**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Государственное бюджетное образовательное учреждение высшего профессионального образования

**УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Факультет информационных систем и технологий

Кафедра «Измерительно-вычислительные комплексы»

**Пояснительная записка**

**к массовой многопользовательской онлайн-игре “Танчики”**

Выполнил:

ст. гр. ИСТбд-32 Пестряков А.С. / Шигабутдинова Р.Ш.

(Фамилия И.О.)

Проверил:

преподаватель Кандаулов В.М.

(Фамилия И.О.)

Ульяновск, 2017

**Содержание**

[Введение](#_bhpne1l18u9w) 3

[Техническое задание](#_1qw58za9ee44) 4

[1.1. Общее описание проекта](#_a2x23kn0abq5) 4

[1.2. Описание серверной части и ее функционала](#_n23q3a72et5g) 4

[1.3. Описание клиентской части и ее функционала](#_lo52cc9ddiqz) 5

[Прикладное программное обеспечение системы](#_nvzauaszk90l) 5

[2.1. Общая характеристика прикладного программного обеспечения](#_xjjo4nx7qybu) 5

[2.2. Состав прикладного программного обеспечения](#_ne5n9qjjrnnm) 6

[2.3. Описание основных методов приложения](#_wnd07ngoye3i) 6

[Руководство пользователя](#_j4k2joa7dna7) 7

[Заключение](#_g1er4j1sqbmk) 10

[Список источников](#_uyrbjwhe54ny) 11

[Приложения](#_gsk4zig5syd6) 12

# 

# 

# **Введение**

Танк — бронированная боевая машина, чаще всего на гусеничном ходу, как правило, с пушечным вооружением в полноповоротной башне в качестве основного. Своим появлением танки обязаны Первой мировой войне.

Танки всегда вызывали интерес у создателей всевозможных action-игр, как и любая военная техника.

Первая игра по танкам была создана в 1985 году. Это была японская игра «Battle City» (с англ. — «Город битв») — компьютерная игра для игровых приставок Famicom и Game Boy. В России и странах СНГ выпускалась на пиратских картриджах как в оригинальном виде, так и в модификации Tank 1990, и известна под неофициальным названием «Танчики».

Сейчас существует множество вариаций данной игры, но в целом процесс игры всегда остается практически одинаковым. Чаще всего они предлагают пользователю выбор среди различных видов танков, позволяет играть в компании друзей.

Проект является очень актуальным в данный момент времени, сейчас интернет предлагает огромное разнообразие всевозможных игр и пользователям будет особенно интересно поиграть в танчики, так как любая вариация этой игры вызывала неизменно высокий интерес у игроков. Также данный проект является неплохой возможностью продемонстрировать работу с JavaScript.

# **Техническое задание**

## **1.1. Общее описание проекта**

Танчики – игра между множеством противников на квадратном поле 5500 x 5500 (не включая границы).

Игрок может управлять одним из танков на поле. При этом он противостоит всем остальным игрокам, управляющим танками-противниками. Основная задача игрока – уничтожить все танки противников и набрать максимальное количество очков.

У каждого игрока есть ограниченное количество здоровья (брони танка). Стреляя в своих противников, игрок сокращает количество их здоровья, как только здоровье полностью кончается, танк игрока считается уничтоженным.

В случае, если танк игрока будет уничтожен, он может начать игру заново, при этом, его количество очков будет прежним.

Игра ведет статистику по количеству уничтоженных игроком танков и по количеству полной потери здоровья.

В самом начале игры, игроки могут выбирать себе любой танк из трех возможных вариантов:

* Легкий танк (самый быстрый танк с самой слабой прочностью, скорострельность);
* Средний танк (универсальный танк, средняя скорость, прочность ниже, чем у мощного танка, но выше, чем у легкого танка);
* Тяжелый танк (самый медленный танк с самой мощной прочностью, большой урон).

## **1.2. Описание серверной части и ее функционала**

Сервернаячасть осуществляет все действия пользователей и синхронизацию с клиентом. Сервер является авторитарным и принимает от клиента только его действия (нажатие клавиш и положение курсора). Синхронизация игрового мира с клиентом осуществляется с заданной частотой (21 раз в секунду), а сам мир просчитывается от действия пользователя с частотой 60 раз в секунду. Статистика и игровые события 2 раза в секунду. Данные частоты были выбраны для оптимальной скорости работы сервера.

## **1.3. Описание клиентской части и ее функционала**

Клиентская часть осуществляет прием и отправку данных о действия пользователя. Так как частота обмена не высока, то применяется аппроксимация данных игрока для обеспечения плавности игрового процесса. Соединение с сервером осуществляется в отдельном процессе, что снижает нагрузку на основной поток приложения.

# **Прикладное программное обеспечение системы**

## **2.1. Общая характеристика прикладного программного обеспечения**

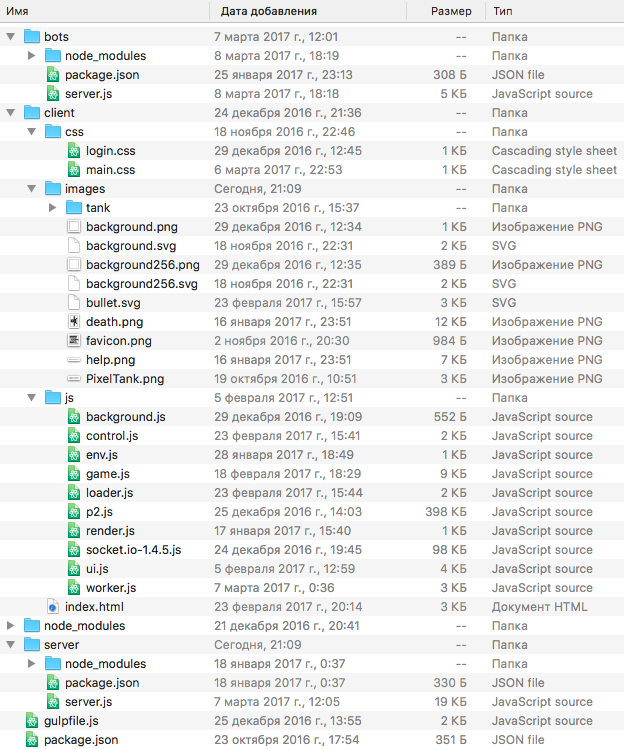
Данное ПО представляет собой серверную и клиентскую часть. Где каждая часть имеет различное представление: серверная часть в виде консольного приложения; клиентская часть имеет вид web-страницы. Такая организация уменьшает количество затрат пользователя для возможности использования данного ПО.

Для осуществления связи клиента и сервера используется WebSocket — протокол полнодуплексной связи (может передавать и принимать одновременно) поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени.

В настоящее время в W3C осуществляется стандартизация API WebSockets.

WebSocket разработан для воплощения в web-браузерах и web-серверах, но он может быть использован для любого клиентского или серверного приложения. Протокол WebSocket — это независимый протокол, основанный на протоколе TCP. Он делает возможным более тесное взаимодействие между браузером и web сайтом, способствуя распространению интерактивного содержимого и созданию приложений реального времени.

## **2.2. Состав прикладного программного обеспечения**



## **2.3. Описание основных методов приложения**

Для данного проекта использовался язык программирования JavaScript. Клиентская часть исполняется в браузере пользователя, а серверная часть с использование Node.JS и набор различных библиотек: socket.io - реализует клиент-серверное соединение, p2 - физический движок для реализации столкновени игровых объектов, uws - реализация WebSocket на C++. Сборка и запуск проекта осуществляется в Gulp и набор модулей. В браузере используется WebSocket, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени, WebWorkers предоставляют простое средство для запуска скриптов в фоновом потоке. Поток Worker'а может выполнять задачи без вмешательства в пользовательский интерфейс и используется для соединения с сервером. Canvas, элемент HTML5, предназначенный для создания растрового двухмерного изображения при помощи скриптов, обычно на языке JavaScript.

# **Руководство пользователя**

Заходя в игру пользователь видит стартовое меню, представленное на рис. 1.

