Учреждение образования

«Гомельский Государственный технический университет имени П.О.Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Отчет по лабораторной работе №6

««Реализация программ для применения графических фильтров»

По дисциплине «Алгоритмические основы современной компьютерной графики»

Выполнил студент гр.ИТЗ-21

Касаткин Д.О.

Проверил преподаватель

Комракова М.В.

Гомель 2019

**Цель работы:** получить теоретические и практические знания по

созданию и применению графических фильтров.

**Задание:** разработать программу, использующую следующие графические

фильтры:

– инверсия цветов;

– размытие;

– увеличение резкости;

– тиснение;

– акварельный эффект.

Листинг программы:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Program

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

public static Bitmap image;

public static string full\_name\_of\_image = "\0";

public static UInt32[,] pixel;

//открытие изображения

private void открытьToolStripMenuItem\_Click(object sender, EventArgs e)

{

OpenFileDialog open\_dialog = new OpenFileDialog();

open\_dialog.Filter = "Image Files(\*.BMP;\*.JPG;\*.GIF;\*.PNG)|\*.BMP;\*.JPG;\*.GIF;\*.PNG|All files (\*.\*)|\*.\*";

if (open\_dialog.ShowDialog() == DialogResult.OK)

{

try

{

full\_name\_of\_image = open\_dialog.FileName;

image = new Bitmap(open\_dialog.FileName);

this.Width = image.Width + 40;

this.Height = image.Height + 75;

this.pictureBox1.Size = image.Size;

pictureBox1.Image = image;

pictureBox1.Invalidate();

//получение матрицы с пикселями

pixel = new UInt32[image.Height, image.Width];

for (int y = 0; y < image.Height; y++)

for (int x = 0; x < image.Width; x++)

pixel[y, x] = (UInt32)(image.GetPixel(x, y).ToArgb());

}

catch

{

full\_name\_of\_image = "\0";

DialogResult rezult = MessageBox.Show("Невозможно открыть выбранный файл",

"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

//сохранение изображения

private void сохранитьКакToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (pictureBox1.Image != null)

{

SaveFileDialog savedialog = new SaveFileDialog();

savedialog.Title = "Сохранить картинку как...";

savedialog.OverwritePrompt = true;

savedialog.CheckPathExists = true;

savedialog.Filter = "Image Files(\*.BMP)|\*.BMP|Image Files(\*.JPG)|\*.JPG|Image Files(\*.GIF)|\*.GIF|Image Files(\*.PNG)|\*.PNG|All files (\*.\*)|\*.\*";

savedialog.ShowHelp = true;

if (savedialog.ShowDialog() == DialogResult.OK)

{

try

{

image.Save(savedialog.FileName, System.Drawing.Imaging.ImageFormat.Jpeg);

}

catch

{

MessageBox.Show("Невозможно сохранить изображение", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

//Повышение резкости

private void повыситьРезкостьToolStripMenuItem\_Click(object sender, EventArgs e)

{

//for(int i=0;i<5;i++)

if (full\_name\_of\_image != "\0")

{

pixel = Filter.matrix\_filtration(image.Width, image.Height, pixel, Filter.N1, Filter.sharpness);

FromPixelToBitmap();

FromBitmapToScreen();

}

}

//размыть

private void размытьToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (full\_name\_of\_image != "\0")

{

pixel = Filter.matrix\_filtration(image.Width, image.Height, pixel, Filter.N2, Filter.blur);

FromPixelToBitmap();

FromBitmapToScreen();

}

}

//преобразование из UINT32 to Bitmap

public static void FromPixelToBitmap()

{

for (int y = 0; y < image.Height; y++)

for (int x = 0; x < image.Width; x++)

image.SetPixel(x, y, Color.FromArgb((int)pixel[y, x]));

}

//преобразование из UINT32 to Bitmap по одному пикселю

public static void FromOnePixelToBitmap(int x, int y, UInt32 pixel)

{

image.SetPixel(y, x, Color.FromArgb((int)pixel));

}

//вывод на экран

public void FromBitmapToScreen()

{

pictureBox1.Image = image;

}

private void акварельToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (full\_name\_of\_image != "\0")

{

pixel = Filter.matrix\_filtration(image.Width, image.Height, pixel, Filter.N3, Filter.aqua);

pixel = Filter.matrix\_filtration(image.Width, image.Height, pixel, Filter.N3, Filter.aqua\_after);

