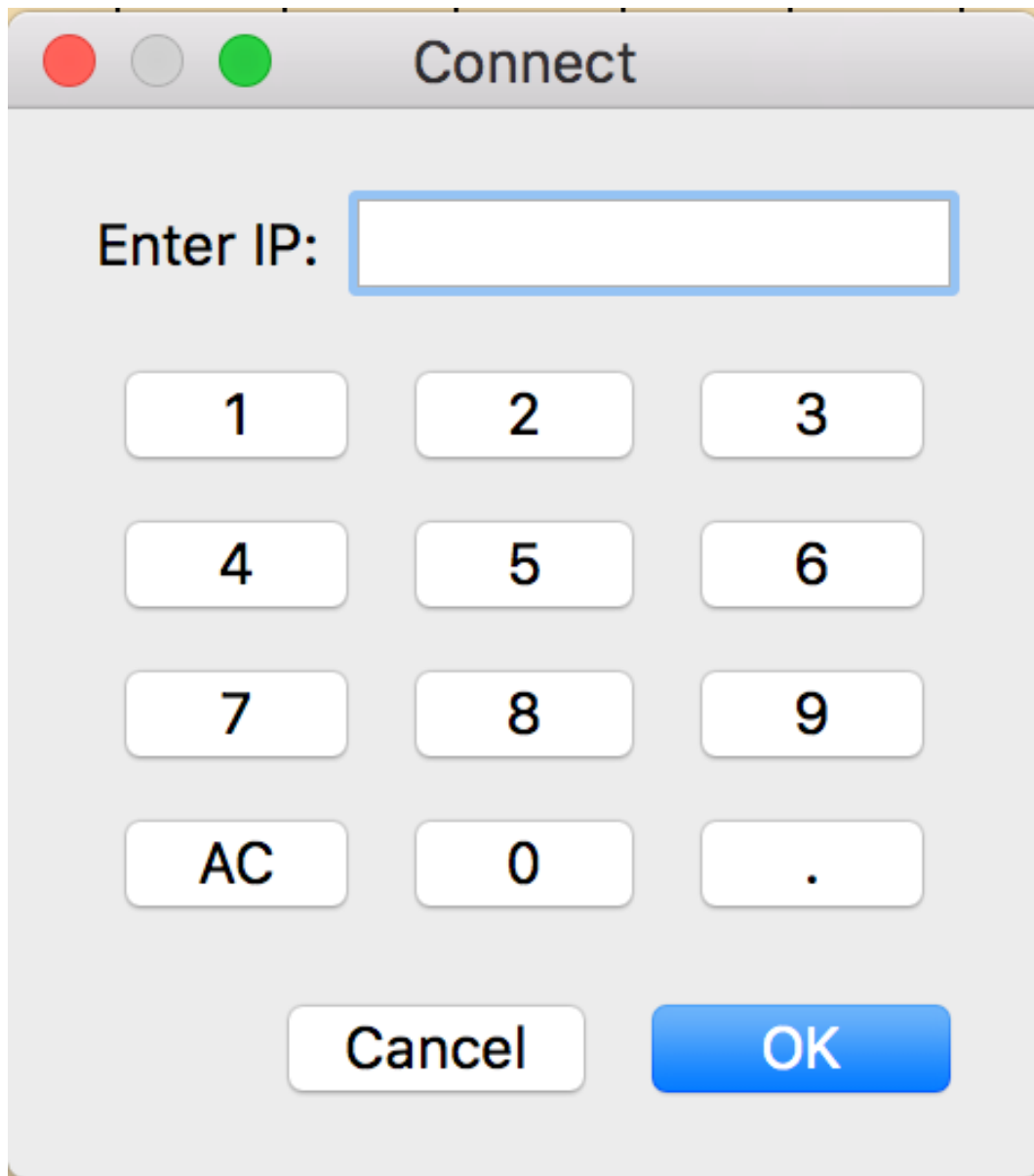
提示框

右侧自上而下提示框依次为连接状态、所执棋子、当前回合。

5.2 Screen Images

下面是软件使用过程中的截图。





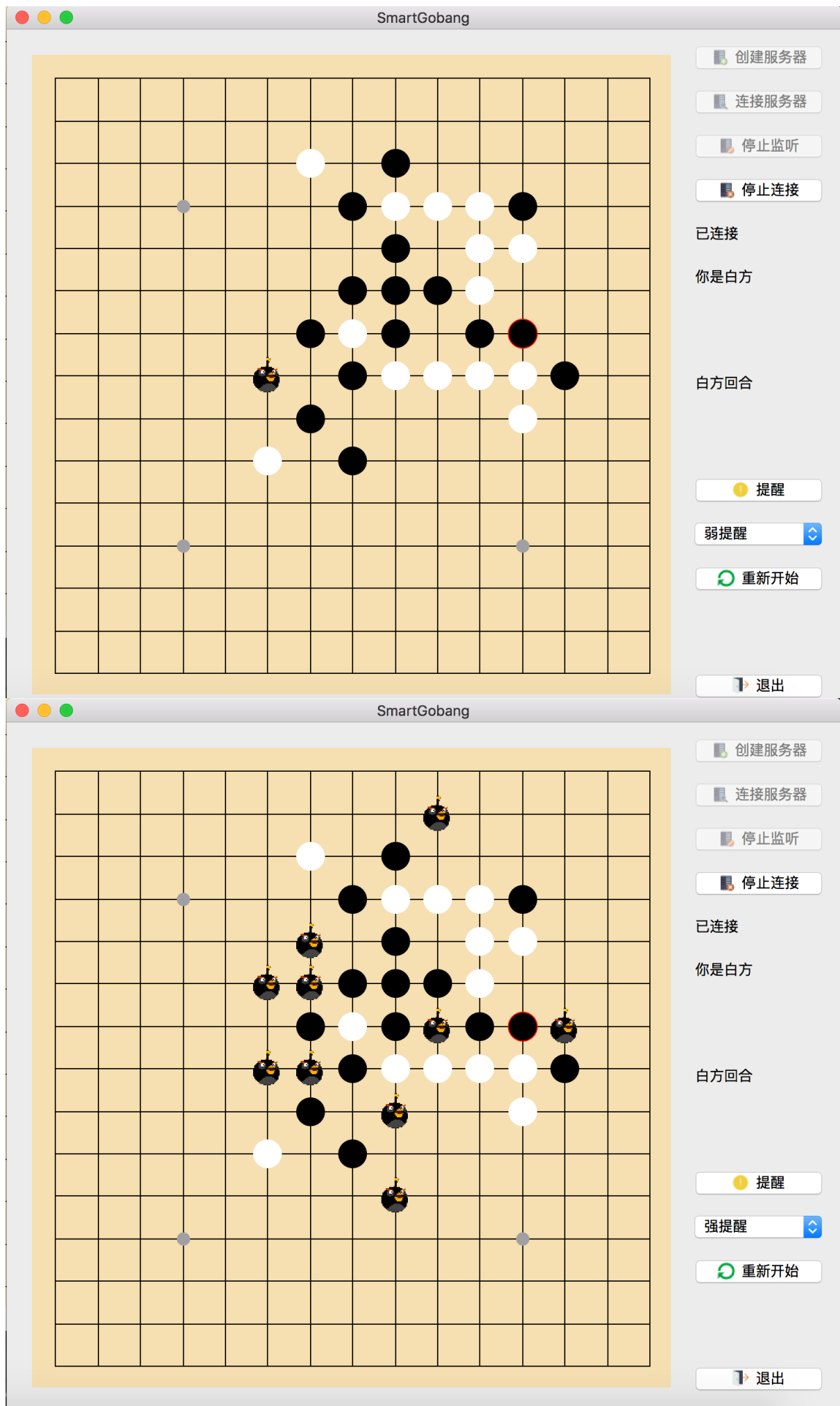
A macOS-style dialog box titled "Connect". It features a title bar with red, yellow, and green window control buttons. The main area contains the text "Enter IP:" followed by a text input field. Below the input field is a numeric keypad with buttons for digits 1-9, 0, a decimal point, and an "AC" (All Clear) button. At the bottom are "Cancel" and "OK" buttons.

Connect

Enter IP:

1	2	3
4	5	6
7	8	9
AC	0	.

Cancel OK



6 SUMMARY

以上就是SmartGobang全部的设计文档，SmartGobang 是一个精致小巧的桌面游戏。希望能给您的生活带来一些快乐。

但由于精力有限，代码实现中可能会有一些不尽如人意的地方，可能会出现一些错误，还请各位使用者多多包涵。同时请包涵设计文档中可能出现的笔误。

在本文的最后，感谢这些天来帮助我的同学，感谢热心网友的帮助文档。