Министерство образования и науки 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 № 7**

**по дисциплине**

**«Программирование компьютерной графики»**

**По теме: «Алгоритмы обработки растровых изображений»**

Выполнила:

ст. гр. ПРИ-118

Левченко В.П

Принял:

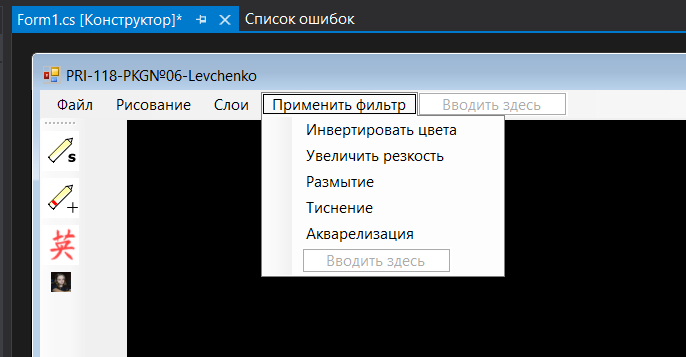
Жигалов И.Е

Владимир, 2021

**Цель работы**

Изучение методов преобразования растровых изображений, приводящих к их визуальному изменению с целью улучшения и получения различных специальных эффектов.

**Выполнение работы**

  
Рисунок 1. Добавление пунктов меню

Обработчики событий в классе anEngine:

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

{

ProgrammDrawingEngine.Filter\_0();

}

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

{

ProgrammDrawingEngine.Filter\_1();

}

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

{

ProgrammDrawingEngine.Filter\_2();

}

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

{

ProgrammDrawingEngine.Filter\_3();

}

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

{

ProgrammDrawingEngine.Filter\_4();

}

Фильтры событий в классе anEngine:

public void Filter\_0()

{

((anLayer)Layers[ActiveLayerNom]).Invers();

}

public void Filter\_1()

{

float[] mat = new float[9]; mat[0] = -0.1f;

mat[1] = -0.1f;

mat[2] = -0.1f;

mat[3] = -0.1f;

mat[4] = 1.8f;

mat[5] = -0.1f;

mat[6] = -0.1f;

mat[7] = -0.1f;

mat[8] = -0.1f;

((anLayer)Layers[ActiveLayerNom]).PixelTransformation(mat, 0, 1, false);

}

public void Filter\_2()

{

float[] mat = new float[9];

mat[0] = 0.05f;

mat[1] = 0.05f;

mat[2] = 0.05f;

mat[3] = 0.05f;

mat[4] = 0.6f;

mat[5] = 0.05f;

mat[6] = 0.05f;

mat[7] = 0.05f;

mat[8] = 0.05f;

((anLayer)Layers[ActiveLayerNom]).PixelTransformation(mat, 0, 1, false);

}

public void Filter\_3()

{

float[] mat = new float[9];

mat[0] = -1.0f;

mat[1] = -1.0f;

mat[2] = -1.0f;

mat[3] = -1.0f;

mat[4] = 8.0f;

mat[5] = -1.0f;

mat[6] = -1.0f;

mat[7] = -1.0f;

mat[8] = -1.0f;

((anLayer)Layers[ActiveLayerNom]).PixelTransformation(mat, 0, 2, true);

}

public void Filter\_4()

{

float[] mat = new float[9];

mat[0] = 0.50f;

mat[1] = 1.0f;

mat[2] = 0.50f;

mat[3] = 1.0f;

mat[4] = 2.0f;

mat[5] = 1.0f;

mat[6] = 0.50f;

mat[7] = 1.0f;

mat[8] = 0.50f;

((anLayer)Layers[ActiveLayerNom]).PixelTransformation(mat, 0, 2, true);

mat[0] = -0.5f;

mat[1] = -0.5f;

mat[2] = -0.5f;

mat[3] = -0.5f;

mat[4] = 6.0f;

mat[5] = -0.5f;

mat[6] = -0.5f;

mat[7] = -0.5f;

mat[8] = -0.5f;

((anLayer)Layers[ActiveLayerNom]).PixelTransformation(mat, 0, 1, false);

