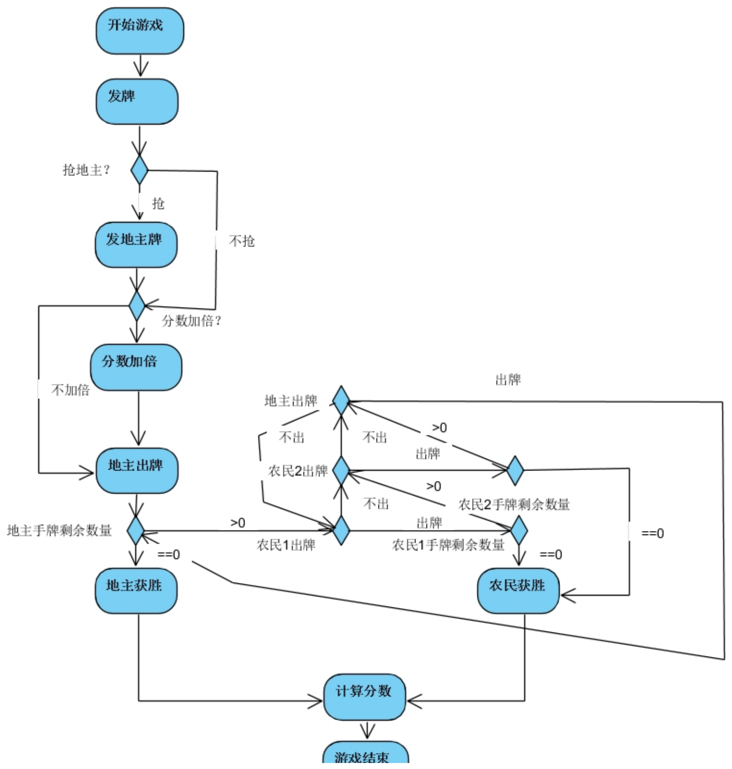
叶崇南 95448916

1. Activity Diagram



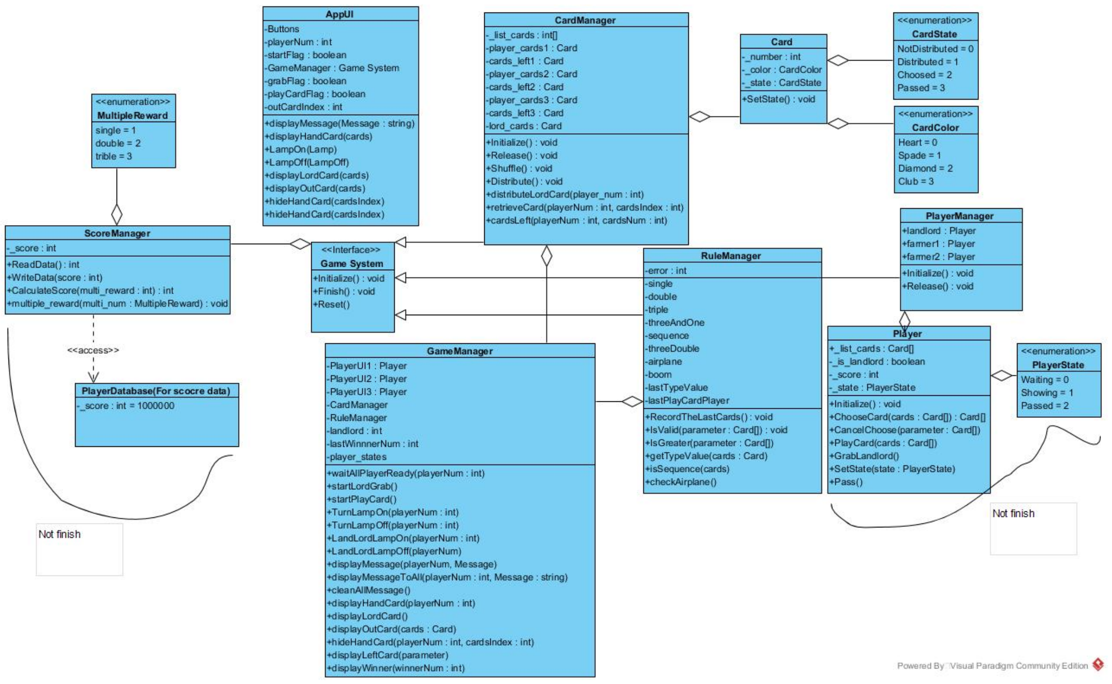
Once the player clicks the “开始游戏” button, the system will Shuffle poker cards into four groups: 3 groups with 17 cards and a group with 3 cards (landlord cards). The three main groups of card will be randomly distributed to three players. In the version 1.0 software, to simplify the rule and implementation, the players would be the landlord once he click “抢地主” button. If he do not want to be the landlord, clicking “不抢” the system will set one of the other players to be the landlord.

