Silesian University of Technology

Faculty of Automatic Control, Electronics and Computer Science

Engineer thesis

Accessible Sudoku

author: Adam Gajewski

supervisor: Krzysztof Dobosz, DSc PhD

Gliwice, January 2019

załącznik nr 2 do zarz. nr 97/08/09

Oświadczenie

Wyrażam zgodę / Nie wyrażam zgody* na udostępnienie mojej pracy dyplomowej / rozprawy doktorskiej*.

Gliwice, dnia 1 stycznia 2019

. . . .
(podpis)

. . . .
(poświadczenie wiarygodności podpisu przez Dziekanat)

* podkreślić właściwe

Oświadczenie promotora

Oświadczam, że praca „Accessible Sudoku” spełnia wymagania formalne pracy dyplomowej inżynierskiej.

Gliwice, dnia 1 stycznia 2019

. . . .

(podpis promotora)

Contents

Chapter 1

Introduction

- introduction into the problem domain
- settling of the problem in the domain
- objective of the thesis
- scope of the thesis
- short description of chapters

1.1 Introduction into the problem domain

Short description of accessibility

Accessibility is a feature that lets blind people to use smartphones. Even if an app was not designed for blind people, they mostly still can use it thanks to screen readers like Talkback (an accessibility service for Android) reading out loud the content of app to the user. It is a small help compere to many problems connected to being blind, but we can increase this help, by creating applications that are more accessible friendly.

Due to the fact that in order to know what is the content of something that is currently displayed on the screen, a specific part of the screen must be tapped, the application should not use the same event as the chose action. In Talkback this event is overwritten by double tap, so after first tap a button (for example) Talkback tells the user what was just tapped. After a double tap the application calls the same action as if this element was tapped once with Talkback turned off. Using Talkback changes this and many other behaviors of applications, which can make many applications unable to work properly. To change this back to how the application worked originally without Talkback, the application have to be designed either disable some of Talkback features for parts of itself, or overwrite its reactions to touch events.

The easiest accessible thing to implement is some text to read. To any other elements should be provided appropriate labels. Thanks to that most of visual features can be forgotten and the user can proceed to use the app. One of problems of this solution is time necessary for the audio to be played. Because of that application requiring faster response are a lot harder to be accessible, so there are not many accessible games on smartphones.

1.2 Settling of the problem in the domain

Short description of accessible games in general

Making an accessible game we can take one of two approaches: Either we make the UI simple and decrease number of elements on screen to minimum, or we take the risk of making something more complicated. In the first case we just have to think of an idea that does not require from the user to chose from many options or to react very fast. It leaves us with really lot of options: Quizzes, text RPG's, crosswords, and so on. Simple design does not make a game boring, everything depends on the execution. In the second case we must put trust in the user and make a bit more complicated UI with more elements. When the user will get used to the game this difference in difficulty should decrease, but the time for learning how to play will be usually longer, than in the first case.

1.3 Objective of the thesis

The main objective is to create a working accessible Sudoku game that can be solved both by a blind person and someone fully sighted.

Chapter 2

[Problem analysis]

- problem analysis
- state of the art, problem statement
- literature research (all sources in the thesis have to be referenced [1, 2, 4, 3])
- description of existing solutions (also scientific ones, if the problem is scientifically researched), algorithms, location of the thesis in the scientific domain

2.1 Problem Analysis

Description of some accessible solutions

The smartphone accessible games have been available for a few years, and the conventional ones were more common for even longer. Puzzles like Rubik cube with braille alphabet on it can be found easily, and there exists a similar solution for the Sudoku puzzle, with numbers represented by small blocks with braille number on it, that can be putted into the puzzle. That solution can not be used in an electronic device not having any form of system showing braille alphabet. There exists a solution on the descktop computer using keyboard and audio reading content of the sudoku, but agin nowadays smartphones usully do not have keyboaeds.

2.2 Description of existing solutions

Other accessible sudoku applications

Durring working on this thesis there heve not been found any accessible sudoku applications, and any of normal sudoku applications were compatible with the TalkBack.

Figure 2.1 normal sudoku application.

