

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №1
по дисциплине «Web-технологии»
Тема: Тетрис на JavaScript

Студент гр. 3343

Пименов П.В.

Преподаватель

Беляев С.А.

Санкт-Петербург

2025

Задание

Целью работы является изучение работы web-сервера nginx со статическими файлами и создание клиентских JavaScript web-приложений.

Для достижения поставленной цели требуется решить следующие задачи:

- генерация открытого и закрытого ключей для использования шифрования (<https://www.openssl.org/>);
- настройка сервера nginx для работы по протоколу HTTPS;
- разработка интерфейса web-приложения;
- обеспечение ввода имени пользователя;
- обеспечение создания новой фигуры для тетриса по таймеру и ее движение;
- обеспечение управления пользователем падающей фигурой;
- обеспечение исчезновения ряда, если он заполнен;
- по окончании игры – отображение таблицы рекордов, которая хранится в браузере пользователя.

Выполнение работы

Файл `src/index.html` описывает структуру страницы с игрой. Файл `src/style.css` задаёт стили для элементов из `src/index.html`.

`src/main.js`:

В начале файла задаются различные игровые константы (`cols`, `rows`, `cell`, etc) и получает элементы страницы через `document.getElementById()`

Функция `createPiece()` создает новое случайное тетрамино (возвращает словарь с его параметрами: форма и стартовые координаты).

Функция `startGame()` сбрасывает переменные состояния в начальные значения и начинает новую игру.

Функция `draw()` отрисовывает текущее состояние игры на холст (`canvas`).

Функция `drawCell()` отрисовывает клетку, из которых состоит игровое поле, на холсте.

Функция `drawGhost()` отрисовывает вспомогательное тетрамино, показывая пользователю, куда «приземлится» настоящее тетрамино.

Функция `drawNextPiece()` отрисовывает окошко с следующим тетрамино.

Функция `collides()` проверяет, будет ли находится тетрамино при применении сдвига внутри игрового поля и будет ли сталкиваться с уже поставленными тетрамино.

Функция `mergePiece()` «встраивает» падающее тетрамино в уже поставленные, проверяет окончена ли игра (проигрыш) и обновляет заполненные линии (вызывая `clearLines`).

Функция `clearLines()` обновляет заполненные линии и выдает игровые очки пользователю.

Функция `rotate()` осуществляет поворот текущего тетрамино на 90 градусов по часовой стрелке.

Функция `hardDrop()` осуществляет резкое «падение» вниз текущего тетрамино.

Функция `updateScore()` обновляет значение текстового поля с игровыми очками.

Функция `updateLevel()` проверяет, нужно ли обновить уровень игры, и если нужно, то применяет обновление (быстрее падают фигуры).

Функция `startLoop()` запускает игровой цикл (таймер).

Функция `stopLoop()` останавливает игровой таймер.

Функция `addLeaderboard()` загружает таблицу очков из `local storage`.

Функция `renderLeaderboard()` отрисовывает таблицу очков.

Функция `escapeHtml()` является вспомогательной функцией, которая экранирует специальные HTML-символы в строке.

В конце файла добавлены `event listeners` к кнопке начала игры и к странице для организации корректного начала игры и обработки нажатия кнопок управления. Также загружается таблица очков.

nginx.conf:

Файл содержит конфигурацию `nginx`, написанную согласно требованиям лабораторной работы (доступ к странице по `https` и порту `443`). Также отключено кэширование файлов игры.

Dockerfile:

Файл содержит инструкции по сборке образа `nginx` с тетрисом. Используется базовый образ `nginx:alpine`.

docker-compose.yml:

Файл описывает инфраструктуру для запуска приложения (порты, сертификаты).

Сертификаты для `localhost` сгенерированы с помощью утилиты `mkcert`, установлены ей же в систему и переданы в `nginx`.

