**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(МИИТ)**

|  |
| --- |
| **ИНСТИТУТ ТРАНСПОРТНОЙ ТЕХНИКИ И СИСТЕМ УПРАВЛЕНИЯ**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Кафедра «Управление и защита информации»

Лабораторная работа №3

**«приложение под управлением JavaScript»**

по дисциплине

**«Web программирование»**

|  |  |
| --- | --- |
|  | Выполнил: студент группы ТКИ-541  Куминов В. П.  Проверил: к.т.н., доц.  Сафронов А. И. |

**Москва – 2023 г.**

**1. Цель:**

Построить приложение под управлением Java Script.

**2. Постановка задачи:**

Разработать типовое или узкоспециализированное небольшое одностраничное web-приложение под управлением «чистым» JavaScript, для которого уместна компактная карта технологического процесса, изображаемая в нотации Сетей Петри.

Базовая структура отчёта по работе: 1. Цель работы. 2. Формулировка задачи. 3. Содержательная часть (код web-приложения + отображение в браузере + Сеть Петри). 4. Вывод.

Рекомендации: разработка web-приложения, в первую очередь, должна быть интересна и актуальна в использовании самим автором (должна иметь место высокая мотивация).

Примеры: выполняет типовой расчёт по одной из учебных дисциплин текущего семестра, рассчитывает один из фрагментов или целиком курсовую работу / курсовой проект предыдущего или текущего семестра, моделирует работу локальной сети, помогает в выполнении рутинной работы на производстве (в случае трудоустройства), помогает друзьям и/или коллегам в выполнении каких-либо аспектов их трудовой деятельности или сферы личных увлечений.

**3. Ход работы**

Построим сеть Петри:

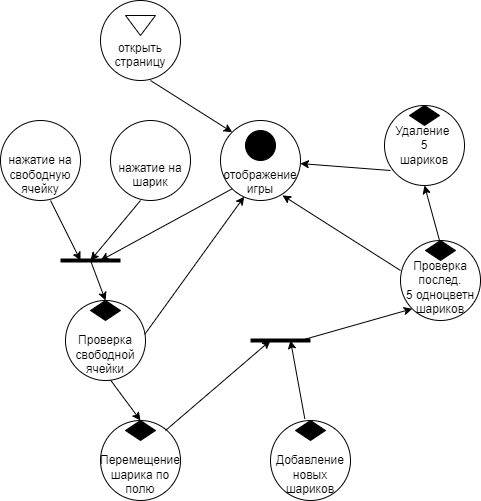


Рисунок 1. Сеть Петри

**Код программы:**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Шарики</title>

<style>

canvas {

border: 1px solid black;

}

</style>

</head>

<body>

<h1>Шарики</h1>

<p>Цель игры: построить 5 шаров в ряд или по диагонали одного цвета, передвигая по одному шару за ход по свободным ячейкам. После хода на поле в рандомном месте появляются ещё 3 шарика различных цветов, которые перед ходом изображены как потенциально возможные, но не перекрывающие клетку. Если собрано 5 шаров одного цвета даётся 10 очков. Игра заканчивается, когда на поле не остаётся свободных ходов.</p>

<p>Очки: <span id="score">0</span></p>

<canvas id="canvas" width="450" height="450"></canvas>

<script>

// Константы

const ROWS = 9; // Количество строк

const COLS = 9; // Количество столбцов

const SIZE = 50; // Размер клетки в пикселях

const COLORS = ["red", "orange", "yellow", "green", "blue", "indigo", "violet"]; // Цвета шариков

const BALLS = 5; // Количество шариков изначально

const NEW\_BALLS = 3; // Количество новых шариков после хода

const LINE = 5; // Количество шариков в ряд для выигрыша

const POINTS = 10; // Количество очков за линию

// Переменные

let canvas = document.getElementById("canvas"); // Элемент холста

let ctx = canvas.getContext("2d"); // Контекст рисования

let score = document.getElementById("score"); // Элемент счета

let grid = []; // Матрица поля

let selected = null; // Выбранный шарик

let nextBalls = []; // Следующие шарики

let gameOver = false; // Флаг окончания игры

// Функция для создания пустого поля

function createGrid() {

for (let i = 0; i < ROWS; i++) {

grid[i] = [];

for (let j = 0; j < COLS; j++) {

grid[i][j] = null;

