**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 № 2

Изучение алгоритмов отрисовки растровых линий с применением сглаживания и гамма-коррекции

Выполнил: студент гр. M3102

Муслимов О. С.

Преподаватель: Скаков П.С.

Санкт-Петербург

2020

**Цель работы:** изучить алгоритмы и реализовать программу, рисующую линию на изображении в формате PGM (P5) с учетом гамма-коррекции sRGB.

**Описание работы**

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

Аргументы передаются через командную строку:

**program.exe <имя\_входного\_файла> <имя\_выходного\_файла> <яркость\_линии> <толщина\_линии> <x\_начальный> <y\_начальный> <x\_конечный> <y\_конечный> <гамма>**

где

* <яркость\_линии>: целое число 0..255;
* <толщина\_линии>: положительное дробное число;
* <x,y>: координаты внутри изображения, (0;0) соответствует левому верхнему углу, дробные числа (целые значения соответствуют центру пикселей).
* <гамма>: (optional)положительное вещественное число: гамма-коррекция с введенным значением в качестве гаммы. При его отсутствии используется sRGB.

**Частичное решение**: <толщина линии>=1, <гамма>=2.0, координаты начала и конца – целые числа, чёрный фон вместо данных исходного файла (размеры берутся из исходного файла).

**Полное решение**: всё работает (гамма + sRGB, толщина не только равная 1, фон из входного изображения) + корректно выделяется и освобождается память, закрываются файлы, есть обработка ошибок.

Если программе передано значение, которое не поддерживается – следует сообщить об ошибке.

Коды возврата:

0 - ошибок нет

1 - произошла ошибка

В поток вывода ничего не выводится (printf, cout).

Сообщения об ошибках выводятся в поток вывода ошибок:

С: fprintf(stderr, "Error\n");

C++: std::cerr

Следующие параметры гарантировано не будут выходить за обусловленные значения:

* <яркость\_линии> = целое число 0..255;
* <толщина\_линии> = положительное вещественное число;
* width и height в файле - положительные целые значения;
* яркостных данных в файле ровно width \* height;
* <x\_начальный> <x\_конечный> = [0..width];
* <y\_начальный> <y\_конечный> = [0..height];

**Теоретическая часть**

Уравнение прямой из точки в точку имеет вид:

Для рисовании линии используется алгоритм Ву. ***Алгоритм Ву*** — это алгоритм разложения отрезка в растр со сглаживанием. В алгоритме Ву на каждом шаге устанавливается не одна, а две точки. Например, если основной осью является Х, то рассматриваются точки с координатами (х, у) и (х, у + 1). В зависимости от величины ошибки, которая показывает, как далеко ушли пиксели от идеальной линии по неосновной оси, распределяется интенсивность между этими двумя точками. Чем больше удалена точка от идеальной линии, тем меньше её интенсивность.

В данной лабораторной работе я использовал такое понятие, как Гамма-коррекция. Гамма – это отношение между численным значением пикселя и действительной совместимостью. Идея гаммы-коррекции заключается в том, чтобы применить инверсию гаммы монитора к окончательному цвету перед выводом на монитор.

*– прямая формула гамма-коррекции*

*– обратная формула гамма-коррекции*

Также, я использовал такое понятие, как sRGB. sRGB является стандартом представления цветового спектра с использованием модели RGB. В отличие от большинства других цветовых пространств RGB, гамма в sRGB не может быть выражена одним числовым значением, так как функция коррекции состоит из линейной части около чёрного цвета, где гамма равна 1.0, и нелинейной части до значения 2.4 включительно.

*– прямая формула sRgb*

*– обратная*

Еще есть такое понятие, как Альфа-смешивание. ***Альфа-смешение*** есть процесс комбинирования двух объектов на экране с учётом их альфа-каналов. Альфа-смешение используется для: антиалиасинга; создания прозрачности, теней, зеркал, тумана.

**R = B + (F − B) ⋅ A**, где В – яркость фонового пикселя, F – яркость накладываемого пикселя, А – непрозрачность накладываемого пикселя, R – результат.

**Экспериментальная часть**

Язык программирования: C++ 20.

Так как Алгоритм работает только для линий, у которых приращение больше по иксу, чем по игреку, поэтому, учитывая это, мы будем менять местами х и у, и в дальнейшем, при рисовании точки, координаты меняются обратно. Также, необходимо поменять начало и конец линии, если линию задали справа налево. Также, мы должны применять гамму-коррекцию (или sRGB, когда отсутствует значение гаммы) и альфа-смешивание. Еще был написан класс, который фиксирует ошибки введенных аргументов в консоль и ошибки, которые могут присутствовать в файле.