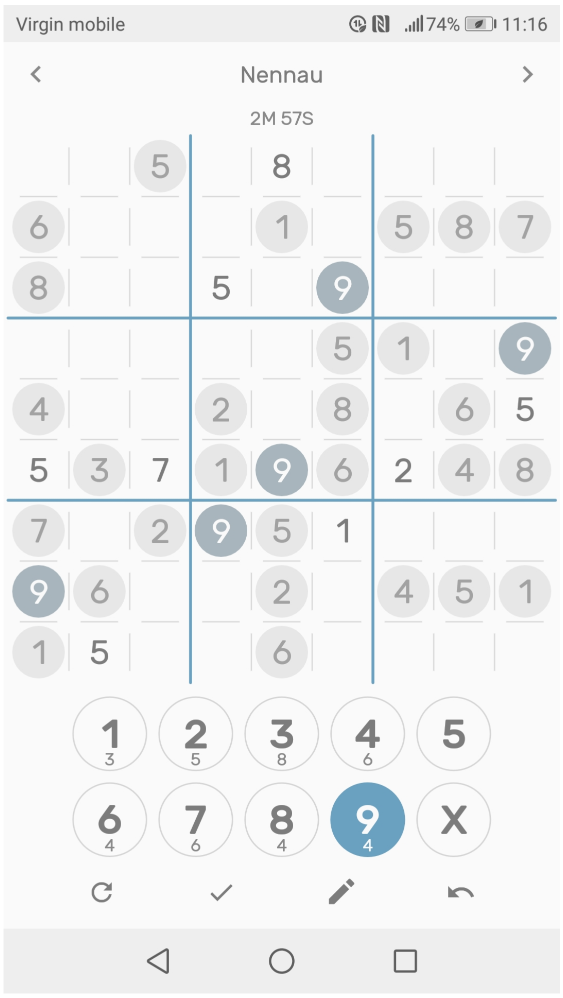


Figure 2.1: Sudoku.

Chapter 3

Requirements and tools

- functional and nonfunctional requirements
- use cases (UML diagrams)
- description of tools
- methodology of design and implementation

3.1 Functional requirements

There have been chosen to use both of previously mentioned approaches in the game, by creating a tutorial level with Sudoku 6 by 6 instead of 9 by 9. This way number of elements is decreased by over 55%, so there are supposed to be two algorithms implemented: one for 6 on 6 and one for 9 on 9. Beside that the application will also need a validation to see if the Sudoku was solved correctly.

Usage scenario:

1. Instruction on how to play is played to the user(he can skip it if he wants to).
2. User chose size of the Sudoku.
3. User fills in the Sudoku.
4. The program validates if the Sudoku is correct.

3.2 NonFunctional requirements

The user interface of the application should work similar to the way that Talkback is working:

- Taping an element of Sudoku (number) should result in the interface reading the label to the user.
- To edit an element, user must hold finger on it for a longer period of time, then the content of the element will be changed to the number that the element was tapped.
- In both cases, when the user try to edit a starting element (element that the game have started with), or when the user taps an empty element, a short message will be played to inform about it.

To make it easier for blind user to find number on the screen of smartphone, and be aware of its localization,

- It should be possible to zoom in to one cell with only 6 or 9 elements (depending on the type of Sudoku).
- It would be possible to move between cells by swipe events, and another after zooming out the swipe event can be used to go back to the menu.

Other requirements:

- Turning on the TalkBack should not interfere with usage of the application.
- Taking into account that this application can be used also by sand-blind people, there can be made a facilitation for them, by displaying white numbers on black background. With this contrast they will be easier to see.

3.3 Description of tools

Tools used in works of the thesis

The chosen design tool is Android Studio with Java as the chosen programming language. Android Studio have been chosen as the most advanced tool in the field of creating android applications at that time. And Java was chosen due to the popularity of this programming language, which ensures wither range of existing solutions of many problems.

Chapter 4

External specification

- hardware and software requirements
- installation procedure
- types of users
- user manual
- example of usage
- working scenarios (with screenshots or output files)

4.1 Hardware and software requirements

Device required to use the application

The application requires a device with any version of android. Because the whole program uses only basic android functionalitis, there is no point in setting any higher version, this way the application will be available on more devices.

