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. 1304 _____ Шаврин А.П.

Преподаватель _____ Беляев С.А.

Санкт-Петербург
2023

Цель работы.

Изучение работы web-сервера nginx со статическими файлами и создание клиентских JavaScript web-приложений.

Задание.

Необходимо создать web-приложение – игру в тетрис. Основные требования:

- сервер – nginx, протокол взаимодействия – HTTPS версии не ниже 2.0;
- отображается страница для ввода имени пользователя с использованием HTML-элементов <input>;
- статическая страница отображает «стакан» для тетриса с использованием HTML-элемента <canvas>, элемент используется для отображения следующей фигуры, отображается имя пользователя;
- фигуры в игре – классические фигуры тетриса (7 шт. тетрамино);
- случайным образом генерируется фигура и начинает падать в «стакан» (описание правил см., например, https://ru.wikipedia.org/wiki/Тетрис);
- пользователь имеет возможность двигать фигуру влево и вправо, повернуть на 90° и «уронить»;
- если собралась целая «строка», она должна исчезнуть;
- при наборе некоторого заданного числа очков увеличивается уровень, что заключается в увеличении скорости игры;
- пользователь проигрывает, когда стакан «заполняется», после чего ему отображается локальная таблица рекордов;
- вся логика приложения написана на JavaScript.

Необязательно: оформление с использованием CSS.

Выполнение работы.

1. Создана пара ssl ключей для работы https протокола nginx
2. Настроены конфигурационные файлы nginx
3. Написан файл index.html, на странице которого располагается приветственная фраза, изображение 3d тетромино и форма для авторизации в игре (поле ввода имени и кнопка подтверждения). Реализован файл index_style.css с заданием стилей для элементов страницы. Создан файл Authorization.js, в котором прописан код действий по кнопке. После нажатия кнопки имя пользователя сохраняется в локальное хранилище и происходит переход на страницу main.html
4. Написан файл main.html, на странице которого располагается поле – «стакан», в котором падают тетромино, и справа от него игровая информация. Поле отображается с помощью элемента canvas, а в игровой информации выводится название игры, имя пользователя, уровень, счет, количество заполненных линий, окно для отображения следующей тетромино, с использованием конструкций div, и кнопка начала игры. Добавлен файл style.css для задания стилей элементам страницы. Создан файл main.js, в котором задана обработка кнопки – вызов функции play, осуществляющей создание главного класса игры и обработка нажатия клавиш передвижения, и вывод имени пользователя.
5. Реализован набор js файлов с классами, хранящими соответствующую названию информацию.
   - Utils.js – файл хранящий все игровые константы
   - Класс Canvas хранит элементы страницы необходимые для отображения (canvas - игровое поле, окно вывода следующей тетромино, элементы вывода уровня, счета, удаленных линий) и методы для реализации отображения.
   - Класс PlayField хранит количество строк, колонок и матрицу поля из нулей и единиц, где 0 – свободная клетка, 1 – занятая клетка, а также методы необходимые для работы с игровым полем (проверка занятости заданной

клетки, проверка принадлежности заданной клетки полю, размещение тетромино, поиск заполненных строк, удаление заполненных строк, смещение всех строк над заданной строкой).

- Класс Tetromino хранит имя тетромино, информацию о том, тень она или нет, его цвет, форму – матрицу n x n, где 0 – пустая клетка, 1 – заполненная клетка и координаты. Также реализован метод поворота матрицы фигуры.

- Класс Tetris хранит данные об уровне, счете, удаленных строках, объект класса PlayField, объект класса Canvas, объекты текущей тетромино, тени текущей тетромино, следующей тетромино, а также методы необходимые для работы (перезапуск таймера, отвечающего за падение тетромино, создания тетромино, создания теневой тетромино, проверки занятости данной конфигурации тетромино, проверка принадлежности полю данной конфигурации тетромино, проверка корректности позиции тетромино, перемещения тетромино, вычисление новой позиции теневой тетромино, обновление данных очков при фиксации тетромино, обновление поля, тетромино и следующей тетромино при фиксации тетромино, вычисления очков, окончания игры).

6. Написан файл game_over.html, на странице которого отображается текст об окончании игры, изображение конца игры и первые 5 записей таблицы рекордов. Создан файл game_over.css с заданием стилей для элементов страницы и файл GameOver.js, реализующий вывод таблицы рекордов.

Основная логика игры:

При нажатии кнопки Play создается объект класса Tetris и устанавливаются обработчики нажатия клавиш. При создании класса Tetris, рисуются текущая тетромино, ее тень, следующая тетромино. По таймеру происходит падение тетромино (вызов функции перемещения тетромино вниз). В функции перемещения проверяется возможность передвижения в заданном направлении и отрисовка измененного положения (при нажатии клавиш то же самое). При

передвижении вниз и бросания тетромино таймер перезапускается. Если при передвижении вниз не удалось изменить положение тетромино, то вызывается метод обновления поля, в котором тетромино размещается на поле, текущее тетромино становится равно следующему, следующее генерируется заново, высчитывается теневое тетромино. Пересчитываются данные о текущем счете, уровне и удаленных линиях. Далее цикл повторяется до падения текущего тетромино.

