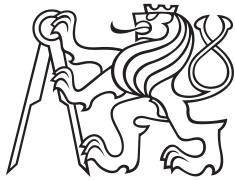


Bachelor thesis



**Czech
Technical
University
in Prague**

F3

**Faculty of Electrical Engineering
Department of Cybernetics**

Development of games for blind users

Denis Řeháček

**Supervisor: doc. Ing. Daniel Novák, Ph.D.
May 2017**

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Control Engineering

BACHELOR PROJECT ASSIGNMENT

Student: **Řeháček Denis**

Study programme: Cybernetics and Robotics
Specialisation: Systems and Control

Title of Bachelor Project: **Development of games for blind users**

Guidelines:

1. Familiarize yourself with user centered UI design
2. Implement 2-3 games based on adversory search in state space (e.g. tick tack toe, checkers etc.)
3. Implement the games in JAVA language
4. Test the games on a sample of 5 blind users

Bibliography/Sources:

- [1] Audio-Based Puzzle Gaming for Blind People, Jaime Carvalho, Tiago Guerreiro, Luis Duarte, Luis Carriço, University of Lisbon, 2012
- [2] Game Accesibility guidelines, <http://gameaccessibilityguidelines.com/>, 2016
- [3] Elmar Krajnc, Johannes Feiner, Stefan Schmidt, User Centered Interaction Design for Mobile Applications Focused on Visually Impaired and Blind People, FH JOANNEUM University of Applied Sciences, Werk-VI-Strasse 46, A-8605 Kapfenberg, Austria, 2010

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

Valid until the summer semester 2017/2018

L.S.

prof. Ing. Michael Šebek, DrSc.
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, February 21, 2017

Acknowledgements

I express my gratitude to my supervisor doc. Ing. Daniel Novák, Ph.D. and his whole team. Furthermore I express thanks to people of SONS organization for cooperation on user interface design and for testing the final applications.

Finally I thank to my friends and family especially my mom for supporting me during all my studies.

A special thanks goes to Mr. President Václav Havel for the opportunity to study what I want in a democratic country.

Declaration

I declare that this work is all my own work and I have cited all sources I have used in the bibliography.

Prague, May 14, 2017

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 14. května 2017

Abstract

This thesis describes analysis and implementation of smartphone games designed for visual impaired people. There was an emphasis on the user interface, sounds and text-to-speech output. Two board games were created - Battleship and Gomoku. Both applications are developed under Android mobile platform.

Keywords: visual impaired, game accessibility, Android, Battleship, Gomoku, AI

Supervisor: doc. Ing. Daniel Novák, Ph.D.

Abstrakt

Tato práce popisuje analýzu a implementaci mobilních aplikací navržených pro zrakově postižené. Důraz byl kladen na uživatelské rozhraní a zvukový výstup. Byly vytvořeny dvě deskové hry - lodě a piškvorky. Obě aplikace jsou vyvinuty pro mobilní platformu Android.

Klíčová slova: nevidomí, hra pro zrakově hendikepované, Android, lodě, piškvorky

Překlad názvu: Vývoj her pro slepé uživatele

Contents

1 Introduction	1	7.1.1 On the fly testing	27
1.1 Motivation	1	7.2 Usability testing	28
1.2 Goal of this work	2	7.2.1 Usability lab	28
2 Smart phone accessibility focused on visual impaired and blind people	3	7.2.2 Participants	28
2.1 Operation system support	3	7.2.3 Questionnaire	28
2.2 Current UI	4	7.2.4 User testing output	29
2.2.1 Touch control	4	8 Conclusion	33
2.2.2 Text-to-speech	5	8.1 Development for Android platform	33
3 Current status	7	8.2 Development for visual impaired	33
3.1 Current games	7	8.3 Reflextion	33
3.1.1 A Blind Legend	7	Bibliography	35
3.1.2 Blind Runner	7	A Information about participants	37
3.1.3 Board games	8	B Pre-test questionnaire	39
3.2 SONS	8	C Post-test questionnaire	41
4 Games control	9	D Attachments	43
4.1 Common control	9		
4.2 Gomoku specific control	9		
4.3 Battleship specific control	10		
4.3.1 Placing ships	10		
4.3.2 Placing ships - new version . .	11		
4.3.3 The battle	11		
4.3.4 The battle - new version	11		
5 Gomoku	13		
5.1 History	13		
5.2 Rules	13		
5.3 Gomocup – The Gomoku AI Tournament	13		
5.3.1 Gomocup communication protocol	13		
5.3.2 Wine - the Gomoku engine . .	15		
5.4 Gomoku developing	16		
5.4.1 Gomoku class diagram	17		
5.4.2 Minimax	17		
5.4.3 Alpha-beta pruning	19		
5.4.4 Running C++ code on Android	21		
6 Battleship	25		
6.1 History	25		
6.2 Rules	25		
6.3 Battleship developing	26		
6.3.1 Strategy	26		
7 Testing	27		
7.1 User testing	27		

Figures

2.1 Example of Current UI.	4
4.1 Example of Gomoku board.	10
5.1 View and controller diagram.	16
5.2 Game tree.	17
5.3 Game tree after Helen first step.	18
5.4 Game tree after Stavros first step.	19
5.5 Game tree of the perfect game.	19
5.6 Alpha beta pruning.	20

Tables

2.1 OS accessibility.	3
5.1 Examples of the communication protocol commands.	15
5.2 Primitive types that match up with Java equivalents.	22
6.1 Names and sizes of the ships.	25
A.1 Information about participants.	37
A.2 Shortcut explanation.	37

Chapter 1

Introduction

All over the world there are 285 million visually impaired people. The International Classification of Diseases describes 4 level of visual function:

- A normal vision
- B moderate visual impairment
- C severe visual impairment
- D blindness.

Visual impairment is describe as a decreased ability to see not fixable by glasses, contact lenses etc. As of 2014 there were 246 million people had moderate or severe visual impairment and 39 million were blind.

Approximately 90% of visually impaired people live in developing countries and 65% are people of age 50 and over. Nowadays about 20% of the world's population is in that age category. Due to aging of develop world population and ongoing demographics revolution the percentage of 50+ age group may increase in next decades.[5]

1.1 Motivation

A statistic based on screen reader survey [6] claims that manority of respondents (53%) owns a smartphone, regular phone were owned by 32% of respondents and 37% are tablet users. 72% use a screen reader in some kind of mobile device.

The survey was conducted on 1465 valid responses. 95% of them claim there were using a screen reader due to a disability.

Furthermore, there were 1.86 billion of smartphone users worldwide in 2015. The number is constantly rising especially in developing countries where lives the risk group.

Consider expectation of increasing number of visual impaired people it seems to be desirable open up as much smartphone's features as possible for this community.

■ 1.2 Goal of this work

The goal of this work is developed mobile phone games suitable to be played by visual impaired and blind people.

