Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Developing a Jass AI Server

Bachelor's Thesis

Jonas Althaus

jalthaus@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Oliver Richter
Prof. Dr. Roger Wattenhofer

March 18, 2021

Acknowledgements

I would like to thank my supervisor Oliver Richter for guiding me throughout this project and always giving me valuable feedback and ideas.

Next, I thank Noël Rimensberger for developing the Jass AI and Niklaus Kappeler for enabling HTTPS on our WebApp.

Lastly, I want to thank Schweizer Jassverzeichnis[1] for publishing our project on their blog as well as all the players for having played over 130 games of Jass.

[1]https://www.jassverzeichnis.ch

Abstract

To evaluate various self-play reinforcement learning agents in the domain of Jass, we have built a web application in which players can compete against each other and the Artificial Intelligence. Since the outcome of a stochastic trick-taking game such as Jass depends on the distribution of the cards, we propose a new game mode which eliminates the luck factor and provides a fair game dynamic. Analysing those fair games enables us to compare different agents with human players under equal conditions. Our results show one agent playing on a par with human Jass players, frequently even outperforming them. Furthermore, we demonstrate the fairness of the new game mode by showing that fair games are more balanced than regular Jass games and thus lead to a more competitive game environment.

Contents

Chapter 1

Introduction

Schieber Jass is a traditional trick-taking Swiss card game which is played by two teams of two players each. As a challenge for game AI research, Noël Rimensberger has developed four different Jass bots using self-play reinforcement learning [1]. In order to ensure accurate evaluation of each agent, it is necessary to analyse more of their games against human Jass players. Therefore, we developed a web application (WebApp) in which people from all over the world can play a game of Jass against each other and the AI.

Even though Jass is not considered as gambling in Switzerland [2], being a stochastic game, it contains a relevant luck factor. The outcome of a game depends on the random distribution of the starting hands, as not every hand is equally strong. In order to mitigate that luck factor, we propose a new fair game mode that provides equal conditions for each team, i.e. every starting hand will be played twice, once for each team. Consequently, we can analyse the agents when starting with equal hands and compare their plays against one another and to the choices made by human players. Furthermore, we address the effects and potential issues of the fair game mode and carry out a user study to count in the experiences made by the players themselves.

Our main goal of this project is to generate interest among Jass players to test our WebApp and thus play alongside and/or against the AI. Therefore, the primary objective is to develop a bug-free server and a capable user interface [3]. Additionally, we provide various features to enhance the overall experience, including a replay mode that allows users to revisit their best games, analyse their mistakes and potentially discover better plays. Revisiting previously played games is a feature exclusively available on our WebApp and not found on other Jass websites[1].

[1] www.schieber.ch, www.jasse.ch, www.swisslos.ch/de/jass/schieber/spielen.html and www.jassfederal.ch

Chapter 2

Background

In this chapter, we explain relevant information for the rest of the thesis.

2.1 Jass Terminology

In a Schieber Jass *game*, two opposing teams of two players each play against one another. First, a 36 card deck is randomly shuffled and evenly distributed amongst the four players, giving every player their starting *hand*. Then the first *round* begins. The starting player makes a *trump call* by either choosing one of the suits (Roses, Shields, Acorns, Bells), or one of the trumps *bottom-up* or *top-down*. Alternatively, s/he may let the partner decide for him/her (*to shove*), hence the name Schieber (shoving in German). Then s/he plays the first card, followed by the other players in counter-clockwise direction. They must always play the same suit or a trump card. If they can't, they may discard any card into play. Then, the player who played the highest card or the strongest trump card wins the *trick*, earns the points from the four cards and begins the next trick. After 9 tricks have been played, the next round begins with the new starting player being the one counter-clockwise to the right. As soon as one team reaches a set number points, they win the game. In each round, 157 points can be made - with the exception of 257 points for a *match* (winning all 9 tricks).

2.2 Server

Our Jass server was built upon the implementation of the Zühlke Jass Challenge [4].