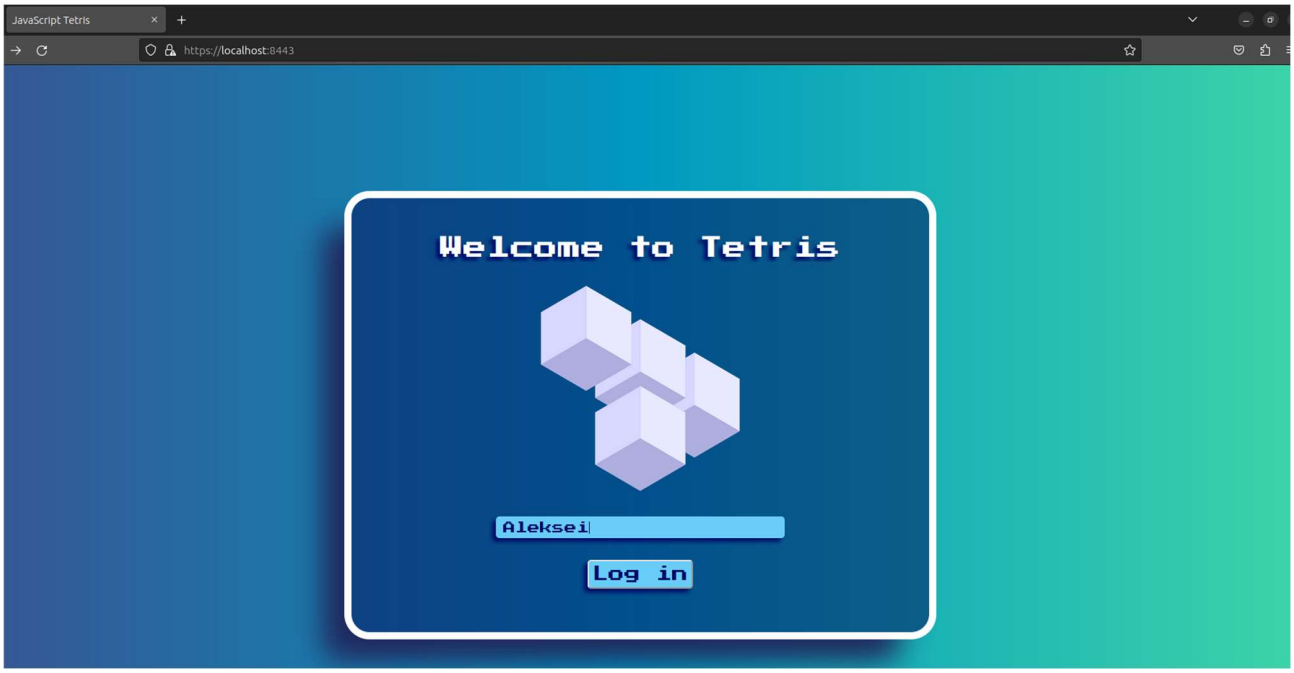

Рис 1. Стартовая страница.

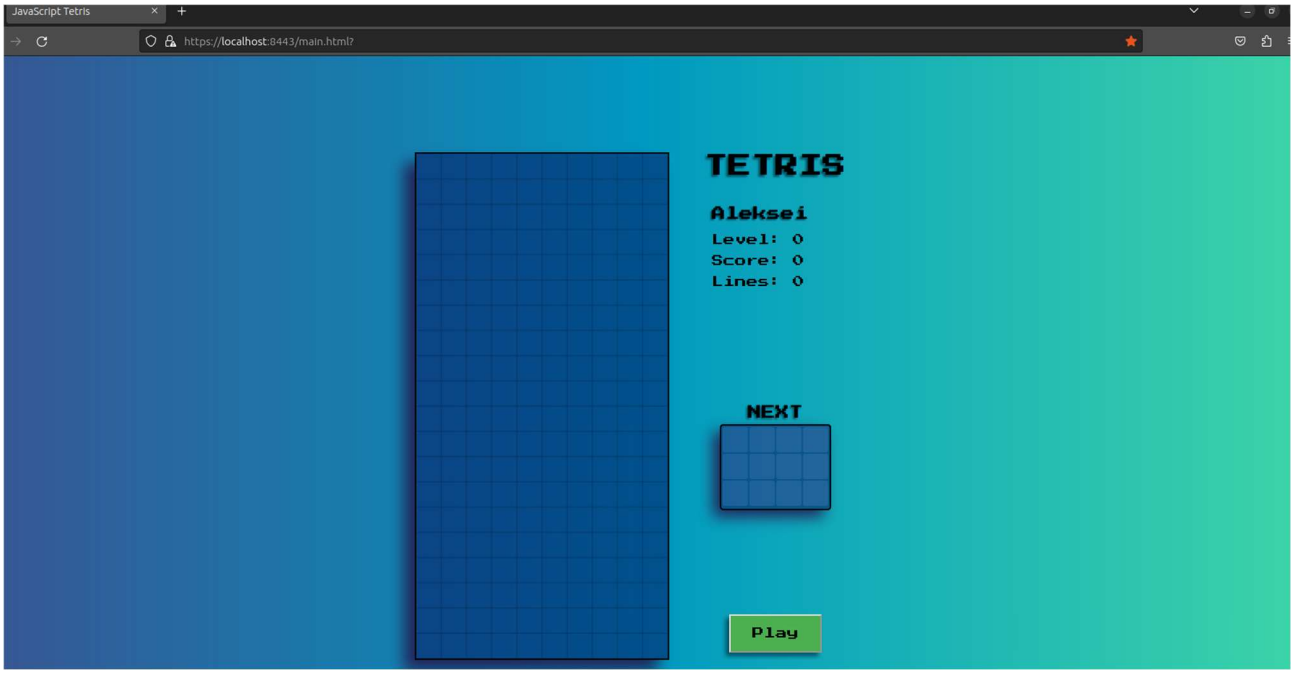

Рис 2. Главная страница до нажатия кнопки старта.

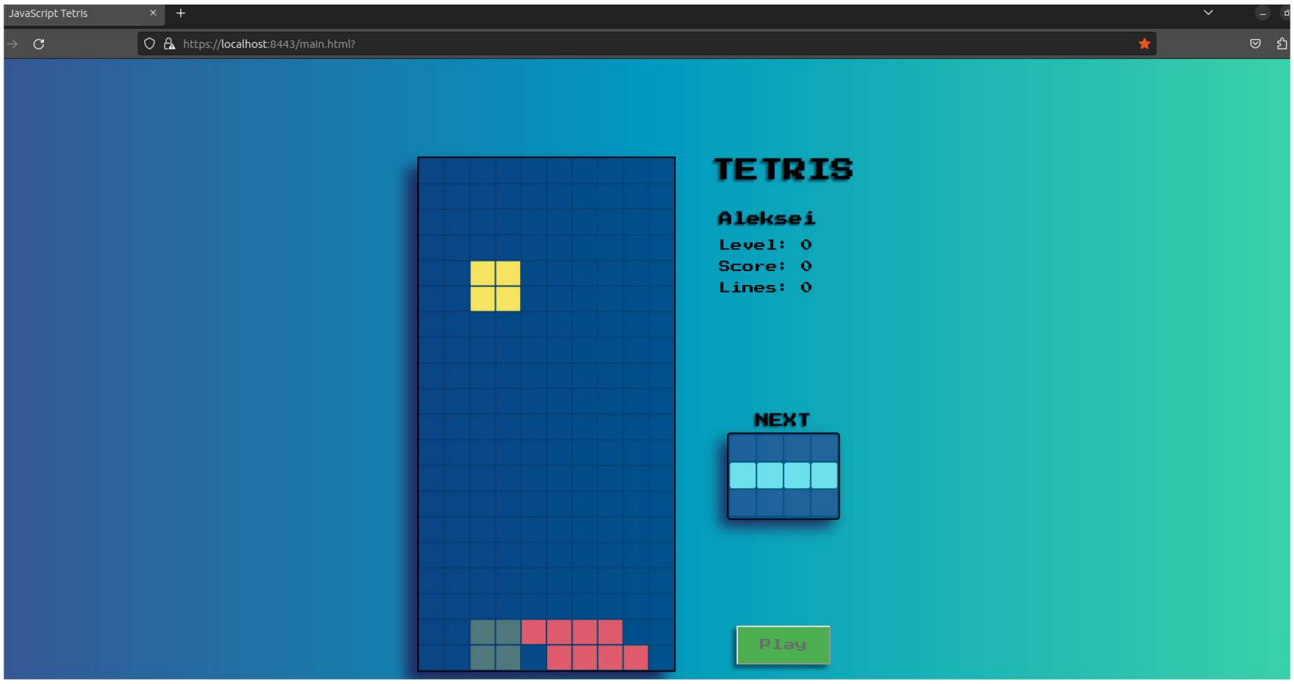

Рис 3. Главная страница во время игры.

