CS104 Course Project
Minesweeper Cricket

Sanskar Shaurya

Contents

1 Introduction

In this project, I have created an exciting game that combines elements of Minesweeper and Cricket. The objective of the game is to click on blocks within a grid and accumulate runs while avoiding fielders. Let's dive into the details of the game and explore the additional customizations that have been implemented. The game begins with a grid of blocks, where each block represents a potential score or a fielder. The player's goal is to reveal blocks and accumulate as many runs as possible without encountering a fielder.

2 Customisations

2.1 Two Game Modes

I have given the player the option to either play this game alone or with a friend, when the game starts the player has to choose which game mode he wants to play. The screen corresponding to this is:

The mechanics of the two game modes are:

- In the Single Player mode, the player has to score as many runs as he can before getting out
- In the Double Player mode, two players play this game turn-wise. The HTML asks for the names of both the players first and then starts the game. When the first player gets out, the second player continues playing till he gets out too. The player with the highest number of runs wins.

2.2 Variable Grid Size

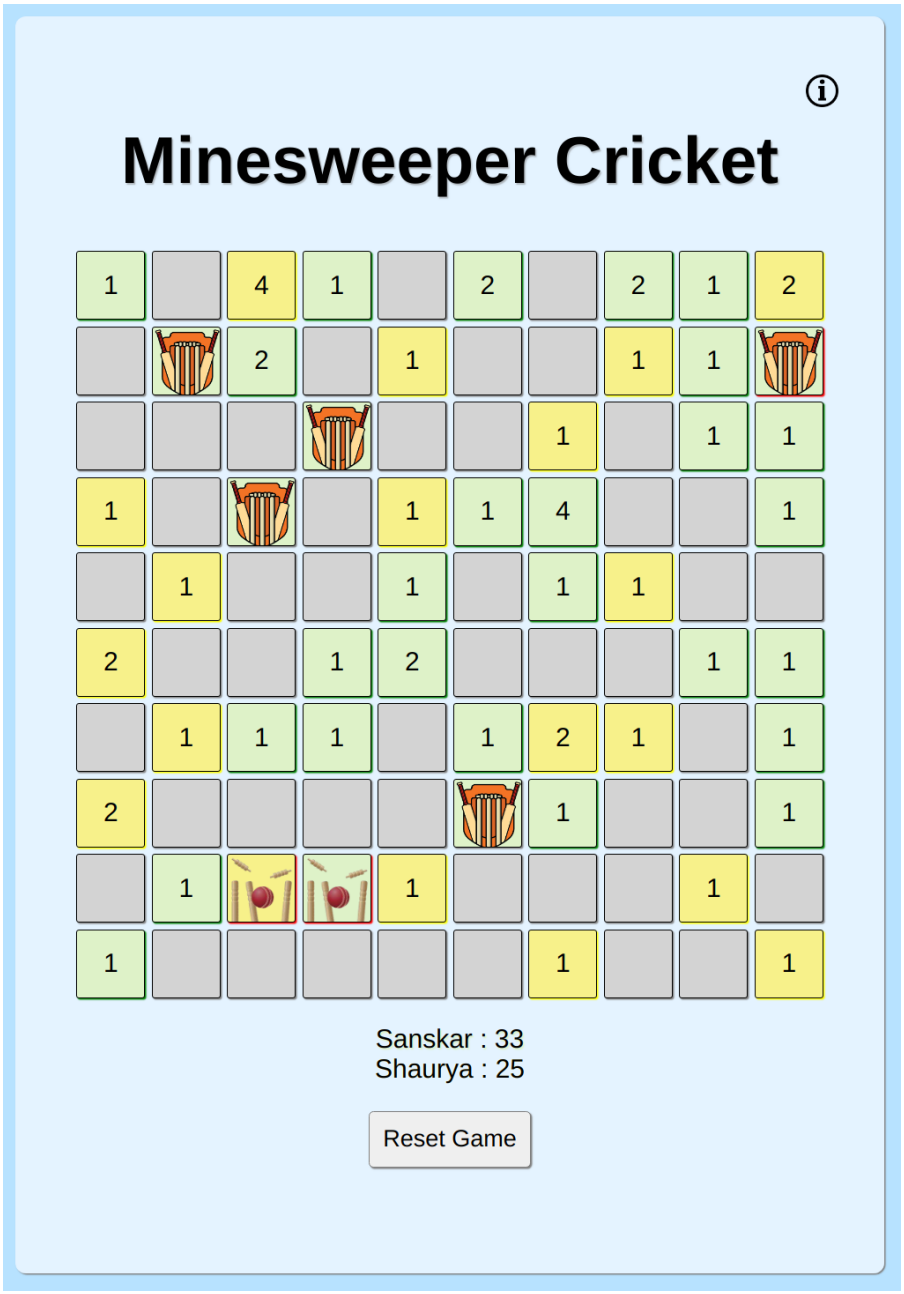
Before starting the game in any of the two above modes, the player has to choose from one of the 5 different available grid sizes, ranging from 6 x 6 to 10 x 10. The game will then produce a grid of the appropriate dimension with 11 fielders spread randomly throughout the "field" (grid).

