不围棋实验报告

日期：2022 年 12 月 24 日

组员：杨子涵、范佳玥

一、代码架构

1、界面处理

图案绘制

基础功能

void Button(int x1, int x2, int y1, int y2, TCHAR* text)	绘制一个主菜单按钮
void black(int x, int y) void white (int x, int y)	绘制黑子或者白子，并加上作为上一步提示的红圈。同时将相应位置的 chess[x][y]值改为-1/1，将记录步数的变量 steps 值加一，并将这一步记入记录下子顺序的二维数组 memo
void draw_chess()	绘制渐变色背景，画出棋盘，并绘制右侧主菜单
int input_chess(void)	让玩家通过鼠标按键控制落子位置：读取鼠标数据，当按键在一格点附近区域时判为在该点落子，通过 regret 函数实现悔棋功能，确定下子后通过调用 ends 函数判断是否分出输赢，若已分出，调用 player_win/lose 函数执行相关操作。当处于双人对局模式时，若已经分出输赢，返回值 2 给 two_player 用以标识。落子期间同时允许操控主菜单任意键。
void start(void)	执行按下 start 键后的操作:左下角绘制黑白棋子图标，读取鼠标信息，让玩家通过按键挑选执棋颜色，将玩家与 bot 棋子颜色记入变量 playercolor 与 botcolor
void play(void)	人机对弈核心算法整合：通过变量 playercolor 与 botcolor 判断当前下子为何方。 若下子的为人：调用 input_chess 落子； 若下子的为机器：当未满足 algorithm 条件，调用 efficiency 函数获得最佳落子坐标并落子；当满足 algorithm 条件，调用 UCB 函数，遍历寻找 UCB 值最高的点进行落子，若已然处于必输局面，则随机落子。 机器落子后调用 ends 判断输赢，与 input_chess 中输赢相关操作类似。
void read(void) void save(void)	**存盘读盘**：按下 save 键将关键数据存入 txt 文件；按下 read 键，从 txt 文件读取数据并根据数据重新绘制出之前的棋局
int main(void)	BGM 以及界面整合：绘制主界面；**实现附加功能四：**用 mciSendString 函数打开并播放 MP3 文件；读取鼠标数据根据按键位置调用同名的 start(),save(),read(),two_player() 以及 play() 等函数执行相关程序

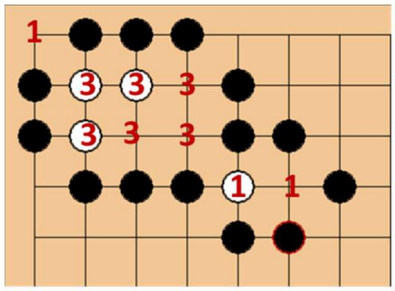
附加功能	void two_players(void)	**实现附加功能一：双人对局**——在左下角绘制黑白棋子图样，通过红圈表明下一步棋子颜色。调用 input_chess 函数轮流下棋，当返回值是 2 时绘制分出输赢的标识
	void rev_last(void)	**实现附加功能二：上一步提示**——当下完新的一步棋后从二维数组 memo 中读取上一步棋子所在位置，并消去棋子周围作为提示的红圈
	int regret(int x, int y)	**实现附加功能三：悔棋**——读取鼠标数据，当在 confirm 框内按键时继续游戏；当在 second chance 框内按键时抹去之前的棋子，同时将 chess 二维数组、变量 color，step 回退至前一步，并重新等待玩家下子
	void player_win(void) void player_lose(void)	显示人机对弈分出输赢时的提示，即结束界面
2、棋局判定	bool whether_end() int ends (int x, int y)	同为判断是否分出胜负的函数，whether_end()遍历全局，ends 则针对刚下完的一步，调用 notsuicide 与 legal 两个函数判断其是否是自杀或者是不合法的
	bool border(int x, int y)	判断棋子是否在棋盘内
	bool air_exist(int x, int y , int copyX, int copyY)	判断该点还有多少气
	bool legal(int tx, int ty)	判断该点下子是否合法——即第一步不可在正中央，该点不能有子
	bool notsuicide(int x, int y)	调用 air_exist 与 border 两个函数，判断该点下子是否属于自杀——即不可让下子　后该店没有气或者让四周的点没有气
3、AI 算法*	int link(int x, int y, int copyX, int copyY)	递归数附近己方连起来的棋子数
第一阶段	int count_point (int x, int y int z)	判断我/对方占该点后，增加了多少个自己点+减少了多少对手点，从而判断每一步棋多“划算”
	void draw_map()	找出我/对手能走的位置
	void efficiency()	引用以上函数，得出确切落子位置。
切换	int algorithm(void)	算法切换边界判断：当前可行的格数小于等于 7 时从步法搜索转入 UCT 算法
第二阶段	void copytosim(void) void copytochess(void)	两个函数实现 chess，simchess 之间数据拷贝，用于 MCTS 的回溯
	void calculate_nj(int x, int y)	计算模拟的盘面有几个合法且非自杀的落子点
	void playout(void)	递归，回溯，模拟棋局的进行直至分出胜负
	void UCB(void)	遍历计算下一步有几个合法且非自杀的落子点，并且调用 calculate_nj 与 playout 函数计算胜率，以此计算每个子节点 UCB 值，对于盘面进行评估

