|  |  |
| --- | --- |
|  | <!-- Ryan Davis |
|  | 9/1/17 |
|  | CS 4500 |
|  | Student ID: 12171429 |
|  | Student SSO: rpd4g5 |
|  | Hw2 |
|  | \*\* - Note to Dr. Miller - \*\*: |
|  | I did not have a |
|  | partner on this assignment |
|  | because I missed class on |
|  | Monday due to the Wake, and |
|  | was not assigned a partner. |
|  | --> |
|  |  |
|  | <!-- This game is an HTML, CSS, and JavaScript |
|  | based game called "A Pretty Strange Game." |
|  |  |
|  | This game simulates a ball moving around a grid. |
|  | The ball always starts in the bottom-left corner |
|  | of the grid, and when the game is simulated the |
|  | ball moves all over the board at random until |
|  | it lands in the top-right corner, in which case |
|  | you win. If the ball has not made it to the |
|  | top-right corner within 1,000,000 cell touches |
|  | (1 cell touch per move; even if the ball doesn't |
|  | move anywhere due to out of bounds restraints) |
|  | then you lose. |
|  |  |
|  | In this iteration of the software you can specify |
|  | the number of columns and rows you would like |
|  | to have as long as they are between 5 and 20 |
|  | inclusively. |
|  |  |
|  | I added the option to move the ball one space |
|  | at a time as an additional feature although |
|  | I created it mostly for debugging to be honest |
|  | however it adds a nice bit of functionality to |
|  | the game in my opinion. |
|  |  |
|  | Due to some time constraints and an initial |
|  | oversight in my design I did not implement |
|  | a method for tracking the number of times |
|  | an individual cell gets touched although I could |
|  | add this in future versions of the product without |
|  | too much re-work I would guess. |
|  |  |
|  | Thanks and enjoy the code! |
|  | --> |
|  |  |
|  | <!DOCTYPE html> |
|  | <html> |
|  | <head> |
|  | <meta charset="UTF-8"> |
|  | <title>A Pretty Strange Game</title> |
|  | <style> |
|  | td.circle::before{ |
|  | content: attr(marker); |
|  | display: block; |
|  | background: #f00; |
|  | width: 10px; |
|  | height: 10px; |
|  | line-height:10px; |
|  | text-align:center; |
|  | border-radius: 50%; |
|  | margin:0 auto; |
|  | } |
|  | h2.enterInput { |
|  | font-family: Arial, Helvetica, sans-serif; |
|  | } |
|  | </style> |
|  | </head> |
|  | <body> |
|  | <div> |
|  | <h1>A Pretty Strange Game</h1> |
|  | </div> |
|  | <!-- input the number of columns and rows --> |
|  | <div> |
|  | <div> |
|  | <br> |
|  | <h2 class="enterInput">Enter 5-20 (inclusive) for Columns and Rows:</h2> |
|  | </div> |
|  | Columns:<br> |
|  | <input type="text" id="columns"> |
|  | <br> |
|  | Rows:<br> |
|  | <input type="text" id="rows"> |
|  | <br> |
|  | <button onclick="getUserInput()">Submit</button> |
|  |  |
|  | </div> |
|  | <!-- javascript for taking the user input --> |
|  | <script> |
|  | //Default values for the columns and rows of the grid: |
|  | var columns = 10; |
|  | var rows = 10; |
|  | var tableReDrawn = false; |
|  | var cell\_touch\_count; |
|  | var max\_cell\_touches; |
|  | var original\_square\_pos; |
|  | var previous\_square\_pos; |
|  | var new\_square\_pos; |
|  | var td; |
|  | var att; |
|  |  |
|  | function getUserInput() { |
|  | tableReDrawn = true; |
|  | table\_structure.value = ""; //erases the old table so a new one can be drawn |
|  | console.log('DEBUG: getUserInput()'); |
|  | columns = document.getElementById("columns").value; |
|  | rows = document.getElementById("rows").value; |
|  |  |
|  | if (columns < 5 || columns > 20) { |
|  | var colsInput = document.getElementById("columns"); |
|  | colsInput.value = ""; |
|  | while (columns < 5 || columns > 20) { |
|  | columns = prompt("Please Enter 5 - 20 Columns (inclusive)"); |
|  | console.log(columns); |
|  | } |
|  | } |
|  | if (rows < 5 || rows > 20) { |
|  | var rowsInput = document.getElementById("rows"); |
|  | rowsInput.value = ""; |