2.3 Variable Number of Runs

Each block has a probability of getting one of the following runs allocated to it: 1, 2, 4, 6, or the shield power-up. I have modified these probabilities such that the chance of getting a boundary-score is less than that of a single or a double.

2.4 Shield Power Up

Each block also has a chance of containing the shield power-up. This power up when activated, will allow the player to not get out in the next three turns even if they click on a block with a fielder.



An example of the game being played between two players showcasing some of the customisations

3 Stylesheet

3.1 Colors

I have used a combination of blue, light blue, gray, and white to make the webpage visually appealing. The elements also have a shadow which makes them pop on the screen.

3.2 Info and Back Button

I have also added an info button which, when clicked, will open up an information box that takes up the entire screen. There also is a back button that will take the player back to the previous screen. (Note: This button is not available when the game gets started)

3.3 Grid Style [1]

The boxes, when clicked, also get colored corresponding to the player who is playing. Green for Player-1 and Yellow for Player-2 in double-player mode and just Green in single-player mode. When hovering on a block, the block gets highlighted and when it gets clicked, it undergoes a scale-up animation[2, 3] which makes the game more visually appealing. I have also added some border-radius to the grid-blocks so that the blocks don't look very sharp.

3.4 Font Used

I have used the "Gill Sans" font for this website. The color of the font is either black or white, depending on the background on which the text is written.

4 The JavaScript [4]

Brief explanations of some of the functions I used:

- **single() :** This function sets the *playerChoice* variable to 1, which makes the game proceed with the single player mode. This function is called when the player clicks on the Single Player button.
- **double() :** This function sets the *playerChoice* variable to 2, which makes the game proceed with the double player mode. This function is called when the player clicks on the Double Player button.
- **backButton.addEventListener('click') :** This function is for the back button, makes the screen go to the previous one.[5]
- **showGameGrid(gridSize) :** This function is executed when the player clicks on one of the grid-size options, it sets the variable *size* equal to the value clicked by the user and proceeds onto the next screen corresponding to the game-mode selected.
- **startSingleGame(gridSize) :** This is the main function of the single-player mode, it contains of all the sub-functions required for this mode to work properly. It generates blocks and gives them scores or shield based on some probability. It has some sub-functions which are listed below:
  - **handleBlockClick(event) :** This function is the main working part of the single-player mode. This function is called whenever the player clicks on any of the grid-blocks. Based on what the block contains, this function gives the player some runs, gets them a shield or gets them out and calls the *endGame()* function.

  - **endGame(score) :** This function is called when the player clicks on a fielder block, It gives an alert giving the final score of the player and makes the reset button visible.

- **nameForm.addEventListener('submit') :** This function is for when the user submits the name form in the double player mode. It sets the player-1 and player-2 name corresponding to the values written in the text-fields.

- **startDoubleGame(gridSize) :** This is the main function of the double-player mode, it contains of all the sub-functions required for this mode to work properly. It generates blocks and gives them scores or shield based on some probability. It has some sub-functions which are listed below:

  - **handleBlockClick(event) :** This function handles the clicking of blocks for the double-player-mode. This checks who the current player is and applies the game logic to that player. It calls the *togglePlayer()* function to the change the current player and the *endGame()* function when both the players get out.
  - **togglePlayer() :** This is a simple function which just checks who the current player is and then swaps to the next player.
  - **endGame() :** This function ends the game when both of the players get out and makes the reset button visible.

- **resetButton.addEventListener('click') :** This is the function for the reset button, this makes every score go back to the default value of 0. This also clears the grid and makes it empty.

- **generateRandomPositions(gridSize, count) :** This is the function which generates random fielder positions. It returns a 2D array which has the row and column indices of the fielder positions as elements and there are a total of *count* elements which we have used as 11.

