目 录

1. 引言 ................................................................ 2
2. 需求分析，程序界面与控件设计......................................... 3
3. 黑白棋的规则与实现................................................... 5
   1. 棋子与棋盘......................................................... 5
   2. 落子和判定胜负的规则与实现......................................... 5
4. 图形化界面绘制....................................................... 10
   1. 棋子.............................................................. 10
   2. 棋盘.............................................................. 11
5. 消息映射与处理....................................................... 12

5.6 ON\_WM\_LBUTTONUP()核心函数......................................... 16

1. 项目运行效果展示..................................................... 17

结论

致谢

参考文献

**与基于强化学习的AI对战的黑白棋（Reversi）小游戏**

**摘要：**随着时代的发展，人工智能在我们的生活中的运用场景日益广泛，同时随着人们娱乐需求的增加，各种新形式电子游戏也逐渐兴起。初学经典棋类游戏，如象棋，围棋，黑白棋的玩家，常常苦于在新手阶段没有合适的对手去进行对弈，使得人工智能在这类游戏领域内有了新的的用武之地。学期末我们结合在高级程序设计课程中学习的基于MFC框架进行面向对象的windows应用开发的知识制作了游戏的界面，并使用了DQN算法（Q-learning 算法和神经网络的结合）来设计与训练AI，制作了做了一个与基于强化学习的AI对战的黑白棋小游戏。该程序能让玩家与AI进行对抗，磨炼新手的技术。

**关键词：**强化学习;DQN算法;MFC框架;黑白棋;Windows应用开发;

**Reversi games against AI based on Reinforcement Learning**

**Abstract:** With the development of present era, artificial intelligence is more and more widely used in our life. At the same time, with the increase of people's entertainment demand, various new forms of electronic games are also gradually emerging. Players who are beginning to learn classic chess games, such as chess, go or reversi, often suffer from having no suitable opponent to play chess at the novice stage, which makes artificial intelligence have its own place in the field of such games. At the end of the semester, we combined the knowledge of object-oriented windows application development based on MFC framework learned in the advanced programming course to make the game interface, and used DQN algorithm (the combination of Q-learning algorithm and neural network) to design and train AI, made a small reversi game to compete with AI based on reinforcement learning. This program allows players to confront AI and hone the novices' skills.

**Key words:** reinforcement learning; DQN algorithm; MFC framework; Reversi; Windows application development;

1. **引言**

在本学期的课程中，我们学习了基于微软基础类库（英语：Microsoft Foundation Classes，简称MFC）框架进行面向对象的windows应用开发的相关知识，并收获了许多使用的技能。这个框架使用C++语言封装了许多Windows API，可以非常高效地帮助我们开发一些桌面应用程序。在设计期末大作业实验时，本组成员结合人工智能的知识，选择了DQN算法这一结合了Q-learning 算法和神经网络的强化学习算法[1]，设计了一款可以使玩家与AI对战的小游戏。本次项目主要分为基于MFC的前端图形化界面，AI服务器后端，使前两者通信的网络模块，以及依赖管理四个部分。本文主要介绍基于MFC的前端图形化界面相关的设计思路与具体实现过程，其他模块相关请参考本组其他同学的论文。

