ネットワークプログラミングⅡ
―麻雀ゲームの作成―

学籍番号：16418
5 年 電子·情報工学科 16 番
多田 洋輔

学籍番号：16426
5 年 電子·情報工学科 24 番
福澤 大地

提出日：2020 年 8 月 6 日

1 作成したプログラム

オンライン対戦可能な麻雀ゲームを作成した。ルールはオーソドックスな日本のリーチ麻雀であるが、3人/4 人対戦どちらも対応していたり、AI を交えた対戦が可能であったりと、細かな機能を充実させた。また、操作は CUI で行うが、OpenCV を用いることで卓上の状態をグラフィカルに表示した。

使用したプログラミング言語は、Python 3.7.4 である。

2 動作説明

麻雀で行う主な動作について、サーバー側とプレイヤー側でどのような動作をしているかを示す。

2.1 ポン

図 1 から図 3 にポンをした時の動作を示す。

図 1　ポンの動作（サーバー）

図 2　ポンの動作（コンソール）

図 3　ポンの動作（GUI）

　サーバーではポンをしたプレイヤーの手牌からポンの構成牌がなくなっている。コンソールではポンするかどうかに対し、y(yes) を入力している。これに対し GUI にはポンの様子が表示されている。

2.2 チー

図 4 から図 6 にチーをした時の動作を示す。

図 4　チーの動作（サーバー）

図 5　チーの動作（コンソール）

図 6　チーの動作（GUI）

　サーバーでは、チーをしたプレイヤーが以前にポンをしていたため、手牌から 2 鳴き分の構成牌が消えている。コンソールではチーするかどうかに対し、y(yes) を入力している。これに対し GUI にはチーの様子が表示されている。

2.3　暗カン

　図 7 から図 8 に暗カンをした時の動作を示す。

図 7　暗カンの動作（サーバー）

図 8　暗カンの動作（GUI）

　サーバーでは暗カンをしたプレイヤーの手牌から暗カンに使われた牌が消えている。AI が暗カンをしたためコンソールの表示はないが、GUI には暗カンの様子が表示されている。
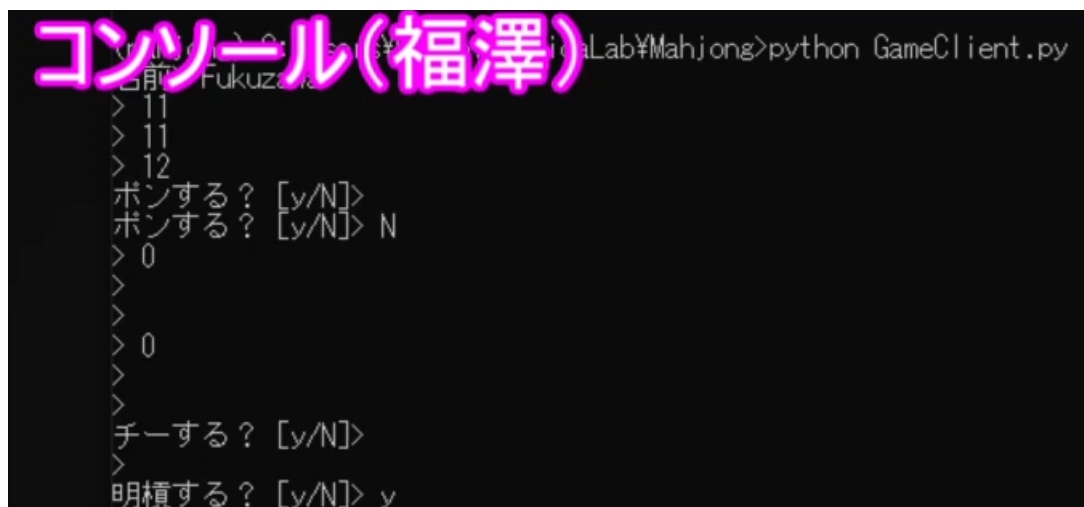
2.4 明カン

図 9 から図 11 に明カンをした時の動作を示す。

図 9　明カンの動作（サーバー）

図 10　明カンの動作（コンソール）

図 11　明カンの動作（GUI）

　サーバーでは明カンを舌プレイヤーの手牌から明カンに使われた牌が消えている。コンソールでは暗カンするかどうかに対し、y(yes) が入力されている。これに対し GUI には明カンの様子が表示されている。

2.5　ロン

　図 12 から図 14 にロンをした時の動作を示す。

図 12　ロンの動作（サーバー）

図 13　ロンの動作（コンソール）

図 14　ロンの動作（GUI）

　サーバーではロンに関わったプレイヤーと、役の内容が表示されている。コンソールではロンするかどうかに対し y(yes) が入力されている。これに対し GUI ではロンをした牌が赤みがかり、ロンを表現している。

2.6　ツモ

　図 15 から図 17 に打牌時の動作を示す。

図 15　ツモの動作（サーバー）

図 16　ツモの動作（コンソール）

図 17 ツモの動作（GUI）

サーバーではツモをしたプレイヤーと、役の内容が表示されている。コンソールではツモをするかどうかに対し y(yes) を入力している。これに対し GUI にはツモった牌を表示している。

2.7 流局

図 18 から図 19 に流局した時の動作を示す。
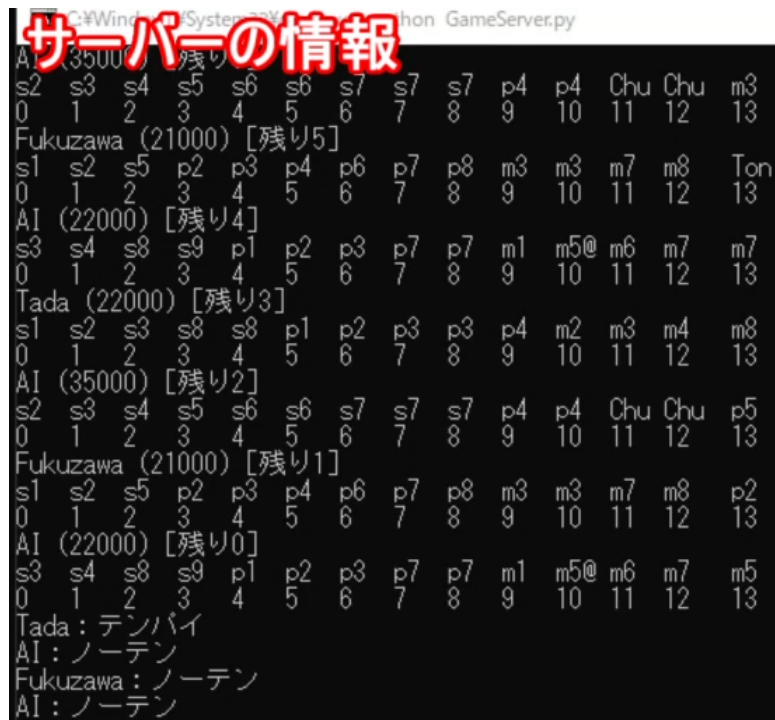
図 18　流局の動作（サーバー）

図 19　流局の動作（GUI）

サーバーでは各プレイヤーのテンパイ状況を表示している。これに対し GUI ではテンパイのプレイヤーの

12

手牌が倒されて表示されている。

3 クラス図

本システムのクラス図を図 20 に示す。詳細な説明は省略するが、ゲームの進行を表す `Game` クラスを `GameServer` クラスに継承し、ソケット通信でオンライン対戦を行う機能をオーバーライドしている。

4 ファイル構造

本システムのファイル構造を図 21 に示す。`GameServer.py` はサーバーサイドのプログラム、`GameClient.py` はクライアントサイドのプログラムである。その他の麻雀システムのプログラムは `mjgame` フォルダ内にまとめられている。

```
/mahjong
├── GameServer.py
├── GameClient.py
└── /mjgame
    ├── __init__.py
    ├── /system
    │   ├── __init__.py
    │   ├── core.py
    │   ├── game.py
    │   ├── player.py
    │   └── tehai.py
    ├── /players
    │   ├── __init__.py
    │   ├── human.py
    │   └── ai.py
    └── /graphic
        ├── __init__.py
        ├── graphic.py
        ├── /image
        ├── /mjhai
        └── /font
```

図 21 ファイル構造

Game

Attributes
+calculator: HandCalculator
+hai_set: Hai[]
+players: Player[]
+player_num: int
+start_point: int
+game_reader: GameReader
+bakaze: int
+kyoku: int
+honba: int
+kyotaku: int
+turn: int
+cur_player: Player
+yama: Yama
+dora_hais: Hai[]
+uradora_hais: Hai[]
+doras: tuple<int, int>[]
+uradoras: tuple<int, int>[]

Methods
+__init__(hai_set: Hai[], players: Player[], start_point: int)
+start()
+start_kyoku(): tuple<bool, bool>
+next_kyoku(rencha: bool, ryukyoku: bool)
+next_player()
+change_player(chicha: int)
+add_dora()
+update_game_reader(open_players: Player[], uradora: bool)
+yaku(player: Player, is_tsumo: bool, agari_hat: Hai, chankan: bool): HandResponse
+dora_kind(hai: Hai): tuple<int, int>
+kyoku_name(): str
+jikaze(player: Player): int
+haipai(player: Player)
+tsumo_pop(player: Player)
+dahai(player: Player, index: int): KawaHai
+richi(player: Player)
+tsumo(player: Player)
+ron(player: Player, target: KawaHai, whose: Player, chankan: bool)
+ankan(player: Player, hais: Hai[])
+kakan(player: Player, hais: Hai[])
+minkan(player: Player, hais: Hai[], target: KawaHai, whose: Player)
+pon(player: Player, hais: Hai[], target: KawaHai, whose: Player)
+chi(player: Player, hais: Hai[], target: KawaHai, whose: Player)
+ryukyoku()
+check_tsumo(player: Player): bool
+check_ron(player: Player, target: Hai, whose: Player): bool
+check_ankan(player: Player): Hais[][]
+check_kakan(player: Player): Hais[]
+check_minkan(player: Player, target: Hai): Hais[][]
+check_pon(player: Player, target: Hai): Hais[][]
+check_chi(player: Player, target: Hai): Hais[][]
+on_dahai(player: Player): int
+on_richi(player: Player): bool
+on_tsumo(player: Player): bool
+on_ron(player: Player, target: Hai, whose: Player): bool
+on_ankan(player: Player, target: Hai): bool
+on_minkan(player: Player, hais: Hai[], target: Hai, whose: Player): bool
+on_kakan(player: Player, hais: Hai[], target: Hai): bool
+on_pon(player: Player, hais: Hai[], target: Hai, whose: Player): bool
+on_chi(player: Player, hais: Hai[], target: Hai, whose: Player): bool

