1. Исходные данные:

Для отрисовки графика треугольника я использовала HTML элемент - <canvas> , поэтому получились следующие входные данные:

Координаты вершин треугольника: (200; 0), (0; 400), (400; 200)

RGB-представление выбранных цветов: (0; 255; 137), (127; 0; 255), (254; 204; 2)

1. Формулу аффинного преобразования плоскости в трехмерное пространство цветовой схемы RGB в матричном виде:



*Основные вычисления:*















*Промежуточные вычисления:*







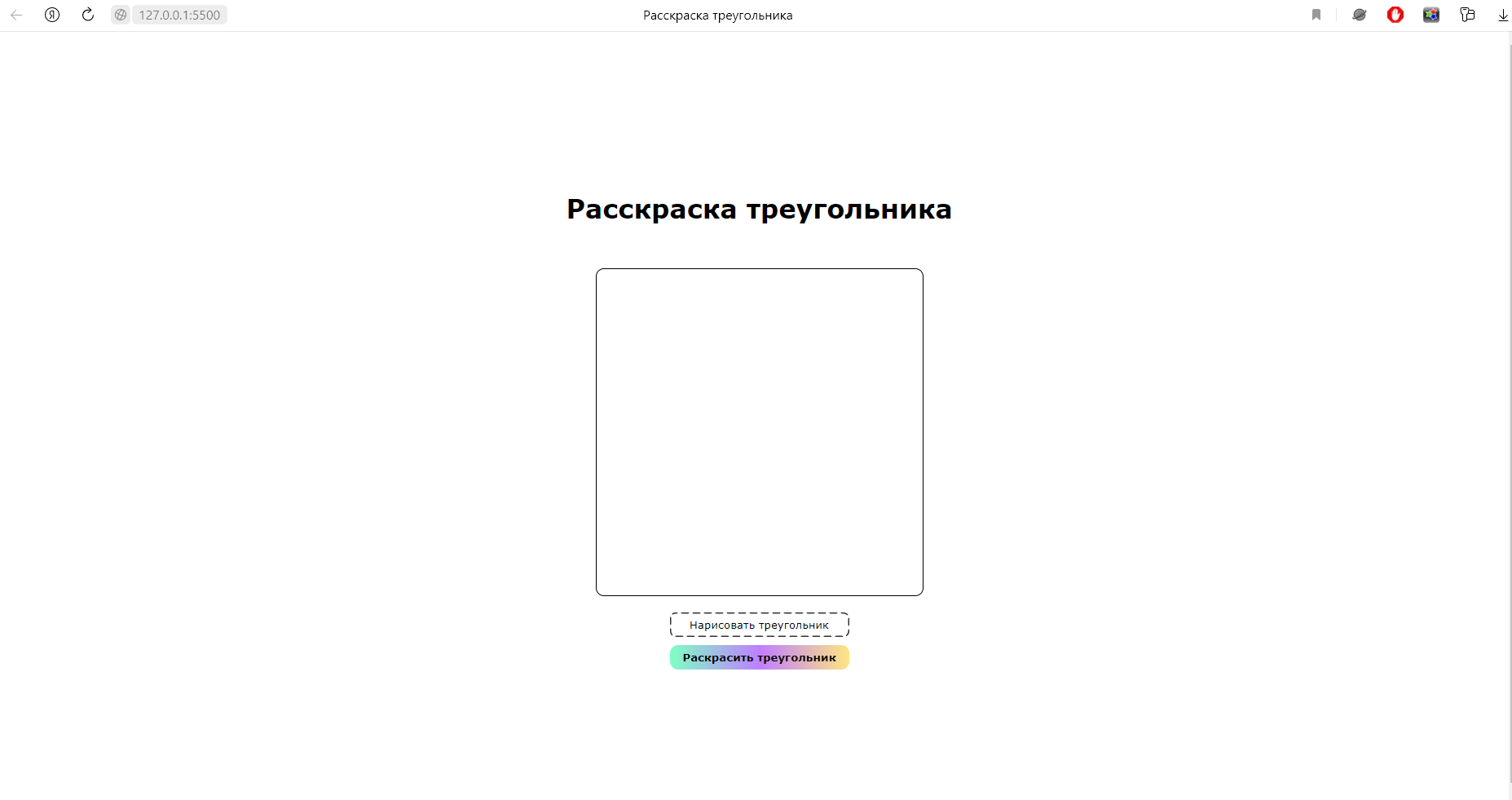






1. Формулу аффинного преобразования плоскости в трехмерное пространство цветовой схемы RGB в матричном виде:

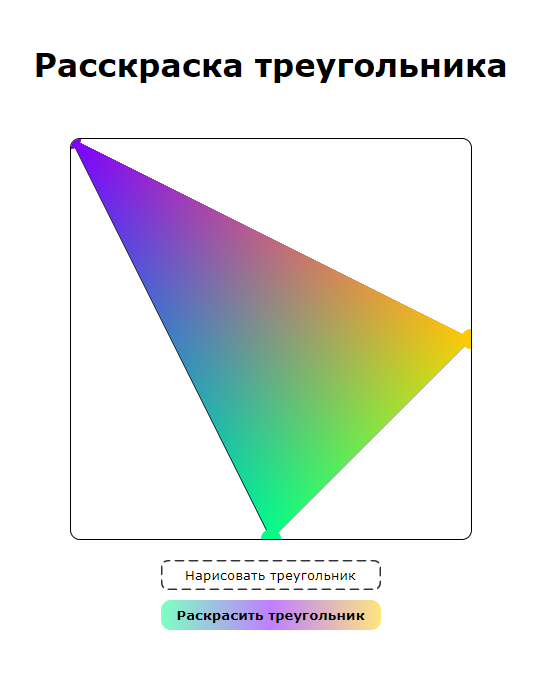
Мое ИД представляет собой веб-приложение, если запустить код, то увидим:



Нажав на кнопку “Нарисовать треугольник”, в окошке нарисуется треугольник с заданными в п.1. координатами вершин, которые помечены соответствующими цветами из п.1.



Нажав на кнопку “Раскрасить треугольник”, запуститься ключевой алгоритм, ради которого мы здесь сегодня собрались в этом файлике. После этого увидим следующее:



***Код из файла main.html :***

<!DOCTYPE html>

<html>

<head>

    <title>Расскраска треугольника</title>

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

</head>

<body>

    <h1>Расскраска треугольника</h1>

    <div class="block">