1. **需求分析，程序界面与控件设计**

在程序设计的前期需求分析阶段，我们组内经过讨论，对程序界面进行了以下设计

电脑屏幕截图

描述已自动生成

该程序只需要一个主界面即可，因此选择Visual Studio 2022中基于对话框的MFC应用模板

对话框主界面，大小为510\*540，有一个字符串与四个按钮

图形用户界面

中度可信度描述已自动生成

图形用户界面

中度可信度描述已自动生成

其中字符串用于实时显示当前棋盘上黑棋与白棋各自的数量

#define IDC\_CHESSCOUNT 1001

各按钮依次为：

开始游戏按钮，点击后开始一局新游戏。

#define IDC\_START 1002

提示按钮，默认不可点击，点击会在当前执棋方可以落子的空格中心显示一个红点。

#define IDC\_PROMPT 1003

结束本局按钮，默认不可点击，点击后会结束本局游戏。

#define IDC\_ENDGAME 1004

退出按钮，结束后会直接退出程序。

#define IDC\_QUIT 1005

另外有两个光标资源，用于在之后替换鼠标箭头光标，显示当前执棋方颜色

图片包含 游戏机, 笼子

描述已自动生成 图表

描述已自动生成

光标的ID设置为：

#define IDC\_BLACK 131

#define IDC\_WHITE 132

1. **黑白棋的规则与实现**
   1. 棋子与棋盘

棋子：黑白棋的棋子呈圆饼形，每颗有黑白两面，共64个。棋局开始时棋盘中心预先布置两黑两白共四颗棋子。本次实验图形化界面为2D平面，所以使用圆形棋子。

棋盘：黑白棋棋盘以8\*8的大小最为常见。棋盘上的64格为正方形格子，落子时将棋子放在格子中间，而不是像围棋或五子棋那样放在格线的交叉点。本次实验中采取8\*8棋盘，棋盘内信息保存在8\*8的二维数组当中。

以上的实现方法是，在对话框类的头文件ReversiDig.h中定义成员变量，各变量含义见代码注释。

文本

描述已自动生成

* 1. 落子和判定胜负的规则与实现

①玩家只能将自己所执颜色的棋子放在棋盘的空格中，而不是像围棋或者五子棋那样放在格线的交点。开局时棋盘中心预先布置两黑两白共四枚棋子，黑方先落子。

②落子后，如果放下的棋子在上，左上，下，左下，左，右上，右，右下八个方向内有一个相同颜色的棋子，这两枚棋子中间的所有对方棋子的颜色会全部翻转为该颜色的棋子。

③只有在落子后可以成功翻转对方棋子的位置才是合法的落子位置。也就是说每次落子至少要翻转对手的一个棋子，否则就不能落子。

④一方成功落子后换手至另一方。如果一方没有合法的落子位置，那么他将跳过本轮落子，由他的对手一直继续落子，直到他有合法的落子位置。如果一方至少有一个合法落子位置，他就必须落子，不得跳过本轮。

⑤当棋盘填满，或者双方都无合法落子位置时，游戏结束。棋盘上存在的哪种颜色的棋子总数多，哪一方获胜。如果两种颜色的棋子一样多，则宣布平局。

在讲解实现思路前要说明的是，计算机中数组下标是先行后列（先y轴后x轴），与我们在平时在平面直角坐标系表达位置的习惯先列后行（先x轴后y轴）不符，所以本次实验中为了符合直觉，方便运算，先创建两个函数转换这个差异。

屏幕上有字

描述已自动生成

* + 1. 规则①实现为：

在处理单次落子的函数bool RefreshChessBoard(int x, int y);中， 每次落子前判断该位置是否为空，不为空则落子失败，函数直接返回。

文本

描述已自动生成

* + 1. 规则③实现为：

落子前先判断该坐标是否是符合要求的可落子的点，其中，bool ChessPlaceableAt(int, int, int);函数会在上，左上，下，左下，左，右上，右，右下八个方向调用CPoint GetNextSameColorChessPos(int, int, int, int);进行穷举搜索，寻找与己方同色的棋子的坐标，以及判断两个坐标中间是否有可翻转的对方棋子。如果可以落子，则返回True，否则返回False。（上图中仅保留y轴方向的代码用于举例）

文本

描述已自动生成

其中，CPoint CReversiDlg::GetNextSameColorChessPos(int nx, int ny, int direction, int color)函数会返回指定方向上下一个同色棋子的坐标，如果找不到，则返回事先确定的的错误值

