人智综合作业 1——斗地主报告

自 92 李永健 2019010819

目录

1 算法分析

1.1 发牌

1.1.1 随机发牌

首先生成一个从 1~54 按顺序排列的数组，再利用 random.shuffle 将其打乱顺序，根据输入的牌的数量 pokerCnt，截取前 pokerCnt 个元素作为手牌，利用扑克牌类 Poker 生成一个扑克牌数组，利用 sort 函数将手牌从小到大进行排序，存入初始节点中作为初始状态。

1.1.2 指定手牌

输入一个扑克牌数组，其中每个元素为包含扑克牌的点数和花色的元组，利用该数组生成初始手牌。

1.2 搜索可以出的手牌组合

对于每个节点，在构造函数中首先遍历存入的扑克牌状态数组，得到每个点数的牌的数量，存在一个字典 pokerCtn 中，再利用 OrderedDict 生成一个可以出的扑克牌组合的字典 possibleStep，字典的键从三顺子到单牌依次添加，这样在后续迭代搜索中可以按照需要的优先级顺序进行遍历。

首先根据根据每个点数扑克牌的数量添加可以出的单牌，添加的顺序按照扑克牌数量从小到大进行添加，这样可以减少拆牌的可能性，每种点数的牌只添加一张牌。对子牌、三张牌和炸弹也按照这种方式进行添加，为了便于后续的计算，火箭也算作对子牌进行添加。

之后搜索三带一、三带二的牌的组合，依次对得到的三张牌的列表和单牌的列表进行遍历，遍历过程中注意判断三张牌和单排是否为同一点数，由于单牌再搜索时按照扑克牌数量进行添加，可以保证三带一带走的单牌优先为一张单牌。三带二的搜索也同理，我在搜索中认为火箭不算对子牌。

搜索四带二、四带二对和上述的方法类似，遍历四张牌的列表和单牌的列表，如果单牌和四张牌点数不同，再对单牌列表中该单牌后续的牌进行遍历搜索，这样可以避免重复，我在作业中认为对子牌也可以认为是两张单牌，因此四带二搜索过程中除了搜索四带两张不同的单牌外还需搜索四带一对。四带二对的搜索过程类似，注意火箭不算对子牌。

搜索顺子牌时，遍历 pokerCnt 中的元素，得到当前牌的点数，如果该点数牌的数量满足要求，就将其存入一个临时列表中，并利用 Poker 中的 get_next_poker_num 的函数得到下一个需要的点数，如果下一个需要的点数的牌的数量同样满足要求，就将其 append 入列表，直到对应后续点数牌的数量不满足要求，或者搜索到 2 或双王为止。如果该临时列表的长度满足顺子的要求，就将该临时列表存入 possibleStep 中，并不断去除该列表的最后一个元素，如果新的临时列表满足长度要求，就把去除最后一个元素后的列表存入 possibleStep 中，重复上述步骤直到长度不满足要求为止。

1.3 搜索可以最快出完所有牌的出牌策略

利用 A* 算法实现该功能，启发函数的值为当前节点的牌的数量除以当前节点可以出的最长的牌的数量再向上取整，相当于从当前节点开始每次出牌数量均为当前节点的最长出牌数，由于每出完牌，下一个节点的最长出牌数一定减少或不变，因此可以保证真实的出牌步骤一定大于等于该启发函数的值，因此启发函数满足可采纳性，同时该问题为树搜索，启发函数满足可采纳性就足够了。

仅仅利用 A* 算法而不进行剪枝，整体的复杂度还是非常高的，为了减少分支的节点数量，将顺子和非顺子类的牌分开进行考虑，由于每次出牌的过程一定包含顺子和非顺子两部分，因此可以先出完一种牌再出另一种牌，在本次作业中，就采用先出顺子后出非顺子的策略，即如果开始出非顺子类的牌，就不再出顺子类的牌，基于这种搜索策略，即使没有顺子也可以较好地实现最快出牌策略的搜索，速度也比较快，几乎任何初始手牌组合都可以在 1 分钟之内跑出结果。