*详情请见 AI 逻辑部分

二、算法逻辑

第一阶段：步法搜索

由于前期合法落子点数目较多，这里使用了一层步法搜索。



上图所有黑子的气为三个黑包围圈外的总格点数。

将落子后己方可行格点数增加，对方可行格点数额外减少视为划算的。遍历全局，寻找对自己和对对手最划算的落点位。两相比较，选择下己方最优位置，或抢占对方最优位置。此处截枝：若是划算的，四周必产生气为一的格子，形成一方可下，一方不可下的局面。

气：在对手棋界（被对手的棋或边界四周框住）内剩余多少空点（包括自身）

若对敌我两方都不存在划算的落点位（即在任何合法位置落子都无法使得落子方增加可行点，对方额外减少可行点），则统计棋盘上每个空点的四周空点数（2 ~ 4）优先在空点数小处四周可行位置落子，从而趋向产生气为一的格子。例：空棋盘时，会选择角落如第二行第一列处落子。

若空点数一致，则比较附近己方连起来的棋子数，连接棋子数多处优先。将己方棋子串联，不宜被对方分散，有利局势。

bool border	判断边界
int air_exist	递归数气：对手棋界内剩余多少空点（包括自身）
bool legal	输入合法 与 不自杀
bool notsuicide	
void draw_map	找出我/对手能走的位置
int count_point	判断我/对方占该点后，增加了多少个自己点+减少了多少对手点，从而判断每一步棋多“划算”
int link	递归数附近己方连起来的棋子数
void efficiency	引用以上函数，得出确切落子位置。

第二阶段：UCT

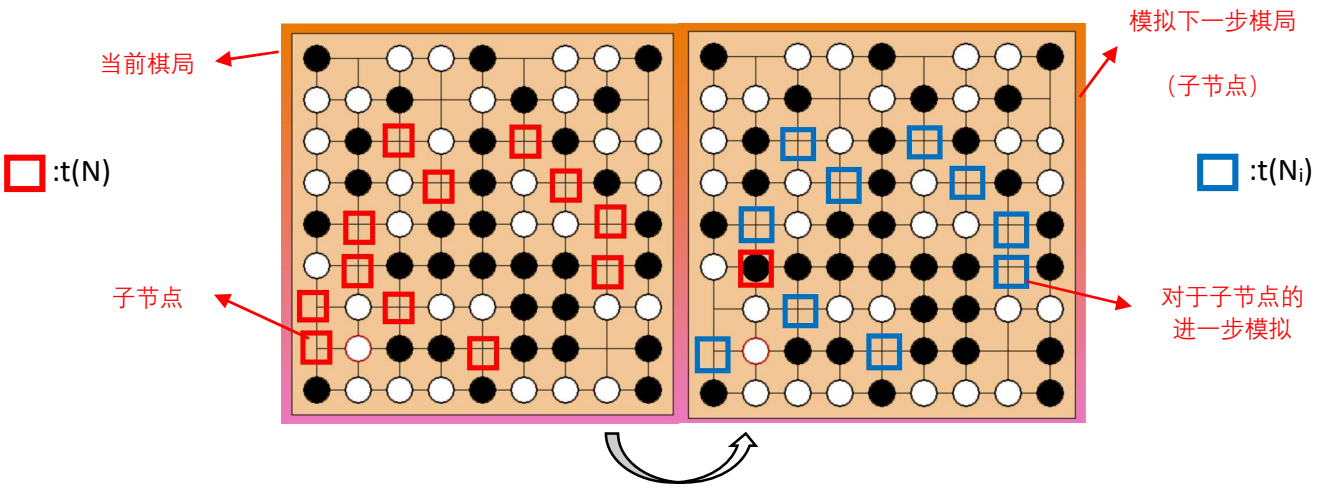
合法落子点数目小于等于七(有函数 algorithm 判断)，进入第二阶段。第二阶段中主要运用了上限信心界树搜索（UCT）算法——UCB 算法与蒙特卡

