Poker Game

Specification

Group 25
Author: 王子瑜 15355832

Iteration plan 0 : 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 plan 1 : Add a database with players' information. At the beginning of the game, the player can register his information to the database. Add a score system, at the end of the game, the system will adjust players' score based on whether he won this game., Then players' score will be updated to the database.

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.

Software Specification

To achieve these requirements, we designed four classes, which are GameManager, CardManager, RuleManager and PlayerUI.

GameManager: the engine of the game, integrate the other three parts, determine The progress of the game.

CardManager: manage the players' hand cards, do cards shuffle and cards distribution.

RuleManager: verify whether the cards that players play out is valid or not.

PlayerDB: store the players' information, their names and scores.

PlayerUI : The user interface of the players, displays the cards information and the process of the game to the player, all the players' operations are on the PlayerUI.

Brief Specifications:

Here we first lists some brief specificaions , the details can be found in the *Implementation details* part.

S1: Start game

There is a 'Start Game' button on the PlayerUI. While the player clicked, the GameManger will get the information that this player is ready. When all three players clicked the 'Start Game' button, the game begin, and their own hand cards will be displayed.

S2: Display Cards information

All the cards information, including his own hand cards, the cards on the table,and the number of cards of other player are displayed by the PlayerUI. Note that, the PlayerUI is just for displaying, which cards to display is determined by the GameManager.

S3: Landlord grabbing

The landlord grabbing is implemented in the GameManager and PlayerUI. Players clicks the button on the PlayerUI to chooces to grab the landlord or not. The GameManager will track the player choices, which is the value of the buttons, to determine the landlord.

S4: Card verification

While the player has chosen which cards to play on the PlayerUI, he can click the 'Play' button, the information of the cards (indices) will sent to GameManager. Then, the GameManager retrieves the cards with its information from the CardManager. After that, send the cards to the RuleManager for verification. If the cards is invalid, the PlayUI will display the message to the player.

S5: Winner check

Once a player plays some cards, the GameManager will track the hand cards remaining of the player. If it turns to be 0, GameManager will determine the winner, and display the information on the PlayUI.

S6: Player register or login

At the begin of the game , the players need to enter a name into the playerUI, if his name already in the the database, the database will retrieve his score, and display his information. If his name not in the database, he will be automatically registered into the database and has a initial score, which is 0.

S7: Adjust players score.

When one player plays out all his cards, the GameManager will determine the winner and lose of the game, then adjust their score into the database and display the variation of the scores to all players' playerUI. If farmers win, their score +3 , landlord's score -6. If landlord wins, his score +6, for each farmer's score -3.

Implementation details:

The following the code document, which shows the details of implementations.

Class: **GameManager** (handle)

properties

PlayerUI1 **% the UI of the players**

PlayerUI2

PlayerUI3

CardManager

RuleManager

PlayerDB

landlord %the landlord of this round of game

lastWinnnerNum = 1 %the winner of last round of game, default player 1,

%Use for determining who will start the next round of landlord grab

player_states = [0,0,0];

%init [0,0,0], when the ith player click the 'Start Game' button,

%His states player_states(i) changes to 1

grabRound = 1; %the round of the grabbing landlord, max 2, initially 1

grabSequence = []; % the sequence of the players to grab the landlord

end

methods

function waitAllPlayerReady(obj,playerNum)

Wait for all of the three player to click the 'Start Game' button, set the states of the player to 1 corresponding to the playerNum .

When all players' states are set to 1, which is [0,0,0] -> [1,1,1], the game starts . The own hand cards of the three players will be displayed .

Then the landlord grab begins.

function startLordGrab(obj)

Start grabbing the landlord , the initial landlord is the player corresponding to the lastWinnerNum, he can choose to be the landlord or not, then the rest two players can grab the landlord.

There landlord grab will be two rounds at most. At the begging, The order of grabbing the landlord will be determined. Eg. Player 2 is winner of the last round game, the order of grabbing is [2, 3, 1].

There is three 'TurnLamp' in each player UI when it turns to green, means that now is the corresponding player to make a choice. So at the beginning , turn the landlord's Lamp on and make his GrabLandLordPanel to be visible in the playerUI.

Note that: <u>in our landlord grabbing , the winner of the last round game don't have the button ' I want to be the landlord' or 'not', but he has the highest priority to grab the landlord.</u>

function processGrabLordInput(obj,playerNum,grabFlag)

The function will be called when the player clicks the 'Grab the landlord' button or 'Not grab' button. Then ask the next player to grab the landlord.