FromPixelToBitmap();

FromBitmapToScreen();

}

}

private void инверсияToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (Form1.full\_name\_of\_image != "\0")

{

UInt32 p;

for (int i = 0; i < Form1.image.Height; i++)

for (int j = 0; j < Form1.image.Width; j++)

{

p = Filter.Invers(Form1.pixel[i, j]);

Form1.FromOnePixelToBitmap(i, j, p);

}

FromBitmapToScreen();

}

}

private void тиснениеToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (full\_name\_of\_image != "\0")

{

pixel = Filter.matrix\_filtration(image.Width, image.Height, pixel, Filter.N3, Filter.tisnenie);

// pixel = Filter.matrix\_filtration(image.Width, image.Height, pixel, Filter.N3, Filter.aqua\_after);

FromPixelToBitmap();

FromBitmapToScreen();

}

}

}

}

using System;

namespace Program

{

struct RGB

{

public float R;

public float G;

public float B;

}

class Filter

{

public static UInt32[,] matrix\_filtration(int W, int H, UInt32[,] pixel, int N, double[,] matryx)

{

int i, j, k, m, gap = (int)(N / 2);

int tmpH = H + 2 \* gap, tmpW = W + 2 \* gap;

UInt32[,] tmppixel = new UInt32[tmpH, tmpW];

UInt32[,] newpixel = new UInt32[H, W];

//заполнение временного расширенного изображения

//углы

for (i = 0; i < gap; i++)

for (j = 0; j < gap; j++)

{

tmppixel[i, j] = pixel[0, 0];

tmppixel[i, tmpW - 1 - j] = pixel[0, W - 1];

tmppixel[tmpH - 1 - i, j] = pixel[H - 1, 0];

tmppixel[tmpH - 1 - i, tmpW - 1 - j] = pixel[H - 1, W - 1];

}

//крайние левая и правая стороны

for (i = gap; i < tmpH - gap; i++)

for (j = 0; j < gap; j++)

{

tmppixel[i, j] = pixel[i - gap, j];

tmppixel[i, tmpW - 1 - j] = pixel[i - gap, W - 1 - j];

}

//крайние верхняя и нижняя стороны

for (i = 0; i < gap; i++)

for (j = gap; j < tmpW - gap; j++)

{

tmppixel[i, j] = pixel[i, j - gap];

tmppixel[tmpH - 1 - i, j] = pixel[H - 1 - i, j - gap];

}

//центр

for (i = 0; i < H; i++)

for (j = 0; j < W; j++)

tmppixel[i + gap, j + gap] = pixel[i, j];

//применение ядра свертки

RGB ColorOfPixel = new RGB();

RGB ColorOfCell= new RGB();

for (i = gap; i < tmpH - gap; i++)

for (j = gap; j < tmpW - gap; j++)

{

ColorOfPixel.R = 0;

ColorOfPixel.G = 0;

ColorOfPixel.B = 0;

for (k = 0; k < N; k++)

for (m = 0; m < N; m++)

{

ColorOfCell = calculationOfColor(tmppixel[i - gap + k, j - gap + m], matryx[k, m]);

ColorOfPixel.R += ColorOfCell.R;

ColorOfPixel.G += ColorOfCell.G;

ColorOfPixel.B += ColorOfCell.B;

}

//контролируем переполнение переменных

if (ColorOfPixel.R < 0) ColorOfPixel.R = 0;

if (ColorOfPixel.R > 255) ColorOfPixel.R = 255;

if (ColorOfPixel.G < 0) ColorOfPixel.G = 0;

if (ColorOfPixel.G > 255) ColorOfPixel.G = 255;

if (ColorOfPixel.B < 0) ColorOfPixel.B = 0;

if (ColorOfPixel.B > 255) ColorOfPixel.B = 255;

newpixel[i - gap, j - gap] = build(ColorOfPixel);

}

return newpixel;

}

//вычисление нового цвета

public static RGB calculationOfColor(UInt32 pixel, double coefficient)

{

RGB Color = new RGB();

Color.R = (float)(coefficient \* ((pixel & 0x00FF0000) >> 16));

Color.G = (float)(coefficient \* ((pixel & 0x0000FF00) >> 8));

Color.B = (float)(coefficient \* (pixel & 0x000000FF));

return Color;