Player *(abstract)*

+name: str
+tehai: Tehai
+kawa: Kawa
+richi: bool
+furo: bool
+ippatsu: bool
+rinshan: bool
+game: GameReader
+chicha: int
+point: int

+__init__()
+relative(other: Player): int
+setup(game: GameReader, chicha: int, start_point: int)
+reset()
+on_dahai(): int
+on_richi(): bool
+on_tsumo(): bool
+on_ron(target: Hai, whose: int): bool
+on_ankan(target: Hai): bool
+on_minkan(hais: Hai[], target: Hai, whose: int): bool
+on_kakan(hais: Hai[], target: Hai): bool
+on_pon(hais: Hai[], target: Hai, whose: int): bool
+on_chi(hais: Hai[], target: Hai, whose: int): bool

Yama

+hais: Hai[]
+remain: int
+doras: Hai[]
+uradoras: Hai[]
+__init__(hai_set: Hai[])
+pop(): Hai
+add_dora()

GameServer

+screen: Screen
+__init__(hai_set: Hai[], players: Player[], start_point: int)
+connect()
+receive(player: Player): bytes
+request(player: Player, event_kind: EventKind): bytes
+update_game_reader(open_players: Player[], uradora: bool)
+start()
+start_kyoku(): tuple<bool, bool>
+on_dahai(player: Player): int
+on_richi(player: Player): bool
+on_tsumo(player: Player): bool
+on_ron(player: Player, target: Hai, whose: Player): bool
+on_ankan(player: Player, target: Hai): bool
+on_minkan(player: Player, hais: Hai[], target: Hai, whose: Player): bool
+on_kakan(player: Player, hais: Hai[], target: Hai): bool
+on_pon(player: Player, hais: Hai[], target: Hai, whose: Player): bool
+on_chi(player: Player, hais: Hai[], target: Hai, whose: Player): bool

Human

+name: str
+__init__()
+confirm(message: str, default: bool): bool
+on_dahai(): int
+on_richi(): bool
+on_tsumo(): bool
+on_ron(target: Hai, whose: int): bool
+on_ankan(target: Hai): bool
+on_minkan(hais: Hai[], target: Hai, whose: int): bool
+on_kakan(hais: Hai[], target: Hai): bool
+on_pon(hais: Hai[], target: Hai, whose: int): bool
+on_chi(hais: Hai[], target: Hai, whose: int): bool

Menzen

+name: str
+__init__()
+on_dahai(): int
+on_richi(): bool
+on_tsumo(): bool
+on_ron(target: Hai, whose: int): bool
+on_ankan(target: Hai): bool
+on_minkan(hais: Hai[], target: Hai, whose: int): bool
+on_kakan(hais: Hai[], target: Hai): bool
+on_pon(hais: Hai[], target: Hai, whose: int): bool
+on_chi(hais: Hai[], target: Hai, whose: int): bool

GameReader

+hai_set: Hai[]
+player_num: int
+start_point: int
+bakaze: int
+kyoku: int
+honba: int
+kyotaku: int
+players: PlayerReader[]
+turn: int
+cur_player: PlayerReader
+yama_remain: int
+dora_hais: Hai[]
+uradora_hais: Hai[]
+doras: tuple<int, int>[]
+uradoras: tuple<int, int>[]
+__init__(player_num: int, start_point: int)

EventKind *(enumeration)*

UPDATE
DAHAI
RICHI
TSUMO
RON
ANKAN
MINKAN
KAKAN
PON
CHI
RESULT
COMPLATE
OVER
END

PlayerReader

+name: str
+point: int
+tehai: TehaiReader
+kawa: Kawa
+richi: bool
+furo: bool
+ippatsu: bool
+__init__(start_point: int)

Kawa

+hais: KawaHai[]
+__init__()
+append(hai: Hai, tsumogiri, richi)

Tehai

+calculator: Shanten
+hais: Hai[]
+furos: Furo[]
+tsumo_hai: Hai
+hai_num: int
+__init__(hais: Hai[], furos: Furo[])
+store()
+tsumo(hai: Hai)
+append(hais: Hai[])
+insert(index: int, hai: Hai)
+pop(index: int): Hai
+remove(hai): Hai
+find(kind: int, num: int): Hai
+find_multi(kinds: int[], nums: int[]): Hai[]
+sort()
+anako_able(): Hais[][]
+kakan_able(): Hais[]
+minkan_able(target: Hai): Hais[][]
+pon_able(target: Hai): Hais[][]
+chi_able(target: Hai): Hais[][]
+ankan(hais: Hai[])
+kakan(hai: Hai)
+minkan(hais: Hai[], target: Hai, direct: int)
+pon(hais: Hai[], target: Hai, direct: int)
+chi(hais: Hai[], target: Hai, direct: int)
+show()
+shanten(): int

TehaiReader

+hais: Hai[]
+furos: Furo[]
+hai_num: int
+tsumo_hai: Hai
+exist_tsumo_hai: bool
+__init__()

KawaHai

+hai: Hai
+tsumogiri: bool
+richi: bool
+furo: bool
+hoju: bool
+__init__(hai: Hai, tsumogiri: bool, richi: bool)

Furo

+hais: Hai[]
+kind: FuroKind
+direct: int
+__init__(hais: Hai[], kind: FuroKind, direct: int)

Hai

+Kind: int
+num: int
+red: bool
+__init__(kind: int, num: int, red: bool)
+__eq__(other: Hai): bool
+__lt__(other: Hai): bool
+__gt__(other: Hai): bool