}

}

}

// Функция для рисования сетки

function drawGrid() {

ctx.strokeStyle = "black";

ctx.lineWidth = 1;

for (let i = 0; i <= ROWS; i++) {

ctx.beginPath();

ctx.moveTo(0, i \* SIZE);

ctx.lineTo(COLS \* SIZE, i \* SIZE);

ctx.stroke();

}

for (let j = 0; j <= COLS; j++) {

ctx.beginPath();

ctx.moveTo(j \* SIZE, 0);

ctx.lineTo(j \* SIZE, ROWS \* SIZE);

ctx.stroke();

}

}

// Функция для рисования шарика в клетке (i, j)

function drawBall(i, j) {

let color = grid[i][j];

if (color) {

ctx.fillStyle = color;

ctx.beginPath();

ctx.arc(j \* SIZE + SIZE / 2, i \* SIZE + SIZE / 2, SIZE / 3, 0, Math.PI \* 2);

ctx.fill();

}

}

// Функция для рисования всех шариков на поле

function drawBalls() {

for (let i = 0; i < ROWS; i++) {

for (let j = 0; j < COLS; j++) {

drawBall(i, j);

}

}

}

// Функция для рисования выделенного шарика

function drawSelected() {

if (selected) {

let i = selected.i;

let j = selected.j;

ctx.strokeStyle = "white";

ctx.lineWidth = 3;

ctx.beginPath();

ctx.arc(j \* SIZE + SIZE / 2, i \* SIZE + SIZE / 2, SIZE / 3, 0, Math.PI \* 2);

ctx.stroke();

}

}

// Функция для рисования следующих шариков

function drawNextBalls() {

for (let k = 0; k < nextBalls.length; k++) {

let color = nextBalls[k];

ctx.fillStyle = color;

ctx.beginPath();

ctx.arc((COLS + 1) \* SIZE + SIZE / 2, (k + 1) \* SIZE + SIZE / 2, SIZE / 3, 0, Math.PI \* 2);

ctx.fill();

}

}

// Функция для рисования всего поля

function drawField() {

ctx.clearRect(0, 0, canvas.width, canvas.height); // Очистить холст

drawGrid(); // Рисовать сетку

drawBalls(); // Рисовать шарики

drawSelected(); // Рисовать выделенный шарик

drawNextBalls(); // Рисовать следующие шарики

}

// Функция для получения случайного цвета из массива цветов

function getRandomColor() {

let index = Math.floor(Math.random() \* COLORS.length);

return COLORS[index];

}

// Функция для добавления случайных шариков на поле

function addRandomBalls(n) {

let emptyCells = []; // Массив пустых клеток

for (let i = 0; i < ROWS; i++) {

for (let j = 0; j < COLS; j++) {

if (grid[i][j] == null) {

emptyCells.push({i: i, j: j}); // Добавить пустую клетку в массив

}

}

}

if (emptyCells.length == 0) { // Если нет пустых клеток

gameOver = true; // Игра окончена

return;

}

for (let k = 0; k < n; k++) { // Добавить n шариков

if (emptyCells.length > 0) { // Если есть пустые клетки

let index = Math.floor(Math.random() \* emptyCells.length); // Выбрать случайный индекс из массива пустых клеток

let cell = emptyCells[index]; // Получить координаты пустой клетки

grid[cell.i][cell.j] = getRandomColor(); // Добавить случайный цвет в пустую клетку

emptyCells.splice(index, 1); // Удалить пустую клетку из массива

}

}

}

// Функция для проверки, есть ли линия из n шариков одного цвета по горизонтали, вертикали или диагонали

function checkLine(n) {

let points = 0; // Количество очков за линию

for (let i = 0; i < ROWS; i++) { // Проверить все строки

let count = 1; // Счетчик одинаковых цветов подряд

for (let j = 1; j < COLS; j++) {

if (grid[i][j] != null && grid[i][j] == grid[i][j - 1]) { // Если текущий цвет совпадает с предыдущим в строке

