**Министерство образования и науки Российской Федерации**

федеральное государственное автономное образовательное

учреждение высшего образования

**Санкт-Петербургский исследовательский университет**

**Информационных технологий, механики и оптики**

Факультет информационных технологий и программирования

Дисциплина: Компьютерная геометрия и графика

**Отчет**

по лабораторной работе №1

***Изучение простых преобразований изображений***

Выполнила: студент гр. M3101

Семина А. Д.

Преподаватель: Скаков П. С.

Санкт-Петербург

2020

**Цель работы:** изучить алгоритмы и реализовать программу выполняющую простые преобразования серых и цветных изображений в формате PNM.

**Описание работы**

Программа должна поддерживать серые и цветные изображения (варианты PNM P5 и P6), самостоятельно определяя формат по содержимому.

Аргументы программе передаются через командную строку:

**lab#.exe <имя\_входного\_файла> <имя\_выходного\_файла> <преобразование>**

где <преобразование>:

0 - инверсия,

1 - зеркальное отражение по горизонтали,

2 - зеркальное отражение по вертикали,

3 - поворот на 90 градусов по часовой стрелке,

4 - поворот на 90 градусов против часовой стрелки.

Программа должна быть написана на C/C++ и не использовать внешние библиотеки.

**Частичное решение:** работают преобразования 0-2; имена файлов и преобразование, возможно, написаны в исходном коде или читаются с консоли, а не берутся из командной строки.

**Полное решение:** всё работает + корректно выделяется и освобождается память, закрываются файлы, есть обработка ошибок: не удалось открыть файл, формат файла не поддерживается, не удалось выделить память.

|  |  |  |
| --- | --- | --- |
|  | До преобразования | После преобразования |
| По горизонтали |  |  |
| По вертикали |  |

**Теоретическая часть**

Простые форматы хранения изображений portable pixmap (иногда определяемые как PNM): цветных (PPM), полутоновых (PGM) и чёрно-белых (PBM) определяют правила для обмена графическими файлами. Эти форматы могут обеспечивать промежуточное представление данных при конвертации растровых графических файлов трёх перечисленных типов между разными платформами. Некоторые приложения поддерживают эти три формата напрямую, определяя их как формат PNM (portable anymap).

Файл расширения pnm имеет вид:

“P”<число от 1 до 6, которое определяет формат данных и вид изображения>

<число колонок> <число рядов>

<максимальное значение цвета>

<пиксели картинки>

**Инверсия:** каждому пикселю изменяется значение с текущего на [<максимальное значение цвета> - <текущее значение>].  
**Отражение:** переставляются столбцы или строки в обратном порядке.

**Поворот:** строки нового изображения – это столбцы исходного.

**Экспериментальная часть**

Язык программирования: C++ 17

Данные считываются в формате, указанном в описании работы. Если данные не соответствуют формату, то выводится ошибка и программа завершает свою работу. В зависимости от последнего аргумента вызываются функции, в которых производится инверсия, отображение или поворот изображения.

**Выводы**

В ходе проделанной работы были получены следующие результаты:

Для поворота изображения требуется выделение дополнительной памяти, для инверсии и отражения этого можно избежать, но в данной программе она все равно выделяется (вектор image1).

Время работы алгоритмов каждой функции - O(w \* h) для формата pgm (начинается с “P5”) и O(3 \* w \* h) = O(w \* h) для формата ppm (начинается с “P6”, 3 байта на каждый пиксель), где w и h – высота и ширина изображения. Сделать быстрее не получится, так как хотя бы раз мы должны пройти по всем пикселям изображения.

**Листинг**

**Название исходного файла : Source.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <algorithm>

#include <vector>

#include <string>

//using namespace std;

//bool flag\_error = false;

int transformation;

char symbol;

int number, h, w, max\_color;

std::vector<unsigned char> image1, image2;

FILE \*fin, \*fout;

void inversion()

{

for (int i = 0; i < image2.size(); ++i)

image2[i] = max\_color - (int)image1[i];

}

void reflection\_gorisontal()

{

if (number == 5)

{

for (int i = 0; i < w; ++i)

for (int j = 0; j < h; ++j)

image2[i \* h + j] = image1[i \* h + h - j - 1];

}

else

{

for (int i = 0; i < w; ++i)

for (int j = 0; j < 3 \* h; j += 3)

