Министерство науки и высшего образования Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение**

**высшего 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»**

**ОБРАБОТКА РАСТРОВЫХ ИЗОБРАЖЕНИЙ**

**Выполнил**:

ст. гр. ПРИм-124

Д. А. Грачев

**Принял**:

Жигалов И. Е.

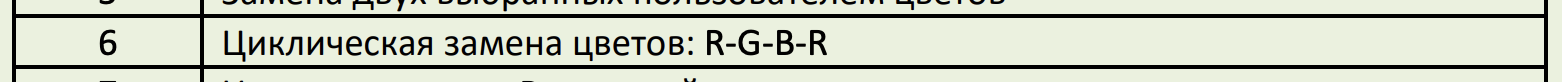
Владимир, 2024

ЦЕЛЬ РАБОТЫ

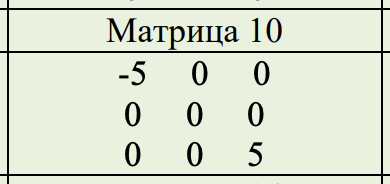
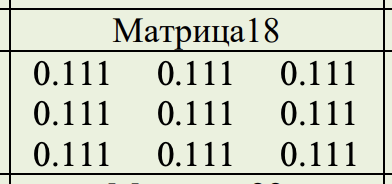
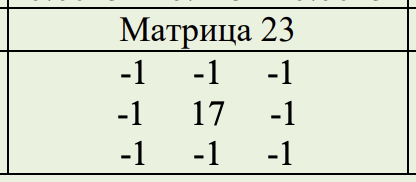
Изучение способов обработки растровых изображений при формировании моделей графических объектов с использованием WebGL.

ЗАДАНИЕ

1. Модифицировать код программы раздела 6.4, добавив к списку способов обработки изображения операцию точечного преобразования исходного изображения в соответствие с вариантом задания



1. Модифицировать код программы раздела 6.4, добавив к списку способов обработки изображения алгоритмы свертки исходного изображения с заданными матрицами. Вид матрицы-маски выбирается пользователем из списка (из нескольких заданных в соответствие с вариантом). Определить получающиеся эффекты преобразования изображения и использовать их названия в меню. При необходимости использовать в матричных операциях корректирующие коэффициенты D и F.

1. Модифицировать код программы раздела 6.4, добавив к списку способов обработки изображения алгоритм последовательного преобразования изображений с помощью двух матриц из заданных в соответствие с вариантом (п.3)
2. Модифицировать код программы раздела 6.4, добавив к списку способов обработки изображения одно из заданных преобразований (п.3) с помощью матрицы размерности 5х5.

ВЫПОЛНЕНИЕ РАБОТЫ

**Задание 1**

Добавим новую операцию в объект filters

CyclicColorShift: {

weight: 1.0,

kernel: [0, 0, 0, 0, 1, 0, 0, 0, 0],

},

Изменим функцию main в шейдере:

uniform bool uCyclicColorShift;

void main(void) {

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

vec4 colorSum =

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

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

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

texture2D(uSampler, vTextureCoord + pixelSize \* vec2(-1, 0)) \* uKernel[3] +

texture2D(uSampler, vTextureCoord + pixelSize \* vec2( 0, 0)) \* uKernel[4] +

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

vec4 originalColor = texture2D(uSampler, vTextureCoord);

if (uCyclicColorShift) { // Используем флаг для CyclicColorShift

gl\_FragColor = vec4(originalColor.g, originalColor.b, originalColor.r, 1.0); // R->G->B->R

} else {

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

}

}

В функции drawScene() добавьте установку значения uCyclicColorShift перед отрисовкой

var cyclicColorShiftLocation = gl.getUniformLocation(

shaderProgram,

"uCyclicColorShift"

);

gl.uniform1i(

cyclicColorShiftLocation,

selectedFilter === "CyclicColorShift" ? 1 : 0

);

Добавим новую опцию

<option value="CyclicColorShift">Циклическая замена цветов</option>

**Задание 2**

Добавим новые фильтры для каждой матрицы:

Matrix10: {

weight: 1.0,

kernel: [-5, 0, 0, 0, 0, 0, 0, 0, 5],

},

Matrix18: {

weight: 1.0,

kernel: [

0.111, 0.111, 0.111, 0.111, 0.111, 0.111, 0.111, 0.111, 0.111,

],

},

Matrix23: {

weight: 1.0,

kernel: [-1, -1, -1, -1, 17, -1, -1, -1, -1],

},

Добавим новые опции для возможности выбора матриц

<option value="Matrix10">Matrix 10</option>

<option value="Matrix18">Matrix 18</option>

<option value="Matrix23">Matrix 23</option>

**Задание 3**

В drawScene() добавьте обработку второго фильтра

var secondFilter = filters["Matrix18"];

var secondKernelLocation = gl.getUniformLocation(

shaderProgram,

"uKernel[1]"

);

gl.uniform1fv(secondKernelLocation, secondFilter.kernel);

**Задание 4**

Добавим новую матрицу 5х5:

LargeMatrix: {

weight: 1.0,

kernel: [

1, 1, -5, 0, 1,

1, 1, 0, 0, 0,

1, 1, 0, 0, 5,

1, 1, 1, 1, 1,

1, 1, 1, 1, 1,

],

},

Добавим опцию для выбора матрицы:

<option value="LargeMatrix">Матрица 5х5</option>

Скриншоты работы программы:

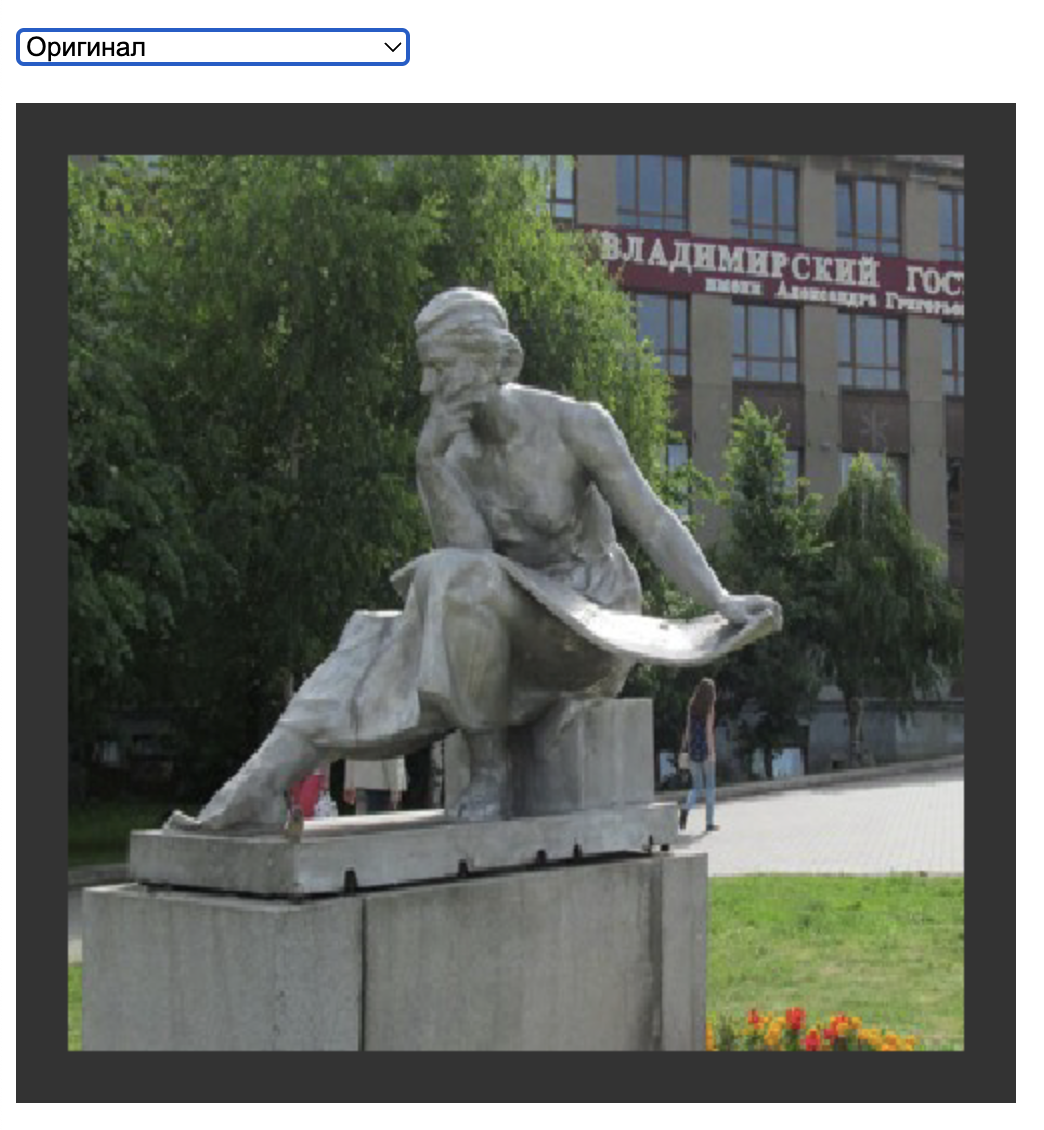


Рисунок 1. Оригинальное изображение

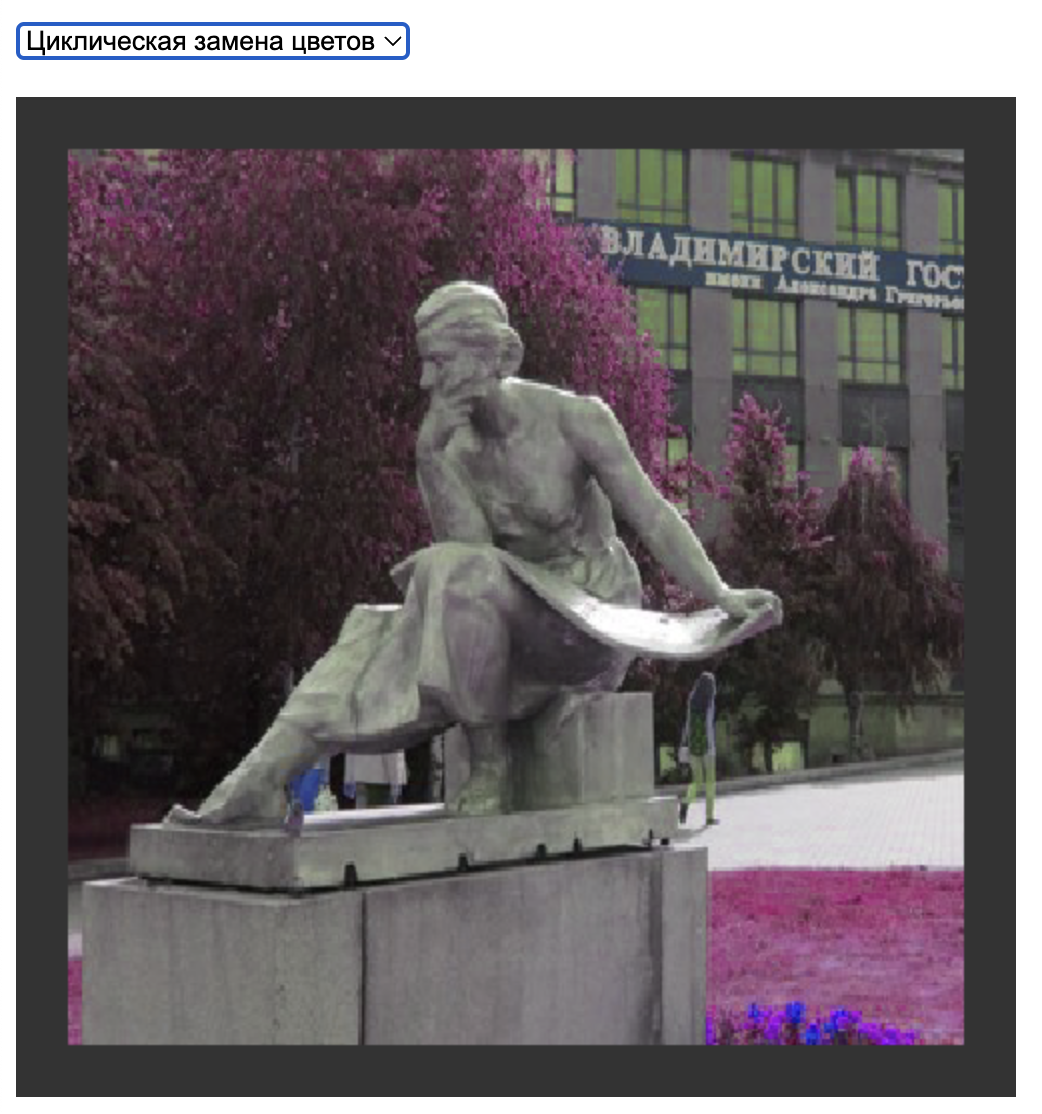


Рисунок 2. Изображение с циклической заменой цвета

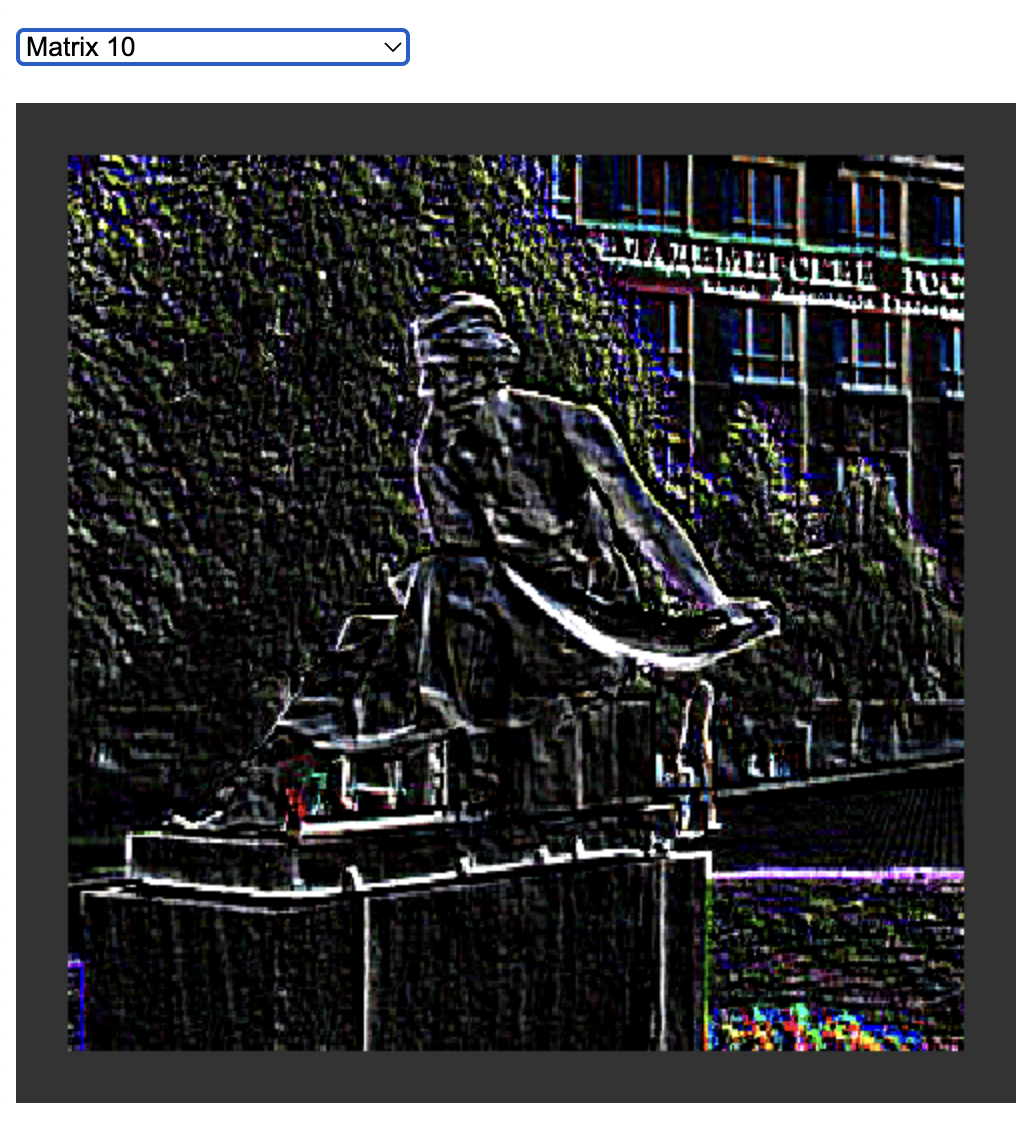


Рисунок 3. Изображение по матрице 10



Рисунок 4. Изображение по матрице 18



Рисунок 5. Изображение по матрице 23



Рисунок 6. Изображение по большой матрице

ВЫВОД

В ходе выполнения работы были изучены способы обработки растровых изображений при формировании моделей графических объектов с использованием WebGL.