Липецкий государственный технический университет

Кафедра АСУ

Лабораторная работа №1

по организации графических систем и систем мультимедиа

Обработка изображений

Студент Бровкин А.Н.

Группа М-АС-19

Руководитель Кургасов В.В.

Липецк

2020

**Задание кафедры**

Реализовать программный продукт, в котором есть возможность применения не менее 3-х фильтров к загружаемому изображению и экспорт результата в виде измененного изображения. Необходимо наличие графического меню пользователя и при обработке изображений должен использоваться CPU.

**Цель работы**

Освоить на практике преобразование загружаемых графических файлов.

# Теоретические сведения

Изначально JavaScript был создан, чтобы «сделать веб-страницы живыми». Программы на этом языке называются скриптами. Они могут встраиваться в HTML и выполняться автоматически при загрузке веб-страницы.

Скрипты распространяются и выполняются, как простой текст. Им не нужна специальная подготовка или компиляция для запуска.

Сегодня JavaScript может выполняться не только в браузере, но и на сервере или на любом другом устройстве, которое имеет специальную программу, называющуюся «движком» JavaScript.

У браузера есть собственный движок, который иногда называют «виртуальная машина JavaScript».

Современный JavaScript – это «безопасный» язык программирования. Он не предоставляет низкоуровневый доступ к памяти или процессору, потому что изначально был создан для браузеров, не требующих этого.

Возможности JavaScript сильно зависят от окружения, в котором он работает. Например, Node.JS поддерживает функции чтения/записи произвольных файлов, выполнения сетевых запросов и т.д.

В браузере для JavaScript доступно всё, что связано с манипулированием веб-страницами, взаимодействием с пользователем и веб-сервером.

Например, в браузере JavaScript может:

* Добавлять новый HTML-код на страницу, изменять существующее содержимое, модифицировать стили.
* Реагировать на действия пользователя, щелчки мыши, перемещения указателя, нажатия клавиш.
* Отправлять сетевые запросы на удалённые сервера, скачивать и загружать файлы (технологии AJAX и COMET).
* Получать и устанавливать куки, задавать вопросы посетителю, показывать сообщения.
* Запоминать данные на стороне клиента («local storage»).

# Основная часть

## «Серый» фильтр

Чтобы изображение имело только оттенки серого (от черного до белого) требуется преобразовать каждый пиксель изображения следующим образом:

AVG = (R+G+B)/3,

R’ = AVG,

G’ = AVG,

B’ = AVG.

Где R, G, B – красный, зеленый, синий соответственно цвета модели пикселя исходного изображения, R’, G’, B’ - красный, зеленый, синий соответственно цвета модели пикселя преобразованного изображения.

## Фильтр «Негатив»

Чтобы изображение стало негативным требуется каждый пиксель сделать негативным, то есть требуется инвертировать каждый пиксель. Так как максимальное значение каждого цвета в пикселе равняется 255, то достаточно вычесть из этого значения исходные цвета, то есть:

R’ = 255 - R,

G’ = 255 - G,

B’ = 255 - B.

## Регулирование яркости

Для изменения яркости изображения достаточно вычисление значений каждого из каналов пикселя по заданному коэффициенту яркости (K), используя следующую формулу:

R’ = R + K,

G’ = G + K,

B’ = B + K.

Если K > 0, то изображение становиться ярче, а если K < 0, то темнее

# Листинг кода

var ctx

const TypeFilter = {

GRAY: 1,

NEGATIVE: 2,

BRIGH: 3

}

function saveImage(){

const aElement = document.createElement('a')

aElement.setAttribute('download', "image.png")

aElement.href = canvasElement.toDataURL("image/png")

aElement.click()

}

function setGrayImageData(imageData){

for (i = 0; i < imageData.data.length; i += 4){

let rgb = [imageData.data[i], imageData.data[i+1], imageData.data[i+2]]

let avg = (rgb[0] + rgb[1] + rgb[2]) / 3

imageData.data[i] = imageData.data[i+1] = imageData.data[i+2] = avg

}

return imageData

}

function setNegativeImageData(imageData){

for (i = 0; i < imageData.data.length; i += 4){

let newRGB = [255 - imageData.data[i], 255 - imageData.data[i+1], 255 - imageData.data[i+2]]

imageData.data[i] = newRGB[0]

imageData.data[i+1] = newRGB[1]

imageData.data[i+2] = newRGB[2]

