# AI五子棋实验报告

-2023080901009邓皓天

## 实验目的：

1.通过c语言编写程序，辅以raylib图形库，实现ai五子棋人机对战游戏

2.实现五子对战的各种功能，包含画棋盘，落子的函数，判断游戏是否结束的函数，判断哪一方取得了胜利，对落子点进行估值的分数表函数，对落子点进行往四个方向进行遍历的函数，对落子点进行打分的估值函数，对整个棋局进行评分的评估函数，寻找最佳落子点的极大极小值搜索及alpha-beta剪枝的博弈树函数，悔棋函数。下面将详细赘述。

3.通过各种数据结构与算法的使用，完善了游戏的功能，增强玩家的互动性，增强ai的算力和智能性。

## 流程图：

## 

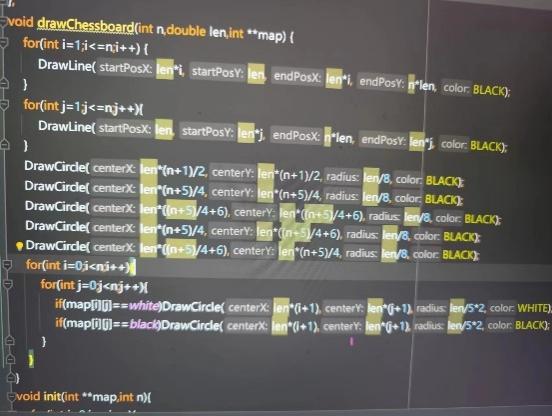
玩家通过鼠标选择是否先手

将坐标传递给绘制棋子的函数

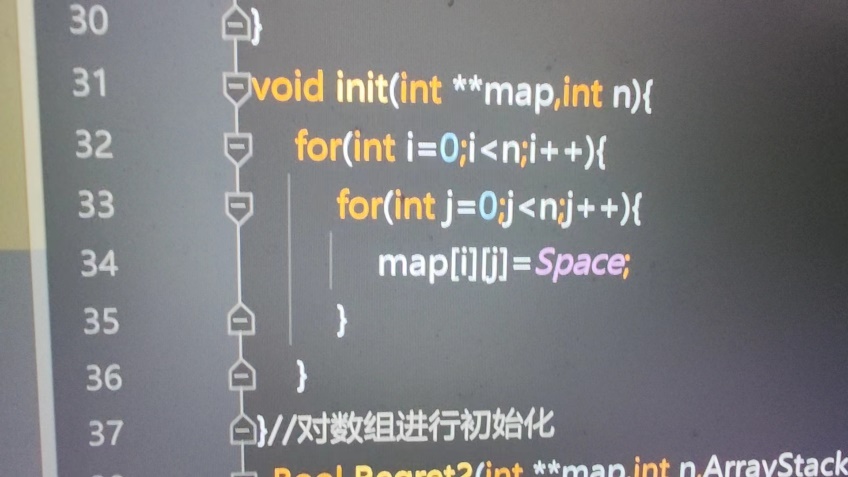
玩家通过鼠标选择是否先手

## 实验过程：

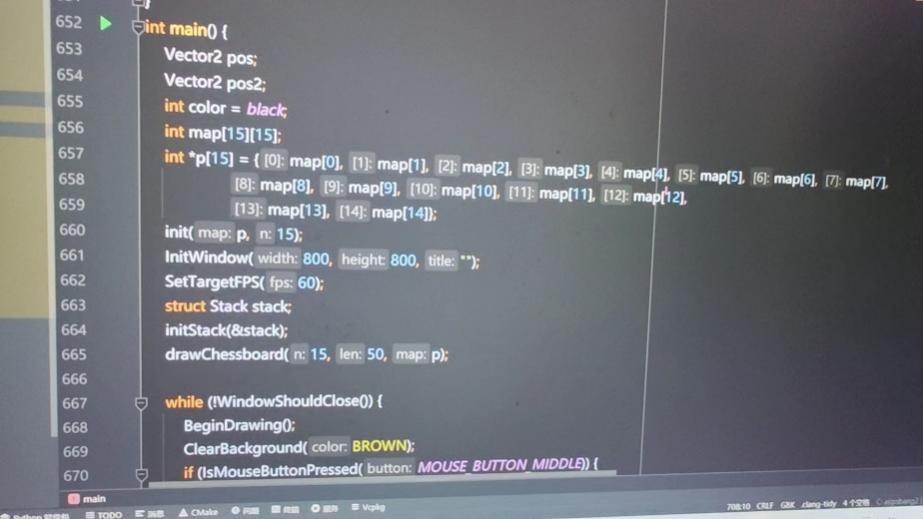
1. 首先利用线与线的组合，加上几个标致点，形成了基本的棋盘，将该棋盘的坐标与含枚举类型数据的二维数组的元素一一对应，遍历整个二维数组，实现画棋子的功能，枚举类型分别为Space，black，white。