At beginning, it will check the grabRound.

If grabRound = 1:

When player clicks the 'Grab the landlord'button , the grabFlag = 1, he can be in the second round of the landlord grab. The GameManager will ask the next player whether he want to grab or not.

When player clicks the 'Not grab'button, the grabFlag = -1, he will not in the

second round of the landlord grab, his corresponding playNum in the grabSequence will be set to 0. Eg. [2,3,1] -> [0,3,1] if player 2 decides not to be the landlord.

Note: If no player choose to grab the landlord at first round , the cards will be reshuffled, their handcard will be redistributed. The grabbing landlord part will restart.

If grabRound = 2:

if the player chooses to grab the landlord, the landlord grabbing finishes, he becomes the landlord. Eg. Order [0,3,1], if player3 chooses to becomes the landlord, he becomes the landlord, and player2 will not be asked to grab the landlord

After the landlord is determined, display three landlord cards to all the players, display who is the landlord and add these three new cards to the landlord's hand cards.

function startPlayCard(obj)

Their is a TurnLamp' will also show that who is playing the cards at this moment.

Make the landlord's PlayCardPanel to be visible, display the message " Please play cards" to him. And make his TurnLamp to be on to all players, which shows that it is his turn to play cards.

function processPlayCard(obj,player_num,playFlag,cardsIndex)

The function will be called when the player click the 'Play' or 'Pass' button.

When the player clicks the 'Play' button on his playerUI, the playFlag = 1 . The GameManager will get the information of the cards he chose using the cardsIndex based on CardManager. Then sents the cards to the RuleManager to verify if the cards are valid. If valid, display the cards on the tables of all the players' UI, adjust the cards left of the corresponding player; if not , ask the player to re-select cards.

When the current playing-card player clicks the 'Pass'button, the playFlag = -1. The message ' Pass' will be displayed to all players, then make the next player to play cards.

If one player' cards left turns to 0, the GameManager will determine who is the winner of the game, adjust their score and record the winner as the lastWinnerNum.

function score = retrieveScore(obj,playerNum,playerName)

Retrieve the score of the player from the databse given the playerName.

If the players name not in the database, he will be automatically registered.

function logged = isNameLogged(obj,playerName)

Check whether the player is logged or not in the current game.

function reshuffleCard(obj)

Reshuffle all the cards of the player, update their handcard, and display the message "The cards reshuffled.

function TurnLampOn(obj,playerNum)

Now it is the turn of the player corresponding to the playerNum, turn his correponding 'TurnLamp' on at all the three player UIs.

function TurnLampOff(obj,playerNum)

Turn the ‘TurnLamp’ of the player corresponding to the playerNum off at all the three player UIs.

function LandLordLampOn(obj,playerNum)

The ‘playerNum’ is the landlord, turn the ‘LandlordLamp’ on at all the three player UIs.

function LandLordLampOff(obj,playerNum)

Turn the ‘LandlordLamp’ according to playerNum off at all the three player UIs.

function displayMessage(obj,playerNum,Message)

Display the ‘message’ on the playerUI corresponding to the playerNum.

function displayMessageToAll(obj,playerNum,Message)

Display the ‘message’ of the player corresponding to the playerNum for all three playerUIs. Such that the other two players can see the decision that the (playerNum)th player did.

function cleanAllMessage(obj)

Clean all the message of the three playerUIs.

function displayHandCard(obj,playerNum)

Display the hand cards of the player corresponding to the playerNum on his playerUI.

function displayLordCard(obj)

Display the three landlord cards to all the playerUIs.

function displayOutCard(obj,cards)

Display the ‘cards’ that the player played on the table to all the three playerUIs.

function hideHandCard(obj,playerNum,cardsIndex)

‘cardsIndex’ is indices of cards that the (‘playerNum’)th player chooses to play, hide the corresponding hand cards on the corresponding playerUI.

function displayLeftCard(obj)

Display the hand cards left of each player, on all the three playerUIs.

function adjustment = adjustScore(obj, winnerNum)

Adjust all the players scores based on the winnerNum.

function displayResult(obj,adjustment)

Display the win or lose for all the players and their score adjustments on each playerUI.