文本

描述已自动生成

经过以上两步判定后，可确认落子成功，修改数组中对应位置的值。

* + 1. 规则②实现为：

在处理单次落子的函数bool RefreshChessBoard(int x, int y);中，如果确认落子成功，则在上，下，左，右，左上，右上，左下，右下八个方向依次穷举会被翻转的对方棋子，并修改棋盘中对应位置的值为我方棋子，完成翻转。（图中保留两个方向的代码用于举例）

文本

描述已自动生成

* + 1. 规则④实现为：

在bool CReversiDlg::RefreshCurrentColor()函数中，每次成功落子后，都会先用对手颜色在整个棋盘的所有位置穷举，如果对手有可落子位置，则交换执棋方。如果对手无法落子，则用己方颜色在整个棋盘所有位置穷尽，如果己方有可落子位置，则己方继续落子，如果己方也无法落子，则返回false，结算本局游戏。

文本

描述已自动生成

文本

描述已自动生成

* + 1. 规则⑤实现为：

在void CReversiDlg::RefreshChessConut()函数中，会扫描棋盘所有位置，统计其颜色，保存在变量中，同时在界面底部的字符串中更新显示。如果双方都无法落子，需要判断胜负时，则会调用该函数更新变量中的统计数据，并结算。

文本

描述已自动生成

1. 图形化界面绘制
   1. 棋盘

棋盘每格大小为100\*100，棋盘左上角从界面左上角50\*50位置开始

文本

中度可信度描述已自动生成

在void CReversiDlg::OnPaint()函数中执行绘制

* 1. 棋子

void CReversiDlg::DrawAllChess()函数会穷举棋盘每个位置并且调用void CReversiDlg::DrawOneChess(int nx, int ny)函数绘制棋子

文本

描述已自动生成

void CReversiDlg::DrawOneChess(int nx, int ny)会先根据棋盘形状，计算出该坐标对应的棋盘格子的中心位置，以此为圆心，绘制半径为40的圆。圆形的填充颜色会根据该棋子的值，设定在Cbrush中。

文本

描述已自动生成

1. 消息映射与处理

本次实验中主要对以下消息进行处理

文本

描述已自动生成

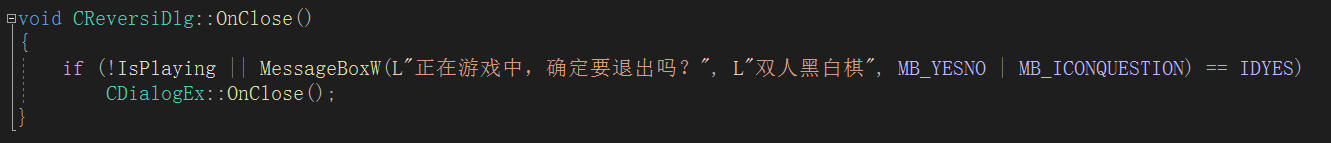
其中：

ON\_WM\_QUERYDRAGICON()完全使用系统默认设置不再赘述

ON\_WM\_PAINT()前文绘图部分已经讲解

* 1. ON\_WM\_CLOSE()在点击界面右上角X号时触发

如果此时正在游戏中，该函数会弹出框口，要求二次判断。如果不在游戏中或者二次确认，则会退出整个程序。



* 1. ON\_WM\_SETCURSOR()用于提示玩家当前执棋的是哪一方

该函数触发时，如果不在游戏中或者鼠标在棋盘范围外，则设置鼠标光标为默认光标。如果正在游戏中，且鼠标在棋盘上，则会根据当前执棋方的颜色设置鼠标为之前自定义的光标资源。当前执棋方为黑色，则鼠标光标显示为一个黑色的小棋子。白方同理。

文本

描述已自动生成

当按钮“开始游戏“被点击时，会触发 ON\_BN\_CLICKED(IDC\_START, &CReversiDlg::OnBnClickedStart)