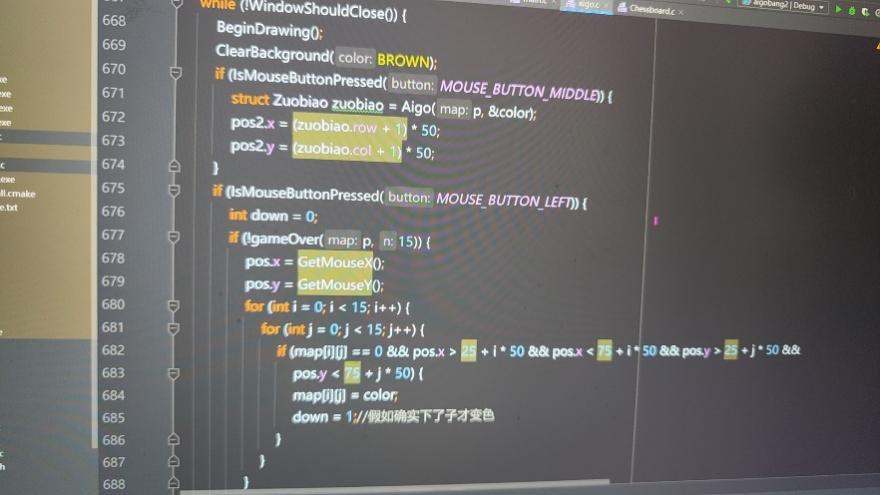
1. 为了防止数组出现枚举类型以外数据的情况，需对数组进行初始化。



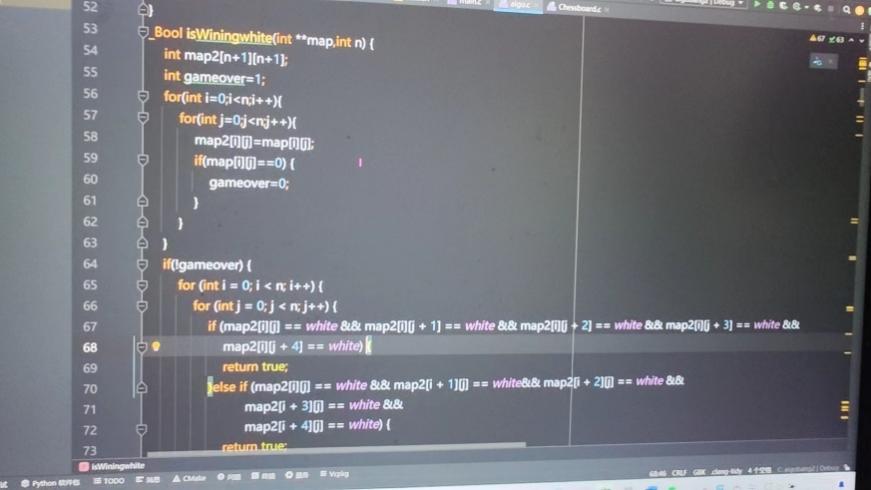
1. 创建一个窗口800\*800的窗口，设置帧率为60fps，初始化二维向量栈（之后的悔棋函数会用到），对二维数组初始化，让指针数组获得二维数组的地址



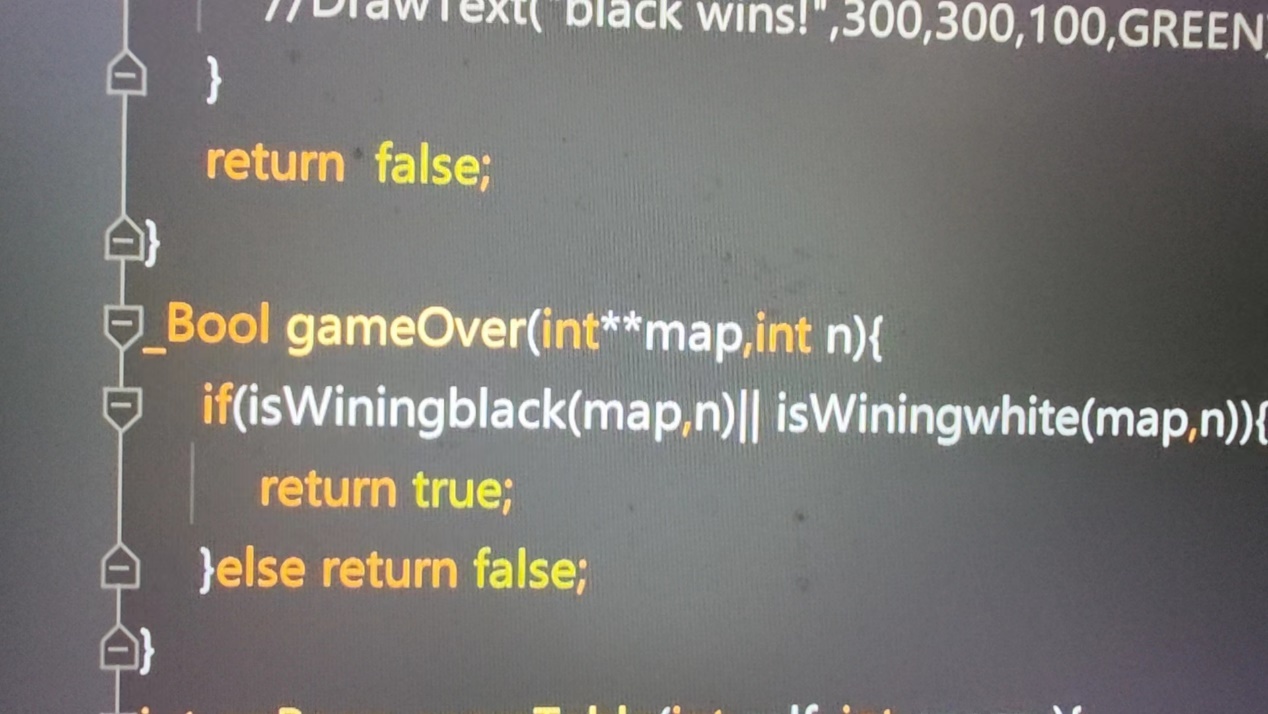
4.设置背景色为棕色，假如鼠标中键点击，让AI先下，否则玩家先下，玩家通过鼠标左键来落子，并且用二维向量栈储存好AI和玩家的落子点坐标



