**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работe №1**

**по дисциплине «Web - технологии»**

Тема: «Тетрис на JavaScript»

|  |  |  |
| --- | --- | --- |
| Студент гр. 0382 |  | Литягин С.М. |
| Преподаватель |  | Беляев С.А. |

Санкт-Петербург

2022

**Цель работы**

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

**Задачи**

Для достижения поставленной цели требуется решить следующие задачи:

– генерация открытого и закрытого ключей для использования шифрования;

– 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.

**Описание решения**

Сначала был сгенерирован открытый и закрытый ключи с использованием openssl. Выполненная команда представлена на рисунке 1.

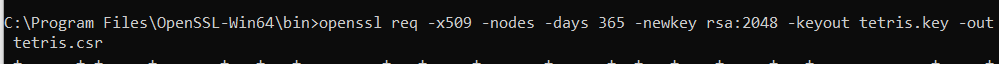


Рисунок 1 – Генерация ключей шифрования

Далее была выполнена настройка конфигурации nginx. Для этого в nginx.conf были внесены изменения, представленные на рисунке 2.

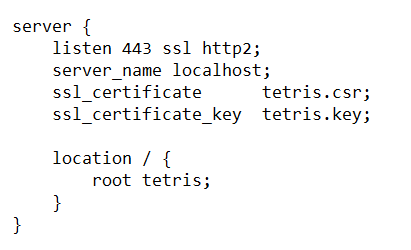


Рисунок 2 – Изменения в конфигурации nginx

Было создано три экрана игры: приветственный экран – ввод ника игрока (username.html); экран с тетрисом (tetris.html); экран с рекордами (records.html). Экраны представлены на рисунках 3-5.