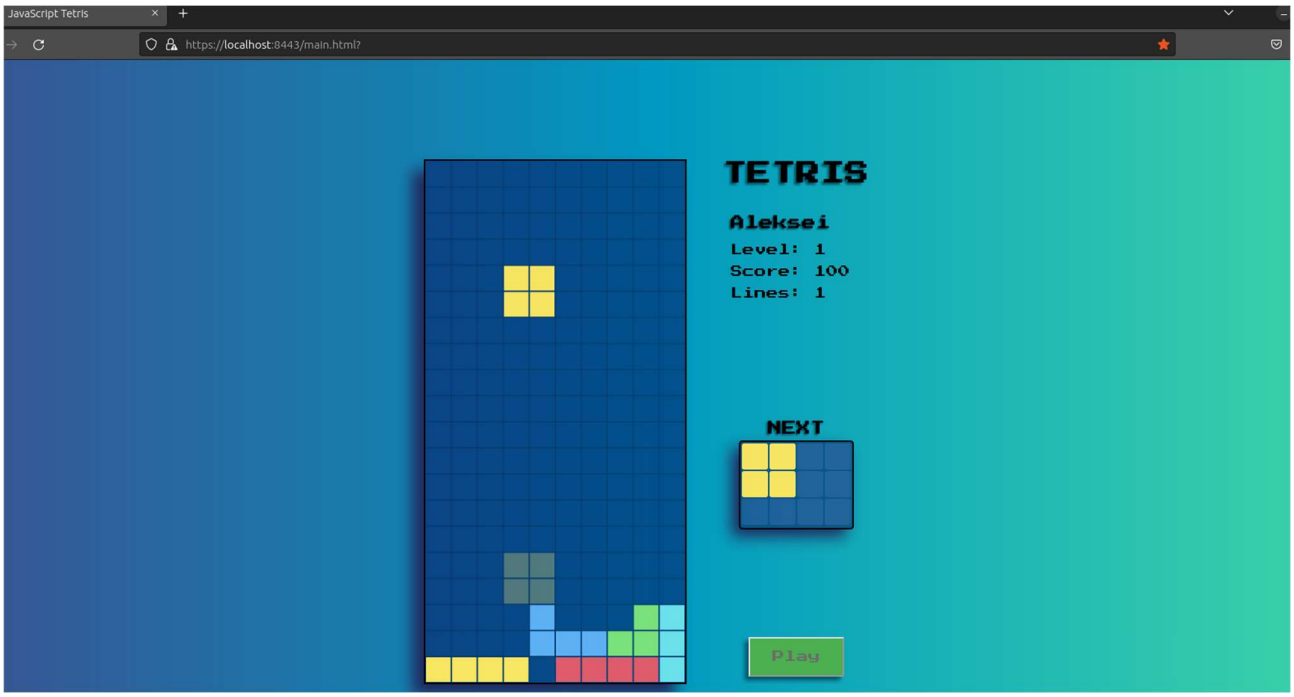

Рис 4. Главная страница с обновленными результатами.

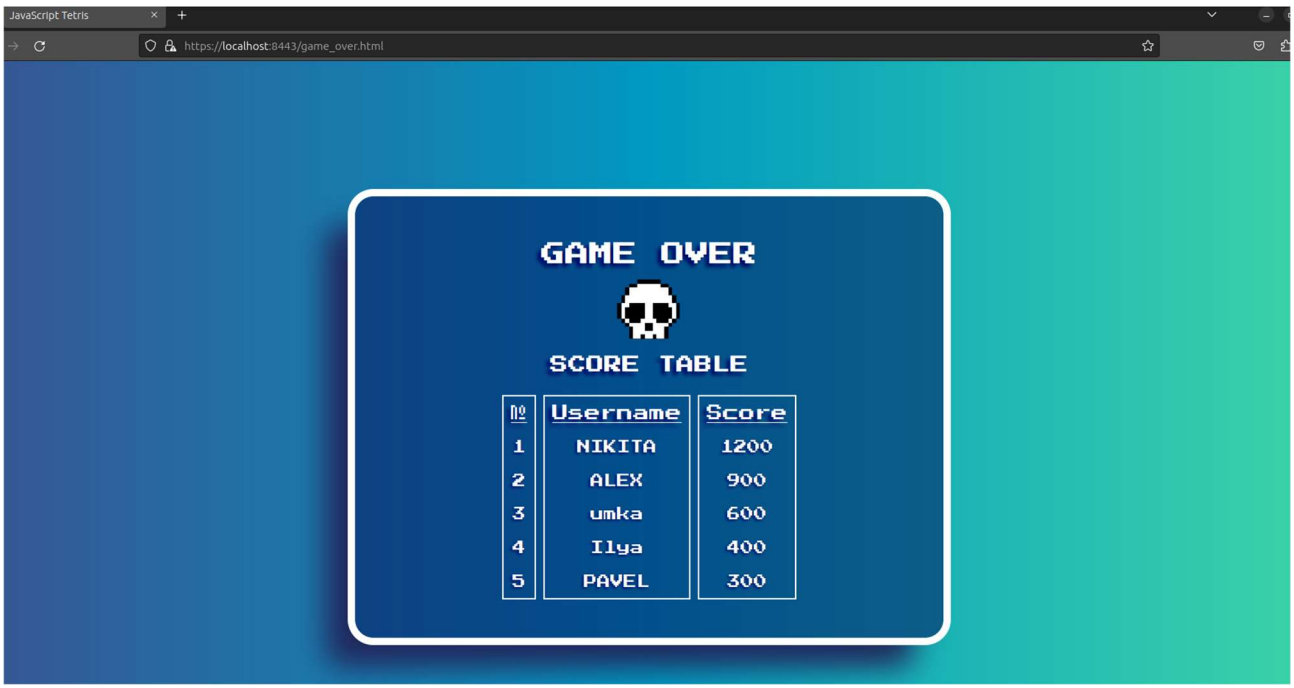

Рис 5. Страница результатов.