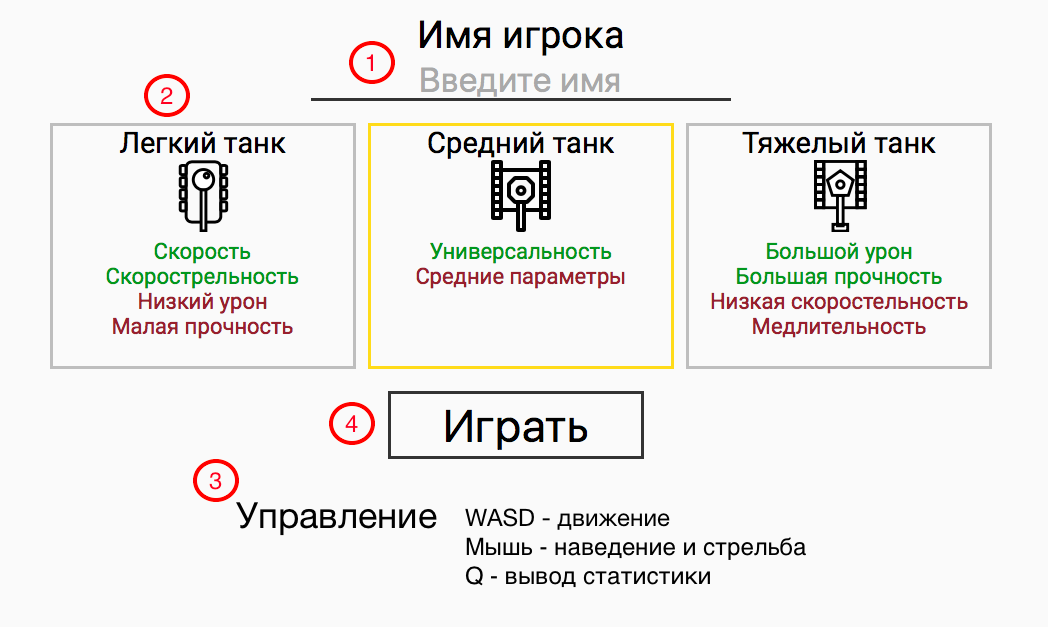


Рис. 1. Стартовое меню

В стартовом меню пользователь может увидеть четыре поля, с которыми необходимо ознакомится в самом начале игры. Пользователь может оставить данные пункты без внимания, тогда настройки игры будут установлены по умолчанию.

Все поля стартового меню:

1. В данном поле игроку необходимо ввести имя, под которым он собирается играть, в случае, если это поле оставить незаполненным игрок будет играть с именем по умолчанию (NONAME);
2. Выбор танка, в случае, если не осуществить выбор танка, игра начнется со средним танком;
3. Управление танком. В данном пункте описаны основы управления игры. С данным полем необходимо ознакомиться, чтобы понять, как осуществляется управление в игре;
4. Начало игры. После нажатия на данную кнопку, пользователь перейдет непосредственно к процессу игры.

Начав играть пользователь видит перед собой игровое поле (Рис. 2).

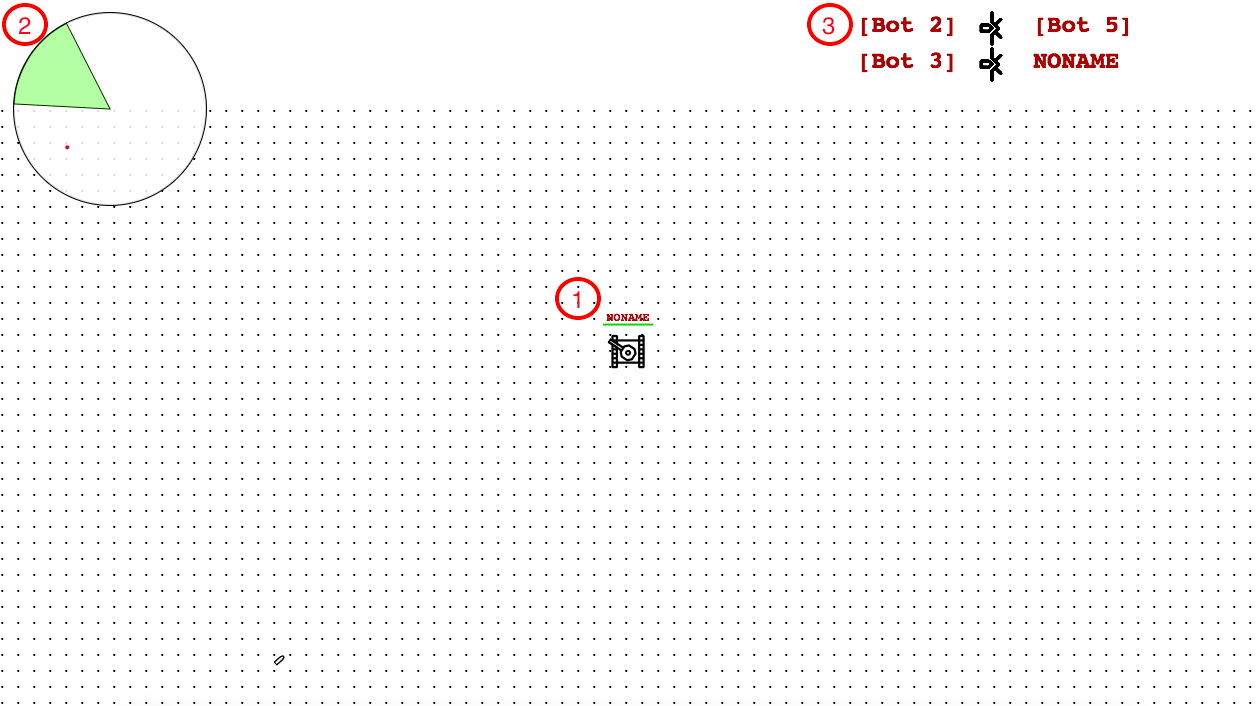


Рис. 2. Игровое поле

1. Над танком указано имя пользователя и количество здоровья, как только здоровье кончится, игрок возрождается в случайном месте на случайном танке;
2. Радар, на котором отображаются другие пользователи.
3. История пяти последних событий c информацией об уничтоженных танках.

Нажав клавишу Q для отображения статистики пользователь увидит 10 лучших игроков (Рис. 3).

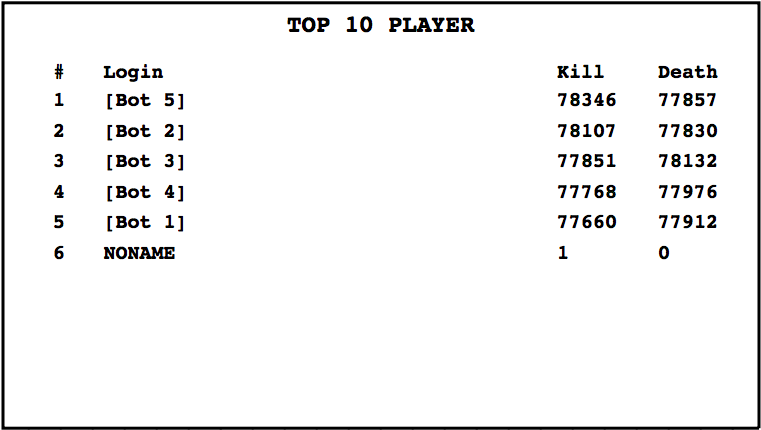


Рис. 3. Статистика игры

# 

# **Заключение**

Проектирование и реализация курсовой работы были выполнены в соответствии с утвержденным техническим заданием.

Особенные сложности были вызваны незнанием некоторых особенностей используемого языка программирования, однако данные сложности были преодолены в процессе работы над проектом.

Логика игры максимально приближена к правилам оригинальной игры.

К достоинствам данной игры можно отнести:

* простота в использовании. Игроку совершенно не нужно устанавливать какие-либо программы на компьютер, чтобы насладится игрой;
* простой и удобный интуитивно понятный интерфейс.

Недостатки:

* TCP не может создавать широковещательные пакеты для обмена данными между клиентами и сервером, что сильно затрудняет работу с сетью.

