TDT4240

Software Architecture

Spring 2012

Group A17
Android SDK

Steinar Haram
Emil Andreas Mork
Stian Sørebø
Ole Jørgen Rishoff

Implementation Document

Primary focus:
Maintainability

Secondary focus:
Usability

Contents

1 Introduction **3**

2 Design and implementation details **3**

2.1 Skiller multiplayer framework 3

2.2 Activities 4

2.3 MVC 4

2.4 States 6

3 User manual **6**

3.1 Functional requirements 6

3.2 Running the application 6

3.2.1 Emulator 6

3.2.2 On Android device 7

3.3 Game rules 7

3.4 Creating Skiller account 8

3.5 How to play 8

3.5.1 Choosing game mode 8

3.5.2 Placing, selecting, moving, and removing pieces 8

3.5.3 Hotseat mode 10

3.5.4 Online mode 10

4 Test report **10**

4.1 Functional requirements testing 10

4.2 Quality requirements testing 13

5 Relations to the architecture **13**

6 Issues **13**

6.1 Gained experieces 14

References **15**

List of Tables

1 Testing of FR1 10
2 Testing of FR2 10
3 Testing of FR3 11
4 Testing of FR4 11
5 Testing of FR5 11
6 Testing of FR6 12
7 Testing of FR7 12
8 Testing of FR8 12
9 Testing of FR9 13
10 Testing of M1 13

List of Figures

1 Application activities 4
2 MVC structure 5
3 State structure 6
4 Available game modes 8

1 Introduction

This document contains implementation details for our developed version of the classical *Nine Men's Morris* game. The game is developed as a native Android application. The applications primary attribute is modifiability, while its secondary attribute is usability. The second chapter will highlight the design and implementation details. The following chapter contains a user manual, while chapter four contains a brief description of the testing of functional and quality requirements. The relation between the implementation and the planned architecture will be reviewed in chapter five. Chapter six highlights encountered problems and gained experience.

2 Design and implementation details

Here you describe a more detailed view of the various parts of the architecture describing how the robot controller or game was designed.

2.1 Skiller multiplayer framework

Due to the desire of developing a fully functional multiplayer game, the *Skiller multiplayer framework* [**?**] has been used. This is a third party COTS software, and its usage has sped up the development process. Registration was needed in order to gain access to the Skiller SDK. When the registration was done, a new game could be created, and an application ID, an application key, and an application secret was supplied. These are used in the code to identify the specific application.

This framework supplies a server solution for turn based games, and it has been implemented in the network class. When playing a network game, the GameController class tells the Game model to network class sends event messages to the server, and the server delivers it to the opponent.

2.2 Activities

Figure 1 shows an overview of the application's different activities, and how the user interactions can change them. The user can navigates back to the menuActivity by pressing the back button.

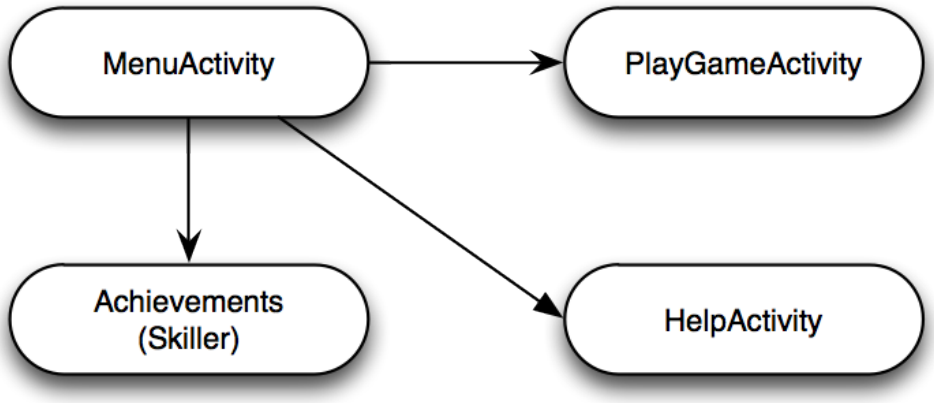


Figure 1: Application activities

The MenuActivity shows a menu consisting of five items, allowing the user to create or join a multiplayer game, start a local game, check achievements, or check the game rules. The PlayGameActivity is responsible for creating or joining multiplayer games, and starting local games. The HelpActivity is responsible for showing the game rules. The achievement screen is supplied by *Skiller*.

Screencaps of the different activities, and the achievement screen, are shown in section 3.5.