实际算法实现过程中，对于顺子类的牌，不同顺子组合之间的关联性很高，很难进行剪枝，因此每个节点在扩展时会将所有可能出的顺子类的牌都加以扩展，而对于非顺子类的牌，每个节点只需要扩展一次，这是由于非顺子类的牌相关性较弱，可以比较容易地找到一种方式使其以最少的步骤出完，每个节点可以直接扩展一次找到最佳的出牌步骤，因此对于非顺子类的牌，可以在线性时间复杂度跑出结果，极大地减少了分支数量。

对于非顺子类的牌，我在作业中优先出四带二，且优先出带两张牌数为 1 的单牌的情况，如果带的两张单牌中有一个的牌数不为一，说明牌数为 1 的单牌最多只有一个，此时考虑四带二对效果更好，如果四带二对中有一对的牌数等于 2，则出四带二对，这是由于即使另一对的牌的牌数大于 2，由于搜索可以出的手牌组合中的优先级顺序，牌数等于 2 的牌也只能有一对，这样即使拆掉了一个三张牌或者一个炸弹，最慢也能一部出完被拆掉的牌，整体上所需要的步骤仍然最小。而如果四带二对的两对牌每对牌的牌数都大于等于 2，则仍然出四带二而不是四带二对，这是由于四带二对可能导致炸弹被拆为两对，导致出完四带二对后还需出两步才能把被拆的牌也出完，而四代一即使把炸弹拆了仍然可以剩三张牌，可以利用三带一或者三带二更快地出完所有牌。

对于三带二和三带一可以进行类似的考虑，如果三带二的对子的牌数大于 2，则考虑三带一，如果三带一的单牌的牌数等于 1，则出三带一，否仍然出三带二，剩下的牌就可以按照炸弹、三张牌、对子、单牌的顺序依次出直到出完所有牌。容易证明按该顺序出牌一定能得到一种最优的出牌策略。

特殊情况处理：由于四带二对可以包括四带四，如果非顺子考虑四带四的话会很麻烦，因此四带四按照顺子进行处理。

1.4 按照最优 score 搜索出牌策略

由于按照 score 进行优化如果采用 A* 搜搜很难找到有效的启发函数，无论怎么变换 score 一定是包含对数项的，因此最后采用的是深度优先搜索实现的优化。每个节点中顺子的扩展方式和搜索最少出牌步骤的搜索的扩展相同，四带四同样按照顺子进行处理，而对于非顺子则有一定的差别，比如当有一个炸弹和两个三张牌时，按照炸弹、三张牌、三张牌的顺序出牌的得分会比按照四带二加两个对子的出牌方式效率要高，由于情况很复杂，只能同样利用深度优先搜索进行扩展，但可以针对部分特殊情况进行剪枝处理，比如如果四带二的两个单牌的牌数都等于 1，四带二对的两对牌

的牌数都等于 2，以及三带一和三带二的对应情况都可以进行剪枝，对于只剩对子和单牌的情况也可以直接搜到出完所有牌，但整体上而言当牌数大于 30 之后速度仍然会比较慢，目前并没有找到更好的加速方法。

我曾想再每次深搜扩展时计算当前的 score 值，判断其与已有的 score 值比是否能更小，如果更小就直接进行剪枝处理，但我并不能证明再本题的条件下，score 一定随步数增加而递减，因此最后并没有进行如上处理。

1.5 1v1 对战

本次作业中，我用两副牌分别给双方随机发牌，根据对方出的牌得到自己下一步要出的牌，如果对方没有出牌或者对战刚开始，则按照三顺子、间隔三顺子、··· 对子、单牌的顺序进行选择第一个可以出的手牌组合打出，对方如果出牌了，就根据对方的牌型在自己所有可能的出牌步骤里搜索，找到对应的牌型，如果自己有比对方大的同牌型的牌，则打出，注意顺子要保证长度相同。如果一方手牌为 0，则对战结束。