Разработанный программный код см. в приложении А.

Выводы.

В ходе выполнения работы изучена работа web-сервера nginx со статическими файлами, его настройка, изучены основы языка JavaScript, HTML – языка гипертекстовой разметки и CSS – каскадной таблицы стилей.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: nginx.config

```
server
{
    listen 8443 ssl http2;

    ssl_certificate /etc/ssl/certs/example.csr;
    ssl_certificate_key /etc/ssl/private/example.key;

    location /
    {
        root /home/alex/Programming/Leti/5_semester/Web/lab_1/src/;
        index index.html;
    }
}
```

Название файла: index.html

```
<!DOCTYPE html>
<html land="eng">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel="stylesheet" type="text/css"
href="./Css/index_style.css">
    <link
href="https://fonts.googleapis.com/css?family=Press+Start+2P"
rel="stylesheet"/>
    <script type="module" src="./Scripts/Authorization.js"></script>

    <title>JavaScript Tetris</title>
</head>

<body>
    <div class="authorization">
        <h1 class="preview_text">Welcome to Tetris</h1>
        <div class="preview_img">
            <img src="./Images/prev.png" alt="Tetris" width="300px"
height="300px">
        </div>
        <form id="authorization_form" action="./main.html"
method="get">
            <label>
                <input class="username_input" id="username_input"
placeholder="Enter your username"><br>
            </label>
            <br>
            <input class="log_in_button" id="log_in_button"
disabled="disabled" type="submit" value="Log in">
        </form>
    </div>
</body>
</html>
```

Название файла: main.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta  name="viewport"  content="width=device-width,  initial-
scale=1.0">
    <link rel="stylesheet" type="text/css" href="./Css/style.css">
    <link
href="https://fonts.googleapis.com/css?family=Press+Start+2P"
rel="stylesheet"/>
    <title>JavaScript Tetris</title>

    <script type="module" src="./Scripts/main.js"></script>
</head>

<body>
    <div class="grid">
        <canvas class="play_field" id="play_field"></canvas>
        <div class="game_info_container">
            <div class="game_info">
                <h1 class="game_name">TETRIS</h1>
                <h3 id="username"></h3>
                <p        class="game_info_row">Level:        <span
id="level">0</span></p>
                <p        class="game_info_row">Score:        <span
id="score">0</span></p>
                <p        class="game_info_row">Lines:        <span
id="lines">0</span></p>
            </div>
            <div class="next_tetromino_conteiner">
                <h3 class="next_tetromino_text">NEXT</h3>
                <div class="next_tetromino_grid">
                    <div></div>
                    <div></div>
                    <div></div>
                    <div></div>
                    <div></div>
                    <div></div>
                    <div></div>
                    <div></div>
                    <div></div>
                    <div></div>
                    <div></div>
                    <div></div>
                </div>
            </div>
            <button       class="play_button"       id="play_button"
type="button">Play</button>
        </div>
    </div>
</body>
</html>
```

Название файла: game_over.html

```
<!DOCTYPE html>
<html land="eng">
<head>
    <meta charset="UTF-8">
    <meta   name="viewport"   content="width=device-width,   initial-
scale=1.0">
    <link              rel="stylesheet"              type="text/css"
href="./Css/game_over.css">
    <link
href="https://fonts.googleapis.com/css?family=Press+Start+2P"
rel="stylesheet"/>
    <script type="module" src="./Scripts/GameOver.js"></script>

    <title>JavaScript Tetris</title>
</head>

<body>
    <div class="game_over">
        <h1 class="game_over_text">GAME OVER</h1>
        <div class="game_over_img">
            <img  src="./Images/game_over.png"  alt="game_over_img"
width="100px" height="100px">
        </div>
        <h2 class="score_table_text">SCORE TABLE</h2>
        <div class="game_over_grid">
            <div class="column" id="numbers_column">
                <div class="column_name">№</div>
            </div>
            <div class="column" id="names_column">
                <div class="column_name">Username</div>
            </div>
            <div class="column" id="scores_column">
                <div class="column_name">Score</div>
            </div>
        </div>
    </div>
</body>
</html>
```

Название файла: index_style.css

```
*
{
    font-family: 'Press Start 2P';
    padding: 0;
    margin: 0;
    box-sizing: border-box;
    overflow: hidden;
}

body
{
    min-height: 100vh;
    background: linear-gradient(90deg, #395492,  #0099c0, #3dd5a8);
    display: flex;
    justify-content: center;
```

```
    align-items: center
}

.authorization
{
    width: 800px;
    height: 600px;

    display: flex;
    flex-direction: column;
    justify-content: space-between;


    text-align: center;

    background-color: rgba(1, 8, 89, 0.5);
    box-shadow: -40px 40px 40px #21265c;

    outline: 10px solid #ffffff;
    border-radius: 25px;
    padding: 50px;
}

.preview_text
{
    color: #ffffff;
    text-shadow: -5px 5px 5px #000a66;
}

.username_input
{
    background-color: #69cdf8;
    color: #000a66;
    outline: none;
    box-shadow: -5px 5px 5px #000a66;
    font-size: 17px;

    padding-left: 10px;
    padding-right: 50px;
    padding-top: 5px;
    padding-bottom: 5px;

    border: 0px;
    border-radius: 5px;
}

.log_in_button
{
    text-align: center;
    font-size: 22px;

    color: #000a66;
    background-color: #69cdf8;

    border-radius: 5px;
    padding: 5px;
    margin: 10px;
```

```
    cursor: pointer;
    box-shadow: -5px 5px 5px #000a66;
}

.log_in_button:hover
{
    color: #000a66;
    background-color: #69cdf8;
}

.log_in_button:disabled
{
    color: #000a665b;
    background-color: #69cdf886;
}

.log_in_button:active
{
    color: #000000;
    background-color: #ffffff;
}
```

Название файла: style.css

```
*
{
    font-family: 'Press Start 2P';
    padding: 0;
    margin: 0;
    box-sizing: border-box;
    overflow: hidden;
}

body
{
    min-height: 100vh;
    background: linear-gradient(90deg, #395492,  #0099c0, #3dd5a8);
    display: flex;
    align-items: center;
    justify-content: center;
}

.grid
{
    display: grid;
    grid-template-columns: auto auto;
    grid-template-rows: max-content;
    grid-column-gap: 20px;
    padding: 40px;
}

.play_field
{
    width: 352px;
    height: 702px;
    border: 2px solid;
    background-color: rgba(1, 8, 89, 0.5);
    box-shadow: -15px 15px 20px #21265c;
```

```
}

.game_info_container
{
    display: flex;
    flex-direction: column;
    justify-content:space-between;
    align-items: center;
}