After the landlord is chosen, each player has an option to choose whether to double the score reward. Click “加倍” will double the reward and also penalty if the player lose. Click “不加倍” button to let the reward stay the same.

The above step is initialization of the game, then we are going to play the poker game and it’s like an infinite loop until one of the player card is empty. First of all, the play who is landlord plays card he wants. Secondly, the other player could play cards with rules (greater than the previous), or choose not to play card and let the next player play.

If anyone’s poker cards are empty, the winner will gain reward, the loser will get penalty and the game is over.

2. Class Diagram



This graph is the main structure of the software system. Here are many classes named with *Manager which are the "manager" to manage specific ingredient of the system. Then we discuss these in details:

class GameSystem:

This is the kernel of the software and also the key for user to interact with game system by clicking UI button. Once the software is running, The system will use ScoreManager and use ReadData() function to read the score data for each player and show it on the screen.

If the player click "开始游戏", the GameSystem class will call Initialize() function to initialize game, including initialing the other Manager class and start the game.

class CardManager:

The part which manipulate poker cards, including

Shuffle(): Shuffle the cards randomly into 4 groups (3 groups with 17 cards, 1 group with 3 cards) once the game starts.

Distribute(): Distribute 3 shuffled cards to 3 players.

DistributeLeftCard(): Distribute the left 3 cards to the landlord.

class Card:

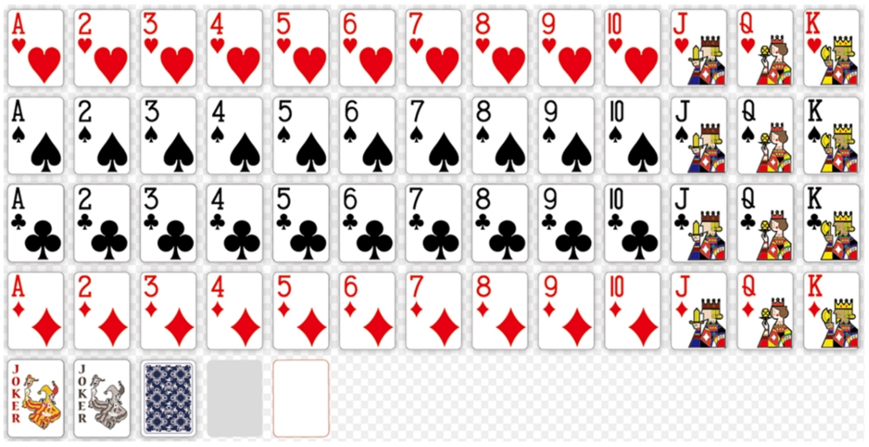
_number: integer from 3, 4, 5...10, J, Q, K, A, 2, middle joker, big joker

_color: enumerate including Heart, Spade, Diamond, and Club

_state: enumerate including NotDistributed, Distributed, Choosed, Passed

setState(): public API to set the card's state

Card UI:



class RuleManager:

Manager to hold the game rule which will be discussed later. Also, the class and function in diagram is only a simple routine to guide the coder writing the code. The details should be implemented individually and will be communicated in further version of requirement analysis document.

RecordTheLastCards(): record the cards which are played by the previous player. This is useful for later comparison.

IsValid(Card[]): Decide if the cards that are going to be played is valid. If not, ban the player's operation.

IsGreater(Card[]): Decide if the cards that are going to be played is greater than previous one in the rule. If not, ban the player's operation.

Rules:

Key part: The cards you are going to play should be larger than the previous one. If there's not the previous cards, the option could be casual.

single card: 3<4<…<10<J<Q<K<A<2<middle joker<big joker

pair card: Two cards with the same number. The comparison is the same before.

triple card: Three cards with the same number. Could followed with casual single or pair cards.

sequence cards: sequence of cards which have at least 5 single cards.

sequence pair cards: sequence of cards which have at least 3 pair cards.

sequence triple card: sequence of cards which have at least 2 triple cards. Could followed with two casual single or two pair cards.

four card: Four cards with the same number. Greater than the previous rules. Double the reward if it's played.

rocket: the largest one with two cards: middle joker and big joker

class PlayerManager:

Class to administrate three players and interaction between them.

