南 开 大 学

本科生毕业论文（设计）

中文题目：基于 Java 的五子棋游戏设计

外文题目：Design of Gobang Game Based on Java

学　　号：2013599

姓　　名：田佳业

年　　级：2020 级

专　　业：计算机科学与技术

系　　别：计算机

学　　院：计算机学院

指导教师：刘嘉欣

完成日期：2021.12.23

摘 要

五子棋老少皆宜，即使是在智能设备已经完全普及的今天，依旧是日常生活中经典的休闲游戏。本文阐述了 Java 版五子棋程序的编写思路和重要的 Java 程序设计思想，并在基本功能的基础上介绍了五子棋 AI 相关的算法，同时对 AI 设计所必要的五子棋规则和棋型进行了简要介绍最后说明了程序的亮点以及不足。

首先，本文介绍了问题提出的背景。在学习了 Java 程序设计并完成对应作业后我们通过课程设计的方式进行总结复习。本程序完成了五子棋基本对战功能，包括下棋、输赢判断；必要的辅助功能，包括悔棋、重开、复盘等；以及网络对战和聊天功能。额外的，本程序提供了 3 种方式可供选择，分别为自娱自乐（Self Play）、人机对战（Play with AI）和网络对战（Play online）。

而后，本文对窗口之间的交互，类的设计以及重要方法的实现作了简要介绍。其中结合程序设计的基本情况对关键之处进行了解释，并对主要采用的设计模式进行了介绍。

最后介绍了五子棋 AI 的设计的基本思路和算法，简要说明了五子棋基本局势以及判断的方法。并总结在程序设计中的用心之处以及遇到的困难和问题，对程序进行了简要的评价。

关键词：五子棋；Java；MVC 设计模式；AI 设计

Abstract

Gobang is suitable for all ages. Even today when smart devices are fully popular, it is still a classic casual game in daily life. This article expounds the programming ideas of the Java version of Gobang program and important Java programming ideas, and introduces the algorithms related to Gobang AI based on the basic functions; at the same time, it briefly introduces the Gobang rules and chess patterns necessary for AI design; Finally, the highlights and shortcomings of the program are explained.

First, this article introduces the background of the question. After learning Java programming and completing the corresponding homework, we summarized and reviewed through the way of course design. This program has completed the basic functions of Gobang, including playing chess, winning or losing judgment; necessary auxiliary functions, including regret, reopening, and replaying; as well as online battle and chat functions. In addition, this program provides a total of 3 ways to choose from, namely Self Play, Play with AI, and Play online.

Then, this article briefly introduces the interaction between windows, the design of classes and the realization of important methods. Among them, the key points are explained based on the basic situation of program design, and the main design patterns adopted are introduced.

Finally, the basic ideas and algorithms of the design of Gobang AI are introduced, and the basic situation of Gobang and the method of judgment are briefly explained. And summarized the intentions in the program design and the difficulties and problems encountered, and made a brief evaluation of the program.

Key words：Gobang; Java;MVC design patten; AI design

目 录

第一章 问题提出背景及设计目标

在经历了一学期的 Java 学习和编程实践后，有必要对所学知识进行复习和总结，体会完成一个完整项目的所需的思想和方法，并在提高编写程序的速度、准确性，优化代码风格和提高程序健壮性上努力。

考虑到五子棋程序设计具有以下特征：1）规则简单，容易上手，且在设计时较好的遵循软件设计的基本设计模式；2）不涉及过多额外的知识的前提下能够对所学内容进行较为综合的利用；3）具有良好的可拓展性等。因此我们课程设计选择了五子棋为设计目标。

本程序主要设计的重点分为四个部分，接下来将一一介绍并阐述了每个部分需要应用的知识和关键部分。

界面部分：

知识：GUI 编程（图形化用户界面）；重点：对 Swing 中各个组件的理解和灵活运用；难点：界面设计的简洁、美观、易用。

游戏逻辑部分：

知识：Java 基本语句的使用、持有对象、设计模式；重点;对游戏基本规则的把握、游戏设计流程的清晰；难点：代码的简洁易读，执行效率的提高。

AI 部分：

作为附加的部分，主要运用的知识为 AI 相应的算法以及 Java 基本语句的运用；重点在于游戏规则的深刻理解以及代码分析和调试的功能的熟练运用；难点在于效率的提升和算法或参数的优化。

网络部分：

网络部分是本课程和程序设计的重点。知识：输入输出流、多线程、网络编程。重点：服务器与客户端的信息交流；难点：线程逻辑处理、反馈顺序安排、关闭线程时处理事件等。

第二章 程序整体框架概述

下图展示了程序中所构建的全部类总览。主要可分为窗口类、核心部分以及辅助功能类。在一下的几章内将会作详细的说明。（注意：图示中灰色方框用来表示各类的功能，不再表示“对象”的含义）

