Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра информационных систем и программной инженерии

Лабораторная работа № 6

по дисциплине "Программирование графических приложений"

ТЕМА РАБОТЫ:

Обработка растровых изображений

Выполнил:

студент гр. ПРИм-124

Парахин К.В.

Принял:

Жигалов И.Е.

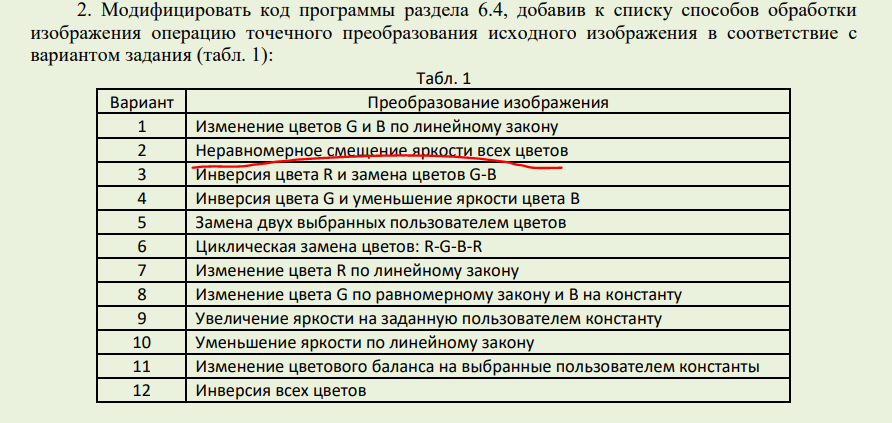
Владимир 2024 г.

Цель работы:

Изучение способов обработки растровых изображений при формировании моделей графических объектов с использованием WebGL.

Выполнение работы:

Индивидуальный вариант 2 (14 % 12 = 2)



Добавляем матрицу для фильтра ColorsOffset

        ColorsOffset: {

            weight: 1.0,

            kernel: [

                -0.5, 1.5, -0.5,

                0.5, 5, -2

                -0.5, 0.5, -0.5

            ]

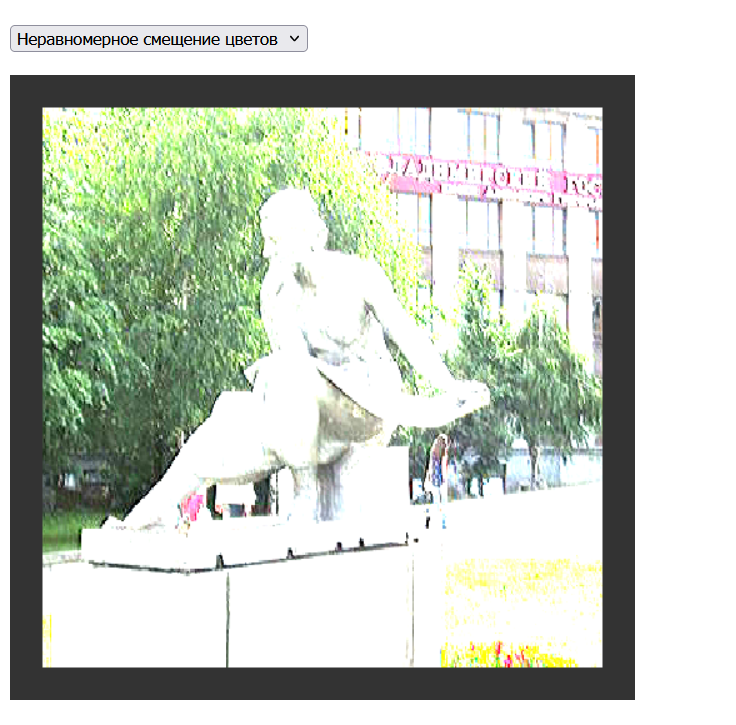
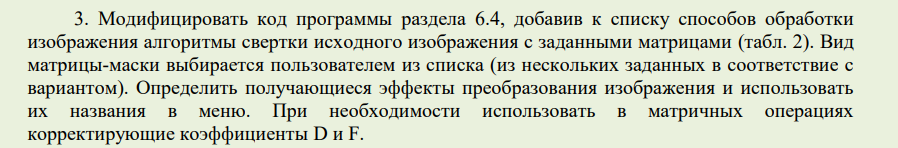
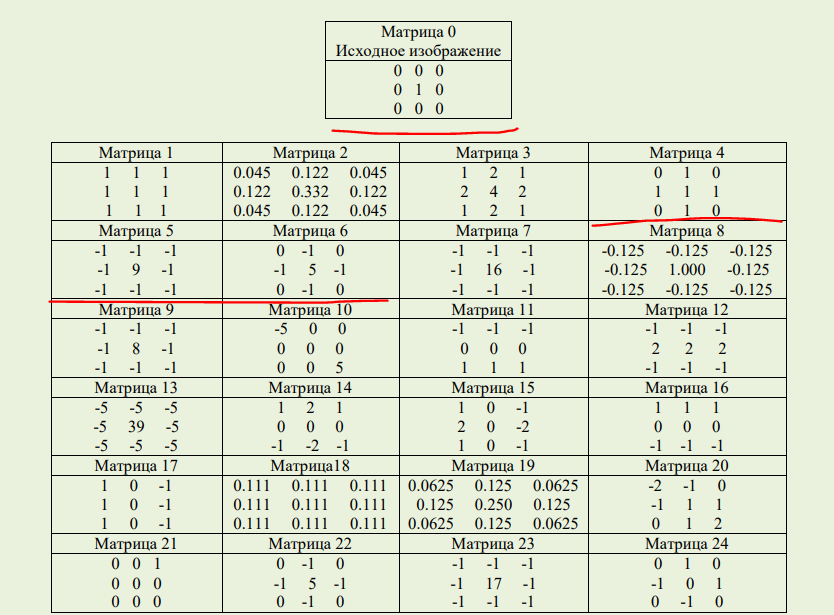


