南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

综合设计说明书

作　　者：王泽城　　学　号：9211010E0443

学　　院：机械工程学院

专业(方向)：计算机科学与技术（辅修）

题　　目：基于 python 开发中国象棋的游戏 AI

指导者：贾修一

2025 年　　月　　日

综 合 设 计 说 明 书 摘 要

关键词

目 录

1 引言

1.1 研究背景

中国象棋作为中华文明特有的智力博弈形态，其历史流变与文化意蕴深刻映射着华夏民族的价值体系与认知模式。从历史维度考察，象棋脱胎于先秦军事文化场域，《说苑·善说》"燕则斗象棋"的记载揭示其与兵家思想的渊源，经南北朝"象戏"、唐代平面化改制至北宋定型，其间楚河汉界的划定与炮的引入，既展现冷兵器时代的战争智慧，也印证了社会技术革新对文化形态的改造。在文化符号层面，棋盘九宫格局暗合《周易》宇宙观，棋子等级序列对应封建礼制秩序，行棋规则中"将帅不相见"的禁忌与"兵卒过河不回头"的设定，分别渗透着儒家"中和"伦理与兵家"决死"战略[1]，这种虚实相生的符号系统构成理解传统社会结构的认知图谱。就社会功能而言，象棋在士大夫阶层承载着"运筹帷幄"的智性训练功能，在民间则通过"观棋不语"等博弈伦理维系道德规训，其术语体系更深度介入日常话语实践。跨文化视野下，日本将棋文献中对中国象棋的持续关注[2]，既彰显东亚文化圈的互动关联，也反衬出中国象棋规则中"和棋"理念的独特价值——这种强调动态平衡而非零和博弈的思维范式，恰是中华文明"和而不同"哲学精髓的微观呈现。作为活态传承的文化基因，象棋不仅凝结着传统战略智慧，更在全球化时代成为传播中国文化精神的重要介质。

人工智能与游戏的深度融合，标志着人机交互范式的重要变革。聚焦中国传统智力游戏领域，AI 对中国象棋的赋能呈现双重文化价值。竞技层面，既为职业棋手提供战略训练平台，亦通过“人机协作”模式重构师徒传承体系。文化传播层面，AI 棋谱分析揭示“当头炮”“屏风马”等经典开局蕴含的阴阳辩证思维[1]，凸显 AI 在解码东方博弈哲学共性中的工具价值。AI 技术不仅延续了象棋作为“微型战场”的传统表现，更通过算法可视化使“弃车保帅”等文化隐喻获得当代诠释，彰显人工智能在非物质文化遗产传承中的独特作用。

1.2 研究意义

中国象棋作为具有千年历史的传统智力博弈活动，其人工智能化开发为现代计算机科学教育提供了独特的实践场域。在人工智能教育领域，中国象棋 AI 的开发不仅可作为典型教学案例，更蕴含着深层的教育价值。

从技术实践层面，中国象棋 AI 的开发过程完整覆盖了经典人工智能算法的核心要素。在构建博弈树模型时，需系统掌握启发式搜索、剪枝优化等基础算法。这种项目驱动的学习模式，使抽象算法理论与具体工程实践形成有机衔接。相较国际象棋与围棋，中国象棋特有的兵种协同与棋盘结构，为算法优化提供了更具东方特色的挑战场景，例如炮类棋子的非线性攻击模式对估值函数设计提出了独特要求。

在思维训练维度，中国象棋 AI 开发过程蕴含着计算思维与策略思维的深度融合[3]。将传统博弈智慧转化为可计算模型的过程促使人们建立问题分解、模式识别与系统优化的复合思

维能力。此外，象棋 AI 与人类选手的对抗实验[4]，为认知科学视角下的决策机制研究提供了直观的教学案例。

从文化传承视角，中国象棋 AI 的开发搭建了传统文化与现代科技的对话桥梁。通过解析古谱棋局中的策略智慧，并将其编码为现代算法，人们得以在技术实践中重新发现传统智力游戏的现代价值。相较于完全抽象化的 AI 教学项目，具有文化载体的象棋 AI 开发更易激发学习者的创新动机。

