Инженерная и компьютерная графика
6 семестр (диф.зачет)

Лектор:

Таранцев Игорь Геннадьевич

Доцент ФИТ НГУ, ИАиЭ, «СофтЛаб-НСК»

Создатели курса:

Дебелов Виктор Алексеевич

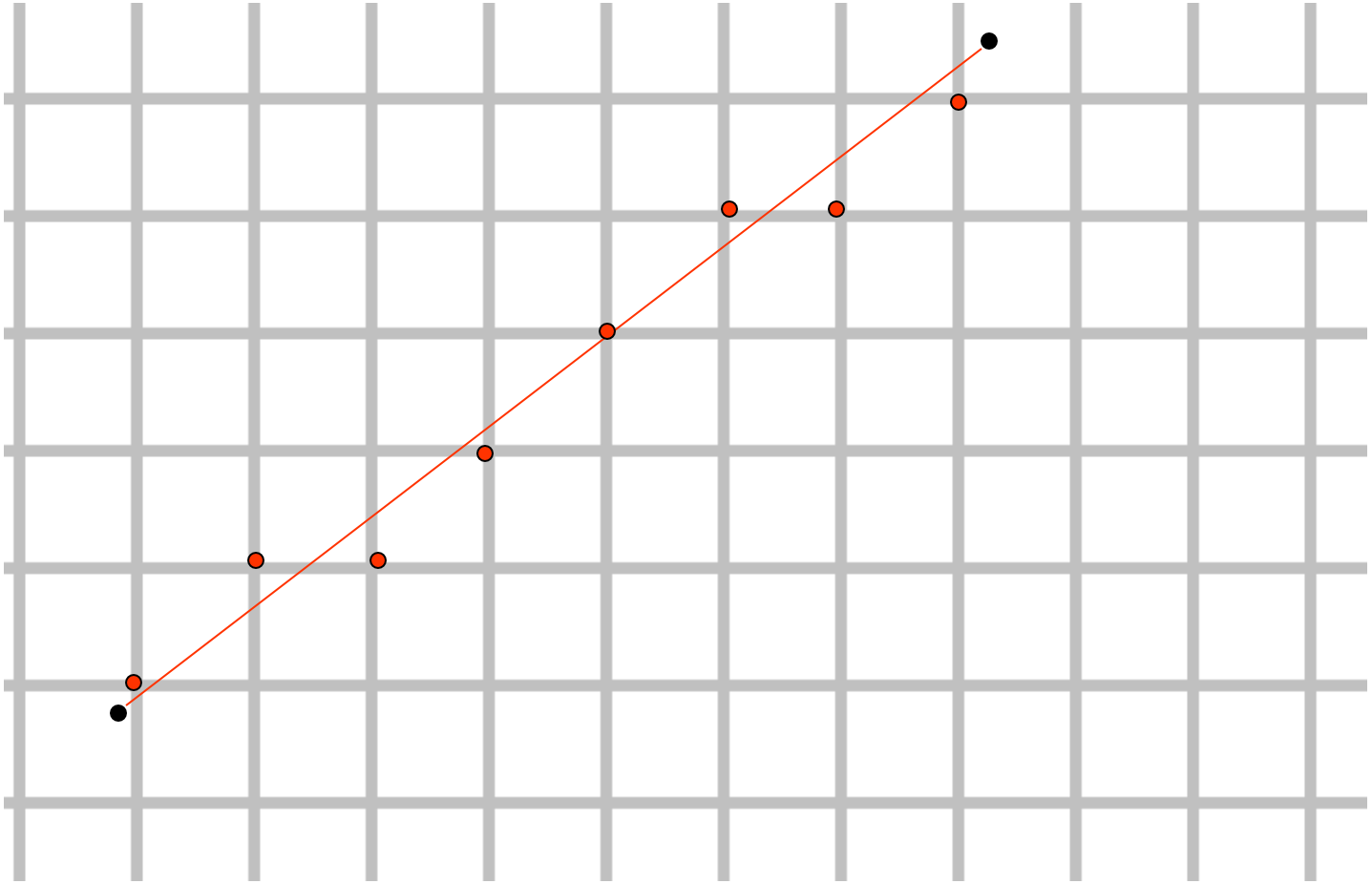
Валеев Тагир Фаридович