# **Список источников**

1. p2 // NPM URL:<https://www.npmjs.com/package/p2>
2. gulp // NPM URL:<https://www.npmjs.com/package/gulp>
3. uws // NPM URL:<https://www.npmjs.com/package/uws>
4. socket.io // NPM URL:<https://www.npmjs.com/package/socket.io>
5. socket.io-client // NPM URL:<https://www.npmjs.com/package/socket.io-client>
6. p2 docs // GitHub URL:<http://schteppe.github.io/p2.js/docs/>
7. Optimizing WebSockets Bandwidth // Build New Games URL:[http://buildnewgames.com/optimizing-websockets-bandwidth](http://buildnewgames.com/optimizing-websockets-bandwidth/)
8. Node.js v7.7.2 Documentation // Node.js URL:<https://nodejs.org/dist/latest-v7.x/docs/api/>

# **Приложения**

Серверная часть:

'use strict';

var io = require('socket.io')(3600);

io.set('transports', ['websocket']);

io.engine.ws = new (require('uws').Server)({

noServer: true,

clientTracking: false,

perMessageDeflate: false

});

var p2 = require('p2');

var bullets = []; // Снаряды

var users = []; // Пользователи

var cd = []; // Перезарядка орудия

var stats = []; // Статистика

var control = [];

var death = [];

var fps = 60; // Частота обновлений в секунду

var fps\_send = 21; // Частота отправки обновлений в секунду

var BULLET\_SPEED = 750;

var tank = {

light: {

hp: 180, // текущее хп

max\_speed: 230, // Макс. скорость

acceleration: 145, // Скорость разгона (px/sec)

reload: 0.35, // Период между выстрелами (в секундах)

damage: 55, // Урон наносимый снарядом

chassis: {

angle: 96 // Скорость поворота шасси (в градусах)

},

turret: {

angle: 180 // Скорость башни шасси (в градусах)

}

},

medium: {

hp: 275,

max\_speed: 205,

acceleration: 110,

reload: 0.68,

damage: 85,

chassis: {

angle: 68

},

turret: {

angle: 150

}

},

heavy: {

hp: 420,

max\_speed: 150,

acceleration: 55,

reload: 1.15,

damage: 125,

chassis: {

angle: 48

},

turret: {

angle: 120

}

}

};

console.log('Starting server');

//ssh root@194.87.144.13 cnZR4Spa

//node -nouse-idle-notification -expose-gc -max-old-space-size=4096 server.js

// DEBUG=socket.io\* node server.js

// /\* WORLD

var world = new p2.World({

gravity:[0, 0]

});

var PhysicsObject=[];

// Добавление танка в мир

function addTank(id,tank,x,y,angle) {

let mass;

switch (tank) {

case 'light':

mass=15.5;

break;

case 'medium':

mass=36;

break;

case 'heavy':

mass=188.9;

break;

default:

mass=15.5;

}

let body = new p2.Body({ mass: mass, position: [x,y],angle: angle, damping: .99,angularDamping: .95 });

PhysicsObject[id]=body.id;

switch (tank) {

case 'light':

//light

body.fromPolygon([[22,26],[22,19.6],[34,19.6], [34,-19.6],[22,-19.6],[22,-26],

[-30,-26],[-30,-19.6],[-34,-19.6], [-34,19.6],[-30,19.6],[-30,26] ]);

break;

case 'medium':

//medium

body.fromPolygon([[32.5,32.5],[32.5,20],[25,20], [25,-20],[32.5,-20],[32.5,-32.5],

[-32.5,-32.5],[-32.5,-20],[-25,-20], [-25,20],[-32.5,20],[-32.5,32.5] ]);

break;

case 'heavy':

//heavy

body.addShape(new p2.Box({ width: 54, height: 57.6 }));

break;

default:

//light

body.fromPolygon([[22,26],[22,19.6],[34,19.6], [34,-19.6],[22,-19.6],[22,-26],

[-30,-26],[-30,-19.6],[-34,-19.6], [-34,19.6],[-30,19.6],[-30,26] ]);

}

world.addBody(body);

}

// Добавление границ в мир

function addBorder(x,y,angle) {

let body = new p2.Body({position:[x,y],angle: angle});

body.addShape(new p2.Plane());

world.addBody(body);

}

// Изменение танка в мире по ид пользователя

function changeTankSpeed(id,speed=0) {

let changeTankId = world.getBodyById(PhysicsObject[id]);

changeTankId.velocity[0] = speed\*Math.cos(changeTankId.angle);

changeTankId.velocity[1] = speed\*Math.sin(changeTankId.angle);

}

function changeTankAngle(id,angle=0) {

world.getBodyById(PhysicsObject[id]).angularVelocity=angle;

}

// Получение информации о танке из мира по ид пользователя

function getTank(id) {

return world.getBodyById(PhysicsObject[id]);

}

// Удаление танка из мира по ид пользователя

function removeTank(id) {

world.removeBody(world.getBodyById(PhysicsObject[id]));

PhysicsObject.splice(id, 1);

}

addBorder( 0, 2530,Math.PI); // bottom

addBorder(-2560, 0,-Math.PI/2); // left

addBorder( 2530, 0,Math.PI/2); // right

addBorder( 0,-2560,0); // top

// \*/ WORLD

// /\* STATS

// Изменение статистики игрока по ид пользователя

function kdStat(id\_kill, id\_death) {

for(var i = 0, l = stats.length; i < l; i++) {

if (stats[i].id == id\_kill)

stats[i].kill++;

if (stats[i].id == id\_death)

stats[i].death++;

}

}

// Добавление статистики игрока по ид пользователя

function addStat(id) {

stats.unshift({

id: id,

kill: 0,

death: 0

});

}

// Удаление статистики игрока по ид пользователя

function removeStat(id) {

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

if (id == stats[i].id)

stats.splice(i, 1);

}

}

// \*/ STATS

setInterval(function Refresh() {

// Полет снарядов

for(var bi = 0; bi < bullets.length; bi++) {

// Передвижение снаряда на величину скорости

bullets[bi].x +=s(BULLET\_SPEED)\*Math.cos(bullets[bi].angle);

bullets[bi].y +=s(BULLET\_SPEED)\*Math.sin(bullets[bi].angle);

bullets[bi].life -=s(1);

if(bullets[bi].life < 0) bullets.splice(bi, 1);

}

// Просчет столкновения со снарядом

for(var bi = 0; bi < bullets.length; bi++) {

for(var ui = 0, ul = users.length; ui < ul; ui++) {

if (users[ui].id != bullets[bi].user)

if(Math.pow(Math.abs(bullets[bi].x-users[ui].x),2)+Math.pow(Math.abs(bullets[bi].y-users[ui].y),2) <= 56\*56) {

if(CollisionBullet(users[ui].x,users[ui].y,users[ui].chassis.angle,bullets[bi].x,bullets[bi].y)) {

//console.log(bullets[bi].user + ' повредил ' + users[ui].id + ' на ' + bullets[bi].damage + ' ('+ users[ui].hp + '/150)');

if (users[ui].hp-bullets[bi].damage >= 0 && users[ui].hp > 0) {

users[ui].hp-=bullets[bi].damage;

} else {

//console.log(bullets[bi].user + ' уничтожил ' + users[ui].id);

// Отправка данных о смерти

kdStat(bullets[bi].user, users[ui].id);

//io.sockets.emit('death', {log: {killer: bullets[bi].user,death: users[ui].id,life : 3}, stats});

death[death.length]=[bullets[bi].user,users[ui].id];

let plname = users[ui].name;

let usersid = users[ui].id;

removeTank(users[ui].id);

users.splice(ui, 1);

// Возрождение, но нужно выделить в отдельный месседж

// в виде replay, а саму смерть оптравлять в виде death

////////////////

var pltank=(Math.random() > .66) ? 'heavy' : (Math.random() > .5) ? 'light' : 'medium';

let x = getRandomInt(-2400, 2400);

let y = getRandomInt(-2400, 2400);

users.unshift({

x : x,

y : y,

speed : 0,

turret : {

angle: -Math.PI/2

},

chassis : {

angle: -Math.PI/2

},

hp : tank[pltank].hp,

tank : pltank,

name : plname,// заготовка для ника

id : usersid

});

addTank(usersid,pltank,x,y,-Math.PI/2);

///////////////////

}

bullets.splice(bi, 1);

break;

