

**本 科 毕 业 论 文**

**­­­­­­­­­**

基于QT的中国象棋游戏的设计

**Design of Chinese Chess Game Based on QT**

学 院 名 称： 计算机科学与信息工程学院

专 业 班 级： 计算机科学与技术（信息技术）13-1

学 生 姓 名： 曹志勇

学 号： 13031010205

指导教师姓名： 李爱玲

指导教师职称： 教 授

2017年 5月

**毕业设计（论文）原创性声明和使用授权说明**

**原创性声明**

本人郑重承诺：所呈交的毕业设计（论文），是我个人在指导教师的指导下进行的研究工作及取得的成果。尽我所知，除文中特别加以标注和致谢的地方外，不包含其他人或组织已经发表或公布过的研究成果，也不包含我为获得安阳工学院及其它教育机构的学位或学历而使用过的材料。对本研究提供过帮助和做出过贡献的个人或集体，均已在文中作了明确的说明并表示了谢意。

作 者 签 名：　　 　　 日　 期：

指导教师签名：　 日　 期：

**使用授权说明**

本人完全了解安阳工学院关于收集、保存、使用毕业设计（论文）的规定，即：按照学校要求提交毕业设计（论文）的印刷本和电子版本；学校有权保存毕业设计（论文）的印刷本和电子版，并提供目录检索与阅览服务；学校可以采用影印、缩印、数字化或其它复制手段保存论文；在不以赢利为目的前提下，学校可以公布论文的部分或全部内容。

作者签名：　　 　 　 日　 期：

目 录

[摘要 I](#_Toc483515635)

[Abstract II](#_Toc483515636)

[引言 1](#_Toc483515637)

[第1章 绪论 2](#_Toc483515638)

[1.1 课题背景 2](#_Toc483515639)

[1.2 课题目的和意义 3](#_Toc483515640)

[1.3 设计思想 3](#_Toc483515641)

[1.4 开发技术简介 4](#_Toc483515642)

[1.5 环境简介 4](#_Toc483515643)

[第2章 系统需求分析 6](#_Toc483515644)

[2.1 系统需求分析 6](#_Toc483515645)

[2.2 可靠性和可行性需求分析 6](#_Toc483515646)

[第3章 系统设计 8](#_Toc483515647)

[3.1 功能模块设计 8](#_Toc483515648)

[3.2 系统流程设计 8](#_Toc483515649)

[3.3 系统数据结构设计 9](#_Toc483515650)

[3.4 走法设计 10](#_Toc483515651)

[第4章 系统实现 12](#_Toc483515652)

[4.1 棋盘界面的实现 12](#_Toc483515653)

[4.1.1棋盘的绘制 12](#_Toc483515654)

[4.1.2 棋子的绘制 13](#_Toc483515655)

[4.2 走棋与悔棋 14](#_Toc483515656)

[4.2.1 走法的实现 14](#_Toc483515657)

[4.2.2 悔棋的实现 16](#_Toc483515658)

[4.3 人工智能的实现 17](#_Toc483515659)

[4.3.1 搜索算法的实现 17](#_Toc483515660)

[4.3.2 估值函数的实现 18](#_Toc483515661)

[4.3.3极大极小值算法和Alpha-Beta算法的实现 18](#_Toc483515662)

[4.4 对战的实现 21](#_Toc483515663)

[4.4.1 人机对战 22](#_Toc483515664)

[4.4.2 人人对战和网络对战 23](#_Toc483515665)

[4.4.3 多种对战方式的选择 24](#_Toc483515666)

[4.5 判输检测 25](#_Toc483515667)

[第5章 系统测试 26](#_Toc483515668)

[5.1 测试方法 26](#_Toc483515669)

[5.2 对弈模式选项模块测试 26](#_Toc483515670)

[5.3 人工智能模块测试 27](#_Toc483515671)

[5.4 走法的测试 28](#_Toc483515672)

[5.5 网络对战模块测试 29](#_Toc483515673)

[5.6 将军测试 30](#_Toc483515674)

[总 结 32](#_Toc483515675)

[致 谢 33](#_Toc483515676)

[参考文献 34](#_Toc483515677)

基于QT的中国象棋游戏的设计

摘要：本课题研究的是以中国象棋为背景的跨平台棋类游戏，主要由界面程序和人工智能两大部分组成。其中界面程序包括对战模式界面设计，棋盘界面设计。在对战模式选择界面，需要考虑人人对战，人机对战，网络对战（服务端），网络对战（客户端）的不同需求。而在人工智能部分，主要体现在人走棋后计算机会采取合适有效的应对策略，主要包括一步象棋人工智能，两步象棋人工智能，走n步的人工智能，Alpha-Beta算法，极大极小值算法等的设计。

系统采用QT作为开发平台，并用C++作为编程语言，设计并实现适合于普通大众的中国象棋游戏。QT是一款支持多平台的开发库，并且对于GUI程序的开发和非GUI程序的开发均得到了支持，可以较好的满足设计一款中国象棋游戏的需求。

关键字：QT；中国象棋；人工智能；C++；Alpha-Beta算法

**Design of Chinese Chess Game Based on QT**

**Abstract:** This topic is the study of Chinese chess as the background of the cross-platform chess game, mainly by the interface program and artificial intelligence composed of two parts. The interface program includes battle mode interface design, checkerboard interface design. In the battle mode selection interface, need to consider the different demands of Man-man battle, Man-machine battle, Network battle (server), Network battle(client).In the artificial intelligence part, mainly reflected in the computer will take appropriate and effective coping strategies after people take a move, including One-step chess artificial intelligence, Two-step chess artificial intelligence, walking n-step artificial intelligence, Alpha-Beta algorithm, Minimax Algorithm and so on.

The system uses QT as the development platform, and uses C ++ as the programming language to design and realize the Chinese chess game suitable for the general public. QT is a multi-platform development library, and for the development of GUI programs and non-GUI program development are supported, it can better meet the design of a Chinese chess game needs.

**Key words:** QT; Chinese Chess; Artificial Intelligence; C ++; Alpha-Beta Algorithm

# 引言

在现在快节奏的生活状态下，棋类游戏是一个很好的休闲替代品。同时，棋类游戏的人工智能部分的完善程度也是计算机科学发展程度的标志之一。而中国象棋兼具棋类游戏的益智属性和中国传统文化属性。鉴于此，本课题致力于实现一个跨平台的中国象棋游戏的开发，以此丰富同学们的课余生活，增进大家对传统文化的了解。同时，通过课题开发也可以加强自身对人工智能的了解。

随着人类计算机水平的不断进步，人工智能也变得越来越火热。从卡斯帕罗夫与“深蓝”的世纪大战，到李世石与“阿尔法狗”的激烈碰撞，人工智能的发展取得了长足的进步。虽然人类有着既希望战胜机器又希望机器发展出完美的智力的矛盾心理，但不可否认的是人工智能的发展对于未来的人类生活来说及其重要。人工智能的发展，将会使我们的生活更加便捷，我们有理由去不断的发掘人工智能的发展潜力。

基于QT的中国象棋游戏采用QT（Quasar Technologies）作为开发平台，并用C++作为编程语言，加入适量的人工智能思想，设计并实现适合于普通大众的中国象棋游戏。本软件界面简洁，操作简单，对战模式较为齐全，可以满足不同的对弈需求。而且依托于QT的跨平台特性，可以满足用户在不同的平台进行游戏，因此这是一项不错的中国象棋游戏设计。