4.2 Installation procedure

How to install

The installation is based on downloading the APK file to the device and running the installation procedure appropriate to this device.

4.3 Types of users

This application has only one universal mode for all types of users. The Sudoku will still have simple visual elements, so the game could be played relying only on the eyesight. The content is read to the user that lets play the game without looking at the screen. And the white on black background contrast could help the partially blind players. This way the application is made as much accessible as it is possible.

4.4 User manual

Application instructionl

All the basic instructions are shown and played every time after starting the application, and can be skipped with tapping the screen.

Editing the content of Sudoku number is enabled by a long tap, after long tapping the number it can be incremented by tapping it, after reaching the limit the number goes back to zero. Moving between cells to access more numbers is done by swipping a finger on the screen.

When the talkback is on, the swipe must be done with two fingers.

At the beginning user is given a simple instruction on how to play the game:

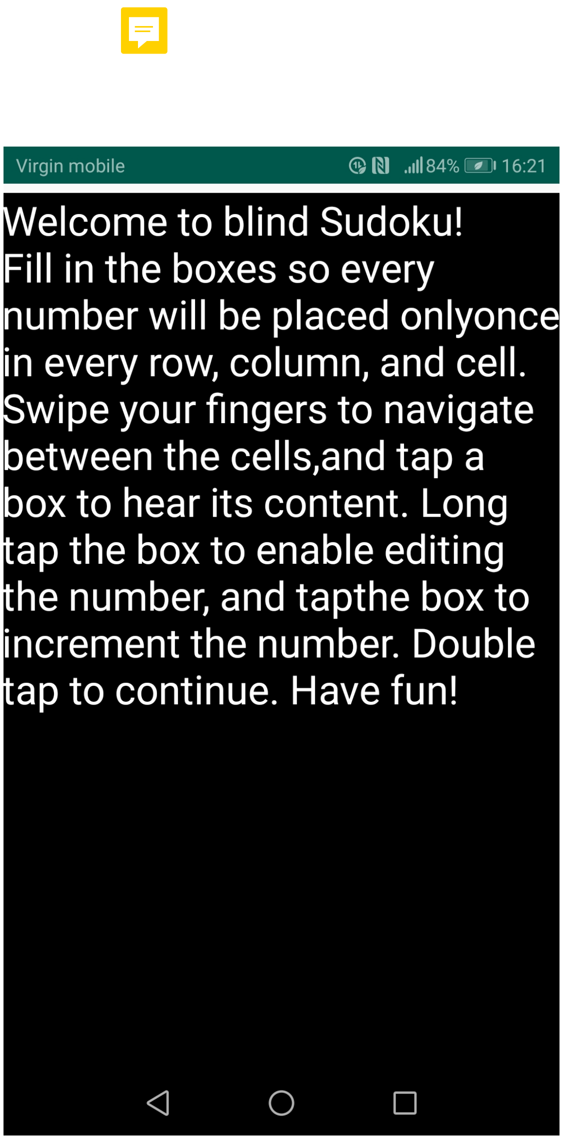
> Welcome to blind Sudoku! Fill in the boxes so every number will be placed only once in every row, column, and cell. Swipe your fingers to navigate between the cells, and tap a box to hear its content. Long tap the box to enable editing the number, and tap the box to increment the number. Double tap to continue. Have fun!

4.5 Working scenarios (with screenshots or output files)

This application uses two xml views. One for the instructions and one for the cell filled with numbers.

Figure 4.1 welcome message.

Figure 4.2 game view with Sudoku cell.



Welcome to blind Sudoku!
Fill in the boxes so every number will be placed onlyonce in every row, column, and cell. Swipe your fingers to navigate between the cells,and tap a box to hear its content. Long tap the box to enable editing the number, and tapthe box to increment the number. Double tap to continue. Have fun!

