

面向对象程序设计（C++） 课程设计报告

题目：五子棋游戏

专业：人工智能

设计日期：2024 年 4 月 29 日 至 2024 年 6 月 13 日

小组成员人数：1 人

小组成员名单：

组长 班级： 学号： 姓名： 成绩：

报告撰写的内容与要求

1. 项目简介：介绍本次课程设计选题的目的、意义、任务概况、本人在项目中的分工等内容。
2. 项目内容：系统的设计与实现的全面描述，介绍系统整体结构、系统框架图、UML 类图、程序流程图、重点难点分析及解决方案、调试难点及解决方法、系统交互界面及结果展示等。本部分内容应以记叙或白描手法为基调，在完整叙述的基础上，对自己认为有重要意义或需要研究解决的问题进行重点叙述，其它内容则可简述。
3. 总结或体会：对课设效果进行综合评价，着重介绍自身的收获与体会，内容较多时可列出小标题，逐一列举。总结或体会的最后部分，应针对实习中发现的自身不足，简要地提出今后学习，努力的方向。
4. 报告正文一律采用计算机排版、A4 纸**双面**打印，正文字体为**小四号宋体**，**1.35 倍**行距，正文页数不少于 10 页、不多于 20 页(其中代码不超过 3 页，**中文字数**不少于 5000 字)。要求语句通顺、论述严谨、规范、正确。
5. 请注意封面页、扉页、评语页等的打印及装订顺序。分别为①封面页（其背面为扉页“实习报告的内容与要求”）、②目录、③正文、④其他附件（如有）。

目录

一 . 项目简介	1
1.1 项目内容	1
1.2 文件与执行环境	1
1.2.1 电子文档打包文件名及文件列表	1
1.2.2 编译执行环境与运行步骤	2
1.3 任务分工	2
二 . 项目内容	3
2.1 系统的整体架构	3
2.2 UML 类图	4
2.3 系统的整体设计	12
2.4 功能模块设计	12
2.5 重点难点分析及解决方案	15
2.5.1 重难点分析	15
2.5.2 解决办法	15
2.6 调试难点及解决方法	16
2.6.1 调试难点	16
2.6.2 解决方案	16
2.7 结果展示	16
三 . 总结与体会	19

一. 项目简介

1.1 项目内容

项目简介:

利用 C++ 的知识和 QT 以及一些自行拓展的新知识, 实现五子棋游戏。

使自己更加深层的理解和掌握 C++ 并在程序中展现出来, 同时开发编程的思想和能力, 以及扩展知识面, 学习一些课上没有涉及的内容。同时通过阅读一定量他人的代码, 使自己更加熟练的看懂别人的思想及做法从中学习, 提高自己的编程能力。

主要功能:

- (1) 实现本五子棋游戏系统中登录、注册新账号、找回密码、游戏目录、五子棋游戏教程、设置账号信息、“练习对战”模式、“双人对战”模式、“人机对战”模式等等页面。
- (2) 实现五子棋游戏逻辑, 并基于游戏逻辑和游戏理解、动态规划的思想实现了双重优化的 Alpha-Beta 剪枝算法并用于局面评分、人机对战, 其中使用了 Ac 自动机、Zobrist 哈希算法;
- (3) 用文件接口实现了棋谱保存、导入功能, 当玩家未保存当前棋局时直接关闭界面将会提示, 防止误触;
- (4) 不同模式中相应应有“悔棋”功能, 所有模式中均有“重新开始一局”、展示手顺列表、展示指定的前几步的棋局、复盘棋局功能, 其中练习对战模式中可以任意生成棋局“变例”并在表格中、在能量条中展示局面分数, 直观且方便;
- (5) 实现数据库存储用户注册信息, 包括用户名、密码、邮箱和头像 URL 信息, 并使用链表作为临时存储单元;
- (6) 使用 TCP 协议实现注册时发送验证码、找回密码时发送密码等功能, 这两个功能分别继承自基础类 tcp, 实现了继承与多态。

1.2 文件与执行环境

1.2.1 电子文档打包文件名及文件列表

电子文档压缩包内文件列表如下:

GoBang_studio

项目文件:

GoBang_studio.pro

源程序:

acstring.cpp, chessboard.cpp, computerwidget.cpp, duelwidget.cpp, fileio.cpp, forgetpwd.cpp, gobangai.cpp, guidewidget.cpp, inputchecker.cpp, main.cpp, login.cpp, menu.cpp, practicewidget.cpp, settingswidget.cpp, signup.cpp, sqlio.cpp, tcpemail.cpp, userlist.cpp

头文件:

acstring.h, chessboard.h, computerwidget.h, duelwidget.h, fileio.h, forgetpwd.h, gobangai.h, guidewidget.h, inputchecker.h, login.h, menu.h, practicewidget.h, settingswidget.h, signup.h, sqlio.h, tcpemail.h, userlist.h

UI 文件:

computerwidget.ui, duelwidget.ui, forgetpwd.ui, guidewidget.ui, login.ui, menu.ui, practicewidget.ui, settingswidget.ui, signup.ui

资源文件:

ResourceFile.qrc, sounds\press.mp3, pix\gomoku.ico

1.2.2 编译执行环境与运行步骤

编译环境为 Qt 框架下的 Desktop Qt 5.14.1 MinGW 64-bit 编译环境, 运行环境为 Windows10 操作系统, 数据库使用 MySQL8.0.37 版本, 数据库文件为本地 *gobang_studio_db* 数据库下的表 *1_userinfo*。在 Qt 5.14.1 中打开 .pro 文件并按键 **ctrl + R** 即可编译运行, 启动程序。

1.3 任务分工

任务概况:

本次任务艰巨需要在一定的时间内完成 QT 的学习, 并且能应用代码进行编写窗口, 将任务分为一下内容:

1. 实现本五子棋游戏系统中登录、注册新账号、找回密码、游戏目录、五子棋游戏教程、设置账号信息、“练习对战”模式、“双人对战”模式、“人机对战”模式等等页面;
2. 实现五子棋游戏逻辑;

3. 基于游戏逻辑和游戏理解、动态规划的思想实现了双重优化的 Alpha-Beta 剪枝算法并用于局面评分、人机对战，其中使用了 Ac 自动机、Zobrist 哈希算法；
4. 用文件接口实现了棋谱保存、导入功能，当玩家未保存当前棋局时直接关闭界面将会提示，防止误触；不同模式中相应“悔棋”功能，所有模式中均有“重新开始一局”、展示手顺列表、展示指定的前几步的棋局、复盘棋局功能，其中练习对战模式中可以任意生成棋局“变例”并在表格中、在能量条中展示局面分数，直观且方便；
5. 实现数据库存储用户注册信息，包括用户名、密码、邮箱和头像 URL 信息，并使用链表作为临时存储单元；
6. 使用 TCP 协议实现注册时发送验证码、找回密码时发送密码等功能，这两个功能分别继承自基础类 tcp，实现了继承与多态。

任务分工：

项目的任务分工如表一所示。

表一 任务分工表

姓名	完成内容
***	实现本五子棋游戏系统中登录、注册新账号、找回密码、游戏目录、五子棋游戏教程、设置账号信息、“练习对战”模式、“双人对战”模式、“人机对战”模式等等页面。本系统页面友好，操作流畅，实现五子棋游戏逻辑，并基于游戏逻辑和游戏理解、动态规划的思想实现了双重优化的 Alpha-Beta 剪枝算法并用于局面评分、人机对战，其中使用了 Ac 自动机、Zobrist 哈希算法；用文件接口实现了棋谱保存、导入功能，当玩家未保存当前棋局时直接关闭界面将会提示，防止误触；不同模式中相应“悔棋”功能，所有模式中均有“重新开始一局”、展示手顺列表、展示指定的前几步的棋局、复盘棋局功能，其中练习对战模式中可以任意生成棋局“变例”并在表格中、在能量条中展示局面分数，直观且方便；实现数据库存储用户注册信息，包括用户名、密码、邮箱和头像 URL 信息，并使用链表作为临时存储单元；使用 TCP 协议实现注册时发送验证码、找回密码时发送密码等功能，这两个功能分别继承自基础类 tcp，实现了继承与多态。

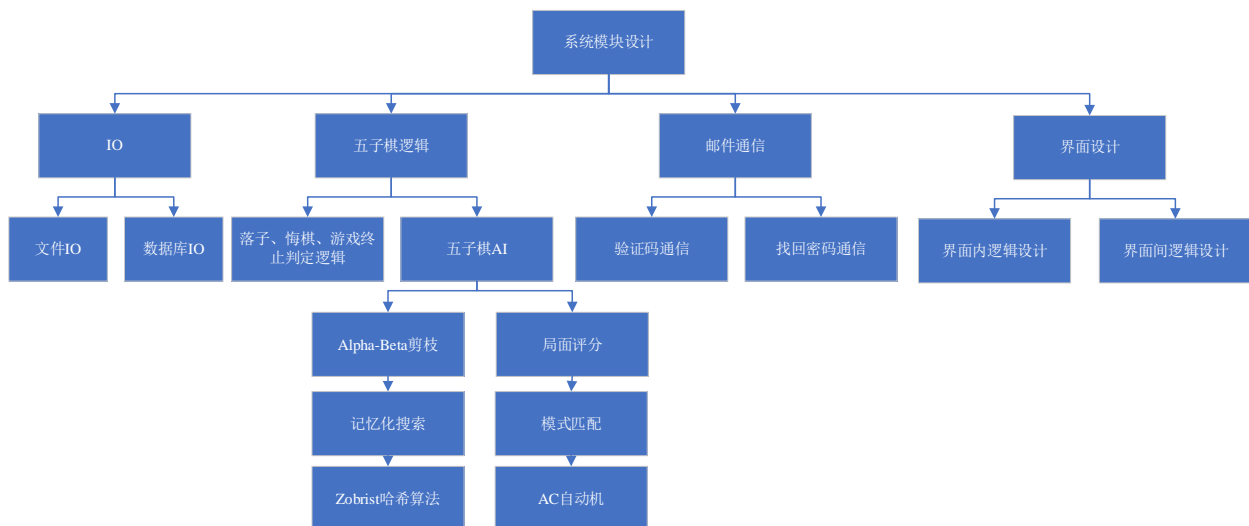
二. 项目内容

2.1 系统的整体架构

一个游戏的设计与实现，首先一定要完成最基本的框架，最基础的底层设施，上层建筑才能更加方便快捷的建筑；首先要确定我们需要的最基本的功能，包括几个界面，功能大概需要怎么实现。不仅要包括游戏本身的界面，还要有登陆和欢迎的界面以及规则讲解界面。本系统由欢迎与登录界面登入后选择开始游戏。

