Хабаровский институт инфокоммуникаций (филиал)

Федерального государственного бюджетного образовательного учреждения высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

кафедра «Информационных технологий»

##### Лабораторная работа №4

по дисциплине Интернет-технологии

на тему: игра «Tetris»

Выполнил студент 2 курса

группы ПОВТ-20Д

ФИО Леуненко А.О.

Шифр: 09.03.01.01

Проверил: Манжула И.С.

Хабаровск 2021 г.

Код файла разметки

**index.html**

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="utf-8">  
 <title>Тетрис</title>  
 <link rel="stylesheet" href="assets/css/style.css" />  
  
 <link rel="preconnect" href="https://fonts.googleapis.com">  
 <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>  
 <link href="https://fonts.googleapis.com/css2?family=Oswald:wght@500&display=swap" rel="stylesheet">  
</head>  
  
<body>  
<!-- рисуем поле для игры -->  
<canvas width="320" height="640" id="game"></canvas>  
  
<!-- добавляем область для вывода информации о текущей игре -->  
<canvas style="border: 0" width="320" height="100" id="score" ></canvas>  
  
<script src="assets/js/main.js"></script>  
</body>  
</html>

Код файла стиля

**style.css**

/\*настройки стилей страницы\*/  
  
html, body {  
 height: 100%;  
 margin: 0;  
 font-family: 'Oswald', sans-serif;  
}  
/\*делаем чёрный фон и выравниваем всё по центру\*/  
body {  
 background: black;  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 font-family: 'Oswald', sans-serif;  
}  
/\*толщина обводки холста — 1 пиксель, белый цвет\*/  
canvas {  
 border: 1px solid white;  
 font-family: 'Oswald', sans-serif;  
}

Код файла Javascript

**main.js**

// License CC0 1.0 Universal  
// https://tetris.fandom.com/wiki/Tetris\_Guideline  
  
// получаем доступ к основному холсту  
const ***canvas*** = ***document***.getElementById('game');  
const ***context*** = ***canvas***.getContext('2d');  
  
// получаем доступ к холсту с игровой статистикой  
const ***canvasScore*** = ***document***.getElementById('score');  
const ***contextScore*** = ***canvasScore***.getContext('2d');  
  
// размер квадратика  
const ***grid*** = 32;  
// массив с последовательностями фигур, на старте — пустой  
var ***tetrominoSequence*** = [];  
  
// с помощью двумерного массива следим за тем, что находится в каждой клетке игрового поля  
// размер поля — 10 на 20, и несколько строк ещё находится за видимой областью  
var ***playfield*** = [];  
  
// заполняем сразу массив пустыми ячейками  
for (let row = -2; row < 20; row++) {  
 ***playfield***[row] = [];  
  
 for (let col = 0; col < 10; col++) {  
 ***playfield***[row][col] = 0;  
 }  
}  
  
// как рисовать каждую фигуру  
// https://tetris.fandom.com/wiki/SRS  
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],  
 ]  
};  
  
// цвет каждой фигуры  
const ***colors*** = {  
 'I': 'cyan',  
 'O': 'yellow',  
 'T': 'purple',  
 'S': 'green',  
 'Z': 'red',  
 'J': 'blue',  
 'L': 'orange'  
};  
  
// счётчик  
let ***count*** = 0;  
// текущая фигура в игре  
let ***tetromino*** = getNextTetromino();  
// следим за кадрами анимации, чтобы если что — остановить игру  
let ***rAF*** = null;  
// флаг конца игры, на старте — неактивный  
let ***gameOver*** = false;  
  
// количество набранных очков на старте  
let ***score*** = 0;  
// рекорд игры  
let ***record*** = 0;  
// текущий уровень сложности  
let ***level*** = 1;  
// имя игрока с наибольшим рейтингом  
let ***recordName*** = '';  
  
// спрашиваем имя игрока при запуске  
***name*** = prompt("Ваше имя", "");  
  
  
// Узнаём размер хранилища  
var ***Storage\_size*** = ***localStorage***.length;  
// Если в хранилище уже что-то есть…  
if (***Storage\_size*** > 0) {  
 // …то достаём оттуда значение рекорда и имя чемпиона  
 ***record*** = ***localStorage***.record;  
 ***recordName*** = ***localStorage***.recordName;  
}  
  
// Функция возвращает случайное число в заданном диапазоне  
// https://stackoverflow.com/a/1527820/2124254  
function getRandomInt(min, max) {  
 min = ***Math***.ceil(min);  
 max = ***Math***.floor(max);  
  
 return ***Math***.floor(***Math***.random() \* (max - min + 1)) + min;  
}  
  
// создаём последовательность фигур, которая появится в игре  
//https://tetris.fandom.com/wiki/Random\_Generator  
function generateSequence() {  
 // тут — сами фигуры  
 const sequence = ['I', 'J', 'L', 'O', 'S', 'T', 'Z'];  
  
 while (sequence.length) {  
 // случайным образом находим любую из них  
 const rand = getRandomInt(0, sequence.length - 1);  
 const name = sequence.splice(rand, 1)[0];  
 // помещаем выбранную фигуру в игровой массив с последовательностями  
 ***tetrominoSequence***.push(name);  
 }  
}  
  