.game_name
{
    margin-bottom: 30px;
    text-shadow: -3px 3px 3px #005266;
}

#username
{
    text-shadow: -2px 2px 2px #005266;
    padding: 5px;
    margin-left: 5px;
}

.game_info_row
{
    margin: 10px;
}

.next_tetromino_conteiner
{
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    width: 100%;
    padding: 50px;
}

.next_tetromino_text
{
    padding: 5px;
    text-shadow: -2px 2px 3px #005266;
}

.next_tetromino_grid
{
    display: grid;
    grid-template-columns: repeat(4, auto);
    grid-template-rows: repeat(4, auto);

    gap: 2px;
    padding: 2px;
    border: 2px solid;
    border-radius: 5px;

    background-color: rgba(1, 8, 89, 0.5);
    box-shadow: -10px 10px 20px #21265c;
}
```

```
.next_tetromino_grid>div
{
    --cell-size: 35px;
    min-height: var(--cell-size);
    min-width: var(--cell-size);
    border-radius: 3px;
    background-color: rgba(126, 150, 221, 0.2);
}

.next_tetromino_grid>div.I
{
    background-color: #6be1ec;
}

.next_tetromino_grid>div.J
{
    background-color: #5eb0f3;
}

.next_tetromino_grid>div.L
{
    background-color: #f2965b;
}

.next_tetromino_grid>div.O
{
    background-color: #f7e562;
}

.next_tetromino_grid>div.S
{
    background-color: #7be17b;
}

.next_tetromino_grid>div.Z
{
    background-color: #de5c6b;
}

.next_tetromino_grid>div.T
{
    background-color: #b276f3;
}


.play_button
{
    background-color: #4caf50;
    box-shadow: -5px 5px 5px #005266;

    font-size: 16px;
    padding: 15px 30px;
    margin-bottom: 10px;
    cursor: pointer;
}
```

Название файла: game_over.css

```
*
{
    font-family: 'Press Start 2P';
    padding: 0;
    margin: 0;
    box-sizing: border-box;
    overflow: hidden;
}

body
{
    min-height: 100vh;
    background: linear-gradient(90deg, #395492,  #0099c0, #3dd5a8);
    display: flex;
    justify-content: center;
    align-items: center
}

.game_over
{
    width: 800px;
    height: 600px;

    display: flex;
    flex-direction: column;
    justify-content: space-between;


    text-align: center;

    background-color: rgba(1, 8, 89, 0.5);
    box-shadow: -40px 40px 40px #21265c;

    outline: 10px solid #ffffff;
    border-radius: 25px;
    padding: 50px;
}

.game_over_text
{
    color: #ffffff;
    text-shadow: -5px 5px 5px #000a66;
    padding: 10px;
}

.score_table_text
{
    color: #ffffff;
    text-shadow: -5px 5px 5px #000a66;
    padding: 10px;
    margin-top: 10px;
}

.game_over_grid
{
    min-width: 60%;
```

```
        min-height: 60%;
        display: grid;
        justify-content: center;
        grid-template-columns: repeat(3, min-content);
        grid-template-rows: max-content;
        column-gap:10px;
        padding: 20px;
    }

    .column
    {
        height: 100%;
        display: flex;
        flex-direction: column;
        justify-content: space-around;
        align-items: center;
        border: 2px solid #ffff;
    }

    .column_name
    {
        padding: 10px;
        font-size: 22px;
        color: #ffff;
        text-shadow: -5px 5px 5px #000a66;
        text-decoration: underline;
    }

    .row
    {
        font-size: large;
        color: #ffff;
        margin: 7px;
        padding: 5px;
        text-shadow: -2px 2px 4px #000a66;
    }
```

Название файла: Authorization.js

```
    let username_element = document.getElementById("username_input")
    let log_in_button_element = document.getElementById("log_in_button")
    let form_element = document.getElementById("authorization_form")

    username_element.addEventListener("input", checkUsernameInput)
    form_element.addEventListener("submit", saveUsername)

    function checkUsernameInput(event)
    {
        /* This function activates and disables the button depending on
the user's name input */

        if (username_element.value.length === 0)
        {
            log_in_button_element.setAttribute("disabled", "disable")
        } else
        {
            log_in_button_element.removeAttribute("disabled")
        }
```

```
}

function saveUsername(event)
{
    /*
    This function saves the username to local memory
    and creates an entry for the high score table as needed
    */

    localStorage["tetris.username"] = username_element.value
    if (localStorage["tetris.score_table"] === undefined)
    {
        localStorage["tetris.score_table"] = JSON.stringify([])
    }
}
```

Название файла: main.js

```
import {Tetris} from "./Tetris.js"
import { Canvas } from "./Canvas.js"
import { MOVEMENT_ACTIVITIES } from "./Utils.js"


let play_button_element = document.getElementById("play_button")
play_button_element.addEventListener("click", play)

let username_element = document.getElementById("username")
username_element.innerText = `${localStorage["tetris.username"]}`


const canvas = new Canvas()

function play() {
    /* This function launches the game */

    /* Disable the button */
    play_button_element.setAttribute("disabled", "disable")
    play_button_element.blur()

    const tetris = new Tetris(canvas)

    document.addEventListener("keydown", onKeydown)

    function onKeydown(event)
    {
        /* This function handles key presses */
        switch (event.key){
            case 'ArrowDown':
            case 's':
                tetris.moveTetromino(MOVEMENT_ACTIVITIES.Down)
                break
            case 'ArrowLeft':
            case 'a':
                tetris.moveTetromino(MOVEMENT_ACTIVITIES.Left)
                break
            case 'ArrowRight':
            case 'd':
                tetris.moveTetromino(MOVEMENT_ACTIVITIES.Right)
```

```
                    break
                case 'ArrowUp':
                case 'w':
                    tetris.moveTetromino(MOVEMENT_ACTIVITIES.Rotate)
                    break
                case ' ':
                    tetris.moveTetromino(MOVEMENT_ACTIVITIES.Drop)
                    break
                default:
                    break
            }
        }
    }
```

Название файла: Utils.js

```
    /* play field consts */
    export const PLAYFIELD_ROWS = 20
    export const PLAYFIELD_COLUMNS = 10
    export const BLOCK_SIZE = 35

    /* next tetromino window consts*/
    export const NT_WINDOW_ROWS = 4
    export const NT_WINDOW_COLUMNS= 4

    /* level consts */
    export const MAX_LEVEL = 20
    export const LEVEL_TIME_INCREASE = 50

    /* movement enum */
    export const MOVEMENT_ACTIVITIES = {Rotate: 0, Down: 1, Left: 2,
Right: 3, Drop: 4}

