­­­Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

Новосибирский государственный технически университет

*Факультет автоматики и вычислительной техники*

Лабораторная работа №1

по дисциплине «Компьютерная и инженерная графика»

***«Генерация векторов »***

Студент: Богомолов Д.А.

Группа: АВТ-310

Преподаватель: Дружинин А. И.

Новосибирск

2014

***Цель:***

Изучить алгоритмы разложения отрезка в растр: алгоритм цифрового дифференциального анализатора (ЦДА), алгоритм Брезенхема. Сравнить результаты применения алгоритмов между собой.

***Задание:***

1. Центр отрезка находится в середине экрана.

1.1. Разложить в растр отрезок по методу цифрового дифференциального анализатора (ЦДА) с произвольными координатами.

1.2. Разложите в растр отрезок, используя алгоритм Брезенхема с произвольными координатами.

2. Используя разные цвета, разложите в растр отрезок на одной системе координат с помощью метода ЦДА и Брезенхема. Сравните полученные результаты. Сделайте выводы.

3. Нарисуйте "настоящий" отрезок и сравните результаты работы алгоритмов.

4. Сравните алгоритмы ЦДА и Брезенхема. Приведите примеры их сходства и различия.

**Инструкция по применению:**

1. Ввести координаты начала и координаты конца отрезка в специально предназначенные для этого поля. Ориентироваться относительно осей координат, обозначенными на сетке чёрными жирными линиями. 3. Для выполнения алгоритма ЦДА (отрезок из зелёных пикселей) отметить галочку «ЦДА». Для выполнения алгоритма Брезенхема (отрезок из синих пикселей) отметить галочку «Брезенхем». Для построения реального отрезка отметить «Нарисовать вектор». Можно комбинировать одновременное построение графика с построением растра.

***Теоретический материал:***

Назначение генератора векторов – соединение двух точек изображения отрезком прямой. Далее будут рассмотрены следующие алгоритмы:

* алгоритм ЦДА – цифрового дифференциального анализатора (DDA – Digital Differential Analyser) для генерации векторов – симметричный и несимметричный;
* алгоритм Брезенхема для генерации векторов.

Перед рассмотрением конкретных алгоритмов сформулируем общие требования к изображению отрезка:

* концы отрезка должны находиться в заданных точках;
* отрезки должны выглядеть прямыми;
* яркость вдоль отрезка должна быть постоянной и не зависеть от длины и наклона;
* построение должно быть быстрое.

Ни одно из этих условий не может быть точно выполнено на растровом дисплее в силу того, что изображение строится из пикселов конечных размеров, а именно:

* концы отрезка в общем случае располагаются на пикселах, лишь наиболее близких к требуемым позициям и только в частных случаях координаты концов отрезка точно совпадают с координатами пикселов;
* отрезок аппроксимируется набором пикселов и лишь в частных случаях вертикальных, горизонтальных и отрезков под 45° они будут выглядеть прямыми, причем гладкими прямыми, без ступенек только для вертикальных и горизонтальных отрезков (рис. 1.1);
* яркость для различных отрезков и даже вдоль отрезка в общем случае различна, так как, например, расстояние между центрами пикселов для вертикального отрезка и отрезка под углом 45° различно (рис. 1.1).

*Рис. 1.1*. Растровое представление различных векторов

## 1.1. Цифровой дифференциальный анализатор (ЦДА)

С помощью алгоритма ЦДА решается дифференциальное уравнение отрезка, имеющее вид:

,

где  *–* приращение координат отрезка по оси *Y*;  – приращение координат отрезка по оси *X*; *X*2, *Y*2, *X*1,*Y*1 – начальная и конечная координаты отрезка.

При этом ЦДА формирует дискретную аппроксимацию непрерывного решения этого дифференциального уравнения. В симметричном ЦДА тем или иным образом определяется количество узлов *N*, используемых для аппроксимации отрезка. Затем за *N* циклов вычисляются координаты очередных узлов:

,

.

Получаемые значения  и  преобразуются в целочисленные значения координаты очередного подсвечиваемого пиксела. Генератор векторов, использующий этот алгоритм, имеет тот недостаток, что точки могут прописываться неоднократно, что увеличивает время построения, или отрезок будет выглядеть прерывистым.

Кроме того, из-за независимого вычисления обеих координат нет предпочтительных направлений и построенные отрезки кажутся не очень красивыми. Субъективно лучше смотрятся вектора с единичным шагом по большей относительной координате (несимметричный ЦДА). Для  (при ) это означает, что координата по *X* направлению должна увеличиться на 1 *Px* раз, а координата по *Y* – направлению должна также *Px*раз увеличиться, но на . Т.е. количество узлов аппроксимации берется равным числу пикселов вдоль наибольшего приращения. Для генерации отрезка из точки (*X*1*,Y*1) в точку (*X*2*,Y*2) в первом октанте алгоритм несимметричного ЦДА имеет вид:

*x – integer*;

*Вычислить приращения координат:  
*; **;

*Занести начальную точку отрезка  
PutPixel* (*x*1*,y*1);

*Сгенерировать отрезок  
while* (*x*1 *< x*2) { *x*1*:= x*1 *+* 1; *y*1*:= y*1 *+ Py/Px*; *PutPixel* (*intl* (x1), *intl* (*y*1));}

*Примечание*: intl – приближение к нижнему, т.е. 5.3 = 5; 5.6 = 5; –5.3 = –6; –5.6 = –6.

Пример генерации отрезка по алгоритму несимметричного ЦДА приведен на рис. 1.2.