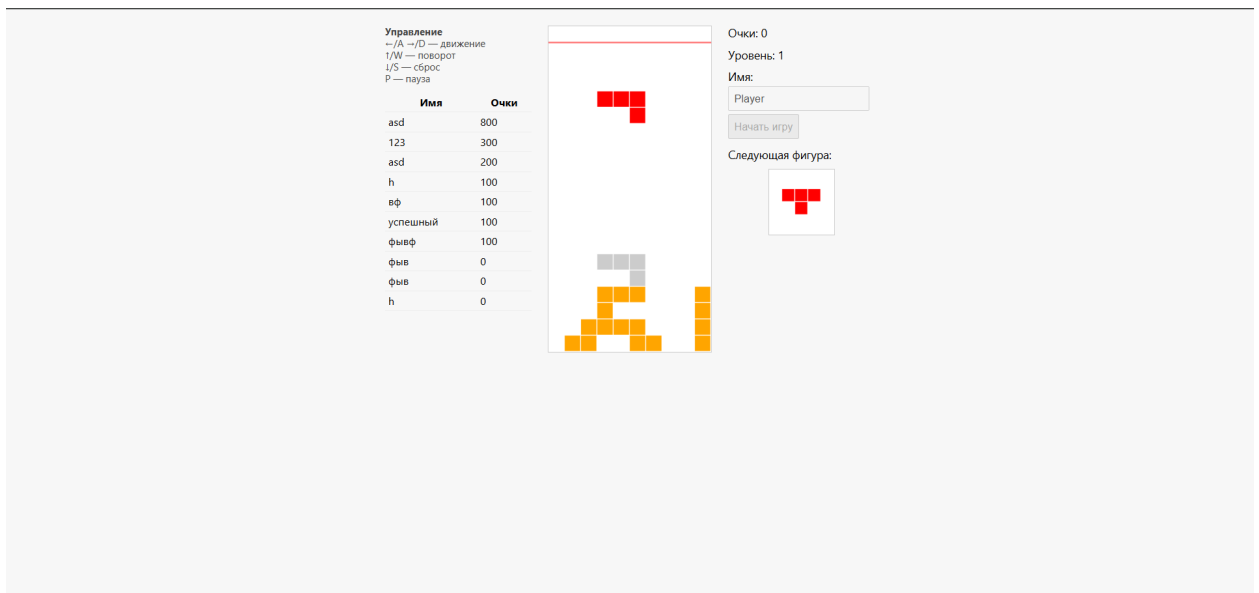


Рисунок 1 – Интерфейс страницы с игрой

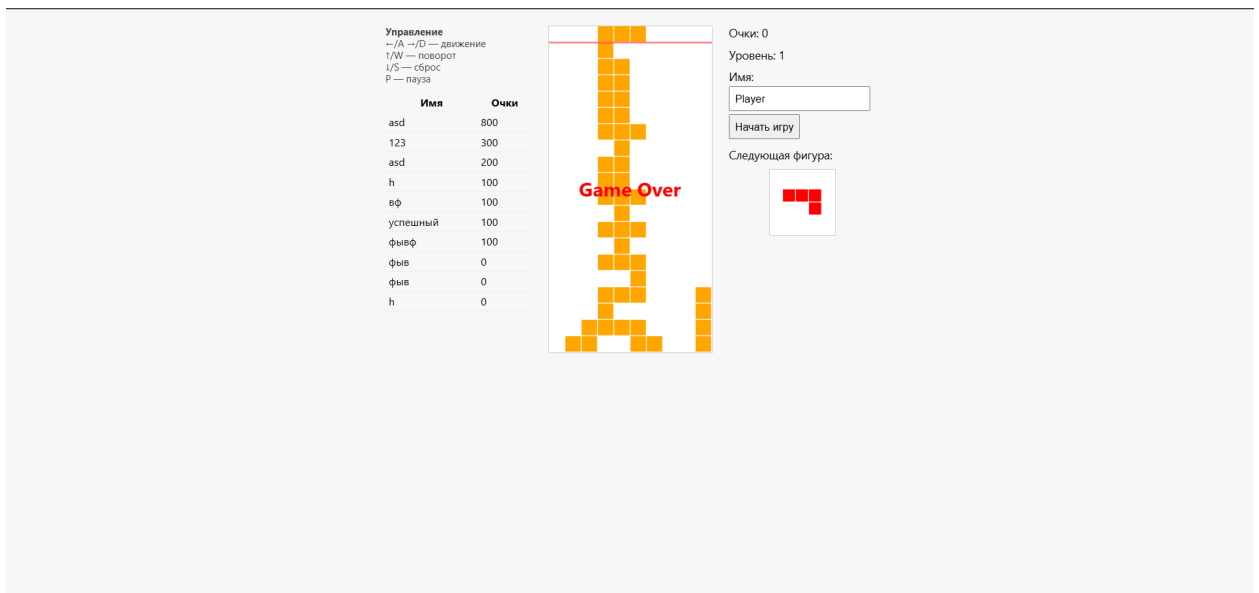


Рисунок 2 – Игра окончена

Вывод

Успешно создана игра Тетрис на языке JavaScript. Использован веб-сервер Nginx для сервировки страницы. Сгенерированы сертификаты для доступа к странице по https. Написаны Dockerfile и docker-compose.yml для запуска приложения в Docker контейнере. Разработан пользовательский интерфейс игры согласно требованиям лабораторной. Создан движок игры Тетрис. Все компоненты работают корректно.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

