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. Where student divisions engage in non-lethal battle against each other.

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.

Space
A button on the map which gets its name from the location it is placed at on the map. The owner's color and amount of units is displayed on the space.

Area

A collection of **spaces**. If a player owns all of the **spaces** of an area it will receive more units to deploy on their next turn.

2 Requirements

2.1 User Stories

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

```
Story Identifier: CC001
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: CC002
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 control 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. (as far as possible)

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

Story Identifier: CC003
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: CC004
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: CC005
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: CC006
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: CC007
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: CC008
Story Name: Online Gameplay

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

Acceptance:
   - You can connect to a game and play with friends without being
     at the same place.

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

2.2 Definition of Done

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

2.3 User interface

Views used in both singleplayer and multiplayer

The Start Menu

Users are welcomed by a starting page with navigation options. The user can choose to start a local game, to start an online game or to quit the game.

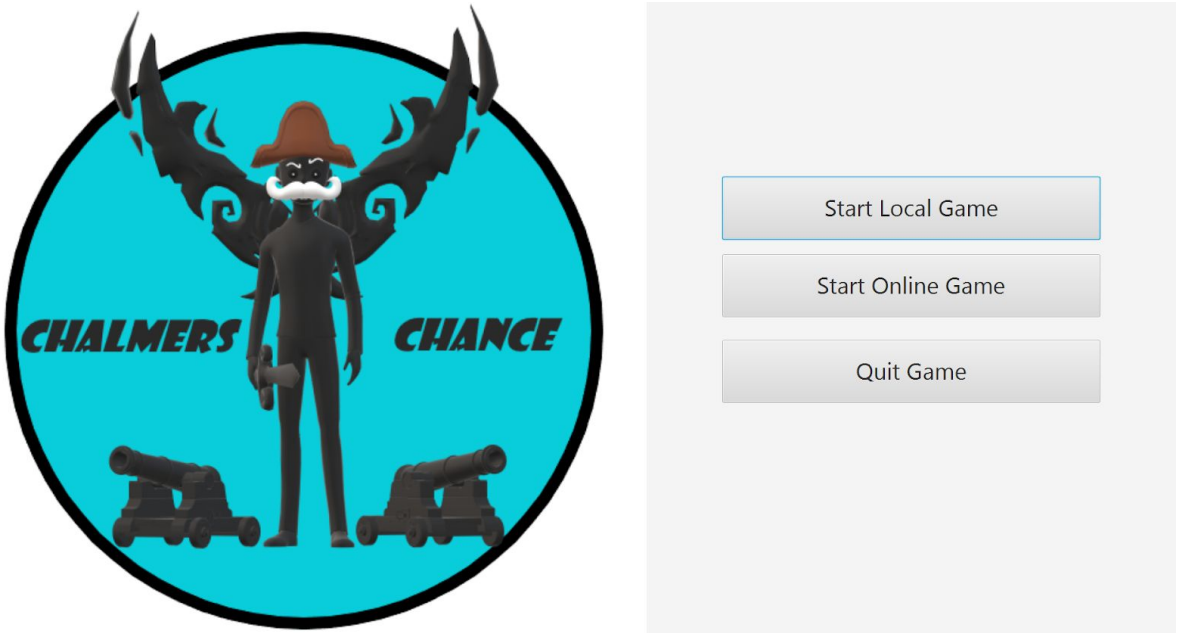

Figure 1: The starting-page of the application.

The Chalmers Board

The active player is displayed in the bottom right corner and the phases can be navigated through using the buttons. Areas och spaces are named and colorcoded. The same board is used for both multiplayer and singleplayer with a few additions in the multiplayer-mode. While in that mode you can see what the other players are selecting and it will display the players' usernames.

