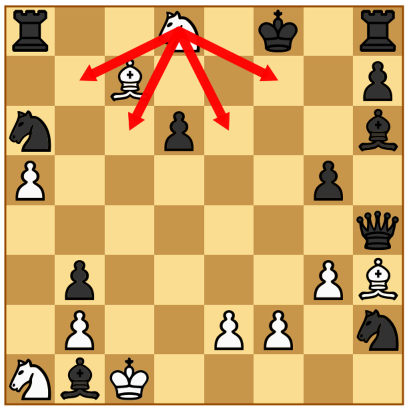
Project Report Group 9: Visual Tutorials for Games

Rick van Bellen, Max Knappe, Andreas Petridis, Abel de Wit

20 January 2021



Abstract

The effectiveness of a player in a specific game is directly dependent to whether the player fully understood the rules and the concept of that game. Textual instructions and rule books exist in almost every game to serve that purpose. However, as *"a picture is worth a thousand words"*, visual tutorials are demanded to improve the player's comprehension of the game's rules. In this project, a framework is developed to automatically generate visual tutorials for games.

Contents

1 Introduction

1.1 Context and Motivation

Currently, the rules explanation of games are often composed by hand. The process of creating content which explains the workings of a game well is time consuming. Additionally, it can be hard to tie textual information to what is visually perceived. Visual explanations are faster to understand, more intuitive, and more accessible. It is therefore interesting to automatically generate visual explanations which can describe the rules of the game to humans.

Additionally, such a process would be potentially translatable to teaching the strategy of games. While rules are concerned with defining what is allowed at a single moment in a game's state, strategy is related to plans across sequences of actions that lead to better play.

The project's goal is to develop a framework which can automatically generate visual tutorials from games described with the Ludii general game system, using an approach that can be generalized to work outside of the Ludii system as well.

1.2 Objectives

For this project, our objective is the creation of an algorithm that, for a given game, will be able to automatically:

- Extract the rules of the game from game information.
- Analyze which moves are important to explain.
- Cluster the moves based on piece or move type.
- Generate images explaining the important moves.
- Combine the images into a coherent tutorial.

This algorithm will be implemented as an extension of the Ludii package, but the approach explained in this report should be usable for any other game system. Examples of existing tutorials such as Figure 1 and tutorials that we want to achieve such as Appendix E, show how a game might be explained using imagery.

1.3 Ludii General Game System

Ludii is a general game system developed for the ERC-funded Digital Ludeme Project [Browne, 2018]. Ludii is designed to play, evaluate and design a wide range of games such as board games, card games, dice games, mathematical games, etc. Ludii's consists of a great database of games, which can be played both by humans and AI or random agents. Ludii provides to the players important information of the game such as:

- played moves
- rules of the game
- analysis on the game
- corresponding Ludeme [Parlett, 2017] of the game
- description of the game
- origin of the game and similar games

Ludii also contains many in-game features that can increase the experience of playing, for instance restart the game, see last move or possible legal moves. Last but not least, Ludii contains many tools that could be used during the project, such as a way to play game trials automatically, and a way to create screenshots from the game board that could be used in the visualizations.

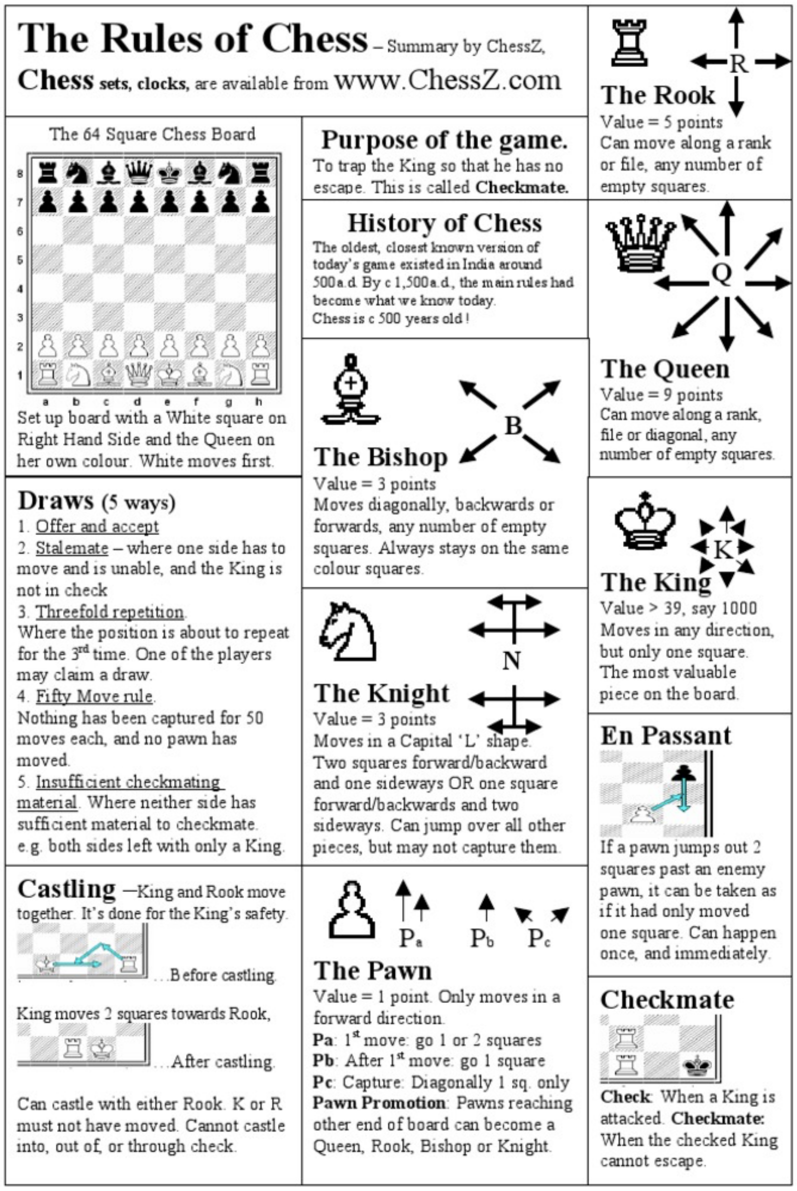
1.4 Research Questions

The project gave rise to the following research questions:

- Is it possible to explain game rules for a game, for example Breakthrough, using only automatically generated images?
- Can the generated tutorials provide full understanding of the game?
- Can this approach be generalized to work for any game?
- How to ensure that all possible move types and piece types are covered in the generated tutorial?

