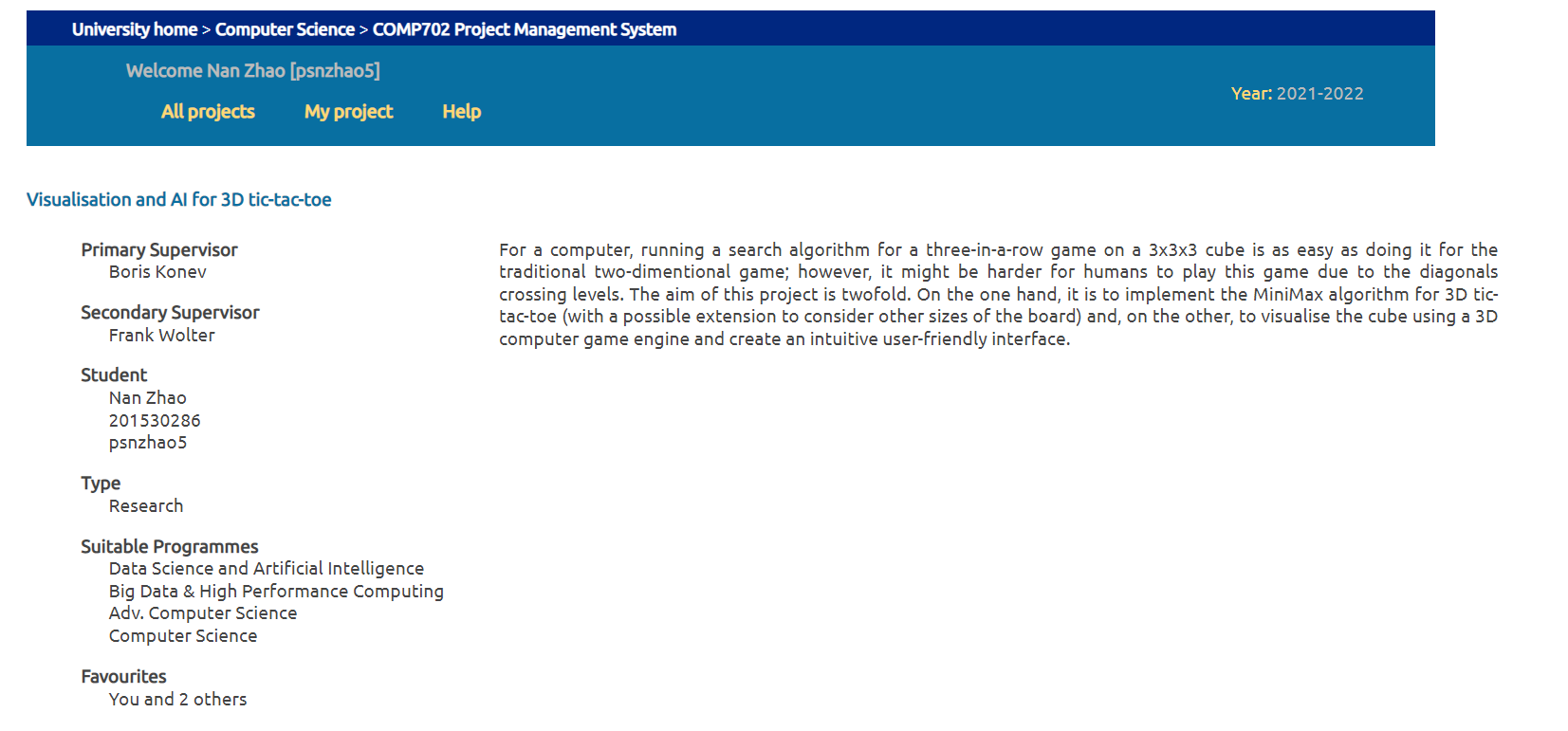
**题目简介：**



对于计算机而言，在 3x3x3 立方体上运行三合一游戏的搜索算法与传统的二维游戏一样容易； 然而，由于对角线交叉关卡，人类可能更难玩这个游戏。 这个项目的目的是双重的。 一方面，它是实现 3D 井字游戏的 MiniMax 算法（可能扩展以考虑其他尺寸的棋盘），另一方面，使用3D 计算机游戏引擎可视化立方体并创建直观的用户友好界面。

**ASSESSMENTS:**

The following three components will be assessed (each submitted through the Coursework Submission System <https://sam.csc.liv.ac.uk/COMP/Submissions.pl?strModule=COMP702>):

* Specification and Proposed Design (15% of the mark)
  1. A single zip file containing:
     + A short report in pdf format (recommended approx 10 pages A4),
     + A set of presentation slides in pdf format,
  2. A 10-minute oral presentation the week after the slides were submitted.

