Requirements and Analysis Document for Chalmers Chance

William Andersson, Vilhelm Hedquist, Felix Holmesten,
Måns Josefsson and Oscar Johansson

02/10/2020 - Version 1

1 Introduction

Chalmers Chance puts a new spin on the classic board game Risk. This time at the campus of Chalmers University of Technology.

The game is an online version of Risk and follows the same rulebook. The target audience is students at Chalmers and the game allows them to fulfill fantasies of conquering Chalmers for their student division of choice, but even if you are not a student at Chalmers you can learn about the campus or at the very least play a game of Risk.

1.1 Definitions, acronyms, and abbreviations

Risk:
A classic board game, which revolves around conquering the world and battle against opponents for each other's territories using dice. (For more information: https://en.wikipedia.org/wiki/Risk_(game))

Chalmers Chance:
The name of the game.

Student division:
A student division in the context of the game is a local organization of Chalmers student union, where each of the 16 student divisions are connected to a unique study programme.

2 Requirements

2.1 User Stories

The following User Stories are used in the creation of the application:

```
Story Identifier: 001
Story Name: Set Up Game
```

As a User, I want to set up the game, so that I can configure the game and get started.

Acceptance:
- The application can read input from users.
- The user can choose the amount of players to play the game.
- The user can choose what student division each player represents.
- The game can only start when everything is set up correctly.
- Configuration is not available after the game is started.

-

Story Identifier: 002
Story Name: Creation of Gameboard

As a Player, I want to get a game board, so that I can start playing and having fun.

Acceptance:
- A map over Chalmers is displayed.
- Spaces that players can display are displayed on the map.
- At the start of the game each player controls the same amount of spaces. (as far as possible)
- At the start of the game each player controls the same amount of units.

-

Story Identifier: 003
Story Name: Take a turn

As a Player, I want to take a turn, so that I can try to win.

Acceptance:
- A player can only play during their turn.
- There is a visual cue that shows whose turn it is.
- The player can end their turn.
- A turn consists of three phases; Deployment, Attack, Movement.
- The player can navigate through the phases.
- The game should indicate which phase is active.

Story Identifier: 004

Story Name: Deployment Phase

As a Player, I want to deploy units, so that I can become more powerful.

Acceptance:
- The player gets a set amount of units to deploy at the start of the Deployment phase, based on the spaces they hold.
- The player can deploy units on spaces they control.
- Units can’t be saved to deploy during later stages of the game

---
-

Story Identifier: 005
Story Name: Attack Phase

As a Player, I want to attack other players' spaces, so that I can take over the spaces.

Acceptance:
- The player can choose a space that they control.
- The player can then choose a neighbouring space that they don’t control.
- The player can initiate the attack.
- The game calculates the casualties of the attack and updates the spaces accordingly.

-

Story Identifier: 006
Story Name: Movement Phase

As a Player, I want to move my units, so that I can position them strategically on the gameboard.

Acceptance:
- The player can choose a space that they control.
- The player can then choose a neighbouring space that they control.
- The player can choose the amount of units to move from the first space to the other.

-

```
Story Identifier: 007
Story Name: The End of the Game

As a Player, I want the game to end, so that I can win (or lose)

Acceptance:
   -  The game ends when one player controls all spaces.
   -  The winner is displayed to the players.

-----------------------------------------------------------------
-

Story Identifier: 008
Story Name: Online Gameplay

As a User, I want to play online, so that I can play against remote
friends.

Acceptance:
   -  The game can be played online
   -  TBD
   -  Not implemented yet

-----------------------------------------------------------------
-
```

2.2 Definition of Done

- Each class should have a general JavaDoc-description of its purpose.
- All public or package private methods should be well documented with JavaDoc, as well as tested with at least one JUnit-test.
- The application should always be runnable.
- Every acceptance-criteria described in the User Stories should be met.

2.3 User interface

Users are welcomed by a starting page with limited navigation options. The user can start the game or exit right away.