    /* tetromino enums */
    export const TETROMINO_NAMES = ['I', 'J', 'L', 'O', 'S', 'Z', 'T']
    export const TETROMINO_COLORS = {
        I: ["rgba(107, 225, 236, 1)", "rgba(107, 225, 236, 0.3)"],
        J: ["rgba(94, 176, 243, 1)", "rgba(94, 176, 243, 0.3)"],
        L: ["rgba(242, 150, 91, 1)", "rgba(242, 150, 91, 0.3)"],
        O: ["rgba(247, 229, 98, 1)", "rgba(247, 229, 98, 0.3)"],
        S: ["rgba(123, 225, 123, 1)", "rgba(123, 225, 123, 0.3)"],
        Z: ["rgba(222, 92, 107, 1)", "rgba(222, 92, 107, 0.3)"],
        T: ["rgba(178, 118, 243, 1)", "rgba(178, 118, 243, 0.3)"]
    }
    export const TETROMINO_SHAPES = {
        '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]
        ],
        'S': [
            [0, 1, 0],
            [1, 1, 1],
            [0, 0, 0]
        ]
    }
```

Название файла: Tetromino.js

```
    import { TETROMINO_SHAPES, TETROMINO_NAMES, PLAYFIELD_COLUMNS} from
"./Utils.js"

    export class Tetromino {
        /* This is a tetromino class */

        constructor (tetromino_name, is_clone = false)
        {
            this.name = tetromino_name !== undefined ? tetromino_name:
TETROMINO_NAMES[0]
            this.is_clone = is_clone
            this.shape = TETROMINO_SHAPES[this.name]
            this.coordinates = {x: Math.floor(PLAYFIELD_COLUMNS / 2) - 2,
y: 0}
        }

        rotate ()
        {
            /* This method rotates the tetromino matrix */
            const N = this.shape.length
            const rotated_matrix = []
            for (let i = 0; i < N; i++) {
                rotated_matrix[i] = []
                for (let j = 0; j < N; j++) {
                    rotated_matrix[i][j] = this.shape[N - j - 1][i]
                }
            }
            this.shape = rotated_matrix
        }
    }
```

Название файла: PlayField.js

```
export class PlayField {
    /* This is a play field class */
    constructor (count_rows, count_columns) {
        this.rows = count_rows
        this.columns = count_columns
        this.play_field = this.generateEmptyPlayField()
    }

    generateEmptyPlayField()
    {
        /* This function creates an empty field */
        return new Array(this.rows).fill(0)
        .map(() => new Array(this.columns).fill(0))
    }

    isFreeCell(x, y)
    {
        /* This function checks whether a given field cell is free
*/
        return this.play_field[y][x] === 0
    }

    isBelongingFieldCoordinates(x, y)
    {
        /* This function checks whether a given cell is inside a
field */
        let isBelonging = true
        if (y < 0 || x < 0 || y >= this.rows || x >= this.columns)
        {
            isBelonging = false
        }
        return isBelonging
    }

    placeTetromino(tetromino)
    {
        /*
        This function marks the cells of the field as occupied that
        correspond to the final position of the tetrominoes
        */
        if (tetromino !== undefined && tetromino.shape !== undefined)
        {
            let start_x = tetromino.coordinates.x
            let start_y = tetromino.coordinates.y

            tetromino.shape.forEach((row, y) =>
            {
                row.forEach((value, x) =>
                {
                if (value > 0){
                    let current_x = start_x + x
                    let current_y = start_y + y

                    this.play_field[current_y][current_x] = 1
                }
                })
```

```
                })
            }
        }

        findFilledRows()
        {
            /* This function returns an array with the indexes of the
filled rows */
            const filled_rows = new Array()
            for (let row = 0; row < this.rows; row++)
            {
                if (this.play_field[row].every(cell => Boolean(cell)))
                {
                    filled_rows.push(row)
                }
            }
            return filled_rows
        }

        removeFilledRows(filled_rows)
        {
            /* This function removes all filled lines */
            if (filled_rows.length != 0)
            {
                filled_rows.forEach(row => {
                    this.dropRowsAbove(row)
                })
            }
        }

        dropRowsAbove(row_to_delete){
            /*
            This is a helper function for deleting rows,
            it shifts all rows down above the row being deleted
            */
            for (let row = row_to_delete; row > 0; row--)
            {
                this.play_field[row] = this.play_field[row - 1]
            }
            this.play_field[0] = new Array(this.columns).fill(0)
        }
    }
```

Название файла: Canvas.js

```
    import   {   PLAYFIELD_COLUMNS,   PLAYFIELD_ROWS,   BLOCK_SIZE   ,
TETROMINO_COLORS, NT_WINDOW_ROWS} from "./Utils.js"

    export class Canvas {
        /* This canvas class is used to display all game information */

        constructor () {
            this.canvas = document.getElementById('play_field')
            this.context = this.canvas.getContext('2d')
            this.level = document.getElementById("level")
            this.score = document.getElementById("score")
            this.lines = document.getElementById("lines")
```

```
            this.next_tetromino_cells                                    =
document.querySelectorAll('.next_tetromino_grid>div')

            /* Set canvas dimensions and zoom */
            this.canvas.width = PLAYFIELD_COLUMNS * BLOCK_SIZE
            this.canvas.height = PLAYFIELD_ROWS * BLOCK_SIZE
            this.context.scale(BLOCK_SIZE, BLOCK_SIZE)

            this.drawPlayFieldGrid(this.canvas.width,
this.canvas.height)

            /* For drawing tetromino */
            this.left_up_gap = 1 / BLOCK_SIZE
            this.right_down_gap = 2 * this.left_up_gap
        }

        drawPlayFieldGrid(width, height)
        {
            /*
            This method draws a field grid from coordinate
            (0, 0) of a given width and height, but with a fixed cell
size
            */

            this.context.scale(1 / BLOCK_SIZE, 1 / BLOCK_SIZE)
            this.context.lineWidth = 1
            this.context.strokeStyle = "rgba(0, 0, 0, 0.15)"

            for (let y = 0; y < height; y += BLOCK_SIZE)
            {
                for (let x = 0; x < width; x += BLOCK_SIZE)
                {
                    this.context.strokeRect(x, y, BLOCK_SIZE, BLOCK_SIZE)
                }
            }
            this.context.scale(BLOCK_SIZE, BLOCK_SIZE)
        }

        removeFilledRows(filled_rows)
        {
            /* This function removes all filled lines */

            if (filled_rows.length != 0)
            {
                filled_rows.forEach(row => {
                    this.dropRowsAbove(row)
                })
                this.drawPlayFieldGrid(this.canvas.width,
filled_rows.length * BLOCK_SIZE)
            }
        }

        dropRowsAbove(row_to_delete)
        {
            /*
            This is a helper function for deleting rows,
            it shifts all rows down above the row being deleted
            */
```

