

# 桌游类小游戏开发框架

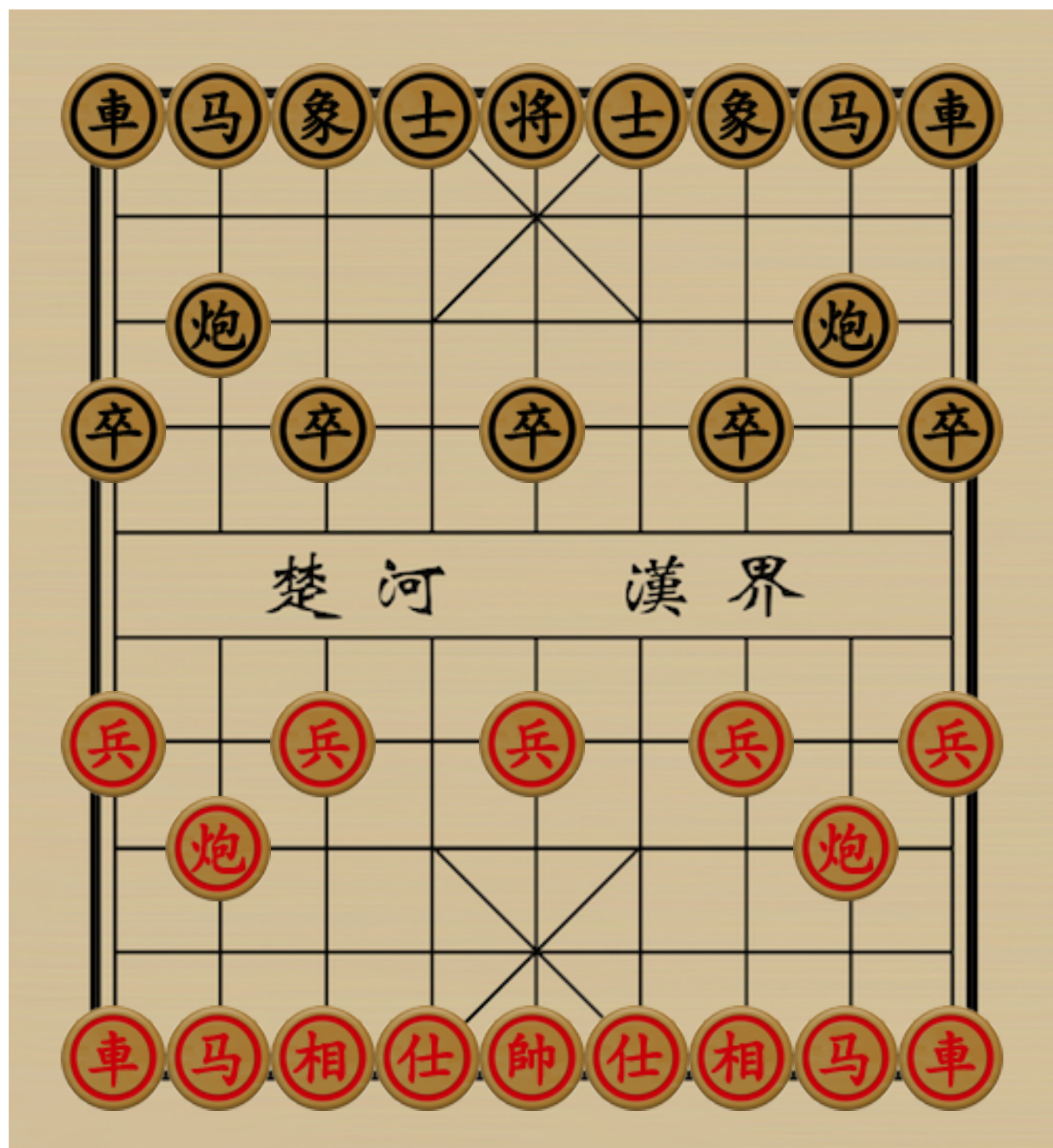
南方小智

2020 年 10 月 17 日

# 目录

<b>第一章 概述</b>	<b>4</b>
1.1 安装与设置 . . . . .	4
<b>第二章 基本元素</b>	<b>5</b>
2.1 Board . . . . .	5
2.2 Action . . . . .	6
2.3 Status . . . . .	6
2.4 Player . . . . .	6
2.5 Judge . . . . .	6
2.6 Game . . . . .	7
<b>第三章 卡坦岛设计</b>	<b>8</b>
3.1 游戏状态转换 . . . . .	8
3.2 坐标系设计 . . . . .	9
<b>第四章 AI 设计</b>	<b>10</b>
4.1 AIPlayer . . . . .	10
4.2 Evaluator . . . . .	10
4.3 MaxMinAIPlayer . . . . .	10
4.4 置换表 . . . . .	11
4.5 历史表 . . . . .	11
<b>第五章 对弈平台</b>	<b>12</b>

<b>第六章</b>	<b>TODO</b>	<b>13</b>
6.1	基础设施 . . . . .	13
6.2	中国象棋 . . . . .	13
6.3	五子棋 . . . . .	14
6.4	卡坦岛 . . . . .	14



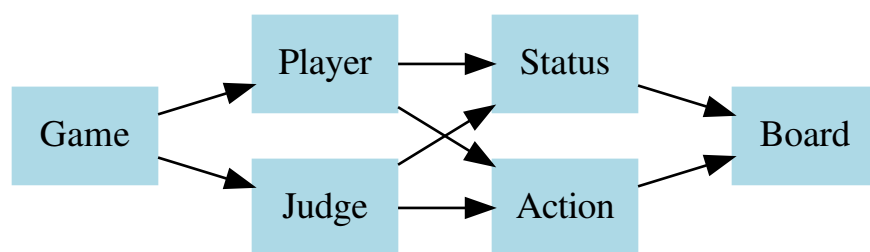
# 第一章 概述

本项目用于实现各类桌游小游戏的游戏流程，AI 玩家，对弈平台等。目前正在开发中国象棋的基本游戏流程和 AI 算法。

## 1.1 安装与设置

```
1 python3 -m pip install termcolor
```

## 第二章 基本元素



桌游小游戏基本元素