2 UI 界面

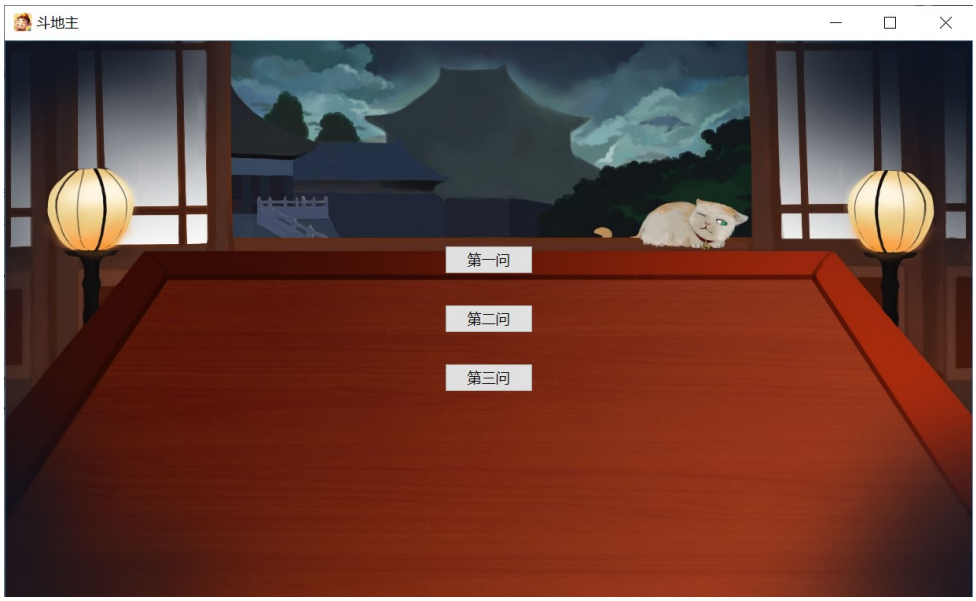

图 1 启动界面

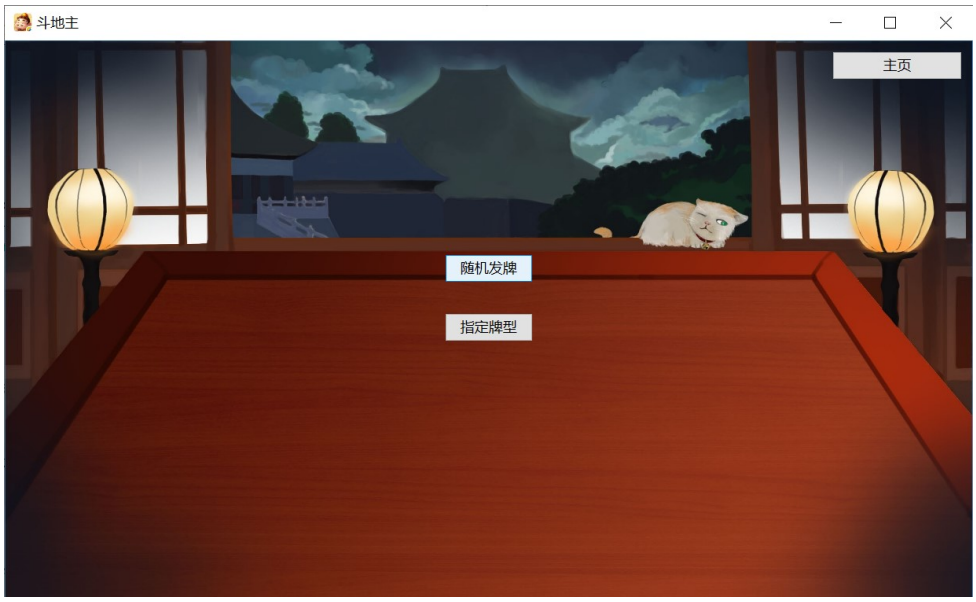

图 2 选择发牌方式界面

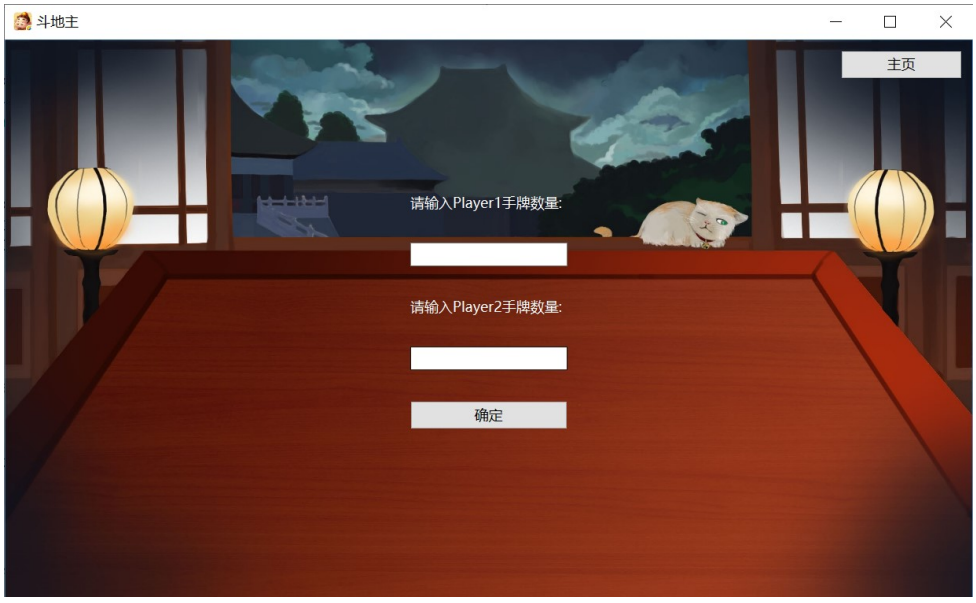

图 3 输入两个 player 的手牌数量的界面

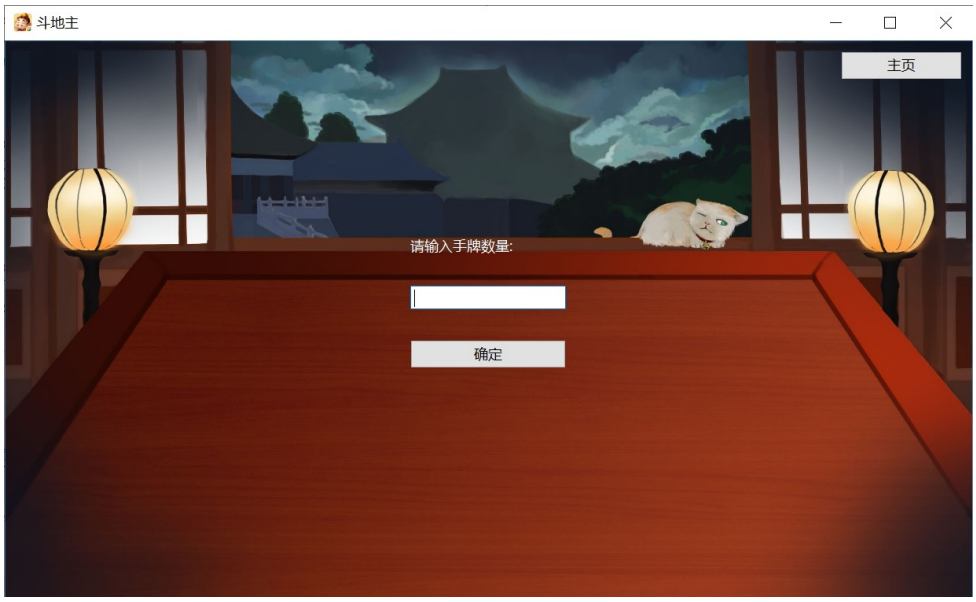