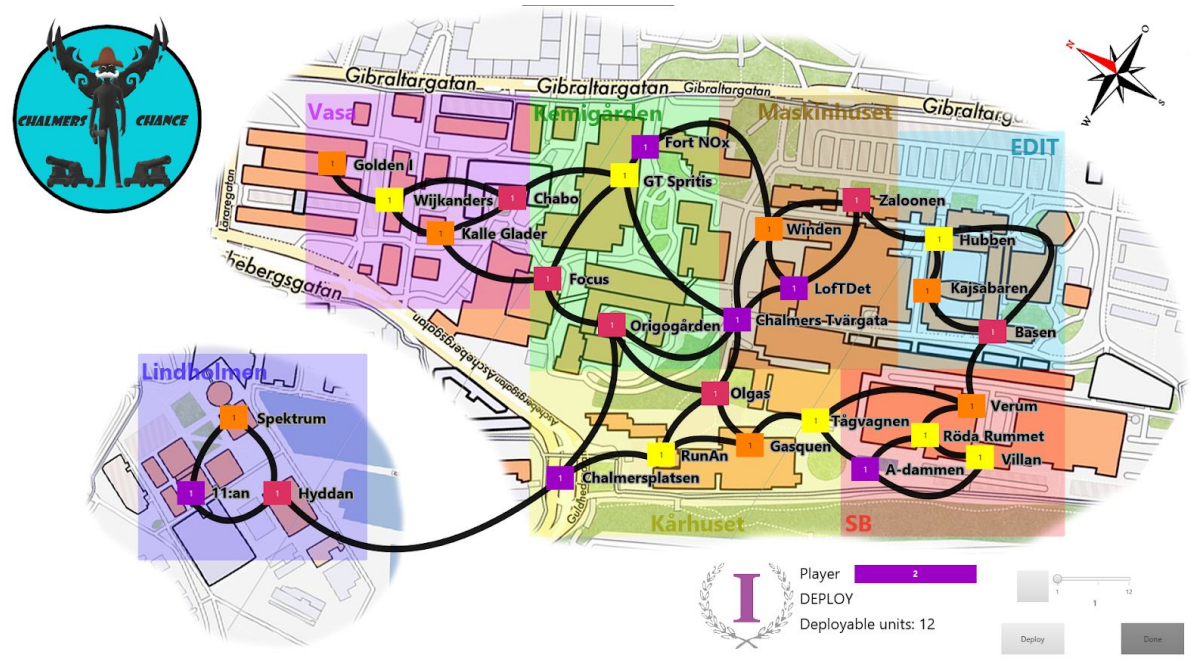

Figure 2: Board in a single player session

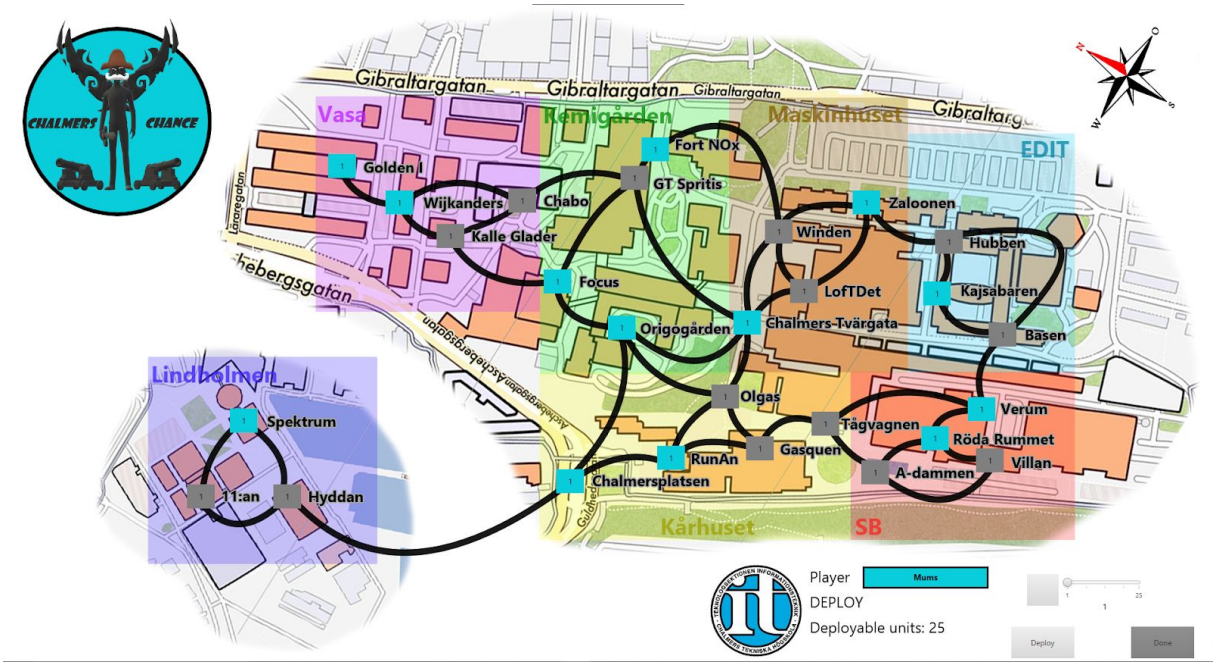

Figure 3: Board in a multiplayer session

The Pause Menu