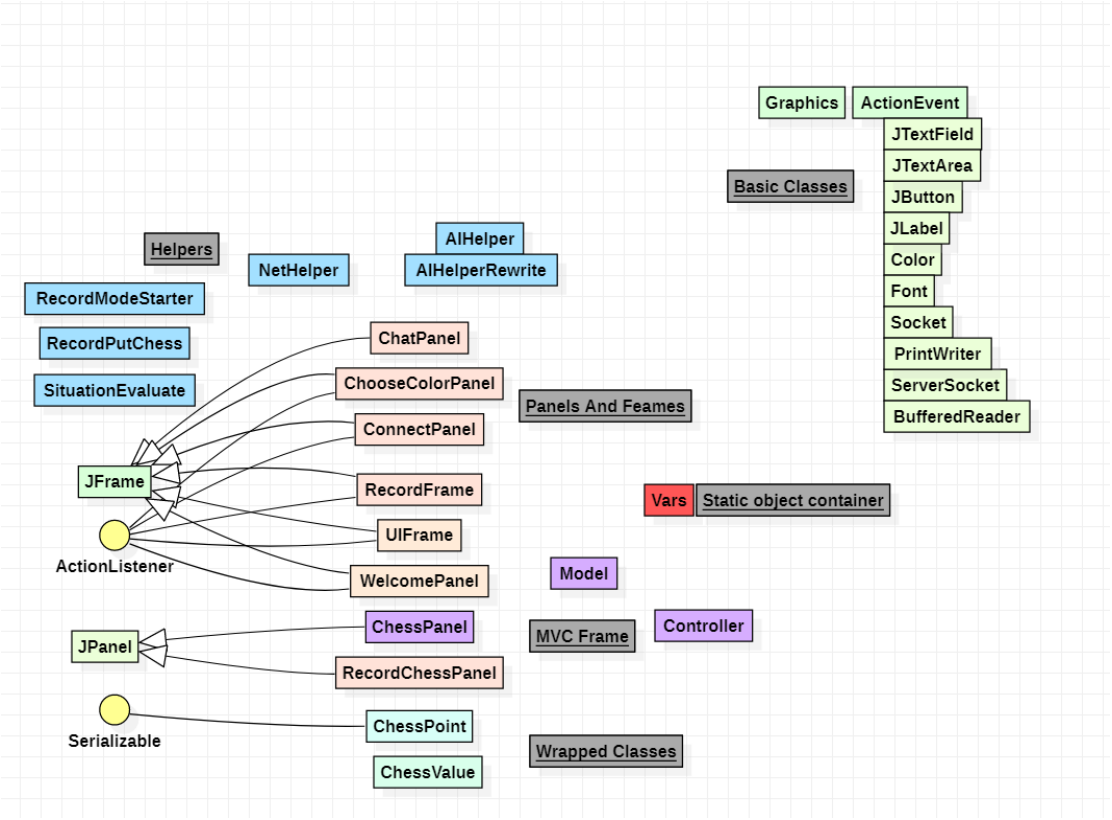


图 2.1 程序框架图

第三章 窗口及其对应类的设计

第一节 程序入口

本程序的一大亮点便是实现了多窗口之间的交互。相比大多数只设计了一个大面板的程序更加简洁明了，容易引导用户操作，并使程序的设计更加整体有序。当然，这也带来了额外的工作量，并需要合理的在窗口的展示和隐藏时进行事件处理。

3.1.1 入口界面

入口界面对应于类 WelcomePanel，同时也是主函数的入口。由于其特殊性，在后面部分窗口的构造函数中传入了该对象的引用，以便在必要的时候设置其显示和隐藏。同时，在该界面选择需要进入的模式中，并将其传入控制器中。

图 3.1 入口界面

3.1.2 WelcomePanel 类

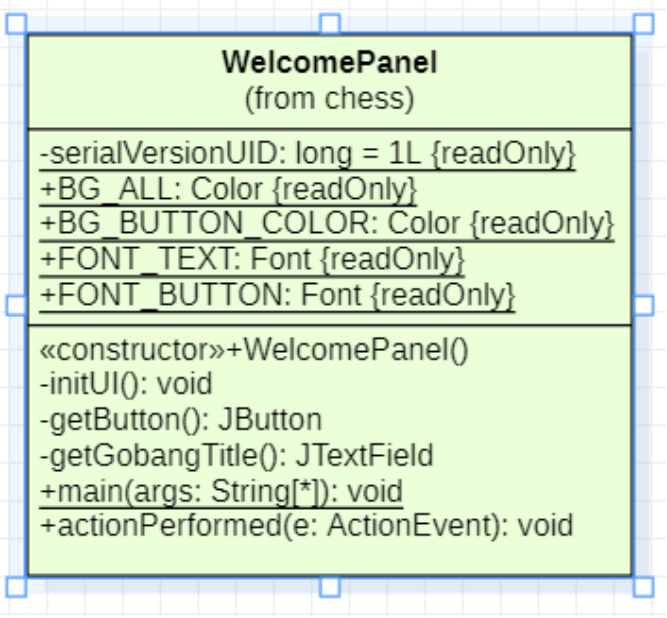


图 3.2 入口界面实体类类图

部分关键方法的说明：

1）private JButton getButton()

为 Button 添加统一的样式，返回“标准化”的 button

2）private JTextField getGobangTitle()

返回“GoBang”的样式

3）public void actionPerformed(ActionEvent e)

按钮的事件监听，监听按钮按下时的操作

第二节 不同模式间的过渡

当在入口界面点击 Self Play 后，会直接进入棋盘界面；点击 Play with AI 后会提示先选择棋子的颜色，之后进入 AI 对战棋盘界面；点击 Play Online 后会进入服务器连接界面；在此界面选择服务器 IP 并输入自己的名字。点击“Listen”可作为服务器端邀请朋友加入对战；点击“Connect”可作为客户端接受邀请，进入网络对战棋盘界面进行对战。下面将对其逐一进行介绍。

3.2.1 颜色选择类 ChooseColorPanel

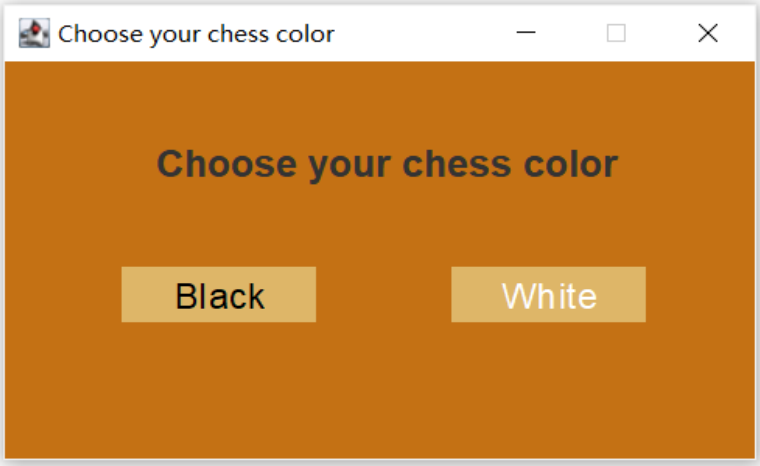


图 3.3 颜色选择界面

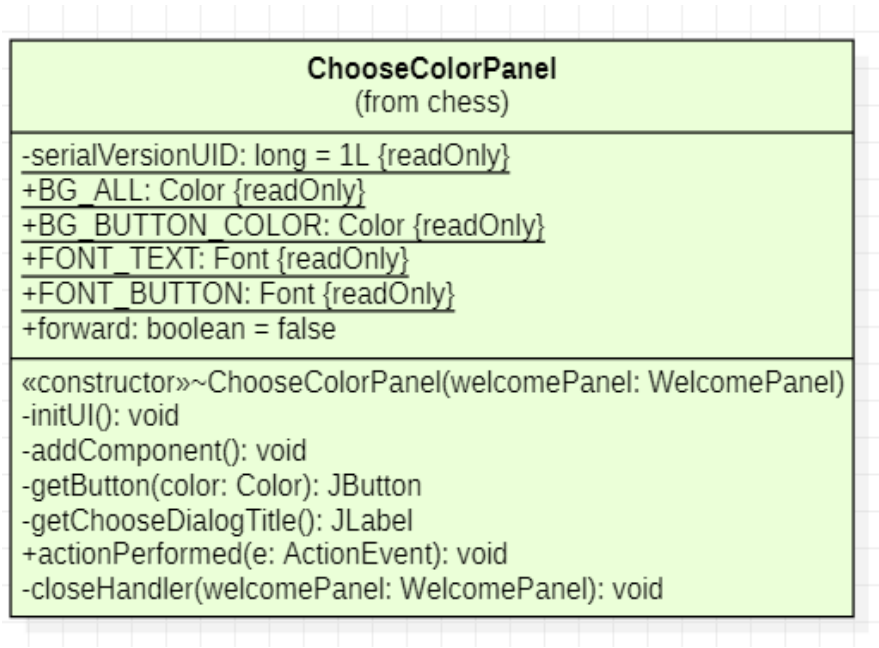


图 3.4 颜色选择类类图

部分关键方法的说明：

1）public ChooseColorPanel()

构造函数。调用 initUI 进行 UI 初始化；保存 WelcomePanel 对象，添加 CloseHandler 监听器

2）private void closeHandler()

