Development of games for users with visual impairment

Czech Technical University in Prague

Faculty of Electrical Engineering

Dina Chernova

January 2017

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Computer Graphics and Interaction

BACHELOR PROJECT ASSIGNMENT

Student: **Dina Chernova**

Study programme: Software Engineering and Management
Specialisation: Web and Multimedia

Title of Bachelor Project: **Development of games for users with visual impairment**

Guidelines:

1. Study the user-centered design for blind and visually impaired users.
2. Implement two games in Java language (e.g. chess, RPG, checkers).
3. Test the user experience on 5 visually impaired users.

Bibliography/Sources:

[1] BGF toolkit, Blind Faith Game, http://en.blind-faith-games.e-ucm.es/, 2016

[2] Game Accesibility guidelines, http://gameaccessibilityguidelines.com/, 2016

Bachelor Project Supervisor: Ing. Daniel Novák, Ph.D.

Valid until the end of the summer semester of academic year 2017/2018

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
-7-
L.S.

[illegible]
prof. Ing. Jiří Žára, CSc.
Head of Department

[illegible]
prof. Ing. Pavel Ripka,CSc.
Dean

Prague, November 21, 2016

Acknowledgement

I would first like to thank Bc. Honza Hadáček for his valuable advice. I am also very grateful to my supervisor Ing. Daniel Novák, Ph.D. and to all participants that were involved in testing of my application for their precious time.

I must express my profound gratitude to my loved ones for their support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them. Thank you.

Declaration

I declare that I have developed this thesis on my own and that I have stated all the information sources in accordance with the methodological guideline of adhering to ethical principles during the preparation of university theses.

In Prague 09.01.2017

Author

Abstract

This bachelor thesis deals with analysis and implementation of mobile application that allows visually impaired people to play chess on their smart phones. The application control is performed using special gestures and text–to–speech engine as a sound accompanier. For human against computer game mode I have used currently the best game engine called Stockfish. The application is developed under Android mobile platform.

Keywords: chess; visually impaired; Android;

Bakalářská práce se zabývá analýzou a implementací mobilní aplikace, která umožňuje zrakově postiženým lidem hrát šachy na svém smartphonu. Ovládání aplikace se provádí pomocí speciálních gest a text–to–speech enginu pro zvukové doprovázení. V režimu člověk versus počítač jsem použila současně nejlepší herní engine Stockfish. Aplikace je vyvinuta pro mobilní platformu Android.

Klíčová slova: šachy; nevidomí; Android;

Contents

Chapter 1

Introduction

1.1 Motivation

1.1.1 Current situation

With the constant rapid evolution of technics, a person's life is becoming easier and different types of gadgets are becoming more affordable for the masses. More and more people are purchasing them for the daily use. Nowadays it's hard to imagine one's life without a smart phone. By the year 2016 the number of smart phone users is predicted to reach 2.1 billion. And this number is constantly rising. So that, by the year 2019 it is forecasted to reach and pass 5 billion mark. [1] With the technical progress, mobile devices are also becoming more popular among people with disabilities. According to the Web AIM Screen Reader User Survey from the year 2014 smart phones with screen readers are trending to be used by blind and visually impaired population. 44% of the respondents assert to be using those as much or even more frequently than personal computers with screen readers. [2]

1.1.2 Why chess

Chess is a very old, sophisticated game with a well established rule set and infinite game outcomes. It is believed that first game is originated in India before the 6th century AD and has been spread all over the world with time. [3] This fourteen century old game is played by people of many nationalities and different religious believes. It also has a static interface, therefore is favourite among people with visual disabilities.

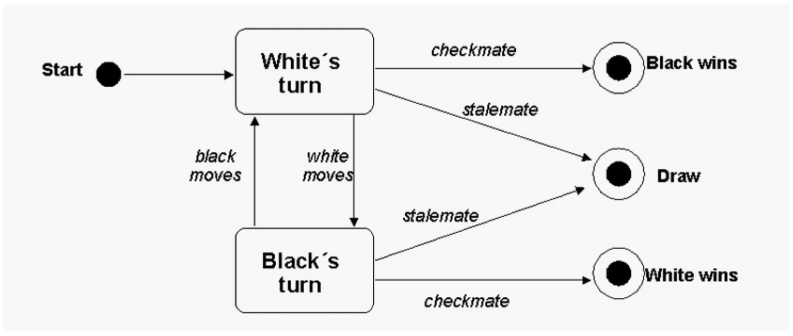


Figure 1.1: Simplified chess game state diagram [4]

1.2 The scope of the work

The scope of this work is to build a mobile application that will allow blind or visually impaired user to play chess. I have named this app “Blind Chess”. Chess game will be implemented in the Java programming language for Android operating system.

The requirements for the application have been decided in collaboration with SONS (Sjednocená organizace nevidomých a slabozrakých České republiky) [5].

On the final stage of this work the application will be tested on five blind users with usability tests. Some of the testing will take place in SONS office and some at my participants' houses. I will be testing this app on the members of SONS and some other visually impaired people I know in order to get feedback for possible improvement.

Chapter 2

Used methods

2.1 Visually impaired and chess

Sight is not necessary to play chess. In fact, there's a way to play chess without even seeing a board or touching any of the pieces. It is called blindfold. But not all the people are able to memorize current positions of the pieces on the board and at the same time keep track of the game. Therefore, the classic chessboard was supposed to be customized according to the needs of blind and visually impaired .

The blind chessboard looks almost the same as a classic board. But there are few major differences:

1. The black squares are higher than the whites.
2. Pieces have a downside nail at the base.
3. Each square is made with a whole in the center, so that each piece is inserted and held still.
4. To determine whether the piece is black or white the black ones are provided with pin on the top of them.

With the help of the following modifications a visually impaired person is able to determine a colour of the square and the piece. It is also easy to figure out whether it is a Queen, King, Pawn, Rook, Bishop or Knight just by feeling a chess piece. [6]

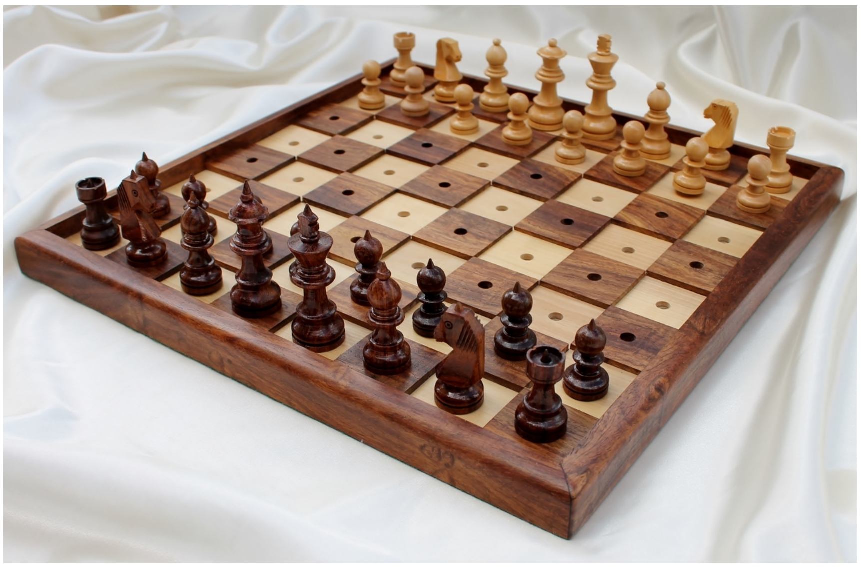


Figure 2.1: Chess board designed for visually impaired users [7]

2.2 Existing applications

2.2.1 Chess–wise

Chess–wise is an application created for Apple's iOS operating system for playing chess. [8] This app is prefered by visually impaired for its ability to be played using VoiceOver feature. VoiceOver is a screen reader native application running on Apple's macOS, iOS, tvOS, watchOS, and iPod operating systems. It is created for blind and low–vision users. VoiceOver has its own way of controlling the device it is running on.

Essentially, this chess game app has all the features that my application has. Human–human and human–PC modes, announcement of the chess board squares and pieces on them when touching the screen, it also says who's turn it is. Instead of moving pieces on chess board user can also type his move in.

There are few disadvantages that I have noticed in this app:

1. Application doesn't announce captured pieces.

2. This app has complicated access to "Undo move" and "New game" features.

Chess–wise is an app created mainly for people with good sight and, to my opinion, is not fully suitable for visually impaired.

2.2.2 KChess Elite

KChess Elite is a chess game application for Windows operating system, such as Windows 3.1 and Windows 9x/ME/NT/2000/XP. [9] It is suitable for use by anyone who is blind or has any visual impairment. All moves can be controlled through the keyboard. Similarly to my application KChess allows for sound feedback. But comparing to my app KChess give more information about game state, like: full move list, details about the most recent move, a narration about the opening and state of the board, and a board layout in FEN format.

On the other hand, it has few cons:

1. User can't play chess outside of her/his house or office, unless she/he brings his personal computer with him which is quite rare for visually impaired.

2. User needs to memorize a great deal of shortcuts.

Chapter 3

Development

3.1 General description

3.1.1 Application functions

Concepts listed below are all of the functions that this application has.

1. Chess board is graphically represented as a XML layout. Logically it is a two dimensional array of TextViews.

2. Chess pieces are objects like Queen, King, Pawn, Rook, Knight, Bishop. Each of them has its own set of possible moves.

3. Black and white players are defined.

4. Chess piece selection is enabled.

5. "Undo move" reverses the last move of the player and his opponent if she/he has made any. In case of human–PC mode two moved are being reversed.