count++; // Увеличить счетчик на единицу

} else { // Иначе

count = 1; // Сбросить счетчик на единицу

}

if (count >= n) { // Если счетчик достиг n или больше

points += POINTS; // Добавить очки за линию к общему счету

for (let k = 0; k < n; k++) { // Удалить n шариков из линии

grid[i][j - k] = null; // Установить цвет клетки в null

}

}

}

}

for (let j = 0; j < COLS; j++) { // Проверить все столбцы

let count = 1; // Счетчик одинаковых цветов подряд

for (let i = 1; i < ROWS; i++) {

if (grid[i][j] != null && grid[i][j] == grid[i - 1][j]) { // Если текущий цвет совпадает с предыдущим в столбце

count++; // Увеличить счетчик на единицу

} else { // Иначе

count = 1; // Сбросить счетчик на единицу

}

if (count >= n) { // Если счетчик достиг n или больше

points += POINTS; // Добавить очки за линию к общему счету

for (let k = 0; k < n; k++) { // Удалить n шариков из линии

grid[i - k][j] = null; // Установить цвет клетки в null

}

}

}

}

for (let d = -(ROWS - n); d <= COLS - n; d++) { // Проверить все главные диагонали

let count = 1; // Счетчик одинаковых цветов подряд

for (let i = Math.max(0, -d) + 1, j = Math.max(0, d) + 1; i < ROWS && j < COLS; i++, j++) {

if (grid[i][j] != null && grid[i][j] == grid[i - 1][j - 1]) { // Если текущий цвет совпадает с предыдущим по диагонали

count++; // Увеличить счетчик на единицу

} else { // Иначе

count = 1; // Сбросить счетчик на единицу

}

if (count >= n) { // Если счетчик достиг n или больше

points += POINTS; // Добавить очки за линию к общему счету

for (let k = 0; k < n; k++) { // Удалить n шариков из линии

grid[i - k][j - k] = null; // Установить цвет клетки в null

}

}

}

}

for (let d = -(ROWS - n); d <= COLS - n; d++) { // Проверить все побочные диагонали

let count = 1; // Счетчик одинаковых цветов подряд

for (let i = Math.max(0, -d) + 1, j = Math.min(COLS - 1, COLS - d) - 1; i < ROWS && j >= 0; i++, j--) {

if (grid[i][j] != null && grid[i][j] == grid[i - 1][j + 1]) { // Если текущий цвет совпадает с предыдущим по диагонали

count++; // Увеличить счетчик на единицу

} else { // Иначе

count = 1; // Сбросить счетчик на единицу

}

if (count >= n) { // Если счетчик достиг n или больше

points += POINTS; // Добавить очки за линию к общему счету

for (let k = 0; k < n; k++) { // Удалить n шариков из линии

grid[i - k][j + k] = null; // Установить цвет клетки в null

}

}

}

}

return points;

}

// Функция для перемещения шарика из клетки (i1, j1) в клетку (i2, j2)

function moveBall(i1, j1, i2, j2) {

if (grid[i1][j1] != null && grid[i2][j2] == null) { // Если есть шарик в клетке (i1, j1) и нет шарика в клетке (i2, j2)

let color = grid[i1][j1]; // Запомнить цвет шарика

grid[i1][j1] = null; // Удалить шарик из клетки (i1, j1)

grid[i2][j2] = color; // Добавить шарик в клетку (i2, j2)

}

}

// Функция для проверки, есть ли путь из клетки (i1, j1) в клетку (i2, j2)

function checkPath(i1, j1, i2, j2) {

let visited = []; // Матрица посещенных клеток

for (let i = 0; i < ROWS; i++) {

visited[i] = [];

for (let j = 0; j < COLS; j++) {

visited[i][j] = false;

