

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ**)

Факультет «Информатика и вычислительная техника»  
Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Отчёт по лабораторной работе № 2

«Игра Тетрис»

Вариант № 11

Выполнил:

Студент группы ВМО11

Дышко Н. А.

Принял:

Попова О.А.

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

(оценка) (дата) (подпись)

Ростов-на-Дону

2024

Цель работы:

Создать простую игру с помощью JavaScript.

В игре должны присутствовать следующие элементы:

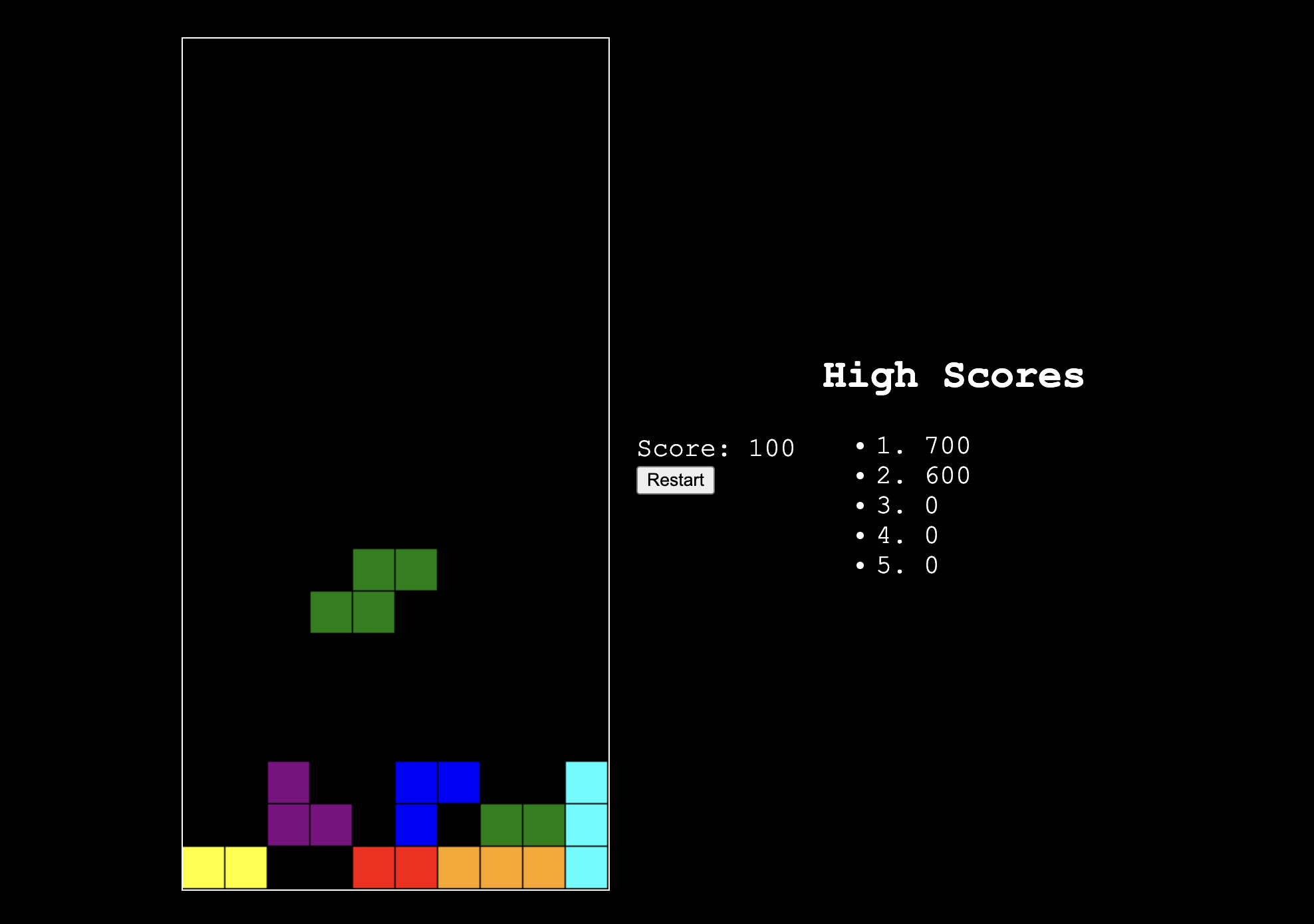
-Кнопки: новая игра, таблица рекордов.

