

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| Институт цифровых  интеллектуальных систем | Кафедра  компьютерных систем управления |

Дисциплина «Основы системного программного обеспечения»

# Отчет по лабораторной работе № 3

|  |  |  |
| --- | --- | --- |
| **Выполнили**  **студент гр. АДБ-20-06:** | *(дата) (подпись)* | **Шагов Д.А.** |
| **Проверил к.т.н., доцент** | *(дата) (подпись)* | **Ковалев И.А.** |

**Москва 2023 г.**

**Код индивидуальных заданий:**

**//first question**

**function max(a,b) {**

**if (a > b){**

**console.log('Maximum is: ', a)**

**}**

**else{**

**console.log('Maximum is: ', b)**

**}**

**}**

**max(5,10)**

**//second question**

**let admin**

**let name = 'Denis'**

**admin = name**

**console.log("admin:" + admin)**

**//third question**

**function ask\_age(){**

**let age = prompt('Введите ваш возраст','Введите ваш возраст');**

**if(!confirm("Вам точно - "+age + " лет?")){**

**ask\_age()**

**}**

**}**

**ask\_age()**

**//fourth question**

**text\_field = document.querySelector('input')**

**h1\_text = document.querySelector('h1')**

**function change\_text(){**

**console.log(h1\_text)**

**console.log(text\_field)**

**h1\_text.innerHTML = text\_field.value + " Denis"**

**}**

**button = document.querySelector('button');**

**button.addEventListener("click", change\_text)**

**'use strict';**

**Код для анимации перебора букв**

**var tetrominos = [{**

**// box**

**colors : ['rgb(59,84,165)', 'rgb(118,137,196)', 'rgb(79,111,182)'],**

**data : [[0, 0, 0, 0],**

**[0, 1, 1, 0],**

**[0, 1, 1, 0],**

**[0, 0, 0, 0]]**

**},**

**{**

**// stick**

**colors : ['rgb(214,30,60)', 'rgb(241,108,107)', 'rgb(236,42,75)'],**

**data : [[0, 0, 0, 0],**

**[0, 0, 0, 0],**

**[1, 1, 1, 1],**

**[0, 0, 0, 0]]**

**},**

**{**

**// z**

**colors : ['rgb(88,178,71)', 'rgb(150,204,110)', 'rgb(115,191,68)'],**

**data : [[0, 0, 0, 0],**

**[0, 1, 1, 0],**

**[0, 0, 1, 1],**

**[0, 0, 0, 0]]**

**},**

**{**

**// T**

**colors : ['rgb(62,170,212)', 'rgb(120,205,244)', 'rgb(54,192,240)'],**

**data : [[0, 0, 0, 0],**

**[0, 1, 1, 1],**

**[0, 0, 1, 0],**

**[0, 0, 0, 0]]**

**},**

**{**

**// s**

**colors : ['rgb(236,94,36)', 'rgb(234,154,84)', 'rgb(228,126,37)'],**

**data : [[0, 0, 0, 0],**

**[0, 1, 1, 0],**

**[1, 1, 0, 0],**

**[0, 0, 0, 0]]**

**},**

**{**

**// backwards L**

**colors : ['rgb(220,159,39)', 'rgb(246,197,100)', 'rgb(242,181,42)'],**

**data : [[0, 0, 1, 0],**

**[0, 0, 1, 0],**

**[0, 1, 1, 0],**

**[0, 0, 0, 0]]**

**},**

**{**

**// L**

**colors : ['rgb(158,35,126)', 'rgb(193,111,173)', 'rgb(179,63,151)'],**

**data : [[0, 1, 0, 0],**

**[0, 1, 0, 0],**

**[0, 1, 1, 0],**

**[0, 0, 0, 0]]**

**}];**

**var Tetris = function(x,y,width,height){**

**this.posX = x || 0;**

**this.posY = y || 0;**

**this.width = width || window.innerWidth;**

**this.height = height || window.innerHeight;**

**this.bgCanvas = document.createElement('canvas');**

**this.fgCanvas = document.createElement('canvas');**

**this.bgCanvas.width = this.fgCanvas.width = this.width;**

**this.bgCanvas.height = this.fgCanvas.height = this.height;**

**this.bgCtx = this.bgCanvas.getContext('2d');**

**this.fgCtx = this.fgCanvas.getContext('2d');**

**this.bgCanvas.style.left = this.posX + 'px';**

**this.bgCanvas.style.top = this.posY + 'px';**

**this.fgCanvas.style.left = this.posX + 'px';**

**this.fgCanvas.style.top = this.posY + 'px';**

**document.body.appendChild(this.bgCanvas);**

**document.body.appendChild(this.fgCanvas);**

**this.init();**

**};**

**Tetris.prototype.init = function(){**

**this.curPiece = {**

**data : null,**

**colors : [0,0,0],**

**x : 0,**

**y : 0,**

**};**

**this.lastMove = Date.now();**

**this.curSpeed = 50+Math.random()\*50;**

**this.unitSize = 20;**

**this.linesCleared = 0;**

**this.level = 0;**

**this.loseBlock = 0;**

**// init the board**

**this.board = [];**

**this.boardWidth = Math.floor(this.width / this.unitSize);**

**this.boardHeight = Math.floor(this.height / this.unitSize);**

**var board = this.board,**

**boardWidth = this.boardWidth,**

**boardHeight = this.boardHeight,**

**halfHeight = boardHeight/2,**

**curPiece = this.curPiece,**

**x = 0, y = 0;**

**// init board**

**for (x = 0; x <= boardWidth; x++) {**

**board[x] = [];**

**for (y = 0; y <= boardHeight; y++) {**

**board[x][y] = {**

**data: 0,**

**colors: ['rgb(0,0,0)', 'rgb(0,0,0)', 'rgb(0,0,0)']**

**};**

**if(Math.random() > 0.15 && y > halfHeight){**

**board[x][y] = {**

**data: 1,**

**colors: tetrominos[Math.floor(Math.random() \* tetrominos.length)].colors**

**};**

**}**

**}**

**}**

**// collapse the board a bit**

**for (x = 0; x <= boardWidth; x++) {**

**for (y = boardHeight-1; y > -1; y--) {**

**if(board[x][y].data === 0 && y > 0){**

**for(var yy = y; yy > 0; yy--){**

**if(board[x][yy-1].data){**

**board[x][yy].data = 1;**

**board[x][yy].colors = board[x][yy-1].colors;**

**board[x][yy-1].data = 0;**

**board[x][yy-1].colors = ['rgb(0,0,0)', 'rgb(0,0,0)', 'rgb(0,0,0)'];**

**}**

**}**

**}**

**}**

**}**

**var self = this;**

**window.addEventListener('keydown', function (e) {**

**switch (e.keyCode) {**

**case 37:**

**if (self.checkMovement(curPiece, -1, 0)) {**

**curPiece.x--;**

**}**

**break;**

**case 39:**

**if (self.checkMovement(curPiece, 1, 0)) {**

**curPiece.x++;**

**}**

**break;**

**case 40:**

**if (self.checkMovement(curPiece, 0, 1)) {**

**curPiece.y++;**

**}**

**break;**

**case 32:**

**case 38:**

**curPiece.data = self.rotateTetrimono(curPiece);**

**break;**

**}**

**});**

**// render the board**

**this.checkLines();**

**this.renderBoard();**

**// assign the first tetri**

**this.newTetromino();**

**this.update();**

**};**

**Tetris.prototype.update = function() {**

**var curPiece = this.curPiece;**

**if (!this.checkMovement(curPiece, 0, 1)) {**

**if (curPiece.y < -1) {**

**// you lose**

**this.loseScreen();**

**return true;**

**} else {**

**this.fillBoard(curPiece);**

**this.newTetromino();**

**}**

**} else {**

**if (Date.now() > this.lastMove) {**

**this.lastMove = Date.now() + this.curSpeed;**

**if (this.checkMovement(curPiece, 0, 1)) {**

**curPiece.y++;**

**} else {**

**this.fillBoard(curPiece);**

**this.newTetromino();**

**}**

**}**

**}**

**this.render();**

**var self = this;**

**requestAnimationFrame(function(){self.update();});**

**};**

**// render only the board.