**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

Курсовая РАБОТА

**по дисциплине «Web-технологии»**

Тема: Игра на JavaScript

|  |  |  |
| --- | --- | --- |
| Студент гр. 9304 |  | Прокофьев М.Д. |
| Преподаватель |  | Беляев С.А. |

Санкт-Петербург

2020

**ЗАДАНИЕ**

**на курсовую работу (курсовой проект)**

|  |  |  |
| --- | --- | --- |
| Студент Прокофьев М.Д. | | |
| Группа 9304 | | |
| Тема работы: Игра на JavaScript | | |
| Исходные данные:  1. Минимум 2 уровня игры  2. Реализованы все менеджеры в соответствии с  учебным пособием (УП)  3. Есть таблица рекордов  4. Есть препятствия  5. Есть «интеллектуальные» противники и «бонусы»  6. Используются tiles с редактором Tiled | | |
|  | | |
|  | | |
| Дата выдачи задания: 06.09.2021 | | |
| Дата сдачи реферата: 22.11.2021 | | |
| Дата защиты реферата: 22.11.2021 | | |
| Студент(ка) |  | Прокофьев М.Д. |
| Преподаватель |  | Беляев С.А. |

**Аннотация**

Курсовой проект представляет из себя игру, написанную на языке JavaScript(ES6).

**Summary**

The course project is a game written in JavaScript (ES6).**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 5 |
| 1. | Формальная постановка задачи | 6 |
| 2. | Ход выполнения работы | 7 |
| 2.1. | Предисловие к ходу выполнения работы | 7 |
| 2.2. | Менеджер карты | 7 |
| 2.3 | Менеджер спрайтов | 7 |
| 2.4 | Менеджер событий | 8 |
| 2.5 | Менеджер физики | 8 |
| 2.6 | Менеджер звуков | 9 |
| 2.7 | Менеджер игры | 9 |
| 2.8 | Таблица рекордов | 9 |
| 3. | Описание и тестирование приложения | 10 |
| 3.1. | Описание приложения | 10 |
| 3.2. | Тестирование приложения | 10 |
|  | Заключение | 17 |
|  | Список использованных источников | 18 |
|  | Приложение А. Название приложения | 19 |

**введение**

**Цель работы:**

Изучение языка JavaScript(ES6), структуризации игрового web-проекта. Создание web-приложения, которое представляет из себя игру, написанную на JavaScript(ES6).

**1. Формальная постановка задачи**

Написать веб-приложение, игру, на JavaScript(ES6) в соответствии с следующими требованиями:

1. Минимум 2 уровня игры

2. Реализованы все менеджеры в соответствии с

учебным пособием:

- Менеджер карты

- Менеджер игры

- Менеджер спрайтов

- Менеджер событий

- Менеджер физики

- Менеджер аудио

3. Есть таблица рекордов

4. Есть препятствия

5. Есть «интеллектуальные» противники и «бонусы»

6. Используются tiles с редактором Tiled

**2. ход выполнения работы**

**2.1. Предисловие к ходу выполнения работы**

Для выполнения работы были реализованы менеджеры, которые в совокупности при компиляции образуют игру. Каждый из менеджеров по-своему структурирован и выполняет свои функции.

**2.2. Менеджер карты**

Менеджер карты предназначен для загрузки карты. Передвижение на карте осуществляется по клеткам, соответственно каждый тайл = клетка. Соответственно, менеджер карты содержит информацию о клетках карты. Для загрузки карты используется метод LoadMap(), данный метод предполагает загрузку карты из JSON файла. ParseMap() используется для считывания 'ландшафта' из JSON файла. ParseEntities() используется для считывания объектов в JSON файле.

**2.3. Менеджер спрайтов**

Менеджер спрайтов представляет из себя менеджер, который генерирует спрайты, текущее изображение объекта. Данный менеджер представляет из себя несколько методов, которые предполагают загрузку и отображение объекта. Метод loadImg() используется для загрузки изображения, данный метод записывает в менеджер загруженное изображение. loadAtlas() используется для загрузки атласа – набора изображений для спрайта, при этом, данный метод принимает два параметра – JSON файл атласа и изображение . Метод parseAtlas() записывает атлас в спрайт. drawSprite() служит для рисования спрайта, а getSprite() – соответственно, для получения.

**2.4. Менеджер событий**

Данный менеджер служит для обработки нажатий клавиш пользователем. Используются 5 клавиш: ‘W’,’A’,’S’,’D’ – для перемещения вверх, влево, вниз, вправо и пробел – для выстрелов. Функцией инициализации данного менеджера является функция ‘setup’, в ней задаются переназначения клавиш и методов. В данном менеджере функциями событий(в данном случае, нажатий) являются функции: onMouseDown(), onMouseUp() – соответственно, для нажатия и отпуска мыши, а также функции: onKeyDown() и onKeyUp() – для нажатия и отпуска клавиш клавиатуры.

**2.5. Менеджер физики**

Данный менеджер предназначен для обработки физики в игре. Он содержит методы взаимодействия объектов, как с окружающей средой, так и друг с другом. Методы moveObject() и moveObjectNotDraw() предназначены за смещение объекта на карте. В обоих методах перед смещением передвижения проверяются некоторые условия, а именно, выходит ли объект за границы карты и сталкивается ли объект с препятствием, если оба условия неверны, то обьект перемещается в нужную позицию. Отличия методов moveObject() и moveObjectNotDraw() заключаются в том, что в одном случае объект рисуется, в другом нет. Также, в этом менеджеры присутствуют и другие методы. Такие как update() – Метод, определяющий каждый такт игры новое положение объекта на карте, в зависимости от его текущей скорости и прошлого расположения, также в этом методе проверяется коллизия текущего объекта с другими объектами. Данная проверка осуществляется с помощью метода entityAtXY(). Данный метод принимает исходный объект и координаты ‘x’ и ‘y’. В данном методе по координате ‘x’ и ‘y’ производится поиск сущности, имя которой не соответствует имени исходного объекта, если такой сущности нет, возвращается false, иначе – true.

**2.5. Менеджер звуков**

В данном менеджере используется обработка и загрузка звуков. Инициализация менеджера представляет функция init(), где производятся первоначальные настройки. Функция load() используется для загрузки звука, также в данной функции используется play(), которая служит для воспроизведения соответствующего звука.

**2.7. Менеджер игры**

Менеджер игры представляет из себя менеджер, который использует("собирает") остальные менеджеры для создания полноценной игры, где имеется управление игрока, смена спрайтов, физика, звуки, а также отображение карты. Помимо этого, менеджер игры содержит фабрики сущностей, которые используются в игре: Игрок, Враг, Бонус. Также, данный менеджер содержит дополнительные методы для реализации игрового процесса, такие как: toWin() - метод, обрабатывающий выигрыш игрока и его переход на следующий уровень. На рисунке 2 представлена реализация менеджера игры

**2.8. Таблица рекордов**

В игре реализована таблица рекордов, где по окончанию игры в локальную память записываются данные имени и количества жизней игрока, при прохождении уровня.

**3. описание и тестирование приложения**

**3.1. Описание приложения**

Веб-программа игры была написана на JavaScript(ES6).