Козлов Дмитрий Сергеевич

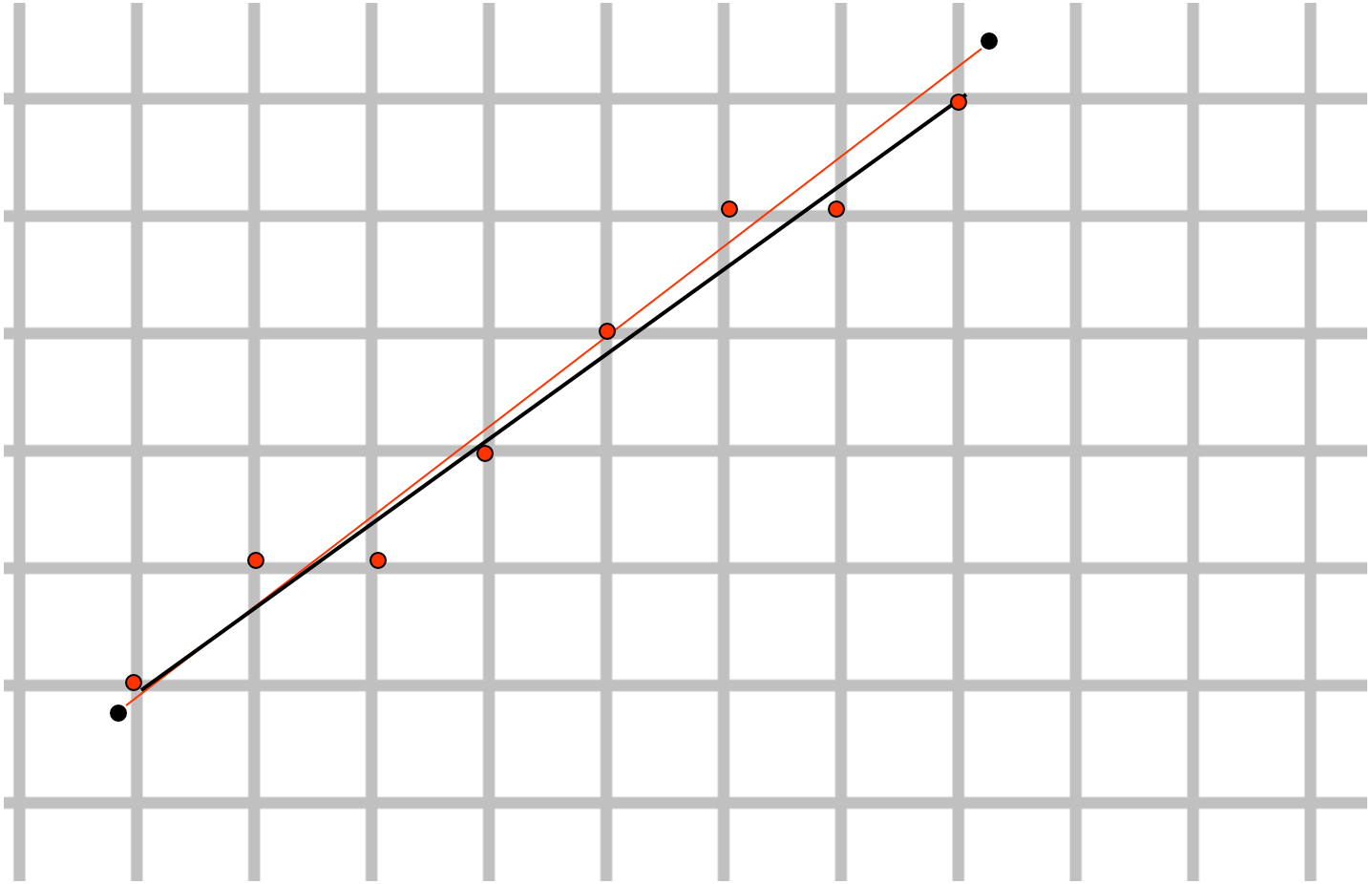
Задача №1

Алгоритм Брэзенхема
Спан заливка

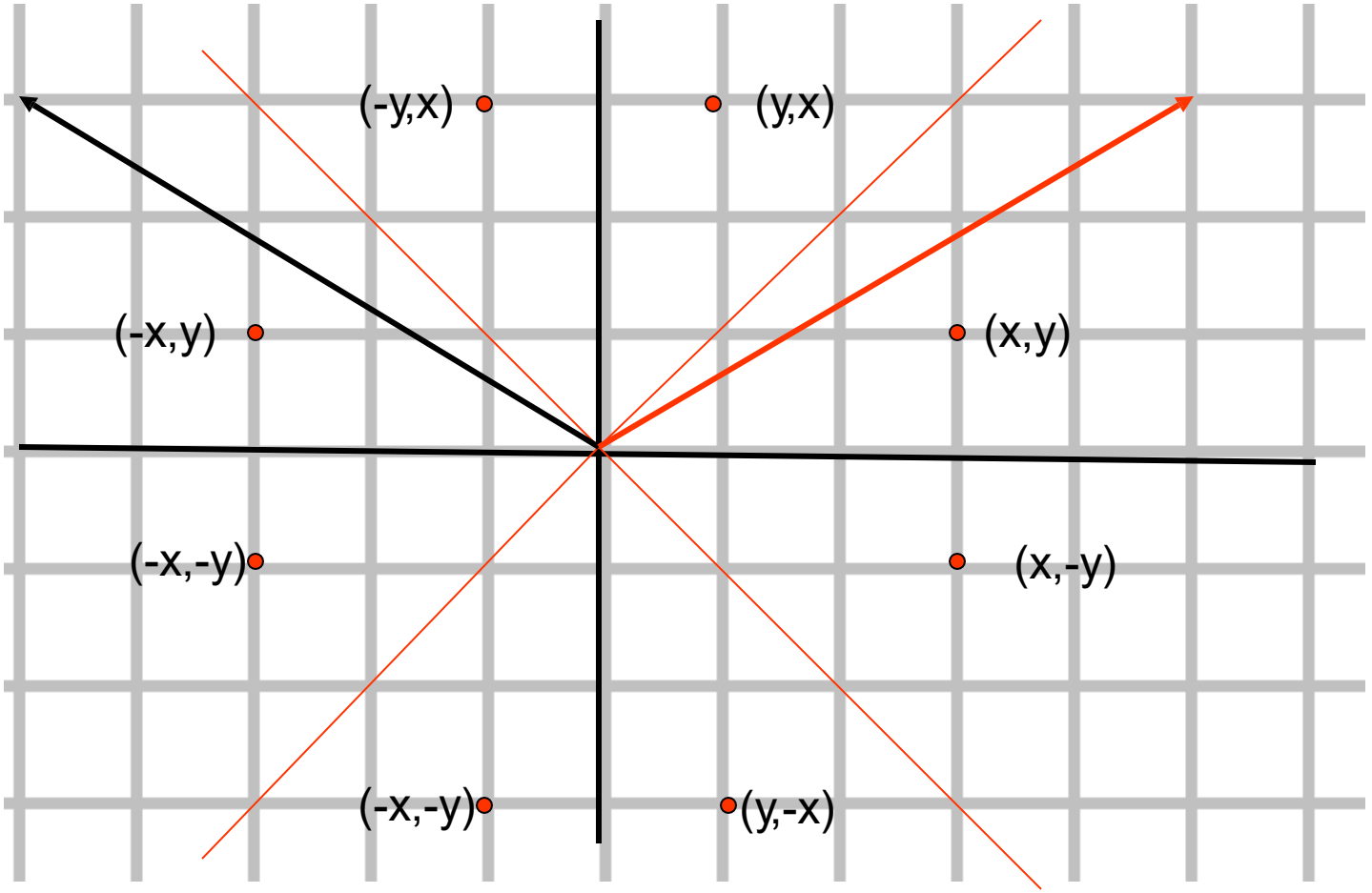
Растеризация отрезков



Растеризация отрезков

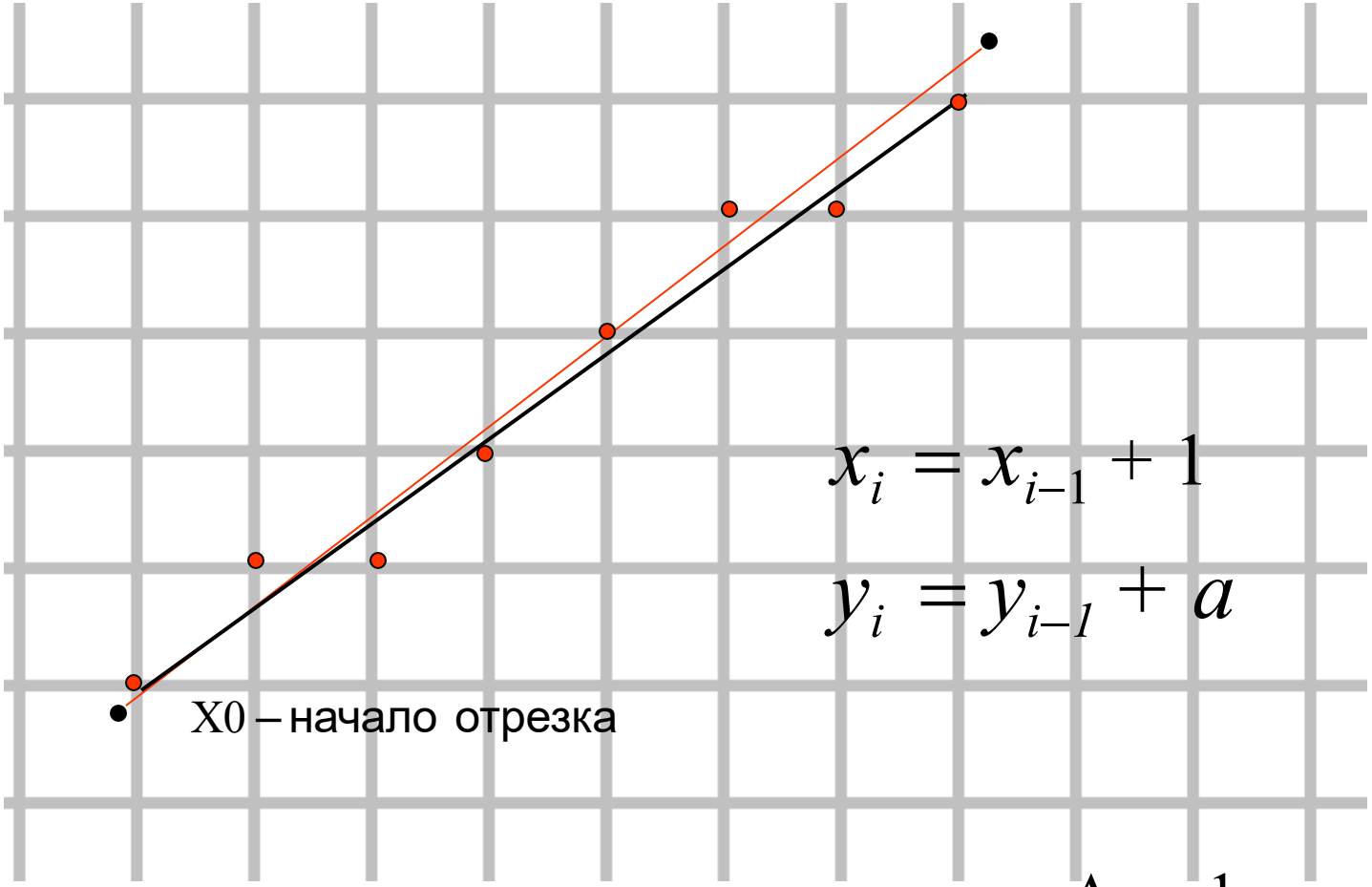