}

}

}

}

for(var ui = 0, ul = users.length; ui < ul; ui++) {

// Обновление данных о направление орудия

users[ui].turret.angle = RotateTurret(users[ui].turret.angle,control[users[ui].id].a,s(tank[users[ui].tank].turret.angle\*Math.PI/180));

// Вычисление изменений скорости

if(control[users[ui].id].m == 1) {

users[ui].speed += s(tank[users[ui].tank].acceleration\*ac(users[ui].speed,tank[users[ui].tank].max\_speed));

if (users[ui].speed > tank[users[ui].tank].max\_speed) users[ui].speed = tank[users[ui].tank].max\_speed;

}

if(control[users[ui].id].m == -1) {

users[ui].speed -= s(tank[users[ui].tank].acceleration\*0.6\*ac(users[ui].speed,tank[users[ui].tank].max\_speed));

if (users[ui].speed < -tank[users[ui].tank].max\_speed\*0.3) users[ui].speed = -tank[users[ui].tank].max\_speed\*0.3;

}

if(control[users[ui].id].m == 0) {

if (users[ui].speed < 10 && users[ui].speed > -10) {

users[ui].speed = 0;

}

if (users[ui].speed <= -10) {

users[ui].speed += s(tank[users[ui].tank].acceleration\*3\*ac(users[ui].speed,users[ui].max\_speed));

}

if (users[ui].speed >= 10) {

users[ui].speed -= s(tank[users[ui].tank].acceleration\*3\*ac(users[ui].speed,users[ui].max\_speed));

}

}

// Поворот танка

let angleTank = 0;

if(control[users[ui].id].r == -1) angleTank -=tank[users[ui].tank].chassis.angle\*Math.PI/180\*ac(users[ui].speed,tank[users[ui].tank].max\_speed);

if(control[users[ui].id].r == 1) angleTank +=tank[users[ui].tank].chassis.angle\*Math.PI/180\*ac(users[ui].speed,tank[users[ui].tank].max\_speed);

changeTankAngle(users[ui].id, angleTank);

// Передвижение танка на величину скорости

changeTankSpeed(users[ui].id,users[ui].speed);

//Уменьшение перезарядки

cd[users[ui].id] -=s(1);

if(cd[users[ui].id] <= 0) cd.splice(users[ui].id, 1);

// Стрельба

if(control[users[ui].id].f) {

// Проверка возможности стрельбы

if(!(cd[users[ui].id]>0)) {

// Включение блокировки на стрельбу (перезарядка)

cd[users[ui].id]=tank[users[ui].tank].reload;

// Добавление снаряда

bullets.unshift({

x: users[ui].x+36\*Math.cos(users[ui].turret.angle),

y: users[ui].y+36\*Math.sin(users[ui].turret.angle),

life: 1.5,

angle: users[ui].turret.angle,

damage: tank[users[ui].tank].damage,

user: users[ui].id

});

}

}

}

world.step(1/fps);

// Считывание данных с физ. мира

for(var ui = 0, ul = users.length; ui < ul; ui++) {

let tmp = getTank(users[ui].id);

users[ui].x = tmp.position[0];

users[ui].y = tmp.position[1];

users[ui].chassis.angle = tmp.angle;

}

}, 1000/fps);//1000/fps

setInterval(function SendStats() {

// Отправка данных игроков //sockets

io.compress(true).emit('stats', {data: {stats,death}});

death=[];

}, 750);

setInterval(function SendRefresh() {

// Отправка данных игроков //sockets

let t = Date.now();

io.compress(true).emit('refresh', {data: SendArray(), t});

}, 1000/fps\_send);

/\*

++count;

var count=0;

setInterval(function () {

console.log(count);

count=0;

},1000)

\*/

io.on('connection', function (socket) {

// Обработка полученых от пользователя данных

socket.on('refresh', function (m) {

control[socket.id]=m;

});

// Вход нового игрока и создание данных о нем

socket.on('join', function (m) {

// Проверка

switch(m.tank) {

case 'heavy':

break;

case 'medium':

break;

case 'light':

break;

default:

m.tank=(Math.random() > .66) ? 'heavy' : (Math.random() > .5) ? 'light' : 'medium';

break;

}

if (m.name == '') {

m.name="NONAME";

}

// Вывод в лог информации о новом игроке

console.log('Join new client: ' + socket.id);

// Отправка данных соединения

socket.compress(true).emit('init', {tank,stats,bs:BULLET\_SPEED});

// Добавление данных нового пользователя

let x = getRandomInt(-2000, 2000);

let y = getRandomInt(-2000, 2000);

users.unshift({

x: x,

y: y,

speed: 0,

turret: {

angle: -Math.PI/2

},

chassis: {

angle: -Math.PI/2

},

hp: tank[m.tank].hp,

tank: m.tank,

name: m.name,// заготовка для ника

id: socket.id

});

addStat(socket.id);

addTank(socket.id,m.tank,x,y,-Math.PI/2);

control[socket.id]={

m: 0,

r: 0,

a: 0,

f: false

};

});

// Обработка дисконекта

socket.on('disconnect', function (m) {

//io.emit('user disconnected');

// Удаление данных удаленного игрока

io.clients(function(error, clients) {

if (error) throw error;

users.forEach(function (value, index) {

let del=true;

clients.forEach(function (val) {

if (val == value.id) {

del=false;

}

});

if(del) {

removeTank(value.id);

removeStat(value.id);

users.splice(index, 1);

control.splice(value.id, 1);

}

});

});

// Вывод в лог информации о дисконекте

console.log("Disconnect: " + m);

});

});

// Оптимизация отправляемых данных

function SendArray() {

let array=[[],[]];

for(var ui = 0, ul = users.length; ui < ul; ui++) {

array[0][ui]=[

((0.5 + users[ui].x\*100) | 0)/100,

((0.5 + users[ui].y\*100) | 0)/100,

((0.5 + users[ui].speed\*100) | 0)/100,

((0.5 + users[ui].turret.angle\*100000) | 0)/100000,

((0.5 + users[ui].chassis.angle\*100000) | 0)/100000,

users[ui].hp,

users[ui].tank,

users[ui].name,

users[ui].id

];

}

for(var bi = 0, bl = bullets.length; bi < bl; bi++) {

array[1][bi]=[

((0.5 + bullets[bi].x\*100) | 0)/100,

((0.5 + bullets[bi].y\*100) | 0)/100,

((0.5 + bullets[bi].angle\*100000) | 0)/100000

];

}

return array;

}

// Функция плавного поворота на угол angle со скоростью speed

function RotateTurret(ta,ma,s) {

if (ta < 0)

ta = ta + 2\*Math.PI;

let a = ta - ma;

if (a > Math.PI)

a = a - 2\*Math.PI;

else if (a < -Math.PI)

a = a + 2\*Math.PI;

if (a >= -s && a <= s)

ta = ma;

else if (a > s)

ta = ta - s;

else if (a < -s)

ta = ta + s;

return ta;

}

//self.ang - текущий угол поворота

function CollisionTank(x,y,a,bx,by,ba) {

// Можно один из танков повернуть на угол обоих танков

// и воспринимать как один из них как квадрат

// другой как набор точек

return DetectCollision(

[[x-40\*Math.cos(a),y-40\*Math.sin(a)],

[x-40\*Math.cos(a),y+40\*Math.sin(a)],

[x+40\*Math.cos(a),y+40\*Math.sin(a)],

[x+40\*Math.cos(a),y-40\*Math.sin(a)] ]

,

[[bx-40\*Math.cos(ba),by-40\*Math.sin(ba)],

[bx-40\*Math.cos(ba),by+40\*Math.sin(ba)],

[bx+40\*Math.cos(ba),by+40\*Math.sin(ba)],

[bx+40\*Math.cos(ba),by-40\*Math.sin(ba)] ]

);

}

function CollisionBullet(x,y,a,bx,by) {

/\*

x,y - координаты танка

a - поворот танка

bx,by - координаты снаряда

x1,y1 --- x2,y2

| |

| x,y |

| |

x3,y3 --- x4,y4

\*/

var R = Math.sqrt(Math.pow(bx-x,2)+Math.pow(by-y,2));

bx = Math.cos(a) \* R + x;

by = Math.sin(a) \* R + y;

var x1=x-40;

var x4=x+40;

var y1=y-40;

var y4=y+40;

if(bx > x1 && by > y1

&& bx < x4 && by < y4)

return true;

return false;

}

function DetectCollision(one, another) {

/\*

Определяется так. Предположим, у нас есть 3 точки: А(х1,у1), Б(х2,у2), С(х3,у3). Через точки А и Б проведена прямая. И нам надо определить, как расположена точка С относительно прямой АБ. Для этого вычисляем значение:

D = (х3 - х1) \* (у2 - у1) - (у3 - у1) \* (х2 - х1)

- Если D = 0 - значит, точка С лежит на прямой АБ.

