Project Title: Web Coding

Kanishk Sharma

June 14, 2023

Description

The purpose of this code is to create a Minesweeper Cricket game using HTML, CSS, and JavaScript. The code consists of three main parts: the HTML code, the CSS code, and the JavaScript code.

The HTML code defines the structure of the game interface. It includes elements such as a grid size select dropdown, a difficulty level select dropdown, a game board, a score card, a lives card, a start button, and links to the rules and owner pages. It also includes an audio element for playing a win sound.

The CSS code defines the styling for various elements of the game interface. It sets the background color and image, defines the layout of the game board, sets the styles for different cards and buttons, and positions the elements on the page.

The JavaScript code handles the game logic. It defines variables to store the number of fielders, number of no balls, the run values, the fielder locations, the no ball locations, the score, and the lives. It also defines arrays for the difficulty levels and grid sizes. The code includes functions to generate random fielder and no ball locations, check if a block contains a fielder or a no ball, handle block clicking, reveal all unclicked grids, get the image for a run value, update the score and lives, end the game, play a win audio, reset the game, and initialize the game board.

Overall, this code creates a Minesweeper Cricket game where the player clicks on blocks on the game board to reveal runs, fielders, and no balls. The player earns points for runs and loses lives for fielders. The game ends when the player reaches the score limit for the selected difficulty level or runs out of lives.

Implementation Details

HTML

HTML is used to structure the game interface and define the elements on the page. The code includes elements such as buttons, text, and the game board. HTML is the standard markup language for creating web pages and provides the basic structure of the game's user interface.

CSS

CSS is used to style the game interface and make it visually appealing. The code sets the background color and image, applies styles to various elements such as buttons, score cards, and the game board, and positions them on the screen. CSS helps in creating a visually consistent and attractive layout for the game.

JavaScript

JavaScript is used for the game's logic and interactivity. The code handles various functionalities, such as generating random locations for fielders and no balls, checking block contents, updating scores and lives, and responding to user clicks. JavaScript provides the necessary scripting capabilities to make the game interactive and dynamic.

No external frameworks or libraries

The implementation of the game does not seem to rely on any external frameworks or libraries. It appears to be a pure JavaScript implementation using the core functionalities provided by the language itself. This keeps the code lightweight and avoids unnecessary dependencies.

Random generation of fielders and no balls

The code uses a random generation algorithm to determine the positions of fielders and no balls on the game board. This adds an element of unpredictability to each game, making it challenging and replayable.

Dynamic block clicking

The Click() function is triggered when a block is clicked, and it dynamically updates the score and lives based on the block's contents. This allows the player to interact with the game and receive immediate feedback on their actions.

Responsive layout

Although the code you provided does not explicitly show responsiveness, the use of HTML and CSS allows for potential responsiveness to different screen sizes and devices. By leveraging responsive design techniques, the game could adapt its layout and elements to provide an optimal experience across various devices.

Audio feedback

The code includes a playWinAudio() function that plays an audio file when the player wins the game. This adds a sound element to the game, providing auditory feedback to the player's achievements.

Code Files

1. **main.html**

   Purpose: The main entry point for the Minesweeper Cricket game's user interface. Functionality: Defines the overall structure of the game interface. Renders the game board, allowing the player to interact with the game. Displays buttons and controls for starting and resetting the game. Includes JavaScript code for handling user interactions and game logic. References CSS stylesheets for visual appearance and layout.

2. **styles.css**

   Purpose: Defines the visual appearance and layout of the game interface. Functionality: Specifies colors, fonts, dimensions, and positioning of game elements. Styles the game board, buttons, and other visual components. Ensures a consistent and appealing visual design for the game.

3. **script.js**

   Purpose: Implements the game logic and handles user interactions. Functionality: Controls the game flow, including starting, resetting, and ending the game. Manages the game board, including generating mines and updating cell states. Handles user clicks and reveals cells, updating the score and game state. Implements game rules and win/loss conditions. Updates the UI based on game events and user interactions.

4. **rules.html**

   Purpose: Provides information and instructions about the game rules. Functionality: Displays the rules of the Minesweeper Cricket game to the player. Provides guidance on how to play the game and understand its mechanics. Supports the player in learning and following the game's rules.

5. **rulestyle.css**

   Purpose: Defines the visual appearance and layout for the rules page. Functionality: Specifies styles for headings, paragraphs, and other text elements. Formats and structures the content of the rules page for readability. Ensures a visually appealing presentation of the game rules.

6. **owner.html**

   Purpose: Displays information about the owner/developer of the game. Functionality: Shows details about the creator of the Minesweeper Cricket game. Provides contact information or additional information about the game's development. May include acknowledgments or credits to contributors.

7. **owner.css**

   Purpose: Defines the visual appearance and layout for the owner page. Functionality: Specifies styles for headings, paragraphs, and other text elements. Formats and structures the content of the owner page for readability. Ensures a visually appealing presentation of the owner information.

Report Files

1. **report.pdf**

   Purpose: Comprehensive documentation of the project. Functionality: Describes project details, implementation, customization, instructions to compile and run the code, includes screenshots, references external sources, and provides recommendations.

2. **report.tex**

   Purpose: LaTeX source file to generate the report.pdf. Functionality: Formats the report document, defines structure, layout, manages cross-references, citations, integrates visual elements, and allows customization.

3. **report.bib**

   Purpose: BibTeX file with bibliographic references. Functionality: Stores references for cited works, enables proper citation, attribution, and acknowledgment in the report.tex file.

Customizations

1. Dynamic Grid Size:

   Added the option to choose between 6x6 and 7x7 grid sizes. Provides different levels of challenge and gameplay experience.

2. No Ball Feature:

   Introduced a special item that grants players extra lives. Increases player longevity and adds a strategic element to the game.

3. Difficulty Levels:

   Implemented three difficulty levels: easy, medium, and tough. Each level offers a different level of challenge to cater to players with varying skills.

4. Variable Runs:

   Randomized the number of runs in each round (1, 2, 4, or 6). Adds unpredictability and keeps the gameplay fresh and exciting.

5. Restarting the Game:

   Enabled players to restart the game at any point. Provides flexibility and allows players to improve their score or start over if desired.

6. Styling Customizations:

   Adjusted colors, fonts, and sizes to enhance the visual appearance. Achieved a desired aesthetic appeal and improved user experience.

7. Hyperlinks for Rules and Information:

   Added hyperlinks to access the game rules and information about the developer. Facilitates easy access to important information within the game interface.

Screenshots

[Insert screenshots here]

Conclusion

The project involved creating a customized game with dynamic grid sizes, a no ball feature, difficulty levels, variable runs, and personalized styling. The achievements are implementing these features successfully. The challenges faced were related to game balancing, responsiveness, and styling. Future improvements could include adding power-ups, multiplayer functionality, a high score system, sound effects, and LeaderBoard.