}

return imageData

}

function setBrighterImageData(imageData){

for (i = 0; i < imageData.data.length; i += 4){

imageData.data[i] += 10

imageData.data[i+1] += 10

imageData.data[i+2] += 10

}

return imageData

}

function setDarkerImageData(imageData){

for (i = 0; i < imageData.data.length; i += 4){

imageData.data[i] -= 10

imageData.data[i+1] -= 10

imageData.data[i+2] -= 10

}

return imageData

}

function setFilter(type){

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

var resultImageData

switch (type){

case TypeFilter.GRAY: resultImageData = setGrayImageData(imageData); break

case TypeFilter.NEGATIVE: resultImageData = setNegativeImageData(imageData); break;

case TypeFilter.BRIGH: resultImageData = setBrighterImageData(imageData); break;

case TypeFilter.DARK: resultImageData = setDarkerImageData(imageData); break;

}

ctx.putImageData(resultImageData, 0, 0, 0, 0, resultImageData.width, resultImageData.height);

}

function handleFile(files){

if (files.length > 0){

var file = files[0]

var reader = new FileReader()

reader.readAsDataURL(file)

var img = new Image()

ctx = canvasElement.getContext("2d")

img.onload = function() {

canvasElement.width = img.width

canvasElement.height = img.height

ctx.drawImage(img, 0, 0);

}

img.src = URL.createObjectURL(file);

} else {

alert("Выберите файл")

}

}

# Результаты выполнения работы

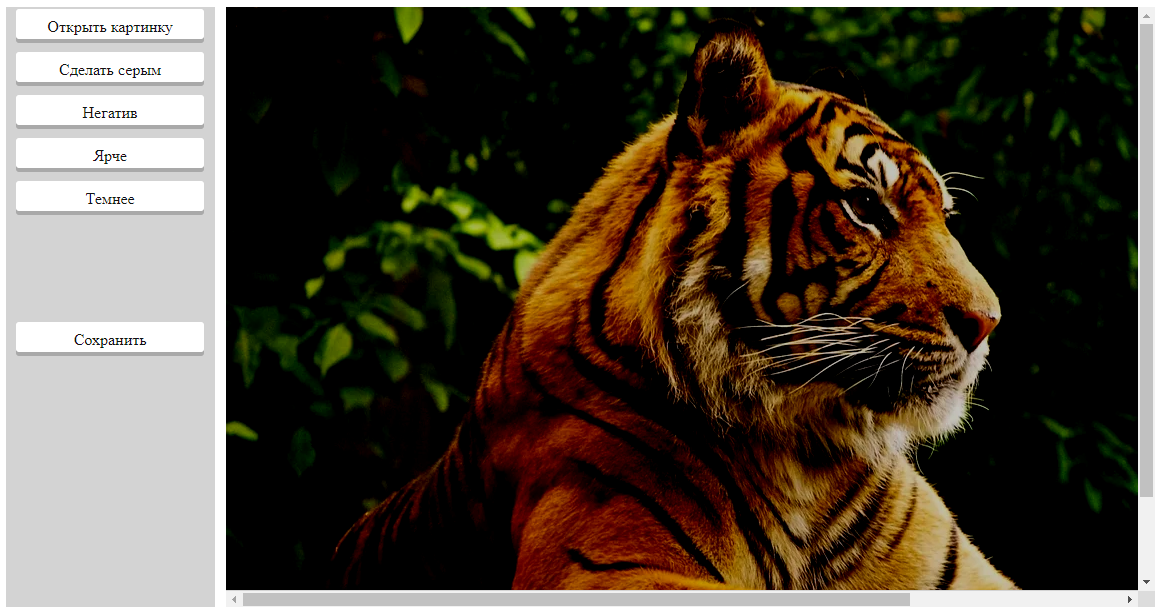
На рисунках 1-5 показаны результаты выполнения работы.



1. Оригинал изображения



1. Изображение с увеличенной яркостью



1. Изображение с пониженной яркостью



1. Изображение в негативе



1. Серое изображение

**Вывод**

В ходе выполнения лабораторной работы были получены навыки в программировании на языке Swift. Изучены основы синтаксиса языка, работа с базовыми типами данных, управляющими конструкциями, функциями и замыканиями.