2.2.1 Setup

The back-end of our WebApp runs on a Express server in Node.js [5][6], whereas the front-end was developed using React.js [7]. As soon as a client accesses our website[1], he connects to the server via a new websocket. The WebSocket Protocol enables two-way communication over a persistent TCP connection between a web browser and a web server [8]. This allows the back-end to send messages to the client without having to wait for a request. On connection, the **SessionHandler** module starts sending requests to the browser. By starting a new game, the client sends back a response. With that information, the **SessionHandler** creates a new Jass **Session** to which the player's websocket is added. The game starts once four players have joined the session. Throughout the game, the server sends information about the current state to all the clients and/or requests the next move from them.

2.2.2 Bots

Instead of waiting for 3 other players, the client can fill the remaining slots with python bots. In [1], Rimensberger has created four different agent types that can play a game of Jass, namely bot A, bot B, bot C and bot D. We decided to only use bot C and bot D for our WebApp, as they achieved the highest performance in testing [1]. Bot D is a mixed agent having a policy trained by Proximal Policy Optimization [9] for the trump phase and a Determinized Monte Carlo Tree Search (DMCTS) [10] agent with 40 determinizations of 400 iterations each for the card play phase. Bot C uses a DMCTS agent with 10 determinizations of 100 iterations each for both phases [1].

2.3 Data

All played games are saved to a SQL database [11] using phpMyAdmin [12]. For each round, we store the round number, trump call, starting player, game mode (see Chapter 3.1) and the points, as well as the four starting hands. In addition, to be able to reconstruct every game, each trick is also saved. In total, the collected information consisted of over 14'500 rows of data. However, that included many rounds and tricks from prematurely finished games. Therefore, we pre-processed the data by filtering out all the games that lasted less than 6 rounds. This resulted in over 11'800 rows of data to analyse.

[1] https://jass.ethz.ch

Chapter 3

Extensions to the WebApp

Users now have the possibility to log in to our WebApp, which allows us to track multiple games from the same player. After logging in, the user name is saved to the local storage of the browser. When setting up a new Jass game, the player is given several options. S/he can create a private table which requires entering a password to join or watch the game, and s/he can set the number of points needed for winning, e.g. 1000 points. But most importantly, the user is given the choice between the standard game mode and the fair game mode.

3.1 Fair Game Mode

Jass is considered partially a game of luck, as each starting hand of 9 random cards determines the chance of winning the round. In order to eliminate this luck factor, we created a fair game mode with the intention of allowing both teams to start with same hands. This gives each team an equal opportunity and would level the playing field, thus making the game fair.

Letting a player start with the same hand as an opposing player in a previous round requires dealing the same cards again, i.e. repeating the round.

Definition 3.1 (Repeated Round)**.** We say round r' repeats round r if both have the same starting hands and round r' is played after round r. Round r' is called the repeated round, whereas round r is called the original round. Together they form a round pair (r, r').

Thus, a fair game consists of only round pairs, i.e. every round is part of a round pair. However, the team starting in a repeated round might have a disadvantage, as an opposing player could remember the cards from the original round. To balance this out, both teams must start in an equal number of original rounds. In addition, before dealing the cards, the server randomly permutes the suits to hide which rounds are repeated (Figure 3.1). Note that games remain fair under suit permutation, as all four suits are equally strong prior to the trump call, i.e. swapping any two suits in a set of starting hands doesn't change the

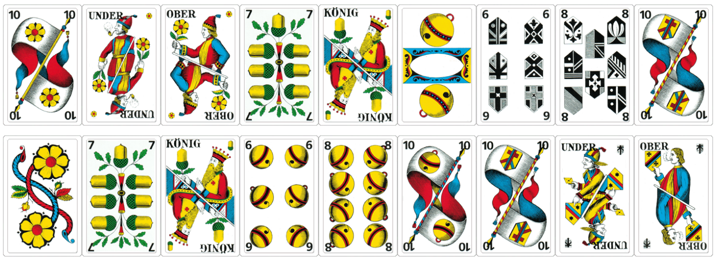


Figure 3.1: An original and its repeated hand. Note that the values are the same, just the suits are permuted.

game dynamic and outcome. Further, we can permute the order of the games under the restriction that each team starts in the same number of original rounds. In practice, when a new fair game with n rounds is started, the server calculates a random valid sequence d of original and repeated rounds.

$$d = [x_1, x_2, ..., x_n] \text{ where } x_i = \begin{cases} 0 & \text{if round } i \text{ is an original round} \\ t & \text{if round } i \text{ repeats round } t \end{cases}$$

This sequence is then used throughout the game to decide which cards to deal to the players.

