Rusty Connect-4 & TOOT-and-OTTO Design Document

By: Robert Mah, Ze Hui Peng, Charlie Chen

Major Innovations

- Our board allows the user to resize the board so the players are not limited to using the general 7x6 board for Connect-4 and general 6x4 board for TOOT and OTTO. Users may choose between a 7x6, 5x4, 6x4, 6x5, 8x7, 9x7, 10x7 or an 8x8 board.
- We also allow users to check on game history and champion players for each of the board games, and the user to clear the game history if they wish.
- We disallowed players to have empty names, we did a check on the player names and will alert the user if the player name field is empty

Rationale for Augmented Decisions

1) What can we do on a computer that we can't do on a printed board?

For a computer, we can dynamically set the dimensions of a board so the game is not limited to a 7x6 board(or 6x4 board for TOOT-and-OTTO). For a printed board, another human opponent is usually required. This is not the case for a computer as one can play against an artificial opponent in the form of a simple AI. We can also connect our games to a database for statistics, play connect4 and other games with our friends while practicing social distancing, and other things like play validation and easier win condition checking.

2) What is a computerized opponent? What are its objectives? (Remember, not everyone is an expert.)

- What characteristics should it possess? Do we need an opponent or opponents?

A computerized opponent is some logic that can play the game and calculate where to place the next disc. In the case of Connect 4 and TOOT-and-OTTO, it is essentially trying to maximize the score to the AI itself, and the algorithm considers that to be winning. Or minimize the score to the human player, which the algorithm considers that to be preventing the human player from winning.

3) What design choices exist for the Interface components?
- Color? Font? Dimensions of Windows? Rescale-ability? Scroll Bars? ….
For color, the traditional red and yellow colours for the chips in Connect4 are acceptable for people with typical colour blindnesses of protanopia, deuteranopia, or tritanopia. The traditional yellow colour was substituted for a more orangey hue since yellow appears like white for people with deuteranopia. The colour options were checked using https://gka.github.io/palettes in order to ensure the colours worked in as divergent colours. Furthermore, the chips also have an R or Y printed on them to help people with more severe color blindness and for general accessibility. For TOOT-and-OTTO the color are the same for both players, since for this game only the text matters(i.e. ‘T’ and ‘O’).
For the font and dimension of windows, we just used the same configuration from the MEAN stack code as the original GUI looked good for us.
For rescale-ability, we have the ability to hide the side menu and only display the main menu with a header text “Connect 4 with Rust” if the user has a small window size (ie. mobile users). The main design came from the MEAN stack code so mobile users will be required to turn to landscape mode (or the browser needs to be sufficiently wide) in order to access the sidebar.
For scroll bars, we didn’t really have any scroll bars designed, the entire GUI should be visible in any screen of at least 13-inches, the browser provides a scroll bar if the html components don't fit on the screen.

4) What does exception handling mean in a GUI system*?*
It means showing a warning to a user in the worst-case scenario. If a user attempts to perform an action that would result in an error, then a model dialogue should appear and tell the user that they cannot perform the action. If exception handling is not performed, then the system will break and the user does not understand what happened unless the user knows how to look at the console terminal. Furthermore, for browsers there are different ways the underlying engine works so there can be any number of things that are broken. The objective is to have the system fail gracefully such that any errors can occur, but the GUI still remains responsive instead of freezing or having deadlock.

5) Do we require a command-line interface for debugging purposes????? (The answer is yes by the way – please explain why)
Having a command-line interface(CLI) allows us to debug our Model(which is code related to game logic) without needing the View or the Controller. The benefit of this is if the View and/or the Controller stopped working for some reason, we can still use the CLI to test our Model and make sure the Model is free of error.

6) What are Model-View-Controller and Model-View-Viewmodel? Are either applicable to your design? Justify your answer.
Model-View-Controller (MVC) is a software design pattern, commonly used for user interfaces.
The model contains the state and logic of the system; the view takes the state to generate the display elements, and the controller handles how state is updated in the model when the user interacts with the view.

Model-View-Viewmodel (MVVM) is another software design pattern. The model and view is similar to the model-view-controller. However, the viewmodel is responsible for converting the data from the model and binding it to the view such that updates in the view or model are received by each other.

