斗地主AI算法——第七章の被动出牌(1)

原文地址：<https://blog.csdn.net/sm9sun/article/details/70828085>

本章开始讲被动出牌的逻辑算法。首先我们先把架子搭起来，被动出牌我们肯定是要知道场上目前打出的是什么牌型。

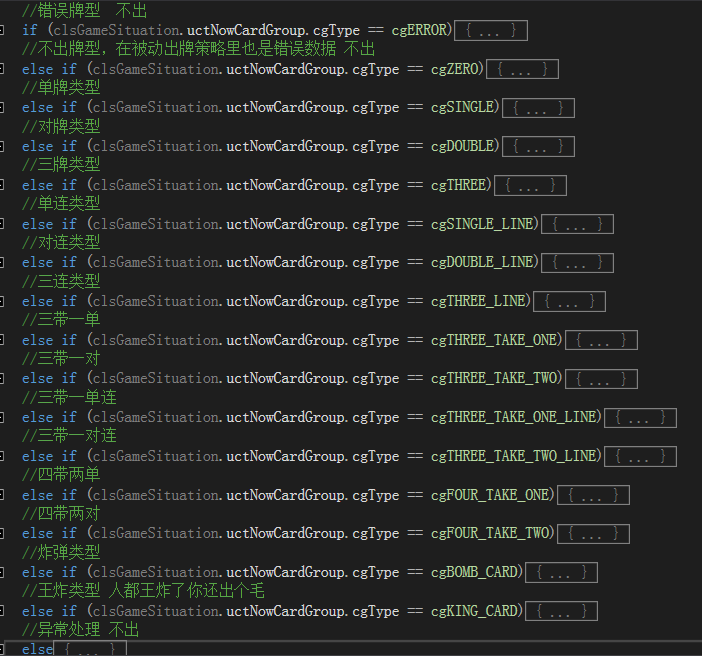
在第二章数据结构里我们定义过，游戏全局类里面有一个存放当前牌型结构的成员，即

[cpp] [view plain](https://blog.csdn.net/sm9sun/article/details/70828085) [copy](https://blog.csdn.net/sm9sun/article/details/70828085)

//当前打出牌的类型数据，被动出牌时玩家根据这里做出筛选

CardGroupData uctNowCardGroup;

我们即将通过他进行类型的筛选，所以肯定是要枚举各类牌型的，也就是这个样子的。



是的不要质疑！就是这个样子的~~



当然了，虽然是2.0初级版，我们还是要给予一定的灵性的，比如说当最后只剩两手牌且存在王炸的话，王炸优先出。

[cpp] [view plain](https://blog.csdn.net/sm9sun/article/details/70828085) [copy](https://blog.csdn.net/sm9sun/article/details/70828085)

/\*王炸——当前策略只处理王炸作为倒数第二手的优先出牌逻辑，后续版本会在此基础上优化\*/

**if** (clsHandCardData.value\_aHandCardList[17] > 0 && clsHandCardData.value\_aHandCardList[16] > 0)

{

    clsHandCardData.value\_aHandCardList[17] --;

    clsHandCardData.value\_aHandCardList[16] --;

    clsHandCardData.nHandCardCount -= 2;

    HandCardValue tmpHandCardValue = get\_HandCardValue(clsHandCardData);

    clsHandCardData.value\_aHandCardList[16] ++;

    clsHandCardData.value\_aHandCardList[17] ++;

    clsHandCardData.nHandCardCount += 2;

**if** (tmpHandCardValue.NeedRound == 1)

    {

        clsHandCardData.value\_nPutCardList.push\_back(17);

        clsHandCardData.value\_nPutCardList.push\_back(16);

        clsHandCardData.uctPutCardType = clsGameSituation.uctNowCardGroup = get\_GroupData(cgKING\_CARD, 17, 2);