}

//сборка каналов

public static UInt32 build(RGB ColorOfPixel)

{

UInt32 Color;

Color = 0xFF000000 | ((UInt32)ColorOfPixel.R << 16) | ((UInt32)ColorOfPixel.G << 8) | ((UInt32)ColorOfPixel.B);

return Color;

}

//повышение резкости

public const int N1 = 3;

public static double[,] sharpness = new double[N1, N1] {{-1, -1, -1},

{-1, 9, -1},

{-1, -1, -1}};

//размытие

public const int N2 = 3;

public static double[,] blur = new double[N1, N1] {{0.111, 0.111, 0.111},

{0.111, 0.111, 0.111},

{0.111, 0.111, 0.111}};

//применения размытия для акварели

public const int N3 = 3;

public static double[,] aqua = new double[N1, N1] {{0.062,0.125,0.0625},

{0.125,0.25, 0.125},

{0.062,0.125,0.0625}};

//акварель подчеркивание контуров

public const int N4 = 3;

public static double[,] aqua\_after = new double[N1, N1] {{-0.5, -0.5, -0.5},

{-0.5, 5, -0.5},

{-0.5, -0.5, -0.5}};

public const int N5 = 3;

public static double[,] tisnenie = new double[N1, N1] {{0,1,0},

{-1,0,1},

{0,-1,0}};

public static UInt32 Invers(UInt32 point)

{

int R;

int G;

int B;

// int N = (100 / lenght) \* poz; //кол-во процентов

R = (int)(255 - ((point & 0x00FF0000) >> 16));

G = (int)(255 - ((point & 0x0000FF00) >> 8));

B = (int)(255 - (point & 0x000000FF));

//контролируем переполнение переменных

if (R < 0) R = 0;

if (R > 255) R = 255;

if (G < 0) G = 0;

if (G > 255) G = 255;

if (B < 0) B = 0;

if (B > 255) B = 255;

point = 0xFF000000 | ((UInt32)R << 16) | ((UInt32)G << 8) | ((UInt32)B);

return point;