**

**Tetris.prototype.renderBoard = function(){**

**var canvas = this.bgCanvas,**

**ctx = this.bgCtx,**

**unitSize = this.unitSize,**

**board = this.board,**

**boardWidth = this.boardWidth,**

**boardHeight = this.boardHeight;**

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

**for (var x = 0; x <= boardWidth; x++) {**

**for (var y = 0; y <= boardHeight; y++) {**

**if (board[x][y].data !== 0) {**

**var bX = (x \* unitSize),**

**bY = (y \* unitSize);**

**ctx.fillStyle = board[x][y].colors[0];**

**ctx.fillRect(bX, bY, unitSize, unitSize);**

**ctx.fillStyle = board[x][y].colors[1];**

**ctx.fillRect(bX+2, bY+2, unitSize-4, unitSize-4);**

**ctx.fillStyle = board[x][y].colors[2];**

**ctx.fillRect(bX+4, bY+4, unitSize-8, unitSize-8);**

**}**

**}**

**}**

**};**

**// Render the current active piece**

**Tetris.prototype.render = function() {**

**var canvas = this.fgCanvas,**

**ctx = this.fgCtx,**

**unitSize = this.unitSize,**

**curPiece = this.curPiece;**

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

**for (var x = 0; x < 4; x++) {**

**for (var y = 0; y < 4; y++) {**

**if (curPiece.data[x][y] === 1) {**

**var xPos = ((curPiece.x + x) \* unitSize),**

**yPos = ((curPiece.y + y) \* unitSize);**

**if (yPos > - 1) {**

**ctx.fillStyle = curPiece.colors[0];**

**ctx.fillRect(xPos, yPos, unitSize, unitSize);**

**ctx.fillStyle = curPiece.colors[1];**

**ctx.fillRect(xPos+2, yPos+2, unitSize-4, unitSize-4);**

**ctx.fillStyle = curPiece.colors[2];**

**ctx.fillRect(xPos+4, yPos+4, unitSize-8, unitSize-8);**

**}**

**}**

**}**

**}**

**};**

**// Make sure we can mov where we want.**

**Tetris.prototype.checkMovement = function(curPiece, newX, newY) {**

**var piece = curPiece.data,**

**posX = curPiece.x,**

**posY = curPiece.y,**

**board = this.board,**

**boardWidth = this.boardWidth,**

**boardHeight = this.boardHeight;**

**for (var x = 0; x < 4; x++) {**

**for (var y = 0; y < 4; y++) {**

**if (piece[x][y] === 1) {**

**if (!board[posX + x + newX]) {**

**board[posX + x + newX] = [];**

**}**

**if (!board[posX + x + newX][y + posY + newY]) {**

**board[posX + x + newX][y + posY + newY] = {**

**data: 0**

**};**

**}**

**if (posX + x + newX >= boardWidth || posX + x + newX < 0 || board[posX + x + newX][y + posY + newY].data == 1) {**

**return false;**

**}**

**if (posY + y + newY > boardHeight) {**

**return false;**

**}**

**}**

**}**

**}**

**return true;**

**};**

**// checks for completed lines and clears them**

**Tetris.prototype.checkLines = function() {**

**var board = this.board,**

**boardWidth = this.boardWidth,**

**boardHeight = this.boardHeight,**

**linesCleared = this.linesCleared,**

**level = this.level,**

**y = boardHeight+1;**

**while (y--) {**

**var x = boardWidth,**

**lines = 0;**

**while (x--) {**

**if (board[x][y].data === 1) {**

**lines++;**

**}**

**}**

**if (lines === boardWidth) {**

**linesCleared++;**

**level = Math.round(linesCleared / 20) \* 20;**

**var lineY = y;**

**while (lineY) {**

**for (x = 0; x <= boardWidth; x++) {**

**if (lineY - 1 > 0) {**

**board[x][lineY].data = board[x][lineY - 1].data;**

**board[x][lineY].colors = board[x][lineY - 1].colors;**

**}**

**}**

**lineY--;**

**}**

**y++;**

**}**

**}**

**};**

**// Lose animation**

**Tetris.prototype.loseScreen = function() {**

**var ctx = this.bgCtx,**

**unitSize = this.unitSize,**

**boardWidth = this.boardWidth,**

**boardHeight = this.boardHeight,**

**y = boardHeight - this.loseBlock;**

**for(var x = 0; x < boardWidth; x++){**

**var bX = (x \* unitSize),**

**bY = (y \* unitSize);**

**ctx.fillStyle = 'rgb(80,80,80)';**

**ctx.fillRect(bX, bY, unitSize, unitSize);**

**ctx.fillStyle = 'rgb(150,150,150)';**

**ctx.fillRect(bX+2, bY+2, unitSize-4, unitSize-4);**

**ctx.fillStyle = 'rgb(100,100,100)';**

**ctx.fillRect(bX+4, bY+4, unitSize-8, unitSize-8);**

**}**

**if(this.loseBlock <= (boardHeight+1)){**

**this.loseBlock++;**

**var self = this;**

**requestAnimationFrame(function(){self.loseScreen();});**

**}else{**

**this.init();**

**}**

**};**

**// adds the piece as part of the board**

**Tetris.prototype.fillBoard = function(curPiece) {**

**var piece = curPiece.data,**

**posX = curPiece.x,**

**posY = curPiece.y,**

**board = this.board;**

**for (var x = 0; x < 4; x++) {**

**for (var y = 0; y < 4; y++) {**

**if (piece[x][y] === 1) {**

**board[x + posX][y + posY].data = 1;**

**board[x + posX][y + posY].colors = curPiece.colors;**

**}**

**}**

**}**

**this.checkLines();**

**this.renderBoard();**

**};**

**// rotate a piece**

**Tetris.prototype.rotateTetrimono = function(curPiece) {**

**var rotated = [];**

**for (var x = 0; x < 4; x++) {**

**rotated[x] = [];**

**for (var y = 0; y < 4; y++) {**

**rotated[x][y] = curPiece.data[3 - y][x];**

**}**

**}**

**if (!this.checkMovement({**

**data: rotated,**

**x: curPiece.x,**

**y: curPiece.y**

**}, 0, 0)) {**

**rotated = curPiece.data;**

**}**

**return rotated;**

**};**

**// assign the player a new peice**

**Tetris.prototype.newTetromino = function() {**

**var pieceNum = Math.floor(Math.random() \* tetrominos.length),**

**curPiece = this.curPiece;**

**curPiece.data = tetrominos[pieceNum].data;**

**curPiece.colors = tetrominos[pieceNum].colors;**

**curPiece.x = Math.floor(Math.random()\*(this.boardWidth-curPiece.data.length+1));**

**curPiece.y = -4;**

**};**

**var width = window.innerWidth,**

**boardDiv = 20\*Math.round(window.innerWidth/20),**

**boards = 8,**

**bWidth = boardDiv/boards,**

**tetrisInstances = [];**

**for(var w = 0; w < boards; w++){**

**tetrisInstances.push(new Tetris(20 \* Math.round((w\*bWidth)/20), 0, bWidth));**

**}**

**Вывод:** В ходе работы сделал страницу на github.io