**return**;

    }

}

算法思路：若存在手牌17（大王）和16（小王），那么先去除这两张牌

然后通过get\_HandCardValue获取剩余轮次。

再回溯到原有状态。若只剩一手，则打出王炸。

出牌的操作也很简单，将需要打出的牌进入clsHandCardData.value\_nPutCardList数组，且通过第四章提到的get\_GroupData函数获取类型结构再赋值给手牌类以及游戏全局类相应的成员变量。当确定好出牌策略后，直接return。因为被动出牌的分支只会走一个，为了节约时间，所以每个分支里都有return，若没有走入任何分支则视为错误数据。

我们先把最简单的类型写出来，就是别人出王炸时的策略。

[cpp] [view plain](https://blog.csdn.net/sm9sun/article/details/70828085) [copy](https://blog.csdn.net/sm9sun/article/details/70828085)

//王炸类型 人都王炸了你还出个毛

**else** **if** (clsGameSituation.uctNowCardGroup.cgType == cgKING\_CARD)

{

    clsHandCardData.uctPutCardType = get\_GroupData(cgZERO, 0, 0);

**return**;

}

怎么样，是不是很简单？



若不出牌，我们只更新自己手牌类型就好了。若出牌时，不但要更新自己手牌信息，也要更新游戏全局类里面的当前出牌信息。不过我更推荐后期嵌入的时候通过服务器来获取当前出牌的信息。比如我的测试函数里会加上：

[cpp] [view plain](https://blog.csdn.net/sm9sun/article/details/70828085) [copy](https://blog.csdn.net/sm9sun/article/details/70828085)

**if** (arrHandCardData[indexID].uctPutCardType.cgType != cgZERO)

{

    clsGameSituation.nCardDroit = indexID;

    clsGameSituation.uctNowCardGroup = arrHandCardData[indexID].uctPutCardType;

}

另外，出牌前记得清空一下出牌序列，就是在一开始加：

[cpp] [view plain](https://blog.csdn.net/sm9sun/article/details/70828085) [copy](https://blog.csdn.net/sm9sun/article/details/70828085)

clsHandCardData.ClearPutCardList();

所以，整个函数的架子是这样的，假设我们啥都管不上。

[cpp] [view plain](https://blog.csdn.net/sm9sun/article/details/70828085) [copy](https://blog.csdn.net/sm9sun/article/details/70828085)

/\*

2.0版本策略  根据场上形势决定当前预打出的手牌——被动出牌

\*/

**void** get\_PutCardList\_2\_limit(GameSituation &clsGameSituation, HandCardData &clsHandCardData)

{

    clsHandCardData.ClearPutCardList();

    /\*王炸——当前策略只处理王炸作为倒数第二手的优先出牌逻辑，后续版本会在此基础上优化\*/

**if** (clsHandCardData.value\_aHandCardList[17] > 0 && clsHandCardData.value\_aHandCardList[16] > 0)

    {

        clsHandCardData.value\_aHandCardList[17] --;

        clsHandCardData.value\_aHandCardList[16] --;

        clsHandCardData.nHandCardCount -= 2;

        HandCardValue tmpHandCardValue = get\_HandCardValue(clsHandCardData);

        clsHandCardData.value\_aHandCardList[16] ++;

        clsHandCardData.value\_aHandCardList[17] ++;

        clsHandCardData.nHandCardCount += 2;

**if** (tmpHandCardValue.NeedRound == 1)

        {

            clsHandCardData.value\_nPutCardList.push\_back(17);

            clsHandCardData.value\_nPutCardList.push\_back(16);

            clsHandCardData.uctPutCardType = clsGameSituation.uctNowCardGroup = get\_GroupData(cgKING\_CARD, 17, 2);

**return**;

        }

    }

    //错误牌型  不出

**if** (clsGameSituation.uctNowCardGroup.cgType == cgERROR)

    {

        clsHandCardData.uctPutCardType = get\_GroupData(cgERROR, 0, 0);

**return**;

    }

    //不出牌型，在被动出牌策略里也是错误数据 不出

**else** **if** (clsGameSituation.uctNowCardGroup.cgType == cgZERO)

    {

        clsHandCardData.uctPutCardType = get\_GroupData(cgZERO, 0, 0);