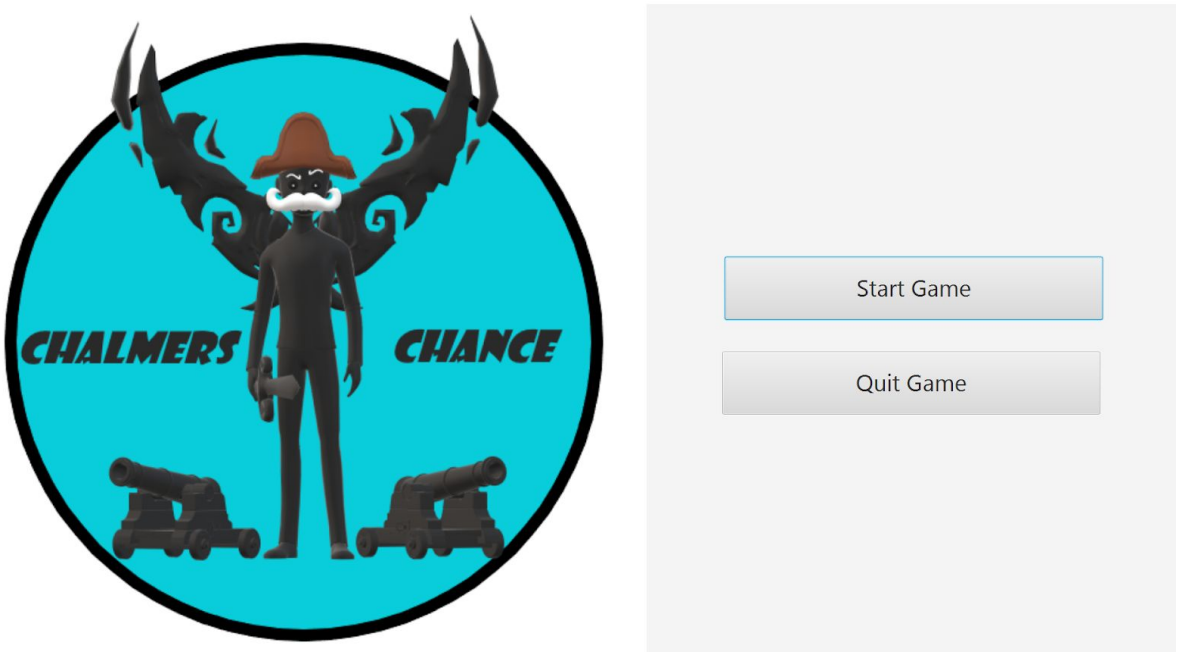

Figure 1: The starting-page of the application.

Starting the game takes the user to the player selection, where the player can choose the amount of players and which student division they want to represent in-game. The user can thereafter opt to start the game and is taken to the game board.

Figure 2: The player selection view.

The active player is displayed at the top left corner and the phases can be navigated through using the buttons.

Figure 3: The gameboard

The game can be paused by using the escape-key on the keyboard.