Example 3.2. Assume a fair game with 12 rounds was started. Hence, there will be 6 original and 6 repeated rounds, 3 of each for both teams. The server then calculates a random valid sequence d, e.g. [0, 0, 2, 0, 0, 5, 4, 0, 8, 1, 0, 11]. In this example, the first two rounds are original rounds and played with randomly distributed cards. Then, round 3 repeats round 2, and the server deals the same cards as in round 2 to the players. But in order to make it harder to recognize a repeated hand, the suits are randomly permuted. Rounds 4 and 5 are again original rounds, whereas round 6 repeats round 5. The game continues until all 12 rounds have been played. Finally, the team with more points wins.

3.2 Replay Mode

A logged in user has access to all of his/her previously played games and is given the possibility to replay them. When a game is replayed, the server first fetches all the relevant information about that game from the database and then displays it to the user, including every card in each players hand. Then, s/he can play through the game card by card by choosing one of the four following actions (refer to Figure 3.2).

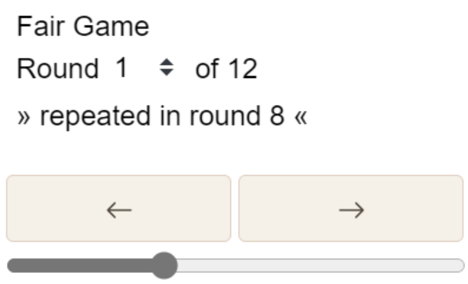


Figure 3.2: Actions in the replay mode.

1. **Play the next card:** Places the next card that was played in that game from its hand onto the table.
2. **Play the previous card:** Removes the previously played card from the table and puts it back into the respective hand.
3. **Move the slider:** Allows the user to jump to any played card of the current round. Plays/removes all cards up to that point.
4. **Switch to a different round:** Removes all cards on the table and from the hands, and loads the newly selected round.

Furthermore, if the replayed game was a fair game, the user can directly switch between an original and a repeated round, and also sees the difference of points made when starting in those rounds compared to the other team.

Definition 3.3 (Round point advantage)**.** The round point advantage RPA(r_1) of round r_1 that forms a round pair with round r_2 is defined as the difference of their starting team points.

By Definition 3.3, it follows that

$$RPA(r_2) = -RPA(r_1) \tag{3.1}$$

for every round pair (r_1, r_2). The round point advantage (from now on RPA) reveals the better playing team in repeated rounds. In general, the replay mode enables users to analyse their plays, identify potential mistakes and discover better strategies.

Chapter 4

Evaluation

As of this writing, over 130 Jass games have been played on the WebApp. In this section, we will analyse the performance of the AI and evaluate the fair game mode.

4.1 Performance of the Bots

In order to assess the performance of the bots, we will evaluate all games stored in the database (see Chapter 2.3).

4.1.1 General Performance

Table 4.1 shows the number of games played by each team and player type as well as their win rate and average point advantage. Overall, bot D achieved the best performance and won over half of its games, whereas bot C was losing 3 out of 5 games on average. As a team, the D bots won 31 of their 57 games (5 of 9 against two human players), giving them the second highest winning rate of 54%. On the other hand, the C bots lost 34 out of their 54 games as a team, thus having a significantly smaller winning rate of 37%. In particular, they won only once against a team of two human players while losing four times. The highest win rate as well as the largest point advantage of 79.20 points was achieved by human players who played alongside a D bot. One reason might be that they faced a team of two C bots in the majority of their games, which are notably the weakest players, and therefore often won by a high margin.

Overall, this evaluation mostly matches the results from Rimensberger [1]. In his experiments, a team of C bots had an almost identical win rate of 0.38 compared to the one shown in Table 4.1. The same applies to D bots with a 0.04 lower win rate of 0.5. Only human players had a notably higher win rate of 0.68. However, this result includes games played against a pair of the much weaker A and B bots (win rate of 0.0 resp. 0.2).