2.3 MVC

Our implementation of the MVC structure is based interface communication. In regular Java it is more common to create a MVC structure where the view has model and the controller contains view and model. Android has its own

MVC structure where an activity represent controller and view, which makes it difficult to build ut the MVC in a regular way. The most favorable way of createing MVC in android is to use interfaces to update the view when there are changes in the model. Figure 2 shows an overview of the MVC structure, with BoardView (View), GameController (Controller) and Game (Model). When the user interact with the view the controller handles the user action and tells the model what to do. When the model is updated, the view receive a message via the interface, and updates the screen.

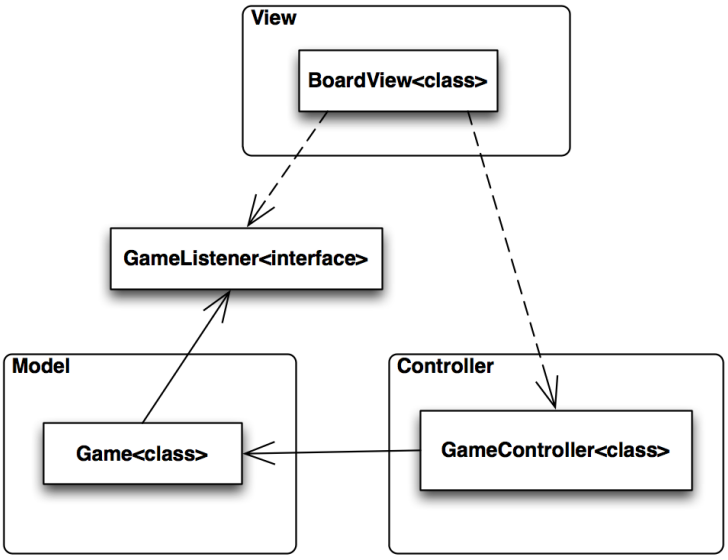


Figure 2: MVC structure

2.4 States

The Game class is implemented with associated states. The states change on runtime depending on player interaction. All states have their own way of controlling user possibilities, and telling the user what to do while playing.

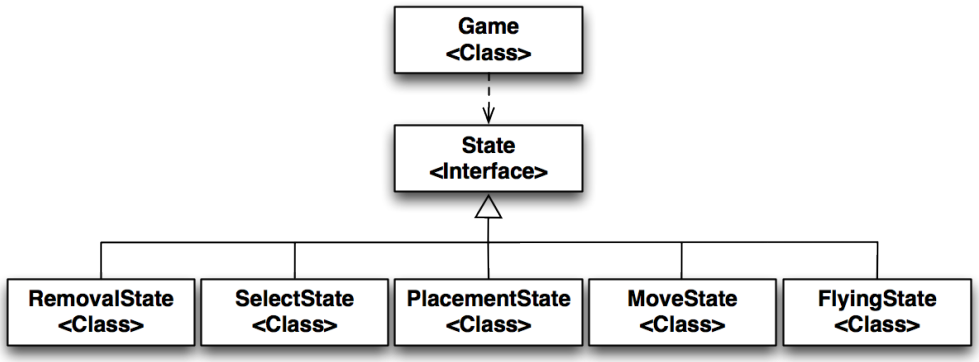


Figure 3: State structure

3 User manual

How to run the robot, how to compile it, run it, etc.

3.1 Functional requirements

- Android OS v2.2

3.2 Running the application

The application is avalible at URL.
The Eclipse-project is avalible at ANOTHER URL.

3.2.1 Emulator

To run the application in the emulator, the user needs to open the project in Eclipse. File -> Open project -> Existing source code -> path to downloaded project

3.2.2 On Android device

The user has to connect the Android device and load the application file.
DETAILED (STEP BY STEP) DESCRIPTION NEEDED!

3.3 Game rules

The game is implemented with the same set of rules as the classic board game *Nine Men's Morris* [?]. The goal of the game is to either block any opponent moves, or to reduce your opponent's piece number to less than three. If you get three pieces in a row, you enter a morris state, and are allowed to remove one of your opponent's pieces. Pieces that are in a morris state, i.e. forms three in a row either horizontally or vertically, are not removable.