游戏中的基本元素有，Board，Status，Action，Player，Judge，Game 等。外部程序通过设置 Game，Players，和 Config，可以调用该框架。

### 2.1 Board

游戏中最基本的元素是 Board，相当与棋盘，牌桌。其中包含各类游戏物件以及他们的位置，比如棋子，纸牌和骰子等。

Board 提供对这些游戏物件的访问和改变，比如改变某个物件的位置。同时也对 Board 上的基本状态进行检测，比如在中国象棋中判断当前局面上将帅是否照面。

## 2.2 Action

Action 代表某个玩家可以在 Board 上的一个操作。比如把某个子移动到另一个地方，称为 MoveAction。可以通过继承 Action 实现更高级的动作，比如悔棋操作。MoveAction 应当是可以撤销的 (roll\_back)，这样可以支持悔棋的功能，以及在 AI 搜索算法中可以利用其实现回溯。

认输也可以是一个 Action。

## 2.3 Status

当前游戏的所有状态，包括 Board 的格局，当前玩家，获胜玩家，Action 的历史栈等。获得 Status 就可以知道游戏从初始到现在所有的需要的状态数据，也就是可以通过 Status 对整个游戏过程进行复盘。

## 2.4 Player

Player 的主要任务是，在自己为主的一轮当中，通过当前的 Status，按照顺序作出一个或多个 Action。

Player 可以是自动的 AIPlayer，也可以接收输入的真人玩家，网络玩家等。

## 2.5 Judge

Judge 负责判定 Player 产生的 Action 是否合法，并最终负责执行合法的 Action。Judge 还负责判断游戏是否已经结束，谁是胜利玩家等。Judge 通过 Rule 来进行这些判定，不同的 Rule 集合可以产生略微不同的游戏规则，比如，可以在象棋中去掉将帅不可照面的规则等。

