## МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЕ РФ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

## “ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Лабораторная работа №2

По дисциплине: «Веб-технологии»

На тему «Веб-игра «Лабиринт с автоматической генерацией» на HTML JS»

Выполнил :

Студент группы ВМО31

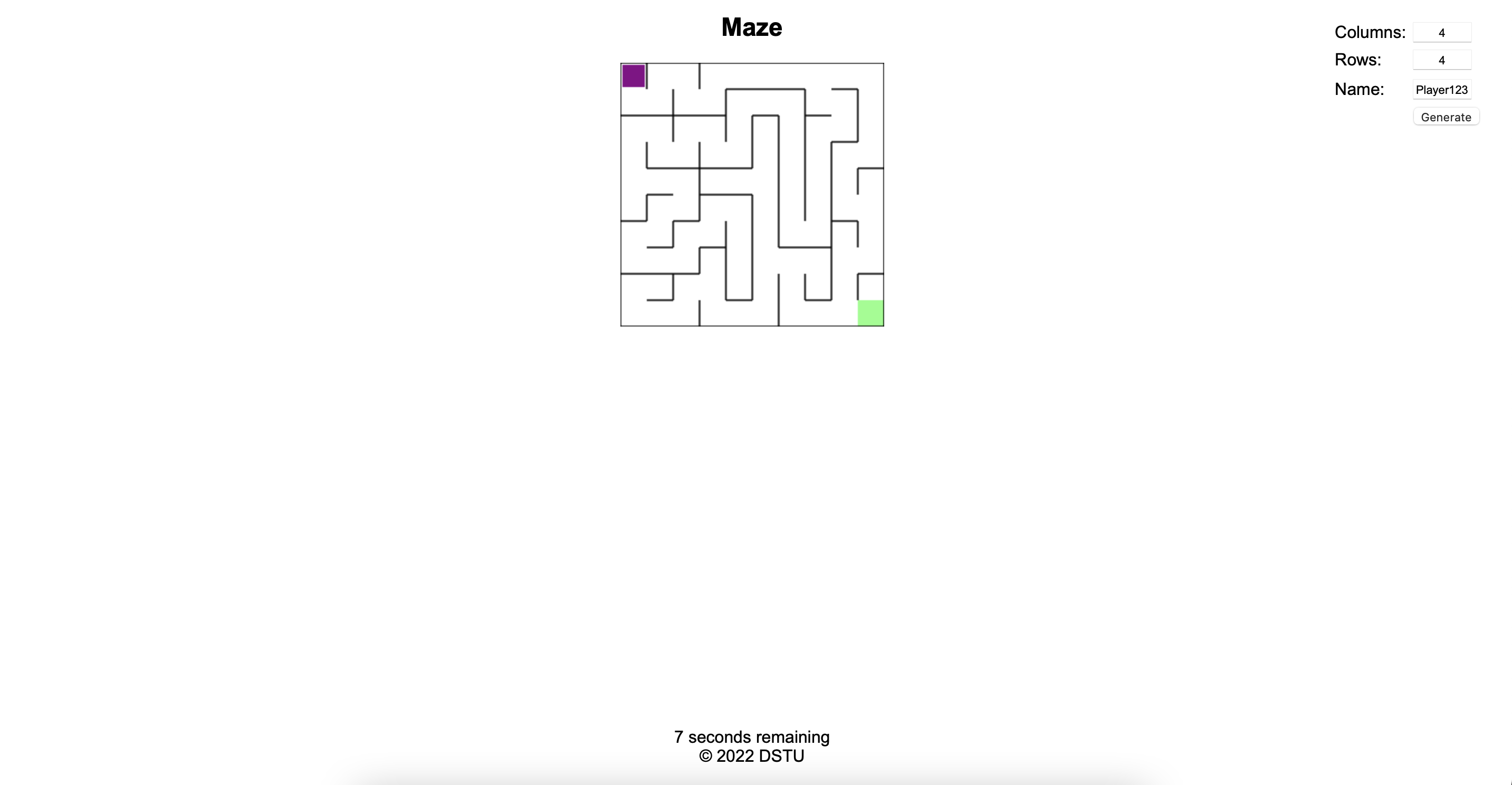
Оганесьянц К.П.