根据 forward 变量的值（即根据因为前往棋盘界面而关闭还是因为用户放弃选择该模式而手动点击了×号）作出不同的事件处理。如果手动点击了×号，将 welcomePanel 置为可见，以使程序正常退出。

3.2.2 网络连接类 ConnectPanel

与 ChooseColorPanel 类似，只是由于网络连接的原因与其有两个主要的不同：一是在点击按钮时需要借助下面将要介绍的 netHelper 功能类进行连接及连接消息的发送；二是在关闭窗口时如果是没有连接成功或主动关闭窗口，应当在回到主界面的同时将服务器关闭。具体的方法除网络部分外与 ChooseColorPanel 相似，重复的部分不再赘述，仅介绍网络相关方法。

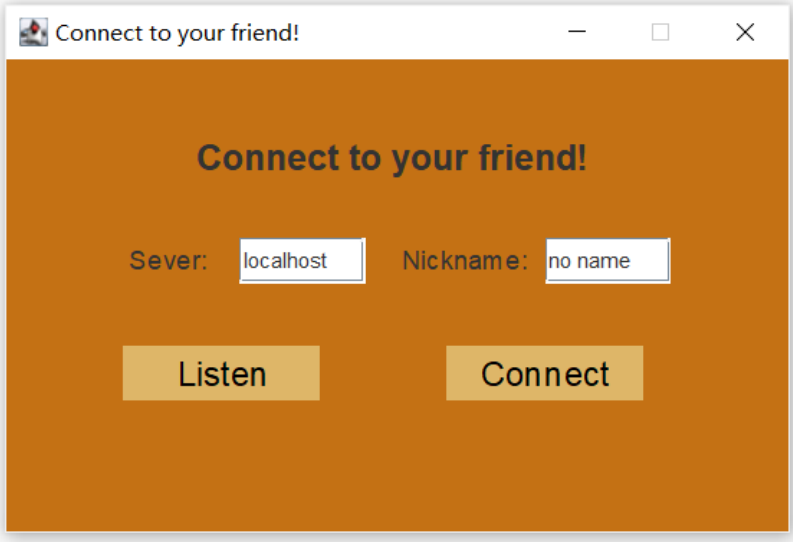


图 3.5 网络连接窗口

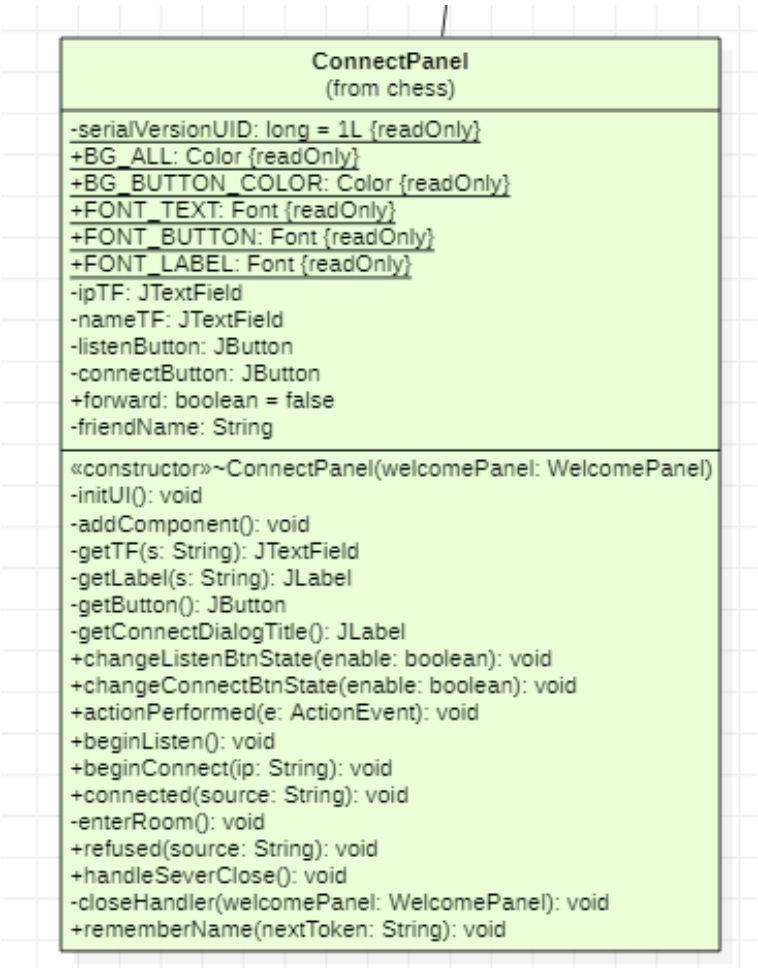


图 3.6 网络连接类类图

1）beginConnect ()和 beginListen()

调用 netHelper 请求连接

2）connected ()

若为服务器，提示成功连接，并向客户发送成功连接的信息，提示控制器设置为先手（黑棋）；若为客户，提示成功连接，设置为后手（白棋）。同时调用进入房间 enterRoom()方法,开始游戏。

3）private void closeHandler()

根据 forward 变量的值（即根据因为前往棋盘界面而关闭还是因为用户放弃选择该模式而手动点击了×号）做出不同的事件处理。如果手动点击了×号，将 WelcomePanel 置为可见，以使程序正常退出。

