Dots and Boxes Web-Based GUI, Fall 2019

Goal
While the Gamescrafters group already has dozens of games created with GUI’s in Gamesman Classic, Fall 2019 is the first semester where the games are now being migrated over to the Gamescrafters website, and thus each game needs to have its own new, web-based GUI implementation. The goal of this project was to create such a GUI for Dots and Boxes.

Members
Grace Kull, Frank Cuoco, Lucas Huang, and Danyal Shahroz

Responsibilities included fully comprehending how the current framework for GamesmanUni works, navigating the GitHub (cloning, pushing, and pulling), learning brand new programming languages (JavaScript, CSS, HTML, SVG, etc.), coding the GUI frontend and connecting it to the interactions of the backend, keeping the design consistent with the Gamescrafter’s format (hints, color schemes, etc.), and handling all user inputs such as clicks and hovers.

Overview
For Dots and Boxes, the goal of the game is to create as many boxes as you can by placing down one line at a time across two dots. The player on each turn is allowed to place down a single line, and if the player puts down a line that completes a box, they capture that box. The player with the most captured boxes when there are no more moves left is the winner. In the Gamescrafters’ implementation, the board is a 3x3 set of dots, which allows for 12 lines to be put down and 4 boxes to be created in a playthrough.

When designing the GUI, we wanted to mirror the simplicity of the gameplay, and only show the user what they needed to see. We first put down a 3x3 grid of light grey circles to signify the dots. Next, we programmed each line to be the length of exactly two dots, or half the board, and translated/rotated each line object as necessary. Then, our framework included having all of the lines showing their respective “hint” color, with each line having a separate “clickable” that would change the color of each line selected to the respective player color. Finally, if a player completed a box, a circle in that player’s color would appear in the middle of the box to signify that it was captured by them.
Problems and Bugs

For most of the frameworks and technologies we used, it was our first time using them. For example, none of us had Yarn installed on our laptops, and figuring out the proper way to install it took some trial and error. However, with Shein’s help, we were able to properly get it on our laptops, as he has an excellent understanding of more backend software. Additionally, Vue was a language most of us had never used before, but it was similar enough to other languages we had seen such as JavaScript, so we eventually we able to understand it.

As far as implementing the GUI, the biggest obstacle we had to overcome was figuring out how to connect each of the parts to the respective backend server game data. Each line represents one out of 12 moves, and converting the position binary string to a game board took a lot of work, especially since we also had to figure out how to use most of the languages behind it. The crux of this issue was figuring out how to loop through each move and update the board every time with the player’s choice. It was difficult to truly understand the backend implementation of the game, and it took much debugging with Shein to figure out how things were being represented in the backend code so that we could properly reference it in the frontend.

In terms of adding style components to the board, we learned to make use of SVG’s components such as circle and line objects. While making each component look how we wanted for the Dots and Boxes game board, it took some trial and error to make sure all the components were in the proper place and interacted with each other how we intended.

Final Product and Future Development

The board design and implementation is finished. The design and drawing for each line, token, and completed box indicator and how it will be visualized is also complete. The only thing left to do is to figure out how to reliably loop through the moves of the string representation of the board from the backend so that hints and clickables can be implemented as intended.

The pseudocode for this should be effective once this one issue and solved. Like many other front end teams, our starting point for this board was TTT, so this logic is built off that because TTT and DnB have very similar game logic.

```python
for line_pos in moves:
    if line_pos == '0':
        return hint_line at board[line_pos]
    else if line_pos == '1':
        return curr_player_line at board[line_pos]
```
else if line_pos == 'x':
    return player1_completebox at board[line_pos]
else if line_pos == 'o':
    return player2_completebox at board[line_pos]