For our project, we used MVC, where the model is the game logic that handles placing a disc, checking winning conditions, board state, and CanvasModel acts as a proxy model that the web can interact with while maintaining the states. View is the pages with html components that will be displayed on the web, and controller is the on click handlers with the controller logic being in the update method. We also use the MVVM implicitly with yew where ShouldRender true returns cause the data bound within the model struct to be updated on the display. The viewmodel is done with the change method which is only used in the components as props change from passed parameters.

List of Known Errors, Faults, Missing Functionality

Sometimes the AI needs around 1-2 seconds to calculate the next disc to place, but that should not be a big issue in the game. Considering it as a human requiring time to think about where to place its piece. There is no UI indicator that tells the player that it is their turn. The player is free to take his turn once the AI has placed its chip.

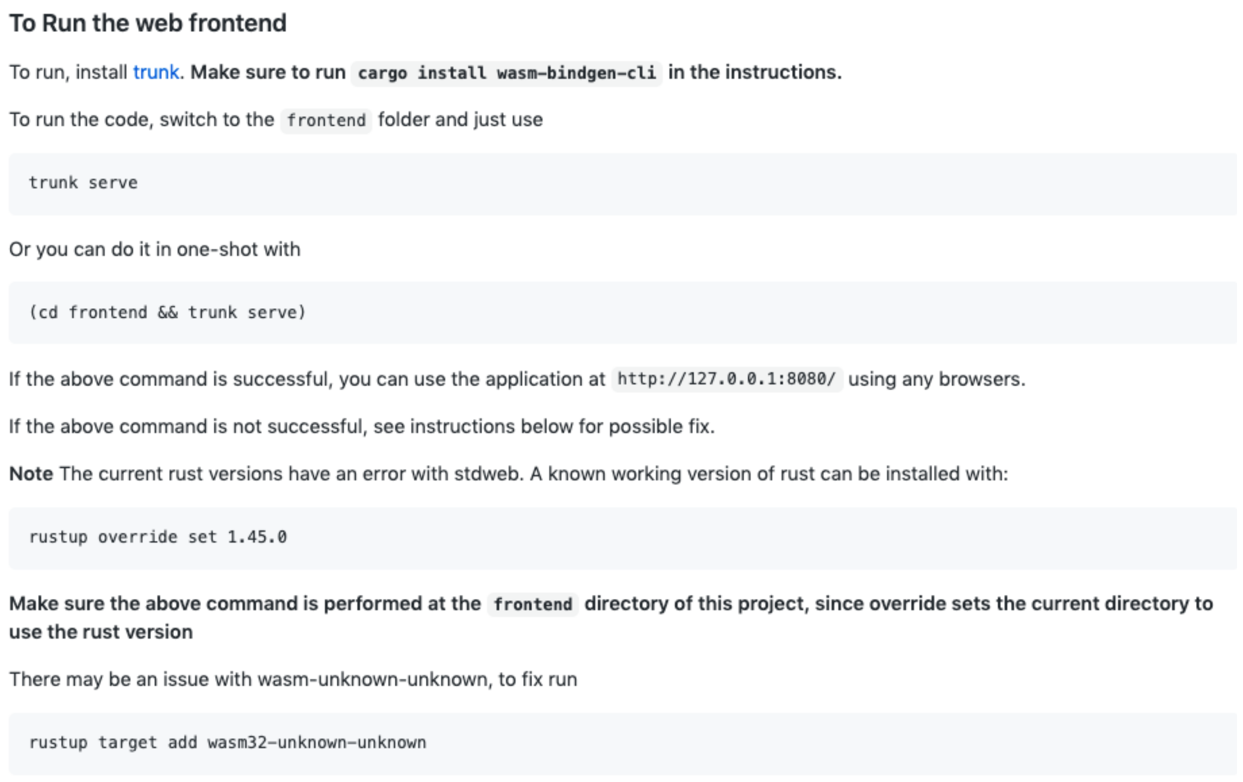
Detailed Description of MEAN stack code

The only thing left over from the MEAN stack is MongoDB, which is rewritten using the official Rust mongodb driver and the font styles which are used for the font and dimension of the windows. Everything else has been replaced with Rust and WebAssembly. In our project we have no developer written javascript and the only HTML used is what is required to load our webassembly.

User Manual

The README.md file should have all details required to run the application, a screenshot of the README is included in this document as reference.