5 Source Code

5.1 HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>Minesweeper Cricket</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
  </head>
  <body>
    <div id="everything">
      <span id="open"
        ><img src="./images/info.png" width="30px" height="30px" id="info-button"
      /></span>

      <div id="info-window" class="info-window">
        <h3>Game Information</h3>
        <hr />
        <p style="margin-top: 20px">
          Welcome to this crazy combination of Cricket and Minesweeper, fielders
          are spread randomly on the field and you need to pick blocks such that
          they don't contain fielders.
        </p>
        <p>
          Different blocks have different scores, clicking on one may give you a
          boundary score or just a single. Clicking on blocks where fielders are
          present will make the game stop.
        </p>
        <p>
          There are two game options present, you can either play alone or play
          with a friend in the Double Player mode. The double player mode
          introduces a turn based system where each player will pick a block
          each turn. The one with the highest score when both the players get
          out will win the game!
        </p>
        <p>
          You can also choose the grid size of the game, there are 5 options
          available, from 6x6 to 10x10.
        </p>
        <p>
          There are some hidden shields in the field, which will protect you if
          find a fielder in the next 3 turns, good luck on locating them!
        </p>
        <span id="cl"><button id="close-button">X</button></span>
      </div>

      <div id="overlay" class="overlay"></div>
      <h1>Minesweeper Cricket</h1>
      <div id="start-window">
        <button class="startbtn" id="single" onclick="single()">
          Single Player
        </button>
        <button class="startbtn" id="double" onclick="double()">
          Double Player
        </button>
      </div>
      <img src="./images/back.png" id="back-button" style="display: none" />
      <div id="name-prompt" style="display: none">
        <form id="player-names-form">
          <label for="player1-name">Player 1 Name:</label>
          <input type="text" id="player1-name" required />
          <br />
          <label for="player2-name">Player 2 Name:</label>
          <input type="text" id="player2-name" required />
```

```
          <br />
          <button type="submit" id="start">Start Game</button>
        </form>
      </div>
      <div id="grid-size-form" style="display: none">
        <p style="margin: 10px">Select Grid Size:</p>
        <div class="grid-size-option" onclick="showGameGrid(6)">6x6</div>
        <div class="grid-size-option" onclick="showGameGrid(7)">7x7</div>
        <div class="grid-size-option" onclick="showGameGrid(8)">8x8</div>
        <div class="grid-size-option" onclick="showGameGrid(9)">9x9</div>
        <div class="grid-size-option" onclick="showGameGrid(10)">10x10</div>
      </div>
      <div id="game-container" style="display: none">
        <div id="game-grid"></div>
        <div id="score">Score: <span id="run-score">0</span></div>
        <div id="double-score">
          <span id="player1-score" style="display: inline-block">0</span><br />
          <span id="player2-score" style="display: inline-block">0</span>
        </div>
      </div>
      <button id="reset-button" style="display: none">Reset Game</button>
      <script src="script.js"></script>
    </div>
  </body>
</html>
```

5.2 CSS

```
* {
  margin: 0;
  padding: 0;
}
body {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  background-color: rgb(183, 226, 255);
  height: 100vh;
  font-family: "Gill Sans", "Gill Sans MT", Calibri, "Trebuchet MS", sans-serif;
  margin: 0;
}
h1 {
  margin-bottom: 40px;
  text-shadow: 1px 1px 1px rgba(128, 128, 128, 0.534);

  font-size: 50px;
}
#everything {
  position: relative;
  text-align: center;
  background-color: rgba(240, 248, 255, 0.8);
  padding: 80px;
  border-radius: 0.5em;
  box-shadow: 1px 1px 1px gray;
}
#game-grid {
  display: grid;
  grid-template-columns: repeat(var(--grid-size), minmax(0, 1fr));
  grid-gap: 5px;
  width: 0; /* Adjust the width based on the desired size */
  height: calc(
    var(--grid-size) * 56.6px
  ); /* Adjust the height based on the desired size */
  justify-content: center;
  align-content: center;
```

```
  margin: auto;
}

.block {
  width: 50px;
  height: 50px;
  background-color: lightgray;
  border: 1px solid black;
  border-radius: 0.1em;
  box-shadow: 1px 1px 1px gray;
  text-align: center;
  line-height: 50px;
  font-size: 20px;
  cursor: pointer;
}

