# Introduction

# Implementation

User interface:

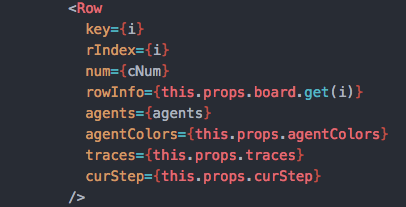
React.js is used in this project as a framework to build the user interface. It not only a JavaScript library, but also a thought of building user interface.

Componentization is one of the features of React.js and it heavily affected the way we develop the user interfaces. Writing React.js code is like using building blocks. We first use some basic components to build some higher-level components and then use those higher-level components to build some other components on top of them. The whole web application is built up in this way. This makes the UI structure clear and allows us to write reusable code. If a component is used in many parts of the code and we want to change its style or add some stuff in it, we only need to change one part of the code in where this component is built up. This saved us a lot of time and made it possible to spend some time on making the application looks better and easier to use.

Figure 3.1 – 3.3 shows an example of componentization used in this project.



3.1 Using some divs to build the “Row” component



3.2 Using the “Row” component to build the “Board” component (part of the code)



3.3 Using “Board” components to build the “Visualization” component (part of the code)

Another core feature or thought in React.js is “state”. We took this thought and implemented the user interface. A “state” is a storage of an attribute that will affect the appearance of user interface. In a traditional way, if a user changed some data after some operation and the data have some relations to the user interface (like the position of an agent), then there should be some code to update the user interface with these new data. It would be much complex if there are so much logic and elements on the user interface. While in the “state” way, these data are stored in “state”, what we only need to focus on is the relations between the “state” and the user interface. After these relations is built, the changes on a “state” will automatically reflects on user interface. In this project, “state” appears everywhere. Most updates on the user interface (like agents’ positions, visited spaces, target lists and the showing of graph views) are achieved by changing the “state” using the “setState” method.

For the graph view, we decided to use canvas to draw the graph. Compare to just using some basic HTML elements (like div) with CSS to draw a graph, canvas is much more flexible and powerful. Besides, there are a lot of excellent third-part libraries for constructing a canvas graph. D3 is selected as the library to develop the graph view, because some team members have already learned a little about it. A function called “graph” is implemented to build the graph. This function takes three parameters: region, traces and step. The “region” tells the structure of selected region, the “traces” tells the trace of each agent and the step tells the current step. When another region is selected or the “run step” button is clicked, this function will be called to build the new graph.

# Technology and tools

Npm. It is used in this project to install and manage JavaScript packages.

Webpack. It is a module bundler for JavaScript applications. We used this tool to achieve modular development using JavaScript.

Babel. It is a compiler. By using it, we can write JavaScript code with new features without caring about the compatibility in old browsers. It will compile the code to old version that supported by old browsers. It also compiles JSX which used in React.js to plain JavaScript.

Eslint. It is a tool for identifying and reporting on patterns found in ECMAScript/JavaScript code, with the goal of making code more consistent and avoiding bugs [1].

D3

VSCode

Node.js

Express

ECMAScript 2015

Git

Github

MongoDB

Linux

# Challenges

Because this project is developed by three team members, the interactions between different modules developed by different people is a big problem. Therefore, at the very beginning, the team reached an agreement on the structures of data transferred between different modules. Besides, the team also discussed on how a module should be used in other modules and reached an agreement. After doing this, the three people just focus on developing their own modules and we did not encounter too much trouble when combining these modules together into one application.

# Learning

Before this project, I did not have much experience on a project which developed by more than two people. By developing this project, I realized that even three people are much different than two people. The difficulty of the management on project increased a lot. The communication between team members is much harder than just two people. The meetings gave us a good way to communicate with each other on the project and solve problems. Face to face communication is much more effective than communicating on Internet.

I learned Scrum a lot. Though developing this project may not be the same as developing a project in a real company, it still gave me a picture of how Scrum works and how it would affect the developing process. We need to care about sprint tasks, we need to care about both coding and testing. We also need to adapt to the changes on requirements.

I also learned many new technologies in the web world. All code is written in ECMAScript 2015 (also known as ECMAScript 6). There are many new features like arrow functions, let, const and promises. In the next few years, these features will be used more and more frequently in the area of web development. We also used React.js, Webpack, Babel, Node.js which also let me learned a lot.

[1] http://eslint.org/