2 Related Work

Tutorials are usually the first thing a player faces in any type of game or software. Prior research exists in different approaches on the tutorials topic. In [Li et al., 2013], an automated tutorial generator is presented that creates visual and textual instructions to help users learn AutoCAD software. In [Green et al., 2018, Green et al., 2017, Therrien, 2011], tutorials on video games are studied. In [Green et al., 2017] tutorials are classified in four dimensions, and three common tutorial types (*"Teaching using instructions, "Teaching using examples", "Teaching using experience"*) are highlighted. [Therrien, 2011], introduces how the tutorials in video games evolve through time. In [Iida et al., 1995], a search strategy is introduced for tutoring board games while playing, where a tutor is needed to coordinate a player. In [Björnsson, 2012], an algorithm for learning game rules of simplified board games is proposed and in [Gregory et al., 2016], it is extended. In [Hagen, 2013], the author examines if the use of aesthetics and gameplay design can give a better understanding of the games instead of tutorials. However, it turns out that tutorials give better understanding of the rules.



The Rules of Chess – Summary by ChessZ,
Chess sets, clocks, are available from www.ChessZ.com

The 64 Square Chess Board

Set up board with a White square on Right Hand Side and the Queen on her own colour. White moves first.

Purpose of the game.
To trap the King so that he has no escape. This is called **Checkmate.**

History of Chess
The oldest, closest known version of today's game existed in India around 500a.d. By c 1,500a.d., the main rules had become what we know today.
Chess is c 500 years old !

The Rook
Value = 5 points
Can move along a rank or file, any number of empty squares.

The Queen
Value = 9 points
Can move along a rank, file or diagonal, any number of empty squares.

The Bishop
Value = 3 points
Moves diagonally, backwards or forwards, any number of empty squares. Always stays on the same colour squares.

Draws (5 ways)
1. Offer and accept
2. Stalemate – where one side has to move and is unable, and the King is not in check
3. Threefold repetition.
Where the position is about to repeat for the 3rd time. One of the players may claim a draw.
4. Fifty Move rule.
Nothing has been captured for 50 moves each, and no pawn has moved.
5. Insufficient checkmating material. Where neither side has sufficient material to checkmate. e.g. both sides left with only a King.

The King
Value > 39, say 1000
Moves in any direction, but only one square. The most valuable piece on the board.

The Knight
Value = 3 points
Moves in a Capital 'L' shape. Two squares forward/backward and one sideways OR one square forward/backwards and two sideways. Can jump over all other pieces, but may not capture them.

En Passant
If a pawn jumps out 2 squares past an enemy pawn, it can be taken as if it had only moved one square. Can happen once, and immediately.

Castling –King and Rook move together. It's done for the King's safety.
…Before castling.
King moves 2 squares towards Rook,
…After castling.
Can castle with either Rook. K or R must not have moved. Cannot castle into, out of, or through check.

The Pawn
Value = 1 point. Only moves in a forward direction.
Pa: 1st move: go 1 or 2 squares
Pb: After 1st move: go 1 square
Pc: Capture: Diagonally 1 sq. only
Pawn Promotion: Pawns reaching other end of board can become a Queen, Rook, Bishop or Knight.

Checkmate
Check: When a King is attacked. **Checkmate:** When the checked King cannot escape.

Figure 1: An example of a visual tutorial for chess.

Despite the fact that people play board games for thousands of years [Piccione, 1980] and board games exist as a hobby for decades, not much research has been done on automated generation of visual tutorials for board games.

3 Methodology

To achieve the goals of this research, the process was divided in three main components.

1. Data Generation
2. Move analysis
3. Visualisation

Each of the design choices and workings of these components will be explained in the sections below.

3.1 Data Generation

To be able to infer what types of moves are played often, or are interesting, the game should be played many times. Ludii provides a method that, given player algorithms, thinking time, and how many moves should be played, generates a so called playout of the game at hand. This playout will contain all of the properties that are needed for our analysis; the moves that have been played, what type of move this was, and which player won in the end. The Ludii framework already possesses multiple algorithms, of which an Alpha-Beta algorithm that adapts for each game, and a random algorithm that will just play random moves. Because our research needs a lot of played games, even Alpha-Beta with restricted thinking time took too long to generate a substantial amount of data, hence the choice was made to use the random algorithm for performance. Other than performance, the random algorithm also makes sense when generating a large amount of moves. The goal is to explain what moves result in legal moves, captures and eventually, winning moves. If for future work it is attempted to find strategic moves it would be necessary to use an algorithm that actually thinks about the moves it makes, for merely explaining possible moves this would be overkill.

3.2 Move analysis

After a game is played, the complete move list of the game is passed to the next component of our work, the generalisation and analysis. In the game Breakthrough, a piece can move forward either diagonally or vertically if that location is empty, and it can only capture the opponent's pieces diagonally[Ludoteka.com,]. Whether, for example, a piece moves from A1 to B2 or from C3 to D4 does not matter. These moves can both be generalized to a 'diagonal move to the top-right'. This is done for each move that is passed to the generalisation method.