Рисунок 1. Неравномерное распределение яркости





    var filters = {

        Original: {

            weight: 1.0,

            kernel: [

                0, 0, 0,

                0, 1, 0,

                0, 0, 0

            ]

        },

        Filter1: {

            weight: 3.0,

            kernel: [

                0, 1, 0,

                1, 1, 1,

                0, 1, 0

            ]

        },

        Filter2: {

            weight: 1.0,

            kernel: [

                -1, -1, -1,

                -1, 9, -1,

                -1, -1, -1

            ]

        },

        Filter3: {

            weight: 1.0,

            kernel: [

                0, -1, 0,

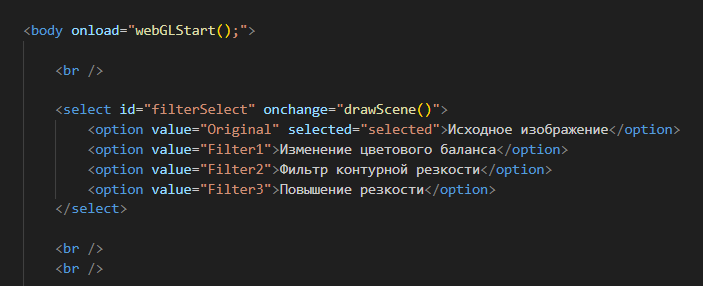
                -1, 5, -1,

                0, -1, 0

            ]

        }

    };



