Requirements and Analysis Document for Project Robo Rally (RAD)

Version:
ver 1.1 - All participants
This version overrides all previous versions.

1. Introduction

This project aims to create a fully playable computer based version of the popular boardgame RoboRally™. The game will follow the original game rules and be played by players on the same computer or over network.

1.1 Purpose of application

Purely for entertainment purposes.

1.2 General characteristics of application

Turn-based board game. Multiplayer over network with 1-8 players.
Possible to play offline on a single computer against persons in the room as well.

Each player has their own Robot which is controlled by picking different cards that have move-orders on them. The goal is to race to all checkpoints in the correct order, first, second checkpoint etc, and avoid falling off the GameBoard or taking damage from other Robots lasers.

1.3 Scope of application

A complete digital representation of the original board game RoboRally™ by Avalon Hill Games. All of the game-rules and win-conditions should be implemented. It is to be played in multiplayer over network or on the same computer.

1.4 Objectives and success criteria of the project

Playable with all necessary functions for adequate gameplay. Multiplayer offline is a must. Online if possible within time frame.

1.5 Definitions, acronyms and abbreviations

- GUI, the graphical interface i.e all the buttons and look of the game.
- Java, platform independent programming language.
- JRE, the Java Runtime Environment. Additional software needed to run a Java application.
- IntelliJ IDEA, Integrated Development Environment. An advanced text editor for writing and producing Java code with various additions.
- Host, a computer where the game will run.

- Card, a card which a specific robot movement is written on.
- Round, starts with the choosing of cards for all players. A round consists of five Turns.
- Turn, during a turn all players cards are revealed in order, five cards total, and all actions and effects from the moving the player is executed.
- GameBoard, the board where the robots move around. Consists of different types of tiles.
- Tiles, have different attributes that affects the robots in some way.
- Checkpoint, a tile that has the attribute of a checkpoint, all players must reach these checkpoints in the right order to win the game.

2. Requirements

In this section we specify all requirements

2.1 Functional requirements

1. Start new game
2. Choose how many players
3. Input name for player
4. Choose map
5. Player should have access to rulebook
6. Click ready
7. Start Round
8. Shuffle cards
9. Deal cards
10. Pick registercards
11. Choose whether to power down or not
12. Do five turns
13. Move robots (according to register each turn)
14. Robots take damage and die
15. Robots can fall off edge
16. Robots can spawn on last checkpoint
17. Robots can be moved on Conveyors
18. Players can use Robot-upgrades
19. Cards gets shuffled between each round
20. Players gets new cards according to how many damagetokens
21. Players registers can become locked if to much damage is taken
22. Players lose lifetokens and eventually dies
23. Player can win on last checkpoint or when everyone else is dead
24. Player can withdraw and end game
25. Player can restart game

2.2 Non-functional requirements

2.2.1 Usability

Simple, clean and understandable user interface is our goal. Anyone who has played a boardgame or computergame should be able to play and understand the game with help from rulebook.

2.2.2 Reliability

The user should be able to set up a game without experiencing any bugs.

2.2.3 Performance

The game should be responsive and have zero loading times.

2.2.4 Supportability

Desktop application. Our goal is to have Multiplayer support.

2.2.5 Implementation

The application will be runnable and adapted for Desktop/Laptop, OS X, Window osv..

2.2.6 Packaging and installation

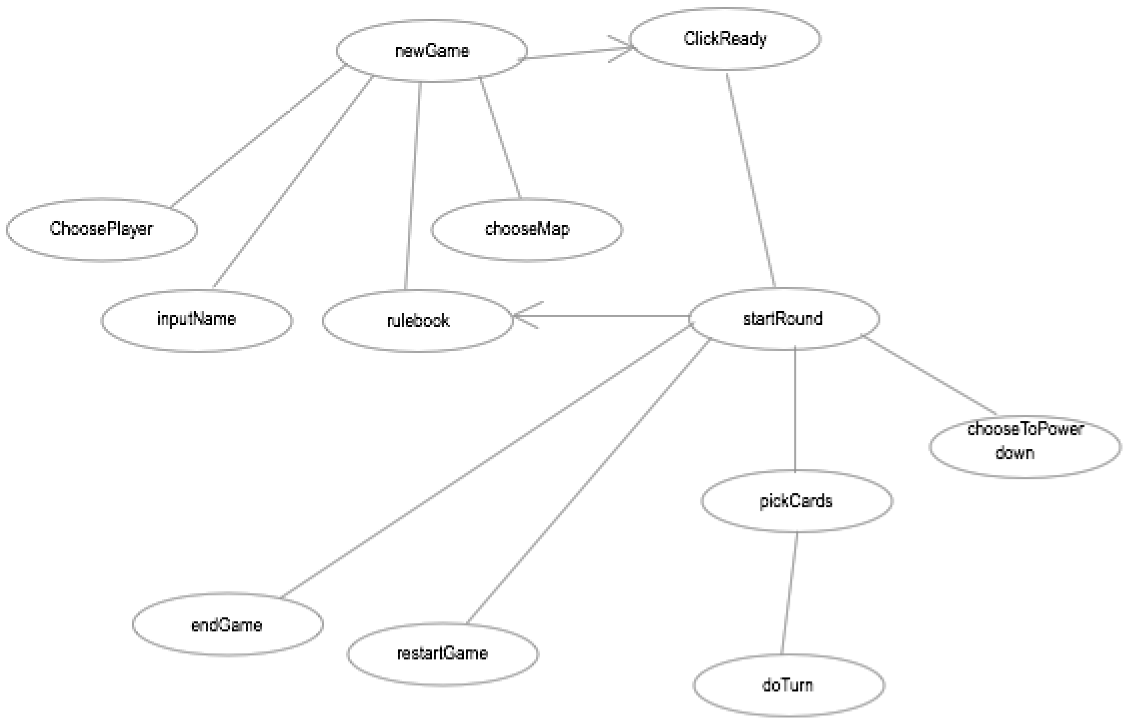
The application will be distributed through a GIT repository which will contain all of the necessary files to compile and run the application. It will also contain a READMEfile that explains what has to be done to be able to start the application.

2.2.7 Legal

We don’t own the rights to the original game so we cannot publish it.

2.3 Application models

2.3.1 Use case model



2.3.2 Use cases priority

newGame	**High** (See appendix 2.4.3 Sequence Maps)
startRound	**High**
doTurn	**High** (See appendix 2.4.3 Sequence Maps)
chooseMap	**High**
pickCards	**High**
conveyRobot	**High**
Robot into pit	**Medium**
fireLaser	**Medium**
repairRobot	**Low**
usePerk	**Low**

