**Министерство образования и науки Российской Федерации**

федеральное государственное автономное образовательное

учреждение высшего образования

**Санкт-Петербургский исследовательский университет**

**Информационных технологий, механики и оптики**

Факультет: ФИТиП

Дисциплина: Компьютерная графика и геометрия

**Отчет**

по лабораторной работе № 1

***Изучение простых преобразований изображений***

Выполнила: студент гр. M3101

Дымчикова А. Ч.

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.

**Теоретическая часть**

В данной лабораторной работе необходимо проводить преобразования серых и цветных изображений в формате PNM (варианты P5 и P6), благодаря чему необходимо понимать структуру файлов в данных форматах и различия в вариантах P5 и P6. Простые форматы хранения изображений portable pixmap (иногда определяемые как PNM): цветных (PPM), полутоновых (PGM) и чёрно-белых (PBM) определяют правила для обмена графическими файлами. Некоторые приложения поддерживают эти три формата напрямую, определяя их как формат PNM (portable anymap). В зависимости от распознанного формата файла, система простых графических файлов portable pixmap может различать три сходных формата файлов, каждый в двух версиях:

PBM — чёрно-белый (portable bitmap) (P1/P4) — 1 бит на пиксель

PGM — полутоновый (portable graymap) (P2/P5) — 8 бит на пиксель

PPM — цветной (portable pixmap) (P3/P6) — 24 бита на пиксель (по 8 на красный, зелёный и синий)

В каждом случае нижняя версия (P1, P2 или P3) относятся к удобочитаемым, основанным на ASCII форматам, а верхние версии (P4, P5 и P6) относятся к двоичным форматам, менее удобным для разбора, но более эффективным для экономии места в файле и более удобным для разбора из-за отсутствия пробелов.

В данной работе могут использоваться только форматы P5 и P6.

Рассмотрим формат P5:

Первой строкой в данном файле должна являться строка «P5». Далее может следовать строка комментария, отделенная знаком «#». Следующие два числа задают ширину и высоту. Затем идет число, обозначающее максимальное значение цвета. И матрица пикселей, где каждый байт задает цвет одного пикселя.

Рассмотрим формат P6:

Первой строкой в данном файле должна являться строка «P6». Далее может следовать строка комментария, отделенная знаком «#». Следующие два числа задают ширину и высоту. Затем идет число, обозначающее максимальное значение цвета. И матрица пикселей, где каждый пиксель задается тремя байтами, причем первый байт отвечает за красную составляющую, второй – за зеленую, третий – за голубую.

Необходимо реализовать следующие пункты:

* Инверсия
* Зеркальное отражение по горизонтали и вертикали
* Поворот на 90 градусов по/против часовой.

Для инвертирования изображения необходимо изменить цвет каждого пикселя на противоположный, т. е. заменить пиксель с содержимым A на разность DEPTH – A, где DEPTH – глубина цвета изображения (максимальное значение цвета). В случае с форматом P6, каждая составляющая цвета инвертируется независимо от другой. По предварительным оценкам данная операция занимает не меньше, чем O(size) времени, где size – размер массива пикселей.

При зеркальном отображении необходимо найти центральную строку(столбец) и поменять местами симметричные относительно нее(него) строки(столбцы). При этом в случае формата P6, составляющие цвета одного пикселя не меняют своего взаимного расположения друг относительно друга. По предварительным оценкам данные операции занимают не меньше, чем O(size) времени.

При повороте на 90 градусов меняются местами и разворачиваются строки и столбцы в некотором установленном порядке. По предварительным оценкам данная операция занимает не меньше, чем O(size) времени.

**Экспериментальная часть**

Язык программирования: С++17.

В процессе реализации для удобства использования создаются два класса: PNM\_Header и PNM\_File, где хранятся данные файла. В конструкторе классов происходит считывание данных и проверка на корректность.

PNM\_Header представляет собой совокупность полей:

* format – отвечает за формат файла, может принимать значения 5 и 6;
* width – ширина матрицы пикселей
* height – высота матрицы пикселей
* depth – глубина цвета (максимальное значение цвета)

PNM\_File представляет собой совокупность полей:

* header – поле типа PNM\_Header c необходимыми данными
* data – массив пикселей
* size – размер массива

Далее следует описание реализации всех операций:

*Инвертирование*: для каждого пикселя выполняется следующая операция: замена пикселя с содержимым A на разность DEPTH – A, где DEPTH – глубина цвета изображения (максимальное значение цвета). В случае с форматом P6, каждая составляющая цвета инвертируется независимо от другой.

*Зеркальное отображение по вертикали*: строки, симметричные относительно центральной, меняются местами. При этом в случае формата P6, составляющие цвета одного пикселя не меняют своего взаимного расположения друг относительно друга.

*Зеркальное отображение по горизонтали*: столбцы, симметричные относительно центрального, меняются местами. При этом в случае формата P6, составляющие цвета одного пикселя не меняют своего взаимного расположения друг относительно друга.

