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 _____ Поршнев Р.А.

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 градусов 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. Логика программы

Вся логика программы написана на языке программирования JavaScript, которая запускается в файле main.js посредством создания объекта класса Game и запуска метода run().

Для реализации логики тетриса было реализовано 5 классов, такие как Game, Field, Figure, Display и Sound. Каждый класс строго отвечает своим задачам.

Класс Game отвечает за запуск всей игры и за управление текущей фигурой. Каждый кадр игры начинается в данном классе, здесь же происходит вызов инициализации начальных данных на игровом поле. В данном классе реализованы следующие методы:

- run() – данный метод запускает игровой цикл #figureAutoPilot(), а также слушателя клавиатуры пользователя #controller() для последующего управления фигурами;
- #figureAutoPilot() – данный метод управляет игровым циклом, а именно контролирует момент приземления фигуры, конца игры, удаления линий на поле, ускорения игры, отображения поля на экране и конца игры, а также добавления данных игры пользователя в базу данных;
- #controller() – данный метод отслеживает клавиатуру пользователя и в зависимости от клавиш вызывает метод класса Field.InitDataForMovement(figure, command) с определёнными значениями переменной command.

Класс Field отвечает за инициализацию поля, проверяет возможность поставить фигуру в определённые координаты, рисует фигуру в заданных координатах, стирает фигуру в заданных координатах, удаляет заполненные полосы и начисляет за это очки, двигает, поворачивает и сбрасывает фигуры. Реализованы следующие методы:

- isGameOver() – данный метод даёт ответ на вопрос о факте окончании игры, что является возвращаемым значением;
- hasFigureLanded() – данный метод даёт ответ на вопрос о факте приземления фигуры, что является возвращаемым значением;
- #checkInstallOfFigure(startX, startY, endX, endY, figureMatrix, shiftX) – данный метод на вход получает координаты поля, начиная с которых нужно проверить возможность постановки фигуры, матрицу фигуры figureMatrix, а

также смещение shiftX, которое нужно для корректной проверки на возможность установки фигуры в данной области в том случае, если левый верхний угол фигуры находится за границами поля; выходными данными является ответ на вопрос о факте возможности размещения фигуры в данной области;

- #drawFigure(startX, startY, endX, endY, figureMatrix, shiftX) – данный метод на вход получает область в виде координат, которую нужно закрасить, матрицу фигуры figureMatrix и смещение shiftX, которое выполняет ту же роль, что и в методе #checkInstallOfFigure(startX, startY, endX, endY, figureMatrix, shiftX);
- spawnFigure(figure) – данный метод получает на вход фигуру figure, проверяет возможность появления новой фигуры и в случае одобрения новая фигура появляется на карте; в ином случае объявляется окончание игры;
- #removeFigure(startX, startY, endX, endY, figureMatrix, shiftX) – данный метод принимает на вход координаты области, в которой нужно стереть фигуру, что нужно в том случае, когда фигура пытается поменять своё положение карте, а для этого нужно стереть старые координаты и попытаться установить новые; также данный метод принимает на вход матрицу фигуры figureMatrix и смещение shiftX, которое выполняет ту же роль, что и в методе #checkInstallOfFigure(startX, startY, endX, endY, figureMatrix, shiftX);
- InitDataForMovement(figure, command) – данный метод принимает на вход фигуру figure и команду command, которую нужно выполнить фигуре, а в зависимости от значения command вызываются определённые методы, отвечающие за движения фигуры на карте, и стирается текущая фигура с карты;
- #deleteLines(linesToDelete) – данный метод удаляет заполненные строки на карте, индексы которых хранятся во входном массиве linesToDelete;
- #addScores(numberOfLines) – данный метод принимает на вход количество линий, которое нужно удалить, и в соответствии с их количеством добавляет фиксированное количество очков;
- searchForScores() – данный метод ищет индексы заполненный линий на карте;