Figure 4.1: Instructions.

cell.jpg

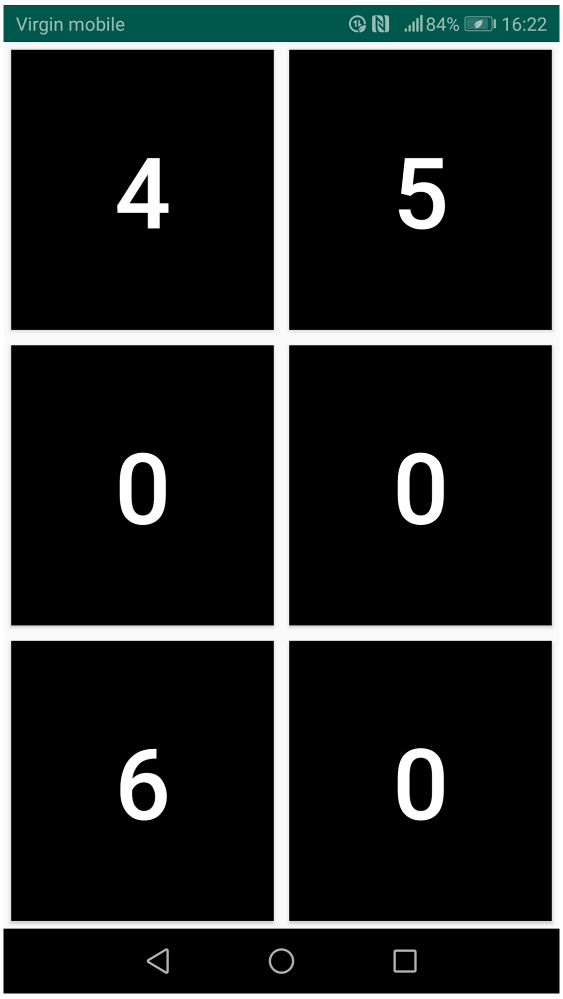


Figure 4.2: Instructions.

Chapter 5

Internal specification

- components, modules, libraries, resume of important classes (if used)
- resume of important algorithms (if used)
- details of implementation of selected parts
- applied design patterns
- UML diagrams

Use special environment for inline code, eg **descriptor** or **descriptor_gaussian**. Longer parts of code put in the figure environment, eg. code in Fig. **??**. Very long listings–move to an appendix.

5.1 Components, modules, libraries, resume of important classes

The whole program is based mostly on two classes. The sudoku logic in class Sudoku made of: generating Sudoku, validating the Sudoku, and all the other methods needed to work with the Sudoku. The MainActivity class responsible for using the Sudoku class and connecting it to the other elements of the game.

5.2 Resume of important algorithms

Generation of the 6 by 6 Sudoku was problematic, due to the fact that a square cannot be made of six square elements (what is required by most algorithms). This is why this program uses a simpler algorithm:

1. Generate numbers from one to six in random order, and put them into the first row.
2. Go to the next row.
3. Copy the content of the previous row and:
    (a) If the row number can be divided by 3, shift the numbers of this row by one place.
    (b) If the row number can not be divided by 3, shift the numbers of this row by two places.
4. If the Sudoku is not full yet, go back to step number two.
5. The Sudoku is ready.

Later on the generated data is copied from one array to another during the copping process the correct indexes of the arrays are compared with a ready made SmallTemplate object to create holes for the user to fill in. Letter on both the first and the second array can be compared to see it the user solved the second correctly.

The application behavior with Talkback should be similar to the one with Talkback turned off as much as possible. Firstly the reading of buttons labels by Talkback was turned off, so it would be done by the implementation in the program. It was done by setting buttons android:importantForAccessibility to "no" in the XML file. Secondly the reaction to tapping the buttons had to be overeaten, so the user would not have to tap twice more times and to override the previously mentioned reading of labels. It was done by using the setOnHoverListener on buttons, and implementing there the same behavior as in the onClick method, where we test if the talkback is on by using the AccessibilityManager.

5.3 Details of implementation of selected parts

The most important parts of implementation

The game is based mostly on only one xml view, that is one Sudoku cell, with numbers changing after moving between cells, and editing the content inside them. After every swipe the current cell number is adjusted and the function updating the view, with data from Sudoku, is called.

5.4 UML diagrams

Figure 5.1 UML of the project.

