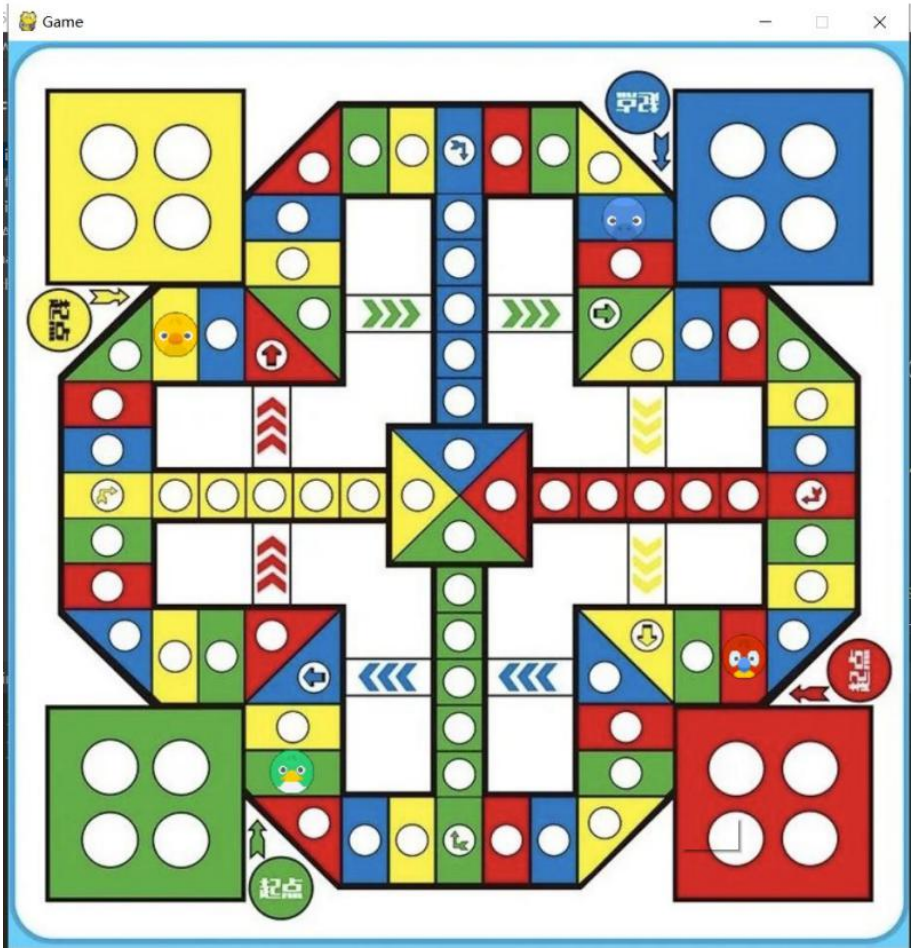
Online Aeroplane Chess

Week 12 Report



Group member:

Li Zi

Gou Xichun

Yu Yingying

Su Zix

Contents

General Introduction

Game introduction

This project aims to develop a traditional flying chess game with the basic rules as follows:

1) Players need to roll dice to get 6 to take off;

2) After take-off, players advance according to the number of dice they have thrown this turn;

3) Each time a 6 is gained, a piece can take off or a piece that has already taken off can continue to advance;

4) Chess pieces with the same color as the drop point can jump to the next drop point with the same color during driving;

5) During driving, chess pieces with the same color as the drop point with the flight path can leap to the corresponding drop point through the flight path;

6) If there are chess pieces of other colors at the landing point during driving, it will be reset back to the starting point;

7) In the process of driving, if the drop point is the chess piece of the same color, it will be stacked. Other pieces in the stack cannot pass;

8) The first player to place four pieces in the end zone wins

Assignment Requirements

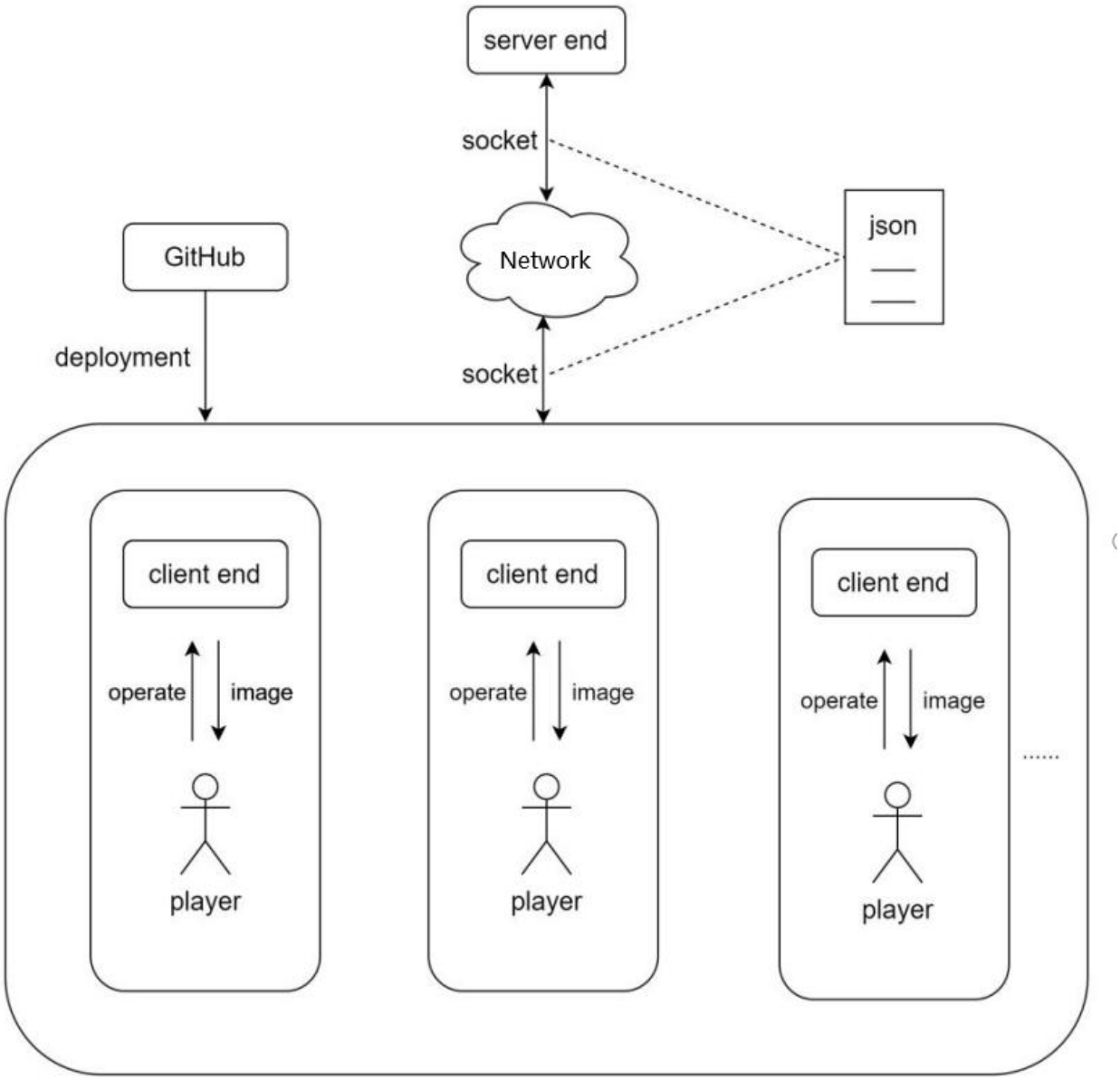
- Minimum number of players 3
- Played over a network
- Cross platform Windows, OS/X and Linux
- Getting a fully playable and stable version of the game implemented is the main priority
- Key that the game implement AI to replace a human player, or play/complete a human player's turn for them if they take too long.

Game Constraints

- Up to 4 players (There are only 4 colored chess pieces in the map)
- 76 board positions
- 16 chess pieces in total, 4 for each player (4 kinds of planes, 4*4=16): yellow, blue, green, red
- 4 different colored airfields
- 4 special board positions: Flying Chess (the player can go directly to a fixed position according to the map).
- By rolling the dice, players can move 1-6 spaces once.

Architecture/Design

Our high level design is very different from the fourth week and same as eighth week. After the update, we abandoned some useless ideas (Due to the game is developed by pygame, we have to give up the webpage development and Internet cloud server connection) and formulated the high level design in line with our development process, as shown below:



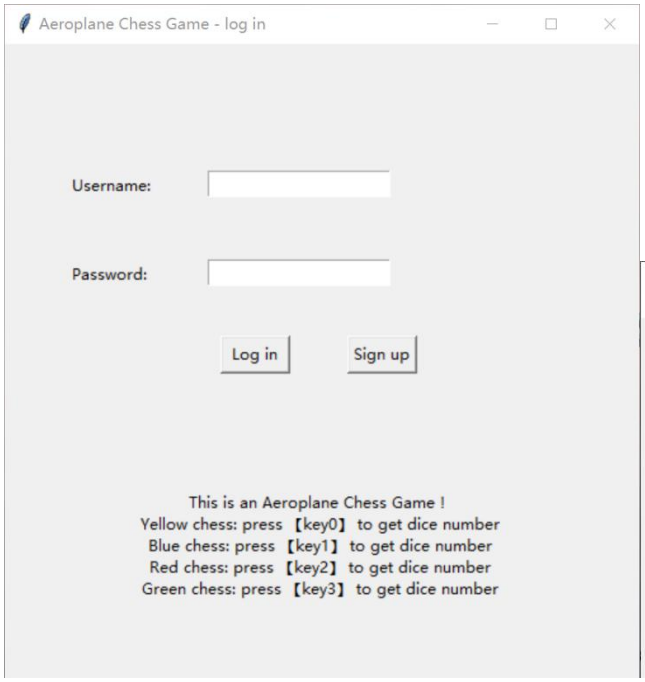
In the interest of simplicity, we did away with the use of cloud platforms as servers (typically accessing them in China might require a VPN). Instead, we use another computer as a server to provide the overall service.

