МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР

ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра ПИ

ОТЧЕТ

по лабораторной работе № 5

по теме

“ Создание самодокументирующегося кода”

Выполнил:

ст. группы ПИ-15Б

Луков П.Р.

Преподаватели:

Грищенко Д.

Донецк 2018

/\*

\* хранит canvas

\* var HTMLElement canvas

\*/

var canvas = document.getElementById("myCanvas");

/\*

\* хранит контекст рисования на холсте

\* var any ctx

\*/

var = canvas.getContext("2d");

/\*

\* радиус шара

\* var int ballRadius

\*/

var ballRadius = 10;

/\*

\* координата x шара

\* var double x

\*/

var x = canvas.width / 2;

/\*

\* координата y шара

\* var double y

\*/

var y = canvas.height - 30;

/\*

\* скорость передвижения шара по x

\* var int dx

\*/

var dx = 2;

/\*

\* скорость передвижения шара по y

\* var int dy

\*/

var dy = -2;

/\*

\* высота передвижной досточки

\* var int paddleHeight

\*/

var paddleHeight = 15;

/\*

\* ширина передвижной досточки

\* var int paddleWidth

\*/

var paddleWidth = 75;

/\*

\* координата x передвижной досточки

\* var double paddleX

\*/

var paddleX = (canvas.width - paddleWidth) / 2;

/\*

\* сдвиг вправо

\* var bool rightPressed

\*/

var rightPressed = false;

/\*

\* сдвиг влево

\* var bool leftPressed

\*/

var leftPressed = false;

/\*

\* кол-во строк досточек

\* var int brickRowCount

\*/

var brickRowCount = 5;

/\*

\* кол-во колонок досточек

\* var int brickColumnCount

\*/

var brickColumnCount = 3;

/\*

\* ширина досточки

\* var int brickWidth

\*/

var brickWidth = 75;

/\*

\* высота досточки

\* var int brickHeight

\*/

var brickHeight = 20;

/\*

\* отступ между досточками

\* var int brickPadding

\*/

var brickPadding = 10;

/\*

\* отступ от верха

\* var int brickOffsetTop

\*/

var brickOffsetTop = 30;

/\*

\* отступ от левого края

\* var int brickOffsetLeft

\*/

var brickOffsetLeft = 30;

/\*

\* счет

\* var int score

\*/

var score = 0;

/\*

\* жизни

\* var int lives

\*/

var lives = 3;

/\*

\* массив досточек

\*/

var bricks = [];

for (var c = 0; c < brickColumnCount; c++) {

bricks[c] = [];

for (var r = 0; r < brickRowCount; r++) {

bricks[c][r] = { x: 0, y: 0, status: 1 };

}

}

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

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

document.addEventListener("mousemove", mouseMoveHandler, false);

/\*

\* работа со свойствами rightPressed и leftPressed

\* если код нажатой кнопки равен 39 rightPressed устанавливается true

\* если код нажатой кнопки равен 37 leftPressed устанавливается true

\*/

function keyDownHandler(e) {

if (e.keyCode == 39) {

rightPressed = true;

}

else if (e.keyCode == 37) {

leftPressed = true;

}

}

/\*

\* работа со свойствами rightPressed и leftPressed

\* если код отпущенной кнопки равен 39 rightPressed устанавливается false

\* если код отпущенной кнопки равен 37 leftPressed устанавливается false

\*/

function keyUpHandler(e) {

if (e.keyCode == 39) {

rightPressed = false;

}

else if (e.keyCode == 37) {

leftPressed = false;

}

}

/\*

\* работа со свойством paddleX

\* если координата x курсора больше 0 и меньше ширины холста

\* передвижной дощечке устанвливается этот x

\*/

function mouseMoveHandler(e) {

var relativeX = e.clientX - canvas.offsetLeft;

if (relativeX > 0 && relativeX < canvas.width) {

paddleX = relativeX - paddleWidth / 2;

