Run step

1. Go main function and get argc and argv
2. GO SANJUAN_MAIN function to parse function and draw(button.c to draw buttons) menu for player to choose
3. If player go setting, they will call setting.c to change some game's parameter like language and resolution
4. If player go play, they can choose(button.c) how many players and how the difficulty is.
5. After entering game, player will be initialize by player_init() (player.c) function and do the normal game process by loops.
6. After game, it will show who is the winner(draw.c) and free all player(player.c) and exit()

Button.c

Because SDL doesn't offer a interface for button. So I encapsulate one and here is how to use.

```
typedef struct _buttonItem{
    char *msg;
    SDL_Rect rect;
} buttonItem

void show_Button(buttonItem *buttons);
int get_ButtonID(int x, int y, buttonItem *buttons);
```

1. You can create a buttonItem array and store massage in the array you want to present and use show_button to present it.

2. You can use get_ButtonID to check which button in the presented buttonItem array is clicked.

Phase_t

I divide the game step to 12 step which decide when player can use there card's effect and what action can a player to do.

Card_t

Card_t is defined in Sanjuandata.c and has follow members

```
typedef struct _card_t{
    SDL_Texture *texture;
    phase_t phase;
    struct _card_t *tokens;
    int victoryPoint;
    int cost;
    building_t buildingType;
    void (*effect)(player_t *player);
    struct _card_t *next;
    char *cardName;
    SDL_Rect pos;
} card_t;
```

It store all necessary information for a card.

For my convenience, I use struct _card_t *next as linking list to present my hand card and ground card.

effect() is for card effect.

building_t is set for distinguishing factory and normal building

Other is obvious so I think I don't have to waste times to explain.

player_t

Player_t is defined in Sanjuandata.c and has follow members

```
struct _player_t {
    card_t *hand;
    card_t *ground;
    int victoryPoint;
    int8_t totalBuilding;
    int8_t privillageTimes;
    int8_t totalPerSell;
    int8_t totalPerProduce;
    int8_t cost_off;
    int8_t maxCost;
    int8_t totalHand;
    int8_t maxHand;
    int8_t playerNum;
    career_t (*chooseCareer)(player_t *player);
    card_t* (*chooseCard)(player_t *player,
card_t **cards);
};
```

It store all necessary information for a player.

hand and ground are linking list.

And by putting different chooseCareer function pointer and chooseCard function pointer. We can decide which player_t object is bot or is player.

Card.c

It defined all operation function for linking_list card and instantiation of all cards and initialize a deck.

```
//for every card's effect
void <buildingName>_effect();
```

```
//load cards' texture. Success return true, else return false
int load_card_texture();
//destroy all cards' texture
void destroy_card_texture();
//load all card_t in the deck, deck is an array for card_t
void init_deck();
//a linking list add element for card_t type
card_t *add_card(card_t *cards, card_t addCard);
//free all cards' resource
void free_card_list(card_t *cards);
//check all cards' on the ground if the cards' effect have to be execute
void check_effect(player_t *player);
//discard all cards' in the linking list to the discard area
void discard_all_card(card_t *cards);
//get trading price in every trade phase
void get_Price();
//sell a card and return how many card you have to draw
int sell(card_t *card)
//delete a card from a card linking list(No free the resource)
card_t *delete_card_from_list(card_t *cardList, card_t *card);
//for player to choose card(draw choose list on the screen and let player to click) return a card_t which player choose
card_t *choose_card(card_t *cards);
//check if the card are same(same type)
bool same_card(card_t *card1, card_t *card2);
//check if the building is on the ground
bool has_<buildingName>(card_t *ground);
```