$$I(N_i)=\overline{X(N_i)}+C\sqrt{\frac{\ln t(N)}{t(N_i)}}.$$

洛方法的结合，通过 UCB1 公式（图一）对于盘面进行评估。

N 是给定节点，即当前局面，N_i 则是 N 的一个子节点，即模拟的一个可能盘面，t(N)是对于 N 进行模拟的次数，$t(N_i)$ 是对于节点 N_i 进行模拟的次数，$X(N_i)$则是 N_i 盘面的平均收益值，即对应盘面的胜率[2]。而 C 则是给定的常数，当 C 大于零时已被证实可以通过将节点遍历次数纳入考量提高不围棋的胜率[1]。算法通过该公式对于模拟的盘面进行了评估，通过寻找 UCB 值最大的模拟盘面确定下一步落子位置。

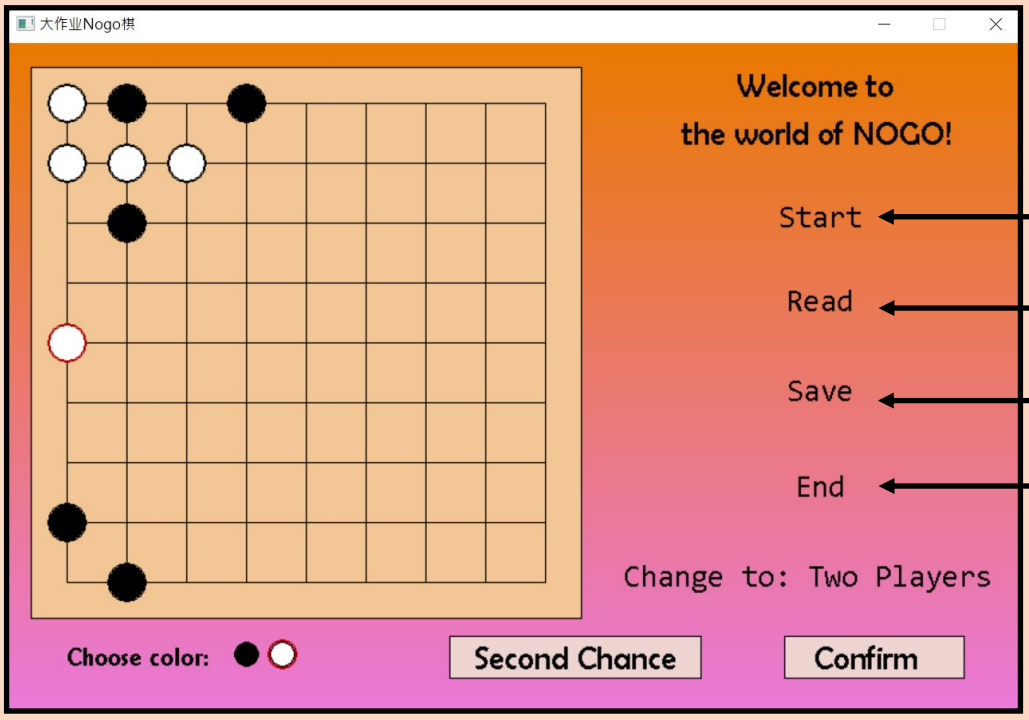
本次实验中用 calculate_nj()函数计算当前盘面下子节点模拟的次数，即下一步所有可能盘面的个数，playout()函数不断进行模拟直至分出胜负，UCB()函数则利用 playout()所得模拟结果以及子节点模拟的次数计算得到该子节点胜率，又通过计算当前盘面下所有子节点个数 N 最终得出各个子节点的 UCB 值，最终选择 UCB 值最大的点进行落子。