图 4 随机发牌界面

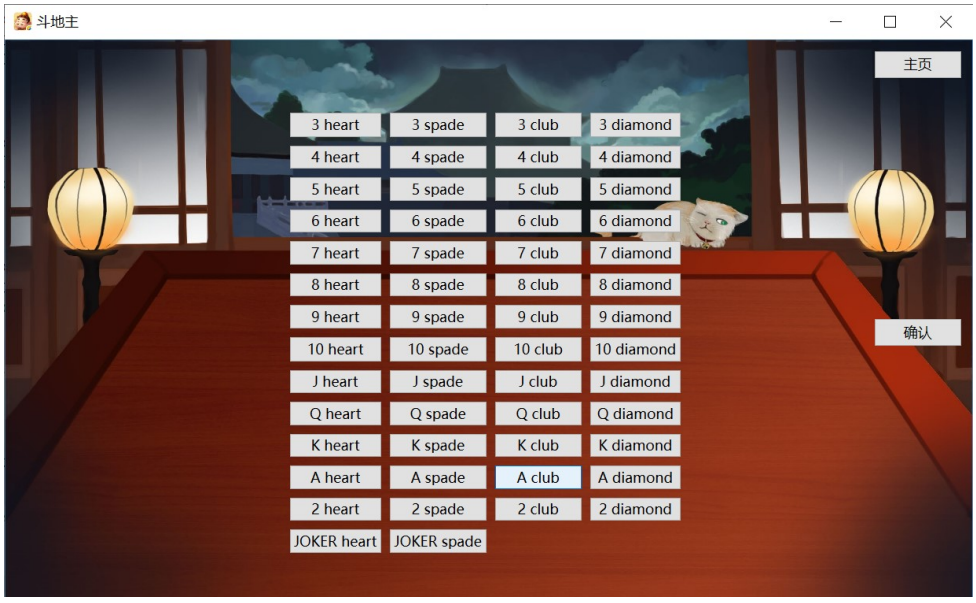

图 5 指定牌型界面

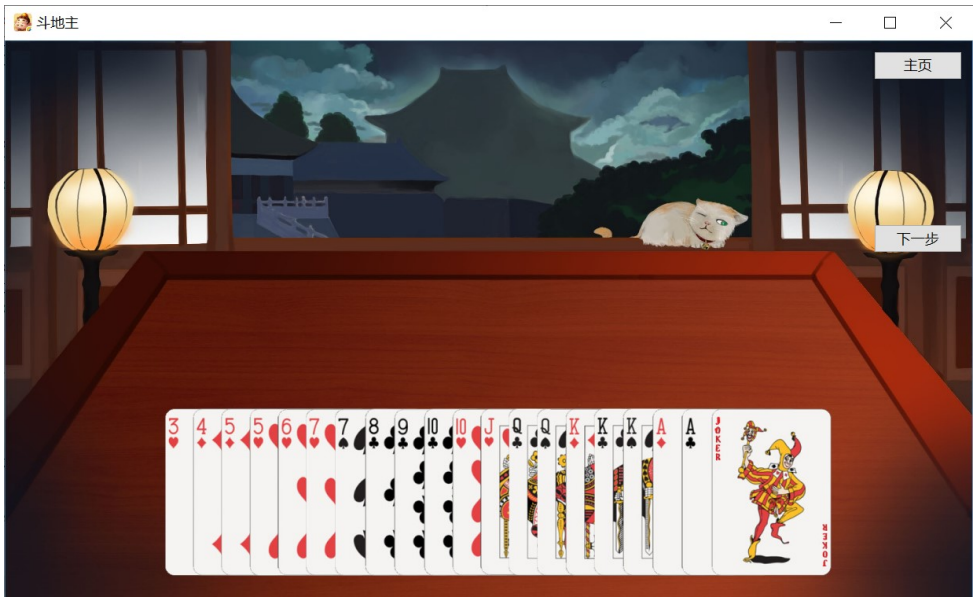

图 6 第一问和第二问的出牌界面

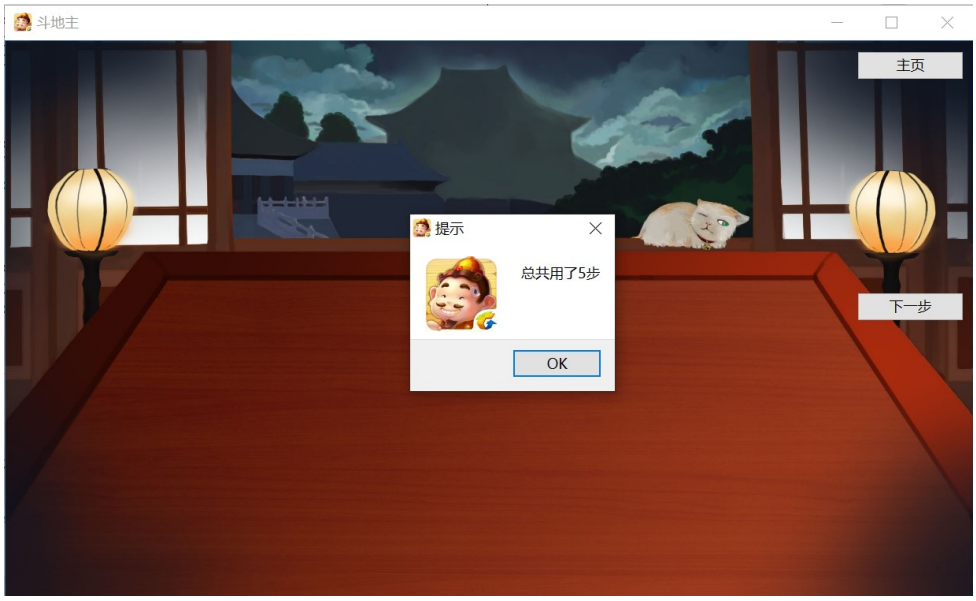

图 7 题目一提示框

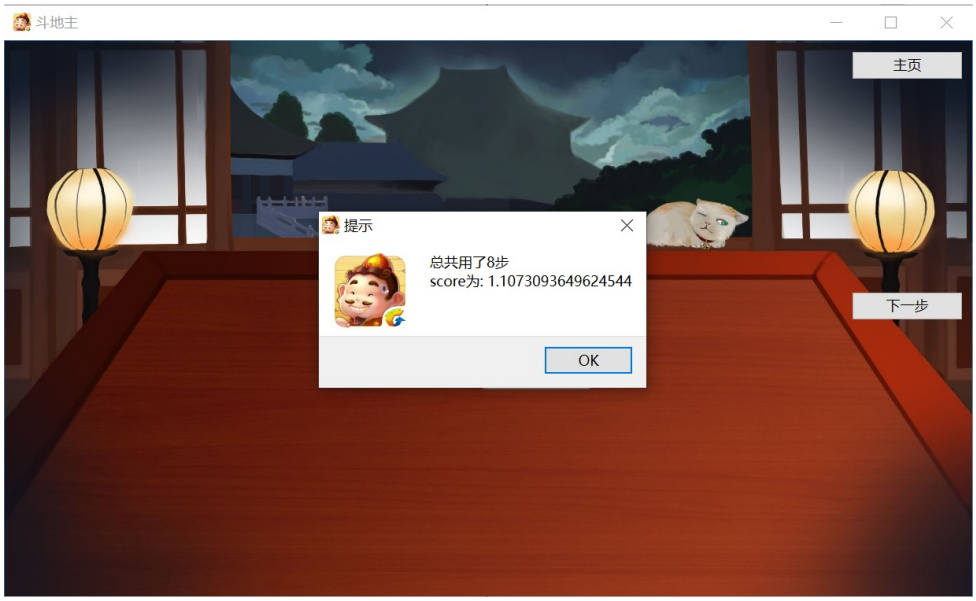

图 8 题目二提示框

