МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____О. Г. Локтионова
«10» 02 2022 г.

АЛГОРИТМЫ РАСТЕРИЗАЦИИ ОТРЕЗКОВ

Методические указания по выполнению лабораторной работы
по дисциплине «Компьютерная графика»
для студентов всех форм обучения направления подготовки
09.03.04 «Программная инженерия»

Курск 2022

УДК 004.92
Составитель Е.А. Петрик

Рецензент
Кандидат технических наук, доцент Т.И.Лапина

Алгоритмы растеризации отрезков: методические указания по выполнению лабораторной работы / Юго-Зап. гос. ун-т; сост.: Е. А. Петрик. Курск, 2022. 12 с.: ил.4. Библиогр.: с.12.

Содержат краткие теоретические сведения об алгоритмах растеризации отрезков, а также приведены примеры и задания для лабораторной работы.

Методические указания соответствуют требованиям программы по направлению подготовки бакалавров: 09.03.04 «Программная инженерия»

Предназначены для студентов всех форм обучения направления подготовки бакалавров 09.03.04 «Программная инженерия»

Текст печатается в авторской редакции

Подписано в печать Формат 60х84 1/16.
Усл. печ. л. Уч. – изд. л. .Тираж экз. Заказ . Бесплатно.
Юго - Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Цель работы

Изучение алгоритмов растеризации отрезков, создание программы для визуализации работы алгоритмов.

Основные понятия

Дисплей накладывает технические ограничения на представление графической информации. Изображение представляется в виде точек и каждой точке соответствует вполне определенный атрибут - цвет и интенсивность. Таким образом, графические (а точнее - дисплейные) примитивы хранятся в памяти в виде совокупности образующих их точек, называемых *пикселями/*

Количество точек растра называется *разрешающей способность* (или разрешением) монитора.

В процессе построчного вывода изображение (кадр) помещается в битовую карту (видеопамять), которая организована по линейному принципу и расположена в адресном пространстве центрального процессора. Но прежде, чем будет сформирован растровый образ в видеопамяти, файл должен быть преобразован в соответствующий формат вывода. Чтобы преобразовать двумерный точечный образ, необходимо проанализировать и преобразовать значение каждой точки, что резко снижает производительность видеоподсистемы и системы в целом. Когда изображение может быть представлено в виде совокупности графических примитивов, т.е. в виде аналитического описания, каким и является векторный формат файлов, то процесс преобразования сводится к растеризации отдельных отрезков, окружностей, областей заливки и т.д. Следовательно, количество анализируемых и синтезируемых точек намного меньше, чем при обработке растровых изображений. К тому же, файлы векторного формата в среднем на порядок меньше по объему растровых файлов, поскольку последние хранят описание каждого пикселя и объем резко увеличивается с увеличением разрешения.

Базовые алгоритмы предназначены для реализации (растеризации) графических примитивов, таких как отрезки, окружности, эллипсы и т.д. (Графические примитивы, описанные на языке высокого уровня, - это векторная форма изображения.) Кроме того, в эту группу попадают такие операции машинной графики как

закраска сплошных областей и отсечение отрезков и многоугольников.

Необходимость разработки специальных алгоритмов построения геометрических объектов объясняется тем, что сами эти объекты имеют непрерывную (аналоговую) природу, а изображение строится, как правило, на экране растрового дисплея, то есть получаемое изображение носит дискретный характер. Это означает, например, что нельзя построить непосредственно отрезок, соединяющий две заданные точки экрана (см. рис.1). Процесс нахождения пикселей, наилучшим образом аппроксимирующих заданный отрезок, называется разложением отрезка в растр.