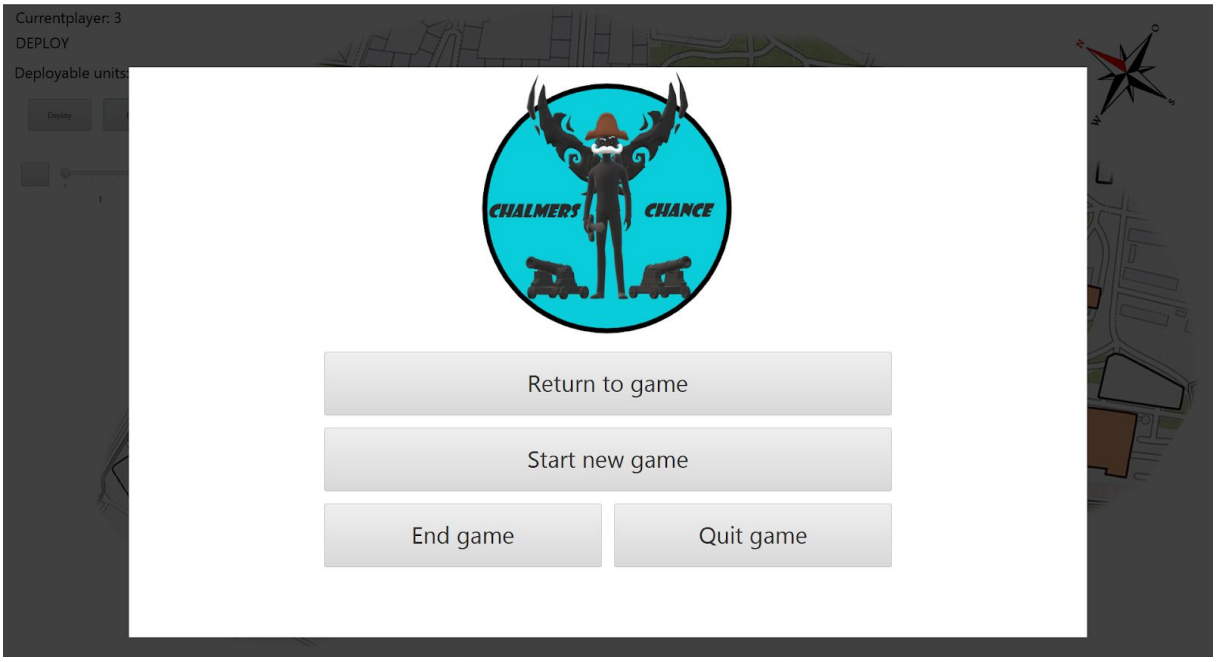

Figure 4: The pause screen.

From the pause screen the player can choose to resume the game, start a new game, quit the game completely or end the game and reach the end game screen. This is currently the only way to reach the end game screen at the moment, as a winner is yet not implemented.

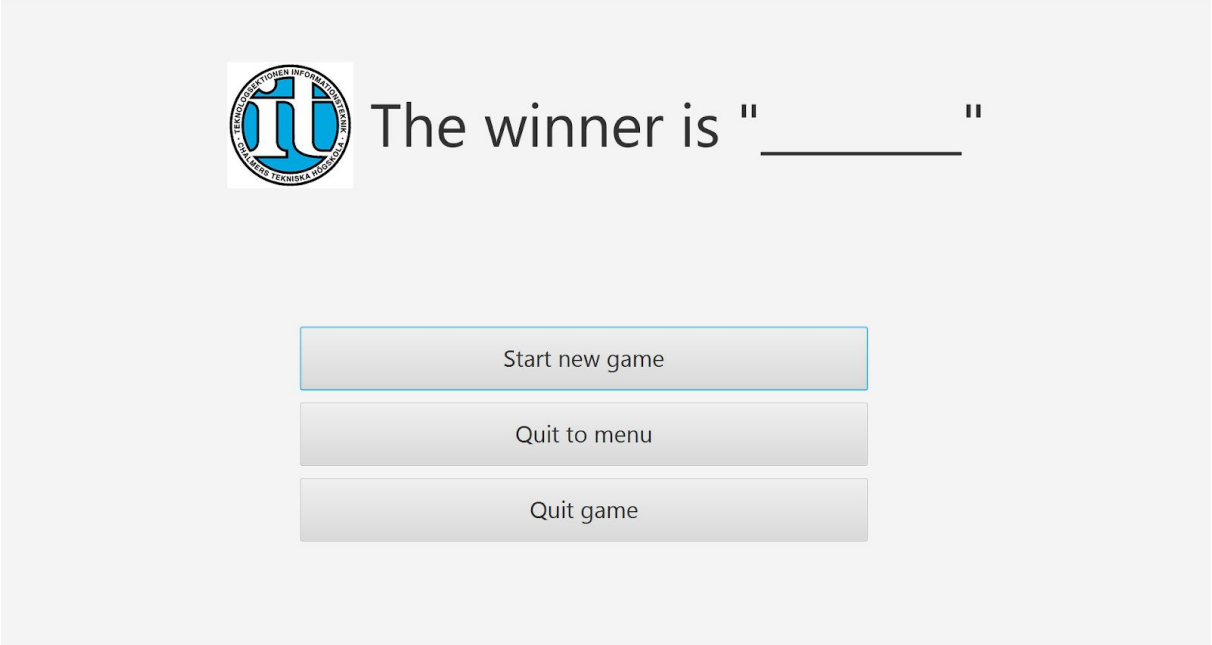

Figure 5: The end game screen.

The end game screen has similar options as the pause screen. Starting a new game will take the user to the start page again.