        <canvas id="myCanvas" width="400" height="400"></canvas>

        <button id="draw" style="margin-top: 20px;" onclick="drawTriangle()">Нарисовать треугольник</button>

        <button id="color" style="margin-top: 10px;" onclick="colorTriangle()">Раскрасить треугольник</button>

    </div>

    <script src="main.js"></script>

</body>

</html>

***Код из файла main.css :***

html{

    width: 100%;

    height: 100%;

}

body{

    width: 100%;

    height: 100%;

    display: flex;

    align-items: center;

    justify-content: center;

    flex-direction: column;

}

canvas {

    border: 1px solid black;

    transform: scaleY(-1);

    border-radius: 10px;

}

h1{

    font-family: Verdana, Geneva, Tahoma, sans-serif;

    margin-bottom: 50px;

}

.block{

    display: flex;

    flex-direction: column;

    width: 400px;

    height: 500px;

    align-items: center;

    justify-content: center;

}

button{

    height: 30px;

    width: 220px;

    border-radius: 10px;

    border: 0px solid;

    font-family: Verdana, Geneva, Tahoma, sans-serif;

}

button:hover{

    transition: 0.2s;

    -webkit-box-shadow: 0px 0px 10px 2px rgba(34, 60, 80, 0.2);

    -moz-box-shadow: 0px 0px 10px 2px rgba(34, 60, 80, 0.2);

    box-shadow: 0px 0px 10px 2px rgba(34, 60, 80, 0.2);