3.4 Creating Skiller account

3.5 How to play

3.5.1 Choosing game mode

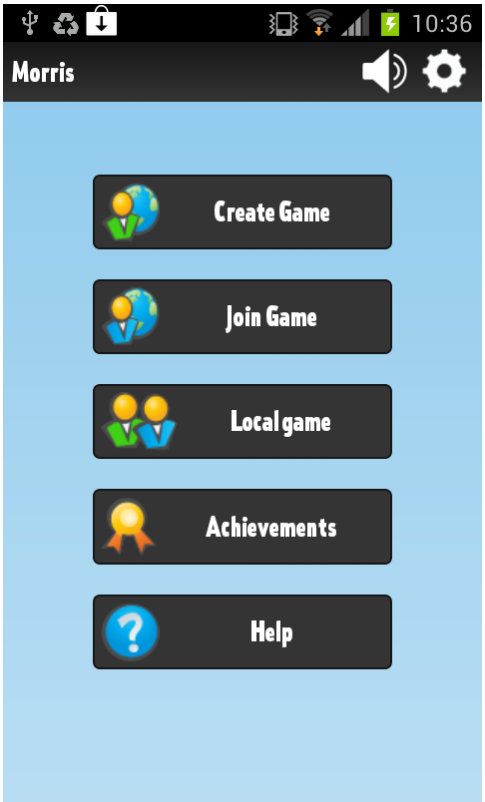
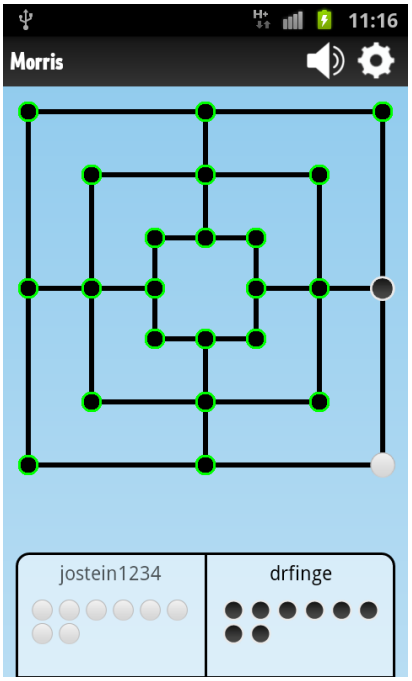


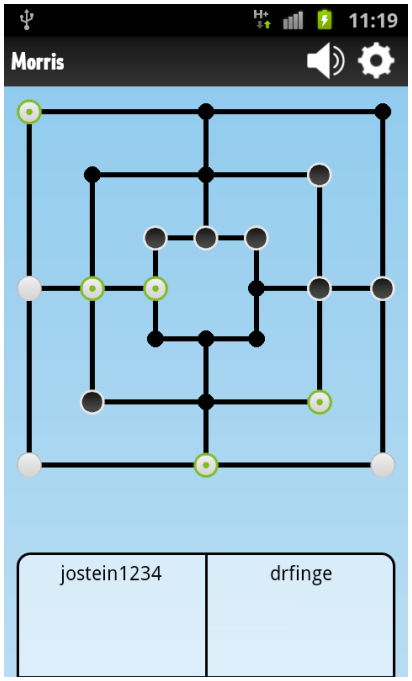
Figure 4: Available game modes

3.5.2 Placing, selecting, moving, and removing pieces

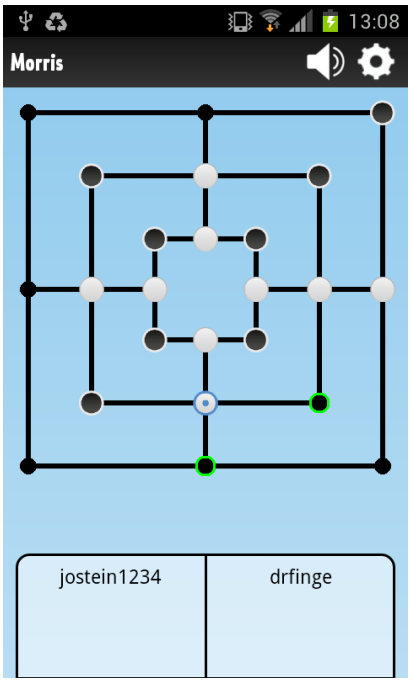
When it is your turn to move, either the board or your pieces will be highlighted. In addition, the name of the current player will be blinking as the game progresses.



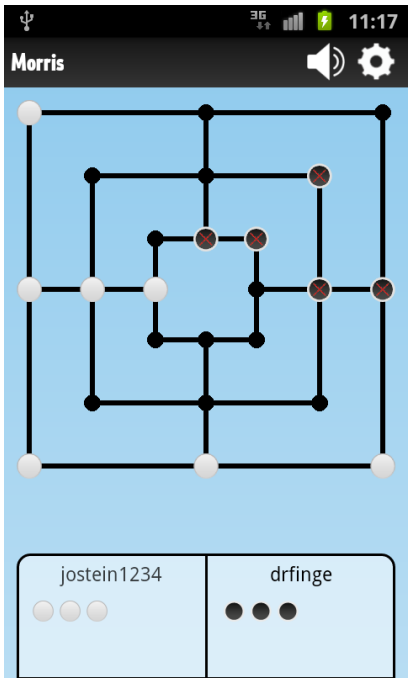
(a) Green indicator shows where you can place a piece



(b) Highlights selectable pieces



(c) Highlights selected piece, green indicator on possible moves



(d) Highlights removable pieces with a red cross

3.5.3 Hotseat mode

If you start a local game as described in section 3.5.1, you can control both players from the same device.