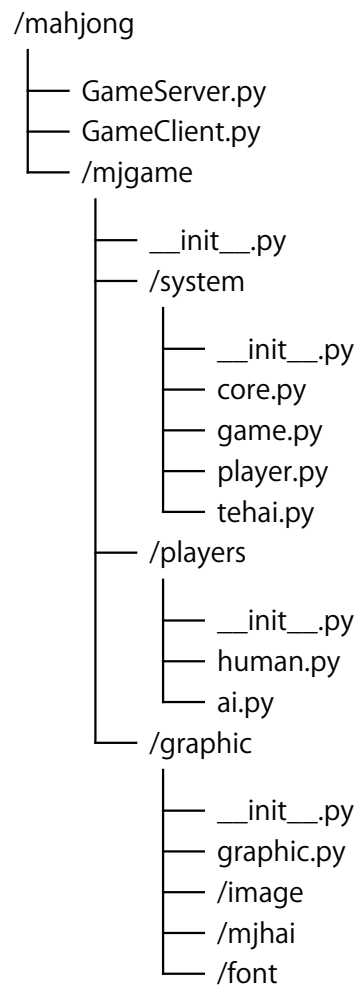## FuroKind *(enumeration)*

PON
CHI
ANKAN
MINKAN
KAKAN

図 20　クラス図

15

5 プログラムリスト

　サーバーサイドのプログラムをリスト 1 に、クライアントサイドのプログラムをリスト 2 に示す。その他の
麻雀システムのプログラムはリスト 3–13 に示す。

リスト 1　GameServer.py

```python
import time
import random
import copy
import pickle
import socket

import mjgame.system as mj
import mjgame.players as mp

# 通信する処理の種類
class EventKind():
    UPDATE   = b"upd"
    DAHAI    = b"dah"
    RICHI    = b"ric"
    TSUMO    = b"tmo"
    RON      = b"ron"
    ANKAN    = b"aka"
    MINKAN   = b"mka"
    KAKAN    = b"kka"
    PON      = b"pon"
    CHI      = b"chi"
    RESULT   = b"res"
    COMPLETE = b"com"
    OVER     = b"ovr"
    END      = b"end"

# サーバーサイド
class GameServer(mj.Game):
    def __init__(self, hai_set, players, start_point):
        super().__init__(hai_set, players, start_point)

        self.conns = {}
        self.addrs = {}

    # 接続が確率されるまで待機
    def connect(self):
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            # サーバーの設定
            s.bind(("0.0.0.0", 50005))
            s.listen(self.player_num)

            num = 1

            for i in range(self.player_num):
                player = self.players[i]

                if self.players[i].__class__.__name__ != "Human":
                    continue
```

```python
49
50                    # 接続待機
51                    print("No. {} Player: Waiting...".format(num))
52                    conn, addr = s.accept()
53
54                    # リストに追加
55                    self.conns[i] = conn
56                    self.addrs[i] = addr
57
58                    # 名前を受信
59                    player_name = self.receive(player)
60                    player.name = player_name.decode()
61
62                    print("No. {} Player: Connected".format(num))
63                    print("Name: {}".format(player.name))
64
65                    num += 1
66
67      # 受信
68      def receive(self, player):
69          conn = self.conns[player.chicha]
70          recv_data = b""
71
72          # データが終端に達するまで受信
73          while True:
74              data = conn.recv(4096)
75              recv_data += data
76
77              if recv_data[-3:] == EventKind.END:
78                  break
79
80          return recv_data[:-3]
81
82      # 選択を要求
83      def request(self, player, event_kind):
84          conn = self.conns[player.chicha]
85          conn.sendall(event_kind + EventKind.END)
86          return self.receive(player)
87
88      # ゲームリーダーを更新
89      def update_game_reader(self, open_players = [], uradora = False):
90          super().update_game_reader(open_players, uradora)
91
92          # 全てのクライアントにゲームリーダーを送信
93          for i in range(self.player_num):
94              player = self.players[i]
95
96              if player.__class__.__name__ != "Human":
97                  continue
98
99              send_data = pickle.dumps(self.players[i])
100             self.conns[i].sendall(EventKind.UPDATE + send_data + EventKind.END)
101
102         # 全てのクライアントから応答があるまで待機
103         for i in range(self.player_num):
104             player = self.players[i]
```

```
105
106                 if player.__class__.__name__ != "Human":
107                     continue
108
109                 self.receive(player)
110
111         # ゲーム開始
112         def start(self):
113             super().start()
114
115             # 終局
116             for i in range(self.player_num):
117                 player = self.players[i]
118
119                 if player.__class__.__name__ != "Human":
120                     continue
121
122                 self.conns[i].sendall(EventKind.OVER + EventKind.END)
123
124         # 局開始
125         def start_kyoku(self):
126             renchan, ryukyoku = super().start_kyoku()
127
128             # 終局
129             for i in range(self.player_num):
130                 player = self.players[i]
131
132                 if player.__class__.__name__ != "Human":
133                     continue
134
135                 self.conns[i].sendall(EventKind.RESULT + EventKind.END)
136
137             # 全てのクライアントから応答があるまで待機
138             for i in range(self.player_num):
139                 player = self.players[i]
140
141                 if player.__class__.__name__ != "Human":
142                     continue
143
144                 self.receive(player)
145
146             return renchan, ryukyoku
147
148         # 選択
149         def on_dahai(self, player):
150             if player.__class__.__name__ == "Human":
151                 data = self.request(player, EventKind.DAHAI)
152                 return int.from_bytes(data, "big", signed=True)
153             else:
154                 return super().on_dahai(player)
155
156         # 立直するか
157         def on_richi(self, player):
158             if player.__class__.__name__ == "Human":
159                 data = self.request(player, EventKind.RICHI)
160                 return bool.from_bytes(data, "big", signed=True)
```

```
161            else:
162                return super().on_richi(player)
163
164        # ツモ和了するか
165        def on_tsumo(self, player):
166            if player.__class__.__name__ == "Human":
167                data = self.request(player, EventKind.TSUMO)
168                return bool.from_bytes(data, "big", signed=True)
169            else:
170                return super().on_tsumo(player)
171
172        # ロン和了するか
173        def on_ron(self, player, target, whose):
174            if player.__class__.__name__ == "Human":
175                data = self.request(player, EventKind.RON)
176                return bool.from_bytes(data, "big", signed=True)
177            else:
178                return super().on_ron(player, target, whose)
179
180        # 暗槓するか
181        def on_ankan(self, player, target):
182            if player.__class__.__name__ == "Human":
183                data = self.request(player, EventKind.ANKAN)
184                return bool.from_bytes(data, "big", signed=True)
185            else:
186                return super().on_ankan(player, target)
187
188        # 明槓するか
189        def on_minkan(self, player, hais, target, whose):
190            if player.__class__.__name__ == "Human":
191                data = self.request(player, EventKind.MINKAN)
192                return bool.from_bytes(data, "big", signed=True)
193            else:
194                return super().on_minkan(player, hais, target, whose)
195
196        # 加槓するか
197        def on_kakan(self, player, target):
198            if player.__class__.__name__ == "Human":
199                data = self.request(player, EventKind.KAKAN)
200                return bool.from_bytes(data, "big", signed=True)
201            else:
202                return super().on_kakan(player, target)
203
204        # ポンするか
205        def on_pon(self, player, hais, target, whose):
206            if player.__class__.__name__ == "Human":
207                data = self.request(player, EventKind.PON)
208                return bool.from_bytes(data, "big", signed=True)
209            else:
210                return super().on_pon(player, hais, target, whose)
211
212        # チーするか
213        def on_chi(self, player, hais, target, whose):
214            if player.__class__.__name__ == "Human":
215                data = self.request(player, EventKind.CHI)
216                return bool.from_bytes(data, "big", signed=True)
```

```
217            else:
218                return super().on_chi(player, hais, target, whose)
219
220    if __name__ == "__main__":
221        # 牌をセット
222        hai_set = []
223
224        # 数牌
225        for i in range(3):
226            for j in range(1, 10):
227                hai_set.extend(mj.Hai(i, j, j == 5 and k == 3) for k in range(4))
228
229        # 字牌
230        for i in range(1, 8):
231            hai_set.extend(mj.Hai(3, i) for j in range(4))
232
233        # プレイヤー
234        players = [mp.Human(), mp.AI(), mp.Human(), mp.AI()]
235
236        game = GameServer(hai_set, players, 25000)
237        game.connect()
238        game.start()
```

<div align="center">リスト 2　GameClient.py</div>

```
1    import sys
2    import socket
3    import pickle
4
5    import mjgame.system as mj
6    import mjgame.players as mp
7    import mjgame.graphic as mg
8
9    from GameServer import EventKind
10
11   # 名前を入力
12   player_name = input("名前> ")
13
14   with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
15       # サーバーに接続し名前を送信
16       s.connect(("localhost", 50005))
17       s.sendall(player_name.encode() + EventKind.END)
18
19       while True:
20           recv_data = b""
21
22           # データを受信
23           while True:
24               data = s.recv(4096)
25               recv_data += data
26
27               if recv_data[-3:] == EventKind.END:
28                   break
29
30           # ヘッダと本文を切り離す
31           event_kind = recv_data[:3]
```

```python
            recv_data = recv_data[3:-3]

            # ゲームリーダーのアップデート
            if event_kind == EventKind.UPDATE:
                s.sendall(EventKind.COMPLETE + EventKind.END)
                player = pickle.loads(recv_data)

                screen = mg.Screen(player.game, [player], player, "Mahjong")
                screen.draw()

            # 打牌
            elif event_kind == EventKind.DAHAI:
                ret = player.on_dahai()
                s.sendall(ret.to_bytes(1, "big", signed=True) + EventKind.END)

            # リーチ
            elif event_kind == EventKind.RICHI:
                ret = player.on_richi()
                s.sendall(ret.to_bytes(1, "big") + EventKind.END)

            # ツモ
            elif event_kind == EventKind.TSUMO:
                ret = player.on_tsumo()
                s.sendall(ret.to_bytes(1, "big") + EventKind.END)

            # ロン
            elif event_kind == EventKind.RON:
                ret = player.on_ron(None, None)
                s.sendall(ret.to_bytes(1, "big") + EventKind.END)

            # 暗槓
            elif event_kind == EventKind.ANKAN:
                ret = player.on_ankan(None)
                s.sendall(ret.to_bytes(1, "big") + EventKind.END)

            # 明槓
            elif event_kind == EventKind.MINKAN:
                ret = player.on_minkan(None, None, None)
                s.sendall(ret.to_bytes(1, "big") + EventKind.END)

            # 加槓
            elif event_kind == EventKind.KAKAN:
                ret = player.on_kakan(None)
                s.sendall(ret.to_bytes(1, "big") + EventKind.END)

            # ポン
            elif event_kind == EventKind.PON:
                ret = player.on_pon(None, None, None)
                s.sendall(ret.to_bytes(1, "big") + EventKind.END)

            # チー
            elif event_kind == EventKind.CHI:
                ret = player.on_chi(None, None, None)
                s.sendall(ret.to_bytes(1, "big") + EventKind.END)

            # 終局
```

```
88          elif event_kind == EventKind.RESULT:
89              screen = mg.Screen(player.game, [player], player, "Mahjong")
90              screen.draw()
91
92              if input("Press Enter Key...") == "q":
93                  sys.exit()
94
95              s.sendall(EventKind.COMPLETE + EventKind.END)
96
97          # ゲーム終了
98          elif event_kind == EventKind.OVER:
99              break
```

リスト 3　mjgame/__init__.py

```
1  from .system import *
2  from .players import *
3  from .graphic import *
```

リスト 4　mjgame/system/__init__.py

```
1  from .core import *
2  from .player import *
3  from .game import *
```

リスト 5　mjgame/system/core.py

```
1   import enum
2
3   # 河
4   class Kawa():
5       def __init__(self):
6           self.hais = []
7
8       # 追加
9       def append(self, hai, tsumogiri=False, richi=False):
10          self.hais.append(KawaHai(hai, tsumogiri, richi))
11
12  # 河の麻雀牌
13  class KawaHai():
14      def __init__(self, hai, tsumogiri=False, richi=False):
15          self.hai = hai
16          self.tsumogiri = tsumogiri
17          self.richi = richi
18          self.furo = False
19          self.hoju = False
20
21  # ゲームリーダー
22  class GameReader():
23      def __init__(self, hai_set, player_num, start_point):
24          self.hai_set = hai_set
25          self.player_num = player_num
26          self.start_point = start_point
27
28          self.bakaze = 0
29          self.kyoku = 0
```

```python
            self.honba = 0
            self.kyotaku = 0

            self.players = [PlayerReader(i) for i in range(self.player_num)]
            self.turn = 0
            self.cur_player = self.players[self.turn]

            self.yama_remain = 0
            self.dora_hais = []
            self.uradora_hais = None
            self.doras = []
            self.uradoras = None

# プレイヤーリーダー
class PlayerReader():
    def __init__(self, chicha):
        self.name = ""
        self.chicha = chicha
        self.point = 0
        self.tehai = TehaiReader()
        self.kawa = Kawa()
        self.richi = False
        self.furo = False
        self.ippatsu = False

# 手牌リーダー
class TehaiReader():
    def __init__(self):
        self.hais = None
        self.furos = []
        self.hai_num = 0
        self.tsumo_hai = None
        self.exist_tsumo_hai = False

# 副露
class Furo():
    def __init__(self, hais, kind, direct=0):
        self.hais = hais
        self.kind = kind
        self.direct = direct

# 副露の種類
class FuroKind(enum.Enum):
    PON = enum.auto()
    CHI = enum.auto()
    ANKAN = enum.auto()
    MINKAN = enum.auto()
    KAKAN = enum.auto()

# 麻雀牌
class Hai():
    """
    kind:
      0 ... 索子
      1 ... 筒子
      2 ... 萬子
```

```
 86          3 ... 字牌
 87
 88      num:
 89        1-9 ... 数字
 90        or
 91        1-7 ... 東～中
 92      """
 93
 94      def __init__(self, kind, num, red=False):
 95          self.kind = kind # 種類
 96          self.num = num   # 数字
 97          self.red = red   # 赤ドラかどうか
 98
 99          COLOR_NAME = ["s", "p", "m"]
100          JIHAI_NAME = ["Ton", "Nan", "Sha", "Pei", "Hak", "Hat", "Chu"]
101
102          # 名称
103          if self.kind <= 2:
104              self.name = "{}{}{}".format(
105                  COLOR_NAME[self.kind],
106                  self.num,
107                  "@" if self.red else ""
108              )
109          else:
110              self.name = "{}{}".format(
111                  JIHAI_NAME[self.num - 1],
112                  "@" if self.red else ""
113              )
114
115      # 比較演算子
116      def __eq__(self, other):
117          return (self.kind, self.num, self.red) == (other.kind, other.num, other.red)
118
119      def __lt__(self, other):
120          return (self.kind, self.num, self.red) < (other.kind, other.num, other.red)
121
122      def __gt__(self, other):
123          return (self.kind, self.num, self.red) > (other.kind, other.num, other.red)
```