- #shiftFigureDown(figure, xBorder, yBorder) – данный метод принимает на вход объект класса Figure и первоначальные границы работы метода по осям xBorder и yBorder; его назначение – попытка сдвинуть фигуру на карте на 1 клетку вниз, что осуществляется за счёт проверки на сдвиг с помощью метода #checkInstallOfFigure(startX, startY, endX, endY, figureMatrix, shiftX) и независимо от результата проверки вызывается метод #drawFigure(startX, startY, endX, endY, figureMatrix, shiftX) для рисования фигуры, но с разными аргументами в случае успешного прохождения проверки и неуспешной (в случае неуспеха на карте рисуется фигура в тех координатах, где она была ранее);
- #shiftFigureLeft(figure, xBorder, yBorder) – данный метод отличается от предыдущего лишь тем, что фиггfура сдвигается на 1 клетку влево;
- #shiftFigureRight(figure, xBorder, yBorder) – данный метод отличается от предыдущего лишь тем, что фигура сдвигается на 1 клетку вправо;
- #figureRotate(figure, xBorder, yBorder) – данный метод принимает на вход объект класса Figure и первоначальные границы работы метода по осям xBorder и yBorder; его назначение – попытка повернуть фигуру на 90 градусов по часовой стрелке с помощью метода #checkInstallOfFigure(startX, startY, endX, endY, figureMatrix, shiftX) и независимо от результата проверки вызывается метод #drawFigure(startX, startY, endX, endY, figureMatrix, shiftX) для рисования фигуры, но с разными аргументами в случае успешного прохождения проверки и неуспешной (в случае неуспеха на карте рисуется фигура в том же виде, в котором была ранее);
- #figureDrop(figure) – данный метод принимает на вход объект класса Figure и его назначение – это сбросить фигуру вниз, что осуществляется за счёт циклического вызова метода сдвига фигуры вниз на 1 клетку.

Класс Figure предназначен для хранения информации и фигуре, такой как её тип, матрица, размер, начальные координаты её левого верхнего угла на карте и координаты фигуры в окне, которое показывает следующую фигуру. В данном классе реализован метод #generateNextFigure(), который возвращает случайный тип фигуры.

Класс Display предназначен для отображения всех элементов игры на экране пользователя. Реализованы следующие методы:

- #initNextFigureField() – данный метод предназначен для инициализации матрицы, в которой будет отображаться следующая фигура;
- showField(field) – данный метод получает на вход поле field и на основе данной матрицы рисует игровое поле на холсте в непосредственно в браузере;
- showWindow(field, score, level) – данный метод принимает на вход матрицу field, количество очков и уровень пользователя; его назначение – обновить данные об очках и уровне игрока и нарисовать игровое поле на холсте на основе поданной на вход матрицы;
- #insertNextFigureInNextFigureField(figure) – данный метод принимает на вход объект класса Figure и предназначен для заполнения матрицы следующей фигуры, которая отображается в специальном окошке;
- #clearNextFigureField() – данный метод предназначен для очистки матрицы, в которой хранится информация о прототипе следующей фигуры;
- showNextFigure(figure) – данный метод принимает на вход объект класса Figure и предназначен для отображения на холсте следующей фигуры;
- showGameOver() – данный метод отображает информацию о том, что игра окончена;
- addRecordInTable() – данный метод предназначен для добавления игрового результата текущего игрока в список результатов;
- showRecordTable() – данный метод предназначен для отображения таблицы рекордов;

Класс Sound предназначен для воспроизведения некоторых событий в игре, таких как приземление фигуры, исчезновение одной или нескольких линий и окончание игры. Реализованы следующие методы:

- land() – данный метод воспроизводит звук приземления фигуры;

- gameOver() – данный метод предназначен для воспроизведения звука конца игры;
- linesToDelete() – данный метод воспроизводит звук удаления одной или нескольких линий.

2. Разметка и дизайн

Разметка была реализована на HTML, а дизайн на CSS. Особенностью данной игры является использование специфического шрифта, близкого к тому, что использовался в оригинальной игре. Связь HTML и CSS элементов с JavaScript осуществлялась преимущественно через идентификаторы элементов. Особенностью дизайна является использование эффектов затухания и расцвета элементов в такие моменты, как начало игры, проигрыш и отображение таблицы рекордов.

Тестирование.

Страница авторизации представлена на Рисунке 1.