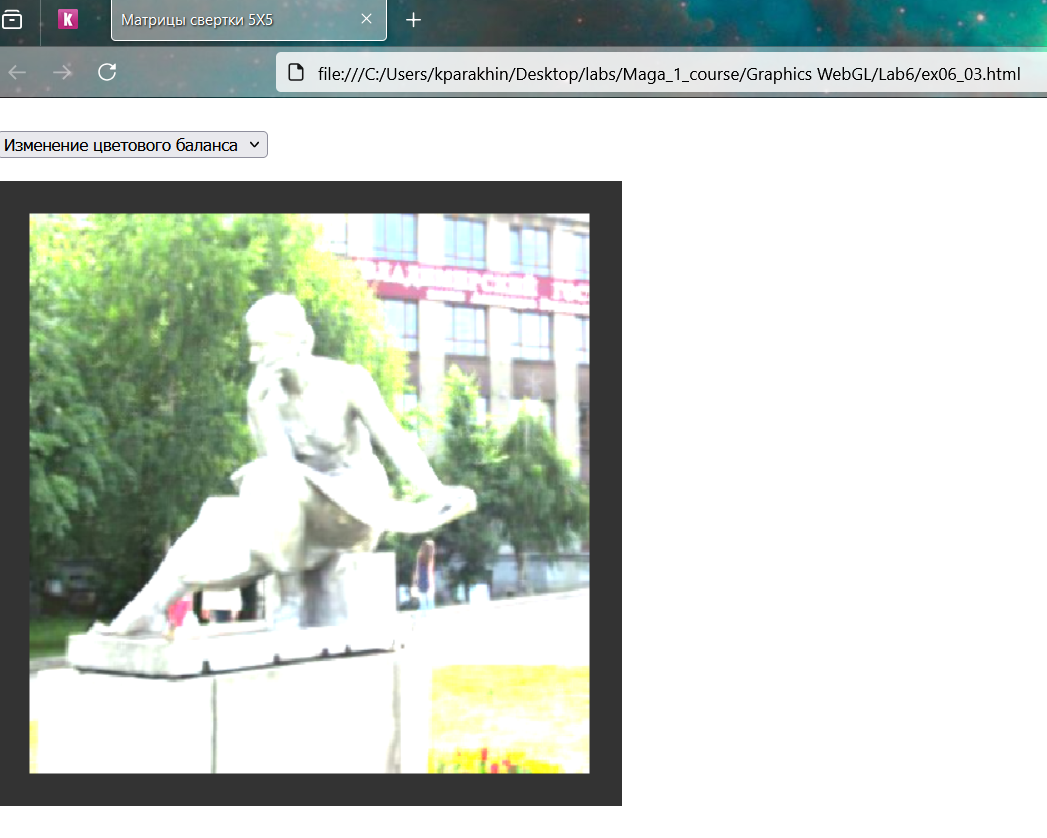


Рисунок 2.1. Фильтр 1 – изменение цветового баланса



Рисунок 2.2. Фильтр 2 – увеличение контурной резкости

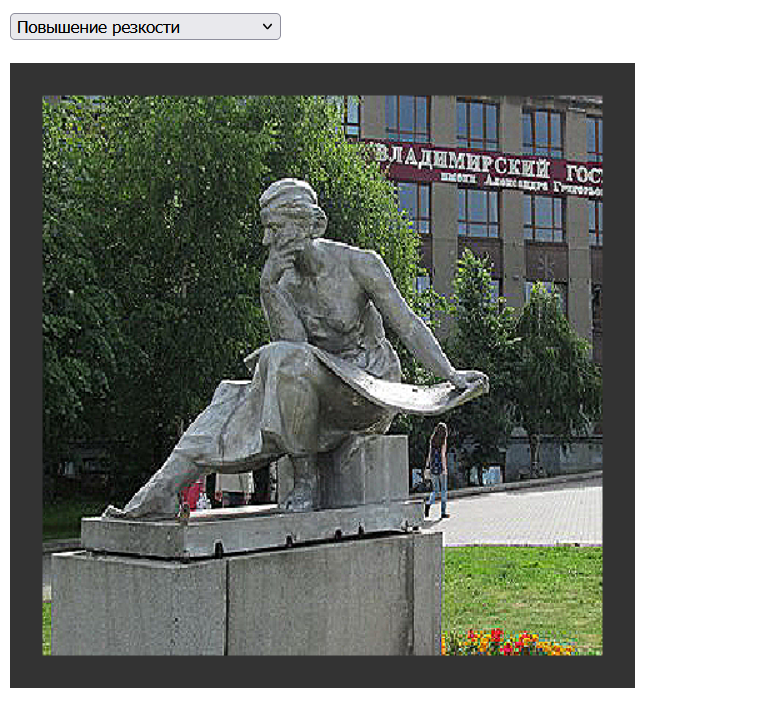
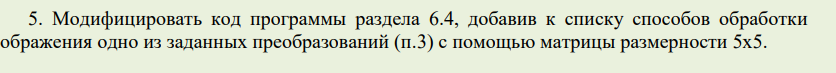


Рисунок 2.3 – простое увеличение резкости (с малым корректирующим коэффициентом)