		Bot C	Bot D	Human Player
All games				
Overall	Games Played	158	165	169
	Win Rate	0.40	0.57	0.52
	Point Advantage	-53.41	30.02	20.63
As a team	Games Played	54	57	34
	Win Rate	0.37	0.54	0.47
	Point Advantage	-77.72	8.02	-7.47
With a human	Games Played	50	51	-
	Win Rate	0.48	0.63	-
	Point Advantage	0.9	79.20	-
Fair games				
Overall	Games Played	126	122	124
	Win Rate	0.40	0.56	0.55
	Point Advantage	-67.43	36.49	32.62
Decisive Rounds	Rounds Played	45	46	47
	Win Rate	0.33	0.59	0.57
	Average RPA	-33.78	19.26	13.49

Table 4.1: Results from the games played on the WebApp. The first row shows the performance of each player type independent of its team partner. The middle row displays the results of two bots or two human players playing as a team. In the third row, we see the outcome of human players playing alongside a bot. The final two rows indicate their performance in fair games.

4.1.2 Performance in Fair Games

Definition 4.1 (Decisive Round)**.** A round pair including the round with the highest RPA is called a decisive round pair and its rounds are the decisive rounds.

Decisive rounds usually determine the winner of a fair game. In fact, in 86% of the games played, the winner of the decisive round pair has won the entire game. To determine why bot C has the lowest win rate, we can analyse their strategies in the decisive rounds.

In Table 4.1, we see the average RPA for each player type when being the starting player in decisive rounds. As expected, bot C put in the weakest performance with an average RPA of -33.78 points. Bot D outclassed human players once more with an average RPA of 19.26 points compared to the latter's 13.49 points. As both starting players have an equal hand in decisive round pairs, we can compare their trump calls to determine which one lead to winning the round. Figure 4.1 shows that in 39% of the time, both players chose the same call. Surprisingly though, 33% of all decisive round pairs were won by the player who decided to shove when the other player did not. And in fact, bot D made the shoving call in 19 out of the 46 decisive rounds and won 15 of them (see Figure 4.2a), implying that its partner most likely had a better starting hand in those rounds. On the flip side, bot C lost most decisive rounds after not shoving, as shown in Figure 4.2b.

We can conclude that shoving often resulted in a high RPA and therefore a decisive round. Unlike bot C, human players and bot D regularly let their partner decide the trump call when starting with a weak hand. This ability to deal with poor starting hands was a decisive factor in many games and might be one reason for the performance difference.

4.2 Fair Game Mode Evaluation

Fair games should turn out to be closer on average, as they are more balanced. Figure 4.3 compares the distribution of the point advantage from the winning team in both fair and standard games. And in fact, fair games tend to be significantly more hard-fought, as half of them were won by a margin smaller than 66 points. On the flip side, in standard games, the median of the winner's point advantage was 91 points, an increase of 38% compared to the fair game mode. However, many fair games (33%) still ended up having a clear winner, i.e. a point advantage of over 120 points. Nevertheless, this does not contradict the game's fairness, as if two teams of different strength play a balanced game, the better team is likely to win by a high margin. In the standard game mode on the other hand, the weaker team could open with better starting hands, giving it a fighting chance.