}

}

let queue = []; // Очередь для обхода в ширину

queue.push({i: i1, j: j1}); // Добавить начальную клетку в очередь

visited[i1][j1] = true; // Пометить начальную клетку как посещенную

while (queue.length > 0) { // Пока очередь не пуста

let cell = queue.shift(); // Извлечь первую клетку из очереди

let i = cell.i;

let j = cell.j;

if (i == i2 && j == j2) { // Если достигнута конечная клетка

return true; // Вернуть true

}

let neighbors = [ // Массив соседних клеток

{i: i - 1, j: j},

{i: i + 1, j: j},

{i: i, j: j - 1},

{i: i, j: j + 1}

];

for (let k = 0; k < neighbors.length; k++) { // Для каждой соседней клетки

let ni = neighbors[k].i;

let nj = neighbors[k].j;

if (ni >= 0 && ni < ROWS && nj >= 0 && nj < COLS && grid[ni][nj] == null && !visited[ni][nj]) { // Если клетка в пределах поля, пустая и не посещенная

queue.push({i: ni, j: nj}); // Добавить клетку в очередь

visited[ni][nj] = true; // Пометить клетку как посещенную

}

}

}

return false; // Вернуть false

}

// Функция для обработки клика мыши на холсте

function handleClick(event) {

if (!gameOver) { // Если игра не окончена

let x = event.offsetX; // Получить координату x клика относительно холста

let y = event.offsetY; // Получить координату y клика относительно холста

let i = Math.floor(y / SIZE); // Получить номер строки по координате y

let j = Math.floor(x / SIZE); // Получить номер столбца по координате x

if (i >= 0 && i < ROWS && j >= 0 && j < COLS) { // Если клик был в пределах поля

if (grid[i][j] != null) { // Если в клетке есть шарик

selected = {i: i, j: j}; // Запомнить выбранный шарик

} else if (selected != null && checkPath(selected.i, selected.j, i, j)) { // Иначе если есть выбранный шарик и есть путь до пустой клетки

moveBall(selected.i, selected.j, i, j); // Переместить выбранный шарик в пустую клетку

selected = null; // Сбросить выбранный шарик

let points = checkLine(LINE); // Проверить, есть ли линии из LINE шариков одного цвета

if (points > 0) { // Если есть линии

score.textContent = Number(score.textContent) + points; // Добавить очки к счету

} else { // Иначе

addRandomBalls(NEW\_BALLS); // Добавить NEW\_BALLS новых шариков на поле

checkLine(LINE); // Проверить, есть ли линии из LINE шариков одного цвета

}

}

}

drawField(); // Перерисовать поле

if (gameOver) { // Если игра окончена

alert("Игра окончена! Ваш счет: " + score.textContent); // Вывести сообщение с счетом

}

}

}

// Функция для инициализации игры

function initGame() {

createGrid(); // Создать пустое поле

addRandomBalls(BALLS); // Добавить BALLS случайных шариков на поле

for (let k = 0; k < NEW\_BALLS; k++) { // Сгенерировать NEW\_BALLS следующих шариков

nextBalls[k] = getRandomColor();

}

drawField(); // Нарисовать поле

canvas.addEventListener("click", handleClick); // Добавить обработчик клика мыши на холсте

}

initGame(); // Запустить игру

</script>

</body>

</html>

Результат работы кода можно увидеть на рисунках 2–5.