Class: **CardManager** (handle)

properties

list_cards =

[3 4 5 6 7 8 9 10 11 12 13 14 15 3 4 5 6 7 8 9 10 11 12 13 14 15 3 4 5 6 7 8 9 10 11 12 13 14 15 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17;

1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 5 6];

% The total cards of the game are saved in a 2*54 matrix, the first row is the cards

%number, note that 'A' = 14 , '2' = 15 , 'Joker(small)' = 16 , 'Joker(big)' = 17

%The second row is the color of the cards.

%' '♥ ' (Spade) = 1, '♠ '(Heart) = 2, '♣'(Club) = 3, '♦'(Diamond) = 4

% 'Black'(small joker) = 5 , 'Red'(Big joker) = 6,

player_cards1 %The cards of the player 1 (2*N matrix)

cards_left1 %The number of remaining hand cards of player 1

player_cards2

cards_left2

player_cards3

cards_left3

lord_cards % The three landlord cards (2*3 matrix)

end

methods

function obj = shuffle(obj)

Shuffle the cards stored in the list_cards, set the lord_cards to be the last three cards (52:54 of list_cards) in the shuffled list_cards.

function obj = distribute(obj)

Distribute the shuffled cards to the player_cards1, player_cards2 and player_cards3.

The distribute method is that:

1:17 cards of the list_cards to the player_cards1

18:34 cards of the list_cards to the player_cards2

35:51 cards of the list_cards to the player_cards3

Then sort the player_cards1, player_cards2 and player_cards3 respectively.

function obj = distributeLordCard(obj,player_num)

Distribute the landlord cards to the player_cards corresponding to the player_num, then sort it.

function cards = retrieveCard(obj, playerNum, cardsIndex)

Retrieve the cards stored in the player_cards according to ‘playerNum’ and ‘cardsIndex’.

Note that: ’cardsIndex’ stores the indices of the cards in the player_cards that the player wants to play.

function cardsLeft = updateLeftCard(obj,playerNum,cardsNum)

Update the cards_left corresponding to the ‘playerNum’, ‘cardsNum’ is the number of cards have been played. Return the new cards_left.

Eg. playerNum = 1, cardsNum = 10 , So set cards_left1 = cards_left1 - cardsNum, return the new cards_left1.

Class: **RuleManager** (handle)

properties

error = 0; %Type : error

single = 1; % Type: single card (单张）

double = 2; %Type: two identical cards (对子）

triple = 3; %Type: three identical cards (三张)

threeAndOne = 4;

%Type : three identical cards and one arbitrary card(三带一)

sequence = 5;

%Type : five consecutive cards, no ‘2’ or two ’Joker’s (五连顺）

threeDouble = 6;

%Type : three consecutive sets of two cards(三连对)

airplane = 7;

%Type: two consecutive sets of three cards and two arbitrary cards (飞机)

boom = 8; %four identical cards or a pair of ‘Joker’ (炸弹 或 王炸）

fourAndTwo = 9; %四带二

lastTypeValue = [-1,-1];

% Store the last played cards type and its value

%initially [-1,-1], the first element is the type, the second element is the cards value of that type.

lastPlayCardPlayer = 0; %the playerNum of player who played the last time

end

methods

function valid = isValid(obj,cards,playerNum)

First, call the getTypeValue() function to get the type and the value of the input cards.

Then, call the isGreater() function with the cards’ type ,value and who plays them(playerNum) to check whether the cards is greater than the last played cards.

If it is greater, return 1; smaller or not valid, return 0

function typeValue = getTypeValue(obj,cards)

Input cards, return its type and value. ([type , value])

The way to get its type and value:

First check the number of the cards, the valid input number of cards should be between 1 and 8. Then for the corresponding number of cards, check its type.

If the type is valid, give it a value:

single, double, triple : value = the first card number

boom (four identical cards): value = the first card number

boom (a pair of jokers): value = 9999

threeAndOne : value = the card number of the three identical cards.

sequence : value = the largest card number of the five cards.

threeDouble: value = the largest card number of the six cards.

airplane: value = the larger card number of the two consecutive sets of three cards.

If the type is not valid , value = 0.

function result = isGreater(obj,typeValue,playerNum)

These function check the input typeValue with the lastTypeValue (the last played cards' type and value).

First , the function will check whether the type is valid. If not, return 0.

Second, check whether playerNum is the same as the lastPlayCardPlayer.

If they are equal, means that in this round of cards playing, other two players didn't play. So the cards is valid, update the lastTypeValue to the input typeValue, return 1.