本论文基本框架结构如下：第一章绪论，主要阐明了基于QT的中国象棋的设计的课题背景、研究目的以及实际意义，同时也简要介绍了本软件依托的设计思想和理论技术基础等；第二章分析了一个中国象棋游戏的系统需求，第三章结合第二章的分析，做出了一些合理的设计，为后面的系统实现打下了良好的基础；第四章是本论文的核心内容，主要讲述了实现基于QT的中国象棋游戏的具体过程；第五章主要针对系统设计出来的功能进行测试，并且将发现的一些漏洞填补上；最后，就是对于本次设计的一些经验总结，对在软件设计过程中提供过帮助的人表示感谢，以及罗列设计过程中用到的参考文献。

# 第1章 绪论

现代生活节奏疾驰如车，而棋类游戏是一个很好的休闲替代品。基于QT的中国象棋游戏主要面向的是工作繁忙，无暇长时间娱乐的人群。本软件界面简洁，只要掌握了基本的走棋规则，就能体验到传统象棋给用户带来的乐趣。同时，该软件还加入了适量的人工智能模块，即便是一个人也可以下棋，大大增加了娱乐性。下面，我们将从本软件的课题背景等相关信息出发，为大家论述一个基于QT的中国象棋游戏的完整开发过程。

## 1.1 课题背景

与国际象棋相比，中国象棋软件研究较晚。台湾学者在上世纪80年代初才开始进行电脑棋类方面的研究。在大陆方面，南开大学于1983年设计出一款残棋程序，之后全局程序又被设计出来。虽然这个阶段的棋力很低，但是却为计算机象棋的发展奠定了坚实的基础。里程碑式的中国象棋软件出现于1986年，由台湾著名人工智能专家俞锡顺先生撰写的“国际象棋大师”成为在未来十年的一方霸主，直到本世纪初垄断才被打破。

二十一世纪初，“象棋世家”和“棋隐”开始商业化，后来的“楚汉棋缘”“谢谢大师”等紧随其后。之后的“象棋精灵”是一款集象棋教学、人机对弈等功能于一身的软件。之后优秀的人工智能方面的专家黄晨，更是建立了名为象棋百科全书网的信息平台，确立了相关的行业标准。

行业标准成立后，一些优秀的象棋软件也相继出现。“象棋倚天”及其姐妹版“象棋旋风”是那个时期最受追捧的象棋软件。象棋旋风引擎棋力强大，功能完善，局面判断准确，运行速度快，界面美观，自带连接，无需第三方连接器即可连接到主要游戏网站。“倚天象棋”采用最先进的人机博弈算法，2008年诞生后便在北京举行的第十三届电脑奥林匹克大赛中国象棋组中夺得冠军。

“象棋名手”于2005年开发，随后参加了2007年在重庆举行的第二届中国机器博弈锦标赛，之后经过不断地测试和完善，棋力大大提高。象棋名手在2011年中国机器博弈锦标赛中获得11胜1和的成绩获得中国象棋软件冠军。之后象棋名手连续四年荣获中国象棋软件冠军。是近年来表现最好的象棋软件之一。

要改进软件引擎，有必要反复让软件与人类强者博弈，然后将问题反馈到软件的设计中。从难以匹敌人类到势均力敌，再到战胜人类，象棋软件经历了漫长的过程。

## 1.2 课题目的和意义

在现在快节奏的生活状态下，棋类游戏是一个很好的休闲替代品。同时，棋类游戏的人工智能部分的完善程度也是计算机科学发展程度的标志之一。而中国象棋兼具棋类游戏的益智属性和中国传统文化属性。鉴于此，本课题致力于实现一个跨平台的中国象棋游戏的开发，以此丰富同学们的课余生活，增进大家对传统文化的了解。同时，通过课题开发也可以加强自身对人工智能的了解。

中国象棋是一种起源于中国的棋戏，有着悠久的历史，在民间有着广泛的群众基础。我们经常可以看到街边有人进行象棋对战，然后有一群人围观。象棋集文化、科学、艺术、竞技于一身，可以说，象棋是一种老少兼宜，修身养性，怡神益智的有益活动。随着物质生活水平的提高，单纯的实体象棋已无法满足人们快节奏的生活。但是依托于计算机科学的发展，这个问题便有了可以解决的方案。利用编程实现中国象棋的在不同平台的开发，可以满足人们在快节奏的生活状态下可以随时随地展开象棋对战的需求，以此来放松心情，陶冶情操，甚至在对战过程中增进友谊，提高知识水平。基于此等原因从而提出对基于QT的跨平台中国象棋游戏的研究进行实施与探讨。

## 1.3 设计思想

本软件主要由界面程序和人工智能两大部分组成。其中界面程序包括客户端界面设计，棋盘界面设计。在对战模式界面，需要考虑人人对战，人机对战，服务端，客户端的不同需求。而在人工智能部分，主要体现在人走棋后计算机会采取合适有效的应对策略，主要包括一步象棋人工智能，两步象棋人工智能，走n步的人工智能，优化-剪枝算法，最小值最大值算法等的设计。设计思路如图1-1所示。



图1-1 设计思路图

## 1.4 开发技术简介

本课题采用QT作为开发平台，并用C++作为编程语言，设计并实现适合于普通大众的中国象棋游戏。QT是支持跨平台的[C++](http://baike.baidu.com/item/C%2B%2B) GUI应用程序开发平台。它既可以开发图形界面相关的GUI程序，又可以制作控制台工具和服务器等非GUI程序。QT的图形用户界面的基础是QWidget，它派生出了QT中所有的GUI组件。QT放弃了传统的callback机制来进行对象之间的沟通，而采用了信号与槽机制来完成这一过程。利用QT的编写的源代码无需做任何修改就可以很好的移植到不同的操作平台。基于QT的这些特点，可以较好的满足软件设计的需求。

## 1.5 环境简介

软件开发环境是基于软件所需的基本硬件和数字软件相结合的开发平台。基于QT的中国象棋游戏的开发环境如下：

1.系统的软硬件平台

内存：4GB ；

CPU：Intel(R) Celeron(R) CPU 1007U @ 1.50GHz 1.50GHz；

硬盘：500GB；

操作系统： Windows 10(64位)；

2.开发语言和工具

开发语言：C++；

开发工具：QT 5.4.1

QT是一个跨平台的C++应用程序开发框架。QT的源码对用户开放，大部分主流编译器都得到它的支持，用户常用的GCC的C++编译器和微软的VS系列也在其中。使用其开发的软件，在不需要修改任何源代码的情况下，同样的代码可以在其支持的任何平台上编译和运行，只是会由于平台的限制而表现出不同的图形界面风格。而QT支持的平台几乎包括了Windows、MAC、Linux等在内的所以主流平台。因此对于开发跨平台的应用，QT是不二之选。

# 第2章 系统需求分析

系统需求分析的主要内容是对系统应包含的功能和所受到的约束的一种描述，这些需求反映了用户需要系统帮助解决的一些问题。系统需求分析应该是准确的，它有时也被称为系统的功能描述。软件定义阶段的最后一步即是需求分析，它确定了系统必须完成哪些工作，即对目标系统提出清晰、完整、具体、准确的要求。做好系统需求分析，可以节省时间和精力，我们才能顺利完成软件设计工作。