5.为了实现真正的有输有赢，我们需要有一个判断胜负的函数，以下以白棋为例，黑棋同理

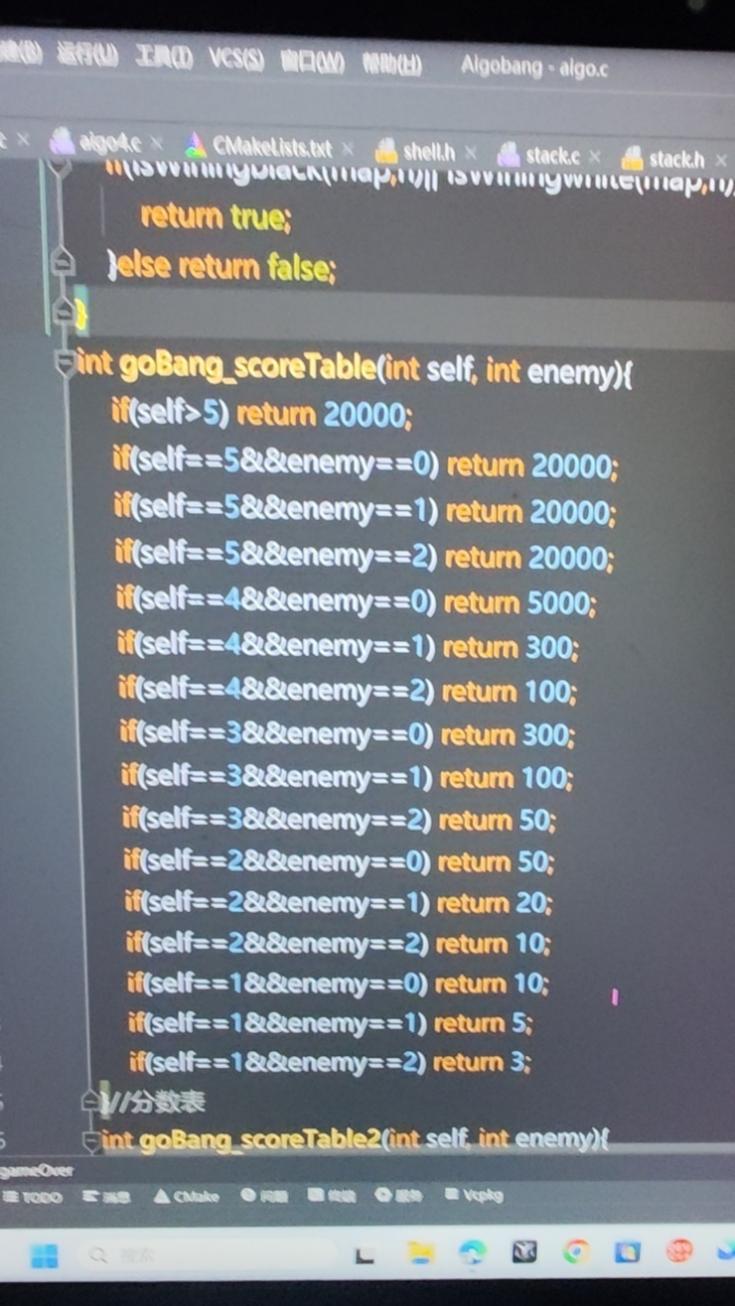


6.还需要一个函数统合两个输赢函数，方便之后游戏结束的判断和使用

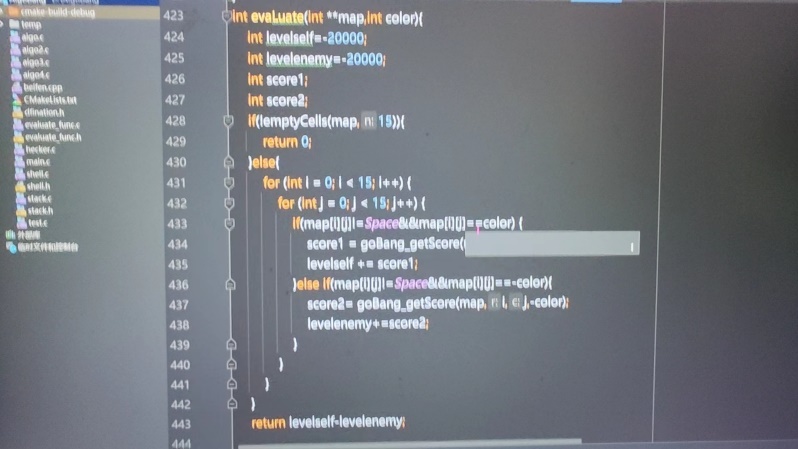


7.落子点的分数表函数，分别遍历左右，竖直，左斜，右斜方向的棋子，如遇到自己本身就令self+1，如遇到空白或对方棋子就停止遍历，并且遇到敌方停止的话也让enemy+1，

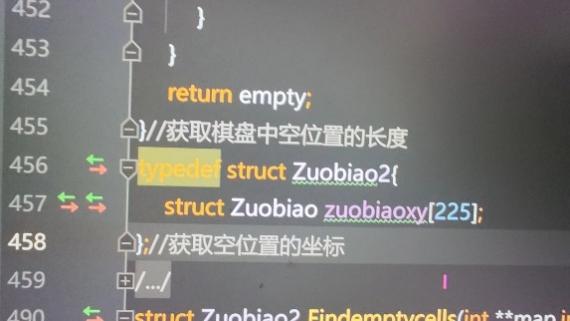
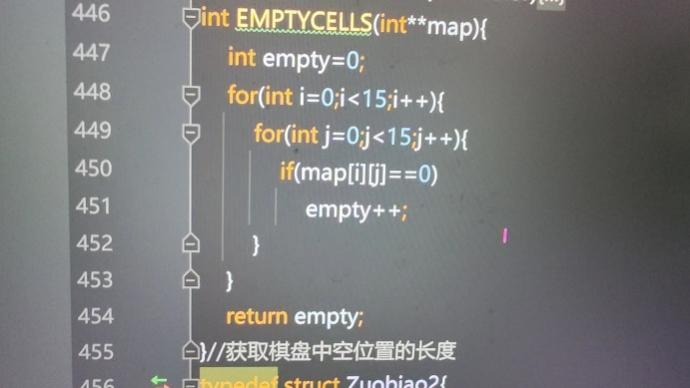
这样就得到了活五到活一，眠五到眠一，死五到死一等情况的分值，值得眠四，活三，活四与其他眠三，死三，死四情况的分值差距较大，让棋子会优先形成活三，活四，眠四



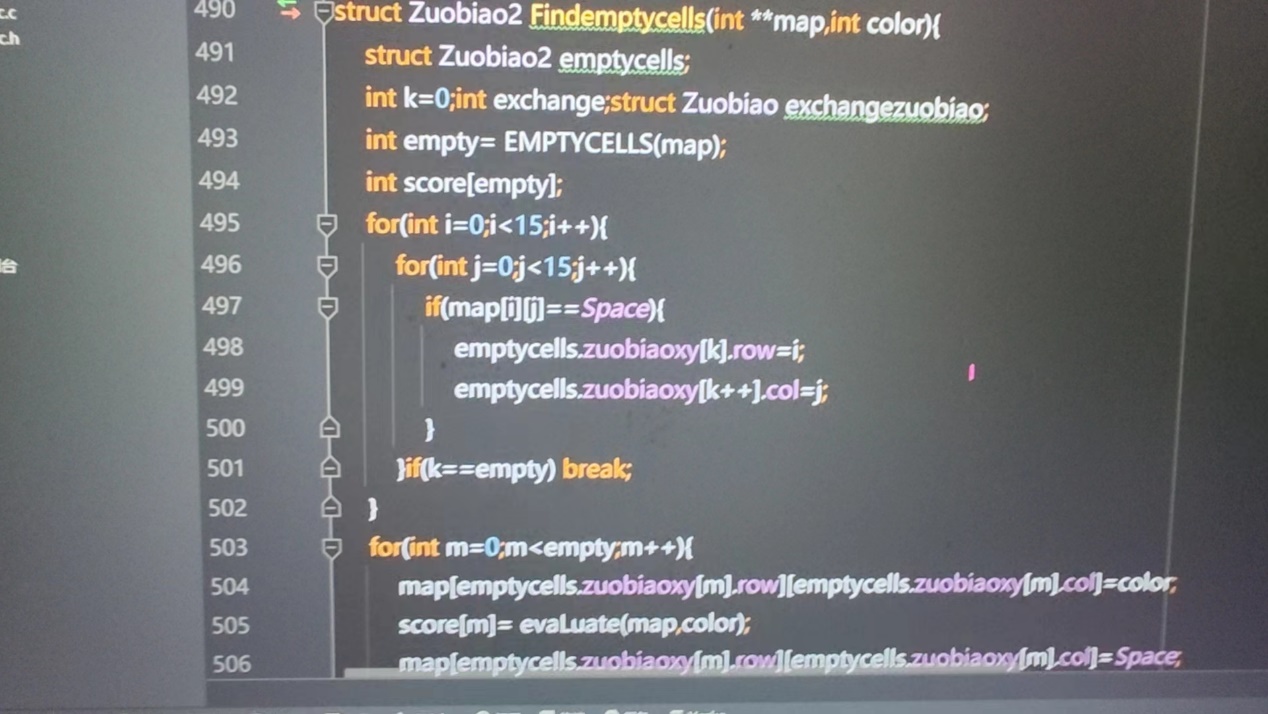
8.获取到某个点的分数后，我们还需要对整个棋盘的所有本方的落子点进行估值打分，以便得到博弈树根结点的最佳局势，采用对自身和对方的局势分别打分进行相减的方法来得到局势分