The generalised string and it's respective move are then sorted in four lists with sub-lists:

- Orientation (*Horizontal, Vertical, Diagonal*)
- Direction (*top, top-right, right, down-right, down, down-left, left, top-left*)
- Move length (*Step, Leap*)
- Move type (*Move, Capture, Jump-over*)

After this sorting has been completed for each move of each game that is played, analysis of the results can be done.

By looking at the sum of each sub-list and the time a move has been played, it can be inferred what the most common move is to play first. Even though it is random, in for example a thousand games, this still holds information about how to start a game. The same goes for the move that occurred most as the last move, indicating the most general way of ending the game.

The above mentioned moves are easily distinguished because the moment in the game is the most important factor to filter on. To show all allowed moves and captures that happen during a game, another technique is needed.

A method was created that will search for moves that occur in multiple lists, this opens up the possibility to iterate over each move type, and see whether for that type, moves occurred in a specific direction. For example, one could be searching for capture moves, so the parameter that is passed to the method is "Capture" and the method will return for each direction a capture if there exists one. In our example of Breakthrough, this results in a returned list with only a move for top-left and top-right.

The program can then call this method for every piece in the game, for every move type there is, and it will such gather a comprehensive list of moves in all directions for a certain type of move. At this point, a move for the opening and closing of the game is extracted. As is each possible move for the pieces on the board.

3.3 Visualisation

With the extracted move, the next method will generate new playouts and check whether the move that is wanted is part of this trial. When the method finds a trial that contains the move, it will create a trial file that contains that playout and return the place in time on which the move occurs.

The Ludii framework then has methods that allow for loading of this file and jumping to the number of the move. These methods are called with the output of the trial search and open the corresponding trial at the time that is wanted. The screenshot method of Ludii is then called, the board is cropped out of the picture of the whole GUI and the generated image is saved with a suffix of the type of move that was captured (*Opening, Closing, Move, Capture,* *etc.*). The steps described are repeated for each move that was chosen by the analysis resulting in a handful of images showing how to open the game, how to win the game, and how to move around and capture pieces. With these images, a program can be written that will combine all of the images into a full tutorial, using the above mentioned suffixes as sections of the tutorial. A general introduction into the game such as the game description that Ludii provides could also be added automatically from the ludeme, providing a fully fledged tutorial.

4 Experiments

To test the implementation, Breakthrough, which is briefly explained in section 3, as well as Chess are used to see if all of the important rules are found. We decided to use those specific board games, as breakthrough is a symmetric and easily explained game, with only one type of piece. Breakthrough consists only a small amount of rules, and therefore they can simply and precisely be inspected. On the other hand, chess is one of the most known classical board games. Its basic rules and some insight on the game are well known to the public. Chess consists of 6 different types of pieces which have their own movement and some special moves (e.g. *castling*). It is chosen, to check if the tutorials work, and in which grade, with a more complicated game consisting of more rules, pieces and moves. Both Breakthrough and Chess can be played on a square board, by two players who have exactly the same pieces. Tutorials for both games are developed with random agents playing and Ludii AI agents playing. In the final tutorial of Breakthrough it needs to be conveyed that a piece can move forward 1 step vertically or diagonally, and capture only diagonally. Chess, since it has way more complicate rules that need to be explained, is an interesting test case to see how many of those rules are conveyed through the visualizations. To create these tutorials, the games are loaded into Ludii, and the number of random trials is set for each game.

4.1 Breakthrough

The Breakthrough tutorial with random agents playing, is created using 1000 random trials. Every game is inevitably converging to win for either player, since the only option is to move forward, which always results in either reaching the other side, or losing all of your pieces. After all of the trials are run and the images are created, they are manually combined into tutorials. These tutorials are then compared to already existing tutorials, namely the rules as explained on [Ludoteka.com,]. To check whether tutorials generated by a random agent indeed perform at least as good as tutorials generated by an AI agent, the experiments are done again, but with the Ludii AI agent making the moves. The Ludii AI agent is configured using alpha-beta search, with a thinking time of 0.01 seconds. This thinking time unfortunately has to be low, since otherwise the runtime of the tutorial generation increases significantly. The tutorials are generated using 50 trials. The results are compared to the results gained when using a random agent.

4.2 Chess

The Chess tutorial with random agents playing, is created using 50 trials, a significant small amount in contrast to the corresponding in Breakthrough tutorial. This is because a random game of chess can take a very long, since the game does not always converge to a win for either player as it happens in Breakthrough. After the trials are run and the images are created, the tutorials are created manually consisting those images. The tutorials are compared to existing tutorials, namely the rules as explained in Figure 1. To check whether tutorials generated by a random agent achieve similar results as the tutorials generated by AI agents, the same process as in Breakthrough is followed again using 50 trials. The results are again compared to the results gained using a random agent.

5 Results

5.1 Breakthrough

The tutorials created with random agents for Breakthrough and chess can be found in Appendix A and B respectively. The tutorials created with the Ludii AI can be found in Appendix C and D. From Appendix A and C, it can be concluded that all possible moves are shown by at least one image for both the random agent and Ludii AI. However, one of the capture move images has no arrows for the random agent, and one regular move has no arrows for the Ludii AI. The reason for this has not been identified as of yet. Additionally, for both versions three images are created for the regular move, while this could be conveyed with only one. This is because one image is created for every type of move (to the top-left, top, and top-right), and every additional legal move is added to the image. This ensures that every move is shown, but it does make some images redundant. The winning move image for the random agent shows that it does not matter whether or not you reach the end by capturing or moving regularly, which is a nice clarification.