Guidelines:

- Implementing 2-3 games based on adversary search in state space (e.g. tick tack toe, checkers etc.).
- Implementation the games in JAVA language for Android operation system.
- Collaborate with SONS (Sjednocená organizace nevidomých a slabozrakých České republiky)
- Create a clean accessible design
- Test the games on a sample of 5 blind users.

Chapter 2

Smart phone accessibility focused on visual impaired and blind people

At the first sight it might sounds ridiculous to recommend a touch screen smartphone to visual impaired people but it turns out it is not. Smartphone and other modern technology can improve their life and integrate them to the society.

2.1 Operation system support

All current operation systems come with accessibility services created to help visual impaired people to control the phone providing several features.

Name	Description
Large text	sets larger font for most of the device texts
TalkBack	screen reader on Android
VoiceOver	screen reader on iOS
Narrator	screen reader on Windows Phone
Text-to-speech output	tool that reads any highlighted text
High contrast	sets clean design with high contrast
Magnifier	enable to zoom a part of the screen

Table 2.1: OS accessibility.

All of these function make the phone UI sort of suitable for visual impaired or even for blind people. However it seems not to be the best solution. Even with all the accessibility services turn on a lot of native application shipped on Android iOS devices are inaccessible or just not meet with productivity requirements of many visually impaired users.[6]

Finally, as an another possibility there is complete user interface design for visual impaired people created by Petr Svobodník (hereinafter referred to as the "current UI"). [7]

2.2 Current UI

Convention system accessibility stand on TalkBack (VoiceOver/Narrator) function. All controls staying the same. That is quite impractical for visual impaired who are forced to move all over the display to find all the controls on current screen. This makes the phone control slower than old button phone with text-to-speech function.

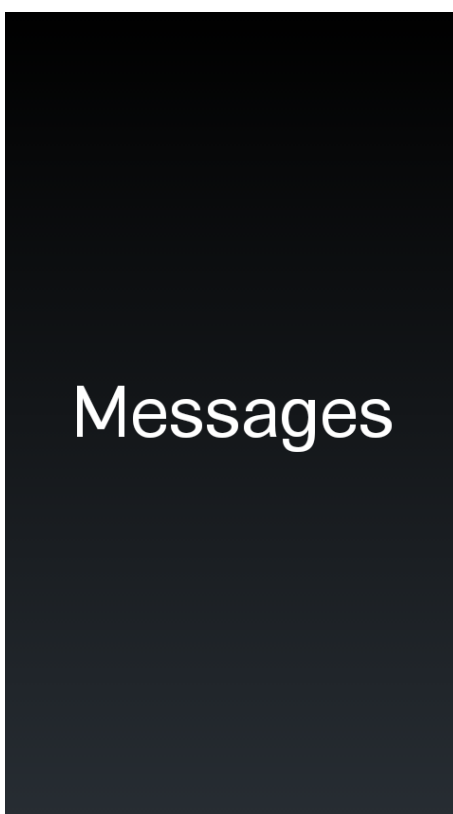


Figure 2.1: Example of Current UI.

2.2.1 Touch control

The touch control in the current UI 2.2 is design to be as simply and understandable as possible. Each item of a menu has its own screen with its name in the middle written in large font and white color on black background. Every menu items and phone actions are read by text-to-speech function 2.2.2.

■ Menu gestures

The UI is operated by few simply gestures:

One finger tap

This gesture is for navigation in menus or lists. The screen is vertically divided into two parts - left and right. Tapping left part moves the user to previews item, tapping right to next item.

One finger hold

Holding finger on screen for at least half a second stands for confirmation of present item.

Two finger hold

Basically an opposite of previous gesture. Holding two finger on screen for at least half a second moves user back in the menu, rejects call, alarms etc.

■ Keyboard gestures

Slightly different gestures are used for operating a keyboard. By touching the screen with one finger, the phone reads a key (if any) on that point. A confirmation of the key is done by tapping the screen with second finger about an inch or 1.5 cm next to the first finger.

■ 2.2.2 Text-to-speech

Also known as Speech synthesis, is the artificial production of human speech.

Text to speech is included in Android SDK since API level of 4 and is used in current UI 2.2 as well.

Chapter 3

Current status

As written in previous chapter, there are many alternatives for every day applications blind and visual impaired people mind need. But the situation is getting worse when comes to games. TalkBack (VoiceOver, Narrator) function are usually unavailable in most of the games which makes them unusable by the community.

3.1 Current games

Nowadays it is pretty hard to find any mobile phone game that could be controlled by a visual impaired person.

3.1.1 A Blind Legend

1

A Blind Legend is audio-only adventure game. The users orient in 3D environment based on sounds only, no visual art can be found. To recognize the source of a sound in 3D space headphones are compulsory. A movement is controlled by a joystick on the phone's screen.

Since the game was installed on more than 500 000 devices [2] it is obvious that games aimed on visual impaired people are attractive.

3.1.2 Blind Runner

2

Another audio-only adventure game similar to A Blind Legend. A movement is control by easy gestures. By a sound responds user can determinate if is getting closer or farther away to a danger etc. After each step a sound of steps is played or sound of collision if user crashes into a obstacle.

¹<https://play.google.com/store/apps/details?id=com.dowino.ABlindLegend>

²<https://play.google.com/store/apps/details?id=osphi.blindrunner>

■ 3.1.3 Board games

As suitable games for visual impaired seems to be board games. We can find several Chess implementation one of them [3] based on the current UI 2.2.

Another accessible board game is RapiTap!³. In this game whole screen is divided into several squares. An user is supposed to click on a square as fast as possible after its coordinate is read. A finale score is a summation of tapped squares in given time.

■ 3.2 SONS

SONS - United Organization of Blind and Visually Impaired Czech Republic is an organization helping blind and visual impaired community. The organization provides informations, advice, training, seek employment, train guiding dog and also gives consultation to developers.

The people in SONS were sceptical about our first idea to develop a chess or a checkers due to quite a difficult rules in these games. Many people could be rebound by getting read all the rules before they would even started to play. After all, we decided to develop two other games: Gomoku and Battleship. These two games are already well known in the “paper version” and has much simpler and more understandable rules.

³<https://play.google.com/store/apps/details?id=au.com.twentytwopoint.rapitap>

Chapter 4

Games control

To preserve cohesive control of the current UI 2.2. All game menus were implemented according to its gestures 2.2.1.

The challenge comes with the game board control. How to guide the user through a game board? How to announce opponents move? How to start new game? And many other question comes to mind. However most of them were already answered in Dina Chernova' work [3].

4.1 Common control

The control is inspired by the keyboard gestures 2.2.1. While swiping a finger on the screen. The phone is continuously reading each square of the board currently pointing at. If the user is not pointing at either the board or any control button underneath the phone just wait until the user simply swipe into one of them.

