Министерство образования и науки Украины

Черниговский национальный технологический университет

Кафедра информационных и компьютерных систем

Проект   
«Игра «Вертолет»»

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Исполнители:

студенты гр. КИ-131 Квитницкий Р.Н.

Ткаченко Е.Е

Полторак А.П.

2016

Требования, предъявляемые к разрабатываемой ИГРЕ

* Запуск посредством локального сервера
* Исправное отображение всех элементов
* Отображение подгружаемого js файла
* Окно браузера должно служить в качестве консоля
* Реализация в текстовом редакторе Sublime
* Исправное отображение подгружаемых иконок объектов
* Старт игры от обновления страницы
* Окончание игры по количеству столкновений с приградами

**ОПИСАНИЕ МЕТОДОВ**

Метод spawnEnemy – цикл создания индексов врагов, при столкновении на которых у объекта вертолет отнимается очко. Передается в качестве параметра максимальное количество врагов.

Метод startCreatingEnemies – создание количества врагов, заданное переменной spawnAmount через интервал времени spawnTime.

Метод stopCreatingEnemies() – очищает интервал от врагов.

Метод Player() описывает фигуру игрока, а Enemy() – врагов.

Методы checkKeyDown(), checkKeyUp() позволяют пользователю управлять объектом самолет.

**ТЕКСТ ПРОГРАММЫ**

Текст файла game.js

window.onload = init;

var map;

var ctxMap;

var drawBtn;

var clearBtn;

var stats;

var ctxStats;

var pl;

var ctxPl;

var enemyCv;

var ctxEnemy;

var gameWidth = 800;

var gameHeight = 480;

var bg = new Image();

bg.src = "img/bg.png";

var plTiles = new Image();

plTiles.src = "img/player1.png";

var enemyTiles = new Image();

enemyTiles.src = "img/enemy1.png";

var player;

var enemies = [];

var isPlaying;

//For creating enemies

var spawnInterval;

var spawnTime = 3800;

var spawnAmount = 6;

var requestAnimFrame = window.requestAnimationFrame ||

window.webkitRequestAnimationFrame ||

window.mozRequestAnimationFrame ||

window.oRequestAnimationFrame ||

window.msRequestAnimationFrame;

function init(){

map = document.getElementById("map");

ctxMap = map.getContext("2d");

map.width= gameWidth;

map.height = gameHeight;

pl = document.getElementById("map");

ctxPl = map.getContext("2d");

enemyCv = document.getElementById("enemy");

ctxEnemy = map.getContext("2d");

stats = document.getElementById("stats");

ctxStats = map.getContext("2d");

pl.width= gameWidth;

pl.height = gameHeight;

stats.width= gameWidth;

stats.height = gameHeight;

ctxStats.fillStyle = "#303031";

ctxStats.font = "bold 15pt Arial"

drawBtn = document.getElementById("drawBtn");

clearBtn = document.getElementById("clearBtn")

drawBtn.addEventListener("click", drawRect,false );

clearBtn.addEventListener("click", clearRect,false );

player = new Player();

// enemy = new Enemy();

drawBg();

// drawPlayar();

// draw();

startLoop();

updateStats();

document.addEventListener("keydown", checkKeyDown, false);

document.addEventListener("keyup", checkKeyUp, false);

}

function spawnEnemy(count){

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

enemies[i] = new Enemy();

}

}

function loop(){

if(isPlaying){

draw();

update();

requestAnimFrame(loop)

}

}

function startCreatingEnemies(){

stopCreatingEnemies();

spawnInterval = setInterval(function(){spawnEnemy(spawnAmount)},spawnTime);

}

function stopCreatingEnemies(){

clearInterval(spawnInterval);

}

function startLoop(){

isPlaying = true;

loop();

startCreatingEnemies();

}

function stopLoop(){

isPlaying=false;

}

function draw(){

player.draw();

// enemy.draw();

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

enemies[i].draw();

}

}

function update(){

player.update();

// enemy.update();

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

enemies[i].update();

}

}

function Player(){

this.srcX = 0;

this.srcY = 0;

this.drawX = 0;

this.drawY = 0;

this.width = 130;

this.height = 77;

this.isUp = false;

this.isDown = false;

this.isRight = false;

this.isLeft = false;

this.speed = 5;

}

function Enemy(){

this.srcX = 0;

this.srcY = 0;

this.drawX = Math.floor(Math.random()\*gameWidth)+gameWidth;

this.drawY = Math.floor(Math.random()\*gameHeight );

this.width = 77;

this.height = 62;

this.speed = 7;