1. 在欢迎与登录界面需要实现登录、用户注册、找回密码等按钮，点击相应的按钮可触发相应的游戏菜单界面、注册界面、找回密码界面的显示。
2. 本系统涵盖四个主要模块，包括 IO 设计、五子棋逻辑设计（含五子棋 AI 设计）、邮件通信设计、界面设计四个主要部分，如图一所示。
3. 登录界面中，当使用者打开登录界面后需要输入用户名和密码，首次登录需要先注册。点击“注册新账号”进入注册界面，点击“忘记密码”进入忘记密码界面。登录成功后进入主菜单界面。
4. 在主菜单界面中，点击“登出”将返回登陆页面，点击“退出”将退出程序，点击“教程”将显示新手教程页面，点击“设置”将显示设置界面。其中，设置页面可重新设置头像、密码。点击相应的游戏模式按钮将进入相应的游戏模式主界面，包括练习对战模式、双人对战模式、人机对战模式。
5. 在游戏主界面中，“保存棋谱”、“导入棋谱”用于棋谱文件操作，“重新开始”用于重新开始一个棋局，右下角“超级前进”、“前进”、“播放”、“后退”、“超级后退”按钮用于查看过去的手顺，在表格中会相应进行聚焦。练习模式中将展示手顺的机器评分，双人模式、人机对战模式将显示悔棋按钮。其中，如果有未保存的操作，将会弹窗提示保存棋谱文件，十分人性化。

系统模块设计如图一所示：



图一 系统整体结构图

2.2 UML 类图

下面是程序中类的 UML 类图，包含类之间的继承与多态关系。

表二 UML 类图

AcString
- maxState: int - nodes: vector<ACNode> - pattern: vector<string>
+ AcString(patterns: vector<Pattern>) + ~AcString() + ACSearch(text: string): vector<int> + LoadPattern(pattern: vector<string>): void + BuildGotoTable(): void + BuildFailTable(): void - AddState(parent: int, ch: char): void
chessboard
- cb: int[15][15] - isOver: bool - isBlackToMove: bool - winner: int - moveCnt: int - moveList: vector<PSS> - rating: vector<double>
+ chessboard(movelist: vector<PSS>) + moveCount(): int + getIthMove(i: int): PSS + isValidPosition(row: int, col: int): bool + isReadyToAddMove(row: int, col: int): int + isReadyToAddMoveAfterIthMove(row: int, col: int, i: int): int + addMove(row: int, col: int): int + delMove(): int + getIthSlice(i: int): chessboard* + pieceSequentialNeighbours(row: int, col: int, player: int, direction: int): int + pieceGeneratesFiveInARow(row: int, col: int, player: int): bool + gameIsOver(): int + gameWinner(): int + evaluate(): int + alphaBeta(depth: int, alpha: int, beta: int, maximizingPlayer: bool): int + getAI(depthcfg: int = 4): GoBangAI::point*
computerWidget
- ui: Ui::computerWidget* - menuPtr: menu*

<ul style="list-style-type: none"> - click_position_row: int - click_position_col: int - is_to_click: bool - cb: chessboard* - show_piece_num: int - depth: int - timer: QTimer* - need_to_save: bool - closing: bool - lastMoveIsFromHuman: bool
<ul style="list-style-type: none"> + computerWidget(menuptr: menu*, parent: QWidget* = nullptr) + ~computerWidget() + on_returnBtn_clicked(): void # paintEvent(event: QPaintEvent*): void # mouseMoveEvent(event: QMouseEvent*): void # mousePressEvent(event: QMouseEvent*): void # closeEvent(event: QCloseEvent*): void + on_openBtn_clicked(): void + on_saveBtn_clicked(): void + on_tableWidget_cellClicked(row: int, column: int): void + on_forwardBtn_clicked(): void + on_superForwardBtn_clicked(): void + on_backwardBtn_clicked(): void + on_superBackwardBtn_clicked(): void + on_playBtn_clicked(): void + on_regretBtn_clicked(): void + on_startBtn_clicked(): void