Смысл игры следующий: имеется главный герой и несколько противников. Изначально, игрок имеет 2 жизни. Если противник коснётся игрока, игрок получит ранение, и у него отнимется 1 жизнь. В случае, если игрок потеряет все жизни, игра прекращается. Герой может неограниченно ходить(кнопки: WASD) и стрелять(кнопка: Space) в четырёх направлениях: вверх, вправо, вниз, влево. Направление снаряда зависит от текущего поворота игрока после его движения. Если снаряд касается врага, враг погибает. Цель игры - уничтожить всех врагов. В случае уничтожения всех врагов, происходит переход на следующий уровень(уровней всего 2).

**3.2. Тестирование**

1) Игра состоит из двух уровней. Соответствующие уровни представлены на рисунках 1-2:

****

Рисунок 1 – Иллюстрация первого уровня

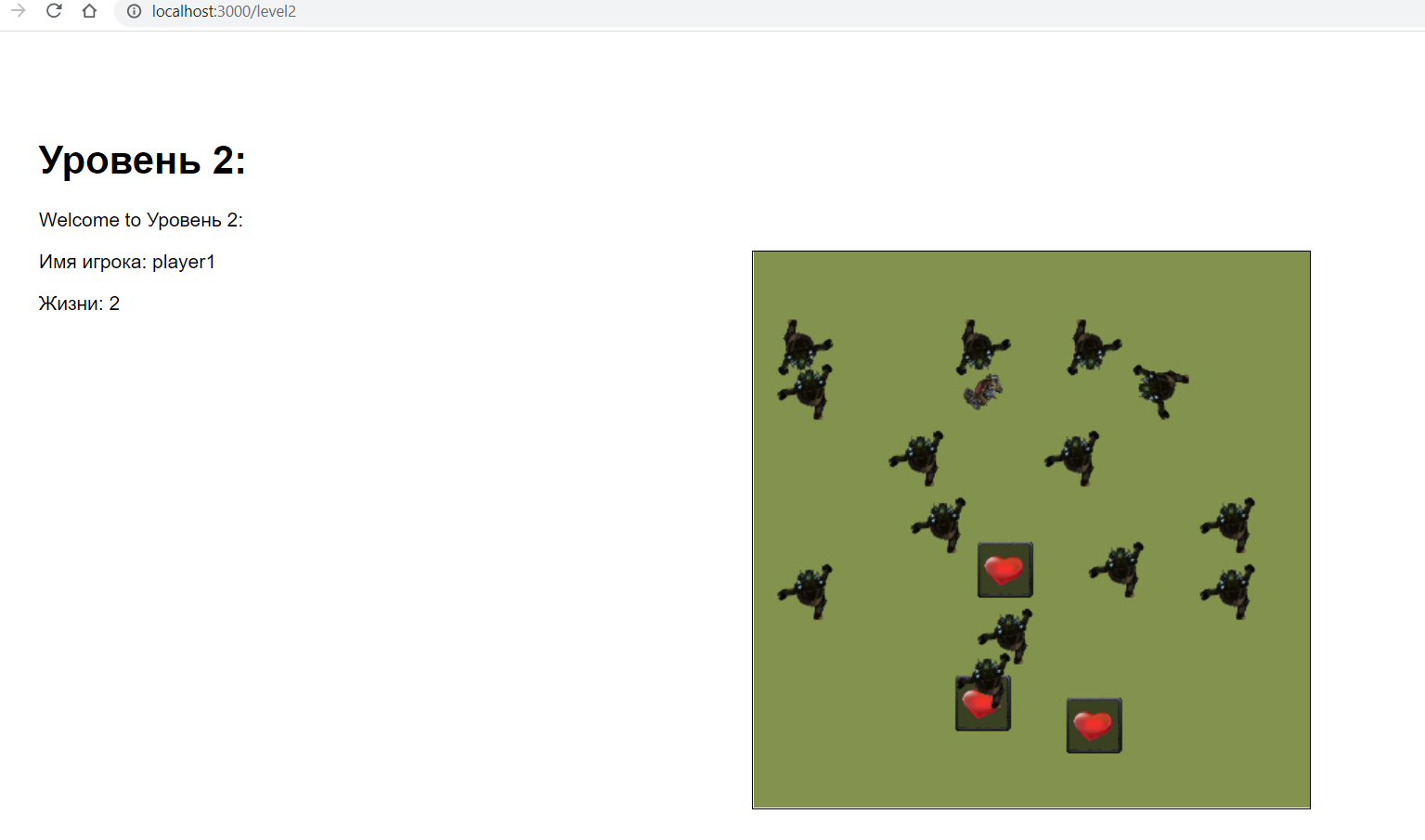
****

Рисунок 2 – Иллюстрация второго уровня

2) Для выстрелов игрок использует клавишу ‘space’(пробел). Иллюстрация стрельбы представлена на рисунке 3:

****

Рисунок 3 – Иллюстрация стрельбы

3) При столкновении с врагом игрок теряет жизни, игра завершается, если игрок теряет все жизни. Данные ситуации продемонстрированы на рисунках 4-5:

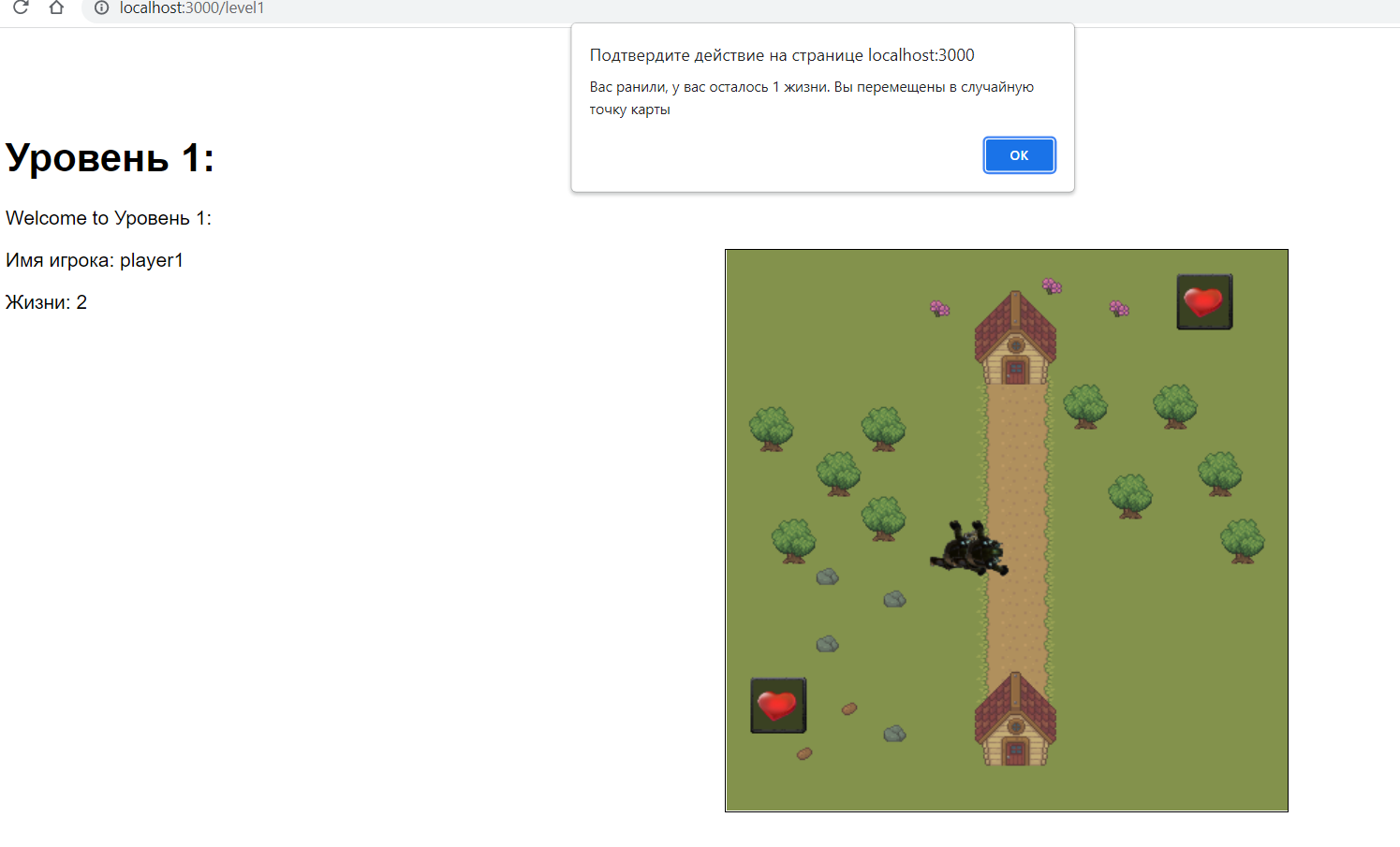
****

Рисунок 4 – Ситуация столкновения игрока с врагом, если жизни игрока больше единицы

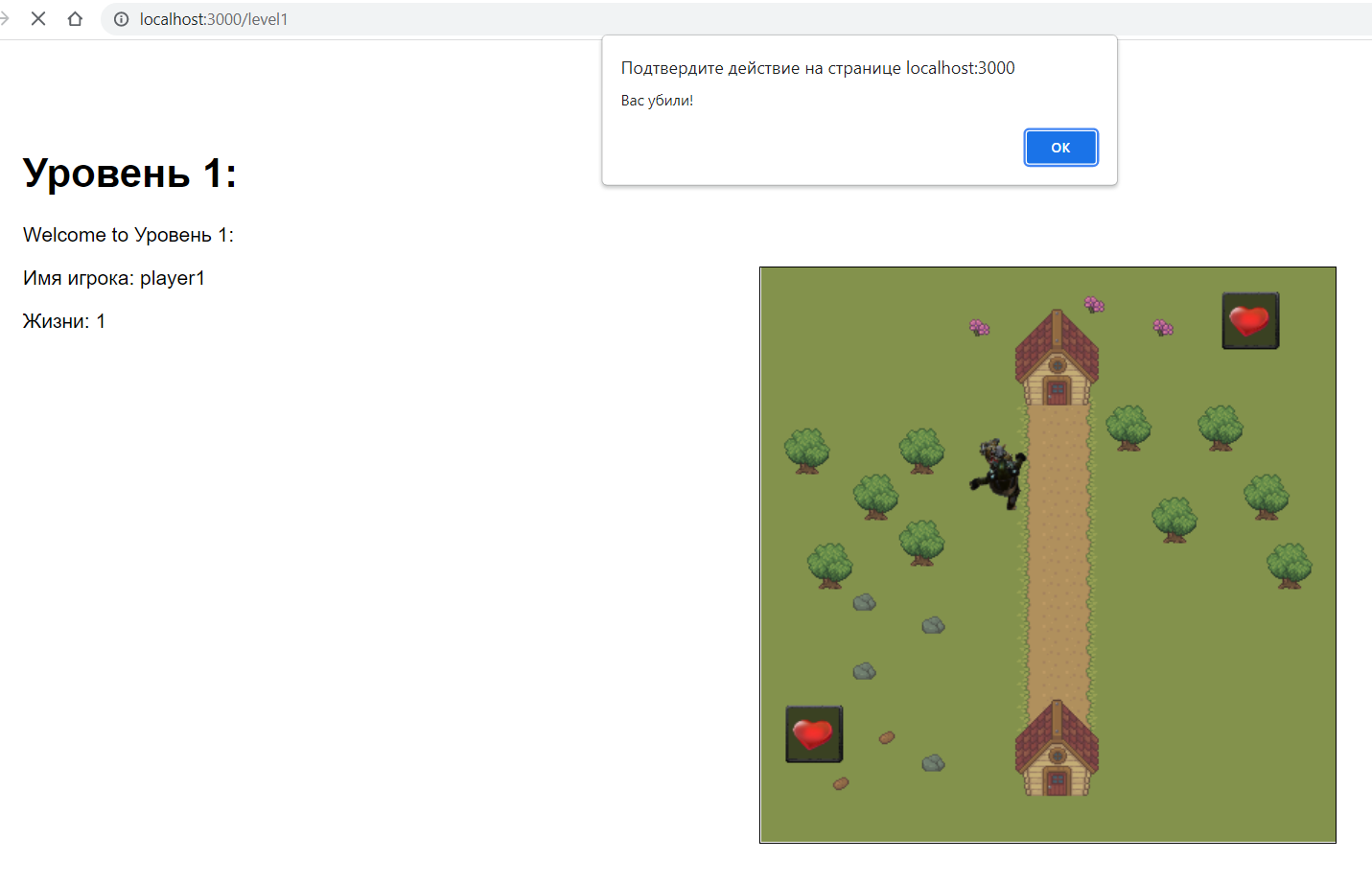


Рисунок 5 – Ситуация столкновения игрока с врагом, если игрок потерял все жизни

4) В случае, если игрок подбирает бонусные жизни, выводится соответствующее сообщение, а также увеличивается параметр жизни на единицу. Это показано на рисунке 6:



Рисунок 6 – Игрок подбирает бонус

5) В случае выигрыша на первом уровне, игрок переходит на второй. В случае выигрыша на втором уровне, игра заканчивается. Это показано на рисунках 7-8