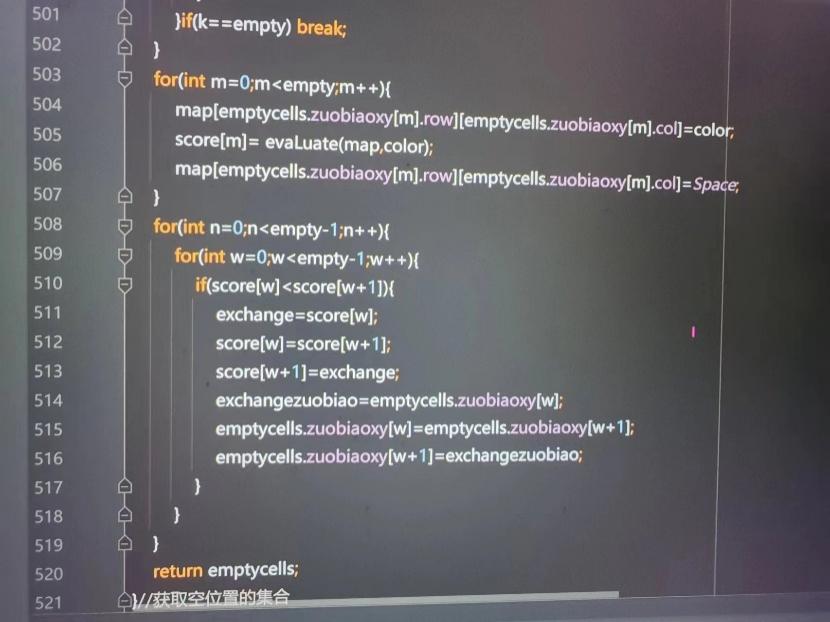
9.为了减少遍历过程中的循环嵌套和多余的空位置判断的问题，我们可以通过函数获取空位置和获取空位置的多少来得到一组储存了所有空位置坐标的结构体数组，结构体数组为一个储存了坐标的数组