}

}

Метод в классе anLayer

public void Invers()

{

for (int Y = 0; Y < Heigth; Y++)

{

for (int X = 0; X < Width; X++)

{

DrawPlace[X, Y, 0] = 255 - DrawPlace[X, Y, 0];

DrawPlace[X, Y, 1] = 255 - DrawPlace[X, Y, 1];

DrawPlace[X, Y, 2] = 255 - DrawPlace[X, Y, 2];

}

}

}

Функция для преобразований:

public void PixelTransformation(float[] mat, int corr, float COEFF, bool need\_count\_correction)

{

// массив для получения результирующего пикселя

float[] resault\_RGB = new float[3];

int count = 0;

// проходим циклом по всем пикселям слоя

for (int Y = 0; Y < Heigth; Y++)

{

for (int X = 0; X < Width; X++)

{

// цикл по всем составляющим (0-2, т.е. R G B)

for (int c = 0, ax = 0, bx = 0; c < 3; c++)

{

// обнуление составляющей результата

resault\_RGB[c] = 0;

// обнуление счетчика обработок

count = 0;

// два цикла для захвата области 3х3 вокруг обрабатываемого пикселя

for (bx = -1; bx < 2; bx++)

{

for (ax = -1; ax < 2; ax++)

{

// если мы не попали в рамки, просто используем центральный пиксель, и продолжаем цикл

if (X + ax < 0 || X + ax > Width - 1 || Y + bx < 0 || Y + bx > Heigth - 1)

{

// считаем составляющую в одной из точек, используем коэфицент в матрице (под номером текущей итерации), коэфицент усиления (COEFF) и прибовляем коррекцию (corr)

resault\_RGB[c] += (float)(DrawPlace[X, Y, c]) \* mat[count] \* COEFF + corr;

// счетчик обработок = ячейке матрицы с необходимым коэфицентом

count++;

// продолжаем цикл

continue;

}

// иначе, если мы укладываемся в изображение (не пересекаем границы), используем соседние пиксели, корректируем ячейку массива параметрами ax, bx

resault\_RGB[c] += (float)(DrawPlace[X + ax, Y + bx, c]) \* mat[count] \* COEFF + corr;

// счетчик обработок = ячейке матрицы с необходимым коэфицентом

count++;

}

}

}

// теперь для всех составляющих корректируем цвет

for (int c = 0; c < 3; c++)

{

// если требуется разделить результат до приведения к 0-255, разделив на количество проведенных операций

if (count != 0 && need\_count\_correction)

{

// выполняем данное деление

resault\_RGB[c] /= count;

}

// если значение меньше нуля

if (resault\_RGB[c] < 0)

{

// - приравниваем к нулю

resault\_RGB[c] = 0;

}

// если больше 255

if (resault\_RGB[c] > 255)

{

// приравниваем к 255

resault\_RGB[c] = 255;

}

// записываем в массив цветов слоя новое значение

DrawPlace[X, Y, c] = (int)resault\_RGB[c];

