

École polytechnique de Louvain

Graphical Game Design Interface: Game Board and Setup

Author: **Sébastien VERCAMMEN**

Supervisor: **Eric PIETTE**

Readers: **Hélène VERHAEGHE, Achille MORENVILLE, Alice BURLATS**

Academic year 2025–2026

Master [120] in Computer Science and Engineering

CONTENTS

Acknowledgments	v
1 Introduction	1
2 Background	5
2.1 General Game Playing	5
2.2 General Game System	5
2.3 Game Description Formalisms	6
2.3.1 Game Description Language	6
2.3.2 Ludii's ludemic approach.	8
3 Implementation	11
3.1 Game structure.	11
3.2 Equipments	12
3.2.1 Boards	13
3.2.2 Pieces	16
3.3 Rules.	17
4 Interface	21
4.1 Selection of the graphic library.	21
4.1.1 SWT/JFace	21
4.1.2 Swing	21
4.2 Boardmaker	22
4.2.1 Data classes	22
4.2.2 Views	25
5 Results	33
5.1 Board creation as validation	33
5.2 User study	34
5.3 Limitations.	35
6 Future Work	37
6.1 Rule creation.	37
6.2 Feature completion.	37
7 Conclusion	39
A Appendix	41
A.1 Validation games description.	41
A.1.1 ConHex	41
A.1.2 Bam Blang Beh Khla	42
A.1.3 Hund Efter Hare	43

A.1.4	Halusa	44
A.1.5	Surakarta.	44
Bibliography		47

ACKNOWLEDGMENTS

I would first like to thank my supervisor Eric Piette for his advices and guidance through this work.

I would also like to thank the members of my jury Hélène Verhaeghe, Achille Morenville and Alice Burlats for accepting to be part of my jury and for the time taken to read the present work.

The Claude Sonnet tool was used during this work to help me with rephrasing as well as translations and spell check.

1

INTRODUCTION

Since their creation, games have been played by humans all around the world [1][2], mainly as a leisure activity, but some civilizations used primitive form of games as a way to read and interpret oracles from the local divine entities [3]. In our modern days, their usage has spread out into different domains:

- **Artificial Intelligence (AI):** with the emergence of AI, games, whether ancient or contemporary, have proven to be quite efficient testbeds and have lead to major advances in the field [4][5]. We can take for example Deep Blue¹[6], who has beaten the reigning chess world champion Gary Kasparov in 1997, or more recently AlphaGo², defeating professional Go player Lee Sedol³ in 2016 and whose algorithm can be applied to other domains than Go, like economy or online security assessment as explained in [7].
- **Education:** games can be an amazing tool for schools to teach and practice various concepts; some concrete like game design principles or AI technique [8], some abstract like cooperation, communication or negotiation [9]. Moreover, [10] shows that games, either board games or video games, used for a learning activity increases the motivation, engagement and intellectual activity of the learners.
- **Healthcare:** board games, especially abstract strategy games (like Go, Chess or Shogi), have shown an improvement in the condition of adults with a decline of their cognitive functions, but also of patients suffering ADHD, Alzheimer or conditions such as anxiety or depression [11][12].
- **Anthropology:** as the concept of game was somewhat shared between civilizations, reconstructing and studying characteristics of ancient games can give us insights on the impact on the cultural landscape of a given civilization through its interactions with others [13][14]. In addition, games, being part of the cultural heritage, can be used in a perspective of "Cultural Memory", which consists of the ways a culture

¹[https://en.wikipedia.org/wiki/Deep_Blue_\(chess_computer\)](https://en.wikipedia.org/wiki/Deep_Blue_(chess_computer))

²<https://en.wikipedia.org/wiki/AlphaGo>

³https://en.wikipedia.org/wiki/Lee_Sedol

interacts with and constructs its past, and to simulate the metaphors of different cultures[15].

Although the majority of ancient board games has disappeared and fell into oblivion, Digital Archaeoludology (DAL) [1], a new field of study, was created with the objectives of reconstruction, modelling and analysis of such games, based on incomplete descriptions and archeological evidences. It was first lead by the Digital Ludeme Project (DLP) [16] and is now mainly explored by its successor the GameTable Network [17].

This marks the introduction of Ludii[18], a tool developed by the DLP to help DAL. Ludii is a General Game System (GGS), meaning that it is able to model and analyze an immense variety of games and allows human or artificial agents to play them. It also possess an interface that let users visualize and play any game implemented. In this system, a game is written as a tree of ludemes, making it human readable, easily understandable and quite simple to implement any kind of supported game. Unfortunately, while it is quite easy and straightforward to create simple games, the code-like structure of the language and its intricacies makes it quite hard to apprehend for new users and makes the creation of more complex games a tedious, time-consuming task. Ludii already possess a graphical interface (see Figure 1.1) that allows to play implemented games, manage AI and to compute various analysis on the games, but it lack tools allowing to easily create or modify games, the only way being through the use of a text editor.

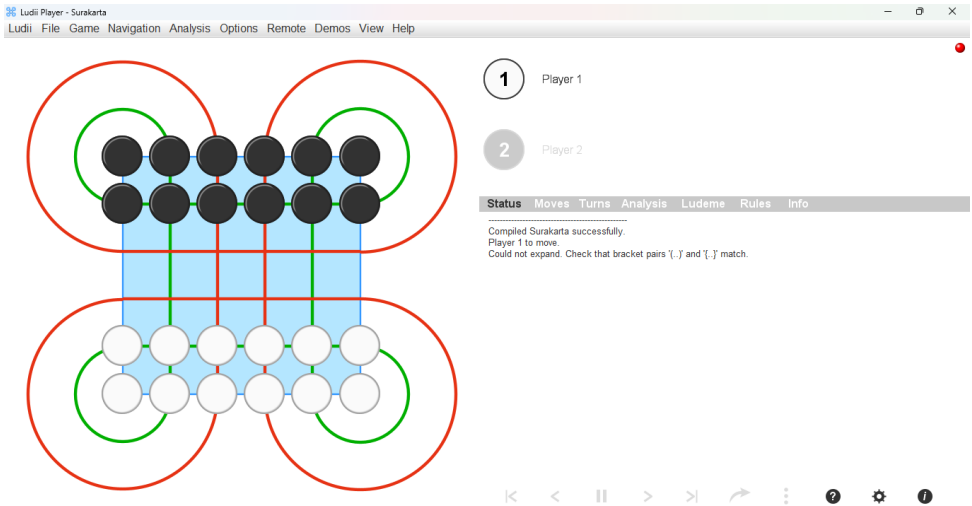


Figure 1.1: Ludii's interface

This thesis marks the first step of a long journey to exploit all the potential and capabilities of the ludemic language. The destination, a Graphical User Interface (GUI) able to create any game that can be described by the language, from the board to the rules, including pieces selection and placement. Having such tool directly inside Ludii would allow :

1. To ease the process of creating game in Ludii. Currently, if someone wants to create a game in Ludii, they have to learn how to write the game description and all the intricacies of such process. With the help of the interface, that person would be able to concentrate solely on the game without having to worry about the description as the interface would handle it for them.
2. To generalize the usage of Ludii in other contexts than DAL or research. Indeed, an easier creation process, as explained in the previous item, could lead to the democratization of Ludii in other domains. With such interface, Ludii could become an amazing tool to be used by game designers to create prototypes of games. But it could also be used in education, by schools in a context of "game-based learning" or by cultural organizations for diverse activities.
3. To help finding errors in currently implemented board related ludemes. Because the interface displays the image of such ludemes and the parameters used, it allows to find combinations of parameters that creates bugs in the generation of graphical elements of a game much easily than by writing then compiling the description. For example, the interface helped finding this description (*board (merge (rotate 180 (shift 0 2 (brick 1))) (scale 2 1 (square 3))) use:Cell*) for which Ludii cannot display the corresponding board.

As this thesis is only the first part of the project, it will only focus on the creation of the board and its setup, which correspond to the selection of pieces and their placement. Hence this work will be organized in the following way : Chapter 2 will present the concepts of General Game Playing, their systems and the formalisms used to describe games, Chapter 3 will overview elements of Ludii needed for this works, Chapter 4 will present the interface while Chapter 5 will discuss the results obtained through different tests and finally, Chapter 6 will present potential future work related to this thesis.

2

BACKGROUND

This chapter will introduce the concepts behind GGS and General Game Playing[19] (GGP).

The field of GGP has emerged as an important area concerning AI research, which lead to the need for systems providing necessary infrastructures to model diverse game and to allow humans as well as artificial agents to play them. Such systems are commonly known as GGS.

2.1 GENERAL GAME PLAYING

The field of GGP is an area of AI that aims at developing agents that are capable to play effectively any kind of games with no other information than the rules which are given at runtime[19][20]. In other words, the objective is to create agents with capabilities similar to us, humans, who can play any new games just by being told the rules.

In opposition to specialized game agents, such as the previously mentioned Deep Blue or AlphaGo, and more recently with AlphaZero[21] and MuZero[22] that, despite being able to play different games, still needs to be trained on the targeted tasks and therefore cannot be considered as GGP, General Game Players provide more value in the domain of AI. Indeed, unlike their specialized counterparts that can be tailored for specific tasks and train on them, GGP agents can adapt to changes in the environment, in our case the rules of the game, and still be effective even though not as much as the specialized agents. Such advancements could be used in domains where there is a real need to adapt rapidly and efficiently to environment changes like autonomous driving or robotics.

2.2 GENERAL GAME SYSTEM

Despite being versatile players, GGP agents need the appropriate playground. This playground is provided by General Game Systems, pieces of software that provides the necessary infrastructure to General Game Players.