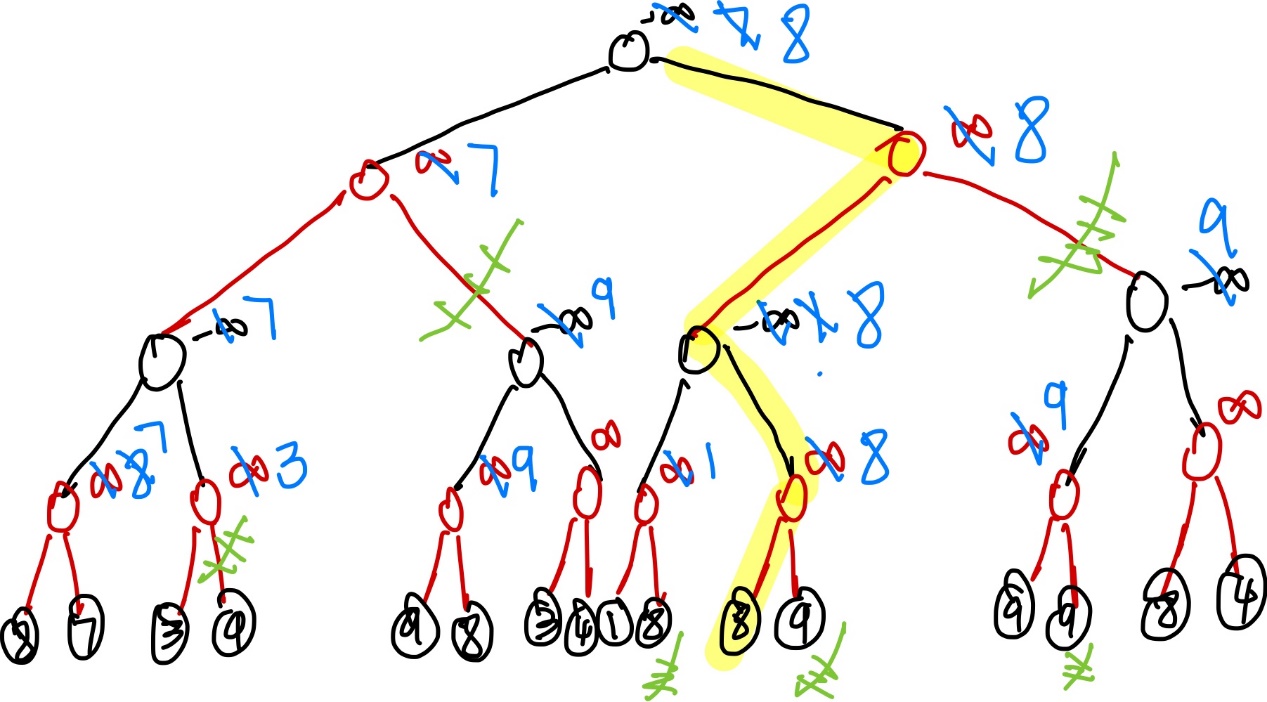
10.找到所有的空位置，储存在结构体数组里面



11.对所有空位置进行打分，打分后按照从大到小的顺序排序，由于排序速度对整个程序来说影响不大，此处采用了最简单的冒泡排序



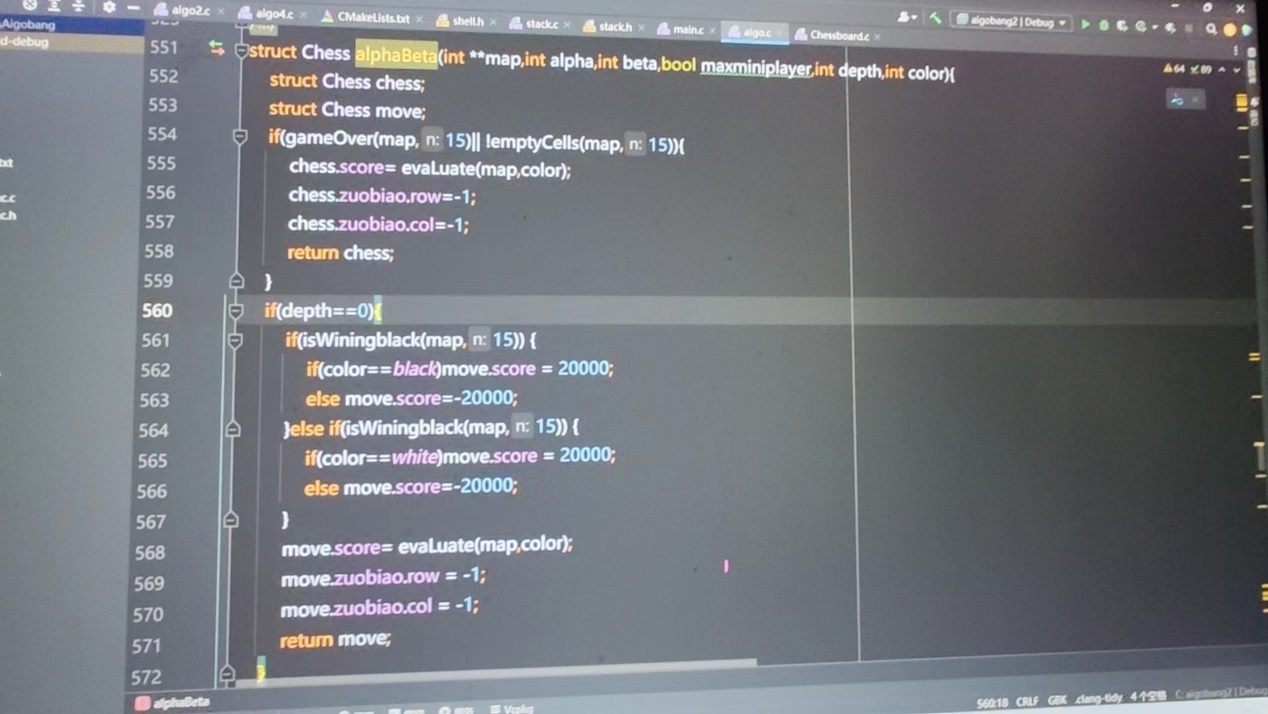
为了方便理解，我们先来看看博弈树的maxminisearch和alpha-beta剪枝的原理



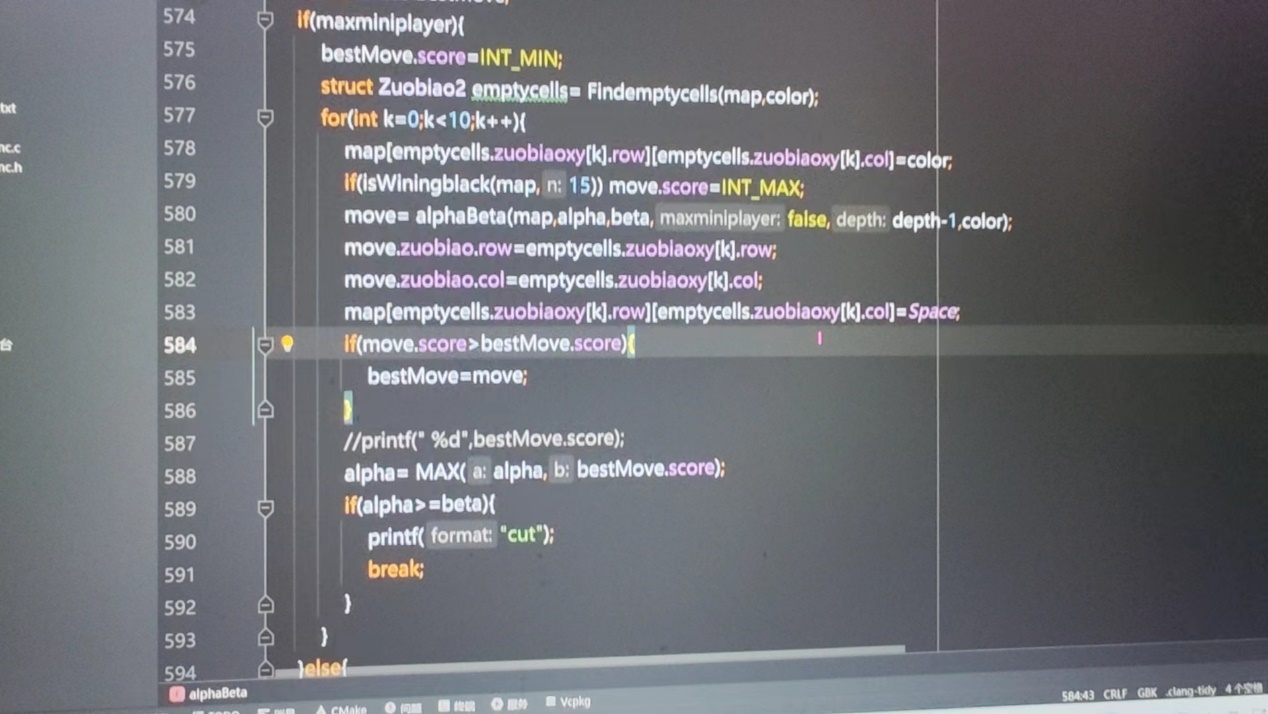
以上图为一个博弈树，此博弈树的alpha层为黑色，beta层为红色，以上演示每个父节点只有两个子节点，且只递归到4层的情况，所有alpha层一开始赋值为INT\_MIN;beta层一开始赋值为INT\_MAX,进行了四次拟落子后得到了最下层的局面分数，从最下层开始，beta层搜刮比它小的子节点的值，并且发现子节点的值比自己大时直接剪掉往下继续搜索的过程，alpha层则搜刮比它大的子结点的值，当发现子结点值比自己小时直接剪掉往下继续搜索的过程，这样就能实现搜索效率的提高，避免了不必要情况的出现。搜索完后，根结点搜刮到的值为四步以内假设对方和自己做出最佳落子而对当前这颗子最有利的情况（即四步以后对自己情况最有利的·情况）。相当于预测了四步。