- Если D < 0 - значит, точка С лежит слева от прямой.

- Если D > 0 - значит, точка С лежит справа от прямой.

проверяеться по часовой стрелке

\*/

// Состояние столкновения

var collision=false;

one.forEach(function (value, index) {

// Если уже столкнулись дальше проверять нет смысла

// Можно конечно проверять кол-во точек столкновения, но зачем мне это надо хз

if (!collision) {

// Проверяемая точка

var C={

x:value[0],

y:value[1]

};

// Прямая через 2 точки

let A={

x:0,

y:0

};

let B={

x:0,

y:0

};

// Кол-во точек сравнения

let length = another.length;

let intcolls=0;

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

if (i==length-1) {

A.x=another[i][0];

A.y=another[i][1];

B.x=another[0][0];

B.y=another[0][1];

//console.log(i+1+') x:'+another[i]+' y:'+another[0]);

} else {

A.x=another[i][0];

A.y=another[i][1];

B.x=another[i+1][0];

B.y=another[i+1][1];

//console.log(i+1+') x:'+another[i]+' y:'+another[i+1]);

}

var D = (C.x - A.x) \* (B.y - A.y) - (C.y - A.y) \* (B.x - A.x);

//console.log(D);

if (D==0) {

//console.log('Точка на прямой');

intcolls++;

} else if (D>0) {

//console.log('Точка справа');

intcolls++;

} else if (D<0) {

//console.log('Точка слева');

}

}

if (intcolls == length) {

//console.log('Столкновение');

collision=true;

}

}

});

return collision;

}

function getRandomInt(min, max) {

return Math.floor(Math.random() \* (max - min + 1)) + min;

}

//Все ниже также входит в модуль апроксимации

function s(v) {

// Все действия происходят в виде Pixel/Second и тд

return v/fps;

}

function ac(speed, max\_speed=120) {

//(2+x^0.1)/(2+x^0.3) (x from 0 to 100)

// получаем коэффицент скорости разворота

// перемножаем на скорость поворота танка

return (1+Math.pow((Math.abs(speed)/max\_speed)\*10,0.1))/(1+Math.pow((Math.abs(speed)/max\_speed)\*10,0.3));

}

/\*

for(var bi = 0, bl = bullets.length; bi < bl; bi++) {

comp\_value=bullets[bi]

comp\_index=bi

for(var ui = 0, ul = users.length; ui < ul; ui++) {

value=users[ui]

index=ui

// Просчет столкновения со снарядом

bullets.forEach(function (comp\_value, comp\_index) {

users.forEach(function (value, index) {

\*/

Клиентская часть:

index.html

<!doctype html>

<html>

<meta charset="utf-8">

<head>

<title>Pixel Tank</title>

<meta charset="utf-8">

<link rel="shortcut icon" href="images/favicon.png">

<script src="js/loader.js"></script>

</head>

<body style="height: 100%; margin: auto;">

<form id="select" class="select select-active">

<br />

Имя игрока<br />

<input class="select-name" type="text" name="select-name" placeholder="Введите имя">

<container>

<input type="radio" name="select-tank" value="light" id="light"/>

<label for="light" class="select-tank select-tank-light">

Легкий танк<br />

<img src="images/tank/light/all.svg" alt="" />

<span class="select-tank--plus">

Скорость<br />

Скорострельность

</span>

<span class="select-tank--minus">

Низкий урон<br />

Малая прочность

</span>

</label>

<input type="radio" name="select-tank" value="medium" id="medium" checked/>

<label for="medium" class="select-tank select-tank-medium">

Средний танк<br />

<img src="images/tank/medium/all.svg" alt="" />

<span class="select-tank--plus">

Универсальность

</span>

<span class="select-tank--minus">

Средние параметры

</span>

</label>

<input type="radio" name="select-tank" value="heavy" id="heavy"/>

<label for="heavy" class="select-tank select-tank-heavy">

Тяжелый танк<br />

<img src="images/tank/heavy/all.svg" alt="" />

<span class="select-tank--plus">

Большой урон<br />

Большая прочность

</span>

<span class="select-tank--minus">

Низкая скоростельность<br />

Медлительность

</span>

</label>

</container>

<button id="button" type="button" class="select-button" name="button">Играть</button>

<br /><br /><img src="images/help.png" alt="">

</form>

<div class="loading">

<div id="progress"><span id="progress--after"></span></div>

</div>

<!-- <canvas id="canvas">Обновите браузер</canvas> -->

<div id="stage">

<canvas id="ui-layer"></canvas>

<canvas id="game-layer"></canvas>

<canvas id="background-layer"></canvas>

</div>

</body>

</html>

<link href="css/main.css" rel="stylesheet">

<link href="css/login.css" rel="stylesheet">

<link href="https://fonts.googleapis.com/css?family=Roboto:400,700" rel="stylesheet">

main.css

\* {padding: 0; margin: 0;}

body {

font-size: 62.5%;

font-family: 'Roboto', sans-serif;

}

.loading {

position: absolute;

z-index: 10;

height: 100vh;

width: 100vw;

background: #fafafa;

background-image: url('/images/PixelTank.png');

background-position: center;

background-repeat: no-repeat;

cursor: default;

}

body.active .loading {

animation-name: back;

animation-duration: 0.5s;

animation-fill-mode: forwards;

}

#stage {

position: relative;

}

canvas { position: absolute; }

#ui-layer { z-index: 3 }

#game-layer { z-index: 2 }

#background-layer { z-index: 1 }

#ui-layer {

cursor: crosshair;

}

#progress {

background: #333;

border-radius: 13px;

height: 20px;

width: 300px;

padding: 3px;

position: absolute;

top: 60%;

left: calc(50% - 150px);

}

#progress--after {

content: '';

display: block;

background: orange;

width: 33%;

height: 100%;

border-radius: 9px;

}

@keyframes back {

0% {

opacity: 1;

}

40% {

opacity: 1;

}

100% {

opacity: 0;

display: none;

z-index: -1;

}

}

login.css

.select {

display: none;

z-index: -1;

text-align: center;

position: absolute;

height: 100%;

width: 100%;

background: rgba(250, 250, 250, 1);

cursor: default;

font-size: 2.4rem;

}

.select-active {

display: block;

z-index: 10;

}

container img {

height: 30%;

width: 30%;

}

container {

margin: 1rem;

display:flex;

flex-direction: row;

flex-wrap: nowrap;

justify-content: center;

align-content: center;

align-items: center;

}

.select-name {

height: 2.4rem;

font-size: 2.1rem;

background: transparent;

border: none;

border-bottom: solid 3px #333;

text-align: center;

}

.select-name:focus {

border-bottom: solid 3px rgba(255, 211, 55, 0.9);

}

.select-button {

height: 4.2rem;

width: 16rem;

font-size: 2.8rem;

background: transparent;

border: solid 3px #333;

}

.select-tank {

max-width: 300px;

flex-grow: 1;

border: solid 3px rgba(51, 51, 51,0.3);

margin: 0.5%;

font-size: 1.8rem;

text-align: center;

height: 15rem;

}

.select-tank--plus {

display: block;

color: #309030;

font-size: 1.4rem;

}

.select-tank--minus {

display: block;

color: #903030;

font-size: 1.4rem;

}

.select-tank--light {

}

.select-tank--medium {

}

.select-tank--heavy {

}

input[type="radio"] {

display: none;

}

input[type="radio"]:checked+label{

border: solid 3px rgba(255, 211, 55, 0.9);