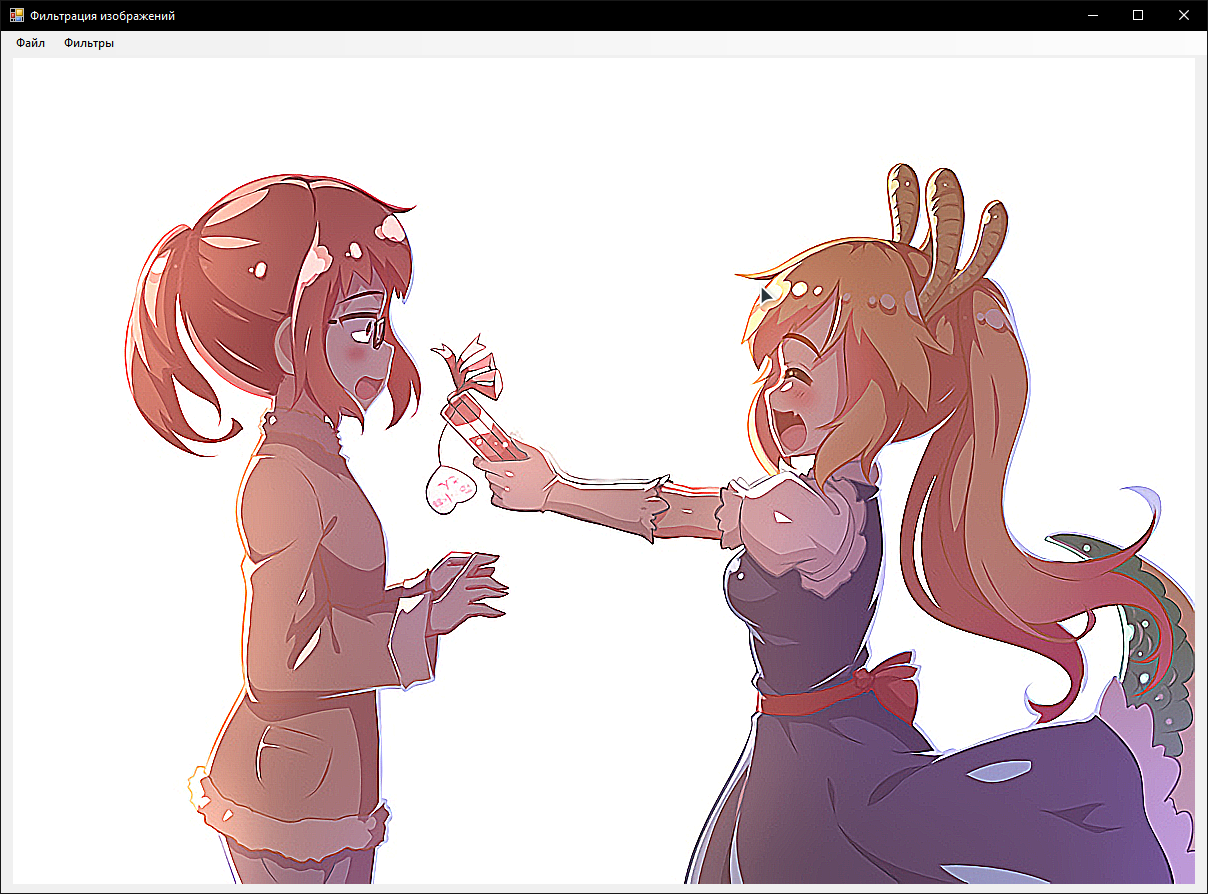
}

}

}



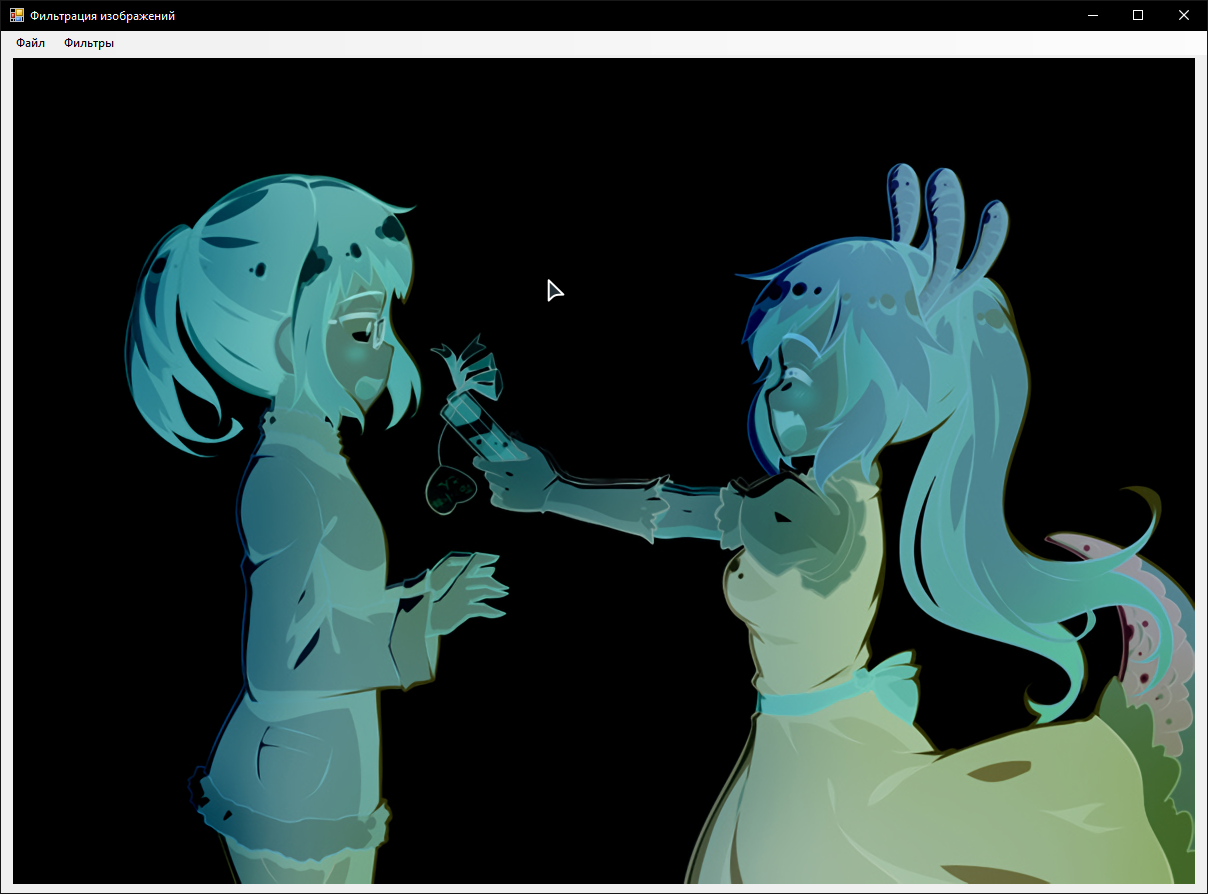
Скриншот 1 : запуск программы.