12.最关键的博弈树函数，可以获得拟落子后最终局面最佳的落子点，此处返回值为一个含有分数和坐标的结构体，传入参数，数组用于拟落子，alpha，beta用于不必要情况的剪枝，

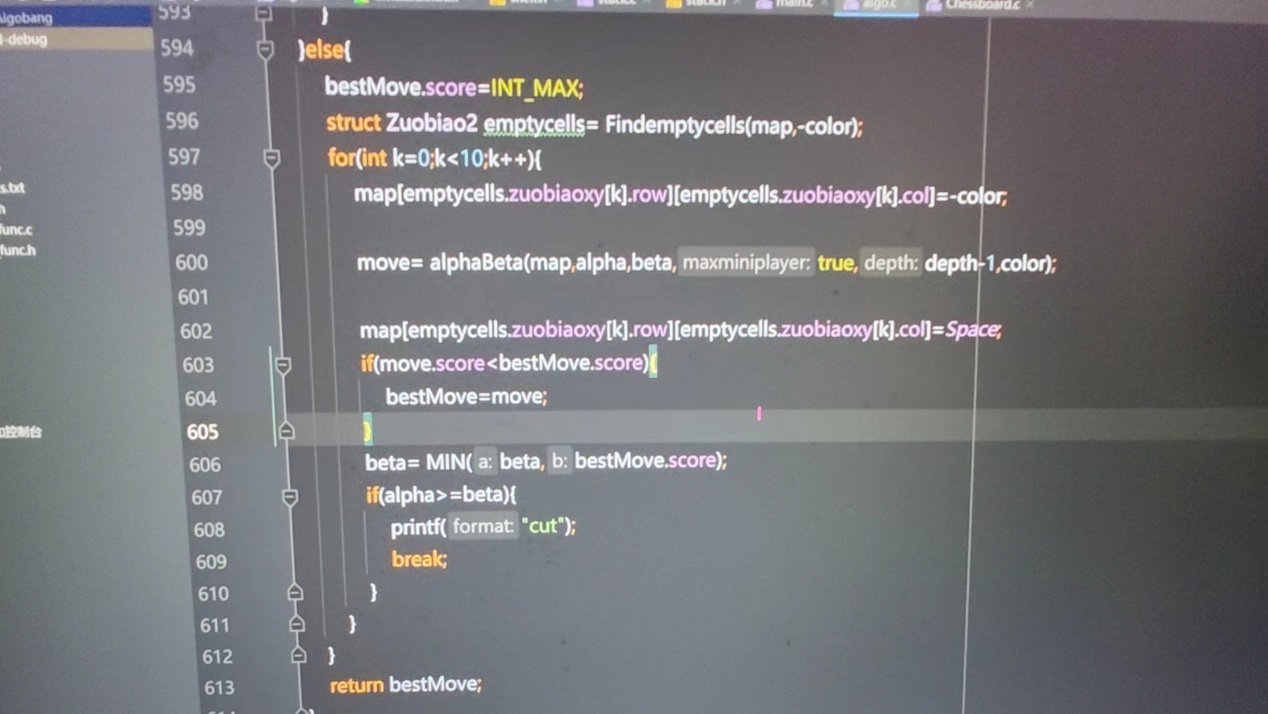
depth用于限制递归深度，bool类型的maxminiplayer用于控制棋手的转换，和递归实现博弈树，当比赛结束且本方赢的话返回一个很大的分数，敌方赢的话返回一个很小的分数，如果以上情况均没有，则返回一个局势分，不需要返回坐标，固此处返回-1，-1



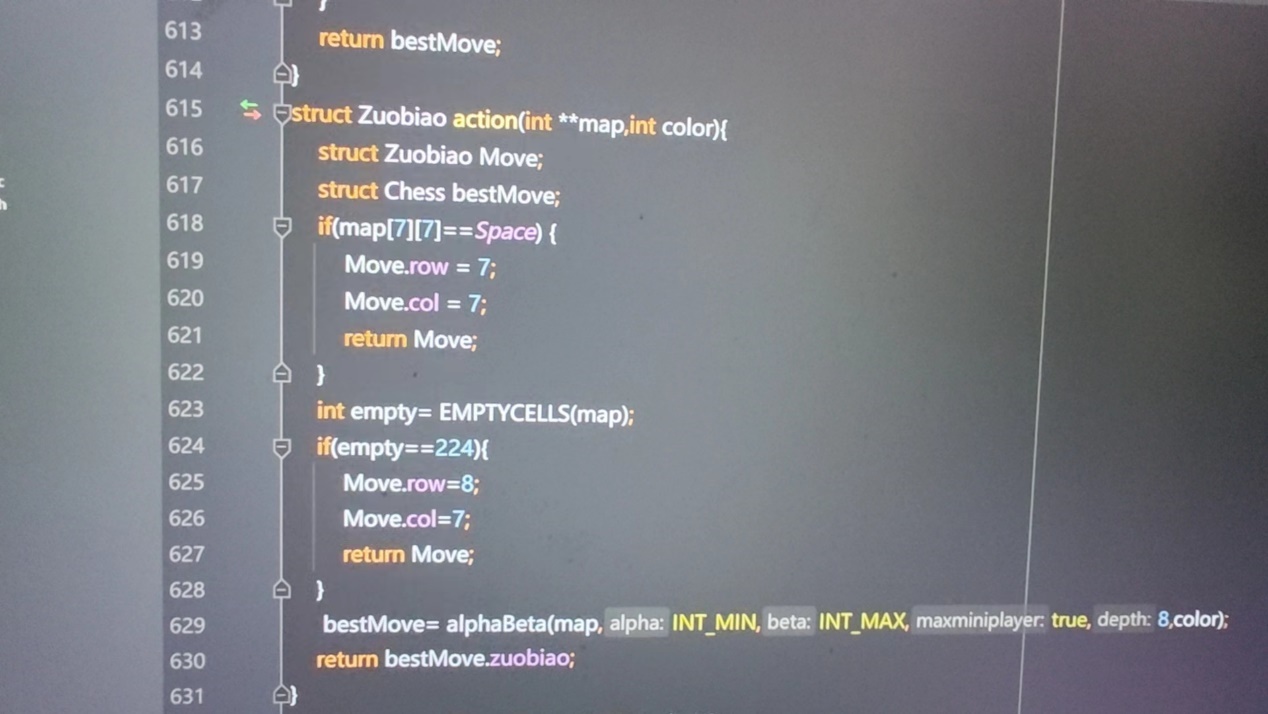
13.对于alpha层，先让alpha层一开始的分数为最小，并且找到分数最高的十个空位置进行拟落子，每次落子评估一下分数再撤回这颗子，alpha如果从子结点找到一个更大的分数，就搜刮它的分数作为自己的分数，如果发现下一层的结点分数比自己还低，则直接弃掉这个结点往下搜索过程（剪枝），最后返回一个含坐标和分数的结构体给上一层