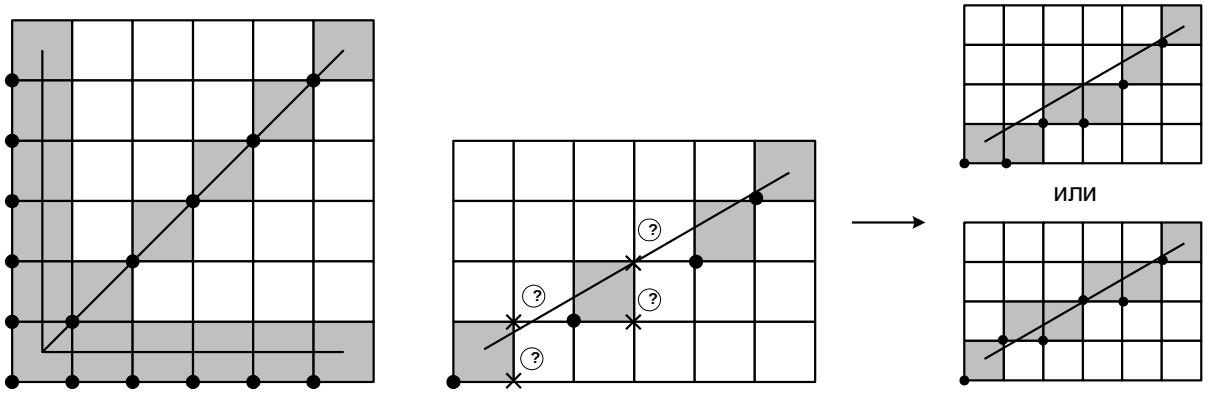


Рисунок 1 – Разложение отрезка в растр

Решение поставленной задачи очевидно лишь для трех типов отрезков: горизонтальных, вертикальных и наклоненных под углом в 45°.

Требования к алгоритмам растеризации отрезков

Во-первых, отрезки должны выглядеть прямыми; начинаться и заканчиваться в заданных точках. Во-вторых, яркость вдоль отрезка должна быть постоянной и не зависеть от длины и наклона. В-третьих, алгоритмы должны работать быстро.

Первое требование в силу дискретной природы растрового дисплея выполнено всегда быть не может. Можно лишь добиться того, что визуально (человеческим глазом) отрезок будет восприниматься прямым. Решение этой задачи может достигаться путем увеличения разрешающей способности экрана дисплея и применения методов устранения ступенчатости.

Второму требованию удовлетворяют также только горизонтальные, вертикальные и наклоненные под углом в 45° отрезки. Однако вертикальные и горизонтальные отрезки по сравнению с отрезками, расположенными под 45°, будут выглядеть ярче, так как расстояние между соседними пикселями у них меньше,

чем у наклонных отрезков. Обеспечение постоянной яркости вдоль отрезка требует высвечивания очередного пикселя яркостью, зависящей от расстояния между пикселями, вычисление которого производится с использованием операций извлечения квадратного корня и умножения. Использование этих операций существенно замедляет работу алгоритма, поэтому второе требование остается, как правило, невыполненным.

Удовлетворение третьего требования достигается путем сведения к минимуму вычислительных операций, использования операций над целочисленными данными, а также реализацией алгоритмов на аппаратном или микропрограммном уровне.

Цифровой дифференциальный анализатор. Многие алгоритмы вычерчивания отрезков и кривых используют пошаговый принцип, суть которого состоит в том, что результат вычисления координат высвечиваемого пикселя зависит от результатов, полученных на предыдущем шаге. Алгоритм вычерчивания отрезков по методу цифрового дифференциального анализатора, кроме этого, основан также на достаточно общем принципе, известном в математике: изучение какого-либо явления на основе дифференциального уравнения или системы таких уравнений, описывающей это явление.

Поскольку прямая линия на плоскости описывается уравнением вида

$ax + by + c = 0,$

где a,b,c - коэффициенты этого уравнения, то производная dy/dx является постоянной.

Заменив дифференциалы конечными разностями, получим

$$\frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1},$$

где x_1y_1 и x_2y_2 – координаты начальной и конечной точек отрезка.