6. Current state announcement feature enables for a user to find out actual game state, which includes information about who's turn it is, pieces that user has captured and in case of human–PC game her/his opponent's captured pieces. Similarly, if a piece is being captured, an app reacts to it by announcing.

7. "New game" starts a new game. Current game is automatically saved after exiting the app and loaded when the app is opened again.

8. Settings consist of game mode (human–human, human–PC) and in case of human–PC game mode also level and player colour selection.

9. Human versus human game is implemented.

10. Human versus computer game is also a part of this application.

11. Board flip is a feature that is enabled in human versus human game that allows each player to perceive the board from her/his perspective.

12. Screen sound feedback in a way that every chess board square and a button is being spoken when user touches it.

13. Manual can be spoken up if a specific button on the smart phone is held.

3.1.2 Gestures

It is essential for an application created for a visually impaired to have a set of rules in order to control it. Selection of either a chess piece or a button on main chess board layout is completed by holding one finger on an item desired to be selected and taping another finger on any other place of the screen. When selecting anything from “Settings” menu one long press is required. For scrolling through menu items one short touch on the left or right side of the screen is needed. Everything is accompanied with vibration.

3.1.3 Text–to–speech

As many visually impaired people say, sound is like colour for the blind. That is why the main part of my app is the sound.

1. Every action taken by the user is assigned a specific phrase. For example, when selecting a white rook from D4 user gets a feedback by the phrase “White rook is selected from A1”.
2. Another thing important for a visually impaired is to have an idea of what is going on in the game, so that’s where current state announcement plays its role. This option is called “Current state”. It provides information about current player’s colour and possibly selected piece. Another thing that “Current state” maintains is captured pieces. In case of human–PC game mode it also delivers information about opponent’s conquered pieces. For example, “White’s turn. White rook is selected from A1. Captured pieces: two pawn, one rook. Opponent’s captured pieces: one bishop.”
3. Furthermore, there’s an option for a user to be aware of the location of each chess piece on the board. By swiping along the chess board every square’s name and possibly chess piece’ name is announced. For example, “White rook on A1” or just “D4”.
4. User gets feedback not only from chess board, but from the control buttons panel. So that when touching a button, button’s name is declared.

3.1.4 Smart phone characteristics

User must have a multitouch smart phone with Android 5.1 operating system. This game is played offline, so there’s no need for Internet access.

3.2 Functional specifications

3.2.1 Why do we need them

A requirement specification describes how a software will function from user’s point of view. It has to do with “what”, rather than “how”. Identifying the functional requirements of the product is one of the most important parts in development process.

Use case approach is considered to be the most efficient. It takes into account a set of so called actors (users) that interact with the system. Actor has a goal that she/he is meaning to fulfil. The process of fulfilling a goal with all its sequences is called a use case. Use case describes main flow of an event and all of its alternative variants, even those that lead to an error state. Use case consists of scenarios, that represent a single path through a use case. [10]

In my case there are two main actors: User and AI engine. Use case diagram consists of totally 9 use cases. And there’s just one system.

3.2.2 Use case diagram

The following diagram describes user's interaction with the system. There is two actors in this app: User and AI engine.

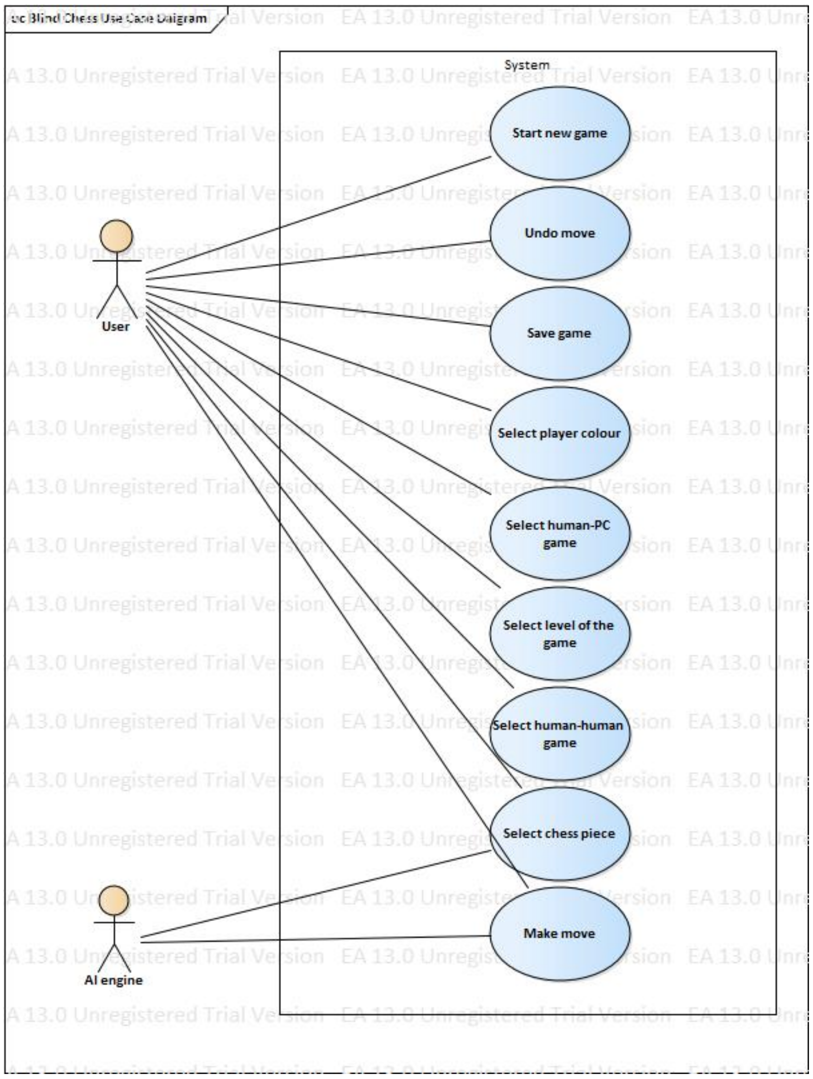


Figure 3.1: Use case diagram

The template for the use cases is created by combining different templates have found into one. It consists of 10 points: use case name, actors, description, trigger, preconditions, postconditions, normal, alternative, exceptional flows and includes.

To explain, trigger is a physical event that triggers a use case to start. In the following example the trigger is user's "Undo" click. Preconditions are conditions that have to be fulfilled in order for a use case to be accomplished. Postconditions clarify the flow of events after a use case has been fulfilled. Basic, alternative and exceptional flows are ways how a use case flow can turn out. Includes are other use cases that are supposed to be accomplished. [10]

As an example of use case description I have decided to illustrate in details a situation when a user wants to undo move. This use case can only happen when an actor is User. In order for this use case to take place User must go through starting new game, selecting a piece and setting selected piece on a position. As you can see this use case has it's normal, alternative and exceptional flow. Existence of two game modes – human–human and human–PC – makes alternative flow possible.

ID	3
Use case name	Undo move
Actors	User
Description	Player undoes move/s
Trigger	Player clicks “Undo”
Preconditions	Player has made a move in order to cancel it
Postconditions	It is initial player’s turn
Basic flow	1. Player makes move 2. IF human–PC game mode enabled THEN PC makes move and it is the initial player’s turn 3. Initial player clicks “Undo” 4. IF human–PC game mode enabled THEN two moves have been undone
Alternative flows	2. IF human–human game mode enabled THEN other player makes move and it is the initial player’s turn 3. IF human–human game mode enabled THEN turn changes and it is the next player’s turn 5. IF human–human game mode enabled THEN only initial player’s move has been undone and the chess board is flipped
Exceptions	Player tries to undo move if none has been performed
Includes	1. Select chess piece 2. Set piece on position

Table 3.1: Use case ”Undo move”

Another use case descriptions are available in Appendix A.

3.2.3 Activity diagram

Activity diagrams represent work flow in the system and are used to illustrate activities. My diagram consists of initial and end points, action and decisions nodes and activity. But there are many more other components that are used in these type of diagrams, like fork node, accept event action and so on…

Activity specifies the way and the order in which the work flow of an application is executed. Flow of the system execution consists of activity nodes, or actions, and activity edges. The whole diagram can consist of number of activities, invocation one another.

Action is single step in activity. One action leads to another. Each of them is taken as a discrete unit. Completion of one action triggers the completion of its successor. [11]

In my diagram of activity I have decided to show the main part of the application. It is the game routine.