duelWidget
<ul style="list-style-type: none"> - ui: Ui::duelWidget* - menuptr: menu* - click_position_row: int - click_position_col: int - is_to_click: bool - cb: chessboard* - show_piece_num: int - timer: QTimer* - need_to_save: bool - closing: bool
<ul style="list-style-type: none"> + duelWidget(menuptr: menu*, parent: QWidget* = nullptr) + ~duelWidget() + on_returnBtn_clicked(): void # paintEvent(event: QPaintEvent*): void

```
# mouseMoveEvent(event: QMouseEvent*): void
# mousePressEvent(event: QMouseEvent*): void
# closeEvent(event: QCloseEvent*): void
+ on_openBtn_clicked(): void
+ on_saveBtn_clicked(): void
+ on_tableWidget_cellClicked(row: int, column: int): void
+ on_forwardBtn_clicked(): void
+ on_superForwardBtn_clicked(): void
+ on_backwardBtn_clicked(): void
+ on_superBackwardBtn_clicked(): void
+ on_playBtn_clicked(): void
+ on_startBtn_clicked(): void
+ on_regretBtn_clicked(): void
```

fileIO

```
+ fileIO()
+ readChessManual(fileName: QString, movelist: vector<pair<short, short>>&): void
+ getChessManual(cb: chessboard*): QString
```

forgetpwd

```
- ui: Ui::forgetpwd*
+ forgetpwd(parent: QWidget* = nullptr)
+ ~forgetpwd()
+ on_backButton_clicked(): void
+ on_submitButton_clicked(): void
```

GoBangAI

```
struct point
- isBlack: bool
- x: int
- y: int
- score: int
+ point()
+ point(isBlack: bool, x: int, y: int)
+ point(isBlack: bool, x: int, y: int, score: int)
- aistep: point
+ GoBangAI(state: vector<point>&, nextisBlack: bool, depthmax: int)
+ ~GoBangAI()
+ getStep(): point
- alphaBetaAlgorithm(state: vector<point>&, nowisBlack: bool, depth: int, alpha: int, beta: int,
```

maxdepth: int): point - evaluateSituation(state: vector<point>&, nextisBlack: bool): int - getScore(isBlack: bool, x: int, y: int, g: int[][N]): int - getScoreInDir(isBlack: bool, x: int, y: int, state: vector<point>, dir: int): int - isValidPosition(x: int, y: int): bool - getPossibleNextStep(state: vector<point>&, nowisBlack: bool): stack<point> + static getZobristKey(state: vector<point>): ULL

guideWidget

- ui: Ui::guideWidget*
+ guideWidget(parent: QWidget* = nullptr)
+ ~guideWidget()
+ on_returnBtn_clicked(): void

inputchecker

+ inputchecker() + check_login_loginBtn(username: QString, pwd: QString): int + check_signup_codeBtn(email: QString): int + check_signup_createBtn(username: QString, pwd: QString, repwd: QString, email: QString): int + check_forgetpwd_submitButton(username: QString, email: QString): int + check_settingsWidget_modifyBtn(newPwd: QString): int - inDigit_Alpha(s: QString): int - inEmail(s: QString): int

login

- ui: Ui::login*
+ login(parent: QWidget* = nullptr)
+ ~login()
+ on_loginBtn_clicked(): void
+ on_signupBtn_clicked(): void
+ on_findpwdBtn_clicked(): void
+ on_showpwdBox_stateChanged(arg1: int): void

menu

- ui: Ui::menu*
- current_user: userlist::usernode*
+ menu(parent: QWidget* = nullptr)
+ ~menu()

+ set_current_user(current_user: userlist::usernode*): void + on_guideBtn_clicked(): void + on_settingsBtn_clicked(): void + on_practiceBtn_clicked(): void + on_duelBtn_clicked(): void + on_computerBtn_clicked(): void + on_quitBtn_clicked(): void + on_reloginBtn_clicked(): void

practiceWidget

- ui: Ui::practiceWidget* - menuptr: menu* - click_position_row: int - click_position_col: int - is_to_click: bool - cb: chessboard* - show_piece_num: int - depth: int - timer: QTimer* - need_to_save: bool - closing: bool

+ practiceWidget(menuptr: menu*, parent: QWidget* = nullptr) + ~practiceWidget() # paintEvent(event: QPaintEvent*): void # mouseMoveEvent(event: QMouseEvent*): void # mousePressEvent(event: QMouseEvent*): void # closeEvent(event: QCloseEvent*): void + on_openBtn_clicked(): void + on_saveBtn_clicked(): void + on_tableWidget_cellClicked(row: int, column: int): void + on_forwardBtn_clicked(): void + on_superForwardBtn_clicked(): void + on_backwardBtn_clicked(): void + on_superBackwardBtn_clicked(): void + on_playBtn_clicked(): void + on_startBtn_clicked(): void + on_returnBtn_clicked(): void
--

settingsWidget

- ui: Ui::settingsWidget* - current_user: userlist::usernode* - menu_ptr: menu*

+ settingsWidget(current_user: userlist::usernode*, menu_ptr: menu*, parent: QWidget* = nullptr) + ~settingsWidget() + on_returnBtn_clicked(): void + on_showPwdCheckBox_stateChanged(arg1: int): void + on_pickAvatarBtn_clicked(): void + on_modifyBtn_clicked(): void

signUp
- ui: Ui::signUp* - identificationCode: QString - targetEmail: QString - avatar_url: QString
+ signUp(parent: QWidget* = nullptr) + ~signUp() + on_createBtn_clicked(): void + on_returnBtn_clicked(): void + on_codeBtn_clicked(): void + on_pickAvatarBtn_clicked(): void + on_showPwdCheckBox_stateChanged(arg1: int): void

