掼蛋在线平台使用说明书

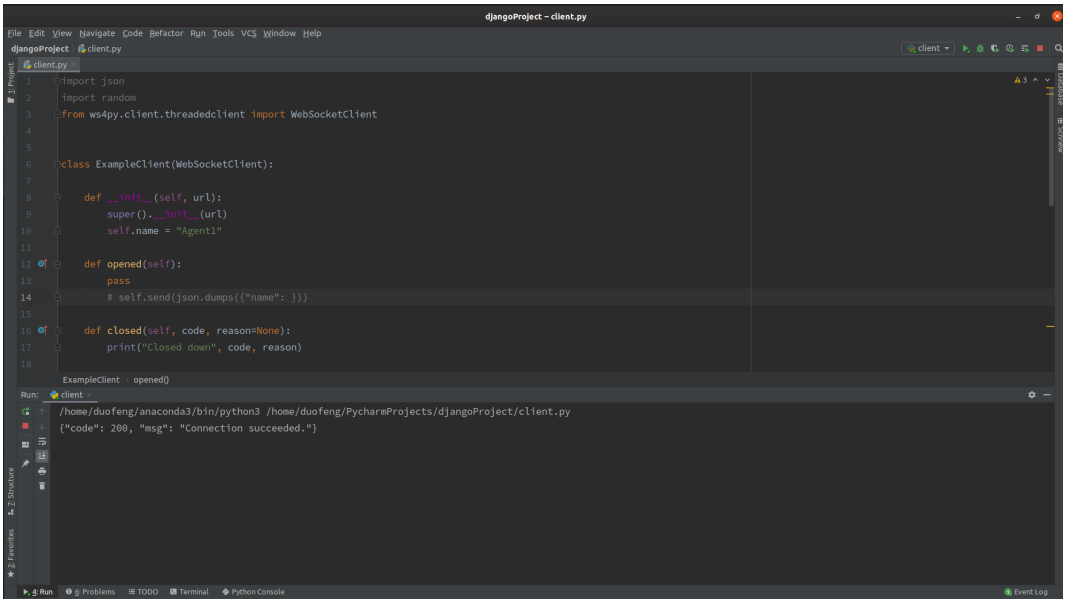
版本说明

1. **在线平台目前处于封闭测试阶段，将于比赛开赛日正式开放**
2. **关于离线版本的实际截图以及实际连接方法请以实际上线时的平台为准**

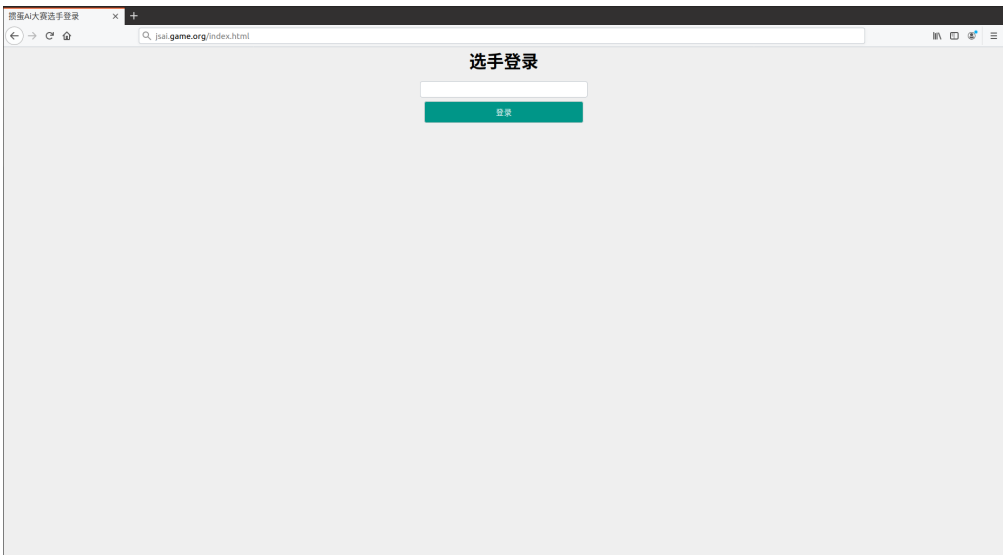
平台运行要求

1. 在线平台为一个开放服务器供大家进行连接进行比赛，要求所有选手的PC配置必须能够连接互联网

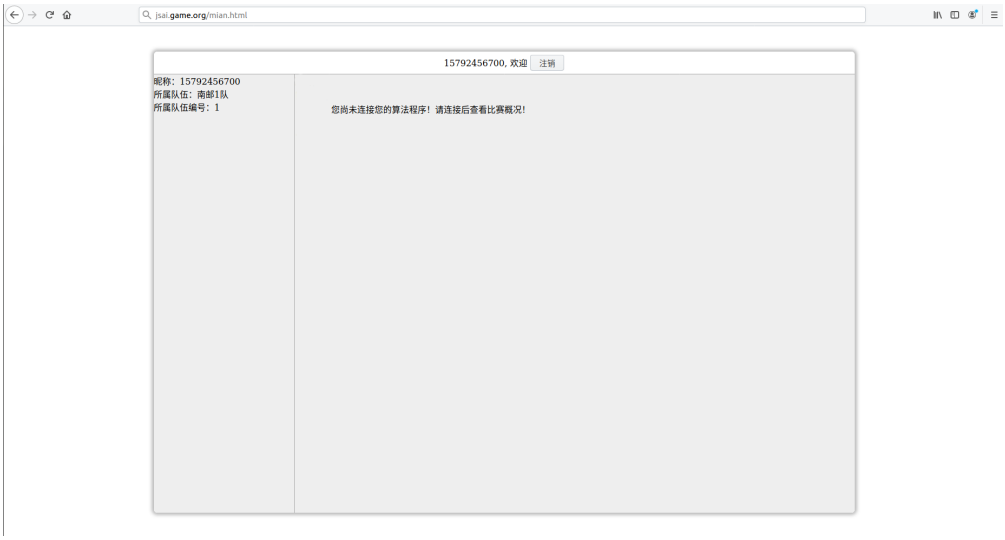
掼蛋AI的编写与连接

1. 不限制编程语言，AI通过Websokcet来连接平台，平台与AI之间通过JSON数据的交互来进行游戏