3.2.3 聊天面板类 ChatPanel

chatPanel 的实现与之前网络编程部分类似，在此不在赘述。

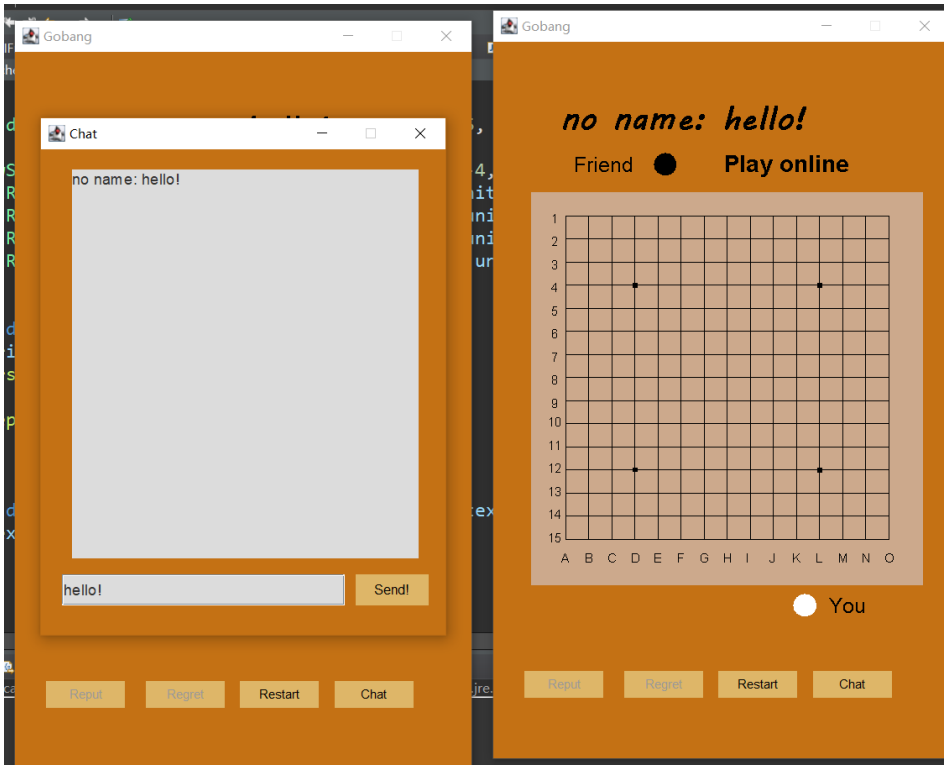


图 3.7 聊天面板

第四章 游戏核心类与对象的设计

第一节 Java MVC 设计模式

MVC 设计模式把一个软件组件区分为三个不同的部分:model、view 和 controller。模型(model)是实现控制数据访问与数据处理功能的应用程序。视图(view)是实现采集用户输入的数据并传递给控制器，或输出控制器中的处理数据给用户功能的应用程序。控制器(controller)是实现模型与视图之间的数据流向与数据转换功能的应用程序。[1]

简单的说，使用 MVC 的架构编写代码能够较好的理清整个项目各部分功能之间的联系，“专门的部分做专门的事”，减少代码的耦合度。在编写代码时，能够较好的使思路集中：比如在编写控制器部分时不需要关心功能的具体实现，编写模型部分时不需要关心界面上显示的大小、位置等信息，做到前后端分离；在调试代码时，能够较快的发现问题所在，并减少问题出现时所涉及的模块，减少测试和修改时的工作量。

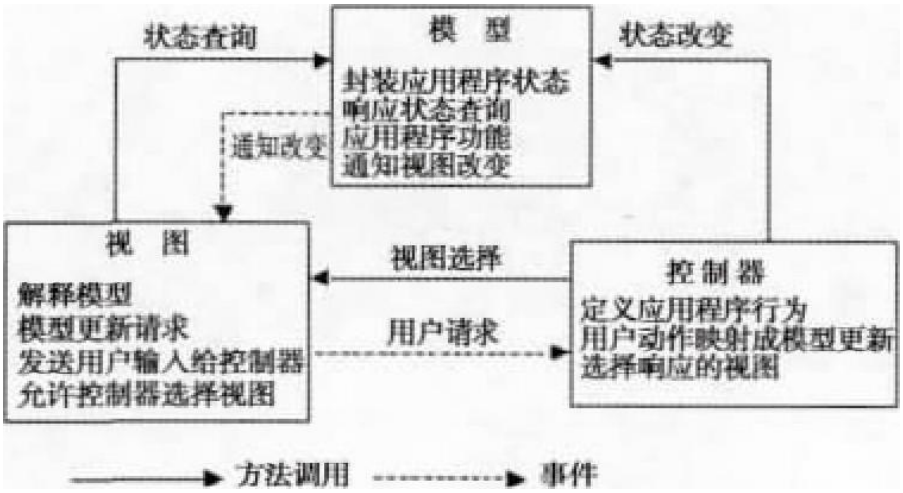


图 4.1 MVC 组件类型的关系和功能

MVC 设计模式的思想在实际软件工程中得到广泛应用。它是 Xerox PARC 在二十世纪八十年代为编程语言 Smalltalk－80 发明的一种软件设计模式，已被广泛使用。后来被推荐为 Oracle 旗下 Sun 公司 Java EE 平台的设计模式，在现代的 Spring、.NET 等框架中也被广泛使用。本次课程设计的核心部分同样采用了

MVC 的设计模式，以更好的学习系统化的编程思想，适应将来实际生活中的编程场景。

第二节 C-控制器类 Controller

图 4.2 Controller 类类图

部分关键方法的说明：

1）get()和 set()类方法：

为其他部分调用提供接口。尽管 Controller 中大部分关键变量都具有包访问权限（事实上更严谨的话完全可以并有必要设置成 private），但考虑到代码的可读性和优雅，我采用了 get()方法来获取诸如 openDoor 等关键变量，同时 setFirstHand()和 setGoteHand()方法将开局设置先后手的代码打包到一起，提高代

码的可读性和复用性。

2）handlexxx()和 ensurexxx()类方法：

handle 方法为按下请求按钮后触发的方法，若为 Self Play 模式或 AI Play 模式，则直接调用 ensurexxx()方法，参数为 true。若为网络对战，则分为请求方和被请求方。根据对方的同意情况改变传入 ensurexxx()方法的参数，并由 NetHelper 类调用。

2）xxputChess ()类方法：

用户点击鼠标后调用 localPutChes()方法。首先根据 openDoor 的值确定是否下棋，轮不到自己下直接 return；成功下棋后通知画板重绘棋子，通知 model 判定输赢后若存在赢家则根据不同的模式采取不同的语言、通知方式和行为。若不存在赢家，按照不同的模式进行下一步棋的操作。

ChessPanel 类通过 paintComponent 整体上通过“画板”的方式实现界面绘制，主要负责棋盘上的的事情，如棋盘和棋子的绘制、棋盘状态的保存，以及模式、对手、棋子状态的指示、同时，聊天中发送的实时消息也可像经典网络游戏“斗地主”一样显示在棋盘上方。

第三节 V-自定义棋盘类 ChessPanel

ChessPanel 类使用 paintComponent()整体上通过“画板”的方式实现界面绘制，主要负责棋盘上的的事情，如棋盘和棋子的绘制、棋盘状态的保存，以及模式、对手、棋子状态的指示、同时，聊天中发送的实时消息也可像经典网络游戏“斗地主”一样显示在棋盘上方。

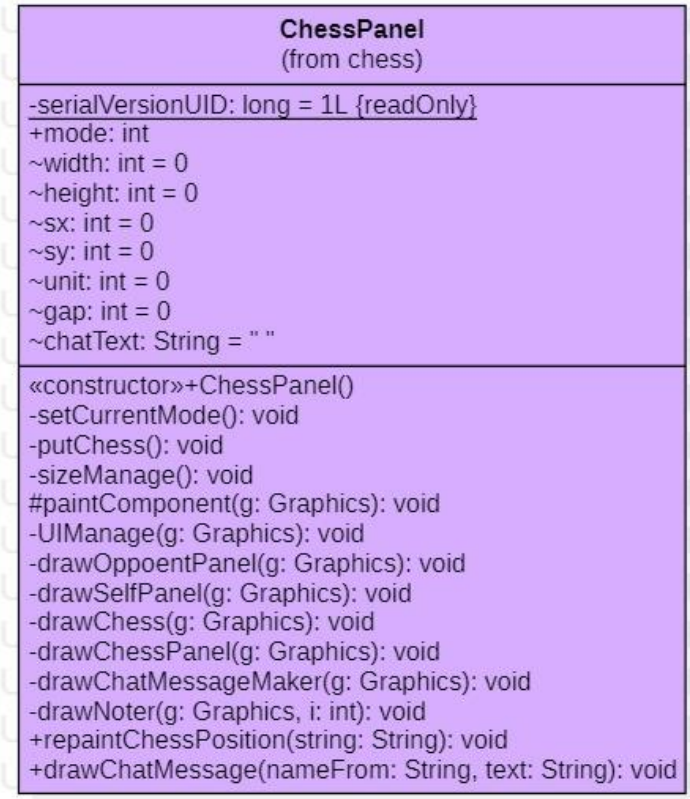


图 4.3 ChessPanel 类类图

部分关键方法的说明：

1）setCurrentMode ()方法

此处 set 的含义是针对自己对象的。获取 Controller 中的 mode，保存在本地变量中。由于在绘制棋盘时时常会根据不同的模式选择不同的绘制内容，故保存下变量，减少函数调用。