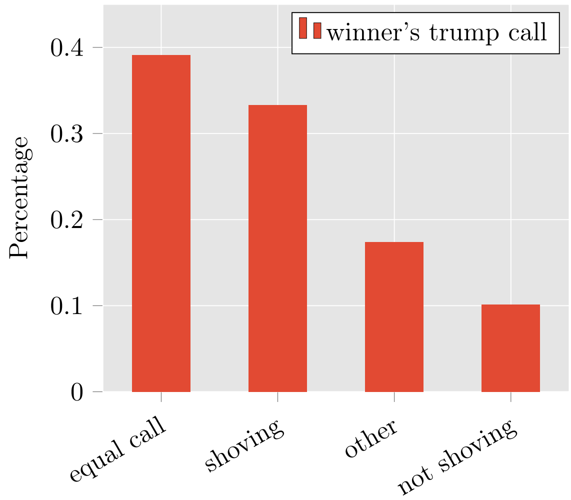
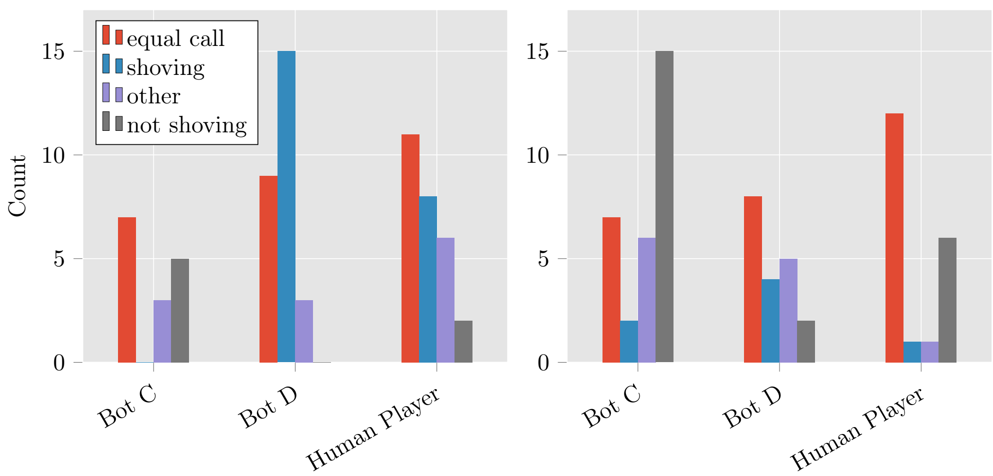


Figure 4.1: Comparison of the two trump calls in decisive round pairs. Either both are the same (*equal call*), only one player decided to shove (*shoving* if that player won, else *not shoving*) or both players didn't shove, but chose a different trump (*other*).



(a) Winning

(b) Losing

Figure 4.2: Trump calls made by each player type when (a) winning or (b) losing a decisive round pair.

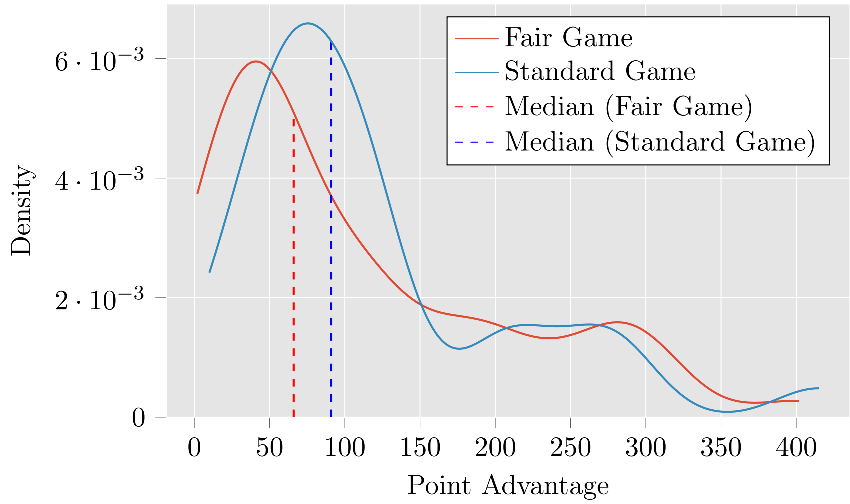


Figure 4.3: Point advantage distribution in fair games and standard games.

4.2.1 User Study

One might argue that repeated hands can easily be recognized despite permuting the suits, which could give the second team that plays a hand an advantage. Even though the overall game would still be fair, as both teams start in the same number of repeated rounds, it might favour more skilled players. Therefore, we started a user study where players were asked to fill out a form about their experience with the fair game mode (see Appendix A for details). Most importantly, we wanted to discover which players, if any, would recognize repeated hands and could take advantage of it.

Most players (87%) weren't able to recognize any repeated hands (Figure 4.4). Out of the 6 players who did, only 2 were stating that this knowledge gave them an advantage. Interestingly, those two players assessed themselves as professional (5) respectively very good (4) Jass players, whereas the other four players rated themselves at lower levels (2, 3, 3 and 4). Figure 4.5 shows higher level players gaining more advantage in the fair game mode. However, only 1 out of 5 professional Jass players could recognize any repeated hands, so more data would be needed in order to verify this tendency. On the other hand, many players thought that this knowledge could give them an advantage in further games. When asked, 17.4% were sure about the future advantage, whereas the majority of 54.3% answered "Maybe". 3 out of 4 players who played multiple games and submitted the form changed their opinion on the potential future advantage. After the first game, all four players answered "Maybe". But, after the succeeding games, they switched to a "No", suggesting that they came to the conclusion that it's too difficult to recognize hands and thereby gain an advantage.