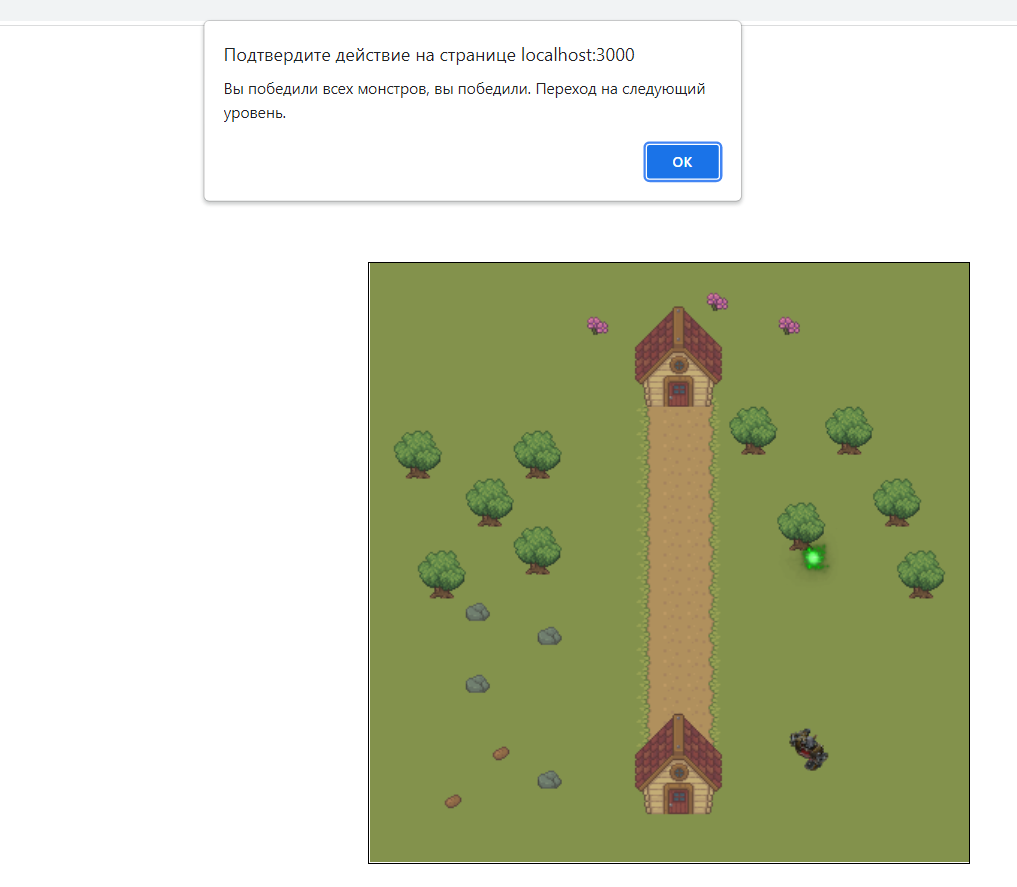


Рисунок 7 – Конец первого уровня

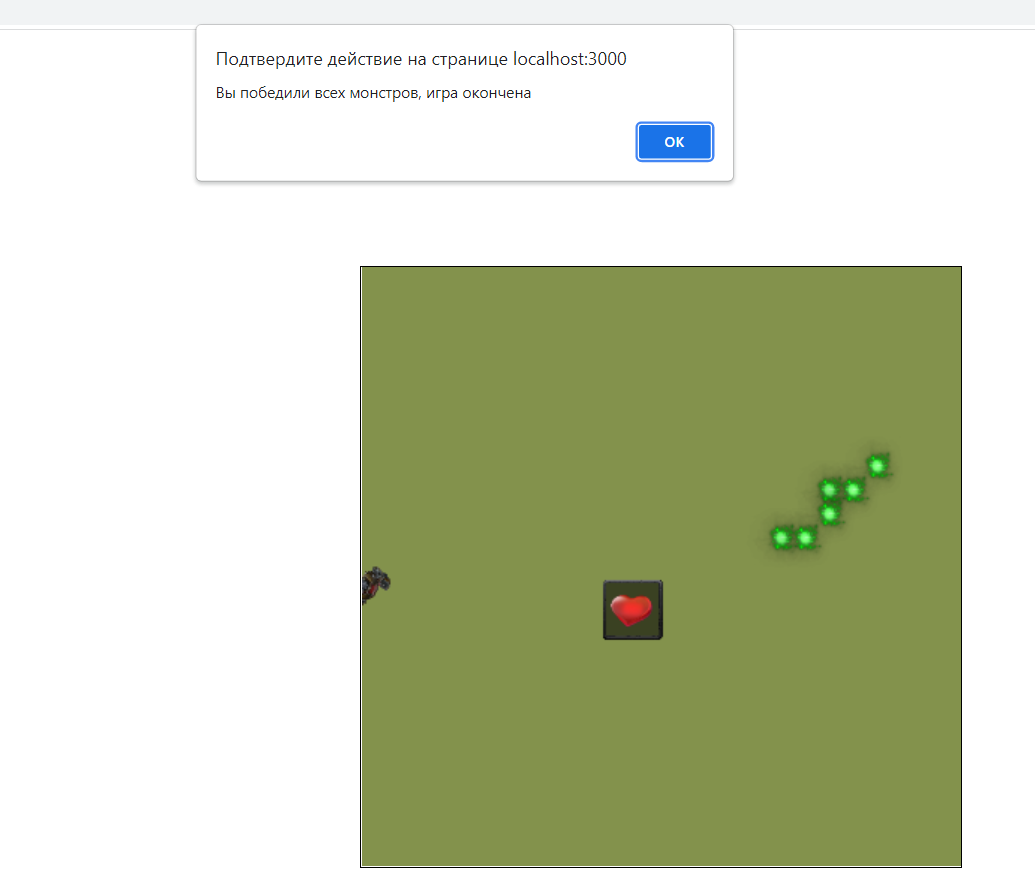


Рисунок 8 – Конец второго уровня

6) По окончанию игры, в случае проигрыша, или выигрыша, результаты игрока и его имя записываются в таблицу рекордов, которая находится на начальной странице (странице регистрации пользователя). Таблица рекордов представлена на рисунке 9:

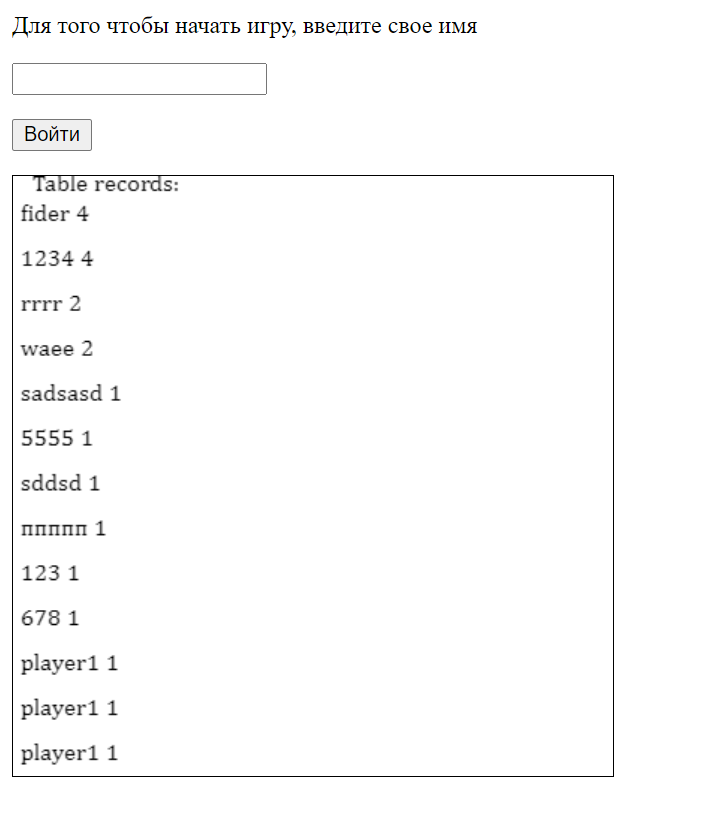


Рисунок 9 – Таблица рекордов на начальной странице

**4. Заключение**

В ходе выполнения работы были получен опыт работы в JavaScript.

Была написана игра, в которой реализовано 6 менеджеров: менеджер карты, менеджер игры, менеджер спрайтов, менеджер событий, Менеджер физики, менеджер аудио

**список использованных источников**

1. Беляев С. А. “Разработка игр на языке JavaScript”. М.: Лань, 2020.