8-симметрия



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

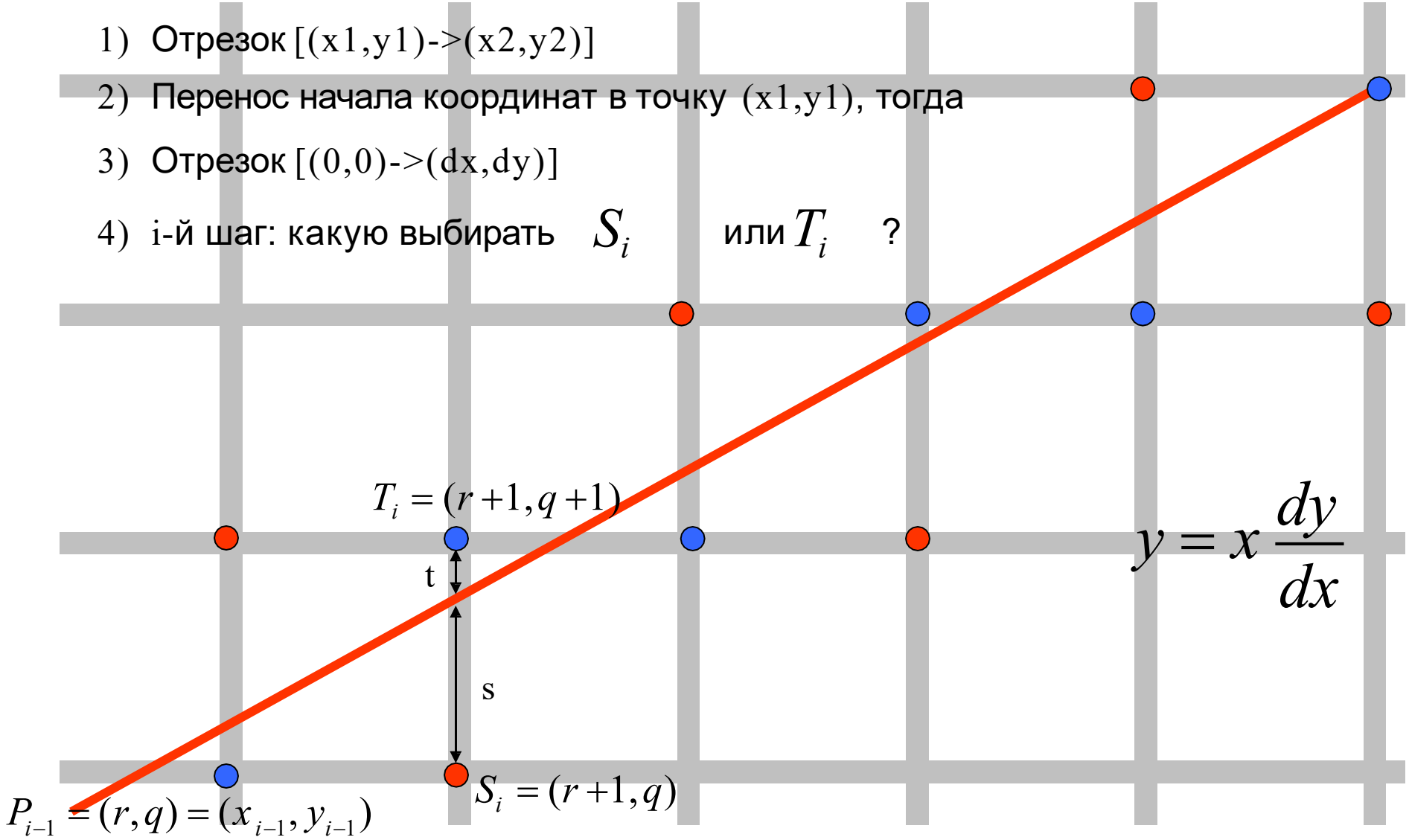
Растеризация отрезков



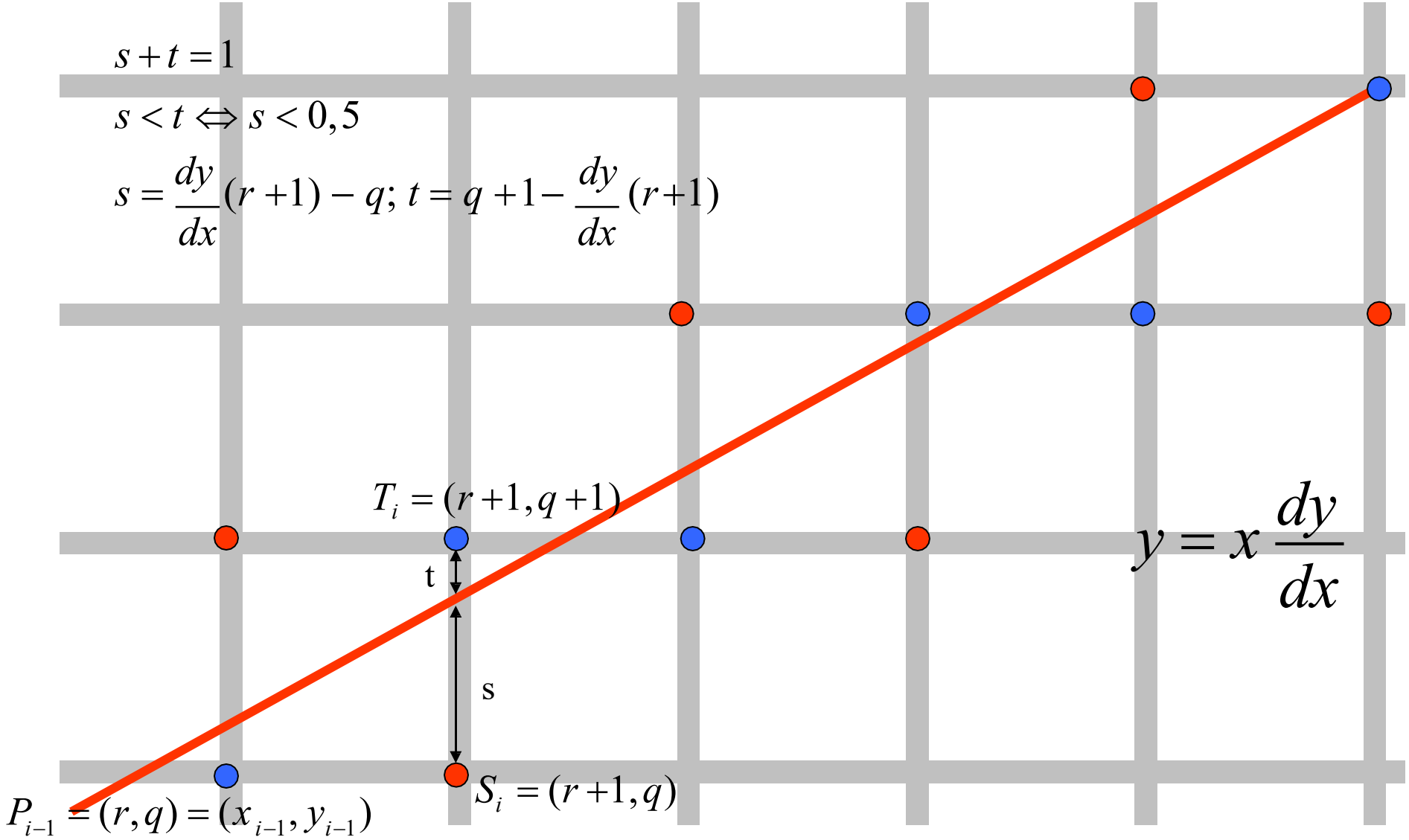
Делаем пересчет в растровую систему координат $\Delta x = 1$