The selection of square or button is done by tapping the screen with second finger about an inch or 1.5 cm next to the first finger 2.2.1. This gesture produce corresponding reaction of the game.

Since each game has it owns specifics the advance control is described individually.

4.2 Gomoku specific control

In the game of Gomoku the control is intuitive. The selection gesture perform on a square adds players stone on it. After the player make his move, the Wine engine5.3.2 analyzed the board and make its move as well. The engine's move is announce by text-to-speech service 2.2.2.

When the game comes to its end the service announces a winner of the match.

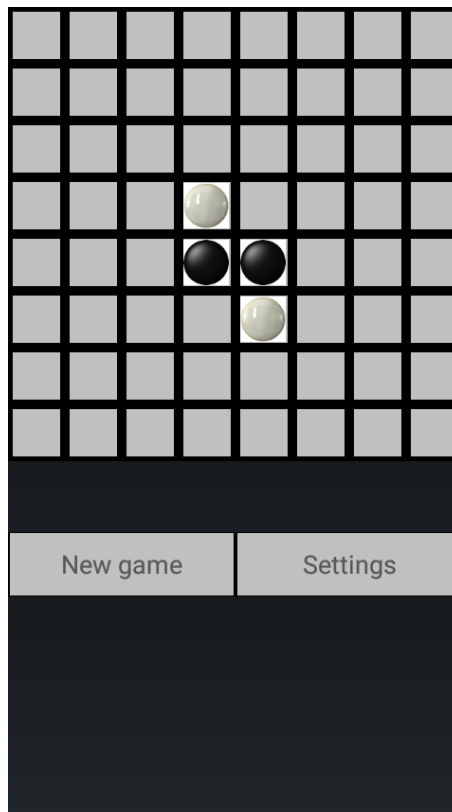


Figure 4.1: Example of Gomoku board.

4.3 Battleship specific control

As described 6.2 the Battleship game is divided into two parts, placing ships and the battle.

4.3.1 Placing ships

For the first part of the game following control was suggested.

1. First selected square is the starting square of the ship. The application do not let user to selected a square where is not enough free space for the currently placing ship. Such as squares between already placed ships and edges of the board if the space between is smaller then the ship's length.
2. While swiping on the board, the phone is announcing facing of the ship - vertically, horizontally or other (that stand for unsupported facing - diagonal or out of board).
3. Submitted the facing, the ship is finally placed.
4. If all the ships are placed, game automatically changes it state to a battle. Otherwise continue with step 1.

■ 4.3.2 Placing ships - new version

After consultation with visual impaired expert in SONS 7.1.1 new version of the placing ship was design.

The ships are sorted in list by length in decreasing order. An empty board is drawn.

1. The biggest ship is taken from the list.
2. Name of the ship and number of unplaced squares is read.
3. User to choose a square on the board where want to place another piece of the ship. It is checked if the square is valid. It means there must be enough space for whole ship. If one or more pieces of the ship are already placed the new one must be bordering with one of them. If two and more pieces are placed the new one must respect its facing. That means if the board is placing horizontally, the new piece must border horizontally as well.
4. If the last piece was not the last unplaced piece of the ship go to step 2.
5. If there is another unplaced ship left got to step 1.
6. The game state is automatically change to a battle.

■ 4.3.3 The battle

At first, the start of the battle is announce. An empty board is displayed and user guest a square where could an opponent ship been hiding by selection gesture 2.2.1. The attack result is announce.

Second the opponent's attack is read and the user board is drawn so the user can check the status of his army.

When the game comes to its end the service announces a winner of the match.

■ 4.3.4 The battle - new version

After consultation with visual impaired user 7.1.1 new version of a battle was design. To keep the simpleness of the control it is no longer possible to switch to users board during the battle. Instead only one empty board is drawn. While swiping finger over the board a square currently pointing is read. If it is not attacked square only the position is read otherwise is read the position and the square's state - HIT or MISS.

After every opponents attack current status of the battle is read. The status of each player is qualified by number of ships pieces that were not hit yet.

■ Sound response

Besides the text-to-speech function 2.2.2 sound respond was implement to announce an attack instead of reading “HIT” or “MISS”. If a player MISS, a sound of a stone falling into water is played. In case of HIT the phone responds with sound of an explosion.

Chapter 5

Gomoku

Gomoku [8] is an abstract strategy board game. Also known as Five in a Row, Go Bang, Pente or Go-Moku. Tick Tack Toe is just a simplified version on 3x3 board. The original game was played with Go pieces (black and white stones) on 19x19 or 15x15 board. Today is often played on board of many sizes and may be played on paper instead of the traditional GO board.

5.1 History

Gomoku is a very old game with its origin in Heian period of Japan. Gomoku spread to the world by its introduction to Britain in 19th century.

5.2 Rules

Two players are placing (or writing in the paper version) stones one by one on empty places on the playing board. This winner is the first player to achieve an unbroken horizontal, vertical or diagonal row of (usually) five stones.

5.3 Gomocup – The Gomoku AI Tournament

Gomocup is an international tournament of AIs playing gomoku which takes place every year since 2000 [9]. All the AI competing in the tournament must implement a simple communication protocol.

5.3.1 Gomocup communication protocol

The protocol [10] describes the communication between a manager (the controller of the game) and a brain (an AI playing the game). The brain must implement the following commands: START, RESTART, BEGIN, TURN, BOARD, TURN, END and optional INFO.

■ **START** [size]

The brain receiving this command is supposed to initialize itself, create an empty board of the size passed as the first argument after the command. In the Gomocup tournament only boards of size 20 are used but it is recommended to support other board sizes.

■ **RESTART**

This command is basically the same as the START command but has no parameters. The board size and rules remain from the previous game. The brain prepares itself for a new game.

■ **TURN** [x] [y]

The parameters are the coordinates of the last opponent's move (indexed from zero). The brain's move is expected after this command.

■ **BEGIN**

This command is sent to one brain only at the beginning of a match. The brain responds with its first move on an empty board. Then the game continues by sending TURN commands.

■ **BOARD**

```
[x] [y] [p]
[x] [y] [p]
...
DONE
```

This command describes a board already containing some stones. It may be sent after START or RESTART command when the board is empty. Parameters x and y are the coordinates of a stone, parameter p determines its color. Value 1 stands for the brain's stone, value 2 for the opponent's stone. The command is ended by passing DONE.

■ **INFO** [key] [value]

Set of optional commands to be sent by the manager to the brain. Most of them set rules of given games (such as a time limit for each turn, time limit of a whole match, memory limit) or any other additional information. The brain can ignore them but if it breaks any limit passed in the INFO command, it automatically loses the whole match.