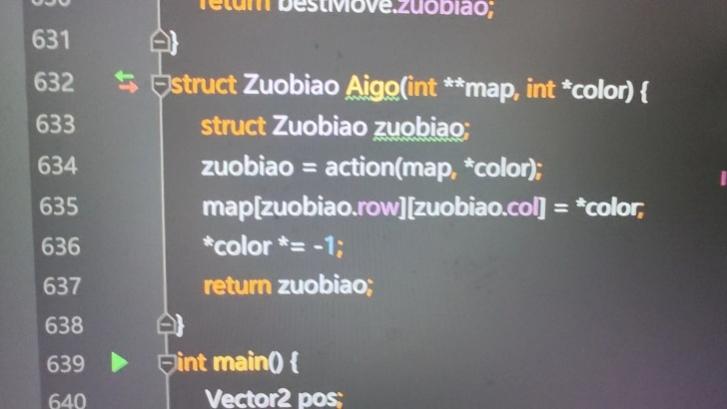
14.同理，beta层则是一开始分数为最大，搜刮比自己分数小的子结点，遇到比自己分数大的子结点直接剪掉，并且会返回一个含坐标和分数的结构体给上一层



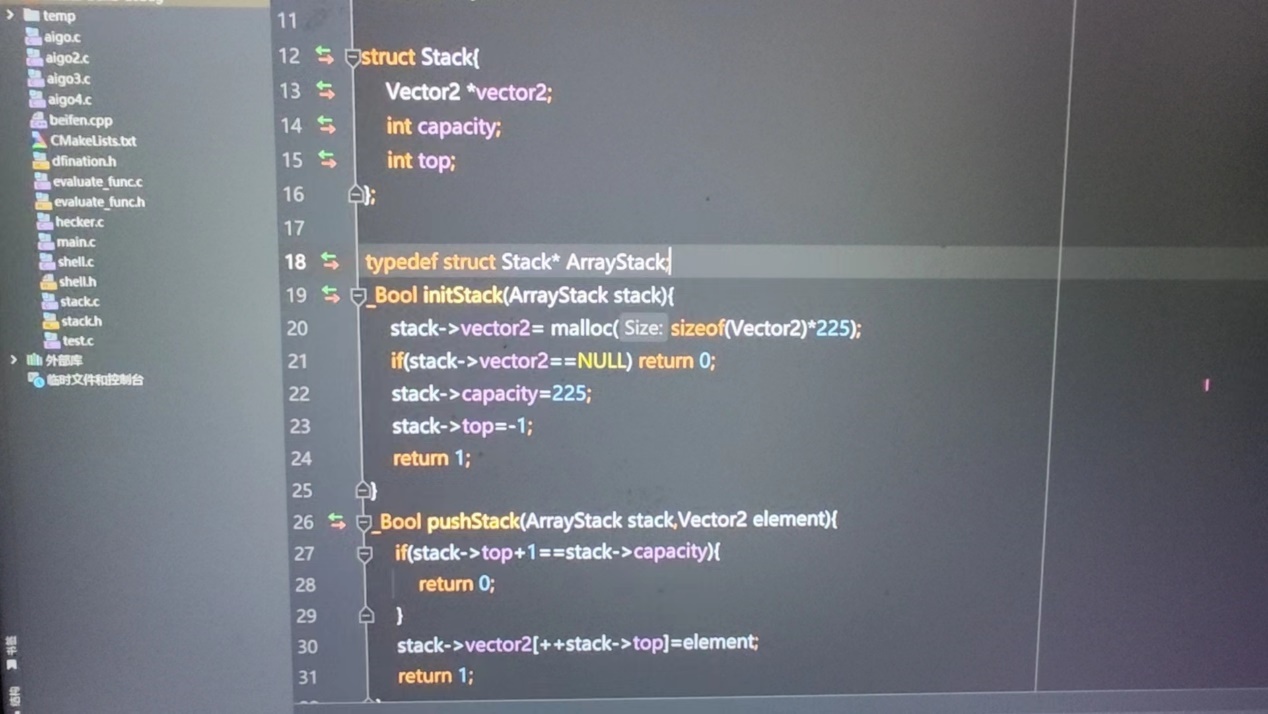
15.通过一个函数简单的定义下第一步的下法，由于一开始评估函数得到的所有点都相同防止落子时会任意下到搜索到的第一个点，让第一个点下到中心或中心附近



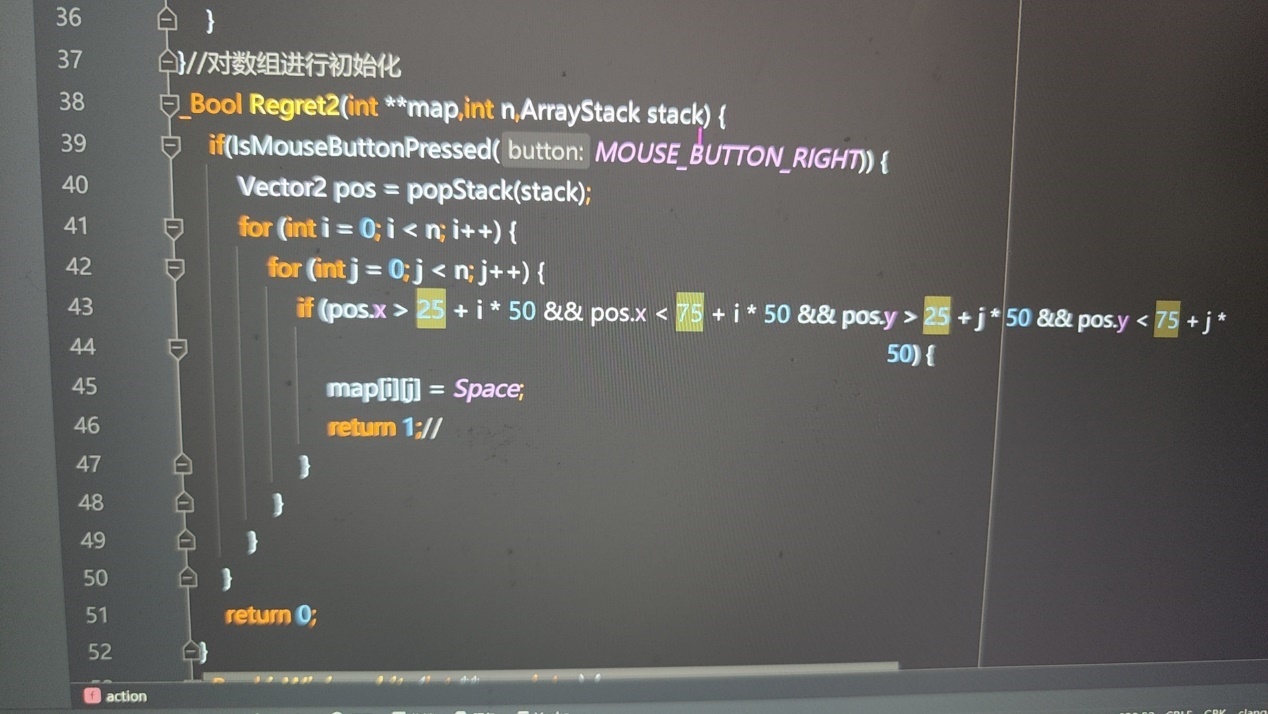
16.控制AI的函数，把染色板地址传入，令其落子后改变染色板的颜色，以便玩家落子