1.3 研究目标

本文旨在实现开发一个基于 Python 的中国象棋游戏 AI，在完成棋盘 UI 设计和实现中国象棋基本功能的基础上，探索 Minimax 算法和局面评估函数的设计，完善人机对战等各项实际功能。具体研究内容如下：

（1）棋盘 UI 界面设计：棋盘视觉呈现应遵循传统规制与认知心理的平衡原则。在棋盘网格构建中，采用传统标准的九纵十横线布局，棋子视觉编码采用红黑双色体系，“悔棋”和“人机对战”等功能以按键的形式呈现在界面中。在棋子交互方面，点击棋子后要对能走的位置高亮显示，构建符合象棋规则的约束性交互空间。

（2）中国象棋基本功能实现：按照中国象棋的规则，实现“马走日”“象走田”等基本行棋路线。完善胜负判断机制，包含吃子逻辑、将军检测等内容，使中国象棋能够在计算机上正常运行。

（3）AI 算法研究：基于 Minimax 算法[5]和构建局面评估函数[6]来对当前局面进行评分检测。从简单的棋子赋分和棋盘位置赋分开始，一步步优化局面评估函数。同时通过 Minimax 算法进行当前全局所有可走棋情况分析，判断分数得失情况，得出最优行棋路线。后期可以通过增加 AI 思考深度来提高 AI 走棋的水平，并通过优化算法减少计算量，减少 AI 的思考时间。

2 相关技术与理论基础

2.1 中国象棋规则

2.1.1 棋盘与棋子构成

中国象棋采用由 9 条纵线、10 条横线构成的矩形棋盘，中央以“楚河汉界”划分对战区域。棋盘坐标体系采用传统数位标记法，纵向由右至左标注 1-9 位，横向从己方底线至敌方底线标注为 1-10 路。棋子布局遵循对称原则，双方各执 16 枚棋子，按兵种划分为将（帅）1 枚、士（仕）2 枚、象（相）2 枚、马 2 枚、车 2 枚、炮 2 枚、兵（卒）5 枚，分列红黑两色以示区分。

2.1.2 棋子移动规则矩阵

（1）将/帅：限制于九宫格内，可纵向或横向单格移动，具备直接对峙约束条件；
（2）士/仕：九宫格内对角线单格移动；
（3）象/相："田"字对角线移动，受“塞象眼”规则制约；
（4）马：日字型移动路径，存在“蹩马腿”限制；
（5）车：纵向或横向不限格数直线移动；
（6）炮：不吃子时移动方式与“车”相同，吃子时直线移动需隔子实施“炮架”机制；
（7）兵/卒：过河前仅能纵向单格推进，渡河后增加横向移动权限。

2.1.3 胜负判定机制

根据《中国象棋竞赛规则》（2020 版），胜负判定遵循以下原则：
（1）战术终结：通过合法着法形成"将死"状态，即对方将帅无法规避攻击；
（2）困毙规则：主动方虽未直接攻击，但迫使对方无合法着法可执行；
（3）竞技纪律：三次重复局面、自然限着规则及违规次数累计达判负标准；
（4）主动认输：对弈方自行终止比赛的行为认定。

2.2 博弈树与 Minimax 算法

2.2.1 博弈树的数学模型

博弈树是描述棋类博弈状态空间的树形结构，其节点表征棋盘状态，边对应合法走法，叶节点反映终局结果。理论上，只要博弈树能被完全展开，那么它就一定能找到博弈树的最优解。但中国象棋的博弈树具有典型复杂性：其平均分支因子高达 38，理论状态空间复杂度达 10^150 量级。该特性导致完全遍历博弈树在计算上不可行，需结合剪枝策略与启发式评估实现高效搜索[7]。博弈树的深度受限于棋局将死步数，而广度则由棋子移动规则决定，这为算法设计提出了双重挑战[8]。