The possible keys are following:

- **timeout_scoreturn** - time limit for each move (milliseconds, 0=play as fast as possible)

- `timeout_scorematch` - time limit of a whole match(milliseconds, 0=no limit)
- `max_memory` - memory limit (bytes, 0=no limit)
- `time_left` - remaining time limit of a whole match (milliseconds)
- `game_type` - 0=opponent is human, 1=opponent is brain,2=tournament, 3=network tournament
- `rule` - bitmask or sum of 1=exactly five in a row win,2=continuous game, 4=renju
- `evaluate` - coordinates X,Y representing current position of the mouse cursor
- `folder` - folder for persistent files

ABOUT

As a response for this command brains is expected to give some information about itself. Such a name, version, author, contact,.. etc.

Some examples of the protocol communication are shown in table 5.1

Command	Response	Description
INFO 8	OK	Board size was set to 8x8
	ERROR	Unsupported size or unspecified error
TURN 5,5	5,6	Player played 5,5 - brain played 5,6
BEGIN	5,5	Brain begun by move 5,5
BOARD	4,5	This command describes current board
5,5,1	4,5	Player 1 has field 5,5
5,6,2		Player 2 has field 5,6
4,4,1		Player 1 has field 4,4
DONE		Brain (player 2) played 4,5
INFO timeout_turn	-none-	Set maximum time for a turn in milliseconds
END	-none-	Terminate the brain
RESTART	OK	Start game in last available configuration

Table 5.1: Examples of the communication protocol commands.

The Gomocup rules says AI must be Win32 compatible program supporting Windows 7 or higher on both platforms x86 and x64.

There are also some open source engines implementing the protocol.

■ 5.3.2 Wine - the Gomoku engine

Wine - an engine made by JinJie Wang developed in C++. The sources are available for free on author's gitlab [11].

5.4 Gomoku developing

The applications core is shown in following class diagram. 5.1

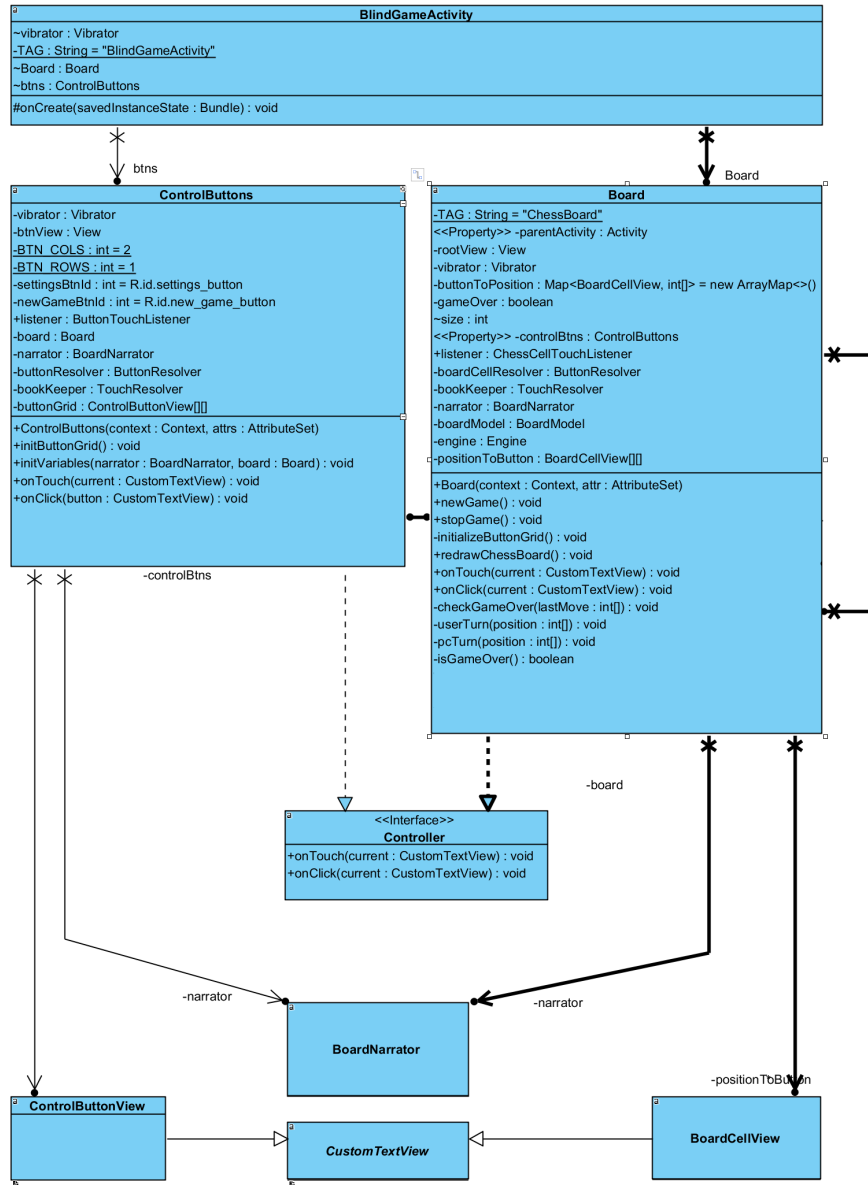


Figure 5.1: View and controller diagram.

Class diagram

Class diagram describes the application structure by showing its classes, attributes, methods and cross object relations.

5.4.1 Gomoku class diagram

The class `BlindGameActivity` contains two classes extending Android SDK's `LinearLayout - ControlButtons` and `Board`. Both of them also implement the interface `Controller` that describes two methods corresponding to the control as described in 4.1. The method `onTouch(CustomTextView current)` is called by a `TouchResolver` when the current `CustomTextView` is touched. The second one `onTouch(CustomTextView current)` is called when the current view is confirmed by tapping with a second finger.

The `ControlButtons` class resolves the input on application's control buttons, the `Board` class on each square of its board. A `Board`'s data model is stored in a `BoardModel` class. Finally, the `Board` contains an engine that computes computer's moves and can be easily replaced by another one or with an implementation of an online player.

Maybe the most important class is `BoardNarrator` which implements the sound response of the game.

5.4.2 Minimax

Games like Gomoku, Chess, Checkers etc are known as "games of perfect information", because it is possible to see all the moves in a game tree [12].

Game tree

Game tree is a directed graph whose nodes represent positions in a game and whose edges are moves. A node without any child is called leaf. Finally, the topmost node of a tree is root. 5.2

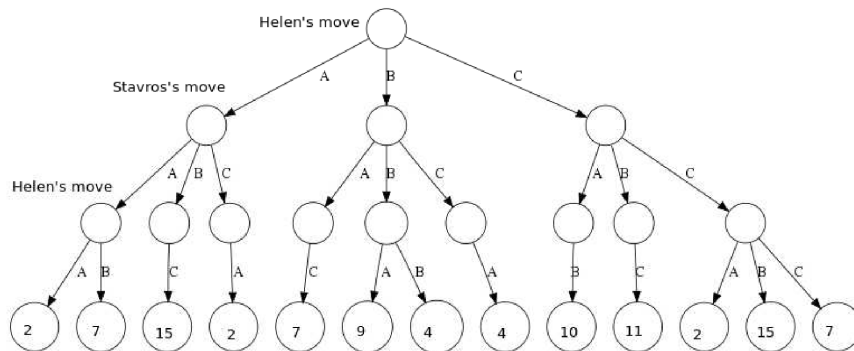


Figure 5.2: Game tree.

Minimax is an algorithm used as a base of many game artificial intelligence. The idea is "minimizing the possible loss for a worst case".

Formally written as:

$$v_i = \max_{a_i} \min_{a_{i-1}} v_i(a_i, a_{i-1}) \quad (5.1)$$

Let say we have a function describing score of current game state for both player, black (playing with black stones) and white (playing with white

stones). Furthermore let black player be a max player, white to be a min player. That means every step the black player needs to choose a move maximizing the outcome after the move. And of course, the white player is choosing a move minimizing the outcome score.

The minimax algorithm takes all the possible moves count, the score for each of them and choose the best. One player trying to gain as high score as possible the second one as low score as possible.

A function resolving a game score for each game state is call heuristic function.

To explain Minimax algorithm exploring game tree, lets consider any perfect information game with game tree 5.2, where the leaves correspond to heuristic function score. The game is played by two players, Helen and Stavros. Helen takes the first move A,B or C. She want to choose the move that maximized her score but she also realized Stavros will minimized her score in the next round. Therefore she needs to search every possibility recursively starting in the bottom.

In the first step 5.3 Helen maximized - chooses the path with the highest score. As can be seen in the left first two nodes of values 2,7 is the biggest value 7 so it is the choose. Same strategy is done on each path in this level of the tree.

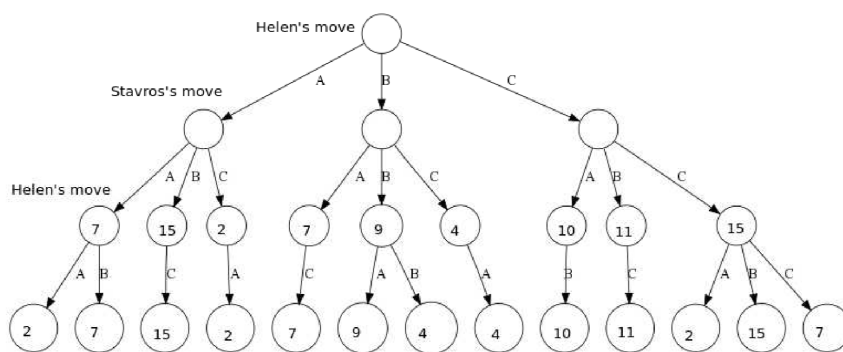


Figure 5.3: Game tree after Helen first step.

In the second step 5.4 Stavros minimized - chooses the path with the lowest score. For the most left node Stavros chooses from values 7, 15, 2. Trying to minimized Helen's score, value 2 is his move.

Now it is obvious choosing option C is the perfect step for Helen 5.5.

Helene repeats the maximizing strategy by choosing path C with the value of 10 which is her final score. The best one if both of the players were playing rational. We have consider the best strategy for Halen and Stavros as well. But if Stavros make an mistake it would be even better score for Helen.

If Stavros would not play the best he could then after Helen play C he would have choose B or C and the game would end by the score of 11 or 15. That would be better score for Helen and even worse score for Stavros obviously.

Now we have an algorithm that always choose the best move but also searches the whole game tree even if it is unnecessary.

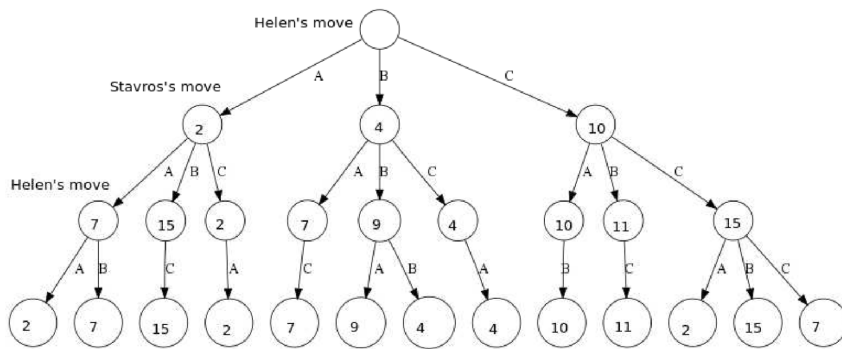


Figure 5.4: Game tree after Stavros first step.

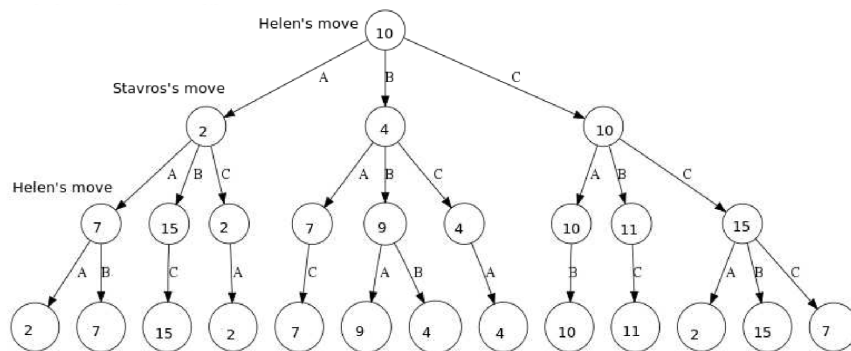


Figure 5.5: Game tree of the perfect game.

The Minimax tree grows exponentially with its depth. For a basic Gomoku 19x19 board it would search a state space of size $19! \times 19!$. Even for a simplified version with 8x8 the size of state space to be search in would be

$$19! \times 19! = 1625702400$$

It would take the algorithm long time to make even a simple decision.

However, efficiency of the Minimax algorithm without any influence on the result may be reach with combination of alpha-beta pruning.

5.4.3 Alpha-beta pruning

Alpha-beta pruning extends Minimax offering decreasing of nodes that are evaluated in given search three. It reach the same node[4] but usually in a shorter time due to pruning branches that do not influence the result.

The difference is that this algorithm stops to evaluation a move when reaching a score that is equal or worse than score of already found branch. This score is a proof than any other move underneath this branch has same or even worse score.

Back to our example 5.2 of Helen vs. Stavros game. Helen needs to knows values of the tree nodes connected from root, not whole tree. Starting from

left she finds a value of 7. Then in the middle branch there is a node of the same value. On the next level Stavros will try to minimize the score. So it is obvious he will not pick anything greater than 7 and also Helen will not pick anything less than 7. Having consider this fact, there is no point to evaluate the rest of middle the branch.