リスト 6   mjgame/system/game.py

```
 1  import sys
 2  import time
 3  import random
 4  import copy
 5  import pickle
 6
 7  from .core import *
 8  from .. import graphic as mg
 9
10  from mahjong.hand_calculating.hand import HandCalculator
11  from mahjong.meld import Meld
12  from mahjong.hand_calculating.hand_config import HandConfig, OptionalRules
13  from mahjong.tile import TilesConverter
14  from mahjong.constants import EAST, SOUTH, WEST, NORTH
15
```

```python
# ゲーム
class Game():
    def __init__(self, hai_set, players, start_point):
        seed = time.time()
        random.seed(seed)
        print("seed = {}".format(seed))

        self.calculator = HandCalculator()

        self.hai_set = hai_set
        self.players = players
        self.player_num = len(self.players)
        self.start_point = start_point

        self.game_reader = GameReader(self.hai_set, self.player_num, self.start_point)

        for i, player in enumerate(self.players):
            player.setup(self.game_reader, i, self.start_point)

        self.bakaze = 0  # 場風
        self.kyoku = 0   # 局
        self.honba = 0   # 本場
        self.kyotaku = 0 # 供託

        # 番
        self.turn = 0
        self.cur_player = self.players[self.turn]

        # 牌山
        self.yama = Yama(self.hai_set)
        self.dora_hais = [self.yama.doras[0]]
        self.uradora_hais = [self.yama.uradoras[0]]
        self.doras = [self.dora_kind(self.dora_hais[0])]
        self.uradoras = [self.dora_kind(self.uradora_hais[0])]

    # ゲーム開始
    def start(self):
        while self.bakaze <= 1:
            renchan, ryukyoku = self.start_kyoku()
            self.next_kyoku(renchan, ryukyoku)

    # 局開始
    def start_kyoku(self):
        # コンソール表示
        print("{} {}本場 供託{}".format(self.kyoku_name(), self.honba, self.kyotaku))

        # 配牌
        for player in self.players:
            self.haipai(player)

        while self.yama.remain > 0:
            # ツモ
            self.tsumo_pop(self.cur_player)

            # コンソール表示
            print("{} ({}) [残り{}]".format(self.cur_player.name, self.cur_player.point,
```

```
                 self.yama.remain))
72               self.cur_player.tehai.show()
73
74               # ツモ判定
75               if self.check_tsumo(self.cur_player) and self.on_tsumo(self.cur_player):
76                   self.tsumo(self.cur_player)
77                   return self.jikaze(self.cur_player) == 0, False
78
79               cont = False
80
81               # 暗槓
82               for cur_hais in self.check_ankan(self.cur_player):
83                   if self.on_ankan(self.cur_player, cur_hais):
84                       self.ankan(self.cur_player, cur_hais)
85                       cont = True
86                       break
87
88               if cont:
89                   continue
90
91               # 加槓
92               for cur_hai in self.check_kakan(self.cur_player):
93                   if self.on_kakan(self.cur_player, cur_hai):
94                       # 槍槓判定
95                       for player in self.players:
96                           # 自身は判定しない
97                           if player == self.cur_player:
98                               continue
99
100                          if self.check_ron(player, cur_hai, self.cur_player) and self.on_ron
      (player, cur_hai.hai, self.cur_player):
101                              self.ron(player, cur_hai, self.cur_player, True)
102
103                              ron = True
104                              if self.jikaze(player) == 0:
105                                  renchan = True
106
107                      if ron:
108                          return renchan, False
109
110                  self.kakan(self.cur_player, cur_hai)
111                  cont = True
112                  break
113
114              if cont:
115                  continue
116
117              while True:
118                  # 打牌
119                  if self.cur_player.richi:
120                      index = -1
121                  else:
122                      index = self.on_dahai(self.cur_player)
123
124                  check_hai = self.dahai(self.cur_player, index)
125
```

```python
                    # 立直
                    if not self.cur_player.richi and not self.cur_player.furo and self.
        cur_player.tehai.shanten() == 0:
                        if self.on_richi(self.cur_player):
                            self.richi(self.cur_player)

                ron = False
                renchan = False
                minkan = False
                furo = False

                # ロン判定
                for player in self.players:
                    # 自身は判定しない
                    if player == self.cur_player:
                        continue

                    if self.check_ron(player, check_hai.hai, self.cur_player) and self.
        on_ron(player, check_hai.hai, self.cur_player):
                        self.ron(player, check_hai, self.cur_player)

                        ron = True
                        if self.jikaze(player) == 0:
                            renchan = True

                if ron:
                    return renchan, False

                # 副露判定
                for player in self.players:
                    # 自身は判定しない
                    if player == self.cur_player:
                        continue

                    # 明槓
                    for cur_hais in self.check_minkan(player, check_hai.hai):
                        if self.on_minkan(player, cur_hais, check_hai.hai, self.cur_player)
        :
                            self.minkan(player, cur_hais, check_hai, self.cur_player)
                            minkan = True
                            break

                    if minkan:
                        break

                    # ポン
                    for cur_hais in self.check_pon(player, check_hai.hai):
                        if self.on_pon(player, cur_hais, check_hai.hai, self.cur_player):
                            self.pon(player, cur_hais, check_hai, self.cur_player)
                            furo = True
                            break

                    if furo:
                        break

                    # チー
```

```
179                    if self.player_num >= 4 and self.cur_player.relative(player) == 1:
180                        for cur_hais in self.check_chi(player, check_hai.hai):
181                            if self.on_chi(player, cur_hais, check_hai.hai, self.cur_player
     ):
182                                self.chi(player, cur_hais, check_hai, self.cur_player)
183                                furo = True
184                                break
185
186                    if furo:
187                        break
188
189                if minkan or not furo:
190                    break
191
192            if not minkan:
193                self.next_player()
194
195        self.ryukyoku()
196        return self.players[self.kyoku].tehai.shanten() == 0, True
197
198    # 次の局へ
199    def next_kyoku(self, renchan, ryukyoku):
200        # 局を更新
201        if not renchan:
202            self.kyoku += 1
203            if self.kyoku >= self.player_num:
204                self.kyoku = 0
205                self.bakaze += 1
206
207        # 本場
208        if renchan or ryukyoku:
209            self.honba += 1
210        else:
211            self.honba = 0
212
213        # 供託
214        if ryukyoku:
215            for player in self.players:
216                if player.richi:
217                    self.kyotaku += 1
218        else:
219            self.kyotaku = 0
220
221        # リセット
222        self.change_player(self.kyoku)
223        self.yama = Yama(self.hai_set)
224        self.dora_hais = [self.yama.doras[0]]
225        self.uradora_hais = [self.yama.uradoras[0]]
226        self.doras = [self.dora_kind(self.dora_hais[0])]
227        self.uradoras = [self.dora_kind(self.uradora_hais[0])]
228
229        for player in self.players:
230            player.reset()
231
232        self.update_game_reader()
233
```

```python
234        # 次のプレイヤーへ
235        def next_player(self):
236            self.change_player((self.turn + 1) % self.player_num)
237
238        # プレイヤーのツモ順を変更
239        def change_player(self, chicha):
240            self.turn = chicha
241            self.cur_player = self.players[self.turn]
242            self.update_game_reader()
243
244        # ドラ追加
245        def add_dora(self):
246            self.yama.add_dora()
247            self.dora_hais.append(self.yama.doras[-1])
248            self.uradora_hais.append(self.yama.uradoras[-1])
249            self.doras.append(self.dora_kind(self.dora_hais[-1]))
250            self.uradoras.append(self.dora_kind(self.uradora_hais[-1]))
251            self.update_game_reader()
252
253        # ゲームリーダーを更新
254        def update_game_reader(self, open_players = [], uradora = False):
255            self.game_reader.bakaze = self.bakaze
256            self.game_reader.kyoku = self.kyoku
257            self.game_reader.honba = self.honba
258            self.game_reader.kyotaku = self.kyotaku
259
260            self.game_reader.turn = self.turn
261            self.game_reader.cur_player = self.game_reader.players[self.game_reader.turn]
262
263            self.game_reader.yama_remain = self.yama.remain
264            self.game_reader.dora_hais = self.dora_hais
265            self.game_reader.doras = self.doras
266
267            if uradora:
268                self.game_reader.uradora_hais = self.uradora_hais
269                self.game_reader.uradoras = self.uradoras
270            else:
271                self.game_reader.uradora_hais = None
272                self.game_reader.uradoras = None
273
274            for i in range(self.player_num):
275                player = self.game_reader.players[i]
276                player_org = self.players[i]
277
278                player.name = player_org.name
279                player.point = player_org.point
280
281                if player_org in open_players:
282                    player.tehai.hais = player_org.tehai.hais
283                    player.tehai.tsumo_hai = player_org.tehai.tsumo_hai
284                else:
285                    player.tehai.hais = None
286                    player.tehai.tsumo_hai = None
287
288                player.tehai.furos = player_org.tehai.furos
289                player.tehai.hai_num = player_org.tehai.hai_num
```

```
290              player.tehai.exist_tsumo_hai = player_org.tehai.tsumo_hai is not None
291              player.kawa = player_org.kawa
292              player.richi = player_org.richi
293              player.furo = player_org.furo
294              player.ippatsu = player_org.ippatsu
295
296      # 役
297      def yaku(self, player, is_tsumo, agari_hai=None, chankan=False):
298          # 手牌
299          tile_strs = [""] * 4
300          win_tile_strs = [""] * 4
301
302          for hai in player.tehai.hais:
303              tile_strs[hai.kind] += "r" if hai.red else str(hai.num)
304
305          for furo in player.tehai.furos:
306              for hai in furo.hais:
307                  tile_strs[hai.kind] += "r" if hai.red else str(hai.num)
308
309              if furo.kind == FuroKind.ANKAN or furo.kind == FuroKind.MINKAN or furo.kind ==
    FuroKind.KAKAN:
310                  tile_strs[furo.hais[0].kind] = tile_strs[furo.hais[0].kind][:-1]
311
312          if is_tsumo:
313              hai = player.tehai.tsumo_hai
314              tile_strs[hai.kind] += "r" if hai.red else str(hai.num)
315              win_tile_strs[hai.kind] += "r" if hai.red else str(hai.num)
316          else:
317              tile_strs[agari_hai.kind] += "r" if agari_hai.red else str(agari_hai.num)
318              win_tile_strs[agari_hai.kind] += "r" if agari_hai.red else str(agari_hai.num)
319
320          tiles = TilesConverter.string_to_136_array(tile_strs[0], tile_strs[1], tile_strs
    [2], tile_strs[3], True)
321          win_tile = TilesConverter.string_to_136_array(win_tile_strs[0], win_tile_strs[1],
    win_tile_strs[2], win_tile_strs[3], True)[0]
322
323          # 副露
324          FURO_MELD = {
325              FuroKind.PON: Meld.PON,
326              FuroKind.CHI: Meld.CHI,
327              FuroKind.ANKAN: Meld.KAN,
328              FuroKind.MINKAN: Meld.KAN,
329              FuroKind.KAKAN: Meld.CHANKAN
330          }
331
332          melds = []
333
334          for furo in player.tehai.furos:
335              furo_strs = [""] * 4
336
337              for hai in furo.hais:
338                  furo_strs[hai.kind] += "r" if hai.red else str(hai.num)
339
340              meld_tiles = TilesConverter.string_to_136_array(furo_strs[0], furo_strs[1],
    furo_strs[2], furo_strs[3], True)
341              melds.append(Meld(FURO_MELD[furo.kind], meld_tiles, furo.kind != FuroKind.ANKAN
```

```
                ))
342
343                 # ドラ
344                 dora_indicators = []
345
346                 for hai in self.dora_hais:
347                     dora_strs = [""] * 4
348                     dora_strs[hai.kind] += "r" if hai.red else str(hai.num)
349
350                     dora_tile = TilesConverter.string_to_136_array(dora_strs[0], dora_strs[1],
           dora_strs[2], dora_strs[3], True)[0]
351                     dora_indicators.append(dora_tile)
352
353                 if player.richi:
354                     for hai in self.uradora_hais:
355                         dora_strs = [""] * 4
356                         dora_strs[hai.kind] += "r" if hai.red else str(hai.num)
357
358                         dora_tile = TilesConverter.string_to_136_array(dora_strs[0], dora_strs[1],
           dora_strs[2], dora_strs[3], True)[0]
359                         dora_indicators.append(dora_tile)
360
361                 # 設定
362                 KAZE_WIND = [EAST, SOUTH, WEST, NORTH]
363
364                 config = HandConfig(
365                     is_tsumo=is_tsumo,
366                     is_riichi=player.richi,
367                     is_ippatsu=player.ippatsu,
368                     is_rinshan=player.rinshan,
369                     is_chankan=chankan,
370                     is_haitei=agari_hai is None and self.yama.remain == 0,
371                     is_houtei=agari_hai is not None and self.yama.remain == 0,
372                     is_daburu_riichi=len(player.kawa.hais) > 0 and player.kawa.hais[0].richi,
373                     is_tenhou=agari_hai is None and self.jikaze(player) == 0 and len(player.kawa.
           hais) == 0,
374                     is_renhou=agari_hai is not None and len(player.kawa.hais) == 0,
375                     is_chiihou=agari_hai is None and self.jikaze(player) != 0 and len(player.kawa.
           hais) == 0,
376                     player_wind=KAZE_WIND[self.jikaze(player)],
377                     round_wind=KAZE_WIND[self.bakaze],
378                     options=OptionalRules(has_aka_dora=True)
379                 )
380
381                 return self.calculator.estimate_hand_value(tiles, win_tile, melds, dora_indicators,
            config)
382
383         # ドラ表示牌に対応するドラ
384         def dora_kind(self, hai):
385             # 数牌
386             if hai.kind <= 2:
387                 next_num = hai.num
388
389                 while True:
390                     next_num += 1
391                     if (next_num > 9):
```

```
392                    next_num = 1
393
394                if Hai(hai.kind, next_num) in self.hai_set or Hai(hai.kind, next_num, True)
       in self.hai_set:
395                    return (hai.kind, next_num)
396
397        # 東南西北
398        elif 1 <= hai.num <= 4:
399            next_num = hai.num + 1
400            if (next_num > 4):
401                next_num = 1
402
403            return (hai.kind, next_num)
404
405        # 白發中
406        elif 5 <= hai.num <= 7:
407            next_num = hai.num + 1
408            if (next_num > 7):
409                next_num = 5
410
411            return (hai.kind, next_num)
412
413    # 局を表す文字列
414    def kyoku_name(self):
415        KAZE_NAME = ["東", "南", "西", "北"]
416        return "{}{}局".format(KAZE_NAME[self.bakaze], self.kyoku + 1)
417
418    # 自風
419    def jikaze(self, player):
420        return (player.chicha - self.kyoku) % self.player_num
421
422    # 配牌
423    def haipai(self, player):
424        player.tehai.append([self.yama.pop() for i in range(13)])
425        player.tehai.sort()
426        self.update_game_reader()
427
428    # ツモ
429    def tsumo_pop(self, player):
430        player.tehai.tsumo(self.yama.pop())
431        self.update_game_reader()
432
433    # 打牌
434    def dahai(self, player, index):
435        tsumogiri = (index == -1 or index == player.tehai.hai_num - 1)
436        player.kawa.append(player.tehai.pop(index), tsumogiri)
437        player.tehai.sort()
438
439        player.ippatsu = False
440        player.rinshan = False
441
442        self.update_game_reader()
443        return player.kawa.hais[-1]
444
445    # リーチ
446    def richi(self, player):
```

```python
447                player.richi = True
448                player.ippatsu = True
449                player.kawa.hais[-1].richi = True
450                self.update_game_reader()
451
452        # ツモ
453        def tsumo(self, player):
454            yaku = self.yaku(player, True)
455
456            # 点数移動
457            for cur_player in self.players:
458                if cur_player == player:
459                    continue
460
461                change_point = yaku.cost["main" if self.jikaze(cur_player) == 0 else "
       additional"] + self.honba * 100
462                player.point += change_point
463                cur_player.point -= change_point
464
465                # リーチ棒
466                if cur_player.richi:
467                    player.point += 1000
468                    cur_player.point -= 1000
469
470            # 供託
471            player.point += self.kyotaku * 1000
472
473            # 表示
474            print("{}:ツモ".format(player.name))
475            print(yaku.yaku)
476            print("{}翻 {}符".format(yaku.han, yaku.fu))
477
478            self.update_game_reader([player], player.richi)
479
480        # ロン
481        def ron(self, player, target, whose, chankan = False):
482            if chankan:
483                hai = target
484            else:
485                hai = target.hai
486                target.hoju = True
487
488            player.tehai.tsumo(hai)
489            yaku = self.yaku(player, False, hai, chankan)
490
491            # 点数移動
492            change_point = yaku.cost["main"] + self.honba * 300
493            player.point += change_point
494            whose.point -= change_point
495
496            # リーチ棒
497            for cur_player in self.players:
498                if cur_player != player and cur_player.richi:
499                    player.point += 1000
500                    cur_player.point -= 1000
501
```

```python
            # 供託
            player.point += self.kyotaku * 1000

            # 表示
            print("{}→{}:ロン".format(whose.name, player.name))
            print(yaku.yaku)
            print("{}翻 {}符".format(yaku.han, yaku.fu))

            self.update_game_reader([player], player.richi)

    # 暗槓
    def ankan(self, player, hais):
        player.tehai.ankan(hais)
        player.rinshan = True

        self.add_dora()

        for player in self.players:
            player.ippatsu = False

        self.update_game_reader()

    # 加槓
    def kakan(self, player, hai):
        player.tehai.kakan(hai)
        player.rinshan = True

        self.add_dora()

        for player in self.players:
            player.ippatsu = False

        self.update_game_reader()

    # 明槓
    def minkan(self, player, hais, target, whose):
        player.tehai.minkan(hais, target.hai, player.relative(whose))
        player.furo = True
        player.rinshan = True
        target.furo = True

        self.change_player(player.chicha)
        self.add_dora()

        for player in self.players:
            player.ippatsu = False

        self.update_game_reader()

    # ポン
    def pon(self, player, hais, target, whose):
        player.tehai.pon(hais, target.hai, player.relative(whose))
        player.furo = True
        target.furo = True

        self.change_player(player.chicha)
```

```python
558
559            for player in self.players:
560                player.ippatsu = False
561
562            self.update_game_reader()
563
564        # チー
565        def chi(self, player, hais, target, whose):
566            player.tehai.chi(hais, target.hai, player.relative(whose))
567            player.furo = True
568            target.furo = True
569
570            self.change_player(player.chicha)
571
572            for player in self.players:
573                player.ippatsu = False
574
575            self.update_game_reader()
576
577        # 流局
578        def ryukyoku(self):
579            open_players = []
580
581            tenpai_cnt = 0
582
583            for player in self.players:
584                if player.tehai.shanten() == 0:
585                    tenpai_cnt += 1
586
587            for player in self.players:
588                # 点数移動
589                if tenpai_cnt != 0 and tenpai_cnt != self.player_num:
590                    if player.tehai.shanten() == 0:
591                        player.point += int(3000 / tenpai_cnt)
592                    else:
593                        player.point -= int(3000 / (self.player_num - tenpai_cnt))
594
595                # 供託
596                if player.richi:
597                    player.point -= 1000
598
599                if player.tehai.shanten() == 0:
600                    print("{}:テンパイ".format(player.name))
601                    open_players.append(player)
602                else:
603                    print("{}:ノーテン".format(player.name))
604
605            self.update_game_reader(open_players)
606
607        # ツモチェック
608        def check_tsumo(self, player):
609            if player.tehai.shanten() != -1:
610                return False
611
612            yaku = self.yaku(player, True)
613            return yaku.cost is not None
```

```
614
615        # ロンチェック
616        def check_ron(self, player, target, whose):
617            player.tehai.tsumo(target)
618            cur_shanten = player.tehai.shanten()
619            player.tehai.pop()
620
621            if cur_shanten != -1:
622                return False
623
624            yaku = self.yaku(player, False, target)
625            return yaku.cost is not None
626
627        # 暗槓チェック
628        def check_ankan(self, player):
629            return player.tehai.ankan_able()
630
631        # 加槓チェック
632        def check_kakan(self, player):
633            return player.tehai.kakan_able()
634
635        # 明槓チェック
636        def check_minkan(self, player, target):
637            if player.richi:
638                return []
639
640            return player.tehai.minkan_able(target)
641
642        # ポンチェック
643        def check_pon(self, player, target):
644            if player.richi:
645                return []
646
647            return player.tehai.pon_able(target)
648
649        # チーチェック
650        def check_chi(self, player, target):
651            if player.richi:
652                return []
653
654            return player.tehai.chi_able(target)
655
656        # 選択
657        def on_dahai(self, player):
658            return player.on_dahai()
659
660        # 立直するか
661        def on_richi(self, player):
662            return player.on_richi()
663
664        # ツモ和了するか
665        def on_tsumo(self, player):
666            return player.on_tsumo()
667
668        # ロン和了するか
669        def on_ron(self, player, target, whose):
```

```
670             return player.on_ron(target, whose.chicha)
671
672        # 暗槓するか
673        def on_ankan(self, player, target):
674             return player.on_ankan(target)
675
676        # 明槓するか
677        def on_minkan(self, player, hais, target, whose):
678             return player.on_minkan(hais, target, whose.chicha)
679
680        # 加槓するか
681        def on_kakan(self, player, target):
682             return player.on_kakan(target)
683
684        # ポンするか
685        def on_pon(self, player, hais, target, whose):
686             return player.on_pon(hais, target, whose.chicha)
687
688        # チーするか
689        def on_chi(self, player, hais, target, whose):
690             return player.on_chi(hais, target, whose.chicha)
691
692  # 山
693  class Yama():
694      def __init__(self, hai_set):
695          self.hais = hai_set[:]
696          random.shuffle(self.hais)
697          self.remain = len(self.hais) - 14
698          self.doras = []
699          self.uradoras = []
700          self.add_dora()
701
702      # 取り出し
703      def pop(self):
704          self.remain -= 1
705          return self.hais.pop()
706
707      # ドラを増やす
708      def add_dora(self):
709          self.doras.append(self.hais[len(self.doras) * 2])
710          self.uradoras.append(self.hais[len(self.uradoras) * 2 + 1])
```