2）public void drawChessPanel (Graphics g)

绘制棋盘。此处用了不少心思对棋盘进行优化。棋盘大小全部采用相对数值，以根据不同的模式选择展示不同大小的棋盘。在离线模式中棋盘较大，在线模式由于组件较多，且要显示聊天文字，故棋盘会小一些。

3）public void repaintChessPosition(String s)

根据玩家不同的操作（悔棋，重放棋子等）改变光标形状或重绘棋盘等。

第四节 M-模型类 Model

Model 类主要是棋盘信息的保存。大部分内容与控制台版五子棋基本一致。在修改 AI 的时候还对 whoWin()的代码作了简化。

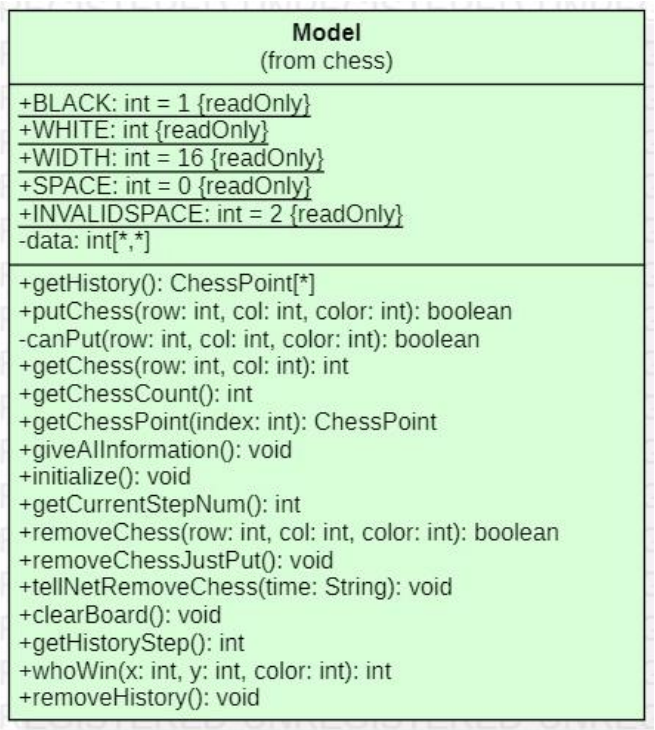


图 4.4. Model 类类图

部分关键方法的说明：

1）initialize()方法和 removeHistory()方法

两者的区别在于 initialize()方法是将棋盘的 invalid space 绘制好，方便 model 中边界条件的判断，在进入下棋界面时被调用；removeHistory()则为同时清除棋盘和历史记录，在重开和退出下棋界面时被调用。

2）其他方法

如提供输赢条件判断、获取棋盘信息和历史记录的接口、告知其他部分利用 model 中的信息进行处理等，在此不再一一赘述。

第五节 UIFrame

UiFrame 主要负责两件事：一是为棋盘和其他功能性按钮提供总面板；二是提供 OptionPanel，Dialog 等信息的统一“接口”，并根据不同情景下的调用绑定不同的事件。

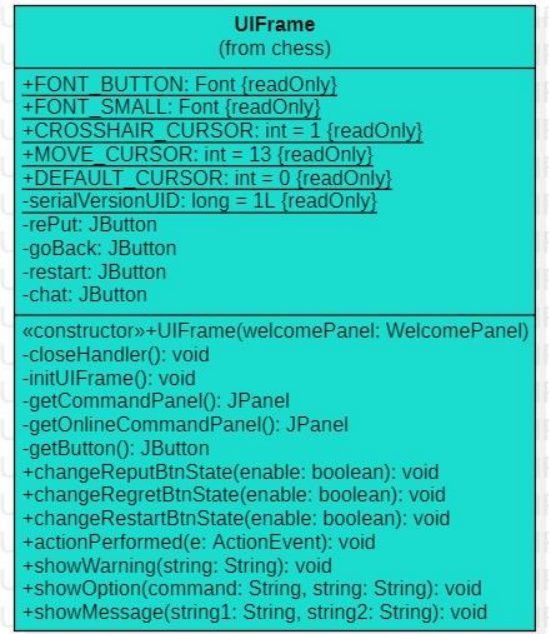


图 4.5 UIFrame 类图

部分关键方法的说明：

1）changexxxBtnState()方法

根据实际情况改变按钮是否可点击，可有效指导用户操作，并避免不必要的会产生异常的操作。比如：重新置子发生在我方已经放置棋子的情况下，因此只在相应的“窗口期”开启此方法的可调用状态；同时，在 AI 模式下由于在经过优化后电脑下棋速度非常快且重新置子会影响 AI 计算的结果，因此不允许重置；悔棋发生在当前未置子的状态下，在经过对方同意后会撤回两步，自己回到上一次己方下棋的情形。在 Self Play 模式中不需要悔棋，仅需不断重放即可。