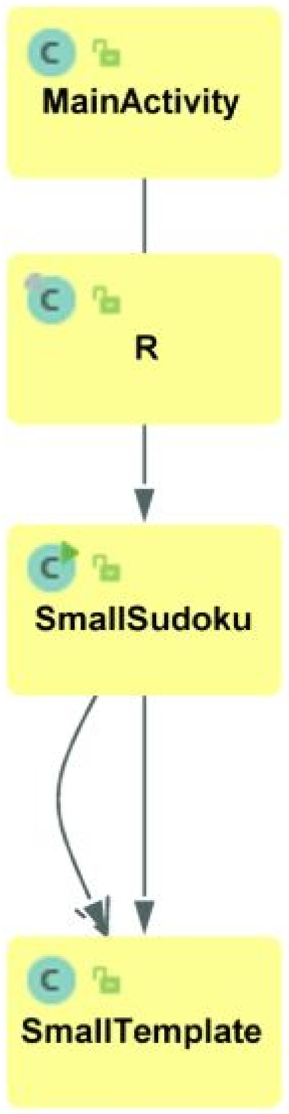


Figure 5.1: UML diagram.

Chapter 6

Verification and validation

- testing paradigm (eg V model)
- test cases, testing scope (full / partial)
- detected and fixed bugs
- results of experiments (optional)

6.1 Testing paradigm

State of the testing metodology

Due to the importance of Talkback compatibility, every feature had to be tested with and without enabled Talkback.

6.2 Test cases

All the most important cases taken into account during testing

Table 6.1: Action «User actions»

Action	**User actions**
Edit empty box	After long tapping the empty box the number editing is enabled, and every tap increments the number. After next long tap the content of box is saved.
Edit box with content generated by application	User is informed that the content cannot be changed.
Single tap a box	Application speaks the number that is inside the box.
Single tap a box when Talkback is on	Application speaks the number that is inside the box.
Swipe vertically and horisontally	The view moves between cells and stops and detects if it can move further.
Fill in the Sudoku correctly	Application informs that the Sudoku is correct.
Fill in the Sudoku incorrectly	Application informs that the Sudoku is incorrect.

6.3 Detected and fixed bugs

All the bugs that have been detected during testing and development

1. The upper limit for vertival cell movement was to high, that resulted in index out of bounds exception.

2. The application was displaying wrong numbers due to a mistake in translation of button number into x and y coordinates of sudoku.

3. Vertical swipe was not always working, because the event was not defined properly.

4. Application was not able to compile. The first usage of the text to speech was accidentally before its initialization.

Chapter 7

Conclusions

- achieved results with regard to objectives of the thesis and requirements
- path of further development (eg functional extension . . .)
- encountered difficulties and problems

7.1 Achieved results with regard to objectives of the thesis and requirements

Comparison between what was stated at the beginning as the goal and what was the final result

The final result of the project is fully functional Sudoku game that can be played without looking at the screen. It can work both with and without turned on Talkback, and the accessibility feature would not change. The only difference in playing the game with Talkback turned on is the way player enables the change of number in button. Instead of just holding the button for a longer period of time, it must be double tapped where the second tap is the one to be held longer.

7.2 Path of further development

Suggestions for what can be done more with the game

At this time the game can be played only on small (six on six) Sudoku with simple algorithm of generation, for better game play results it could be possible to chose size of Sudoku and level of difficulty. The application could be also extended by additional features not necessary for the game mechanics like saving time of the play into a ranking to compere results. Possibility to see the ranking should be made in the menu view, that would be also useful to implement together with going back to menu from the game view.

There are also other features that could make the game more accessible. Possibility of showing the whole Sudoku on the screen, by making the zoom out gesture, and zoom in to go back into the single cell. When the whole Sudoku is visible it is easier to look thru rows and columns, but navigating between 36 elements (or even 81 if the nine on nine version was chosen) could be challenging for a blind person, so both zoom in and zoom out views should be available. Similar result can be achieved by implementing hints for the user: Telling him if there is a certain number in a row or column. How many occurrences of a certain numbers missing in the whole Sudoku. It could also read all numbers in a row or column in the same order as they are placed in the Sudoku.

Leaving the subject of the game itself, there is also place for improvement in the displaying the text. The contrast of white text on black background can help people who still have partial sight, but for large variety of eyesight diseases, it can be possible that for some eyesight disease it would be easier to read with different contrast. This is why it should be possible to change the colors of contrast in the menu settings.

