New chat

Recent

Battleship Game (HTML/JS)

Gems

Test Gem

Brainstormer

Career guide

Show more

Gem manager

Help

Activity

Can you create a battleship game in a 6x6 board, 6x3 top portion for AI, 6x3 bottom portion should be for human player using JavaScript and HTML in a single file. In the first phase, player or ai places 4 different ships in form of shapes and in the second phase, they try to sink each other's ship in alternating turn.

Show drafts

HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Battleship</title>
<style>
table {
  border-collapse: collapse;
}
td {
  width: 30px;
  height: 30px;
  border: 1px solid black;
  text-align: center;
  cursor: pointer;


}
.ship {
  background-color: gray;
}
.hit {
  background-color: red;
}
.miss {
  background-color: blue;
}
</style>
</head>
<body>

<h1>Battleship</h1>

<div id="board"></div>

<script>
const boardSize = 6;
const playerBoard = Array(boardSize).fill(null).map(() => Array(boardSi
const aiBoard = Array(boardSize).fill(null).map(() => Array(boardSize).
const ships = [
  { name: 'Carrier', length: 5 },
  { name: 'Battleship', length: 4 },
  { name: 'Cruiser', length: 3 },
  { name: 'Submarine', length: 3 },
  { name: 'Destroyer', length:

 2 }
];
let playerShipsPlaced = 0;
let aiShipsPlaced = 0;
let playerTurn = true;

// Create the game board
const boardDiv = document.getElementById('board');
const table = document.createElement('table');
for (let i = 0; i < boardSize; i++) {
  const row = document.createElement('tr');
  for (let j = 0; j < boardSize; j++) {
    const cell = document.createElement('td');
    cell.id


 = `cell-${i}-${j}`;
    if (i < boardSize / 2) {
      cell.classList.add('ai');
    } else {
      cell.classList.add('player');
      cell.addEventListener('click', () => handleClick(i, j));
    }
    row.appendChild(cell);
  }
  table.appendChild(row);
}
boardDiv.appendChild(table);

// Place ships randomly for AI
while (aiShipsPlaced < ships.length) {
  const ship = ships[aiShipsPlaced];
  const isHorizontal = Math.random() < 0.5;
  let x, y;
  if (isHorizontal) {
    x = Math.floor(Math.random() * (boardSize - ship.length + 1));
    y = Math.floor(Math.random() * (boardSize / 2));
  } else {
    x = Math.floor(Math.random() * (boardSize));
    y = Math.floor(Math.random() * (boardSize / 2 - ship.length + 1));
  }
  if (canPlaceShip(aiBoard, ship, x, y, isHorizontal)) {
    placeShip(aiBoard, ship, x, y, isHorizontal);
    aiShipsPlaced++;
  }
}

// Function to check if a ship can be placed
function canPlaceShip(board, ship, x, y, isHorizontal) {
  for (let i = 0; i < ship.length; i++) {
    const newX = isHorizontal ? x + i : x;
    const newY = isHorizontal ? y : y + i;
    if (newX < 0 || newX >= boardSize || newY < 0 || newY >= boardSize
      return false;
    }
  }
  return true;
}

// Function to place a ship on the board
function placeShip(board, ship, x, y, isHorizontal) {
  for (let i = 0; i < ship.length; i++) {
    const newX = isHorizontal ? x + i : x;
    const newY = isHorizontal ? y : y + i;
    board[newY][newX] = 1;
    // Uncomment the line below to reveal AI ships during placement for
    // document.getElementById(`cell-${newY}-${newX}`).classList.add('sh
  }
}

// Handle cell click for player
function handleClick(i, j) {
  if (playerTurn && playerShipsPlaced < ships.length) {
    // Place ships
    const ship = ships[playerShipsPlaced];
    const isHorizontal = Math.random() < 0.5; // Placeholder for player
    if (canPlaceShip(playerBoard, ship, j, i, isHorizontal)) {
```

```
      placeShip(playerBoard, ship, j, i, isHorizontal);
      playerShipsPlaced++;
      document.getElementById(`cell-${i}-${j}`).classList.add('ship');
      if (playerShipsPlaced === ships.length) {
        playerTurn = false;
        aiTurn();
      }
    }
  } else if (playerTurn && i < boardSize / 2) {
    // Attack AI
    attack(aiBoard, i, j);
  }
}

// AI's turn to attack
function aiTurn() {
  let x, y;
  do {
    x = Math.floor(Math.random() * boardSize);
    y = Math.floor(Math.random() * (boardSize / 2) + (boardSize / 2));
  } while (playerBoard[y][x] === 2 || playerBoard[y][x] === 3);
  attack(playerBoard, y, x);
}

// Function to handle an attack
function attack(board, i, j) {
  const cell = document.getElementById(`cell-${i}-${j}`);
  if (board[i][j] === 1) {
    cell.classList.add('hit');
    board[i][j] = 2;
    checkWin();
  } else if (board[i][j] === 0) {
    cell.classList.add('miss');
    board[i][j] = 3;
    playerTurn = !playerTurn;
    if (!playerTurn) {
      setTimeout(aiTurn, 1000); // Delay AI turn for better UX
    }
  }
}

// Function to check for a win
function checkWin() {
  let playerShipsRemaining = ships.reduce((sum, ship) => sum + ship.leng
  let aiShipsRemaining = playerShipsRemaining;
  for (let i = 0; i < boardSize; i++) {
    for (let j = 0; j < boardSize; j++) {
      if (i < boardSize / 2 && aiBoard[i][j] === 2) {
        aiShipsRemaining--;
      } else if (i >= boardSize / 2 && playerBoard[i][j] === 2) {
        playerShipsRemaining--;
      }
    }
  }
  if (aiShipsRemaining === 0) {
    alert('You win!');
  } else if (playerShipsRemaining === 0) {
    alert('AI wins!');
  }
}
</script>

</body>
</html>
```

Use code with caution.

This code creates a basic Battleship game with the following features:

- **6x6 board:** The game board is divided into two 6x3 sections for the AI and the player.
- **Ship placement:** The AI randomly places its ships. The player currently has automated placement, but this can be modified to allow manual placement.
- **Alternating turns:** The player and AI alternate turns attacking each other.
- **Hit and miss indicators:** Red indicates a hit, blue indicates a miss.
- **Win condition:** The game ends when either the player or the AI sinks all of the opponent's ships.

To play the game:

1. Save the code as an HTML file (e.g., `battleship.html`).
2. Open the file in a web browser.

Further improvements:

- **Manual ship placement for the player:** Add functionality for the player to drag and drop or click to place their ships.
- **Improved AI:** Implement a more intelligent AI that targets ships more effectively.
- **Game over screen:** Display a clear game over message with the winner and an option to restart the game.
- **Visual enhancements:** Add more styling and graphics to improve the game's appearance.
- **Sound effects:** Add sound effects for hits, misses, and sinking ships.

This is a basic implementation of Battleship, but it provides a good foundation for building a more complete and engaging game.

Sources and related content

github.com
github.com

github.com
github.com

github.com
github.com

Settings

Dhaka, Bangladesh
Based on your places (Home) • Update location

Ask Gemini Advanced

Gemini can make mistakes, so double-check it