This is marked by the first supervisor only.

* Final Presentation (15% of the mark)
  1. A single zip file containing a set of presentation slides in pdf format,
  2. A 15 minute oral presentation **that includes a software demonstration** the week after the slides were submitted.

This is marked by the first and the second supervisor.

* Dissertation (70% of the mark)  
  A single zip file containing:
  1. A written dissertation document in pdf format (approx 15 to 25 pages A4 plus references and appendices),
  2. All project source code in its native format, **not** converted to pdf.

This is marked by the first and the second supervisor.

思路和知识点：

（1）MinMax算法

MinMax算法中，对每种棋盘局面都有一个估价函数，对A方有利的估价为正数，对B方有利的估价为负数。因此，当A方落子时，必然会选择使得局面估价函数最大的一步，对应了Max；反之，当B方落子时，必然会选择使得局面估价函数最大的一步，对应了Min。

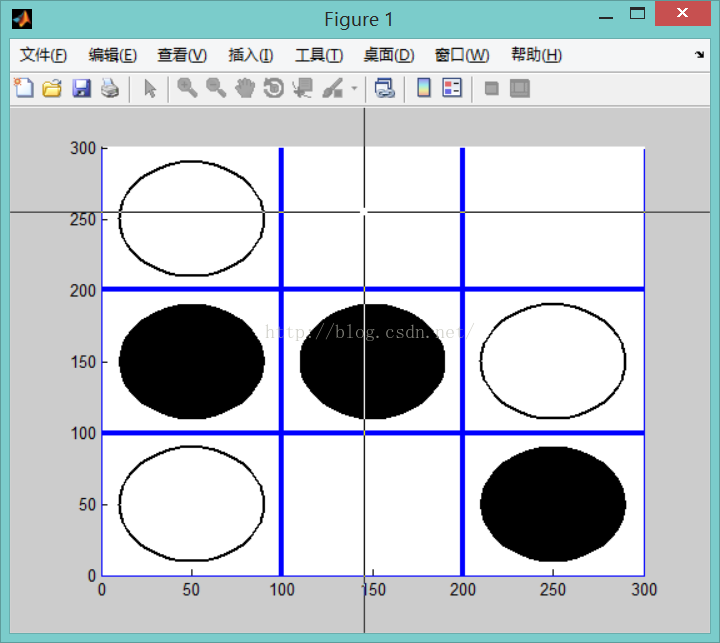
* In the MinMax algorithm, there is an evaluation function for each chessboard position, and the evaluation in favor of party A is positive, and the evaluation in favor of party B is negative. Therefore, when party A makes a move, it must choose the step that maximizes the position evaluation function, which corresponds to Max; conversely, when party B makes a move, it must choose the step that maximizes the evaluation function of the situation, corresponding to Min.

（2）估价函数

估价函数是算法中非常重要的一部分。我采用的估价函数是将所有空着的格子填充成当前玩家的棋子，并计算所有的行、列、对角线有多少连成3个的棋子，并将这些棋子的总个数作为估价值。如果某一方在当前局面下已经胜利，那么估价值设为无穷大（程序中使用一个很大的数值即可）。

* The evaluation function is a very important part of the algorithm. The evaluation function I use is to fill all the empty squares with the current player's pawns, and calculate how many of all the rows, columns, and diagonals are connected into 4 pawns, and use the total number of these pawns as the evaluation value. . If one side has won in the current situation, then the valuation value is set to infinity (a very large value can be used in the program)