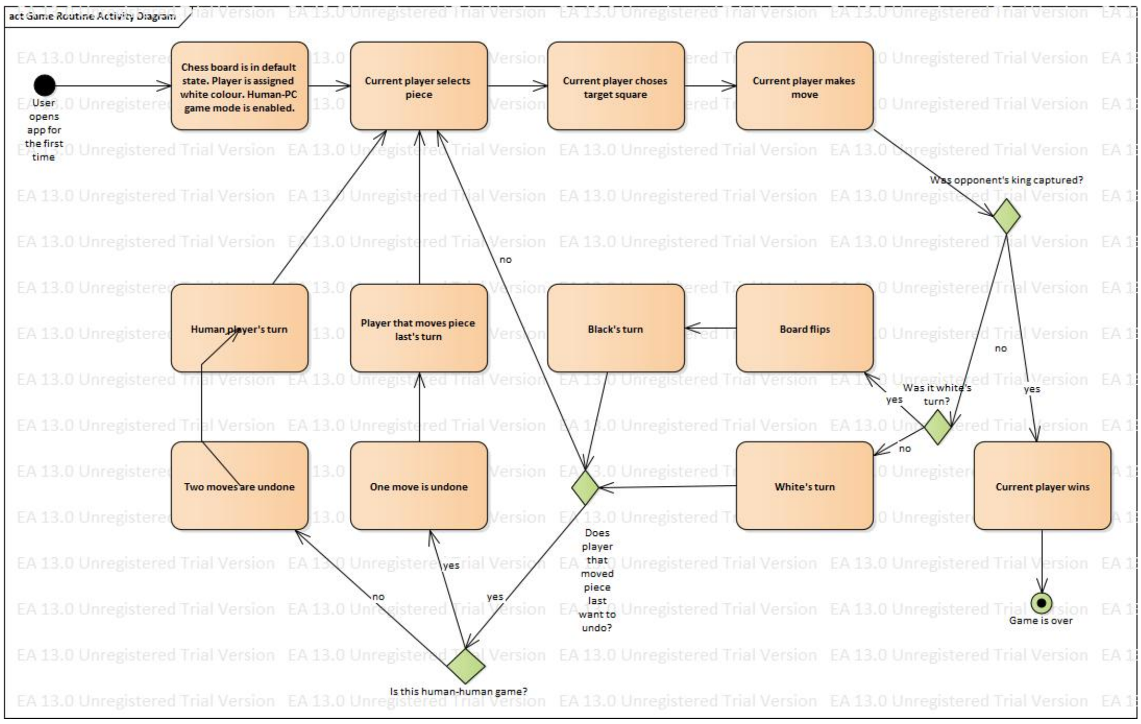

act Game Routine Activity Diagram
User opens app for the first time
Chess board is in default state. Player is assigned white colour. Human-PC game mode is enabled.
Current player selects piece
Current player choses target square
Current player makes move
Was opponent's king captured?
no
yes
Current player wins
Game is over
Board flips
Black's turn
Was it white's turn?
yes
no
White's turn
Does player that moved piece last want to undo?
no
yes
Is this human-human game?
no
yes
Two moves are undone
One move is undone
Human player's turn
Player that moves piece last's turn

Figure 3.2: Activity diagram

3.2.4 Graphical user interface

The chess board interface (including chess pieces) shown in Figure 3.3 is used within all of the gaming process starting from its beginning. It is a default board state. Board is represented as an two dimensional array starting from left top corner (A8) down to right left corner (H1). I have chosen red and grey colours to represent the chess board. Bright colors are generally the easiest to see because of their ability to reflect light, so for partially sighted users it is more convenient to orientate in the app when red colour is used.

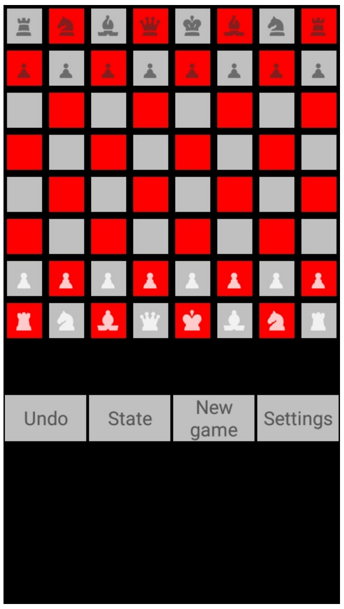


Figure 3.3: Initial GUI

The next screen interface in Figure 3.4 is used once a particular piece has been selected and a target square is chosen and clicked using just one finger. Picked piece's square is assigned green, which also is an easy–to–see colour. Target square changes its colour to an opposite (red or grey).

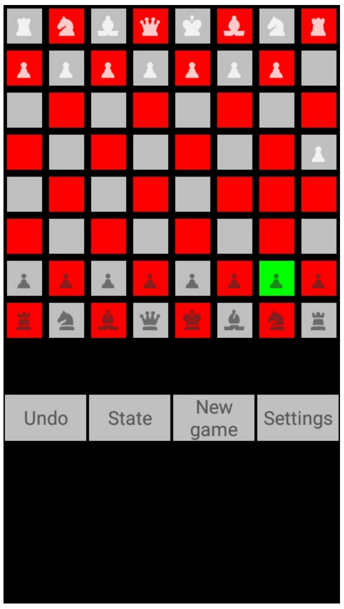


Figure 3.4: Selected piece and target square

The next Figure 3.5 represents a situation when menu button is cliked.

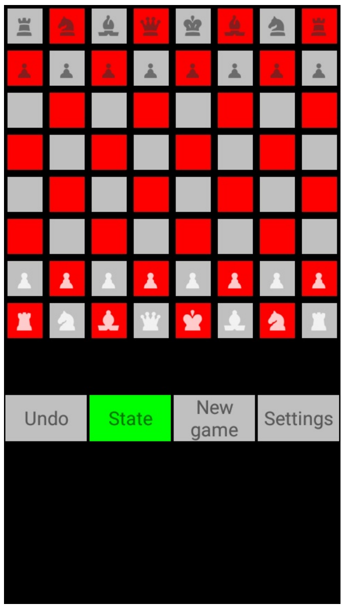


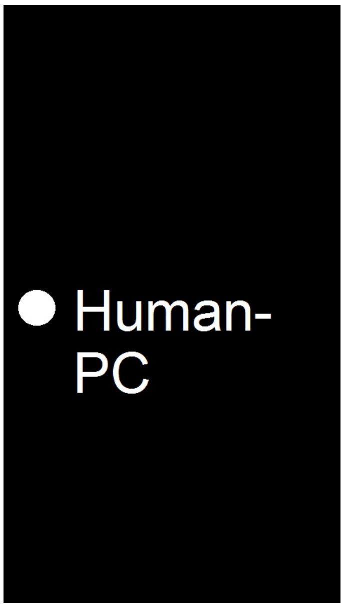
Figure 3.5: Control button clicked

It is possible to set game mode, player colour and game level when clicking on “Settings” in menu layout. Each of these options are shown on Figures 3.6 (a), (b) and (c). Together these menu items are represented as a list. So when clicking on the right or left side of the screen the next or previous menu item is shown.

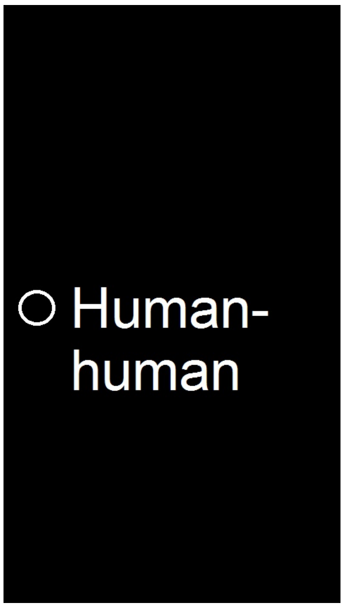
(a) Game mode menu item (b) Player colour menu item (c) Game level menu item

Figure 3.6: Settings menu

When selecting any of these items, selectable submenu is shown. As an example, game mode submenu is pictured on Figures 3.7 (a) and (b).



(a) Human–PC game mode selectable menu item



(b) Human–human game mode selectable menu item

Figure 3.7: Game mode options

Chapter 4

Chess engine

4.1 How do humans play chess

Dr. Fernand Gobet, a professor of Cognitive Psychology at the University of Liverpool, has been studying different aspects of chess psychology. His researches have shown that human being's ability to play chess depends on many factors combined together.

First of all, the time is what counts. Strong players have spent a big amount of time of their lives playing chess. Though, different people achieve different levels of skill in different amount of time. According to the research, some players reach a certain skill level almost 8 times faster than the others.

Secondly, a very important skill to have is pattern recognition. Once a pattern is recognized by a player, it is easier to implement a specific strategy. In average, strong players memorize several hundred thousands of chess patterns.

Furthermore, a great chess player has an ability to search through the problem space. Stronger players tend to inspect deeper through the tree of possible movements.

On top of that, intuition is too has shown to be a relevant skill for playing chess on a master level. [12]

Nevertheless, all of the above is only accurate for human beings. Computers play chess the other way.

4.2 How do computers play chess

There are different programs called chess engines that analyze the board and pick the best possible move. According to a third party source that test engines and rate them, by December 10, 2016 three best chess engines are:

1. Stockfish
2. Komodo
3. Houdini[13]

4.2.1 Bitboards

The way that the board is arranged is called a bitboard. This data structure is used in the most popular and strongest engines, like Houdini and Stockfish. As the board has 64 squares it can be represented as a binary number consisting of 64 bits. By standard, the bit on position 0 represents A1, similarly the one on position 63 represents H8.

The Figure that follows represents the positions of bits coming from left down side to the right up side.

