Компьютерная графика

Лекция 3

Растровая графика.

Генерация линий.

Растровые алгоритмы

- В большинстве случаев графические устройства, такие как монитор или принтер являются растровыми, то есть представляют изображение в виде прямоугольной сетки пикселов.
- Поэтому большое место в компьютерной графике уделяется алгоритмам, которые рисуют на пиксельной сетке различные графические объекты.
- Такие алгоритмы называются растровыми.

Понятие связности

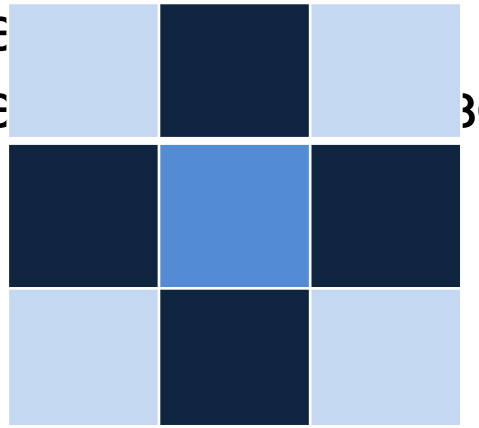
- Важным понятием для растровой сетки является понятие связности - возможность соединения двух пикселов растровой линией, т.е. последовательным набором пикселов.
- При этом возникает вопрос, когда пикселы (x_1, y_1) и (x_2, y_2) можно считать соседними.
- В этом вопросе используют два подхода.

4-связность и 8-связность

4-связность – пикселы считаются соседними при выполнении условия

$$|x_1-x_2|+|y_1-y_2|\leq 1$$

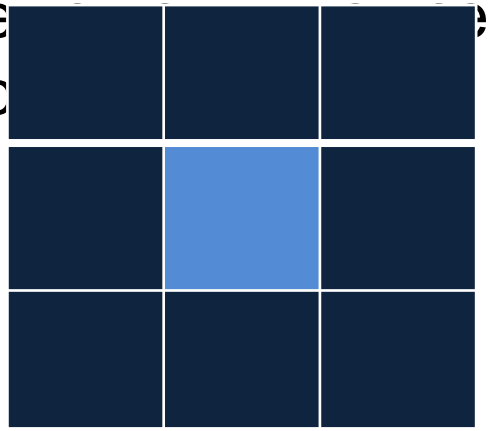
т.е. возможны пе… ве… …зонтали.



8-связность – пикселы считаются соседними при выполнении условия

$$|x_1-x_2|\leq 1 \;\;\&\;\; |y_1-y_2|\leq 1$$

т.е. возможны пере… …ем 8 напр…



Построение отрезков

Рассмотрим следующую задачу. Требуется построить на пиксельной сетке (растровое) изображение отрезка, соединяющего точки (x_1,y_1) и (x_2,y_2).
Стандартными требованиями к алгоритмам вычерчивания линий являются следующие:

- Отрезки должны выглядеть прямыми и начинаться и заканчиваться в заданных точках;
- Яркость должна быть постоянна и независима от наклона;
- Скорость рисования должна быть максимальна.

Цифровой Дифференциальный Анализатор (ЦДА)

Один из методов разложения отрезка в растр состоит в решении дифференциального уравнения, описывающего этот процесс.

Для прямой линии имеем

$dy / dx = const$ или $\Delta y / \Delta x = (y_2 - y_1) / (x_2 - x_1)$

Решение предсталяется в виде

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

$y_{i+1} = y_i + \Delta x (y_2 - y_1) / (x_2 - x_1)$

В простом ЦДА либо Δx, либо Δy (большее из приращений) выбирается в качестве единицы растра.

Реализация ЦДА

```
Len = max(abs(x2 - x1), abs(y2 - y1));
dx = (x2 - x1) / Len; dy = (y2 - y1) / Len;
x = x1 + 0.5 * Sign(dx);  y = y1 + 0.5 * Sign(dy);
for (int i=0; i<Len; ++i)
{   SetPixel(hdc, x, y, 0);
    x+=dx; y+=dy;
}
```

Здесь **Sign** - функция, возвращающая -1, 0, 1 для отрицательного, нулевого и положительного аргумента соответственно.