|  | while (rows < 5 || rows > 20) { |
|  | rows = prompt("Please Enter 5 - 20 Rows (inclusive)"); |
|  | console.log(rows); |
|  | } |
|  | } |
|  | console.log('DEBUG: Final Columns = ' + columns); |
|  | console.log('DEBUG: Final Rows = ' + rows); |
|  | drawTable(rows, columns); |
|  | } |
|  | </script> |
|  | <!-- div tag for the building the grid --> |
|  | <div id="divTable"> |
|  | <!-- javascript section for the divTable grid --> |
|  | <script> |
|  | var divTable = document.getElementById('divTable'); |
|  | var table\_structure; |
|  | var i, j; |
|  | drawTable(rows, columns); |
|  |  |
|  | function drawTable(num\_rows, num\_cols) { |
|  | table\_structure = '<table border="1" width="100%">'; |
|  | for(i = 0; i < num\_rows; i++) { |
|  | table\_structure += '<tr>'; |
|  | for(j = 0; j < num\_cols; j++) { |
|  | table\_structure += '<td height="25" width="25">'; |
|  | } |
|  | table\_structure += '</td>'; |
|  | } |
|  | table\_structure += '</table>'; |
|  | divTable.innerHTML = table\_structure; |
|  |  |
|  | //Move counter metrics: |
|  | cell\_touch\_count = 0.0; |
|  | max\_cell\_touches = 1000001; |
|  | /\* |
|  | missing metrics (as of 8/24/17 @ 7:52am): |
|  | - min. times a particular cell was touched |
|  | - max. times a particular cell was touched |
|  | - number of times each particular cell was touched |
|  | \*/ |
|  |  |
|  | //Game board position vars: |
|  | original\_square\_pos = ((rows \* columns) - (columns)); |
|  | console.log('DEBUG: drawTable(num\_rows, num\_cols'); |
|  | console.log(original\_square\_pos); |
|  |  |
|  | td = document.getElementsByTagName("td")[original\_square\_pos]; //get the 90th (bottom left corner) <td> |
|  | att = document.createAttribute("class"); //creates a class attribute |
|  | att.value = "circle"; //sets the value of the class attribute |
|  | td.setAttributeNode(att); |
|  | } |
|  |  |
|  | //checkForWinner(position) called on each directional move: |
|  | function checkForWinnerOnBtnClick(position) { |
|  | console.log(columns); |
|  | console.log(position); |
|  | if (position == (columns-1)) { |
|  | var button = document.getElementsByTagName("button")[0]; |
|  | button.hidden = true; |
|  | document.getElementById('game\_result').innerHTML = "You Win!!!" |
|  | console.log('DEBUG: ' + 'position parameter = ' + position + ' YOU WIN!') |
|  | } |
|  | } |
|  |  |
|  | //checkForMaxTouches(touch\_count) called on each directional move: |
|  | function checkForMaxTouchesOnBtnClick(touch\_count) { |
|  | if (touch\_count >= max\_cell\_touches) { |
|  | var button = document.getElementsByTagName("button")[0]; |
|  | button.hidden = true; |
|  | document.getElementById('game\_result').innerHTML = "1,000,001 Touches - Game Over!" |
|  | console.log('DEBUG: ' + 'touch\_count parameter = ' + touch\_count + ' YOU LOSE!') |
|  | } |
|  | } |
|  |  |
|  | /\*gameSimulator() calls the moveFunction() and then itself is |
|  | called recursively by the setTimeout function until a conditional |
|  | is met at which point the interval is cleared and the sim stops: |
|  | \*/ |
|  | function gameSimulator() { |
|  | moveFunction(); |
|  | var showSim = setTimeout(function(){ gameSimulator(); }, 200); |
|  |  |
|  | if ((new\_square\_pos == (columns-1)) || cell\_touch\_count > 1000000) { |
|  | clearTimeout(showSim); |
|  | var mv\_button = document.getElementsByTagName("button")[0]; |
|  | var sim\_button = document.getElementsByTagName("button")[1]; |
|  | mv\_button.hidden = true; |
|  | sim\_button.hidden = true; |
|  | if (new\_square\_pos == (columns-1)) { |
|  | document.getElementById('result\_stats').innerHTML = "GAME RESULT = Marker Finish: WINNER"; |
|  | document.getElementById('touch\_total').innerHTML = 'Total Cell Touches = ' + cell\_touch\_count; |
|  | document.getElementById('avg\_touches\_per\_cell').innerHTML = 'Avg. Touches Per Cell = ' + (cell\_touch\_count/100.0); |