リスト 7　mjgame/system/player.py

```
1  from abc import ABCMeta, abstractmethod
2  from .core import *
3  from .tehai import Tehai
4
5  # プレイヤー
6  class Player(metaclass=ABCMeta):
7      def __init__(self):
8          self.name = ""
9          self.reset()
10
11     # 下家・対面・上家
12     def relative(self, other):
```

```python
13             return (other.chicha - self.chicha) % 4
14
15     # 初期設定
16     def setup(self, game, chicha, start_point):
17         self.game = game
18         self.chicha = chicha
19         self.point = start_point
20
21     # 手牌・河をリセット
22     def reset(self):
23         self.tehai = Tehai()
24         self.kawa = Kawa()
25         self.richi = False
26         self.furo = False
27         self.ippatsu = False
28         self.rinshan = False
29
30     # 選択
31     @abstractmethod
32     def on_dahai(self):
33         pass
34
35     # 立直するか
36     @abstractmethod
37     def on_richi(self):
38         pass
39
40     # ツモ和了するか
41     @abstractmethod
42     def on_tsumo(self):
43         pass
44
45     # ロン和了するか
46     @abstractmethod
47     def on_ron(self, target, whose):
48         pass
49
50     # 暗槓するか
51     @abstractmethod
52     def on_ankan(self, target):
53         pass
54
55     # 明槓するか
56     @abstractmethod
57     def on_minkan(self, hais, target, whose):
58         pass
59
60     # 加槓するか
61     @abstractmethod
62     def on_kakan(self, target):
63         pass
64
65     # ポンするか
66     @abstractmethod
67     def on_pon(self, hais, target, whose):
68         pass
```

```
69
70      # チーするか
71      @abstractmethod
72      def on_chi(self, hais, target, whose):
73          pass
```