**Выводы**

В ходе проделанной работы были получены следующие результаты:

Полученные результаты могут не совпадать с теоритическими выкладками, потому что вычисляемые значения точек не целые числа, а координаты пикселя целые и присутствуют погрешности при вычислении координат точек.

**Листинг**

**main.cpp**

|  |
| --- |
| #include <iostream> |
|  | #include "CExpension.h" |
|  | #include "CLine.h" |
|  | #include "CImage.h" |
|  | #include <cmath> |
|  |  |
|  | using namespace std; |
|  |  |
|  | void getAnsw(const char \*fileName, CImage &output) { |
|  | FILE \*new\_f = fopen(fileName, "wb"); |
|  | if (!new\_f) { |
|  | throw CExpension("Output file didn't open", new\_f); |
|  | } |
|  | fprintf(new\_f, "P%i\n%i %i\n%i\n", 5, output.width, output.height, output.max\_val); |
|  | fwrite(output.pix, sizeof(unsigned char), output.size, new\_f); |
|  | fclose(new\_f); |
|  | } |
|  |  |
|  | int main(int argc, char \*argv[]) { |
|  | try { |
|  | SInput input; |
|  | FILE \*f; |
|  | if (argc > 10 && argc < 9) { |
|  | throw CExpension("Wrong amount of arguments"); |
|  | } |
|  | for (int i = 1; i < argc; i++) { |
|  | switch (i) { |
|  | case 1: |
|  | input.inputName = argv[i]; |
|  | f = fopen(input.inputName, "rb"); |
|  | if (!f) { |
|  | throw CExpension("Input File didn't open", f); |
|  | } |
|  | break; |
|  | case 2: |
|  | input.outputName = argv[i]; |
|  | break; |
|  | case 3: |
|  | input.brightness = atoi(argv[i]); |
|  | break; |
|  | case 4: |
|  | input.thickness = atof(argv[i]); |
|  | break; |
|  | case 5: |
|  | input.start.x\_ = atof(argv[i]); |
|  | break; |
|  | case 6: |
|  | input.start.y\_ = atof(argv[i]); |
|  | break; |
|  | case 7: |
|  | input.end.x\_ = atof(argv[i]); |
|  | break; |
|  | case 8: |
|  | input.end.y\_ = atof(argv[i]); |
|  | break; |
|  | case 9: |
|  | input.gamma = atof(argv[i]); |
|  | break; |
|  | } |
|  | } |
|  | CLine line(input); |
|  | CImage image(f); |
|  | image.drawLine(line, input); |
|  | getAnsw(input.outputName, image); |
|  | return 0; |
|  | } catch (CExpension &expension) { |
|  | cerr << expension.getError(); |
|  | if (expension.getFile()) { |
|  | fclose(expension.getFile()); |
|  | } |
|  | exit(1); |
|  | } |
|  | } |

**CException.h**

|  |
| --- |
| #ifndef GRAPH\_CEXPENSION\_H |
|  | #define GRAPH\_CEXPENSION\_H |
|  |  |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | class CExpension { |
|  | public: |
|  | CExpension(string error); |
|  |  |
|  | CExpension(string error, FILE \*file); |
|  |  |
|  | string getError(); |
|  |  |
|  | FILE \*getFile(); |
|  |  |
|  | private: |
|  | string error\_; |
|  | FILE \*file\_ = nullptr; |
|  | }; |
|  |  |
|  |  |
|  | #endif //GRAPH\_CEXPENSION\_H |

**CException.cpp**

|  |
| --- |
| #include "CExpension.h" |
|  | CExpension::CExpension(string error) { |
|  | this->error\_ = error; |
|  | } |
|  | CExpension::CExpension(string error, FILE \*file) { |
|  | this->error\_ = error; |
|  | this->file\_ = file; |
|  | } |
|  |  |
|  | string CExpension::getError() { |
|  | return this->error\_; |
|  | } |
|  | FILE\* CExpension::getFile() { |
|  | return this->file\_; |
|  | } |

**CImage.h**

|  |
| --- |
| #ifndef GRAPHICS\_CIMAGE\_H |
|  | #define GRAPHICS\_CIMAGE\_H |
|  |  |
|  | #include "CExpension.h" |
|  | #include <iostream> |
|  | #include "CLine.h" |
|  |  |
|  | struct IPoint { |
|  | int x; |
|  | int y; |
|  | }; |
|  |  |
|  | class CImage { |
|  | public: |
|  | CImage(FILE \*f); |
|  |  |
|  | ~CImage(); |
|  |  |
|  |  |
|  | friend void getAnsw(const char \*fileName, CImage &output); |
|  |  |
|  | void WuLine(CLine line); |
|  |  |
|  | void drawLine(CLine line, SInput input); |
|  |  |
|  | private: |
|  | FILE \*file; |
|  | int version; |
|  | int width; |
|  | int height; |
|  | int max\_val; |
|  | int size; |
|  | unsigned char \*pix; |
|  |  |
|  | bool inRec(int x, int y, CLine AB, CLine BC, CLine CD, CLine DA); |
|  |  |
|  | void point(bool sw, IPoint p, double brightness, CLine line); |
|  |  |
|  | double Decorrection(double value, CLine line); |
|  |  |
|  | double gammaCorrection(double value, CLine line); |
|  | }; |
|  |  |
|  |  |
|  | #endif //GRAPHICS\_CIMAGE\_H |