某些特殊的中国象棋残局添加了额外的限定，这种 Judge 和 Rule 解耦分开处理的方式有利于实现这些残局游戏。

Judge 还可以用于控制游戏状态和流程，比如实现回退，保存局面等。

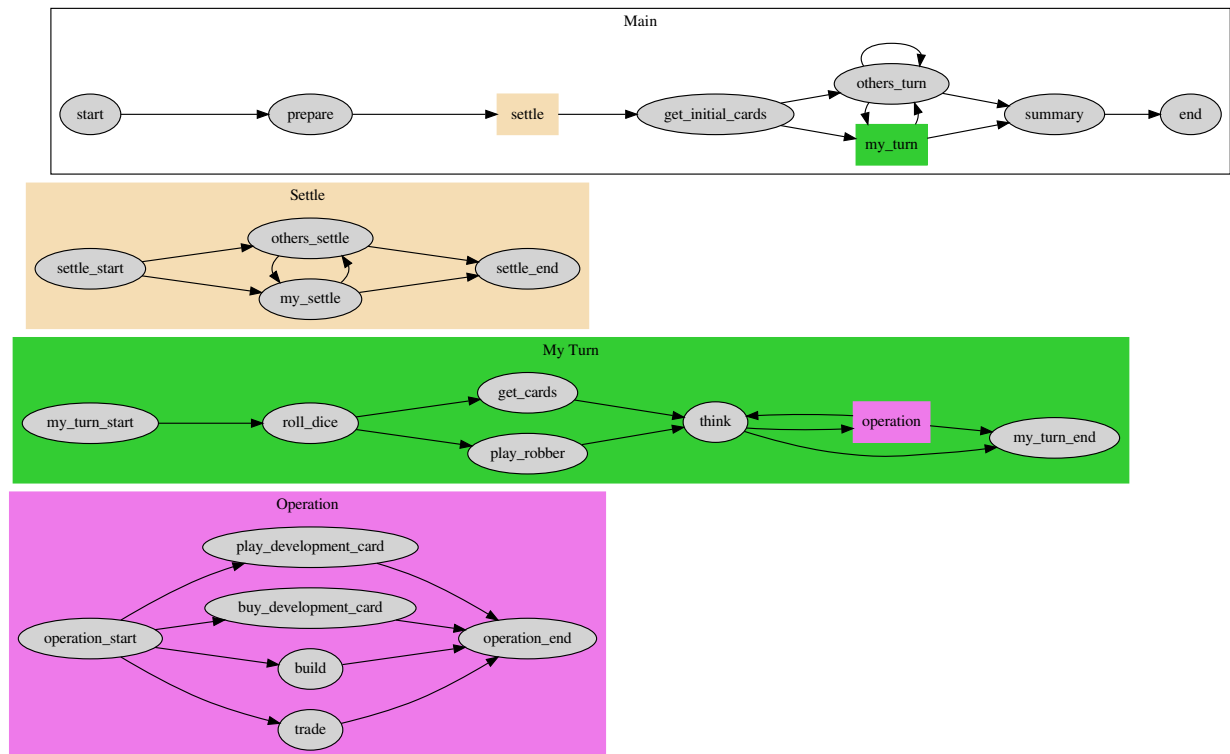
## 2.6 Game

Game 控制整个游戏的流程，每个游戏由准备阶段开始，然后经过若干轮，每轮以其中一位 Player 为主，并由 Judge 执行操作。最后利用 Judge 判断游戏结束和得出胜利玩家列表。