All in all, there is no meaningful increase in quality when using the Ludii AI, while the random agent takes less time to play out. This confirms the fact that using the random agent is the right approach for creating a Breakthrough tutorial.

Comparing the created tutorials to the already existing tutorial[Ludoteka.com,], it can be concluded that indeed all of the possible moves as explained on Ludoteka are present in the tutorials generated by our approach. The rules that are not explained however, are the initial board layout, the fact that the player who uses the white pieces starts the game, and the fact that players take turns alternatively.

5.2 Chess

For chess, the results are relatively inconsistent. Both tutorials only have a good depiction of the possible moves of a subset of piece types. Most of the capture moves are empty, and the one that is not shows a capture move for a rook, which unfortunately does not show that it can capture over longer distances. How a pawn or queen can move is not touched on in these tutorials. By dividing the moves into the various piece types this issue can be solved. The winning moves are taken from games that ended in a draw, and how to win is therefore not conveyed well. Using the Ludii AI with more time to think might solve this issue, by making sure that games end in a checkmate more often.

Comparing the chess tutorial created with the random agent to the one created with the Ludii AI again shows no notable improvements. This might be due to the fact that the thinking time has to be quite low in order to generate the tutorials within a reasonable time frame. As with Breakthrough, since the random agent generates the tutorials faster, using the random agent seems to be the best idea.

When the results are compared to the already existing tutorial in Figure 1, many rules are left out. Aside from multiple ways to move and capture, rules about the initial board state, and ways to win and get a draw are missing. More future research will have to be done in order to include more rules in the chess tutorials.

6 Discussion

6.1 Research Questions

One of the research questions that was posed asked whether it is possible to explain game rules for a game using only automatically generated images. This research poses a first step in that direction by exploring how moves can be categorized based on their type, and how these categories can then be visualized in order to cover each type of move a piece can make. Images are generated in certain sections, and with a small extension of the code, these images can be combined into a full tutorial for the given game, hence it can be assumed that it is possible to explain game rules using only automatically generated images.

The next question that is asked is whether the generated tutorials provide full understanding of the game. This question cannot be answered with a computer program, but would need sufficient user testing to see whether the generated tutorial is sufficient for a first time player to play the game. Users would be presented a game that is unknown to them, and are asked to play the game (in Ludii) using the generated tutorial as their guide. Their performance in the game can then be measured and a questionnaire asking whether the users felt confident playing the game given the tutorial can be provided, giving researchers insight in how well the generated tutorial covers the rules of the given game. This user testing fell outside of the scope of this research, which means that the research question can not be answered with full confidence.

The approach taken in this research focused solely on square board with moving pieces, which is only a very small subset of the possible board games. The methodology of categorizing the moves in certain directions for certain move types limits the expansion to other games where pieces are added throughout the game such as Tic-Tac-Toe or Amazons. However, the idea of binning moves in subsets to extract different types of moves as categories can definitely be expanded to generalize better over a multitude of games, allowing for tutorial generation for most games. Future work might explore on how move types can be extracted automatically, resulting in automated subsets which can then be visualized for the tutorial. The answer to the question whether this approach can be generalized to work for any game is therefore yes.

To ensure that all possible piece types are covered in the generated tutorial, the methods implemented are ready to receive piece types as input to generate an image for a specific type. It was however not possible within the given time to figure out how to extract the amount and id's of piece types and how to extract this information from a playout which means that full coverage of all pieces is ready to be rolled out but not working as of now.

6.2 Improvements

While visual tutorials can get made for specific games using the approach explained in this report, there are quite some things that can be improved. Firstly, the move generalization is now hard coded for square boards as mentioned in 6.1. For example, when a piece moves 1 step in both the x and y direction, then this is classified as a diagonal step to the top-right. However, when the game board is hexagonal, such as with hexagonal chess (Figure 2), this approach does not work anymore. Therefore, the move generalization has to be improved to make sure that it can work for any Ludii game.

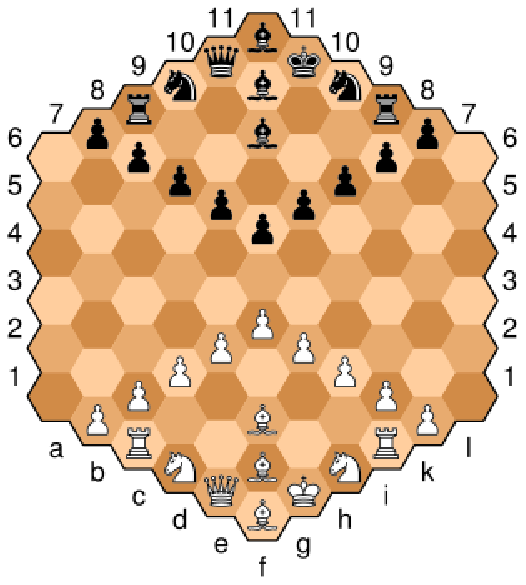


Figure 2: A hexagonal chess board.

Secondly, the Ludii GUI needs to be opened in order to create images for the tutorial. This is far from ideal, since it would be faster to have a way to create images without having to open another application, and new threads are started, which makes it harder to debug. To do this would require the creation of images from scratch, and this was not possible within the allotted time frame. Another issue with this approach is that the screenshots made also include the UI around the board. To make sure only the board itself ends up in the tutorial, only a portion of the frame is captured. However, the only way we found to do this was to take a percentage of the frame, which is different depending on the window size in which Ludii is run. This makes it so that the images created might include too much when the program is run on one device, while on other devices they might be cropped too much.