// получаем следующую фигуру  
function getNextTetromino() {  
 // если следующей нет — генерируем  
 if (***tetrominoSequence***.length === 0) {  
 generateSequence();  
 }  
 // берём первую фигуру из массива  
 const name = ***tetrominoSequence***.pop();  
 // сразу создаём матрицу, с которой мы отрисуем фигуру  
 const matrix = ***tetrominos***[name];  
  
 // I и O стартуют с середины, остальные — чуть левее  
 const col = ***playfield***[0].length / 2 - ***Math***.ceil(matrix[0].length / 2);  
  
 // I начинает с 21 строки (смещение -1), а все остальные — со строки 22 (смещение -2)  
 const row = name === 'I' ? -1 : -2;  
  
 // вот что возвращает функция  
 return {  
 name: name, // название фигуры (L, O, и т. д.)  
 matrix: matrix, // матрица с фигурой  
 row: row, // текущая строка (фигуры стартуют за видимой областью холста)  
 col: col // текущий столбец  
 };  
}  
  
// поворачиваем матрицу на 90 градусов  
// https://codereview.stackexchange.com/a/186834  
function rotate(matrix) {  
 const N = matrix.length - 1;  
 const result = matrix.map((row, i) =>  
 row.map((val, j) => matrix[N - j][i])  
 );  
 // на входе матрица, и на выходе тоже отдаём матрицу  
 return result;  
}  
  
// проверяем после появления или вращения, может ли матрица (фигура) быть в этом месте поля или она вылезет за его границы  
function isValidMove(matrix, cellRow, cellCol) {  
 // проверяем все строки и столбцы  
 for (let row = 0; row < matrix.length; row++) {  
 for (let col = 0; col < matrix[row].length; col++) {  
 if (matrix[row][col] && (  
 // если выходит за границы поля…  
 cellCol + col < 0 ||  
 cellCol + col >= ***playfield***[0].length ||  
 cellRow + row >= ***playfield***.length ||  
 // …или пересекается с другими фигурами  
 ***playfield***[cellRow + row][cellCol + col])  
 ) {  
 // то возвращаем, что нет, так не пойдёт  
 return false;  
 }  
 }  
 }  
 // а если мы дошли до этого момента и не закончили раньше — то всё в порядке  
 return true;  
}  
  
// когда фигура окончательна встала на своё место  
function placeTetromino() {  
 // обрабатываем все строки и столбцы в игровом поле  
 for (let row = 0; row < ***tetromino***.matrix.length; row++) {  
 for (let col = 0; col < ***tetromino***.matrix[row].length; col++) {  
 if (***tetromino***.matrix[row][col]) {  
  
 // если край фигуры после установки вылезает за границы поля, то игра закончилась  
 if (***tetromino***.row + row < 0) {  
 return showGameOver();  
 }  
 // если всё в порядке, то записываем в массив игрового поля нашу фигуру  
 ***playfield***[***tetromino***.row + row][***tetromino***.col + col] = ***tetromino***.name;  
 }  
 }  
 }  
  
 // проверяем, чтобы заполненные ряды очистились снизу вверх  
 for (let row = ***playfield***.length - 1; row >= 0; ) {  
 // если ряд заполнен  
 if (***playfield***[row].every(cell => !!cell)) {  
  
 ***score*** += 10;  
 // считаем уровень  
 ***level*** = ***Math***.floor(***score***/100) + 1;  
 // если игрок побил прошлый рекорд  
 if (***score*** > ***record***) {  
 // ставим его очки как рекорд  
 ***record*** = ***score***;  
 // заносим в хранилище значение рекорда  
 ***localStorage***.record = ***record***;  
 // меняем имя чемпиона  
 ***recordName*** = name;  
 // заносим в хранилище его имя  
 ***localStorage***.recordName = ***recordName***;  
 }  
  
 // очищаем его и опускаем всё вниз на одну клетку  
 for (let r = row; r >= 0; r--) {  
 for (let c = 0; c < ***playfield***[r].length; c++) {  
 ***playfield***[r][c] = ***playfield***[r-1][c];  
 }  
 }  
 }  
 else {  
 // переходим к следующему ряду  
 row--;  
 }  
 }  
 // получаем следующую фигуру  
 ***tetromino*** = getNextTetromino();  
}  
  