[1] Chou, C. -., O. Teytaud, and S. -. Yen. "Revisiting Monte-Carlo Tree Search on a Normal Form Game: NoGo." *Applications of Evolutionary Computation, Pt i*. Edited by C. DiChio, et al. vol. 6624, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[2]Jun Tao,Gui Wu. Application and Design of Advanced Algorithms in NoGo Game of Computer Games[C]//.第 28 届中国控制与决策会议论文集（中）.[出版者不详],2016:1595-1598.

三、游戏介绍



基础功能

开始游戏（人机）

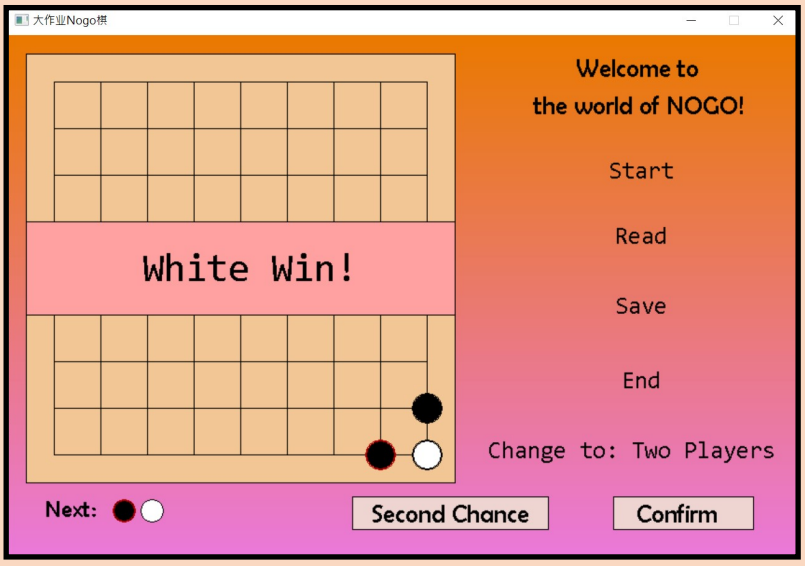
读档

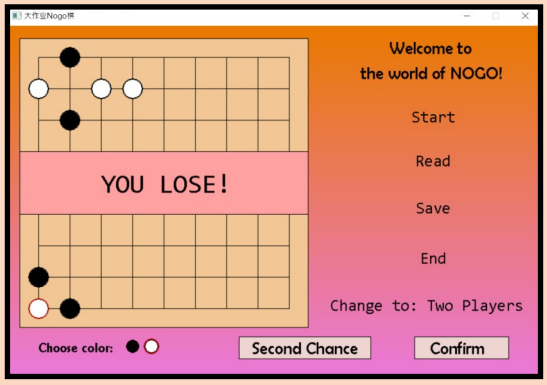
存档

结束游戏

附加功能： BGM、上一步提示、悔棋、双人对战

- 一开始玩家可选择执黑/白（左下 Choose color）
- 棋子上红圈提示上一步
- 玩家通过鼠标按键落子，按 Second Chance 可悔棋重落，按 Confirm 继续
- 输赢显示例子如下





双人对战

- 落子颜色提醒（左下角 Next： ）
- 支持存档、读档等所有基础功能
- 支持悔棋等所有附加功能

BGM： vide cor meum