2. 在线平台连接地址：ws://168.0.1.23:8000/**{PHONE_NUMBER}**
    - 实际连接域名将于开赛前一星期进行公布
    - **{PHONE_NUMBER}** 为参赛选手报名时所填手机号，进行替换即可。若不符合报名时所填手机号则将连接失败
3. 连接成功后平台将发送连接成功的消息给所连接的算法程序（如下图所示），算法程序无需做出回应（做出回应也不会有任何影响）。
    - 

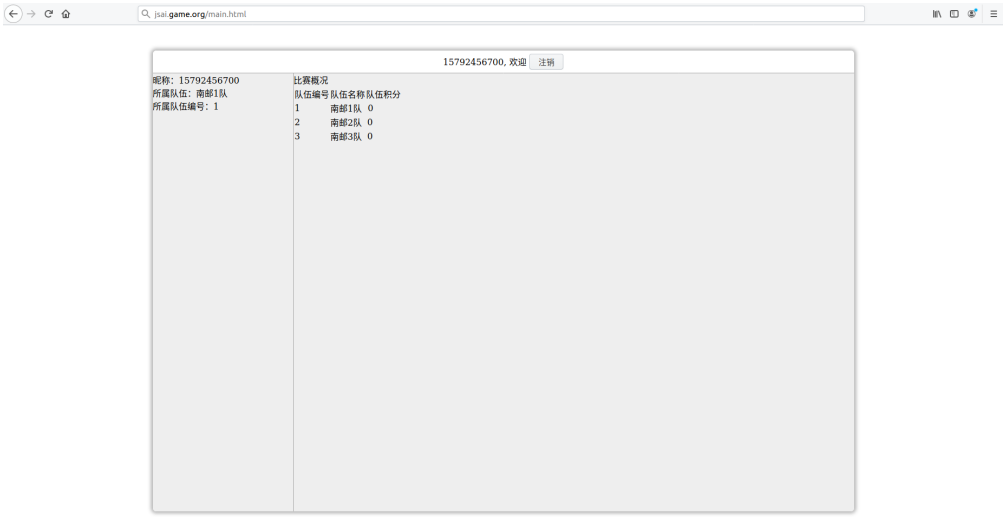
    - 选手亦可登录大赛官方网址所给出的网站上进行登录，即可验证自己的算法程序是否连接成功（下图域名非最终网址）