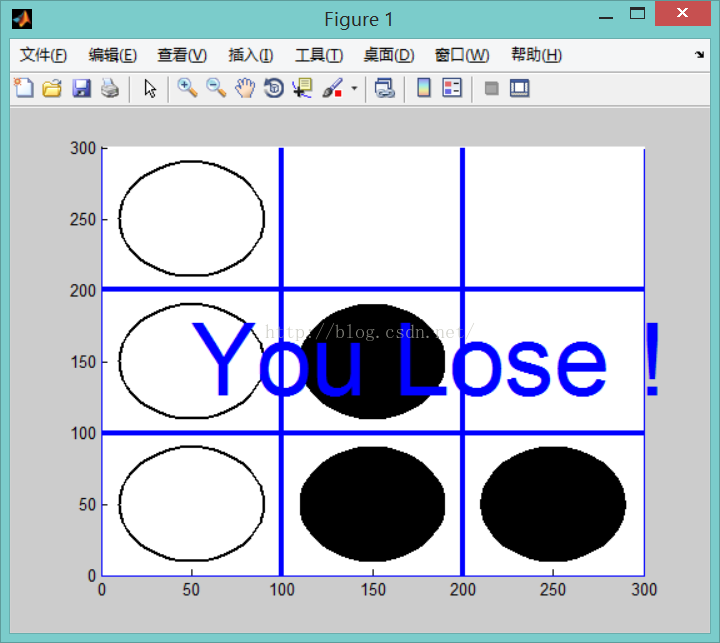
（3）可视化(**源代码下载：**[**http://download.csdn.net/detail/jsgaobiao/9464292**](http://download.csdn.net/detail/jsgaobiao/9464292))

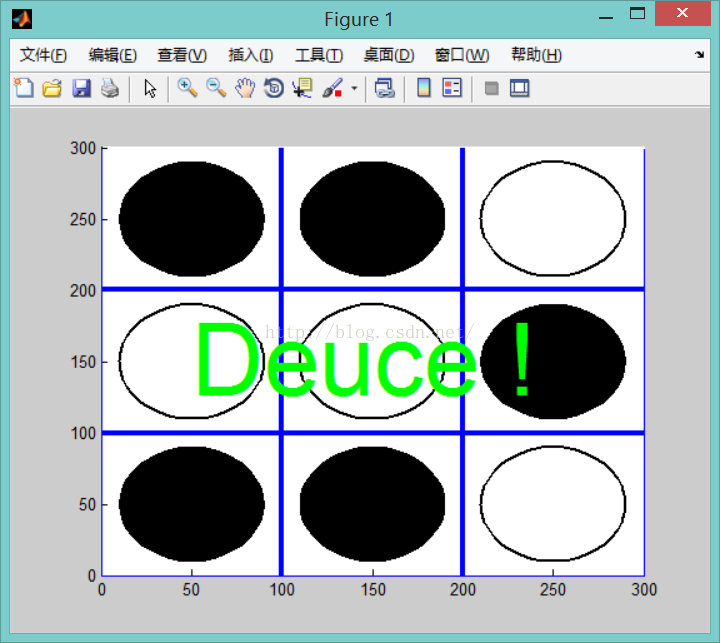


i.用plot函数来绘制棋盘和棋子，并加上了鼠标操作。其中，棋盘是绘制了若干蓝色的直线，棋子是绘制了圆形，通过填充颜色来区分白子和黑子。

ii. 鼠标的控制是在Userinput.m函数中实现的，我通过ginput()函数获取鼠标的坐标位置，并据此判断鼠标当前落在哪一个棋盘格子内，从而获取用户的落子位置。

iii.其中，需要注意的是，外层套了一个while(true)的循环，并在循环内检测鼠标位置是否落在棋盘区域内。这样可以保证用户在随意点击鼠标时，不会被误判为落子。最后还有胜负和平局的判断：