src/index.html:

```
<!doctype html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta content="width=device-width, initial-scale=1.0"
name="viewport">
    <title>Тетрис</title>
    <link href="style.css" rel="stylesheet">
</head>
<body>
<main>
    <div class="left-column">
        <div class="help">
            <strong>Управление</strong><br>
            ←/A →/D – движение<br>
            ↑/W – поворот<br>
            ↓/S – сброс<br>
            P – пауза
        </div>
        <table id="leaderboard">
            <thead>
                <tr>
                    <th>Имя</th>
                    <th>Очки</th>
                </tr>
            </thead>
            <tbody></tbody>
        </table>
    </div>

    <div class="game-column">
        <canvas aria-label="игровое поле" height="400" id="gameCanvas"
width="200"></canvas>
        <div id="overlay"></div>
    </div>

    <div class="right-column">
```

```

<div id="score">Очки: 0</div>
<div id="level">Уровень: 1</div>
<div>
  <label for="playerName">Имя:</label>
  <input id="playerName" placeholder="Игрок">
  <button id="startGame">Начать игру</button>
</div>
<div>Следующая фигура:</div>
<canvas height="80" id="nextCanvas" width="80"></canvas>
</div>
</main>
<script src="main.js"></script>
</body>
</html>

```

src/main.js:

```

const canvas = document.getElementById('gameCanvas');
const ctx = canvas.getContext('2d');
const overlay = document.getElementById('overlay');
const scoreEl = document.getElementById('score');
const levelEl = document.getElementById('level');
const nameInput = document.getElementById('playerName');
const startBtn = document.getElementById('startGame');
const leaderboardBody = document.querySelector('#leaderboard tbody');
const nextCanvas = document.getElementById('nextCanvas');
const nextCtx = nextCanvas.getContext('2d');

const COLS = 10, ROWS = 20, CELL = 20;
const TETROMINOS = [
  [[1, 1, 1, 1]], [[1, 1, 1], [0, 1, 0]], [[1, 1, 0], [0, 1, 1]],
  [[0, 1, 1], [1, 1, 0]], [[1, 1, 1], [1, 0, 0]], [[1, 1, 1], [0, 0,
1]], [[1, 1], [1, 1]]
];

let board = Array.from({length: ROWS}, () => Array(COLS).fill(0)),
currentPiece, nextPiece;

let score = 0, level = 1, intervalId;
let paused = false, gameOver = false, playerName = '';
const loseLine = 1;
const scoreThresholds = [0, 500, 1000, 2000, 4000];
let dropInterval = 500;

draw();

```



```

function createPiece() {
    return {shape: TETROMINOS[Math.floor(Math.random() *
TETROMINOS.length)].map(r => r.slice()), x: 3, y: 0};
}

function startGame() {
    board = Array.from({length: ROWS}, () => Array(COLS).fill(0));
    score = 0;
    level = 1;
    paused = false;
    gameOver = false;
    overlay.textContent = '';
    currentPiece = createPiece();
    nextPiece = createPiece();
    updateScore();
    updateLevel();
    draw();
    drawNextPiece();
    startLoop();
    startBtn.disabled = true;
    nameInput.disabled = true;
}

function draw() {
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    for (let y = 0; y < ROWS; y++) for (let x = 0; x < COLS; x++) if
(board[y][x]) drawCell(x, y, '#ffa500');

    ctx.strokeStyle = 'red';
    ctx.beginPath();
    ctx.moveTo(0, loseLine * CELL);
    ctx.lineTo(COLS * CELL, loseLine * CELL);
    ctx.stroke();

    if (currentPiece && !gameOver) {
        drawGhost();
        currentPiece.shape.forEach((row, dy) => row.forEach((v, dx) => {
            if (v) {
                const X = currentPiece.x + dx, Y = currentPiece.y + dy;
                if (Y >= 0 && Y < ROWS) drawCell(X, Y, 'red');
            }
        }));
    }
}