## 2.1 系统需求分析

功能需求分析包括对系统应该提供的服务、如何对特殊输入做出反应，以及系统在特定条件下的行为的描述。在某些情况下，功能需求还需要声明系统不应该做什么。基于QT的中国象棋游戏作为一款象棋游戏，首要实现的就是能够在已有的象棋规则下进行对弈。同时，为了增加对弈的趣味性和可玩性，合理的人工智能也必不可少。最后结合简洁的界面设计，才能称得上一款不错的象棋游戏。

经过对基于QT的中国象棋游戏的调查分析，其主要应该满足以下功能：

1.要求有良好的人机界面，操作简单，易于玩家使用和掌握。

2.要求有良好的系统兼容性，可以在不同的平台使用。

3.人工智能思考时间尽量合理，减少玩家等待时间，提高游戏效率。

4.人人对战实时传输延时短，效率高。

5.客户端简洁，不占用过多内存空间。

6.人工智能策略尽量合理，避免无限循环的死棋。

## 2.2 可靠性和可行性需求分析

中国象棋本身就存在着竞技性，因此其有严格的棋盘规则来保证竞技的公平公正性。作为一款中国象棋软件，我们也必须严格按照象棋规则进行设计，其主要设计原则包括以下几条：

1.每个棋子必须按照自身的特性来走棋，例如“马走日”，“象走田”，兵不能后退等规则

2.对战双方在一方走棋过程中，另一方不得有走棋操作，避免发生冲突

3.网络端对战时，走棋的网络传输

4.对战时的悔棋，弃局，计时等操作

5.判断棋局输赢的操作

6.设置每个棋子的权值（重要程度），判断当前棋局局势，人工智能作出相应对策

本软件主要由界面程序和人工智能两大部分组成。其中界面程序包括客户端界面设计，棋盘界面设计。在登录界面，需要考虑人人对战，人机对战，服务端，客户端的不同需求。而在人工智能部分，主要体现在人走棋后计算机会采取合适有效的应对策略，主要包括一步象棋人工智能，两步象棋人工智能，走n步的人工智能，优化-剪枝算法，最小值最大值算法等的设计。

在对基于QT的中国象棋游戏进行功能需求分析，可靠性及可行性进行分析之后，我们就有了合理的设计策略，可以着手开始开发设计本象棋游戏了。

# 第3章 系统设计

所谓设计是将一个实际问题转换成相应的解决办法的主动过程，也就是对一种解决办法的描述。而系统设计目的是告诉用户系统具体将要做什么，系统设计在描述系统时，不采用计算机方面的专业术语，重点在描述系统的功能上，用户完全能够理解。根据上一章对基于QT的中国象棋游戏的系统需求分析，我们可以很容易的将该象棋的功能分为以下几个模块。

## 3.1 功能模块设计

系统的功能模块简要概括了一个系统软件将要实现的功能以及各部分功能之间的联系。基于QT的中国象棋游戏的整体功能模块结构如图3-1所示。



图3-1 功能模块图

## 3.2 系统流程设计

系统流程图作为描绘系统物理模型的传统工具。它的基本思想是系统理念的每个部件均利用图形符号并以黑盒子的形式描绘，以此表示信息在各个部件之间流动的情况，而不是表示对信息进行加工处理的控制过程。基于QT的中国象棋游戏的系统流程图简述了游戏进行的逻辑过程，如图3-2所示。



图3-2 系统流程图

## 3.3 系统数据结构设计

数据结构是[计算机](http://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA" \t "_blank)存储、组织[数据](http://baike.baidu.com/item/%E6%95%B0%E6%8D%AE)的方式。数据结构是指彼此之间存在一种或多种特定关系的[数据元素](http://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%85%83%E7%B4%A0)的集合。一般，想要得到高效的运行或者存储[效率](http://baike.baidu.com/item/%E6%95%88%E7%8E%87)，就必须进行精心设计。中国象棋作为一款棋类游戏，必然牵涉到大量的运算，所以选择一项合理地数据结构非常有必要。

基于QT的中国象棋游戏最主要的数据结构是对于象棋棋盘表示的数据结构设计，它通常使用一个二维数组来表示。这里我们可以用的二维数组表示象棋棋盘。每一个元素代表棋盘上的一个交点，元素的值（如0,1,2等）代表该位置是否有棋子或是何种棋子。这种方式看起来合乎实际情况，但对于计算机来说却产生了大量冗余，因为很多时候棋盘上一些位置是空着的，利用到的次数并不太多。所以，我们更希望选择一种紧凑的数据结构来表示棋盘上的信息。

我们知道象棋棋盘由10横线和9条纵线交错而成，这与数学中的平面直角坐标系十分类似。而象棋对战双方合计有32个棋子，每一个棋子都位于横纵交叉点。那么，所有的棋子信息都可以通过一个32个字节的一维数组来表示。每个棋子的横坐标可以由每个字节的高4位来表示，低4位则用来表示该棋子的纵坐标。已经死亡的棋子则用坐标范围外的数来表示。于是，整个棋盘的信息就被装在了这32个字节的数组当中。

当然，对于棋子来说，只包含横坐标和纵坐标的信息是远远不够的。我们还要知道棋子种类，是否死亡，棋子颜色，棋子编号等信息。基于此，我们可以设计一个名为Stone的类来保存棋子信息。Stone类的成员变量包括以下内容。

int \_row; //棋子横坐标

int \_col; //棋子纵坐标

TYPE \_type; //棋子种类

bool \_dead; //死亡标志

bool \_red; //颜色标志

int \_id; //棋子编号

利用Stone类开辟一个类型为Stone的\_s[32]的数组，以此我们就可以保存棋盘信息了。

## 3.4 走法设计

在对棋盘棋子设计完成后，实际上如果对战双方严格要求自己，也基本能够实现对弈了。但是，我们很难保证对战双方会严格按照既定的中国象棋规则走棋，不论是故意胡乱走棋，还是不小心点击了不合法的走棋位置，这些都是不应该允许的。因此，对棋子的移动加以约束，这就是所谓的走法设计。

对于中国象棋的走法设计，我们首先要明白走法是如何产生的。所谓走法产生是指将某个局面的所有可能的走法罗列出来的那一部分程序，然后通知其他部分下一步该往哪走的模块。对于中国象棋来说，由于其有7种（即车、马、象、士、将、炮、兵）不同的棋子，而每种棋子对应的规则也不尽相同，所以对于走法产生模块的设计是一个麻烦但重要的过程。比如对于“将”来说，它的活动范围被限制在“九宫”内，每次只能以直线的方式移动一个。特别的是，当双方的“将”碰面时，对方可以直接吃掉敌方的“将”。基于每个棋子的走棋规则，我们可以设计以下函数。

canMoveJiang(); //将走法产生

canMoveShi(); //士走法产生

canMoveXiang(); //象走法产生

canMoveChe(); //车走法产生

canMoveMa(); //马走法产生

canMovePao(); //炮走法产生

canMoveBing(); //兵走法产生

而函数的具体实现，我们可以在下一章做进一步了解，在此不做过多敷述。通过对中国象棋棋子的走法设计，我们就可以基本实现中国象棋的对战了。

# 第4章 系统实现

软件系统的设计和实现是软件工程过程中一个重要的阶段，在这一部分我们需要开发出可执行的软件系统。经过前几章的分析和设计，我们已经大致清楚了基于QT的中国象棋游戏的设计思路，现在我们需要着手实现各个模块的具体功能了。

## 4.1 棋盘界面的实现

棋盘界面的实现分为棋盘的绘制和棋子的绘制。棋盘界面的绘制我们需要用到QT的QPainter类，以此来调用一些绘制函数。而QPainter类的调用已包含在头文件#include <QPainter>中。棋盘界面完成图如图4-1所示。