sqlIO
+ sqlIO() + init_sql(): bool + get_userlist_sql(): userlist& + get_usernode_by_username_sql(username: QString): userlist::usernode* + get_usernode_by_email_sql(email: QString): userlist::usernode* + add_user_sql(username: QString, pwd: QString, email: QString, avatar_url: QString): bool + modify_user_sql(new_info: userlist::usernode*): bool

tcp
- tcpSocket: QTcpSocket* - email: QString - title: QString - content: QString - annotation: QString - senderEmail: QString - senderKey: QString
+ tcp(email: QString) + ReadFromHost(ExpectedReplyFromHost: QString): int

```

+ CheckConnectState(): void
+ SendHeloSis(): void
+ SendAuthLogin(): void
+ SendMyEmailAddress(): void
+ SendAuthCode(): void
+ SendEmailFrom(): void
+ SendRcptTo(): void
+ StartSendContent(): void
+ SendContent(): void
+ SendingStateConfirm(): void
# setTitle() = 0: void
# setContent() = 0: void
# setAnnotation() = 0: void

```

tcp_code : tcp

```

- code: QString
+ tcp_code(code: QString, email: QString)
+ setTitle(): void
+ setContent(): void
+ setAnnotation(): void

```

tcp_pwd : tcp

```

- pwd: QString
+ tcp_pwd(pwd: QString, email: QString)
+ setTitle(): void
+ setContent(): void
+ setAnnotation(): void

```

userlist

```

struct usernode
- username: QString
- pwd: QString
- email: QString
- avatar_url: QString
- next: usernode*
+ usernode()
+ usernode(username: QString, pwd: QString, email: QString, avatar_url: QString, next:
usernode* = nullptr)
- head: usernode*
- len: int

```

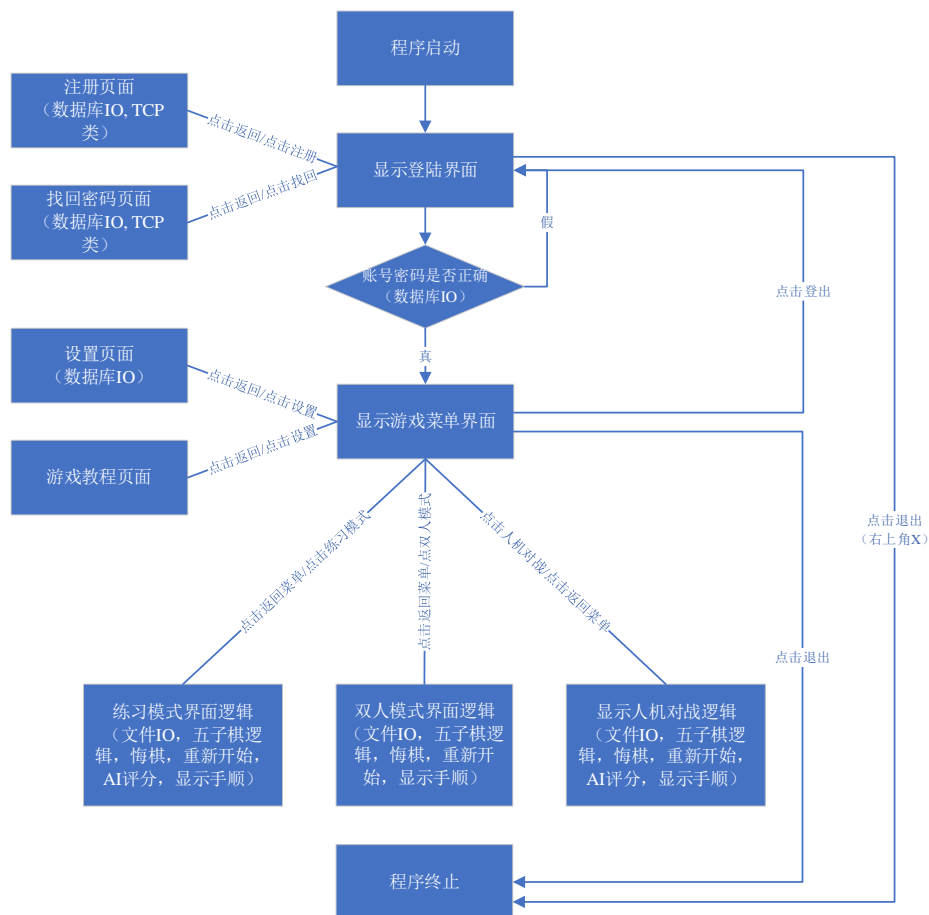
```

+ userlist()
+ ~userlist()
+ getLen(): int
+ getlthNode(i: int): usernode*
+ push_front(node: usernode*): void
+ push_front(username: QString, pwd: QString, email: QString, avatar_url: QString): void
+ push_back(node: usernode*): void
+ push_back(username: QString, pwd: QString, email: QString, avatar_url: QString): void
+ pop_front(): usernode*
+ pop_back(): usernode*