Пример работы ЦДА

Рассмотрим работу ЦДА для отрезка (0,0) (-8,-4)

x1 = 0

y1 = 0

x2 = -8

y2 = -4

Len = 8

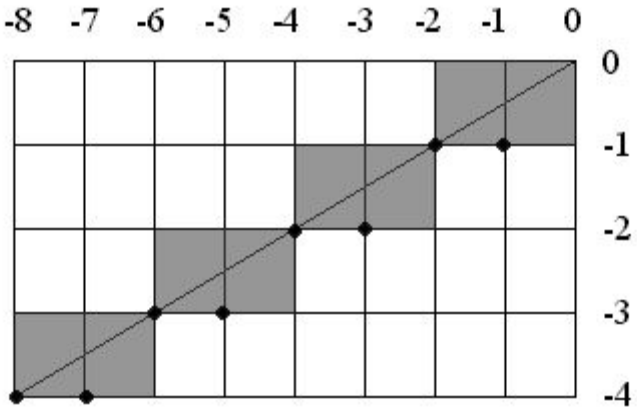
dx = -1

dy = -0,5

x = -0,5

y = -0,5

i	pixel	x	y
		-0.5	-0.5
0	(-1, -1)	-1.5	-1.0
1	(-2, -1)	-2.5	-1.5
2	(-3, -2)	-3.5	-2.0
3	(-4, -2)	-4.5	-2.5
4	(-5, -3)	-5.5	-3.0
5	(-6, -3)	-6.5	-3.5
6	(-7, -4)	-7.5	-4.0
7	(-8, -4)	-8.5	-4.5



Недостатки:

отсутствие симметрии,

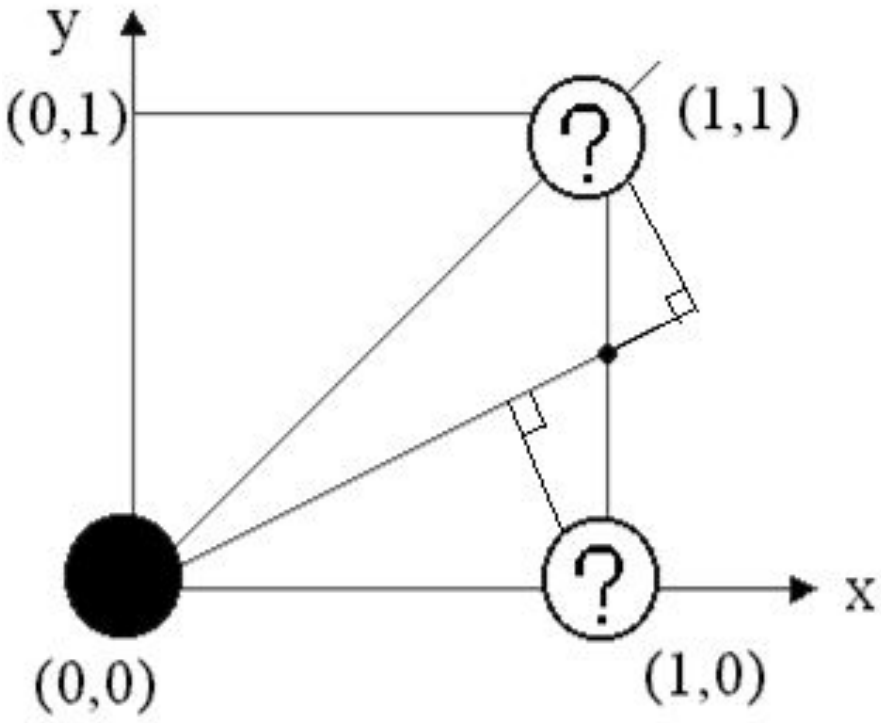
все точки лежат с одной стороны от реального отрезка.

Алгоритм Брезенхема

- Этот алгоритм, разработанный Джеком Е. Брезенхэмом (Jack E. Bresenham) в 1962 году в компании IBM, является одним из самых старых алгоритмов в компьютерной графике.
- Он позволяет получить приближение идеальной прямой точками растровой сетки.
- Он является итерационным. Каждый раз мы переходим к тому из соседних пикселов, расстояние от которого до реального отрезка минимально.

Шаг алгоритма Брезенхема

Рассмотрим случай, когда dx>dy, тогда при построении 8-связной линии можно в качестве соседних точек брать только две справа



Теоретически требуется рассматривать расстояния до прямой, но за счет подобия треугольников можно контролировать только вертикальные смещения.

Работаем с величиной ошибки.

