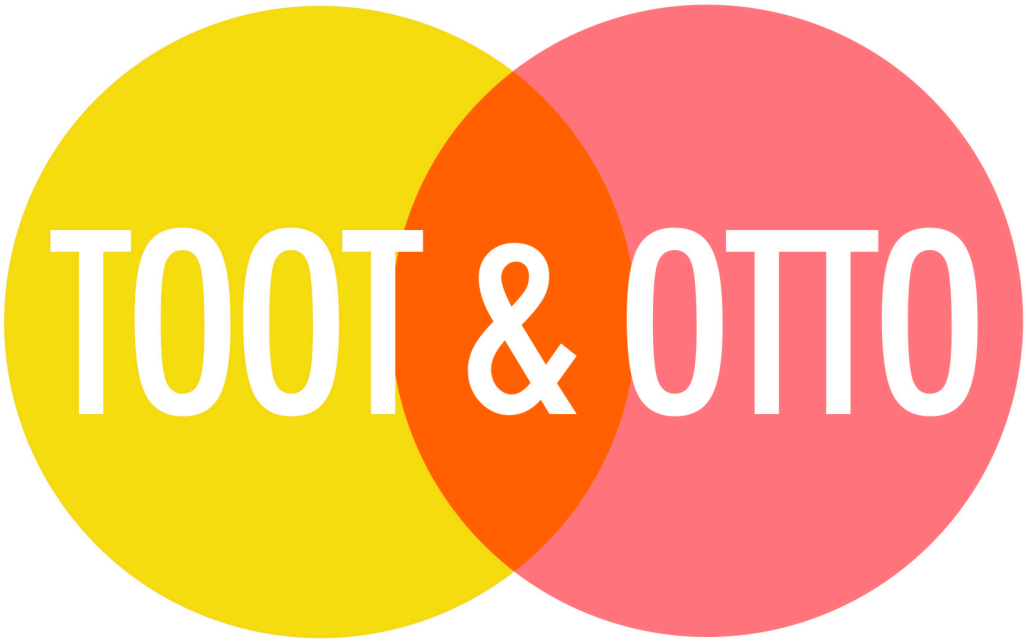

TOOT & OTTO

Major Innovations

The computerised opponent code was influenced by Monte Carlo simulations and heuristics. Similarly to the simulation phase of Monte Carlo search trees, the bot plays numerous random games by considering each potential next move. It then determines the best next move by summing the scores of all potential moves and choosing the highest score. The bot assigns a penalty to losses (-2) and ties (-1) while rewarding wins (+1). This way, difficulty can be controlled by changing the number of simulations the model makes. However since the simulations involve an opponent that also moves randomly, the bot may not always detect an opponent's potential win. Thus it was necessary to implement an override function for blocking an opponent's winning move and a check for ensuring the bot always takes a winning move if it exists. This section was done through brute forcing every next move to ensure it does not result in loss or that it would result in a win.

The code for the computerised opponent is also designed to suggest player moves. During initialization, you may specify the player the bot will make moves for. The bot will then determine the best move for the specified player. This allows for the bot to serve as both a computerised opponent and player, enhancing the versatility of the code.

In the case of an invalid url, the web application will not crash. If the user enters or tampers with the URL, it will bring the user to a blank page where the header is visible. The user can then redirect themselves to the page they would like to visit. This is an example of defensive programming as it is possible, although not likely, to arrive on a page that does not exist in this application.

Lastly, accessibility was a big concern for us in terms of this project. One of the major innovations in respect to the GUI is the customization of token colours for Connect Four. Using the colour picker, the user can colour both their and the bot's token to whatever colour they wish to select. This is an accessibility feature, as those with different types of colour blindness can choose the colours that they can see most easily. Additionally, there are 'X' and 'O' in the tokens which further increases accessibility for users whose colorblindness prevents them from benefiting from the customizable colour options. A byproduct of these innovations is that it may be more engaging to a younger audience, as children can customise their tokens to whatever colours they want.

Detailed Rationale for Augmented Decisions

---

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

A computer-based game offers several advantages over a printed board. This includes solo play through a computerised opponent, move suggestions, game customizability, accessibility, and status messages.

- A computerised opponent creates a learning environment for a player to improve at a game. It also allows for difficulty levels so that the player can adjust their gameplay to their level.
- Similar to a computerised opponent, a computerised game board can make move suggestions, further creating a learning environment or levelling the playing field.
- Since the computer does not require physical tokens, it allows for game customizability such as changing token colours.
- Players receive instant confirmation of game outcomes through printed status messages, eliminating the possibility of overlooking a victory or defeat.
- Using symbols on the tokens allow for accessibility for the color blind and as stated above, colour customization further accommodates people with different viewable colour ranges.

2. What is a computerised opponent? What are its objectives?

The goal of a computerised opponent is to always win and/or prevent an opponent's win. While it's not winning or blocking, its next goal is to find the best next move. We implemented these objectives through a brute force search for winning and blocking and a monte carlo search tree simulation for predicting the best next move (see major innovations for more information).

In the simulations, there are multiple randomised opponents who all play randomly until completion. Those randomised games are then aggregated and scored to find the best next move. Difficulty is determined by the number of randomised opponents. For instance, moves in Connect Four are determined by 5 opponents for each column for easy, 500 per column for medium, and 1000 per column for hard.

3. What design choices exist for the Interface components?

- Colour
    - Friendly and light colours since this webpage is catered towards children and families.
    - Buttons are brighter in colour to bring more attention to them as they are more important than other elements of the page.
    - Buttons and header tabs change colour when the mouse is hovered over to guide the user into these clickable areas.
- Font
    - Large and legible font for ease of reading.
    - Is a more 'round' font than jagged to match the theme of the webpage which is a family oriented game board.
- Using a slider for difficulty level
    - Intuitive UI design where the user can click on the area of difficulty, and the webpage gives proper feedback by moving the slider to that area.
- Using a switch for T and O tokens
    - A more aesthetically pleasing design than using radio buttons.
    - Placed on the right side of the grid and close to the grid so that the user doesn't need to navigate as much.
- Colour pickers
    - The default is white so that people that have trouble distinguishing texts on coloured backgrounds can still play the game with ease.
    - Allows more customization for the users and increases engagement.
- Rescalability
    - Components of the page are programmed with relative and absolute positions so that users on different screens will view the page similarly. This was checked by using different device dimensions using Devtools in the web browser.
- Labelling
    - The column numbers given for hints are one-indexed, as it is a more intuitive counting system for those that don't have a programming background.

4. What does exception handling mean in a GUI system?

In a GUI system, error handling means a couple of different things. Firstly, detection systems must be in place to detect the exception in the first place. This can manifest in the forms of try-catch blocks and event listeners. Afterwards, the exception that has been detected must be handled. In a

GUI system, this can be as simple as logging the error for debugging purposes to attempting to correct the exception if it is not unrecoverable. Assuming the system tries to recover from the exception, there are a couple ways it might attempt to nullify the exception. Firstly, the application state might be reset, hoping the exception does not occur again. Alternatively, the specific operation that caused the exception may be rerun, hoping the exception happened based on a set of circumstances that are outside of the program's control. Lastly, if there are any alternatives to the operation that caused the error the user may be prompted to try those. The last important thing a GUI system should do is inform the user an exception occurred if it could not recover from it, and print a helpful message so that the user may seek support if they deem it necessary.

5. Do we require a command-line interface for debugging purposes?

Yes, a CLI was required for the development of the project. The project is split into a front end and back end, and the CLI was used in the development of both areas. During back end development, the game board was printed out after every move to observe the game state and determine if the functionality was correct. It was used to simulate user behaviour and bot behaviour i.e. placing a token in a column. The common line interface was necessary as it ensured that the base functionality was handling user actions properly and was able to be tested in a simple manner. This is preferable as opposed to testing it with front end components which would be harder to debug due to the increased possible points of failure. Afterwards, logging statements are outputted onto the terminal to ensure that the data transfer using the APIs was successful.

List of Known Errors and Limitations

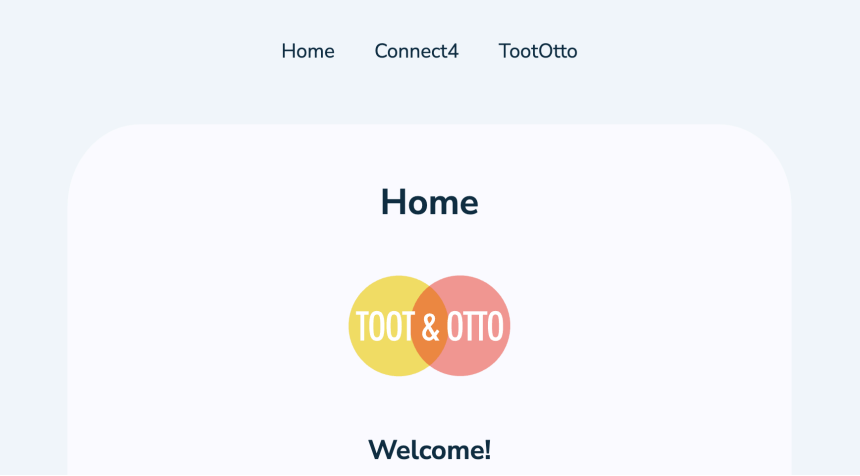
Following the move suggestions won't always result in the player's win, but will result in a win majority of the time.

The time between the bot and the player in Toot Otto does not seem consistent as there was a noticeable simulation time for the hardest difficulty level. The delay time set for Toot Otto is the same across all three levels, therefore the time between the two players is much shorter in the easy level than the hard level.

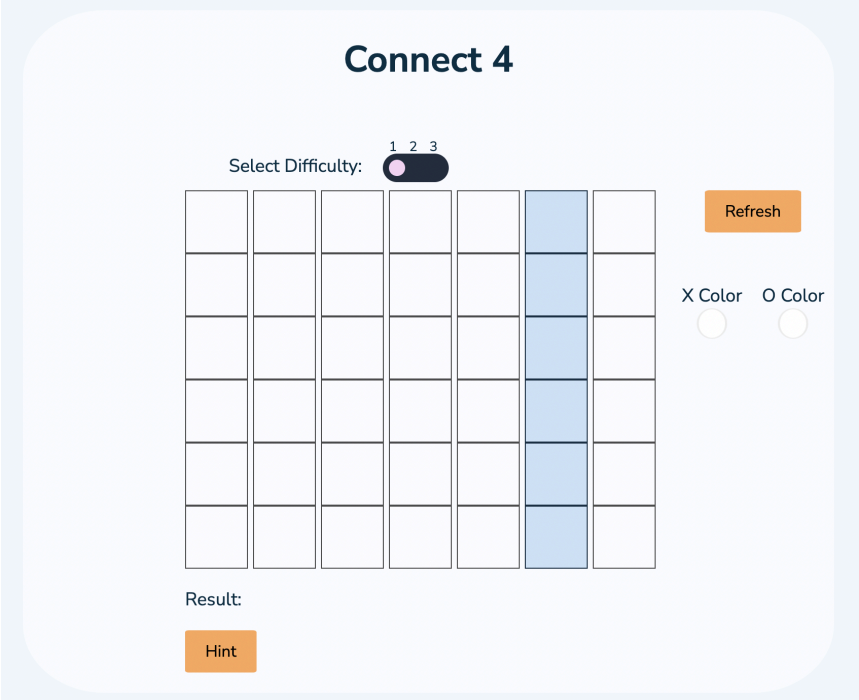
User Manual

This manual assumes that a Rust toolchain has already been installed on the device.

1. Install and configure Rust to nightly version by running these commands in the terminal within the yew-app directory to run the Rocket server:
   ```
   rustup toolchain install nightly
   rustup default nightly
   rustup override set nightly
   ```
2. Execute the following command to run the yew application:
   ```
   rustup target add wasm32 - unknown unknown --toolchain nightly
   ```
3. Install trunk:
   ```
   cargo install trunk
   ```
4. Open a terminal in the ./frontend directory
   a. execute `trunk build`
5. Open a second terminal to start up the rocket server
   a. Ensure that you are in the ./backend directory
   b. execute `cargo run`
   c. Click on the http://localhost:8000 link in the terminal to see the webpage
   d. The second terminal must remain running throughout the process
6. The link will open the home page for the application. The home page contains the game instructions. Follow the links at the top to navigate to the different game pages.

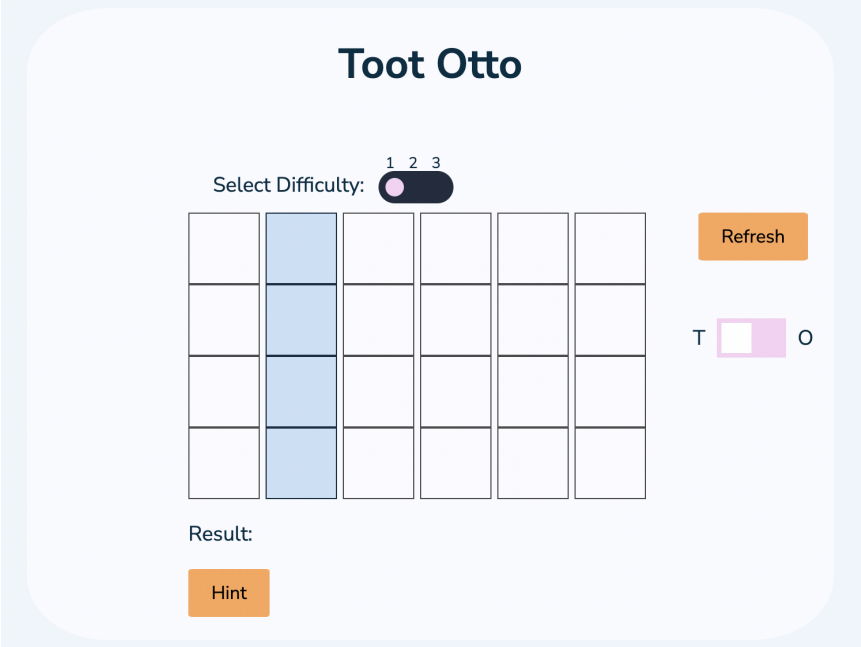


Connect Four: The goal is to get 4 tokens in sequence in any horizontal, vertical, or diagonal. The player is assigned the X token while the computerised opponent's token is O.



- The "Select Difficulty" toggle switches from easy (1) to medium (2) to hard (3). Toggling a different difficulty will reset the game.
- The grid shows the game state with the hovered column highlighted. Clicking on a column will drop the player's token to the lowest unoccupied row.
- The "Refresh" button clears game state, including token colours.
- "X Color" and "O Color" are colour pickers to allow a player to customise token colour.
- The "Result:" text will display a message if either player wins or if a tie occurs.
- The "Hint" button will display a suggested move for the player when clicked.
    - The hint number signifies a column number from 1 to 7, with 1 being the leftmost column and 7 being the rightmost.
    - There will be no hint displayed if the game has already ended.

Toot Otto: The player's goal is to form the word "TOOT" while preventing the computerised opponent from forming the word "OTTO".



- The "Select Difficulty" toggle switches from easy (1) to medium (2) to hard (3). Toggling a different difficulty will reset the game.
- The grid shows the game state with the hovered column highlighted. Clicking on a column will drop the player's token to the lowest unoccupied row.
- The "Refresh" button clears game state.
- "T"/"O" switch toggles the player's token letter between T and O.
- The "Result:" text will display a message if either player wins or if a tie occurs.
- The "Hint" button will display a suggested move for the player.
    - The hint shows a column number (from 1 to 6, with 1 being the leftmost column and 6 being the rightmost) and a token letter T or O.
    - There will be no hint displayed if the game has already ended.