@keyframes scale-in {
  from {
    transform: scale(0);
  }
  to {
    transform: scale(1);
  }
}
.block.revealed {
  pointer-events: none;
  animation: scale-in 0.2s forwards;
}
.block:hover {
  background-color: aliceblue;
}

#game-container {
  text-align: center;
}

#score {
  margin-top: 10px;
  font-size: 20px;
  margin-bottom: 10px;
}

#grid-size-form {
  margin-bottom: 20px;
  text-align: center;
}

.grid-size-option {
  display: inline-block;
  width: 75px;
  height: 75px;
  margin-right: 10px;
  background-color: lightgray;
  border: 1px solid gray;
  border-radius: 0.5em;
  text-align: center;
  line-height: 75px;
  font-size: 18px;
  cursor: pointer;
  box-shadow: 1px 1px 1px rgba(128, 128, 128, 0.534);
  margin-top: 5px;
}

.grid-size-option:hover {
  background-color: gray;
  color: white;
```

```
}
#reset-button {
  border: 1px solid gray;
  border-radius: 0.2em;
  width: auto;
  height: auto;
  margin: 20px auto;
  margin-bottom: 0;
  padding: 10px;
  cursor: pointer;
  box-shadow: 1px 1px 1px gray;
  font-family: "Gill Sans", "Gill Sans MT", Calibri, "Trebuchet MS", sans-serif;
  font-size: large;
}
#reset-button:hover {
  background-color: gray;
  box-shadow: 1px 1px 1px gray;

  color: white;
}

.overlay {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, 0.5);
  z-index: 9999;
  opacity: 0;
  transition: opacity 0.5s;
  pointer-events: none;
}

.info-window {
  position: fixed;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%) scale(0.5);
  width: 500px;
  background-color: white;
  padding: 50px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
  z-index: 10000;
  opacity: 0;
  transition: opacity 0.5s, transform 0.5s;
  pointer-events: none;
  border: 2px solid black;
  border-radius: 0.5em;
  text-align: left;
}
.info-window h3 {
  text-align: center;
  font-size: 30px;
  margin-bottom: 20px;
}
.info-window.show {
  opacity: 1;
  transform: translate(-50%, -50%) scale(1);
  pointer-events: auto;
}
#info-button {
  width: 25px;
  height: 25px;
  padding: 15px;
  font-family: "Gill Sans", "Gill Sans MT", Calibri, "Trebuchet MS", sans-serif;
```

```
  font-size: medium;
  cursor: pointer;
  position: absolute;
  top: 3%;
  right: 3%;
}

#back-button {
  width: 25px;
  height: 25px;
  padding: 0;
  font-family: "Gill Sans", "Gill Sans MT", Calibri, "Trebuchet MS", sans-serif;
  font-size: medium;
  cursor: pointer;
  position: absolute;
  top: 6%;
  left: 5%;
}

#close-button {
  border: 1px solid gray;
  text-align: center;
  border-radius: 50%;
  width: 25px;
  height: 25px;
  padding: 0px;
  cursor: pointer;
  font-family: "Gill Sans", "Gill Sans MT", Calibri, "Trebuchet MS", sans-serif;
  font-size: medium;
}
#close-button:hover {
  background-color: gray;
  color: white;
}
#cl {
  margin: 0;
  position: absolute;
  top: 6%;
  right: 5%;
}
.startbtn {
  border: 1px solid gray;
  border-radius: 0.2em;
  width: auto;
  height: auto;
  margin: 4px;
  padding: 10px;
  cursor: pointer;
  box-shadow: 1px 1px 1px rgba(128, 128, 128, 0.534);

  font-family: "Gill Sans", "Gill Sans MT", Calibri, "Trebuchet MS", sans-serif;
  font-size: large;
}
.startbtn:hover {
  background-color: gray;
  color: white;
  box-shadow: 1px 1px 1px rgba(128, 128, 128, 0.534);
}
#double-score {
  margin-top: 20px;
  font-size: 20px;
  margin-bottom: 10px;
}
form {
  font-size: large;
}
```

```
input {
  margin: 5px;
  border-radius: 1px;
}
#start {
  border: 1px solid gray;
  border-radius: 0.2em;
  width: auto;
  height: auto;
  padding: 10px;
  cursor: pointer;
  font-family: "Gill Sans", "Gill Sans MT", Calibri, "Trebuchet MS", sans-serif;
  font-size: large;
  margin: 15px;
}
#start:hover {
  background-color: gray;
  color: white;
}