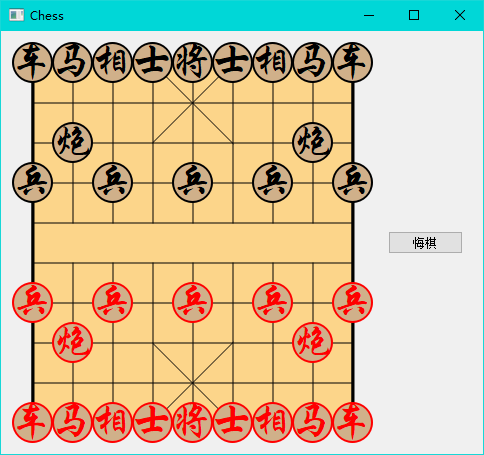


图4-1 棋盘界面

### 4.1.1棋盘的绘制

棋盘总体来说是由9条竖线和10条横线交错而成的盘面，细节部分是“将”和“士”所在的九宫的对角有斜线相连，还有就是“炮”和“兵”所在的位置有一些折角图案。棋盘的绘制程序如下。

//绘制棋盘

void Board::drawPlate(QPainter &p)

{

p.setBrush(QBrush(QColor(252,213,138),Qt::SolidPattern));//设置画刷形式

p.drawRect(\_r,\_r,\_r\*16,\_r\*18);

/\*绘制10条横线\*/

/\*绘制9条竖线\*/

}

//绘制九宫

void Board::drawPlace(QPainter &p)

{

p.setPen(QPen(Qt::black,1,Qt::SolidLine));

p.drawLine(center(0, 3), center(2, 5));

p.drawLine(center(2, 3), center(0, 5));

p.drawLine(center(9, 3), center(7, 5));

p.drawLine(center(7, 3), center(9, 5));

}

上述代码中的函数center()的作用是将某一位置像素化，即将坐标信息转化为像素点位置并返回，而drawLine()就是在两个像素点之间画线。

### 4.1.2 棋子的绘制

棋子的绘制中我们主要用到函数drawEllipse()和函数drawText()。drawElipse()主要作用是绘制一个圆，而drawText()主要作用是在某一位置添加文字。两者结合起来，就可以在某一位置绘制一个棋子。其实现过程如下。

//绘制棋子

void Board::drawStone(QPainter &p, int id)

{

if(isDead(id)) return;

QColor color;

if(red(id)) color = Qt::red;

else color = Qt::black;

p.setPen(QPen(QBrush(color), 2));

if(id == \_selectid) p.setBrush(QColor(0,86,31));

else p.setBrush(QColor(208,176,138));

p.drawEllipse(cell(id));

p.setFont(QFont("华文行楷", \_r\*1.5, 700));

p.drawText(cell(id), name(id), QTextOption(Qt::AlignCenter));

}

## 4.2 走棋与悔棋

### 4.2.1 走法的实现

走法的实现是保证中国象棋能够按照规则走棋的基础，前面的设计中我们已经提到过象棋有7种棋子，所以需要实现7种棋子的走法。而在走棋之前，我们必须先弄清楚要走的是哪个棋子以及该棋子是否有被移动的权利（即棋权），该过程我们称之为走法产生器。我们可以用一个函数sameColor()来检测欲要移动的棋子与将要吃掉的棋子是否为同一种颜色。很明显，相同颜色的棋子是不能相互吃掉的，所以这时候我们要返回错误信息（false）。如果将要走的位置是不同颜色的棋子或该位置为空，我们就要利用switch语句来根据棋子的种类调用不同的走法。实现过程如下所示。

bool Board::canMove(int moveid, int killid, int row, int col)

{

if(sameColor(moveid, killid)) return false;

switch (\_s[moveid].\_type)

{

case Stone::CHE:

return canMoveChe(moveid, killid, row, col);

/\*省略部分代码\*/

}

return false;

}

switch()语句中的每个分支都对应一种棋子的走法。例如函数canMoveJiang()实现了“将”的走法，它的基本思想是：“将”的移动范围被限制在九宫内，这一点我们只要限制其横坐标row和纵坐标col的范围就行了；其次，“将”只能直线移动一步，对于这一限制我们可以通过公式(4-1)来实现。

(4-1)

其中row，col为棋子原位置，rowl，coll为欲移动的位置。很明显，如果棋子上下移动一步，r的值必定为10，如果棋子左右移动一步，r的值必定为1。所以，如果通过计算欲移动的位置返回的r值不为1或10，那么该走法是不合法的；最后，特别的一点是象棋对战时存在双方将帅相见时可以直接吃掉的情况。这是，我们可以将“将”看作是一个“车”，调用“车”的走法来吃掉对方的“将”。具体实现过程如下。

bool Board::canMoveJiang(int moveid, int killid, int row, int col)

{

if(killid != -1 && \_s[killid].\_type == Stone::JIANG)

return canMoveChe(moveid, killid, row, col);

GetRowCol(row1, col1, moveid);

int r = relation(row1, col1, row, col);

if(r != 1 && r != 10) return false;

if(col < 3 || col > 5) return false;

if(isBottomSide(moveid))

{

if(row < 7) return false;

}

else

{

if(row > 2) return false;

}

return true;

}

对于“士”来说，它的走法实现函数为canMoveShi()。实际上“士”和“将”在走法上有许多相似之处。例如它们都被限制在“九宫”的范围内，都只能移动一步。不同的是，“士”走一步只能走斜线，“将”走一步是走直线。对于这一点，我们上面已经说过棋子的走动可以通过公式计算来实现。我们已经知道“将”对应的r为1或10，那么很明显“士”对应的r为11。于是“士”的实现过程如下所示。

bool Board::canMoveShi(int moveid, int, int row, int col)

{

/\*省略部分代码\*/

if(r != 11) return false;

/\*省略部分代码\*/

}

从“将”和“士”的走法实现我们不难看出，不同种类棋子的走法实现其实可以分解为范围限制的实现和走棋合法的判断两个过程。范围限制的实现就是限制棋子能够移动的范围的横坐标row和纵坐标col，而走棋合法的判断只要是计算公式的r值是否对应该棋子的走法。除了上述两个过程外，我们只需额外考虑一些特殊情况（如“蹩马腿”、“塞象眼”等）下的限制即可。显然，只需遵循上述的设计原则，我们可以很快实现另外几种棋子的走法，在此不作过多敷述。

### 4.2.2 悔棋的实现

虽然走法的实现可以规范我们的走棋方式，使其合法合乎规则。但是很多时候即使是按照规则的走棋，也不一定是合乎我们本心的策略，这时候我们就需要悔棋这个功能。为了实现悔棋的功能，我们首先要保存没走一步的局面信息。这里我们创建一个名为Step的类来存储局面局面信息。但我们需要悔棋时，只要提取类Step的对象里保存的数据信息来重绘一下棋盘界面就可以了。实现过程如下。

//保存局面信息

void Board::saveStep(int moveid, int killid, int row, int col, QVector<Step\*>& steps)

{

GetRowCol(row1, col1, moveid);

Step\* step = new Step;

step->\_colFrom = col1;

step->\_colTo = col;

step->\_rowFrom = row1;

step->\_rowTo = row;

step->\_moveid = moveid;

step->\_killid = killid;

steps.append(step);

}

//悔棋

void Board::back(Step \*step)

{

reliveStone(step->\_killid);

moveStone(step->\_moveid, step->\_rowFrom, step->\_colFrom);

}

void Board::backOne()

{

if(this->\_steps.size() == 0) return;

Step\* step = this->\_steps.last();

\_steps.removeLast();

back(step);

update();

delete step;

}

## 4.3 人工智能的实现

上面走法的实现的内容实际上已经体现了部分的人工智能，但走法的实现主要作用是限制规则，保证对弈的公平性，合理性，它贯穿于人人对战、人机对战等各个方面，是对弈的基础。而真正的人工智能体现在人机对战时机器能够采取合理的策略应对人的走棋，而不是看起来很白痴的走棋。在人机对战中，人的优势是能够利用经验采取合理的策略走棋。而机器的优势是计算能力的强大，它可以通过快速高效的计算采取计算范围内最优的策略应对人类。要发挥人工智能的优势，我们需要考虑三个方面的内容，即搜索算法的实现，估值函数的实现，以及局面评估的实现。结合上述三个方面，我们才能开发出合理高效的人工智能程序。

### 4.3.1 搜索算法的实现

假设甲和乙两个人在进行对弈。现在轮到甲走棋，甲可以有40种走法（不考虑优劣）；而对于甲的每一种走法，以也可以有与之对应的若干种走法。然后又轮到甲有若干种走法来应对……如此往复。根据上面的分析，我们可以来构建一颗“博弈树”，将所有的局面罗列出来。在这棵树的根部是开局的局面，根的若干子节点对应的是甲每走一步棋所对应的局面，而这些节点的子节点有连结着乙每走一步棋所对应的若干局面……如此循环往复，直至树的叶节点。此时便是博弈树的结局的局面，对应甲胜或乙胜又或者平局。博弈树是从根部向下递归产生的一颗包含所有可能的走法的搜索树，我们叫它完全搜索树。

可以看出，如果我们能够得到一局对弈的完全搜索树，我们就可以很轻易的预判输赢，进而采取最优的策略。但是，这几乎是不可能的！经统计表明，中国象棋的每一步对弈局面可以有大约20~40种应对策略。如果我们取平均值40种走法计，那么建立一颗（假设双方各走50步）完全搜索树就需要约10160个节点，以目前计算机的处理能力是完全无法胜任的。显然这是不切实际的，我们应该采取更加合理的方法。

通常，我们会采用深度优先搜索（Depth First Search）来解决这个问题。其基本思想是从树的深度遍历树的节点出发，尽可能深的搜索树的分支。当节点v的所在边都己被探寻过，搜索将回溯到发现节点v的那条边的起始节点，直至从源节点可达的所有节点均被访问为止。此时若存在未被访问的节点，则选择其中一个作为源节点，重复以上过程，直至所有节点都被访问为止。之所以采用深度优先搜索的方法，就是利用了深度优先搜索在搜索过程中的任何时候仅仅生成整颗树的一小部分，搜索过的部分被立即删去。显然，这样的算法对内存的要求极低，运算速度上也不低，是一种很合适的算法。

### 4.3.2 估值函数的实现

我们经常听到别人走棋时旁边有人说这是一步好棋或这是一步臭棋，这是对走法的价值的一种评估。作为人工智能，我们当然愿意选择价值对当前局面最优的那一步棋，这就牵扯到估值函数的实现。我们如何评判一步棋的价值呢，通常可以采取一种比较方便的方法，就是为每种棋子赋予一定的价值分。例如车1000分，马499分，炮501分，兵200分，将15000，士100分，象100分。然后我们假设如果执行这一步棋之后，棋盘上黑方棋子分值的总和与红方棋子分值的总和之差，就是走这一步棋的价值。通过选择价值最大的走法，理论上我们就可以得到最优的走棋策略。具体实现过程如下。

int SingleGame::score()

{

static int s[] = {1000, 499, 501, 200, 15000, 100, 100};

int scoreBlack = 0;

int scoreRed = 0;

/\*计算黑方分值\*/

/\*计算红方分值\*/

return scoreBlack - scoreRed;

}

### 4.3.3极大极小值算法和Alpha-Beta算法的实现

上面我们讨论了走法的价值问题，但实际上价值得分最大的一步棋并不一定是最好的一步棋。这是因为，在实际对弈过程中，我们走一步棋不但要获得最大的价值，还要使对方获得最小的价值。这就牵扯到极大极小值算法的思想。假设甲与乙对弈，在博弈树中，甲会选择局面评分最大的一个走法，故此表示甲走棋的节点为极大节点；而乙会选择局面评分最小的一个走法（这里的局面评分都是相对于甲来说），则称乙为极小节点。如果在有限的搜索深度内，甲和乙都会选择对自己最有利的评分，那么我们会就得到最好的走法。如图4-2所示，黑圈表示甲方会选择分值最大的走法，而红圈表示红方相对于黑方会选择分值最小的走法。



图4-2 极大极小值算法

显然，极大极小值算法是一个不错的算法。但是我们忽略了一个重要的事实，那就是如果我们已经确定某个节点是极大值（或极小值），那么这个节点之后的兄弟节点我们就没必要再去搜索了，从而节省了一大部分时间复杂度。这就是Alpha-Beta搜索算法（也称优化剪枝算法）。极小极大搜索基于深度优先的搜索。如图4-3所示，当搜索到第二层的红色节点时，已知其第一个子节点是一个返回值为200的极小节点，那么这个节点得到的值肯定是小于200的。而第二层的第一个蓝色节点的值为700，那么这个节点后面即使都搜索了，也不会超过200，更不会超过700，因此这个节点后面的节点可以忽略，即图4-3中第三层没有数字的节点。这属于Alpha剪枝，相应的也有Beta剪枝。



图4-3 Alpha-Beta算法

Alpha-Beta算法实际上是极大极小值算法的优化，但它们的实现都依赖于深度优先搜索来实现。总和上述几方面的分析，我们可以通过如下代码实现人工智能的功能。

//获取极小值

int SingleGame::getMinScore(int level, int curMin)

{

if(level == 0)

return score();

QVector<Step\*> steps;

getAllPossibleMove(steps);

int minInAllMaxScore = 300000;

while(steps.count())

{

Step\* step = steps.last();

steps.removeLast();

fakeMove(step);

int maxScore = getMaxScore(level-1, minInAllMaxScore);

unfakeMove(step);

delete step;

if(maxScore <= curMin)

{

while(steps.count())

{

Step\* step = steps.last();

steps.removeLast();

delete step;

}

return maxScore;

}

if(maxScore < minInAllMaxScore)

{

minInAllMaxScore = maxScore;

}

}

return minInAllMaxScore;

}

//获取极大值

int SingleGame::getMaxScore(int level, int curMax)

{

/\*与获取极小值类似，省略部分代码\*/

}

//获取搜索深度内最优走法

Step\* SingleGame::getBestMove()

{

Step\* ret = NULL;

QVector<Step\*> steps;

getAllPossibleMove(steps);

int maxInAllMinScore = -300000;

while(steps.count())

{

Step\* step = steps.last();

steps.removeLast();

fakeMove(step);

int minScore = getMinScore(this->\_level-1, maxInAllMinScore);

unfakeMove(step);

if(minScore > maxInAllMinScore)

{

if(ret) delete ret;

ret = step;

maxInAllMinScore = minScore;

}

else

{

delete step;

}

}

return ret;

