Réseau Figure
CURSUS MASTER EN INGÉNIERIE

UFC
UNIVERSITÉ
DE FRANCHE-COMTÉ

ufr st

Javascript Project Report

ROBOCODE

Reynald Barbeaut AND Timothée Guy

L3 CMI Computer-Science
2017 — 2018

Teacher: M. Frédéric Dadeau
FEMTO-ST Institute — DISC Department

Contents

List of Figures

1 Introduction

This report will cover the Javascript project Robocode. Robocode is a two players game where you have to capture flags and bring them back to your base.

The application has been made with the following programming languages:

- HTML5
- CSS3
- Javascript 8
- Node.js

2 Features

The application is available in two different modes: online multiplayer and local multiplayer. Below, you can see the use case diagram of the application, describing the interactions of the user, that will be described further.

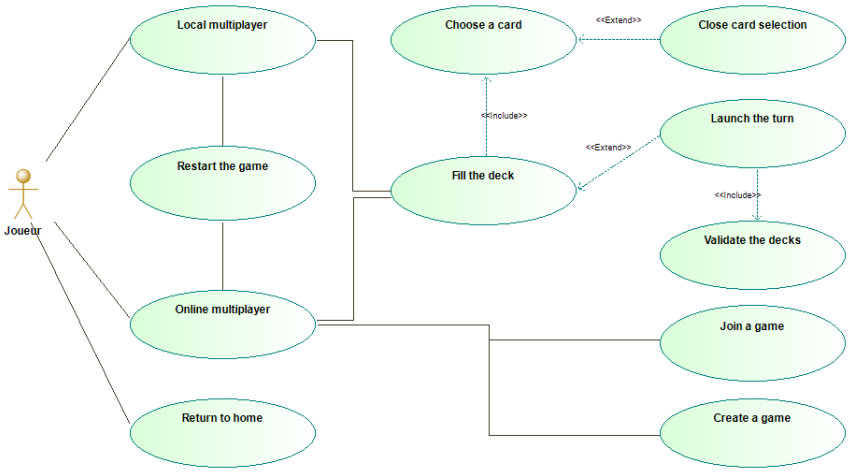

Figure 1 – Use case diagram

2.1 Online/Local multiplayer

To use the online multiplayer, the user will have to join the tchat of Robocode and create the game by using the following command in the tchat: `/invite:username`. Then, the other user will have to join the game by using the following command in the tchat: `/join:username`. When using the local multiplayer mode, the game is launched automatically, because we don't use the tchat to launch it, but a direct HTML link in the homepage of the website.

2.2 Common features

The two modes are distinguished by the way they launch the game, but the core of the game is the same. The game isn't complex, so the user has few numbers of interactions available, he can:

- restart the game: by refreshing the page or clicking on the "restart" button when the game is finished
- fill the deck: when the game is launched, the user can fill his deck of cards that will be executed by the robot
- choose a card: a window will open where the user can pick a card, in extension, the user can close the card selection window
- launch the turn: when either blue and red decks are validated, the player can choose to launch the turn in the local mode, or the server will launch the turn in the online mode
- return to home: the user can also choose to return to the homepage of the website

3 Data Models

In this project, we used three different class : `Robot`, `Flag` and `Card`. We also used a set of global variables which we aren't going to describe here.