#info-window p {
  font-size: 19px;
}
```

5.3 JavaScript

```
// Get form, game container, and reset button elements
const gridSizeForm = document.getElementById("grid-size-form");
const gameContainer = document.getElementById("game-container");
const resetButton = document.getElementById("reset-button");
const startWindow = document.getElementById("start-window");
const backButton = document.getElementById("back-button");
const nameForm = document.getElementById("player-names-form");
const namePrompt = document.getElementById("name-prompt");

let playerChoice = 1;
let size = 0;
let player1Name = "P1";
let player2Name = "P2";

// Function to set the playerChoice to 1
function single() {
  playerChoice = 1;
  startWindow.style.display = "none";
  gridSizeForm.style.display = "block";
  backButton.style.display = "block";
}

// Function to set the playerChoice to 2
function double() {
  playerChoice = 2;
  startWindow.style.display = "none";
  gridSizeForm.style.display = "block";
  backButton.style.display = "block";
}

backButton.addEventListener("click", function () {
  // Go back one screen based on the current screen
  if (gridSizeForm.style.display == "block") {
    startWindow.style.display = "block";
    gridSizeForm.style.display = "none";
    backButton.style.display = "none";
  } else if (namePrompt.style.display == "block") {
    namePrompt.style.display = "none";
    gridSizeForm.style.display = "block";
  }
```

```
});

// Function to show the game grid with the selected grid size
function showGameGrid(gridSize) {
  // Hide the grid size form and show the game container
  gridSizeForm.style.display = "none";
  size = gridSize;
  // Call the function to start the game with the selected grid size based on playerChoice
  if (playerChoice === 1) {
    gameContainer.style.display = "block";
    startSingleGame(gridSize);
  } else if (playerChoice === 2) {
    namePrompt.style.display = "block";
  }
}

// Function to start the game with the specified grid size
function startSingleGame(gridSize) {
  // Inititalize the shield status
  let shieldActive = false;
  let shieldTurns = 0;
  backButton.style.display = "none";
  document.getElementById("score").style.display = "block";
  document.getElementById("double-score").style.display = "none";
  // Initialize game state
  let runScore = 0;
  let blocksRevealed = 0;
  let fielderPositions;
  if(gridSize!=10) fielderPositions = generateRandomPositions(gridSize, 2*gridSize-8);
  else fielderPositions = generateRandomPositions(gridSize,11);
  // Create game grid
  const gameGrid = document.getElementById("game-grid");
  gameGrid.innerHTML = "";
  gameGrid.style.gridTemplateColumns = `repeat(${gridSize}, 1fr)`;
  gameGrid.style.pointerEvents = "auto"; // Enable click events on the game grid

  // Add event listener to the game grid
  gameGrid.addEventListener("click", handleBlockClick);

  for (let i = 0; i < gridSize; i++) {
    for (let j = 0; j < gridSize; j++) {
      const block = document.createElement("div");
      block.classList.add("block");
      block.dataset.row = i;
      block.dataset.col = j;

      // Add a random chance for a block to give a score of two, four, six or shield-power-up
      if (Math.random() < 0.35) {
        block.dataset.score = 2;
      } else if (Math.random() < 0.14) {
        block.dataset.score = 4;
      } else if (Math.random() < 0.10) {
        block.dataset.score = 6;
      } else if (Math.random() < 0.06) {
        block.dataset.score = 10;
      } else {
        block.dataset.score = 1;
      }