One of the main functionality of such system is to model the rules of a game into a formal specification called a game description, a standardization of the game logic to be interpreted by the agent. Depending on the system, this formalism can completely change, from formal logic as used for GGP Base¹ to a completely new language based on the concept of ludemes, as it is the case for Ludii.

On top of that, GGS are able to simulate the game environment, switch between different agents and can sometimes provide a model of the game and its state for the agents.

2.3 GAME DESCRIPTION FORMALISMS

In response to the progress made in GGP, different formalisms have been developed through the years. While differing in scope, expressive power and design principles, they all share the objective of creating declarative descriptions of the game rules for General Game Players.

2.3.1 GAME DESCRIPTION LANGUAGE

Created in 2005 by the Stanford Logic Group², Game Description Language [20] is a generic language that aims at representing any finite board game with perfect information[23][24] (e.g, Chess, Go, Shogi). It has been extended by GDL-II[25] in 2011 to handle games with partial information and stochastic actions (e.g, Poker, Backgammon), and by GDL-III[26] in 2017 to support epistemic games.

With this language, every game element is written in a language inspired by Prolog that relies on first-order logical clauses. This allows the language to be universal and represent any board game[23]. Figure 2.1a shows the GDL description of the game Tic-Tac-Toe.

Figure 2.1a is an illustration that the possibilities the language offers us comes with a cost, and that cost is particularly high when it comes to designing games with GDL. The rest of the section will explain these drawbacks based on the example of the game Tic-Tac-Toe.

First, the core elements of the game such as the board, the legal moves, or the win conditions, must be declared from scratch for every game. In our example, lines 33 to 40 defines what is considered a line for the game, while lines 18 to 22 defines what are the legal moves for every player.

Secondly, as everything must be declared from scratch, it implies that game concepts cannot be reused. Let us consider that we wanted to make a game similar as Tic-Tac-Toe but that plays on 4 by 4 board contrary to the usual 3 by 3 and that we would allow players to form lines of different length. To make these changes we would have to rewrite all the description of our example and we would have to adapt the definition of the board and add the definitions of a line of length 4.

¹<https://github.com/ggp-org/ggp-base>

²<https://ginapp.stanford.edu/overview.html>

```

1 role(x)
2 role(o)
3
4 base(cell(M,N,x)) :- index(M) & index(N)
5 base(cell(M,N,o)) :- index(M) & index(N)
6 base(cell(M,N,b)) :- index(M) & index(N)
7 base(control(R)) :- role(R)
8
9 action(mark(M,N)) :- index(M) & index(N)
10
11 index(1)
12 index(2)
13 index(3)
14
15 init(cell(M,N,b)) :- index(M) & index(N)
16 init(control(x))
17
18 legal(mark(M,N)) :- cell(M,N,b)
19
20 mark(M,N) :: control(R) ==> cell(M,N,R) & ~cell(M,N,b)
21 mark(M,N) :: control(x) ==> ~control(x) & control(o)
22 mark(M,N) :: control(o) ==> ~control(o) & control(x)
23
24 goal(x,100) :- line(x) & ~line(o)
25 goal(x,50) :- line(x) & line(o)
26 goal(x,50) :- ~line(x) & ~line(o)
27 goal(x,0) :- ~line(x) & line(o)
28 goal(o,100) :- ~line(x) & line(o)
29 goal(o,50) :- line(x) & line(o)
30 goal(o,50) :- ~line(x) & ~line(o)
31 goal(o,0) :- line(x) & ~line(o)
32
33 row(M,X) :- cell(M,1,X) & cell(M,2,X) & cell(M,3,X)
34 col(N,X) :- cell(1,N,X) & cell(2,N,X) & cell(3,N,X)
35 diag(X) :- cell(1,1,X) & cell(2,2,X) & cell(3,3,X)
36 diag(X) :- cell(1,3,X) & cell(2,2,X) & cell(3,1,X)
37
38 line(X) :- row(M,X)
39 line(X) :- col(N,X)
40 line(X) :- diag(X)
41
42 terminal :- line(x)
43 terminal :- line(o)
44 terminal :- ~open
45
46 open :- cell(M,N,b)

```

(a) GDL description of Tic-Tac-Toe

```

1 (game "Tic-Tac-Toe"
2   (players 2)
3   (equipment {
4     (board (square 3))
5     (piece "Disc" P1)
6     (piece "Cross" P2)
7   })
8   (rules
9     (play (move Add (to (sites Empty))))
10    (end ("Line3Win")))
11 )
12 )

```

(b) Ludii's description of Tic-Tac-Toe

Figure 2.1: Comparison of the descriptions of Tic-Tac-Toe in different formalisms

Thirdly, because of the use of first-order logic, many of the game elements are interconnected within the description. Consider the case of the addition of a third player in our example, lines 4 to 7 have to be changed to support the new player, lines 20 to 22 must be modified to add this third player's moves and many more changes are required to fully integrate this new player in the game.

In the end, all these defaults of GDL makes the descriptions very verbose and repetitive, as well as hard to decipher for people not akin with the language.

Even though being the only one available for a long time and therefore used for academic research, this formalism has lead to many advances in the field, notably for the Monte Carlo Tree Search (MCTS) algorithm and rule representation[27]. However, the difficulty to describe new games has highlighted that a simpler and more suitable formalism to describe games was necessary.

2.3.2 LUDII'S LUDEMIC APPROACH

Ludii's language, in opposition to Stanford's GDL, relies on a completely different concept that has already been named in this work but not clearly defined, ludemes.

A ludeme[18][28] is a conceptual unit containing game-related information. Whether it is the form of the board, the number of players, the type of the pieces used and how they move, or the conditions to end the game, every characteristic that defines a given game is considered a ludeme. Every ludeme is mapped to a java class. These classes are the key elements of a class grammar used to compile any game description. Once the game is compiled, generates a generalised game representation which can be accessed by different modules to retrieve necessary information. For example, the AI module can retrieve the list of legal moves, while the Graphic module retrieves the image of the board.

Figure 2.1b represents the description of the game Tic-Tac-Toe within Ludii and will serve to illustrate the advantages of ludemes for game description over GDL.

Firstly, as ludemes encapsulate game concepts, they can be given meaningful labels, which in turn makes the description of the game easily readable and understandable while allowing more advanced analysis in the context of DAL such as establishing relationships between games and their components to model the evolution of games through history and their spread across cultures worldwide[18]. This can be seen in the example at line 9 that the only move allowed is for a player to add one of its piece to an empty site.

Secondly, the usage of ludemes makes the different elements of the game "independent" in such a way that we only need to modify a few lines to modify the game. For example, if we wanted to change the number of player we would only have to change line 2 and add the corresponding piece, or to change the size of the board, the only modification needed is on line 4.

Thirdly, again because of the encapsulation of concepts, once a ludeme is supported

by Ludii, it can be reused indefinitely for any game without having to redefine it. For example, if we want to create a completely different game that has the same end condition as Tic-Tac-Toe, we would only have to reuse line 10.

In the end, the usage of such formalism produces shorter and more understandable game descriptions than GDL.

3

3

IMPLEMENTATION

3.1 GAME STRUCTURE

As you have seen in Figure 2.1b, a game description in Ludii's language is centered around a main ludeme, the "Game" ludeme. It serves as the root of the tree of ludeme defining the game and is composed of 4 ludemes[29][30] (see Figure 3.1):

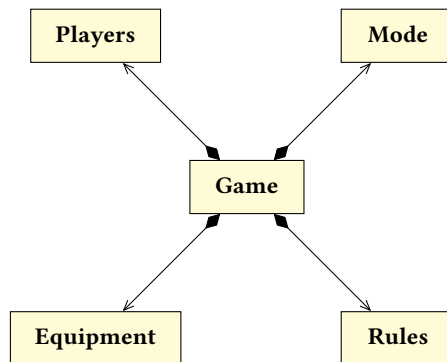


Figure 3.1: Game components

- **Players:** provides information about the player of the game, especially their number. Ludii currently supports games with up to 16 players. For every game, an additional hidden player, Player 0, is added to represent the randomness of game elements like flipping a coin, throwing a dice, etc.
- **Mode:** describes the control flow of the game. There's 3 modes available: "Alternating", "Simultaneous", and "Simulation" corresponding respectively to players playing one after each other, at the same time, or for games with no players such as the famous "Game of life"¹. By default if the ludeme is not present in the description, the game mode will be set to "Alternating".

¹https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

- **Equipment:** defines the different items used for the game. Among them, only the "Board" and "Piece" ludemes are relevant for the present work and will be developed in Section 3.2.
- **Rules:** dictates the behavior of the game. This ludeme establish the initial state of the game, the legal moves of the current player, and the conditions to end the game and the outcome. The different types of rule will be discussed in Section 3.3.

In addition to the "Game" ludeme, the game description can contain another ludeme called "Metadata", containing additional information about the game that are not related to how it is played.

3

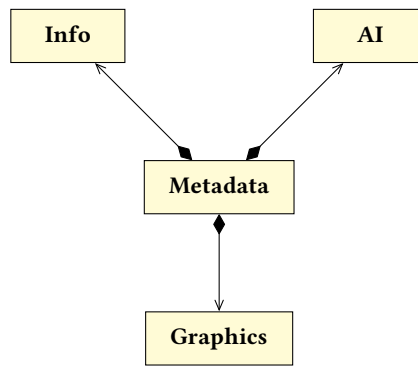


Figure 3.2: Metadata components

Figure 3.2 shows the 3 components that are part of the metadata. The "Info" ludeme contains general information about the game such as the creator, a textual description of the game and its rules, historical information and so on. The "AI" ludeme provides information to be used by the AI playing the game. It can tell for example which AI performs the best on this game as well as provide heuristics to increase its performance on this game. Finally the "Graphics" ludeme is used to provide information about the visualization of the game. It can be used to change the color of cells or pieces, modify the size of pieces, etc.