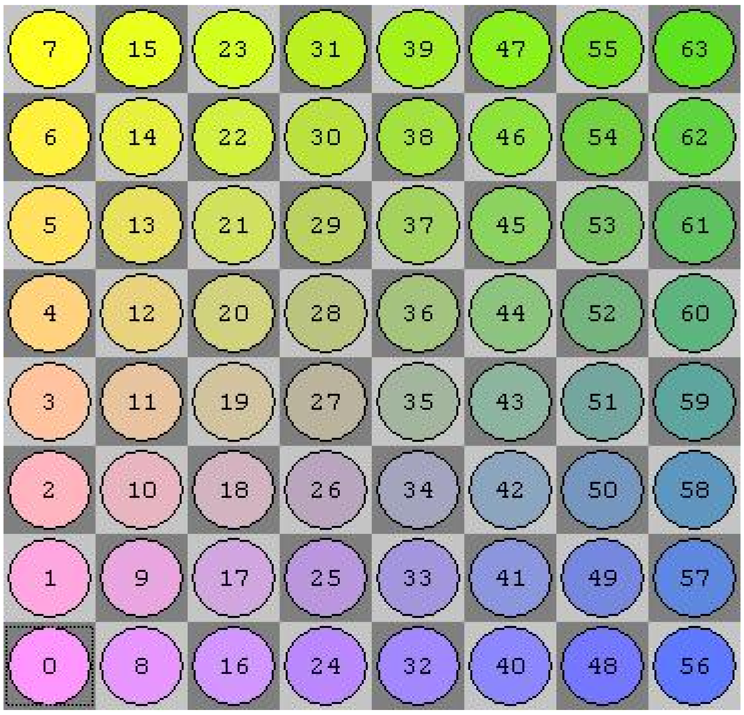


Figure 4.1: Bitboard representation by each bit [14]

Each of the 12th pieces gets its own bitboard. So there are 12 64–bit–long numbers representing the whole board in total. So that, for a white rook placed on A1 the bitboard will look like **10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000**.

The following Figure shows each of the pieces having its own bitboard.

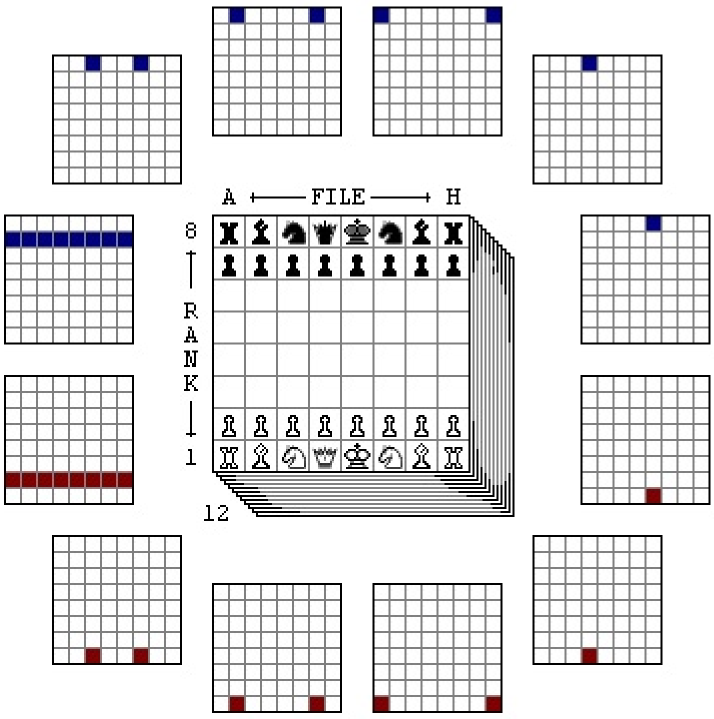


Figure 4.2: Twelve 64–bit–long strings representing the whole board [15]

There are three main advantages of bitboards:

1. Memory Usage: Bitboards encode the board state more efficiently than integer–per–cell encodings.
2. Efficient Operation: Read and write operations can be performed efficiently using bitwise operations.
3. Bitwise–Parallel Operation: Bitwise operations can be applied to all board cells simultaneously. [16]

Bitboards is the most efficient way to store data about current board state. Computer processors are made to work with binary numbers and with the cost of just one logical operation this data structure is able to answer some questions about the game state. For example, if a chess program wants to know if the white player has any rook on A1 it can just compare a bitboard for the white player's rooks with one that has 1 on bit on position 0 using AND operation. If there are no rooks on A1, the result will be zero. [17]

4.2.2 Game tree

Game tree is a data structure that represents available moves in the game starting at the initial position and containing all of the possible moves from each position. In chess it is inconvenient to generate all of the possible moves, that's why partial trees are used. [18] For this kind of trees the depth is given, so when reaching it, the search stops.

This data structure is represented as a directed graph, which means that nodes are connected by edges that have directions associated with them. Nodes represent positions and edges – moves. [18] The initial position of the search is called root. Leaves are nodes which are terminal, that when reaching them the game comes to an end, or the ones where the desired depth from the root is achieved. Each layer of the tree is called ply. Each layer is created when one of the players takes turn. [19]

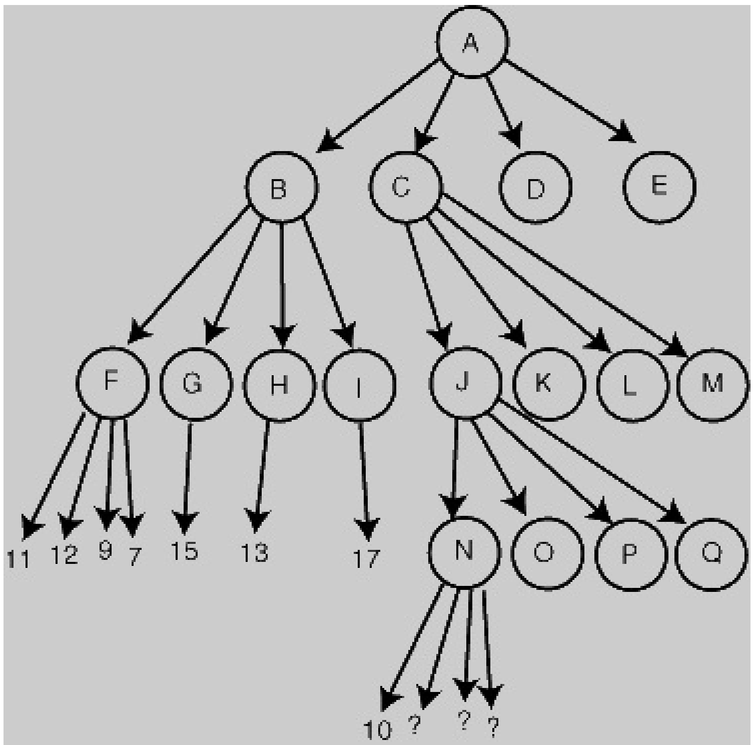


Figure 4.3: An $\alpha - \beta$ search tree [20]

Building a tree is very time consuming since every move must be rated. There are nearly 5 million possibilities after 5 plies (half–moves) and over 3 billion possibilities at a depth of 7 ply. [21]

4.2.3 Searching techniques

Claude Elwood Shannon, an American electrical engineer, mathematician and researcher from MIT, categorizes searches into two groups:

1. Type A

2. Type B

Type A is a "brute force" search which controls every single position in a tree up to a given depth. These types of algorithms are simple, but their con is that the space of search is exponentially growing.

Technique type A with the following improvements will be called type B according to Claude Shannon's Programming a Computer for Playing Chess:

1. Examine forceful variations out as far as possible and evaluate only at reasonable positions, where some quasi–stability has been established.

2. Select the variations to be explored by some process so that the machine does not waste its time in totally pointless variations.

The type B search technique doesn't examine all of the nodes, but selectively cuts off some of them. [22]

Most of the today's programs are closer to type A, but still use selectivity.

4.2.4 Heuristic evaluation function

Heuristic evaluation function is used in game–playing programs to determine the heuristic values of chances to win for the players. In chess this function changes according to different factors that influence the position. Evaluation function often look like this:

$w_1 * f_1 + w_2 * f_2 + \ldots + W_n * f_n$,

where f is a factor that influences the position and w is weight given to that factor. [23]
The following case illustrates an example of evaluation function. Letters with apostrophe are blacks, the others – whites.

f(p) = 9(Q-Q')
+ 5(R-R')
+ 3(B-B'+N-N')
+ (P-P')
- 0.5(D-D'+S-S'+I-I')
+ 0.1(M-M') + . . .

Q, R, B, N, P – number of kings, queens, rooks, bishops, knights and pawns
D, S, I – doubled, blocked and isolated white pawns par
M – mobility of white pieces (the number of legal moves)
The score of the game is returned as a subtraction of current player's scores from his opponent's. [22]

4.2.5 Search algorithms

Minimax

Once moves have been analyzed, the engine must pick an initial move which provides the most promising future. Engine anticipates player's best move in response to each of its best move.

Minimax is the algorithm used to evaluate the best player in the zero–sum game. Zero–sum game is a game, where one player's lose or gain is balanced with the other player's. If the all of the both players' gains are added up and all of the losses are subtracted and result is zero, then the game is zero–sum. [24]

Graham Owen in his paper "Search Algorithms and Game Playing" from 1997 describes minimax algorithm: "The minimax algorithm assigns one player as the maximizer, and the other player as the minimizer. At each ply, the maximum (or minimum) score returned from all of the nodes is backed–up to the node above. This process is followed recursively until the root node is assigned the backed–up score of the target node."

As an example, let's assume all of the pieces have been assigned heuristic values in the way described in the following table. Whites get all of their pieces accredited with the positive values, blacks – negative. [23]

Piece	Value
Pawn	100
Bishop	330
Knight	350
Rook	520
Queen	980
King	10000

Table 4.1: Evaluated chess pieces

The graph that is shown on the Figure 4.4 is 3 ply tree. Leafs have been given specific heuristic values. Some of these values, like 102, 187, 106, 354 represent pieces that can be captured. First three are pawns and the last is knight.