}

background.js

// Отрисовка игрового фона (точек)

function Background() {

ctxbg.save();

//Camera();

if(scale != 1) ctxbg.scale(scale, scale);

let roundedX = (0.5 + -vision.center.x+bg.width/2/scale) | 0;

let roundedY = (0.5 + -vision.center.y+bg.height/2/scale) | 0;

ctxbg.translate(roundedX, roundedY);

for (var x = -2560; x < 2560; x += 256) {

for (var y = -2560; y < 2560; y += 256) {

if (IsVisible(x, y,256,256)) {

ctxbg.drawImage(images.background, x, y,256,256);

}

}

}

ctxbg.restore();

}

control.js

'use strict';

window.onmousemove = event => {

control.mouse.x = event.pageX;

control.mouse.y = event.pageY;

control.mouse.angle = Math.atan2(event.pageY-game.height/2,event.pageX-game.width/2);// / Math.PI \* 180;

};

window.onmousedown = () => {

control.mouse.down = true;

};

window.onmouseup = () => {

control.mouse.down = false;

};

window.onwheel = event => { // если выключить синхр пропадет?

if(event.wheelDelta > 0) {

scale += 0.1;

} else {

scale -= 0.1;

}

if (scale < 0.5)

scale = 0.5;

if (scale > 1.4)

scale = 1.4;

};

window.onresize = () => {

game.width = window.innerWidth-4;

game.height = window.innerHeight-4;

ui.width = window.innerWidth-4;

ui.height = window.innerHeight-4;

bg.width = window.innerWidth-4;

bg.height = window.innerHeight-4;

};

window.onkeydown = event => {

if(event.keyCode == 87) control.key.w=true;

if(event.keyCode == 83) control.key.s=true;

if(event.keyCode == 65) control.key.a=true;

if(event.keyCode == 68) control.key.d=true;

if(event.keyCode == 81) control.key.q=true;

};

window.onkeyup = event => {

if(event.keyCode == 87) control.key.w=false;

if(event.keyCode == 83) control.key.s=false;

if(event.keyCode == 65) control.key.a=false;

if(event.keyCode == 68) control.key.d=false;

if(event.keyCode == 81) control.key.q=false;

};

button.onclick = () => {

PlayerName = document.forms["select"].elements["select-name"].value.trim();

PlayerTank = document.forms["select"].elements["select-tank"].value.trim();

if(PlayerTank == "light" || PlayerTank == "medium" || PlayerTank == "heavy") {

// console.log(PlayerName,PlayerTank);

document.forms["select"].classList.remove('select-active');

Connect();

GameLoop();

BGLoop();

UILoop();

}

};

var server;

worker.onmessage = event => {

switch (event.data.type) {

case 'refresh':

server=event.data.data.server;

break;

case 'init':

tank=event.data.data.tank;

BULLET\_SPEED=event.data.data.bs;

break;

}

};

setInterval(() => {

worker.postMessage({

type: 'control',

data: {

control

}

});

},1000/60);

function Connect() {

worker.postMessage({

type: 'connect',

data: {

name: PlayerName,

tank: PlayerTank

}

});

}

control.js

'use strict';

/// Player select

// Данные получение со страницы входа

var PlayerName; // Логин игрока

var PlayerTank; // Выбраный танк игрока

// Scale

var scale=1; // Коэффицент масштабирования (0.5 - 1.4)

// GameLoop

var fps = 60;

// Connect

var server;

var BULLET\_SPEED;

// Видимость

/\*

Область видимости выглядит так

@ ====

| с |

==== #

\*/

var vision = {

start: { // @ - Верхний левый угол

x: 0,

y: 0

},

center: { // с - Центр

x: 0,

y: 0

},

end: { // # - правый нижний

x: 0,

y: 0

}

};

// Параметры танков

var tank;

/// Control

// передаем нажатие мыши и клавиатуры

var control={

mouse: {

x: 0, // положение мыши по x

y: 0, // положение мыши по y

down: false, // Нажата ли кнопка

angle: 0 // На какой угол от центра находиться указатель

},

key: {

w: false, // ↑

s: false, // ↓

a: false, // ←

d: false, // →

q: false // Stats vision

}

};

game.js

// Центрирование холста на игрока

function Camera() {

if(scale != 1) ctxgame.scale(scale, scale);

let roundedX = ((0.5 + (-vision.center.x+game.width/2/scale)\*10) | 0)/10;

let roundedY = ((0.5 + (-vision.center.y+game.height/2/scale)\*10) | 0)/10;

ctxgame.translate(roundedX, roundedY);

}

// Проверка видимости объекта игроком на холсте

function IsVisible(x,y,w=0,h=0) {

let IsVisible=false;

if (vision.start.x<x+w)

if (vision.start.y<y+h)

if (vision.end.x>x)

if (vision.end.y>y)

IsVisible=true;

return IsVisible;

}

// Возврат среднего значения массива

function ArrayAverage(array) {

return (array.reduce(function(previousValue, currentValue) {

return previousValue + currentValue;

})/array.length);

}

// Отрисовка танков на игровом поле

function DrawTanks() {

ctxgame.save();

Camera(); // Центриование на игрока

for(var ui = 0, ul = server.users.length; ui < ul; ui++) {

if (IsVisible(server.users[ui].x-40, server.users[ui].y-40,80,80)) {// Видит ли танки

var roundedX = ((0.5 + server.users[ui].x\*10) | 0)/10;

var roundedY = ((0.5 + server.users[ui].y\*10) | 0)/10;

switch (server.users[ui].tank) {

case 'light':

// Отрисовка шасси легкого танка

ctxgame.save();

ctxgame.translate(roundedX, roundedY);

ctxgame.rotate(server.users[ui].chassis.angle);

ctxgame.drawImage(images.lchassis, -40,-40, 80,80);

ctxgame.restore();

// Отрисовка башни легкого танка

ctxgame.save();

ctxgame.translate(roundedX, roundedY);

ctxgame.rotate(server.users[ui].turret.angle);

ctxgame.drawImage(images.lturret, -21.5, -40, 80,80);

ctxgame.restore();

break;

case 'medium':

// Отрисовка шасси среднего танка

ctxgame.save();

ctxgame.translate(roundedX, roundedY);

ctxgame.rotate(server.users[ui].chassis.angle);

ctxgame.drawImage(images.mchassis, -40,-40, 80,80);

ctxgame.restore();

// Отрисовка башни среднего танка

ctxgame.save();

ctxgame.translate(roundedX, roundedY+2.5);

ctxgame.rotate(server.users[ui].turret.angle);

ctxgame.drawImage(images.mturret, -33.5, -40, 80,80);

ctxgame.restore();

break;

case 'heavy':

// Отрисовка шасси тяжелого танка

ctxgame.save();

ctxgame.translate(roundedX, roundedY);

ctxgame.rotate(server.users[ui].chassis.angle);

ctxgame.drawImage(images.hchassis, -40,-40, 80,80);

ctxgame.restore();

// Отрисовка башни тяжелого танка

ctxgame.save();

ctxgame.translate(roundedX, Math.round(server.users[ui].y\*10)/10);

ctxgame.rotate(server.users[ui].turret.angle);

ctxgame.drawImage(images.hturret, -27, -40, 80,80);

ctxgame.restore();

break;

}

ctxgame.save();

// Отображение ника игрока

ctxgame.textAlign = "center";

ctxgame.textBaseline = "bottom";

ctxgame.fillStyle = "#a80b0b";

ctxgame.font = 'bold 24px monospace';

ctxgame.fillText(server.users[ui].name,roundedX, roundedY-56);

// Отображение хп

ctxgame.beginPath();

ctxgame.lineWidth=4;

ctxgame.strokeStyle = "#00cc00";

ctxgame.moveTo(roundedX-50,roundedY-54);

ctxgame.lineTo(roundedX-50+(100\*(server.users[ui].hp/tank[server.users[ui].tank].hp)),roundedY-54);

ctxgame.stroke();

ctxgame.beginPath();

ctxgame.strokeStyle = "#000000";

ctxgame.moveTo(roundedX-50+(100\*(server.users[ui].hp/tank[server.users[ui].tank].hp)),roundedY-54);

ctxgame.lineTo(roundedX+50,roundedY-54);

ctxgame.stroke();

ctxgame.restore();