Скриншот 2 : Повышение резкости



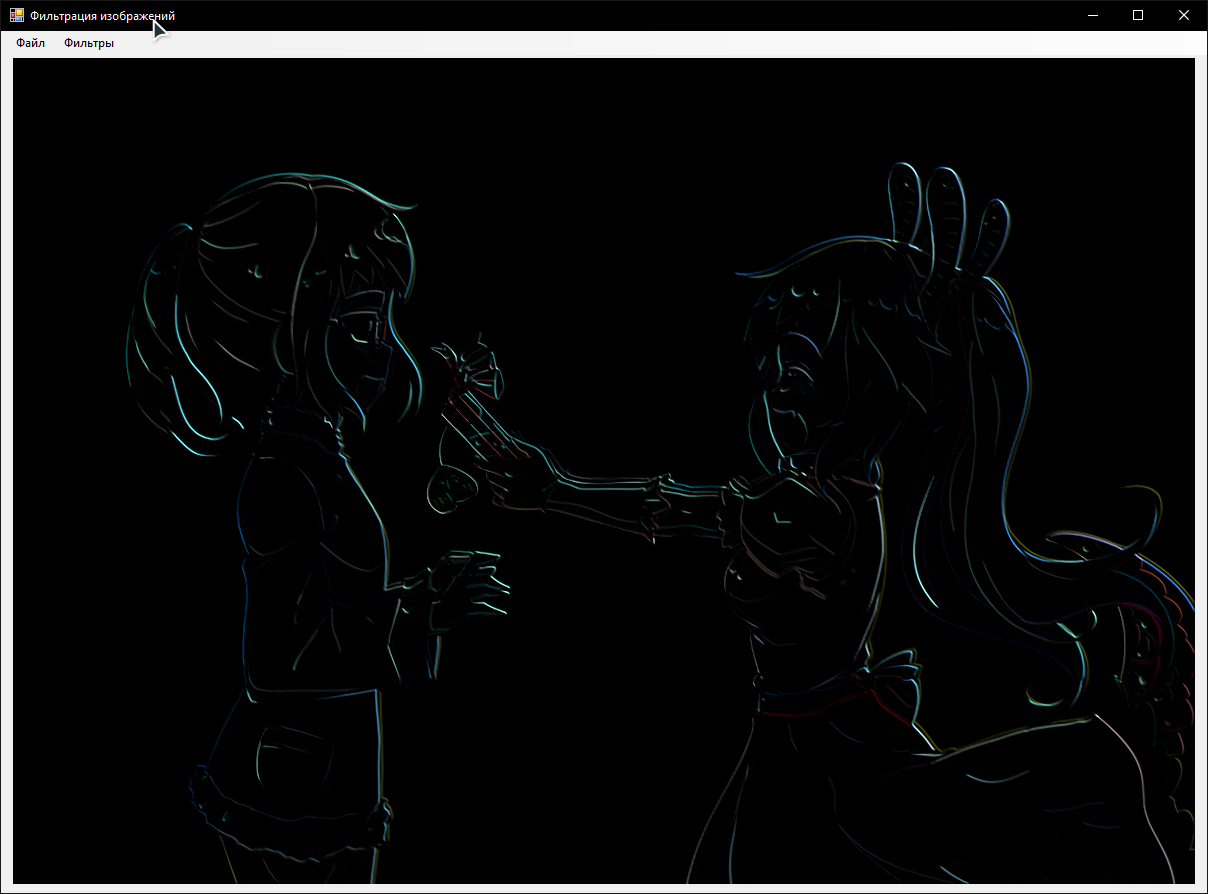
Скриншот 3: размытие



Скриншот 4: инверсия цветов



Скриншот 5: акварель



Скриншот 6: тиснение

**Вывод:** в ходе выполнения работы я получил теоретические и практические знания посозданию и применению графических фильтров.