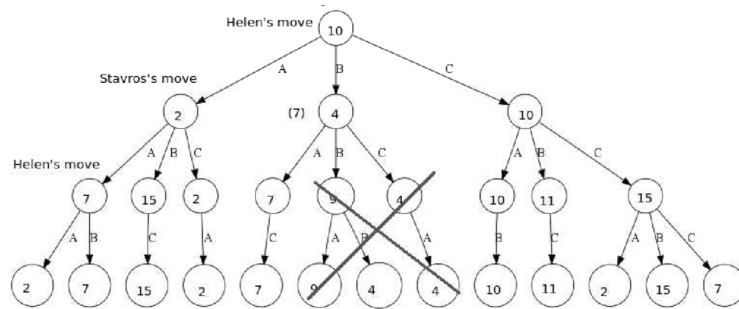


Figure 5.6: Alpha beta pruning.

The base implementation of minimax with alfa-beta pruning is following:

```
private int[] minimax(int depth, BoardModel model,
PieceColour player, int alpha, int beta) {
    //get all the possible moves in current model
    List<int[]> nextMoves = generateMoves(model);
    //init variable to default
    int score;
    int bestRow = -1;
    int bestCol = -1;

    /*
    * if reach the maximum depth or search over
    * all possible moves return best move
    */
    if ( depth == 0 || nextMoves.isEmpty() ) {
        score = score(model);
        return new int[] {score, bestRow, bestCol};
    } else {
        for (int[] move : nextMoves) { // evaluate all the moves
            model.setPiece(move[0], move[1], player);
            if (player == user) {
                score = minimax(depth - 1, model,
                    pc, alpha, beta)[0];
                if (score > alpha) {
                    alpha = score;
                    bestRow = move[0];
                    bestCol = move[1];
                }
            }
        }
    }
}
```

```

        score = minimax(depth - 1, model, user,
                        alpha, beta)[0];
        if (score < beta) {
            beta = score;
            bestRow = move[0];
            bestCol = move[1];
        }
    }
    model.removePiece(move[0], move[1]);
    if (alpha >= beta) { // alpha-beta pruning
        break;
    }
}
}
return new int[] {(player == user) ? alpha : beta,
                  bestRow, bestCol};
}

```

5.4.4 Running C++ code on Android

Using of the Wine engine written in C++ is possible due to Android Native Development Kit (NDK) [13]. Android NDK allows to run system native code written in C/C++. The library written in C/C++ can be called from Java code running under Android Virtual Machine (Dalvik) by following static call:

```
System.loadLibrary("libraryName")
```

Then it is possible to call C/C++ method using native methods. For communication between the Gomoku application and the Library following method were established.

```

/**
 * Native method to pass any command to the library.
 * @param command Command to be pass
 * @param value command value (optional)
 * @return library response
 */
public native String passCommand(String command, int value);

/**
 * Native method to pass TURN command to the library.
 * Coordinates of last move x,y are required.
 * @param x coordinate x of last move
 * @param y coordinate y of last move
 * @return coordinate of the Brain move
 */

```

```
public native int[] passTurn(int x, int y);
```

All the Gomocup engines are Win32 application controlled from command line. To use the Wine engine as a library the engine's controller had to be rewritten to enable communication with Java.

Another obstacle is a lack of compatibility between Java and C++ data types. To deal with this issue Android SDK comes with "jni.h" library providing primitive C types that match up with Java equivalents. Moreover some reference types, in C++.

Java data type	Native data type	Description
typedef unsigned char	jboolean;	unsigned 8 bits
typedef signed char	jbyte;	signed 8 bits
typedef unsigned short	jchar;	unsigned 16 bits
typedef short	jshort;	signed 16 bits
typedef int	jint;	signed 32 bits
typedef long long	jlong;	signed 64 bits
typedef float	jfloat;	32-bit IEEE 754
typedef double	jdouble;	64-bit IEEE 754

Table 5.2: Primitive types that match up with Java equivalents.

According to this knowledge, the C++ opposite of:

```
public native String passCommand(String command, int value);
```

may be declare as:

```
JNIEXPORT jstring JNICALL
Java_com_blindshell_denis_libgomoku_Wine_passCommand
(JNIEnv *env, jobject, jstring jcommand, jint value) {...}
```

Where "JNIEXPORT jstring" definite the return type as jstring - the Java String type. The method name specify whole path to the Class where passCommand(...) native method is declared. Method's parameter "JNIEnv *env" stand for pointer to Java environment used for returning values back to Java world.

Of course, the C/C++ library must be compiled for ARM processor family. Since the compilation of Java is done using Gradle it is natural it for this purpose as well. To do that following lines needs to be added to build.gradle file of the C/C++ module.

```
android {  
    ...  
    //common Android build script  
    ...  
    //adds native library to be build  
    externalNativeBuild {  
        cmake {  
            path "src/main/cpp/CMakeLists.txt"  
        }  
    }  
}
```

This script snippet show usage of **CMake** which calls its configuration file, so called CMakeLists.txt, where build properties of the C/C++ code is defined.

Chapter 6

Battleship

Battleship is a guessing board or paper game also known as Sea Battle.

6.1 History

The game has its origin in the game L'Attaque played in France during World War 1 and in the game Baslinda played in Russia since 1890. Battleship spread to the world by its first commercial version, published in 1931 in the United States of America.

It is also said to be one of the earliest computer games with first released in 1979 for the Z80 CompuColor.

6.2 Rules

Each player has its private board that none of its opponents is authorized to see. All the boards have fixed size, usually of 10x10 squares.

In the first part of the game each player secretly places ships on their private board. Ships can be arranged either vertically or horizontally and must not share a square with another ship. Number and length of the ships as well as the size of the board may vary depending on an agreement established before the match begins.

According to the 1990 Milton Bradley version each player has five ships of the following types and lengths:

Type	Length
Carrier	5
Battleship	4
Cruiser	3
Submarine	3
Destroyer	2

Table 6.1: Names and sizes of the ships.

After all these ships are placed the game itself can begin with a series of rounds. In each round, each player attacks his opponent by announcing a

square of the board. The opponent must respond by saying “HIT” if he had a part of any ship on that square and “MISS” otherwise.

Usually sunk ship should be announced too by saying (“You have sunk my Cruise” e.g.).

The goal is to sink all opponents ships [14].

6.3 Battleship developing

The Battleship game was patterned on the Gomoku. A view and a controller remaining almost the same. Only the board controller implements battleship specific control as described in 4.3 and its engine is completely different. But the architecture is preserved.

However the application data model is more complicated in this case. BoardModel must contain two boards now - user's and opponent's one. In Gomoku a board was defined by 2D array of pieces. In this game the board is extended by a list of ships.

A Ship is an abstract class storing positions of all its pieces. Each implementation must override abstract method `getLength()` describing ships length. Each piece now besides its owner provides a state (HIT or MISS) as well.

