**网络象棋技术文档**

**--1152703 方志晗**

## 坐标转换

CPoint类中的两个主要函数说明

|  |  |
| --- | --- |
| **函数实现** | **函数说明** |
| //转换为棋盘坐标  **public** **static** CPoint changeToCPoint(Point a)  {  CPoint temp =**new** CPoint ();  temp.Cx =(**int**)(((**double**)a.x-initCx)/CPWidth);  temp.Cy =(**int**)(((**double**)a.y-initCy)/CPHeight);  **return** temp;  } | **将java坐标的横纵坐标减去棋盘初位置的横坐标后除以棋子坐标的长度和宽度，即从java坐标转换为棋盘坐标** |
| //转换为坐标  **public** **static** Point changeToPoint(CPoint a)  {  Point temp = **new** Point();  temp.x=(**int**)((**double**)a.getCx()\*(CPoint.*CPWidth*)+CPoint.*initCx*);  temp.y=(**int**)((**double**)a.getCy()\*(CPoint.*CPHeight*)+CPoint.*initCy*);  **return** temp;    } | **将棋盘坐标的横纵坐标乘以棋盘坐标的长宽后加上棋盘的初始位置得到该棋盘点的java坐标，即从棋盘坐标转化为java坐标** |
|  |  |

## 棋子关系

|  |
| --- |
| 基类ChessPiece（抽象类） |
| **public** **abstract** **class** ChessPiece **extends** JLabel **implements** MouseListener {  //构造用的图片  **public** Icon img =**null**;  **public** CPoint CurrentCPoint=**null**; //记录棋子的当前坐标  **public** CPoint LastNode =**null**; //上一步的坐标  **public** **boolean** IsSelect =**false**; //记录棋子是否被选择  **public** **boolean** isWalking =**false**; //记录棋子是否在执行走的动作  **public** ChessType ChessMan = **null**; //记录棋子的种类 /车， 马。。。。  **public** ChessColor chessColor =**null**; //记录棋子的颜色  **public** CPoint InitCPoint =**null**; //起始坐标  **public** **boolean** CanReachable =**false**; //是否可以打到  **public** **boolean** isDead = **false**; //是否死亡  **public** **boolean** isCrossRiver =**false**;  **public** **int** Num; //给每颗棋子编号，便于找到在数组中位置    **public** ChessBoardPanel myBoardPanel = **null**; //棋盘，方便调用上一层  //行走函数，视具体棋子而定走法  **public** **abstract** **void** walkTo(CPoint target);  //判断行走函数  **public** **abstract** **boolean** canWalk(CPoint target);  //基类的行走函数  **public** **void** walk (CPoint target) //可行走到任意位置，在walkTo中调用canWalk和walk来完成每颗棋子的不同走法  //颜色对换函数，实现网络对战时一方转换棋子颜色  **public** **abstract** **void** changeColor();  。。。  } | |

说明：每种棋子继承该基类的三个抽象方法，完成不同走法，具体走法实现见源码；基类中鼠标监听实现所有棋子的鼠标。若棋子未选择,则IsSelect置为真，并将其他棋子的IsSelect置为假，在主类MainChessBoard的线程方法run中对所有棋子的IsSelect进行监听，若为真，则让其执行闪烁效果的函数。若当前棋子已经被选择，则点击时放弃选择。在当前棋子未被选择的点击事件中添加对吃子的响应，即判断是否当前为对方行走，通过判断静态量PresentTurnColor 是否与地方的颜色相同，若是则判断对方是否可吃该子，若是，则执行吃子函数。

## 网络实现

网络通过发送编码和接受解码实现控制命令的传输，网络实现在服务器端用了8个线程，线程作为双发转发消息，具体线程安排如下

|  |
| --- |
| **服务端的线程安排：** |
| **线程** | **说明** |
| **Thread1 Thread2** | **双方棋子对战控制命令的相互转发** |
| **Thread3 Thread4** | **双方普通聊天信息的转发** |
| **Thread5 Thread6** | **双方高级聊天信息的转发** |
| **Thread7 Thread8** | **双方高级聊天字体相关信息的转发** |

控制命令编码方式如下“Ax”，A为命令符号，x为控制符号。接受端解码时首先判断A为哪种命令，找到A的命令类型后，再解码控制符号‘x’，不同控制命令的控制符号有不同的含义。具体说明如下

|  |
| --- |
| **对战命令编码说明：** |
| **网络命令** | **含义** | **举例说明** |
| **Sx** | **敌方选择x号棋子，接收端接收到该命令后，即让本地棋盘**  **的地方x号棋子置为已选择，执行闪烁动画** | **S1表示选中数组中的1号棋子，在这里是放在左上角的“车”** |
| **Ex** | **敌方吃掉我方的x号棋子，接受端接受到这条命令后，即将本地棋盘的地方吃去我们x号棋子** | **E1表示吃掉我方1号棋子，接受端接受到这条** |
| **Mxy** | **对方移动到位置（x,y）,接收端接收到该命令后，自动将其转换为（8-x，9-y），即中心对称为本地敌方棋盘** | **M11在发送端反应为选择的子移动到位置(x,y),接收端译码为选择敌方棋子移动（7，8）位置。** |
| **Fx** | **功能命令，具体如下** |  |
| **F1表示准备好开始游戏，** |
| **F2表示提出投降** |
| **F3表示提出悔棋申请，对方接收到该命令后出现弹框，提出悔棋申请** |
| **F4表示求和申请，对方接收到后出现弹框求和申请** |
| **Jx** | **回复命令，对对方请求的回复** |  |
| **J1表示对方同意提出的和棋申请** |
| **J2表示拒绝和棋，在接受端出现消息框，对方拒绝和棋** |
| **J3表示对方同意悔棋申请** |
| **J4表示对方拒绝悔棋** |

|  |
| --- |
| **高级聊天控制的编码说明：** |
| **TaZbCcBd** | **Ta表示发送端的字体风格，接收端解码后译出编号a的编码** | | **每次发送端发送一个消息的同时发送发送端当前的字体，颜色，大小等，解码时依次通过用string中的substring函数结合indexof函数截取T到Z，Z到C，C到B，以及B到最终的字符串，然后通过**  **Integer.parseInt函数转换为int型** |
| **a编码具体说明如下** | |
|  | **编码** | **字体** |
| **1** | **常规** |
| **2** | **粗体** |
| **3** | **斜体** |
| **4** | **粗斜体** |
| **Zb表示字体，b正好是字体大小，没有设计到编码** | |
| **Cc表示发送端的字体颜色编码为c，接收端接收到后将对方发送过来的消息自动设置成c解码后的颜色，c的具体编码如下** | |
| **1** | **黑色** |
| **2** | **红色** |
| **3** | **蓝色** |
| **4** | **黄色** |
| **5** | **绿色** |
| **Bb表示发送端的背景色为b，接收端接收到b后进行解码，在本地将对方发过来的消息置成解码后的背景色** | |
| **6** | **无色** |
| **7** | **灰色** |
| **8** | **淡红** |
| **9** | **淡蓝** |
| **10** | **淡黄** |
| **11** | **淡绿** |