      gameGrid.appendChild(block);
    }
  }

```

```
104  // Function to handle block click event
105  function handleBlockClick(event) {
106    const clickedElement = event.target;
107    // Check if the shield is active
108    if (shieldTurns != 0) {
109      shieldTurns -= 1;
110      if (shieldTurns == 0) alert("Shield expired");
111    } else {
112      shieldActive = false;
113    }
114    // Check if the clicked element is a block
115    if (!clickedElement.classList.contains("block")) {
116      return; // Ignore click on non-block elements
117    }
118
119    // Get the clicked block as a variable
120    const clickedBlock = clickedElement;
121    const row = parseInt(clickedBlock.dataset.row);
122    const col = parseInt(clickedBlock.dataset.col);
123    const score = parseInt(clickedBlock.dataset.score);
124
125    // Check if the block is already revealed
126    if (clickedBlock.classList.contains("revealed")) {
127      return; // Ignore click on revealed block
128    }
129
130    if (blocksRevealed === gridSize*gridSize) {
131      //End the game in case the shield protects the players from all the fielders
132      endGame(runScore);
133    }
134
135    if (fielderPositions.some((pos) => pos[0] === row && pos[1] === col)) {
136      // Clicked on a block with a fielder
137
138      clickedBlock.style.boxShadow = "1px 1px 1px red";
139      clickedBlock.style.backgroundColor = "#def2c8";
140      clickedBlock.style.backgroundSize = "cover";
141      // If shield is active, then it is not out
142      if (shieldActive) {
143        alert("The shield protected you!");
144        clickedBlock.style.backgroundImage = "url('./images/shield.png')";
145        clickedBlock.classList.add("revealed");
146        blocksRevealed+=1;
147        clickedBlock.removeEventListener("click", handleBlockClick);
148        shieldActive = false;
149        shieldTurns = 0;
150      } else {
151        endGame(runScore);
152        clickedBlock.style.backgroundImage = "url('./images/out.png')";
153      }
154    } else if (score != 10) {
155      // Clicked on a block without a fielder
156      clickedBlock.classList.add("revealed");
157      blocksRevealed+=1;
158      clickedBlock.removeEventListener("click", handleBlockClick);
159      clickedBlock.style.backgroundColor = "#def2c8";
160      clickedBlock.style.boxShadow = "1px 1px 1px green";
161      runScore += score;
162      document.getElementById("run-score").textContent = runScore;
163      clickedBlock.textContent = score;
164    } else {
165      // Code for when the user clicks on a shield block
166      shieldActive = true;
167      shieldTurns = 4;
168      alert("You got a shield for 3 turns!!");
169      clickedBlock.classList.add("revealed");
```

```
      blocksRevealed+=1;
      clickedBlock.removeEventListener("click", handleBlockClick);
      clickedBlock.style.backgroundColor = "#def2c8";
      clickedBlock.style.boxShadow = "1px 1px 1px green";
      clickedBlock.style.backgroundSize = "cover";
      clickedBlock.style.backgroundImage = "url('./images/shield.png')";
    }
  }

  // Function to end the game and display the final score
  function endGame(score) {
    gameGrid.removeEventListener("click", handleBlockClick);
    gameGrid.style.pointerEvents = "none"; // Disable further clicks on the game grid
    resetButton.style.display = "block";
    alert(`Game over! Your final score is ${score}.`);
  }
}

nameForm.addEventListener("submit", function (event) {
  event.preventDefault(); // Prevent the default form submission behavior
  namePrompt.style.display = "none";
  // Get the entered player names
  player1Name = document.getElementById("player1-name").value;
  player2Name = document.getElementById("player2-name").value;

  // Update player names in the UI
  document.getElementById("player1-score").textContent = `${player1Name} : 0`;
  document.getElementById("player2-score").textContent = `${player2Name} : 0`;
  // Start the game using the entered names
  gameContainer.style.display = "block";
  startDoubleGame(size);
});

function startDoubleGame(gridSize) {
  // Initialize game state
  backButton.style.display = "none";
  document.getElementById("score").style.display = "none";
  document.getElementById("double-score").style.display = "block";
  document.getElementById("player1-score").style.textShadow = "1px 1px 1px #def2c8";
  document.getElementById("player2-score").style.textShadow = "0px 0px 0px #F7F18A";
  let currentPlayer = 1;
  let p1out = false;
  let p2out = false;
  let p1shield = false;
  let p2shield = false;
  let p1shieldTurns = 0;
  let p2shieldTurns = 0;
  let player1Score = 0;
  let player2Score = 0;
  let blocksRevealed = 0;
  if(gridSize!=10) fielderPositions = generateRandomPositions(gridSize, 2*gridSize-8);
  else fielderPositions = generateRandomPositions(gridSize,11);

  // Create game grid
  const gameGrid = document.getElementById("game-grid");
  gameGrid.innerHTML = "";
  gameGrid.style.gridTemplateColumns = `repeat(${gridSize}, 1fr)`;
  gameGrid.style.pointerEvents = "auto"; // Enable click events on the game grid

  // Add event listener to the game grid
  gameGrid.addEventListener("click", handleBlockClick);

```