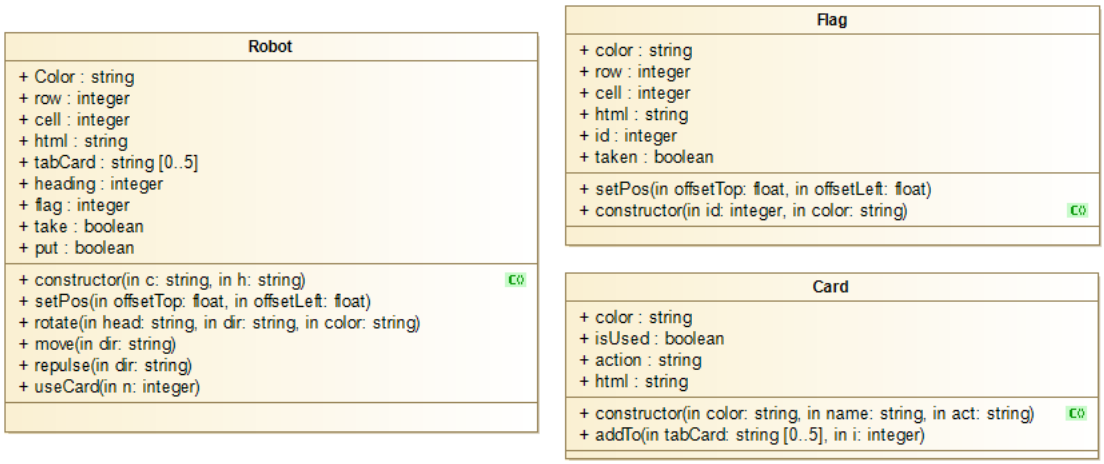

Figure 2 – Class diagram

3.1 Robot

The class `Robot` represent a robot. It's used to represent the blue and red robot in the game. It has 9 variables representing different specifications of the robot, and 6 methods detailed below:

- `constructor`: the constructor of the Class
- `setPos`: sets the position of the robot
- `rotate`: rotates the robot
- `move`: moves the robot in a direction
- `repulse`: repulse another robot
- `useCard`: execute an action of a Card

3.2 Flag

The class `Flag` represent a flag. It's used to represent all the red and blue flags in the game. It has 6 variables and 2 methods detailed below:

- `constructor`: the constructor of the Class
- `setPos`: sets the position of the flag

3.3 Card

The class `Card` represent a card. It's used to represent the different cards in the game. It has 4 variables and 2 methodes detailed below:

- `constructor`: the constructor of the Class
- `addTo`: adds the Card to a deck

4 Animations

There are several animations in the game, one for every movement of the robot (north, south, east, west, eastx2, westx2) and also an animation when the robot is idle or doing an action that doesn't imply moving.

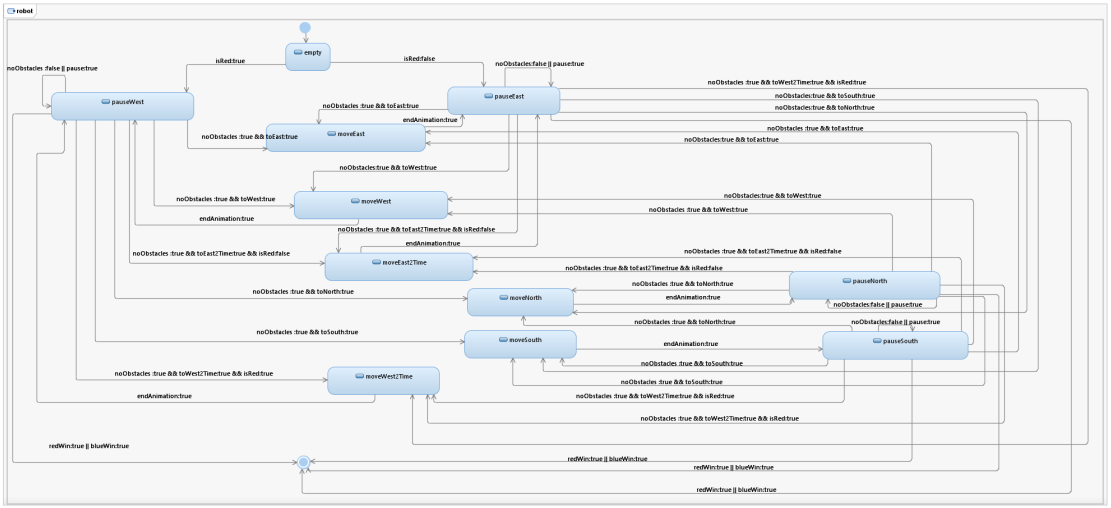

Figure 3 – State and transition diagram

When the robot moves, if the direction doesn't change, it goes straight forward, but if the direction change, it turns and goes forward at the same time. When it doesn't moves, the robot gets a little bit brighter for a second, to show that he's doing an action where it is idle.
On the diagram above, you can see all the different states and conditions of the animations.