```

2.3 系统的整体设计

系统的整体程序流程图如图二所示。



图二 系统整体程序流程图

2.4 功能模块设计

系统中使用的主要方法列表如表三所示，由于类的方法的设计繁多，故每个类挑选较

为关键的方法作为示例进行说明。

表三 数据表

Class	Method	Function
AcString	ACSearch(text: string): vector<int>	在给定的文本中搜索模式字符串，返回所有匹配的结果位置。
AcString	BuildGotoTable(): void	构建 AC 自动机的转移表。
chessboard	addMove(row: int, col: int): int	在棋盘的指定位置添加一个棋子。
chessboard	evaluate(): int	评价当前棋局的状态。
computerWidget	paintEvent(event: QPaintEvent*): void	重绘窗口。
computerWidget	mousePressEvent(event: QMouseEvent*): void	处理鼠标点击事件。
duelWidget	on_saveBtn_clicked(): void	处理保存按钮的点击事件。
duelWidget	on_returnBtn_clicked(): void	处理返回按钮的点击事件。
fileIO	readChessManual(fileName: QString, movelist: vector<pair<short, short>>&): void	从文件中读取棋谱。
fileIO	getChessManual(cb: chessboard*): QString	获取当前棋盘的棋谱。
forgetpwd	on_submitButton_clicked(): void	处理提交按钮的点击事件。
forgetpwd	on_backButton_clicked(): void	处理返回按钮的点击事件。
GoBangAI	getStep(): point	获取 AI 计算得到的最佳下一步。
GoBangAI	alphaBetaAlgorithm(state: vector<point>&, nowisBlack: bool, depth: int, alpha: int, beta: int, maxdepth: int): point	使用 Alpha-Beta 剪枝算法计算最佳下一步。

guideWidget	on_returnBtn_clicked(): void	处理返回按钮的点击事件。
guideWidget	paintEvent(event: QPaintEvent*): void	重绘窗口。
inputchecker	check_login_loginBtn(username: QString, pwd: QString): int	检查登录界面的输入信息。
inputchecker	check_signup_createBtn(username: QString, pwd: QString, repwd: QString, email: QString): int	检查注册界面的输入信息。
login	on_loginBtn_clicked(): void	处理登录按钮的点击事件。
login	on_signupBtn_clicked(): void	处理注册按钮的点击事件。
menu	on_guideBtn_clicked(): void	处理指南按钮的点击事件。
menu	on_settingsBtn_clicked(): void	处理设置按钮的点击事件。
practiceWidget	paintEvent(event: QPaintEvent*): void	重绘窗口。
practiceWidget	on_startBtn_clicked(): void	处理开始按钮的点击事件。
settingsWidget	on_modifyBtn_clicked(): void	处理修改按钮的点击事件。
settingsWidget	on_pickAvatarBtn_clicked(): void	处理选择头像按钮的点击事件。
signUp	on_createBtn_clicked(): void	处理创建按钮的点击事件。
signUp	on_codeBtn_clicked(): void	处理发送验证码按钮的点击事件。
sqlIO	get_userlist_sql(): userlist&	获取用户列表。
sqlIO	add_user_sql(username: QString, pwd: QString, email: QString, avatar_url: QString): bool	添加新用户。
tcp	SendEmailFrom(): void	发送发件人信息。
tcp	SendRcptTo(): void	发送收件人信息。

userlist	push_back(node: usernode*): void	将用户节点添加到链表末尾。
userlist	getIthNode(i: int): usernode*	获取链表中的第 i 个节点。

2.5 重点难点分析及解决方案

2.5.1 重难点分析

1. 如何实现 GoBangAI 类中对 Alpha-Beta 剪枝评估算法的优化；
2. 如何实现由鼠标点击产生下棋坐标。

2.5.2 解决办法

1. 如何实现 GoBangAI 类中对 Alpha-Beta 剪枝评估算法的优化：

原始 Alpha-Beta 剪枝评估算法的缺陷：1) 当下一层中目标局面节点在搜索顺序的最后时，Alpha-Beta 剪枝评估算法退化为不剪枝的情况；2) 局面节点搜索树中，从根节点走不同的路径可能出现相同的局面，例如，现有四个手顺 A, B, C, D，假设 A, B, C, D 是合法的手顺，而且 A, C 与 B, D 来自两个棋手，那么路径(C, B, A, D), (A, B, C, D), (C, D, A, B), (A, D, C, B)均产生相同的局面；3) 局面评估函数的实现依赖游戏理解和具体实现，需要根据不同的棋类游戏的研究来进行实现。

针对问题一，本人用队列存放待搜索的儿子节点，并预先对他们进行评分并根据评分进行排序取前 10 个手顺。这样做是根据预设贪心偏见“下一步走得好，之后的手顺相对来说肯定更好”。

针对问题二，本人用 Zobrist 哈希方法，15*15 棋盘上每个格点均生成不同的随机数，把过往每个手顺均使用^运算共同操作得到一个 unsigned int 类型的整数作为当前棋局的一个 key，存储到一个 unordered_map 中并同时存储该棋局对应的评分，下回查找一个棋局的评分就可以先看看是不是有存储过，没存储过就看上一步有没有存储过，大大提升了运行效率。

