МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СХІДНОУКРАЇНСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

імені ВОЛОДИМИРА ДАЛЯ

Звіт

з лабораторної роботи № 10

з теми: «Отображение фракталов с использованием средств асинхронных вычислений языка JavaScript»

Виконав студент:

Група КН-18д

Скороход С.Г.

Дата 30.04.2020

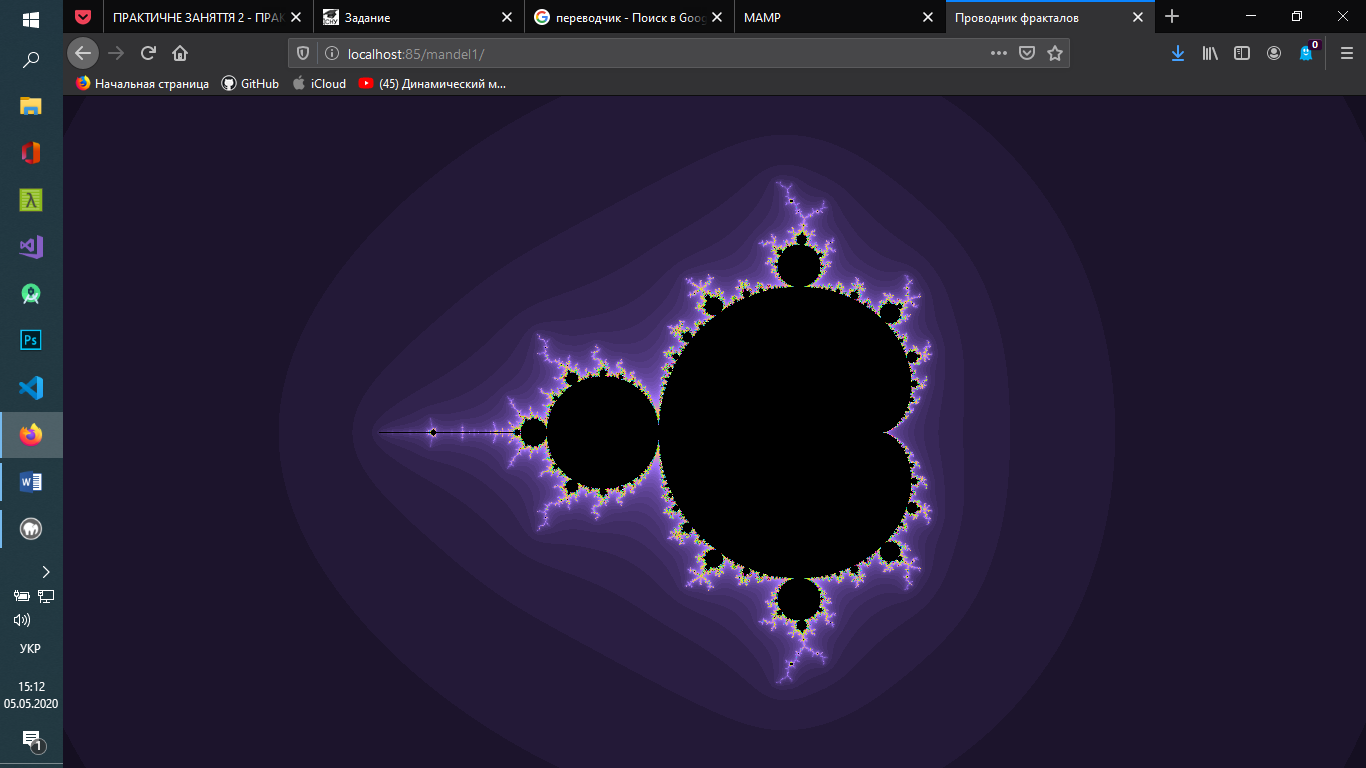
Викладач -

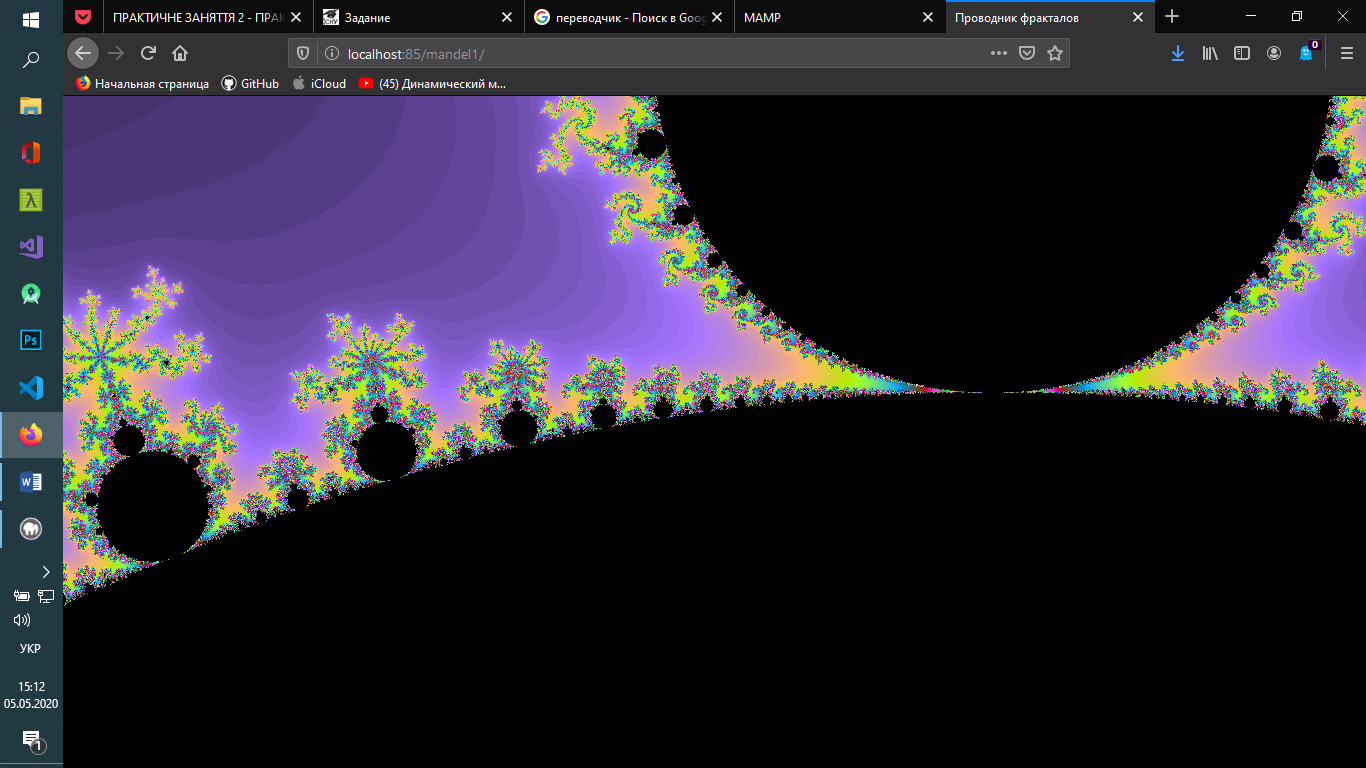
Деркач М. В.

Варіант №8

**м. Северодонецьк**

2020

****

****

**Fractal.css**

html, body, canvas {

    width: 100%;

    height: 100%;

    margin: 0px;

    padding: 0px;

}

canvas {

    display: block;

}

**Index.html**

<!doctype html>

<html lang="en">

<head>

<title>Проводник фракталов</title>

<meta charset="utf-8">

<link rel="stylesheet" href="fractal.css">

<script src="mandeli.js"></script>

<script src="mandel.js"></script>

</head>

<body>

<canvas id="fractal" width="800" height="600"></canvas>

</body>

</html>

**Mandel.js**

var numberOfWorkers = 8;

var workers = [];

var rowData;

var nextRow = 0;

var generation = 0;

window.onload = init;

function init() {

    setupGraphics();

    canvas.onclick = function(event) {

        handleClick(event.clientX, event.clientY);

    };

    window.onresize = function() {

        resizeToWindow();

    };

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

        var worker = new Worker("worker.js");

        worker.onmessage = function(event) {

            processWork(event.target, event.data)

        }

        worker.idle = true;

        workers.push(worker);

    }

    startWorkers();

}

function startWorkers() {

    generation++;

    nextRow = 0;

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

        var worker = workers[i];

        if (worker.idle) {

            var task = createTask(nextRow);

            worker.idle = false;

            worker.postMessage(task);

            nextRow++;

        }

    }

}

function processWork(worker, workerResults) {

    if (workerResults.generation == generation) {

        drawRow(workerResults);