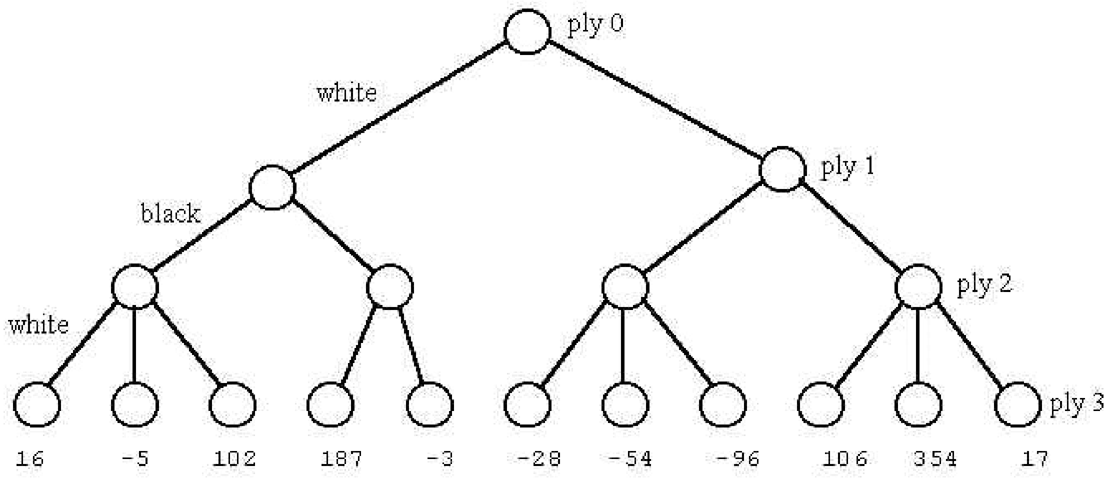


Figure 4.4: Tree with plies and players [23]

In the Figure 4.5 each ply according to max or min command searches for a required value and passes it to the parent node until the root node is assigned 102. This particular example indicates that the best move for whites will lead to capturing a pawn.

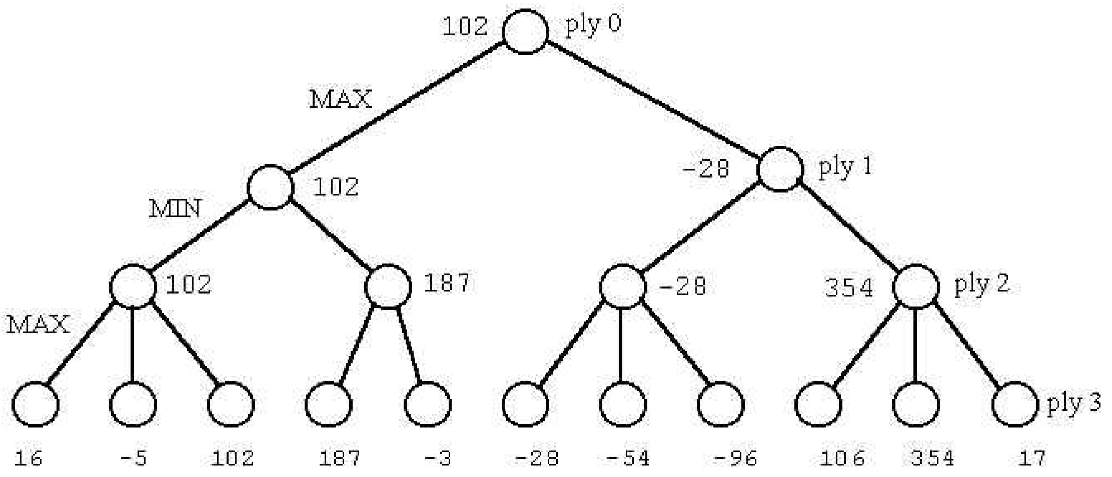


Figure 4.5: Minimaxed tree [23]

The alpha–beta pruning

Instead of checking every possible move that could happen, most modern engines employ some sort of smart pruning in which they avoid analyzing certain moves and thus avoid all moves following that pruned move. This kind of algorithms greatly speed up the engine. Alpha–beta pruning is one of them.

Graham Owen writes in his paper: "The alpha–beta search algorithm is based on maintaining two values, α and β. Initially set to $-\infty$ and $+\infty$ respectively, α representing the value of the best path found so far for the maximizer, and β the value of the best path found so far for the minimizer. An $\alpha - \beta$ search updates the values of α and β as it goes along, with maximizing nodes only able to increase the value of α, and minimizing nodes only able to decrease the value of β. This creates a window which gets narrower as the values of α and β approach each other. Any branches of the tree that have values that fall outside of this range are identified as being worse than the current best choice and are immediately pruned." (Search Algorithms and Game Playing, 1997). [23]

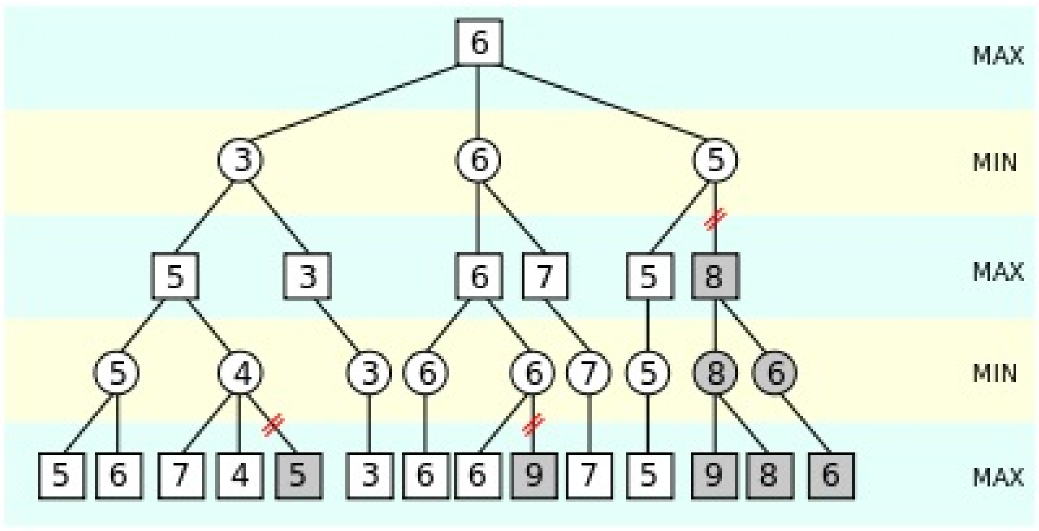


Figure 4.6: $\alpha - \beta$ pruned tree [25]

4.2.6 Move ordering

Iterative deepening

One of these techniques is called iterative deepening. Initially this algorithm was created to solve the problem of choosing depth of tree according to the time given for the search. One ply is first examined for the best move and repeated extension by one ply is applied to the search until we run out of given time. The results of previous iterations can improve the move ordering of new iteration, which is critical for efficient searching. Furthermore, during the game the best move is already found for every ply if this algorithms is used. This is important for games like chess where concrete time limits are set. [23]

Transposition tables

In order for an engine to remember positions that it previously looked at transposition tables are used. All of the known positions are stored in the memory so that they can afterwards be searched and checked if an engine has already seen them. This information not only includes the position, but typically the search depth, the best move, the score, alpha and beta. [26]

The table is organized as a hash map, where each value represents all of the information above transformed into a number using Zobrist hashing method. Zobrist hashing method creates keys, which are 64 bit numbers that almost uniquely represent chess board positions in a particular state. Although it could happen that two different positions get the same value, but this is very rare and there's no need to worry about it. [27]

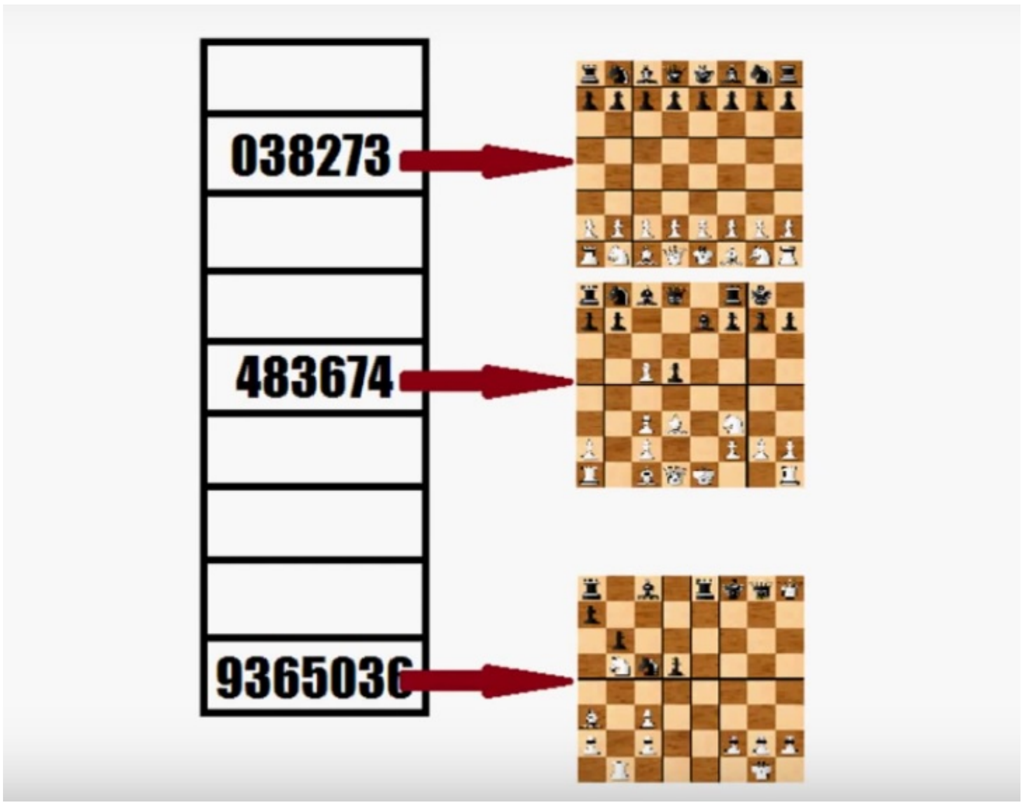


Figure 4.7: Hash for different board states [28]

4.2.7 FEN

Forsyth–Edwards Notation (FEN) is a standard notation for describing a particular board state of a chess game. This standard holds all of the necessary information about the game. It is used to restore game state after a restart of the application. [29]

FEN string consists of 6 main parts:

1. Piece Placement
2. Side to move
3. Castling ability
4. En passant target square
5. Halfmove clock
6. Fullmove counter

Piece placement is performed as a set of chess board rows that are divided by slashes. Pieces are identified as letters: P – Pawn, N – Knight, B – Bishop, R – Rook, Q – Queen, K – King. Uppercase letters are white pieces, lowercase – black pieces. And digits count consecutive empty squares.

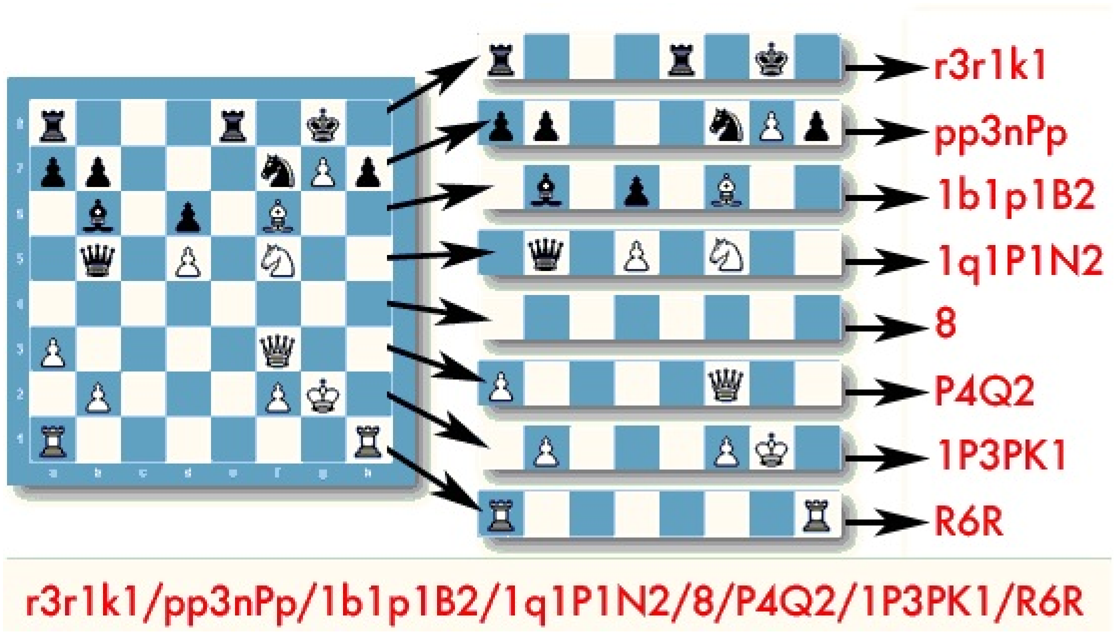


Figure 4.8: FEN piece placement part [30]

Side to move describes whose turn it is. W – white, B – black.

Castling ability specifies how each sides can castle. So, if king's and queen's side of both players can castle this part of FEN string will look like KQkq.

En passant target square is specified after a double push of a pawn, no matter whether an en passant capture is really possible or not. It can look like "e3" if white's pawn has just moved from f2 to f4.

The halfmove clock specifies a decimal number of half moves with respect to the 50 move draw rule. It is reset to zero after a capture or a pawn move and is incremented otherwise.

The number of the full moves in a game starts at 1 and is incremented after each black's move. [31]

By FEN standard a board in the initial state looks like this:
rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1

4.2.8 UCI

A protocol that outlines how a chess engine should communicate with a graphical user interface is called Universal Chess Interface (UCI). UCI was designed and developed by Stefan Meyer–Kahlen and Rudolf Huber in 2000. It is not the only, but is thought to be the best protocol. That is why it is used in most chess engines. [31]

As an example I have taken some commands from communication with the Stockfish engine. The beginning of communication and first move can be performed in the following way:

// GUI tells engine that UCI protocol is used for communication
uci

// Stockfish identifies itself
id name Stockfish 8
id author T. Romstad, M. Costalba, J. Kiiski, G. Linscott

// Stockfish sends the "option" commands to tell the GUI which engine settings the engine

```
// supports
option name Debug Log File type string default
option name Contempt type spin default 0 min -100 max 100
option name Threads type spin default 1 min 1 max 128
option name Hash type spin default 16 min 1 max 2048
option name Clear Hash type button
option name Ponder type check default false
option name MultiPV type spin default 1 min 1 max 500
option name Skill Level type spin default 20 min 0 max 20
option name Move Overhead type spin default 30 min 0 max 5000
option name Minimum Thinking Time type spin default 20 min 0 max 5000
option name Slow Mover type spin default 89 min 10 max 1000
option name nodestime type spin default 0 min 0 max 10000
option name UCI_Chess960 type check default false
option name SyzygyPath type string default ¡empty¿
option name SyzygyProbeDepth type spin default 1 min 1 max 100
option name Syzygy50MoveRule type check default true
option name SyzygyProbeLimit type spin default 6 min 0 max 6

// Stockfish lets GUI know that all parameters are ready
uciok

// GUI sends this command to wait for the engine to be ready again or to ping the engine
to find out if it is still alive
isready

// Stockfish has finished setting up the internal values and is ready to start
readyok

// GUI lets the engine know that new game is starting
ucinewgame

// GUI tells the engine the position to search
position startpos moves e2e4

// GUI tells the engine to start searching with time limit of 100 milliseconds
go movetime 100

// Stockfish sends search information continuously during search including depth, search
// value, time, nodes, speed, and pv line
info depth 1 seldepth 1 multipv 1 score cp 10 nodes 31 nps 7750 tbhits 0 time
4 pv d7d5
info depth 2 seldepth 2 multipv 1 score cp 68 nodes 59 nps 9833 tbhits 0 time
6 pv d7d5 e4d5
info depth 3 seldepth 3 multipv 1 score cp 41 nodes 186 nps 20666 tbhits 0
time 9 pv d7d5 c2c3 d5e4
info depth 4 seldepth 5 multipv 1 score cp 18 nodes 506 nps 33733 tbhits 0
time 15 pv d7d5 e4d5 d8d5 d2d3
info depth 5 seldepth 6 multipv 1 score cp 12 nodes 1950 nps 65000 tbhits 0
time 30 pv g8f6 d2d3 b8c6 b1c3 d7d5
info depth 6 seldepth 6 multipv 1 score cp 11 nodes 3948 nps 58925 tbhits 0
```

```
time 67 pv d7d5 e4d5 d8d5 b1c3 d5d4 d2d3
info depth 7 seldepth 7 multipv 1 score cp -5 nodes 5205 nps 63475 tbhits 0
time 82 pv d7d5 e4d5 d8d5 d2d4 b8c6 c2c3 e7e6
info depth 8 seldepth 8 multipv 1 score cp -22 upperbound nodes 8346 nps
70134 tbhits 0 time 119 pv d7d5 e4d5

// Stockfish returns the best move found
bestmove d7d5
```

4.3 Stockfish

Figure 4.9: Stockfish logo [38]

For the human–PC game mode I have decided to use Stockfish engine. I have chosen this particular engine for it is the strongest open source engine in the world. Stockfish is initially written by Marco Costalba, Joona Kiiski, Gary Linscott and Tord Romstad. Released on November 2, 2008 and is now being developed and maintained by the Stockfish community. [32] This engine has won the ninth season of Top Chess Engine Championship, which has been taking place from May 2016 to December 2016. [33]

It is licensed under GPLv3, which basically means you can share to anybody, sell it as part of a larger project, and change the source code, as long as you either point back to where you got it, or supply the original source code.

Chapter 5

Testing

5.1 Usability testing

Usability testing is a technique of evaluating a product by testing it on users. Usually, this kind of testing is applied to test web sites and applications, computer interfaces, documents and devices. Comparing to another technique called usability inspection, where testers use different methods that do not include testing on actual users, usability testing expresses actual users' remarks. Normally, during this kind of testing users are given tasks that they have to fulfill. Testers are watching, recording the process and taking notes.

The purpose of testing is to reveal if user is satisfied and to collect qualitative and quantitative data. With the help of this technique developers and designers of the product are able to see if users can complete given tasks and analyze how much time it takes in order to do them. Furthermore, it allows for finding out possible future improvements of the product.

Few things are required in order for usability testing to take place such as a screener, pre– and post–test questionnaires and tasks. [34]

5.2 Participants

The target group are visually impaired users using a mobile phone running Android or potential users that play chess or at least know the rules. I have tested the app on 5 people with age ranging from 27 to 62. One of them is a woman. These people work in different fields. Three of them use personal computer, four of them use smart phones. Two of the participants are sand–blind and the others are fully blind.