}

## 4.4 对战的实现

本软件设计了多种的对战方式，包括人机对战，人人对战，网络对战。接下来我们分别讨论它们的实现方式。

### 4.4.1 人机对战

人机对战的实现是本软件最能够体现人工智能的方面，也是本软件设计所围绕的核心，之后的人人对战和网络对战也在一定方面继承了人机对战的功能。对于人机对战来说，对弈双方是人和计算机。人在走法只要符合走法规则，就可以进行下去。而对于计算机来说，我们需要实现的是人工智能对人的走棋这一动作做出反应。首先我们对鼠标的状态做出反应，并判断当前状态下棋权在那一方。这里我们可以创建函数mouseReleaseEvent()检测鼠标状态，如果鼠标已经走棋完毕，就调用函数click(QPoint pt)，它实际上是将点击的像素点位置通过调用其它一系列函数转化为坐标位置信息，然后调用虚函数virtual void click(int id, int row, int col)来调用函数computerMove()实现电脑走棋。具体实现过程如下。

void SingleGame::click(int id, int row, int col)

{

if(\_bRedTurn)

{

Board::click(id, row, col);

if(!\_bRedTurn)

{

QTimer::singleShot(100, this, SLOT(computerMove()));

}

}

}

void SingleGame::computerMove()

{

//将军

if(\_s[4].\_dead || \_s[20].\_dead)

{

Gameover \_over;

\_over.exec();

}

Step\* step = getBestMove();

moveStone(step->\_moveid, step->\_killid, step->\_rowTo, step->\_colTo);

delete step;

update();

}

### 4.4.2 人人对战和网络对战

上面已经说过人人对战和网络对战在一定程度上继承了人机对战的功能，只是把对战双方都换成了由人来操作，免去了计算机的操作步骤。甚至人人对战是棋盘类Board的派生类，我们只要新建一个名为MultiGame的类，在头文件里包含Board.h，使其继承棋盘类Board的功能即可。类似的，网络对战也是Board类的派生，但是它多了网络对战的功能，这里需要说明一下实现方法。

对于网络对战来说，最重要的是要实现网络同步。这一过程不可避免的要发送棋盘信息的报文。这里我们把报文格式的定义如下：

1.初始化占两个字节；

2.鼠标点击信息占四个字节；

3.悔棋信息占一个字节。

网络对战的双方，一方是服务器端，另一方是客户端。服务器端负责监听，其监听端口设置为9899。如果监听到有客户端进入了端口9899，服务器端就可以和客户端建立连接。通过把报文实时传送给对方，并以此更新对战界面，我们就可以实现网络对战了。实现过程如下。

NetGame::NetGame(bool server, QWidget \*parent) : Board(parent)

{

\_server = NULL;

\_socket = NULL;

\_bServer = server;

if(\_bServer)

{

\_server = new QTcpServer(this);

\_server->listen(QHostAddress::Any, 9899);

connect(\_server, SIGNAL(newConnection()), this, SLOT(slotNewConnection()));

}

else

{

\_socket = new QTcpSocket(this);

\_socket->connectToHost(QHostAddress("127.0.0.1"), 9899);

connect(\_socket, SIGNAL(readyRead()), this, SLOT(slotDataArrive()));

}

}

### 4.4.3 多种对战方式的选择

由于本软件采用多种的对战方式，所以必须设计一个选择对战界面。

ChooseDlg::ChooseDlg(QWidget \*parent) : QDialog(parent)

{

QVBoxLayout\* lay = new QVBoxLayout(this);

lay->addWidget(\_buttons[0] = new QPushButton("人机对战"));

/\*省略部分代码\*/

for(int i=0; i<4; ++i)

{

connect(\_buttons[i], SIGNAL(clicked()), this, SLOT(slotClicked()));

}

}

MainWnd::MainWnd(int gameType, QWidget \*parent) : QWidget(parent)

{

\_gameType = gameType;

if(\_gameType == 0)

{

SingleGame\* game = new SingleGame;

CtrlPanel\* panel = new CtrlPanel;

QHBoxLayout\* hLay = new QHBoxLayout(this);

hLay->addWidget(game, 1);

hLay->addWidget(panel);

connect(panel, SIGNAL(sigBack()), game, SLOT(slotBack()));

}

/\*多重判断，省略部分代码\*/

}

## 4.5 判输检测

一局对弈进行下来，总要有分出胜负的时候，因此我们很有必要对输赢的情况进行判断。中国象棋获得胜利的标志只有一个，那就是“将”被吃掉，也就是所谓的“将军”。基于此，我们可以创建一个Gameover类来弹出一个标志着游戏结束的对话框。当我们检测到某一方的“将”（即\_s[4]和\_s[20]）的成员变量（bool \_dead）为真（true）时，表明该“将”已经死亡，可以结束棋局了。该过程的核心代码如下。

Gameover::Gameover(QWidget \*parent) : QDialog(parent)

{

QMessageBox::information(this,"中国象棋","Game Over");

exit(0);

}

void SingleGame::computerMove()

{

//将军

if(\_s[4].\_dead || \_s[20].\_dead)

{

Gameover \_over;

\_over.exec();

}

Step\* step = getBestMove();

moveStone(step->\_moveid, step->\_killid, step->\_rowTo, step->\_colTo);

delete step;

update();

}

# 第5章 系统测试

在完成整个软件的实现之后，我们很有必要对整个系统进行全方位的测试。这是因为测试可以展现我们所预期的系统的功能的样子，同时又可以发现一些在投入使用之前的漏洞。只有系统通过了所有的测试，并对发现的漏洞采取相应的弥补措施，这样才能完成一个合格的可以投入使用的软件。

## 5.1 测试方法

系统的测试方法多种多样，通常我们采用黑盒测试或白盒测试。黑盒测试主要切入点是在软件的宏观表现，我们主要测试所设计的功能是否符合要求，而不关注具体内部实现；白盒测试则恰恰相反，它更加重视系统内部操作是否合乎要求，要对每步的实现进行仔细测试。鉴于白盒测试的测试工作步骤相对复杂，这里我们采用黑盒测试的方法检测系统的完成度。

## 5.2 对弈模式选项模块测试

由于本系统实现了多种对弈模式，需要测试是否能够正确打开相对应的对战模块，打开正确的对战模块能够保证我们在正确的对弈方式下走棋。对弈模式模块包含人机对战、人人对战、网络对战（服务端）、网络对战（客户端）是个选项，如图5-1所示。测试对弈模式选项模块用例分析如表5-1所示。

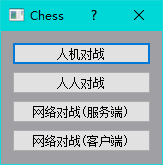


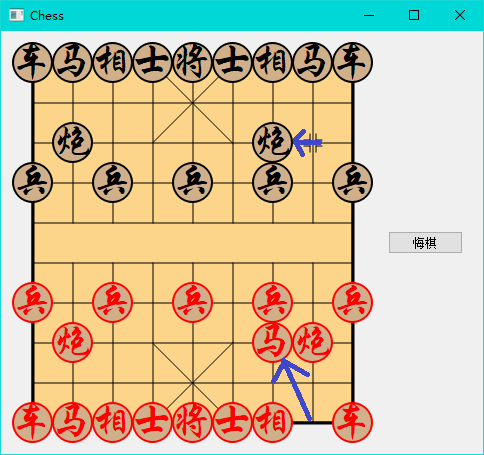
图5-1 对弈模式选择

表5-1 对弈选项测试用例

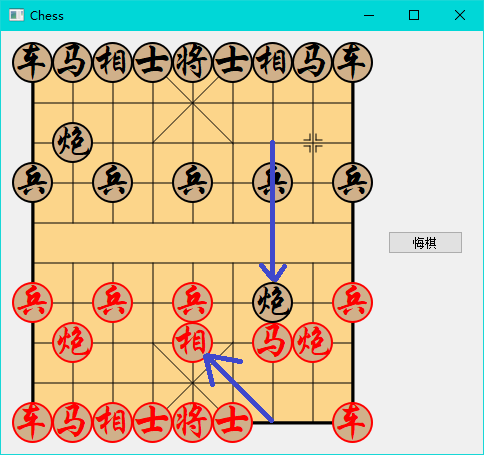
|  |  |  |  |
| --- | --- | --- | --- |
| 测试编号 | 测试目标 | 测试结果 | 错误原因 |
| 1 | 点击“人机对战”打开人机对战界面 | 人机对战界面被打开 | 无 |
| 2 | 点击“人人对战”打开人机对战界面 | 人人对战界面被打开 | 无 |
| 3 | 点击“网络对战（服务端）”打开服务端界面 | 网络对战服务端界面被打开 | 无 |
| 4 | 点击“网络对战（客户端）”打开客户端界面 | 网络对战客户端界面被打开 | 无 |