如果此时正在游戏中，那么会弹出窗口要求二次确认。如果此时不在游戏中，或者二次确认要求重新开始。该函数会开始初始化一局新的游戏。

文本

描述已自动生成

首先会弹出一个窗口，要求玩家选择执黑棋还是白棋，并向服务器发送消息请求开始一局新的游戏。在这一过程中。会将连接服务器的情况进度会显示在下方字符串中。如果连接失败，则会显示错误信息。

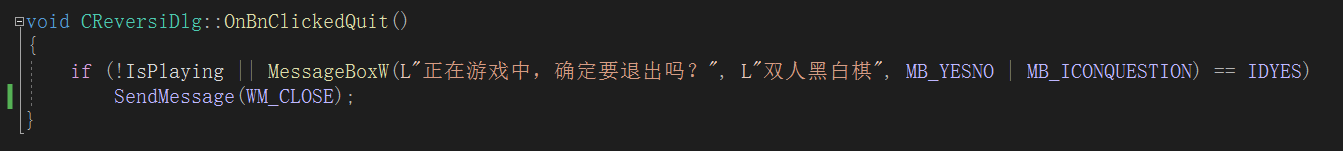
文本

描述已自动生成

之后会设置“开始游戏”按钮为“重玩”按钮，清空棋盘，设置游戏正在进行，并首次向服务器请求获取当前的棋盘。如果无法正确从服务器获取棋盘，也会在下方的字符串中提示，并弹窗提示错误的原因。在之后会设置“开始游戏”和“提示”的按钮为可用状态。获取到的黑白棋初始的棋盘有两黑两白共四个个棋子，最后会调用绘制棋盘的函数并重新统计棋子数量并显示在下方字符串中。

* 1. 当点击“退出”按钮时，会调用ON\_BN\_CLICKED(IDC\_QUIT, &CReversiDlg::OnBnClickedQuit)退出程序

如果此时正在游戏中，该函数会弹出框口，要求二次判断。如果不在游戏中或者二次确认，则会发出退出整个程序的消息。



* 1. 当点击“结束本局”按钮时，会调用 ON\_BN\_CLICKED(IDC\_ENDGAME, &CReversiDlg::OnBnClickedEndgame)

如果此时正在游戏中，该函数会弹出框口，要求二次判断。如果不在游戏中或者二次确认，则会调用void CReversiDlg::EndGame()函数结束本局游戏，而不是退出程序。

文本

中度可信度描述已自动生成

该函数会清空棋盘，设置游戏状态结束，重置各按钮的文字与状态。

文本

描述已自动生成

* 1. 点击“提示“按钮会触发 ON\_BN\_CLICKED(IDC\_PROMPT, &CReversiDlg::OnBnClickedPrompt)

该函数会扫描整张棋盘，找出所有当前执棋者可以落子的位置，并在对应位置画一个半径为10的红色圆

文本

描述已自动生成

清除提示的小红圆的函数为void CReversiDlg::CleanPrompt()该函数会设置画笔与画刷为背景色RGB（240,240,240），并在所有没有棋子的位置，都会画一个半径为11的圆，盖住之前的红色圆。

文本

描述已自动生成

* 1. 在程序页面任意位置点击都会触发ON\_WM\_LBUTTONUP()，该函数为本程序核心函数

该函数首先会将鼠标点击的位置计算转化为棋盘坐标，并调用落子函数 bool RefreshChessBoard(int x, int y);在对应棋盘坐标落子。如果落子函数判定该位置不能落子，则结束该函数。如果落子成功，则调用 bool RefreshCurrentColor();更新执棋方，并保存该函数返回的能否继续进行游戏的判断结果。。在落子过程中玩家可能点击过提示按钮，因此要消除提示的小红点。然后更新棋子数量的统计，重新绘制棋盘，之后还要根据新的执棋方更新鼠标光标的颜色。