}

button:active{

    transition: 2s;

    -webkit-box-shadow: 0px 0px 10px 2px rgba(34, 60, 80, 0.2) inset;

    -moz-box-shadow: 0px 0px 10px 2px rgba(34, 60, 80, 0.2) inset;

    box-shadow: 0px 0px 10px 2px rgba(34, 60, 80, 0.2) inset;

}

#draw{

    background-image: url("data:image/svg+xml,%3csvg width='100%25' height='100%25' xmlns='http://www.w3.org/2000/svg'%3e%3crect width='100%25' height='100%25' fill='none' rx='7' ry='7' stroke='%23333' stroke-width='3' stroke-dasharray='7' stroke-dashoffset='6' stroke-linecap='round'/%3e%3c/svg%3e");

    border-radius: 7px;

    background-color: white;

}

#color{

    background: linear-gradient(to right,rgb(0, 255, 137,0.5), rgb(127, 0, 255,0.5), rgb(254, 204, 2,0.5));

    font-weight: 600;

}

***Код из файла main.js :***

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

var ctx = canvas.getContext('2d');

function drawTriangle() {

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

    ctx.beginPath();

    ctx.moveTo(200, 0);

    ctx.lineTo(0, 400);

    ctx.lineTo(400, 200);

    ctx.closePath();

    ctx.stroke();

    drawCircle(200, 0, 10, "(0, 255, 137)"); // Вершина треугольника (200, 0)

    drawCircle(0, 400, 10, '(127, 0, 255)'); // Вершина треугольника (0, 400)

    drawCircle(400, 200, 10, '(254, 204, 2)'); // Вершина треугольника (400, 200)

}

function drawCircle(x, y, radius, color) {

    ctx.beginPath();

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

    ctx.fillStyle = "rgb" + `${color}`;

    ctx.fill();

    ctx.closePath();

}

function colorTriangle() {

    var imageData = ctx.getImageData(0, 0, canvas.width, canvas.height);

    var data = imageData.data;

    for (var y = 0; y < canvas.height; y++) {

        for (var x = 0; x < canvas.width; x++) {

            if (isPointInsideTriangle(x, y)) {

                var xK = { x: 0.635, y: 0.255, z: -388 / 600 };

                var yK = { x: -0.635, y: 0.51, z: 17 / 600 };

                var K = { x: 127, y: -204, z: -799 / 3 };

                var xResultVec = { x: xK.x \* x, y: xK.y \* x, z: xK.z \* x };

                var yResultVec = { x: yK.x \* y, y: yK.y \* y, z: yK.z \* y };

                var resultVec = { x: xResultVec.x - yResultVec.x - K.x, y: xResultVec.y - yResultVec.y - K.y, z: xResultVec.z - yResultVec.z - K.z };

                var pixelIndex = (y \* canvas.width + x) \* 4;

                data[pixelIndex] = resultVec.x;

                data[pixelIndex + 1] = resultVec.y;

                data[pixelIndex + 2] = resultVec.z;

                data[pixelIndex + 3] = 255;

            }

        }

    }

    ctx.putImageData(imageData, 0, 0);

}

function isPointInsideTriangle(x, y) {

    var p0 = { x: 200, y: 0 };

    var p1 = { x: 0, y: 400 };

    var p2 = { x: 400, y: 200 };

    var dX = x - p2.x;

    var dY = y - p2.y;

    var dX21 = p2.x - p1.x;

    var dY12 = p1.y - p2.y;

    var D = dY12 \* (p0.x - p2.x) + dX21 \* (p0.y - p2.y);

    var s = dY12 \* dX + dX21 \* dY;

    var t = (p2.y - p0.y) \* dX + (p0.x - p2.x) \* dY;

    if (D < 0) {

        return s <= 0 && t <= 0 && s + t >= D;

    } else {

        return s >= 0 && t >= 0 && s + t <= D;

    }

}