-Подсчет очков (и/или времени игры) и вывод их по завершению игры, если очков набрано достаточно для попадания в таблицу рекордов - спросить у пользователя его имя.

-Ведение таблицы рекордов: таблица из 10 лучших партий в порядке убывания (имя - очки и/или время игры). Данные хранить в Cookie. -картинка в шапке сайта.

-Изменение уровней сложности ("\*" в номере - игра не предполагает уровней сложности).

-НИКАКОЙ ASCII-графики!

На рисунке представлен скриншот главной страницы игры.

В листинг 1 представлен код главной страницы игры – файл Tetris.html,

Листинг – Код html, CSS, Javascript

<!DOCTYPE html>

<html>

<head>

<title>Тетрис</title>

<style>

html, body {

height: 100%;

margin: 0;

}

body {

background: black;

display: flex;

align-items: center;

justify-content: center;

}

canvas {

border: 1px solid white;

}

#score-container {

color: white;

font-family: monospace;

font-size: 20px;

margin-left: 20px;

}

#highscores {

color: white;

font-family: monospace;

font-size: 20px;

margin-left: 20px;

}

</style>

</head>

<body>

<canvas width="320" height="640" id="game"></canvas>

<div id="score-container">

<div>Score: <span id="score">0</span></div>

<button onclick="restartGame()">Restart</button>

</div>

<div id="highscores">

<h2>High Scores</h2>

<ul id="highscore-list"></ul>

</div>

<script>

const canvas = document.getElementById('game');

const context = canvas.getContext('2d');

const scoreElement = document.getElementById('score');

const highscoreList = document.getElementById('highscore-list');

const grid = 32;

let tetrominoSequence = [];

let score = 0;

let playfield = [];

for (let row = -2; row < 20; row++) {

playfield[row] = [];

for (let col = 0; col < 10; col++) {

playfield[row][col] = 0;

}

}

const tetrominos = {

'I': [

[0,0,0,0],

[1,1,1,1],

[0,0,0,0],

[0,0,0,0]

],

'J': [

[1,0,0],

[1,1,1],

[0,0,0],

],

'L': [

[0,0,1],

[1,1,1],

[0,0,0],

],

'O': [

[1,1],

[1,1],

],

'S': [

[0,1,1],

[1,1,0],

[0,0,0],

],

'Z': [

[1,1,0],

[0,1,1],

[0,0,0],

],

'T': [

[0,1,0],

[1,1,1],

[0,0,0],

]

};

const colors = {

'I': 'cyan',

'O': 'yellow',

'T': 'purple',

'S': 'green',

'Z': 'red',

'J': 'blue',

'L': 'orange'

};

let count = 0;

let tetromino = getNextTetromino();

let rAF = null;

let gameOver = false;

let dropCounter = 0;

let dropInterval = 1000;

const speedIncreaseInterval = 10000;

let highscores = JSON.parse(localStorage.getItem('highscores')) || [];

function getRandomInt(min, max) {

min = Math.ceil(min);

max = Math.floor(max);

return Math.floor(Math.random() \* (max - min + 1)) + min;

}

function generateSequence() {

const sequence = ['I', 'J', 'L', 'O', 'S', 'T', 'Z'];

while (sequence.length) {

const rand = getRandomInt(0, sequence.length - 1);

const name = sequence.splice(rand, 1)[0];

tetrominoSequence.push(name);

}

}

function getNextTetromino() {

if (tetrominoSequence.length === 0) {

generateSequence();

}

const name = tetrominoSequence.pop();

const matrix = tetrominos[name];

const col = playfield[0].length / 2 - Math.ceil(matrix[0].length / 2);

const row = name === 'I' ? -1 : -2;

return {

name: name,

matrix: matrix,

row: row,

col: col

};

}

function rotate(matrix) {

const N = matrix.length - 1;

const result = matrix.map((row, i) =>

row.map((val, j) => matrix[N - j][i])

);

return result;

}

function isValidMove(matrix, cellRow, cellCol) {

for (let row = 0; row < matrix.length; row++) {

for (let col = 0; col < matrix[row].length; col++) {

if (matrix[row][col] && (

cellCol + col < 0 ||

cellCol + col >= playfield[0].length ||

cellRow + row >= playfield.length ||

playfield[cellRow + row][cellCol + col])

) {

return false;

}

}

}

return true;