While the images created nicely depict what moves are possible, they do not always convey what is not possible. As an example, Figure 3 shows that a vertical capture move is impossible by showing every capture move that *is* possible, which are only the diagonal ones. The visualizations should be created in a way that impossible moves are always clear. This could be done by using the sandbox mode present in Ludii, in which a particular board state can be constructed by hand. Using this sandbox mode, placing an opponent piece at every location that an allied piece can move in shows all capture moves that are possible, and in turn all capture moves that are impossible.

It would also be very useful if the generated tutorials could be verified automatically. The tutorial should cover all of the move and piece types present in the game. An approach that could be used to do this is to turn off rules in the

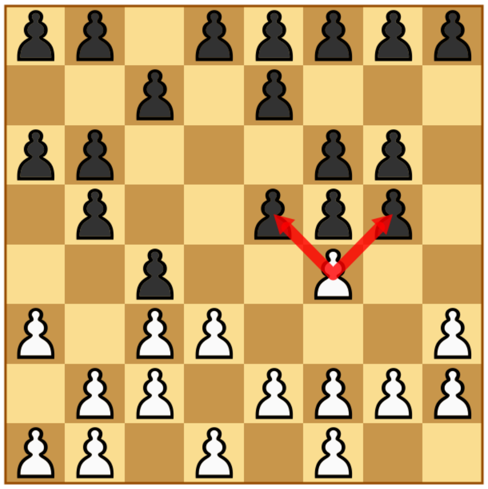


Figure 3: Image that shows that forward capture is impossible.

game description one by one, and see if this has an effect on the generated tutorial. If this is the case, then that means that a rule must be missing. Another approach would be to run a number of trial games and check if every move done in the trials is covered by the tutorial. The downside of this approach is that rules still can be missed, if they for example only occur once in a million games. The last approach to the validation problem is discussed already in 6.1.

Since the tutorials are generated from the perspective of player 1, it does not work for asymmetric games. This issue could be solved by creating a separate tutorial for player 2, and checking if they both contain the same rules using the same approach as explained in the previous paragraph.

Further research could also include creating tutorials that explain strategies that can be used in the game, instead of just explaining the rules. This idea can be extended upon by creating interactive tutorials, where the program automatically creates puzzles that the user has to solve by picking the correct move given a board state.

Finally, the images created currently still have to be combined manually into a tutorial. There was not enough time to find a way to automatically combine them into a pdf. However, with some further research this task is definitely possible. The moves are already separated into opening moves, regular moves, capture moves, and ending moves, so these distinctions can be used as chapter headers in the final tutorial.

6.3 Initial approaches

The initial plan was to use the AtDelfi[Green et al., 2018] approach in combination with the GRL [Gregory et al., 2016] system in order to create visual tutorials. However, during the second phase it was decided that this approach was not relevant for this project. This is because AtDelfi creates written tutorials as opposed to images, and these tutorials are designed for general games, not board games. The underlying graph architecture was too complex for the purpose of this project. In the AtDelphi game system, for every interaction of entities a graph is created. When talking about video games, this makes sense, as many different entities may interact in many different ways with one another. For board games, this is not the case. Board game interactions are mostly based on the position of game piece, and in contrast to video games, the piece cannot move freely. AtDelphi could model the interaction of for example one piece capturing another in chess, but wouldn't be able to model the different ways of moving, as a general interaction between a chess piece and an empty field is not given. Since most of the time in the first phase was allotted to understanding AtDelfi, unfortunately most of this time spent is not visible in the final result. Some of the problems with the current implementation could maybe have been improved if this approach was scrapped sooner.

Initially, the plan was to signify a capture move with a red cross on the location of the captured piece. However, feedback was given that this approach made it look like this move was impossible to do, and that the red cross was there to signify that the piece blocked the location. Therefore, this idea was scrapped.

7 Conclusion

In conclusion, the resulting algorithm is able to meet the first four objectives that we have set, only just for two specific games. It is able to extract the rules of these games, analyze the important moves, cluster these moves based on move type, and generate images explaining them. However, the images do not automatically get combined into a coherent tutorial, and the approach used in this report does not work for every Ludii game. The initial goal of this research was to create an algorithm that is able to generate a tutorial for any game, and while that has not been achieved, the groundwork has been laid for future researchers to expand on the move type binning to allow for a more general approach of type detection and visualization.

To answer the research questions posed in section 1.4, it is possible to explain game rules for Breakthrough using only automatically generated images. This approach can be generalized to work for other games, for example chess, but not for every game. For instance, games with hexagonal boards can not be generalized using the techniques explained in this report. The generated tutorials sometimes cover all the rules, and thus provide a full understanding of the game, but in some cases some rules are missing from the final result.

The objectives and research questions have been answered for a small subset of games and for just two games the produced results are shown. This research does, however, propose a way of approaching future studies in the automated generation of visual tutorials for board games, as the approach of move binning that is taken in this research is one that could be adapted for other board games, and maybe even generalised in such way that it is modular enough to work for any board game. Suggestions are also made on how to validate the tutorials once they are generated, which will provide researchers with even more insight in what type of tutorials do or do not work.

References

[Björnsson, 2012] Björnsson, Y. (2012). Learning rules of simplified boardgames by observing. In *Proceedings of the 20th European Conference on Artificial Intelligence*, ECAI'12, page 175–180, NLD. IOS Press.

[Browne, 2018] Browne, C. (2018). Modern techniques for ancient games. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE.

[Green et al., 2017] Green, M., Khalifa, A., Barros, G., and Togellius, J. (2017). " press space to fire": Automatic video game tutorial generation. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 13.

[Green et al., 2018] Green, M. C., Khalifa, A., Barros, G. A. B., Machado, T., Nealen, A., and Togelius, J. (2018). Atdelfi: Automatically designing legible, full instructions for games. *CoRR*, abs/1807.04375.