针对问题三，依据前人的研究和本人的实践，“活四”、“双活三”等等通常预示着即将到来的胜利，而如果用一般遍历棋盘或者棋子的方法通常会导致重复计算同一个形状的特征，或者代码逻辑过于纷繁复杂。本人通过联想到 AC 自动机多模式匹配，把每条直线、斜线中连着的棋子的特征转化为一个字符串，并计算该字符串中对应的模式串

的匹配次数和相应评分，得到黑方和白方的评分后进行运算即可得到相应的局面评分。通过不断实践和尝试，本人的评分函数在多数情况下是有效的。

2. 如何实现由鼠标点击产生下棋坐标：

为想要实现点下鼠标就能绘画出棋子这样的功能，需要包含鼠标事件头文件，并且重写界面父类 `QWidget` 的 `mousePressEvent(QMouseEvent *event)` 函数，并且每当鼠标点击时，都通过调用 `update()` 函数来调用重写的 `paintEvent(QPaintEvent *event)` 函数（该函数用于绘制界面上面的图案）。当点击鼠标时，函数体内自定义的变量就会直接利用 `event` 获得 `Qpoint`；根据 `Qpoint` 获得鼠标位置，依据预设好的界面像素坐标和棋盘位置之间的转换关系，存入到私有变量 `click_position_row` 和 `click_position_col`，等待后续传入 `chessboard*` 实例中进行进一步的逻辑判断。

2.6 调试难点及解决方法

2.6.1 调试难点

本次 C++ 课程设计所用的是 Qt Company 开发的跨平台 C++ 图形用户界面应用程序开发框架。由于 `printf` 函数首先将待输出的部分保存在缓冲区而不会伴随运行时指令立即输出，所以无法立即获得需要打印的信息，而是在程序终止运行时一并进行打印，不利于对程序运行时过程中的分析。

2.6.2 解决方案

方案一：使用头文件 `<QDebug>` 中的 `QDebug()` 宏进行实时信息的打印，并在打印的信息中包含打印该信息代码所在的文件与方法中；

方案二：在类 C 语言程序的编写中，空指针、野指针通常是导致程序崩溃的根源，而断点、Qt 自带的 Debug 程序会将程序崩溃的节点反应出来，从而加快分析程序运行错误的根源所在。

2.7 结果展示

登录界面如图三所示。当使用者打开登录界面后需要输入用户名和密码，首次登录需要先注册。点击“注册新账号”进入注册界面，点击“忘记密码”进入忘记密码界面。登录成功后进入主菜单界面。

login

请登录

用户名 注册新账号

密码 忘记密码

☐ 显示密码

登录

图三 登陆界面

Form

注册新账号

用户名 (头像)

密码

重复密码 选择头像

☐ 显示密码

邮箱 获得验证码

验证码

注册

返回

图四 注册界面

Form

忘记密码

用户名

邮箱


向您的邮箱发送密码

返回

图五 忘记密码界面

在主菜单界面中，点击“登出”将返回登陆页面，点击“退出”将退出程序，点击“教程”将显示新手教程页面，点击“设置”将显示设置界面。其中，设置页面可重新设置头像、密码。点击相应的游戏模式按钮将进入相应的游戏模式主界面，包括练习对战模式、双人对战模式、人机对战模式。