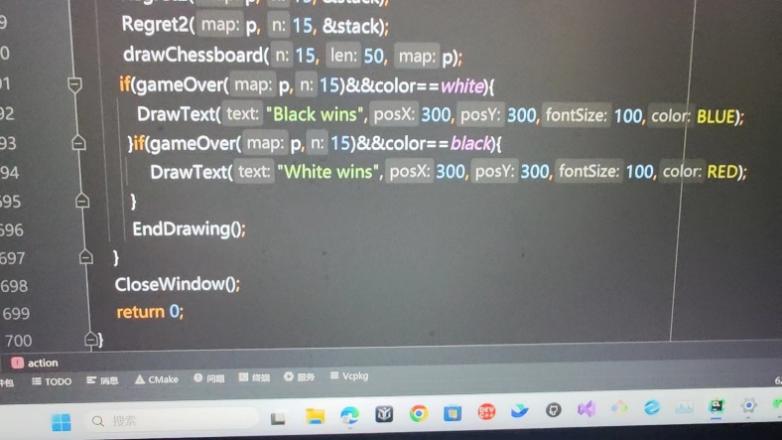
17.为了实现悔棋，此处加入了raylib库自带的结构体Vector2二维向量作为栈的数据类型，并且申请了一段足够储存棋盘所有点位置的内存

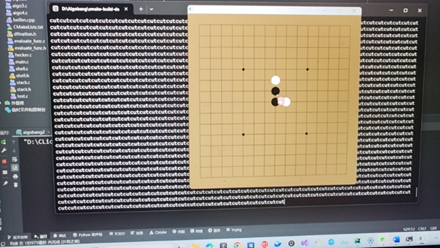


18.悔棋函数的实现，把之前落子后按照后进先出的顺序进行出栈，并对出栈后的坐标对应的位置进行撤子，并且为了公平，每次悔棋都悔两颗，回到悔棋前的状态



20.为了方便知道是谁赢了，在主函数末尾，把赢家呈现于窗口中，代码与效果如下

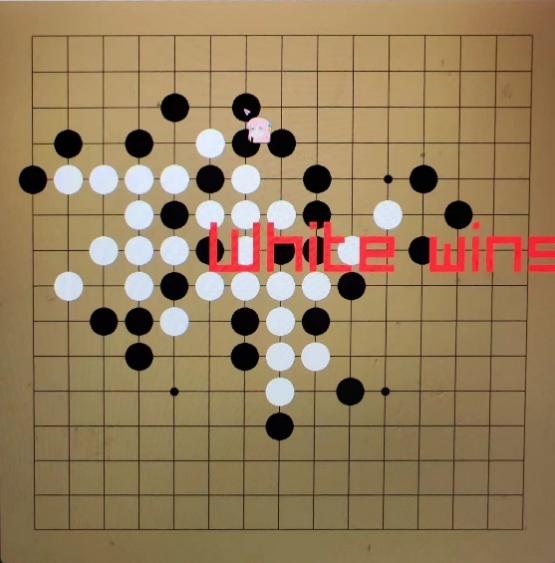


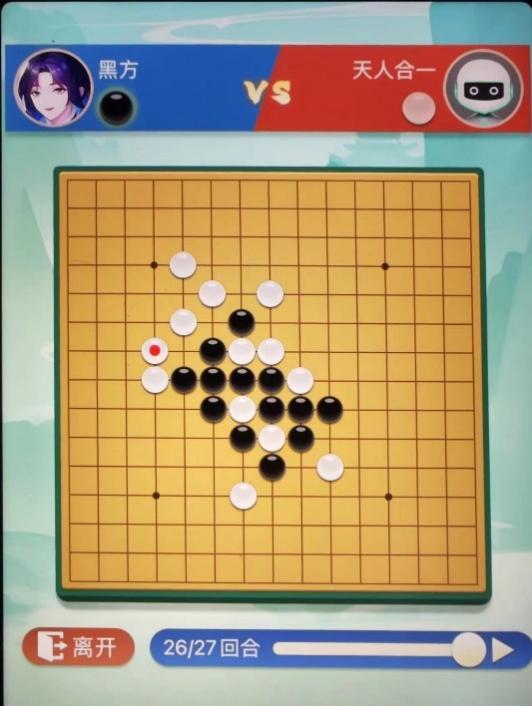


## 实验结果：至此，我们已经完成了AI五子棋人机对战的游戏制作，目前测试来看比一般人要强，采用十个根结点时递归层数可以达到8，几秒内出一颗子，采用6个结点时递归层数可以达到12，其中alpha-beta剪枝确实生效成功把“printf（“cut“）；“语句实现，而且能够越下越快，由于本人找不到棋力较强的棋手来测试，难以赢下AI，遂找来Apple Store上一款名为五子棋的游戏上的最高难度AI “天人合一“ 进行了对战。结果如下，可以看到我的AI分别在先手和后手的情况下赢了”天人合一“（此处采用搜索层数8）



## 先手时27步完成胜利，后手时54步完成胜利





## 本人还与同班同学进行了模拟对战，先手落子的情况下胜率极高