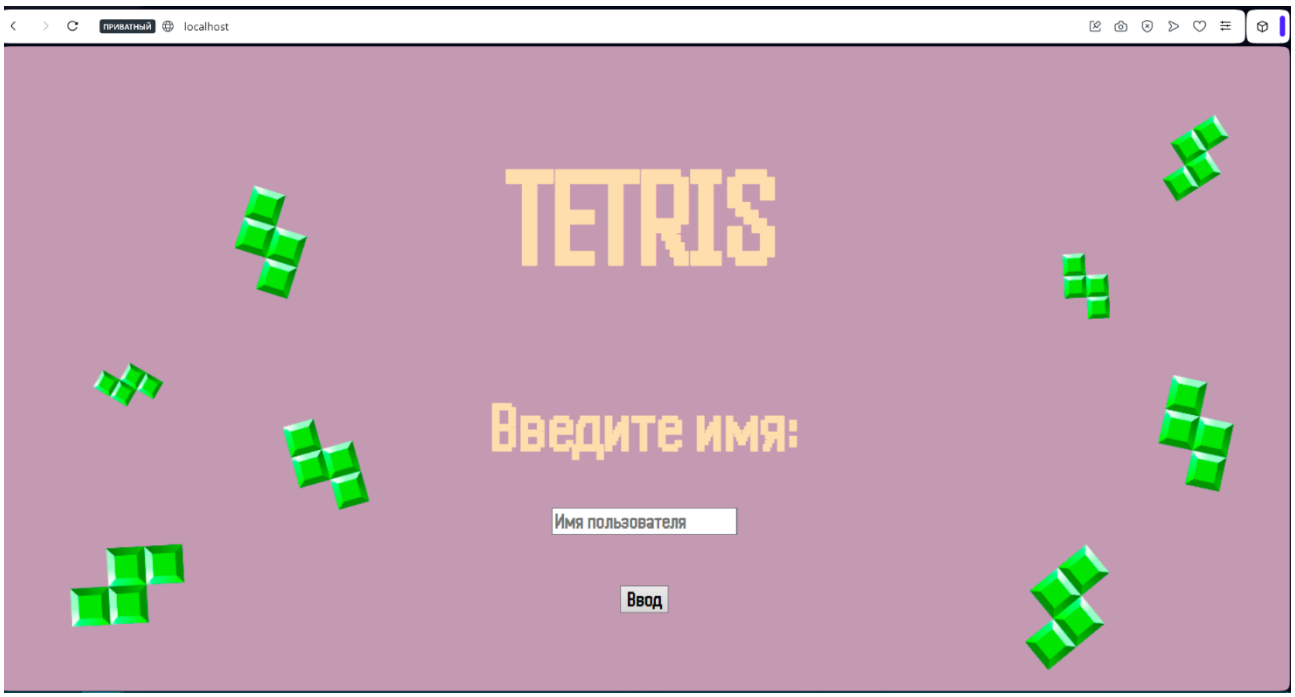

Рисунок 1 – Страница авторизации

Демонстрация игрового процесса представлена на Рисунке 2.