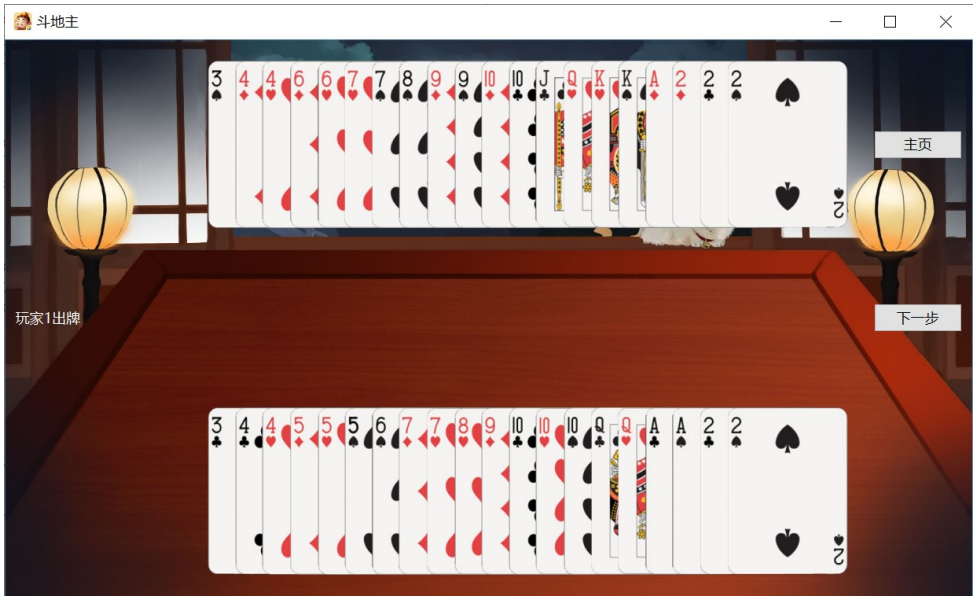

图 9 1v1 对战界面

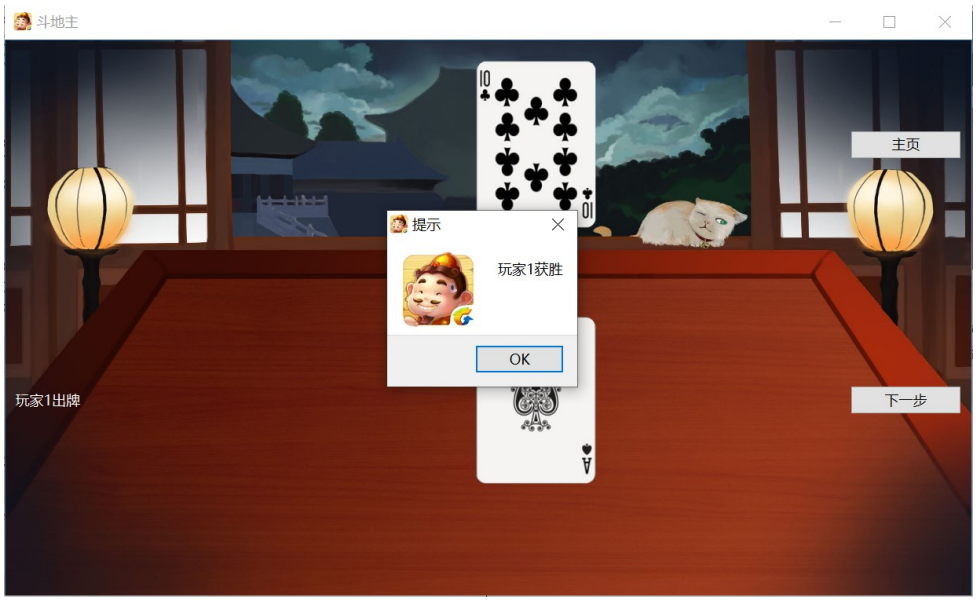

图 10 对战胜者提示框

点击启动界面的第一问和第二问按钮，都会进入到图2选择发牌方式界面中，启动界面中选择第三问按钮会进入到图3的题目三的随机发牌界面，在所有界面中点击主页按钮会回到图1启动界面。