The game can be paused by using the escape-key on the keyboard. The player can choose to resume the game, go to the start menu, or quit the game.

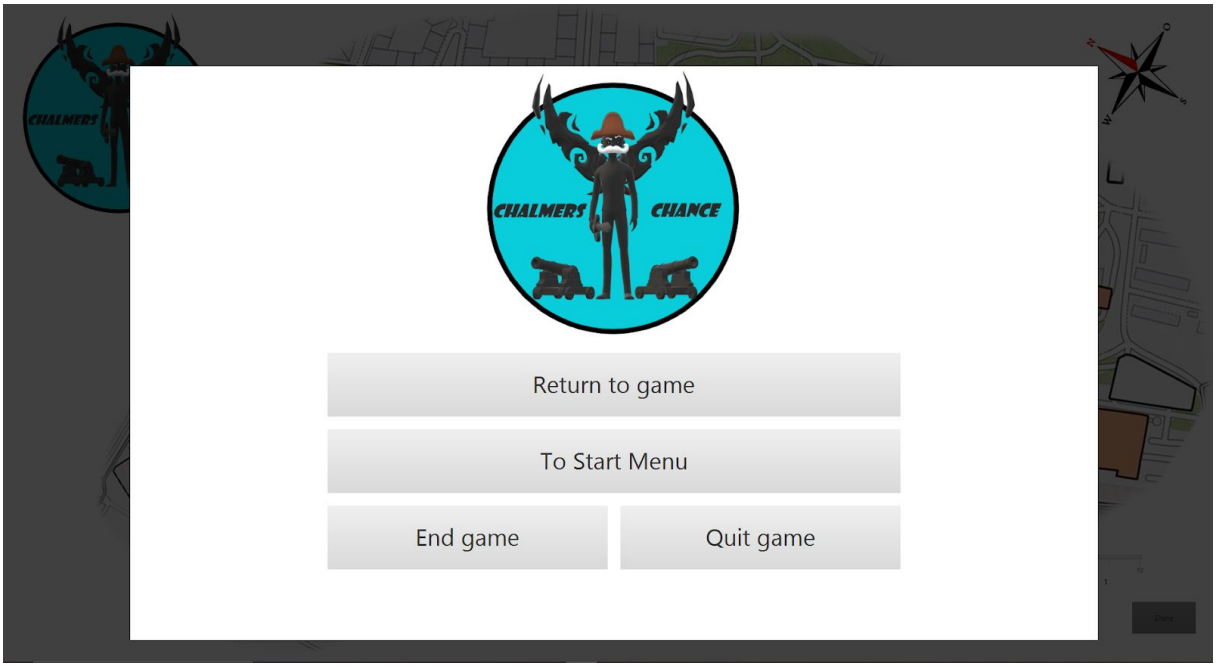

Figure 3: The pause screen.