**приложение А**

**Исходный код приложения**

Название файла: eventManager.js

var eventsManager = {

bind: [],

action: [],

setup: function (canvas) {

this.bind[87] = 'up';

this.bind[65] = 'left';

this.bind[83] = 'down';

this.bind[68] = 'right';

this.bind[32] = 'fire';

canvas.addEventListener("mousedown", this.onMouseDown);

canvas.addEventListener("mouseup", this.onMouseUp);

document.body.addEventListener("keydown", this.onKeyDown);

document.body.addEventListener("keyup", this.onKeyUp);

},

onMouseDown: function (event) {

eventsManager.action["fire"] = true;

},

onMouseUp: function (event) {

eventsManager.action["fire"] = false;

},

onKeyDown: function (event) {

var action = eventsManager.bind[event.keyCode];

active\_player.move\_x = 1;

if (action)

eventsManager.action[action] = true;

},

onKeyUp: function (event) {

var action = eventsManager.bind[event.keyCode];

if (action)

eventsManager.action[action] = false;

}

};

Название файла: gameManager.js

let factory = {}

let Entities = []

let fireNum = 0;

let player = 0;

let laterKill = []

let ended = 0;

let isGo = 1;

const player\_name\_inner = document.getElementById('player\_name');

const scores\_inner = document.getElementById('scores');

function getRandomInt(max) {

return Math.floor(Math.random() \* max);

}

function initPlayer(obj) {

active\_player = obj

}

function kill(obj) {

laterKill.push(obj);

}

function draw(ctx) {

active\_player.draw(ctx)

}

var Entity = {

pos\_x: 0, pos\_y: 0,

size\_x: 0, size\_y: 0,

path\_image: "",

extend: function (extendProto) {

var object = Object.create(this);

for (var property in extendProto) {

if (this.hasOwnProperty(property) || typeof object[property] === 'undefined') {

object[property] = extendProto[property];

}

}

return object;

},

draw\_self(ctx) {

var local\_x = this.pos\_x;

var local\_y = this.pos\_y;

var main\_image = new Image();

main\_image.onload = function () {

ctx.drawImage(main\_image, local\_x \* 16 - 15, local\_y \* 16 - 12, 40, 40)

}

main\_image.src = this.path\_image;

},

kill() {

//console.log(this)

laterKill.push(this)

}

};

var active\_bonus = Entity.extend( {

pos\_x: 0, pos\_y: 0,

move\_x: 0, move\_y: 0,

speed: 1,

path\_image: './images/hero.png',

name\_direction: "bonus",

draw: function (ctx) {

spriteManager.drawSprite(ctx, this.name\_direction, this.pos\_x \* 16 - 15, this.pos\_y \* 16 - 15);

},

update: function () {

physicManager.update(this)

},

function(obj) {},

fire: function () {},

onTouchEntity: function (obj) {

if(obj.name === 'Player') {

active\_player.lifetime++;

alert('Вы подобрали бонус с жизнями, теперь у вас: '+active\_player.lifetime+' жизней')

eventsManager.action["up"] = false

eventsManager.action["left"] = false

eventsManager.action["right"] = false

eventsManager.action["down"] = false

eventsManager.action["fire"] = false

scores\_inner.innerHTML = 'Жизни: ' + active\_player.lifetime;

//console.log(this)

this.kill()

}

}

})

var active\_fireball = Entity.extend( {

pos\_x: 0, pos\_y: 0,

move\_x: 0, move\_y: 0,

speed: 1,

path\_image: './images/hero.png',

name\_direction: "fireball",

draw: function (ctx) {

spriteManager.drawSprite(ctx, this.name\_direction, this.pos\_x \* 16 - 15, this.pos\_y \* 16 - 15);

},

update: function () {

physicManager.update(this)

},

function(obj) {},

fire: function () {},

onTouchEntity: function (obj) {

if(obj.name === 'Enemy') {

//console.log(this)

this.kill()

}

}

})

var active\_enemy = Entity.extend({

pos\_x: 0, pos\_y: 0,

lifetime: 100,

move\_x: 0, move\_y: 0,

speed: 1,

path\_image: './images/hero.png',

name\_direction: "enemy\_right",

draw: function (ctx) {

spriteManager.drawSprite(ctx, this.name\_direction, this.pos\_x \* 16 - 15, this.pos\_y \* 16 - 15);

},

update: function () {

physicManager.update(this)

},

function(obj) {},

fire: function () {},

onTouchEntity: function (obj) {

if(obj.name === 'Fireball') {

//console.log(this)

this.kill()

}

}

});

var active\_player = Entity.extend({

pos\_x: 0, pos\_y: 0,

lifetime: 2,

move\_x: 0, move\_y: 0,

speed: 1,

path\_image: './images/hero.png',

name\_direction: "hero\_right",

draw: function (ctx) {

spriteManager.drawSprite(ctx, this.name\_direction, this.pos\_x \* 16 - 15, this.pos\_y \* 16 - 15);

},

update: function () {

physicManager.update(this)

},

function(obj) {},

fire: function () {

soundManager.play('../audio/click\_combo.wav', {looping: false, volume: 100})

var r = Object.create(active\_fireball);

r.name = "Fireball";

r.pos\_x = active\_player.pos\_x;

r.pos\_y = active\_player.pos\_y;

if(this.name\_direction === "hero\_right")

setInterval(()=> moveObject(r,1,0), 50)

if(this.name\_direction === "hero\_left")

setInterval(()=> moveObject(r,-1,0), 50)

if(this.name\_direction === "hero\_up")

setInterval(()=> moveObject(r,0,-1), 50)

if(this.name\_direction === "hero\_down")

setInterval(()=> moveObject(r,-1,1), 50)

Entities.push(r);

},

onTouchEntity: function (obj){

if((obj.name === 'Enemy')&&(this.lifetime===1)) {

//console.log('collized');

cur\_scores = active\_player.lifetime

window.location.href = '/'

alert('Вас убили!')

if(ended == 0) {

writeRec({name: player\_name, scores: cur\_scores});

ended = 1;

}

scores\_inner.innerHTML = 'Жизни: ' + active\_player.lifetime;

}

if((obj.name === 'Enemy')&&(this.lifetime!==1)) {

this.lifetime--

do {

this.pos\_x = getRandomInt(mapManager.xCount)

this.pos\_y = getRandomInt(mapManager.yCount-3)

} while(!mapManager.Cell\_class[this.pos\_x][this.pos\_y].isFree())

alert('Вас ранили, у вас осталось '+this.lifetime+' жизни. Вы перемещены в случайную точку карты')

eventsManager.action["up"] = false

eventsManager.action["left"] = false

eventsManager.action["right"] = false

eventsManager.action["down"] = false

eventsManager.action["fire"] = false

scores\_inner.innerHTML = 'Жизни: ' + active\_player.lifetime;

}

}

});

function update() { // oбhobлehиe иh$opmaции

if(this.active\_player === null)

return;

this.active\_player.move\_x = 0;

this.active\_player.move\_y = 0;

mapManager.draw(ctx);

if (eventsManager.action["up"]) {

active\_player.name\_direction = "hero\_up"

moveObjectNotDraw(this.active\_player, 0,-1)

}

if (eventsManager.action["down"]) {

active\_player.name\_direction = "hero\_down"

moveObjectNotDraw(this.active\_player, 0,1)

}

if (eventsManager.action["left"]) {

active\_player.name\_direction = "hero\_left"

moveObjectNotDraw(this.active\_player, -1,0)

}

if (eventsManager.action["right"]) {

active\_player.name\_direction = "hero\_right"

moveObjectNotDraw(this.active\_player, 1,0)

