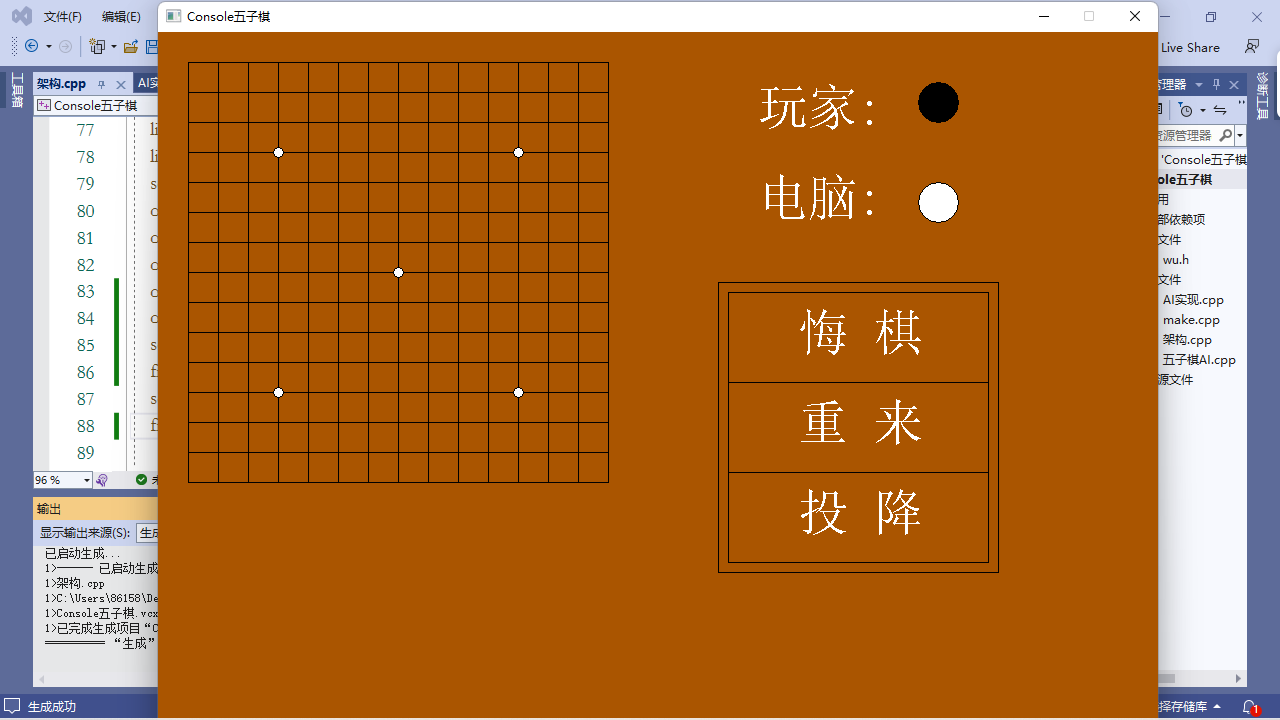
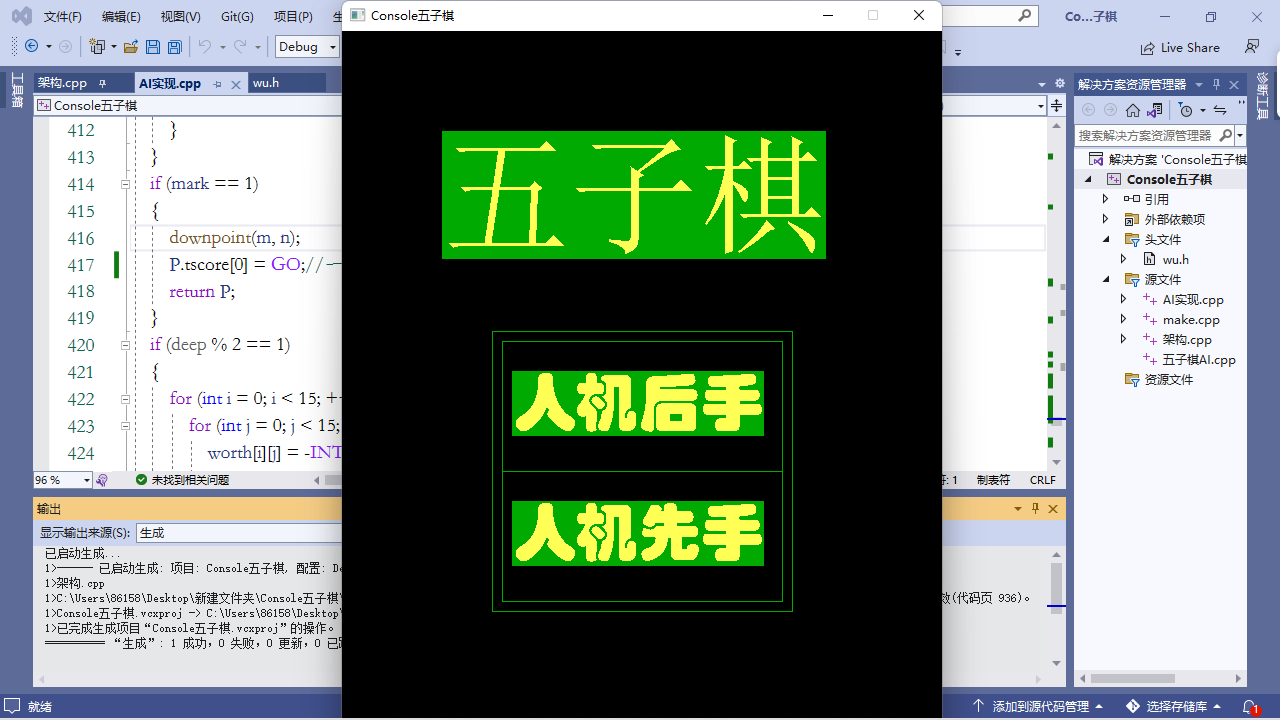
**五子棋实验报告**