3.2 EQUIPMENTS

Ludii provides different items to be used as an equipment (see Figure 3.3), and the majority of them can be split between 2 categories: the containers and the components.

A container (see Figure 3.4) is a structure that holds the playable sites of the game. There exist 4 types of container in Ludii but every game within Ludii has at least 1 board. Its purpose is to provide the different playable sites on which to place the different items to be used by the players during the game. For games that doesn't require boards like card games, it can be hidden with the aforementioned graphics metadata to give players the illusion of the absence of board.

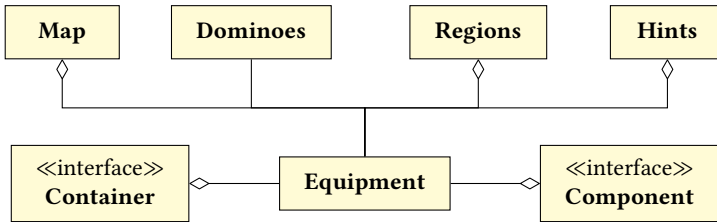


Figure 3.3: Available equipments

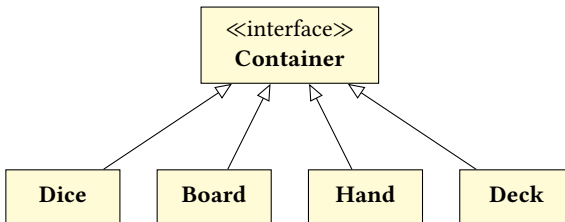


Figure 3.4: Available Containers

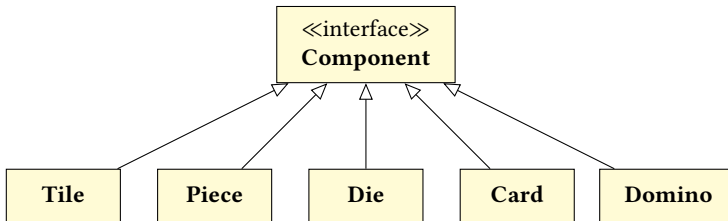


Figure 3.5: Available Components

A component (see Figure 3.5) is an item that can be manipulated by the players during the game.

As the goal of this thesis is to build an interface able to create a board and its initial setup, we will only focus on the board container and the piece component.

3.2.1 BOARDS

In Ludii, boards, just like any other container, are internally modeled by a graph[31]. This graph is defined by a set V of vertices, a set E of edges and a set C of cells, also called faces. All of these graph elements denote playable sites to be used by the players during the game.

Figure 3.6 shows 2 games played on different type of sites. As all graph elements can be used to place a component, Ludii uses a default type of site whenever we refer to them. This type is by default the site type defined when the board was created and in the case this type is not defined, the default value is set to Cell.

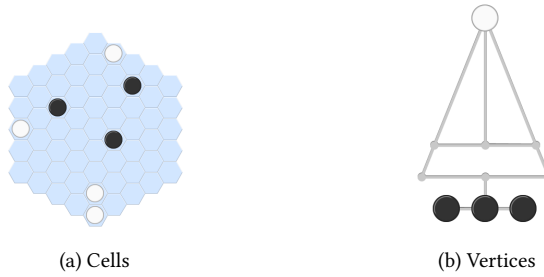


Figure 3.6: Games played on different sites

GRAPH FUNCTIONS

In order to build the boards, Ludii uses graph functions. As the name suggests, the role of such function is to create the graph of the board by computing the Cartesian coordinates of each vertex, edge and faces. These functions are separated in 2 categories: base graph functions and graph operator functions.

Base graph functions are used to create the initial graph of the board. Graphs created this way are raw and free of any transformation. Boards issued from such function will be called basic boards throughout this work.

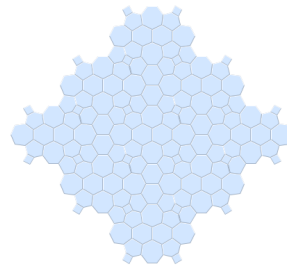
Graph operator functions, for their part, are used to transform and modify other graph functions, whether they are base graph or other graph operator functions. This is done by modifying the graph to reflect the operation.

```

1 (board
2   (dual
3     (subdivide
4       (dual
5         (subdivide
6           (tiling T33434 3 1)
7             min:4)
8         )
9       min:3)
10   )
11 )

```

(a) Description



(b) Graph

Figure 3.7: Example of a complex graph

The combinations of these functions provides an easy and quick way to create various board topologies from a simple description. Figure 3.7 shows a visually interesting graph made that way.

OTHER BOARDS

In addition to boards created with the usual "Board" ludeme, Ludii supports 3 other types of boards.

"Boardless" which creates a board whose size is extended during the game, Carcassonne² is an example of such game. Clémentine Sacré's thesis aimed to upgrade the representation of such board in Ludii[32]. This type of board hasn't been considered for this work.

3

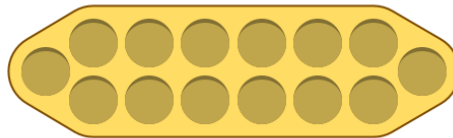
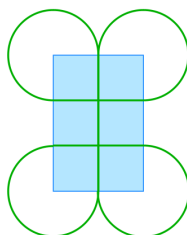
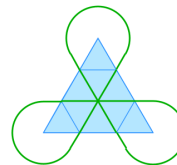


Figure 3.8: A mancala board

"MancalaBoard" (see Figure 3.8) defines the board of games belonging to the Mancala family of games. These boards are always composed of rows of holes, with possible holes on the left and right sides of the board.



(a) Rectangular



(b) Triangular

Figure 3.9: Surakarta boards

²[https://en.wikipedia.org/wiki/Carcassonne_\(board_game\)](https://en.wikipedia.org/wiki/Carcassonne_(board_game))

Finally, "SurakartaBoard" creates a board similar to the one used in the Surakarta game³ for rectangular and triangular boards (see Figure 3.9). Boards belonging to this family come with inherent tracks and this ludeme has been created to simplify the description of games using this kind of board.

3.2.2 PIECES

Pieces are the most used component in Ludii. They are mainly created based on their name, which can be denoted as their global name, and their owner with the usage of the "Piece" ludeme. The name is used to identify the different pieces and the image to display for each piece. The owner parameter is used to determine which player owns the piece. Other parameters can be given to specify the direction the piece is facing or the moves it can do.

3

Owners	Description
Each	Each player possess its own version of the component.
Shared	The component is owned by all the players.
Neutral	The component is owned by no player.
PX	Only player X owns this component.

Table 3.1: Owners

Table 3.1 shows the most common values for the owner parameter. When a piece is created, the system creates a number of components depending on this value. "Each" means that a piece is created for each player of the game, "Shared" means that a single piece is created and that piece is owned by all the players, "Neutral" creates a single piece not owned by any player, and "PX" creates a single piece specifically for the player X. These pieces are then internally named as the concatenation of its name parameter and the id of its owner, so the id of the player for "Each" and "PX", 0 for "Neutral" and no id for "Shared".

1 (piece "Ball" Each)

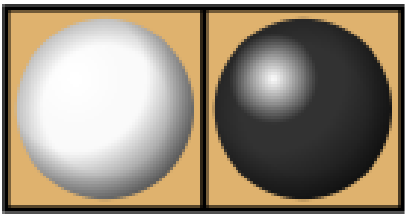


Figure 3.10: Piece creation in a 2 player game

Figure 3.10 shows that a piece "Ball" owned by each of the player is created. As each player have their own copy of the piece, 2 components are created if there is 2 players.

³[https://en.wikipedia.org/wiki/Surakarta_\(game\)](https://en.wikipedia.org/wiki/Surakarta_(game))

Player 1 owns the white piece called "Ball1" while the other player owns the black piece called "Ball2".

3.3 RULES

As previously said, the "Rules" ludeme is responsible for the behavior and the logic of the game. When creating a game, its rules can be divided into 4 categories (see Figure 3.11). While only the starting rules play a part in the setup of the game, basic knowledge about the other rules is necessary to have a complete understanding of a game description.

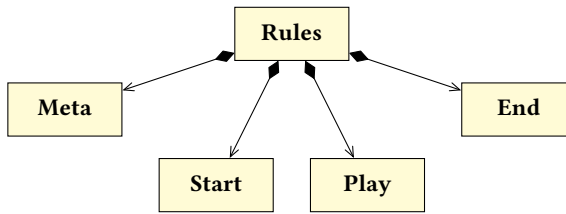


Figure 3.11: Rule types

META RULES

A meta rule is an optional type of rule that is applied at each turn, whatever the conditions. Among them, some are able to modify the state of the game or add/remove certain moves from the list of legal moves.

START RULES

Start rules are a set of optional rules that are applied just after the compilation of the game, but before its beginning. These rules creates a list of moves that are applied on a game state with no piece placed and all the variables set at their default. This list is then used to create the initial state of the game by placing the pieces or setting the variables to their targeted value. Among those rules, the mostly used is the "Place" ludeme that is used to place a piece on one or multiple cells.

Figure 3.12 shows an example of Start rules, with the description of these rules on Figure 3.12a and the corresponding initial state on Figure 3.12b. On the example, line 3 places a piece called "Dog" owned by the player 1 on the sites corresponding to the bottom row of the board while line 4 places a piece called "Cat" owned by the player 2 on the sites corresponding to coordinates "A3", "C4" and "D2".