2）其他方法

主要为弹窗，消息发送等的功能，因为 parentComponent 需要设置为该面板，也就是由这部分来显示，这一部分方法便放置到了 UIFrame 中。

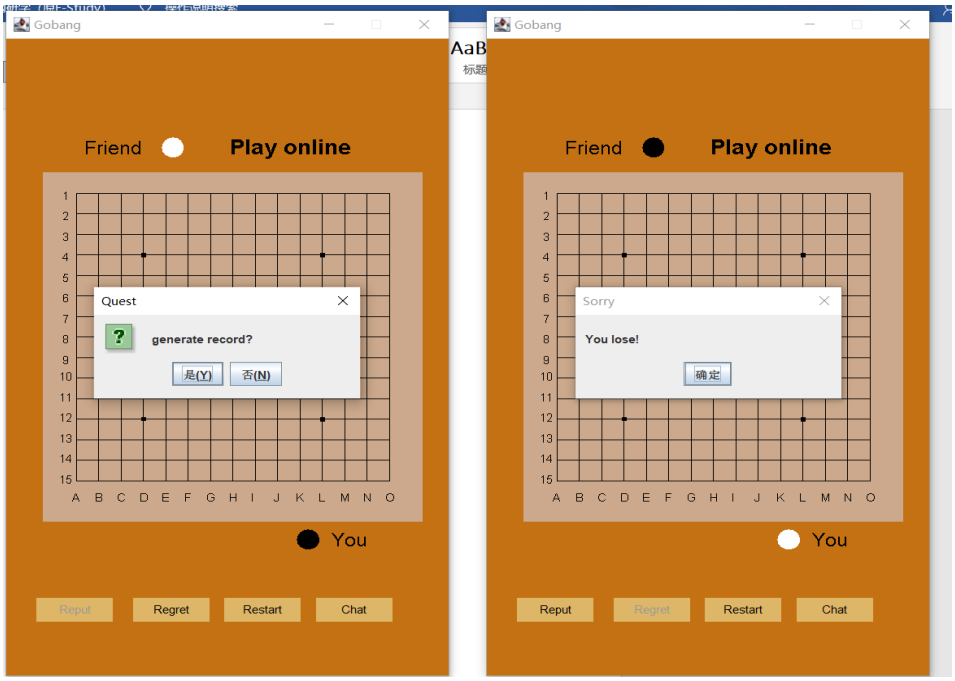


图 4.6　UIFrame 展示对局实战状况

第五章 辅助类与对象的设计

第一节 单例模式

单例模式(Singleton Pattern)：确保某一个类最多只有一个实例，并向整个系统提供这个实例，即该类需提供一个访问唯一实例的全局方法，这个类称为单例类。[2]

这种模式可以保证一个类仅有一个实例，并提供一个访问它的全局访问点。它能够在类与类之间联系较多时避免类的频繁创建和销毁，节省系统资源并提供一种优雅的方式使用单例对象。

当然，单例模式也有一些缺点。比如没有接口，不能继承等。当然，在课程设计中并没有涉及到此类问题，因此使用单例模式能够较好的实现程序的设计。

第二节 单例保存类 Vars

将需要频繁调用的类保存在静态变量中，以在需要的时候调用。当然，作为单例模式的局限性之一，我们也必须要关心类何时被实例化，以避免调用时的空指针问题。

第三节 网络功能提供类 NetHelper

Chess 类主要负责逻辑上的各个事件，如逻辑上落子、AI 局势分析、胜负判定等，Chess 类的设计如图 3 所示。

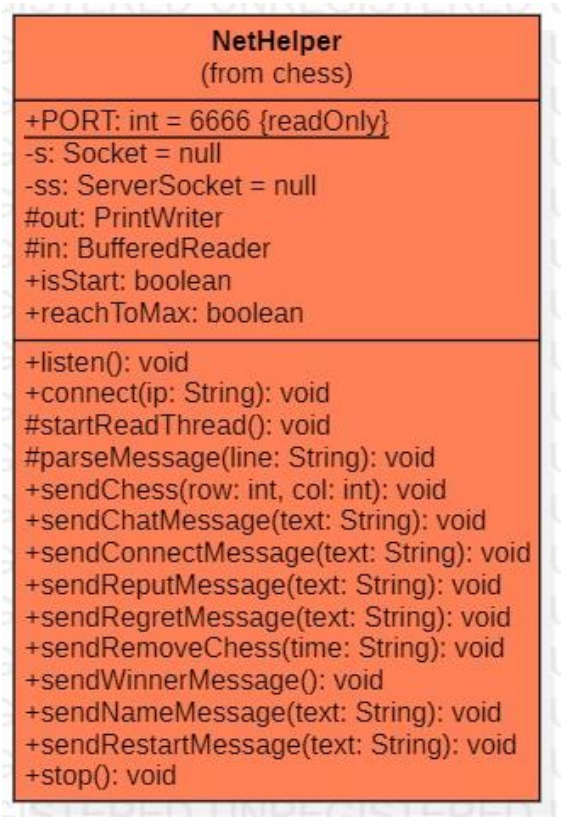


图 5.1 NetHelper 类类图

部分关键方法的说明：

1）sendxxx (String text)

发送不同类型的消息。当然，此处也可以更加简化：抽取 SendMessange 方法并增加区分类型的参数。

第四节 AI 功能提供类 AIHelper

此处代码有所重构，存在一些已废弃的方法，在此不再展示。这一部分将会在 AI 设计中详细介绍。

第五节 其他辅助类

5.5.1 ChessPoint 和 ChessValue 类

ChessPoint 类为棋子历史记录提供统一的包装类；ChessValue 在 AI 设计中

提供统一的包装类。

5.5.2 SimulationEvaluate 类

对当前棋盘局势进行分析。将在 AI 部分详细讲解。

5.5.3 Record 相关类

当在决出输赢后弹出“Generate record？”中选择“是”的话，将会另外弹出独立的窗口，在这个窗口可以对对战进行回放。我为 Record 相关类设置了独立的 MVC 架构，在 RecordModeStarter 的构造函数中只需要传入必要的信息（棋盘、棋子历史记录以及当前模式和己方持子颜色）即可独立于主程序流程之外进行操作。只要主程序不关闭，随时可以查看这局的回放，并根据边界情况合理的引导用户的操作。复盘界面如下图所示。

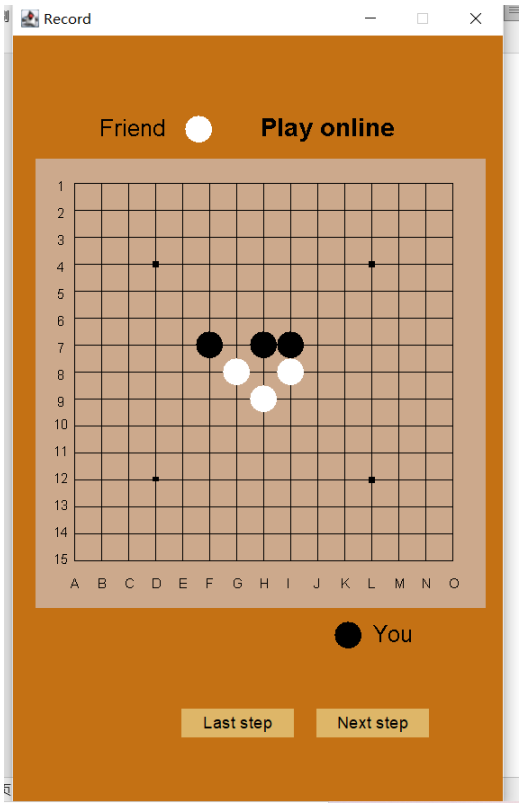


图 5.2 复盘界面

第六章 人工智能的设计

第一节 AI 实现局势判断

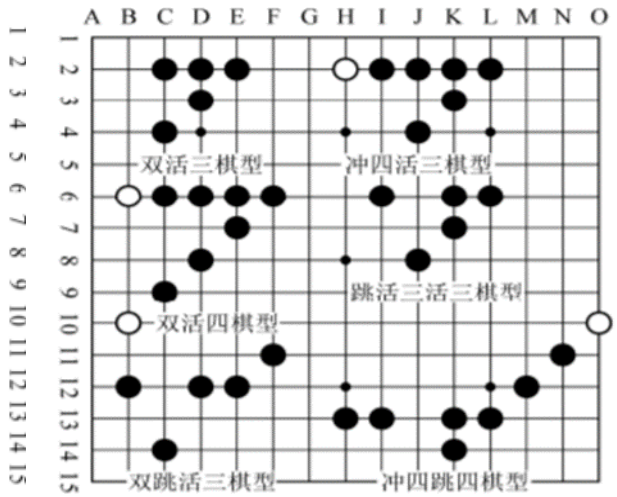
我们时常会想，五子棋 AI 是怎么确定下在哪个位置上的，首先最容易想到的是，玩家落子后会形成一个局势，在该局势下 AI 依次为每个空位打分，得分最高的位置就是 AI 落子的位置。

那么问题是：怎么为空位打分？

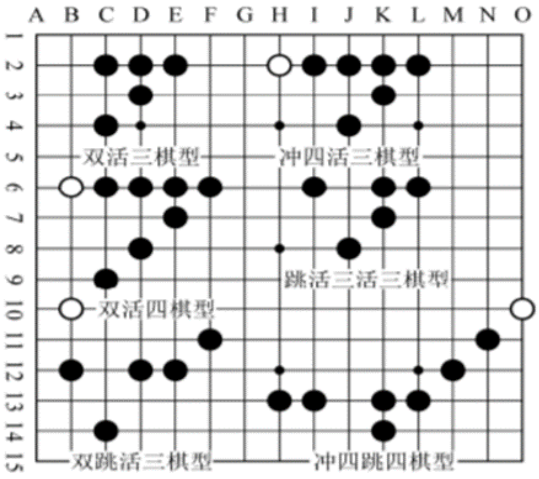
五子棋中有成五、活四、冲四、死四、活三等诸多概念，我们需要对可行的落子位置进行局势的判定，也就是 SituationEvaluate 类所做的工作。至此，我们需要先对五子棋的基本棋型进行归纳和介绍。

6.1.1 五子棋基本棋型

在五子棋博弈中，活三和冲四是五子棋着棋过程最重要的棋型，活三两头均可进攻，冲四具有绝对先手，连续的冲四和活三的进攻是五子棋获胜的关键技术．而活三的一些变体棋型，如有一边空一格被对方棋子拦住，如图 5.1a 的棋型 g 所示．此外跳活三、跳四，也是十分重要的棋型．跳四和冲四一样，具备绝对先手。



(a)



(b)

图 6.1 五子棋基本棋型

在 SituationEvaluate 中，考虑了五连，活 4，特殊情况下的双活 3 等必胜局势，以及冲四、跳三、眠三以及活二、眠二棋型，对于图中列出的其余组合情况，考虑到代码的复杂程度并没有完全穷举。

6.1.2 评估函数

评估函数是一个对单个可行位置评分的方法，比如它对某个可行位置 pos 进行评估，评估步骤如下：

1）横向扫描

A.以可行位置 pos 的左侧为中心，向左扫描

如果遇到空格，记录下左侧为空格，停止向左扫描

如果遇到己方棋子，棋子个数加 1，继续向左扫描

如果遇到对方棋子，记录下左侧为对方棋子，停止向左扫描

如果已到达最左侧，记录下左侧为墙，停止向左扫描

B.以可行位置 pos 为中心，向右扫描

如果遇到空格，记录下右侧为空格，停止向右扫描

如果遇到己方棋子，棋子个数加 1，继续向右扫描

如果遇到对方棋子，记录下右侧为对方棋子，停止向右扫描

如果已到达最右侧，记录下右侧为墙，停止向右扫描

C.根据棋子个数、评分表为该位置打分，该空位得分 score1

2）纵向扫描

扫描纵向上的相连己方棋子个数，根据棋子个数、评分表为该位置打分，该空位得分 score2

3）正斜向扫描

扫描正斜向上的相连己方棋子个数，根据棋子个数、评分表为该位置打分，该空位得分 score3

4）反斜向扫描

扫描反斜向上的相连己方棋子个数，根据棋子个数、评分表为该位置打分，该空位得分 score4

5）对以上四个方位之后可以得到 score=score1+score2+score3+score4，我们将总得分作为该空位的评分，这样就能使 AI 考虑到周围四个方向。

6.1.3 算法效果

AI 下的位置就是在所有可行位置中评分结果最高的位置，评分结果最高意味着 AI 觉得这个位置对自己是最有利的，这个算法效果怎么样呢？ AI 是白棋，玩家先手，由图，AI 的第 6 手是个败笔，AI 第 6 手应该封堵黑棋，但是它并没有这个意识，可知如果仅是静态的考虑当前的棋局，这样的 AI 是非常脆弱的，不具备”防”的意识，只有“攻”的意识。原始算法的效果如图 6.2 所示。

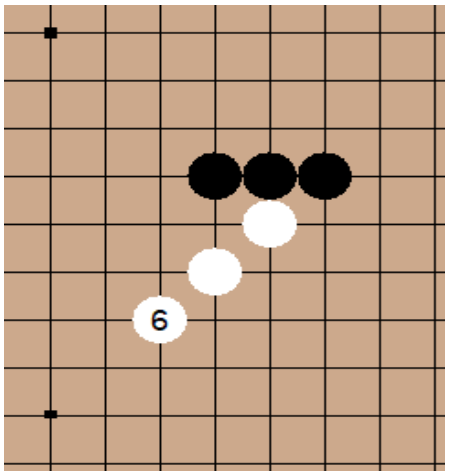


图 6.2 初始算法效果图

第二节 考虑对手的博弈算法

6.2.1 MIN-MAX 博弈树

根据理性博弈人的基本假设，为了简化搜索逻辑，我们可以做出这样的简化：己方在进行判断时，总是为自己选择利益最大的行为，在模拟其对方判断时，总是选择使得己方利益最小的行为。此时，整个的博弈过程为：轮到己方行为时，进行搜索，在当前局面下所有可能的行为中选择对己方利益最大的一种行为。但在实际情况中，一次考虑总是远远不够，因此需要一定的搜索深度。即交替的为双方进行模拟行为，并以此为己方优化决策。[3]按照如上假设及搜索的基本思想，便是极大极小搜索，其基本思想如下：在每一次迭代中，为每一个局面生产所有可能的子局面，通过极大极小的原则，为不同的局中人选择行为，在向上回溯的过程中，通过子局面的评估，给出父局面的评价。继续向上回溯,依次类推,一直到当前局面,最终相对的最优行为也就搜索出来了。

在第一版的 AI 算法中，我将搜索深度设置为 4。此时发现电脑用了太长时间去计算棋子的位置。大约需要 10 秒才可以返回结果，而且返回的结果也并不理想。这显然不能满足实际的要求。

棋盘上有 225 个交叉点，仅仅第二层就会有 225 个分叉，第三层有 225*2 个分叉，不得不说计算量太大了，对 AI 来说，仅仅预测三步的计算量都是非常大的。所以要用到 α－β 剪枝提高搜索效率。