|  | } else { |
|  | document.getElementById('result\_stats').innerHTML = "GAME RESULT = 1,000,001 Touches: LOSER"; |
|  | document.getElementById('touch\_total').innerHTML = 'Total Cell Touches = ' + cell\_touch\_count; |
|  | document.getElementById('avg\_touches\_per\_cell').innerHTML = 'Avg. Touches Per Cell = ' + (cell\_touch\_count/100.0); |
|  | console.log('DEBUG: ' + 'touch\_count parameter = ' + cell\_touch\_count + ' YOU LOSE!'); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | //moveFunction() controls the movement logic of the marker on the board: |
|  | function moveFunction() { |
|  | //Place the marker in bottom-left corner to begin the game: |
|  | console.log('cell touch count = ' + cell\_touch\_count); |
|  | if (cell\_touch\_count < 1 && tableReDrawn == false) { |
|  | previous\_square\_pos = original\_square\_pos; //set this for the first turn |
|  | console.log('Original table; tableReDrawn == false'); |
|  | } else { |
|  | console.log('DEBUG: Cell Count > 1 || tableReDrawn == false'); |
|  | previous\_square\_pos = new\_square\_pos; //when tableReDrawn == true it clips this logic on the way back if |
|  | //if cell\_touch\_count > 1 |
|  | } |
|  |  |
|  | //if you are re-sizing the table then you will need to reset the previous\_square\_pos: |
|  | if (cell\_touch\_count < 1 && tableReDrawn == true) { |
|  | console.log('original\_square\_pos = ' + original\_square\_pos); |
|  | console.log('A new table; tableReDrawn == true AND cell\_count < 1'); |
|  | previous\_square\_pos = original\_square\_pos; |
|  | } |
|  |  |
|  | console.log('DEBUG: previous\_sqaure\_pos = ' + previous\_square\_pos); |
|  | console.log('DEBUG: moveFunction() executed'); |
|  | //Randomly choose a direction: up, down, left, or right: |
|  | var D = Math.floor((Math.random() \* 4) + 1); |
|  | //Randomly choose a number of steps: 0, 1, or 2: |
|  | var N = Math.floor((Math.random() \* 3) + 0); |
|  | console.log('DEBUG: D = ' + D + ' and N = ' + N); |
|  |  |
|  | console.log('DEBUG: new\_square\_pos = ' + previous\_square\_pos); |
|  |  |
|  | switch(D) { |
|  | case 1: |
|  | new\_square\_pos = previous\_square\_pos-(N\*10); |
|  | document.getElementsByTagName("td")[previous\_square\_pos].removeAttribute("class"); |
|  | console.log('DEBUG: new\_sqaure\_pos = ' + new\_square\_pos); |
|  |  |
|  | if (new\_square\_pos >= 0 && new\_square\_pos < (rows \* columns)) { |
|  | //count square touch (add to million count for analytics) |
|  | cell\_touch\_count += 1; |
|  | td = document.getElementsByTagName("td")[new\_square\_pos]; |
|  | att = document.createAttribute("class"); //creates a class attribute |
|  | att.value = "circle"; //sets the value of the class attribute |
|  | td.setAttributeNode(att); |
|  | } else { |
|  | //don't move |
|  | console.log('DEBUG: location ' + new\_square\_pos + ' out of bounds'); |
|  | new\_square\_pos = previous\_square\_pos; //reset so no move on o.o.b. |
|  | cell\_touch\_count += 1; |
|  | td = document.getElementsByTagName("td")[new\_square\_pos]; |
|  | att = document.createAttribute("class"); //creates a class attribute |
|  | att.value = "circle"; //sets the value of the class attribute |
|  | td.setAttributeNode(att); |
|  | } |
|  | checkForMaxTouchesOnBtnClick(cell\_touch\_count); |
|  | checkForWinnerOnBtnClick(new\_square\_pos); |
|  | break; |
|  | case 2: |
|  | new\_square\_pos = previous\_square\_pos+(N\*10); |
|  | previous\_square\_pos = new\_square\_pos-(N\*10); |
|  | document.getElementsByTagName("td")[previous\_square\_pos].removeAttribute("class"); |
|  |  |
|  | if (new\_square\_pos >= 0 && new\_square\_pos < (rows \* columns)) { |
|  | //count square touch (add to million count for analytics) |
|  | cell\_touch\_count += 1; |
|  | td = document.getElementsByTagName("td")[new\_square\_pos]; |
|  | att = document.createAttribute("class"); //creates a class attribute |