An important point from README is make sure to run the **frontend** folder with Rust (not nightly) 1.45.0(preferred, higher version has compatibility issues with the **stdweb** crate). And make sure to run the **backend** folder with Rust 1.53.0-nightly(preferred). Detailed instructions on how to set or switch the Rust versions are mentioned in README.



To Run the web frontend

To run, install trunk. **Make sure to run `cargo install wasm-bindgen-cli` in the instructions.**

To run the code, switch to the `frontend` folder and just use

```
trunk serve
```

Or you can do it in one-shot with

```
(cd frontend && trunk serve)
```

If the above command is successful, you can use the application at `http://127.0.0.1:8080/` using any browsers.

If the above command is not successful, see instructions below for possible fix.

Note The current rust versions have an error with stdweb. A known working version of rust can be installed with:

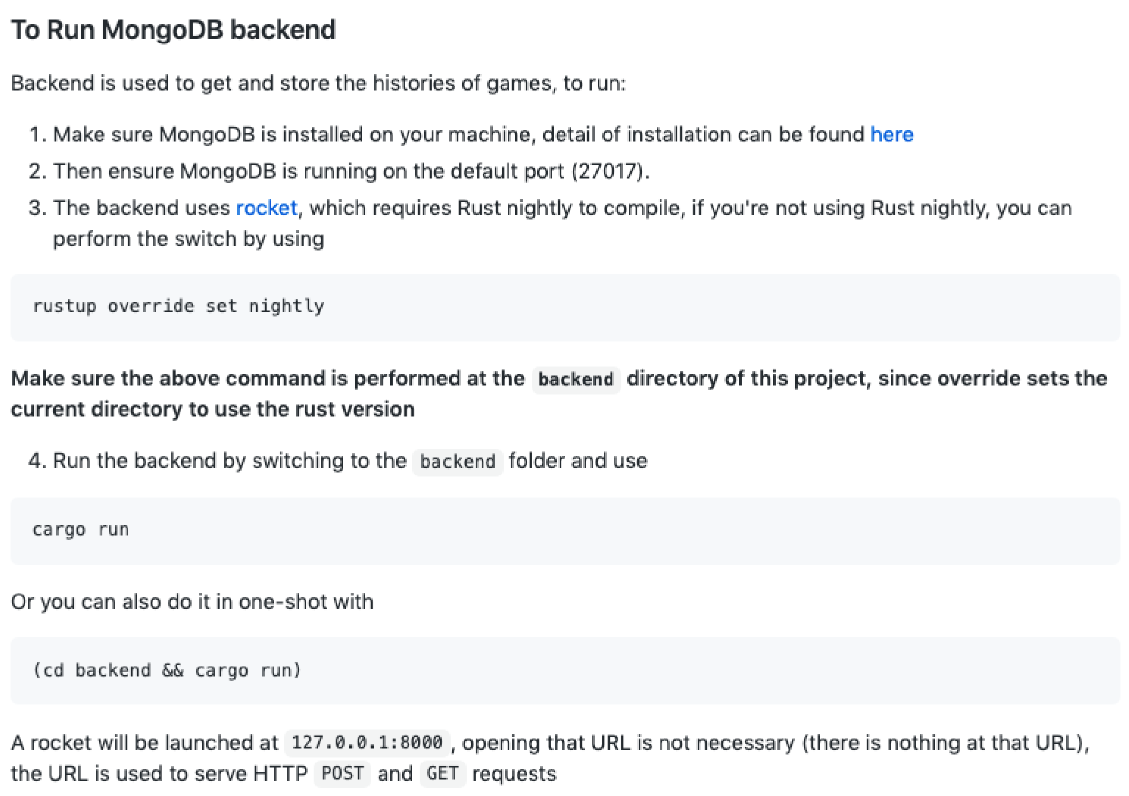
```
rustup override set 1.45.0
```

Make sure the above command is performed at the `frontend` directory of this project, since override sets the current directory to use the rust version

There may be an issue with wasm-unknown-unknown, to fix run

```
rustup target add wasm32-unknown-unknown
```

Link to **trunk:** https://crates.io/crates/trunk



To Run MongoDB backend

Backend is used to get and store the histories of games, to run:

1. Make sure MongoDB is installed on your machine, detail of installation can be found here
2. Then ensure MongoDB is running on the default port (27017).
3. The backend uses rocket, which requires Rust nightly to compile, if you're not using Rust nightly, you can perform the switch by using

```
rustup override set nightly
```

Make sure the above command is performed at the `backend` directory of this project, since override sets the current directory to use the rust version

4. Run the backend by switching to the `backend` folder and use

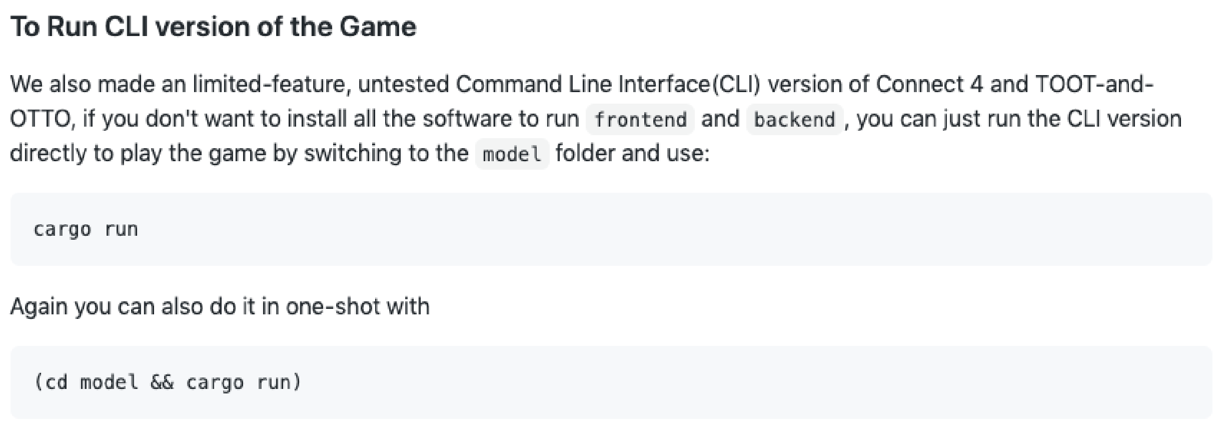
```
cargo run
```

Or you can also do it in one-shot with

```
(cd backend && cargo run)
```

A rocket will be launched at `127.0.0.1:8000`, opening that URL is not necessary (there is nothing at that URL), the URL is used to serve HTTP `POST` and `GET` requests

Link to **here**(MongoDB installation): https://docs.mongodb.com/manual/installation/
Link to **rocket**: https://github.com/SergioBenitez/Rocket/tree/v0.4



To Run CLI version of the Game

We also made an limited-feature, untested Command Line Interface(CLI) version of Connect 4 and TOOT-and-OTTO, if you don't want to install all the software to run `frontend` and `backend`, you can just run the CLI version directly to play the game by switching to the `model` folder and use:

```
cargo run
```

Again you can also do it in one-shot with

```
(cd model && cargo run)
```

To Use the Program:
Once the frontend and backend is running, the user should copy and paste the generated link into the address bar of their browser.

The link is located in the command line output after the frontend is run with “trunk serve”, at line “INFO server running at <link>”. The <link> is the address of the webpage for the application. By default it should be http://127.0.0.1:8080/.

The instructions for playing the game are located in the sidebars. Click on “How to Play Connect4” for instructions on how to play Connect4. Click on “How to Play TOOT-OTTO” for instructions on how to play TOOT-OTTO.

The application also gives users the ability to play against an AI opponent. The user may select either “Play Connect4 With Computer” or “Play Connect4 With Computer”. The user will then be greeted with a textbox and 2 drop down menus. The text box is the player name and must be filled in. The first drop down menu selects the difficulty of the AI opponent: easy, medium, hard and insane. The second drop down menu allows the user to choose the dimensions of the board from preset dimensions: 7x6, 5x4, 6x4, 6x5, 8x7, 9x7, 10x7 and 8x8. The “Start Game” button starts the game with the settings currently selected in the drop down menus. Once a game has concluded, the user may choose to play again by clicking on the game board, which will allow the user to change their player name, and the settings for the board and AI opponent.

The user can choose to view their game history by selecting “View Game History” from the sidebar. The user may choose to clear their game history by clicking on the “Clear Game History” button.