リスト 8   mjgame/system/tehai.py

```
 1  import copy
 2  import itertools
 3  import collections
 4  from .core import *
 5
 6  from mahjong.shanten import Shanten
 7  from mahjong.tile import TilesConverter
 8
 9  # 手牌
10  class Tehai():
11      def __init__(self, hais=[], furos=[]):
12          self.calculator = Shanten()
13          self.hais = hais[:]
14          self.furos = furos[:]
15          self.tsumo_hai = None
16          self.hai_num = len(self.hais)
17
18      # ツモ牌を手牌に格納
19      def store(self):
20          if self.tsumo_hai is not None:
21              self.hais.append(self.tsumo_hai)
22              self.tsumo_hai = None
23
24      # ツモ
25      def tsumo(self, hai):
26          self.store()
27          self.tsumo_hai = hai
28          self.hai_num = len(self.hais) + 1
29
30      # 追加
31      def append(self, hais):
32          self.store()
33          for hai in hais:
34              self.hais.append(hai)
35          self.hai_num = len(self.hais)
36
37      # 挿入
38      def insert(self, index, hai):
39          self.store()
40          self.hais.insert(index, hai)
41          self.hai_num = len(self.hais)
42
43      # 番号で取り出し
44      def pop(self, index=-1):
45          self.store()
46          pop_hai = self.hais.pop(index)
47          self.hai_num = len(self.hais)
48          return pop_hai
```

```
49
50      # 牌を指定して取り出し
51      def remove(self, hai):
52          self.store()
53          self.hais.remove(hai)
54          self.hai_num = len(self.hais)
55          return hai
56
57      # 牌の種類を指定して検索
58      def find(self, kind, num):
59          for hai in self.hais + [self.tsumo_hai]:
60              if hai is not None and hai.kind == kind and hai.num == num:
61                  return hai
62          return None
63
64      # 複数の牌を検索
65      def find_multi(self, kinds, nums):
66          if len(kinds) != len(nums):
67              return None
68
69          find_num = len(kinds)
70          found_hais = []
71
72          for hai in self.hais + [self.tsumo_hai]:
73              if hai is None:
74                  continue
75
76              for kind, num in zip(kinds, nums):
77                  if hai.kind == kind and hai.num == num:
78                      found_hais.append(hai)
79                      kinds.remove(kind)
80                      nums.remove(num)
81                      break
82
83          if len(found_hais) == find_num:
84              return found_hais
85          else:
86              return None
87
88      # 並べ替え
89      def sort(self):
90          self.store()
91          self.hais.sort()
92
93      # 暗槓可能な牌
94      def ankan_able(self):
95          return_hais = []
96
97          for hai in self.hais + [self.tsumo_hai]:
98              for return_hai in return_hais:
99                  if hai.kind == return_hai[0].kind and hai.num == return_hai[0].num:
100                     break
101             else:
102                 found_hais = self.find_multi([hai.kind for i in range(4)], [hai.num for i
    in range(4)])
103
```

```
104                 if found_hais is not None:
105                     return_hais.append(found_hais)
106
107         return return_hais
108
109     # 加槓可能な牌
110     def kakan_able(self):
111         return_hais = []
112
113         for furo in self.furos:
114             if furo.kind == FuroKind.PON:
115                 for hai in self.hais + [self.tsumo_hai]:
116                     # 明刻と同じ牌だったら
117                     if furo.hais[0].kind == hai.kind and furo.hais[0].num == hai.num:
118                         return_hais.append(hai)
119                         break
120
121         return return_hais
122
123     # 明槓可能な牌
124     def minkan_able(self, target):
125         found_hais = self.find_multi([target.kind for i in range(3)], [target.num for i in
    range(3)])
126
127         if found_hais is None:
128             return []
129         else:
130             return [found_hais]
131
132     # ポン可能な牌
133     def pon_able(self, target):
134         found_hais = self.find_multi([target.kind for i in range(2)], [target.num for i in
    range(2)])
135
136         if found_hais is None:
137             return []
138         else:
139             return [found_hais]
140
141     # チー可能な牌
142     def chi_able(self, target):
143         if target.kind == 3:
144             return []
145
146         return_hais = []
147
148         for i in range(-2, 1):
149             kinds = []
150             nums = []
151
152             for j in range(i, i + 3):
153                 if j == 0:
154                     continue
155
156                 kinds.append(target.kind)
157                 nums.append(target.num + j)
```

```
158
159              found_hais = self.find_multi(kinds, nums)
160
161              if found_hais is not None:
162                  return_hais.append(found_hais)
163
164          return return_hais
165
166      # 暗槓
167      def ankan(self, hais):
168          append_hais = [self.remove(hai) for hai in hais]
169          self.furos.append(Furo(append_hais, FuroKind.ANKAN))
170          self.sort()
171
172      # 加槓
173      def kakan(self, hai):
174          for furo in self.furos:
175              if furo.kind == FuroKind.PON:
176                  # 明刻と同じ牌だったら
177                  if furo.hais[0].kind == hai.kind and furo.hais[0].num == hai.num:
178                      furo.kind = FuroKind.KAKAN
179                      furo.hais.append(self.remove(hai))
180
181      # 明槓
182      def minkan(self, hais, target, direct):
183          append_hais = [self.remove(hai) for hai in hais]
184          append_hais.insert(int((direct - 1) * 1.5), target)
185          self.furos.append(Furo(append_hais, FuroKind.MINKAN, direct))
186
187      # ポン
188      def pon(self, hais, target, direct):
189          append_hais = [self.remove(hai) for hai in hais]
190          append_hais.insert(direct - 1, target)
191          self.furos.append(Furo(append_hais, FuroKind.PON, direct))
192
193      # チー
194      def chi(self, hais, target, direct):
195          append_hais = [self.remove(hai) for hai in hais]
196          append_hais.insert(direct - 1, target)
197          self.furos.append(Furo(append_hais, FuroKind.CHI, direct))
198
199      # 表示
200      def show(self):
201          for hai in self.hais:
202              print(format(hai.name, "<4s"), end="")
203
204          if self.tsumo_hai is not None:
205              print(" {}".format(self.tsumo_hai.name))
206          else:
207              print()
208
209          for j in range(len(self.hais)):
210              print(format(j, "<4d"), end="")
211
212          if self.tsumo_hai is not None:
213              print(" {}".format(j + 1))
```

```
214        else:
215            print()
216
217    # シャンテン数
218    def shanten(self):
219        tile_strs = [""] * 4
220
221        for hai in self.hais:
222            tile_strs[hai.kind] += str(hai.num)
223
224        if self.tsumo_hai is not None:
225            tile_strs[self.tsumo_hai.kind] += str(self.tsumo_hai.num)
226
227        tiles = TilesConverter.string_to_34_array(tile_strs[0], tile_strs[1], tile_strs[2],
       tile_strs[3])
228
229        return self.calculator.calculate_shanten(tiles)
```