5.3 Screener

Screener is a questionnaire by which you can determine whether a person is suitable for application testing. Furthermore, it is useful to receive basic information about the participant, such as her/his age, hobbies, etc. All screeners are attached to this thesis in Appendix B.

5.4 Pre–test questionnaire

This questionnaire is used to determine additional information about the selected participants. Questions are usually open type and participant has to answer them before the start of actual testing. The focus of these questions is close to the topic of the testing product. Used pre–testing questionnaire is also attached in Appendix B.

5.5 Post–test questionnaire

After completion of the passage of the set tasks participant is asked a few questions from post–test questionnaire. These questions relate to the actual testing and usability application. It serves as an appropriate complement of the knowledge gained during the tasks completion. This questionnaire is provided in Appendix B.

5.6 Testing process

Normally, usability testing is held in a special usability testing laboratory and participant is left by her/himself in a room. But in case of testing an app on visually impaired people, there's no need for a special environment. Also, it is better to have a moderator in the same room as a participant, though moderator can't help a participant with given tasks. Moderator is only allowed to help with assistance in orientation in a room, handing over a smart phone and questionnaires. The whole testing is held in Czech language. All questions and answers are translated to English for this thesis.

The testing starts with moderator reading pre–test questions to a participant in order for him to answer them. After that moderator gives a participant tasks from the task list. It is assumed the application is running before first task is given.

While participant is fulfilling given tasks, it is best if she/he comments the process of her/his thoughts. Moderator writes down her/his observations. After completing the tasks a participant is ought to answer post–test questionnaires.

5.7 Analysis

All five participants had a smart phone with Android OS with BlindChess game which I have provided. It was a bit of a problem for them to control the app. But all of them were convinced that it would take time to learn all the gestures.

Most of the participants had a problem with selecting a chess piece and target square because the squares were too small. Though, it is not possible to make the squares bigger for this concrete model of a smart phone that I have used due to its monitor size. One sand–blinded participant had a problem with recognizing a currently selected chessboard square and another participant had made a remark about lack of information. Most of the participants agreed that the application was better than they expected.

The speed with which the participants fulfilled the given tasks depended on the fact if this person has ever used a smart phone for the visually impaired.

Overall impression was positive. Participants were satisfied. All of them agreed that it is now more convenient for a visually impaired to play chess wherever she/he wants.

5.8 Improvements

No major disabilities were found. Nevertheless, participants have pointed out on some things which can improve the app.

1. Implement a bright–coloured focus when sliding across chess board squares.
2. Add information about last moves made when clicking on "State".
3. Change the way the app is letting a user know about captured pieces. Do not sort on players, but on colour of those pieces.
4. Make chess pieces bigger and the white ones brighter.

Chapter 6

Conclusion

This thesis describes the analysis and implementation of mobile application for playing chess for the blind. It also describes the way this game is played against the strongest chess engine – Stockfish – which gives a reader a deeper understanding of things that happen behind the curtain.

The main goal of this work was to create an application with a user–friendly GUI that gives a visually impaired community an opportunity to play chess whenever and wherever they wish and an expansion of the use of a smart phone. To make that happen I analyzed two already existing apps running on iOS and Windows and selected the most relevant and wanted features. For more concrete details of my application I also collected system requirements by communicating with SONS community. Application was tested on five potential users and their remarks have been noted. The results were positive – there were no major problems with usability and users were satisfied with the application.

The points listed above have led to a initial goal accomplishment.

Chapter 7

Literature

[1] *Statista.com.* [online]. 2016 [quoted 2016–12–02]. Available at:
*https : //www.statista.com/statistics/*330695*/number−of−smartphone−users−worldwide/*

[2] AVILA, Jonathan. Trends in mobile device use by people with disabilities. *Access iQ.* [online]. 12.03.2014 [quoted 2016–11–05]. Available at:
*http : //www.accessiq.org/news/features/*2014*/*03*/trends−in−mobile−device−use − by − people − with − disabilities*

[3] REMUS, Horst. The origin of chess and the silk road. *The Silk Road.* [online]. [quoted 2017–01–08]. Available at:
http : //www.silk − road.com/newsletter/volumeonenumberone/origin.html

[4] SYSTEM MODELING. *SE Candies.* [online]. 23.11.2013 [quoted 2017–01–06]. Available at: *https : //secandies.wordpress.com/*2013*/*11*/*23*/system − modeling/*

[5] O nás. *SONS: Sjednocená organizace nevidomých a slabozrakých ČR.* [online]. 2012 [quoted 2016–12–28]. Available at: *https : //www.sons.cz/onas*

[6] How Visually Impaired Play Chess. *IBCA: International Braille Chess Association.* [online]. 2009 [quoted 2016–12–28]. Available at: *http : //www.ibca − info.org/how − visually − impaired − play − chess.php*

[7] New Release 33cm Chess Set for the Blind or Visually Impaired. *Chess Store.* [online]. [quoted 2016–11–09]. Available at: *http : //www.chessstore.com.au/new − release −* 33*cm −* 13 *− chess − set − for − the − blind − or − visually − impaired/*

[8] Chess–wise 3. *Apple Store.* [online]. [quoted 2016–11–08]. Available at: *https : //itunes.apple.com/us/app/chess − wise −* 3*/id*788594027*?mt* = 8

[9] KChess Elite. *ARK ANGLES.* [online]. [quoted 2017–01–02]. Available at: *http : //www.arkangles.com/kchess/elite.html*

[10] STILLER, Evelin, LEBLANC, Cathie. *Project Based Software Engineering: an object–oriented approach.* Boston, MA, USA: Addison–Wesley Longman Publishing Co., 2001 [quoted 2016–11–06]. ISBN 020174225X.

[11] Activity Diagrams : A Formal Framework to Model Business Processes and Code Generation. *The Journal of Object Technology.* Switzerland: Zürich : Chair of Software Engineering, ETH Eidgenössische Technische Hochschule, January–February 2009 [quoted 2016–12–10]. ISSN 1660–1769.

[12] ADDISON, Bill. The Cognitive Psychology of Chess. *Chess.com.* [online]. 21.06.2010 [quoted 2017–01–06]. Available at: https://www.chess.com/article/view/the-cognitive-psychology-of-chess

[13] CCRL 40/40 Rating List — All engines. *CCRL (Computer Chess Rating Lists).* [online]. 2005 [quoted 2016–12–15]. Available at: $http://www.computerchess.org.uk/ccrl/4040/rating_list_all.html$

[14] Efficient Generation of Sliding Piece Attacks. *Chess Programming Wiki.* [online]. 21.11.2008 [quoted 2016–12–15]. Available at: $https://chessprogramming.wikispaces.com/Efficient+Generation+of+Sliding+Piece+Attacks$

[15] Bitboard Board–Definition. *Chess Programming Wiki.* [online]. [quoted 2016–12–18]. Available at: $https://chessprogramming.wikispaces.com/Bitboard+Board-Definition$

[16] BROWNE, Cameron. *Bitboard methods for games.* QUT, Brisbane, Australia: ICGA Journal, 2015 [quoted 2016–12–19]. 2016–12–15ISSN 1389–6911.

[17] SAN SEGUNDO, Pablo, Ramón GALÁN, Fernando MATÍA, Diego RODRÍGUEZ–LOSADA and Agustín JIMÉNEZ. *EFFICIENT SEARCH USING BITBOARD MODELS.* Madrid, 2006 [quoted 2016–12–23]. Universidad Politécnica de Madrid.

[18] A. ELNAGGAR, Ahmed, GADALLAH, Mahmoud, ABDEL AZIEM, Mostafa and EL–DEEB Hesham. *A Comparative Study of Game Tree Searching Methods.* England and Wales: International Journal of Advanced Computer Science and Applications, 2014 [quoted 2016–12–23]. ISSN 2156–5570.

[19] PLAAT, Aske, Jonathan SCHAEFFER, Wim PIJLS a Arie DE BRUIN. *Exploiting Graph Properties of Game Trees.* Portland, Oregon, 1996 [quoted 2016–12–23]. Erasmus University, Dept. CS, Rotterdam, The Netherlands and University of Alberta, Dept. CS, Edmonton, AB, Canada.

[20] Search Tree. *Chess Programming Wiki.* [online]. [quoted 2016–12–26]. Available at: $https://chessprogramming.wikispaces.com/Search+Tree$

[21] Perft Results. *Chess Programming Wiki.* [online]. [quoted 2016–12–28]. Available at: $http://chessprogramming.wikispaces.com/Perft+Results$

[22] E. SHANNON, Claude. *XXII. Programming a Computer for Playing Chess.* Bell Telephone Laboratories, Inc. Murray Hill, NJ.: Philosophical Magazine, 1950 [quoted 2016–12–28]. ISSN 1478–6435.

[23] OWEN, Graham. *Search Algorithms and Game Playing: A general discussion on the minimax and alpha–beta search procedures and their application in computer programs that play chess.* 1997 [quoted 2016–12–30].

[24] Zero-Sum Game. *INVESTOPEDIA.* [online]. [quoted 2017–01–08]. Available at: $http://www.investopedia.com/terms/z/zero-sumgame.asp$

[25] KNUDSEN, Martin. Creating a chess engine (Part 2: Move search). *Chess.com.* [online]. 18.04.2010 [quoted 2017–01–02]. Available at: $https://www.chess.com/blog/zaifrun/creating-a-chess-engine-part-2$

[26] STREJCZEK, Marek. *Introduction – Historical look on chess programming.* Lódz, Poland, 2004 [quoted 2017–01–02]. Politechnika Łódzka.