PLAY RULES

The playing rules defines the available legal moves of the mover for the current turn. It is a mandatory ludeme that, if not present, prevents the game to compile. This ludeme is evaluated at the beginning of each turn and returns a list of legal moves. These moves are defined with the use of the "Move" ludeme either in the "Play" ludeme of the rules or when creating a piece within the "Piece" ludeme. Moves created in such a way are only legal for

```

1 (rules
2   (start {
3     (place "Dog1" (sites bottom))
4     (place "Cat2" {"A3" "C4"
5               "D2"})
6   })
7 )

```

(a) Start rules



(b) Initial setup

Figure 3.12: Illustration of the "Place" starting rule

the given type of pieces.

Figure 3.13 shows an example of the Play rule for the game breakthrough (Figure 3.13a) side by side with a picture of the available legal moves for the selected piece (Figure 3.13b). Breakthrough is a game where players can only move their pieces forward vertically or diagonally and a piece can only capture another piece diagonally. The game ends when a player reaches the opponent's edge of the board or capture all of its pieces. Therefore, the Play rule in Figure 3.13a works this way: it only uses inherent moves from the pieces as legal moves. These moves are defined from line 9 to line 11. Line 9 defines the move that allows a piece to move forward if the site in front of it is empty. Lines 10 and 11 defines the move allowing a piece to move diagonally or possibly capture a piece.

END RULES

The End rule describes the conditions required to terminate a play, as well as the outcome for at least 1 player. It is the second type of rule to be mandatory. It is evaluated at the end of each turn. For games with 2 players or less, this rule describes a terminal game state and the game stops; but for games with more than 2 players, the game can continue if the play is not over for at least 2 players. For the outcome, it is possible to directly specify it by using the ludeme "Result" or it is possible to rank players according to a given score via the ludeme "ByScore". When no players can make a move and the conditions of the End rule are not met, the game ends on a draw.

Figure 3.14 shows an example of the End rule for the game Tic-Tac-Toe (Figure 3.14a) next to a picture of a terminal state (Figure 3.14b). As you probably know, the game Tic-Tac-Toe ends when a player manages to align 3 of its pieces in a line or when the grid is complete. The End rule of Figure 3.14a correspond to this behavior as at the end of each turn, the system checks if there is a line of size 3 formed by the pieces of the player. If it is

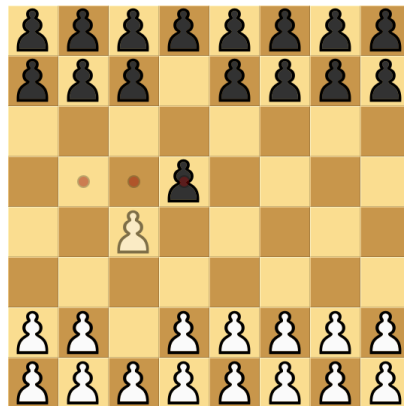
```

1 (game "Breakthrough"
2   (players { (player N) (player S) })
3   (equipment {
4     ...
5     (piece "Pawn"
6       Each
7       (or
8         {
9           (move Step Forward (to if: (is Empty (to))))
10          (move Step (directions { FR FL })
11            (to if: (or (is Empty (to)) (is Enemy (who at: (to)))) (apply
12              (remove (to))))
13          )
14        }
15      )
16    )
17  )
18  (rules
19    ...
20    (play (forEach Piece))
21    ...
22  )
23 )

```

3

(a) Description



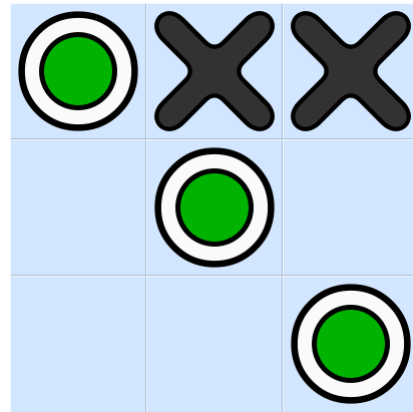
(b) Available moves for the selected Pawn

Figure 3.13: Illustration of the playing rules in the game Breakthrough.

the case, the game ends and the current player, alias the Mover, wins the game. If not, the game resume.

```
1 (end
2   (if (is Line 3)
3       (result Mover Win)
4   )
5 )
```

(a) Tic-Tac-Toe end rule



(b) End state of Tic-Tac-Toe

Figure 3.14: Illustration of the end rule in the game Tic-Tac-Toe

4

INTERFACE

4

4.1 SELECTION OF THE GRAPHIC LIBRARY

Before delving into the presentation the interface, a quick explanation of the choice of the graphic library is needed as this choice will have a major impact on the interface. Indeed, different graphic libraries doesn't deliver the same performance and doesn't offer the same features. Some graphic libraries can present different implementations for the same graphic component, while some components can simply not be supported by some libraries.

This section therefore presents the reasons behind the choice of the graphic library.

4.1.1 SWT/JFACE

The SWT/JFace[33] framework is a combination of graphic libraries for developping GUIs in Java-based applications. While SWT gives access to the native operating system UI widgets and event systems, JFace provides a higher level of abstraction for SWT patterns, particularly for event handling and component management.

The most prominent aspect of SWT/JFace is the usage of heavyweight component; i.e., the native UI components of the underlying platform. This approach makes SWT/JFace applications visually consistent with the host platform's user interface while achieving performance similar to native applications. In other words, such application, when run on Windows, will look like a native Windows application, but will look like a native macOS application when run on a Mac.

Another key characteristic of this framework is that it doesn't rely on automatic garbage collection. It instead uses a semi-automatic garbage collection where any created resource must be explicitly disposed of but when a parent resource is freed, all of its child resources will automatically be freed.

4.1.2 SWING

Swing[34] is a graphic library integrated within Java and the library used for the current interface of Ludii. Built on top of AWT, the initial graphic library of Java, Swing provides a

more complete and sophisticated set of GUI component but still extensively rely on AWT's event-handling mechanism. However, the library is nowadays only maintained but not actively developed.

In opposition to SWT/JFace and even AWT, Swing doesn't rely on heavyweight components but rather on lightweight components entirely implemented in Java. This design makes these components look and behave identically regardless of the operating system. It also allows for the modification at runtime of the look-and-feel (i.e. the appearance and behavior) of each components.

The other difference with SWT/JFace is that Swing takes advantage of Java's garbage collector. While automatic garbage collection can be difficult and error-prone when dealing with low-level resources across different platforms, it isn't a large concern for Swing's high level components[33].

Based on these 2 presentations, Swing seemed to be the most adapted graphic library for the present work. Despite the increased performance that SWT/JFace could offer, the relatively low complexity of the Boardmaker interface shouldn't be an issue. Furthermore, Swing's ease of use due to the automatic resource management given by Java's garbage collector and its previous use in Ludii's interface reinforced the choice.

4.2 BOARDMAKER

The present section will provide a thorough explanation of the interface created for this work called the Boardmaker.

The first major design choice was to define the amount of information to show to the user. Indeed, too little information would make the GUI harder to use while too many unnecessary information would confuse the user[35]. This lead to the idea to split the different functionalities between different views, with each view taking care of its own functionality. This allows to reduce the amount of information present on the screen at one time.

The second major design choice concerned the layout of the different views. Inspired by the current interface of Ludii and the interface of the IntelliJ IDE, each view of the Boardmaker was split in multiple area regrouping related elements.

4.2.1 DATA CLASSES

The first part of this section will present the different classes implemented to efficiently store and easily retrieve all the data required to create a board and its setup.

GRAPHINFO

One of the main concern of the Boardmaker interface is the ability to remember and distinguish all the different graph functions used in the creation of a board. A solution could be to use Ludii's graph function classes but upon closer inspection it becomes clear that this solution is not adapted to the interface. Indeed, the majority of the base graph

functions are intermediate class.

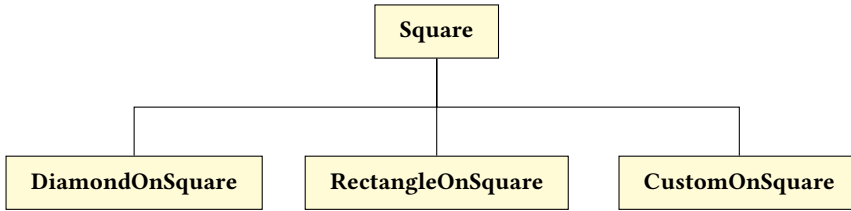


Figure 4.1: Square graph function intermediate class

Figure 4.1 shows the case of the "Square" graph function. When a game using a board based on the "Square" ludeme is created, the corresponding class will act as an intermediate and call the appropriate class based on a certain parameter, generally the shape. For example, if the description was (*board (square Diamond 3)*), the Square class would delegate the creation of the graph to the DiamondOnSquare class.

This problem has been solved with the creation of the GraphInfo interface. Every graph function, base or operator, can be represented by a subclass of GraphInfo. These class, in addition to the parameters related to the graph function they represent, hold the object creating the graph. To illustrate, let's use the example of the previous paragraph. With this system, the Boardmaker creates an object of the class SquareInfo that stores all the parameters of the ludeme as well as the DiamondOnSquare object that will be used to create the graph of the board.

In order to generate the description in the ludemic language, every subclass of GraphInfo implements the *description()* function that returns the description of the graph function.

CONTAINERINFO

The ContainerInfo interface is used to manipulate and store various data about a board. Every type of board mentioned in Section 3.2.1 is represented by one of its subclasses as shown in Figure 4.2.