文本

描述已自动生成

之后该函数会构造消息，将本次落子的情况告知服务器。服务器中的AI会根据玩家落子结果，计算自己的落子结果，然后返回新的棋盘数据，新的执棋方，并计算游戏能否继续进行。在与服务器通信的过程中，下方字符串会显示当前进度，并在出现错误时给出提示。然后根据服务返回的新的棋盘，重新绘制界面，更新棋子的数量并显示，以及更新光标的颜色。

文本

描述已自动生成

如果本函数之前的流程中判断双方都无字可下，则游戏结束，进入结算。根据双方棋子的数量判定胜负，然后调用EndGame()函数结束本局游戏。

屏幕上有字

描述已自动生成

1. 项目运行效果展示
   1. 运行/Reversi\_AI/backend.py，启动AI服务器后端

电脑屏幕的截图

描述已自动生成

* 1. 运行/Release/Reversi\_online.exe进入主界面

表格

中度可信度描述已自动生成

* 1. 点击开始游戏

玩家选择黑色先手，棋盘初始化为两黑两白，下方“开始游戏”按钮文字替换为“重玩”，“提示”与“结束本局”按钮激活为可以被点击。

图表, 气泡图

描述已自动生成

* 1. 落子

玩家点击期棋盘中方格落子，期间，鼠标光标也会被替换为对应执棋方的颜色。点击“提示”按钮可以显示当前执棋方所有可以落子的位置

图表, 气泡图

描述已自动生成

* 1. 游戏期间服务器也会打印接口的请求日志

文本

中度可信度描述已自动生成

* 1. 一局游戏结束，根据黑白棋子数量结算胜负

图表, 形状, 气泡图

描述已自动生成

结论

MFC是一款成熟且功能强大的应用开发框架，它能使我们用C++面向对象的方式调用Windows API，使得开发过程变得非常方便且敏捷。基于以上特点，它能很轻松的与Windows API或驱动程序结合[3]，底层程序以C语言为基础的底层程序和驱动程序，MFC在一些领域，比如工业控制和自动化方面有着巨大的优势。这也使得mfc整体的学习门槛，还是要比新兴的图形化窗口框架，比如QT要高上一些。但是在微软的不断更新优化和强大的Visual Studio的帮助下，我认为MFC至今还有较强的生命力。

致谢

在本学期的课程中，我们学习了基于MFC框架进行面向对象的windows应用开发的相关知识，并收获了许多使用的技能。在这期间，老师细致通透的讲解，认真负责的教学态度让我非常感动和敬佩。在平时的学习和实验过程中常常能非常耐心的解答我们提出的问题，这一过程极大的加深了我对所学知识的理解，使我能够迅速成长。在本次的期末大作业中我与ACD,3位同学组成小组，共同完成了这一项目。在这期间我们举行了多次讨论与互相指导，非常高效地检验了自己所学的知识，查漏补缺取长补短。这一过程极大的锻炼了我的需求分析能力，编程实战能力，项目管理能力，沟通交流能力和团队协作能力，可谓收获颇丰。

最后，再次向老师和A，C，D，3位同学表示感谢。

参考文献：

1. 许聪.基于强化学习的小游戏设计与实现[J].信息与电脑(理论版),2022,34(07):91-93.
2. 郭潇逍,李程,梅俏竹.深度学习在游戏中的应用[J].自动化学报,2016,42(05):676-684.DOI:10.16383/j.aas.2016.y000002.
3. 刘变莲,刘东波,柴晓芳.MFC与Windows程序[J].电脑知识与技术,2015,11(32):64-65.DOI:10.14004/j.cnki.ckt.2015.3122.
4. 黄维通, 贾续涵. Visual C++ 面向对象与可视化程序设计[M]. 第三版. 清华大学出版社, 2011年6月.
5. 黄维通, 解辉. Visual C++ 面向对象与可视化程序设计实例解析[M]. 第4版. 高等教育出版社, 2016年8月.