If they are not equal, go to the third step.

Third, check the type of typeValue and lastTypeValue.

If their type is the same, compares their values. If the input typeValue is greater than the lastTypeValue, update the lastTypeValue and lastPlayCardPlayer, return 1; if not , return 0.

If their type is different, return 0.

Class: **PlayerUI**

properties

playerNum; %The player's number, which will be set in class constructor .

GameManager;

outCardsIndex;

% The matrix which stores the cards indices that the player wants to play, it is corresponding to the Button_1 to Button_20 in the HandCardPanel.

StartGameButton:

% When clicked, the call back function will display the other panels, and call the function waitAllPlayerReady() in the GameManager with the playerNum.

OutCardPanel:

OutCard_1 ;OutCard_2 ;OutCard_3;OutCard_4 ;
OutCard_5 ;OutCard_6 ;OutCard_7 ;OutCard_8 ;

% Eight texts to display the cards that players play on the table. Each text can display one card.

MessagePanel:
Message
% the text of the playerUI's message.(The first player, me)

MessagePanel_2:
Message_2
% the text of the second player's message

MessagePanel_3:
Message_3
% the text of the third player's message

HandCardPanel:
Button_1;Button_2 ;Button_3 ;Button_4 ;Button_5;
Button_6 ;Button_7 ;Button_8 ;Button_9;Button_10;
Button_11 ;Button_12;Button_13;Button_14;Button_15;
Button_16; Button_17; Button_18; Button_19 ; Button_20 ;

% 20 buttons, each button's text can display one hand card information,
% If the button is pressed, its value is 1, means that the card of the button is being chosen by the player.
% If the button is released, its value is 0, means that the card of the button is not chose.

PlayerNumText
%the text of the playerNum who using the playerUI, set in class constructor.

TurnLamp
% if it is turn to green, means that now is the first player(user) to make choice.
TurnLamp_2
% if it is turn to green, means that now is the second player to make choice.
TurnLamp_3
% if it is turn to green, means that now is the second player to make choice.

PlayButton
% if the button is clicked, the call back function will set the playCardFlag to 1, and scan the Button from 1 to 20 in the HandCardPanel to record which cards are being chosen, then update the outCardsIndex.

PassButton

% if the button is clicked, the call back function will set the playCardFlag to -1

CardsLeftPanel_2:

CardsLeft2

CardsLeftPanel_3:

CardsLeft3

% The hand cards remained of the second player and the third player

LandLordLamp

LandLordLamp_2

LandLordLamp_3

% display which player is the landlord

EnterNamePanel

EnterNameEditField

%Enter the player names

ConfirmButton

%Confirm the name, send the name to GameManager.

EnterNameMessage

%Display the message whether the name is valid and whether the name has been %logined in the current game.

GrabLandLordPanel:

GrabtheLandlordButton

% If clicked, the grabFlag sets to 1

NotGrabButton

% If clicked, the grabFlag sets to -1

LordCardPanel:

LordCard_1

LordCard_2

LordCard_3

%display the three landlord cards

ResultPanel

%Display the result of the game, including the winner and loser, and their score %adjustment.

methods

function displayMessage(app,Message)

%display the message on the Message textArea of the MessagePanel

function displayHandCard(app,cards)

% display the hand cards on buttons of the HandCardPanel.

%The input cards is sorted in a 2*17 or 2*20 matrix.

%The first row are the numbers, the second row are their colors.

%Display the cards on the button's text. The indices of the button is the same as the %indices of the cards in the GameManager.CardManager.player_cards

function LampOn(app,Lamp)

% change the input Lamp.color to green.

function LampOff(app,Lamp)

% change the input Lamp.color to grey.

function displayLordCard(app,cards)

%display three landlord cards

function displayOutCard(app,cards)

%display the played cards on the table(OutCardPanel)

function hideHandCard(app,cardsIndex)

%Set the hand card button.Visible to 'off'

function displayLeftCard(app,player2Left,player3Left)

%display the hand cards remained of the second player and the third player

function displayResult(app,scoreList,adjustment)

%Display the winner and loser, and their score adjustment on the ResultPanel.

Class: **PlayerDB** (handle)

function register(obj,playerName)

%Register the a new player given the playerName, give it a initial score 0 .

function score = retrieve(obj,playerName)

%retrieve the player score given the playerName from the database.

function update(obj,playerName,score)

%update the player score to database given the playerName.