}

Enemy.prototype.draw = function (){

ctxEnemy.drawImage(enemyTiles, this.srcX, this.srcY, this.width,

this.height, this.drawX, this.drawY, this.width, this.height);

}

Enemy.prototype.update = function (){

this.drawX -=7;

if(this.drawX + this.width < 0){

this.destroy();

// this.drawX=Math.floor(Math.random()\*gameWidth)+ gameWidth;

// this.drawY=Math.floor(Math.random()\*gameHeight);

}

// ctxEnemy.drawImage(enemyTiles, this.srcX, this.srcY, this.width,

// this.height, this.drawX, this.drawY, this.width, this.height);

}

Enemy.prototype.destroy = function(){

enemies.splice(enemies.indexOf(this),1);

}

Player.prototype.draw = function (){

clearCtxEnemy();

drawBg();

ctxPl.drawImage(plTiles, this.srcX, this.srcY, this.width,

this.height, this.drawX, this.drawY, this.width, this.height);

}

Player.prototype.update = function (){

if(this.drawX <0) this.drawX = 0;

if(this.drawX > gameWidth - this.width) this.drawX = gameWidth - this.width;

if(this.drawY <0) this.drawY = 0;

if(this.drawY >gameHeight - this.height) this.drawY = gameHeight - this.height;

this.chooseDir();

}

Player.prototype.chooseDir = function (){

if(this.isUp){

this.drawY -=this.speed;

}

if(this.isDown){

this.drawY +=this.speed;

}

if(this.isRight){

this.drawX +=this.speed;

}

if(this.isLeft){

this.drawX -=this.speed;

}

}

function checkKeyDown(e){

var keyId = e.keyCode || e.which;

var keyChar = String.fromCharCode(keyId);

if(keyChar == "W"){

player.isUp = true;

e.preventDefault();

}

if(keyChar == "S"){

player.isDown = true;

e.preventDefault();

}

if(keyChar == "D"){

player.isRight = true;

e.preventDefault();

}

if(keyChar == "A"){

player.isLeft = true;

e.preventDefault();

}

}

function checkKeyUp(e){

var keyId = e.keyCode || e.which;

var keyChar = String.fromCharCode(keyId);

if(keyChar == "W"){

player.isUp = false;

e.preventDefault();

}

if(keyChar == "S"){

player.isDown = false;

e.preventDefault();

}

if(keyChar == "D"){

player.isRight = false;

e.preventDefault();

}

if(keyChar == "A"){

player.isLeft = false;

e.preventDefault();

}

}

function drawRect(){

ctxMap.fillStyle = "#3D3D3D";

ctxMap.fillRect(10, 10, 100, 100);

}

function clearRect(){

ctxMap.clearRect(0, 0, 800, 500);

}

function drawBg(){

ctxMap.drawImage(bg, 0, 0, 800, 480, 0, 0, 800, 480);

}

function clearCtxPlayer(){

ctxPl.clearRect(0, 0, gameWidth, gameHeight);

}

function clearCtxEnemy(){

ctxEnemy.clearRect(0, 0, gameWidth, gameHeight);

}

function updateStats(){

ctxStats.clearRect(0, 0, gameWidth, gameHeight);

ctxStats.fillText("Player", 10, 30 )

}

function drawPlayar(){

ctxMap.drawImage(plTiles, 0, 0, 130, 77, 0, 0, 130, 77);