The End Menu

The end menu will be reached when a winner is decided. From here you can leave the game or go back to the start menu.

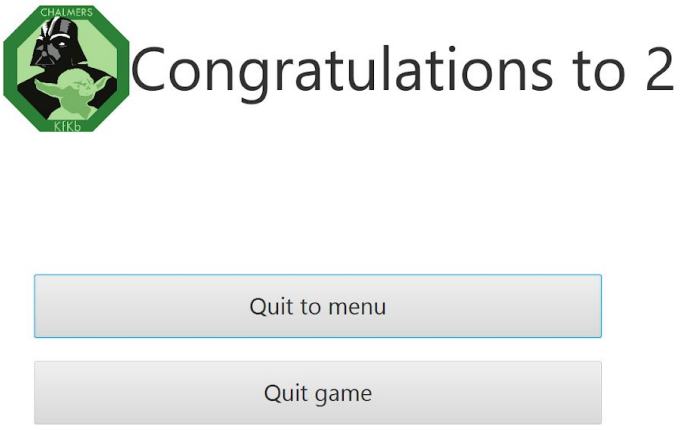

Figure 4: The end game menu

Singleplayer

The Setup Menu (singleplayer)

Starting a local game takes the user to the setup menu, 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 5: The player selection view for single player.

Multiplayer

The Lobby Select Menu

This menu will be displayed upon clicking “start online game”. Here the player can choose which lobby to join.

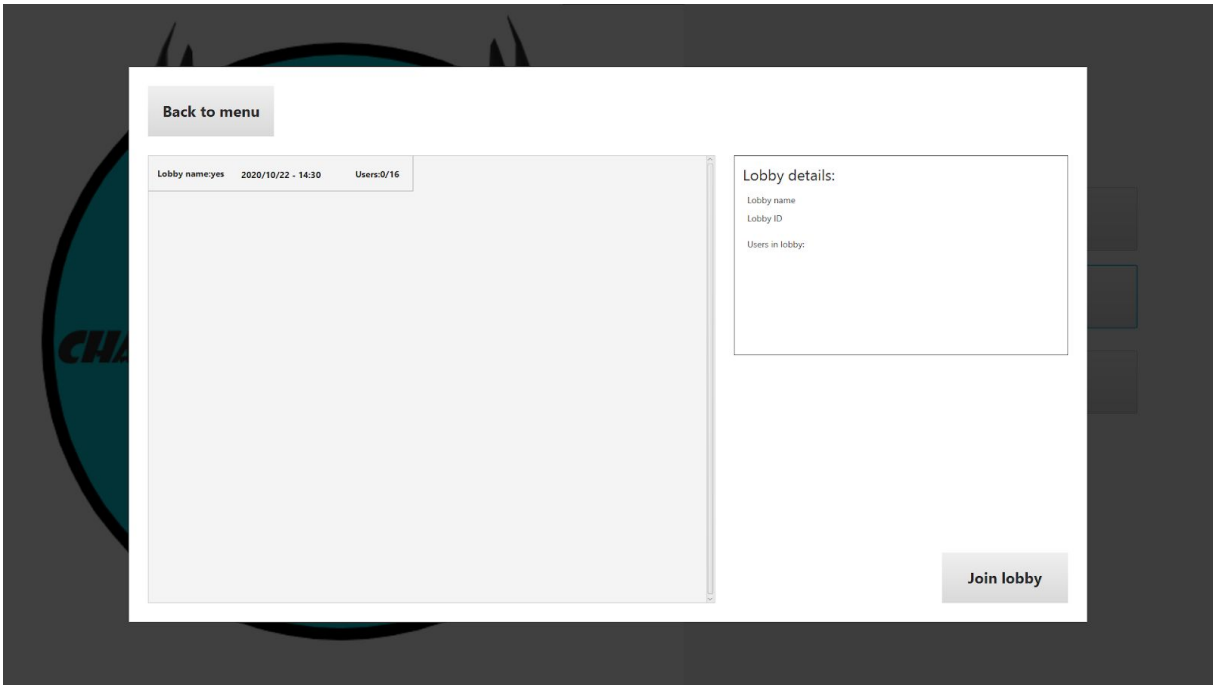

Figure 7: The lobby select screen with a list of lobbies

The Setup Menu (multiplayer)
Setup in multiplayer is a little bit different from setup in single player. Here the player gets to choose a name and student division for themselves but of course not the others. Student divisions which already have been chosen are greyed out.