リスト 9　mjgame/players/__init__.py

```
1  from .human import *
2  from .ai import *
```

リスト 10　mjgame/players/human.py

```
1   import sys
2   from .. import system as mj
3
4   # 人間
5   class Human(mj.Player):
6       def __init__(self):
7           super().__init__()
8           self.name = "Human"
9
10      # 確認メッセージを表示
11      def confirm(self, message, default=True):
12          while True:
13              select_input = input("{} [{}]> ".format(message, "Y/n" if default else "y/N"))
14
15              if select_input == "":
16                  return default
17
18              elif select_input == "Y" or select_input == "y":
19                  return True
20
21              elif select_input == "N" or select_input == "n":
22                  return False
23
24      # 選択
25      def on_dahai(self):
26          # 入力
27          while True:
28              select_input = input("> ")
29
30              if select_input == "q":
31                  sys.exit()
```

```
32
33                  # ツモ切り
34              elif select_input == "":
35                  return -1
36
37              elif select_input.isdecimal() and 0 <= int(select_input) <= self.tehai.hai_num:
38                  break
39
40          return int(select_input)
41
42      # 立直
43      def on_richi(self):
44          return self.confirm("立直する？", False)
45
46      # ツモ和了
47      def on_tsumo(self):
48          return self.confirm("ツモる？", True)
49
50      # ロン和了
51      def on_ron(self, target, whose):
52          return self.confirm("ロンする？", True)
53
54      # 暗槓
55      def on_ankan(self, target):
56          return self.confirm("暗槓する？", False)
57
58      # 明槓
59      def on_minkan(self, hais, target, whose):
60          return self.confirm("明槓する？", False)
61
62      # 加槓
63      def on_kakan(self, target):
64          return self.confirm("加槓する？", False)
65
66      # ポン
67      def on_pon(self, hais, target, whose):
68          return self.confirm("ポンする？", False)
69
70      # チー
71      def on_chi(self, hais, target, whose):
72          return self.confirm("チーする？", False)
```

リスト 11　mjgame/players/ai.py

```
1   import copy
2   from .. import player
3
4   # 手なりAI
5   class Tenari(player.Player):
6       def __init__(self):
7           super().__init__()
8           self.name = "Tenari"
9
10      # 選択
11      def on_dahai(self):
12          select_index = -1
```

```
13              effect_max = 0
14              cur_shanten = self.tehai.shanten()
15
16              # 残っている牌
17              remain_hai = self.game.hai_set[:]
18
19              # 自身の手牌
20              for hai in self.tehai.hais:
21                  remain_hai.remove(hai)
22
23              for player in self.game.players:
24                  # 副露
25                  for furo in player.tehai.furos:
26                      for hai in furo.hais:
27                          remain_hai.remove(hai)
28
29                  # 河
30                  for kawa_hai in player.kawa.hais:
31                      if not kawa_hai.furo:
32                          remain_hai.remove(kawa_hai.hai)
33
34              # ドラ
35              for dora in self.game.dora_hais:
36                  remain_hai.remove(dora)
37
38              for i in range(self.tehai.hai_num):
39                  # 1枚ずつ切ってみる
40                  pop_hai = self.tehai.pop(i)
41
42                  # シャンテン数が進むなら
43                  if self.tehai.shanten() <= cur_shanten:
44                      effect_count = 0
45
46                      # 残り全ての牌をツモってみる
47                      for hai in remain_hai:
48                          # 有効牌の数をカウント
49                          self.tehai.tsumo(hai)
50                          if self.tehai.shanten() < cur_shanten:
51                              effect_count += 1
52                          self.tehai.pop()
53
54                      # 有効牌が一番多い牌を選択
55                      if effect_count >= effect_max:
56                          effect_max = effect_count
57                          select_index = i
58
59                  self.tehai.insert(i, pop_hai)
60
61              return select_index
62
63          # 立直するか
64          def on_richi(self):
65              return True
66
67          # ツモ和了するか
68          def on_tsumo(self):
```

```
 69            return True
 70
 71        # ロン和了するか
 72        def on_ron(self, target, whose):
 73            return True
 74
 75        # 暗槓するか
 76        def on_ankan(self, target):
 77            # シャンテン数が下がらないなら暗槓
 78            temp_tehai = copy.deepcopy(self.tehai)
 79            temp_tehai.ankan(target)
 80            return temp_tehai.shanten() <= self.tehai.shanten()
 81
 82        # 明槓するか
 83        def on_minkan(self, hais, target, whose):
 84            # 門前だったら明槓しない
 85            if not self.furo:
 86                return False
 87
 88            # シャンテン数が下がらないなら明槓
 89            temp_tehai = copy.deepcopy(self.tehai)
 90            temp_tehai.minkan(hais, target, 1)
 91            return temp_tehai.shanten() <= self.tehai.shanten()
 92
 93        # 加槓するか
 94        def on_kakan(self, target):
 95            # シャンテン数が下がらないなら加槓
 96            temp_tehai = copy.deepcopy(self.tehai)
 97            temp_tehai.kakan(target)
 98            return temp_tehai.shanten() <= self.tehai.shanten()
 99
100        # ポンするか
101        def on_pon(self, hais, target, whose):
102            # シャンテン数が進むならポン
103            temp_tehai = copy.deepcopy(self.tehai)
104            temp_tehai.pon(hais, target, 1)
105            return temp_tehai.shanten() < self.tehai.shanten()
106
107        # チーするか
108        def on_chi(self, hais, target, whose):
109            # シャンテン数が進むならチー
110            temp_tehai = copy.deepcopy(self.tehai)
111            temp_tehai.chi(hais, target, 1)
112            return temp_tehai.shanten() < self.tehai.shanten()
113
114    # 面子手のみ
115    class AI(Tenari):
116        def __init__(self):
117            super().__init__()
118            self.name = "AI"
119
120        def on_minkan(self, hais, target, whose):
121            return False
122
123        def on_pon(self, hais, target, whose):
124            return False
```

```
125
126       def on_chi(self, hais, target, whose):
127           return False
```