Among the information related to the board like the GraphInfo of the graph function in the case of BoardInfo or the parameters for MancalaInfo and SurakartaInfo, a ContainerInfo also contains data about the image of the board. The most important being its SVG representation that allows the interface to quickly recompute the image.

In the same way as the GraphInfo interface, ContainerInfo also implements a *description()* function that generates the description of the board.

PIECEINFO

All information relative to the pieces of a board such as the pieces available and their placement is stored within the PieceInfo class.

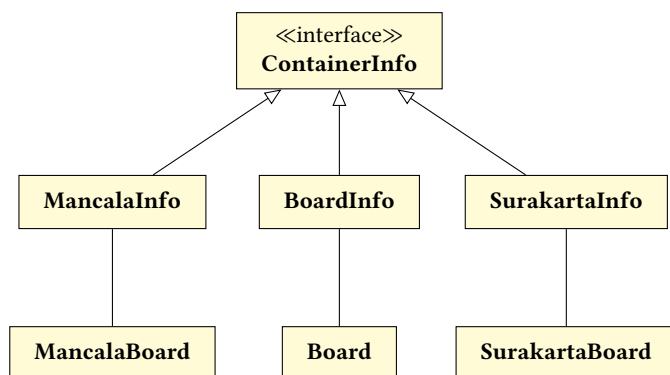


Figure 4.2: ContainerInfo subclasses and the boards they represent

Every piece that is available to the players is stored in the *piecesUsed* variable. It is a list storing Piece objects following the same format as the "Piece" ludeme. In other words, a Piece object stores the global name of a piece and its owner.

As for the placement, sites containing pieces are held in 2 variables having different purposes. The first variable, *piecesPlacedbySite*, is a Hashmap whose keys are the coordinates of the sites and whose values are the internal name of the piece placed on it. An example of such key-value pair could be "A4"-*"Ball1"* for a ball belonging to player 1, as defined in Figure 3.10, placed on site "A4". This map is used to quickly get on which site a piece must be drawn. The second variable is also an Hashmap called *piecesPlacedbyName*. In opposition to *piecePlacedbySite*, this variable maps the internal name of a piece to the list of sites on which it is placed.

The PieceInfo class also provides functions to easily get different parts of the game description. *pieceDescription()* generates the definition of the different pieces for the equipment bloc of the game description and *setupDescription()* generates the starting rules. Those 2 functions make an effective use of *piecesUsed* and *piecesPlacedbyName* respectively.

BOARDMAKERSTATE

The BoardMakerState data class is used to represent the current state of the Boardmaker. It contains the data of all the boards created as well as their associated set of pieces represented as lists of ContainerInfo and PieceInfo respectively called *boards* and *pieces*. In addition, it also holds the index of the currently displayed board.

The usage of lists allows to easily get the necessary information and switch between different boards. Whenever a new board is created, its ContainerInfo is added at the end of *boards*, while an empty PieceInfo is created and added at the end of *pieces* to maintain the link between the 2 items.

4.2.2 VIEWS

This second part will present the different views of the interface as well as the different components that constitute them.

The presentation order of the views has been organized in the same way that an user would encounter them using the Boardmaker interface.

WELCOME VIEW

Upon starting the Boardmaker, the first view the user encounters is the "Welcome view" (see Figure 4.3). This view allows the user to specify some information about the game they want to create: the name of the game, number of players, game mode and the type of sites on which they want to play.

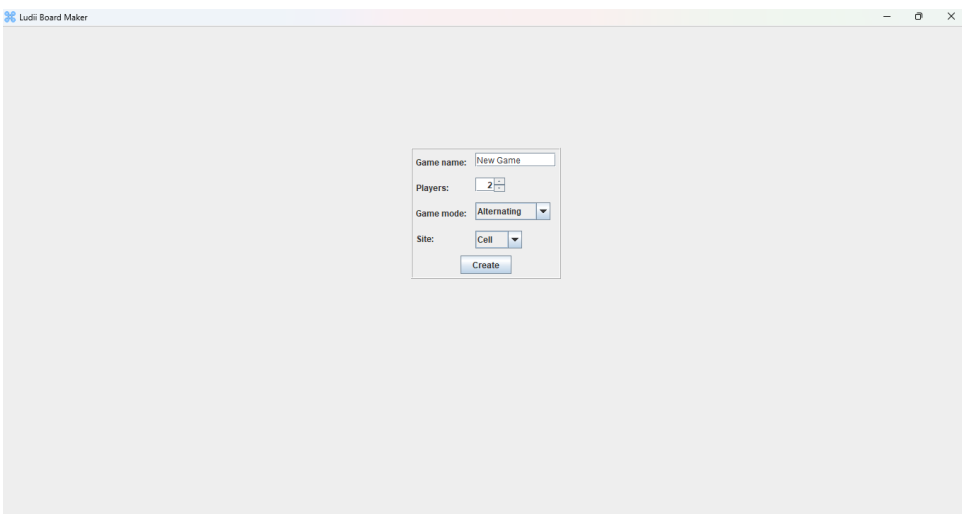


Figure 4.3: Welcome view

Once these information are provided, the user has to click on the *Create* button in order to go to the "Main view" and be able to create the board of its game.

MAIN VIEW

The "Main view" is the core view of the interface. It is from there that all the functionalities are accessible, notably the creation and modification of boards. This view can be split in 3 area depicted in Figure 4.4.

The first area (red area in Figure 4.4) is the "Toolbar". It serves to display the information about game that were provided in the "Welcome view" but also to provide a quick access to certain functionalities. The toolbar is the only component of the GUI to be displayed in all views with the exception of the "Welcome view". It can be decomposed in 5 parts separated by vertical lines as shown in Figure 4.5. From left to right we have:

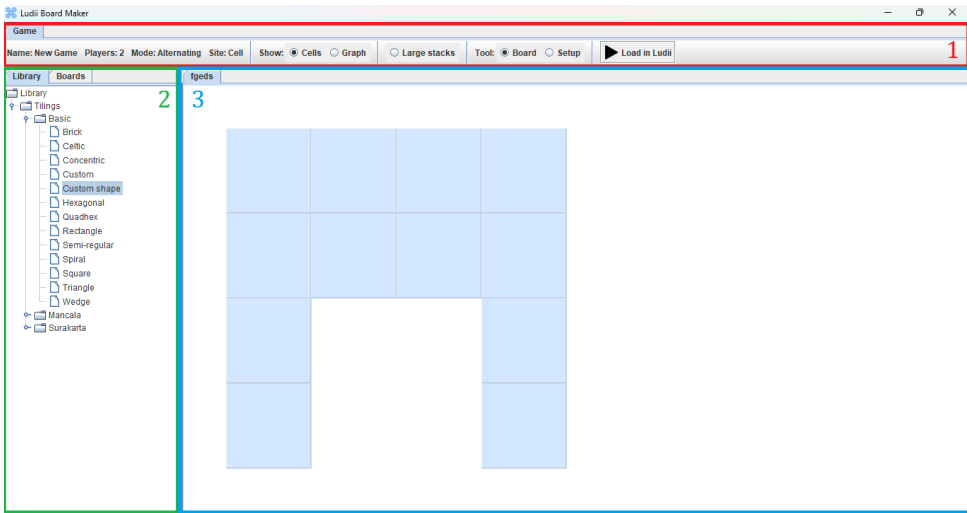


Figure 4.4: Boardmaker's main view. Coloured rectangles mark the **toolbar** (1), the **west area** (2) and the **board area** (3).

- The "Information" part. Its purpose is to display the general information of the game which are its name, number of players, mode and playable sites.
- The "Show" part. This part allows the user to switch the displayed sites of the board. By clicking on the "Cells" button, the cells of the board will be displayed as in Figure 3.6a while the "Graph" button will switch to the graph of the board, thus displaying the vertices and edges as in Figure 3.6b. This has no effect if the board is a mancala or surakarta board as they have a specific design.
- The "Largestack" part. In Ludii, boards tolerate stack of up to 32 pieces by default but the board ludeme has a parameter that allows higher stacks. Even though the Boardmaker doesn't handle the creation of stack of pieces, this part serves to activate or deactivate this property.
- The "Tool" part. This part allows the user to switch between the "Main view" and the "Setup view" where pieces are placed or removed from the board. Clicking on the "Board" button displays the "Main view" while the "Setup" button displays the "Setup view".
- The "Load in Ludii" button. Once this button has been clicked, the interface will generate the file containing the description of the game, open the game in Ludii then close the Boardmaker. It is important to note that the interface will consider the currently selected board and its set of pieces to create the description.

The second area (green area in Figure 4.4) is called the "West area". This area hosts 2 elements of the Boardmaker, the "Library" and the "Board collection".