}

function placeTetromino() {

for (let row = 0; row < tetromino.matrix.length; row++) {

for (let col = 0; col < tetromino.matrix[row].length; col++) {

if (tetromino.matrix[row][col]) {

if (tetromino.row + row < 0) {

return showGameOver();

}

playfield[tetromino.row + row][tetromino.col + col] = tetromino.name;

}

}

}

for (let row = playfield.length - 1; row >= 0; ) {

if (playfield[row].every(cell => !!cell)) {

for (let r = row; r >= 0; r--) {

for (let c = 0; c < playfield[r].length; c++) {

playfield[r][c] = playfield[r-1][c];

}

}

score += 100;

scoreElement.textContent = score;

} else {

row--;

}

}

tetromino = getNextTetromino();

}

function showGameOver() {

cancelAnimationFrame(rAF);

gameOver = true;

context.fillStyle = 'black';

context.globalAlpha = 0.75;

context.fillRect(0, canvas.height / 2 - 30, canvas.width, 60);

context.globalAlpha = 1;

context.fillStyle = 'white';

context.font = '36px monospace';

context.textAlign = 'center';

context.textBaseline = 'middle';

context.fillText('GAME OVER!', canvas.width / 2, canvas.height / 2);

saveHighScore(score);

displayHighScores();

}

function loop(timestamp) {

rAF = requestAnimationFrame(loop);

context.clearRect(0,0,canvas.width,canvas.height);

for (let row = 0; row < 20; row++) {

for (let col = 0; col < 10; col++) {

if (playfield[row][col]) {

const name = playfield[row][col];

context.fillStyle = colors[name];

context.fillRect(col \* grid, row \* grid, grid-1, grid-1);

}

}

}

if (tetromino) {

if (timestamp - dropCounter > dropInterval) {

tetromino.row++;

dropCounter = timestamp;

if (!isValidMove(tetromino.matrix, tetromino.row, tetromino.col)) {

tetromino.row--;

placeTetromino();

}

}

context.fillStyle = colors[tetromino.name];

for (let row = 0; row < tetromino.matrix.length; row++) {

for (let col = 0; col < tetromino.matrix[row].length; col++) {

if (tetromino.matrix[row][col]) {

context.fillRect((tetromino.col + col) \* grid, (tetromino.row + row) \* grid, grid-1, grid-1);

}

}

}

}

}

document.addEventListener('keydown', function(e) {

if (gameOver) return;

if (e.which === 37 || e.which === 39) {

const col = e.which === 37

? tetromino.col - 1

: tetromino.col + 1;

if (isValidMove(tetromino.matrix, tetromino.row, col)) {

tetromino.col = col;

}

}

if (e.which === 38) {

const matrix = rotate(tetromino.matrix);

if (isValidMove(matrix, tetromino.row, tetromino.col)) {

tetromino.matrix = matrix;

}

}

if(e.which === 40) {

const row = tetromino.row + 1;

if (!isValidMove(tetromino.matrix, row, tetromino.col)) {

tetromino.row = row - 1;

placeTetromino();

return;

}

tetromino.row = row;

}

if(e.which === 32) {

while (isValidMove(tetromino.matrix, tetromino.row + 1, tetromino.col)) {

tetromino.row++;

}

placeTetromino();

}

});

setInterval(() => {

if (dropInterval > 100) {

dropInterval -= 100;

}

}, speedIncreaseInterval);

function saveHighScore(score) {

highscores.push(score);

highscores.sort((a, b) => b - a);

highscores = highscores.slice(0, 5); // Keep only top 5 scores

localStorage.setItem('highscores', JSON.stringify(highscores));

}

function displayHighScores() {

highscoreList.innerHTML = '';

highscores.forEach((highscore, index) => {

const li = document.createElement('li');

li.textContent = `${index + 1}. ${highscore}`;

highscoreList.appendChild(li);

});

}

function restartGame() {

playfield = [];

for (let row = -2; row < 20; row++) {

playfield[row] = [];

for (let col = 0; col < 10; col++) {

playfield[row][col] = 0;

}

}

tetromino = getNextTetromino();

score = 0;

scoreElement.textContent = score;

gameOver = false;

dropInterval = 1000;

dropCounter = 0;

rAF = requestAnimationFrame(loop);

}

displayHighScores();

rAF = requestAnimationFrame(loop);

</script>

</body>

</html>