```

```

    }
  }

function drawCell(x, y, color) {
  ctx.fillStyle = color;
  ctx.fillRect(x * CELL, y * CELL, CELL - 1, CELL - 1);
}

function drawGhost() {
  let dropY = currentPiece.y;
  while (!collides(currentPiece.shape, currentPiece.x, dropY + 1))
    dropY++;
  currentPiece.shape.forEach((row, dy) => row.forEach((v, dx) => {
    if (v) {
      const X = currentPiece.x + dx, Y = dropY + dy;
      if (Y >= 0 && Y < ROWS) drawCell(X, Y,
'rgba(128,128,128,0.4)');
    }
  }));
}

function drawNextPiece() {
  nextCtx.clearRect(0, 0, nextCanvas.width, nextCanvas.height);
  const piece = nextPiece.shape;
  const rows = piece.length;
  const cols = piece[0].length;
  const cellSize = 16;
  const offsetX = Math.floor((nextCanvas.width - cols * cellSize) /
2);
  const offsetY = Math.floor((nextCanvas.height - rows * cellSize) /
2);
  piece.forEach((row, dy) => row.forEach((v, dx) => {
    if (v) {
      nextCtx.fillStyle = 'red';
      nextCtx.fillRect(offsetX + dx * cellSize, offsetY + dy *
cellSize, cellSize - 1, cellSize - 1);
    }
  }));
}

function collides(piece, xOffset, yOffset) {
  for (let y = 0; y < piece.length; y++) for (let x = 0; x <
piece[y].length; x++) if (piece[y][x]) {

```

```

        const nx = x + xOffset, ny = y + yOffset;
        if (nx < 0 || nx >= COLS || ny >= ROWS) return true;
        if (ny >= 0 && board[ny][nx]) return true;
    }
    return false;
}

function mergePiece() {
    currentPiece.shape.forEach((row, dy) => row.forEach((v, dx) => {
        if (v) {
            const X = currentPiece.x + dx, Y = currentPiece.y + dy;
            if (Y >= 0) board[Y][X] = 1;
        }
    }));
    clearLines();
    if (currentPiece.y < loseLine) {
        gameOver = true;
        stopLoop();
        overlay.textContent = 'Game Over';
        addLeaderboard();
        draw();
        startBtn.disabled = false;
        nameInput.disabled = false;
        return;
    }
    currentPiece = nextPiece;
    nextPiece = createPiece();
    drawNextPiece();
    if (collides(currentPiece.shape, currentPiece.x, currentPiece.y)) {
        gameOver = true;
        stopLoop();
        overlay.textContent = 'Game Over';
        addLeaderboard();
        draw();
        startBtn.disabled = false;
        nameInput.disabled = false;
    }
}

function clearLines() {
    let cleared = 0;
    for (let y = ROWS - 1; y >= 0; y--) {
        if (board[y].every(v => v)) {

```

```

        board.splice(y, 1);
        board.unshift(Array(COLS).fill(0));
        cleared++;
        y++;
    }
}

if (cleared) {
    score += cleared * 100;
    updateScore();
    updateLevel();
}

}

function rotate(piece) {
    const h = piece.length, w = piece[0].length;
    const rotated = Array.from({length: w}, () => Array(h).fill(0));
    for (let y = 0; y < h; y++) for (let x = 0; x < w; x++) rotated[x][h
- 1 - y] = piece[y][x];
    return rotated;
}

function hardDrop() {
    if (gameOver || paused) return;
    while (!collides(currentPiece.shape, currentPiece.x, currentPiece.y
+ 1)) currentPiece.y++;
    mergePiece();
    draw();
}

function updateScore() {
    scoreEl.textContent = `Очки: ${score}`;
}

function updateLevel() {
    for (let i = scoreThresholds.length - 1; i >= 0; i--) {
        if (score >= scoreThresholds[i]) {
            level = i + 1;
            dropInterval = 500 - Math.min(400, (level - 1) * 50);
            levelEl.textContent = `Уровень: ${level}`;
            if (intervalId) {
                clearInterval(intervalId);
                startLoop();
            }
        }
    }
}

```

```

        break;
    }
}

function startLoop() {
    stopLoop();
    intervalId = setInterval(() => {
        if (!paused && !gameOver) {
            if (!collides(currentPiece.shape, currentPiece.x,
currentPiece.y + 1)) currentPiece.y++; else mergePiece();
            draw();
        }
    }, dropInterval);
}

function stopLoop() {
    if (intervalId) clearInterval(intervalId);
    intervalId = null;
}

function addLeaderboard() {
    const arr = JSON.parse(localStorage.getItem('tetris') || '[]');
    arr.push({n: playerName, s: score});
    arr.sort((a, b) => b.s - a.s);
    localStorage.setItem('tetris', JSON.stringify(arr.slice(0, 10)));
    renderLeaderboard();
}

function renderLeaderboard() {
    const arr = JSON.parse(localStorage.getItem('tetris') || '[]');
    leaderboardBody.innerHTML = arr.map(x => `<tr><td>${escapeHtml(x.n)}
</td><td>${x.s}</td></tr>`).join('');
}

function escapeHtml(s) {
    return String(s).replace(/([<>"']/g, c => ({
        '&': '&amp;',
        '<': '&lt;',
        '>': '&gt;',
        '"': '&quot;',
        "'": '&#39;'
    }[c])));
}