}

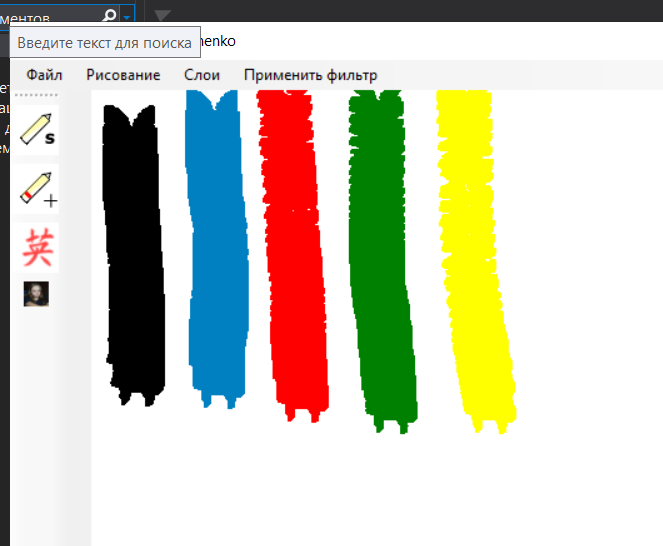
}

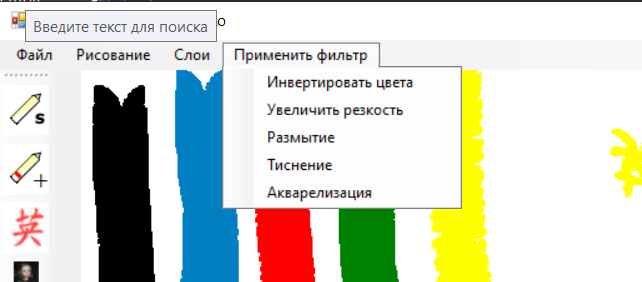
}

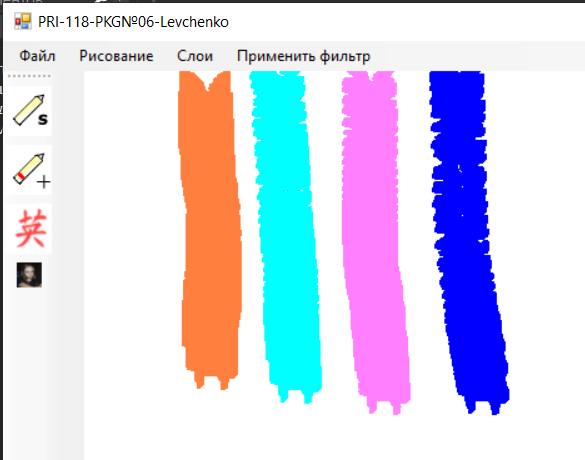
}

}

}

  
Рисунок 2. Первоначальный вид, до применения фильтров

  
Рисунок 3. Возможные фильтры

  
Рисунок 4. Инвертирование цветов

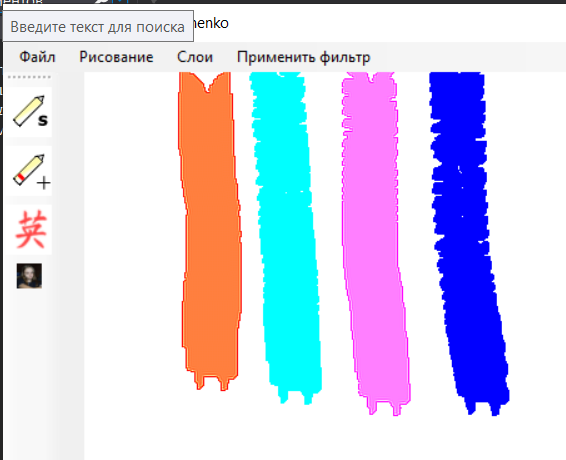
  
Рисунок 5. Увеличение резкости х3

  
Рисунок 6. Размытие х8

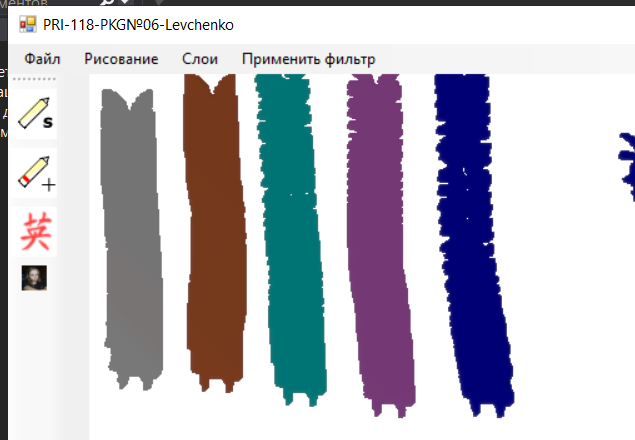
  
Рисунок 7. Тиснение

  
Рисунок 8. Акварелизация

**Вывод**

Изучены методы преобразования растровых изображений, приводящие к их визуальному изменению с целью улучшения и получения различных специальных эффектов.