- 

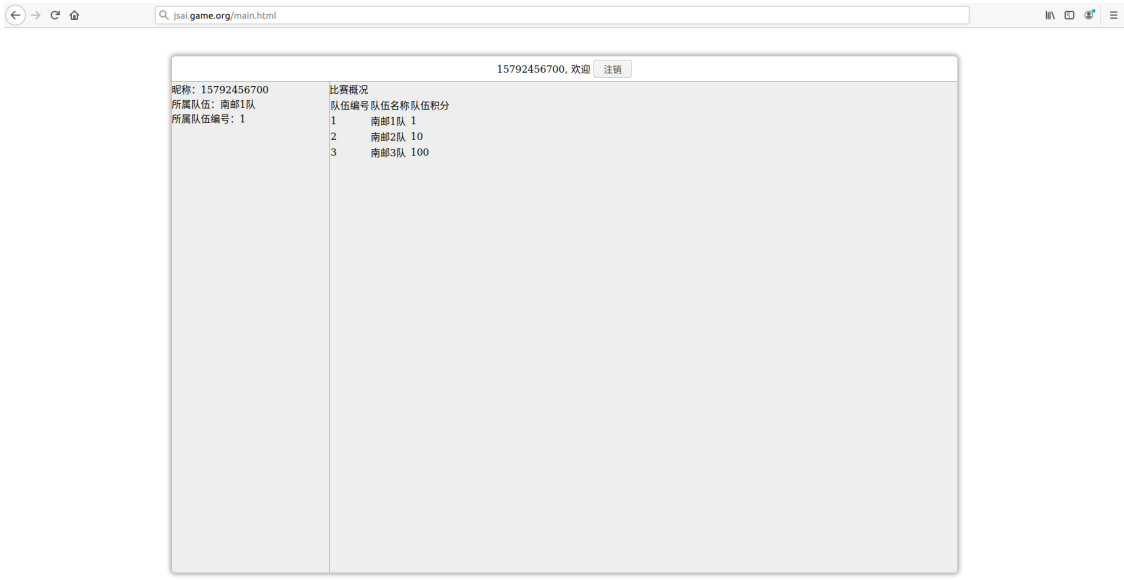
- 若参赛选手未连接算法程序，则会出现如下图所示的页面



- 若参赛选手已经成功连接算法程序并成功登录，能够看见所有准备就绪的队伍，如下图所示的页面



4. 按大赛规则，开赛后选手登录后可实时观看到自己队伍所得的成绩，如下图所示



5. **AI程序应针对不同的JSON数据进行相应的解析，并进行相对应的动作。JSON数据结构的说明仅有1点与离线平台的JSON数据有所区别，具体见下**
6. **提供随机AI程序详见文档末尾的附件**

JSON数据说明

1. **为尽量降低丢包等网络问题对大赛造成影响，在线平台的JSON格式需要在原有的离线平台的JSON上加上一个msg_index的字段，该字段表示服务器所发送的数据序列号，算法程序应在进行动作中包含对应数据的序列号。**
    - 以随机示例代码为例，content为反序列化服务器JSON所得到的序列化对象，在send时（即发送消息时），需要多包含一个msg_index字段，其值与content["msg_index"]中的值相同
    - ```
      def received_message(self, message):
              content = json.loads(str(message))
              if "action_list" in content and content["action_list"]:
                  card_type =
      random.choice(list(content["action_list"].keys()))
                  print("Choose type ", card_type)
                  rank = random.choice(list(content["action_list"]
      [card_type].keys()))
                  action = random.choice(list(content["action_list"]
      [card_type][rank]))
                  print("Choose action:", action)
                  self.send(json.dumps(
                      {"action": action, "type": card_type, "rank": rank,
      "msg_index":content["msg_index"]}
                  ))
      ```
    - 其他内容与离线平台保持不变
2. 卡牌：[{suit}, {rank}]
    - 一张卡牌表示为一个列表数据结构，其长度始终为2
    - 参数suit表示卡牌花色，类型为整型，取值范围为[0, 1, 2, 3]
    - 参数rank表示卡牌点数，类型为字符串，取值范围为['A', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'JOKER']
    - 如[3, '2']表示方片2，[0, 'Q']表示黑桃Q；特别地[0, 'JOKER']表示小王，[1, 'JOKER']表示大王，[0, 'PASS']表示PASS
      ```