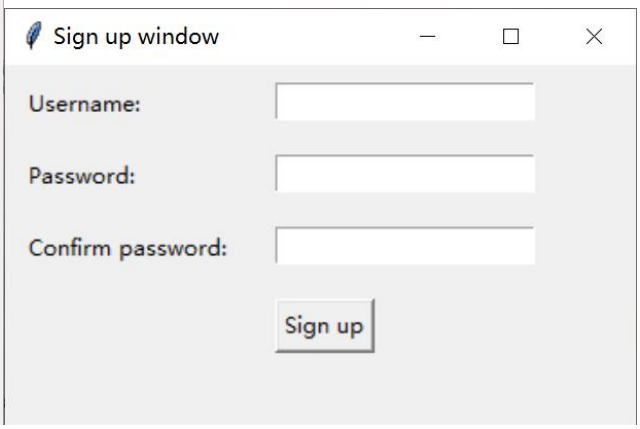
In this design, we plan to embed sockets into the game code so that there are both client and server programs in one project. For this reason, if a server is run on a computer device, the device is a server. Other players can download the project directly from Github, configure the environment locally, install Python, Pygame and Socket library, and then run the client program to connect and control the game. The game screen and command control are provided and monitored by each user's local IDE, and broadcast through JSON files to keep the game screen synchronized.

For the user interface, it will be set in each client. When the player runs the client, the native program creates a user interface that lets them enter a username and password. If you are a new user, you can click the "Sign up" button to register. The registered user name must be different from the name of the

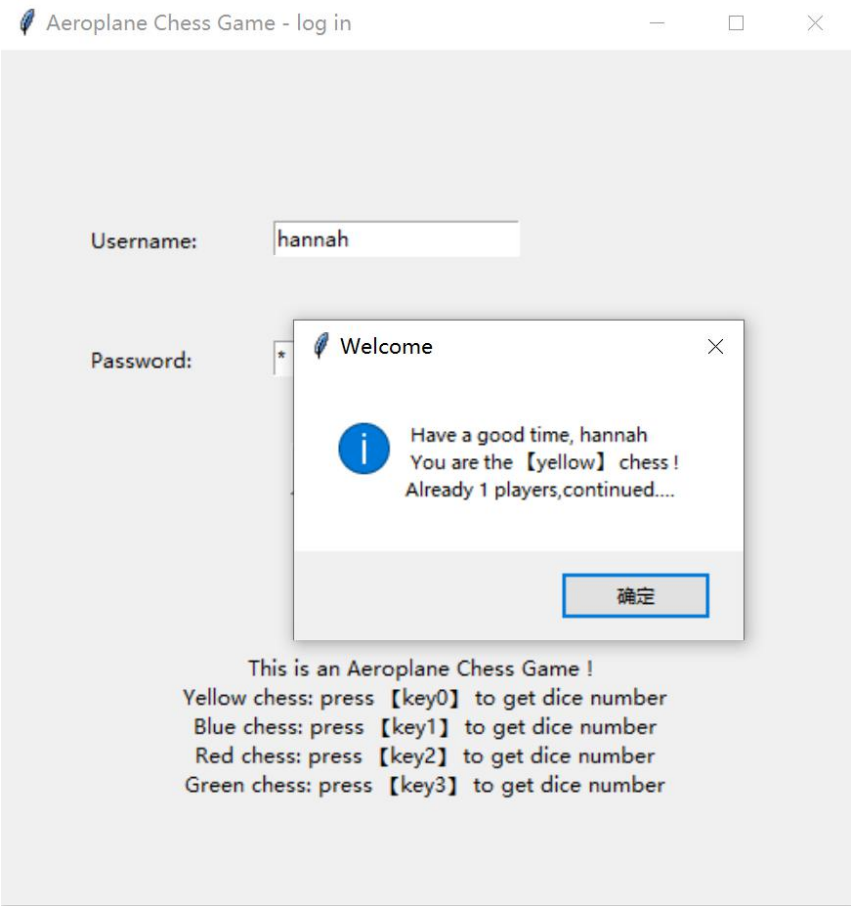
existing user. After successful registration, you need to return to the login interface to log in. In addition, at the bottom of the login screen, the basic rules of the game are shown to the player. After the user logs in successfully, a pop-up box will be triggered, which contains a lot of important information, including: the user name of the player, the color of the pieces used by the player, and the number of current players (including the player).

Login and registration interface:





Pop-up triggered after successful login:



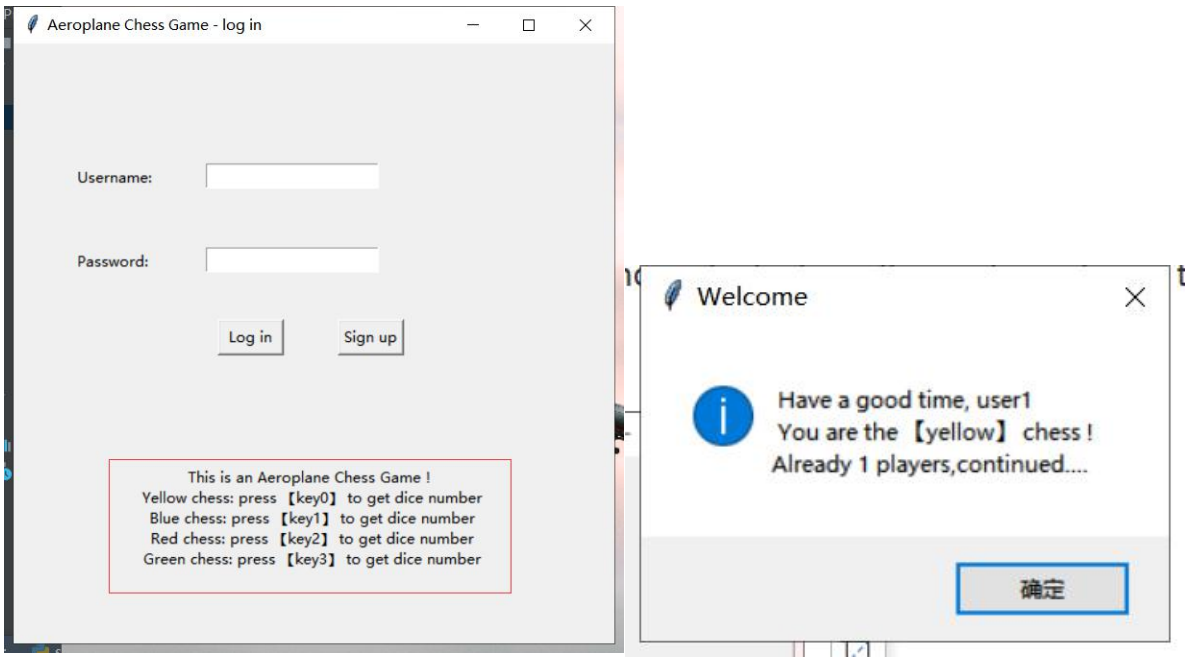
The process of communication can be described as following stage:

- When the server is running, it will continue to listening the JSON data from client.
- When client running, it will send a message to server. If connection succeeds, the server will send a message back to the client.
- When all clients are ready, the first player will announce game start. The first client will send a message to server and server will continue to sending the message to other clients make them sync with the first client.
- Than when the game begins, when a player do his turn, the client will send server a massage about what the player has done in his turn. And the server will pass the message to other client to make them sync, until the game finish.

QA

End user testing (UI)：

When the game was complete but not yet online, we interviewed 20 players. Asked a few questions about the game and the post game experience. They said it was difficult for them to play for the first time and needed to be told how to do it, so we gave them a hint under the login screen. They thought the game was easy to understand and the overall feel was good. And for some user said they do not know how to control the game and do not know which chess they got, we design the function in user interface to achieve this suggestion:



PyTest:

In testFunctions.py, Unittest mainly tests the basic methods of the cell and chess classes, as well as the functions used throughout the program, which are stored in test1.py. Throughout the game, the main logic is in func.py and index.py, and due to some limitations, we added the logical methods we used to the test1.py file. The test1.py file is used to refer to all the functions in the test. We can see that the test1.py file does not have very high coverage because it contains cell and chess initializations and representations of maps.

Module ↑	statements	missing	excluded	coverage
cell.py	33	0	0	100%
chess.py	10	0	0	100%
test1.py	297	111	0	63%
testFunctions.py	184	2	0	99%
Total	**524**	**113**	**0**	**78%**

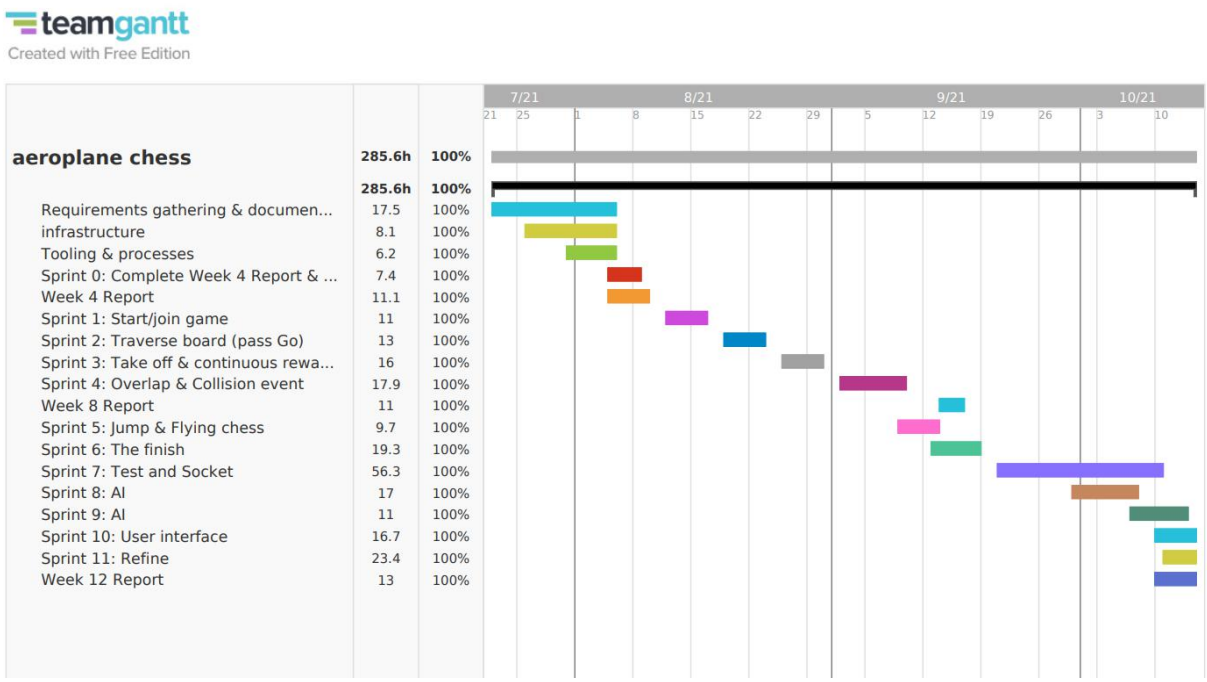
How to run：

-coverage run testFunctions
The results of running the tests will be displayed.
-coverage html
The code coverage will be stored in the index.html file in the htmlcov folder.