```

```

    }

    document.addEventListener('keydown', e => {
        if (gameOver) return;
        switch (e.code) {
            case 'ArrowLeft':
            case 'KeyA':
                if (!paused && !collides(currentPiece.shape, currentPiece.x
- 1, currentPiece.y)) currentPiece.x--;
                break;
            case 'ArrowRight':
            case 'KeyD':
                if (!paused && !collides(currentPiece.shape, currentPiece.x
+ 1, currentPiece.y)) currentPiece.x++;
                break;
            case 'ArrowUp':
            case 'KeyW':
                if (!paused) {
                    const r = rotate(currentPiece.shape);
                    if (!collides(r, currentPiece.x, currentPiece.y))
currentPiece.shape = r;
                }
                break;
            case 'ArrowDown':
            case 'KeyS':
                if (!paused) hardDrop();
                break;
            case 'KeyP':
                if (!gameOver && startBtn.disabled) {
                    paused = !paused;
                    overlay.textContent = paused ? 'Paused' : '';
                }
                break;
        }
        draw();
    });

    startBtn.addEventListener('click', () => {
        if (!startBtn.disabled) {
            const name = nameInput.value.trim();
            if (!/^[a-яA-ЯёЁa-zA-Z0-9]{1,20}$/.test(name)) {
                alert("Имя: 1-20 символов, буквы или цифры");
                return;
            }
        }
    });

```

```

        }
        playerName = name;
        startGame();
    }
});

```

```
renderLeaderboard();
```

src/style.css:

```

body {
    margin: 0;
    background: #f7f7f7;
    font: 14px/1.2 system-ui, Segoe UI, Roboto, Arial;
    padding: 20px;
}

main {
    display: flex;
    gap: 20px;
    align-items: flex-start;
    justify-content: center;
}

.left-column {
    display: flex;
    flex-direction: column;
    gap: 10px;
    width: 180px;
}

.help {
    font-size: 12px;
    color: #444;
}

table {
    border-collapse: collapse;
    width: 100%;
    font-size: 13px;
}

th, td {
    border-bottom: 1px solid #eee;
}

```

```

        padding: 4px;
    }

    .game-column {
        position: relative;
    }

    canvas#gameCanvas {
        background: #fff;
        border: 1px solid #ccc;
        display: block;
    }

    #overlay {
        position: absolute;
        top: 0;
        left: 0;
        width: 200px;
        height: 400px;
        display: flex;
        align-items: center;
        justify-content: center;
        font-size: 24px;
        font-weight: bold;
        color: red;
        pointer-events: none;
    }

    .right-column {
        display: flex;
        flex-direction: column;
        gap: 10px;
        width: 180px;
    }

    input, button {
        padding: 6px;
        font-size: 13px;
        margin-top: 4px;
    }

    canvas#nextCanvas {
        background: #fff;

```



```

border: 1px solid #ccc;
display: block;
margin: 0 auto;
}

```

docker-compose.yml:

```

services:
  webserver:
    build: .
    ports:
      - "8443:443"
    volumes:
      - ./certificates/localhost.pem:/etc/ssl/certs/localhost.pem:ro
      - ./certificates/localhost-key.pem:/etc/ssl/private/localhost-
key.pem:ro

```

Dockerfile:

```

FROM nginx:alpine

COPY nginx.conf /etc/nginx/conf.d/default.conf

COPY src/* /usr/share/nginx/html/

EXPOSE 443

CMD ["nginx", "-g", "daemon off;"]

```

nginx.conf:

```

server {
    listen 443 ssl;
    server_name _;

    ssl_certificate /etc/ssl/certs/localhost.pem;
    ssl_certificate_key /etc/ssl/private/localhost-key.pem;

    root /usr/share/nginx/html;
    index index.html;

    location / {
        try_files $uri /index.html;

        add_header Cache-Control "no-store, no-cache, must-revalidate,
proxy-revalidate, max-age=0";
    }
}

```

```
        add_header Pragma "no-cache";
        add_header Expires "0";
    }
}
```