3 Domain model

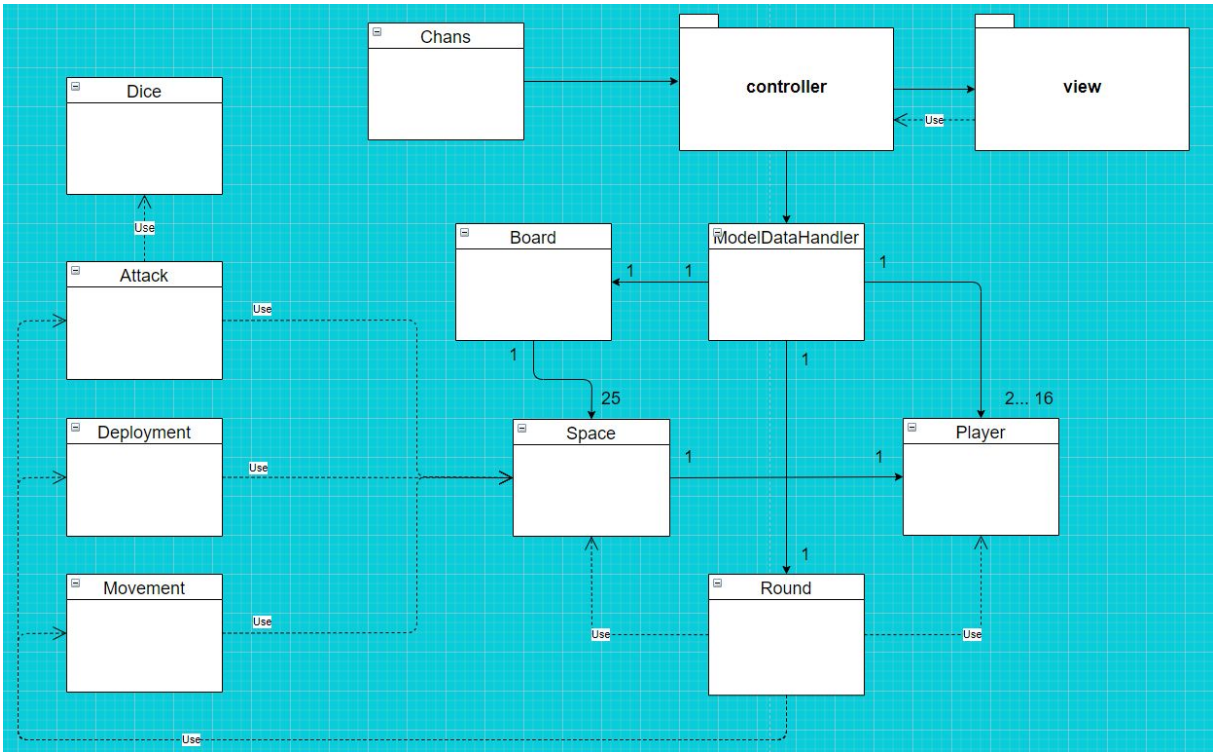

Figure 6: Domain model with dependencies between classes. Technical packages are also displayed on a very high level.

3.1 Class responsibilities

Below follows explanations of the classes seen in the domain model, see Figure 4.

Chans - This class initiates the application and sets the stage for the front page.

ModelDataHandler - This class is the main gateway from the model to other parts of the code and is responsible for initiating the game and setting up a correct game board.
Board - This class represents the board and is keeping track of the board.
Space - This class holds methods that´s controlling the spaces. Mostly creating them and updating them.
Player - This class creates a player.
Round - This class is controlling the different phases for one round.
Deployment - This class makes it possible for the current player to deploy units in the marked space IF the marked space is owned by the current player and that player has any units left to deploy.
Movement - This class makes it possible for a player to move units from one space to another.
Attack - This class makes it possible for the current player to attack other players' spaces. It then calculates the results of the attack and if all units on the other player's space dies then it moves all units except one.
Dice - This class makes it possible to calculate an attack depending on the value of the dices.

4 References

JavaFX - Library used to create the GUI of the application. (See: https://openjfx.io/)

Scene Builder - Integrated with JavaFx and used to design the GUI with simple Drag and Drop features. (See: https://gluonhq.com/products/scene-builder/)

IntelliJ IDEA - IDE used for programming. (See: https://gluonhq.com/products/scene-builder/)