3.5.4 Online mode

If you start an online game as described in section 3.5.1, you are taken to the board screen, and need to wait for another player to join your game. The guest, i.e. the one who joins the game, will get the initial move. Your own pieces will always be white.

4 Test report

The report should contain test reports for both functional requirements and quality requirements (quality scenarios).

4.1 Functional requirements testing

FR1 - Placement of pieces	
Executor	Ole Jørgen Rishoff
Date	12.04.2012
Time used	5 minutes
Evaluation	The players successfully placed all nine pieces.

Table 1: Testing of FR1

FR2 - Moving pieces	
Executor	Ole Jørgen Rishoff
Date	12.04.2012
Time used	3 minutes
Evaluation	The players successfully moved their pieces one length at the time.

Table 2: Testing of FR2

FR3 - Morris state	
Executor	Ole Jørgen Rishoff
Date	12.04.2012
Time used	3 minutes
Evaluation	When placing three pieces in a row, the game successfully changed state, and a piece was removed from the opponent.

Table 3: Testing of FR3

FR4 - Flying pieces	
Executor	Ole Jørgen Rishoff
Date	12.04.2012
Time used	3 minutes
Evaluation	When the player had three pieces left, the game successfully changed state to Flying state, and the player was allowed to move to any vacant field.

Table 4: Testing of FR4

FR5 - Multiplayer	
Executor	Ole Jørgen Rishoff
Date	12.04.2012
Time used	10 minutes
Evaluation	Ole and Emil connected to each other via the Skiller framework, and successfully played a whole game.

Table 5: Testing of FR5

FR6 - Game board	
Executor	Ole Jørgen Rishoff
Date	12.04.2012
Time used	1 minute
Evaluation	The game has a board conforming with the layout of *Nine Men's Morris*.

Table 6: Testing of FR6

FR7 - Setting player name	
Executor	Ole Jørgen Rishoff
Date	12.04.2012
Time used	5 minutes
Evaluation	A player can set his own name when creating a Skiller account.

Table 7: Testing of FR7

FR8 - Denied Morris state	
Executor	Ole Jørgen Rishoff
Date	12.04.2012
Time used	10 minutes
Evaluation	The game automatically ends a players turn when he enters a Morris state, and all the opponents pieces also is in a Morris state

Table 8: Testing of FR8

FR9 - Game over	
Executor	Ole Jørgen Rishoff
Date	12.04.2012
Time used	5 minutes
Evaluation	When a player has only two pieces left, or cannot move any of his or her pieces, the game successfully ends.

Table 9: Testing of FR9

4.2 Quality requirements testing

FR11 - Game over	
Executor	Ole Jørgen Rishoff
Date	23.04.2012
Stimuli	Addition of a new game variant
Expected response	The architecture should allow an easy extension to *Twelve Men's Morris*.
Observed response	The system is flexible and an extension can easily be added.
Evaluation	Successful

Table 10: Testing of M1

5 Relations to the architecture

This section should list the inconsistencies between your architecture and the implementation. Give the reasons for these inconsistencies. Discuss whether they could have been discovered at an earlier point, for instance during the ATAM evaluation.

6 Issues

It turned out that the Skiller framework was poorly documented and it gave some unreadable exception messages. This slowed down the testing quite a

bit. In addition, because we normally have only had two Android devices at our disposal, much of the testing have been done in the emulator. This is of course not very effective. Also, when running the application in the emulator and creating games, players would automatically join without user interaction. The Skiller team was unable to give us any good answers to why we experienced this problem.

We underestimated the importance of groupwide understanding of the framework, and the implementation should have been done collectively. As it was, we assigned one person to this task, and the rest of the group were unable to do anything with the framework in his absence. We also should have a done a more thourogh research regarding the use of the framework and its documentation.

6.1 Gained experieces

In future projects we will spend more time researching possibilities when choosing to work with a third party framework. We did not look into the documentation of the framework before implementing it, and if we had, we would probably have chosen a different framework. We will also perform a quick search for problems related to it before making a final decision.

Some members of the group went into this project with more experience than others, developing native Android applications. Due to good communication we have been able to use this experience to our advantage.

We have gotten a better understanding of the patterns we chose to implement. The experience of implementing a MVC pattern in a native Android application has been very useful. The state pattern and its usage was unknown for all of us, but we now have a good understanding of how it can be used. We could perhaps made this pattern a bit more clear, and assign more responsibilities to the different states.

References