Project Statistics

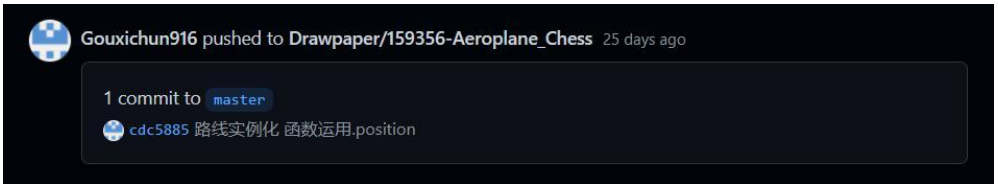
Issue tracking

Overview:



I will list some of the sprint issues and give some explanations :

Sprint 1:(issue)

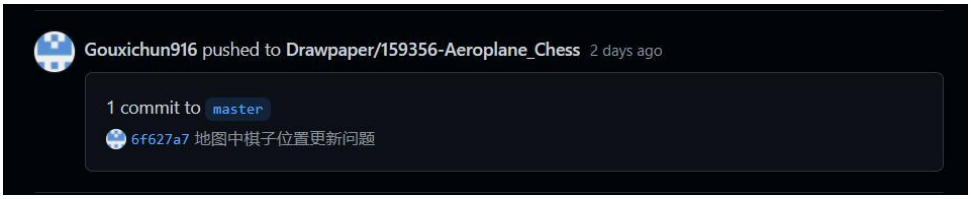


When we examine Sprint 1, we see that each cell in the diagram is still not instantiated. In this step, the group leader reported this problem to another group member through WeChat and asked them to make modifications. The above is the version that the team member revised and submitted, and the team leader reviewed and found correct. The issue is closed.

Sprint 3:(bug)

During the Sprint 3 review, we found that one of the functions did not perform as expected. So the team leader rejected the application of close Sprint3 and reported the problem to another team member for modification. When the changes are correct, the team leader announces that the Sprint is closed.

Sprint 4, Sprint 1:(issue)

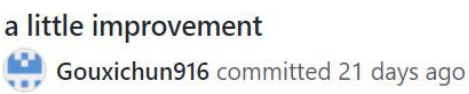


After Sprint 4 was completed, the team leader noticed that some of the positions of the pieces were in conflict with the positions of the instances in Sprint 1. The team leader announces the reopening of Sprint 1 and Sprint 4 and assigns team members to fix the conflicts. After modification, the team leader checked the results and announced the closure of Sprint 1 and Sprint 4 after confirmation.

Sprint 5:(issue)

Basic is implemented in the logic of the game, we had a complete demo of the game, in the process of demo, focus on the placement is in line with the logic of pieces, found in the "fly" and "checkers" on these two rules, logical sequencing problem, need to prioritise "flies", and then execute "checkers", we made changes to this issue in a timely manner.

Sprint 6:(bug)



cell is not iterable
Gouxichun916 committed 15 days ago

In the process of trial play, when the game lasted more than 10 minutes, the game would display an error, indicating "Cell is not iterable". Through the code position given by the error, we located the error and corrected the error. In order to ensure that the error would not occur, we tried more than 10 times. Each trial should last no less than 10 minutes.

Sprint 7:(issue)

fixed tests
yuyingying-666 committed 24 days ago

getOptions 2.0
yuyingying-666 committed 5 days ago

new tests
yuyingying-666 committed yesterday

Test for us, is a very important part, we invested a lot of time and energy on the test program, through continuous improvement of the test, we make the function and the function of attention are covered by the test, in the process of test, we found the getOptions function of a logic issue, and has carried on the correction, Make the function of the game more perfect. In addition, we are constantly updating the testing section as we improve the functional logic and add features to the game.

Sprint 10:(issue)

turn window and push error
Gouxichun916 committed 1 hour ago

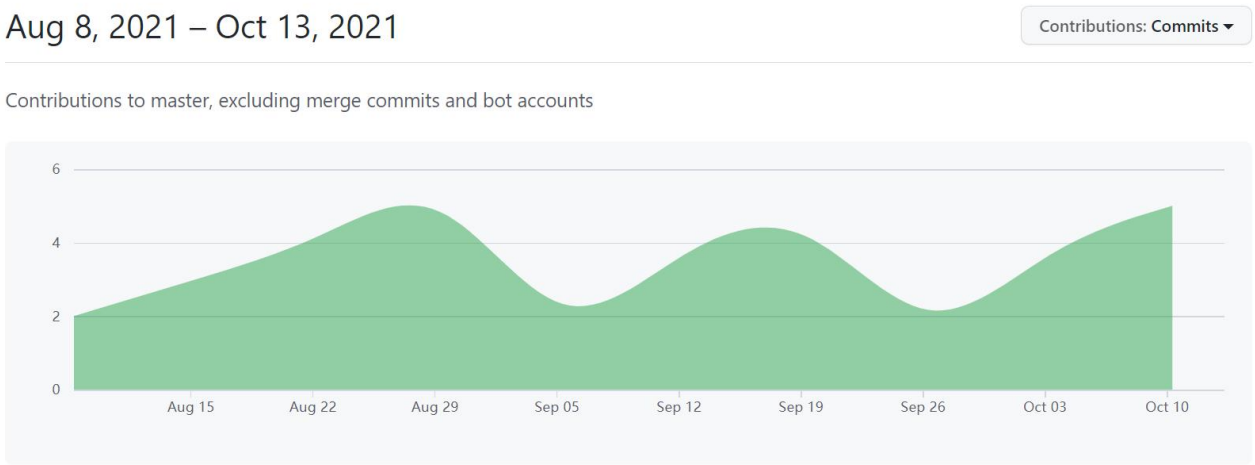
turn window
Gouxichun916 committed 3 hours ago

After implementing the user interface for login and registration, we communicated with our teacher, and when reporting our recent achievements to the teacher, the teacher suggested that we add an interface to remind the player that "it's his turn". We unanimously approved the teacher's proposal. In addition to making the login successful information more specific, we added a pop-up window to remind players, which enhanced players' game experience and made the whole game more complete.

This is our summary of the development and how we are tracking the sprints that have been completed. All the questions and the detailed work done in each section have been recorded by the group leader.

Repository Activity

Commits

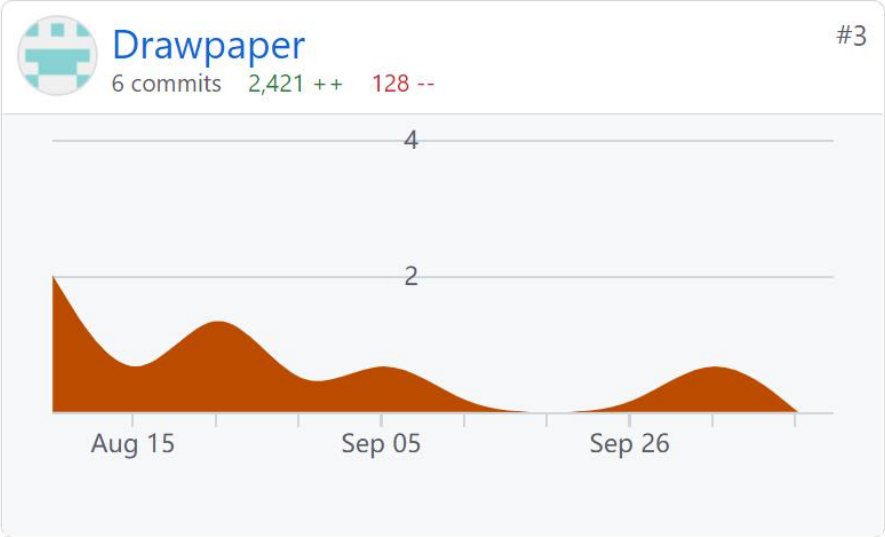


Contributors

Yu Yngying:

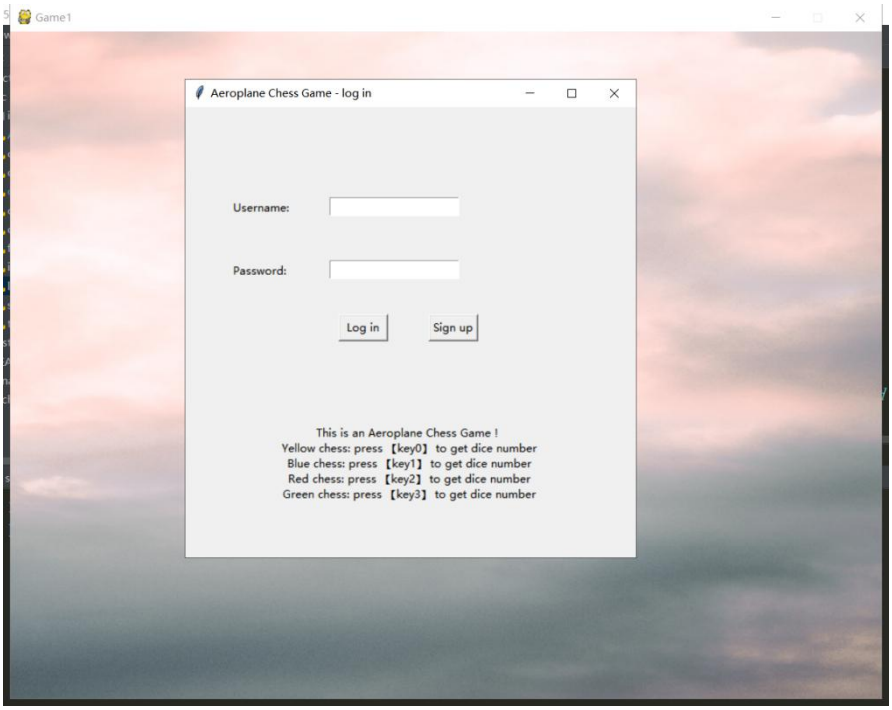
Gou Xichun:

Li Zi (leader):



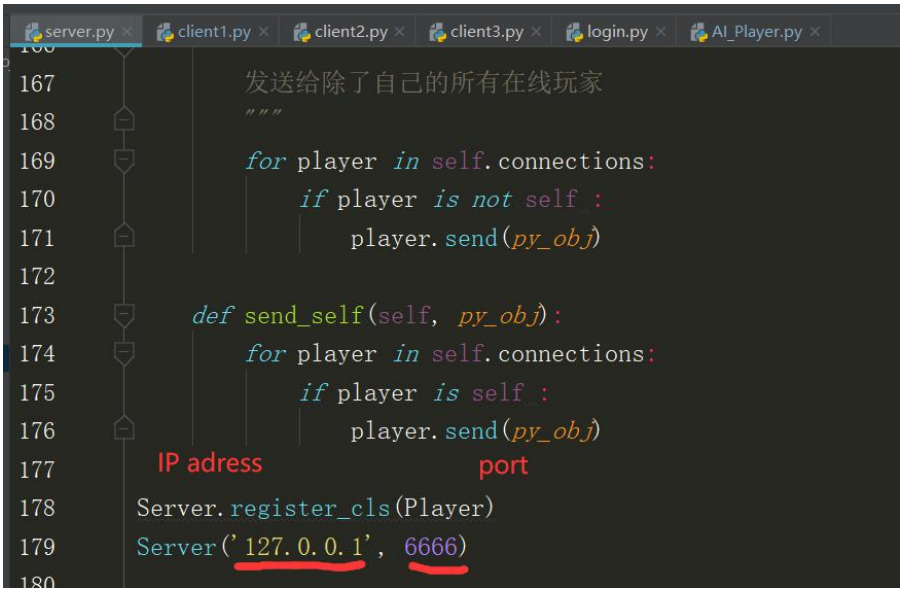
Customization Manual

The customisable game styles are housed in the file data.csv. The file houses the data for the username and password using by the player to login the game.