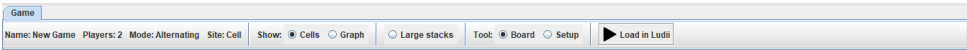


Figure 4.5: Boardmaker's toolbar

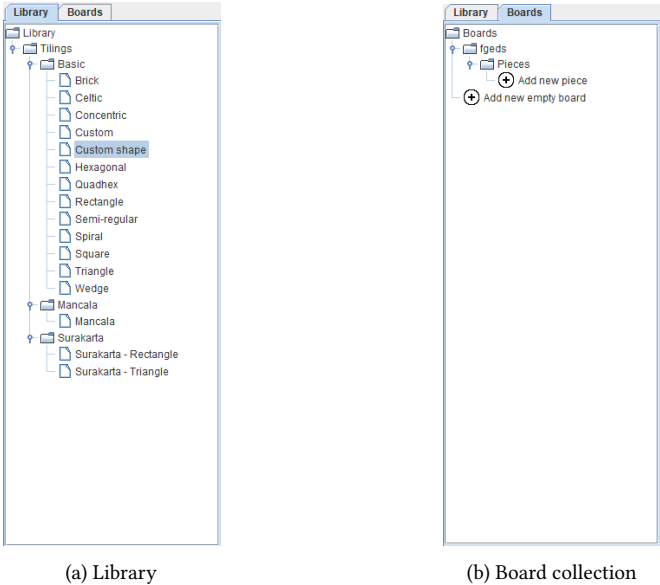


Figure 4.6: Components of the "West area"

The "Library" (see Figure 4.6a) lists all the type of boards available which are the simple board, mancala board and surakarta board, and in the case of the simple board, lists all the base graph function available to the user. By double clicking on one of the item with the left mouse button, the user can create the corresponding board. It is important to note that upon their creation, any new board will override the displayed board so the user must add a new empty board before hand. In that case, the interface will change the view to the "Creation view".

The "Board collection" (see Figure 4.6b) stores all the boards and each set of pieces is stored under the board it is linked to. A double click with the left mouse button on the name of a board will change the selected board. In the case of Figure 4.6b, the clicks would happen on "fgeds". Double left clicks will also trigger an action if they happen on the "Add empty board" or "Add new piece" lines. The former adds a new empty board to the collection and is the only way to create multiple boards; the latter allows to add a new piece to the set it is located under and opens the "Piece dialog".

The "Piece dialog" (see Figure 4.7) is used to create and add a new piece to set of pieces. Available pieces are stored in different categories indicated by the tabs. The image of a piece is displayed on the left of its name. The owner can be selected the drop-down list on

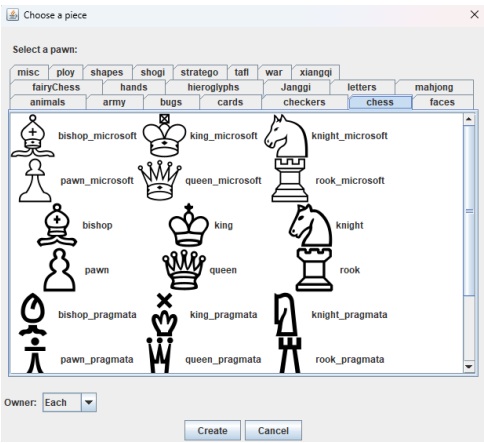


Figure 4.7: Piece dialog

the right of the "Owner" field. To effectively create the piece, the user needs to click on the "Create" button.

The "Board area" (blue box in Figure 4.4) is the third and last area of the "Main view". It is used to display the selected board, which can be identified by its name in the tab located in the upper left corner, and to apply transformations on this board. In order to transform the board, the user can right click on its image and select a transformation function from the popup-menu appearing on the screen (see Figure 4.8). While some transformation are quite straightforward and only require to click on an item of the menu or to enter some value into a dialog, others require direct interaction with the board.

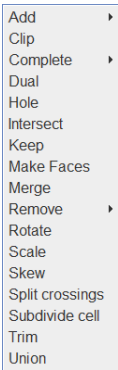


Figure 4.8: Transformation popup-menu

The "Add" transformation allows, as its name says, to add either vertices, edges or cells to the board. This is achieved by simply clicking on the "Board area" to add a vertex, by clicking on 2 vertices to add an edge between them and by placing successive edges

composing a loop to add a cell. As this transformation may require to add an element off-limits of the area, the user has the possibility to move the board around. This can be done by dragging the mouse in the desired direction while pressing its wheel button. An example of addition is shown on Figure 4.9.



Figure 4.9: Example of an addition of cells

4

The "Remove" transformation is the opposite of "Add". It is used to remove an element of the graph. Given the selected type of element, the interface will display its center point that the user can click on to select it. For a vertex, the interface will display the point directly on it, for an edge it will be its middle point while for a cell it will be its center of gravity. It is sometimes necessary to use the "Makefaces" transformation to recompute the faces after a removal. An example of removal is shown on Figure 4.10.



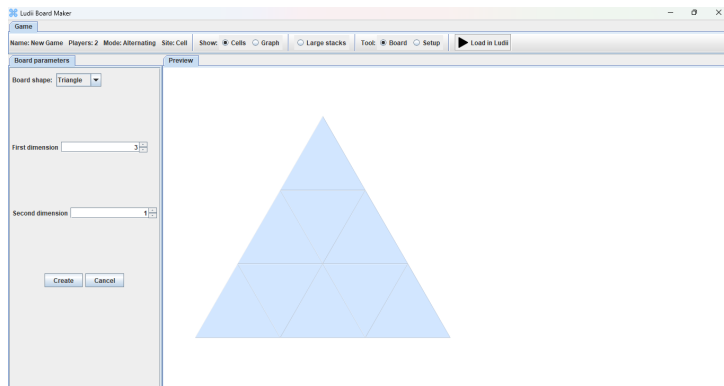
Figure 4.10: Example of a removal of edges

CREATION VIEW

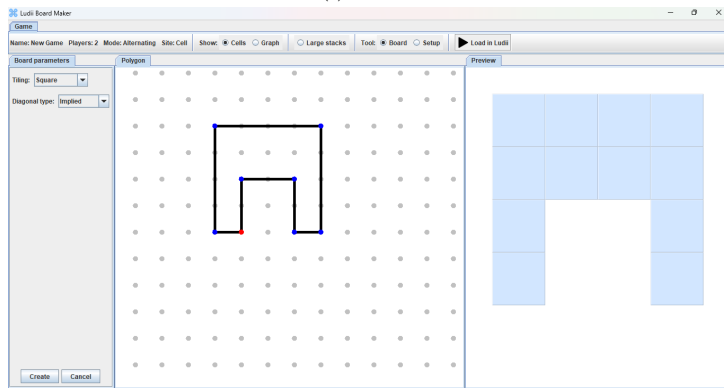
The "Creation view" is the view responsible for the creation of the board. It is from there that the user enters the different parameters necessary to the construction of the graph of the board. The majority of the options present in the "Library" leads to a generic view while "Custom shape" and "Custom" each leads to their own view.

The layout of the generic "Creation view" (see Figure 4.11a) is similar to the "Main view". We still find the "Toolbar" but the "West area" and "Board area" have been replaced by the "Parameter area" and "Preview area".

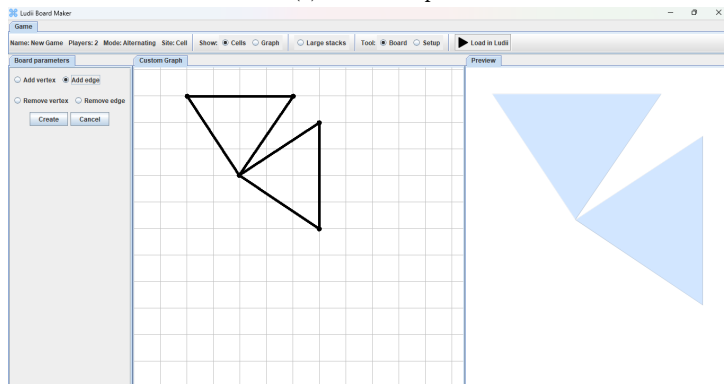
The "Parameter area" is the zone where the user enters the different parameters to create the graph of the board. Each item in the "Library" has their own set of parameters. The "Create" button is used to validate the board. Whether the "Create" or "Cancel" button



(a) Generic



(b) Custom shape



(c) Custom graph

Figure 4.11: Possible "Creation views"

is clicked, the interface will revert to the "Main view" or "Setup view" depending on the selected option in the "Toolbar".

The only purpose of the "Preview area" is to display the image of the board for the current set of parameters in the "Parameter area". The image is recomputed each time a parameter changes.

Figure 4.11b shows the "Creation view" issued from "Custom shape". It includes all the component of the generic "Creation view" with the addition of a dotted grid in its center called the "Polygon area". The role of this area is for the user to draw a polygon that will serves as the shape of the board. A polygon vertex can only be added on a dot of the grid. Each dot is separated by 1 unit in the coordinate system of the board, meaning that the coordinates of a dot on the right side of a dot located in (x, y) are $(x + 1, y)$.

Figure 4.11c illustrates the view issued from "Custom". It includes all the component of the generic "Creation view" with the addition of a grid in its center called the "Graph area". This area lets the user creates a customized graph that will be used to build the board. A vertex or edge can be added or removed by selecting the appropriate option in the "Parameter area" then by clicking somewhere in the "Graph area" in the case of a vertex or, in the case of an edge, by clicking on the 2 vertices that defines it. Similarly to the "Polygon area", the grid is composed of unit squares in the coordinate system of the board.

SETUP VIEW

The last view of the Boardmaker is the "Setup view". In this view, the user is able to create and modify the setup of the game. The view is identical to the "Main view" with the exception that it is not possible to switch to the "Creation view" as in the "Main view". Just like in the "Board area", a right click on the image of the board will open a popup menu. This menu gives the option to add a certain piece from the set of pieces or to remove a selection of pieces already placed on the board.



Figure 4.12: Example of setup creation

When adding or removing a piece, the interface highlights selectable sites. In the case of an addition, highlighted sites are the ones without piece (see Figure 4.12a). In the case of a removal, sites with a piece will be highlighted (see Figure 4.12b).

5

RESULTS

5

5.1 BOARD CREATION AS VALIDATION

To validate the functionality and robustness of the Boardmaker, a comprehensive testing approach was undertaken through the recreation of existing game boards from Ludii. This validation method serves multiple purposes: verifying that the interface can produce boards equivalent to those created manually, ensuring that the generated descriptions are syntactically correct and compile properly in Ludii, as well as identifying any limitations or bugs in the implementation.

