**Checkers Game Final Report**

Cleveland State University

CIS434 - Software Engineering

04/28/2022

Group 12

Connor Van Etten

Tytionna Williams

Esmeralda Xhyliu

**Individual Contributions Breakdown**

**Connor Van Etten**

·         HTML elements (table, pieces)

·         CSS visuals for game

·         Creation of Player Versus Computer Backend

·         GitHub Management

·         Assistance in Debugging Player versus Player

·         Assistance in CSS elements

**Tytionna Williams**

·         HUD and Visuals of Pages

·         JavaScript Framework

·         Creation of Player Versus Player Backend

·         JavaScript and HTML assistance

·         Assistance in Debugging

·         Assistance in Logical Theories

**Esmeralda Xhyliu**

·         Creation of Rules Pages

·         Oversaw Meetings and Communication

·         Helped Clean and Tidy Code

·         Pivot in planning and Timeline

·         Main Tester

·         Assistance in PvC mode

·         Reports and Presentations

**Abstract**

The purpose of this project was to create a checkers game. With the classic game of checkers involving two separate players playing on an 8x8 board we set out to create a virtual version. Our project was created with HTML, CSS, and JavaScript, allowing the user to play against another player or against a computer. HTML served as the basis of the board, and piece to help give the Web application a framework. Then CSS was added to help with our visual changes, and JavaScript handled the background and logic of the game. With three team members working on this project, used the GitHub version control platform to ensure that changes were going to be tracked and any edits can be managed, in that manner we were able to revert changes to their previous state and see at what point a change was made that broke the project so that we can fix it. This also allowed for better code reviews and enhanced teamwork. One advantage of using GitHub is the built-in testing, because of that any commits that were pushed were automatically tested so that we know right away if a commit will break the project.

**Objectives**

         For this project, the objective was to use HTML, CSS, and JavaScript to produce a game of Checkers that allows two users to play against one another, and/or against a “computer opponent.” Creating a virtual remake of the board game classic created many goals that we set out to achieve. The objective was to have a player versus player mode that would react just as it would if you were playing in person but give it incorporated features such as guided moves to enhance the experience. While playing solo, allowing the computer to play with a level of intelligence when responding to a user’s movement, or the board’s current state was a pivotal part of our designing. This would give our “AI” a way to make the best possible move given the supplied logic. When interacting with our implementation, players were given the option to restart the game or quit it, which took the players back to the menu page where they could select another option to either play in a different mode, to check the rules, or to quit the game altogether. Due to limited experience as the semester progressed, we realized we had spent extra time in the planning stage and had fallen behind our initial goals. If the development process proceeded as planned, ideas for multiple different features such as difficulty modes for AI, a timer, and hosting our web app were on the table.

**Project Description**

The development process started with the creation of an 8x8 board, which as well as the rest of the project, was designed with HTML, Javascript and CSS. Following that, the menu page was designed to have four options: Player versus Player, Player versus Computer, Checkers Rule Book and an option to exit the game. Exiting the game will immediately close the current window of the program. Both the Player versus Player, and Player versus Computer options, when clicked, open two respective HTML pages which display identical boards and pieces. Both modes will share the same game rules but will operate with a slight difference. During the Player versus Computer mode, it is according to the rules that the user goes first as they will be the red piece. On the two-player option, the red piece goes first which is the bottom half of the board.

Initially, we wanted to design the pieces to visually drag across the board, but we learned that moving the pieces in this way would limit other features that we valued as more important. The feature we eagerly wanted to work on was the availability to highlight available movements using an event listener that changes the HTML page with the getAvailableSpaces function. It searches for the checker pieces’ available movements in the four directions of top left, top right, bottom left, and bottom right. It then gives the square a temporary yellow circle for non-AI players, showing it is open and available. This fun visual is provided by the function selectionIdentifier. After a piece is clicked and an available square, the function Move makes the move by adding or subtracting 7,9, 14, and 18. Because the board is essentially an array adding and subtracting moves up or down the rows of the board. For both versions of the game, we implemented a visual indication of turn using the function changePlayer, which, using an if-else statement, switches the players’ colors above from light gray to red or white.

The single player (Player versus Computer) version was developed once the player versus player game logic was completed, as it is largely based off of the same functionality. The difference is that we use a random number generator to check the pieces’ available moves and moves with a jump will rank highest most often. It is a relatively simple AI, but it does go for a jump most of the time allowing the user a fun and competitive experience.

Diagram

Description automatically generated

Figure 1: Logic Flow Chart

  In the beginning, we wanted so much more, but we realized how much harder it was to code all of our desires. Features such as choosing the color of the pieces and allowing to pick if AI or user goes first were dropped as the creation of the AI and the highlight available movement features took precedence. A guided logic flow chart of the program can be seen in Figure 1.

