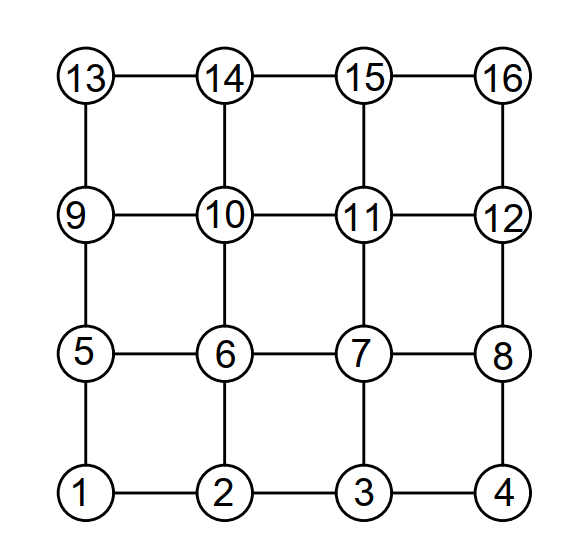
程序文字描述部分

1、棋格拓扑关系部分使用四个一组的拓扑关系，棋格拓扑关系如下表格所示：



2、拓扑的词典是设置为下面的表格，在c++中使用二维数组实现该词典，字典使用时顺针方向存储，存储的顺序为：左->上->右->下，使用CheckerboardClass类进行提供该拓扑词典（ChessTopologyArray）的算法。

拓扑词典表格为：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **拓扑编号** | **拓扑链接T1（左）** | **拓扑链接T2（上）** | **拓扑链接T3（右）** | **拓扑链接T4（下）** |
| **1** | **0** | **5** | **2** | **0** |
| **2** | **1** | **6** | **3** | **0** |
| **3** | **2** | **7** | **4** | **0** |
| **4** | **3** | **8** | **0** | **0** |
| **5** | **0** | **9** | **6** | **1** |
| **6** | **5** | **10** | **7** | **2** |
| **7** | **6** | **11** | **8** | **3** |
| **8** | **7** | **12** | **0** | **4** |
| **9** | **13** | **10** | **5** | **0** |
| **10** | **9** | **14** | **11** | **6** |
| **11** | **10** | **15** | **12** | **7** |
| **12** | **11** | **16** | **0** | **8** |
| **13** | **0** | **14** | **9** | **0** |
| **14** | **13** | **0** | **15** | **10** |
| **15** | **14** | **0** | **16** | **11** |
| **16** | **15** | **0** | **0** | **12** |

机器人行走算法（没有给与每一步详细的计算过程）

1. 通过随机算法，随机移动机器人要走的子，通过拓扑关系移动该子可以移动的方向，判断要移动的拓扑点是否有别的子在哪里，如果没有其他子占用就移动到该位置。
2. 如果有其他子占用该子的位置，改变方向。如果该子没有位置可以移动，就需要再次随机移动其他的子再次判断是都可以行走。

人方面的算法（每次行走前需要判断人的子是否有可以行走的方向）

1. 人移动子时也需要在拓扑方向中进行比较只要合法的方向才可以行动，其他方向都是不能想走子的。

移动玩子后要进行第四部的吃子判断，以及胜负判断。

3、流程图如下图所示：



4、主动部分是否可以吃掉对方棋子判断，这是棋子最为关键的程序代码部分。主要分成纵横两部分。

当蓝方的子从T1移动T2位置事，横向（列）部分：

该部分的代码使用一个MapCoordSequence来存储拓扑点的坐标点，坐标使用point（x行，y列），使用0、1表示坐标表示的红字和黑子。通过T2拓扑值，在MapCoordSequence中查找到对应的坐标点，然后在行列词典（ColumnRowDictionary）中找出对应行的所有拓扑点。行列词典的如下所示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列词典 | | | | |
| 行号 | 词典值 | | | |
| 1 | 1 | 2 | 3 | 4 |
| 2 | 5 | 6 | 7 | 8 |
| 3 | 9 | 10 | 11 | 12 |
| 4 | 13 | 14 | 15 | 16 |
| 列号 | 词典值 | | | |
| 1 | 1 | 5 | 9 | 13 |
| 2 | 2 | 6 | 10 | 14 |
| 3 | 3 | 7 | 11 | 15 |
| 4 | 4 | 8 | 12 | 16 |

该部分的算法描述为：

1. 拓扑值寻找到对应的行列坐标点
2. 通过坐标点寻找出对应的行列词典
3. 通过行列词典在MapbuleChessman和MapRedChessman进行对比存储对应的词典，再两个表格中没有查找到的拓扑点，给该点的状态置位-1，红色的点置位0，蓝色的点置为1。
4. 一次判断行列词典的第一位和最后一位有且仅有一个为-1并且该行（列）棋子数量一定为3个，否则退出吃子判断
5. 判断吃子的条件为红方吃蓝方为001，100即可，蓝方吃红方是否为110，011状态即可。

程序界面部分



1. 在线匹配好友，寻找好友进行游戏
2. 开始游戏，游戏时长不受限制。
3. 再来一局，重新开始游戏。
4. 是否让棋可以让对方再走一步。
5. 人机对战，与机器进行对战，这里提供一算法机器行走算法。

第一阶段只实现1.0界面开发部分。

程序伪代码部分

程序动态数据（提供的删除，更新，保存等状态，外部存档数据）及静态数据。

1. 该程序中需要的数据为：存储红蓝棋子拓扑关系的map表。

MapbuleChessman、MapRedChessman

1. 棋格静态拓扑词典（左上右下的顺序）、行列拓扑点（从左到右，从下到上的书序）,坐标点Map。

ChessTopologyArray、ColumnRowTopoArray，MapCoord <point>。

3、程序需要的临时数据：ChessmanStatus主要用来判断吃子过程中使用。

主要程序需要提供的内容类

ChessClass：程序界面调用的主类，初始化，胜负判定。

提供的方法主要有：

initUI（重置棋局）、IsSuccess、getChessTopologyArray（int bT/rT），getColumnRowTopoArray（int bT/rT），getMapCoord（int bT/rT）。

CheckerboardClass：提供对ChessTopologyArray、ColumnRowTopoArray、MapCoord。的查询，加载。

其他内容详细见程序代码。

未来的展望

首先实现版本1.0的程序，只是开发了该棋局的基本走棋，提供局域网人人匹配，机器行走算法，没有提供智能行走算法。

其次在版本2.0中添加人工智能的部分，提高机器人的算法功能，同时引入语音系统可以实现语音走棋。同时为了更好的开发，提供标准的API供初学者使用。

希望该游戏能集思广益开发出更多该系列的棋局走法，BF,BS,BE,BT,4,6,8,12等子决，后续开发的希望能够对棋子进行不同功能的分类，提升游戏的复杂度。

当然我业余开发者希望有志之士能够给出更好的算法，改进游戏开发性能，我的代码代码公布在<https://github.com/mu116699/Battle>-of-Four，希望可以改进的游戏能够上传到github。该小游戏基于QT开发，源码基础程序使用C++可以在不同平台进行编译，可在Windows、Mac、IOS、Android上运行。