}

ObjectDraw(this.active\_player)

if (eventsManager.action["fire"]) this.active\_player.fire();

Entities.forEach(function (e){

e.update()

e.draw(ctx)

})

for(var i = 0; i < laterKill.length; i++) {

var idx = Entities.indexOf(laterKill[i]);

if(idx > -1)

Entities.splice(idx, 1);

// console.log(Entities)

}

if(laterKill.length > 0)

laterKill.length = 0;

//\*//\*

//mapManager.centerAt(this.active\_player.pos\_x, this.active\_player.pos\_y);

//draw(ctx);/\*\*/

}

function updateAi() {

var numbEntities = 0;

Entities.forEach(function (e) {

if((e.name==='Enemy')/\*&&(isGo === 1)\*/) {

numbEntities++;

if(active\_player.pos\_x<e.pos\_x)

{

e.name\_direction = "enemy\_left"

//mapManager.Cell\_class[e.pos\_x][e.pos\_y].SetFree();

moveObjectNotDraw(e, -1, 0)

//mapManager.Cell\_class[e.pos\_x][e.pos\_y].SetBlock();

}

if(active\_player.pos\_x>e.pos\_x)

{

e.name\_direction = "enemy\_right"

moveObjectNotDraw(e, 1, 0)

}

if(active\_player.pos\_y>e.pos\_y)

{

e.name\_direction = "enemy\_down"

moveObjectNotDraw(e, 0, 1)

}

if(active\_player.pos\_y<e.pos\_y)

{

e.name\_direction = "enemy\_up"

moveObjectNotDraw(e, 0, -1)

}

ObjectDraw(e)

}

})

if(numbEntities === 0) {

toWin();

}

}

function toWin() {

if(level === 0) {

window.location.href='/level2'

cur\_scores = active\_player.lifetime

alert('Вы победили всех монстров, вы победили. Переход на следующий уровень.')

}

if(level === 1) {

window.location.href='/'

cur\_scores = active\_player.lifetime

writeRec({name: player\_name, scores: cur\_scores});

alert('Вы победили всех монстров, игра окончена')

}

}

function play() {

setInterval(() => updateAi(), 450);

setInterval(() => update(), 100);

}

factory['Player'] = active\_player; // иhициaли3aция $aбpики

factory['Enemy'] = active\_enemy;

factory['Bonus'] = active\_bonus;

var canvas = document.getElementById("canvasId")

var ctx = canvas.getContext('2d')

mapManager.loadMap(level\_path);

mapManager.parseEntities();

mapManager.draw(ctx);

spriteManager.loadAtlas("./atlas/atlas.json", "./atlas/atlas.png");

setTimeout(() => active\_player.draw\_self(ctx), 300);

eventsManager.setup(canvas);

play()

let player\_name = localStorage["tetris.username"];

let cur\_scores = 0;

player\_name\_inner.innerHTML = 'Имя игрока: ' + player\_name;

scores\_inner.innerHTML = 'Жизни: ' + active\_player.lifetime;

let readed = readRec();

if(readed!=[]) {

for(i=0;i<readed.length;i++) {

if(readed[i].name === player\_name) {

active\_player.lifetime = readed[i].scores;

cur\_scores = readed[i].scores

}

}

}

Название файла: index.js

function store(){

let input = document.getElementById("id001");

let username = input.value;

if(username == ""){

window.alert("Input your username!");

}else{

window.location.href = "/level1";

localStorage["tetris.username"] = input.value;

}

}

function read(source){

if(localStorage["tetris.username"] != undefined){

source.value = localStorage["tetris.username"];

}

}

function reading() {

localStorage.setItem('records', JSON.stringify([]));

}

let a = JSON.parse(localStorage.getItem("records"));

console.log(a)

if((a==[])||(a==undefined)){

window.addEventListener('load', reading);

reading();

}

printRecords();

Название файла: mapManager.js

class CellClass {

reserved = false;

constructor() {

this.reserved = false;

}

isFree() {

if(this.reserved===false)

return true;

}

setBlock(){

this.reserved = true;

}

setFree(){

this.reserved = false;

}

}

var mapManager = {

mapData: null,

tLayer: null,

bLayer: null,

xCount: 0,

yCount: 0,

tSize: {x: 64, y: 64},

mapSize: {x: 64, y: 64},

tilesets: [],

imgLoadCount: 0,

imgLoaded: false,

jsonLoaded: false,

view: {x: 0, y: 0, w: 400, h: 400},

moving\_map: false,

Cell\_class: [[]],

loadMap(path) {

var request = new XMLHttpRequest();

request.onreadystatechange = function () {

if (request.readyState === 4 && request.status === 200) {

mapManager.parseMap(request.responseText)

}

};

request.open("GET", path, true);

request.send()

},

parseMap(tilesJSON) {

this.mapData = JSON.parse(tilesJSON);

this.xCount = this.mapData.width;

this.yCount = this.mapData.height;

this.tSize.x = this.mapData.tilewidth;

this.tSize.y = this.mapData.tileheight;

this.mapSize.x = this.xCount \* this.tSize.x;

this.mapSize.y = this.yCount \* this.tSize.y;

if (this.mapSize.x > this.view.w || this.mapSize.y > this.view.h) this.moving\_map = true;

for (let i = 0; i < this.mapData.tilesets.length; i++) {

var img = new Image();

img.onload = function () {

mapManager.imgLoadCount++;

if (mapManager.imgLoadCount === mapManager.mapData.tilesets.length) {

mapManager.imgLoaded = true;

}

};

img.src = this.mapData.tilesets[i].image;

var t = this.mapData.tilesets[i];

var ts = {

firstgid: t.firstgid,

image: img,

name: t.name,

xCount: Math.floor(t.imagewidth / mapManager.tSize.x),

yCount: Math.floor(t.imageheight / mapManager.tSize.y),

};

this.tilesets.push(ts)

}

this.jsonLoaded = true;

for (var i = 0; i < this.xCount; i++) {

this.Cell\_class.push([]);

}

for (var i = 0; i < this.xCount; i++) {

for (var j = this.Cell\_class[i].length; j < this.yCount; j++) {

this.Cell\_class[i].push(new CellClass);

}

}

this.Cell\_class.pop()

this.draw\_self();

},

draw(ctx) {

ctx.rect(this.view.x, this.view.y, this.view.w, this.view.h);

ctx.fillStyle = "rgba(0,0,0,0.63)";

ctx.fill();

if (!mapManager.imgLoaded || !mapManager.jsonLoaded) {

setTimeout(function () {

mapManager.draw(ctx);

}, 100)

} else {

if (this.tLayer === null || this.bLayer === null)

for (let id = 0; id < this.mapData.layers.length; id++) {

var layer = this.mapData.layers[id];

if (layer.type === "tilelayer") {

if (layer.name === "tile\_front") {

this.tLayer = layer;

continue;

}

if (layer.name === "tile\_back") {

this.bLayer = layer;

}

}

}

for (let i = 0; i < this.tLayer.data.length; i++) {

if (this.bLayer.data[i] !== 0) {

let tile = this.getTile(this.bLayer.data[i]);

let pX = (i % this.xCount) \* this.tSize.x;

let pY = Math.floor(i / this.xCount) \* this.tSize.y;

if (!this.isVisible(pX, pY, this.tSize.x, this.tSize.y))

continue;

pX -= this.view.x;

pY -= this.view.y;

ctx.drawImage(tile.img, tile.px, tile.py, this.tSize.x, this.tSize.y, pX, pY, this.tSize.x, this.tSize.y);