}

Текст файла index.php

<html>

<head>

<meta charset="UTF-8">

<title>Game</title>

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

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

</head>

<body>

<canvas id="map" ></canvas>

<canvas id="player" ></canvas>

<canvas id="enemy" ></canvas>

<canvas id="stats"></canvas>

<input id="drawBtn" type="button" value="Draw"></input>

<input id="clearBtn" type="button" value="Clear"></input>

</body>

</html>

**РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

1. Ввести в адресную строку браузера ссылку на игру

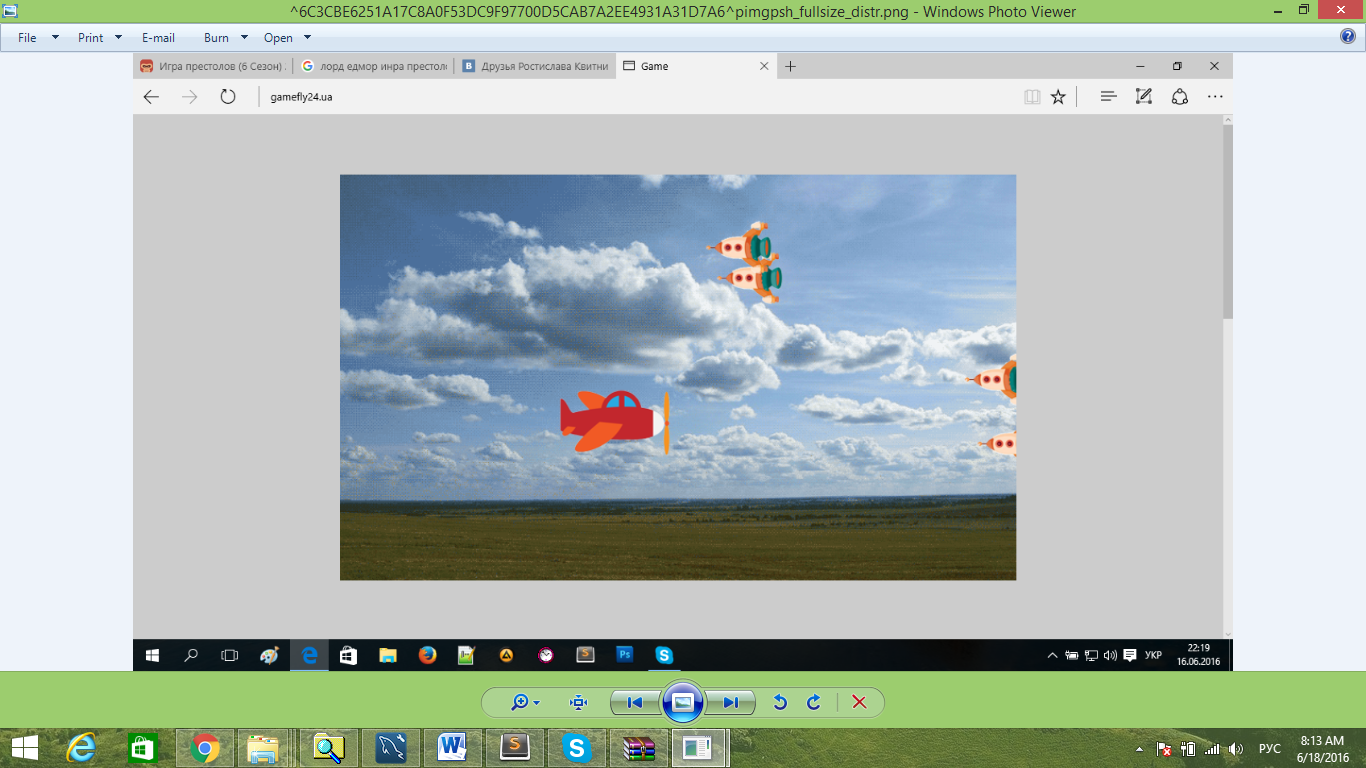


Рис. 1.1 Адресная строка со ссылкой на игру

1. Зажимать клавиши управления
   1. W - вверх
   2. A - влево
   3. D - вниз
   4. S – вправо
2. Управлять объектом вертолет