TODO: Something confusion here, should be discussed with coder in the future.

class Player:

_list_cards: Each player has its poker cards which are distributed at the beginning

_is_landlord: If this player is the landlord.

_score: Player's score in the memory. It should be written back to database when game is finish.

_state: enumeration include Waiting, Showing, Passed.

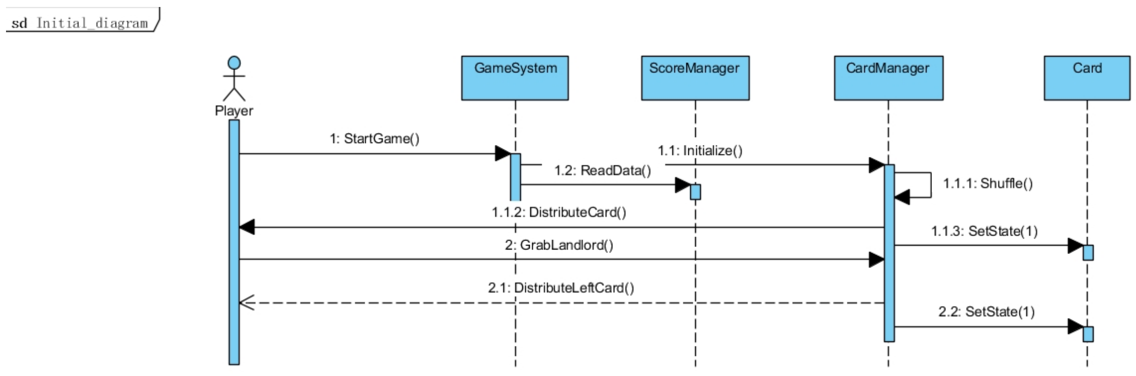
ChooseCard(): Choose the cards you want to play

CancelChoose(): Cancel choosing the card.

PlayCard(): Play the cards you choose.