Алгоритм Брезенхэма

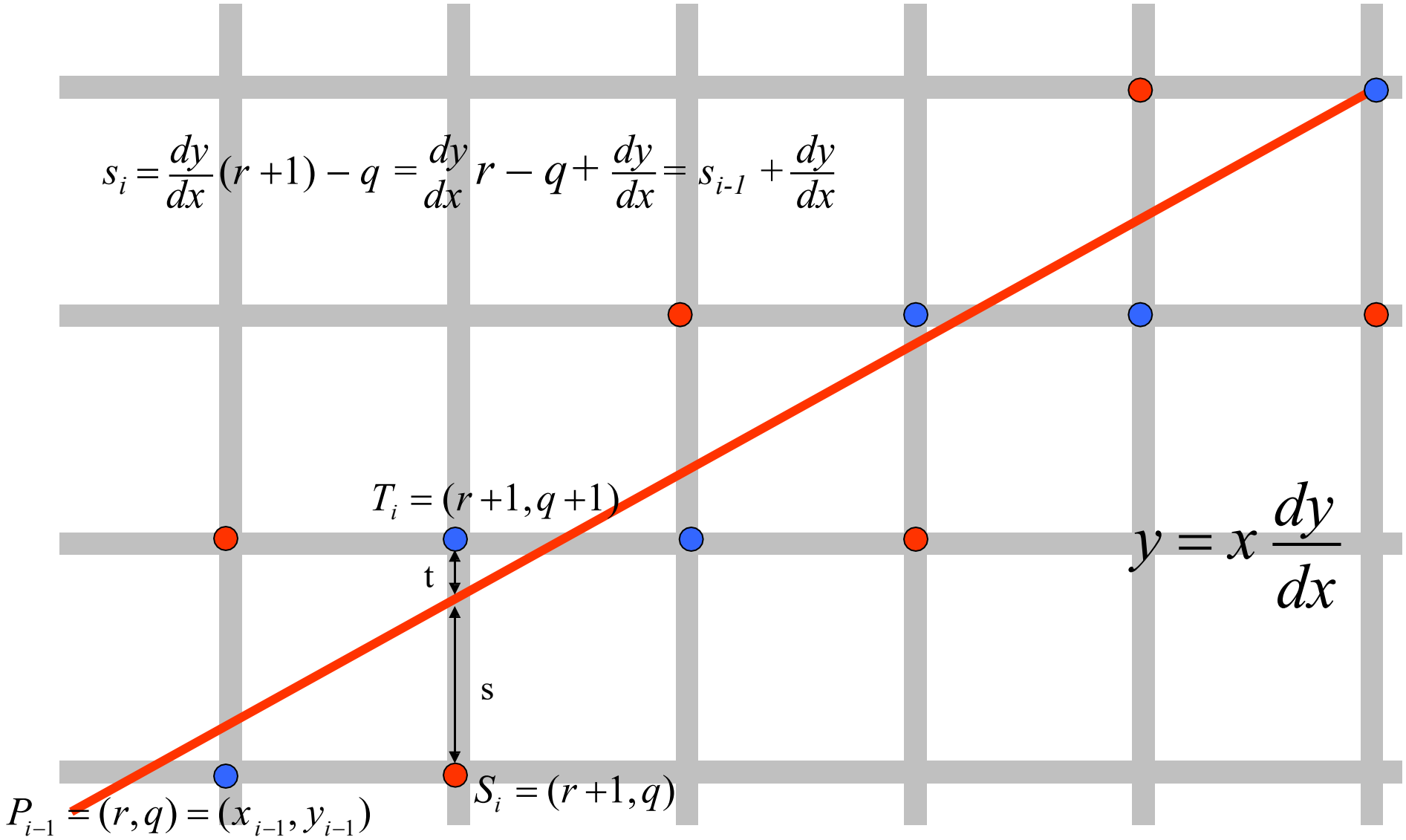
1) Отрезок [(x1,y1)->(x2,y2)]
2) Перенос начала координат в точку (x1,y1), тогда
3) Отрезок [(0,0)->(dx,dy)]
4) i-й шаг: какую выбирать S_i или T_i ?



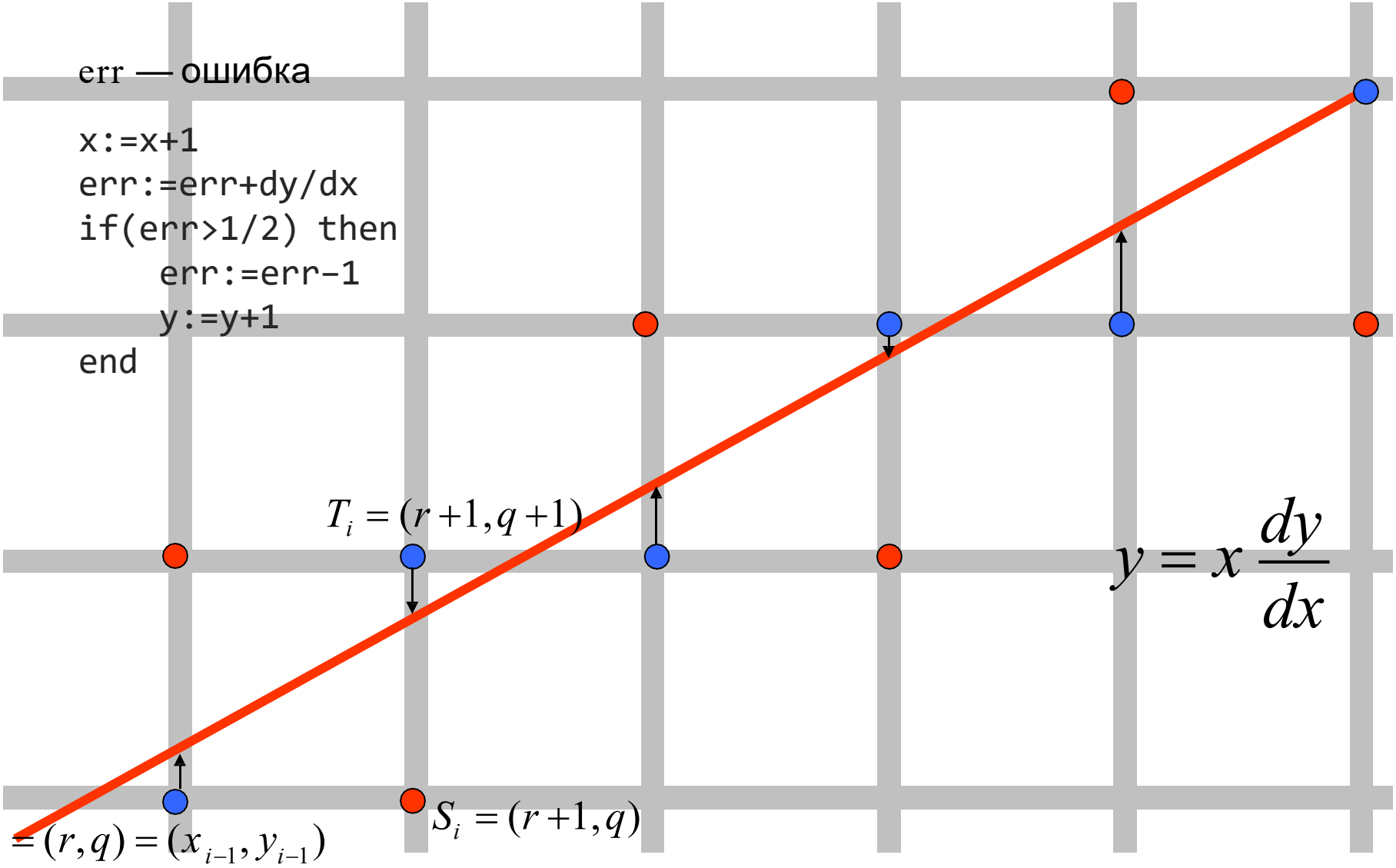
Алгоритм Брезенхэма



Алгоритм Брезенхэма



Алгоритм Брезенхэма

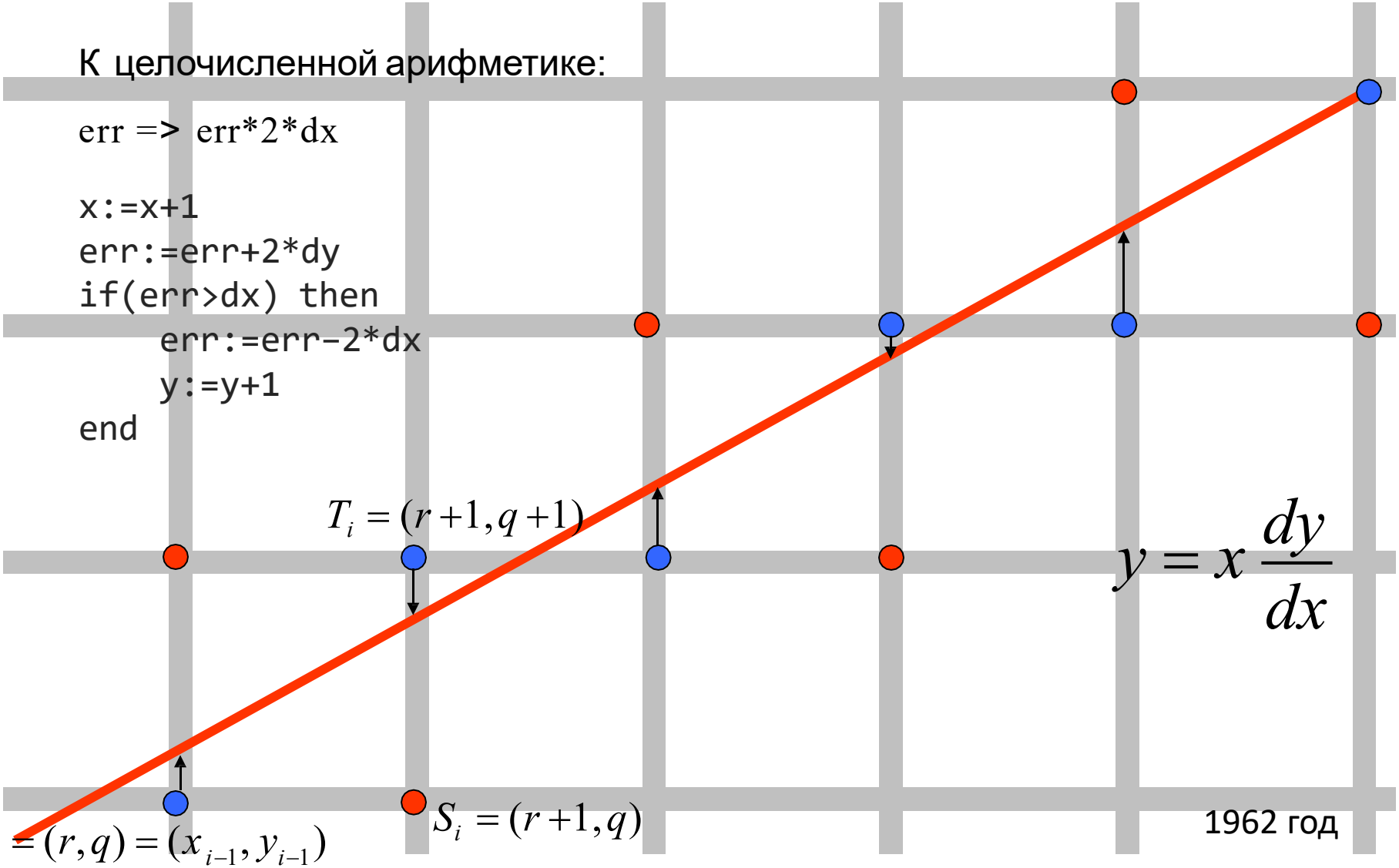


Алгоритм Брезенхэма

К целочисленной арифметике:

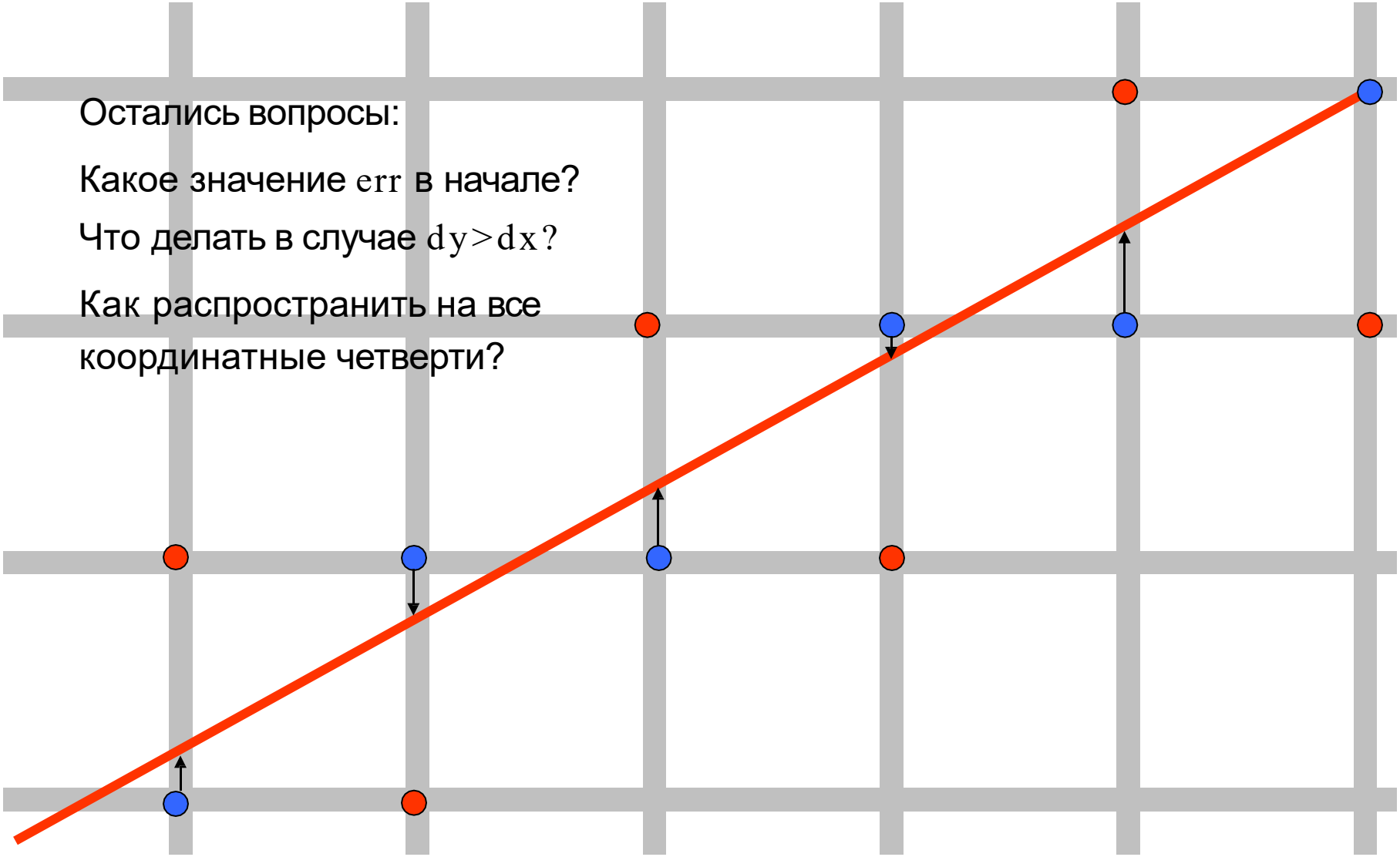
```
err => err*2*dx

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



1962 год

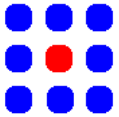
Алгоритм Брезенхэма



Пиксельные области (ПО)

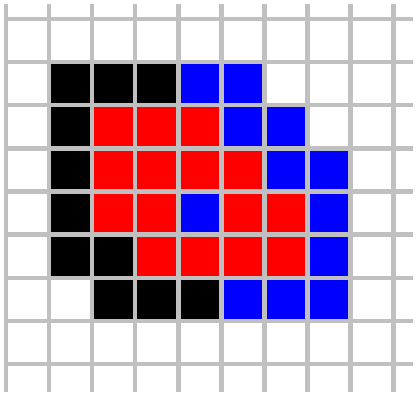
Каждый пиксель определяет значение – это некоторое целое число, т.е. либо индекс палитры, либо значение цвета RGB. Как правило, слово "область" подразумевает связность. Понятие границы (или, что то же самое, – связности) ПО неоднозначно:

4-связность – путь от любого пикселя до любого другого может состоять из 4 элементарных движений,



8-связность – путь от любого пикселя до любого другого может состоять из 8 элементарных движений

Определение области

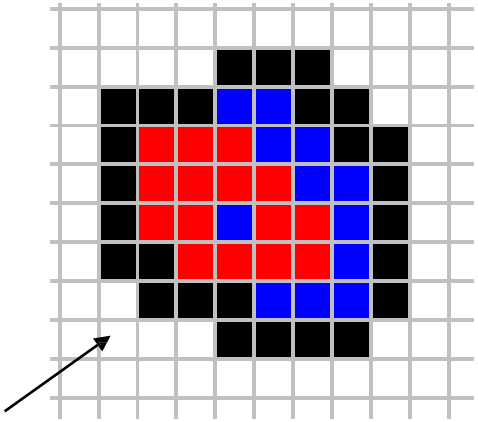


Внутренне определенные ПО. Все внутренние пиксели области имеют одно и то же значение old_value. Граница области может быть многосвязной (напр., дырки). На таких областях работают *внутренне заполняющие* алгоритмы:

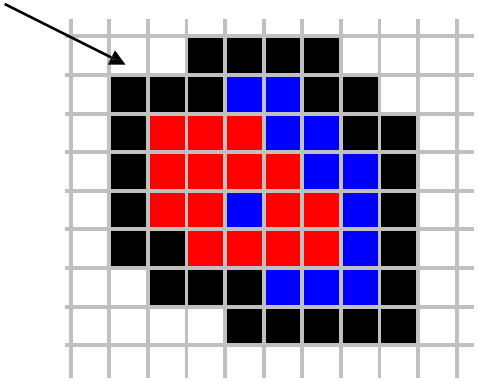
```
if (pixel == old_value)
    pixel = new_value
```

Может быть и более сложный предикат принадлежности к пиксельной области

Определение области



Связность?



Гранично определенные ПО. Пикселы границы имеют значения boundary_value. *Гранично заполняющие* алгоритмы каждому пикселю области присваивают одно и то же значение new_value. Заметим, что многие реализации требуют, чтобы ни один пиксель границы области не имел значения new_value.

Для 8-связных областей необходимо задавать 4-связную границу.

Затравка Seed

- Важная особенность алгоритмов заполнения ПО – им надо задать какой-либо внутренний пиксел области. В этом, например, заключается определенная трудность заполнения областей с криволинейными (параметрические, алгебраические кривые) границами. Хотя с применением интерактивного режима эти задачи решаются очень просто. Сначала рисуем (растеризуем) кривую нужной связности. Затем указываем мышью на любой внутренний пиксель для заливки области.
- И в заключение отметим, что ПО очень важны, поскольку многие редакторы рисунков могут решать различные задачи для отдельных областей, а не для рисунка целиком.

Заполнение шаблоном

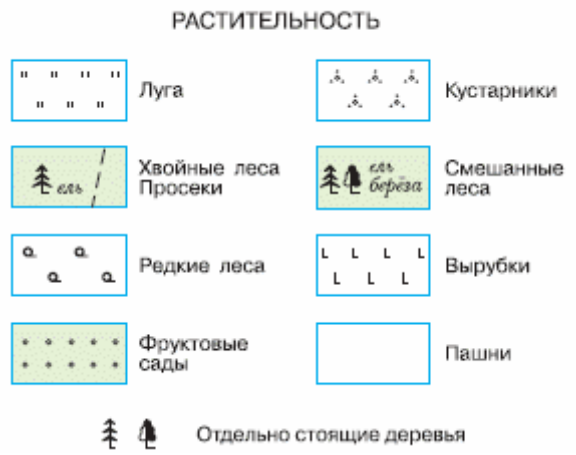
Задача заполнения пиксельной области другим рисунком.

Рассмотрим виртуальную растровую плоскость **P**, точки которой **(x,y)** могут иметь целочисленные координаты от $-\infty$ до $+\infty$. Ограниченная часть ее около начала координат – это наш растр:

R[n × m] = {(x,y), x=0…(n-1), y=0…(m-1)}

В нем выделена пиксельная область **O**. Будем считать, что у нас есть характеристическая функция этой области – **Fo(x,y)**. Таким образом мы можем задавать «независимые от растра» фигуры, которые будут иметь всегда гладкие границы даже при многократном увеличении.

Шаблон –растр обычно меньше – **Pt[s,t]**. точку привязки шаблона – **(x0,y0)** Вся плоскость **P** покрывается шаблоном как кафельной плиткой.

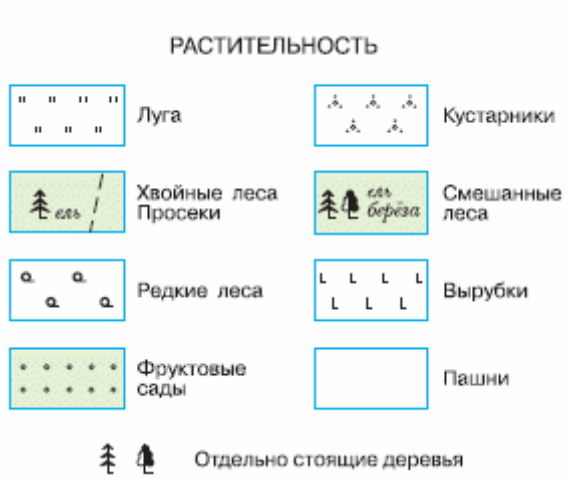


Алгоритм заполнения шаблоном

Вцелом данный метод является простейшим способом текстурирования и лежит в основе всех остальных текстурных покрытий.

Идея алгоритма может быть выражена следующим образом (для несимметричного шаблона):

```
for (j = 0; j < m; j++) // построчное сканирование растра
  for (i = 0; i < n; i++) {
    if (Fo(i, j) == 1) {
      // определить координаты "внутри своей копии плитки"
      is = (i – x0) % s;
      jt = (j – y0) % t;
      // переписать значение из шаблона
      R[i][j] = Pt[is][jt];
    }
  }
```



Алгоритм заполнения шаблоном

Вцелом данный метод является простейшим способом текстурирования и лежит в основе всех остальных текстурных покрытий.

Идея алгоритма может быть выражена следующим образом (для несимметричного шаблона):

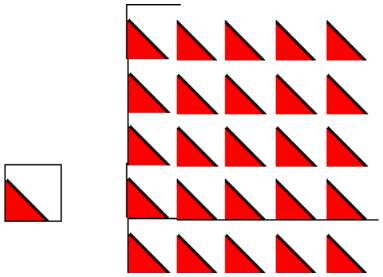
```
for (j = 0; j < m; j++) // построчное сканирование растра
  for (i = 0; i < n; i++) {
    if (Fo(i, j) == 1) {
      // определить координаты "внутри своей копии плитки"
      is = (i – x0) & (s-1);
      jt = (j – y0) & (t-1);
      // переписать значение из шаблона
      R[i][j] = Pt[is][jt];
    }
  }
```

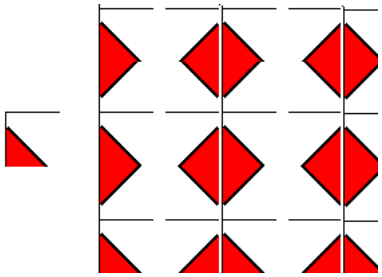
если:

$s = 2^N$

$t = 2^M$

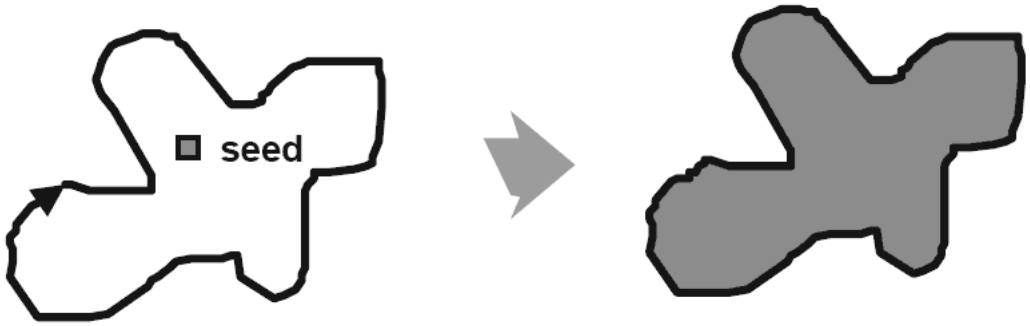
Для симметричных





Такие возможности предлагает, например, OpenGL. Использование симметрии позволяет экономить память под шаблоны. Очевидно, что можно предложить покрытие, когда по горизонтали идет просто повторение, а по вертикали зеркальное отражение, и наоборот. Другими словами, за счет симметрии мы для случая шаблон размером $2N \times 2N$ представили растром размером $N \times N$

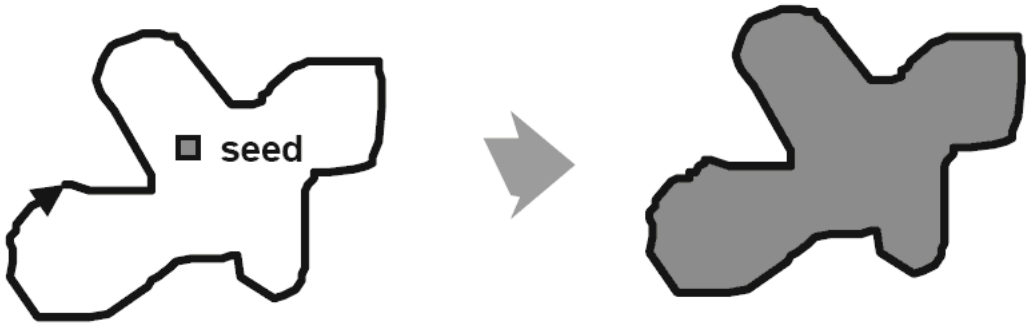
Заливка с затравкой (seed)




