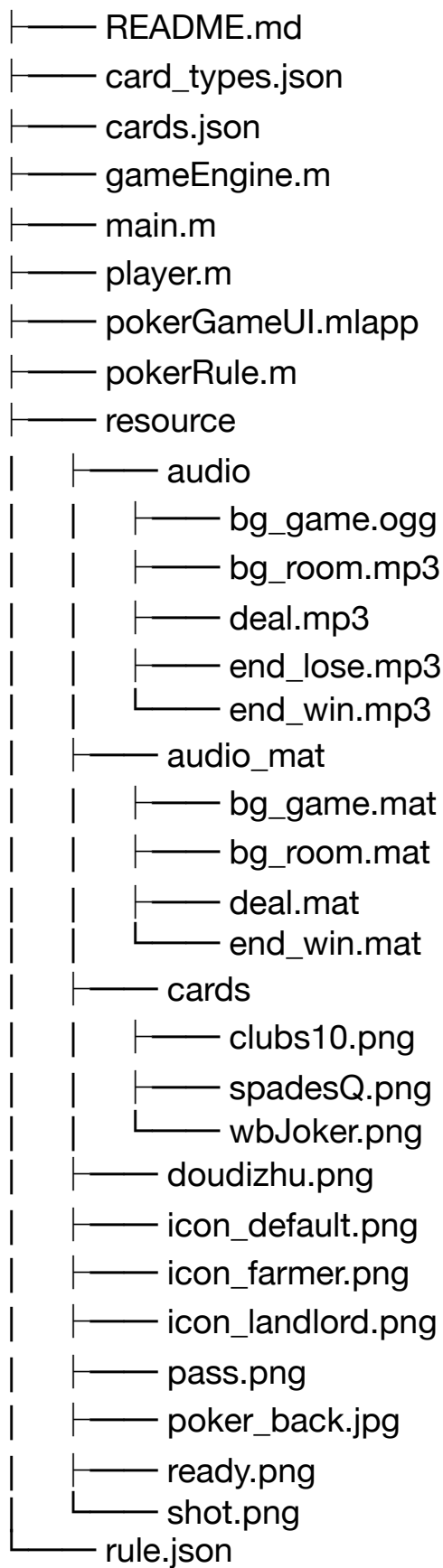


Poker Game Specifications

Program structure	2
Class: pokerRule	3
Properties	3
Methods	4
Class: gameEngine	5
Properties	5
Methods	7
function assignRole(eg)	7
function distributeCards(eg)	7
function determineWinner(eg)	8
function sortCard(eg, player)	8
function displayCard(eg)	8
function dispShotCards(eg)	8
function startGame(eg)	9
function endGame(eg)	9
Class: player	10

Program structure



Class: pokerRule

```
1  classdef pokerRule < handle
2      % properties of poker game rule
3      properties
4          % 'rocket', 'bomb',
5          cardType = ["single", "pair", "trio", "trio_pair", "trio_single",
6          cardRule = jsondecode(fileread('rule.json'));
7          compare_result % -3 -> found; -2 -> not found; -1 -> unkown type;
8          gameEngine
9      end
10
11     % methods of poker game rule
12     methods
13         % find the indx, which represents magnitude, and cards type
14         function index_of(pkRule, typeName, ele) ...
33        % determine the cards type
34        function card_type(pkRule, cards) ...
59        % determine cards value: return cards type and value
60        function cards_value(pkRule, cards) ...
77        % compare the poker hands and set compare value
78        function compare_poker(pkRule, preCards, selectedCards) ...
137    end
138 end
```

Properties

1. **cardType**: 储存所有的斗地主牌型的数组，内容为下：

```
"single", "pair", "trio", "trio_pair", "trio_single",
"seq_single5", "seq_single6", "seq_single7", "seq_single8", "seq_single9",
"seq_single10", "seq_single11", "seq_single12",
"seq_pair3", "seq_pair4", "seq_pair5", "seq_pair6", "seq_pair7", "seq_pair8",
"seq_pair9", "seq_pair10",
"seq_trio2", "seq_trio3", "seq_trio4", "seq_trio5", "seq_trio6",
"seq_trio_pair2", "seq_trio_pair3", "seq_trio_pair4", "seq_trio_pair5",
"seq_trio_single2", "seq_trio_single3", "seq_trio_single4", "seq_trio_single5",
"bomb_pair", "bomb_single"
```

2. **cardRule**: 由 rule.json 导入。

遍历所有可能斗地主打出手牌的可能性，以牌型归类。同一类别内，手牌按规则从小到大排列，即某一手牌在某一手牌类型中的Index代表了该手牌的大小，Index 越大，则手牌越大。