}

}

ctxgame.restore();

}

function DrawBullets() {

ctxgame.save();

Camera();

for(var bi = 0, bl = server.bullets.length; bi < bl; bi++) {

if (IsVisible(server.bullets[bi].x, server.bullets[bi].y)) {

ctxgame.save();

ctxgame.translate(server.bullets[bi].x, server.bullets[bi].y);

ctxgame.rotate(server.bullets[bi].angle);

ctxgame.drawImage(images.bullet, -12.5,-12.5,25,25);

ctxgame.restore();

}

}

ctxgame.restore();

}

//Модуль апроксимации

function Approximation() {

// Полет снарядов

for(var bi = 0, bl = server.bullets.length; bi < bl; bi++) {

if (IsVisible(server.bullets[bi].x-12.5, server.bullets[bi].y-12.,25,25)) {

// Передвижение снаряда на величину скорости

server.bullets[bi].x +=s(BULLET\_SPEED)\*Math.cos(server.bullets[bi].angle);

server.bullets[bi].y +=s(BULLET\_SPEED)\*Math.sin(server.bullets[bi].angle);

server.bullets[bi].x = ((0.5 + server.bullets[bi].x\*10) | 0)/10;

server.bullets[bi].y = ((0.5 + server.bullets[bi].y\*10) | 0)/10;

}

}

// Передвижение танка

for(var ui = 0, ul = server.users.length; ui < ul; ui++) {

if (server.users[ui].id == server.id) {

// Обновление данных о направление орудия

//server.users[ui].turret.angle = mouse.angle;

server.users[ui].turret.angle = RotateTurret(server.users[ui].turret.angle,control.mouse.angle,s(tank[server.users[ui].tank].turret.angle\*Math.PI/180));

// Вычисление изменений скорости

if(control.key.w) {

server.users[ui].speed += s(tank[server.users[ui].tank].acceleration\*ac(server.users[ui].speed,tank[server.users[ui].tank].max\_speed));

if (server.users[ui].speed > tank[server.users[ui].tank].max\_speed) server.users[ui].speed = tank[server.users[ui].tank].max\_speed;

}

if(control.key.s) {

server.users[ui].speed -= s(tank[server.users[ui].tank].acceleration\*0.6\*ac(server.users[ui].speed,tank[server.users[ui].tank].max\_speed));

if (server.users[ui].speed < -tank[server.users[ui].tank].max\_speed\*0.3) server.users[ui].speed = -tank[server.users[ui].tank].max\_speed\*0.3;

}

if((!control.key.s || server.users[ui].speed > 0) && !control.key.w) {

if (server.users[ui].speed < 10 && server.users[ui].speed > -10) {

server.users[ui].speed = 0;

}

if (server.users[ui].speed <= -10) {

server.users[ui].speed += s(tank[server.users[ui].tank].acceleration\*3\*ac(server.users[ui].speed,server.users[ui].max\_speed));

}

if (server.users[ui].speed >= 10) {

server.users[ui].speed -= s(tank[server.users[ui].tank].acceleration\*3\*ac(server.users[ui].speed,server.users[ui].max\_speed));

}

}

// Поворот танка

if(control.key.a) server.users[ui].chassis.angle -= s(tank[server.users[ui].tank].chassis.angle\*Math.PI/180\*ac(server.users[ui].speed,tank[server.users[ui].tank].max\_speed));

if(control.key.d) server.users[ui].chassis.angle += s(tank[server.users[ui].tank].chassis.angle\*Math.PI/180\*ac(server.users[ui].speed,tank[server.users[ui].tank].max\_speed));

vision.start.x = server.users[ui].x-game.width/2/scale;

vision.start.y = server.users[ui].y-game.width/2/scale;

vision.center.x = server.users[ui].x;

vision.center.y = server.users[ui].y;

vision.end.x = server.users[ui].x+game.width/2/scale;

vision.end.y = server.users[ui].y+game.height/2/scale;

}

if (IsVisible(server.users[ui].x-40, server.users[ui].y-40,80,80)) {

// Передвижение танка на величину скорости

server.users[ui].x +=s(server.users[ui].speed)\*Math.cos(server.users[ui].chassis.angle);

server.users[ui].y +=s(server.users[ui].speed)\*Math.sin(server.users[ui].chassis.angle);

// Граница карты

if(server.users[ui].x > 2560) server.users[ui].x = 2560;

if(server.users[ui].y > 2560) server.users[ui].y = 2560;

if(server.users[ui].x < -2560) server.users[ui].x = -2560;

if(server.users[ui].y < -2560) server.users[ui].y = -2560;

}

}

}

// Функция плавного поворота башни

function RotateTurret(ta,ma,s) {

if (ta < 0)

ta = ta + 2\*Math.PI;

let a = ta - ma;

if (a > Math.PI)

a = a - 2\*Math.PI;

else if (a < -Math.PI)

a = a + 2\*Math.PI;

if (a >= -s && a <= s)

ta = ma;

else if (a > s)

ta = ta - s;

else if (a < -s)

ta = ta + s;

return ta;

}

function s(v) {

// Все действия происходят в виде Pixel/Second и тд

return v/fps;

}

function ac(speed, max\_speed=120) {

//(2+x^0.1)/(2+x^0.3) (x from 0 to 100)

// получаем коэффицент скорости разворота

// от 1 до почти 0.616

// перемножаем на скорость поворота танка

return (1+Math.pow((Math.abs(speed)/max\_speed)\*10,0.1))/(1+Math.pow((Math.abs(speed)/max\_speed)\*10,0.3));

}

loader.js

(function( w ){

var loadJS = function( src, cb ){

"use strict";

var ref = w.document.getElementsByTagName( "script" )[ 0 ];

var script = w.document.createElement( "script" );

script.src = src;

script.async = true;

ref.parentNode.insertBefore( script, ref );

if (cb && typeof(cb) === "function") {

script.onload = cb;

}

return script;

};

// commonjs

if( typeof module !== "undefined" ){

module.exports = loadJS;

}

else {

w.loadJS = loadJS;

}

}( typeof global !== "undefined" ? global : this ));

// Данные о изображениях

var images = {};

var loadedImages = 0;

var numImages = 0;

var sourceImages = {

//background: "images/background.svg",

background: "images/background.png",

lchassis: "images/tank/light/chassis.svg",

lturret: "images/tank/light/turret.svg",

mchassis: "images/tank/medium/chassis.svg",

mturret: "images/tank/medium/turret.svg",

hchassis: "images/tank/heavy/chassis.svg",

hturret: "images/tank/heavy/turret.svg",

bullet: "images/bullet.svg",

death: "images/death.png",

};

// Данные о JS

var loadedJS = 0;

var numJS = 0;

var sourceJS = {

game: "js/game.js",

ui: "js/ui.js",

background: "js/background.js",

render: "js/render.js",

control: "js/control.js",

env: "js/env.js"

};

// Подсчет загружаемых файлов

for(var src in sourceImages) {

numImages++;

}

for(var src in sourceJS) {

numJS++;

}

for(var src in sourceImages) {

images[src] = new Image();

images[src].src = sourceImages[src];

images[src].onload = function () {

loadedImages++;

Loading();

};

}

for(var src in sourceJS) {

loadJS(sourceJS[src],function () {

loadedJS++;

Loading();

});

}

function Loading() {

if(loadedJS+loadedImages == numJS+numImages) {

document.body.classList.add('active');

}

document.getElementById('progress--after').style.width

=Math.round((loadedJS+loadedImages)/(numJS+numImages)\*100) + '%';

console.log(Math.round((loadedJS+loadedImages)/(numJS+numImages)\*100) + "% resource loaded");

}

var worker = new Worker('js/worker.js');

render.js

'use strict';

var game = document.getElementById("game-layer"),

ctxgame = game.getContext('2d');

game.width = window.innerWidth-4;

game.height = window.innerHeight-4;

var interation=0;

function GameLoop() {

ctxgame.clearRect(0, 0, game.width, game.height);

Approximation();

DrawBullets();

DrawTanks();

requestAnimationFrame(GameLoop);

interation++;