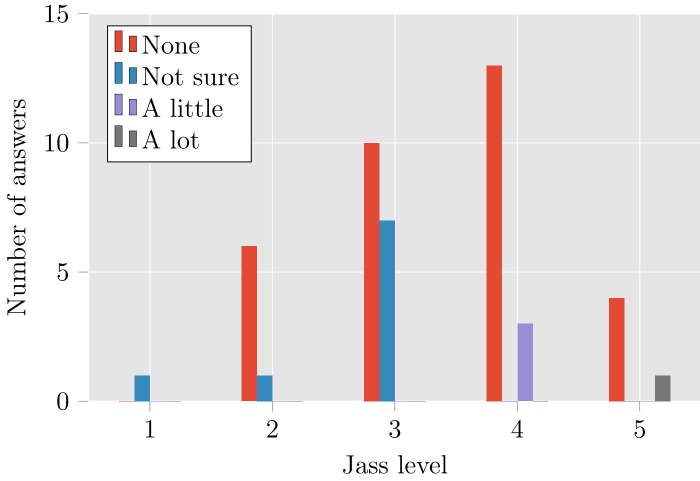


Figure 4.4: Number of recognized hands in one fair game according to players of different Jass levels.

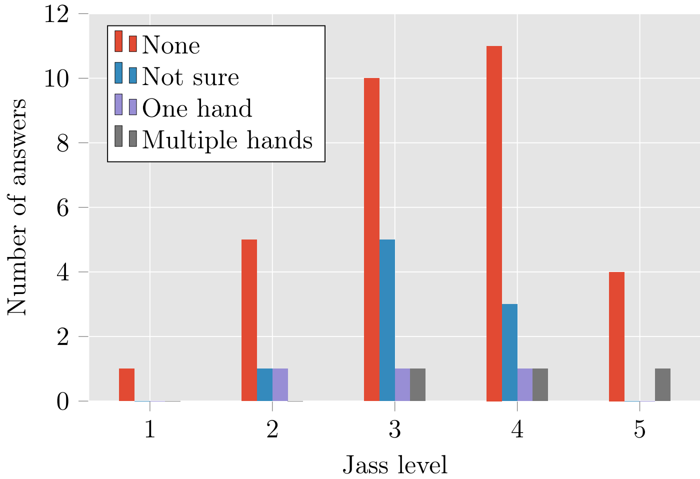


Figure 4.5: Gained advantage by recognizing repeated hands in one fair game according to players of different Jass levels.

Chapter 5

Conclusion

The feedback on the WebApp was mostly positive. Users have mentioned the strength of D bots, reflecting our evaluation of their performance - even having a positive win rate against a team of two human players. On the other hand, bot C didn't perform as well as the other player types, but could win games nonetheless. One of its weaknesses seemed to be choosing the most optimal trump call. As a matter of fact, multiple players have stated that bot C made incomprehensible and confusing trump calls.

In this thesis, we have introduced a new fair game mode to eliminate the luck factor in Jass and allowing users to play a fair and more balanced game. Actually, most games on the Jass server were played in this mode, resulting in a sufficient amount of data to analyse the fairness and its consequences. We discovered that fair games tend to be closer than standard Jass games, implying that the fair game mode functions as intended. In addition, we learned in a user study that most players could neither recognize repeated hands nor gain an advantage from that.

For a future version of the Jass server, we would only consider bot D for playing against and/or alongside human players, as they offer the greatest challenge. In addition, we want to implement a user management system such that players can create an account which facilitates keeping track of their games. This would also allow for a ranked mode in which players can compete in a tournament-style setting against each other. Such a ranking might attract more skillful Jass player leading to the possibility of evaluating the agents in a more competitive environment.

Bibliography

[1] N. Rimensberger, "Developing a jass ai," in *Distributed Computing Group, Computer Engineering and Networks Laboratory, ETH Zürich*, Aug. 2020.