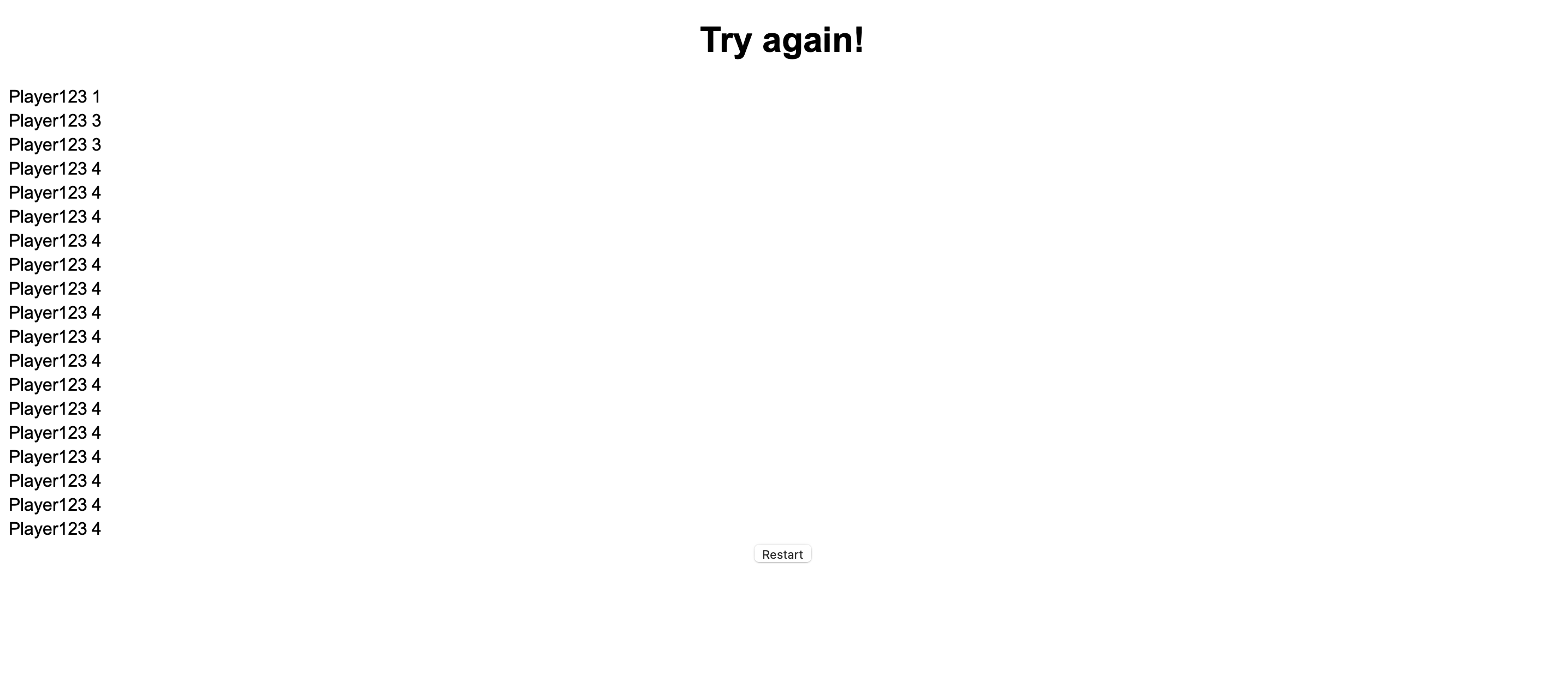
Проверил:

Чугунный К.А.

Ростов-на-Дону 2022 г.

Внешний вид:





Код программы:

**Index.html**

<!**DOCTYPE** html>

<**html** lang="en">

<**head**>

<**link** rel="stylesheet" href="style.css">

<**meta** charset="UTF-8">

<**script** src="script.js"></**script**>

<**title**>Maze</**title**>

</**head**>

<**body** onload="onLoad();">

<**h2**>Maze</**h2**>

<**canvas** id="mainForm"></**canvas**>

<**div** id="settings">

<**table**>

<**tbody**>

<**tr**>

<**td**><**label** for="cols">Columns:</**label**></**td**>

<**td**><**input** id="cols" type="text" size="5" value="4" /></**td**>

</**tr**><**tr**>

<**td**><**label** for="rows">Rows:</**label**></**td**>

<**td**><**input** id="rows" type="text" size="5" value="4" /></**td**>

</**tr**><**tr**>

</**tr**><**tr**>

<**td**><**label** for="rows">Name:</**label**></**td**>

<**td**><**input** id="name" type="text" size="5" value="Player123" /></**td**>

</**tr**><**tr**>

<**td**>&nbsp;</**td**>

<**td**><**input** id="generate" type="button" value="Generate" /></**td**>

</**tr**>

</**tbody**>

</**table**>

</**div**>

<**div** id="footer">

<**div** id="some\_div"></**div**>

&copy; 2022 DSTU

</**div**>

</**body**>

</**html**>

**Script.js**

**let** ctx;

**let** canvas;

**let** maze;

**let** mazeHeight;

**let** mazeWidth;

**let** player;

**class** Player {

**constructor**() {

**this**.col = 0;

**this**.row = 0;

}

}

**class** MazeCell {

**constructor**(col, row) {

**this**.col = col;

**this**.row = row;

**this**.eastWall = **true**;

**this**.northWall = **true**;

**this**.southWall = **true**;

**this**.westWall = **true**;

**this**.visited = **false**;

}

}

**class** Maze {

**constructor**(cols, rows, cellSize) {

**this**.backgroundColor = "#ffffff";

**this**.cols = cols;

**this**.endColor = "#88FF88";

**this**.mazeColor = "#000000";

**this**.playerColor = "#880088";

**this**.rows = rows;

**this**.cellSize = cellSize;

**this**.cells = [];

**this**.generate()

}

generate() {

**const** count = localStorage.getItem('levels')

localStorage.setItem('levels', Number(count)+1)

mazeHeight = **this**.rows \* **this**.cellSize;

mazeWidth = **this**.cols \* **this**.cellSize;

canvas.height = mazeHeight;

canvas.width = mazeWidth;

canvas.style.height = mazeHeight;

canvas.style.width = mazeWidth;

**for** (**let** col = 0; col < **this**.cols; col++) {

**this**.cells[col] = [];

**for** (**let** row = 0; row < **this**.rows; row++) {

**this**.cells[col][row] = **new** MazeCell(col, row);

}

}

**let** rndCol = Math.floor(Math.random() \* **this**.cols);

**let** rndRow = Math.floor(Math.random() \* **this**.rows);

**let** stack = [];

stack.push(**this**.cells[rndCol][rndRow]);

**let** currCell;

**let** dir;

**let** foundNeighbor;

**let** nextCell;

**while** (**this**.hasUnvisited(**this**.cells)) {

currCell = stack[stack.length - 1];

currCell.visited = **true**;

**if** (**this**.hasUnvisitedNeighbor(currCell)) {

nextCell = **null**;

foundNeighbor = **false**;

**do** {

dir = Math.floor(Math.random() \* 4);

**switch** (dir) {

**case** 0:

**if** (currCell.col !== (**this**.cols - 1) && !**this**.cells[currCell.col + 1][currCell.row].visited) {

currCell.eastWall = **false**;

nextCell = **this**.cells[currCell.col + 1][currCell.row];

nextCell.westWall = **false**;

foundNeighbor = **true**;

}

**break**;

**case** 1:

**if** (currCell.row !== 0 && !**this**.cells[currCell.col][currCell.row - 1].visited) {

currCell.northWall = **false**;

nextCell = **this**.cells[currCell.col][currCell.row - 1];

nextCell.southWall = **false**;

foundNeighbor = **true**;

}

**break**;

**case** 2:

**if** (currCell.row !== (**this**.rows - 1) && !**this**.cells[currCell.col][currCell.row + 1].visited) {

currCell.southWall = **false**;

nextCell = **this**.cells[currCell.col][currCell.row + 1];

nextCell.northWall = **false**;

foundNeighbor = **true**;

}

**break**;

**case** 3:

**if** (currCell.col !== 0 && !**this**.cells[currCell.col - 1][currCell.row].visited) {

currCell.westWall = **false**;

nextCell = **this**.cells[currCell.col - 1][currCell.row];

nextCell.eastWall = **false**;

foundNeighbor = **true**;

}

**break**;

}

**if** (foundNeighbor) {

stack.push(nextCell);

}

} **while** (!foundNeighbor)