```
232  for (let i = 0; i < gridSize; i++) {
233    for (let j = 0; j < gridSize; j++) {
234      const block = document.createElement("div");
235      block.classList.add("block");
236      block.dataset.row = i;
237      block.dataset.col = j;
238
239      // Add a random chance for a block to give a score of two
240      if (Math.random() < 0.35) {
241        block.dataset.score = 2;
242      } else if (Math.random() < 0.14) {
243        block.dataset.score = 4;
244      } else if (Math.random() < 0.10) {
245        block.dataset.score = 6;
246      } else if (Math.random() < 0.06) {
247        block.dataset.score = 10;
248      } else {
249        block.dataset.score = 1;
250      }
251
252      gameGrid.appendChild(block);
253    }
254  }
255
256  // Function to handle block click event
257  function handleBlockClick(event) {
258    const clickedElement = event.target;
259    if (currentPlayer == 1 && p1shield) {
260      p1shieldTurns -= 1;
261      if (p1shieldTurns === 0) {
262        p1shield=false;
263        alert(`${player1Name}'s shield ran out!!`);
264      }
265    }
266    if (currentPlayer == 2 && p2shield) {
267      p2shieldTurns -= 1;
268      if (p2shieldTurns === 0) {
269        p2shield = false;
270        alert(`${player2Name}'s shield ran out!!`);
271      }
272    }
273    // Check if the clicked element is a block
274    if (!clickedElement.classList.contains("block")) {
275      return; // Ignore click on non-block elements
276    }
277
278    const clickedBlock = clickedElement;
279    const row = parseInt(clickedBlock.dataset.row);
280    const col = parseInt(clickedBlock.dataset.col);
281    const score = parseInt(clickedBlock.dataset.score);
282
283    // Check if the block is already revealed
284    if (clickedBlock.classList.contains("revealed")) {
285      return; // Ignore click on revealed block
286    }
287    clickedBlock.classList.add("revealed");
288    blocksRevealed+=1;
289    if (fielderPositions.some((pos) => pos[0] === row && pos[1] === col)) {
290      // Clicked on a block with a fielder
291      clickedBlock.style.backgroundImage = "url('./images/out.png')";
292      clickedBlock.style.backgroundSize = "cover";
293      clickedBlock.removeEventListener("click", handleBlockClick);
294      clickedBlock.style.animation = "fade-in 0.5s";
295      clickedBlock.style.boxShadow = "1px 1px 1px red";
296      // Update scores for the respective players
297      if (currentPlayer === 1) {
```

```
        if(p2out===false) togglePlayer();
        document.getElementById(
          "player1-score"
        ).textContent = `${player1Name} : ${player1Score}`;
        clickedBlock.style.backgroundColor = "#def2c8";
        if (p1shield == false) {
          p1out = true;
          if (p2out === false)
            alert(`${player1Name} got out at ${player1Score} runs!!`);
        }
        if (p1shield) {
          alert(`${player1Name} got protected by the shield!!`);
          clickedBlock.style.backgroundImage = "url('./images/shield.png')";
          p1shield = false;
          p1shieldTurns = 0;
        }
      } else if (currentPlayer === 2) {
        clickedBlock.style.backgroundColor = "#F7F18A";
        if(p1out===false) togglePlayer();
        document.getElementById(
          "player2-score"
        ).textContent = `${player2Name} : ${player2Score}`;
        if (p2shield == false) {
          p2out = true;
          if (p1out === false)
            alert(`${player2Name} got out at ${player2Score} runs!!`);
        } else {
          alert(`${player2Name} got protected by the shield!!`);
          clickedBlock.style.backgroundImage = "url('./images/shield.png')";
          p2shield = false;
          p2shieldTurns = 0;
        }
      }

      if (p1out && p2out) {
        endGame();
      }
    } else if (score != 10) {
      // Clicked on a block without a fielder
      clickedBlock.classList.add("revealed");
      clickedBlock.removeEventListener("click", handleBlockClick);

      // Update scores for the respective players
      if (currentPlayer === 1) {
        player1Score += score;
        clickedBlock.style.backgroundColor = "#def2c8";
        clickedBlock.style.boxShadow = "1px 1px 1px green";
        if (p2out === false) togglePlayer();
        document.getElementById(
          "player1-score"
        ).textContent = `${player1Name} : ${player1Score}`;
      } else if (currentPlayer === 2) {
        player2Score += score;
        clickedBlock.style.backgroundColor = "#F7F18A ";
        clickedBlock.style.boxShadow = "1px 1px 1px yellow";
        if (p1out === false) togglePlayer();
        document.getElementById(
          "player2-score"
        ).textContent = `${player2Name} : ${player2Score}`;
      }
      clickedBlock.textContent = score;
    } else {
      if (currentPlayer === 1) {
        alert(`${player1Name} got a shield for 3 turns!!`);
        p1shield = true;
        p1shieldTurns = 4;
```