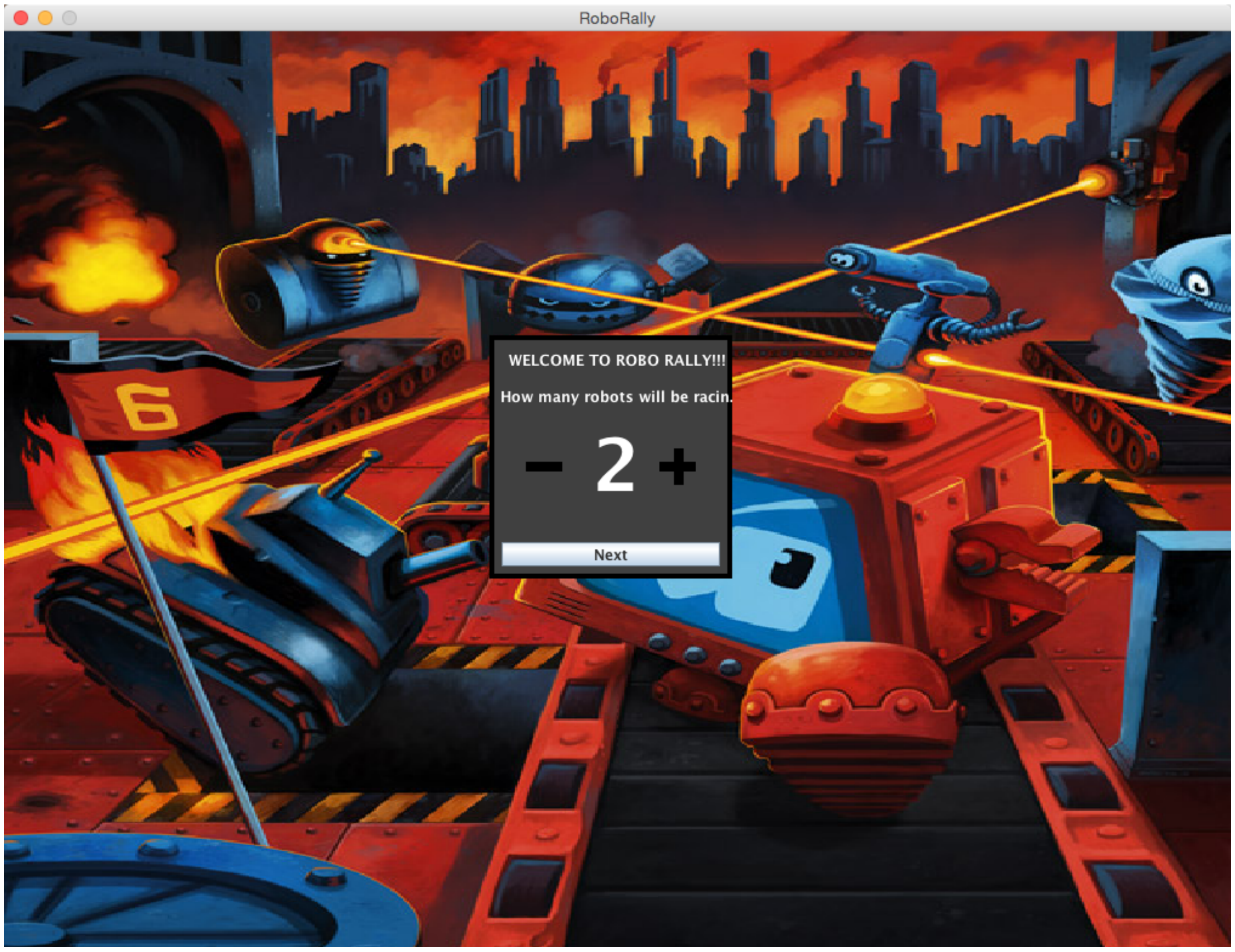
2.3.3 Domain model (See appendix)