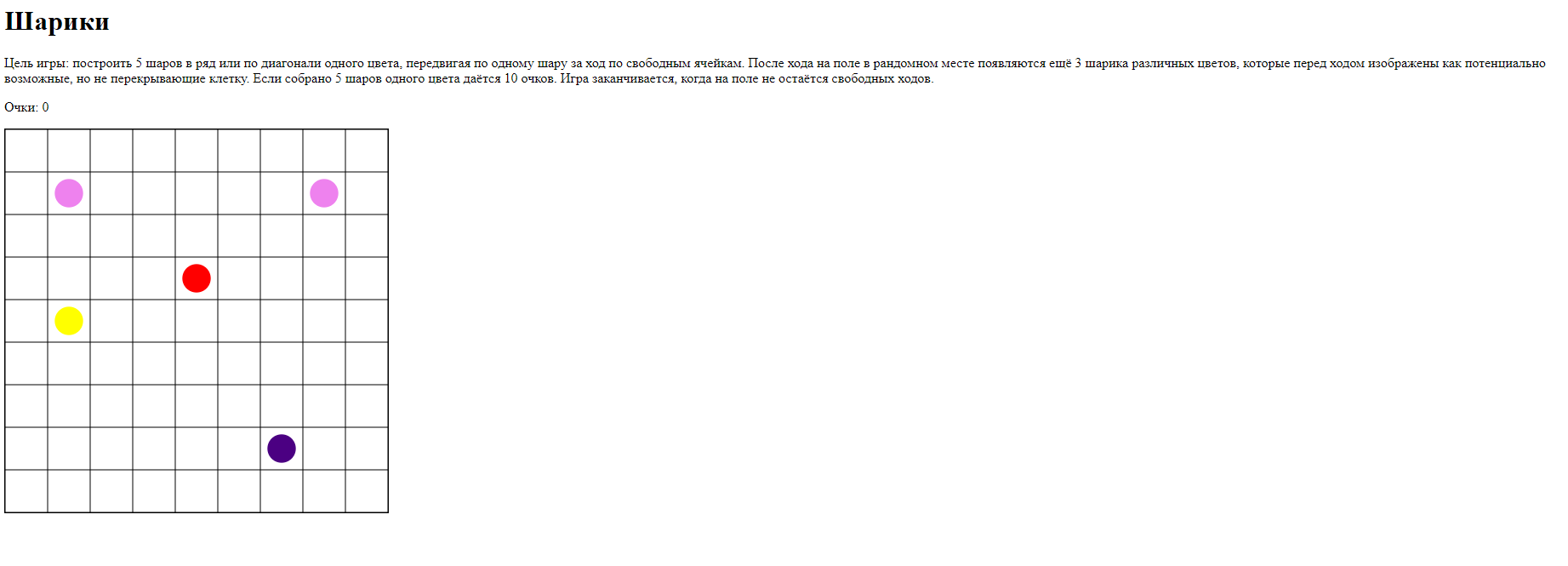


Рисунок 2. Начало игры.