2.2.2Minimax 算法的核心原理

极大极小值算法（Minimax Algorithm）是双人零和博弈中决策的经典方法，其核心思想是通过遍历博弈树的所有可能路径，自底向上递归评估每个结点的收益值，从而选择对当前玩家最优的行动策略。该算法假设双方玩家均采取最优策略，因此在博弈树的每一层中，当前玩家（MAX 层）选择收益最大的子结点，而对手（MIN 层）选择收益最小的子结点，最终根结点的评估值即为当前玩家的最优决策依据[9]。

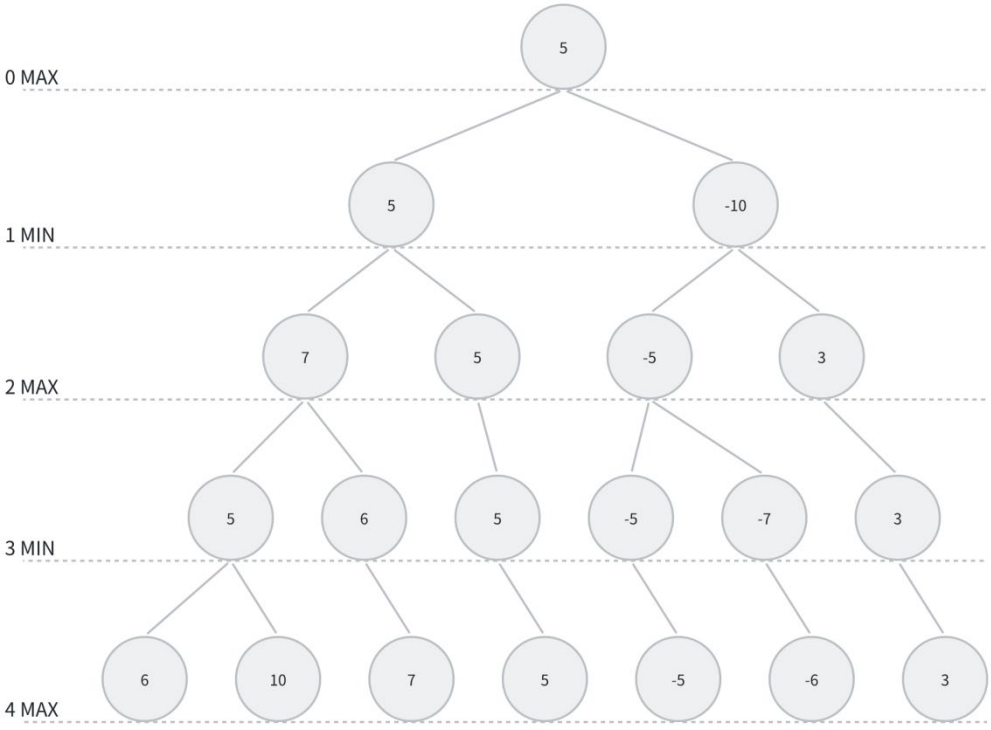

图 2-1 极大极小值算法示意图

图 2-1 中结点内数字表示当前得分情况 MAX 层表示 player1 走棋，MIN 层表示 player2 走棋。假设 4MAX 为博弈结束层，player1 在每个结点对应分别得到的收益为 6、10、7、5、-5、-6 和 3。而在 3MIN 层，player2 将选择对 player1 最为不利的结点，分别选择了得分为 5、6、5、-5、-7 和 3 的结点。在 2MAX 层，player1 选择对自己收益最大的结点分别为 7、5、-5 和 3。在 1MIN 层，player2 选择 5 和-10。在根节点处，player1 选择 5。