5 Protocols of communication

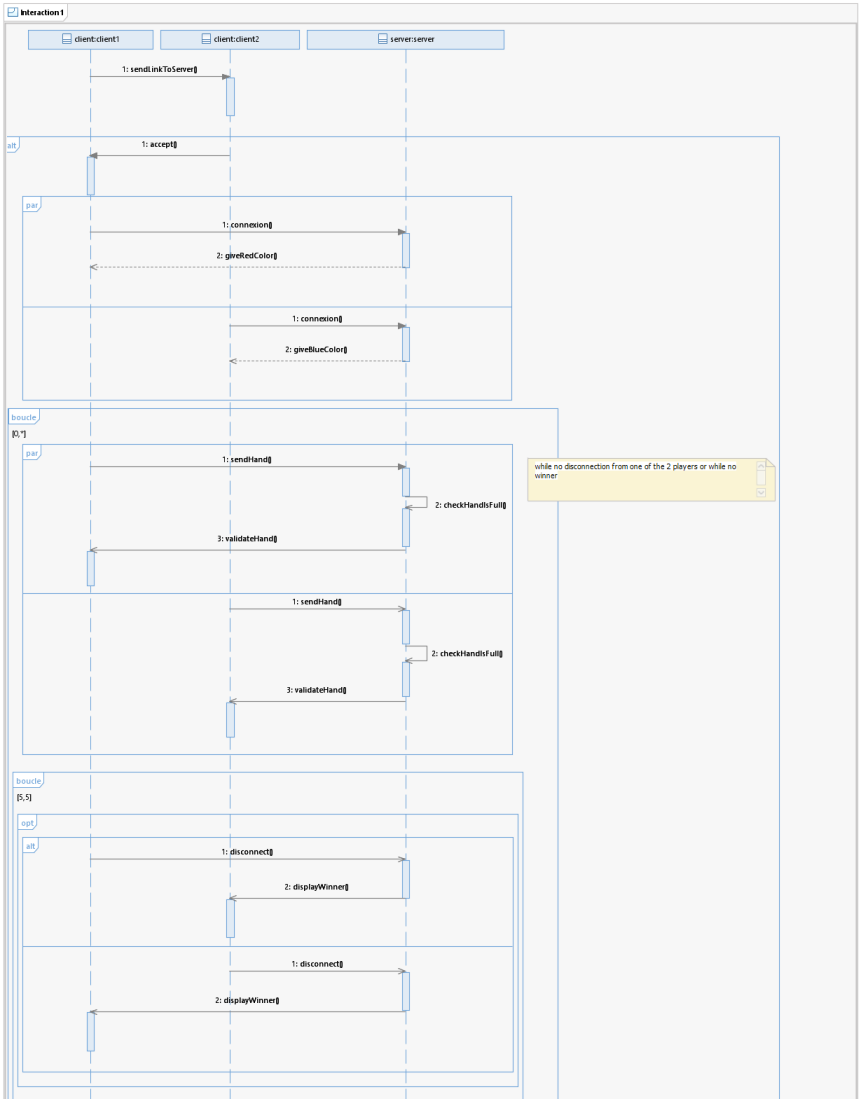

Figure 4 – Sequence diagram pt.1

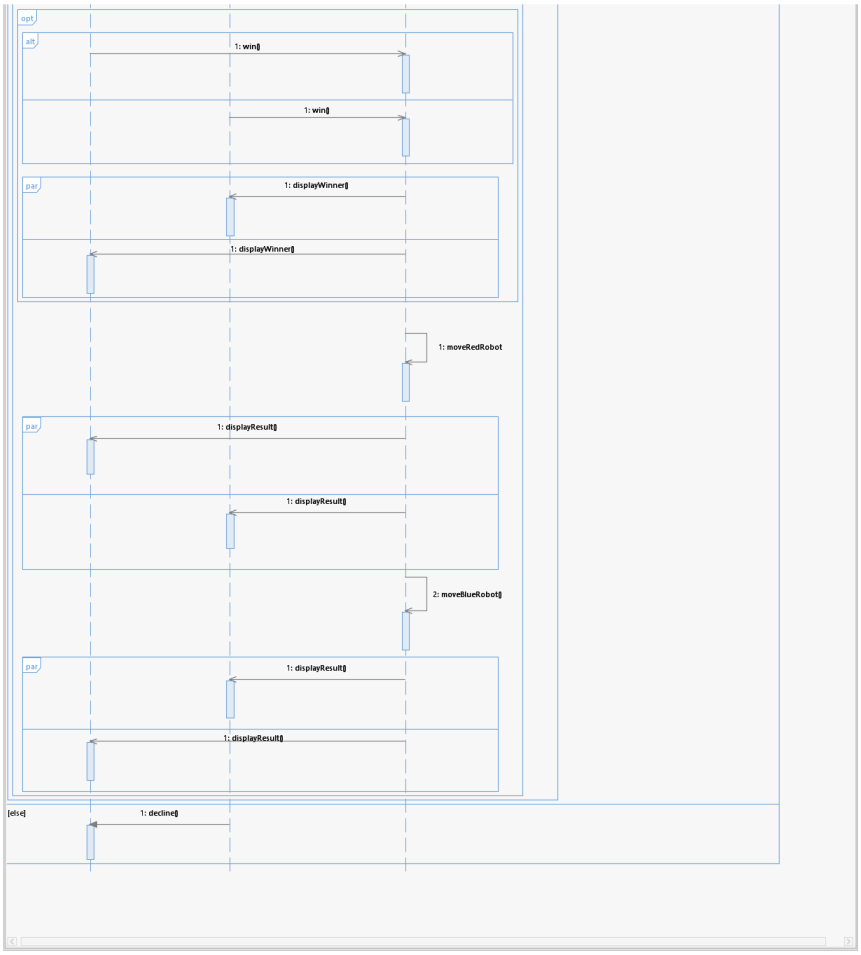

Figure 5 – Sequence diagram pt.2

The first player send an invitation to the second one, then, ask the server for the creation of a game. The server gives each player a color when they connect to it. Then the game starts. Each player gives its hand to the server. Then, the server displays the game to the two players until one of theim win or disconnect. In that case, the server sends a message to the player which stayed. If a player wins, it sends a message to both players. Also if a player doesn't accept the game the server doesn't launch it.

6 Work process

The work was equally split between us. One of us was in charge of implementing the card selection system and the server modification, when the other was in charge of implementing the progress