Form

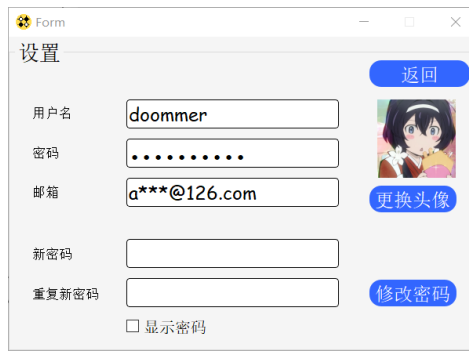
 doommer

登出 退出

教程 设置

练习对战 双人对战 人机对战

图六 主菜单界面

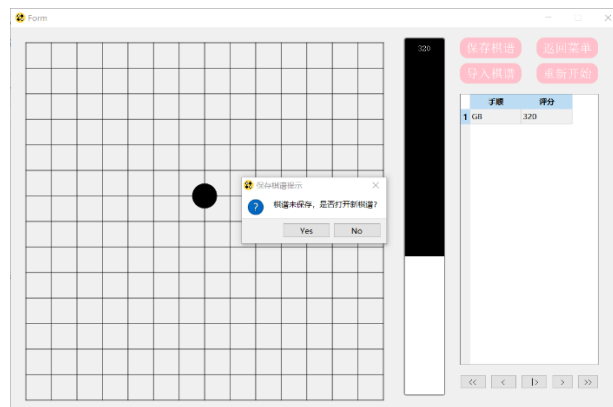


图七 设置界面

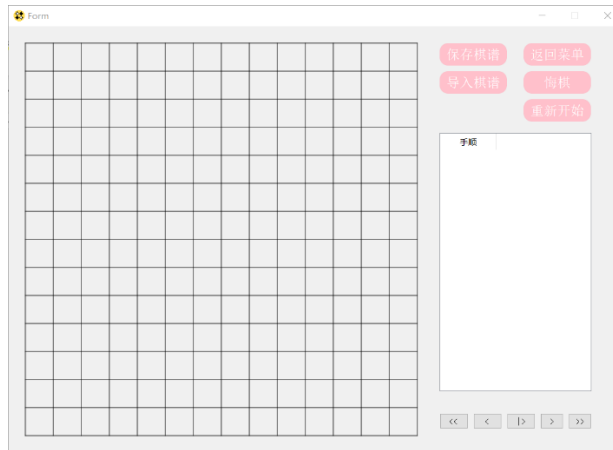


图八 新手教程界面

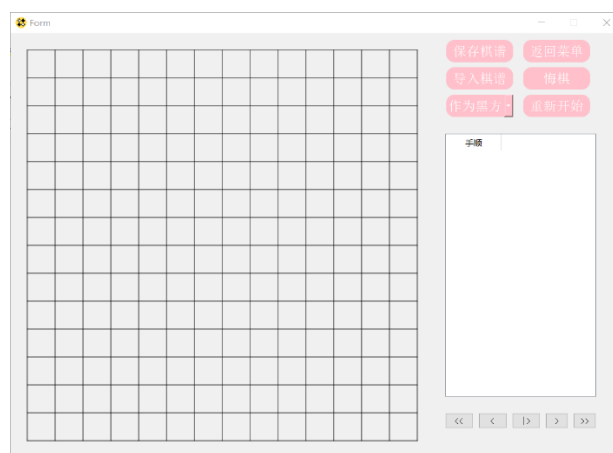
在游戏主界面中，“保存棋谱”、“导入棋谱”用于棋谱文件操作，“重新开始”用于重新开始一个棋局，右下角“超级前进”、“前进”、“播放”、“后退”、“超级后退”按钮用于查看过去的手顺，在表格中会相应进行聚焦。练习模式中将展示手顺的机器评分，双人模式、人机对战模式将显示悔棋按钮。其中，如果有未保存的操作，将会弹窗提示保存棋谱文件，十分人性化。



图九 练习模式界面（落子后点击导入棋谱）



图十 双人对战模式界面



图十一 人机对战模式界面

三. 总结与体会

收获和体会：

本人早就听闻过深蓝计算机（deep blue）的大名，它曾经打败了国际象棋特级大师、当时的国际象棋世界冠军卡斯帕罗夫。作为棋类运动爱好者，虽然十分爱好中国象棋、国际象棋、日本将棋，但自小到大没有研究过五子棋的历史和理论。此次课程设计本人在互联网上初步学习了五子棋相关理论，并且学习了 Qt 中绘制棋盘、棋子、鼠标点击逻辑进一步实现五子棋游戏和五子棋游戏评分算法，并通过数据结构的知识进一步优化五子棋游戏评分算法的实现。从一次次的调试，到最终程序的正常运行，本人也逐步感受到从实践想法的乐趣。

在这次课程设计中，本人首先对系统的整体功能进行了构思，将整个系统的功能与界面划分为几个模块，再根据每个模块的功能编写代码。本人在程序编写的过程中，尽可能地分析所有情况来让程序的逻辑性更强。

不足之处：

本程序中五子棋评分算法中的局面评分函数虽几经调试，但是在某些极端情况下并不能很好地反映当前棋局的战况。本人认为这是因为模式匹配分数仍需要一定的精细化调整以及进一步的优化，当前的评分准则是“欠拟合”的。

本人在队友的选择上存在一定的疏漏，为我更为谷老师增添了一定的麻烦，在此处本人想对谷老师表达我的歉意。

改进与建议：

在本人有解决不了的问题时，本人会主动查阅相关的资料，或向学长学姐、老师们询问，在这个过程中，本人收获了很多的知识。在编写 GoBangAI 类的过程中，最开始我并不知道如何下手。因此，我将类的逻辑以流程图的形式在纸上画了下来，并翻阅教科书，将可能用到的思想和代码在纸上罗列。本程序中五子棋评分算法中的局面评分函数仍然需要进一步的改进和精细化调整。本人在未来的人生路上应当多磨练同他人合作的能力，这不仅仅是针对本科学习阶段，而且包含未来的工作生活阶段，这仍然是本人应当加强的能力之一。

致谢

最后的最后，要感谢此次为我们学生辛苦付出的谷老师。您认真地传授我们课程知识，安排了这样一个课程设计的机会对我们课上学习的知识进行锻炼，并且耐心地验收我们的课设，并提出改进意见，使本人对 C++ 课程知识的了解也更加深入，对语言的学习能力也得到锻炼，掌握了学习新知识的方法，并且也培养了本人的团队组织协调能力，谢谢谷老师的培养，谷老师辛苦了！

参考资料：

郑莉.《C++ 语言程序设计》清华大学出版社.2015

霍亚飞《Qt Creator 快速入门》北京航空航天大学出版社.2017

黑马程序员 《Qt 从入门到实战》bilibili 网站.2019