[Gregory et al., 2016] Gregory, P., Schumann, H. C., Björnsson, Y., and Schiffel, S. (2016). The grl system: Learning board game rules with piece-move interactions. In Cazenave, T., Winands, M. H., Edelkamp, S., Schiffel, S., Thielscher, M., and Togelius, J., editors, *Computer Games*, pages 130–148, Cham. Springer International Publishing.

[Hagen, 2013] Hagen, J. (2013). Teaching game mechanics in casual games without tutorials. In *WINONA COMPUTER SCIENCE UNDERGRADUATE RESEARCH SYMPOSIUM*, page 21.

[Iida et al., 1995] Iida, H., Matsubara, H., and Uiterwijk, J. W. (1995). A search strategy for tutoring in game playing. In *IJCAI-95 Workshop Proc., Entertainment and AI/Alife*, pages 14–18. Citeseer.

[Li et al., 2013] Li, W., Zhang, Y., and Fitzmaurice, G. (2013). Tutorialplan: automated tutorial generation from cad drawings. In *Twenty-Third International Joint Conference on Artificial Intelligence*. Citeseer.

[Ludoteka.com,] Ludoteka.com. Breakthrough. `https://www.ludoteka.com/v2019/breakthrough-en.html/`.

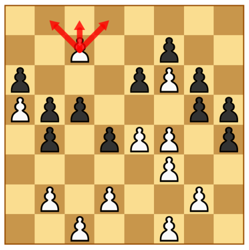
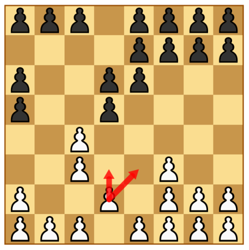
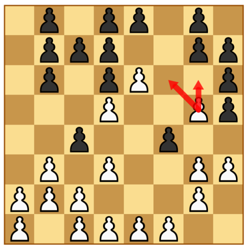
[Parlett, 2017] Parlett, D. (2017). What'sa ludeme? *Game & Puzzle Design*, 2(2):81.

[Piccione, 1980] Piccione, P. A. (1980). *In search of the meaning of Senet*. Archaeological Institute of America New York.

[Therrien, 2011] Therrien, C. (2011). " to get help, please press x" the rise of the assistance paradigm in video game design. In *DiGRA Conference*.

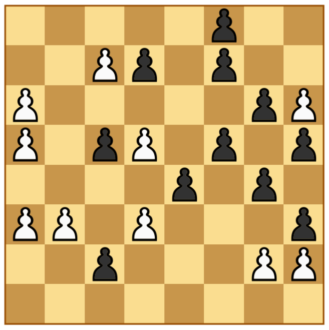
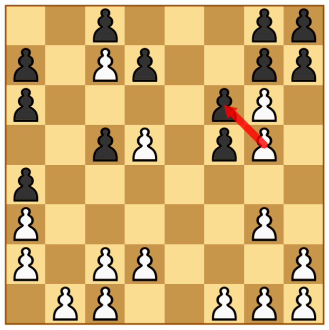
Appendix A: Breakthrough Tutorial (Random agent)

moves

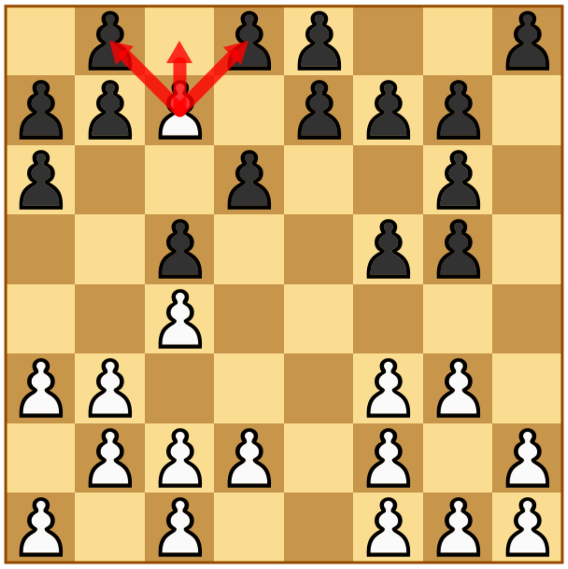
Diagonal move up-left, vertical move up, and diagonal move up-right.

capture moves

Diagonal capture up-right, and diagonal capture up-left.

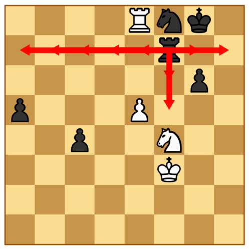
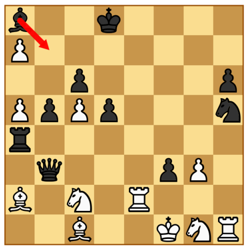
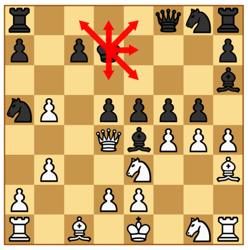
winning



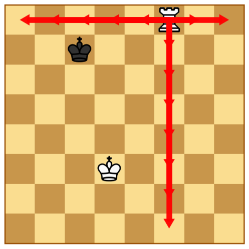
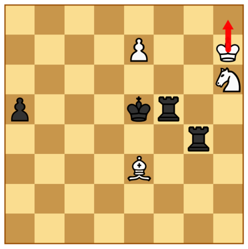
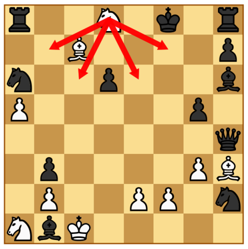
White player win.