}

var ui = document.getElementById("ui-layer"),

ctxui = ui.getContext('2d');

ui.width = window.innerWidth-4;

ui.height = window.innerHeight-4;

function UILoop() {

if(interation == 2) {

ctxui.clearRect(0, 0, ui.width, ui.height);

DrawLog();

Radar();

DrawStat();

interation=0;

}

requestAnimationFrame(UILoop);

}

var bg = document.getElementById("background-layer"),

ctxbg = bg.getContext('2d');

bg.width = window.innerWidth-4;

bg.height = window.innerHeight-4;

function BGLoop() {

ctxbg.clearRect(0, 0, bg.width, bg.height);

Background();

requestAnimationFrame(BGLoop);

}

socket.js

// Отрисоквка радара

function Radar() {

// Отрисовка фона радара

ctxui.save();

ctxui.beginPath();

ctxui.lineWidth = 2;

ctxui.fillStyle = "rgba(255, 255, 255, 0.85)";

ctxui.arc(112, 112, 96, 0, Math.PI\*2, false);

ctxui.closePath();

ctxui.stroke();

ctxui.fill();

//Данные об игроке

let angle;

let x,y;

server.users.forEach(function (value, index) {

if (value.id == server.id) {

angle = value.turret.angle;

x = value.x;

y = value.y;

}

});

// Данные об окружающих игроках

server.users.forEach(function (value, index) {

if (value.id != server.id) {

let r = Math.sqrt(Math.pow(Math.abs(value.y-y),2)+Math.pow(Math.abs(value.x-x),2));

if(r <= 2000) {

let degree = Math.atan2(value.y-y,value.x-x);

ctxui.beginPath();

ctxui.fillStyle = "rgb(221, 49, 49)";

ctxui.arc(112+96\*(r/2000)\*Math.cos(degree), 112+96\*(r/2000)\*Math.sin(degree), 2, 0, Math.PI\*2, false);

ctxui.closePath();

ctxui.fill();

}

}

});

// Отрисовка сектора направления орудия

ctxui.beginPath();

ctxui.lineWidth = 1;

ctxui.fillStyle = "rgba(76, 255, 3, 0.33)";

ctxui.arc(112,112,96,angle-(Math.PI/6),angle+(Math.PI/6));

ctxui.lineTo(112, 112);

ctxui.closePath();

ctxui.stroke();

ctxui.fill();

ctxui.restore();

}

// Отрисовка логов о смертях и убийствах

function DrawLog() {

server.death.forEach(function (value, index) {

ctxui.save();

let death\_name="",killer\_name="";

server.users.forEach(function (player) {

if (player.id == value.killer)

killer\_name = player.name;

if (player.id == value.death)

death\_name = player.name;

});

ctxui.textBaseline = "top";

ctxui.fillStyle = "#a80b0b";

ctxui.font = 'bold 24px monospace';

ctxui.textAlign = "start";

ctxui.fillText(death\_name, game.width-225, 12.5+36\*index);

ctxui.textAlign = "end";

ctxui.fillText(killer\_name, game.width-300, 12.5+36\*index);

ctxui.drawImage(images.death, game.width-280, 12.5+36\*index, 26, 38);

ctxui.restore();

});

}

// Отрисовка логов о смертях и убийствах

function DrawStat() {

if (control.key.q) {

ctxui.save();

ctxui.beginPath();

ctxui.lineWidth=3;

ctxui.strokeStyle = "#000000";

ctxui.moveTo(game.width\*.8,game.height\*.8);

ctxui.lineTo(game.width\*.8,game.height\*.2);

ctxui.lineTo(game.width\*.2,game.height\*.2);

ctxui.lineTo(game.width\*.2,game.height\*.8);

ctxui.lineTo(game.width\*.8,game.height\*.8);

ctxui.fillStyle = "white";

ctxui.fill();

ctxui.stroke();

ctxui.textBaseline = "top";

ctxui.fillStyle = "#000000";

ctxui.font = 'bold 24px monospace';

ctxui.textAlign = "center";

ctxui.fillText("TOP 10 PLAYER", game.width\*.5, game.height\*.21);

ctxui.textBaseline = "top";

ctxui.fillStyle = "#000000";

ctxui.font = 'bold 20px monospace';

ctxui.textAlign = "start";

server.stats.sort(sDecrease);

ctxui.fillText('#', game.width\*.24, game.height\*(.28));

ctxui.fillText('Login', game.width\*.28, game.height\*(.28));

ctxui.fillText('Kill', game.width\*.64, game.height\*(.28));

ctxui.fillText('Death', game.width\*.72, game.height\*(.28));

server.stats.forEach(function (value, index) {

if (index < 10) {

let name;

server.users.forEach(function (val) {

if(value.id == val.id) name = val.name;

});

ctxui.fillText(index+1, game.width\*.24, game.height\*(.32+0.043\*index));

ctxui.fillText(name, game.width\*.28, game.height\*(.32+0.043\*index));

ctxui.fillText(value.kill, game.width\*.64, game.height\*(.32+0.043\*index));

ctxui.fillText(value.death, game.width\*.72, game.height\*(.32+0.043\*index));

}

});

ctxui.restore();

}

}

function sDecrease(i, ii) { // По убыванию убийств

if (i.kill > ii.kill)

return -1;

else if (i.kill < ii.kill)

return 1;

else

return 0;

}

worker.js

self.importScripts('socket.io-1.4.5.js');

//self.importScripts('p2.js');

var socket = io('http://localhost:3600/',{transports: ['websocket']});

//var socket = io('https://test.mertico.ru/',{transports: ['websocket'], secure: true});

var server={

death:[],

bullets:[],

users:[],

stats:[],

id:0

};

var timeStamp=0, startTime=0;

// передаем нажатие мыши и клавиатуры

var control={

mouse: {

x: 0, // положение мыши по x

y: 0, // положение мыши по y

down: false, // Нажата ли кнопка

angle: 0 // На какой угол от центра находиться указатель

},

key: {

w: false, // ↑

s: false, // ↓

a: false, // ←

d: false // →

}

};

socket.on('connect', () => { server.id=server.id; });

onmessage = event => {

var message = event.data;

switch (message.type) {

case 'connect':

//socket.on('connect', function () {

socket.emit('join', {name: message.data.name, tank: message.data.tank});

//socket.emit('join', {name: 123, tank: 13});

//});

break;

case 'control':

control=message.data.control;

break;

}

};

socket.on('init', function (m) {

server.stats=m.stats;

postMessage({

type: 'init',

data: {

tank: m.tank,

bs: m.bs

}

});

});

socket.on('stats', function (m) {

m.data.death.forEach(function(element, index) {

server.death.unshift({killer: element[0], death: element[1], life : 3});

});

server.death.splice(5, 1);

server.stats=m.data.stats;

});

socket.on('refresh', function (m) {

// Проверка временой метки (насколько актуален пакет данных)

if(m.t >= timeStamp) {

timeStamp = m.t;

[server.users,server.bullets] = InSend(m.data);

postMessage({

type: 'refresh',

data: {

server

}

});

}

});

setInterval(() => {

socket.emit('refresh', sendControl(control));

},1000/30);

setInterval(() => {

server.death.forEach(function (value, index) {

value.life-=1;

if(value.life<=0)

server.death.splice(index, 1);

});

},1000);

function sendControl(control) {

return {

m: control.key.w && 1 ||

control.key.s && -1 ||

0,

r: control.key.d && 1 ||

control.key.a && -1 ||

0,

a: ((0.5 + control.mouse.angle\*100000) | 0)/100000,

f: control.mouse.down

};

}

function InSend(data) {

let users=[], bullets=[];

for(var index = 0, length = data[0].length; index < length; index++) {

users[index] = {

x: data[0][index][0],

y: data[0][index][1],

speed: data[0][index][2],

turret: {

angle: data[0][index][3]

},

chassis: {

angle: data[0][index][4]

},

hp: data[0][index][5],

tank: data[0][index][6],

name: data[0][index][7],

id: data[0][index][8]

};

}

for(var index = 0, length = data[1].length; index < length; index++) {

bullets[index] = {

x: data[1][index][0],

y: data[1][index][1],

angle: data[1][index][2]

};

}

return [users, bullets];

}