Ордината очередного пикселя y_{i+1} может быть вычислена по известной ординате предыдущего пикселя y_i следующим образом:

$y_{i+1} = y_i + \Delta y$

Подставляя Δy, получим

$$y_{i+1} = y_i + \frac{y_2 - y_1}{x_2 - x_1}\Delta x.$$

Остается определить величину приращения Δx. Как правило, большее из приращений (Δx или Δy) выбирается в качестве единицы растра, а приращение вдоль другой координатной оси подлежит определению. Если же поступить по-другому (меньшее из приращений взять равным единице), то отрезок на экране может получиться "дырявым", то есть состоящим из отдельных точек, не расположенных вплотную друг к другу.

Алгоритм разложения отрезка в растр по методу ЦДА может быть записан следующим образом:

1. Ввод исходных данных x_1y_1, x_2y_2.

2. Проверка вырожденности отрезка. Если отрезок вырожден, то высвечивание точки и переход к п.7.

3. Вычисление $L:=|x_2-x_1|$, если $|x_2-x_1| \geq |y_2-y_1|$
$|y_2-y_1|$, если $|y_2-y_1| > |x_2-x_1|$

4. Вычисление $dx:=(x_2-x_1)/L$, $dy:=(y_2-y_1)/L$.

5. Задание координатам текущей точки начальных значений:

$x := x_1 + 0.5 \times Sign(dx)$, $y := y_1 + 0.5 \times Sign(dy)$,

где Sign - функция, возвращающая -1,0,1 для отрицательного, нулевого и положительного аргумента соответственно, использование этой функции делает алгоритм пригодным для всех квадрантов.

6. Цикл от i:=1 до i=L+1 с шагом 1:

высвечивание точки с текущими координатами (E(x),E(y)), где E - операция округления до ближайшего меньшего целого;

вычисление координат следующей точки:

x:=x+dx, y:=y+dy.

7. Конец.

На рис.2 представлена блок-схема алгоритма ЦДА.