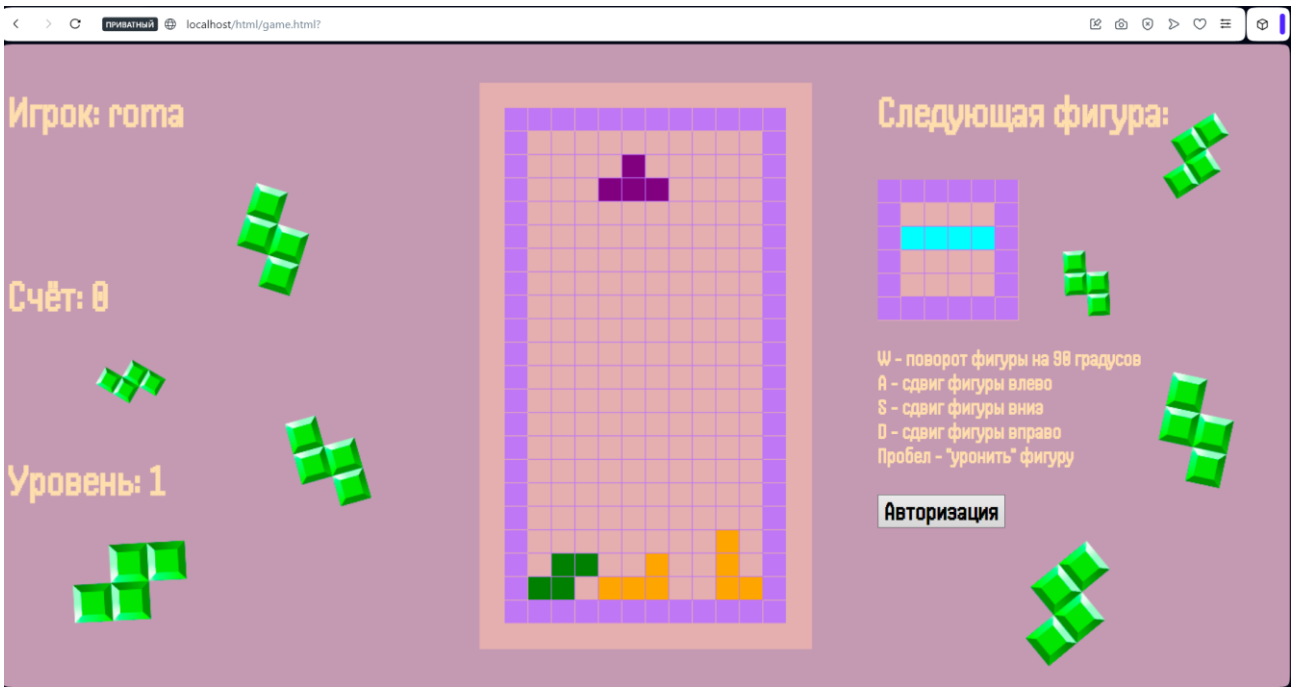

Рисунок 2 – Демонстрация игрового процесса

Демонстрации окончания игры представлена на Рисунке 3.

Рисунок 3 – Демонстрация окончания игры

Демонстрация таблицы рекордов представлена на Рисунке 4

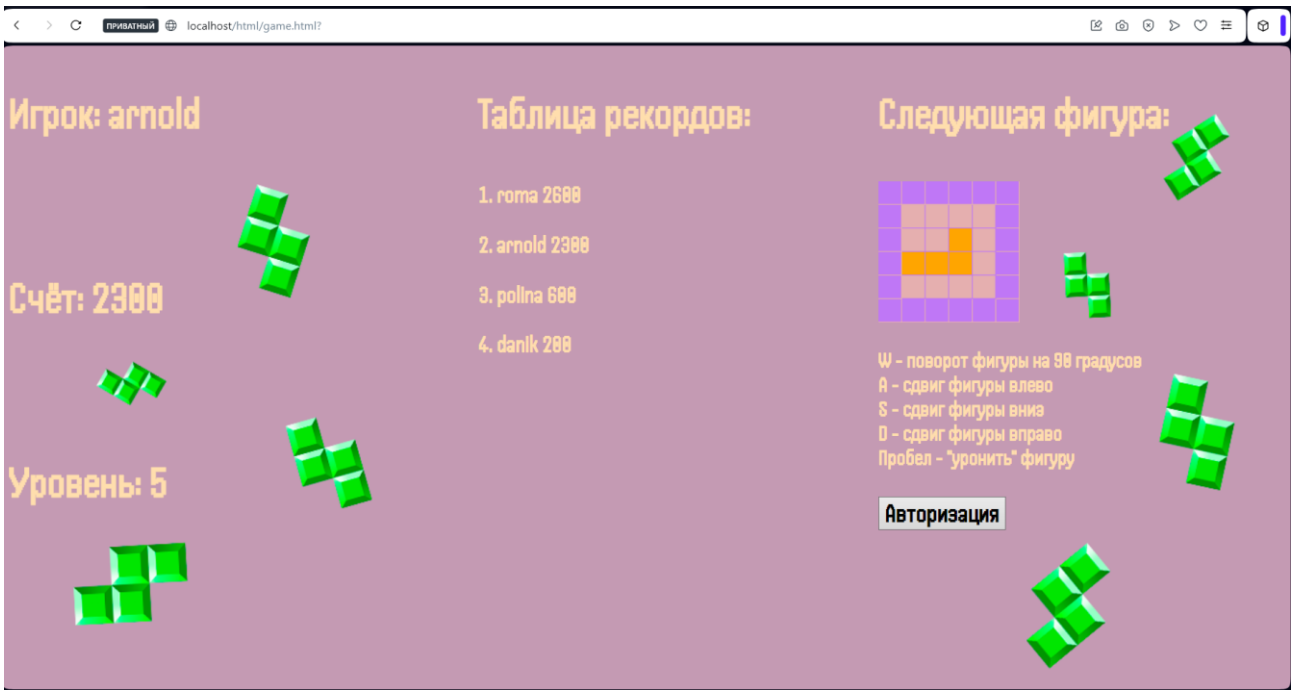

Рисунок 4 – Демонстрация таблицы рекордов

Выводы.

Изучена работа web-сервера nginx со статическими файлами и создание клиентских JavaScript web-приложений на примере игры “Тетрис”.

Разработана игра “Тетрис” на языке JavaScript в объектно-ориентированной стиле, сделана гипертекстовая разметка двух страниц HTML и оформление к ним на CSS.