For this test, games were selected to cover the whole spectrum of boards supported by the interface and utilizing different ludemes. This goes from boards issued from simple graph functions to boards resulting from complex modifications, including specialized boards like mancala and surakarta.

For each game, the procedure was the following: the description was first analyzed to identify the different components of the game, after that, an attempt to recreate the board and setup using only the Boardmaker was made. Once done and compiled, the interface-generated description was compared with the original description based on the number of tokens (tokens related to features not supported by the Boardmaker were not counted). The number of tokens composing the description of a game is an useful metrics that indicates the simplicity, clarity and understandability of a description. A low number indicates that the game description is simple, short, and therefore clearer and more understandable than a longer description.

Table 5.1 shows that the description produced by the Boardmaker is generally a little bit longer but not more complex than the original description. This is due to the fact that in the Boardmaker, sites, whether to place pieces or to modify the board, are selected one by one whereas Ludii provides advanced function to select subsets of sites. In addition, it is possible to merge more than 2 boards at the same time in Ludii, which is another reason to the longer descriptions as the Boardmaker only merges boards two by two.

Game	Tokens (Ludii)	Tokens (Boardmaker)
ConHex	52	56
Bam Blang Beh Khla	73	86
Hund Efter Hare	112	161
Halusa	30	27
Surakarta	34	58

Table 5.1: Token counts between the game description in Ludii compared to the description produced by the Boardmaker

5.2 USER STUDY

User studies have the advantage that they allow to test the robustness of the interface in a realistic environment while collecting feedback to improve it at the same time.

5

The study was conducted on individuals both within and outside the Gametable network. In this study, participants were tasked to create a board and its setup, with complete freedom in terms of design. They were only given indications on how to use the interface. The task was split in 4 parts following this order: creation of a basic board, modification of this board, creation of a set of pieces for the board and in the end, placement of the pieces on the board. After each step, users had to answer a few questions about the clarity, control, overall usability and possible problems of the part of the interface involved in this step. Eventually, they were asked their general opinion about the Boardmaker as well as suggestions to improve it.

Metrics	Board creation	Modification	Piece selection	Setup
Clarity	4.4	3.33	4.47	4.6
Controls	4.53	3.53	4.47	4.6
Usability	4.07	3.4	4.13	4.47

Table 5.2: Average of the notes given by users to the different steps, on a scale of 5

Table 5.2 summarizes the results obtained during the user study. These results shows that choices made in the design of the interface were appropriate. However, the drop in the ratings concerning the board modification step proves that the interface can still be improved.

Regarding the system as a whole, users found it relatively easy to use with an average score of 3.93. From the feedback, 2 major inconveniences have been identified, the lack of information about the different board modification and the lack of feature allowing to undo some actions. Despite these inconveniences, the majority of surveyed users that had already created game description for Ludii confirmed that the interface was better.

5.3 LIMITATIONS

The creation of the validation boards and the user study allowed to identify several limitations of the interface:

1. The lack of features allowing to undo actions made by mistake or experimentation causes users to restart the creation of their board from the beginning.
2. As the dimensions of the boards gets bigger, the clickable area to select a site or an element becomes smaller, making the creation of boards with considerable size a delicate task.
3. In its current iteration, the interface doesn't allow to save one's work in order to finish it later. This prevents the user to close the interface in the mean time or forces them to complete the task in one go.
4. The modification of a board with pieces placed on it can disrupt their placement if topological changes are too consequent.
5. In Ludii, the "Merge" ludeme and its alike (i.e. the "Union" and "Intersect" ludeme) can support more than 2 graph functions. This is not the case with the interface, which implies that merging (or else) more than 2 boards using the interface requires to merge them sequentially, which produces a longer game description.
6. The interface doesn't support the setup for mancala boards as the ludeme required to place pieces on this type of boards is not supported by the Boardmaker.

6

FUTURE WORK

6

6.1 RULE CREATION

As a reminder, this work is part of a greater project with the goal to create an interface capable of creating any board game that can be described in Ludii. Since the Boardmaker successfully handles the creation of the board and the placement of the pieces of the game, the next step of the project is to create either an extension of the Boardmaker or a separate interface that would enable users to define game rules with a graphical interface. This would complete the process of creating a game without writing the description.

6.2 FEATURE COMPLETION

Board related features present in the Boardmaker only represent a subset of the features implemented in Ludii. Among those, tracks and regions are the most used. A track creates a set of successive sites as generally used in race games. A region is merely a set of sites that can possibly be assigned to a player. In addition to these features, there is also features related to pieces such as the possibility to stack them on the board or to modify the direction they are facing, but also features allowing to customize the image of the board through the use of the "Metadata" ludeme.

Furthermore, Ludii provides other types of containers and components than just boards and pieces as shown in Figures 3.3, 3.4 and 3.5. While some of them, such as "Hand" or "Die", can be regularly used in board games, others are only used in logic puzzles and card games, both of them being the object of thesis aiming to add these types to Ludii, respectively written by Pierre Accou [36] and Alexandre Verlaine [37].

Adding all these features to the Boardmaker would bring it one step closer to its objective of creating all kind of games entirely with a GUI.

7

CONCLUSION

To conclude, this thesis focused on the creation of a graphical user interface that allows the creation of boards and the placement of pieces entirely through graphical means. This work is part of a greater project that aims to the creation of tools that addresses the challenge of generating game descriptions without writing a single line of code.

The Boardmaker successfully demonstrates that complex board topologies, whether created through basic graph functions or combinations of graph operators, can be generated through direct manipulation of such functions.

The validation through board recreation and the user study confirms that the interface achieves its primary objective: simplifying the game creation process while maintaining the full power of Ludii.

Although the Boardmaker meets its objectives, it still encounters some defaults and limitations, and lacks some of the advanced features Ludii can offer. The correction of these defaults and limitations, implementation of these advanced features, and the creation of a tool to handle the creation of the game logic through graphical means provides hints for possible future work related to this thesis.



APPENDIX

A.1 VALIDATION GAMES DESCRIPTION

A.1.1 ConHex

```

1 (game "ConHex"
2   (players 2)
3   (mode Alternating)
4   (equipment
5     {
6       (board
7         (keep
8           (splitCrossings
9             (merge
10              {
11                (rotate
12                  45.00
13                  (dual
14                    (square
15                      Square
16                      12
17                      diagonals:Concentric
18                    )
19                  )
20                )
21                (shift
22                  3.00
23                  3.00
24                  (scale
25                    6.00
26                    6.00
27                    (square Square 1 diagonals:Implied)
28                  )
29                )
30              }
31            )
32          )
33        (poly
34          {
35            {2.774 2.908} {9.159 2.908} {9.092 9.259}
36            {2.774 9.226}
37          }
38        )
39      )
40    use:Cell

```

A

```

41 )
42 (piece "rectangle" Each)
43 (piece "disc" Each)
44 }
45 )
46 (rules
47 (play (forEach Piece))
48 (end (if (no Moves Next) (result Mover win)))
49 )
50 )

```

A.1.2 BAM BLANG BEH KHLA

```

1 (game "Bam Blang Beh Khla"
2 (players 2)
3 (mode Alternating)
4 (equipment
5 {
6 (board
7 (add
8 (remove
9 (merge
10 {
11 (merge
12 {
13 (shift
14 0.40
15 -1.00
16 (scale
17 0.60
18 0.60
19 (rectangle
20 1
21 3
22 diagonals:Implied
23 )
24 )
25 )
26 (shift
27 0.20
28 0.00
29 (scale 0.80 2.00 (wedge 2 3))
30 )
31 }
32 )
33 (shift
34 0.00
35 -0.50
36 (scale 1.00 2.50 (wedge 2 3))
37 )
38 }
39 )
40 edges:{ { 3 7 } { 3 8 } { 3 9 } }
41 )
42 edges:{
43 { { 0.200 0.000 } { 0.000 -0.500 } }
44 { { 1.800 0.000 } { 2.000 -0.500 } }
45 { { 1.000 -0.500 } { 1.000 -1.000 } }
46 }
47 )
48 use:Vertex
49 )
50 (piece "goat" P2)
51 (piece "tiger" P1)
52 }
53 )

```

```

54 (rules
55   (start
56     { (place "tiger1" { "B4"}) (place "goat2" { "A1" "B1" "C1" }) }
57   )
58   (play (forEach Piece))
59   (end (if (no Moves Next) (result Mover win)))
60 )
61 )

```

A.1.3 HUND EFTER HARE

```

1 (game "Hund Efter Hare"
2   (players 2)
3   (mode Alternating)
4   (equipment
5     {
6       (board
7         (add
8           (remove
9             (merge
10              {
11                (merge
12                  {
13                    (merge
14                      {
15                        (rectangle
16                          3
17                          5
18                          diagonals:Implied
19                        )
20                        (shift
21                          -1.00
22                          1.00
23                          (rectangle
24                            1
25                            7
26                            diagonals:Implied
27                          )
28                        )
29                      }
30                    )
31                    (shift
32                      1.00
33                      0.00
34                      (scale
35                        0.50
36                        0.50
37                        (rectangle
38                          5
39                          1
40                          diagonals:Implied
41                        )
42                      )
43                    )
44                  }
45                )
46                (shift
47                  3.00
48                  0.00
49                  (scale
50                    0.50
51                    0.50
52                    (rectangle 5 1 diagonals:Implied)
53                  )
54                )
55              }