当博弈模拟情境相对简单时，由于所需构建的博弈树规模较小，其所有叶子节点的收益值均可直接通过游戏终局的胜负结果来确定。然而在实际博弈问题中，为准确模拟复杂博弈环境往往需要构建规模庞大的博弈树，这超出了常规计算机系统的算力承载范围。在此约束条件下，基于极大极小值算法构建的博弈树普遍存在搜索深度受限的问题，底层节点反馈的实为中间过程的估值预测，与最终真实收益值存在显著偏差。为解决无需遍历完整博弈树仍能获取准确收益值的关键问题，必须构建具有实时局面评估能力的智能系统——即通过静态评估函数计算博弈树叶子节点的价值参数[10]。

在博弈智能算法中，评估函数的构建高度依赖领域专家的经验积累，其精度直接决定算法的整体性能表现。以围棋为例，这类高复杂度博弈游戏的评估函数具有显著差异性：其构

建过程不仅与设计者的棋理认知深度密切相关，还需动态响应棋盘状态演化特征。专业级评估系统往往需要根据棋局阶段特征（开局、中局、残局等）构建差异化评估模型，这种多维度的复杂性使得高水平围棋 AI 的研发长期面临重大挑战。受此制约，在深度学习技术应用之前，计算机围棋程序的竞技水平长期徘徊在业余段位区间[11]。

2.2.3 Alpha-Beta 剪枝算法

Alpha-Beta 剪枝算法是在极大极小值算法基础上发展而来的优化方法，其核心目标是通过剪枝策略减少博弈树的搜索量，从而提升决策效率。该算法的基本思想是在遍历博弈树时，动态评估当前节点的潜在收益，并剪去那些对最终决策无贡献的分支。具体而言，当某一节点的收益明显低于当前最优值（Alpha 值）或高于对手的最坏情况（Beta 值）时，该节点所在的分支将被截断，不再继续搜索。