Реализация алгоритма Брезенхема

```
void LineBrez(HDC hdc, int x1, int y1, int x2, int y2)
{   int  dx=x2-x1, dy=y2-y1, x=x1, y=y1;
    double D=double(dy)/dx,  e=0;
    SetPixel(hdc, x, y, 0);
    for (x++; x<x2; x++)
    {e+=D;
     if (e>0.5) {e--; y++;}
     SetPixel(hdc, x, y, 0);
    }
}
```

Что можно улучшить

- В процессе работы алгоритма наблюдаются вычисления с вещественными величинами.
- Заметим, что они касаются только переменных D и e. При этом, эти переменные не связаны непосредственно с координатами x и y.
- В знаменателях вещественных переменных имеем dx и 2. Умножим на них все, что связано с D и e.
- Получим целочисленный вариант

Реализация целочисленного алгоритма

```
void LineBrez2(HDC hdc, int x1, int y1, int x2, int y2)
{   int dx=x2-x1, dy=y2-y1, x=x1, y=y1;
    int D=dy*2,  e=-dx;
    SetPixel(hdc, x, y, 0);
    for (x++; x<x2; x++)
    {e+=D;
     if (e>0) {e-=2*dx; y++;}
     SetPixel(hdc, x, y, 0);
    }
}
```

Алгоритм для произвольной прямой

- Можно избавиться от предположения о том, что угол наклона прямой от 0 до π/4.
- При больших углах необходимо на каждом шаге цикла менять координату y, подбирая при этом x исходя из величины ошибки.
- Необходимо рассматривать направление движения по x и по y в зависимости от знаков dx и dy.
- В итоге имеем целочисленный алгоритм.

Резюме по алгоритму Брезенхема для прямой

- Алгоритм простой для реализации
- Все вычисления в целочисленной арифметике
- Допускает аппаратную реализацию
- Дает идеальное растровое приближение реальной прямой
- Легко модифицируется для случаев 4-связной и 8-связной линии

ЗАДАЧИ

- Задано окно координатами своих вершин. В нем заданы две точки. Нарисовать отрезок их соединяющий методом цифрового дифференциального анализатора.
- В условиях предыдущей задачи нарисовать отрезок методом Брезенхема.
- Визуализировать процесс построения отрезка методом ЦДА. Нарисовать сетку пикселей и показывать каждый шаг алгоритма с возможным и реальным выбором следующего пикселя. Использовать связность типа 4.
- Визуализировать процесс построения отрезка методом ЦДА. Нарисовать сетку пикселей и показывать каждый шаг алгоритма с возможным и реальным выбором следующего пикселя. Использовать связность типа 8.
- Визуализировать процесс построения отрезка методом Брезенхема. Нарисовать сетку пикселей и показывать каждый шаг алгоритма с возможным и реальным выбором следующего пикселя. Использовать связность типа 4 (8).

ТЕСТЫ

- Связность может быть типа

a) 2
b) 3
c) 6
d) 8

- Цифровой дифференциальный анализатор -

a) Это метод разложения отрезка в растр путем решения дифференциального уравнения
b) Это решение задачи Коши численными методами
c) Анализ решения дифференциального уравнения
d) Метод построения произвольной линии

- Алгоритм Брезенхема построения прямой лучше метода ЦДА ,так как

a) позволяет получить приближение идеальной прямой точками растровой сетки
b) все точки лежат с одной стороны от реального отрезка
c) можно использовать связность типа 8
d) не требуется решать дифференциальное уравнение

ТЕСТЫ

- Связность

a) это возможность соединения двух пикселов растровой линией
b) возможность вставить между двумя пикселами несколько других
c) непрерывная последовательность нескольких пикселов

- Общие принципы алгоритма Брезенхема

a) целочисленный итерационный алгоритм получения минимальных расстояний от прямой до реальных пикселов
b) целочисленный итерационный алгоритм, не допускающий аппаратную реализацию, получения минимальных расстояний от прямой до реальных пикселов
c) целочисленный алгоритм построения построения 8-связной линии

- Стандартные требования к алгоритмам вычерчивания линий

a) Скорость рисования должна быть максимальна, а яркость зависит от наклона линии
b) Отрезки должны выглядеть прямыми и начинаться и заканчиваться в заданных точках
c) Скорость рисования должна быть максимальна, яркость постоянна, отрезки не обязательно выглядят прямыми