在图2界面中选择随机发牌和指定牌型分别会进入图4和图5界面中。在随机发牌界面输入要发的牌数，或在指定牌型界面选择想要输入的手牌后点击确认按钮，会进入图6的出牌界面中（由于计算出牌步骤步骤是在这一阶段完成的，如果输入的牌数过多会有所卡顿），在出牌界面点击下一步按钮可以得到本步出的手牌和剩余手牌，出完所有手牌后会弹出提示框，题目一会显示出牌的总步数，如图7，题目二会显示出牌的总步数和最终的 score，如图8。

在题目三的随机发牌界面中指定双方的手牌数量后点击确认，会进入图9进入 1v1 对战界面，在界面中点击下一步按钮可以看到下一步出牌方式，如果一方出完了所有手牌，会弹出提示框显示胜者，如图10。

3 总结

本次作业是第一次利用上课所学的算法对一个实际的问题进行处理，让我深刻体会到了理论和实际之间的区别，算法理论更多的是在宏观层面上考虑问题，但处理实际问题时，会需要在各种地方进行多方面的改进，各种细节部分也需要进行很多的调整，让算法真正落到实处仅仅用上课所学的那一部分知识是远远不够的，跟需要具体问题具体分析，这也是我能力中所需要提升的很重要的一个方面。