3. **Compare_result**: 储存当前两组牌型对比结果

- 3: found, 已在**cardRule**找到;
- 2: not found, 未在**cardRule**找到;
- 1: unkown type, 非法牌型;

0 : not bigger, 当前牌组小于上一轮牌组;
>0 : bigger, 当前牌组大于上一轮牌组;

4. **gameEngine**: 用于接收当前牌组及信息同步。

Methods

1. *function index_of(pkRule, typeName, ele)*

输入对应牌型名称, 待查找 牌组字符串。若查找到, 将当前牌组代表的加权值及牌型赋值给 *gameEngine.cards_value_selected*, *gameEngine.cards_type_selected*, 设置 *Compare_result*; 若未查找到, *Compare_result* 赋对应的值, 返回函数。

2. *function card_type(pkRule, cards)*

调用 *function index_of(pkRule, typeName, ele)* , 判断 *Compare_result* 的值, 若找到, 函数结束; 若未找到, 在相应玩家的 UI 的 *UnknownTypeLabel* 显示相应的系统信息, 并且报错 (该报错为主动吊起, 非程序崩溃), 在命令框显示当前牌型。
~~~~~

### 3. *function cards\_value(pkRule, cards)*

为当前赋加权值 (*gameEngine.cards\_value\_selected*) , 用于与上一局牌组比较。

- A. 王炸, 单独提出, 赋值: 2000;
- B. 炸弹, 单独提出, 赋值: 1000 + index (即该组排在炸弹数组中的位置)
- C. 其余牌型通过调用 *function card\_type(pkRule, cards)* 设置加权值

### 4. *function compare\_poker(pkRule, preCards, selectedCards)*

比较主函数, 比较输入牌组与上一局牌组的大小。

- i) 调用 *function cards\_value(pkRule, cards)* 确定牌型及牌组加权值
- ii) 若当前牌组和上一局牌组任一为空, 即可判断孰大孰小
- iii) 若为相同牌型, 则比较结果为输入牌型加权值与一局牌组加权值之差
- iv) 其余情况皆为输入牌型加权值小于一局牌组加权值之差, 在相应玩家的 UI 的 *UnknownTypeLabel* 显示相应的系统信息。

# Class: gameEngine

## Properties

```
2      properties
3          % All players and thier apps
4          player_0
5          player_0_UI
6          player_1
7          player_1_UI
8          player_2
9          player_2_UI
10
11         rule          % the instance of poker game rule
12         landlord = -1 % -1 -> defalut
13         whoseTurn = -1; % -1 -> defalut
14         passNum = 0; % passNum should not more than 2
15         cardNum = 54; % Total card Num = 54
16         winner = -1; % winner=-1 -> have not determined winner
17         isEnd = false; % determine whether to end the game
18         isStart = false; % determine whether to start the game
19         isBGM = true; % determine if play BGM
20
21         % In order to sychronize, Distribute cards as soon as one player
22         % push 'Ready'. Then create cards compoents and invisible them
23         % until game started.
24         isDistribute = false;
25
26         % import cards data
27         cardsData = transpose(struct2cell(jsondecode(fileread('cards.json'))));
28
29         % store the cards that has shotted: 0 -> last turn
30         % row 1: str num % row 2: lable
31         % row 3: img source % row 4 num
32         cards_shotted_0 = {};
33         cards_selected = {};
34         % used to compare selected cards with last turn's
35         cards_type_0;
36         cards_value_0;
37         cards_type_selected;
38         cards_value_selected;
39
40         % for bgm
41         bg_room = load('./resourse/audio_mat/bg_room.mat');
42         bg_game = load('./resourse/audio_mat/bg_game.mat');
43         deal = load('./resourse/audio_mat/deal.mat');
44         end_win = load('./resourse/audio_mat/end_win.mat');
45         player; % used to play bgm
46     end
```

### 1. **player\_#, player\_#\_UI**

三位游戏玩家及其游戏界面的实例。

### 2. **rule**

**pokerRule** 的实例，用于实现斗地主规则判断。

### 3. **landlord{-1, 0, 1, 2}**

储存地主的玩家编号，-1为缺省值，{0, 1, 2} 代表对应的玩家编号。

### 4. **whoseTurn{-1, 0, 1, 2}**

储存当前出牌玩家编号，-1为缺省值，{0, 1, 2} 代表对应的玩家编号。

### 5. **winner{-1, 0, 1, 2}**

储存获胜玩家编号，-1为缺省值，{0, 1, 2} 代表对应的玩家编号。

### 6. **isEnd{true, false}**

表示游戏是否结束，false 为缺省值（游戏为开始），true 代表游戏结束。

### 7. **isStart {true, false}**

表示游戏是否开始，false 为缺省值（游戏未开始），true 代表游戏开始。

### 8. **isBGM{true, false}**

表示是否播放背景音乐（音效），true 为缺省值（播放音效），false 代表暂停音效。

### 9. **isDistribute{true, false}**

表示是否已经发牌，false 为缺省值（未发牌），true 代表已经发牌。为实现三个玩家洁面信息同步，当任一玩家点击“准备”按钮，即发牌，但所有的牌为不可见，知道三个玩家都已经切换至“准备”状态。

### 10. **cardsData**

一副扑克牌的所有信息，从 [cards.json](#) 文件导入。[cards.json](#) 文件中储存一副扑克牌中所有纸牌信息，包括其代表的数字（字符），大小（int），图片源文件地址，花色。

### 11.cards\_shotted\_0

储存上一局打出牌组的所有信息。

第一列：字符；第二列：花色；第三列：图片源文件地址；第四列：大小（int）。

### 12.cards\_selected

储存当前玩家尝试打出的牌组的所有信息。

第一列：字符；第二列：花色；第三列：图片源文件地址；第四列：大小（int）。

### 13.cards\_type\_0

储存上一局打出牌组的牌组类型。

### 14.cards\_type\_selected

储存当前玩家尝试打出的牌组类型。

### 15.cards\_value\_0

储存上一局打出牌组的牌组加权值。

### 16.cards\_value\_selected

储存当前玩家尝试打出的牌组加权值。

### 17. bg\_room, bg\_game, deal, end\_win

保存音效的工作区。分别为进入房间的背景音乐，开始游戏的背景音乐，发牌的背景音乐，游戏结束的背景音乐。

### 18.Player

背景音乐播放器实例，通过设置采样幅度值、采样频率来控制播放的背景音效。

## Methods

---

#### *function assignRole(eg)*

引擎为同一房间的三个玩家随机分配角色：地主手牌比平民手牌多三张。

---

#### *function distributeCards(eg)*

与 **Rule** 类似每张纸牌的所有信息存至 **cards.json** 文件，导入程序以供使用。