of the game (the turn, the robot action, ...) and making the report. Of course, we also helped each other and worked together when required.
Also, we used GitLab to share our work and manage the different versions of the application.

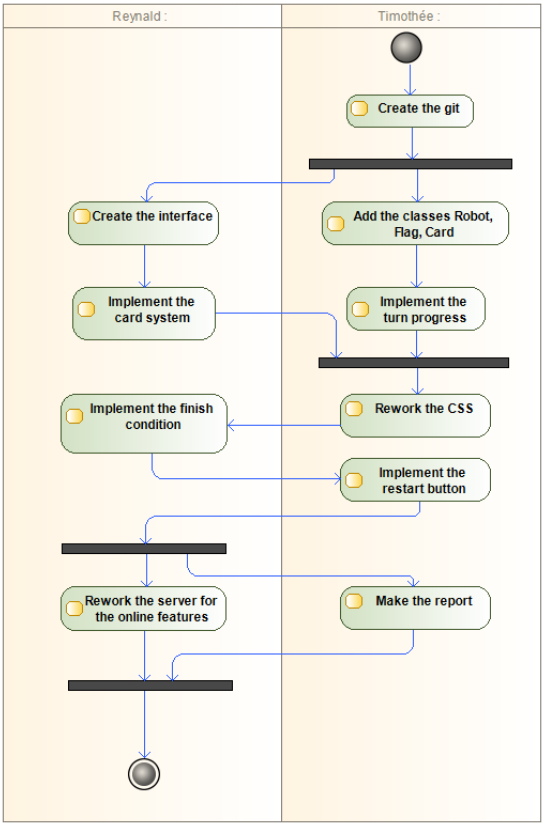

Figure 6 – Activity diagram

7 Conclusion

We managed to make the local multiplayer game without major difficulties, except the management of the animations that was leading to bugs of position of the HTML elements like the robots or the flags.
Unfortunaly, we didn't succeed in making the online game working. The players can only create or join a game, choose their decks and that's all. Also we didn't succeed in resolving differents problems like ragequit. The project also extended our practice of Javascript, CSS and HTML, and we are now more at ease with those programming languages and the interactions between them.