# 第三章 卡坦岛设计

## 3.1 游戏状态转换



卡坦岛玩家状态转换

## 3.2 坐标系设计

## 第四章 AI 设计

AIPlayer 可以实现自动的游戏过程，一些通用的 AIPlayer 可以被不同的游戏重复利用。

### 4.1 AIPlayer

最基本的 AIPlayer 是随机的 AIPlayer。首先根据规则，获取所有可能的 Action，然后在其中随机选择一个 Action 返回。

### 4.2 Evaluator

Evaluator 是对当前 Board 局面的一个评估函数，返回一个  $[0, 1]$  的值，0 代表是最糟糕的局面，1 代表是最好的局面。

通过 Evaluator，我们可以获得稍好于随机操作的 AIPlayer。同样的，可以首先根据规则，获取所有可能的 Action。然后执行每个 Action，对新的局面调用评估函数，可以得出执行哪一个 Action 会得到对自己最有利的局面，然后返回这个 Action。

### 4.3 MaxMinAIPlayer

通过极大极小搜索和 AlphaBeta 剪枝实现多步的搜索。同样的会在搜索的尾部，调用 Evaluator 对局面进行评估。目前的单机搜索深度能够达到 4 到 5 层。

## 4.4 置换表

置换表可以用于保存某个局面的搜索评分，这样，下一次搜索到同一个局面时，可以检查是否已经有可用的评分了。通过哈希函数可以将局面映射到一个哈希键值，以及为局面加锁以防止冲突局面。

## 4.5 历史表

历史表对每种移动走法打分，并对打分高的的走法优先搜索，使得 AlphaBeta 剪枝的效率更高。本框架未实现历史表，而是通过对 Action 导致的新局面的评估函数作为该 Action 的分数进行排序。

## 第五章 对弈平台

目前尚未开始这个阶段的工作。

# 第六章 TODO

## 6.1 基础设施

- 平台化
  - 制作游戏 UI。
- Better Engineering
  - 完善 Readme 文档。
  - Modify all assert to exception

## 6.2 中国象棋

- 将中国象棋相关代码移动到一个 folder 下。
- 判断和棋：长将，长捉，50 步无吃子，双方无子可以过河等
- 棋型监测：双炮，马后炮，双车错，侧面虎等。
- 借助 latex 项目，实现静态棋盘的图片的生成，以及实现动态棋局 gif 图的生成。
- 整理以前写的[Javascript 版本的象棋引擎](#)

### 6.2.1 AI 设计

AI 相关内容更加复杂，需要更长时间的 test，先 Focus 在基础设施的建设上，为 AI 的开发提供足够的工具集合。

- 添加 Unit Test，对每个部件单独进行测试，比如 player, evaluator, judge 等。现在还有许多需要改进的点，每个点都建立对应的 test cases。
- 制作工具快速添加新棋局到 data 中。

- 残局库：将象棋软件中的残局库添加到 data 中，可以通过残局对 AI 进行测试。
- 加强残局能力：对将军的 action 优先搜索，或者对将军的局面提高评分。可以同时计算所有的控子情况。通过对局面计算所有的 actions，然后对所有吃子进行加成。
- 假设后手先走，现将后手可能的杀招计算出来，从而能够对 actions 的排序更加精确。
- 如何训练和制作开局库？
- 添加多线程实现 AI 加速。
- 当 AI 意识到自己输了的时候，AI 也要选取最长路径来走。
- MCT:
  - MCT 还有很多不明确的点，先确认算法，然后实现一种更加 readable 的版本，并对每个步骤提供足够的 unit test。
  - 似乎后手的搜索不太起作用。考虑一个残局，先手优先可以吃子，但是后手可以将死先手。这时，先手陷入到了吃子的陷阱，并没有发现会被将死。测试”data/JJ 象棋残局第 98 关.bd”。
- 实现 AI 的自动博弈，并计算每次结局的基本情况和整体胜率。
- 实现训练框架对评估函数进行训练。
- 参加中国象棋在线比赛，构建中国象棋在线比赛平台：[UCCI 中国象棋通用引擎协议版本：3.0](#)

## 6.3 五子棋

- 扩展到五子棋的游戏流程。
- 将 MaxMinAIPlayer 通用化，使得在五子棋等其他游戏中也可以直接复用。

## 6.4 卡坦岛