}

if (this.tLayer.data[i] !== 0) {

let tile = this.getTile(this.tLayer.data[i]);

let pX = (i % this.xCount) \* this.tSize.x;

let pY = Math.floor(i / this.xCount) \* this.tSize.y;

if (!this.isVisible(pX, pY, this.tSize.x, this.tSize.y))

continue;

pX -= this.view.x;

pY -= this.view.y;

ctx.drawImage(tile.img, tile.px, tile.py, this.tSize.x, this.tSize.y, pX, pY, this.tSize.x, this.tSize.y);

}

}

}

},

getTile(tileIndex) {

var tile = {

img: null,

px: 0,

py: 0

};

var tileset = this.getTileset(tileIndex);

tile.img = tileset.image;

var id = tileIndex - tileset.firstgid;

var x = id % tileset.xCount;

var y = Math.floor(id / tileset.xCount);

tile.px = x \* mapManager.tSize.x;

tile.py = y \* mapManager.tSize.y;

return tile;

},

getTileset(tileIndex) {

for (var i = mapManager.tilesets.length - 1; i >= 0; i--)

if (mapManager.tilesets[i].firstgid <= tileIndex) {

return mapManager.tilesets[i];

}

return null;

},

isVisible(x, y, width, height) {

if (x + width < this.view.x || y + height < this.view.y || x > this.view.x + this.view.w || y > this.view.y + this.view.h)

return false;

return true;

},

parseEntities() {

if (!mapManager.imgLoaded || !mapManager.jsonLoaded) {

setTimeout(function () {

mapManager.parseEntities();

}, 100)

} else

for (var j = 0; j < this.mapData.layers.length; j++)

if (this.mapData.layers[j].type === 'objectgroup') {

var entities = this.mapData.layers[j];

for (var i = 0; i < entities.objects.length; i++) {

var e = entities.objects[i];

try {

//console.log(factory)

var obj = Object.create(factory[e.type]);

obj.name = e.name;

//console.log(obj.name)

obj.pos\_x = Math.floor(e.x/16);

//console.log(obj.pos\_x)

obj.pos\_y = Math.floor(e.y/16);

//console.log(obj.pos\_y)

obj.size\_x = e.width;

obj.size\_y = e.height;

if (e.type === "Box" || e.type === "Finish") {

//if (e.type === "Box") gameManager.count\_target++;

obj.color = e.properties[0].value;

console.log(obj)

//console.log(e.properties[0].value)

}

Entities.push(obj);

console.log('entities: '+Entities)

if (obj.name === "Player")

initPlayer(obj);

} catch (ex) {

console.log("Ошибка создания: [" + e.gid + "]" + e.type + "," + ex)

}

}

}

},

centerAt(x, y) {

if(x < this.view.w / 2) this.view.x = 0;

else

if(x > this.mapSize.x - this.view.w / 2) this.view.x = this.mapSize.x - this.view.w;

else

this.view.x = x - (this.view.w / 2);

if(y < this.view.h / 2) this.view.y = 0;

else

if(y > this.mapSize.y - this.view.h / 2) this.view.y = this.mapSize.y - this.view.h;

else

this.view.y = y - (this.view.h / 2);

},

redraw\_player(ctx, x, y) {

let loc\_i = (y\*25+x)/16;

for(var j = -25; j<100; j+=25) {

var time\_i = loc\_i-25;

if(time\_i<0) time\_i = 0;

var n = 1

if(time\_i-n<0) n = 0;

for (var i = time\_i-n; i < loc\_i+j + 4; i++) {

if (this.bLayer.data[i] !== 0) {

let tile = this.getTile(this.bLayer.data[i]);

let pX = (i % this.xCount) \* this.tSize.x;

let pY = Math.floor(i / this.xCount) \* this.tSize.y;

pX -= this.view.x;

pY -= this.view.y;

ctx.drawImage(tile.img, tile.px, tile.py, this.tSize.x, this.tSize.y, pX, pY, this.tSize.x, this.tSize.y);

}

if (this.tLayer.data[i] !== 0) {

let tile = this.getTile(this.tLayer.data[i]);

let pX = (i % this.xCount) \* this.tSize.x;

let pY = Math.floor(i / this.xCount) \* this.tSize.y;

pX -= this.view.x;

pY -= this.view.y;

ctx.drawImage(tile.img, tile.px, tile.py, this.tSize.x, this.tSize.y, pX, pY, this.tSize.x, this.tSize.y);

}

}

}

},

getTilesetIdx(x, y){

var wY = y;

var wX = x;

var idx = Math.floor(wY / this.tSize.y) \* this.xCount + Math.floor (wX / this.tSize.x);

return this.tLayer.data[idx];

},

to\_get\_blocks() {

let returned\_blocks = [];

for (var i = 0; i < this.tLayer.data.length; i++) {

if (this.tLayer.data[i] !== 0) {

let local\_x = Math.floor(i/25)

let local\_y = i;

while(local\_y>25) {

local\_y-=25;

}

returned\_blocks.push({local\_x: local\_y, local\_y: local\_x})

}

}

return returned\_blocks;

},

draw\_self() {

var canvas = document.getElementById("canvasId")

var ctx = canvas.getContext('2d')

mapManager.draw(ctx);

},

draw\_self\_object\_clear(x,y) {

var canvas = document.getElementById("canvasId")

var ctx = canvas.getContext('2d')

mapManager.redraw\_player(ctx,x,y);

this.to\_generate\_blocks();

},

to\_generate\_blocks() {

var canvas = document.getElementById("canvasId")

var ctx = canvas.getContext('2d')

let get\_blocks = mapManager.to\_get\_blocks();

for(let i = 0; i<get\_blocks.length; i++) {

if((get\_blocks[i].local\_x<25)&&(get\_blocks[i].local\_y<25))

this.Cell\_class[get\_blocks[i].local\_x][get\_blocks[i].local\_y].setBlock();

}

}

}

Название файла: physicManager.js

function moveObject(object, value\_x, value\_y) {

if((object.pos\_y+value\_y<mapManager.yCount-4)&&(object.pos\_y+value\_y>=0)) {

if((object.pos\_x+value\_x<mapManager.xCount)&&(object.pos\_x+value\_x>=0)){

if(mapManager.Cell\_class[object.pos\_x+value\_x][object.pos\_y].isFree()){

if(mapManager.Cell\_class[object.pos\_x][object.pos\_y+value\_y].isFree()) {

object.pos\_y+=value\_y;

object.pos\_x+=value\_x;

object.draw(ctx);

mapManager.to\_generate\_blocks()

} else if(object.name === 'Fireball') object.kill();

} else if(object.name === 'Fireball') object.kill();

} else if(object.name === 'Fireball') object.kill();

} else if(object.name === 'Fireball') object.kill();

}

function moveObjectNotDraw(object, value\_x, value\_y) {

if((object.pos\_y+value\_y<mapManager.yCount-1)&&(object.pos\_y+value\_y>=0)) {

if((object.pos\_x+value\_x<mapManager.xCount)&&(object.pos\_x+value\_x>=0)){

if(mapManager.Cell\_class[object.pos\_x+value\_x][object.pos\_y].isFree()){

if(mapManager.Cell\_class[object.pos\_x][object.pos\_y+value\_y].isFree()) {

object.pos\_y+=value\_y;

object.pos\_x+=value\_x;

mapManager.to\_generate\_blocks()