[27] ZOBRIST, Albert. *A new hashing method with application for game playing.* Madison, Wisconsin, 1970 [quoted 2017–01–02]. The University of Wisconsin.

[28] WARKENTIN, Jonathan. Transposition Tables Zobrist Keys – Advanced Java Chess Engine Tutorial 30. *Youtube.* [online]. 24.06.2014 [quoted 2017–01–03]. Available at: $https : //www.youtube.com/watch?v = QYNRvMolN20t = 2s$

[29] BERENT, Adam. Guide to Programming a Chess Engine. *AdamBerent.com.* [online]. 24.11.2014 [quoted 2017–01–03]. Available at:
$http : //www.adamberent.com/documents/GuideToProgrammingChessEngine$
.pdf

[30] FEN Made Easy. *chessgames.com.* [online]. [quoted 2017–01–04]. Available at: $http : //www.chessgames.com/fenhelp.html$

[31] KAHLEN, Stefan–Meyer. The UCI protocol as publiced by Stefan–Meyer Kahlen (ShredderChess). *WBEC Ridderkerk.* [online]. April 2004 [quoted 2017–01–04]. Available at: $http : //wbec - ridderkerk.nl/html/UCIProtocol.html$

[32] MAETSCHKE, Klein. *Stockfish Chess.* [online]. 2010 [quoted 2017–01–04]. Available at: $https : //stockfishchess.org/$

[33] TCEC Web GUI. *TCEC Top Chess Engine Chapmionship.* [online]. 2010 [quoted 2017–01–05]. Available at: $http : //tcec.chessdom.com/archive.php?se = 9st = fga = 15$

[34] S. DUMAS, Joseph, C. REDISH, Janice. *A Practical Guide to Usability Testing.* UK: Intellect Books Exeter, 1999 [quoted 2017–01–05]. ISBN 9781841500201.

Appendix A

Use cases

A.1 Start new game

ID	1
Use case name	Start new game
Actors	User
Description	Player starts new game
Trigger	Player opens the app or clicks “New game”
Preconditions	User must have software installed and running
Postconditions	New game is started and player/s can play
Basic flow	1. Player opens the app 2. The board is presented to a player in a default state, human–PC game mode is enabled and white colour is assigned to a player
Alternative flows	1. Player clicks “New game”
Exceptions	
Includes	

A.2 Select chess piece

ID	2
Use case name	Select chess piece
Actors	User, AI engine
Description	Player selects a piece to make move
Trigger	Game is in progress
Preconditions	Player clicks on desired piece, holds it and taps somewhere else with his another finger
Postconditions	1. Piece is selected 2. IF player sets piece on desired position THEN move is made ELSE IF player chooses another piece of his colour THEN selected piece is changed to the one just picked
Basic flow	1. Player selects desired chess piece
Alternative flows	
Exceptions	Player selects an opponent's piece or an empty square
Includes	Make move

A.3 Make move

ID	4
Use case name	Make move
Actors	User, AI engine
Description	Player makes move with a selected chess piece
Trigger	IF player is user THEN player clicks on a desired chess board square ELSE square is assigned a chess piece programmatically
Preconditions	Player has chess pieces selected
Postconditions	1. IF chessboard square contains opponent's piece THEN opponent's piece is captured ELSE player's chess piece is set on new position 2. IF human–PC game mode enabled THEN the boards flips
Basic flow	1. Player chooses chess board square 2. Player moves selected piece to that square
Alternative flows	
Exceptions	IF chessboard square contains a chess piece that has the same colour as the player THEN player's selected piece is changed to the one in the square
Includes	Select chess piece

A.4 Select human–human/human–PC game mode

ID	5
Use case name	Select human–human/human–PC game mode
Actors	User
Description	Player selects game mode
Trigger	Player clicks “Game mode” in Settings menu
Preconditions	User has software installed and running
Postconditions	1. New game is started with a mode selected 2. IF game is running with a game mode that is different to a mode that has just been selected THEN new game is started ELSE initial game continues
Basic flow	1. Player clicks “Settings” 2. Player finds “Game mode” in Settings menu and selects human–human or human–PC
Alternative flows	
Exceptions	IF chessboard square contains a chess piece that has the same colour as the player THEN player’s selected piece is changed to the one in the square
Includes	Start new game

A.5 Select player colour

ID	6
Use case name	Select player colour
Actors	User
Description	Player selects her/his colour
Trigger	Player clicks “Player colour” in Settings menu
Preconditions	Human–PC game mode is enabled
Postconditions	1. Player’s colour is changed to desired 2. IF player’s colour is different to a colour that has just been selected THEN new game is started ELSE initial game continues
Basic flow	1. Player clicks “Settings” 2. Player finds “Player colour” in Settings menu and selects desired colour
Alternative flows	
Exceptions	Player wants to select player colour when human–human game mode is enabled
Includes	Start new game

A.6 Select level of the game

ID	7
Use case name	Select level of the game
Actors	User
Description	Player selects level of the game
Trigger	Player clicks “Game level” in Settings menu
Preconditions	Human–PC game mode is enabled
Postconditions	1. Game level is changed to desired 2. IF game is running with a level that is different to a level that has just been selected THEN new game is started ELSE initial game continues
Basic flow	1. Player clicks “Settings” 2. Player finds “Game level” in Settings menu and selects desired level
Alternative flows	
Exceptions	Player wants to select player colour when human–human game mode is enabled
Includes	Start new game

Appendix B

Testing

B.1 Tasks

1. Listen to manual.
2. Pick up a pawn.
3. Move this piece two squares ahead.
4. Play until you capture a piece.
5. Find out the current state of the game.
6. Undo your last move.
7. Start new game.

B.2 Questions

B.2.1 Screener

1. How old are you?
2. What kind of job do you have?
3. Do you use a smart phone?
4. Have you ever participated in usability testing?
5. Do you play chess?

B.2.2 Pre–test questionnaire

1. How do you prefer to play chess (some app, real chess game)?
2. How often do you play chess?
3. What smart phone do you use?
4. What would you like to improve in it?
5. What do you expect from my app?

B.2.3 Post–test questionnaire

1. Were tasks hard to fulfill?
2. Which was the hardest?
3. What is your overall impression from this application?
4. What would you improve?
5. Would you prefer to use another app for playing chess or play with a person to "Blind Chess"?

B.3 Collected data

B.3.1 Participant 1

Sand–blind.

Screener

1. 37.
2. IT specialist in SONS.
3. Yes.
4. Yes.
5. Yes.

Pre–test questionnaire

1. Real chess.
2. Not often.
3. iPhone and Android.
4. A lot of things. For example, to ease the way I call someone.
5. Make chess game on a smart phone available for the visually impaired people.

Post–test questionnaire

1. No.
2. None.
3. I was pleasantly surprised. This app turned out to be actually useful for the visually impaired.
4. I would shorten the names of control buttons. E.g., “Nová” instead of “Nová hra”, or “Zrušit” instead of “Zrušit tah” so it takes less space. It would be best if when I slide across chess board squares each one of them was highlighted with a bright colour creating a focus. I would also make chess pieces bigger and the white ones even more whiter, so I can see them better.
5. I would totally like to try using this game.

B.3.2 Participant 2

Blind.

Screener

1. 46.
2. Administration of the library in SONS.
3. Yes.
4. Yes.
5. Yes.

Pre–test questionnaire

1. Real chess games.
2. Rarely.
3. iPhone.
4. I would like to add text editing.
5. I expect that I will be able to play chess wherever I want.

Post–test questionnaire

1. No.
2. To move pieces was the hardest. Chess squares are too little.
3. It is fine.
4. I would make chess squares bigger and implement entering moves from keyboard.
5. I would have to learn all the gestures in order to play chess using your app. So for now I'd prefer real chess.

B.3.3 Participant 3

Blind.

Screener

1. 27.
2. Entrepreneur.
3. No.
4. No.
5. Yes.

Pre–test questionnaire

1. Real chess.
2. One a month.
3. I don’t use smart phone.
4. I don’t use smart phone.
5. That I will be able to play chess anywhere I want.

Post–test questionnaire

1. No.
2. To pick a chess piece.
3. Positive.
4. I would make squares bigger.
5. I would try using your app.

B.3.4 Participant 4

Blind.

Screener

1. 62.
2. I'm on retirement.
3. Yes.
4. Yes.
5. Yes.

Pre–test questionnaire

1. I play blindfold.
2. Two times a month.
3. Android.
4. I would like to improve an alarm.
5. I expect this app will bring me an opportunity to test different beginning patterns in chess game.

Post–test questionnaire

1. No.
2. I had problems with making a move.
3. I like it. But i would have to learn how to select things.
4. I would change the way the app is telling me about captured pieces. I would do it in the following way: "Odstraněné bílé figurky: 1 pěšec, odstraněné černé figurky: 1 dama".
5. I don't need nor boards, nor apps. I play blindfold.

B.3.5 Participant 5

Sand–blind.

Screener

1. 55.
2. I'm on retirement.
3. Yes.
4. No.
5. Sometimes.

Pre–test questionnaire

1. If I play, I prefer real chess.
2. Almost never.
3. iPhone.
4. Nothing.
5. I expect that people will be able to play chess anywhere. And maybe online.

Post–test questionnaire

1. No.
2. To select chess piece. To remember the last move.
3. It's good.
4. To add the information about the last move.
5. I don't know any other apps and I almost don't play real chess. So I would probably try using your app.

Appendix C

Empirical part

1. blindChess.zip – my application
2. thesis.zip – tex source files
3. assignment.pdf – assignment of this thesis