```

A

```

56         )
57         edges: { { 6 11 } { 8 13 } { 3 8 } { 1 6 } }
58     )
59     edges: {
60         { { 0.000 2.000 } { -1.000 1.000 } }
61         { { -1.000 1.000 } { 0.000 0.000 } }
62         { { 4.000 0.000 } { 5.000 1.000 } }
63         { { 5.000 1.000 } { 4.000 2.000 } }
64         { { 0.000 2.000 } { 1.000 1.500 } }
65         { { 0.000 1.000 } { 1.000 1.500 } }
66         { { 1.000 1.500 } { 2.000 1.000 } }
67         { { 2.000 1.000 } { 3.000 1.500 } }
68         { { 3.000 1.500 } { 4.000 1.000 } }
69         { { 4.000 2.000 } { 3.000 1.500 } }
70         { { 3.000 1.500 } { 2.000 2.000 } }
71         { { 2.000 2.000 } { 1.000 1.500 } }
72         { { 0.000 1.000 } { 1.000 0.500 } }
73         { { 1.000 0.500 } { 2.000 0.000 } }
74         { { 2.000 0.000 } { 3.000 0.500 } }
75         { { 3.000 0.500 } { 4.000 1.000 } }
76         { { 4.000 0.000 } { 3.000 0.500 } }
77         { { 3.000 0.500 } { 2.000 1.000 } }
78         { { 2.000 1.000 } { 1.000 0.500 } }
79         { { 1.000 0.500 } { 0.000 0.000 } }
80     }
81 )
82 use:Vertex
83 )
84 (piece "dog" P2)
85 (piece "hare" P1)
86 }
87 )
88 (rules
89   (start
90     {
91       (place "hare1" {"C3"})
92       (place "dog2" {"A3" "B5" "B3" "B1"})
93     }
94   )
95   (play (forEach Piece))
96   (end (if (no Moves Next) (result Mover win)))
97 )
98 )

```

A.1.4 HALUSA

```

1 (game "Halusa"
2   (players 2)
3   (mode Alternating)
4   (equipment
5     {
6       (mancalaBoard 2 6 store:None numStores:2)
7       (piece "disc" Each)
8     }
9   )
10  (rules
11    (play (forEach Piece))
12    (end (if (no Moves Next) (result Mover win)))
13  )
14 )

```

A.1.5 SURAKARTA

```

1 (game "Surakarta"

```

```

2  (players 2)
3  (mode Alternating)
4  (equipment
5    {
6      (surakartaBoard (rectangle 6 6 diagonals:Implied) loops:2 from:1)
7      (piece "disc" Each)
8    }
9  )
10 (rules
11   (start
12     {
13       (place
14         "disc1"
15         {
16           "A1" "B1" "C1" "D1" "E1" "F1" "F2" "E2" "D2" "C2" "B2"
17           "A2"
18         }
19       )
20       (place
21         "disc2"
22         {
23           "A6" "B5" "A5" "B6" "C6" "C5" "D5" "D6" "E6" "E5" "F5"
24           "F6"
25         }
26       )
27     }
28   )
29   (play (forEach Piece))
30   (end (if (no Moves Next) (result Mover win)))
31 )
32 )

```


BIBLIOGRAPHY

REFERENCES

- [1] Cameron Browne, Dennis JNJ Soemers, Éric Piette, Matthew Stephenson, Michael Conrad, Walter Crist, Thierry Depaulis, Eddie Duggan, Fred Horn, Steven Kelk, et al. Foundations of digital arch { \ae } oludology. *arXiv preprint arXiv:1905.13516*, 2019.
- [2] Cameron Browne and Eric Piette. Digital archaeoludology. In *Computer Applications and Quantitative Methods in Archaeology:(CAA'19)*. 2019.
- [3] Koichi Masukawa. The origins of board games and ancient game boards. In *Simulation and Gaming in the Network Society*, pages 3–11. Springer, 2016.
- [4] Georgios N Yannakakis and Julian Togelius. *Artificial intelligence and games*, volume 2. Springer, 2018.
- [5] Fernando de Mesentier Silva, Scott Lee, Julian Togelius, and Andy Nealen. Ai-based playtesting of contemporary board games. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*, pages 1–10, 2017.
- [6] Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. Deep blue. *Artificial intelligence*, 134(1-2):57–83, 2002.
- [7] Fangxing Li and Yan Du. From alphago to power system ai: What engineers can learn from solving the most complex board game. *IEEE Power and Energy Magazine*, 16(2):76–84, 2018.
- [8] Matthew Stephenson, Eric Piette, and Cameron Browne. Teaching and learning with ludii:(bgs'19). In *Board Game Studies*. 2019.
- [9] Paola Rizzi. Eat the cabbage, kill the wolf: When a game becomes a useful tool for the simulation of communication, information and decision-making processes. In *Simulation and Gaming in the Network Society*, pages 25–33. Springer, 2016.
- [10] Brianno D Collier and David J Shernoff. Video game-based education in mechanical engineering: A look at student engagement. *International Journal of Engineering Education*, 25(2):308, 2009.
- [11] Shota Noda, Kentaro Shiotsuki, and Mutsuhiro Nakao. The effectiveness of intervention with board games: a systematic review. *BioPsychoSocial medicine*, 13:1–21, 2019.

- [12] Jean François Dartigues, Alexandra Foubert-Samier, Mélanie Le Goff, Mélanie Viltard, Hélène Amieva, Jean Marc Orgogozo, Pascale Barberger-Gateau, and Catherine Helmer. Playing board games, cognitive decline and dementia: a french population-based cohort study. *BMJ open*, 3(8):e002998, 2013.
- [13] M Janković. Archaeology of taste: Board and dice games of moesia superior. *Reflections of Roman Imperialisms*, Cambridge, Scholars Publishing, pages 236–263, 2018.
- [14] Mark A Hall and Katherine Forsyth. Roman rules? the introduction of board games to britain and ireland. *Antiquity*, 85(330):1325–1338, 2011.
- [15] Jason Begy. Board games and the construction of cultural memory. *Games and culture*, 12(7-8):718–738, 2017.
- [16] Cameron Browne. Ai for ancient games: report on the digital ludeme project. *KI-Künstliche Intelligenz*, 34(1):89–93, 2020.
- [17] Éric Piette, Walter Crist, Dennis JNJ Soemers, Lisa Rougetet, Summer Courts, Tim Penn, and Achille Morenville. Gametable cost action: kickoff report. *ICGA Journal*, 46:11–27, 2024.
- [18] Eric Piette, Dennis JNJ Soemers, Matthew Stephenson, Chiara F Sironi, Mark HM Winands, and Cameron Browne. Ludii—the ludemic general game system. In *ECAI 2020*, pages 411–418. IOS Press, 2020.
- [19] Michael Genesereth and Michael Thielscher. *General game playing*. Morgan & Claypool Publishers, 2014.
- [20] Michael Genesereth, Nathaniel Love, and Barney Pell. General game playing: Overview of the aaai competition. *AI magazine*, 26(2):62–62, 2005.
- [21] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.
- [22] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.
- [23] Michael Thielscher. The general game playing description language is universal. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1107, 2011.
- [24] Eric Piette, Matthew Stephenson, Dennis JNJ Soemers, and Cameron Browne. General board game concepts. In *2021 IEEE Conference on Games (CoG)*, pages 01–08. IEEE, 2021.

- [25] Michael Thielscher. A general game description language for incomplete information games. In *Proceedings of the AAAI conference on artificial intelligence*, volume 24, pages 994–999, 2010.
- [26] Michael Thielscher. Gdl-iii: A description language for epistemic general game playing. In *IJCAI*, pages 1276–1282, 2017.
- [27] Maciej Świechowski, HyunSoo Park, Jacek Mańdziuk, and Kyung-Joong Kim. Recent advances in general game playing. *The Scientific World Journal*, 2015(1):986262, 2015.
- [28] Cameron Browne. Everything’s a ludeme well, almost everything. In *XXIII BOARD GAME STUDIES COLLOQUIUM-The Evolutions of Board Games*, 2021.
- [29] Cameron Browne, Éric Piette, Matthew Stephenson, and Dennis JNJ Soemers. Ludii general game system for modeling, analyzing, and designing board games. *Encyclopedia of Computer Graphics and Games*. Springer Nature, 2023.
- [30] Éric Piette, Cameron Browne, and Dennis JNJ Soemers. Ludii game logic guide. *arXiv preprint arXiv:2101.02120*, 2021.
- [31] Cameron Browne, Éric Piette, Matthew Stephenson, and Dennis JNJ Soemers. General board geometry. In *Advances in Computer Games*, pages 235–246. Springer, 2021.
- [32] Clémentine Sacré. *Scalable Board Expansion within a General Game System*. PhD thesis, UCL - Ecole Polytechnique de Louvain, 2025.
- [33] Matthew Scarpino, Laurent Mihalkovic, Stanford Ng, and Stephen Holder. *SWT/JFace in Action: GUI Design with Eclipse 3.0*. Simon and Schuster, 2004.
- [34] Robert Eckstein, Marc Loy, and Dave Wood. *Java swing*. O’Reilly & Associates, Inc., 1998.
- [35] Wendy L Martinez. Graphical user interfaces. *Wiley Interdisciplinary Reviews: Computational Statistics*, 3(2):119–133, 2011.
- [36] Pierre Accou. *Model and Play Logic Puzzles within Ludii*. PhD thesis, UCL - Ecole Polytechnique de Louvain, 2024.
- [37] Alexandre Verlaine. *Introducing Card Games in Ludii*. PhD thesis, UCL - Ecole Polytechnique de Louvain, 2025.