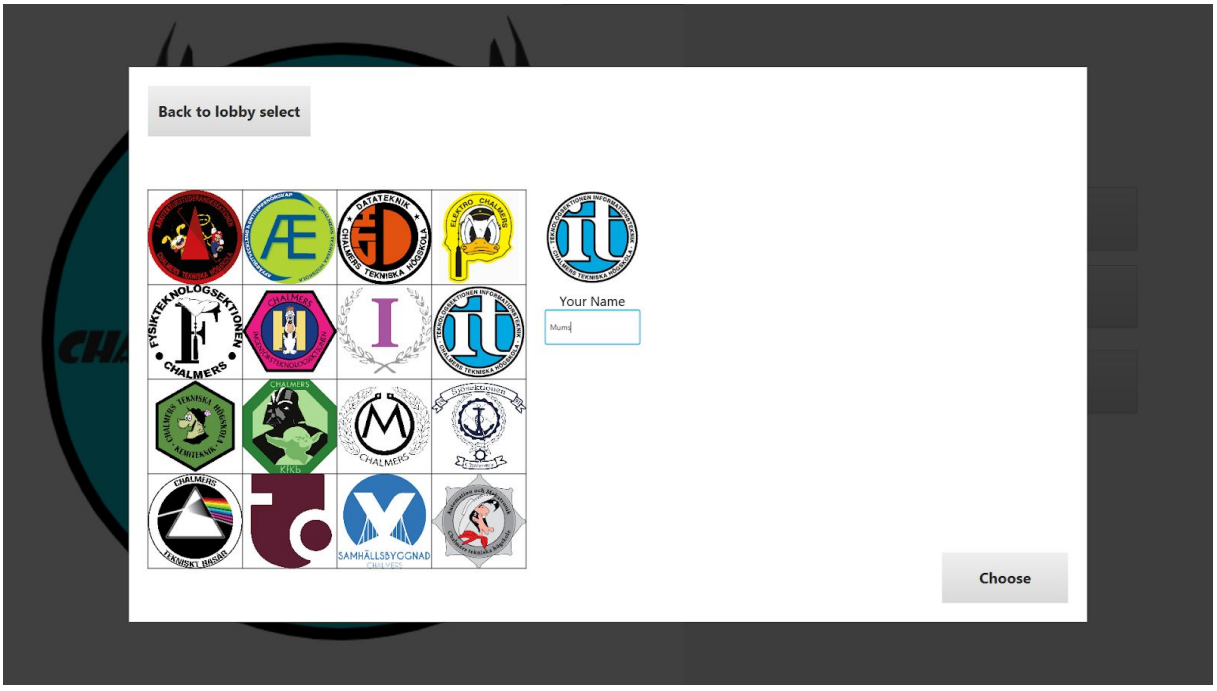

Figure 7: The multiplayer setup menu

The Lobby Ready Menu
After choosing a student division you get sent to the Lobby Ready Menu where you wait for the other players to join before entering the game. You ready up by clicking the button in the bottom-right corner and once everyone are ready the lobby host can decide to start the game.