}

}

/\*

\* Обнаружение столкновения

\* если дощечка еще не уничтожена и координаты шара попадают в ее диапазон

\* ей присваивается статус уничтожена, а счет увеличивается на 1

\* если счет равен кол-во дощечек, значит все дощечки уничтожены

\* и выводится сообщение о победе

\*/

function collisionDetection() {

for (var c = 0; c < brickColumnCount; c++) {

for (var r = 0; r < brickRowCount; r++) {

var b = bricks[c][r];

if (b.status == 1) {

if (x > b.x && x < b.x + brickWidth && y > b.y && y < b.y + brickHeight) {

dy = -dy;

b.status = 0;

score++;

if (score == brickRowCount \* brickColumnCount) {

alert("YOU WIN, CONGRATS!");

document.location.reload();

}

}

}

}

}

}

/\*

\* Отрисовка шара

\*/

function drawBall() {

ctx.beginPath();

ctx.arc(x, y, ballRadius, 0, Math.PI \* 2);

ctx.fillStyle = "#0095DD";

ctx.fill();

ctx.closePath();

}

/\*

\* Отрисовка передвижной досточки

\*/

function drawPaddle() {

ctx.beginPath();

ctx.rect(paddleX, canvas.height - paddleHeight, paddleWidth, paddleHeight);

ctx.fillStyle = "#0095DD";

ctx.fill();

ctx.closePath();

}

/\*

\* Отрисовка досточек

\*/

function drawBricks() {

for (var c = 0; c < brickColumnCount; c++) {

for (var r = 0; r < brickRowCount; r++) {

if (bricks[c][r].status == 1) {

var brickX = (r \* (brickWidth + brickPadding)) + brickOffsetLeft;

var brickY = (c \* (brickHeight + brickPadding)) + brickOffsetTop;

bricks[c][r].x = brickX;

bricks[c][r].y = brickY;

ctx.beginPath();

ctx.rect(brickX, brickY, brickWidth, brickHeight);

ctx.fillStyle = "#0095DD";

ctx.fill();

ctx.closePath();

}

}

}

}

/\*

\* Отрисовка результата

\*/

function drawScore() {

ctx.font = "16px Arial";

ctx.fillStyle = "#0095DD";

ctx.fillText("Score: " + score, 8, 20);

}

/\*

\* Отрисовка жизней

\*/

function drawLives() {

ctx.font = "16px Arial";

ctx.fillStyle = "#0095DD";

ctx.fillText("Lives: " + lives, canvas.width - 65, 20);

}

/\*

\* Отрисовка холста

\* если x шара выходит за рамки холста, то мы инвертируем его скорость по x

\* если y шара выходит за рамки холста вверх, то мы инвертируем его скорость по y

\* если y шара выходит за рамки холста вниз и попадает на передвижную дощечку,

\* то мы инвертируем его скорость по y,

\* если нет то уменьшаем кол-во жизней,

\* если ко-во жизней будет равно 0, игра будет закончена и будет показано сообщение о поражении

\* если свойство rightPressed включено и есть куда сдвинуться мы передвигаем дощечку вправо

\* если свойство leftPressed включено и есть куда сдвинуться мы передвигаем дощечку влево

\* прибавляем скорость шара к его координатм

\*/

function draw() {

ctx.clearRect(0, 0, canvas.width, canvas.height);

drawBricks();

drawBall();

drawPaddle();

drawScore();

drawLives();

collisionDetection();

if (x + dx > canvas.width - ballRadius || x + dx < ballRadius) {

dx = -dx;

}

if (y + dy < ballRadius) {

dy = -dy;

}

else if (y + dy > canvas.height - ballRadius) {

if (x > paddleX && x < paddleX + paddleWidth) {

dy = -dy;

}

else {

lives--;

if (!lives) {

alert("GAME OVER");

document.location.reload();

}

else {

x = canvas.width / 2;

y = canvas.height - 30;

dx = 3;

dy = -3;

paddleX = (canvas.width - paddleWidth) / 2;

}

}

}

if (rightPressed && paddleX < canvas.width - paddleWidth) {

paddleX += 7;

}

else if (leftPressed && paddleX > 0) {

paddleX -= 7;

}

x += dx;

y += dy;

requestAnimationFrame(draw);

}

draw();