{

image2[i \* 3 \* h + j] = image1[i \* 3 \* h + 3 \* h - j - 3];

image2[i \* 3 \* h + j + 1] = image1[i \* 3 \* h + 3 \* h - j - 2];

image2[i \* 3 \* h + j + 2] = image1[i \* 3 \* h + 3 \* h - j - 1];

}

}

}

void reflection\_vertical()

{

if (number == 5)

{

for (int i = 0; i < w; ++i)

for (int j = 0; j < h; ++j)

image2[i \* h + j] = image1[(w - i - 1) \* h + j];

}

else

{

for (int i = 0; i < w; ++i)

for (int j = 0; j < 3 \* h; ++j)

image2[i \* 3 \* h + j] = image1[(w - i - 1) \* 3 \* h + j];

}

}

void clockwise\_rotation()

{

std::swap(w, h);

if (number == 5)

{

int k = 0;

for (int i = 0; i < w; ++i)

for (int j = 0; j < h; ++j)

{

image2[k] = image1[(h - 1 - j) \* w + i];

k++;

}

}

else

{

int k = 0;

for (int i = 0; i < 3 \* w; i += 3)

for (int j = 0; j < h; ++j)

{

image2[k] = image1[(h - 1 - j) \* 3 \* w + i];

image2[k + 1] = image1[(h - 1 - j) \* 3 \* w + i + 1];

image2[k + 2] = image1[(h - 1 - j) \* 3 \* w + i + 2];

k += 3;

}

}

}

void counterclockwise\_rotation()

{

std::swap(w, h);

if (number == 5)

{

int k = 0;

for (int i = 0; i < w; ++i)

for (int j = 0; j < h; ++j)

{

image2[k] = image1[j \* w + w - 1 - i];

k++;

}

}

else

{

int k = 0;

for (int i = 0; i < 3 \* w; i += 3)

for (int j = 0; j < h; ++j)

{

image2[k] = image1[j \* 3 \* w + 3 \* w - 3 - i];

image2[k + 1] = image1[j \* 3 \* w + 3 \* w - 2 - i];

image2[k + 2] = image1[j \* 3 \* w + 3 \* w - 1 - i];

k += 3;

}

}

}

void err(int i)

{

printf("Error\n");

if (i == 1)

{

printf("You should use this structure:\n");

printf("<input\_file\_name> <output\_file\_name> <transformation>\n");

printf("where <transformation> is integer in [0, 4]");

}

else if (i == 2)

printf("Incorrect pnm file");

else if (i == 3)

printf("Input file can't be open");

else if (i == 4)

printf("Output file can't be open");

fclose(fin);

fclose(fout);

system("pause");

exit(0);

}

int main(int argc, char \*argv[])

{

printf("PROCESSING\n");

if (argc == 4)

{

transformation = atoi(argv[3]);

if (transformation >= 5 || transformation < 0)

err(1);

}

else

err(1);

fin = fopen(argv[1], "rb");

if (fin == NULL)

err(3);

int quantity = fscanf(fin, "%c%d\n%d%d\n%d\n", &symbol, &number, &h, &w, &max\_color);

if (quantity != 5 || symbol != 'P' || (number != 5 && number != 6) || h <= 0 || w <= 0 || max\_color <= 0 || max\_color > 255)

err(2);

if (number == 5)

{

image1.resize(h \* w, 0);

image2.resize(h \* w, 0);

quantity = fread(&image1[0], sizeof(unsigned char), image1.size(), fin);

if (quantity != h \* w)

err(2);

}

else

{

image1.resize(3 \* h \* w, 0);

image2.resize(3 \* h \* w, 0);

quantity = fread(&image1[0], sizeof(unsigned char), image1.size(), fin);

if (quantity != 3 \* h \* w)

err(2);

}

switch (transformation)

{

case 0:

inversion();

break;

case 1:

reflection\_gorisontal();

break;

case 2:

reflection\_vertical();

break;

case 3:

clockwise\_rotation();

break;

case 4:

counterclockwise\_rotation();

break;

default:

err(1);

}

//if (!flag\_error)

//{

fout = fopen(argv[2], "wb");

if (fout == NULL)

err(4);

fprintf(fout, "%c%d\n%d %d\n%d\n", symbol, number, h, w, max\_color);

fwrite(&image2[0], sizeof(unsigned char), image2.size(), fout);

//}

fclose(fin);

fclose(fout);

printf("DONE");

system("pause");

return 0;

}