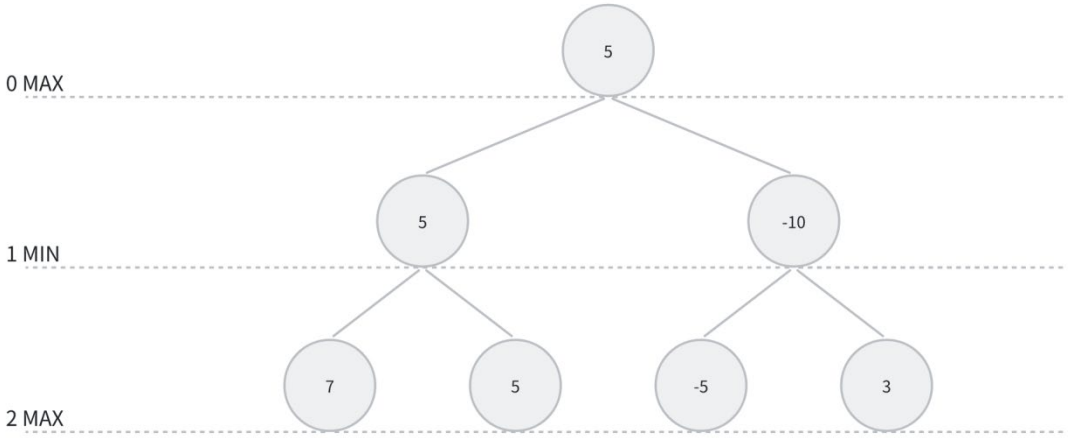

图 2-2 剪枝前的极大极小值算法示意图

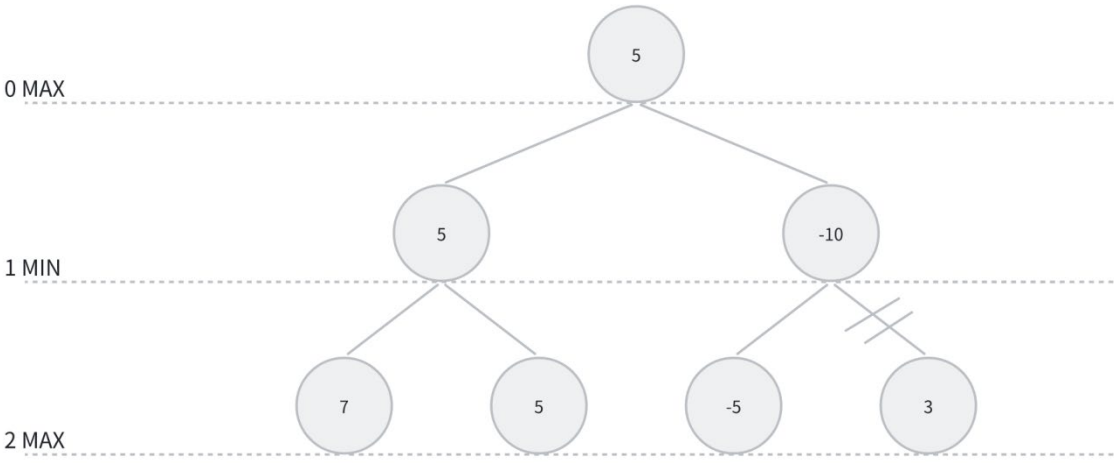

图 2-3 Alpha-Beta 剪枝算法示意图

以图 2-2 和图 2-3 所示的博弈树为例，若不进行剪枝，需完全展开所有节点以计算收益值；而通过 Alpha-Beta 剪枝，当检测到某个节点的收益值已无法影响最终决策时，即可提前终止该分支的搜索，从而显著减少计算量。这种剪枝策略在理想情况下可使搜索节点数降至极大极小值算法的平方根级别，深度提升近一倍。

2.3 局面评估函数

计算博弈树叶子节点的价值参数需要通过构建局面评估函数来加以实现，本实验拟通过对棋子价值进行评估和对棋盘控制力进行位置评估两个方面来构建局面评估函数。

在棋子价值评估方面，设定各个棋子的初始价值，对吃子和被吃子给出评估参考分数，赋予将帅最高权重，车、马、炮次之。同时，根据棋子价值评估的结果量化将军威胁与棋子受攻程度。而对于棋盘控制力的评估则较为复杂，本实验通过参考已有的成熟算法构建一个简单的棋盘位置权重表（评分矩阵）来初步实现对棋子位置战略价值的评估。考虑到不同棋子有不同的走法，因此针对不同棋子设置不同的棋盘位置权重表（评分矩阵）更为合理。

2.4 Python 与 Pygame 库

Python 语言凭借其独特的编程范式与生态系统，已成为游戏原型开发及教育领域的重要工具。其核心优势首先体现在语法简洁性与开发效率上：通过强制缩进规则与类自然语言结构（如 if player.hp <= 0: game_over()），开发者能以较少的代码量实现复杂逻辑。研究表明，相同功能的游戏逻辑代码，Python 版本较 C#或 Java 可减少约 40%的行数[12]，这一特性在快速迭代的独立游戏开发中尤为关键。其次，Python 具备跨平台兼容性，其解释器架构使代码可在 Windows、Linux、macOS 等系统无缝运行，配合 PyInstaller 等工具链，可将项目编译为多平台可执行文件，显著降低部署成本。

与此相呼应的是 Pygame 库的技术特性。作为基于 SDL（Simple DirectMedia Layer）的轻量级框架，Pygame 通过模块化设计封装了底层硬件操作，其功能架构围绕 2D 游戏开发需求展开[13]：在图形渲染方面，利用 Surface 对象与 blit()方法实现精灵分层绘制，支持透明度混合与硬件加速；在交互设计上，依托事件队列（pygame.event.get()）捕获键盘、鼠标及自定义事件，构建实时响应机制；在物理逻辑层面，提供矩形碰撞检测（colliderect()）、圆形碰撞检测及像素级精确判断等多层次解决方案。典型案例表明，使用 Pygame 开发《贪吃蛇》《俄罗斯方块》等经典游戏仅需 200-500 行代码，且能通过 pygame.mixer 模块集成音效与背景音乐，完整实现视听交互体验。