And for other variables to change the style of the game are almost defined in the code and which cannot be changed at will by users.

For the Port and IP address it can be changed in the code:



```
server.py  client1.py  client2.py  client3.py  login.py  AI_Player.py
                drawStartPoints(parrot_chess[i-1].chess_type,-i)
            else:
                continue

    gameControl(random.randint(1, 6),None)
    # 根据用户的操作切换游戏状态
    s = socket.socket()
    s.connect(('127.0.0.1', 6666))
    player_num = None
    cur_num = None
```

IP adress port

When setting the right IP address and port number, the server can communicate with client in the local area network.

Build Instructions and Installation Manual

Prerequisites

Ensure you have the following installed:

- git
- python >= 3.7
- pip
- pygame
- socket
- pycharm or other run environment

Repository, Environment & Required Packages

git clone git@github.com:Drawpaper/159356-Aeroplane_Chess.git
cd 159356-Aeroplane_Chess/src
python -m pip install -U pygame --user
python -m pip install -U socket --user

Configuration

Before running, The initiator of the game (player 1) needs to configure all players' usernames and passwords to ensure that his friends can log in with known usernames and passwords.
Because in our Architecture/Design, we use a computer as server to run the game, therefore we do not using database to store our username and password.Instead, we create a Configuration file in local to make the hoster change the data and other client can get the data from the communication with server.

That is what we done in this part.

The player should enter the data.csv file to change the default username and password:

```
data.csv
username, password
user1, 111111
user2, 222222
user3, 333333
gxc, 123
gg, 12345
```

The login function sends the protocol to the server and receives the protocol sent back from the server:

```
data.csv  login.py  server.py
tk.Label(window, text = 'Username:').place(x = 50, y = 100)
tk.Label(window, text = 'Password:').place(x = 50, y = 170)
rules_intro="This is an Aeroplane Chess Game !\n Yellow chess: press 【key0
            "to get dice number\n Red chess: press 【key2】 " \
                "to get dice number\n Green chess: press 【key3】 to get di
la= tk.Label(window, text= rules_intro).place(x=100, y=350)

def usr_login():
    usr_name = var_usr_name.get()
    usr_pwd = var_usr_pwd.get()

    pro={'username':usr_name,'password':usr_pwd,'protocol': 'login'}
    s.sendall((json.dumps(pro, ensure_ascii=False) + '|#|').encode())

    pro_reply = eval(s.recv(4096).decode('utf8').split('|#|')[0])
    if pro_reply['judge']==1:# 传到server
        if player_num==1:
```

The registration function sends the protocol to the server and receives the protocol sent back from the server:

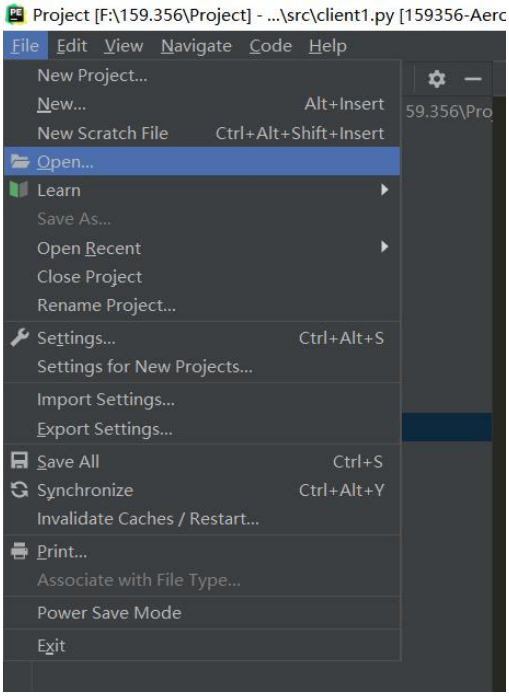
```
data.csv  login.py  server.py
def usr_sign_up():
    def sign_to_Python():

        np = new_pwd.get()
        np_confirm = new_pwd_confirm.get()
        nn = new_name.get()

        pro2={'username':nn,'password':np,'protocol': 'signup'}
        s.sendall((json.dumps(pro2, ensure_ascii=False) + '|#|').encode())
        pro_reply2 = eval(s.recv(4096).decode('utf8').split('|#|')[0])

        if np!=np_confirm:
            tk.messagebox.showerror('Error','the confirm password is differ
        if pro_reply2['judge']==0:
            tk.messagebox.showerror('Error','The user has already signed up
        else:
```

The server receives the protocol for login and registration from the clients:

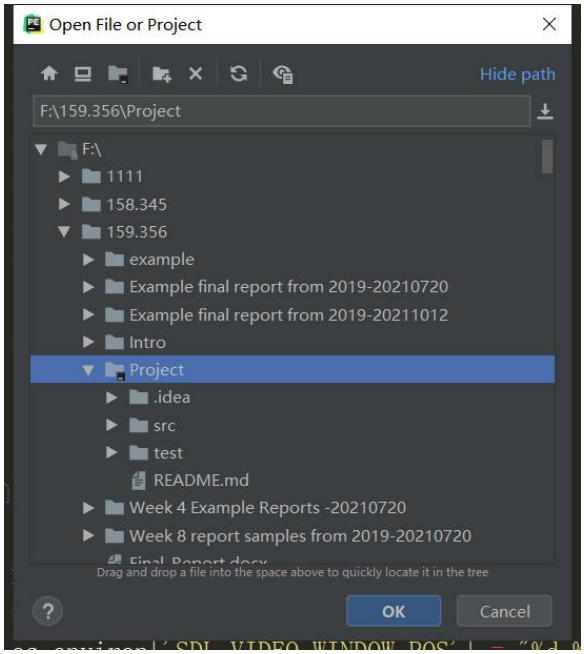
```
data.csv   login.py   server.py

122
123
124     elif bytes['protocol']=='login':# log in
125         filename = 'data.csv'
126         dic_data={}
127         with open(filename) as f:
128             reader = csv.DictReader(f)
129             for row in reader:
130                 username = row['username']
131                 password= row['password']
132                 # print(username)
133                 # print(password)
134                 dic_data[username]=password
135         if bytes['username'] in dic_data and dic_data[bytes['user
136             reply = {
137             'protocol': 'login-reply',
138             'number' : len(self.connections),
139             'judge':1
140             }
141
142             self.send_self(reply)
143     elif bytes['protocol']=='signup': # sign up
144         judge=1
145         filename = 'data.csv'
146         dic_data={}
147         with open(filename) as f:
148             reader = csv.DictReader(f)
149             for row in reader:
150                 username = row['username']
151                 password= row['password']
152                 # print(username)
153                 # print(password)
154                 dic_data[username]=password
155         if bytes['username'] in dic_data:
156             judge=0

Player > deal_data()
```

Running

Using pycharm IDE to open the project:



Then, select the program you download or clone from github:



For the game hoster(who wants to be the server and the first player):

1. Set the port number and your IP address in server.py

```python
server.py
        """
        发送给除了自己的所有在线玩家
        """
        for player in self.connections:
            if player is not self :
                player.send(py_obj)

    def send_self(self, py_obj):
        for player in self.connections:
            if player is self :
                player.send(py_obj)

Server.register_cls(Player)
Server('127.0.0.1', 6666)
```

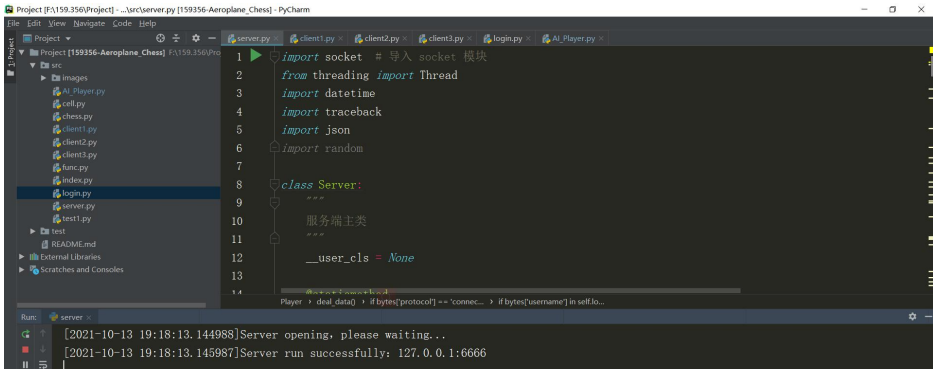
Set your IP address Set your port number