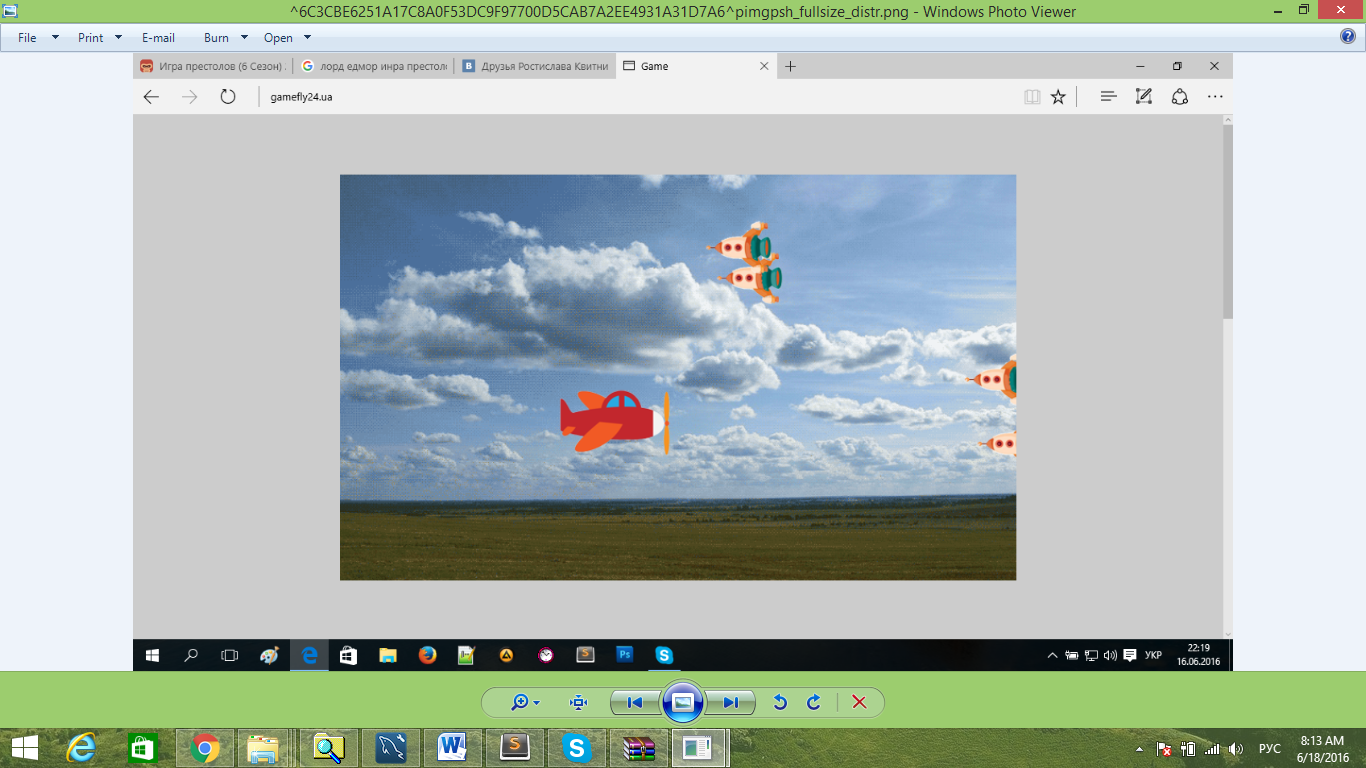


Рис 2.1 Объект «Вертолет»