3. 牌型： {'rank': {rank}, 'type': {type}, 'action': {action}}

- 一个牌型表示为一个字典数据结构，包含三个固定字段：rank，type，action
  - 为避免33A44与33A44的等易混淆情况（A为配牌），故对一个牌型要给出rank和type
- type 用于表示该牌型，类型为字符串，取值范围为['Single', 'Pair', 'Trips', 'ThreePair', 'ThreeWithTwo','TripsPair', 'Straight', 'Straight', 'Bombg','PASS', 'tribute', 'back']
  - 牌型的中英文对照见附件
- rank用于表示该牌型的点数，类型为字符串，取值范围为['A', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'JOKER'， 'PASS']
- action用于表示该牌型所包含的卡牌，类型为列表数据结构
- 特别地，定义如下三种牌型：
  - PASS：{'rank': 'PASS', 'type': 'PASS', 'action': [[0, 'PASS']]}
  - 进贡tribute例子：{'action': [[1, '4']], 'type': 'tribute', 'rank': '4'}
  - 还贡back例子：{'action': [[0, '5']], 'type': 'back', 'rank': '5'}

3 . **整个游戏过程中，AI仅会收到平台发来的3种JSON数据格式**，可通过JSON中字段type的值来判定是何种类型

- **详细字段值说明见《掼蛋API简易文档》，此处只提供简略说明**
- type: 0，表示小局游戏结束，包含字段如下：

```
{
    'msg_index': 1,
    'type': 0,
    'self_rank': {self_rank},
    'opponent_rank': {opponent_rank},
    'current_rank': {current_rank},
    'tringning_times': {tringning_times},
    'winners': {winners}
}
```

- type: 1，表示通知类型，平台用于广播其他AI所发出的动作。
- type: 2，表示动作选择类型，平台单播给当前需要做出动作的AI，AI需要做出响应
- 类型1与类型2的JSON数据格式包含相同的字段，如下所示：

```
{
    'msg_index': 1
    'type': 0,
    'hand_cards': {hand_cards},
    'public': {public},
    'current_player': {current_player},
    'current_stage': {current_stage},
    'self_rank': {self_rank},
    'opponent_rank': {opponent_rank},
    'current_rank': {current_rank},
    'tringning_times': {tringning_times},
    'winners': {winners},
    'action_list': {action_list},
    'action_performed': {action_performed}
}
```

4 . **整个游戏过程中，AI仅需要向平台发送一种固定格式的JSON，即牌型**

- AI仅在收到类型2的JSON格式时，才需向平台做出动作响应

- 如下方表示某AI打出对子Q

```
{'action': [[0, 'Q'], [2, 'Q']], 'type': 'Pair', 'rank': 'Q'}
```

- 某AI选择PASS

```
{'action': [[0, 'PASS']], 'type': 'PASS', 'rank': 'PASS'}
```

- 在需要做出动作时，AI可以自行构造动作，也可从action_list字段中搜寻合法动作

附件-中英文对照表

牌型名称	英文
单张	Single
对子	Pair
三张	Trips
三连对	ThreePair
三带二	ThreeWithTwo
钢板（两个三张）	TripsPair
顺子	Straight
同花顺	StraightFlush
炸弹	Bomb
进贡	tribute
还贡	back
过	PASS

附件-随机AI示例程序Python版本

```
import json
import random
from ws4py.client.threadedclient import WebSocketClient


class ExampleClient(WebSocketClient):

    def __init__(self, url):
        super().__init__(url)
        self.name = "Agent1"

    def opened(self):
        pass

    def closed(self, code, reason=None):
        print("Closed down", code, reason)
```

```
    def received_message(self, message):
        content = json.loads(str(message))
        print(content)
        if "action_list" in content and content["action_list"]:
            card_type = random.choice(list(content["action_list"].keys()))
            print("Choose type ", card_type)
            rank = random.choice(list(content["action_list"][card_type].keys()))
            action = random.choice(list(content["action_list"][card_type]
[rank]))
            print("Choose action:", action)
            self.send(json.dumps(
                {"action": action, "type": card_type, "rank": rank, "msg_index":
content["msg_index"]}
            ))


if __name__ == '__main__':
    try:
        ws = ExampleClient('ws://168.0.1.23:8000/{PHONE_NUMBER}')
        ws.connect()
        ws.run_forever()
    except KeyboardInterrupt:
        ws.close()
```