```
            this.context.scale(1/BLOCK_SIZE, 1/BLOCK_SIZE)
            for (let row = row_to_delete; row > 0; row--)
            {
                let source_x = 0, source_y = (row - 1) * BLOCK_SIZE
                let source_width = this.canvas.width, source_height = 1
* BLOCK_SIZE
                let destination_x = 0, destination_y = row * BLOCK_SIZE
                let     destination_width     =     this.canvas.width,
destination_height = 1 * BLOCK_SIZE
                /* clear filled row */
                this.context.clearRect(destination_x,    destination_y,
destination_width, destination_height)
                /* copying row above */
                this.context.drawImage(
                    this.canvas,
                    source_x,
                    source_y,
                    source_width,
                    source_height,
                    destination_x,
                    destination_y ,
                    destination_width,
                    destination_height
                )
            }
            /* clear top row */
            this.context.clearRect(0, 0, this.canvas.width, BLOCK_SIZE)
            this.context.scale(BLOCK_SIZE, BLOCK_SIZE)
        }

        drawTetromino(tetromino)
        {
            /* This method draws tetrominoes */

            if (tetromino !== undefined && tetromino.shape !== undefined)
            {
                this.context.fillStyle                                  =
TETROMINO_COLORS[tetromino.name][tetromino.is_clone ? 1 : 0]
                tetromino.shape.forEach((row, y) =>
                {
                    row.forEach((value, x) =>
                    {
                    if (value > 0)
                    {
                        this.context.fillRect(tetromino.coordinates.x  +
x + this.left_up_gap,
                                              tetromino.coordinates.y + y
+ this.left_up_gap,
                                              1 - this.right_down_gap,
                                              1 - this.right_down_gap)
                    }
                    })
                })
            }
        }

        clearTetromino(tetromino)
```

```
        {
            /* This method removes tetrominoes */

            if (tetromino !== undefined && tetromino.shape !== undefined)
            {
                tetromino.shape.forEach((row, y) =>
                {
                    row.forEach((value, x) =>
                    {
                      if (value > 0)
                      {
                        this.context.clearRect(tetromino.coordinates.x
+ x + this.left_up_gap,
                                               tetromino.coordinates.y  +
y + this.left_up_gap,
                                               1 - this.right_down_gap,
                                               1 - this.right_down_gap)
                      }
                    })
                })
            }
        }

        drawNextTetromino(next_tetromino)
        {
            /* This method displays the next tetromino */

            if (next_tetromino.shape !== undefined)
            {
                this.next_tetromino_cells.forEach(cell                =>
cell.removeAttribute("class"))

                const size = next_tetromino.shape.length
                for (let y = 0; y < size; y++)
                {
                    for (let x = 0; x < size; x++)
                    {
                        if (!next_tetromino.shape[y][x]) continue
                        const cell_index = y * NT_WINDOW_ROWS + x

this.next_tetromino_cells[cell_index].classList.add(next_tetromino.name)
                    }
                }
            }
        }

        updateData(level, score, lines)
        {
            /* This method displays game information */

            this.level.innerText = level
            this.score.innerText = score
            this.lines.innerText = lines
        }
    }
```

Название файла: Tetris.js

```
import { PlayField } from "./PlayField.js"
import { Tetromino } from "./Tetromino.js"
import { PLAYFIELD_ROWS, PLAYFIELD_COLUMNS, TETROMINO_NAMES,
MOVEMENT_ACTIVITIES, LEVEL_TIME_INCREASE, MAX_LEVEL} from "./Utils.js"


export class Tetris {
    /* This is the Tetris class that controls the entire game */

    constructor(canvas) {
        this.level = 0
        this.score = 0
        this.lines = 0

        this.play_field = new PlayField(PLAYFIELD_ROWS,
PLAYFIELD_COLUMNS)

        this.canvas = canvas
        this.current_tetromino = this.generateTetromino()
        this.shadow_current_tetromino =
this.generateShadowCurrentTetromino()
        this.calculateShadowTetraminoPosition()
        this.next_tetromino = this.generateTetromino()

        this.canvas.drawTetromino(this.current_tetromino)
        this.canvas.drawTetromino(this.shadow_current_tetromino)
        this.canvas.drawNextTetromino(this.next_tetromino)

        this.timeout_id = null
        this.restartFallTetrominoTimeout()
    }

    restartFallTetrominoTimeout(timeout = (MAX_LEVEL *
LEVEL_TIME_INCREASE) - (LEVEL_TIME_INCREASE * this.level))
    {
        /* This method moves the figure according to timeout */

        clearTimeout(this.timeout_id)
        this.timeout_id = setTimeout(() =>
this.moveTetromino(MOVEMENT_ACTIVITIES.Down), timeout)
    }

    generateTetromino()
    {
        /* This method creates tetrominoes */

        let random_tetromino_name_index = Math.floor(Math.random() *
TETROMINO_NAMES.length)
        return new
Tetromino(TETROMINO_NAMES[random_tetromino_name_index])
    }

    generateShadowCurrentTetromino()
    {
        /* This method creates shadow current tetromino*/
```

```
            return new Tetromino(this.current_tetromino.name, true)
        }

        isOccupiedCurrentTetrominoPosition()
        {
            /* This method checks whether the position of the current
tetrominoe is occupied */

            let isOccupied = false
            if     (this.current_tetromino     !==     undefined     &&
this.current_tetromino.shape !== undefined)
            {
                let start_x = this.current_tetromino.coordinates.x
                let start_y = this.current_tetromino.coordinates.y
                const size = this.current_tetromino.shape.length

                labelCancelLoops:
                for (let y = 0; y < size; y++)
                {
                    for (let x = 0; x < size; x++)
                    {
                        let current_x = start_x + x
                        let current_y = start_y + y
                        if   (this.current_tetromino.shape[y][x]   >   0
&& !this.play_field.isFreeCell(current_x, current_y))
                        {
                            isOccupied = true
                            break labelCancelLoops
                        }
                    }
                }
                return isOccupied
            }
            return isOccupied
        }

        isBelongingFieldCurrentTetrominoPosition()
        {
            /* This method checks whether the position of the current
tetromino belongs to the field */

            let isBelonging = true
            let start_x = this.current_tetromino.coordinates.x
            let start_y = this.current_tetromino.coordinates.y
            const size = this.current_tetromino.shape.length

            labelCancelLoops:
            for (let y = 0; y < size; y++)
            {
                for (let x = 0; x < size; x++)
                {
                    let current_x = start_x + x
                    let current_y = start_y + y
                    if    (this.current_tetromino.shape[y][x]    >    0
&& !this.play_field.isBelongingFieldCoordinates(current_x, current_y))
                    {
                        isBelonging = false
                        break labelCancelLoops
```