2. Set the port number and the IP address same as the address and port you used to start the sever in client1.py

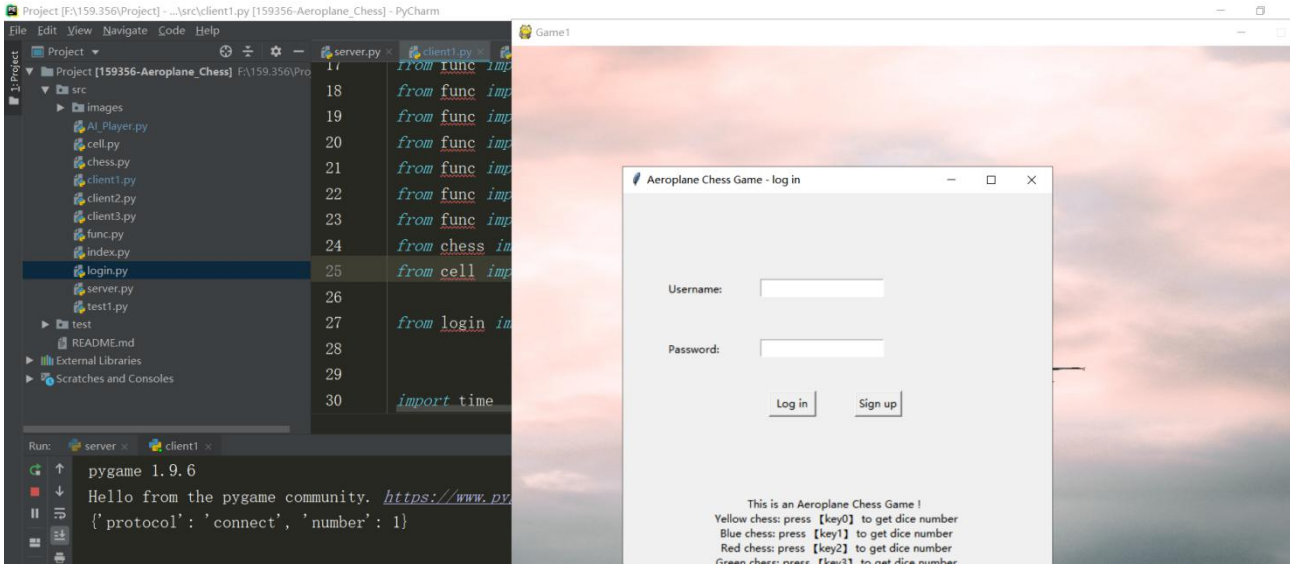
```python
client1.py
                drawStartPoints(parrot_chess[i-1].chess_type,-i)
            else:
                continue

gameControl(random.randint(1, 6),None)
# 根据用户的操作切换游戏状态
s = socket.socket()
s.connect(('127.0.0.1', 6666))
player_num = None
cur_num = None
data = {
            'protocol': 'connect',
            'username': 'user1',
            'password': '111111'
```

Set the IP address and port number same as the server

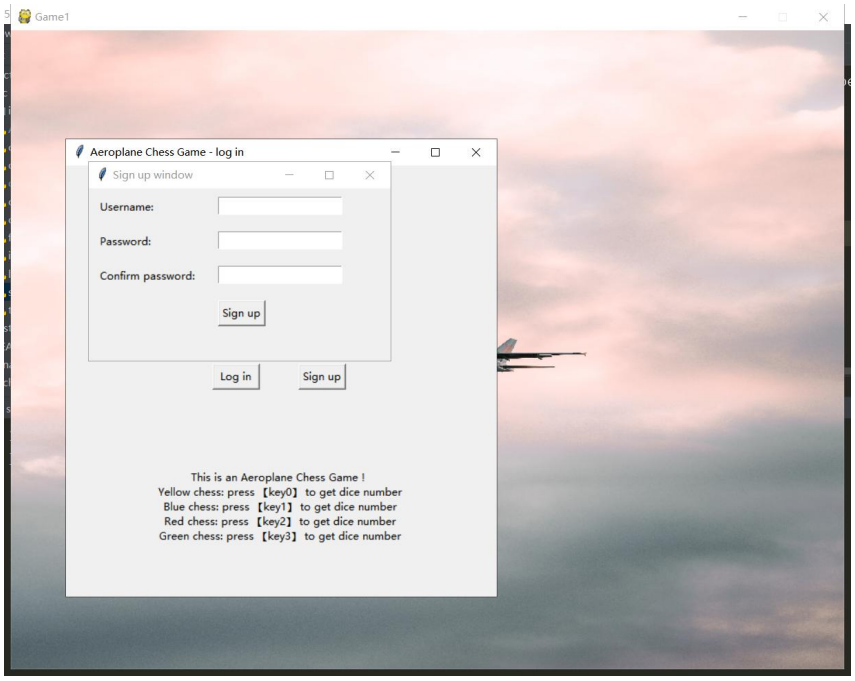
3. Firstly run the server.py



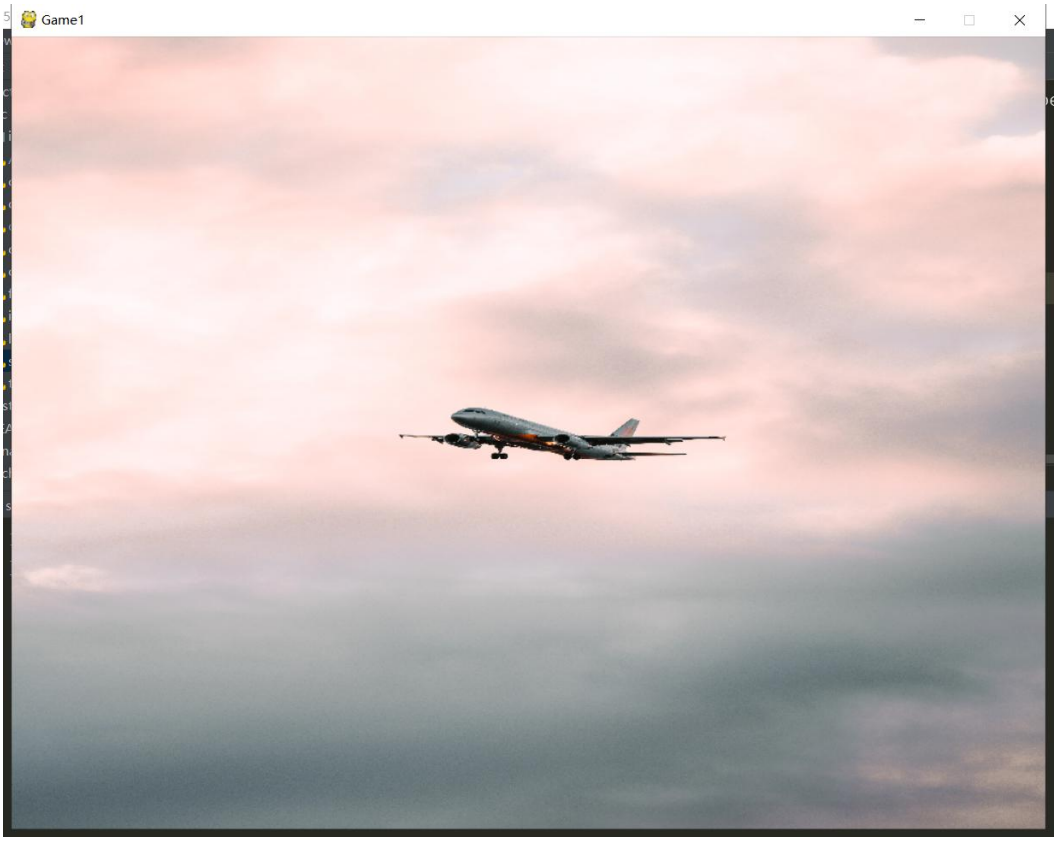
4. Then run the client1.py and login



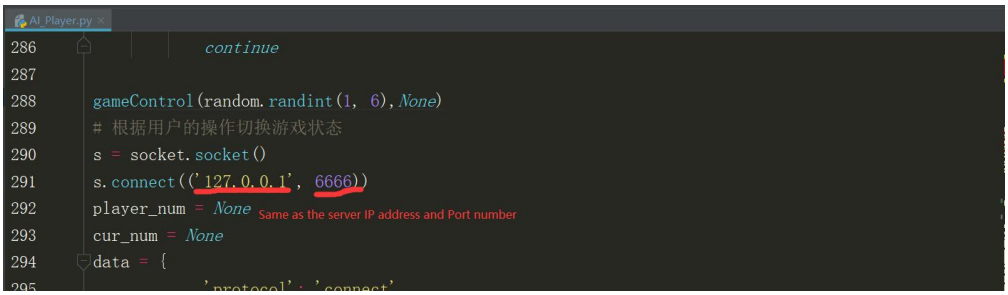
If you do not have username and password please sign up



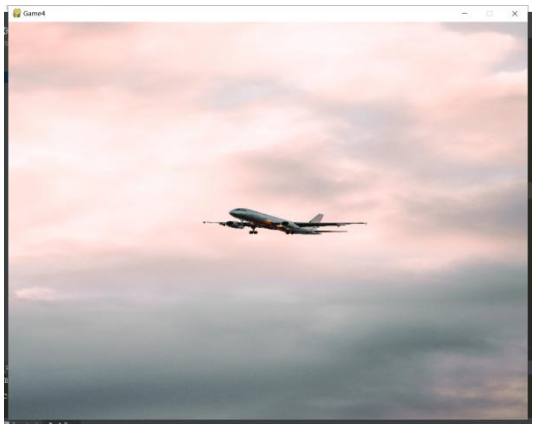
Then you can login, but have to wait in the start page and wait for other player connect until there are four players.There the first player should not click the screen. Only when there are four players, the first player can click the start page to make all player enter the game.



5. If the hoster also want to add an AI player to the game, he should continue to configure the AI_player.py



Next, when there are already three players connected the server, then run AI_player.py .(It must ensure that there are three client connected the server)



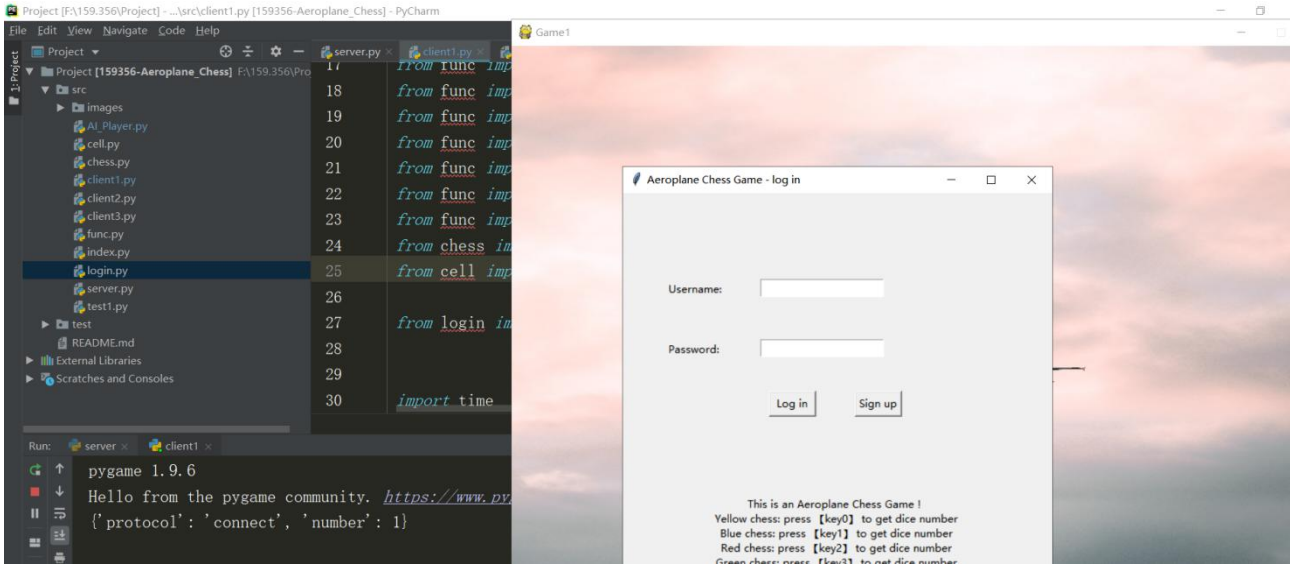
For the other players(who wants to be the server and the first player):