Appendix B: Chess Tutorial (Random agent)

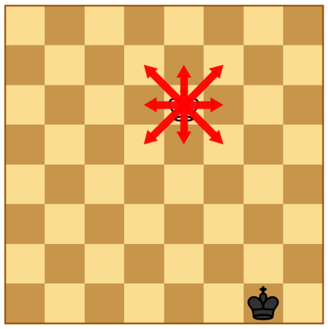
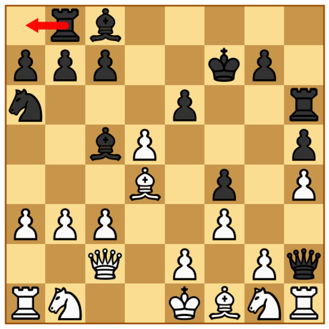
Moves

Rook, bishop, and king move.

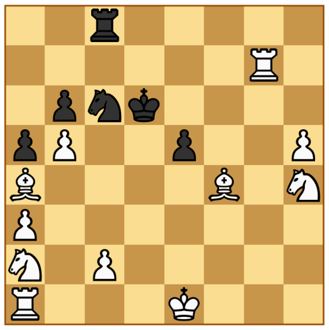
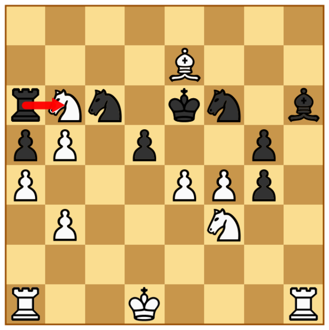




Rook, king, and knight move.

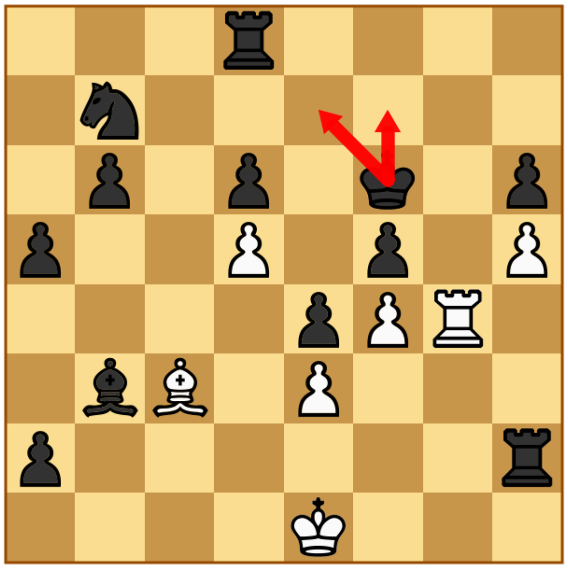
King and rook move.

Capture moves

Rook capture.

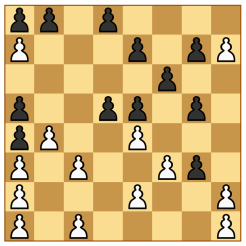
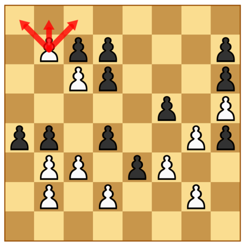
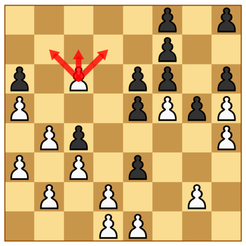
Winning



Draw.

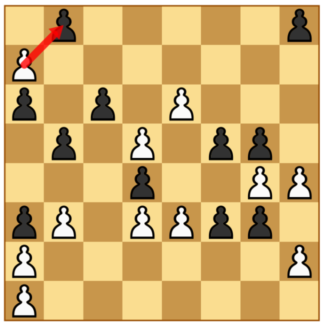
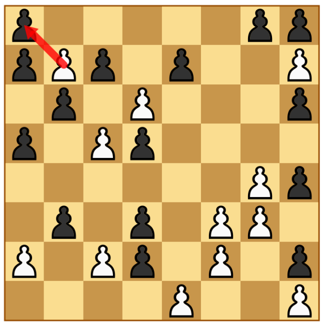
Appendix C: Breakthrough Tutorial (Ludii AI)

moves

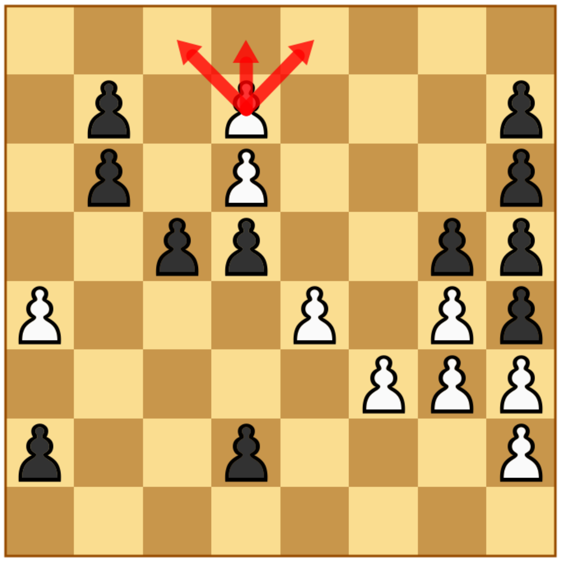
Diagonal move up-left, vertical move up, and diagonal move up-right.

capture moves

Diagonal capture up-right, and diagonal capture up-left.