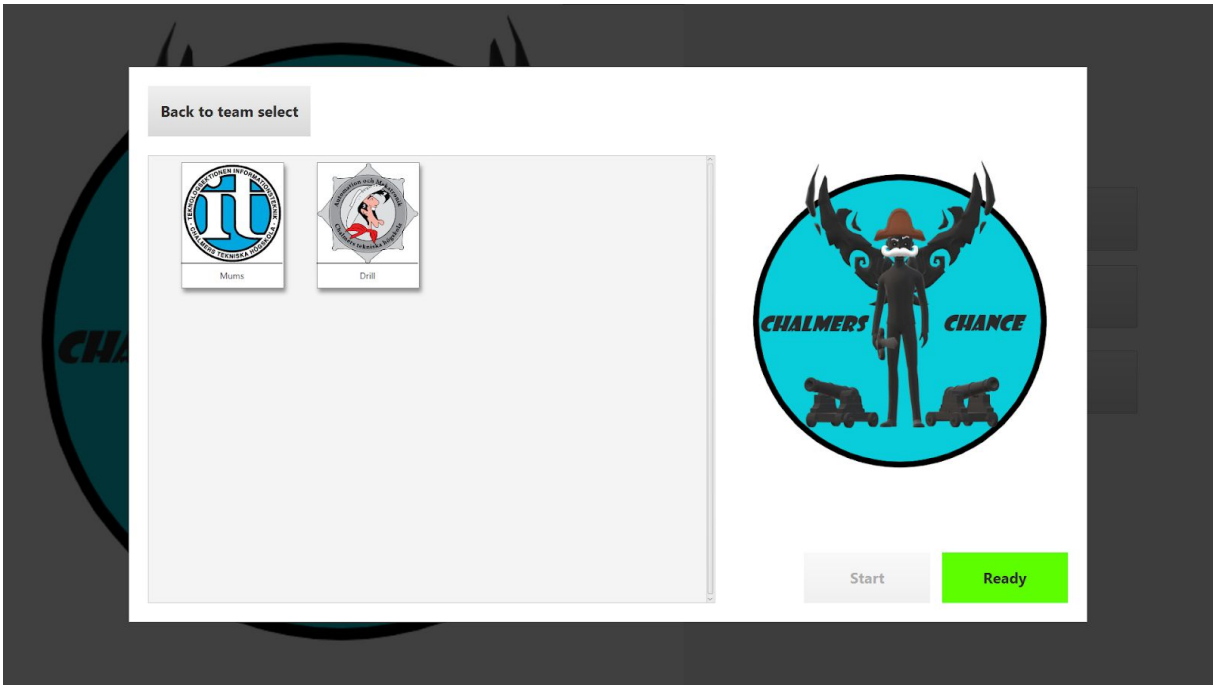

Figure 8: The lobby menu from which you launch a multiplayer game.

3 Domain model

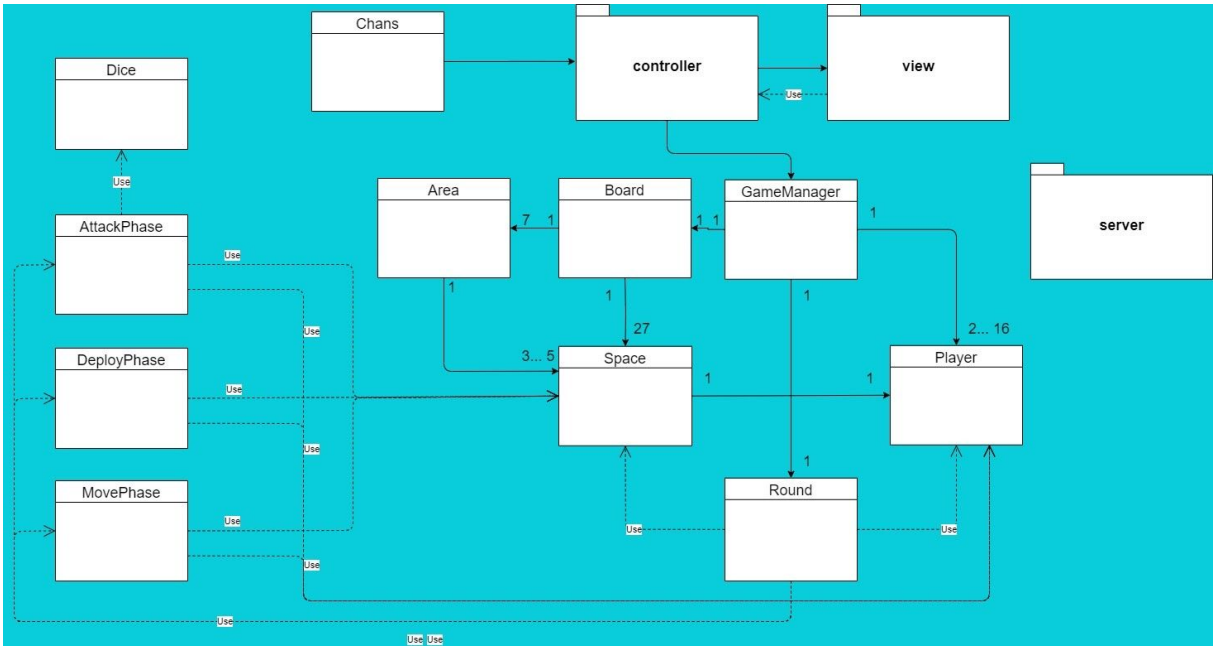

Figure 6: Domain model with dependencies between classes. Technical packages are also displayed on a very high level.

3.1 Class responsibilities

These are short explanations of the classes from the domain model.

Model

- **Chans** - This class initiates the application and sets the stage for the front page.
- **GameManager** - 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.
- **DeployPhase** - 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.
- **MovePhase** - This class makes it possible for a player to move units from one space to another.
- **AttackPhase** - 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.
- **Area** - This class makes it possible to create areas in the game.
- **BoardManager** - This class represents the board and is keeping track of the board and special conditions.
- **ChalmersBoard** - This class is a concrete implementation of the board that is being used in the program. It is a representation of the Campus on Chalmers.
- **ModelToViewFacade** - This is a class that limits the direct access from the view package to the model package. Lowering coupling between packages as well as enabling safer modification of the model.

View

- **AttackView** - This is the view for AttackController. It holds methods that update the AttackView when the AttackController is calling them.
- **MapView** - This is the view for the MapController. It holds methods that update the MapView when the MapController is calling them.

Controller

- **AttackController** - This is the controller for AttackView. It holds all the logic for the buttons in that view and is calling methods from AttackView.
- **EndController** - This is the controller for the endMenu.fxml