Обновленные матрицы свертки размерностью 5x5 (оригинальная и одна из прошлого пункта – матрица изменения цветового баланса)

    var filters = {

        Original: {

            weight: 1.0,

            kernel: [

                0, 0, 0, 0, 0,

                0, 0, 0, 0, 0,

                0, 0, 1, 0, 0,

                0, 0, 0, 0, 0,

                0, 0, 0, 0, 0

            ]

        },

        Filter1: {

            weight: 3.0,

            kernel: [

                0, 0, 1, 0, 0,

                0, 0, 1, 0, 0,

                1, 1, 1, 1, 1,

                0, 0, 1, 0, 0,

                0, 0, 1, 0, 0

            ]

        }

    };

Сам скрипт тоже нужно адаптировать под использование матрицц свертки 5x5:

<script id="shader-fs" type="x-shader/x-fragment">

    precision mediump float;

    varying vec2 vTextureCoord;

    uniform vec2 uTextureSize;

    uniform float uKernel[25];

    uniform float uKernelWeight;

    uniform sampler2D uSampler;

    void main(void) {

        vec2 pixelSize = vec2(1.0, 1.0) / uTextureSize;

        vec4 colorSum =

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2(-2, -2)) \* uKernel[0] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2(-1, -2)) \* uKernel[1] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2( 0, -2)) \* uKernel[2] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2( 1, -2)) \* uKernel[3] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2( 2, -2)) \* uKernel[4] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2(-2, -1)) \* uKernel[5] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2(-1, -1)) \* uKernel[6] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2( 0, -1)) \* uKernel[7] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2( 1, -1)) \* uKernel[8] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2( 2, -1)) \* uKernel[9] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2(-2,  0)) \* uKernel[10] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2(-1,  0)) \* uKernel[11] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2( 0,  0)) \* uKernel[12] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2( 1,  0)) \* uKernel[13] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2( 2,  0)) \* uKernel[14] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2(-2,  1)) \* uKernel[15] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2(-1,  1)) \* uKernel[16] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2( 0,  1)) \* uKernel[17];

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2( 1,  1)) \* uKernel[18] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2( 2,  1)) \* uKernel[19] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2(-2,  2)) \* uKernel[20] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2(-1,  2)) \* uKernel[21] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2( 0,  2)) \* uKernel[22] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2( 1,  2)) \* uKernel[23] +

            texture2D(uSampler, vTextureCoord + pixelSize \* vec2( 2,  2)) \* uKernel[24];

        gl\_FragColor = vec4((colorSum / uKernelWeight).rgb, 1.0);