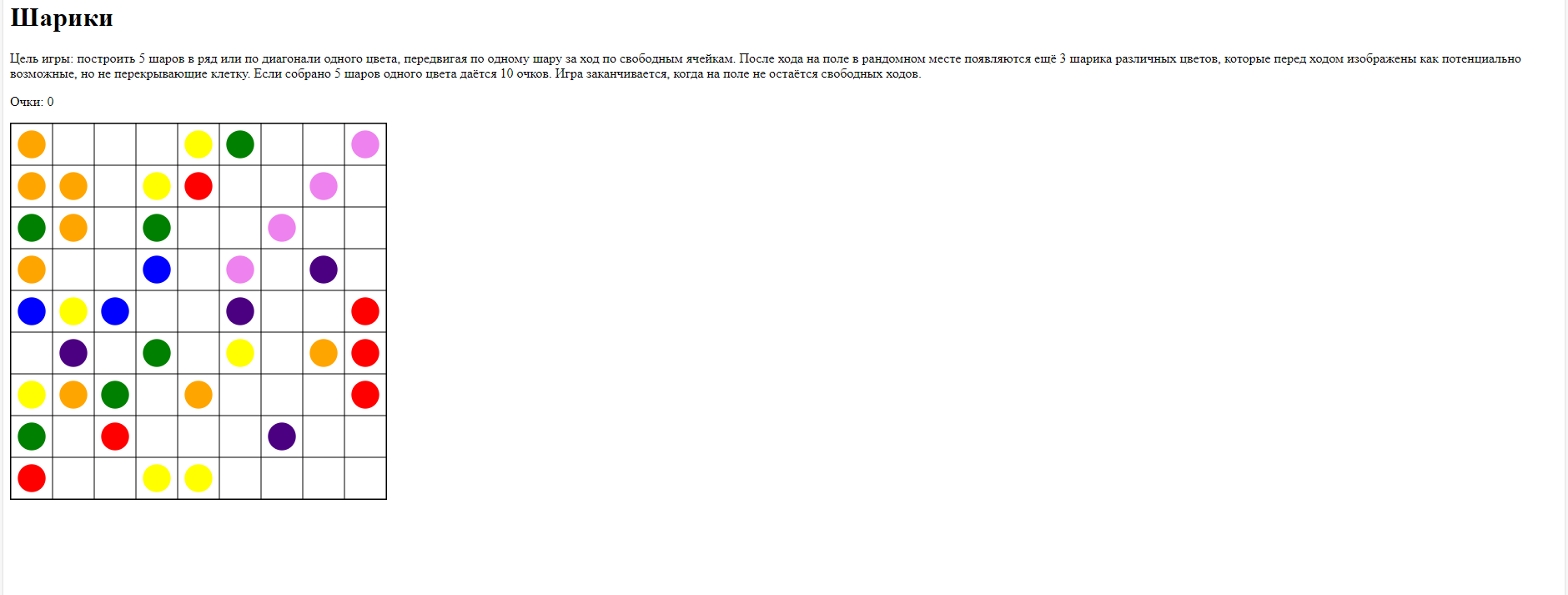


Рисунок 3. 4 шара розового цвета ждут 5-й.

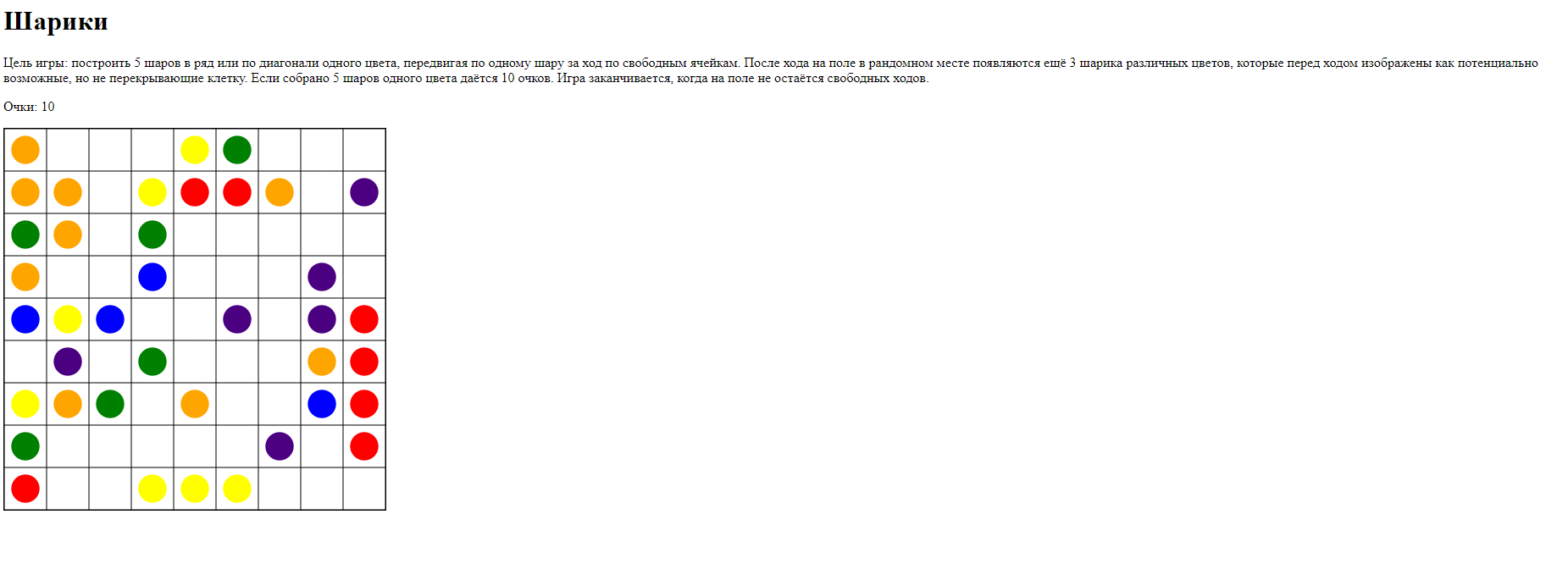


Рисунок 4. Убрали 5 шаров розового цвета

**4. Вывод**

В ходе выполнения задачи, было разработано приложение под управлением Java-Script, который используется для добавления интерактивных элементов на страницу. Это обогащает пользовательский опыт и делает страницу более функциональной.