```

% methods that game engine has:
methods
    % In 0 stage, random assign Role; -1-default, 0-landlord, 1-peasant;
    function assignRole(eg) ...
    % determine which player is winner
    function determineWinner(eg) ...
    % Sort hand cards of players
    function sortCard(eg, player) ...
    % Shuffle and distribute cards
    function distributeCards(eg) ...
    % update related variables in three players and their apps: lable,
    % display cards
    function update(eg) ...
    % display handcards in three UIs
    function displayCard(eg) ...
    % Display the shotted cards in other players' UIs
    function dispShotCards(eg) ...
    % decide whose turn to shot cards
    function nextTurn(eg) ...
    % start game with following process
    function startGame(eg) ...
    % End game with following process
    function endGame(eg) ...
    % control the BGM
    function bgm(eg) ...
end

```

利用Matlab随机生成大小为54，content 为 [1: 54] 的数组。以此为Index，将导入的纸牌信息分发给相应的游戏玩家。

---

### function determineWinner(eg)

若某一玩家手牌首次达0，引擎判定获胜一方，游戏结束。

---

### function sortCard(eg, player)

根据游戏规则制定的大小，为每张手牌设置 int number 属性，据此排列每个玩家的手牌。

---

### function displayCard(eg)

显示手牌。对于每张手牌，在当前玩家的UI里生成对应的Image component。纸牌打出后，该 component 设置为 invisible。

---

### function dispShotCards(eg)

显示打出的手牌，需三个UI同步。当创建每个玩家的UI时，同时在UI中间生成不可见的image components。当有玩家打出手牌时，游戏引擎设置对应图片源，设置components为可见。



---

### **function startGame(eg)**

开始游戏。分配角色，分发手牌。将对应的按钮设置able以及unable。

---

### **function endGame(eg)**

结束游戏。创建每个UI时，UI中间有一不可见的label。游戏引擎判定获胜一方后，该lable可见，并显示获胜信息。

## Class: player

```
1  classdef player < handle
2
3      % properties of poker game engine
4      properties (Access = public)
5          avatar = 'icon_default.png';
6          role = -1;      % -1-default, 0-landlord, 1-peasant;
7          cards = {};     % store the distributed info from *.json
8          cardNum = 0;
9          selectNum = 0; % remember to reset
10         isActive = false;
11         myTurn = false;
12         currUI;         % its own UI
13
14         % row 1 stores the components of cards;
15         % row 2 stores whether the card has selected;
16         % row 3 stores whether the cards has shotted;
17         % row 4 stores the str# of cards;
18         % row 5 stores the lable
19         % row 6 stores the # of cards
20         cards_img = {};
21     end
22
23     % methods that game engine has:
24     methods
25     end
26 end
```