## 5.3 人工智能模块测试

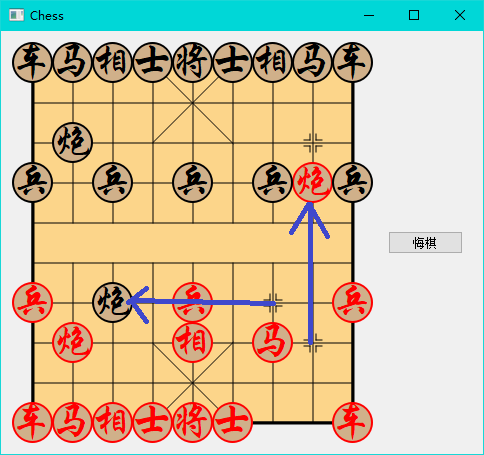
人工智能的测试主要是看在人机对战模式中电脑一方是否能够采取合理的走法。这一模块的测试有很大的主观性，因此如果人工智能的走棋在人看来还算可以，不是很白痴的走棋方式就可以认为人工智能模块是成功的。通过图5-2的几步走棋，我们可以看出，人工智能可以合理的吃掉对手的棋子，在遇到威胁时也会采取相应的措施进行防守，说明人工智能模块的实现是成功的。



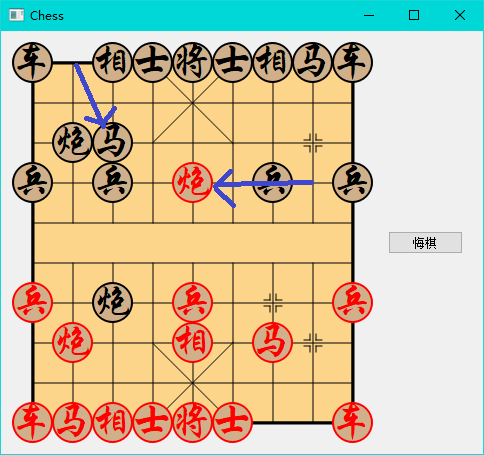
a



b



c



d

图5-2 人工智能测试

## 5.4 走法的测试

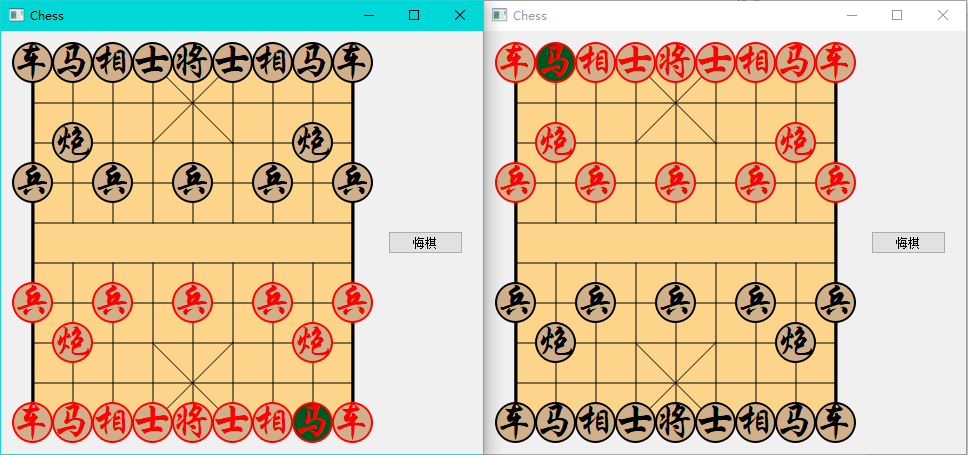
走法的测试旨在期望在走棋过程中对不合法的走法不会产生反应，也就限制了对弈双方不合法的走棋。走法的测试的一些特殊情况主要有“蹩马腿”、“塞象眼”等，具体测试用例如表5-2所示。

表5-2 走法测试用例

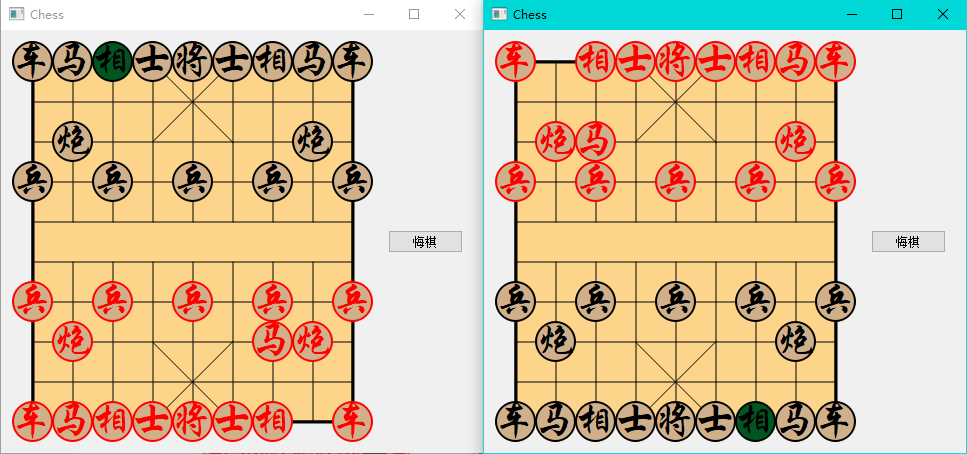
|  |  |  |  |
| --- | --- | --- | --- |
| 测试编号 | 测试内容 | 测试结果 | 错误原因 |
| 1 | “马”被“蹩马腿”后走棋 | 无法走棋 | “马”被蹩腿 |
| 2 | “象”被“塞象眼”后走棋 | 无法走棋 | “象”被塞眼 |
| 3 | “象”过河 | 无法过河走棋 | “象”只能在自己营地走棋 |
| 4 | “士”出“九宫” | 无法出“九宫”走棋 | “士”的活动范围为“九宫“ |
| 5 | “兵”未过河左右走棋 | 无法走棋 | ”兵“在自方营地无法左右移动 |
| 6 | “兵”过河后左右走棋 | 可以左右走棋 | 无 |
| 7 | ”炮“没有炮架吃对方棋子 | 无法吃棋 | “炮”吃棋必须有炮架 |
| 8 | ”炮“有炮架吃对方棋子 | 可以吃棋 | 无 |
| 9 | “车”斜线走棋 | 无法走棋 | “车”只能直线走棋 |
| 10 | “将”吃碰面的“将” | 可以吃棋 | 无 |