**总结：**

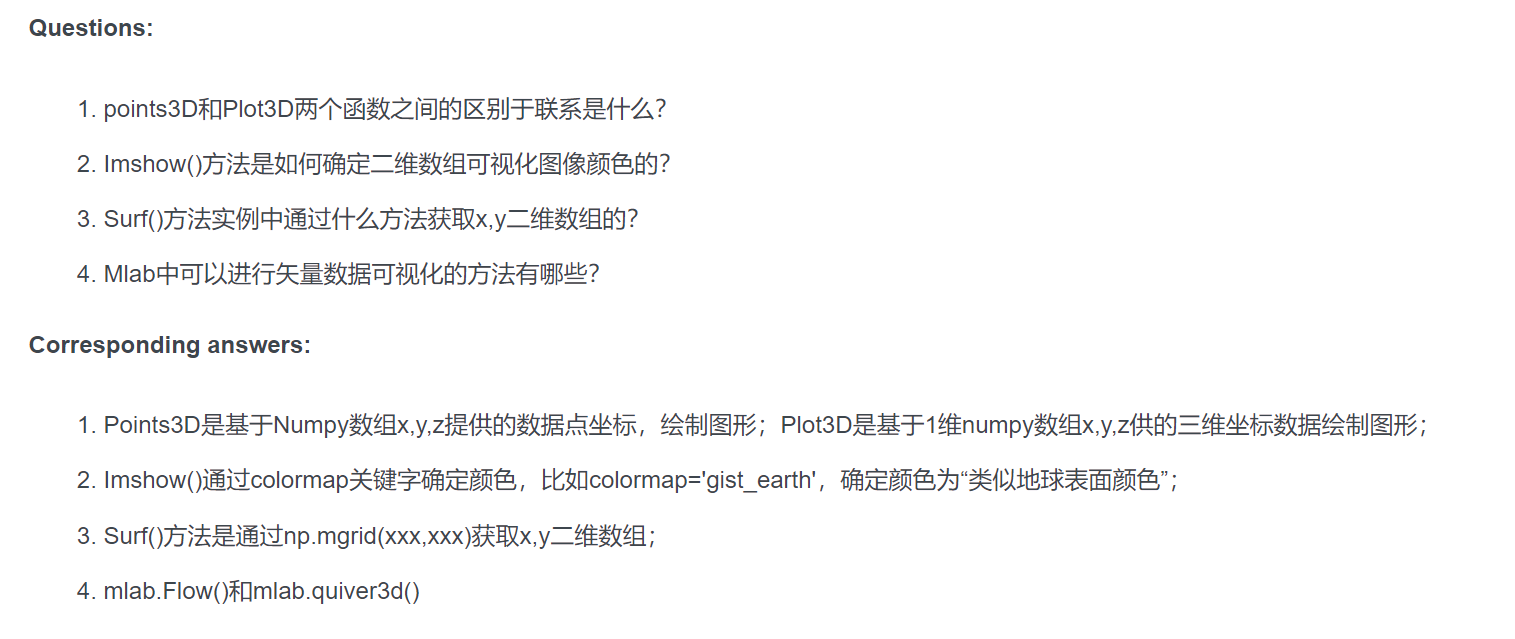
博弈树是很多二人对抗游戏都可以采用的一种AI决策方式，但是整棵博弈树的搜索所产生的状态空间是巨大的并且在时间和空间上难以承受的。

本次作业算法的一个特点就是采用了Negmax和Alpha-Beta剪枝，这样大大减少了博弈树的状态空间，可以使程序检测更多的状态，从而获得更优的决策方案。经过测试，人几乎赢不了这个井字棋的AI，也说明了该算法在井字棋的规则下已经足够优秀。

由于井字棋3\*3的棋盘比较小，导致AI完全可以选择非常优的策略，所以有一点不足就是这个游戏的趣味性就会有些欠缺。如果扩大棋盘，或者改变一下规则，比如连成4子才算胜利等，可能会更有趣。

Minimax 策略预计算

之前我们对每个棋局去计算最佳的下一步，并在此过程中做了剪枝，即当已经找到当前玩家必胜落子时直接返回。这对于单一局面的计算是较优的，但是AI Agent 需要在每一步都重复这个过程，当棋盘大小>3时运算非常耗时，因此我们来做第二种优化。初始空棋盘时使用Minimax来保证遍历所有状态，缓存所有棋局的最佳结果。对于AI Agent面临的每个棋局只需查找此棋局下所有的可能落子位置，并返回最佳决定，这样大大减少了每次棋局下重复的minimax递归计算。



PPT草稿：

井字棋是一个很古老很简单的游戏，两名玩家在一个3X3的[网格](https://so.csdn.net/so/search?q=%E7%BD%91%E6%A0%BC&spm=1001.2101.3001.7020)上画上自己的图标，每回合玩家只能选择一个格子，率先将三个自己图标连成一条直线的玩家获胜（如果在九个格子都被填充后仍没有获胜者，则判为平局）。