1. Стараться избегать столкновений с объектами типа враг

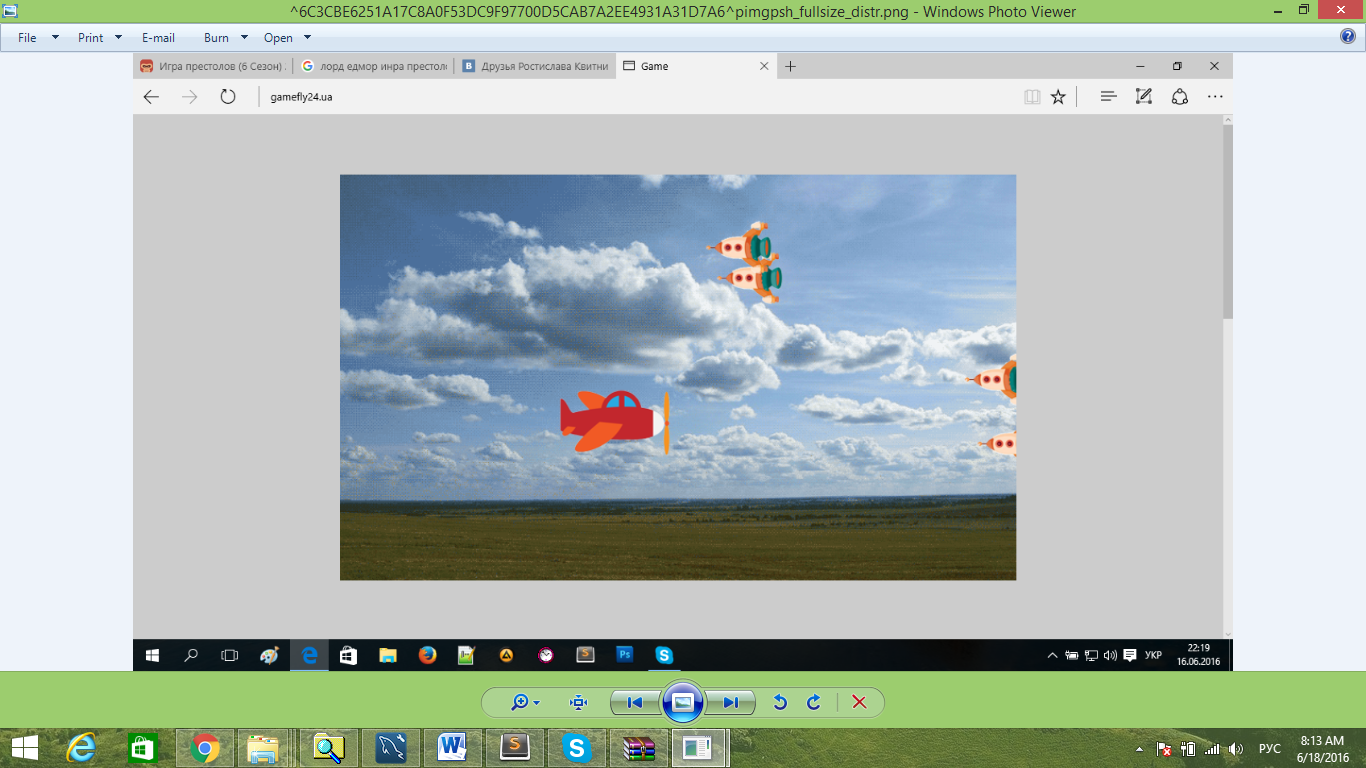


Рис 2.3 Два объекта типа «Враг»

**СКРИНШОТ ИГРЫ**

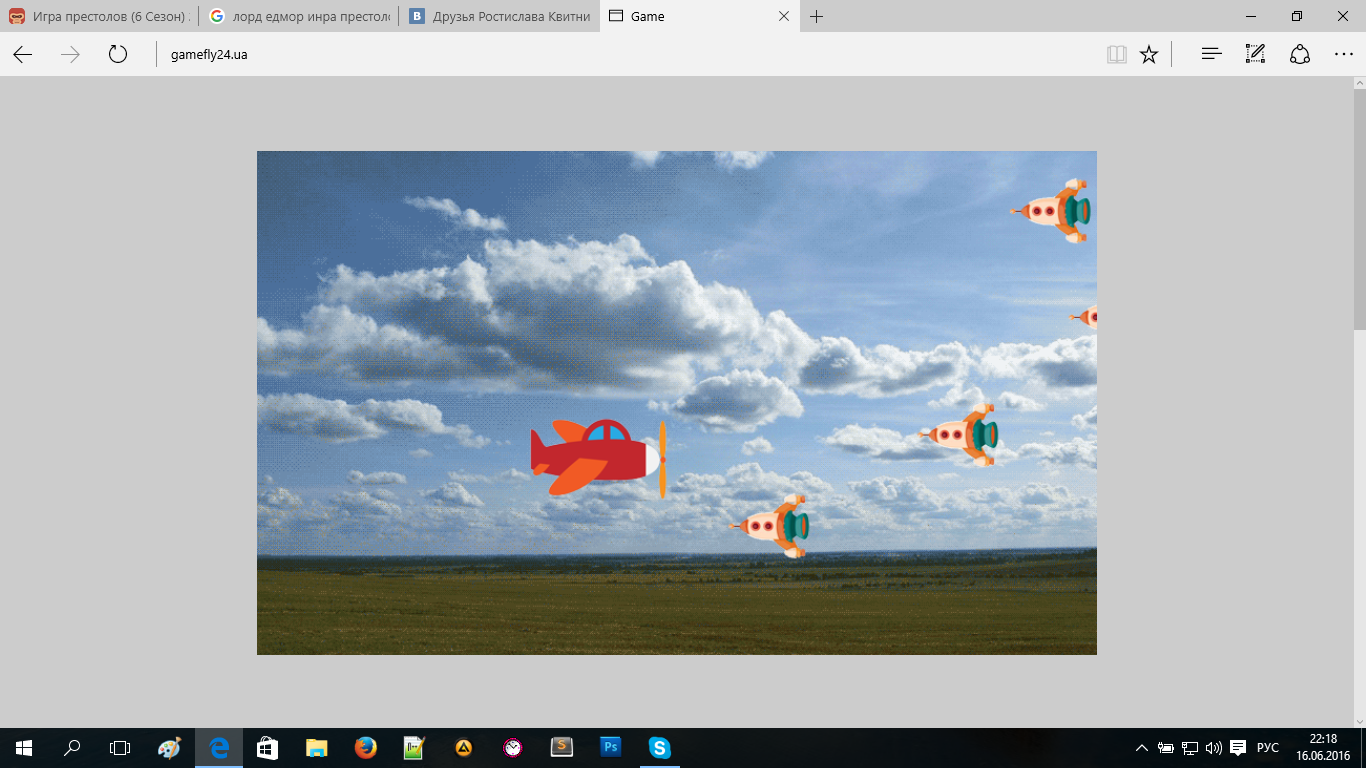


Рис 2.4 Скриншот игры

**ВЫВОДЫ**

При разработке игры «Вертолет» были использованы такие программные средства как языки html, php и javascript. В качестве среды разработки был использован текстовый редактор. Denwer послужил выбранным пакетом сервером. Было использовано подключение внешних файлов иконок и изображений. Игра была реализована в браузере.