```
364         clickedBlock.removeEventListener("click", handleBlockClick);
365         clickedBlock.style.backgroundColor = "#def2c8";
366         clickedBlock.style.backgroundImage = "url('./images/shield.png')";
367         clickedBlock.style.backgroundSize = "cover";
368         if (p2out === false) togglePlayer();
369       } else {
370         p2shield = true;
371         alert(`${player2Name} got a shield for 3 turns!!`);
372         p2shieldTurns = 4;
373         clickedBlock.removeEventListener("click", handleBlockClick);
374         clickedBlock.style.backgroundColor = "#F7F18A";
375         clickedBlock.style.backgroundImage = "url('./images/shield.png')";
376         clickedBlock.style.backgroundSize = "cover";
377         if (p1out === false) togglePlayer();
378       }
379     } if(blocksRevealed == gridSize*gridSize) endgame(); //In case the shield
      protects the players from all the fielders
380   }
381
382   // Function to toggle between players
383   function togglePlayer() {
384     currentPlayer = currentPlayer === 1 ? 2 : 1;
385     if(currentPlayer===1){
386       document.getElementById("player1-score").style.textShadow = "1px 1px 1px #
      def2c8";
387       document.getElementById("player2-score").style.textShadow = "0px 0px 0px #
      F7F18A";
388     }else{
389       document.getElementById("player1-score").style.textShadow = "0px 0px 0px #
      def2c8";
390       document.getElementById("player2-score").style.textShadow = "1px 1px 1px #
      F7F18A";
391     }
392   }
393
394   // Function to end the game and display the final scores
395   function endGame() {
396     gameGrid.removeEventListener("click", handleBlockClick);
397     gameGrid.style.pointerEvents = "none"; // Disable further clicks on the game
      grid
398     resetButton.style.display = "block";
399     if (player1Score > player2Score) {
400       alert(`${player1Name} won the game!!`);
401     } else if (player2Score > player1Score) {
402       alert(`${player2Name} won the game!!`);
403     } else {
404       alert("It's a tie!!");
405     }
406   }
407 }
408
409 // Add event listener to the reset button click event
410 resetButton.addEventListener("click", function () {
411   // Reset the game
412   gridSizeForm.style.display = "none";
413   gameContainer.style.display = "none";
414   resetButton.style.display = "none";
415   backButton.style.display = "none";
416   startWindow.style.display = "block";
417   document.getElementById("run-score").textContent = "0";
418   document.getElementById("player1-score").textContent = "0";
419   document.getElementById("player2-score").textContent = "0";
420   // Remove all blocks from the game grid
421   const gameGrid = document.getElementById("game-grid");
422   gameGrid.innerHTML = "";
423 });
```

```
424
425 // Generate an array of random positions
426 function generateRandomPositions(gridSize, count) {
427   const positions = [];
428   const allPositions = [];
429   for (let i = 0; i < gridSize; i++) {
430     for (let j = 0; j < gridSize; j++) {
431       allPositions.push([i, j]);
432     }
433   }
434   for (let i = 0; i < count; i++) {
435     const index = Math.floor(Math.random() * allPositions.length);
436     positions.push(allPositions.splice(index, 1)[0]);
437   }
438   return positions;
439 }
440
441 window.addEventListener("DOMContentLoaded", () => {
442   const infoWindow = document.getElementById("info-window");
443   const infoButton = document.getElementById("info-button");
444   const closeButton = document.getElementById("close-button");
445   const overlay = document.getElementById("overlay");
446
447   infoButton.addEventListener("click", openInfoWindow);
448   closeButton.addEventListener("click", closeInfoWindow);
449
450   function openInfoWindow() {
451     infoWindow.classList.add("show");
452     overlay.style.opacity = "1";
453     overlay.style.pointerEvents = "auto";
454   }
455
456   function closeInfoWindow() {
457     infoWindow.classList.remove("show");
458     overlay.style.opacity = "0";
459     overlay.style.pointerEvents = "none";
460   }
461 });
```

References

[1] URL: https://www.w3schools.com/css/css_grid.asp.

[2] URL: https://www.w3schools.com/css/css3_transitions.asp.

[3] URL: https://www.w3schools.com/css/css3_animations.asp.

[4] Marin Haverbeke. *Eloquent JavaScript*.

[5] URL: https://www.w3schools.com/js/js_events.asp.