// показываем надпись Game Over  
function showGameOver() {  
 // прекращаем всю анимацию игры  
 cancelAnimationFrame(***rAF***);  
 // ставим флаг окончания  
 ***gameOver*** = true;  
 // рисуем чёрный прямоугольник посередине поля  
 ***context***.fillStyle = 'black';  
 ***context***.globalAlpha = 0.75;  
 ***context***.fillRect(0, ***canvas***.height / 2 - 30, ***canvas***.width, 60);  
 // пишем надпись белым моноширинным шрифтом по центру  
 ***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);  
}  
  
function showScore() {  
 ***contextScore***.clearRect(0,0,***canvasScore***.width,***canvasScore***.height);  
 ***contextScore***.globalAlpha = 1;  
 ***contextScore***.fillStyle = 'white';  
 ***contextScore***.font = '18px Courier New';  
 ***contextScore***.fillText('Уровень: ' + ***level***, 15, 20);  
 ***contextScore***.fillText('Очков: ' + ***score***, 15, 50);  
 ***contextScore***.fillText('Чемпион: ' + ***recordName***, 160, 20);  
 ***contextScore***.fillText('Рекорд: ' + ***record***, 160, 50);  
  
}  
  
// главный цикл игры  
function loop() {  
 // начинаем анимацию  
 ***rAF*** = requestAnimationFrame(loop);  
 // очищаем холст  
 ***context***.clearRect(0,0,***canvas***.width,***canvas***.height);  
  
 // рисуем игровое поле с учётом заполненных фигур  
 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];  
  
 // рисуем всё на один пиксель меньше, чтобы получился эффект «в клетку»  
 ***context***.fillRect(col \* ***grid***, row \* ***grid***, ***grid***-1, ***grid***-1);  
 }  
 }  
 }  
  
 // выводим статистику  
 showScore();  
  
 // рисуем текущую фигуру  
 if (***tetromino***) {  
  
 // фигура сдвигается вниз каждые 36 кадров минус значение текущего уровня. Чем больше уровень, тем быстрее падает.  
 if (++***count*** > (36 - ***level***)) {  
 ***tetromino***.row++;  
 ***count*** = 0;  
  
 // если движение закончилось — рисуем фигуру в поле и проверяем, можно ли удалить строки  
 if (!isValidMove(***tetromino***.matrix, ***tetromino***.row, ***tetromino***.col)) {  
 ***tetromino***.row--;  
 placeTetromino();  
 }  
 }  
  
 // не забываем про цвет текущей фигуры  
 ***context***.fillStyle = ***colors***[***tetromino***.name];  
  
 // отрисовываем её  
 for (let row = 0; row < ***tetromino***.matrix.length; row++) {  
 for (let col = 0; col < ***tetromino***.matrix[row].length; col++) {  
 if (***tetromino***.matrix[row][col]) {  
  
 // и снова рисуем на один пиксель меньше  
 ***context***.fillRect((***tetromino***.col + col) \* ***grid***, (***tetromino***.row + row) \* ***grid***, ***grid***-1, ***grid***-1);  
 }  
 }  
 }  
 }  
}  
  
// следим за нажатиями на клавиши  
***document***.addEventListener('keydown', function(e) {  
 // если игра закончилась — сразу выходим  
 if (***gameOver***) return;  
  
 // стрелки влево и вправо  
 if (e.which === 37 || e.which === 39) {  
 const col = e.which === 37  
 // если влево, то уменьшаем индекс в столбце, если вправо — увеличиваем  
 ? ***tetromino***.col - 1  
 : ***tetromino***.col + 1;  
  
 // если так ходить можно, то запоминаем текущее положение  
 if (isValidMove(***tetromino***.matrix, ***tetromino***.row, col)) {  
 ***tetromino***.col = col;  
 }  
 }  
  
 // стрелка вверх — поворот  
 if (e.which === 38) {  
 // поворачиваем фигуру на 90 градусов  
 const matrix = rotate(***tetromino***.matrix);  
 // если так ходить можно — запоминаем  
 if (isValidMove(matrix, ***tetromino***.row, ***tetromino***.col)) {  
 ***tetromino***.matrix = matrix;  
 }  
 }  
  
 // стрелка вниз — ускорить падение  
 if(e.which === 40) {  
 // смещаем фигуру на строку вниз  
 const row = ***tetromino***.row + 1;  
 // если опускаться больше некуда — запоминаем новое положение  
 if (!isValidMove(***tetromino***.matrix, row, ***tetromino***.col)) {  
 ***tetromino***.row = row - 1;  
 // ставим на место и смотрим на заполненные ряды  
 placeTetromino();  
 return;  
 }  
 // запоминаем строку, куда стала фигура  
 ***tetromino***.row = row;  
 }  
});  
  
// старт игры  
***rAF*** = requestAnimationFrame(loop)

Скриншот работы:

