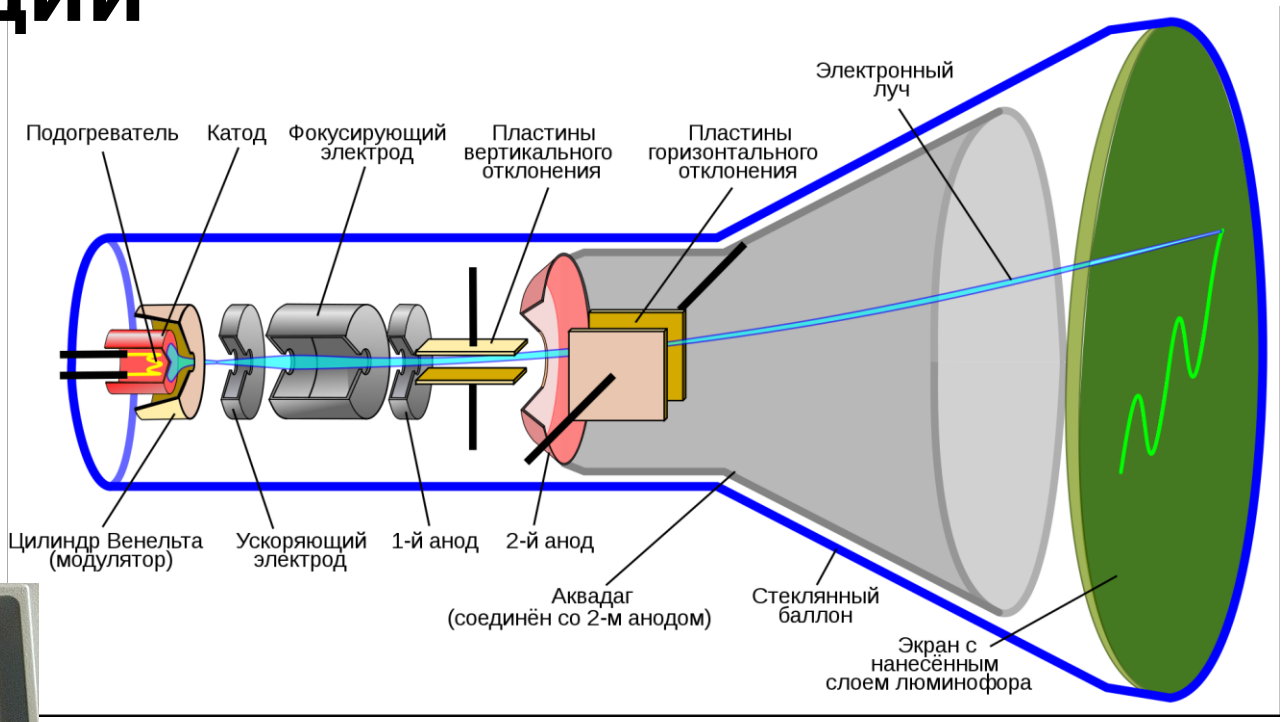
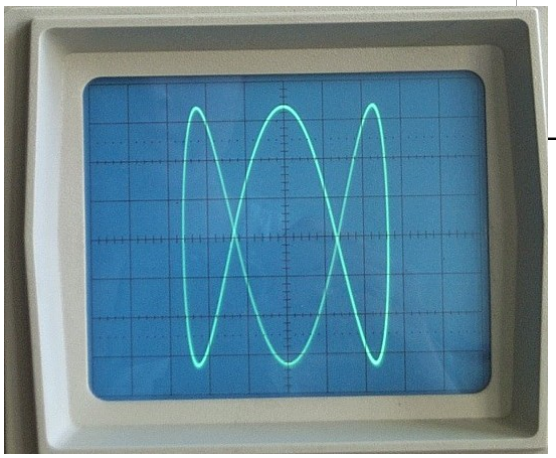
Компьютерная Графика

Растрирование

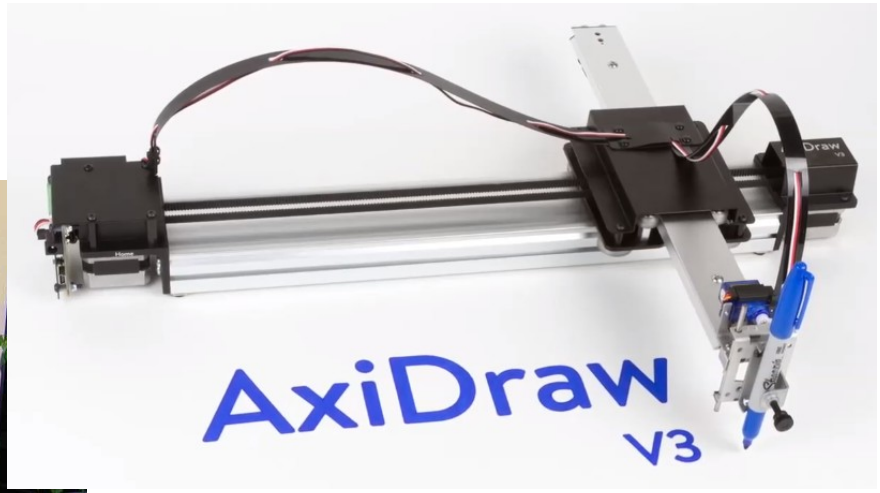
Отображение графической информации





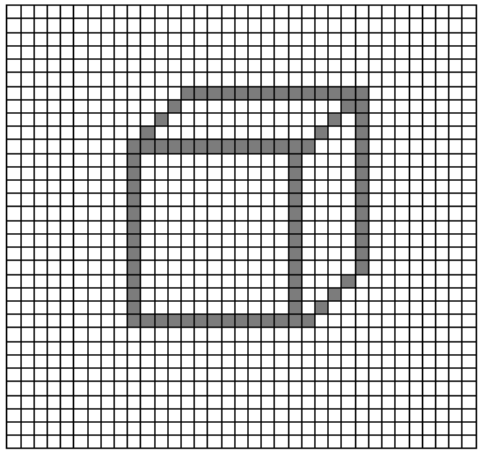
Электронно-лучевая трубка

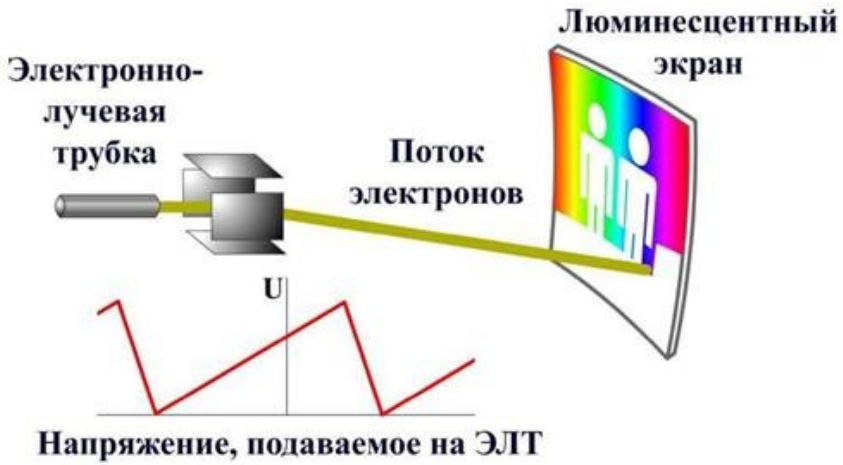
Отображение графической информации



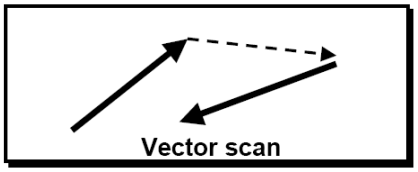
Графопостроители

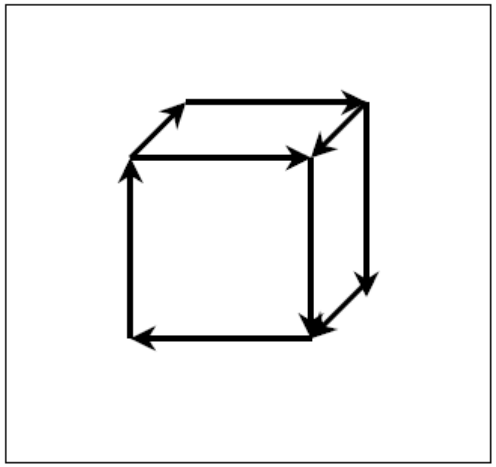
Отображение графической информации





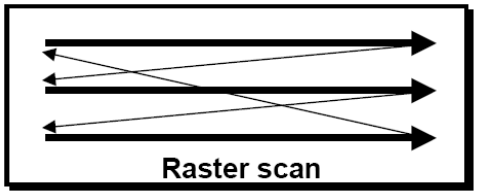
Телевизор (построчная развертка луча)

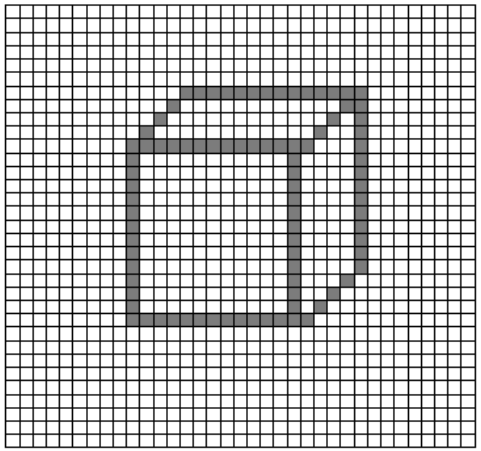




Векторное

- Произвольное разрешение
- Удобно для чертежей
- Можно зарисовать точку дважды
- Трудно делать изменения цвета
- Дисплейный файл
- Частота обновления плавает





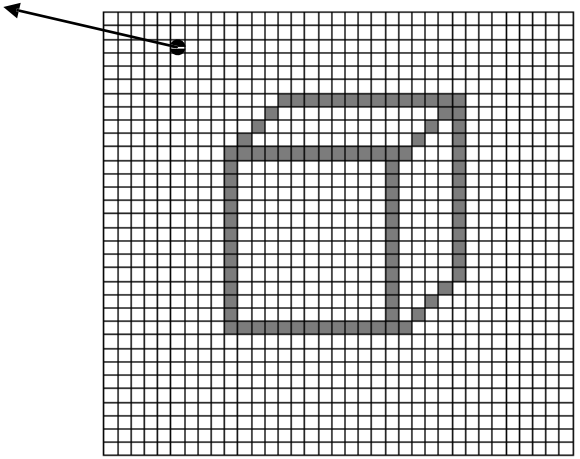
Растровое

- Ограниченный размер
- Удобно для цветных фото
- Общеприменимо
- Ограниченное разрешение (алиасинг)
- Запоминается каждый пиксель
- Частота обновления фиксированная