Career.c

Define all Careers' action and call player's chooseCard function to choose reasonable card in different phase defined by phase_t.

```
//act the careers' action
void <careerName>_action(player_t *player);
//act the careers' previlege then call <careerName>_action
void <careerName>_previ(player_t *player);
//load careers' texture. Success return true, else return false.
int load_career_texture();
//free career's texture.
void free_career_texture();
//destroy careers' linking list(In each round end)
void destroy_career_list(card_t *cards);
//init careers' linking list(In each round start)
card_t *card_t *init_career_list();
//get the present career, return a career_t
career_t get_career(card_t *career);
typedef enum(In SanJunadef.h) {
    CR_BUILDER = 0,
    CR_COUNCILLOR = 1,
    CR_PRODUCER = 2,
    CR_PROSEPCTOR = 3,
    CR_TRADER = 4,
} career_t;
```

Deck.c

Encapsulate discard and draw function which makes us just call one function to get one card or discard one card

```
//shuffle the deck
void shuffle();
//draw a card from the top of the deck and return a card_t
card_t draw();
//discard a card to discard area
void disard_card(card_t card);
```

Draw.c

Draw all candidate card for player to choose and draw ground or other necessary information for player.

```
//draw all information on the ground(bot's ground and our hands)
void draw_ground();
//draw present career text
void draw_presentCareer();
//draw what bot choose and wait player to confirm
void draw_bot_choose_msg_and_wait_for_confirm(int playerID, void *param);
//draw all cards you want player to choose with a card_t linking list as parameter
void draw_choose_card(card_t *cards);
//check which card player click and return the clicked card
card_t *get_choose(card_t *cards, SDL_Point *point);
//draw the price when trading phase
void draw_trading_status();
//draw OK button
void draw_confirm_button();
//check if the mouse click OK button
bool in_ConfirmButton(SDL_Point point);
//draw the massage you pass and give a "Yes", "No" button to wait player to choose
```

```
18 bool draw_confirm_msg_and_choose_answer(char
   *msg);
19
```

Bot.c

for the bots' function

```
1 //the level of bot's choose career function
2 career_t bot_level_<num>_choose_career(player_t
  *player);
3 //the level of bot's choose card function
4 card_t * bot_level_<num>_choose_card(player_t
  *self, card_t **cards)
5 //pass the level function to the chooseCareer and
  chooseCard function defined in player_t depends on
  mode(difficulty)
6 void get_Bot_Func(player_t *player, int mode);
```

GameMain.c

For the whole logic process.

```
1  //print the player's information by player ID(for
   debug)
2  void printPlayerInfo(int j);
3  //handle all logic process
4  int SANJUAN_Main();
5  //handle menu
6  int SANJUAN_Menu();
7  //the loop running for the whole game
8  void SANJUAN_Loop();
9  //init all SDL2 library
10 int Init_Graphic_Lib();
11 //init all resource for the menu(button,
   background)
```

```
int load_start_Resource();
//init all resource for the game(call load_career_texture() and load_card_texture())
int load_in_game_resource();
```

Language.c

Defined all strings in the game.

```
//set the language you want to present
void set_language(char *lang);
//read the language file and copy to the massage string
void load_language();
```

```
//init a player_t object by his playerID(playerNum) and init hand card he has
player_t *init_player(int playerNum, int initCard);
//free all resource for a player_t object
void clear_player(player_t *player);
//for a real player to choose card for player_t
card_t *player_choose_Card(player_t *self, card_t **cards);
//for a real player to choose career for player_t
career_t player_choose_career(player_t *self);
//check if a specific card is on the ground
bool check_card_on_ground(card_t *ground, card_t *card);
```

Sanjuandata.h

for the definition of card_t and player_t and phase_t and price_t

Sanjuandef.h

for the definition of gameMode_t and career_t and building_t

Setting.c

```
//enum for which option you in
typedef enum {
    ST_,
    ST_SOUND,
    ST_LANGUAGE,
    ST_GRAPHIC,
    ST_BACK,
} setting_t;
//handle the whole setting(both display and logic)
void setting();
```

Sound.c

hold the Mix_Music object of SDL2 for play the music

Window.c

Hold all information for window

```
//update the resolution and redraw the screen
void update_window_res();
//destroy all object create for window
void quit_Game();
```