リスト 12　mjgame/graphic/__init__.py

```
1   from .graphic import *
```

リスト 13　mjgame/graphic/graphic.py

```
1   import os
2   import sys
3   import time
4   import glob
5   from PIL import Image, ImageDraw, ImageFont, ImageTk
6   import numpy as np
7   import cv2
8   from .. import system as mj
9
10  # 麻雀牌のサイズ
11  MJHAI_WIDTH = 30
12  MJHAI_HEIGHT = 38
13  SCREEN_SIZE = 12 * MJHAI_HEIGHT + 7 * MJHAI_WIDTH
14
15  # フォントファイル
16  THIS_PATH = os.path.dirname(os.path.abspath(__file__))
17  FONT_FILE = THIS_PATH + "/font/YuGothB.ttc"
18
19  KAZE_NAME = ["東", "南", "西", "北"]
20
21  # 画像ファイル読み込み
22  hai_img = {}
23  mjhai_files = glob.glob(THIS_PATH + "/mjhai/*.png") # ファイル一覧を取得
24
25  for mjhai_file in mjhai_files:
26      img_key, ext = os.path.splitext(os.path.basename(mjhai_file)) # ファイル名を抽出
27      hai_img[img_key] = Image.open(mjhai_file)
28
29  t100_img = Image.open(THIS_PATH + "/image/100.png")
30  t1000_img = Image.open(THIS_PATH + "/image/1000.png")
31  remain_img = Image.open(THIS_PATH + "/image/remain.png")
32
33  # Pillow→OpenCV変換
34  def pil2cv(image):
35      new_image = np.array(image)
36      if new_image.shape[2] == 3: # カラー
37          new_image = new_image[:, :, ::-1]
38      elif new_image.shape[2] == 4: # 透過
39          new_image = new_image[:, :, [2, 1, 0, 3]]
40      return new_image
41
42  # 横向きの画像を生成
43  def draw_side(img):
44      w, h = img.size
45      create_img = Image.new("RGBA", (h, h))
46
```

```python
47          rotate_img = img.rotate(90, expand=True)
48          create_img.paste(rotate_img, (0, h - w))
49
50          return create_img
51
52  # 単色を合成した画像を生成
53  def draw_mix(img, color):
54      create_img = Image.new("RGB", img.size)
55      mix_img = Image.new("RGBA", img.size, color)
56
57      create_img.paste(img)
58      create_img.paste(mix_img, (0, 0), mix_img)
59      return create_img
60
61  # 手牌の画像を生成
62  def draw_tehai(tehai, back=False):
63      create_img = Image.new("RGBA", (14 * MJHAI_WIDTH + 4 * MJHAI_HEIGHT, 2 * MJHAI_HEIGHT))
64      img_draw = ImageDraw.Draw(create_img)
65      img_draw.font = ImageFont.truetype(FONT_FILE, 12)
66
67      x = 0
68      for i, hai in enumerate(tehai.hais):
69          # 番号
70          if not back:
71              w, h = img_draw.textsize(str(i))
72              img_draw.text(
73                  (x + (MJHAI_WIDTH - w) / 2, MJHAI_HEIGHT - h - 4),
74                  str(i)
75              )
76
77          # 麻雀牌
78          mjhai_draw = hai_img["back" if back else hai.name]
79          create_img.paste(mjhai_draw, (x, MJHAI_HEIGHT))
80          x += MJHAI_WIDTH
81
82      if tehai.tsumo_hai is not None:
83          # ツモった牌は離す
84          x += int(MJHAI_WIDTH / 4)
85
86          # 番号
87          if not back:
88              w, h = img_draw.textsize(str(i + 1))
89              img_draw.text(
90                  (x + (MJHAI_WIDTH - w) / 2, MJHAI_HEIGHT - h - 4),
91                  str(i + 1)
92              )
93
94          # 麻雀牌
95          mjhai_draw = hai_img["back" if back else tehai.tsumo_hai.name]
96
97          create_img.paste(mjhai_draw, (x, MJHAI_HEIGHT))
98          x += MJHAI_HEIGHT
99
100     # 副露
101     x = create_img.size[0]
102     for furo in tehai.furos:
```

```python
103             for i, hai in enumerate(furo.hais):
104                 # 麻雀牌
105                 if furo.kind == mj.FuroKind.ANKAN and (i == 0 or i == 3):
106                     mjhai_draw = hai_img["back"]
107                 else:
108                     mjhai_draw = hai_img[hai.name]
109
110                 if furo.kind == mj.FuroKind.KAKAN and i == 3:
111                     # 加槓
112                     create_img.paste(
113                         draw_side(mjhai_draw),
114                         (x + (3 - furo.direct) * MJHAI_WIDTH, MJHAI_HEIGHT - MJHAI_WIDTH)
115                     )
116                 else:
117                     # 他家からの牌は横にする
118                     if furo.kind != mj.FuroKind.ANKAN and furo.direct == i + 1:
119                         create_img.paste(draw_side(mjhai_draw), (x - MJHAI_HEIGHT, MJHAI_HEIGHT
    ))
120                         x -= MJHAI_HEIGHT
121                     else:
122                         create_img.paste(mjhai_draw, (x - MJHAI_WIDTH, MJHAI_HEIGHT))
123                         x -= MJHAI_WIDTH
124
125         return create_img
126
127 # 河の画像を生成
128 def draw_kawa(kawa):
129     create_img = Image.new("RGBA", (5 * MJHAI_WIDTH + MJHAI_HEIGHT, 4 * MJHAI_HEIGHT))
130
131     x = 0
132     y = 0
133
134     for kawa_hai in kawa.hais:
135
136         paste_img = Image.new("RGB", (MJHAI_WIDTH, MJHAI_HEIGHT))
137         paste_img.paste(hai_img[kawa_hai.hai.name])
138
139         # ツモ切りは暗くする
140         if kawa_hai.tsumogiri:
141             paste_img = draw_mix(paste_img, (0, 0, 0, 47))
142
143         # 鳴かれた牌は青くする
144         if kawa_hai.furo:
145             paste_img = draw_mix(paste_img, (0, 63, 255, 47))
146
147         # 放銃した牌は赤くする
148         if kawa_hai.hoju:
149             paste_img = draw_mix(paste_img, (255, 63, 0, 47))
150
151         # 立直宣言牌は横にする
152         if kawa_hai.richi:
153             already_richi = True
154             create_img.paste(draw_side(paste_img), (x, y))
155             x += MJHAI_HEIGHT
156         else:
157             create_img.paste(paste_img, (x, y))
```

```
158              x += MJHAI_WIDTH
159
160          # 6枚で改行
161          if x >= 6 * MJHAI_WIDTH:
162              x = 0
163              y += MJHAI_HEIGHT
164
165      return create_img
166
167  # ゲーム情報
168  def draw_info(game):
169      size = 5 * MJHAI_WIDTH
170      create_img = Image.new("RGBA", (size, size))
171
172      # 局
173      kyoku_name = "{}{}局".format(KAZE_NAME[game.bakaze], game.kyoku + 1)
174
175      img_draw = ImageDraw.Draw(create_img)
176      img_draw.font = ImageFont.truetype(FONT_FILE, 20)
177      w, h = img_draw.textsize(kyoku_name)
178      img_draw.text(((size - w) / 2, 0), kyoku_name)
179
180      # 本場
181      img_draw.font = ImageFont.truetype(FONT_FILE, 16)
182      w, h = img_draw.textsize("× {}".format(game.honba))
183      img_draw.text(
184          (t100_img.size[0] + 2, MJHAI_WIDTH + (t100_img.size[1] - h) / 2),
185          "× {}".format(game.honba)
186      )
187
188      create_img.paste(t100_img, (2, MJHAI_WIDTH))
189
190      # 供託
191      w, h = img_draw.textsize("× {}".format(game.kyotaku))
192      img_draw.text(
193          (size - w - 2, MJHAI_WIDTH + (t1000_img.size[1] - h) / 2),
194          "× {}".format(game.kyotaku)
195      )
196
197      create_img.paste(t1000_img, (size - t1000_img.size[0] - w - 2, MJHAI_WIDTH))
198
199      # 残り
200      w, h = img_draw.textsize("× {:02}".format(game.yama_remain))
201      img_draw.text(
202          ((size + remain_img.size[0] - w) / 2, MJHAI_WIDTH + (remain_img.size[1] - h) / 2 +
203  20),
203          "× {:02}".format(game.yama_remain)
204      )
205
206      create_img.paste(
207          remain_img,
208          (int((size - remain_img.size[0] - w) / 2), MJHAI_WIDTH + 20)
209      )
210
211      return create_img
212
```

```python
213  # ドラ
214  def draw_dora(doras, uradoras):
215      create_img = Image.new("RGBA", (5 * MJHAI_WIDTH, 2 * MJHAI_HEIGHT))
216
217      for i in range(5):
218          # ドラ
219          if i < len(doras):
220              paste_img = hai_img[doras[i].name]
221          else:
222              paste_img = hai_img["back"]
223
224          create_img.paste(paste_img, (i * MJHAI_WIDTH, 0))
225
226          # 裏ドラ
227          if uradoras is not None and i < len(uradoras):
228              paste_img = hai_img[uradoras[i].name]
229          else:
230              paste_img = hai_img["back"]
231
232          create_img.paste(paste_img, (i * MJHAI_WIDTH, MJHAI_HEIGHT))
233
234      return create_img
235
236  # ゲーム画面の画像を生成
237  def draw_screen(game, players, view):
238      create_img = Image.new("RGB", (SCREEN_SIZE, SCREEN_SIZE), "green")
239
240      for i in range(game.player_num):
241          game_player = game.players[i]
242          player = None
243
244          for cur_player in players:
245              if game_player.chicha == cur_player.chicha:
246                  player = cur_player
247
248          paste_img = Image.new("RGBA", (SCREEN_SIZE, SCREEN_SIZE))
249
250          # 手牌
251          if player is None:
252              if game_player.tehai.hais is None:
253                  tehai = mj.Tehai([mj.Hai(0, 0) for j in range(game_player.tehai.hai_num -
     1)], game_player.tehai.furos)
254                  tehai.tsumo(mj.Hai(0, 0))
255
256                  if not game_player.tehai.exist_tsumo_hai:
257                      tehai.store()
258
259                  tehai_img = draw_tehai(tehai, True)
260              else:
261                  tehai = mj.Tehai([hai for hai in game_player.tehai.hais], game_player.tehai
     .furos)
262                  tehai.tsumo(game_player.tehai.tsumo_hai)
263                  tehai_img = draw_tehai(tehai)
264          else:
265              tehai_img = draw_tehai(player.tehai)
266
```

```
267              paste_img.paste(
268                  tehai_img,
269                  (SCREEN_SIZE - tehai_img.size[0], 7 * MJHAI_WIDTH + 10 * MJHAI_HEIGHT),
270                  tehai_img
271              )
272
273              # 河
274              kawa_img = draw_kawa(game_player.kawa)
275              paste_img.paste(
276                  kawa_img,
277                  (6 * MJHAI_HEIGHT + int(0.5 * MJHAI_WIDTH), 7 * MJHAI_WIDTH + 6 * MJHAI_HEIGHT)
     ,
278                  kawa_img
279              )
280
281              # 自風&点数
282              img_draw = ImageDraw.Draw(paste_img)
283              img_draw.font = ImageFont.truetype(FONT_FILE, 16)
284              jikaze = game_player.chicha - game.kyoku % game.player_num
285              w, h = img_draw.textsize("[{}] {}".format(KAZE_NAME[jikaze], game_player.point))
286
287              img_draw.text(
288                  ((SCREEN_SIZE - w) / 2, 6.5 * MJHAI_WIDTH + 6 * MJHAI_HEIGHT - h / 2),
289                  "[{}] {}".format(KAZE_NAME[jikaze], game_player.point),
290                  (255, 255, 0)
291              )
292
293              # 番
294              if game_player == game.cur_player:
295                  x = (SCREEN_SIZE - w) / 2 - 16
296                  y = 6.5 * MJHAI_WIDTH + 6 * MJHAI_HEIGHT - 5
297                  img_draw.rectangle((x, y, x + 10, y + 10), (255, 255, 127), (0, 0, 0))
298
299              # プレイヤー名&シャンテン数
300              draw_str = game_player.name
301              if player is not None:
302                  draw_str += " [{}ST]".format(player.tehai.shanten())
303
304              w, h = img_draw.textsize(draw_str)
305              img_draw.text(
306                  (SCREEN_SIZE - tehai_img.size[0], SCREEN_SIZE - tehai_img.size[1]),
307                  draw_str,
308              )
309
310              # 回転&合成
311              rotate_img = paste_img.rotate((game_player.chicha - view.chicha) * 90)
312              create_img.paste(rotate_img, (0, 0), rotate_img)
313
314      # ゲーム情報
315      info_img = draw_info(game)
316      create_img.paste(
317          info_img,
318          (6 * MJHAI_HEIGHT + MJHAI_WIDTH, 6 * MJHAI_HEIGHT + MJHAI_WIDTH),
319          info_img
320      )
321
```

```python
322        # ドラ
323        dora_img = draw_dora(game.dora_hais, game.uradora_hais)
324        create_img.paste(
325            dora_img,
326            (6 * MJHAI_HEIGHT + MJHAI_WIDTH, 4 * MJHAI_HEIGHT + 6 * MJHAI_WIDTH),
327            dora_img
328        )
329
330        return create_img
331
332  class Screen():
333      def __init__(self, game, players, view=None, title=""):
334          self.game = game
335          self.players = players
336          self.view = view
337          self.title = title
338
339          # ウィンドウ名
340          self.win_name = self.title
341          if view is not None:
342              self.win_name += " [" + view.name + "]"
343
344      # 画面描画
345      def draw(self):
346          cur_view = self.game.cur_player if self.view is None else self.view
347          img = pil2cv(draw_screen(self.game, self.players, cur_view))
348          cv2.imshow(self.win_name, img)
349
350          if cv2.waitKey(1) & 0xff == ord("q"):
351              sys.exit()
```