7.3 Encountered difficulties and problems

The most difficult part of implementing this application was making the application behavior, with and without Talkback, as similar as possible. There were many aspects that had to be taken into account, as many normal behaviors of the phone are overwritten when the Talkback is on.

Bibliography

[1] Name Surname and Name Surname. Title of an article in a journal. *Journal Title*, 157(8):1092–1113, 2016.

[2] Name Surname and Name Surname. *Title of a book.* Publisher, Hong Kong, 2017.

[3] Name Surname, Name Surname, and N. Surname. Title of a web page. `http://somewhere/in/internet.html`. [access date: 2018-09-30].

[4] Name Surname, Name Surname, and N. Surname. Title of a conference article. In *Conference title*, pages 5346–5349, 2006.

Appendices

List of abbreviations and symbols

DNA	deoxyribonucleic acid
MVC	model–view–controller
N	cardinality of data set
μ	membership function of a fuzzy set
$\mathbb{E}$	set of edges of a graph
$\mathcal{L}$	Laplace transformation

Listings

~~(Put long listings in the appendix.)~~

```
partition fcm_possibilistic::doPartition
                              (const dataset & ds)
{
   try
   {
      if (_nClusters < 1)
         throw std::string ("unknown␣number␣of␣clusters");
      if (_nIterations < 1 and _epsilon < 0)
         throw std::string ("You␣should␣set␣a␣maximal␣number␣of␣iteration␣or␣minimal␣difference␣--␣epsilon.");
      if (_nIterations > 0 and _epsilon > 0)
         throw std::string ("Both␣number␣of␣iterations␣and␣minimal␣epsilon␣set␣--␣you␣should␣set␣either␣number␣of␣iterations␣or␣minimal␣epsilon.");

      auto mX = ds.getMatrix();
      std::size_t nAttr = ds.getNumberOfAttributes();
      std::size_t nX    = ds.getNumberOfData();
      std::vector<std::vector<double>> mV;
```

```
    mU = std::vector<std::vector<double>> (_nClusters)
        ;
    for (auto & u : mU)
        u = std::vector<double> (nX);
    randomise(mU);
    normaliseByColumns(mU);
    calculateEtas(_nClusters, nX, ds);
    if (_nIterations > 0)
    {
        for (int iter = 0; iter < _nIterations; iter++)
        {
            mV = calculateClusterCentres(mU, mX);
            mU = modifyPartitionMatrix (mV, mX);
        }
    }
    else if (_epsilon > 0)
    {
        double frob;
        do
        {
            mV = calculateClusterCentres(mU, mX);
            auto mUnew = modifyPartitionMatrix (mV, mX);

            frob = Frobenius_norm_of_difference (mU,
                mUnew);
            mU = mUnew;
        } while (frob > _epsilon);
    }
    mV = calculateClusterCentres(mU, mX);
    std::vector<std::vector<double>> mS =
        calculateClusterFuzzification(mU, mV, mX);

    partition part;
```

```
        for (int c = 0; c < _nClusters; c++)
        {
            cluster cl;
            for (std::size_t a = 0; a < nAttr; a++)
            {
                descriptor_gaussian d (mV[c][a], mS[c][a]);
                cl.addDescriptor(d);
            }
            part.addCluster(cl);
        }
        return part;
    }
    catch (my_exception & ex)
    {
        throw my_exception (__FILE__, __FUNCTION__,
            __LINE__, ex.what());
    }
    catch (std::exception & ex)
    {
        throw my_exceptionn (__FILE__, __FUNCTION__,
            __LINE__, ex.what());
    }
    catch (std::string & ex)
    {
        throw my_exception (__FILE__, __FUNCTION__,
            __LINE__, ex);
    }
    catch (...)
    {
        throw my_exception (__FILE__, __FUNCTION__,
            __LINE__, "unknown␣expection");
    }
}
```

Contents of attached CD

The thesis is accompanied by a CD containing:

- thesis (LaTeX source files and final `pdf` file),
- source code of the application,
- test data.

List of Figures

List of Tables