**return**;

    }

    //单牌类型

**else** **if** (clsGameSituation.uctNowCardGroup.cgType == cgSINGLE)

    {

        //管不上

        clsHandCardData.uctPutCardType = get\_GroupData(cgZERO, 0, 0);

**return**;

    }

    //对牌类型

**else** **if** (clsGameSituation.uctNowCardGroup.cgType == cgDOUBLE)

    {

        //管不上

        clsHandCardData.uctPutCardType = get\_GroupData(cgZERO, 0, 0);

**return**;

    }

    //三牌类型

**else** **if** (clsGameSituation.uctNowCardGroup.cgType == cgTHREE)

    {

        //管不上

        clsHandCardData.uctPutCardType = get\_GroupData(cgZERO, 0, 0);

**return**;

    }

    //单连类型

**else** **if** (clsGameSituation.uctNowCardGroup.cgType == cgSINGLE\_LINE)

    {

        //管不上

        clsHandCardData.uctPutCardType = get\_GroupData(cgZERO, 0, 0);

**return**;

    }

    //对连类型

**else** **if** (clsGameSituation.uctNowCardGroup.cgType == cgDOUBLE\_LINE)

    {

        //管不上

        clsHandCardData.uctPutCardType = get\_GroupData(cgZERO, 0, 0);

**return**;

    }

    //三连类型

**else** **if** (clsGameSituation.uctNowCardGroup.cgType == cgTHREE\_LINE)

    {

    //管不上

        clsHandCardData.uctPutCardType = get\_GroupData(cgZERO, 0, 0);

**return**;

    }

    //三带一单

**else** **if** (clsGameSituation.uctNowCardGroup.cgType == cgTHREE\_TAKE\_ONE)

    {

        //管不上

        clsHandCardData.uctPutCardType = get\_GroupData(cgZERO, 0, 0);

**return**;

    }

    //三带一对

**else** **if** (clsGameSituation.uctNowCardGroup.cgType == cgTHREE\_TAKE\_TWO)

    {

        //管不上

        clsHandCardData.uctPutCardType = get\_GroupData(cgZERO, 0, 0);

**return**;

    }

    //三带一单连

**else** **if** (clsGameSituation.uctNowCardGroup.cgType == cgTHREE\_TAKE\_ONE\_LINE)

    {

        //管不上

        clsHandCardData.uctPutCardType = get\_GroupData(cgZERO, 0, 0);

**return**;

    }

    //三带一对连

**else** **if** (clsGameSituation.uctNowCardGroup.cgType == cgTHREE\_TAKE\_TWO\_LINE)

    {

        //管不上

        clsHandCardData.uctPutCardType = get\_GroupData(cgZERO, 0, 0);

**return**;

    }

    //四带两单

**else** **if** (clsGameSituation.uctNowCardGroup.cgType == cgFOUR\_TAKE\_ONE)

    {

        //管不上

        clsHandCardData.uctPutCardType = get\_GroupData(cgZERO, 0, 0);

**return**;

    }

    //四带两对

**else** **if** (clsGameSituation.uctNowCardGroup.cgType == cgFOUR\_TAKE\_TWO)

    {

        //管不上

        clsHandCardData.uctPutCardType = get\_GroupData(cgZERO, 0, 0);

**return**;

    }

    //炸弹类型

**else** **if** (clsGameSituation.uctNowCardGroup.cgType == cgBOMB\_CARD)

    {

        //管不上

        clsHandCardData.uctPutCardType = get\_GroupData(cgZERO, 0, 0);

**return**;

    }

    //王炸类型 人都王炸了你还出个毛

**else** **if** (clsGameSituation.uctNowCardGroup.cgType == cgKING\_CARD)

    {

        clsHandCardData.uctPutCardType = get\_GroupData(cgZERO, 0, 0);

**return**;

    }

    //异常处理 不出

**else**

    {

        clsHandCardData.uctPutCardType = get\_GroupData(cgZERO, 0, 0);

    }

**return**;

}

当然，啥都管不上肯定是不行的，所以接下来我们会填充各种牌型的策略算法。