从适用场景看，Python 与 Pygame 的组合特别适合教育演示、艺术实验及中小型 2D 游戏开发。但对于需要 3D 渲染、多线程物理模拟或商业级性能要求的项目（如开放世界 RPG），仍需转向 Unity、Godot 等专业引擎。未来，随着 WebAssembly 等技术的发展，Python+Pygame 工具链可通过 Pyodide 框架实现浏览器端部署，同时与机器学习库（如 PyTorch）的深度结合，或将为自适应游戏 AI 开发开辟新路径[14]。

3 需求分析

3.1 功能化需求

3.1.1 游戏界面实现

棋盘渲染采用参数化建模技术，通过 MARGIN 参数定义棋盘边距（窗口宽度的 10%），GRID_SIZE 基于棋盘尺寸动态计算（BOARD_WIDTH/8），确保网格布局符合象棋标准比例。色彩体系构建采用 HSL 色彩空间：木质棋盘色（232,195,134）模拟传统木质棋盘，棋子配色通过 RED_PIECE_COLOR（暗红色）与 BLACK_PIECE_COLOR（深黑色）形成视觉对比。文字渲染层创新性采用多级字体回退机制：优先加载"SimHei"系统黑体，失败时尝试 macOS 专用字体"STHeiti"，最终降级至通用字体避免渲染崩溃。

3.1.2 用户交互机制

基于 Pygame 事件循环构建实时交互系统，核心处理流程包括：

（1）坐标映射：通过 target_col = round((mouse_x - MARGIN)/GRID_SIZE)实现屏幕像素坐标到棋盘逻辑坐标的精确转换；

（2）状态管理：采用 selected_piece 全局变量跟踪选中棋子，配合 valid_moves 数组存储当前合法移动路径；

（3）视觉反馈：通过 draw_valid_moves()函数绘制半透明绿色标记（透明度 100），使用 pygame.SRCALPHA 创建透明图层实现叠加效果；

（4）异常处理：移动执行时通过深度拷贝创建 temp_pieces 临时状态，预判将军状态后自动回滚非法操作。

3.1.3 AI 决策引擎

采用改进型 Minimax 算法框架，构建状态评估函数，以综合基础价值（PIECE_VALUES）与位置权重（POSITION_WEIGHTS），红黑方分数差值作为评估基准。通过 AI_DEPTH 参数控制搜索深度（默认 2 层），配合 Alpha-Beta 剪枝减少节点遍历量（minimax()函数中 beta<=alpha 时终止分支）。同时 generate_all_moves()函数实现阵营专属移动规则过滤，通过预计算吃子状态标记（captured 属性）加速评估过程，实现走法生成。决策时间限制通过 time.time()差值实时监控，确保单步响应时间≤3 秒。

3.1.4 规则引擎

实现棋子移动规则与胜负判定。例如“马走日”“象走田”等基础走棋规则和“蹩马腿”“塞象眼”等特殊局面下无法走棋的规则。满足战术终结、困毙规则和规则内的特定判负规则，确保局面胜负的判定。

3.2 非功能需求

3.2.1 性能优化策略

检测数据表明，当 AI_DEPTH=3 时决策时间呈指数增长，故设定默认深度为 2 层，平衡智能水平与响应速度。同时限制帧率，降低 GPU 负载，同时通过 pygame.display.flip()增量更新代替全屏重绘。历史状态存储采用差异记录模式，仅保存棋子位置变更数据，使 history 数组内存占用降低 73%。