**CImage.cpp**

|  |
| --- |
| #include "CImage.h" |
|  | #include <cstdlib> |
|  | #include <cmath> |
|  |  |
|  | using namespace std; |
|  |  |
|  | CImage::CImage(FILE \*f) { |
|  | file = f; |
|  | if (fscanf(f, "P%i%i%i%i\n", &this->version, &this->width, &this->height, &max\_val) != 4) { |
|  | throw CExpension("Wrong amount data in file", f); |
|  | } |
|  | size = width \* height; |
|  | pix = new unsigned char[size]; |
|  | fread(pix, sizeof(unsigned char), size, f); |
|  | fclose(f); |
|  | } |
|  |  |
|  | CImage::~CImage() { |
|  | delete[] this->pix; |
|  | } |
|  |  |
|  |  |
|  | double CImage::Decorrection(double value, CLine line) { |
|  | value /= 255.0; |
|  | if (line.gamma\_ == -1) { |
|  | double a = 0.055; |
|  | if (value <= 0.0031308) { |
|  | return 255 \* value \* 12.92; |
|  | } |
|  | return 255 \* (1.0 + a) \* pow(value, (double) 1 / 2.4) - a; |
|  | } |
|  | return 255 \* pow(value, (double) 1 / line.gamma\_); |
|  | } |
|  |  |
|  | double CImage::gammaCorrection(double value, CLine line) { |
|  | value /= 255.0; |
|  | if (line.gamma\_ == -1) { |
|  | double a = 0.055; |
|  | if (value <= 0.04045) { |
|  | return 255 \* value / 12.92; |
|  | } |
|  | return 255 \* pow((value + a) / (1.0 + a), 2.4); |
|  | } |
|  | return 255 \* pow(value, line.gamma\_); |
|  | } |
|  | //----------------------Ву-------------------------- |
|  |  |
|  | void CImage::WuLine(CLine line) { |
|  | bool swapped = abs(line.end\_.y\_ - line.start\_.y\_) > abs(line.end\_.x\_ - line.start\_.x\_); |
|  | if (swapped) { |
|  | swap(line.start\_.x\_, line.start\_.y\_); |
|  | swap(line.end\_.x\_, line.end\_.y\_); |
|  | } |
|  | if (line.start\_.x\_ > line.end\_.x\_) { |
|  | swap(line.start\_, line.end\_); |
|  | } |
|  | double dx = line.end\_.x\_ - line.start\_.x\_; |
|  | double dy = line.end\_.y\_ - line.start\_.y\_; |
|  | double gradient = dy / dx; |
|  | double y = line.start\_.y\_ + gradient \* (round(line.start\_.x\_) - line.start\_.x\_); |
|  | int x = (int) round(line.start\_.x\_); |
|  | while (x <= (int) round(line.end\_.x\_)) { |
|  | double b = y - (int) y; |
|  | point(swapped, {x, (int) y}, 1.0 - b, line); |
|  | point(swapped, {x, (int) y + 1}, b, line); |
|  | y += gradient; |
|  | x++; |
|  | } |
|  | } |
|  |  |
|  | bool CImage::inRec(int x, int y, CLine AB, CLine BC, CLine CD, CLine DA) { |
|  | if (y <= AB.start\_.y\_ + x \* (AB.end\_.y\_ - AB.start\_.y\_) / (AB.end\_.x\_ - AB.start\_.x\_) - |
|  | AB.start\_.x\_ \* (AB.end\_.y\_ - AB.start\_.y\_) / (AB.end\_.x\_ - AB.start\_.x\_)) { |
|  | return false; |
|  | } |
|  | if (y >= BC.start\_.y\_ + x \* (BC.end\_.y\_ - BC.start\_.y\_) / (BC.end\_.x\_ - BC.start\_.x\_) - |
|  | BC.start\_.x\_ \* (BC.end\_.y\_ - BC.start\_.y\_) / (BC.end\_.x\_ - BC.start\_.x\_)) { |
|  | return false; |
|  | } |
|  | if (y >= CD.start\_.y\_ + x \* (CD.end\_.y\_ - CD.start\_.y\_) / (CD.end\_.x\_ - CD.start\_.x\_) - |
|  | CD.start\_.x\_ \* (CD.end\_.y\_ - CD.start\_.y\_) / (CD.end\_.x\_ - CD.start\_.x\_)) { |
|  | return false; |
|  | } |
|  | if (y <= DA.start\_.y\_ + x \* (DA.end\_.y\_ - DA.start\_.y\_) / (DA.end\_.x\_ - DA.start\_.x\_) - |
|  | DA.start\_.x\_ \* (DA.end\_.y\_ - DA.start\_.y\_) / (DA.end\_.x\_ - DA.start\_.x\_)) { |
|  | return false; |
|  | } |
|  | return true; |
|  | } |
|  |  |
|  | void CImage::point(bool sw, IPoint p, double brightness, CLine line) { |
|  | double b = round(gammaCorrection(pix[p.x \* width + p.y], line) \* (1.0 - brightness) + line.brightness \* brightness); |
|  | if (sw) { |
|  | swap(p.x, p.y); |
|  | } |
|  | pix[p.y \* width + p.x] = (int) round( |
|  | Decorrection(line.brightness / 255.0 \* b, line)); |
|  | } |
|  |  |
|  | void CImage::drawLine(CLine line, SInput input) { |
|  | if (line.thickness < 0) { |
|  | throw CExpension("Thickness is < 0", file); |
|  | } |
|  | if (line.thickness == 1.0) { |
|  | WuLine(line); |
|  | } else { |
|  | double a = line.end\_.x\_ - line.start\_.x\_ == 0 ? M\_PI / 2 : M\_PI / 2 - abs(atan( |
|  | (line.end\_.y\_ - line.start\_.y\_) / (line.end\_.x\_ - line.start\_.x\_))); |
|  | double l = line.thickness / 2.0; |
|  | SPoint A; |
|  | SPoint B; |
|  | SPoint C; |
|  | SPoint D; |
|  | A.x\_ = line.start\_.x\_ + l \* sin(a); |
|  | A.y\_ = line.start\_.y\_ + l \* cos(a); |
|  | B.x\_ = line.start\_.x\_ - l \* sin(a); |
|  | B.y\_ = line.start\_.y\_ - l \* cos(a); |
|  | C.x\_ = line.end\_.x\_ - l \* sin(a); |
|  | C.y\_ = line.end\_.y\_ - l \* cos(a); |
|  | D.x\_ = line.end\_.x\_ + l \* sin(a); |
|  | D.y\_ = line.end\_.y\_ + l \* cos(a); |
|  | if ((line.end\_.y\_ - line.start\_.y\_) / (line.end\_.x\_ - line.start\_.x\_) < 0) { |
|  | swap(A, D); |
|  | swap(B, C); |
|  | } |
|  | SInput ABinf = input; |
|  | SInput BCinf = input; |
|  | SInput CDinf = input; |
|  | SInput DAinf = input; |
|  | ABinf.start = A; |
|  | ABinf.end = B; |
|  | BCinf.start = B; |
|  | BCinf.end = C; |
|  | CDinf.start = C; |
|  | CDinf.end = D; |
|  | DAinf.start = D; |
|  | DAinf.end = A; |
|  | CLine AB(ABinf); |
|  | CLine BC(BCinf); |
|  | CLine CD(CDinf); |
|  | CLine DA(DAinf); |
|  | WuLine(AB); |
|  | WuLine(BC); |
|  | WuLine(CD); |
|  | WuLine(DA); |
|  | } |
|  | } |