    }

</script>

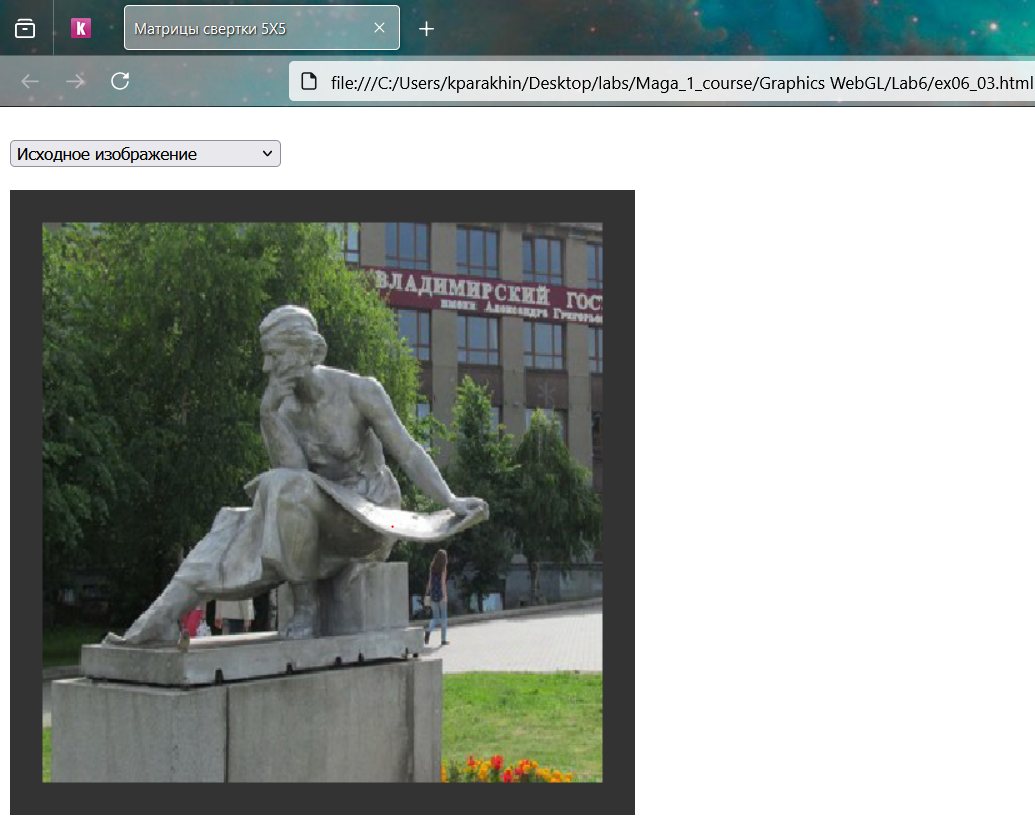


Рисунок 3.1. Использование оригинальной матрицы 5x5

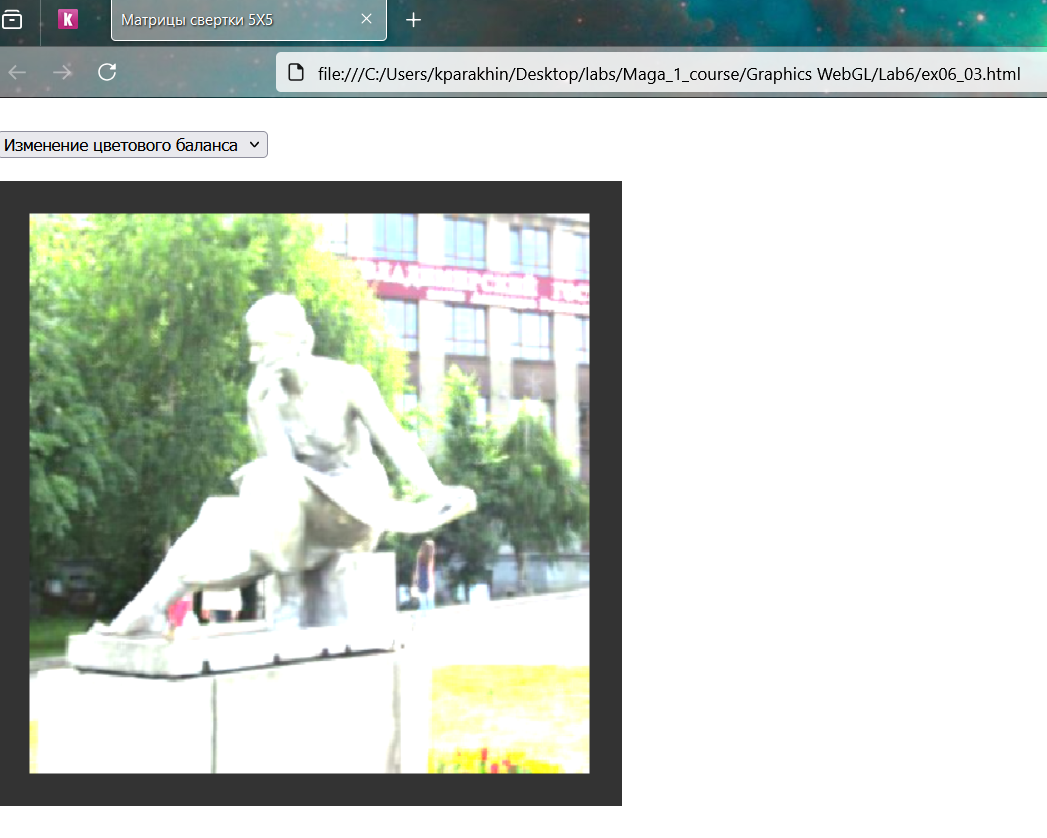


Рисунок 3.2. Фильтр 1 – изменение цветового баланса (c матрице 5x5)

Вывод

В результате выполнения работы я провел изучение способов обработки растровых изображений при формировании моделей графических объектов с использованием WebGL.