ECE 421 Project 3: Let's Make a Rusty Connect-4

You are a freelancer. Recently, you have noticed a job posting at one of the well-known global freelancing platforms that is looking for a 'UI/UX Designer for Board Games'. The client has a 5-star rating and the pay is pretty good. So you have decided to give it a go. The job description looks something like this:

We are working on an initiative to computerize popular board games that we all grew up playing.

The first game we are looking at is **Connect-4**. Traditionally, many families have spent their evenings playing this game; but we want to allow players to test their skill against a computer opponent.

We expect the application to have:

- Computerized opponent(s) for the players with multiple difficulty levels.
- A GUI that can handle the color vision deficiency (a.k.a. Color blindness) in users.
- The game follows the rules of the real version of the game.
- Possible additional features like variable board sizes, disc colors, or next move suggestions

To make things a little more interesting, we want the traditional Connect-4 to be accompanied by a relatively new board game **TOOT and OTTO**. This new game has disks that have the letters **T** or **O** printed on them. Here, one player aims to form the word TOOT, and the other tries to form OTTO. Also, the board size for this game may be different from the board used in Connect-4. In the application, we want users to be able to:

- Seamlessly switch between the two games
- Play TOOT and OTTO with a computerized opponent with multiple difficulty levels

Our company expects to produce a number of games upon this basic theme, and hence, you are invited to design your game to allow it to be efficiently and effectively extended. The company will actively pay freelancers for extensions of this basic theme!

We have previously produced an open-source version: https://github.com/thinking-fun/Connect4-with-TootandOtto. The open-source version provides functionality beyond what we have described above. However, it is implemented using the MEAN stack. We are transitioning it to Rust (and web Assembly). The new version must use this latest technology.

Please note: the author of the MEAN stack solution has left the company, we only have the Github repository.

We are a company heavily invested in redesigning computerized board games. People who have strengths and interests in creative game/simulation projects using Rust, please apply!

Soon after you applied for the job, you got a response from the client accepting your proposal. The client has also mentioned a bonus amount in case you produce additional functionality to the application on top of what has been asked.

Now you have around a month's time to create the application and send it to the client. Good Luck!

Along with the source code for the application, the client also needs documentation that elaborates on your design choices. In this design document, please ponder (and answer) the following questions:

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

3) What design choices exist for the Interface components?
   - Color? Font? Dimensions of Windows? Rescale-ability? Scroll Bars? ….
4) What does exception handling mean in a GUI system?
5) Do we require a command-line interface for debugging purposes????? (The answer is yes by the way – please explain why)
6) What are Model-View-Controller and Model-View-Viewmodel? Are either applicable to your design? Justify your answer.

Deadlines & Submission Instructions

As per your freelancing contract, the client needs the application to be ready and received by **April 12th, 2023 @ 11:59p.m M.S.T.**; this is a hard, absolutely unmovable deadline. Your performance will be measured on:

- The accuracy of the delivered code with the requirements above.
- The performance (efficiency, usability, …) of the solution.
- Bonus credits will be given for new additional features.

By the deadline, you must deliver:

- A copy of the source code for the application.
- A design document outlining:
  - Major innovations – Additional to the project specification.
  - A detailed rationale for your augmented decisions with regard to the above design questions.
  - A list of known errors, faults, defects, missing functionality, etc. Telling us about your system's limitations will score better than letting us find them!
  - A detailed description of any remaining MEAN stack code.
  - A user manual (how to start the system, environment requirements, configuration requirements, etc.…
  - A 2-minute video highlighting the new system – marketing is everything ☺

You must also demo the code live to the client between **April 17th, 2022** and **April 21th, 2023** inclusive.

Please hand in all components by submitting via eclass to the group account, and hence all sub-components, by definition, must be machine-readable. Also, file types should be only .pdf, .mp4, or zipped files for your Rust projects.

Good luck and happy coding!