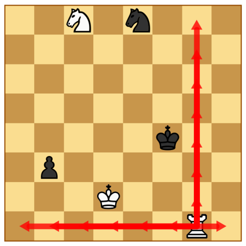
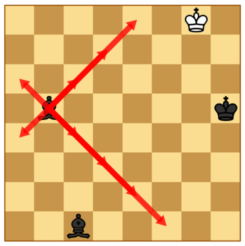
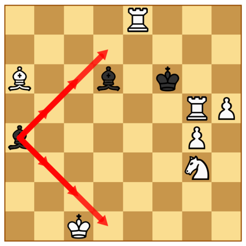
winning



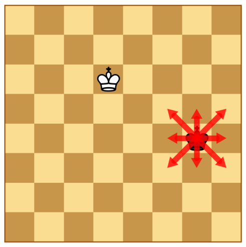
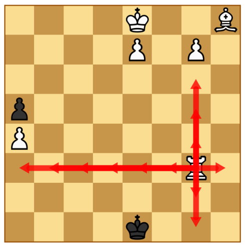
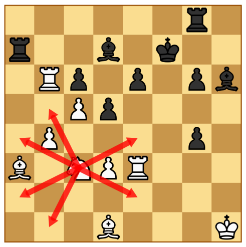
White player win.

Appendix D: Chess Tutorial (Ludii AI)

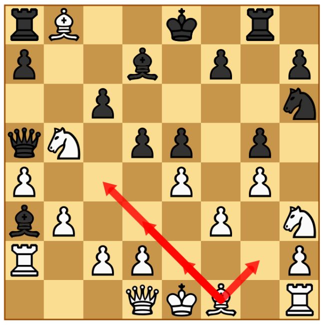
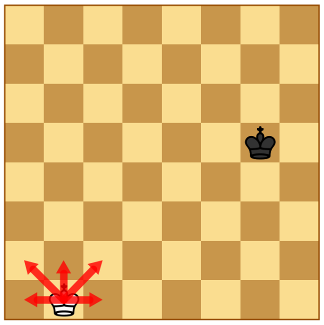
Moves

Rook and bishop move.

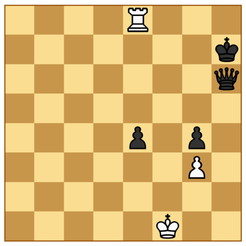
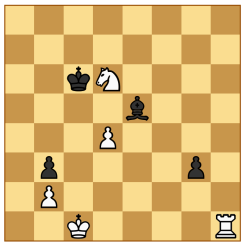
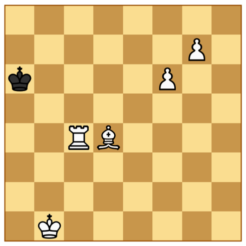




King, rook, and knight move.

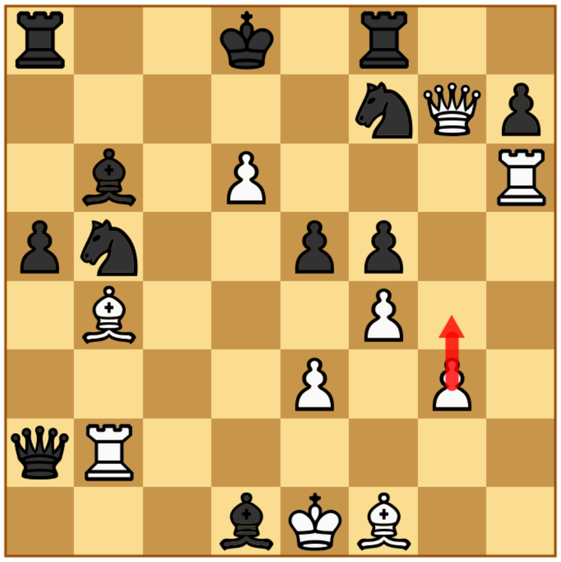



bishop and king move.

Capture moves

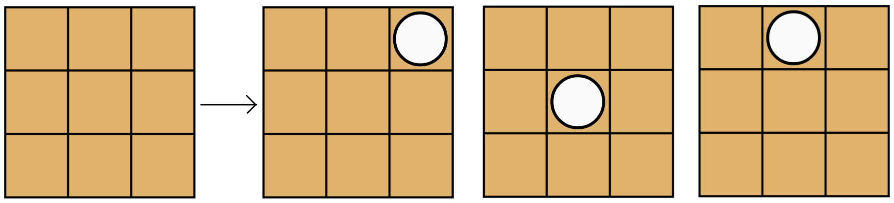
Winning



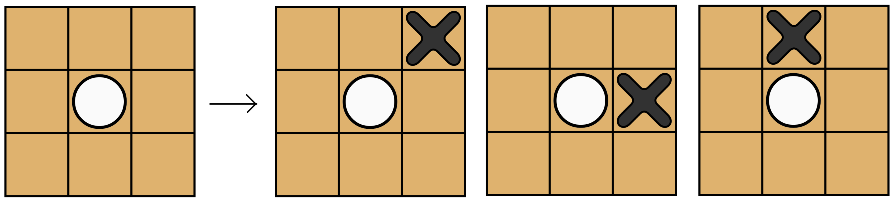
Draw.

Appendix E: Tic-tac-toe Tutorial

Placement

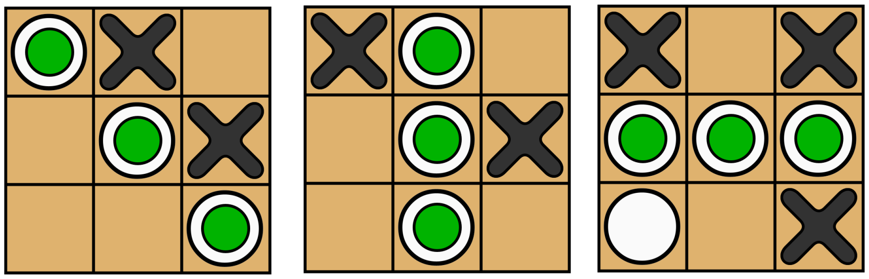


White piece placement



Black piece placement

Winning



White win.