**CLine.h**

|  |
| --- |
| #ifndef GRAPH\_CLINE\_H |
|  | #define GRAPH\_CLINE\_H |
|  |  |
|  |  |
|  | struct SPoint { |
|  | double x\_; |
|  | double y\_; |
|  | }; |
|  |  |
|  | struct SInput { |
|  | const char \*inputName; |
|  | const char \*outputName; |
|  | int brightness; |
|  | double thickness; |
|  | SPoint start; |
|  | SPoint end; |
|  | double gamma = -1; |
|  | }; |
|  |  |
|  | class CLine { |
|  | public: |
|  | CLine(SInput input); |
|  |  |
|  | int brightness; |
|  | double thickness; |
|  | SPoint start\_; |
|  | SPoint end\_; |
|  | double gamma\_ = 2.2; |
|  | }; |
|  |  |
|  |  |
|  | #endif //GRAPH\_CLINE\_H |

**CLine.cpp**

|  |
| --- |
| #include "CLine.h" |
|  | #include "CExpension.h" |
|  | CLine::CLine(SInput input) { |
|  | this->brightness = input.brightness; |
|  | this->start\_ = input.start; |
|  | this->end\_ = input.end; |
|  | this->thickness = input.thickness; |
|  | this->gamma\_ = input.gamma; |
|  | } |