1. Set the sever port number and the IP address in client1.py

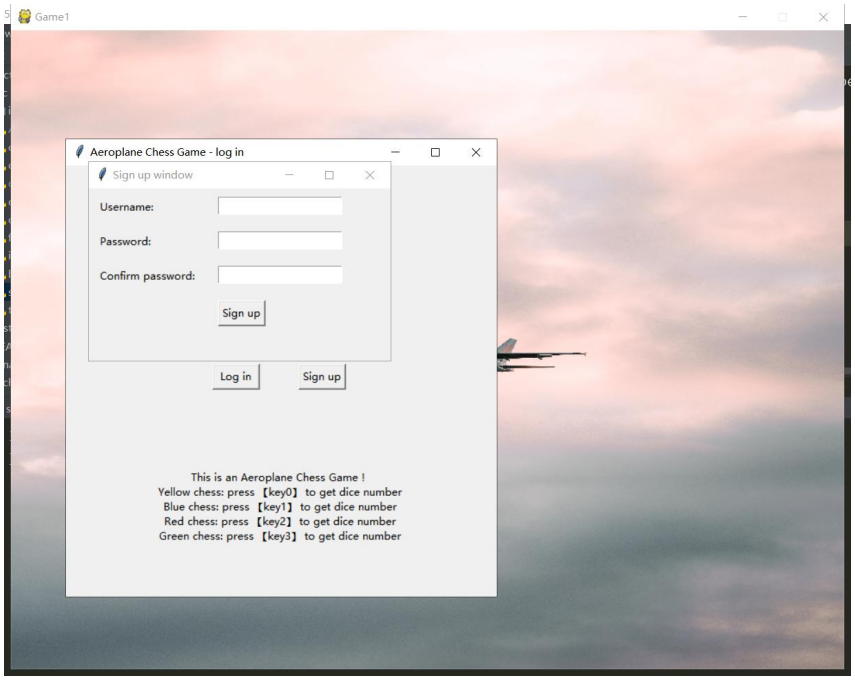
```
client1.py
            drawStartPoints(parrot_chess[i-1].chess_type,-i)
        else:
            continue

gameControl(random.randint(1, 6),None)
# 根据用户的操作切换游戏状态
s = socket.socket()
s.connect(('127.0.0.1', 6666))
player_num = None
cur_num = None
data = {
            'protocol': 'connect',
            'username': 'user1',
            'password': '111111'
```

Set the IP address and port number same as the server

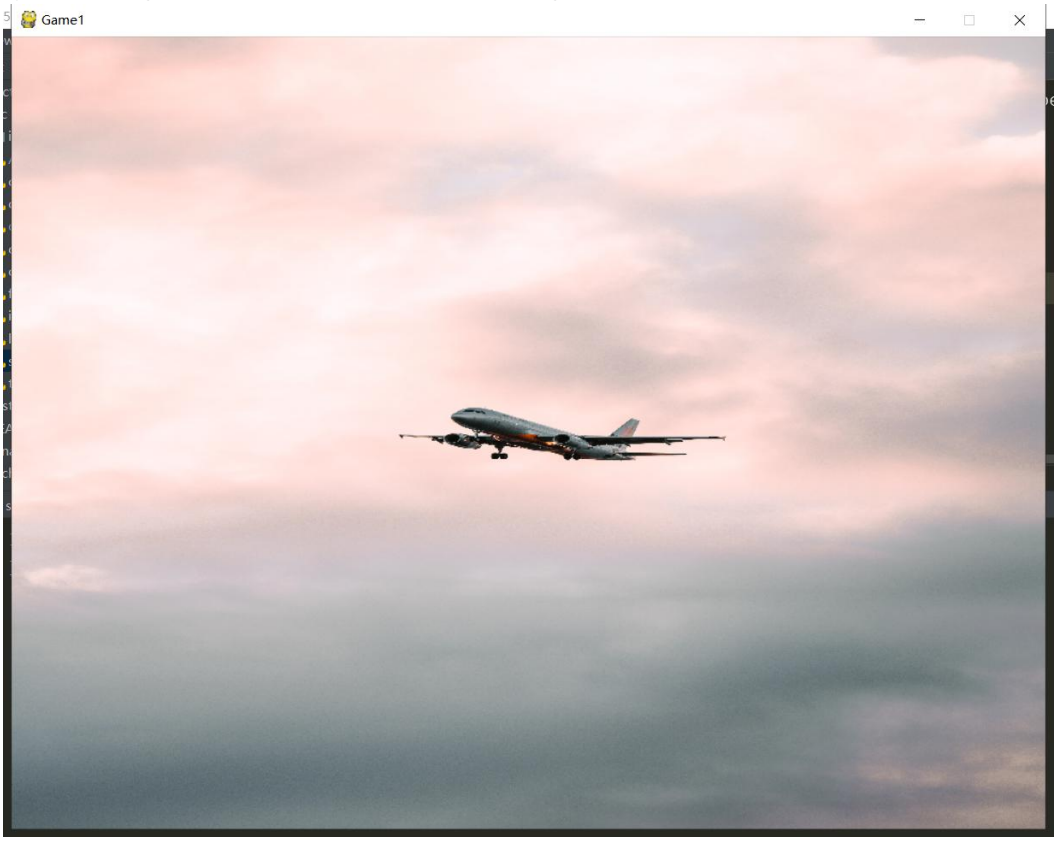
2. Run the run the client1.py and login



If you do not have username and password please sign up



Then you can login, and wait the hoster start the game



When the four players all join the game and login successfully, the first player can click the start page to run the game.

For start a new game, hoster should restart the sever and do the step as above again.And the other player also need to do the step above.

Limitation，know issues and outstanding work

Limitation and improvement

1. The Network

Due to the lack of in-depth understanding of the project requirements in the first four weeks, the project we are currently developing is based on LAN communication. In this project we planned from the beginning to use PyGame and Sockets for design. Unfortunately, later in the development process, we found that games developed on PyGame could not communicate over a wan. We tried a lot of things and

gathered a lot of data, but ultimately we couldn't share pyGame-driven graphics over a wan. Therefore, we turn to the use of socket for the development of LAN connection, in order to meet the requirements of network connection.

However, the connection through LAN is always not as convenient as the wide area network connection and the wide range of transmission. So I figured if we had another chance to redo the project, we would use another game development library that could be connected over a wan instead of PyGame.

2. The User Interface

We don't think the current user interface is aesthetically pleasing because we use MessageBox built into PyGame to implement a basic user interface. Instead of creating a sophisticated user interface like editing a web page, the UI can only implement basic login functions. Since we developed the game logic through PyGame, the content of the game could not be manipulated through the web. So in this section, if we had the opportunity to redevelop, we would use the web to develop the user interface so that the interface would not be crude.

3. The project management method

During development, we used very few project management tools. Lack of experience with large project management tools and an underestimate of how difficult it is to use them. In this development process, in addition to the management tools provided by Git and Github, most of the management activities were managed manually by the leader -- we often communicated with each other through wechat groups, assigned tasks and reviewed projects. Code walk-through and many other operations that can be implemented through automated management tools. This places a considerable burden on the leader. We also tried to use the CI recommended by Mr.Amjed and Mr.Stephen for integration management, but ultimately failed due to configuration failures and inability to access certain services that require VPN. Fortunately, our project was successfully completed in the end. Therefore, in this part, we should confirm in detail the management tools we use and whether we can visit them in China to obtain their services, and get familiar with their management methods in advance.

4. Database development

The only data we need to store in our game is the user name and password. So we don't plan to store it in a database but in a configuration file. But there are a lot of security issues, especially with usernames and passwords. As a result, we now use usernames and passwords that temporarily serve a specific group of players.

5. The running method of our game

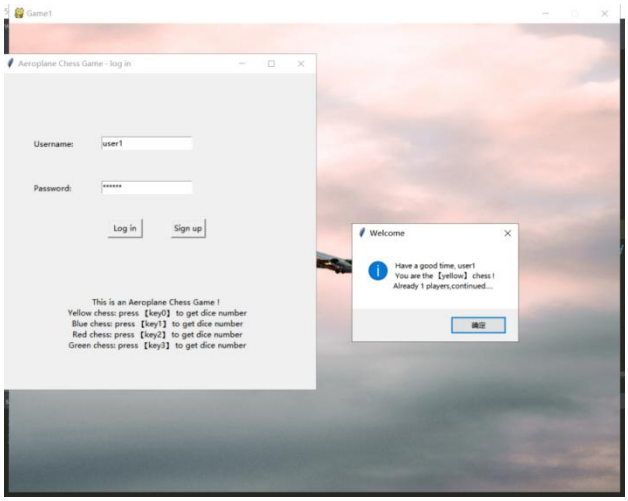
Since our game starts using the easiest way to run py File. So we didn't come up with a better way to package our games. Therefore, our game can only be run through the IDE environment or using the

Command Window Order, instead of being installed on the local machine and running the program through the.exe file like the game downloaded from the network. At present, we still have not thought of a good packing method. Therefore, in order to improve our project, if we have the opportunity, we will learn more about the way to start the project, so that our project operation is more convenient.

known issues

1. The connected player number

In the current code we can display the current number of connected players, but this is only temporary. Once the player clicks OK, the number of connected players will disappear. So The Game Hoster still doesn't know if enough players are playing the game. And if there are not enough 4 player to connect the sever before the first player click the screen, it will happen that the program have no response.We didn't solve the problem in the end because of the time limit. We hope that if development continues, we will solve this problem by showing the current number of players in a pop-up box every time a player connects to the server.



2. Hard to set the port number and IP Address

Just as the run process above, the user should set the port number and IP Address from code. We don’t think it is a convenient way for user to set them. Therefore in the future process, we will add this setting operation to the user interface, to make user conveniently host or connect to a server with specific port number and IP address.