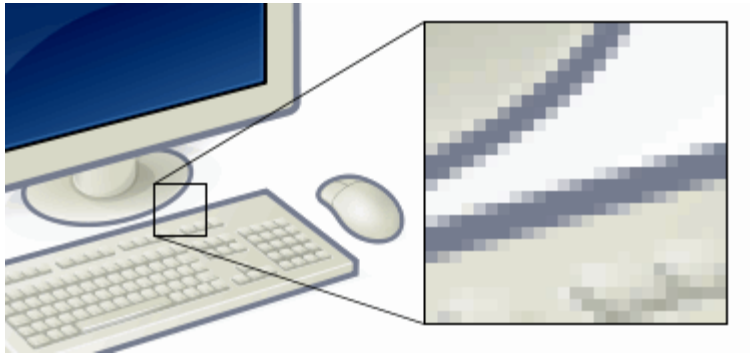
Дисплей и буфер кадра

Изображение – это растр = 2D массив пикселей

`int FB[n][m] - frame buffer  FB[5][2]`



Пиксель (pixel, picture element) мельчайший элемент изображения



Типы пикселей:

- Черно-белый – 1 бит/пиксель
- Монохромные – 2, 4, 8, 12 бит
- Цветные – 2,4,8,15,16,24,32,48,96 б/п
- Супер – 96 б/п

Палитра

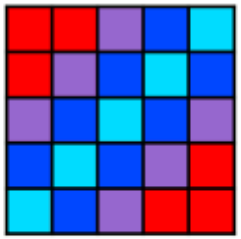
```
const int n=1<<depth;
RGB palette[n]; // LUT
```

0	0	1	2	3
0	1	2	3	2
1	2	3	2	1
2	3	2	1	0
3	2	1	0	0

Цвет точки определяется как **palette**[FB[x][y]];

В ходу были 1, 2, 4, 6, 8, 12-битные палитры

1 бит: 2 цвета (чёрно/белый, чёрно/зелёный и т. д.)
2 бита: 4 цвета адаптер CGA (IBM, 1981, 16 Kb Video RAM)
4 бита: EGA (IBM, 1984, 64-256 Kb)/VGA (IBM, 1987, 256 Kb)
6 бит: Original Amiga chipset (Commodore Amiga, палитра 32 цвета)
8 бит: VGA low res, Super VGA (1989), AGA
12 бит: некоторые станции Silicon Graphics



Палитра (LUT)

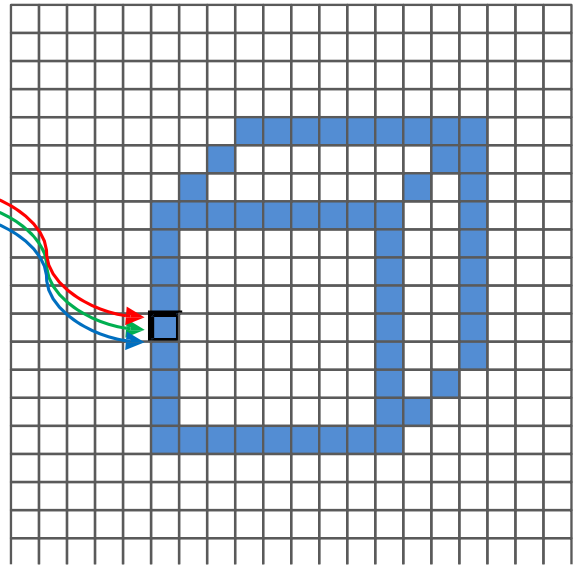
Буфер кадра FB[n,m]

1	1	2	2	2	2
1	1	2	1	1	1
1	1	2	1	1	1
1	1	2	1	1	1
1	1	2	1	1	1
1	1	2	1	1	1
1	1	2	1	1	1

Палитра

Index	R	G	B
0	0	0	0
1	255	255	255
2	84	141	212
…	…	…	…

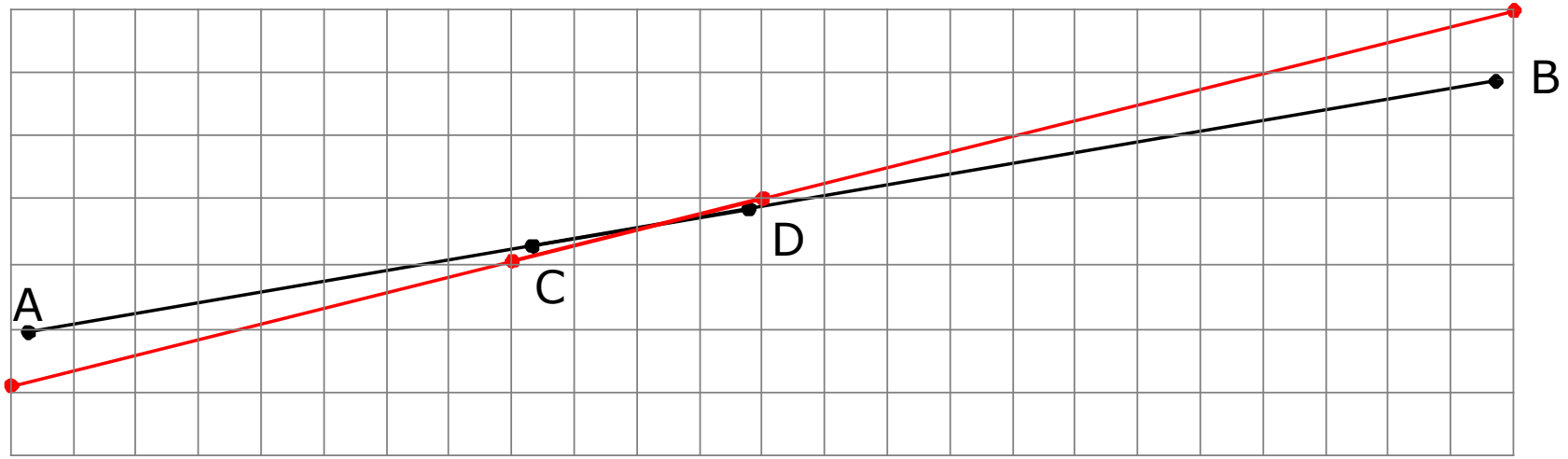
Экран



Полноцветные дисплеи

- 12 бит (4R+4G+4B) = 4096 цветов, некоторые модели сотовых, карманных плееров, КПК
- 15 бит HighColor (5R+5G+5B) = 32768 цветов
- 16 бит HighColor (5R+6G+5B) = 65536 цветов
- 24 бит TrueColor (8R+8G+8B) = 16777216 цветов
- 32 бит TrueColor (8R+8G+8B+8Alpha/empty)
- 48 бит (12R+12G+12B+12Alpha), 96 бит (32 FB+ 16 Z-buffer) * 2, SGI
- 40-64 бита BrilliantColor (пример: 8R+8G+8B+8C+8M+8Y), TI2005

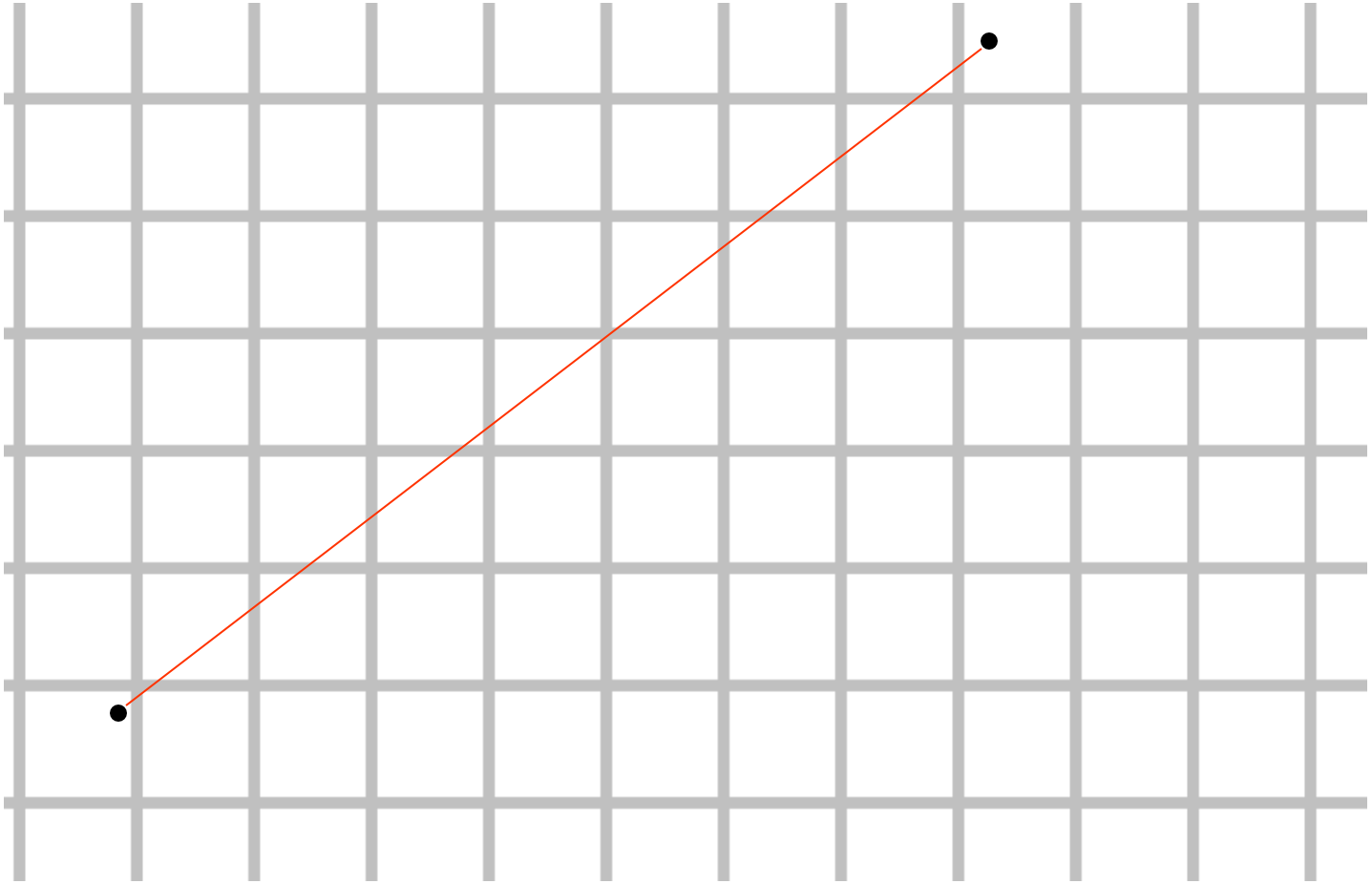
Осторожно – машинная точность



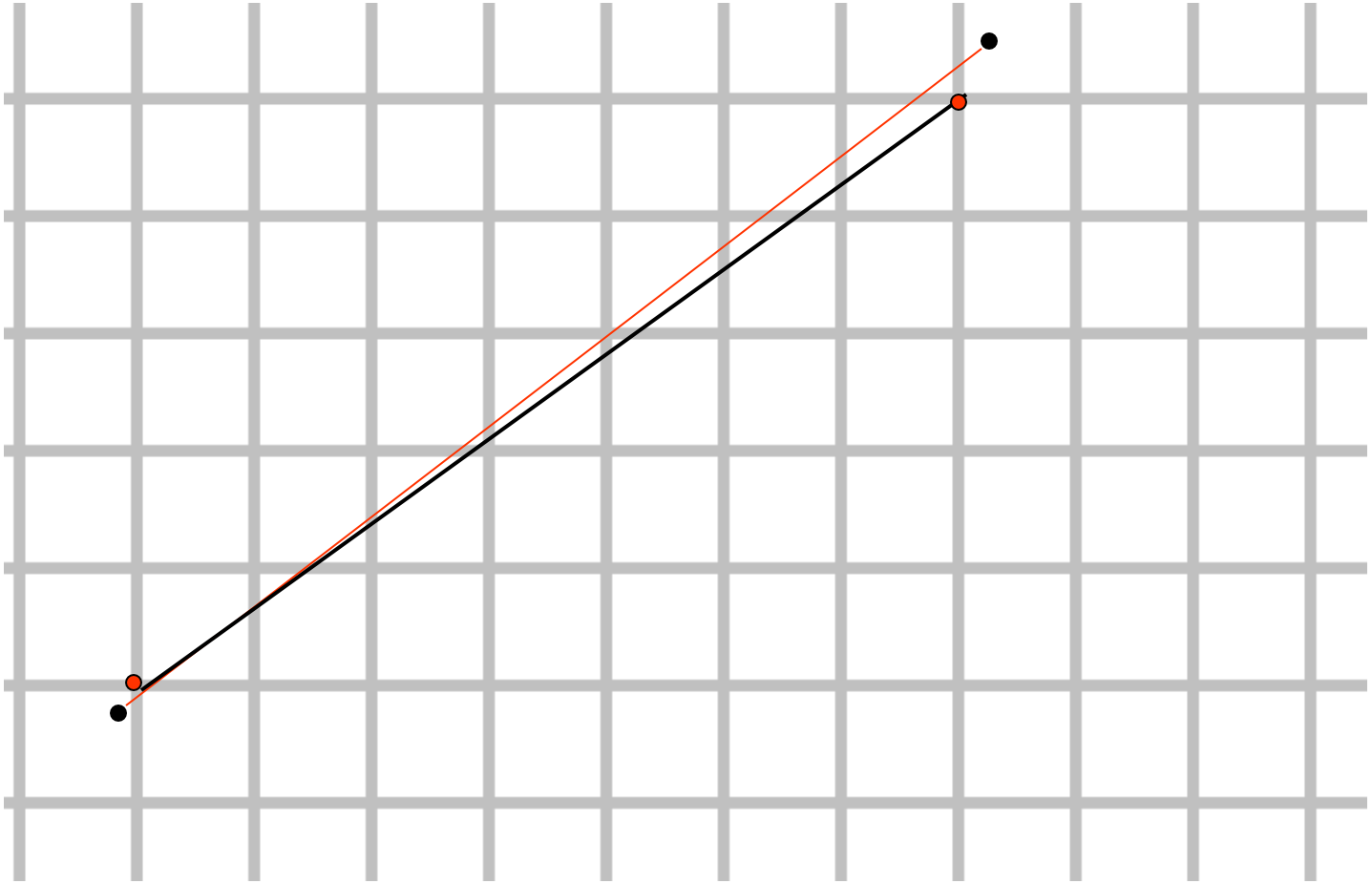
Сетка – это точность представления вещественных чисел. Строим прямую **L(A,B)**, проходящую через черные точки **A** и **B**. На самом деле вместо них мы используем красные точки. Выбираем еще две точки **C** и **D** **є** **L(A,B)**. Строим прямую **L(C,D)**. Как видим, **AB** не принадлежат **L(C,D)**.

Интерполировать можно, экстраполировать – нельзя!

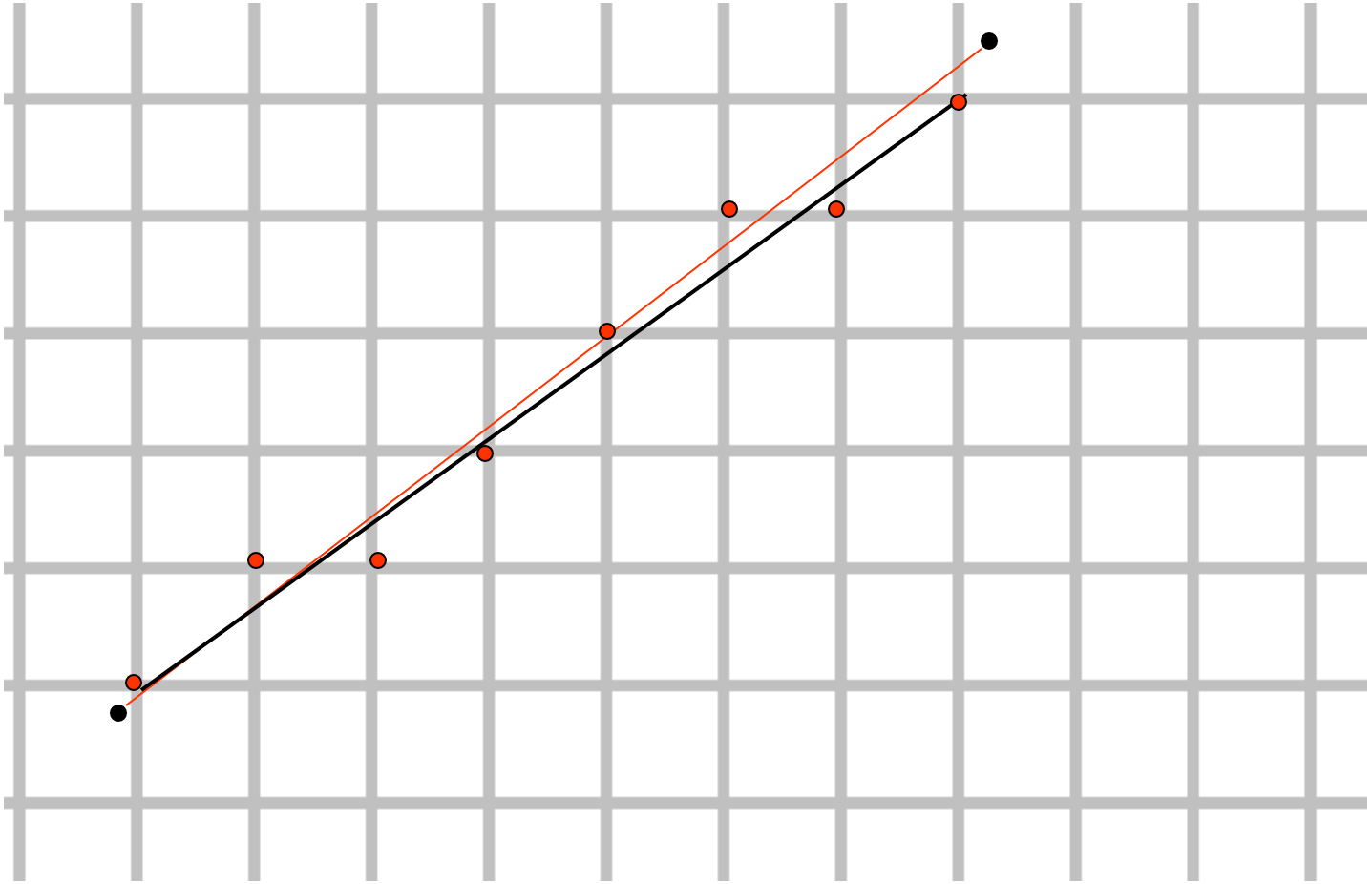
Растеризация отрезков



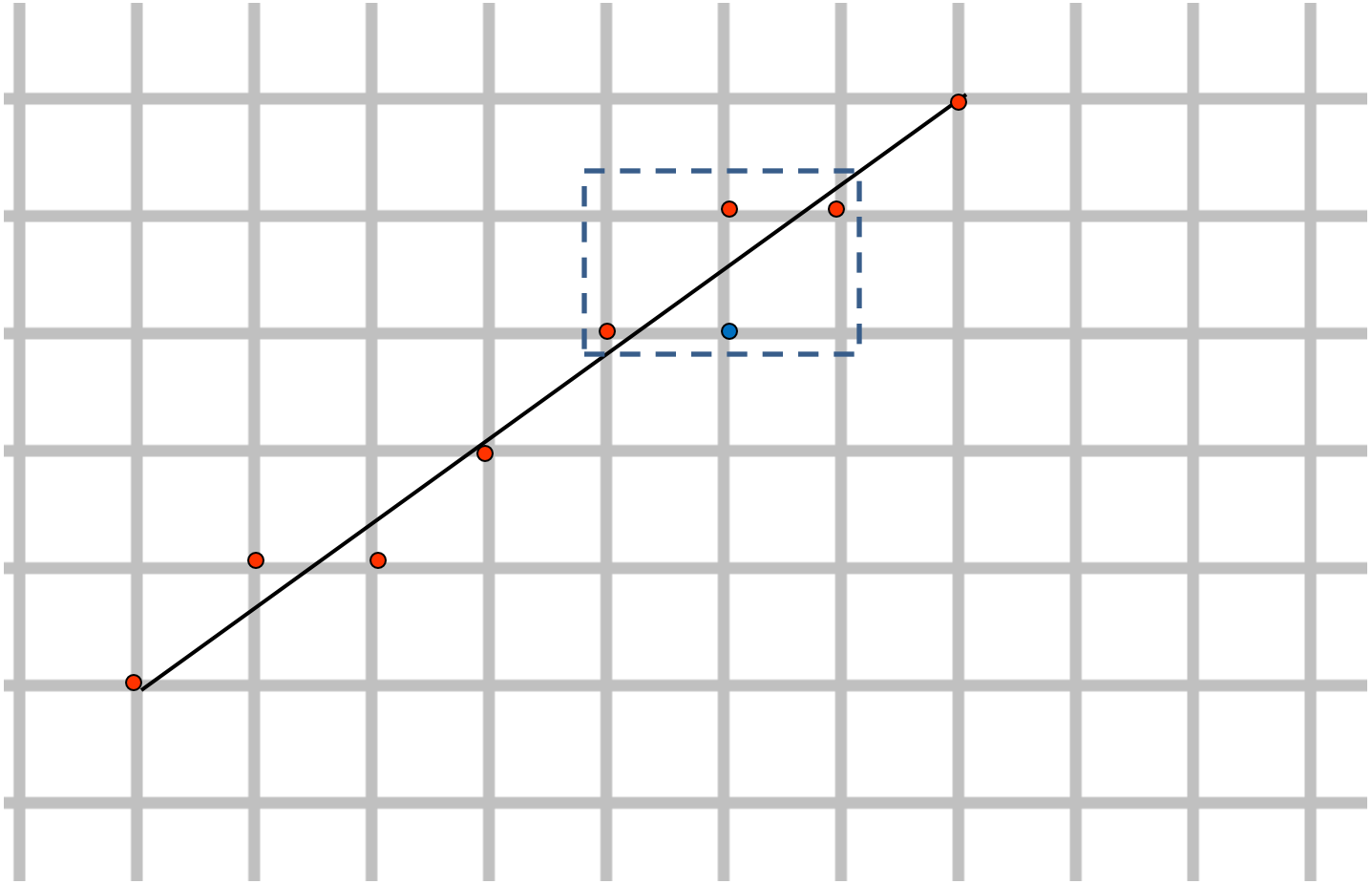
Растеризация отрезков



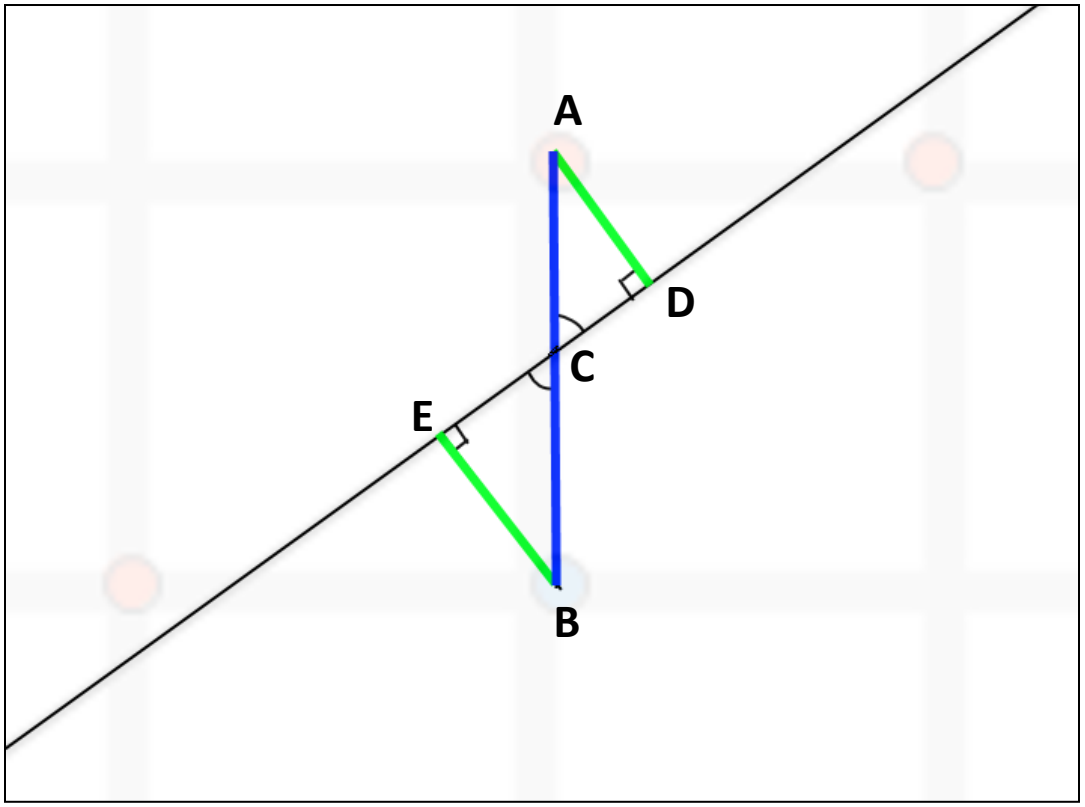
Растеризация отрезков



Растеризация отрезков



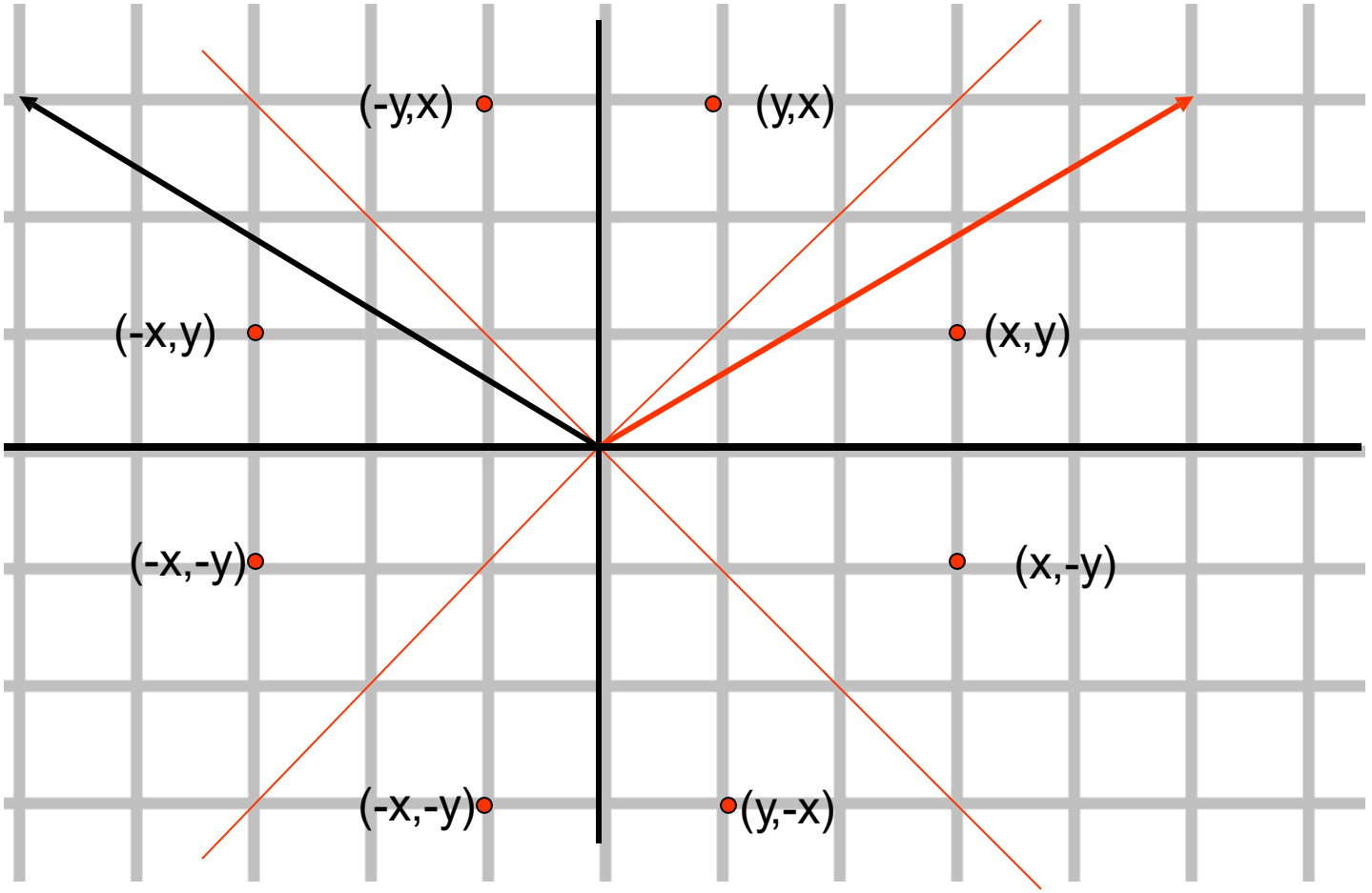
Растеризация отрезков



▲ACD ~ ▲BCE

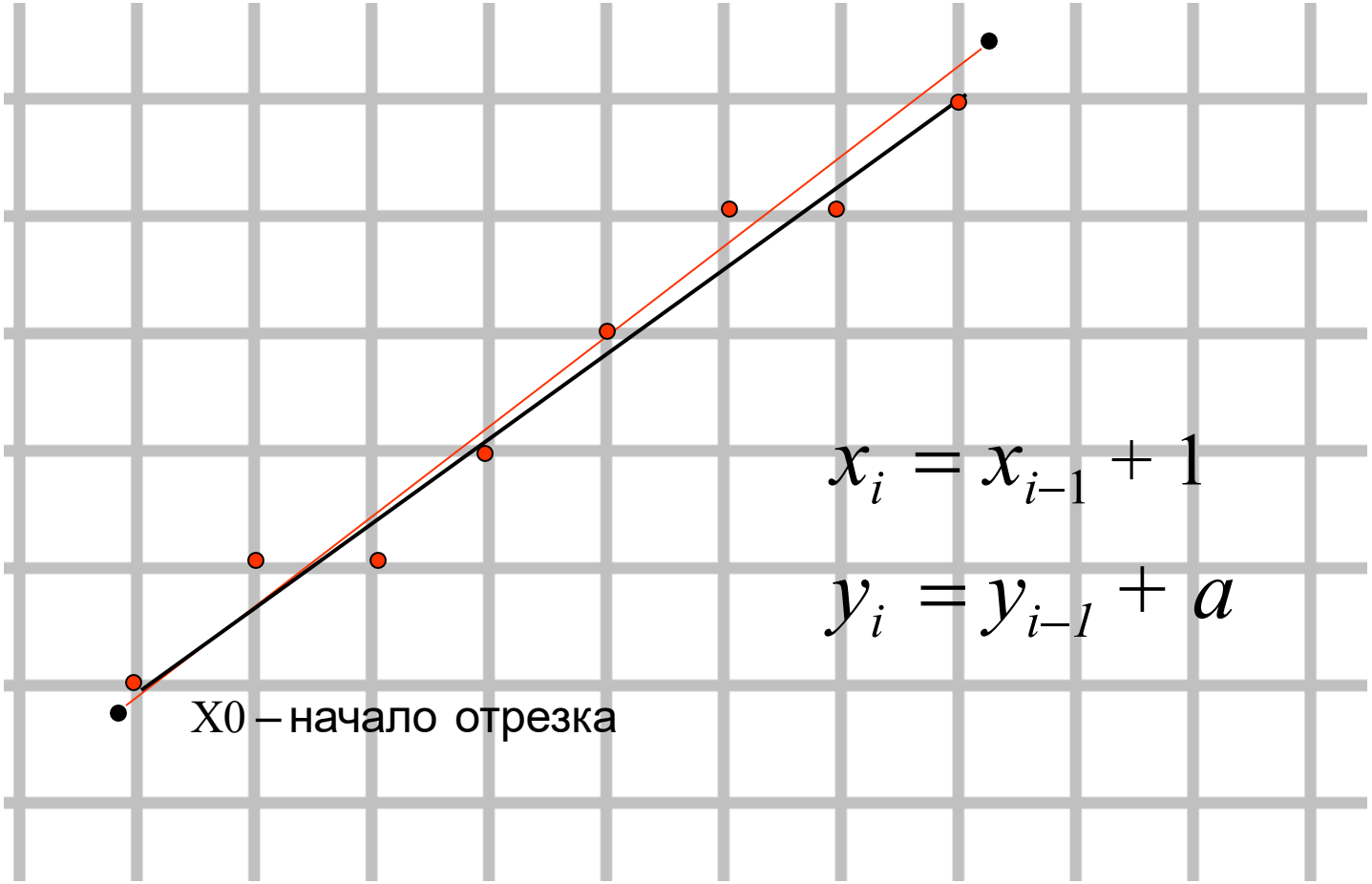
|AD|/|BE|=|AC|/|BC|

8-симметрия



$$y = ax + b, \quad b = 0, \quad 0 \le a \le 1 \quad => \quad y = ax$$

Растеризация отрезков

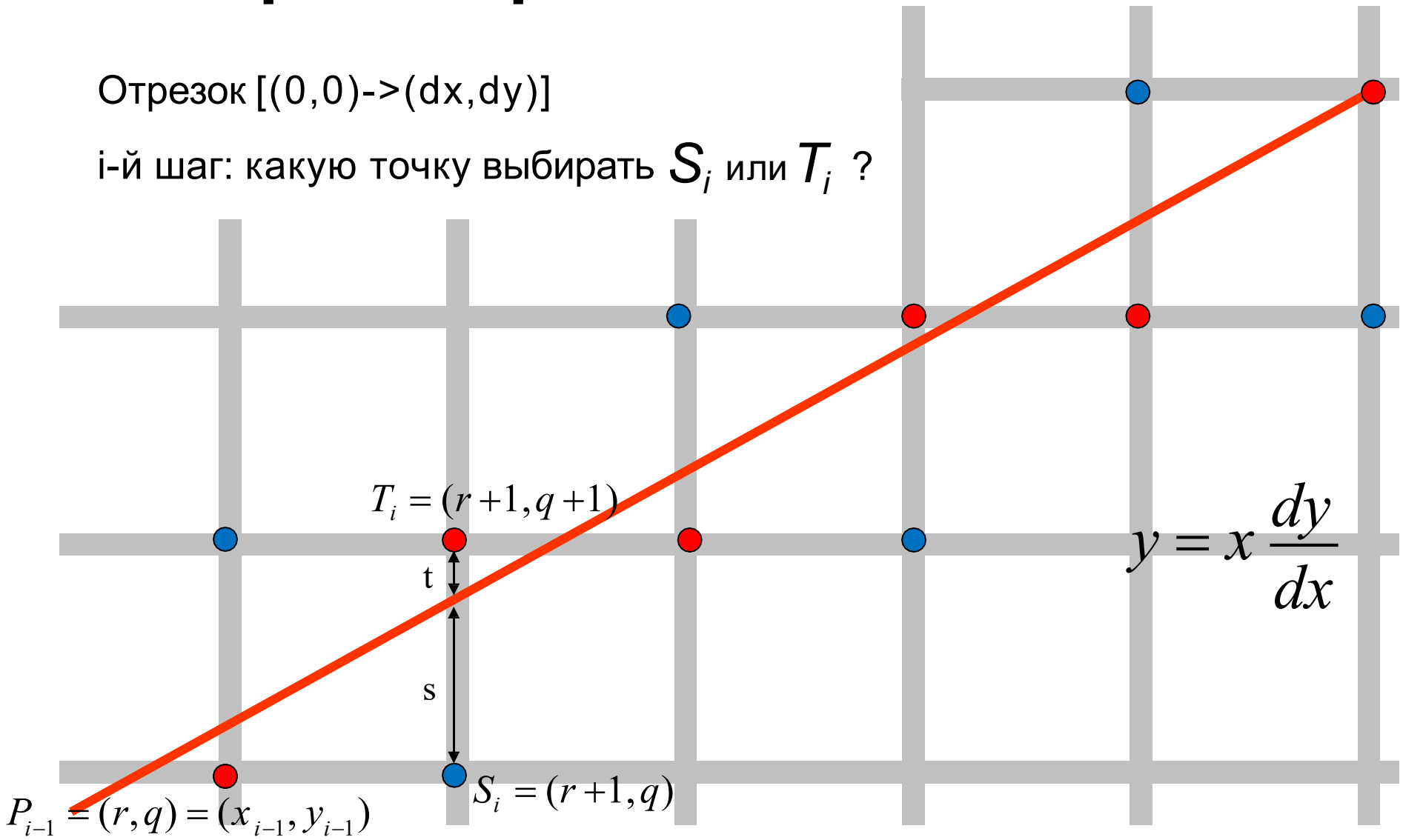


Делаем пересчет в растровую систему координат $\Delta x = 1$

Алгоритм Брезенхэма

Отрезок [(0,0)->(dx,dy)]

i-й шаг: какую точку выбирать S_i или T_i ?

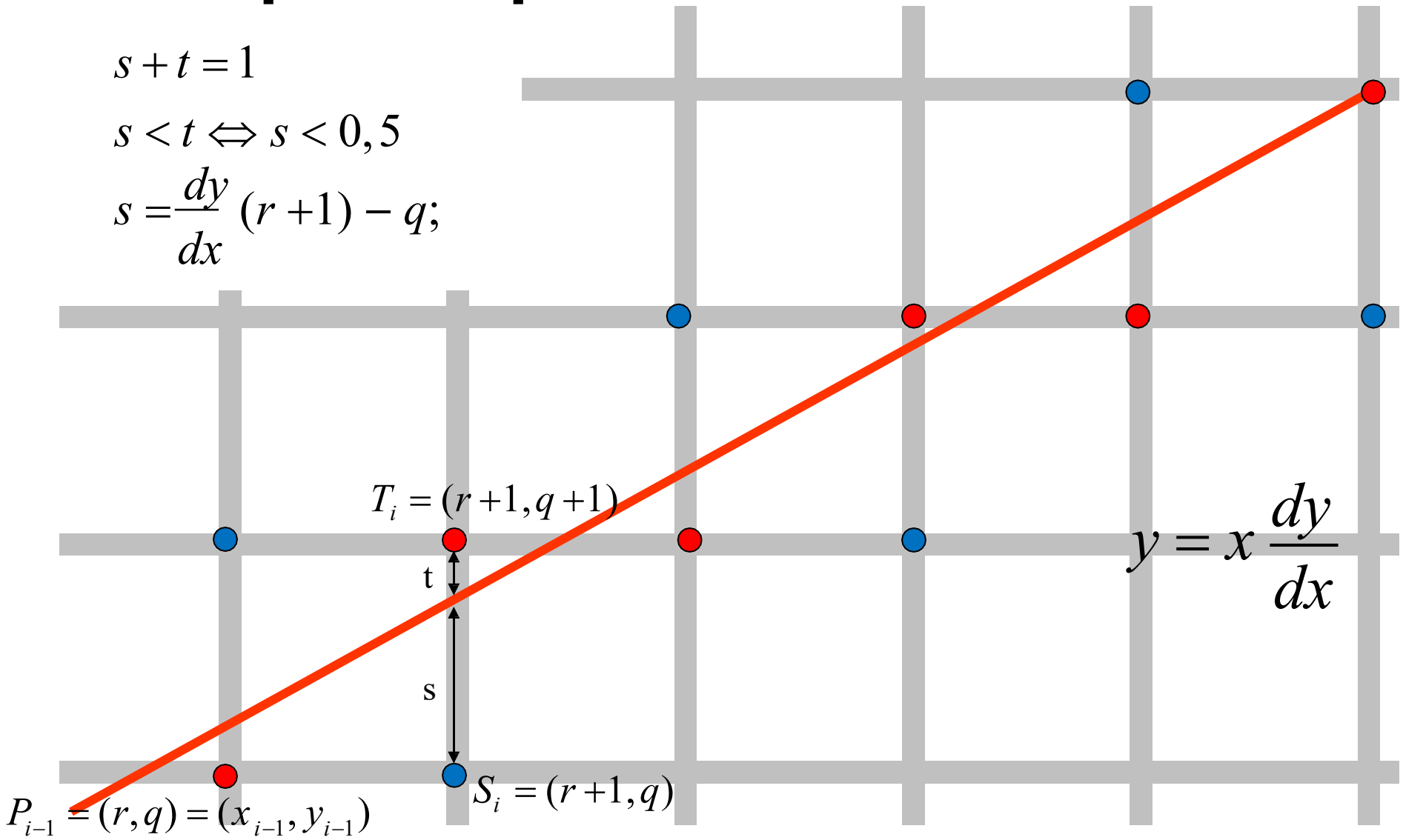


Алгоритм Брезенхэма

$$s + t = 1$$

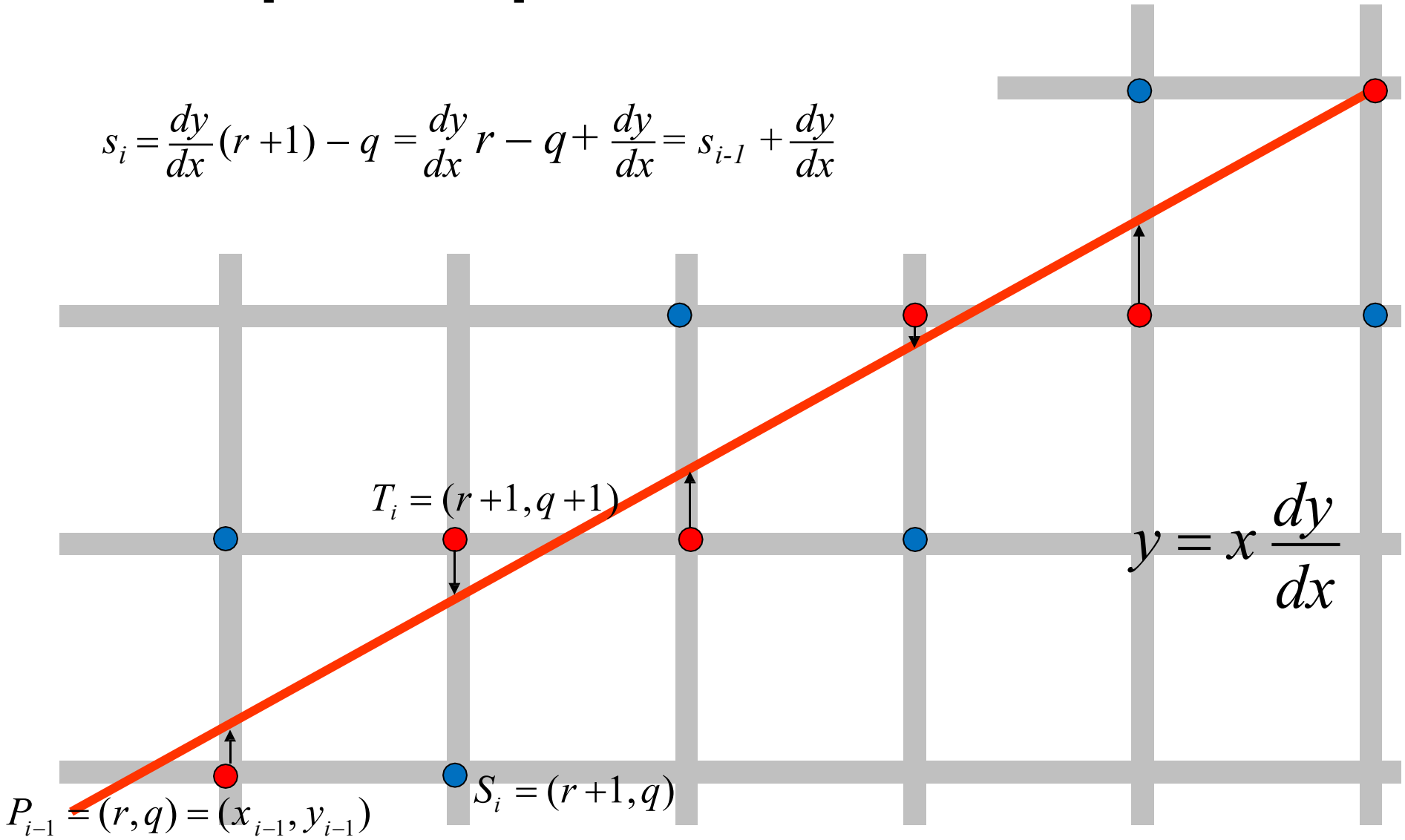
$$s < t \Leftrightarrow s < 0,5$$

$$s = \frac{dy}{dx}(r+1) - q;$$



Алгоритм Брезенхэма

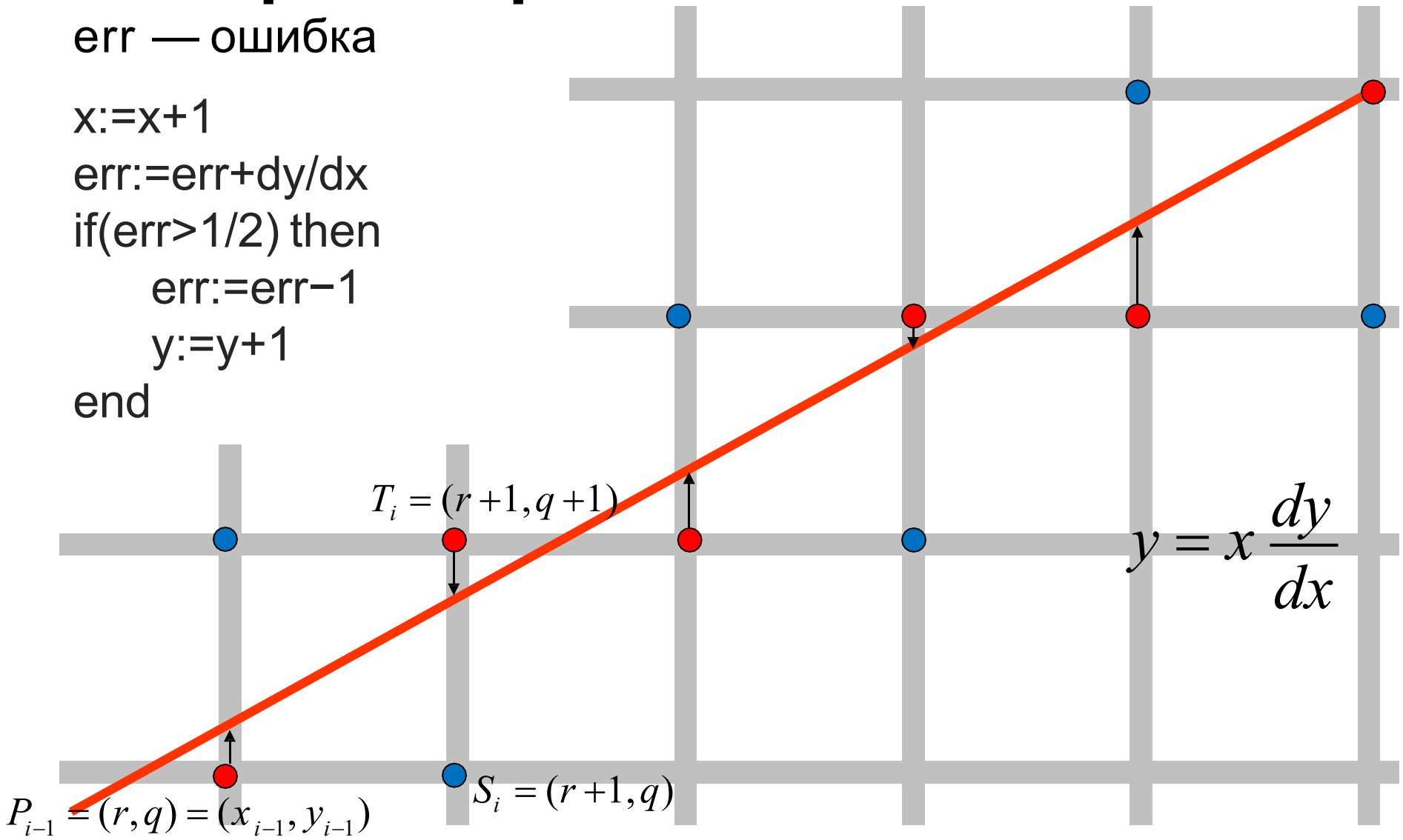
$$s_i = \frac{dy}{dx}(r+1) - q = \frac{dy}{dx} r - q + \frac{dy}{dx} = s_{i-1} + \frac{dy}{dx}$$



Алгоритм Брезенхэма

err — ошибка

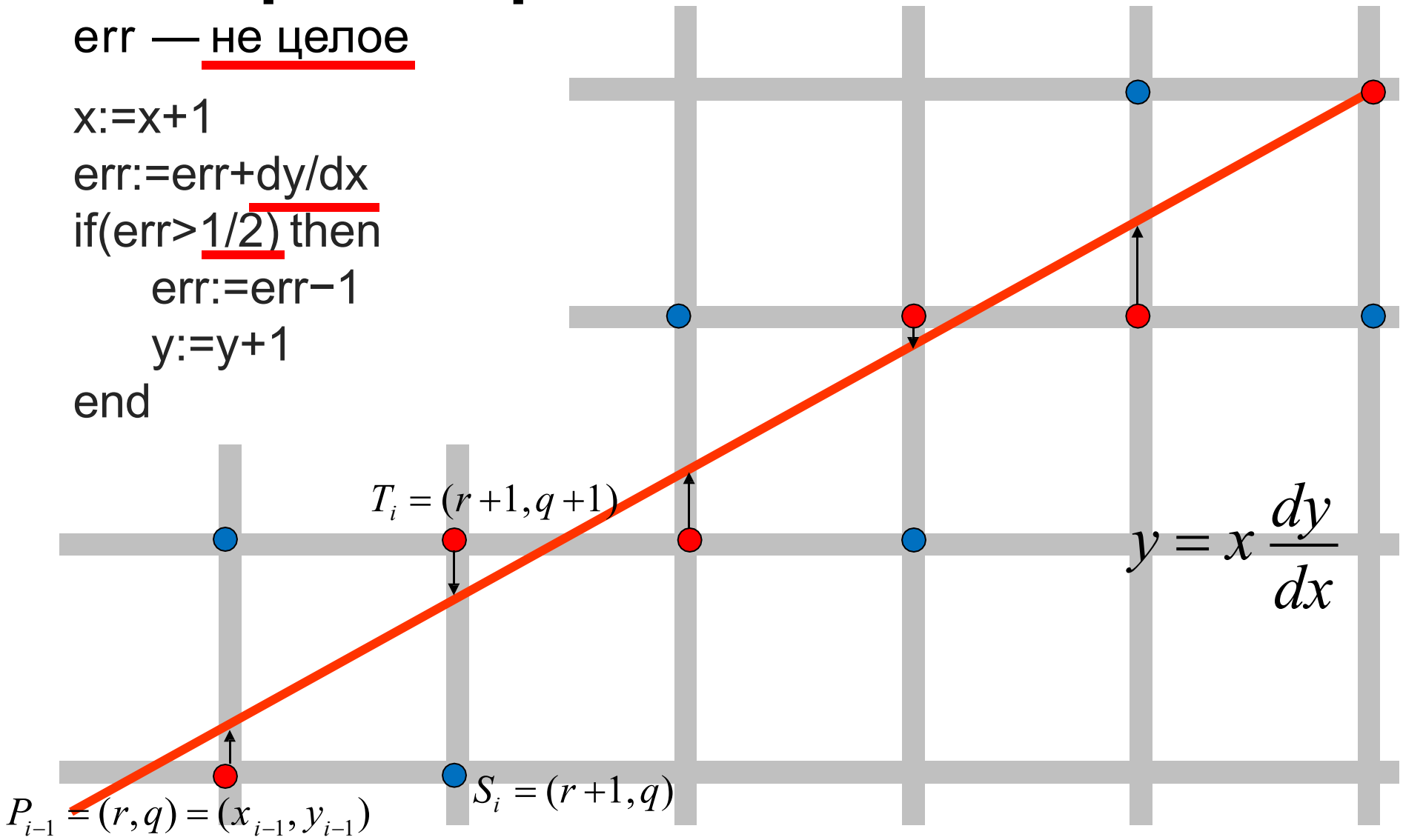
```
x:=x+1
err:=err+dy/dx
if(err>1/2) then
      err:=err−1
      y:=y+1
end
```



Алгоритм Брезенхэма

err — не целое

```
x:=x+1
err:=err+dy/dx
if(err>1/2) then
      err:=err−1
      y:=y+1
end
```