} else {

if(object.name === 'Fireball') object.kill();

}

} else {

if(object.name === 'Fireball') object.kill();

}

} else {

if(object.name === 'Fireball') object.kill();

}

} else {

if(object.name === 'Fireball') object.kill();

if(object.name === 'Player') {

if(object.pos\_y+value\_y>=mapManager.yCount-1)

object.pos\_y = 0;

}

}

}

function ObjectDraw(object) {

object.draw(ctx);

}

let physicManager = {

update: (obj)=>{

let newX = obj.pos\_x + Math.floor(obj.move\_x \* obj.speed);

let newY = obj.pos\_y + Math.floor(obj.move\_y \* obj.speed);

let ts = mapManager.getTilesetIdx(newX + obj.size\_x/2, newY + obj.size\_y/2);

let e = physicManager.entityAtXY(obj, newX, newY);

if(e !== null && obj.onTouchEntity){

console.log(obj.name+' touching '+e.name +' x: '+ e.pos\_x +' y:'+ e.pos\_y);

obj.onTouchEntity(e);

}

if(ts !== 0 && obj.onTouchMap){

obj.onTouchMap(ts);

}

if(ts === 0 && e === null){

obj.pos\_x = newX;

obj.pos\_y = newY;

}else{

return "break";

}

return "move";

},

entityAtXY: (obj, x, y)=>{

for(let i = 0; i < Entities.length; i++){

let e = Entities[i];

if(e.name !== obj.name) {

if( (x + 1 < e.pos\_x || x + 1 > e.pos\_x + 1) ||

(y - 1 > e.pos\_y || y - 1 < e.pos\_y - 1) ){

continue

}

return e;

}

}

return null;

}

}

Название файла: scriptFirstLevel.js

let level = 0;

let level\_path = './tile\_map005.json'

Название файла: scriptSecondLevel.js

let level = 1;

let level\_path = './tile\_map007.json'

Название файла: soundsManager.js

var fire = document.createElement("AUDIO")

document.body.appendChild(fire);

fire.src = '../audio/click\_combo.wav'

var soundManager = {

clips: {},

context: null,

gainNode: null,

loaded: false,

mus000: null,

mus001: null,

mus002: null,

init: ()=>{

soundManager.mus000 = "../audio/mus000.mp3";

soundManager.mus001 = "../audio/mus001.mp3";

soundManager.mus002 = "../audio/mus002.mp3";

fire.src = '../audio/click\_combo.wav'

this.context = new AudioContext();

this.gainNode = this.context.createGain ?

this.context.createGain() : this.context.createGainNode()

this.gainNode.connect(this.context.destination);

},

load: (path, callback)=>{

if(soundManager.clips[path]){

callback(this.clips[path]);

return;

}

let clip = {path: path, buffer: null, loaded: false};

clip.play = (volume, loop)=>{

soundManager.play(this.path, {looping: loop ? loop : false, volume: volume ? volume : 1});

};

soundManager.clips[path] = clip;

let request = new XMLHttpRequest();

request.open('GET', path, true);

request.responseType = 'arraybuffer';

request.onload = ()=>{

console.log('entry?')

console.log(soundManager.context)

soundManager.context.decodeAudioData(request.response, (buffer)=>{

console.log('entry')

clip.buffer = buffer;

clip.loaded = true;

callback(clip);

});

};

request.send();

},

loadArray: (array)=>{

for(let i = 0; i < array.length; i++){

soundManager.load(array[i], ()=>{

if(array.length === Object.keys(soundManager.clips).length){

for(let sd in soundManager.clips){

if(!soundManager.clips[sd].loaded){

return;

}

}

soundManager.loaded = true;

}

});

}

},

play: (path, settings)=>{

console.log(path)

fire.currentTime = 0;

fire.play()

if(!soundManager.loaded){

setTimeout(()=>{

//soundManager.play(path, settings);

}, 1000);

return;

}

let looping = false;

let volume = 1;

if(settings){

if(settings.looping){

looping = settings.looping;

}

if(settings.volume){

volume= settings.volume;

}

}

let sd = this.clips[path];

if(sd === null){

return false;

}

let sound = soundManager.context.createBufferSource();

sound.buffer = sd.buffer;

sound.connect(soundManager.gainNode);

sound.loop = looping;

soundManager.gainNode.gain.value = volume;

sound.start(0);

return true;

},

playSound: function (sound) {

sound.currentTime = 0;

sound.play()

},

};

soundManager.init();

Название файла: spriteManager.js

var spriteManager = {

image: new Image(),

sprites: new Array(),

imgLoaded: false,

jsonLoaded: false,

loadImg(imgName) { // 3aгpy3кa и3oбpaжehия

this.image.onload = function () {

spriteManager.imgLoaded = true;

};

this.image.src = imgName;

},

loadAtlas(atlasJson, atlasImg) {

var request = new XMLHttpRequest();

request.onreadystatechange = function () {

if (request.readyState === 4 && request.status === 200) {

spriteManager.parseAtlas(request.responseText);

}

};

request.open("GET", atlasJson, true);

request.send();

this.loadImg(atlasImg);

},

parseAtlas(atlasJSON) {

var atlas = JSON.parse(atlasJSON);

for (var name in atlas.frames) {

var frame = atlas.frames[name].frame;

this.sprites.push({name: name, x: frame.x, y: frame.y, w: frame.w, h: frame.h});

}

this.jsonLoaded = true;

},

getSprite(name) {

for (var i = 0; i < this.sprites.length; i++) {

var s = this.sprites[i];

if (s.name === name)

return s;

}

return null;

},

drawSprite(ctx, name, x, y) {

if (!this.imgLoaded || !this.jsonLoaded) {

setTimeout(function () {

spriteManager.drawSprite(ctx, name, x, y);

}, 100);

} else {

var sprite = this.getSprite(name);

if (!mapManager.isVisible(x, y, sprite.w, sprite.h))

return;

x -= mapManager.view.x;

y -= mapManager.view.y;

ctx.drawImage(this.image, sprite.x, sprite.y, sprite.w, sprite.h, x, y, 40, 40);

}

}

}

Название файла: tableRecords.js

//fs = require("fs");

function readRec() {

let a = JSON.parse(localStorage.getItem("records"));

return a;

}

function writeRec(user\_test) {

let a = JSON.parse(localStorage.getItem("records"));

a.push(user\_test);

a.sort((x,y) => {

return y.scores - x.scores;

});

localStorage.setItem('records', JSON.stringify(a));

}

function deleteRec(name) {

let a = JSON.parse(localStorage.getItem("records"));

for(var i = 0; i<a.length; i++) {

if(a[i].name === name) {

a.splice(i,1)

break

}

}

}

function printRecords() {

var canvas1 = document.getElementById("table\_records");

var ctx = canvas1.getContext("2d");

ctx.font = "16px cambria";

ctx.clearRect(0,0, 500,500);

ctx.fillText(" Table records:", 5, 10);

let a = readRec();

for(let i = 0; i < a.length; i++) {

ctx.fillText(a[i].name + " " + a[i].scores, 5, 30\*(i+1));

}

}