**Professional Awareness**

We were successful in staying professional over the course of this project through various means. Both our audience and our individual backgrounds were put in consideration during the development process. We strived to develop a usable and enjoyable product that a customer would be happy to utilize, but at the same time implementing a UI that was both straightforward and simple. Keeping an open and respectful mind, was a pivotal part of working within a group setting. Cultivating a safe space for collaboration was one of our better skills as everyone’s opinions and voices were able to be heard. By respecting and valuing each other, the group was able to become closer and more efficient at completing their tasks. Now that we’ve finished developing the initial project, we will be able to continue lifelong learning by expanding upon the finished product. By documenting our work, we can help teach others how to develop a Checkers game using HTML, CSS, and JavaScript. Other ways to be professional include being formal and direct when communicating, being competent and confident in the field one is expected to work in and promoting team morale. Using and learning these techniques keep us not only productive throughout the semester but professional as well.

**Project Timeline**

Chart, timeline

Description automatically generated

Figure 2: Term Project Gantt Chart

There were a few difficulties we faced while working on this term project. One of the biggest problems was finding the time to work on the project together as all of the members of the group have full-time or part-time jobs. However, deciding the work seemed natural as each member had their individual talents. To solve these time issues, we decided to meet on the weekends which was a big sacrifice but absolutely needed in order to finish the project on time. During those days, we utilized video calling applications such as Discord and Zoom to easily share screens and explain the thought process of our coding when comments could not do justice. To maximize time, we watched Youtube videos on coding and language syntax we did not understand. The part of the project that took the longest was the AI creation because the smallest change could affect the white pieces’ functionality, so there were many times we got it working just to have it crash again when adjusting something else in the code like the scoreboard. Alongside all these solutions, we were also involved in pair programming in pairs or with all three of us, to ensure each line was written concisely and securely, and this also helped eliminate many of our issues. A picture of the gantt chart is shared in Figure 2.

**Conclusion**

Overall, throughout this process we achieved the main goals that our group initially declared. Our implementation of checkers included the needed modes and features that were required for our finished product. Upon delivery our game included a PvP mode, a PvC mode as well as a functional homepage and rule book in our complemented implementation. In addition to types of modes, our extra features were able to be added such as guides and a scoreboard. The game reflects the actual layout of the Checkers, including the board’s shape, colors, and pieces with their proper movements. That is, we made our version of the game as like the physical board game as possible, while being able to integrate the abilities of modern computers. Our project was created over the course of 2 months, and there were no legal, security or social issues.

**References**

* Anderson, Andrew. *Checkers: Containing Complete Instructions and Rules for Playing Checkers or Draughts*. Fitzgerald pub. co. *EBSCOhost*,<https://search-ebscohost-com.proxy.ulib.csuohio.edu/login.aspx?direct=true&db=cat02507a&AN=ohiolink.b14294501&site=eds-live&scope=site>.
* Hopper, Millard. *Win at checkers*. New York: Dover Publications, 1956.
* L. Youru, L. Xueping, W. Yajie and X. Yuting, "A systematic research about the computer game of checkers based on the decision-making of AI," *The 27th Chinese Control and Decision Conference (2015 CCDC)*, 2015, pp. 6437-6442, DOI: 10.1109/CCDC.2015.7161978.
* MacDowell, Susan. “What Moves Can a King Make in Checkers?” *Our Pastimes*, 10 Jan. 2019, https://ourpastimes.com/moves-can-king-make-checkers-8068380.html.
* Samuel, A. L. *Some Studies in Machine Learning Using the Game of Checkers, II-Recent Progress*. Pergamon Press, 1969. *EBSCOhost,*<https://search-ebscohost-com.proxy.ulib.csuohio.edu/login.aspx?direct=true&db=cat02507a&AN=ohiolink.b17408675&site=eds-live&scope=site>.
* Schaeffer, Jonathan and Lake, Robert. “Solving the Game of Checkers.” *MSRI Publications*, vol. 29, no. 1, 1996, pp. 119-133.<http://library.msri.org/books/Book29/files/schaeffer.pdf>
* Calin, *A Game of Checker*, A Game of Checkers, Feb. 2019. Accessed on: March. 25, 2022. [Online]. Available: <https://codepen.io/calincojo/pen/wBQqYm>
* G. Carmelo, *Simple Checkers Game In Javascript With Source Code*, Source Code & Projects, Sept. 2018. Accessed on: April. 2, 2022. [Online]. Available: <https://code-projects.org/simple-checkers-game-in-javascript-with-source-code/>
* T. Todorescu, *I’ve Coded Chess.com with Javascript*, Youtube.com, Mar. 2021. Accessed on: April. 11, 2022. [Online]. Available: https://www.youtube.com/watch?v=aqNvMSuLuE0&t=1598s&ab\_channel=TudorTodorescu