6.2.2 α － β 剪枝搜索

α－β 剪枝搜索和剪枝与子节点的排列顺序有关。在 MAX 层节点的值是子节点中的最大值，按节点值从大到小的顺序排列 MIN 层节点的话，计算第一个 MIN 层节点后会更新其父节点即 MAX 层的最大下界，如果在计算后面 MIN 层更新其最小上届的值比其父节点 MAX 层的最大下届还要小，那么该 MIN 层后面的节点就没有继续搜索下去的必要了，因为 MIN 层最大值都比 MAX 层的最小值还要小，直接触发剪枝，没有必要在继续搜索下去。同理 MAX 层按子节点值从小到大排列也会有同样的效果。

如图 5.3 中，第二层中的 B 和 C 节点，按照现在的顺序排列，因为 B 节点的值 3 比较小，更新父节点 A 的最大下界为 3，而节点 C 第一个子节点的值是 6 比 3 大，还要继续计算下一个子节点的值，也就是需要完整计算第二层 C 节点的所有子节点。如果将它们互换位置，因为第一个节点 B 的值 6 比 3 大，计算完后更新父节点 A 的最小下界为 6，计算 C 节点的第一个子节点后更新 C 节点的最大下界为 5，比父节点 A 的最大下界还要小，那么 C 节点后面的子节点就没有继续搜索计算下去的必要了，触发了剪枝。所以，α－β 剪枝的效率和节点排序有很大关系，如果最优的节点能排在前面，则能大幅提升剪枝效率。

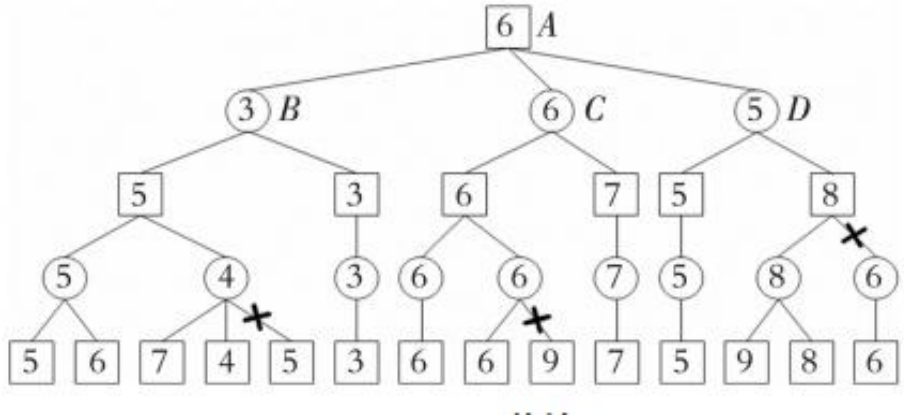


图 6.3 α－β 剪枝搜索示例

在本程序的代码中编写了 α－β 剪枝搜索的代码。然而，在测试的过程中遇到了一些问题。猜测可能的原因是权值算法中对棋子的遍历存在一些问题，从而导致了错误的剪枝，结果便是 AI 总是倾向于将棋子下在棋子的周围。经过细致的检查，已经解决了这个问题，但是由于时间紧迫，在提交的代码中并没有做出修改。

第七章 课程设计的总结和收获

第一节 程序亮点

首先，在本程序中，实现了对模式的拓展，特别是 AI 的设计是一个非常具有挑战性的工作，对以后相关算法编程的内容具有重要的帮助作用。

其次，相比简单的“大板型”界面，本程序较好的实现了界面之间的交互，能够更好的引导用户的使用；同时扁平化的界面更加美观整洁，对各部分组件定制化设计能够较好的弥补 Swing 界面较丑的不足。

再次，通过以上类与方法的分析可以看出，整个程序的设计思路是非常井然有序的，在编写代码时也是不仅注重功能的实现，也注意代码实现方式的优雅。

第二节 不足之处

由于在 AI 部分的工作是十分耗费时间精力的，在其他拓展功能比如倒计时、音乐并没有做太多，当然通过查阅相关的资料相比 AI 的难度还是相对容易的。

相应的，在 AI 部分的代码实现也不算特别理想。在提交的代码中能够使用的版本是使用暴力搜索的方法，在优化后能够达到较好的棋力和效率，当然在对权值进行进一步优化和对更多情况进行考虑，甚至在以后的学习中采用深度学习等更加高级的方法，还能够使 AI 得到进一步提升。

第三节 收获

到目前，课程设计基本实现完成。在程序的设计过程中，我对 Java GUI，网络编程，多线程等知识更加熟练运用，同时更加透彻的理解了 Java 面向对象的特点，学会了在完整的工程中如何合理分配对象的功能以及处理对象与对象之间的通讯和联系。当然，在其中也灵活运用了之前其他作业中实现的内容。

致 谢

这一次的大作业设计让我受益匪浅。它让我全身心投入其中，为新功能的创建和程序的稳定而深入思考。虽然之前也使用过 JS+HTML+CSS 开发过实际的项目，但终究不如这一次在短时间内完成既定功能、并在代码方面对自己有所约束收获更多。

整篇论文的撰写也颇费心思，我并没有因为字数多，时间紧而去“水字数”，而是将项目的框架一一捋清，并渗透在其中用到的思想方法。也正因没有“凑字数”的部分，我只在关键和必需之处引用了文献，将更多的心思花在原创性的工作上，没有掺杂任何与课程设计关系不大的文字。作为大作业的设计引用 10 篇以上文献也未免有些死板，我也有信心在不引用过多文献的前提下能够写出老师满意的论文。

当然仍有做的不足的地方。网上关于五子棋 AI 的资料我也看的不少。思路是相对清晰的，但实践起来是比较有挑战性的。同时由于时间的紧迫，AI 方面有些细节做得不够好。期间参考网上所学以及自己总结思考总共写了 3 次 AI 的实现代码，由于在 AI 方面投入精力过多，最终在其他相比 AI 实现上难度并不大附加功能方面的实现相对也较薄弱，努力和用心的地方有些偏离课程设计的方向。

在课上学的内容难免十分有限，比如数据库，Spring 等在实际工程中广泛应用的内容难以完全渗透，Swing 相比现在迅速发展的前端设计也显得有些“过时”，但课上的内容是我们以后继续深入学习 Java 的基础，以后在实际应用时能够更轻松的上手，有更广阔的发展空间。

最后还要感谢刘嘉欣老师，作为“老程序员”，对 Java 的发展和特性的理解非常深刻透彻，同时也表示“难以在一学期有限的时间内将 Java 所有精髓悉数教于我们”。在程序的设计过程中不说“20 小时写代码，4 小时睡觉”，也将大量的时间精力投入其中，其中两天也熬了夜。作为转专业来到计网的同学，在这次课程设计中也第一次在课程学习的短时间内完成较系统的工程，极大的提高了我编写代码的熟练度、耐心和毅力，受益匪浅，对以后的学习也将会有极大的帮助。

参考文献

[1]侯丽敏,杨俊红,杨志献.MVC 设计模式在 Java 实训项目中的应用[J].郑州铁路职业技术学院学报,2013,25(03):77-79.

[2]王凯琪,兰全祥.Java 中单例设计模式的分析及应用[J].信息技术与信息化,2021(05):112-114.

[3]周子龙.博弈搜索树算法的实现及其优化[J].科学技术创新,2021(18):108-110.