*Поворот на 90 градусов против часовой стрелки*: создается новый массив, в котором последний столбец изначального массива будет лежать в первой строке, предпоследний столбец – во второй и т. д. Старый массив PNM\_File заменяется на новый.

*Поворот на 90 градусов по часовой стрелке*: производится поворот на 90 градусов против часовой стрелки, затем изображение отражается по горизонтали и вертикали.

**Выводы**

В ходе проделанной работы были получены следующие результаты: асимптотика времени работы всех операций совпадает с асимптотикой в теоретической части, все операции, за исключением поворота по часовой стрелке, выполнены с незначительными дополнительными затратами по времени работы. Возможно улучшение времени работы программы при повороте по часовой стрелке в том случае, если приведенный алгоритм заменить на алгоритм без промежуточных преобразований, таких как отражение по горизонтали и вертикали. Также для операций поворота на 90 градусов требуется выделение дополнительной памяти размера O(size).

**Листинг**

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

**using** **namespace** std;

**class** **PNM\_Header** {

**public:**

**char** format;

**int** width;

**int** height;

**int** depth;

PNM\_Header() {};

PNM\_Header(**FILE** \*fin) {

**char** checker;

**int** check = fscanf(fin, "%c%c**\n**%d %d**\n**%d**\n**", &checker, &format, &width, &height, &depth);

**if** (check != **5** || checker != 'P' || format != '5' && format != '6'

|| width <= **0** || height <= **0** || depth != **255**) {

**throw** exception("Invalid data format!**\n**");

}

}

};

**class** **PNM\_File** {

**public:**

PNM\_Header header;

vector <**unsigned** **char**> data;

**int** size = **0**;

PNM\_File() {}

PNM\_File(**FILE** \*fin) {

header = PNM\_Header(fin);

**if** (header.format == '5') {

size = header.width \* header.height;

}

**else** {

size = header.width \* header.height \* **3**;

}

data.resize(size);

**int** check = fread(&data[**0**], **sizeof**(**unsigned** **char**), size, fin);

**if** (check != size) {

**throw** exception("Invalid data format: width or**\a**nd height are wrong!**\n**");

}

}

};

**void** **inverse**(PNM\_File &file) {

**for** (**int** i = **0**; i < file.size; i++) {

file.data[i] = **unsigned** **char**(file.header.depth - file.data[i]);

}

}

**void** **vertical\_reflection**(PNM\_File &file) {

**int** h = file.header.height, w = file.header.width;

**if** (file.header.format == '6') {

w \*= **3**;

}

**for** (**int** i = **0**; i < h / **2**; i++) {

**for** (**int** j = **0**; j < w; j++) {

swap(file.data[w \* i + j],

file.data[w \* (h - i - **1**) + j]);

}

}

}

**void** **horisontal\_reflection**(PNM\_File &file) {

**int** h = file.header.height, w = file.header.width, factor = **1**;

**if** (file.header.format == '6') {

factor = **3**;

}

**for** (**int** k = **0**; k < factor; k++) {

**for** (**int** i = **0**; i < w / **2**; i++) {

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

swap(file.data[(w \* j + i) \* factor + k],

file.data[(w \* (j + **1**) - i - **1**) \* factor + k]);

}

}

}

}

**void** **rotate\_counter\_clockwise**(PNM\_File &file) {

vector <**unsigned** **char**> ndata;

ndata.resize(file.size);

**int** h = file.header.height, w = file.header.width, factor = **1**;

**if** (file.header.format == '6') {

factor = **3**;

}

**for** (**int** k = **0**; k < factor; k++) {

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

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

ndata[(h \* i + j) \* factor + k] = file.data[(w \* (j + **1**) - i - **1**) \* factor + k];

}

}

}

file.data = ndata;

swap(file.header.height, file.header.width);

}

**void** **rotate\_clockwise**(PNM\_File &file) {

rotate\_counter\_clockwise(file);

horisontal\_reflection(file);

vertical\_reflection(file);

}

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

**if** (argc != **4**) {

cout << "Wrong number of arguments!";

**return** **1**;

}

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

**if** (!fin) {

cout << "Input file error!**\n**";

cout << argv[**1**];

**return** **1**;

}

PNM\_File file;

try {

file = PNM\_File(fin);

}

**catch** (exception err) {

fclose(fin);

cout << err.what();

**return** **1**;

}

fclose(fin);

**switch** (argv[**3**][**0**] - '0') {

**case** **0**:

inverse(file);

**break**;

**case** **1**:

horisontal\_reflection(file);

**break**;

**case** **2**:

vertical\_reflection(file);

**break**;

**case** **3**:

rotate\_clockwise(file);

**break**;

**case** **4**:

rotate\_counter\_clockwise(file);

**break**;

**default:**

cout << "Wrong operation!";

**return** **1**;

}

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

**if** (!fout) {

cout << "Output file error!";

**return** **1**;

}

fprintf(fout, "P%c**\n**%d %d**\n**%d**\n**", file.header.format, file.header.width, file.header.height, file.header.depth);

fwrite(&file.data[**0**], **sizeof**(**unsigned** **char**), file.size, fout);

fclose(fout);

**return** **0**;

}