|  | att.value = "circle"; //sets the value of the class attribute |
|  | td.setAttributeNode(att); |
|  | } else { |
|  | // don't move |
|  | console.log('DEBUG: location ' + new\_square\_pos + ' out of bounds'); |
|  | new\_square\_pos = previous\_square\_pos; //reset so no move on o.o.b. |
|  | cell\_touch\_count += 1; |
|  | td = document.getElementsByTagName("td")[new\_square\_pos]; |
|  | att = document.createAttribute("class"); //creates a class attribute |
|  | att.value = "circle"; //sets the value of the class attribute |
|  | td.setAttributeNode(att); |
|  | } |
|  | checkForMaxTouchesOnBtnClick(cell\_touch\_count); |
|  | checkForWinnerOnBtnClick(new\_square\_pos); |
|  | break; |
|  | case 3: |
|  | new\_square\_pos = previous\_square\_pos-(N\*1); |
|  | previous\_square\_pos = new\_square\_pos+(N\*1); |
|  | document.getElementsByTagName("td")[previous\_square\_pos].removeAttribute("class"); |
|  | console.log('DEBUG: new\_sqaure\_pos = ' + new\_square\_pos); |
|  |  |
|  | if (new\_square\_pos >= 0 && new\_square\_pos < (rows \* columns)) { |
|  | //count square touch (add to million count for analytics) |
|  | cell\_touch\_count += 1; |
|  | td = document.getElementsByTagName("td")[new\_square\_pos]; |
|  | att = document.createAttribute("class"); //creates a class attribute |
|  | att.value = "circle"; //sets the value of the class attribute |
|  | td.setAttributeNode(att); |
|  | } else { |
|  | // don't move |
|  | console.log('DEBUG: location ' + new\_square\_pos + ' out of bounds'); |
|  | new\_square\_pos = previous\_square\_pos; //reset so no move on o.o.b. |
|  | cell\_touch\_count += 1; |
|  | td = document.getElementsByTagName("td")[new\_square\_pos]; |
|  | att = document.createAttribute("class"); //creates a class attribute |
|  | att.value = "circle"; //sets the value of the class attribute |
|  | td.setAttributeNode(att); |
|  | } |
|  | checkForMaxTouchesOnBtnClick(cell\_touch\_count); |
|  | checkForWinnerOnBtnClick(new\_square\_pos); |
|  | break; |
|  | case 4: |
|  | new\_square\_pos = previous\_square\_pos+(N\*1); |
|  | previous\_square\_pos = new\_square\_pos-(N\*1); |
|  | document.getElementsByTagName("td")[previous\_square\_pos].removeAttribute("class"); |
|  | console.log('DEBUG: new\_sqaure\_pos = ' + new\_square\_pos); |
|  |  |
|  | if (new\_square\_pos >= 0 && new\_square\_pos < (rows \* columns)) { |
|  | //count square touch (add to million count for analytics) |
|  | cell\_touch\_count += 1; |
|  | td = document.getElementsByTagName("td")[new\_square\_pos]; |
|  | att = document.createAttribute("class"); //creates a class attribute |
|  | att.value = "circle"; //sets the value of the class attribute |
|  | td.setAttributeNode(att); |
|  | } else { |
|  | // don't move |
|  | console.log('DEBUG: location ' + new\_square\_pos + ' out of bounds'); |
|  | new\_square\_pos = previous\_square\_pos; //reset so no move on o.o.b. |
|  | cell\_touch\_count += 1; |
|  | td = document.getElementsByTagName("td")[new\_square\_pos]; |
|  | att = document.createAttribute("class"); //creates a class attribute |
|  | att.value = "circle"; //sets the value of the class attribute |
|  | td.setAttributeNode(att); |
|  | } |
|  | checkForMaxTouchesOnBtnClick(cell\_touch\_count); |
|  | checkForWinnerOnBtnClick(new\_square\_pos); |
|  | break; |
|  | } |
|  | } |
|  | </script> |
|  | </div> |
|  | <!-- move button --> |
|  | <div> |
|  | <button onclick="moveFunction()">single move btn</button> |
|  | </div> |
|  | <!-- game simulator button --> |
|  | <div> |
|  | <button onclick="gameSimulator()">game simulator btn</button> |
|  | </div> |
|  | <!-- game result message --> |
|  | <div> |
|  | <h2 id="game\_result"></h2> |
|  | </div> |
|  | <!-- game stats --> |
|  | <div> |
|  | <h3 id="touch\_total"></h3> |
|  | <h3 id="avg\_touches\_per\_cell"></h3> |
|  | <h3 id="result\_stats"></h3> |
|  | </div> |
|  | </body> |
|  | </html> |
|  |  |