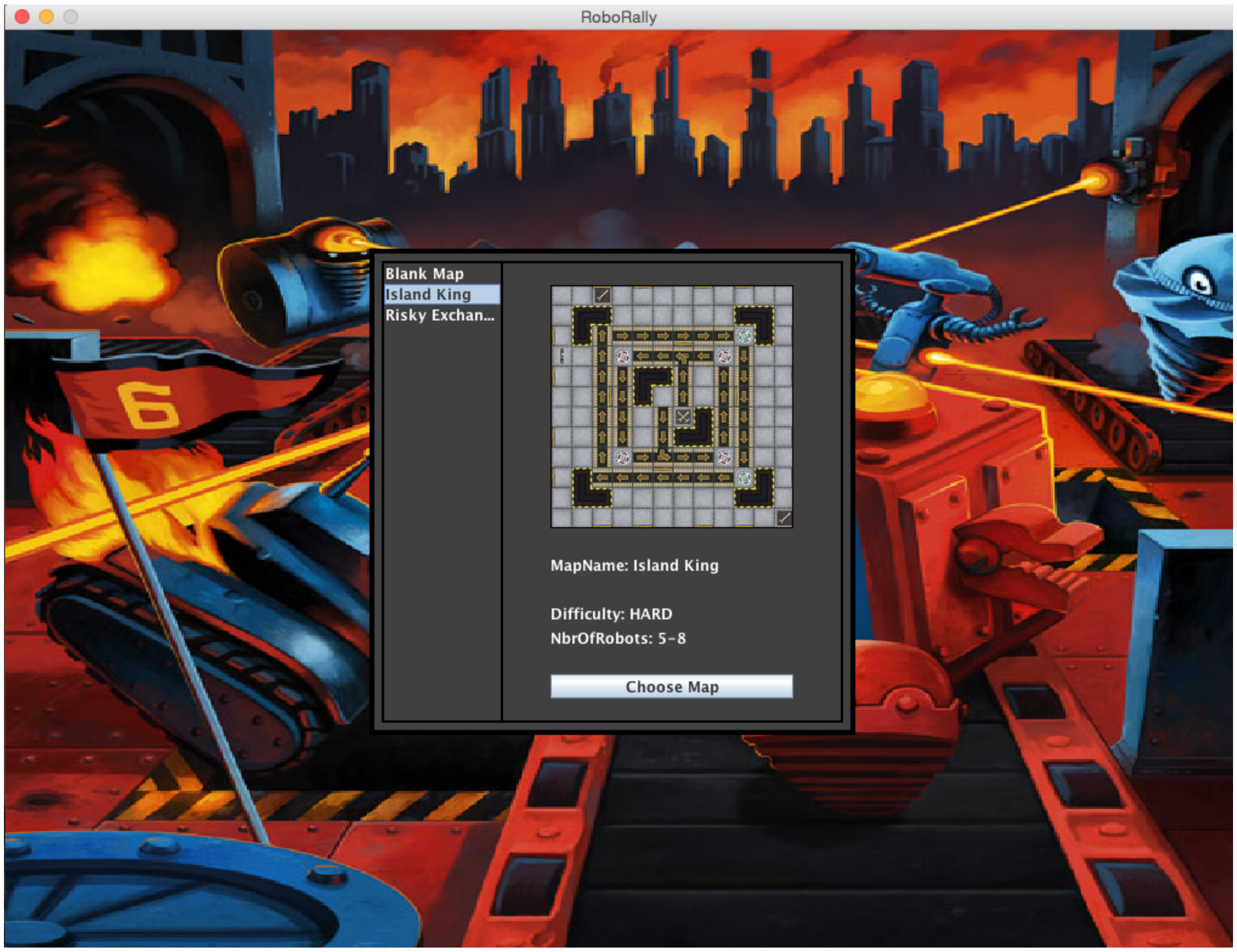
2.3.4 User interface

The GUI will be similar to the original analog board game. Should give the feeling that you're sitting by a table and playing the game.

2.4 Appendix

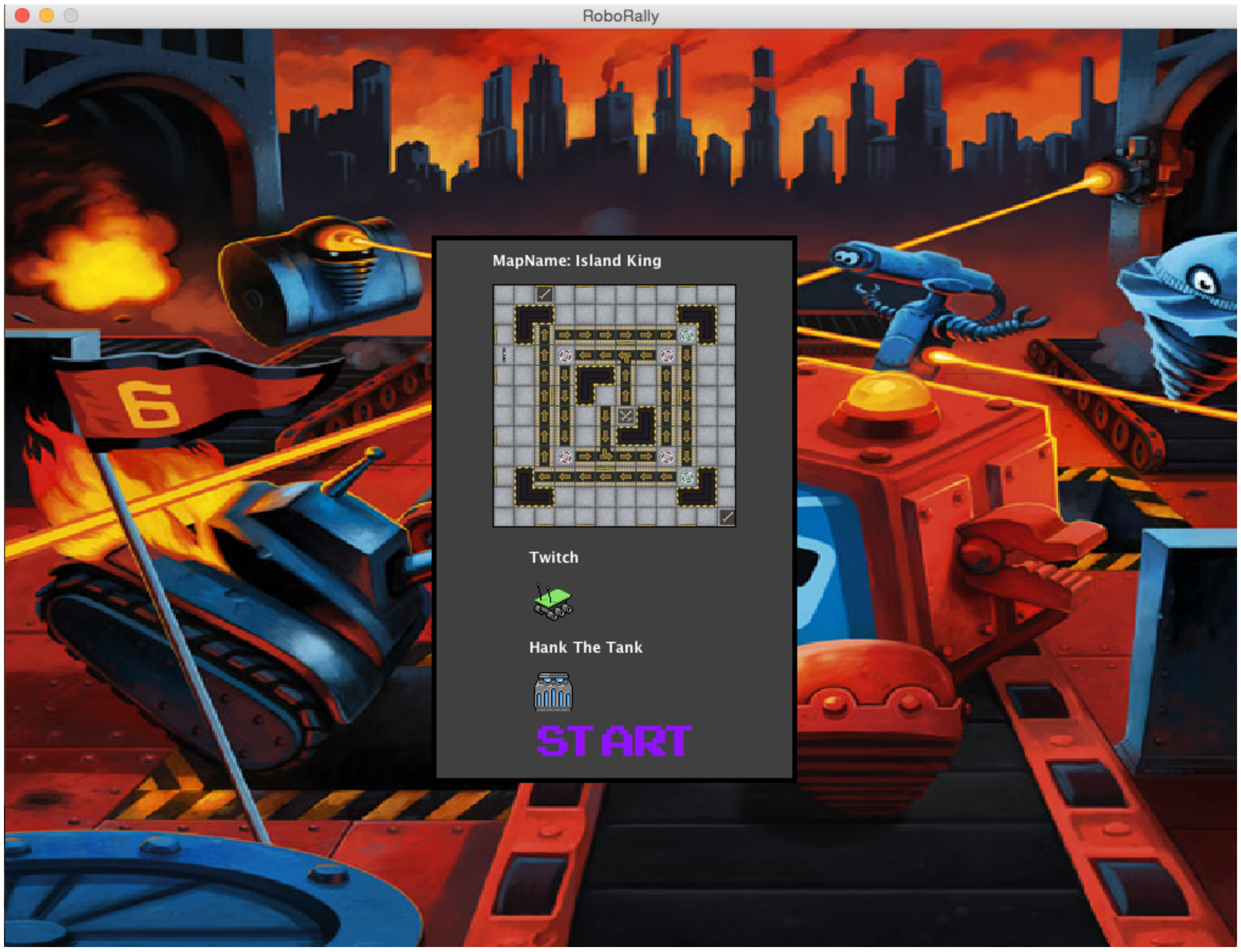
2.4.1 GUI

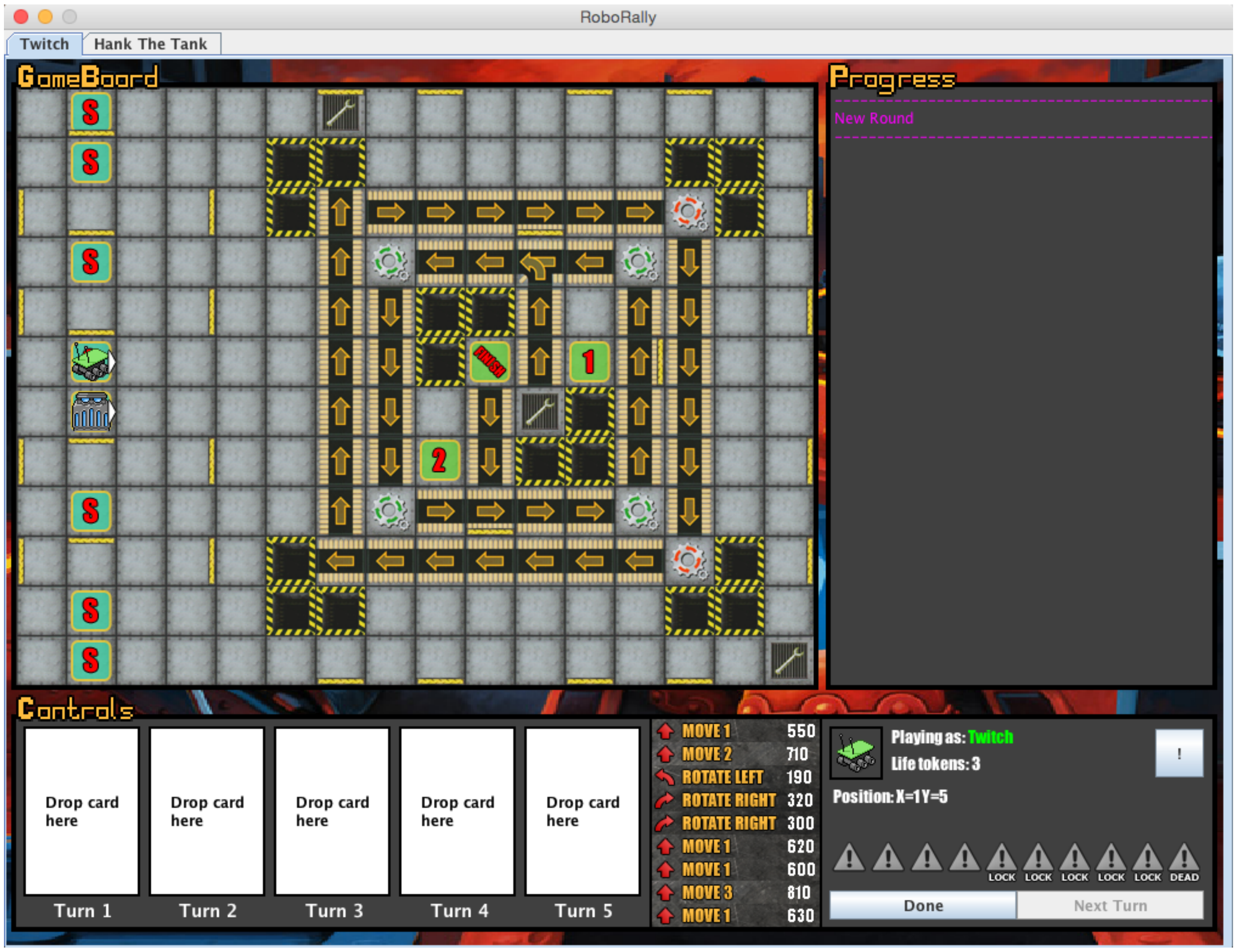




MapName: Island King

Difficulty: HARD
NbrOfRobots: 5-8


RoboRally
MapName: Island King
Twitch
Hank The Tank
START

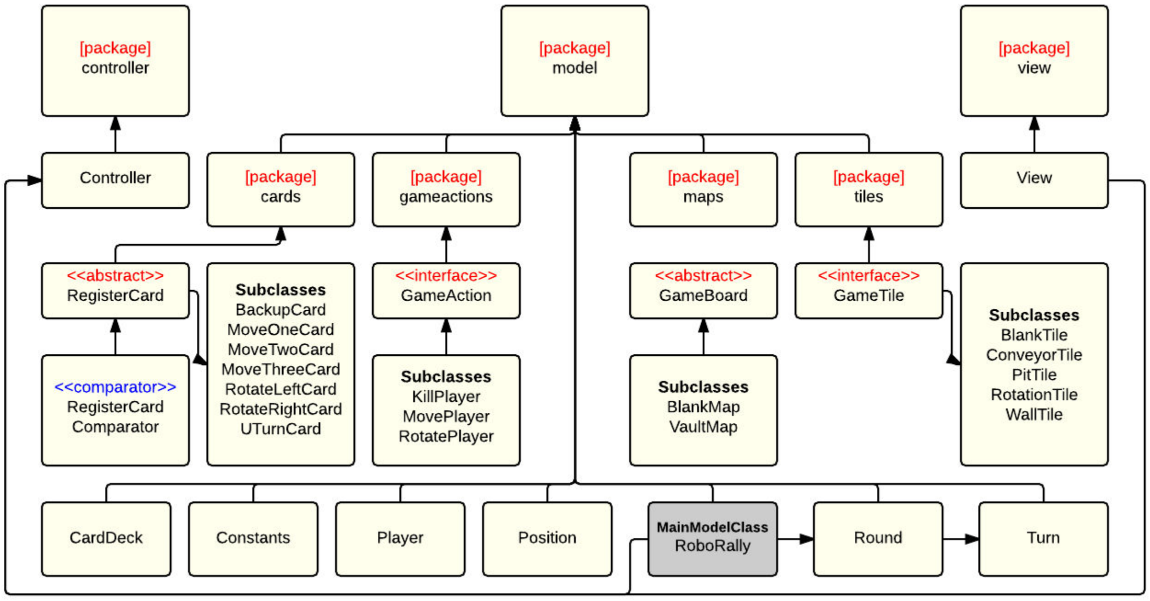


New Round

MOVE 1	550
MOVE 2	710
ROTATE LEFT	190
ROTATE RIGHT	320
ROTATE RIGHT	300
MOVE 1	620
MOVE 1	600
MOVE 3	810
MOVE 1	630

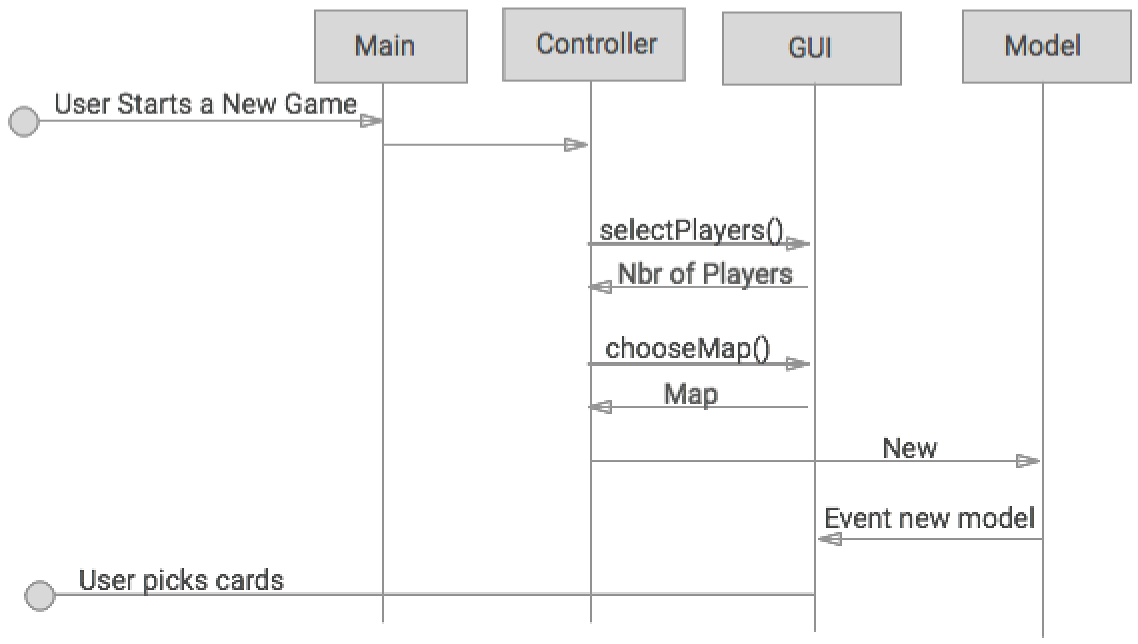
Playing as: Twitch
Life tokens: 3
Position: X=1 Y=5

2.4.2 Domain model

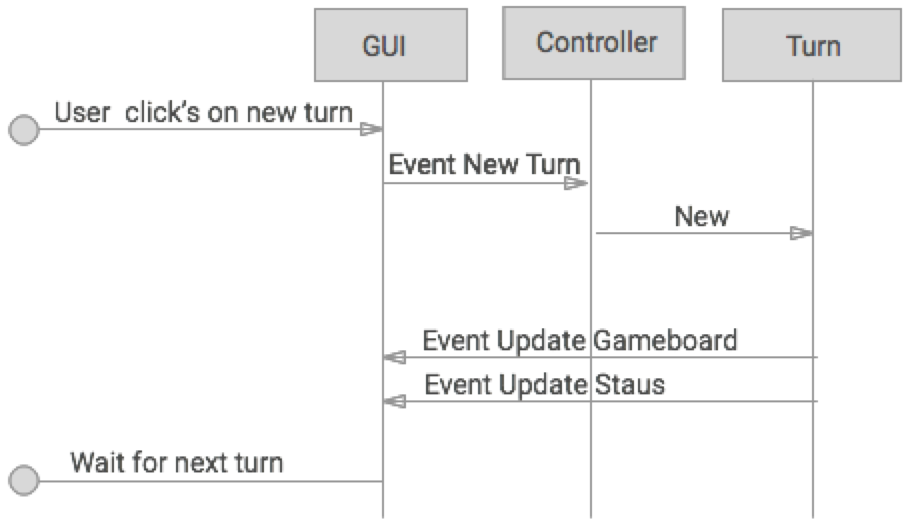


2.4.3 Sequence Maps

Use Case: newGame



Use Case: doTurn


GUI
Controller
Turn
User click's on new turn
Event New Turn
New
Event Update Gameboard
Event Update Staus
Wait for next turn