} **else** {

currCell = stack.pop();

}

}

**this**.redraw();

}

hasUnvisited() {

**for** (**let** col = 0; col < **this**.cols; col++) {

**for** (**let** row = 0; row < **this**.rows; row++) {

**if** (!**this**.cells[col][row].visited) {

**return** **true**;

}

}

}

**return** **false**;

}

hasUnvisitedNeighbor(mazeCell) {

**return** ((mazeCell.col !== 0 && !**this**.cells[mazeCell.col - 1][mazeCell.row].visited) ||

(mazeCell.col !== (**this**.cols - 1) && !**this**.cells[mazeCell.col + 1][mazeCell.row].visited) ||

(mazeCell.row !== 0 && !**this**.cells[mazeCell.col][mazeCell.row - 1].visited) ||

(mazeCell.row !== (**this**.rows - 1) && !**this**.cells[mazeCell.col][mazeCell.row + 1].visited));

}

redraw() {

ctx.fillStyle = **this**.backgroundColor;

ctx.fillRect(0, 0, mazeHeight, mazeWidth);

ctx.fillStyle = **this**.endColor;

ctx.fillRect((**this**.cols - 1) \* **this**.cellSize, (**this**.rows - 1) \* **this**.cellSize, **this**.cellSize, **this**.cellSize);

ctx.strokeStyle = **this**.mazeColor;

ctx.strokeRect(0, 0, mazeHeight, mazeWidth);

**for** (**let** col = 0; col < **this**.cols; col++) {

**for** (**let** row = 0; row < **this**.rows; row++) {

**if** (**this**.cells[col][row].eastWall) {

ctx.beginPath();

ctx.moveTo((col + 1) \* **this**.cellSize, row \* **this**.cellSize);

ctx.lineTo((col + 1) \* **this**.cellSize, (row + 1) \* **this**.cellSize);

ctx.stroke();

}

**if** (**this**.cells[col][row].northWall) {

ctx.beginPath();

ctx.moveTo(col \* **this**.cellSize, row \* **this**.cellSize);

ctx.lineTo((col + 1) \* **this**.cellSize, row \* **this**.cellSize);

ctx.stroke();

}

**if** (**this**.cells[col][row].southWall) {

ctx.beginPath();

ctx.moveTo(col \* **this**.cellSize, (row + 1) \* **this**.cellSize);

ctx.lineTo((col + 1) \* **this**.cellSize, (row + 1) \* **this**.cellSize);

ctx.stroke();

}

**if** (**this**.cells[col][row].westWall) {

ctx.beginPath();

ctx.moveTo(col \* **this**.cellSize, row \* **this**.cellSize);

ctx.lineTo(col \* **this**.cellSize, (row + 1) \* **this**.cellSize);

ctx.stroke();

}

}

}

ctx.fillStyle = **this**.playerColor;

ctx.fillRect((player.col \* **this**.cellSize) + 2, (player.row \* **this**.cellSize) + 2, **this**.cellSize - 4, **this**.cellSize - 4);

}

}

**function** onClick(event) {

maze.cols = document.getElementById("cols").value;

maze.rows = document.getElementById("rows").value;

**const** name = document.getElementById("name").value;

localStorage.setItem('name', name)

localStorage.setItem('levels', 0)

maze.generate();

}

**function** onRestart(event) {

window.location.replace('file:///Users/constantineoganesyants/Documents/GitHub/DSTU/3%20Course/2%20Semester/Web-tech/Lab2/index.html')

}

**function** onKeyDown(event) {

**let** cols = document.getElementById("cols").value;

**let** rows = document.getElementById("rows").value;

**if** (player.row + 1 === Number(cols) && player.col + 1 === Number(rows)) {

player.col = 0;

player.row = 0;

maze.generate();

}

**switch** (event.keyCode) {

**case** 37:

**case** 65:

**if** (!maze.cells[player.col][player.row].westWall) {

player.col -= 1;

}

**break**;

**case** 39:

**case** 68:

**if** (!maze.cells[player.col][player.row].eastWall) {

player.col += 1;

}

**break**;

**case** 40:

**case** 83:

**if** (!maze.cells[player.col][player.row].southWall) {

player.row += 1;

}

**break**;

**case** 38:

**case** 87:

**if** (!maze.cells[player.col][player.row].northWall) {

player.row -= 1;

}

**break**;

**default**:

**break**;

}

maze.redraw();

}

**function** restartPage() {

document.getElementById("restart").addEventListener("click", onRestart);

}

**function** onLoad() {

**var** timeLeft = 10;

**var** elem = document.getElementById('some\_div');

**var** timerId = setInterval(countdown, 1000);

**function** countdown() {

**if** (timeLeft === 0){

**const** finalName = localStorage.getItem('name');

**const** finalLevels = localStorage.getItem('levels');

**const** leaders = localStorage.getItem('leaders');

localStorage.setItem('leaders', leaders.concat(`${finalName}: ${finalLevels};`))

window.location.replace('file:///Users/constantineoganesyants/Documents/GitHub/DSTU/3%20Course/2%20Semester/Web-tech/Lab2/endGame.html')

} **else** {

elem.innerHTML = timeLeft + ' seconds remaining';

timeLeft--;

}

*// if (timeLeft === 0){*

*// const finalName = localStorage.getItem('name');*

*// const finalLevels = localStorage.getItem('level');*

*// const leaders = localStorage.getItem('leaders');*

*// localStorage.setItem([leaders.split(';'), ...`${finalName}: ${finalLevels};`])*

*// console.log(data);*

*// window.location.replace('file:///Users/constantineoganesyants/Documents/GitHub/DSTU/3%20Course/2%20Semester/Web-tech/Lab2/endGame.html')*

*// }*

}

canvas = document.getElementById("mainForm");

ctx = canvas.getContext("2d");

player = **new** Player();

maze = **new** Maze(10, 10, 25);

document.addEventListener("keydown", onKeyDown);

document.getElementById("generate").addEventListener("click", onClick);

}

**EndGame.html**

<!**DOCTYPE** html>

<**html** lang="en">

<**head**>

<**meta** charset="UTF-8">

<**link** rel="stylesheet" href="style.css">

<**meta** http-equiv="X-UA-Compatible" content="IE=edge">

<**meta** name="viewport" content="width=device-width, initial-scale=1.0">

<**script** src="./script.js"></**script**>

<**script** src="./endGame.js"></**script**>

<**title**>Looser</**title**>

</**head**>

<**body** onload="endGame();">

<**h1**>Try again!</**h1**>

<**table**>

<**tbody** class="leadersTable">

</**tbody**>

</**table**>

<**input** id="restart" type="button" value="Restart" onclick="restartPage()" /></**td**>

</**body**>

</**html**>

**Style.css**

\* {

font-family: Arial, sans-serif;

text-align: center;

}

table, tbody, tr, td {

text-align: left;

}

#footer {

bottom: 25px;

left: 25px;

position: absolute;

right: 25px;

}

#settings {

position: absolute;

right: 25px;

top: 25px;

}

**Style.css**

**@font-face** {

font-family: 'Sansita Swashed';

src: URL('SansitaSwashed-VariableFont\_wght.ttf') format('truetype');

}

.left\_menu {

font-family: 'Sansita Swashed';

float:left;

width:24%;

margin-left:1%;

padding-bottom:55px;

}

.main {

font-family: 'Sansita Swashed';

float:right; width:74%;padding-bottom:55px;

}

.Main\_text{

font-family: 'Sansita Swashed';

front-size:14pt

}

footer {

font-family: 'Sansita Swashed';

background:#DCAE7D;

position: fixed;

left : 0 ; bottom : 0;

width:100%;

}

body, h1{

font-family: 'Sansita Swashed';

margin: 0;

padding:0;

}

html {

font-family: 'Sansita Swashed';

height: 100%;

margin: 0;

padding: 0;

background: #F6D7B4;

}

header {

font-family: 'Sansita Swashed';

width: 100%;

height: 400px;

margin: 0;

padding: 0;

background: url(https://www.heathrow.com/content/dam/heathrow/web/common/images/hero/desktop/at-the-airport/shops-a-z/cartier-1920x624.jpg/\_jcr\_content/renditions/cq5dam.web.1680.624.jpeg) center no-repeat;

background-size: cover;

}

.site-title {

font-family: 'Sansita Swashed';

text-align: center;

text-transform: uppercase;

}

table {

font-family: 'Sansita Swashed';

border: 1px solid #F6D7B4;width:100%;

}