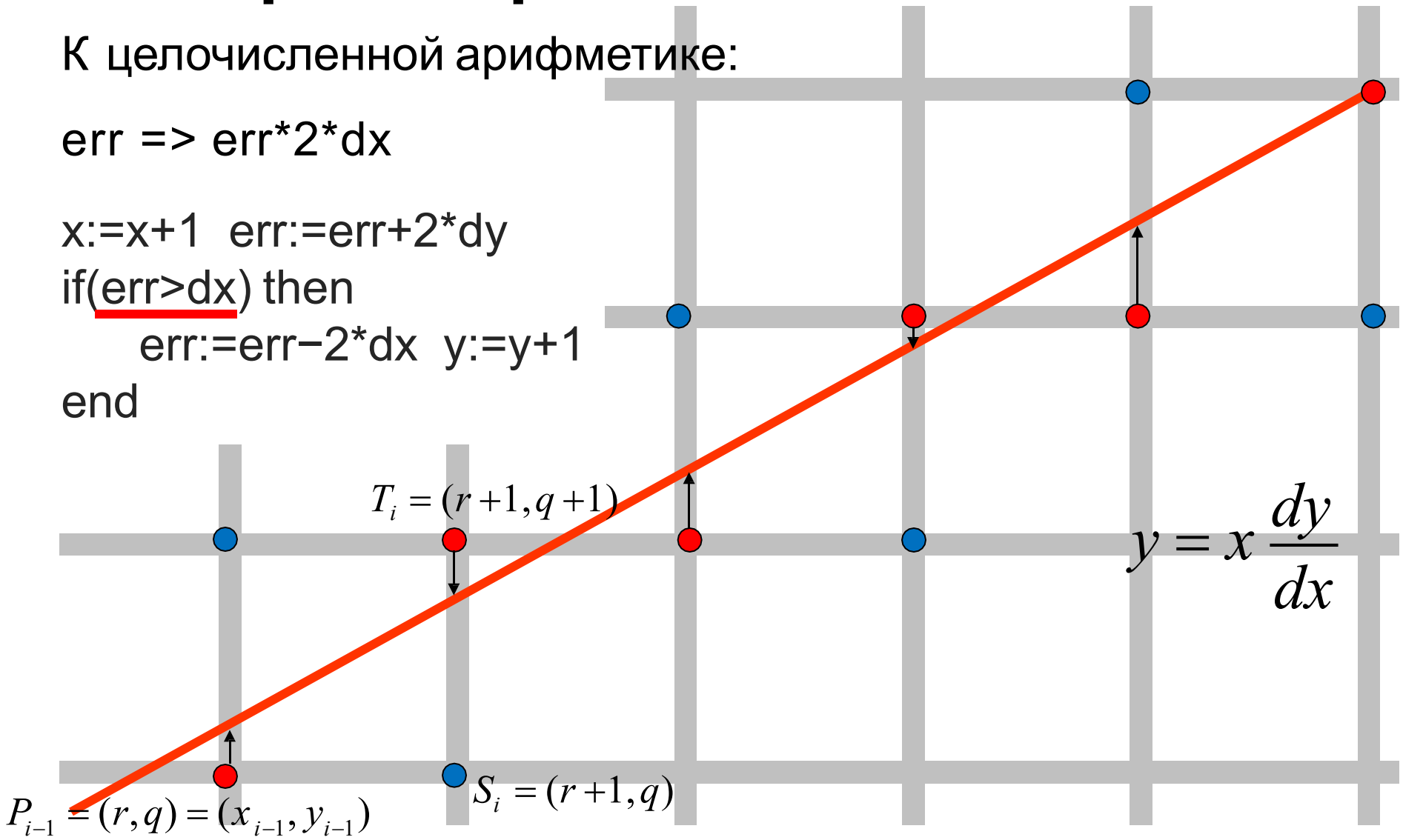


Алгоритм Брезенхэма

К целочисленной арифметике:

err => err*2*dx

x:=x+1 err:=err+2*dy
if(err>dx) then
    err:=err−2*dx y:=y+1
end



Алгоритм Брезенхэма

err => err – dx

err в начале равен –dx

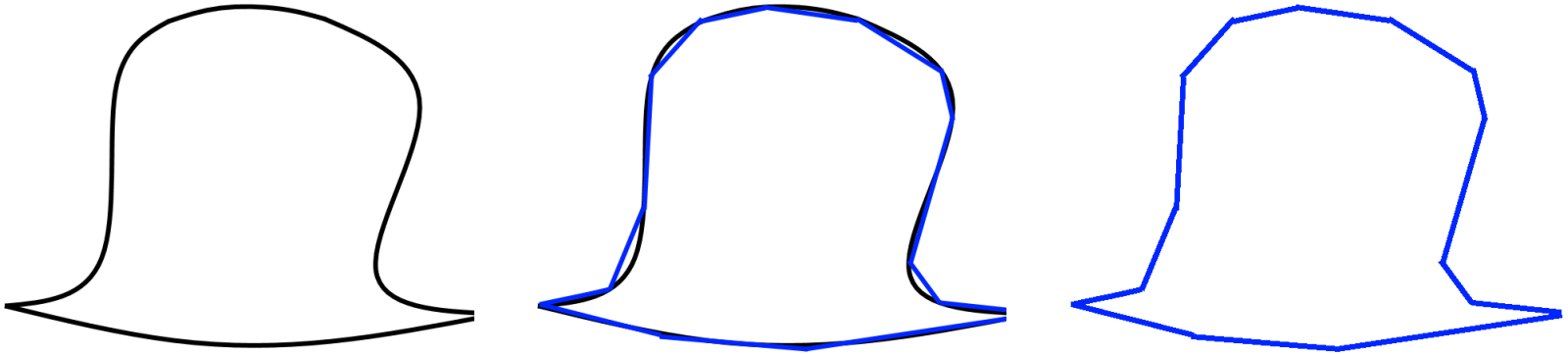
```
x:=x+1
err:=err+2*dy
if(err>0) then
      err:=err−2*dx
      y:=y+1
end
```

C++

```
int x = x0;
int y = y0;
int err = -dx;
while( x<dx ) {
   x++;
   err += 2*dy;
   if (err > 0) {
         err −= 2*dx;
         y++;
   }
   SetPixel(x,y,Color);
}
```

Векторизация кривых

Кривые (сплайны, кривые Безье) приближаются ломаной

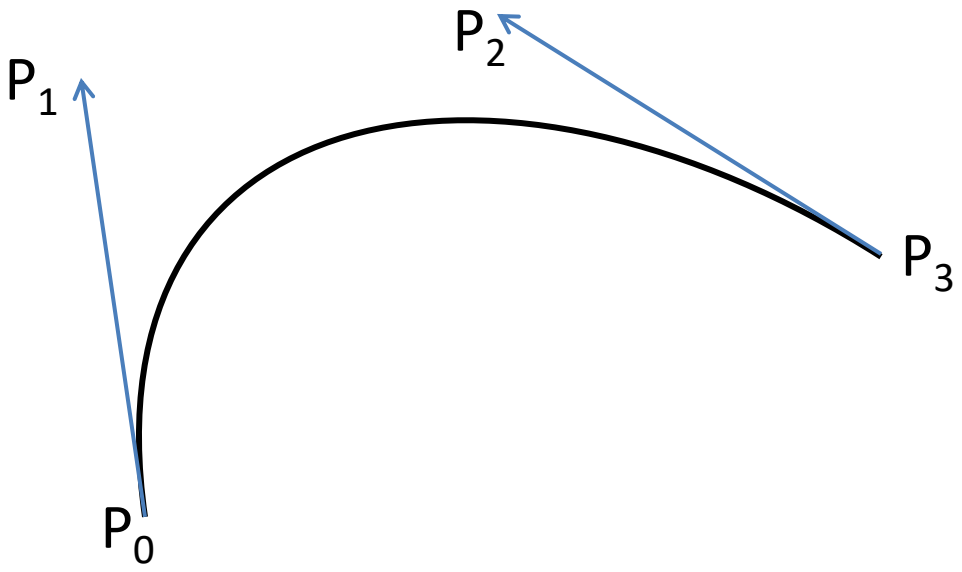


Кривая Безье

$$B(t) = \sum_{i=0}^{n} P_i b_{i,n}(t)$$

где Pi – контрольные точки кривой,

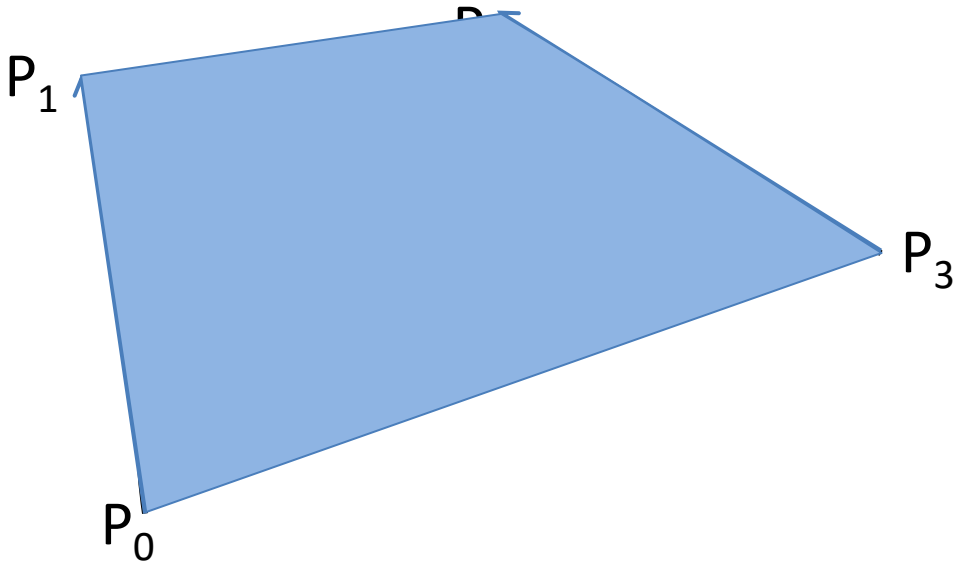
$$b_{i,n}(t) = \frac{n!}{i!\,(n-i)!} t^i (1-t)^{n-i}$$



$$B(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$

Кривая Безье

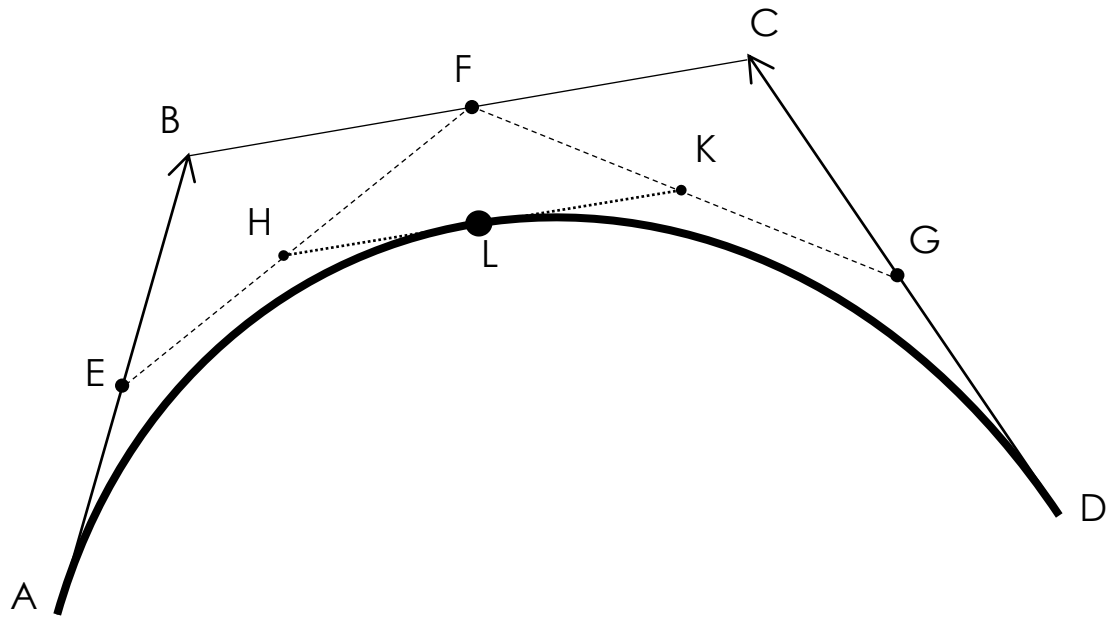
$$B(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$