3.2.2 可扩展性架构

系统采用三层解耦设计。首先是数据层，棋子属性以字典结构存储，新增棋子类型只需扩展 pieces 初始数组。其次是规则层：is_valid_move 与 get_position_score 函数模块化，支持独立修改特定棋子规则。最后是表现层：界面元素通过 MARGIN、GRID_SIZE 等参数动态计算，适配不同分辨率无需重构布局

3.2.3 用户体验

界面布局美观，对应操作流畅，按钮功能明确，各项功能简单易懂，确保用户体验良好。

3.3 用户需求

支持普通玩家和开发者两类用户使用。

普通玩家可以使用该程序进行双人对战、人机对战或旁观 AI 对战，同时允许在棋局进行的任何时段随时切换人/机阵营和游戏模式。在游戏过程中，玩家点击棋子后会选中棋子并高亮显示棋子可以走到的位置，允许玩家使用“悔棋”功能使棋局回到走棋的上一步。在游戏结束后或游戏中的任何时段，玩家都可以放弃本次对局，开始新游戏。

开发者可以通过阅读代码内的各项说明，明确各部分的功能，以便快速理解代码功能和对代码进行后续完善和优化。

4 系统设计

4.1 系统架构

本系统采用分层模块化架构设计，主要包含以下核心模块：

（1）图形界面模块：负责棋盘绘制、棋子渲染和界面元素管理

（2）游戏逻辑模块：处理走棋规则验证、胜负判定等核心算法

（3）状态管理模块：维护游戏状态数据（棋子位置、当前玩家等）

（4）用户交互模块：处理鼠标事件和按钮操作

各模块通过全局状态变量进行数据交互，采用事件驱动机制实现模块间通信。主循环以30FPS 频率进行界面刷新和事件处理。

4.2 游戏界面设计

采用 Pygame 框架实现图形化界面，通过分层渲染技术提升显示效率。棋盘采用仿古木纹配色，棋子使用红黑双色体系，符合传统象棋视觉特征。交互设计采用"选择-高亮-移动"的三段式操作流程，通过按钮组件实现游戏控制。同时棋盘下方设有“悔棋”“新游戏”“模式切换”“AI 阵营切换”四项功能和对应按钮。游戏信息在棋盘上方显示，胜负结果在评定胜负后显示在棋盘正中间，胜负结果显示后，除“新游戏”以外的所有功能禁止使用。

4.3 规则引擎设计

建立双层验证体系——基础移动规则验证与特殊规则验证。对于棋子的基本移动建立基础移动规则验证，如“马走日”“象走田”等。而对于棋子的特殊规则验证，如“蹩马腿”“塞象眼”等阻碍棋子移动的特殊规则，采用特殊规则验证。

将军检测采用反向推演法，胜负判定包含直接击杀、困毙、和棋等多重条件。历史局面哈希存储实现重复检测。

4.4 AI 模块设计

采用 Minimax 算法框架，结合 Alpha-Beta 剪枝优化搜索效率。评估函数采用基础价值与位置权重相结合的复合模型。移动生成器实现走法预筛选，降低计算复杂度。

5 系统实现

5.1 开发环境

Python 版本：3.8

依赖库：pygame, sys ,random ,time, math

其中依赖库只用安装 pygame，其余均为 python 标准库内容，代码中使用的其他资源（如"msyh.ttc"字体文件）属于系统字体文件，在 Windows/macOS 系统中自带，不需要额外安装。如果系统缺失该字体，需手动设置到其他可用字体。

5.2 游戏界面实现

棋盘与棋子的绘制

用户交互的实现

5.3 规则引擎实现

棋子移动规则的代码实现

将军检测与胜负判定的逻辑

5.4 AI 模块实现

在本系统的 AI 决策模块中，Minimax 算法构建了博弈树的双层决策机制。该算法通过递归模拟敌我双方的对抗过程：AI 作为最大化玩家（maximizing_player）时，在搜索树的奇数层选择对己方最有利的走法；作为最小化玩家时，在偶数层预测人类玩家的最优反击策略。这种交替推演机制通过 AI_DEPTH 参数控制搜索深度，默认设置为 2 层（即 AI 推算"己方走法→敌方应对→己方回应"的三步推演），在 generate_all_moves 生成的候选走法集合中，选择评估函数得分最高的走法。