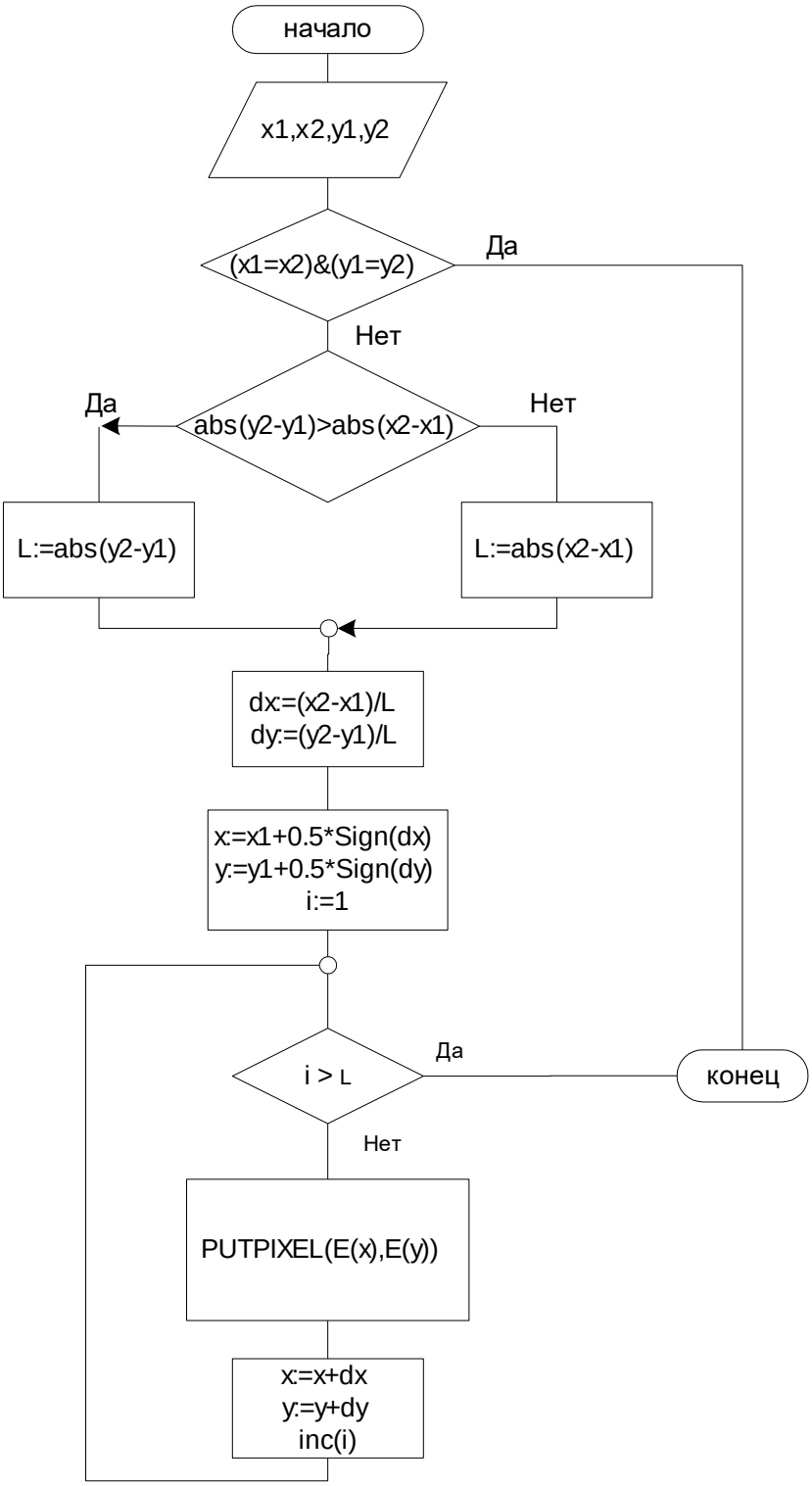
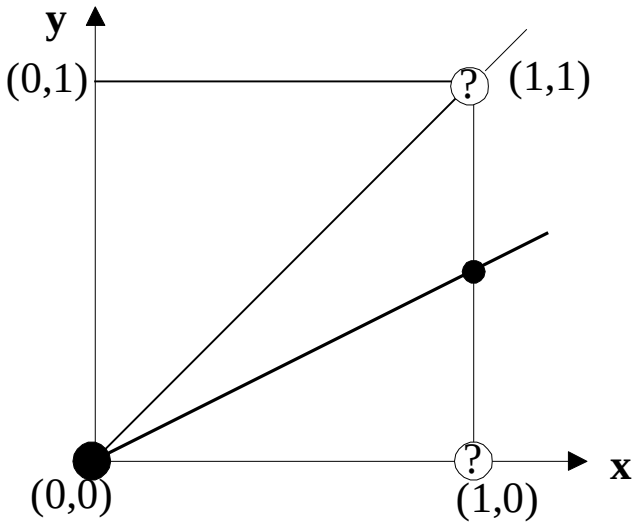


Рисунок 2 – Алгоритм ЦДА

Вещественный алгоритм Брезенхема

Работа алгоритма Брезенхема основывается на использовании понятия ошибка. Ошибкой здесь называется расстояние между действительным положением отрезка и ближайшим пикселем сетки растра, который аппроксимирует отрезок на очередном шаге.

На каждом шаге вычисляется величина ошибки и в зависимости от полученного значения выбирается пиксель, ближе расположенный к идеальному отрезку. Поскольку при реализации алгоритма на ЭВМ удобнее анализировать не само значение ошибки, а ее знак, то истинное значение ошибки смещается на -0,5.