- **LobbyItem** - This is the controller for lobbyItem.fxml
- **LobbyItemCreator** - This class is creating the lobbyItems in the lobby.
- **LobbyReadyController** - This is the controller for lobbyReady.fxml
- **LobbySelectController** - This is the controller for lobbySelect.fxml

- **MapController** - This is the controller for MapView. It holds all the logic for the buttons in that view and is calling methods from MapView.
- **MultiplayerLogoController** - This is the controller for setUpOnline.fxml
- **PauseController** - This is the controller for pauseMenu.fxml
- **PlayerCard** - This is the controller for userCard.fxml
- **SetUpGameController** - This is the controller for setUpGame.fxml
- **StartController** - This is the controller for the startMenu.fxml

Client

- **Client** - This class is a singleton. This class is responsible for the connection with a server.

Server

- **LobbyController** - This class handles lists of lobbies
- **GameLobby** - This class creates a lobby that is being used while playing the game
- **Lobby** - A lobby consists of a list of players, the lobby's name and it's id
- **MenuLobby** - This class creates a lobby that is being used while waiting in the menu
- **ServerManager** - This is the main server class

Interfaces

- **IBoard** - An interface that describes what a concrete implementation of a board needs for the more general BoardManager to work properly
- **IObservable** - An interface that makes it possible for an IObservable to send information to an IObserver without making the IObservable dependant on the IObserver
- **IObserver** - An interface that makes it possible to send objects to the Iobserver without making the IObservable dependant of the IObserver
- **IPhase** - An interface that describes the overall structure of a phase

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://www.jetbrains.com/idea/)