```
                    }
                }
            }
            return isBelonging
        }


        isAbilityCurrentTetrominoPosition()
        {
            /* This method checks the possibility of placing tetrominoes
with the current position */

            let isAbility = true
            if     (this.current_tetromino     !==     undefined     &&
this.current_tetromino.shape !== undefined)
            {
                if     (this.isBelongingFieldCurrentTetrominoPosition()
&& !this.isOccupiedCurrentTetrominoPosition())
                {
                    return isAbility
                }
            }
            return false
        }

        moveTetromino(movement_activity)
        {
            /* This method implements the movement of tetrominoes */
            this.canvas.clearTetromino(this.current_tetromino)
            this.canvas.clearTetromino(this.shadow_current_tetromino)

            let            old_coordinates            =            {x:
this.current_tetromino.coordinates.x,                                    y:
this.current_tetromino.coordinates.y}
            let old_shape = this.current_tetromino.shape

            switch (movement_activity)
            {
                case MOVEMENT_ACTIVITIES.Down:
                    this.current_tetromino.coordinates.y += 1
                    this.restartFallTetrominoTimeout()
                    break
                case MOVEMENT_ACTIVITIES.Left:
                    this.current_tetromino.coordinates.x -= 1
                    break
                case MOVEMENT_ACTIVITIES.Right:
                    this.current_tetromino.coordinates.x += 1
                    break
                case MOVEMENT_ACTIVITIES.Rotate:
                    this.current_tetromino.rotate()
                    this.shadow_current_tetromino.rotate()
                    break
                case MOVEMENT_ACTIVITIES.Drop:
                    this.current_tetromino.coordinates                =
this.shadow_current_tetromino.coordinates
                    this.restartFallTetrominoTimeout()
                    break
            }
```

```
            /* Check changes ability */
            let isAbilityMove = this.isAbilityCurrentTetrominoPosition()
            if (!isAbilityMove)
            {
                this.current_tetromino.coordinates = old_coordinates
                this.current_tetromino.shape = old_shape
                this.shadow_current_tetromino.shape = old_shape
            }
            /* Draw tetrominoes */
            this.canvas.drawTetromino(this.current_tetromino)
            this.calculateShadowTetraminoPosition()
            this.canvas.drawTetromino(this.shadow_current_tetromino)

            /* Check update needed*/
            if  ((movement_activity  ===  MOVEMENT_ACTIVITIES.Down  ||
movement_activity === MOVEMENT_ACTIVITIES.Drop) && (!isAbilityMove))
            {
                this.updatePlayField()
            }
        }

        calculateShadowTetraminoPosition()
        {
            /*  This  method  calculates  the  position  for  the  shadow
tetromino */

            let       current_tetromino_coordinates       =       {x:
this.current_tetromino.coordinates.x,                                y:
this.current_tetromino.coordinates.y}
            while (this.isAbilityCurrentTetrominoPosition())
            {
                this.current_tetromino.coordinates.y += 1
            }
            this.current_tetromino.coordinates.y -= 1
            this.shadow_current_tetromino.coordinates                  =
this.current_tetromino.coordinates
            this.current_tetromino.coordinates                         =
current_tetromino_coordinates
        }


        updatePlayField()
        {
            /* This method places a tetromino on the field and moves on
to the next tetromino */

            this.play_field.placeTetromino(this.current_tetromino)
            this.processFilledRows()
            this.current_tetromino = this.next_tetromino

            /* Game over check */
            if (this.isOccupiedCurrentTetrominoPosition())
            {
                this.gameOver()
            } else {
                this.shadow_current_tetromino                          =
this.generateShadowCurrentTetromino()
```

```
                this.calculateShadowTetraminoPosition()
                this.next_tetromino = this.generateTetromino()
                this.canvas.drawTetromino(this.current_tetromino)
                this.canvas.drawTetromino(this.shadow_current_tetromino)
                this.canvas.drawNextTetromino(this.next_tetromino)
            }
        }

        gameOver(){
            /* This method saves data at the end of the game to local
memory and moves to a new page */

            let                        score_table                        =
JSON.parse(localStorage["tetris.score_table"])
            let isExistsUsername = false

            /* Updating a user's result or creating a new entry */
            for (let i = 0; i < score_table.length; i++)
            {
                if                (score_table[i].name                ==
localStorage["tetris.username"])
                {
                    score_table[i].score = this.score
                    isExistsUsername = true
                    break
                }
            }
            if (!isExistsUsername)
            {
                score_table.push({
                    name: localStorage["tetris.username"],
                    score: this.score
                })
            }

            localStorage["tetris.score_table"]                             =
JSON.stringify(score_table)
            window.location = "../game_over.html"
        }

        processFilledRows()
        {
            /* This method removes filled lines */

            const filled_lines = this.play_field.findFilledRows()
            this.play_field.removeFilledRows(filled_lines)
            this.canvas.removeFilledRows(filled_lines)
            this.updateData(filled_lines.length)
        }

        calculateScore(count_removed_rows)
        {
            /* This method calculates the score */

            if (count_removed_rows == 1)
            {
                return 100
            }
```

```
            return 2 * this.calculateScore(count_removed_rows - 1) + 100
        }

        updateData(count_removed_rows)
        {
            /* This method updates the game data */

            if (count_removed_rows > 0){
                this.score += this.calculateScore(count_removed_rows)
                this.lines += count_removed_rows
                this.level = Math.min(MAX_LEVEL, Math.floor(this.score /
100))
                this.canvas.updateData(this.level,          this.score,
this.lines)
            }
        }
    }
```

Название файла: GameOver.js

```
let score_table = JSON.parse(localStorage["tetris.score_table"])
score_table.sort(compare)

function compare(left, right)
{
    if (left.score < right.score)
    {
        return 1
    }
    return -1
}

/* Display of the table of records (only the first 5) */
let numbers_column = document.getElementById("numbers_column")
let names_column = document.getElementById("names_column")
let scores_column = document.getElementById("scores_column")

for (let i = 0; i < Math.min(5, score_table.length); i++)
{
    let number = document.createElement("div")
    number.setAttribute("class", "row")
    number.innerText = i + 1
    numbers_column.appendChild(number)

    let username = document.createElement("div")
    username.setAttribute("class", "row")
    username.innerText = score_table[i].name
    names_column.appendChild(username)

    let score = document.createElement("div")
    score.setAttribute("class", "row")
    score.innerText = score_table[i].score
    scores_column.appendChild(score)
}
```