3. Unsolvable problem need to continue to develop

The problem is a new one occurred recently. It will sometime effect the program normally running. We still do not figure out why this problem occurred. It is because when we develop the game logic, we assign the project to each member and every one have a part of logic of this game. It will lead to a confused situation that when a bug was occurred after we connect all the function together, we do not

know which function causing the bug. Although we decide a detailed test to test every function run normally, we do not develop unittest to test whether there are bugs when the function connected together. It was a big mistake that we thought as long as each function is ok to achieve the goal the whole program is also normal. This causes when the deadline coming, we find there was an bug in our program. And in the future development if possible, we will develop more test function to find the reason and deal with it.

Outstanding work

In addition to the above mentioned problems, we also made some breakthroughs in this project through our own efforts. I would like to list our efforts in this project one by one:

1. Project management

Since we had not cooperated in large projects before, how to cooperate with our teammates became our first challenge. In the process of implementing the project, we had many conflicts, such as time allocation conflicts (often due to team members having other urgent matters to deal with, but limited time to submit reports). How to transfer the work of this team member to another team member who is not familiar with the current work is a huge challenge), the problem between module handover (because each team member is responsible for different modules, such as the team responsible for socket development and AI development. How to transition between them to ensure the connection of the two modules can be successful together is a huge challenge), the task allocation problem, and development framework for the establishment of (how to develop framework can guarantee won't appear in the subsequent development big problems, and how the tasks of each part of the framework is fair to assigned to each member) and so on. But eventually, we resolved all of these issues by negotiating, asking experienced developers for help, and so on. Despite some flaws, this led to the final project being successfully developed.

Here are some screenshots of our comments assigning tasks and creating specifications during development:

```
server.py  client1.py  client2.py  client3.py  login.py  AI_Player.py
173            return
174
175        #莹莹:
176        #一个函数以用户所选棋子的类为输入，判断棋子的位置（通过判断棋子的sum或cur_cell）
177            #若为none则为起始点，则执行chess中的起飞函数。该函数使当前棋子移动到该棋子路线上的第一个cell，并更新此cell的信息
178            #若不为none则为普通点，则根据调用当前棋子的地图与sum判断前进后的cell，返回该cell
179    ############(修改，增加)调用该cell中的checkCollide函数判断是否有与当前棋子颜色不同的其他棋子在此cell中
180                    #若有则返回这些棋子（注意：可能会有多个相同颜色的棋子在一个cell中），并将这些棋子的类初始化（置于起飞点）
181                    #若无则返回none
182                #调用该cell中的checkJump与checkFly函数。这两个函数以当前chess为输入判断是否该chess是否可以跳棋或飞棋若能返回最终cell，不能返
183    ################(修改，增加)若可以跳棋或飞棋则仍需再次调用cell中的checkCollide函数判断是否有与当前棋子颜色不同的其他棋子在最终cell中
184                        #若有则返回这些棋子（注意：可能会有多个相同颜色的棋子在一个cell中），并将这些棋子的类初始化（置于起飞点）
185                        #若无则返回none
186                #根据得到的最终位置的cell更新当前chess中的sum与cur_cell信息,并更新最终cell与之前cell中cur_chess的信息
187
188        #希纯:
189    ####（修改，增加）判断当前类别所有棋子的位置，若分别位于最后的几个格子内则将final置1，弹框显示‘xxx色的玩家获胜’【√】
190
191
192    另外需要希纯修改的:
193        #每次需要先让用户看到本轮骰子再显示弹框让用户选择 【√】
194        #将叠子棋子用方形图片代替，图片已经上传在image文件夹里（稍有些难需要考虑一下）【√】
```

2. Network connection

Since none of our team members had any previous experience in web development, how to do web development became one of the most difficult problems for the team to solve. But we finally through the selection of the network tool socket, and spent a long time to understand the communication principle of socket and use specifications. Finally, we overcame this difficulty and developed a communication protocol in line with our project while completing socket connection. Finally, we realized the network connection through socket!

```
server   client1
[2021-10-13 20:59:27.364370]Server opening, please waiting...
[2021-10-13 20:59:27.365370]Server run successfully: 127.0.0.1:6666
[2021-10-13 20:59:33.022984]New connection enter, the current number of connection: 1

Information:  {'protocol': 'connect', 'username': 'user1', 'password': '111111'}
```

```
bytes = eval(bytes.decode('utf8').split('|#|')[0])
if bytes['protocol'] == 'connect':

    if bytes['username'] in self.log and self.log[bytes['username']] == bytes['password']:
        data = {
        'protocol': 'connect',
        'number' : len(self.connections)
        }

        self.send_self(data)
elif bytes['protocol'] == 'ready':
    """
    {
        'protocol':'ready'
    }
    """
    data = {
        'protocol':'ready'
    }
    self.send_without_self(data)
```

```
elif bytes['protocol'] == 'running':
    """
    {
        'protocol':'running',
        'cur_number':'1',
        'move_chess':'1',
        'step':'1'
        'if_final':'0'
    }
    """
    step = bytes['step']
    data = {
        'protocol': 'running',
        'step':step,
        'cur_number':bytes['cur_number'],
        'move_chess':bytes['move_chess'],
        'if_final': bytes['if_final']
        }
    self.send_without_self(data)
```

3. AI development

In the game we ended up designing the AI so that one or more players could be replaced by the AI. This was a huge challenge for our designers, but we overcame it and finished the AI.

The victory condition of flying chess is to take the lead in making all their pieces reach the end, so when the pieces move, the number of the steps is the main standard, choose the pieces that can move the most steps. In AI, all four pieces are simulated to move, and the piece with the most moves is chosen as the final piece to move.

In addition, take off and collision should be considered. The pieces can only make the next move after taking off, and they can only take off when the dice roll 6, which is a very small probability, so I think taking off takes precedence over moving forward 6. Also, collision can force the enemy piece back to the airfield, which is an effective way to prevent victory, so collision takes precedence over flying and jumping.

Commercialisation Plan

Advantages and disadvantages and industry status analysis:

Advantages:

LAN games are much more interesting than the traditional console games that could only be played with a computer. They allow multiple players to play against each other instead of the traditional set pieces. Players are more able to use their talents and have real competition, which makes the game more interesting.

Application advantages should be reflected in the following aspects:
1. Simple and easy to use. If the main players understand the basic rules of flying chess, they can play the game. While the design is different from the console version, it is also very simple to operate. You just need to connect to the server, and after booting up, the user can enter from the server and play against your opponent.
2. A real, immersive battle. Because the game is played through LAN connection, the design not only carries out the AI game between people and computers, but also realizes the game between people, giving players a sense of intimacy.
3. Complete functions. The LAN flying chess game can interrupt the process of the game at any time, start a new game.

Disadvantage:

The need for further quality design, the same repetitive gameplay and unpolished game content leads to a high percentage of user churn, which contributes to the overall trend in the industry towards quality games. Considering that the game itself also bears the property of goods, but also need to recover development, operation costs, etc., so the quality of the game can maximize the polish of the quality of the game, recover the cost of the game.

In addition, in China, the current game products are mainly PC+ mobile games, accounting for more than 90% of the total game market. Because the players previously for host class product acceptance is low (independent of equipment purchase and games), but with the development of recent years and the baptism, players for its acceptance begins to increase gradually, given the host class products can provide a more complete, high quality game experience, and users to pay for the acceptance of the genuine experience gradually improve, The market for host products can also be expanded.

Commercialization direction and Opportunities

We can focus on promoting the game to users who are too busy to play large-scale online games, as well as flying chess enthusiasts：

1. Casual games after work: These apps are small games that people play to relax themselves after stressful work. With the flying chess lovers through the LAN to fight, not only relax the feeling of a day of tension, but also find fun in the battle, is a good choice to kill time.

2. Small Flying chess match game: This kind of application is to launch a small flying chess match in a small area. Through the game can be a multi-player showdown, the final evaluation of a champion. So that the flying chess enthusiasts feel the fun of flying chess. And the game need not consider too much network problems, easy to carry out.

Appendix : User guide

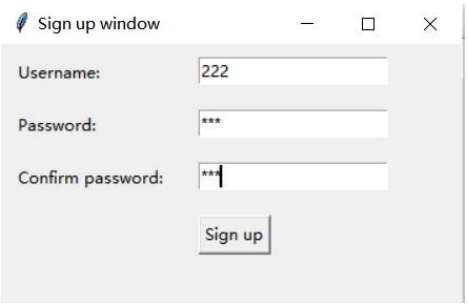
Start a game:

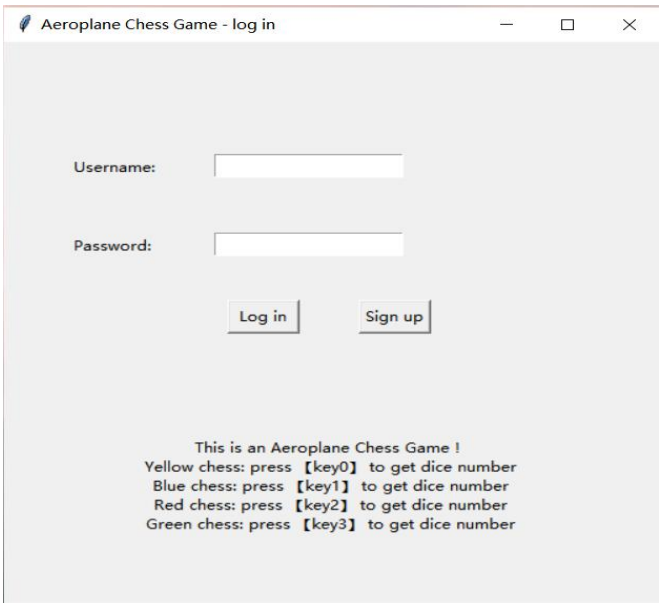
Firstly, run the server.py file, connect to the server. And then run the client1.py, client2.py, client3.py, and client4.py in sequence. User information is exchanged through the server. When four users connect to the server, the state of the server changes from ‘ready’ to ‘running’, and the user can play the game. Noting:only the first player who connect to the server can click the start page to begin a game.

```
[2021-10-13 15:41:59.975507]Server opening, please waiting...
[2021-10-13 15:41:59.978652]Server run successfully: 127.0.0.1:6666
[2021-10-13 15:42:09.143208]New connection enter, the current number of connection: 1

Information:  {'protocol': 'connect', 'username': 'user1', 'password': '111111'}
[2021-10-13 15:42:20.946248]New connection enter, the current number of connection: 2

Information:  {'protocol': 'connect', 'username': 'user1', 'password': '111111'}
[2021-10-13 15:42:27.987272]New connection enter, the current number of connection: 3

Information:  {'protocol': 'connect', 'username': 'user1', 'password': '111111'}
[2021-10-13 15:42:35.745728]New connection enter, the current number of connection: 4

Information:  {'protocol': 'connect', 'username': 'user1', 'password': '111111'}

Information:  {'protocol': 'ready'}

Information:  {'protocol': 'running', 'cur_number': 0, 'move_chess': 0, 'step': 1, 'if_final': 0}

Information:  {'protocol': 'running', 'cur_number': 1, 'move_chess': 0, 'step': 1, 'if_final': 0}

Information:  {'protocol': 'running', 'cur_number': 2, 'move_chess': 1, 'step': 6, 'if_final': 0}

Information:  {'protocol': 'running', 'cur_number': 3, 'move_chess': 0, 'step': 2, 'if_final': 0}

Information:  {'protocol': 'running', 'cur_number': 0, 'move_chess': 1, 'step': 6, 'if_final': 0}
```

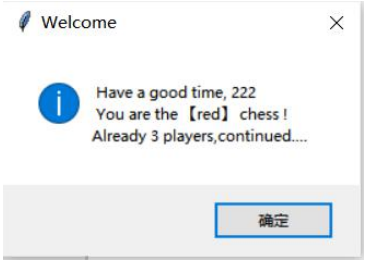
Sign up and Log in：

After entering the game, the first step is to log in. If you don't have an account, sign up.