$0.5 \le \Delta y/\Delta x \le 1$ ($f \ge 0$)

$0 \le \Delta y/\Delta x < 0.5$ ($f < 0$)

Инициализировать f в -0.5

$f := f + \Delta y/\Delta x$

Рисунок 3 – Основная идея алгоритма Брезенхема

Пусть f_i - значение ошибки на очередном i-м шаге, $Y_и$ - ордината идеального отрезка, а Y_i - ордината пикселя, выбранного для аппроксимации отрезка на том же i-м шаге, m тангенс угла наклона отрезка.Поскольку на первом шаге высвечивается пиксель с начальными координатами, то для него f=0, поэтому задаваемое предварительно значение $f = m - 0.5$ является фактически ошибкой для следующего шага.

Ошибка на очередном шаге вычисляется как

$f_{i+1}=Y_{иi+1}-Y_{i+1}=Y_{иi}+m-Y_{i+1}=f_i+m$ (если $Y_{i+1}=Y_i$).

(Поскольку в алгоритме и программе не надо сохранять значения ошибок для всех шагов, то последнее выражение можно записать как f=f+m).

В зависимости от полученного значения ошибки выбирается пиксель с той же ординатой (при f<0) или пиксель с ординатой, на единицу большей, чем у предыдущего пикселя (при f≥0).

Поскольку предварительное значение ошибки вычисляется заранее, то есть f+m вычислено на предыдущем шаге, то во втором случае останется только вычесть единицу из значения ошибки:

f=f-1.

Алгоритм Брезенхема, работающий с действительными величинами, можно записать в следующем виде:

1. Ввод исходных данных Xн,Yн,Xк,Yк.

2. Проверка вырожденности отрезка. Если отрезок вырожденный, то высвечивается точка и осуществляется переход к п.10.

3. Вычисление приращений dX:=Xк-Xн и dY:=Yк-Yн.

4. Вычисление шага изменения каждой координаты пиксела:

SX:=sign(dX), SY:=sign(dY).

5. Вычисление модулей приращения координат:

dX:=|dX|, dY:=|dY|

6. Обмен значений dX и dY в зависимости от углового коэффициента наклона отрезка:

если dY > dX, то выполнить

Temp:=dX, dX:=dY, dY:=Temp, flag:=1;

иначе flag:=0 (flag - флаг, определяющий факт обмена местами координат).

7. Инициализация начального значения ошибки:

f:= dY/dX -0,5 *(для целочисленного алгоритма: fц=2dY-dX)*

8. Инициализация начальных значений координат текущего пиксела:

X:=Xн, Y:=Yн

9. Цикл - пока X≤ Xк делать:

9.1. Высвечивание точки с координатами (X,Y).

9.2. Вычисление координат и корректировка ошибки для следующего пиксела:

Если f≥0, то если flag:=1, то X:=X+SX иначе Y:=Y+SY;

корректировка ошибки f:=f-1 *(fц=fц-2dX)*

Если f<0, то если flag:=1, то Y:=Y+SY иначе X:=X+SX.

9.3. Вычисление ошибки f:=f+ dY/dX (*fц=fц+2dY)*.

10. Конец.

Целочисленный алгоритм Брезенхема

Алгоритм Брезенхема в том виде, как он представлен выше, требует использования арифметики с плавающей точкой и деления(для вычисления углового коэффициента и оценки ошибки).

Быстродействие алгоритма можно увеличить, если использовать только целочисленную арифметику и исключить деление. Для этого выражение (2.3) запишем в виде: f = dY/dX-1/2 и, умножив обе части этого равенства на 2·dX, получим:

2·dX·f = 2·dY - dX.

Обозначив fц=2·dX·f, окончательно получим:

fц = 2·dY - dX (2.5)

(В соответствии с этим выражением должно вычисляться теперь начальное значение ошибки в п.7 алгоритма).

Тогда подсчет нового значения ошибки в п.9 "действительного" алгоритма будет производиться по формулам:

fц = fц + 2·dY и fц = fц - 2·dX.