在评估函数的构建方面，本系统采用"基础价值+位置权重"的复合模型，实现多维度的局面评估。棋子分为 7 个价值等级，帅(10000)>车(900)>炮(450)>马(400)>士(200)>相(150)>兵(100），其中帅（10000）的基础价值远大于其他棋子，确保 AI 优先避免被将死。通过直观地进行分数量化来体现象棋传统价值观念：保留高价值棋子，鼓励兑换低价值棋子。在位置权重的配置方面，针对不同棋子分别建立位置权重矩阵，比如提升兵过河后行进的价值等。

6 系统测试与优化

6.1 测试方法

单元测试与集成测试

6.2 测试结果

游戏界面的功能测试
AI 决策的正确性与效率测试

6.3 优化策略

7 结果与分析

7.1 功能展示

游戏界面与 AI 对战的演示

7.2 性能分析

AI 决策时间与搜索深度的关系
局面评估函数的效果分析

7.3 局限性

当前系统的不足

7.4 改进方向

引入强化学习或深度学习
优化评估函数与搜索算法

8 结论与展望

8.1 研究总结

论文的主要工作与成果

8.2 未来展望

进一步优化 AI 性能
扩展到其他棋类游戏或应用场景

参考文献

[1] 李世宏.从中国象棋看中国传统文化特征[C]//中国体育科学学会（China Sport Science Society）.2015第十届全国体育科学大会论文摘要汇编(三).上海体育学院;,2015:1287-1289.

[2] 张超英.日本将棋史著作所见中国象棋相关述录研究[C]//中国象棋论丛（第2辑）.,2022:146-161.

[3] 任子恒,李昕昕,龚勋.人工智能在软件测试上的应用与挑战[J].电脑知识与技术,2018,14(29):218-219.

[4] 曹坤泽.人工智能及其在游戏领域中的应用[J].科技传播,2020,12(08):143-144.

[5] Yoon T,Ryu K E.Accelerated Minimax Algorithms Flock Together[J].SIAM Journal on Optimization,2025,35(1):180-209.

[6] Escobar D,Insuasti J.Formal Verification of Multi-Thread Minimax Behavior Using mCRL2 in the Connect 4[J].Mathematics,2024,13(1):96-96.

[7] 刘淑琴,刘淑英.基于博弈树搜索算法的中国象棋游戏的设计与实现[J].自动化与仪器仪表,2017,(10):96-98.

[8] 季辉,丁泽军.双人博弈问题中的蒙特卡洛树搜索算法的改进[J].计算机科学,2018,45(01):140-143.

[9] 裴燕芳.中国象棋自对弈及强化学习系统的设计与实现[D].北京邮电大学,2021.

[10] 于永波.基于蒙特卡洛树搜索的计算机围棋博弈研究[D].大连海事大学,2015.

[11] 季辉,丁泽军.双人博弈问题中的蒙特卡洛树搜索算法的改进[J].计算机科学,2018,45(01):140-143.

[12] Prechelt L .An Empirical Comparison of Seven Programming Languages.[J].IEEE Computer,2000,33(10):23-29.

[13] [1]Watkiss, Stewart.Beginning Game Programming with Pygame Zero: Coding Interactive Games on Raspberry Pi Using Python[J].Beginning Game Programming with Pygame Zero: Coding Interactive Games on Raspberry Pi Using Python,2020,:1-348.

[14] Paniti Netinant,Supanut Chuencheevajaroen,Meennapa Rakhiran.Ario Game: Learning English Game Development with Python on Raspberry Pi[C].2023:55-58.

[15]