 – Вычисленные позиции пикселов

– Очередной пиксел

*Рис. 1.2.* Генерация отрезка несимметричным ЦДА

Генерация отрезка:

*if* (*abs* (*x*2 *– x*1) >= *abs* (*y*2 *– y*1)) *then*

*len:= abs* (*x*2 *– x*1)  *else*

*len:= abs* (*y*2 *– y*1);

**; **;

;

;

*i =* 1

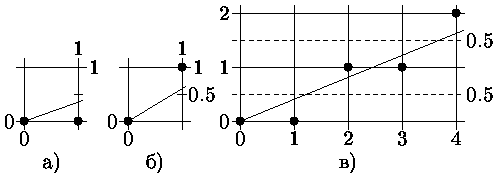
*while* (*i <= len*) { *PutPixel* (*intl* (*x*)*, intl* (*y*)); *x:= x + Px*; *y:= y+Py*; *i:= i +* 1

}

## 1.2. Алгоритм Брезенхема

Так как приращения координат, как правило, не являются целой степенью двойки, то в ЦДА – алгоритме (см. предыдущий пункт) требуется выполнение деления, что не всегда желательно. Алгоритм выбирает оптимальные растровые координаты для представления отрезка. В процессе работы одна из координат – либо *X*, либо *Y* (в зависимости от углового коэффициента) – изменяется на единицу, изменение другой координаты (либо на нуль, либо на единицу) зависит от расстояния между действительным положением отрезка и ближайшими координатами сетки. Такое расстояние назовем *ошибкой*.

Брезенхем предложил алгоритм, построенный так, что требуется проверять лишь знак этой *ошибки*. На рис. 1.3 а,б) это иллюстрируется для отрезка в первом октанте, т.е. для отрезка с угловым коэффициентом, лежащим в диапазоне от нуля до единицы. Из рисунка можно заметить, что если угловой коэффициент отрезка из точки (0,0) больше чем 1/2, то его пересечение с прямой  будет расположено ближе к прямой , чем к прямой . Следовательно, точка растра (1,1) лучше апроксимирует ход отрезка, чем точка (1,0). Если угловой коэффициент меньше 1/2, то верно обратное. Для углового коэффициента, равного 1/2 нет какого-либо предпочтительного выбора. В данном случае алгоритм выбирает точку (1,1). Не все отрезки проходят через точки растра (рис. 1.3 в). Здесь отрезок с тангенсом угла наклона 3/8 сначала проходит через точку растра (0,0) и последовательно пересекает три пиксела. Так как желательно проверять только знак ошибки, то она первоначально устанавливается равной – 1/2.



*Рис. 1.3*. Алгоритм Брезенхема генерации отрезков

Если *Е* < 0 (где *Е* – ошибка), то *Y*-координата не меняется по сравнению с предыдущей точкой. В противном случае *Y* увеличивается на 1.

Для вычисления *Е* положим, что рассматриваемый вектор начинается в точке (0,0) и проходит через точку (4, 1.6) (рис. 1.3 в), т.е. имеет положительный наклон, меньший 1.

Из рис. [1.3](http://ermak.cs.nstu.ru/kg_rivs/kg02.htm#r03_03) в) видно, отклонение для первого шага:

,

поэтому для занесения пиксела выбирается точка (1,0).

Отклонение для второго шага вычисляется добавлением приращения *Y*-координаты для следующей *X*-позиции (рис. 1.3 в):

,

поэтому для занесения пиксела выбирается точка (2,1). Так как отклонение считается от *Y*-координаты, которая теперь увеличилась на 1, то из накопленного отклонения для вычисления последующих отклонений надо вычесть 1:

*E*2 = *E*2 – 1

Отклонение для третьего шага:

,

поэтому для занесения пиксела выбирается точка (3,1).

Возможны случаи:

|  |  |
| --- | --- |
| *E*1 > 0 | *E*1 <= 0 |
| ближайшая точка есть: | |
| *X*1 *= X*0 *+* 1; *Y*1 *= Y*0+1 | X1 = X0 + 1; Y1 = Y0 |
| *E*2 = *E*1 + *Py*/*Px* – 1 | *E*2 = *E*1 + *Py*/*Px* |

Так как интересует только знак *Е*, то можно избавиться от деления умножением *E* на 2*\*Px*:

|  |  |
| --- | --- |
|  | *E*1 *=* 2*\* Py – Px* |
| *E*1 > 0 | *E*2 *= E*1 *+* 2 (*Py – Px*) |
| *E*1 <= 0 | *E*2 *= E*1 *+* 2*\* Py* |

Таким образом получается алгоритм, в котором используются только целые числа, сложение (умножение на 2 – сдвиг влево или сложение с самими числом):

*X = x*1; *Y = y*1; *Px = x*2 *– x*1; *Py = y*2 *– y*1; *E =* 2*\*Py – Px*; *i = Px*; *PutPixel*(*X, Y*); */\* Первая точка вектора \*/  
while* (*i= i –* 1≥ *0*){ *if* (*E  0*){ *X= X +* 1; *Y= Y +* 1; *E= E +* 2*\**(*Py – Px*);} *else  
X= X +* 1; *E= E +* 2*\*Py*; *PutPixel*(*X, Y*); */\* Очередная точка вектора \*/*}

Этот алгоритм пригоден для случая *0  dY  dX.* Для других случаев алгоритм строится аналогичным образом.

На рис. 1.4 приведен пример генерации по алгоритму Брезенхема того же самого отрезка, что и показанного на рис. 1[.2](http://ermak.cs.nstu.ru/kg_rivs/kg02.htm#r03_02) для генерации по алгоритму несимметричного ЦДА.

 – Вычисленные позиции пикселов

– Очередной пиксел

0 1 2 3 4 5 6 7 8

4

3

2

1

*Рис. 1.4.* Генерация отрезка по алгоритму Брезенхема

***Полученные результаты:***