The popup window will tell you which piece you represent and how to get the dice count.



Video link：https://www.bilibili.com/video/BV19P4y187US?share_source=copy_web

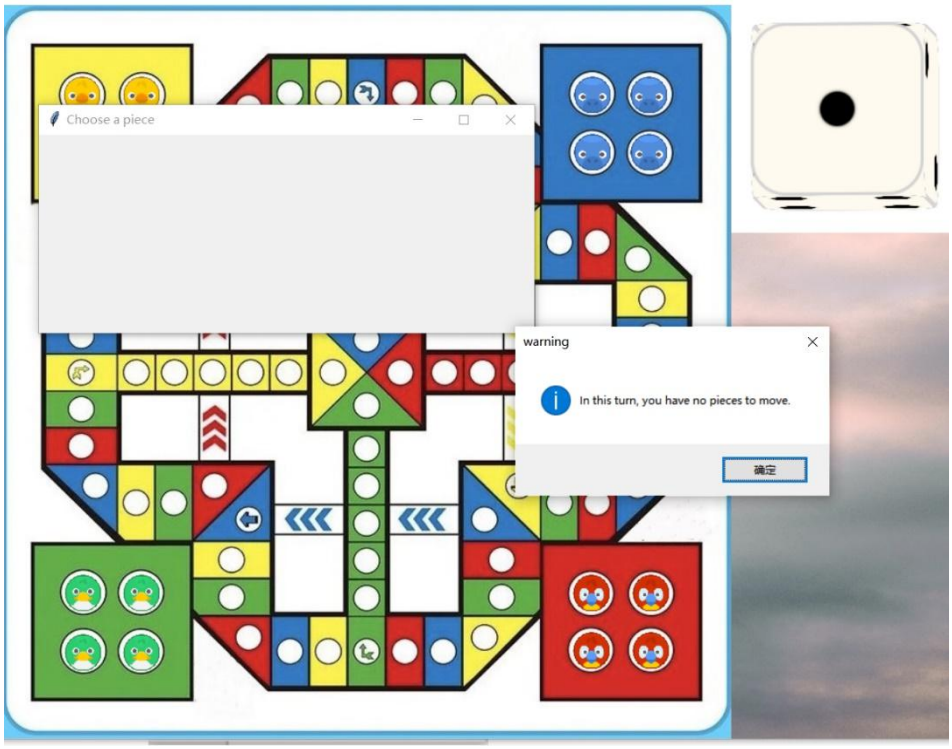
Roll the dice：

When rolling dice, each piece corresponds to a keyboard key. For example, the yellow piece presses 0 and the blue piece presses 1. The player presses the keyboard in his window and the dice change randomly. The user moves based on the roll of the dice.

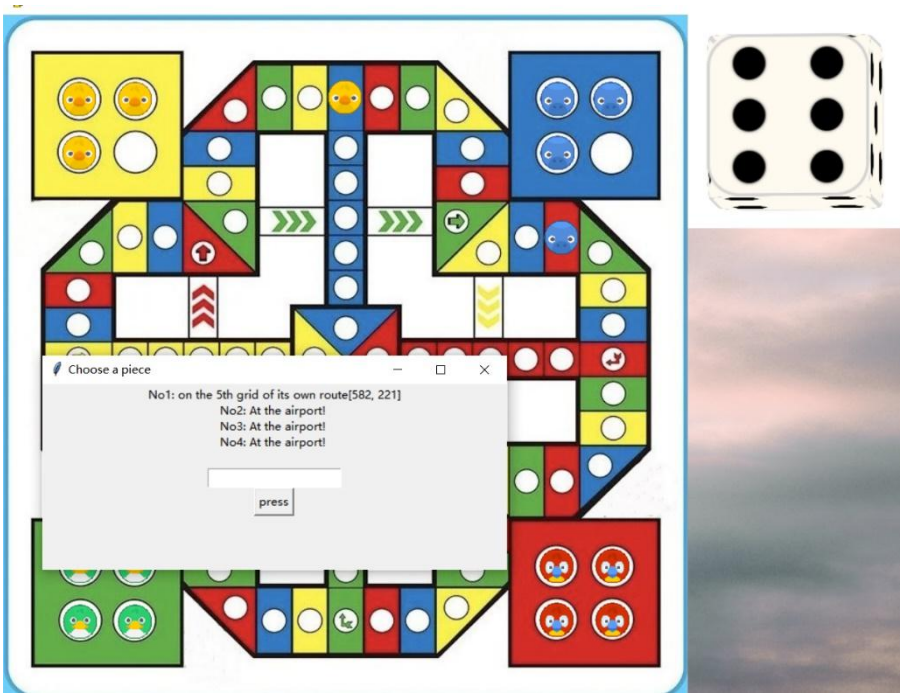
Only when the dice roll to 6, can take off. Players can select a piece in the airport in the popup window to take off.

Video link：https://www.bilibili.com/video/BV1db4y1y7TB?share_source=copy_web

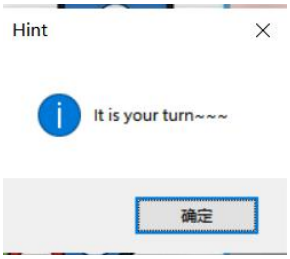
Otherwise, no pieces can be moved and the round is empty.



Only the pieces after takeoff can be moved in the map, popup window will remind you of the pieces that can be moved at present.

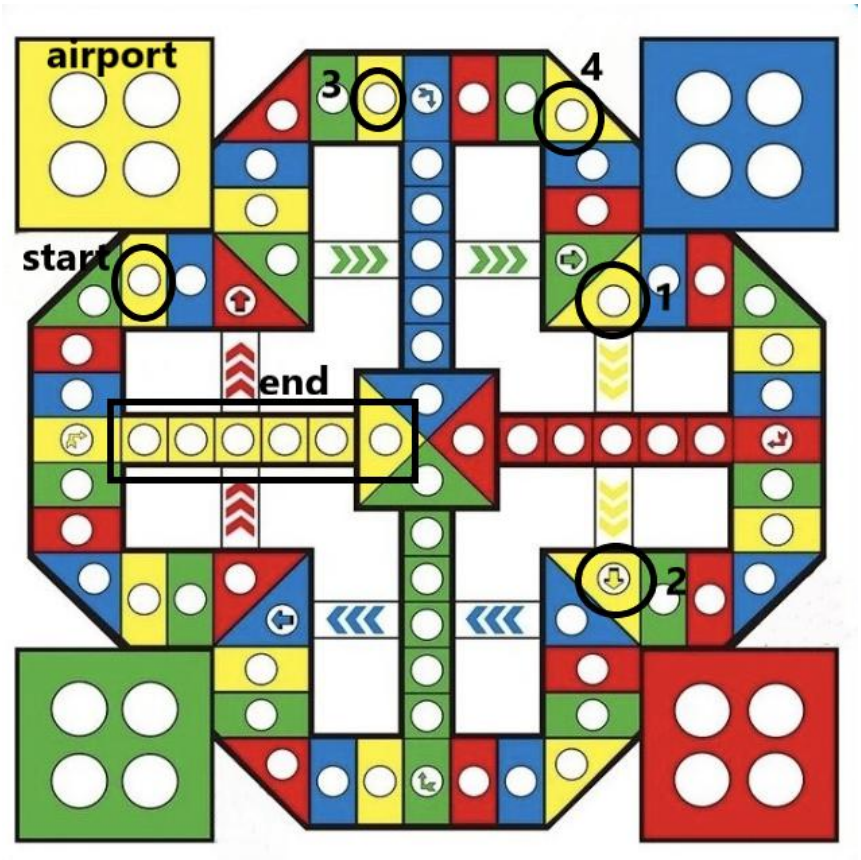


Tip: when one player rolls the dice and moves a piece, the next player will be reminded by a pop-up window.



Special points:

In the case of yellow pieces, a piece needs to move at least 50 squares clockwise from the starting point to the end point. There are 13 jump points and 1 fly point on the map, only the chess piece color is the same as the checkerboard grid color, can jump (3-->4) and fly (1-->2). There are six end points.



Jump & Fly & Collide:

When the player moves to a position that matches his color, the jump point, the player moves forward four more steps.
Video link：https://www.bilibili.com/video/BV1cU4y1A74v?share_source=copy_web

When the player moves to a fly point that matches his color, the player can fly directly to the opposite, moving forward 12 more steps.
Video link：https://www.bilibili.com/video/BV1r34y1S7Eg?share_source=copy_web

When there are two pieces of different colors in the same position, the two pieces collide, the latter one takes the position, and the original piece is forced to return to the airfield to wait for takeoff.
Video link：https://www.bilibili.com/video/BV1xu411Z7d5?share_source=copy_web

Shape of chess pieces:

Under normal circumstances, the pieces are round. The pieces are yellow, blue, red, and green, representing four players. When two pieces of the same color reach the same position, the piece becomes square.

Video link：https://www.bilibili.com/video/BV1dA411F7gf?share_source=copy_web

AI:

The fourth player in the game is set to AI, meaning that there is no need to log in and press the keyboard. When it is the fourth player's turn, the dice roll is automatically triggered, and the piece moves according to the number of dice and the position of other pieces.

Victory conditions:

When all the pieces reach the end, the player wins and the game ends.