Оптимизация алгоритма Брезенхема–прорисовка линии по отрезкам переменной длины. Идея алгоритма Брезенхема – это движение вдоль основной оси по одному пикселю и поддержание текущего отклонения от идеальной прямой в заданных пределах. Если отклонение превышает норму, то делается шаг по неосновной оси, чтобы уменьшить отклонение. Главный момент здесь – это то, что вычисление и проверка производится на каждом шаге.

Для контроля размещения точек необходимо периодически принимать решения относительно их расстановки и делать это желательно как можно быстрее. Можно принимать решения для каждой точки, как в стандартном алгоритме Брезенхема, но большинство проверок в этом случае излишни.

Идея оптимизации (алгоритм с переменной длиной отрезков) – это движение с каждой итерацией по неосновной оси и проверка текущего отклонения горизонтальной координаты.

Посмотрим на задачу так: когда программа проводит линию в 35 пикселей по X и 10 пикселей по Y, то уже есть много информации, часть которой игнорируется стандартным алгоритмом Брезенхема. Например, поскольку наклон линии находится в пределах 1/3-1/4, то уже можно определенно сказать, что составляющие линию ряды точек состоять из 3 или 4 пикселей. Вычисления становятся <на 70%.

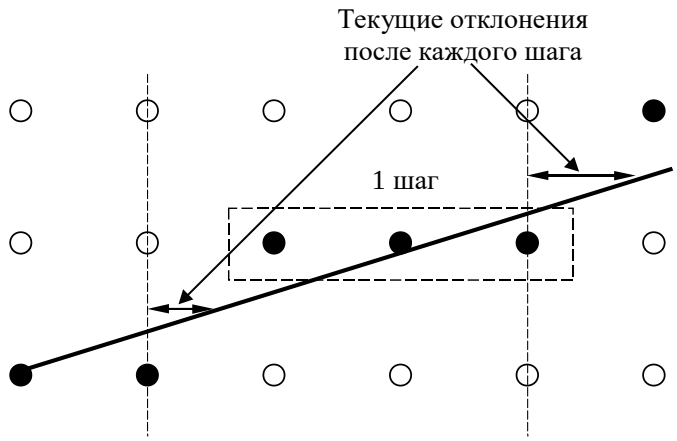


Рисунок 4 – Алгоритм с переменной длиной отрезков

Задание

Написать программу (на языке высокого уровня), реализующую алгоритмы растеризации отрезков с их последующей отрисовкой. Координаты отрезка должны задаваться пользователем. Реализовать:

- алгоритм ЦДА;
- алгоритм Брезенхема;
- целочисленный алгоритм Брезенхема.

Содержание отчета

Отчет должен содержать:

1. Титульный лист.
2. Задание.
3. Блок-схемы алгоритмов.
4. Листинг программы.
5. Пример работы программы.
6. Ответы на контрольные вопросы.

Контрольные вопросы

1. Что такое пиксель?
2. Что такое растровое изображение?
3. Для чего нужны алгоритмы растеризации отрезков?
4. Какие ещё алгоритмы растеризации отрезков существуют?

Список используемой литературы

1. Аммерал, Л. Принципы программирования в машинной графике / Л. Аммерал; пер. с англ. – Москва : Сол Систем, 1992. – 224 с. – ISBN 5-85316-001-X. – Текст : непосредственный.
2. Роджерс, Д. Алгоритмические основы машинной графики / Д. Роджерс; пер. с англ. – Москва : Мир, 1989. – 512 с. – ISBN 5-03-000476-9. – Текст : непосредственный.
3. Роджерс, Д. Математические основы машинной графики / Д. Роджерс, Дж. Адамс; пер. с англ. – Москва : Мир, 2001. – 604 с. – ISBN 5-03-002143-4. – Текст : непосредственный.
4. Шикин, Е. В. Начала компьютерной графики / Е. В. Шикин, А. В. Боресков, А. А. Зайцев. – Москва : ДИАЛОГ-МИФИ, 1993. – 138 с. – ISBN 5-86404-035-5. – Текст : непосредственный.