МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

Лабораторная работа № 2

По дисциплине «Компьютерная графика и геометрия»

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

Выполнил студент группы М3101

Кузьмук Павел Юрьевич

Проверил:

Скаков Павел Сергеевич

САНКТ-ПЕТЕРБУРГ

2020

ЦЕЛЬ РАБОТЫ

Изучить алгоритмы растрирования векторов на существующем изображении и реализовать программу, рисующую линию на изображении в формате PGM (P5) с учетом гамма-коррекции.

ОПИСАНИЕ РАБОТЫ

Полное решение лабораторной работы.

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

Аргументы передаются через командную строку:

lab2 <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.

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:

C: 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

Рисование линий

Уравнение прямой линии:

$$y=\frac{y_1-y_0}{x_1-x_0}(x-x_0)+y_0$$

Существуют несколько видов алгоритмов:

1) Со сглаживанием
2) Без сглаживания

1. Алгоритм Брезенхема

Это простой алгоритм целочисленный, без сглаживания.

Псевдокод:

```
function line(int x0, int x1, int y0, int y1)
     int deltax := abs(x1 - x0)
     int deltay := abs(y1 - y0)
     int error := 0
     int deltaerr := (deltay + 1)
     int y := y0
     int diry := y1 - y0
     if diry > 0
         diry = 1
     if diry < 0
         diry = -1
     for x from x0 to x1
         plot(x,y)
         error := error + deltaerr
         if error >= (deltax + 1)
             y := y + diry
             error := error - (deltax + 1)
```

2. Алгоритм Ву

Этот алгоритм может работать с дробными координатами, со сглаживанием, но он относительно сложный по сравнению с алгоритмом Брезенхема.

Псевдокод:

```
function plot(x, y, c) is
    // рисует точку с координатами (x, y)
    // и яркостью c (где 0 ≤ c ≤ 1)

function ipart(x) is
    return целая часть от x

function round(x) is
    return ipart(x + 0.5) // округление до ближайшего целого

function fpart(x) is
    return дробная часть x

function draw_line(x1,y1,x2,y2) is
   if x2 < x1 then
       swap(x1, x2)
       swap(y1, y2)
   end if

   dx := x2 - x1
   dy := y2 - y1
   gradient := dy ÷ dx

   // обработать начальную точку
   xend := round(x1)
   yend := y1 + gradient × (xend - x1)
   xgap := 1 - fpart(x1 + 0.5)
   xpxl1 := xend  // будет использоваться в основном цикле
   ypxl1 := ipart(yend)
   plot(xpxl1, ypxl1, (1 - fpart(yend)) × xgap)
   plot(xpxl1, ypxl1 + 1, fpart(yend) × xgap)
   intery := yend + gradient // первое y-пересечение для цикла

   // обработать конечную точку
   xend := round(x2)
   yend := y2 + gradient × (xend - x2)
   xgap := fpart(x2 + 0.5)
   xpxl2 := xend  // будет использоваться в основном цикле
   ypxl2 := ipart(yend)
   plot(xpxl2, ypxl2, (1 - fpart(yend)) × xgap)
   plot(xpxl2, ypxl2 + 1, fpart(yend) × xgap)

   // основной цикл
   for x from xpxl1 + 1 to xpxl2 - 1 do
           plot(x, ipart(intery), 1 - fpart(intery))
           plot(x, ipart(intery) + 1, fpart(intery))
           intery := intery + gradient
   repeat
end function
```

3. Алгоритм, придуманный «на коленке», который удовлетворяет требованиям ЛР и дает лучшие результаты(и будет использован):

Абстрагируемся от изображения и координат пикселей, таким образом линия — просто прямоугольник какой-то толщины на плоскости, найдем координаты 4 его вершин. Для этого сначала найдем «медиану» - линия с бесконечно малой толщиной, проходящая через середины боковых сторон прямоугольника, потом через ее концы проведем перпендикуляры длиной thickness / 2, получим очертания линии, которую собираемся рисовать.

Пройдем по всем пикселям изображения, для каждого из них узнаем, какой процент от его площади покрыт прямоугольником. С такой яркостью поверх него мы и наложим цвет линии, которую рисуем. Для высчитывания площади пересечения двух прямоугольников(пикселя и линии) можно использовать геометрию, однако несложно с точки зрения процессорного времени просто разделить пиксель на 100 частей и проверить, сколько из них лежит в прямоугольнике — это и будет искомым процентом.

ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

Полное решение лабораторной работы.

Лабораторная работа выполнена на языке C++. Стандарт языка C++14.

Весь код может быть найден в публичном репозитории https://github.com/corelof/computer_graphics_labs

ВЫВОД

Выполнение данной лабораторной работы позволило узнать об алгоритмах рисования прямых линий со сглаживанием. В ходе данной лабораторной работы был реализован собственный алгоритм растрирования линий произвольной толщины.