已完成工作

完成了五子棋的界面，实现了五子棋的基本逻辑
明确了极大极小算法与相关优化方法
蒙特卡洛算法的初步研究：与极大极小算法的区别、在AlphaGO中的应用

1. 五子棋的界面：

目前程序分为四个类与接口：Board、BoardConfig（接口）、Computer、Main

1.1 Board：

五子棋的面板类，继承自JPanel，实现下棋功能。此类中定义了两个成员内部类：按钮监控类与面板监控类，分别用于按钮的动作监听与面板的鼠标监听。棋局的状态通过面板上一个带滑动条的文本域显示。此文本域默认位置为跟随光标的位置。此面板被JFrame包裹。
（1）当棋手下一步棋时，会另开一个Computer类线程，完成电脑的判断，并通过改变面板类的二维数组，实现电脑落子。
（2）判断五子棋的输赢情况。当存在五子相连时，程序会进行胜负判断显示在文本域上，并不再允许落子以及不允许电脑线程启动，并将胜负标记设置为true（分出胜负）。如果棋局未分出胜负，将不做任何处理。
（3）实现重置棋盘功能。当点击重置棋盘按钮时，胜负标记会被重置为false（未分胜负），存储棋子的二维数组被清空，先手标记会重置为先手方（先手方参数可在BoardConfig接口设置，可选项为BLACK或WHITE）并且如果电脑线程已经启动，将会停止电脑线程。最后清空文本域，并在文本域输出已重置信息。
（4）解决落子冲突。当电脑进程在启动过程中，如果检测到棋手点击棋盘范围，将不产生落子动作。并在文本域上显示提示信息。

1.2 BoardConfig接口：

设置五子棋的参数。可调整的参数为棋盘大小、棋盘的单位格子大小、先手颜色、棋手与电脑执棋颜色。除此之外，其他参数都以相对长度计算。调整参数的目的是在编程时调试程序，实际运行过程中，将不调整任何参数。（其他参数包括：面板大小与位置、按钮大小与位置、窗体大小与位置、棋盘的背景颜色）

1.3 Computer：

电脑进行判断的类。此类实现了Runnable接口，因此可作为线程启动。为了在Computer类中调用Board类中的成员，将Board类作为Computer类的成员变量传入。在电脑类中定义了蒙特卡洛方法，此算法目前还未实现，现在是我们自己定义的在玩家棋的右下角下一颗棋，并用遍历输出数字1到500000的方式模拟出电脑判断的时间延迟。

1.4 Main：

程序的入口，创建Board类对象。

2. 极大极小算法与相关优化

主要包括走法产生、搜索算法、估值函数，还包括一些优化
此部分主要参考文献：
董红安《计算机五子棋博弈系统的研究与实现》
王小春《PC游戏编程（人机博弈）》

2.1 走法产生：

采用深度优先搜索。深度优先的好处在于搜索完成一个分支的时候，可以将此分支删除，这就保证了博弈树不会变得非常大。在博弈过程中，如果甲选择最大化收益，那么乙必定会选择让甲的收益最小。因此在博弈开始时，将甲的收益置为最小，乙的收益最大化（相对于甲而言），就是极小极大算法。

2.2 搜索算法：

由于博弈树很大，难以在有限的时间全部遍历，于是产生了alpha - beta剪枝。并在这基础上产生了负极大值算法（Negamax算法），即每次遍历完成后，都将估值取负值向上带回，这样做与极小极大算法的结果是一样的，但是简化了代码，并且采用这种做法后，只需考虑beta剪枝即可，因为向上带回的时候，这个beta实际上将作为alpha出现。但此算法要求估值函数能够通过正负值反映出是哪一方在下棋。上述无论是极小极大算法还是负极大值算法，窗口都是负无穷到正无穷，这种做法是将窗口不断缩小，直到缩到正确的值上。而还有一种做法是事先划定一个小窗口，采用alpha - beta剪枝，如果不在窗口内，就扩大窗口重新搜索，这个思路是渴望搜索。如果更极端一点，将窗口无限小，那么一开始肯定是无法落在窗口中，肯定需要扩大窗口，结果是窗口正好能满足搜索需求。这种思路是极小窗口搜索（又称PVS搜索或NegaScout搜索）