[2] S. Çeven, "Swisslos: No online-jass for gambler from liechtenstein," Jan. 2021. [Online]. Available: https://www.casinoonline.de/nachrichten/swisslos-online-jass-nicht-mehr-in-liechtenstein-50550

[3] R. Nacheva, "Principles of user interface design: Important rules that every designer should follow," in *Final conference - Science in the Service of Society - 2015*, 10 2015.

[4] F. Lüscher, A. Herzog, and J. Akeret, "Jass challenge," Jun. 2017. [Online]. Available: https://github.com/webplatformz/challenge

[5] "Express: fast, unopinionated, minimalist web framework for node.js," 2021. [Online]. Available: https://expressjs.com

[6] S. Tilkov and S. Vinoski, "Node.js: Using javascript to build high-performance network programs," in *IEEE Internet Computing*, vol. 14, no. 6, 2010, pp. 80–83.

[7] "React: a javascript library for building user interfaces," 2021. [Online]. Available: https://reactjs.org

[8] I. Fette and A. Melnikov, "The websocket protocol," in *Internet Requests for Comments 6455*, Dec. 2011.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," in *arXiv:1707.06347*, Jul. 2017.

[10] D. Whitehouse, E. Powley, and P. Cowling, "Determinization and information set monte carlo tree search for the card game dou di zhu," in *2011 IEEE Conference on Computational Intelligence and Games, CIG 2011*, Oct. 2011.

[11] D. Chamberlin, "Sql," in *Encyclopedia of Database Systems*, L. LIU and M. T. ÖZSU, Eds. Boston, MA: Springer US, 2009, pp. 2753–2760. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_1091

[12] "Phpmyadmin: Bringing mysql to the web," 2021. [Online]. Available: https://www.phpmyadmin.net

Appendix A

User Study

To evaluate the fair game mode, players were asked to fill out a form with the following questions after playing a fair game.

1. Where you able to recognize any repeated hands?
2. Do you think that knowledge gave you an advantage?
3. Do you think that will give you an advantage in future games, as you know better how it works?
4. On a scale from 1 (Beginner) to 5 (Professional), how would you rate your Jass level?

Table A.1 and Table A.2 show the answers of all 46 players who submitted the form.

	Recognized Hands	Advantage	Future Advantage	Level
1	Maybe	No	No	3
2	No	No	Yes. Definitely	4
3	No	No	Maybe	5
4	No	Maybe	Yes. Definitely	1
5	No	No	Maybe	3
6	No	Maybe	Maybe	3
7	No	Maybe	Maybe	3
8	Maybe	No	Yes. Definitely	4
9	No	Maybe	Maybe	3
10	No	No	Maybe	4
11	No	No	No	4
12	No	No	Maybe	5
13	No	No	No	5
14	No	Maybe	Maybe	2
15	No	No	Maybe	3
16	No	No	No	3
17	No	No	Maybe	2
18	No	No	Maybe	2
19	No	No	No	2
20	No	No	Maybe	3
21	No	No	Yes. Definitely	4
22	Maybe	No	No	2
23	No	No	Maybe	2
24	One hand	No	No	3
25	One hand	No	Yes. Definitely	4
26	Maybe	Maybe	No	3
27	One hand	No	No	2
28	Maybe	No	Maybe	3
29	No	No	Maybe	4
30	No	No	Maybe	4

Table A.1: Result of the user study (first 30 answers).

	Recognized Hands	Advantage	Future Advantage	Level
31	No	No	Maybe	3
32	Maybe	Yes. A little	Maybe	4
33	No	No	Maybe	4
34	No	No	No	5
35	Multiple hands	Yes. A lot	Yes. Definitely	5
36	Maybe	Yes. A little	Yes. Definitely	4
37	No	No	Maybe	4
38	No	No	No	4
39	No	No	No	4
40	Maybe	Maybe	Maybe	3
41	Maybe	Maybe	Maybe	3
42	Multiple hands	No	Maybe	3
43	Multiple hands	Yes. A little	Maybe	4
44	No	Maybe	Yes. Definitely	3
45	No	No	Maybe	4
46	No	No	No	3

Table A.2: Result of the user study (remaining 16 answers).