```
// Recursive Flood Fill
void FloodFill4(x,y,Oldval, Newval) {
  if(Pixel(x,y) == Oldval) {
    SetPixel(x,y,Newval);
    FloodFill4(x,y+1,Oldval,Newval);
    FloodFill4(x,y-1,Oldval,Newval);
    FloodFill4(x+1,y,Oldval,Newval);
    FloodFill4(x-1,y,Oldval,Newval);
  }
}
```

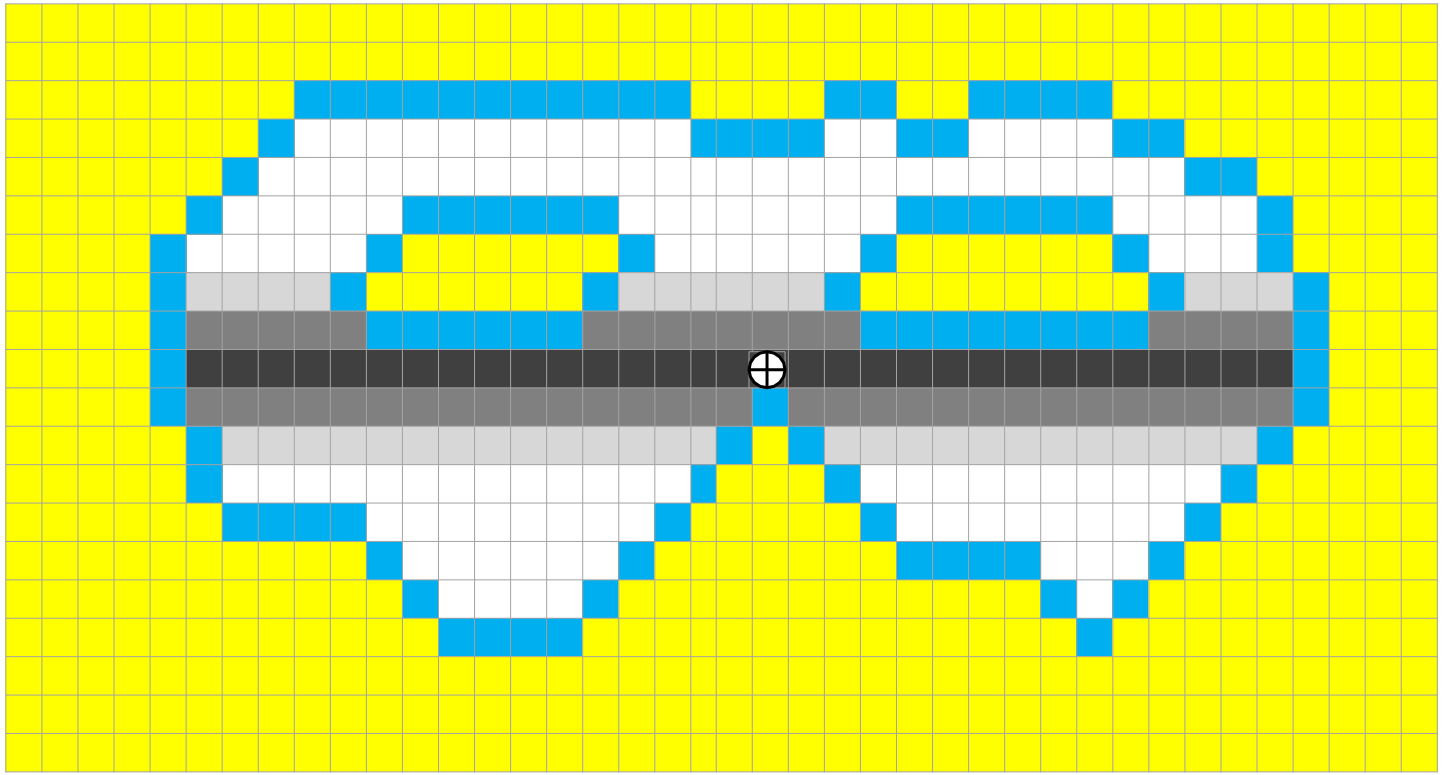
Для 4-х связной области. А как для 8?

Заливка с затравкой (seed)



- **Span filling algorithm**
- Без рекурсии, более эффективен.
- Span – непрерывный горизонтальный «отрезок» из пикселей, которые принадлежат области (имеют значение OV) и ограниченный с обоих концов пикселями, которые не принадлежат ей (имеют другие значения).
- Исходные данные: Пиксельная область ПО с границей. Все пиксели области имеют значение OV. Затравка seed – это пиксель с координатами (Xs,Ys), принадлежащий области.
- Задача: заменить значение OV на значение NV у всех пикселей области

Span



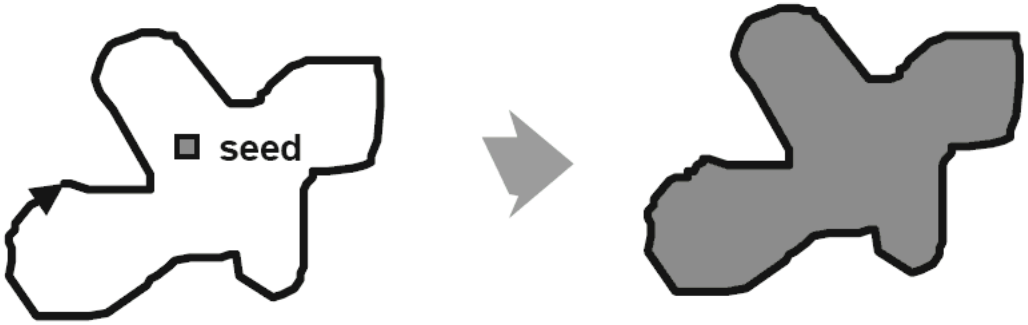
Внешняя ПО (заливать не надо)

Граница ПО

Примеры спанов

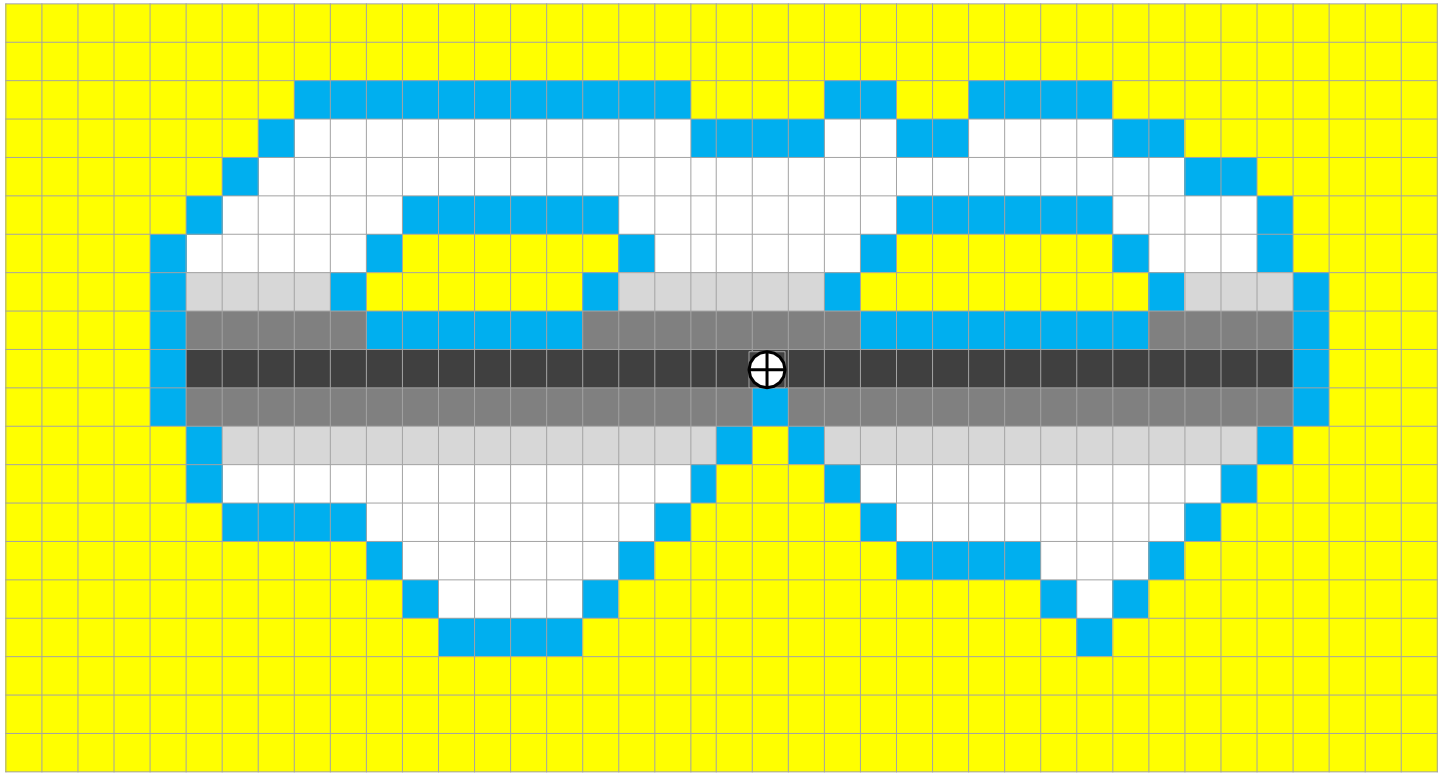
Затравка

Span filling algorithm



- Начало. Определить span, содержащий seed. Поместить его в стек спанов (сначала стек пуст).
- Пока стек не пуст
  - Берем из стека спан C;
  - Заполняем пиксели C значением NV;
  - Поднимаемся вверх на 1 линию растра. Находим все спаны связанные с C (согласно связности 4 или 8) и помещаем их в стек.
  - Опускаемся вниз на 1 линию растра. Находим все спаны связанные с C (согласно связности 4 или 8) и помещаем их в стек.

Span



Внешняя ПО (заливать не надо)

Граница ПО

Примеры спанов

Затравка