Рисунок 3 – Приветственный экран игры

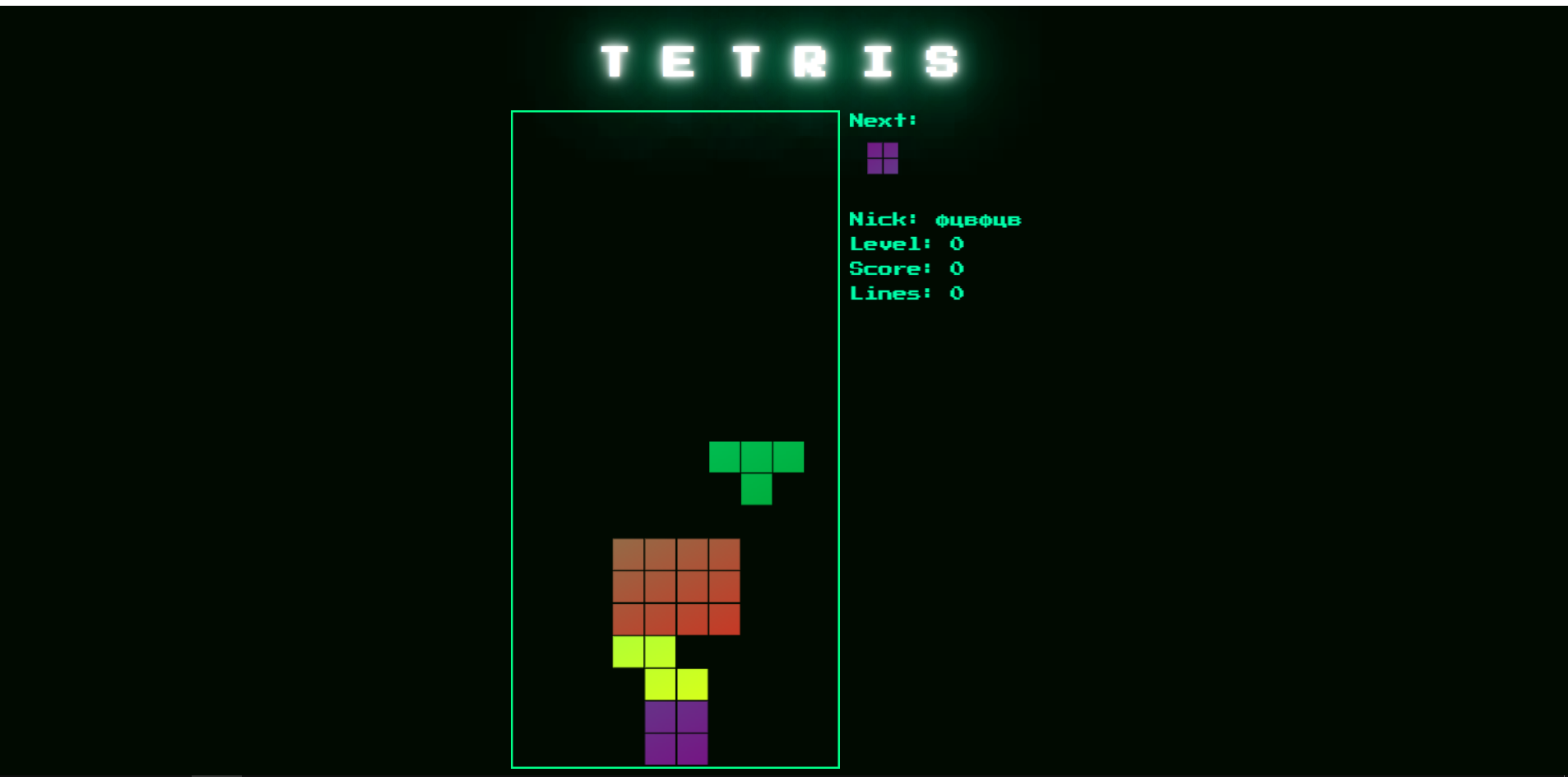


Рисунок 4 – Экран с тетрисом



Рисунок 5 – Экран с рекордами

Информация относительно правил: за удаления 1 линии начисляется 100 очков, 2 линий – 300, 3 линий – 700, 4 линий – 1500; за каждые 10 линий уровень сложности увеличивается на 1; скорость игры регулируется следующей формулой: . Управление осуществляется следующими клавишами: A – сдвинуть влево, D – сдвинуть вправо, S –сдвинуть вниз, F – уронить вниз, E – повернуть фигуру. Фигура появляется над границей игрового поля.

Было написано три основных класса: Tetris, View, Controller. В классе Tetris присутствуют следующие поля:

- playfiels – игровое поле;

- score – число очков;

- lines – число убранных линий;

- level – уровень сложности;

- nextFigure – следующая фигура;

- currentFigure – текущая фигура;

- gameOver – флаг завершения игры.

В данном классе присутствую следующие методы:

- createPlayfield – создает двумерный массив размером 10 колонок на 20 строк, заполненный нулями;

- createFigure – создает новую фигуру, генерируя число от 0 до 1 (не включая), что умножается на число фигур и округляется в меньшую сторону; затем с помощью конструкции else if выбирается, согласно сгенерированному числу, какая фигура будет создана;

- clearLines() – находит заполненные строки в игровом поле, удаляет их и дополняет массив игрового поля строками, содержащими 0;

- updateScore(lines) – обновляет количество очков, согласно числу удаленных линий на шаге;

- rotate() – поворачивает фигуру по часовой стрелке на 90 градусов, если не возникает коллизий с полем или другими фигурами;

- getState() – возвращает объект с количеством очков, уровнем сложности, количеством удаленных линий, с объектом следующей фигуры, флагом завершения игры и игровым полем, на котором зафиксировано текущее положение падающей фигуры;

- методы движения влево, вправо – moveLeft(), moveRight() соответственно;

- moveDown () – метод движения вниз – если после движения возникла коллизия – вызывается фиксирование текущего положения фигуры на поле, создание новой фигуры и замена старой, обновление счета игрока;

- moveFastDown() – быстрой спуск фигуры вниз;

- hasCollision() – функция возвращает true в случае, если фигура выходит за границу поля или наслаивается на уже имеющиеся на поле фигуры;

- fixFigure() – фиксирует положение текущей фигуры;

- newFigures() – создает новую следующую фигуру и заменяет текущую фигуру на старую следующую фигуру.

Класс View имеет множество полей, требующихся для корректной отрисовки экрана тетриса. Имеет следующие методы:

- renderGame() – отвечает за отрисовку игрового поля и панели информации для игрока (следующая фигура, ник, очки и т.п.);

- contextSettingsOne(), contextSettingsTwo() – установка различных параметров context для дальнейшей отрисовки фраз;

- renderStart() – отвечает за отрисовку начальной фразы перед игрой;

- renderGameOver({score}) – отвечает за отрисовку текста после проигрыша, содержащего очки, ник и дальнейшие указания;

- renderPlayField({playfield}) – отрисовывает игровое поле;

- renderPanel({level, score, lines, nextFigure}) – отрисовывает информацию для игрока;

- renderCell(x, y, width, height, color) – отрисовывает клетки, из которых состоят фигуры;

- clearScene() – отчищает холсты.

Класс Controller отвечает за управление таймером движения фигур вниз, а также обработку нажатий клавиш. Имеет методы:

- stopTimer() – останавливает таймер движения фигуры вниз;

- startTimer() – запускает таймер движения фигуры вниз, причем скорость движения зависит от уровня сложности игры;

- addRecords(state) – добавляет с локальное хранилище запись об текущем игроке и его количестве очков после проигрыша;

- render() – вызывает отрисовку либо экрана с тетрисом, либо экрана о проигрыше.

**Выводы**

В ходе работы была изучена работа web-сервера nginx со статическими файлами, а также создано клиентские JavaScript web-приложений – игра Tetris.