    }

    reassignWorker(worker);

}

function reassignWorker(worker) {

    var row = nextRow++;

    if (row >= canvas.height) {

        worker.idle = true;

    } else {

        var task = createTask(row);

        worker.idle = false;

        worker.postMessage(task);

    }

}

function handleClick(x, y) {

    var width = r\_max - r\_min;

    var height = i\_min - i\_max;

    var click\_r = r\_min + ((width \* x) / canvas.width);

    var click\_i = i\_max + ((height \* y) / canvas.height);

    var zoom = 8;

    r\_min = click\_r - width/zoom;

    r\_max = click\_r + width/zoom;

    i\_max = click\_i - height/zoom;

    i\_min = click\_i + height/zoom;

    startWorkers();

}

function resizeToWindow() {

    canvas.width = window.innerWidth;

    canvas.height = window.innerHeight;

    var width = ((i\_max - i\_min) \* canvas.width / canvas.height);

    var r\_mid = (r\_max + r\_min) / 2;

    r\_min = r\_mid - width/2;

    r\_max = r\_mid + width/2;

    rowData = ctx.createImageData(canvas.width, 1);

    startWorkers();

}

**Mandeli.js**

var canvas;

var ctx;

var i\_max = 1.5;

var i\_min = -1.5;

var r\_min = -2.5;

var r\_max = 1.5;

var max\_iter = 1024;

var escape = 100;

var palette = [];

function createTask(row) {

    var task = {

        row: row,

        width: rowData.width,

        generation: generation,

        r\_min: r\_min,

        r\_max: r\_max,

        i: i\_max + (i\_min - i\_max) \* row / canvas.height,

        max\_iter: max\_iter,

        escape: escape

    };

    return task;

}

function makePalette() {

    function wrap(x) {

        x = ((x + 256) & 0x1ff) - 256;

        if (x < 0) x = -x;

        return x;

    }

    for (i = 0; i <= this.max\_iter; i++) {

        palette.push([wrap(7\*i), wrap(5\*i), wrap(11\*i)]);

    }

}

function drawRow(workerResults) {

    var values = workerResults.values;

    var pixelData = rowData.data;

    for (var i = 0; i < rowData.width; i++) {

        var red = i \* 4;

        var green = i \* 4 + 1;

        var blue = i \* 4 + 2;

        var alpha = i \* 4 + 3;

        pixelData[alpha] = 255;

        if (values[i] < 0) {

            pixelData[red] = pixelData[green] = pixelData[blue] = 0;

        } else {

            var color = this.palette[values[i]];

            pixelData[red] = color[0];

            pixelData[green] = color[1];

            pixelData[blue] = color[2];

        }

    }

    ctx.putImageData(this.rowData, 0, workerResults.row);

}

function setupGraphics() {

    canvas = document.getElementById("fractal");

    ctx = canvas.getContext("2d");

    canvas.width = window.innerWidth;

    canvas.height = window.innerHeight;

    var width = ((i\_max - i\_min) \* canvas.width / canvas.height);

    var r\_mid = (r\_max + r\_min) / 2;

    r\_min = r\_mid - width/2;

    r\_max = r\_mid + width/2;

    rowData = ctx.createImageData(canvas.width, 1);

    makePalette();

}

**Worker.js**

function computeRow(task) {

    var iter = 0;

    var c\_i = task.i;

    var max\_iter = task.max\_iter;

    var escape = task.escape \* task.escape;

    task.values = [];

    for (var i = 0; i < task.width; i++) {

        var c\_r = task.r\_min + (task.r\_max - task.r\_min) \* i / task.width;

        var z\_r = 0, z\_i = 0;

        for (iter = 0; z\_r\*z\_r + z\_i\*z\_i < escape && iter < max\_iter; iter++) {

            // z -> z^2 + c

            var tmp = z\_r\*z\_r - z\_i\*z\_i + c\_r;

            z\_i = 2 \* z\_r \* z\_i + c\_i;

            z\_r = tmp;

        }

        if (iter == max\_iter) {

            iter = -1;

        }

        task.values.push(iter);

    }

    return task;

}

onmessage = function (task) {

    var workerResult = computeRow(task.data);

    postMessage(workerResult);

}

**Висновок:** у ході виконання роботи було створено веб-додаток для відображення фракталу з використанням методів асинхронних обчислень