Readme Mankind TicTacToe

2016年4月4日 15:18

1. 简介
   a. 对比cpp的设计，添加了AI先手，以及部分残局处理能力
   b. 注意：AI在前两步中，必须由AI来完成其中的一步或两步，因为需要确定situation_x（先手时确定）或者 situaion_o （后手时确定）
   c. 参数介绍：
   - Int Situation_x situation_o 用来确定是预期中的哪种情况
   - Bool Player 用来确定X和O
   - Bool Flag 用来确定游戏是否结束
   - numArray 存放当前对局
   - Int chess_count 表示当前步数
   - Scope 旋转次数（当AI后手时候，人类放到了非预期位置，则需要加scope进入AI的预期情况）
   - Sum 存放各行、各列、对角的和
   - Static int[][] rotate 旋转前后的对应关系

   d. 情况定义
   - Kill
   - KO
   - Be_kill
   - Be_KO（不可能出现）
   - draw（已经无所谓了，怎么走都是平局了）

   e. 格子定义，分为3类
   - 角
   - 边
   - 中

2. 思路
   a. 将问题分为4类，1是AI先手，另3种是人类先手，根据先手位置不同，分为角、边、中心
   b. 全部情况可以在纸上用遍历解决，事先给予AI处理残局的能力
   c. AI先手时，既然由AI先手，走角落，因为根据预期，走角落的胜率更高
      i. Count == 2 时
         1) 对方拿了中，则放对角
         2) 对方未拿中，则放中，下一步可KO可kill，不再进入others
      ii. count == 4 时
         1) 正常情况下不会出现count==4而且进入others，为了保险起见，若进入了others，则random_move
      iii. count == 6/8
         1) 同count==4，若没KO没kill，则已经平局，random_move即可
   d. AI后手时，记situation_o = 1/2/3 表示 角/边/中心
      之后进入的则是count == 3/5/7的情况

i. Count ==1
   1) 经过分析，该情况优先抢中心，其次抢对角
ii. Count == 3
   1) Situation 1，仅在对方没逼迫的情况下发生，优先拿对角，若不可以拿，则拿边
   2) Situation 2，需要counter KO，若没必要counter，则拿边
   3) Situation 3，仅在对方没逼迫的情况下发生，拿角即可
iii. Count == 5
   1) Situation 1， 优先拿角，拿失败则random
   2) Situation 2 ，random
   3) Situation 3 ，random
iv. Count == 7
   1) Random

3. 代码审计
   a. 缺点
   ○ 大量if-else导致代码过长
   ○ 残局处理能力受限于situation_o是否正确获取到，进入AI预期的范围（只要人类不去通过点击鼠标替AI走某几步的话，AI是不会输的）
   b. 优点
   ○ 拥有处理残局的能力
   ○ 通过Scope、Rotate来将9种后手情况分为3类，更容易去处理，使用numArray[rotate[scope][0~8]]来表示查询AI看起来的布局；使用chess_scope（i）来表示AI看起来的布局所代表的人类看起来的chess；我认为使用Scope和route是一个非常机智的选择
   ○ 模仿人脑的思维过程，完美地构造了一个和人脑一样的if-else机器
4. 运行过程
   a. 略，AI从来不会输，偶尔会因为人类的失误而获胜
   b. 注意：AI在前两步中，必须由AI来完成其中的一步或两步，完成其初始化