6.3.1 Strategy

The PC's ships are placed randomly on the board when starting a new game. During the battle when PC player is supposed to guess, it is obvious that it can know where the player's ship are. But keep it a game the PC player does not have this information.

An algorithm to solve a Battleship game has two parts. First it generates a list of all coordinates of the board.

While trying to find any piece of opponent's ship. It shoots randomly until it makes a HIT for the first time.

The number of hit pieces is stored. Then it is trying to sink the entire ship by shooting around that place. After a ship being sunk its length is subtracted from the number of hit pieces. If the number does not reach zero it is obvious that another ship was in that area and is not sunk yet. So it is shooting around that place until all known ships are sunk.

If the number of HITs reach zero, algorithm is choosing a random position again to find another ship.

And of course, after each HIT its coordinate is removed from the list to never shoot twice on the same place.

Chapter 7

Testing

The development of the games was done by continuous cooperation with SONS.

7.1 User testing

Since the games are aimed at visual impaired community the user testing was the number one priority.

7.1.1 On the fly testing

During to development semi-finished games were consulted with visual impaired people working at the SONS organization. This fact enormously help to create the final product, especially its control, truly adapted to visual impaired community.

Gomoku

The game of Gomoku was accepted with almost no reproach. Only an idea about a possibility of allowing users undo last move was decline. It would mean to add another control button which do not meet with the philosophy of keeping the game control as easy as possible.

Battleship

On the other hand the Battleship game has not met with so positive acceptance for the first time.

The half automatic control during placing ship 4.3.1 was found to be not so intuitive. A visual impaired user does not have the sureness where will be the ship actually placed by choosing only the first square and facing of the rest of the ship. It was found as stressful and uncomfortable.

Instead, new version of placing ship was suggested and later implemented as described in previous chapter 4.3.2.

Another reproach aimed to announcing of current state during a battle.

Reading “HIT” or “MISS” by electronic voice of text-to-speech function is in game environment not only slow but also boring. These small details degrade the overall feeling of the game.

To preserve a game atmosphere the reading of game state was replaced by playing sounds 4.3.4.

■ 7.2 Usability testing

Usability testing refers to evaluating a product by testing it on users. Participants are asked to perform a task on the product while an observer watches them to identify their reaction, listen and measure a time needed for the task.

The goal is to identify problems, determine participants' satisfaction and detect their feelings during usage of the product. Also to get direct input on how real users use it and finally, measure how the product meets its purpose.

Usability testing is not used in software development only but in many other fields such as food, consumer products, devices, documents etc.

■ 7.2.1 Usability lab

A usability test should take place in a usability lab which consists of two separated rooms - participants and observers room. In the first one participants perform given tasks. The room should simulate an environment where the testing product is expected to be used. The observers room is usually connected with the participants room by a one-way mirror or via live stream. In this room observers watch participants' reactions.

Instead of a formal lab as described previously tests of these applications were done in the community center SONS 3.2 and participants' homes because it was more comfortable for them. Also it gives an opportunity to observe participants' reactions to the product in their usual environment [1].

■ 7.2.2 Participants

Participants in usability testing should always correspond to a target group of the product and should cover the whole group.

In this case it means all participants are visually impaired but also cover several age groups (the youngest was 30, the oldest 62 years old) and two genders (3 men and 2 women).

Additional information about the participants can be found in appendix A

■ 7.2.3 Questionnaire

Besides participants' reaction during the testing another output of the test is seized by several questionnaires.

■ Screener

Screener refers to a questionnaire used for choosing right participants sample for given product. It also collects basic information about them like age, sex etc. This questionnaire was skipped due to already known sample of users who were testing other CTU product for visual impaired. The basic information were obtain in pre-test questionnaire instead.

■ Pre-test questionnaire

This questionnaire is given to the participants right before the testing intended to obtain further information about them. The questions are aimed to the testing product issues such as participants experiences with similar products etc.

The pre-test questionnaire can be found in appendix B.

■ Post-test questionnaire

Set of questions to be answered after finishing all tasks the test. The questions are relates to the tasks, tested product and its usability.

The post-test questionnaire can be found in appendix C.

■ 7.2.4 User testing output

Summarization of the main outputs from the user testing.

■ Orientation and board size

Orientation in a board of size 6x6 and bigger was found to be too difficult for two participants. One of them called for opportunity to read a whole row besides of reading single square on touch and for a small vibration when swiping from one square to another. Adding new control button for reading a row is in breach of keeping the control as simple as possible but another vibrate output seems to be a fair observation and was implemented in a following version. But these two users have no previous experience with the Current UI.

Large board in Gomoku game was also found to be too hard to remember for the two participants but they made a note that some blind people can play a game of Chess and remember the whole board. That was confirm by another participants who really play the Chess. For him and one more participants large board was making no problem. They even claimed that smaller boards have no point for them. This is solved by providing several board sizes. By summary of responds the choose of board size should be done before each game round not in game settings.

■ Board control

The control of the board itself was found easy for participant who already had some experiences with the current UI 2.2 due to similarity with controlling its keyboard 2.2.1. Others said that would need more time to get familiarized with the gestures and learn how to used them properly but believed that the control would make them no problem in future using.

■ Gomoku

The Gomoku itself was found hard for imagination but on the other hand good for a memory training. One participant cry for announcing the moment when last change for a win is lost.

A participants with moderate visual impairment claimed that can see black stones but not the white ones due to blending in with the grey board. This can be solve by replacing the drawable picture of white stone by more contrasting one.

■ Battleship

The Battleship game in general was found more entertaining then the Gomoku and surprisingly intuitive to control. A participant who works in SONS said she cannot imagine an elderly to play it but she mentions a blind client she worked with a week before who was only seven years old and claimed the game would be perfect for her. On the contrary the oldest participant thought back to time when he played the game and seemed to be interested.

■ Placing ships

A lack of information was found during placing ships. After last piece of a ship is placed only information about next ship was announced (its name and length). It made troubles with recognizing if a previous ship is placing or already a next one. As a solution was suggested to play a sound of hooting ship. This was implemented in a new version.

One participants found the ship placing a little knotty. He had problems with finishing it while part of a ship was already placed. Sometimes he get lost in a board and tried to place another piece on invalid position several times in a row. To avoid this situation a number of unplaced pieces is announced when tapping on a unfinished ship. If the issue remains a last placed piece's coordination could be announce when trying to place a next piece on invalid square.

He also cry for a hint where could he place next ship. This could be done by adding another control button that would read free positions after being clicked. But that would be long and boring. A better way could be to provide a list of possible position that user would be able to iterate with a possibility to confirm given position that would be automatically used.

As well as during the Gomoku one participants found the board pieces too small to control it without mistakes. This issue could be solve with providing a light weight version of the game played on smaller board with less ships.