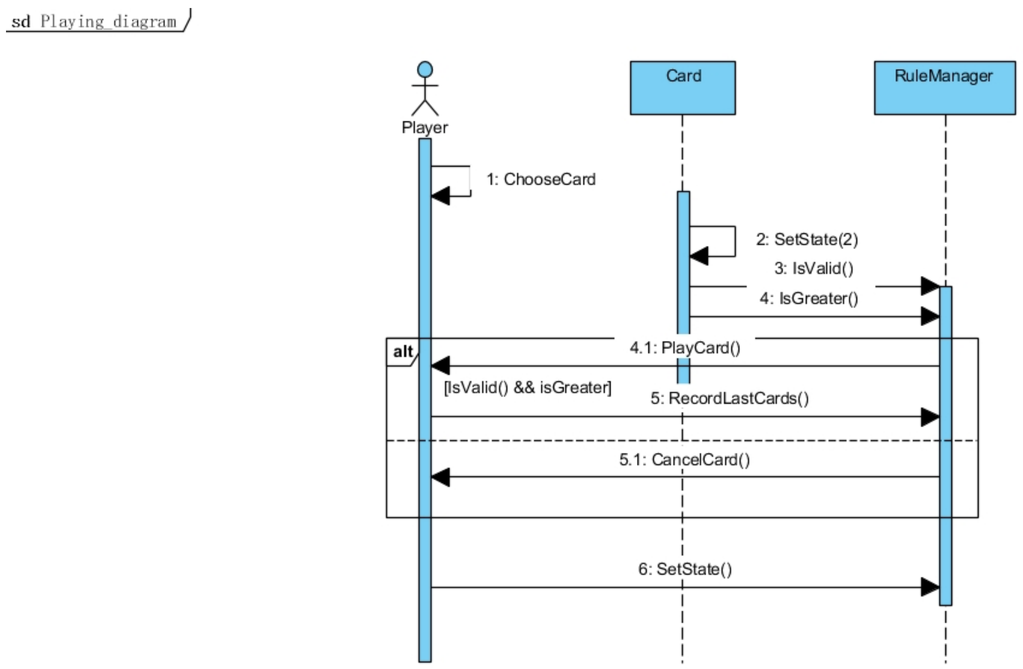
3. Sequence diagram

3.1 Initial Sequence diagram



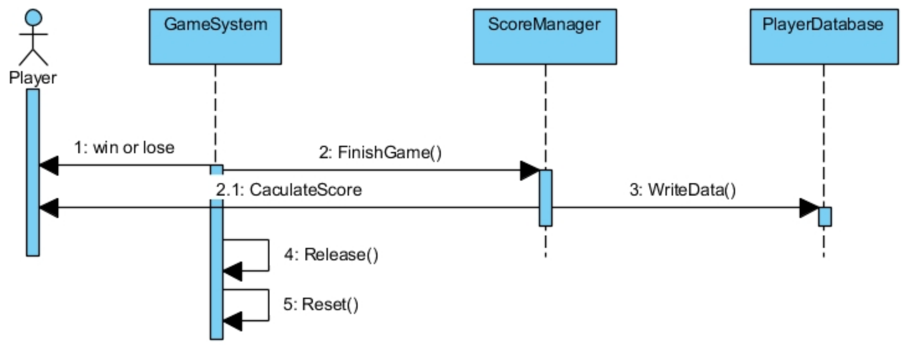
The logic of the game is divided into three parts: Initial part, playing part and finish part. This diagram is initial part which indicates the sequence of activities which describe what will happen at the beginning of the game.

3.2 Playing Sequence diagram



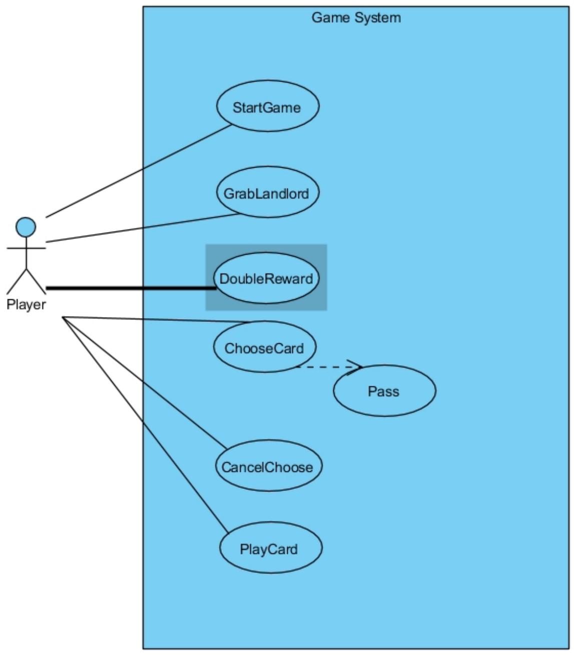
This is playing sequence diagram. If it's player's turn, firstly, he should choose card to play. The RuleManager will check if the cards is valid and greater than the previous one in rules. If it's, Player could play card and switch the turn, else he should choose again or pass.

3.3 Finish Sequence diagram



This part is final. Once the game is over, The ScoreManager will calculate the final score for each player and write the data back to database.

4. Use case



These are the main button, also interface the player can interact with.

5. Requirement prioritization
    1. Basic component of class
    2. Implementation of all of the rules
    3. UI
    4. Network play
    5. AI

6.System iteration plan:

1. version 1.0: The game can run without UI. The players could be three client programs.
2. version 1.1: Game with UI

3. version 2.0: Game can be play through network
4. version 3.0: Game can be play with AI.

Iteration 0 refine:

The basic requirements of poker game. Three players can see their own cards, cards on the table and the number of cards of other player. At the beginning of the game, three players can grab the landlord. While playing , the system can check whether the cards are valid. Once one of the player finished playing his cards, the system can verify who are the winners.

So basically four requirements:

R1: The player can start the game.

R2: The player can see the cards information, including his own hand cards, the cards on the table,and the number of cards of other player.

R3: The player can grab the landlord

R4: The player can play the cards, the system can verify whether the cards are valid.

R5: The player can know whether he wins or not.

Iteration 1 plan:

Implement database of player and score. And also the interaction between the game system and database.

Iteration 1 refine:

Successfully implement the requirement above

Refine:

R1: The player can start the game.

R2: The player can see the cards information, including his own hand cards, the cards on the table,and the number of cards of other player.

R3: The player can grab the landlord.

R4: The player can play the cards, the system can verify whether the cards are valid.

R5: The player can know whether he wins or not.

R6: The player can register or login his information into the database, the database will store his information and score.

R7: At the end of the game, the players scores will be adjusted , update to the database and display to all the players.