**徐瑞 2022080905021**

1. **实验目的**

编写一个可以与玩家对战的五子棋程序，并测试其棋力。

1. **实验主要内容**
2. 基于easyx函数库进行开发，将程序文件分为三个部分，分别用来存放运行的框架，AI实现的过程，以及头文件。
3. 五子棋游戏分为人机先手以及人机后手，在下棋过程中有实现悔棋，投降，重来等功能
4. AI的实现基于极大极小值搜索，以及α、β剪枝算法，在估值函数的帮助下以递归的方式选出落子点。加入搜索已有点周围7\*7范围空点在估值后进行排序的函数来提高速度。

剪枝就是如：当A-MIN层选出最小值6时，B-MIN层在搜索过程中得到一个-6，因为MIN层上的MAX层要选取最大值，B-MIN层最后取值不会超过-6，所以可以不必继续搜索。

MAX层选取下一层的最大值，MIN层选取下一层的最小值

B-MIN

A-MIN（6）

1. AI实现过程中估值函数起到十分重要的作用，这里我将棋子符号化，按不同排序赋予不同数值。如将白子看作1，黑子看作2，空点为0，边界为3。如022230便是一种棋形**。**
2. 一切操作通过单击鼠标左键进行。通过使用easyx中定义的保存鼠标消息的结构体变量ExMessage来获取鼠标的输入信息。

yes

获取鼠标信息

判断左键是否按下

接着读取鼠标左边信息x,y

根据鼠标的位置调用相应的函数，若所在位置没有便不作反应

落点

根据棋盘格的间距对坐标进行修饰确保落点

**三. 程序评估**

1. 在我测试的过程中，AI可以和我下得有来有回，若我疏忽大意，有可能会输。其棋力可以等同未受过训练的成年人。

2. 在4层搜索的情况下程序运行的较为缓慢，如若再加深，AI落子的时间会提到1分钟一颗。而且我将周围空点的局势估值做了排序，选取其中靠前的几个点进行搜索，从10颗到8颗的改变对速度提升不明显，直到四颗时才有了显著提升。选取的点越少运算速度越快，棋力越低。

3. 有时AI落点十分奇怪，不知是我棋力底，无法理解，还是估值函数有问题。

4. 到后面有时会出现玩家有四颗连一起但AI不会去堵的情况。

**四．反思与总结。**

* 1. 程序运行比较缓慢，使用了一定优化方法后无较大提升。
  2. 估值函数还不够优化，对于不同棋形究竟应该怎么赋值还不太清楚，在反复摸索后对于棋力的提高收效甚微。
  3. 部分功能不够模块化，夹杂在了无关函数中。
  4. 到后面有时会出现玩家有四颗连一起但AI不会去堵的情况，这个应该是估值函数有点粗糙的原因。
  5. 一些基本知识还不扎实，比如break与continue在函数中的具体应用还不是很清楚，时常弄混。
  6. 不会在较大程序中的调试。比如像是在五子棋中有递归，程序走得会比较远且比较久的，我不能将程序跳到我想要的位置，即使用断点来到我期望的位置后，继续调试又不知道该怎么做。这给我的改进工作添加了很多麻烦，以后要多去学习调试的知识。

**五 . 改进展望。**

1. 可以尝试不同原理的估值函数，研究其中的不同。
2. 可以加入算杀模组（暂时没搞懂）。
3. 将程序更加模块化，做到一个函数实现一个功能。

**参考：livingsu** <http://t.csdn.cn/bDNrJ>

**Hik**。 http://t.csdn.cn/l4X9K