An idea about a possibility to place ship in text mode (writing its coordinates on keyboard) was definitely decline by 3 of 5 participants. But the remaining two claimed it would be suitable. Both of them had previous experiences with playing the game in this mode. For now the issue was denied. The touch control was accepted positively by all participants and the text mode seems to be just a nostalgia and would break the rule of keeping the application as simple as possible.

■ Battle

One participant found as a big disadvantage the fact, that user is not notified about opponents move. After each step only a sounds of users and opponents attack is played but not the position of it. So the user has no opportunity how to read opponents tactics. Since only one of five respondent open this issue it was not solved yet. But it makes sense to open it again during future implementation of an online version.

Analogously to placing ship, two participants calls for a hint of closed squares (squares that were not attacked yet). It could be implemented by the same menu as were suggested during ship placing. A selection of a coordinate in the menu would started an attack to that square.

■ Online version

All five participants speak out in favour of implementing a possibility to play online against another player instead of an engine. Not only it will make the games more entertaining but creates a social aspect too, they claimed.

■ Is it worth it?

All participants were asked before testing the game if is it worth it to developed accessible smartphone's games for them. Three participants said it definitely is. Rest two said it is but not for them.

After the test they were asked once more. All of them claimed now that it is worth it. Even the two who did not believe in it first time. They said they would not play it on every day base but for example while waiting to a doctor or during traveling in public transport.

Chapter 8

Conclusion

This chapter summarizes the results of this work.

8.1 Development for Android platform

It turns up to be relaxing to use Android SDK. Easy understanding documentation excellently provides a developer through the system. Also it gives a solid base of components that can be easily reused. Another useful feature is Android NDK providing native code support making core of application more efficient.

8.2 Development for visual impaired

This showed up to be a real challenge for a man without any experience with working for visual impaired community. Big effort was given to keep the application's UI clean and simple, to create it both - fast to use as well as gives enough information by sound response.

Luckily, people of the SONS community have many experiences with designing of accessible projects and are open to share them.

8.3 Reflection

In contrary with recommendation in assignment game of checkers was not implemented due to its difficult rules. The case when user is forced to capture opponent's stone was found too demanding on user's imagination. Anyway Checkers were replaced by Battleship which were found much more user friendly.

Finally, rest of the recommendation was fulfilled and the games were accepted with pleasure. After trying the games five of five participants in usability testing claimed that they would appreciate them.



Bibliography

- [1] Carol M. Barnum, *Usability Testing and Research* [quoted 2017-05-14]. Allyn & Bacon, Inc. Needham Heights, MA, USA 2001, ISBN-13: 978-0205315192
- [2] Google Inc. *Google play, A Blind Legend, Additional information* [quoted 2017-05-14]. [online]. available at <https://play.google.com/store/apps/details?id=com.dowino.ABlindLegend>
- [3] Chernova Dina *Development of games for users with visual impairment* [quoted 2017-05-2]. [online]. Chernova Dina, Czech Technical University in Prague, January 2017 available at <https://dspace.cvut.cz/bitstream/handle/10467/66838/F3-BP-2017-Chernova-Dina-thesis.pdf>
- [4] Stuart Russel and Peter Norvig, *Artificial Intelligence - A Modern Approach*, [quoted 2017-05-2]. Prentice Hall, 3 edition, 2010. ISBN 860-1419506989
- [5] World Health Organization *Visual impairment and blindness* [quoted 2017-05-14]. [online]. available at <http://www.who.int/mediacentre/factsheets/fs282/en/>
- [6] WebAIM - web accessibility in mind *Screen Reader User Survey* [quoted 2017-05-14]. [online]. available at <http://webaim.org/projects/screenreadersurvey5/>
- [7] Bc. Petr Svobodník *Zpřístupnění mobilních telefonů se systémem Android pro nevidomé uživatele* [quoted 2017-05-2]. [online]. Bc. Petr Svobodník, Czech Technical University in Prague, May 2013 available at https://dip.felk.cvut.cz/browse/pdfcache/svobop24_2013dipl.pdf
- [8] *Gomoku - Japanese Board Game* [quoted 2017-05-12]. [online]. available at https://web.archive.org/web/20140326000833/http://www.japan-101.com/culture/gomoku_japanese_board_game.htm
- [9] *Gomocup* [quoted 2017-05-12]. [online]. available at <http://gomocup.org/detail-information/>

- [10] *Gomocup - New protocol* [quoted 2017-05-12]. [online]. Petr Lastovicka available at <http://petr.lastovicka.sweb.cz/protocl2en.htm>
- [11] *Wine engine* [quoted 2017-04-29]. [online]. JinJie Wang available at <https://github.com/jinjiebang/wine>
- [12] *Introduction to AI Techniques* [quoted 2017-05-14]. [online]. MIT, June 8, 2009 available at <http://web.mit.edu/sp.268/www/gamesearch.pdf>
- [13] Google Inc. *Android NDK* [quoted 2017-05-14]. [online]. available at <https://developer.android.com/ndk/index.html>
- [14] Wikipedia *Battleship (game)* [quoted 2017-05-10]. [online]. available at [https://en.wikipedia.org/wiki/Battleship_\(game\)](https://en.wikipedia.org/wiki/Battleship_(game))

Appendix A

Information about participants

	M	P	H	V	S
age	38	30	40	56	62
sex	male	female	male	male	female
tester	yes	yes	yes	yes	yes
visual experience	yes	yes	yes	yes	yes
sight loss	d	a	c	d	b
phone	A + TB, I + VO	A + magnifier	I + VO	A + Current UI	I + VO

Table A.1: Information about participants

The sight loss value corresponds to classification describe in introduction.

1

Shortcut	Explanation
A	Android
TB	TalkBack
I	iPhone
VO	VoiceOver

Table A.2: Shortcut explanation.



Appendix B

Pre-test questionnaire


- Your name (for internal needs only)
- Have you ever take a part in a software testing?
- How old are you?
- Do you have visual experience? (Means if you are blind is it lifelong blindness?)
- What is the classification of your handicap?
- What phone do you use?
- Would you appreciate the possibility to play games on your phone?
- Do you have idea about concrete game?
- Do you know Battleship or Gomoku?
- Have you ever play it? If you have how? Digital or board version?



Appendix C

Post-test questionnaire

- Have you got oriented easily on the game board?
- Have you retrieve enough information about opponent's move?
- How do you find the size of the board?
- What do you think about idea to start a game on small board and after winning or ending in a draw automatically switch to bigger board?
- Did you get the games control?
- How do you feel about placing a ship?
- Would you appreciate a possibility to place a ship in text mode by writing its coordination using keyboard?
- How do you find the battleship battle?
- What do you think about online version of the games?
- Do you have any ideas what could be improved?
- What is your general feeling about the application? Is it worth it for you?



Appendix D

Attachments

- battleship.zip sources of the Battleship application
- gomoku.zip sources of the Gomoku application
- thesis.pdf the thesis itself in PDF
- assignment.pdf the assignment of the thesis