2.3 估值函数：

估值函数根据不同的棋类会有所不同，需要人为设置每种棋型的权重，包括基本棋型的影响与棋型间附加的影响。估值函数的优劣将直接影响整个算法的计算时间与精确度

2.4 搜索算法的优化：

置换表：将每一步落子后预测的估值都记录在数组内，如果下次搜索的局面在置换表中有记录，就直接从置换表读取。无序记录与快速读取的性质要求置换表采用哈希表的数据结构。通过系统时间作为种子生成随机数，然后用此随机数产生哈希值，可以减小碰撞几率。
历史启发：alpha - beta剪枝过程对节点顺序要求很高，最糟糕的情况下将不会产生任何剪枝。需要对节点的估值进行从大到小排序，这样可以最大程度地引发剪枝，提高算法效率
迭代加深：刚开始下棋的时候，将会有大量的节点需要搜索，而真正几率高的只有几步。这样会消耗大量的时间用于搜索。因此在刚开始的时候可以设置搜索深度小一点，随着下棋不断增加搜索深度。

2.5 搜索算法的其他优化：

解决水平效应：如果程序在搜索时陷入了局部最优解，则产生了水平效应。解决方法包括扩展搜索与静止期搜索。扩展搜索是增加搜索深度。静止期搜索是当估值发生意外的剧烈变化时，将向下继续搜索几步，看看到底发生了什么。
开局库与残局库：对于五子棋而言有常用26种开局（董红安的论文《计算机五子棋博弈系统的研究与实现》）在开局几步从开局库中搜索，如果库中没有，再根据算法做选择。残局库同理。
代码优化：优化函数调用与内存管理时间、交叉递归。将双方的回合分开递归，不同回合会走不同的递归路线，这样可以减小每回合执行的代码量。
轻视因子：下棋的对手水平参差不齐，因此电脑在进行判断的时候，如果按照对面是高手的判断方式，可能会下出一步对面不明白的棋，虽然在高手的眼中确定有威胁，但是对方棋艺并没有那么好，所以没发现。这有的时候会导致电脑暴露一些低级的失误。采用轻视因子后，电脑会根据对面的水平调节自己的搜索程度：在“若干步后可能会发生的致命错误”与“显而易见的小错误”中做出选择
多线程：相同时间增加线程数会得到更多算力。

3. 蒙特卡洛算法初步研究：与极大极小算法的区别、在AlphaGO中的应用

3.1 蒙特卡洛算法初步研究：

蒙特卡洛算法（MCTS）分为选择，扩展，模拟，反向传播。与极大极小搜索不同的地方在于两点：一是模拟，MCTS最早是应用于围棋，会产生大量的子节点，全部遍历到子节点是不现实的。因此MCTS不会搜索到确定的局面，而是在一定搜索深度后，通过算法模拟出概率。二是反向传播，在模拟完成后，将概率反向传播到根节点。而极大极小算法则是一直扩展到给定的叶子节点，即确定的局面，并且没有反向传播这一步，最多只是将估值存到置换表里。所以蒙特卡洛算法的限制是搜索时间、最大搜索节点数量（即最大算力），而极大极小算法的限制则是到达极限搜索深度。

3.2 蒙特卡洛在AlphaGO中的应用：

此部分主要参考文献：《Mastering the game of Go with deep neural networks》
AlphaGO主要由四个神经网络构成：快速走子网络（准确度约20%）、监督学习（SL）策略网络（准确度约55%）、强化学习（RL）策略网络、价值网络。每一个神经网络都可以当作是一个独立的程序，但是棋力会下降。由于目标是人类顶级水准，因此Deepmind采用四个网络协同工作的方式。快速走子网络结构简单，判断速度比SL策略网络快，可以用于模拟。SL策略网络则存储了大量的人类对弈记录，用于训练棋力。RL策略网络用于复制SL中的数据然后进行自我博弈，博弈完成后将结果传回SL策略网络。因此只要服务器一直开着，AlphaGO的棋力会不断变强。但RL策略网络的作用不仅用于自我博弈，还将结果提供给价值网络，用于在MCTS中做出判断。价值网络判断分为两个方面：总模拟奖励和总访问次数，并由一个值控制这两部分的权重。价值网络并没有直接采用SL策略网络的人类记录，而是RL网络自己模拟出的记录。原因是人类的对弈数据大致相同，有很强的自相关性，这会导致过拟合使判断失误。在蒙特卡洛搜索过程中首先进行扩展，当扩展到一定程度时，通过快速走子网络模拟出剩余步数的胜率。虽然在模拟过程中会存在误差，但是由于模拟的次数很多，误差之间可以相互抵消，因此最终的结果依然可以反映出真正的胜率。然后用价值网络通过权衡奖励和访问次数来确定这个未完全展开点的胜率，最后反向传播回根节点，让系统做出走子判断。

未完成工作

1. 蒙特卡洛树搜索方法的代码深入研究
   1.1 需要明确MCTS如何进行模拟
   1.2 如何将模拟结果表示出胜率
2. 蒙特卡洛树搜索方法的具体代码实现

计划完成时间和拟采取措施

计划完成时间：2020 - 4 - 20
拟采取的措施：查阅资料并设计程序，尽快完成目标