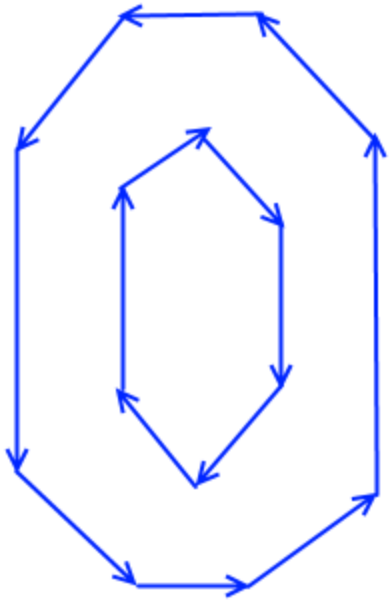
«Разбиение единицы» - все точки кривой лежат внутри выпуклой оболочки контрольных точек

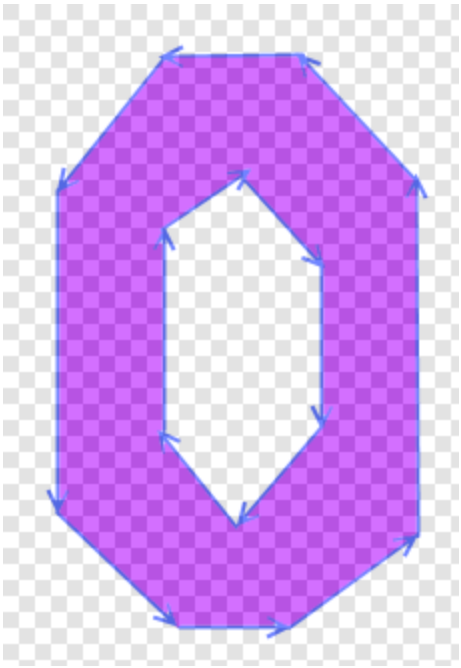
Кривая Безье

$$B(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$

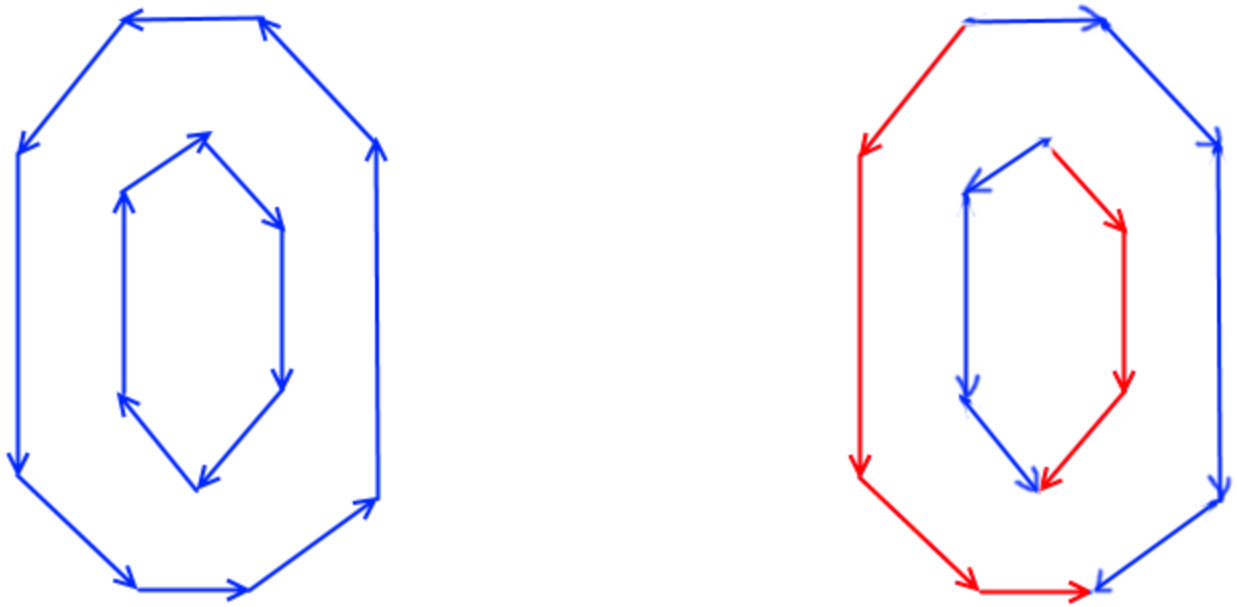


Векторизация текста



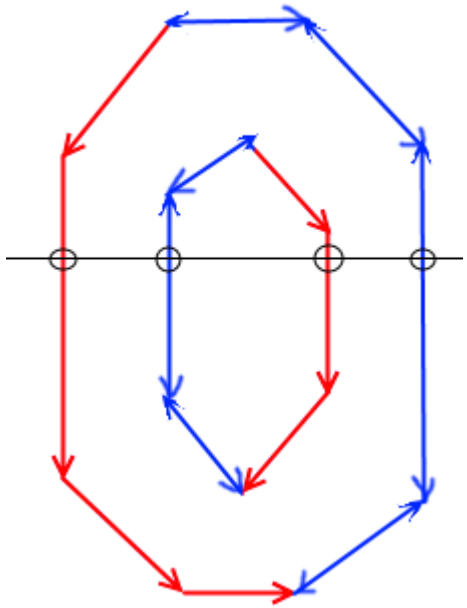


Векторизация текста



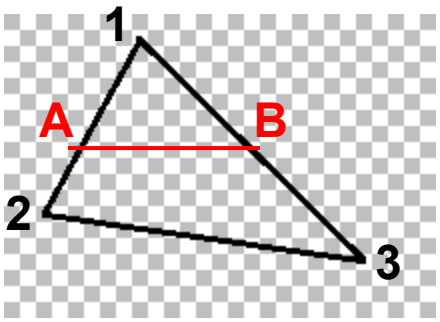
1) Векторизация кривых всех контуров.
2) Разбиение на сегменты, упорядоченные по вертикали (внутри сегмента Y монотонно возрастает).

Векторизация текста



Итерирование по вертикали (по Y)
по всем сегментам:
3) Растрирование первого отрезка для всех активных сегментов (если текущий Y пересекает отрезок) – получаем X. Если отрезок кончился, то переходим к следующему отрезку. Если все отрезки закончились, то удаляем сегмент из списка.
4) Упорядочиваем все X и разбиваем на пары.
5) Рисуем пиксели от первого X в паре до второго X в паре.

Растрирование треугольника



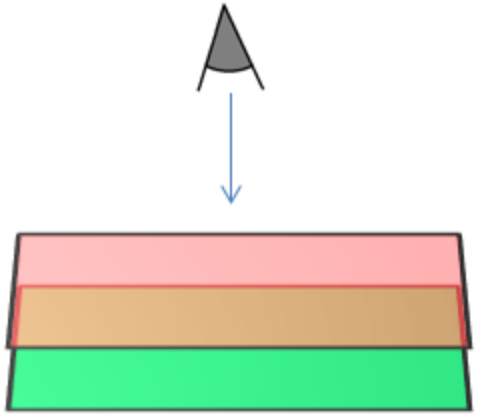
Растрирую отрезок (1,2) и (1,3) для текущего Y – получаю точки A и B. При этом интерполирую все параметры из вершин 1 – 2 и 1 – 3 соответственно (цвет, нормаль, текстурные координаты…).

Вдоль X от A до B рисую все точки. При этом интерполирую все параметры из точек A и B (цвет, нормаль, текстурные координаты…).

Получаю БИЛИНЕЙНУЮ интерполяцию параметров.

Линейное микширование цветов

$$C_R = (1 - \alpha_1) \cdot C_{back} + \alpha_1 \cdot C_1$$



C_1, α_1

C_{back}

Альфа – это непрозрачность:
1 – полностью непрозрачен
0 – полностью прозрачен

Линейная интерполяция

Стопка полупрозрачных слайдов

$$C_{res} = C_1 \bullet A_1 + C_{back} \bullet (1 - A_1)$$ Один слайд (+ фон)

$$C_{res} = C_1 \bullet A_1 + C_2 \bullet A_2 \bullet (1 - A_1) + C_{back} \bullet (1 - A_2) \bullet (1 - A_1)$$ Два слайда (+ фон)

...

$$C_{res} = C_1 \bullet A_1 + C_2 \bullet A_2 \bullet (1 - A_1) + \ldots + C_n \bullet A_n \bullet \prod_{k=1}^{n-1} (1 - A_k) + C_{back} \bullet \prod_{k=1}^{n} (1 - A_k)$$ n слайдов (+ фон)

$$C_{res} = C_x \bullet A_x + C_{back} \bullet (1 - A_x)$$ Искомые цвет и вес всей стопки C_x и A_x

$$A_x = 1 - \prod_{k=1}^{n} (1 - A_k)$$

$$C_x = \frac{C_1 \bullet A_1 + C_2 \bullet A_2 \bullet (1 - A_1) + \ldots + C_n \bullet A_n \bullet \prod_{k=1}^{n-1} (1 - A_k)}{1 - \prod_{k=1}^{n} (1 - A_k)}$$

Стопка полупрозрачных слайдов

Замена – premultiplied color $P = C \bullet A$ и прозрачность $Tr = 1 - A$

$$Tr_x = \prod_{k=1}^{n} Tr_k$$

$$P_x = P_1 + P_2 \bullet Tr_1 + \ldots + P_n \bullet \prod_{k=1}^{n-1} Tr_k$$