## 5.5 网络对战模块测试

网路对战模块测试内容主要是检验服务器端和客户端之间能否实时传送局面信息。这里我们设置服务器端与客户端的通信端口号为“9899”。从图5-3不难看出，当服务器端选中“马”时，客户端的“马”也被标记；当服务器端的“马”走棋后，客户端的“马”也发生了相应的改变；接着在客户端选中“相”，服务器端的“相”也被选中。可以看出，服务器端与客户端之间的通信良好。



a



b

图5-3 网络对战测试

## 5.6 将军测试

在一局对弈结束的时候，游戏需要弹出标志游戏结束的窗口。测试结果如图5-4所示。

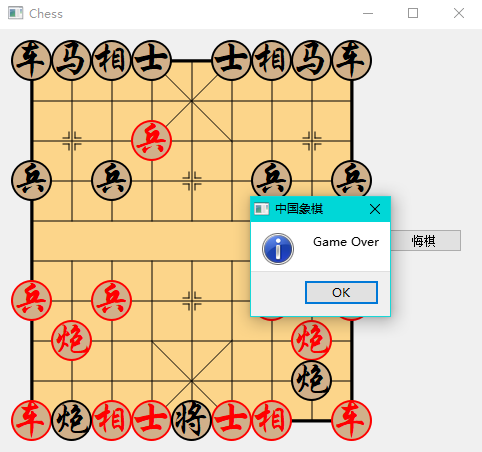


图5-4 将军测试

在对各个模块进行测试后，本软件的所以功能基本都得到了验证。虽然某些部分存在一些小的瑕疵，但是并不影响总体的功能实现，同时也为以后的改进指明了方向。总体来说，本软件的设计是成功的。

# 

# 总 结

最初开始课程设计是在2017年的2月，到现在将课程设计完成已经经过了三个多月了。回想起来，从最开始的迷茫不知道该选择什么样的题目作为毕业设计，到逐渐发掘自己的擅长的方面来确定题目，再到着手一步步实现系统的功能，这中间渗透着无数的汗水与努力，以及老师的谆谆教导和同学们的无私帮助。2017年关于人工智能的新闻层出不穷，从李世石与“阿尔法狗”的人机大战，到即将举办的柯洁与“阿尔法狗”的大战，都激发了我对人工智能的极大兴趣，也是基于此我才决定要做一个关于中国象棋的程序。考虑到人工智能需要极大的运算量，我采用了效率相对更高的C++语言来编写程序。而为了使我所编写的程序有更强的可移植性，我选用了支持跨平台的QT作为开发平台。

通过深入的分析，我逐步抓住了设计一个中国象棋游戏所需的要点。然后，在之后的过程中我根据实际情况不断地修正功能需求，使开发模型日趋完善。由于采用的是基于QT的开发平台，而关于QT方面的知识在本科阶段没有教授相关的课程作为基础，所以我只能从零开始学习。在学习的过程中，我查阅了大量的资料，阅读了一些相关的经典书籍。同时，老师和同学也给我不少学习上的建议。在学习和实践相结合的情况下，我终于学会了关于QT的一些基础的东西。由于我所做的软件的核心还是依靠C++为基础的，所以这些QT的基础知识已经基本能够帮助我完成系统的设计了。虽然我最终顺利完成了系统的设计，但是不可否认的是本软件还是存在一些缺陷和值得改进的地方的。比如界面设计太过简单，也没有加入背景音乐和声效等多媒体文件；人工智能还不够真正的智能，可以进一步提高其运算速度和棋艺；可以添加一些棋谱残局等，是本系统功能更加丰富。我想我会在以后自身技术更加成熟后，逐步完善各项功能，使其成为一个更加成熟的游戏软件。

# 致 谢

时间过得飞快，没想到马上就是要和大学生活说再见的时候了，而我感觉好像昨天才进入大学。而在临近毕业之际，我很感谢学校给了我一次独立完成系统设计的机会，让我本科阶段所学到的知识得到了实践。我很感谢我的指导老师李爱玲老师，她不仅是我的毕业设计老师，在本科阶段也是我的计算机导论和数据结构两门课程的授课老师。所以，李老师不仅在课程设计的过程中给了我许多建设性的指导意见，使我顺利完成毕业设计的任务。在以往的课程学习中，李老师更是我的授业恩师，教会了我许多本专业的知识，传授了我一些有益的人生经验，在此我很感激！当然，对于本科阶段教授过我课程的其他老师，我也由衷的表示感谢！还要感谢的是与我朝夕相处四个年头的同学们。在课程设计的过程中，我也在一些我自身不足以解决的问题上获得了许多同学的帮助。如果不是大家的帮助，我想我会有许多困惑难以解决。所以，在以后的学习和生活中，我会铭记大家给予我的帮助，努力战胜新的挑战。马上就要结束了，虽然很不舍，但是没有办法。所以在此再次对李爱玲老师和同学们表示诚挚的感谢！

# 参考文献

[1]王小春.PC游戏编程（人机编程）[M].重庆：重庆大学出版社，2002.06

[2]《象棋白皮书》课题组.象棋白皮书--关于象棋现状的调查（2016）[R].广东象棋网，2016.6

[3]（英）Mark Summerfield.QT高级编程[M].白建平 等译.北京：电子工业出版社，2011.4

[4]钱能.C++程序设计教程（修订版）：设计思想与实现[M].北京：清华大学出版社，2009.7

[5]（美）Lippman,S.B，（美）Lajoie,J，（美）Moo,B.E.C++ Primer 中文版（第五版）[M].王刚，杨巨峰 译.北京：电子工业出版社，2013.9

[6]严蔚敏，吴伟民.数据结构：C语言版[M].北京：清华大学出版社，2014.11

[7]乔永利.深度优先搜索的地图着色算法[J].测绘科学，2011

[8]韩永亮.棋类游戏程序架构设计[J].硅谷，2012

[9]贾学斌.数据结构中图的深度优先遍历算法与实现[J].中国电子商务，2012

[10] （加）Jasmin Blanchette，（英）Mark Summerfield.C++ GUI Qt4编程(第二版)[M].闫锋欣 等译.电子工业出版社，2008

[11]（韩）金大（臣尔）.Qt5开发实战[M].张红艳 译.人民邮电出版社，2015.9

[12]陆文周.Qt5开发及实例（第2版）[M].电子工业出版社，2015.05

[13]霍亚飞，程梁.Qt5编程入门[M].北京航空航天大学出版社，2015.1

[14]鲍忠贵，王涛，陈凌晖.Qt编程快速入门[M].清华大学出版社，2016.6

[15]霍亚飞.Qt Creator快速入门（第3版）[M].北京航空航天出版社，2017.1

[16]（美）Michael Dawson.C++游戏编程入门（第4版）[M].李军 译.人民邮电出版社，2015.8

[17]（美）Sanjay Madhav.游戏编程算法与技巧[M].刘瀚阳 译.电子工业出版社，2016.10

[18]（美）Mike McShaffry，（美）David “Rez” Graham.游戏编程权威指南（第4版）[M].师蓉，李静，李青翠 译.人民邮电出版社，2016.3

[19]安晓辉.Qt on Android 核心编程[M].电子工业出版社，2015.1

[20]刘汝佳.算法竞赛入门经典（第2版）[M].清华大学出版社，2014.6

[21]熊岳山，祝恩.数据结构与算法[M].清华大学出版社，2013.2