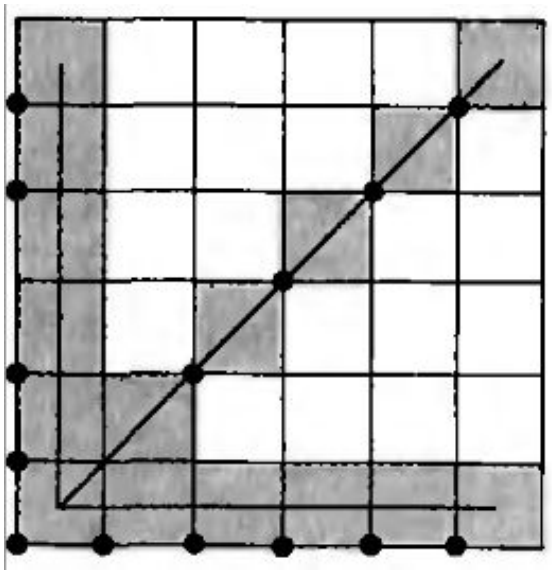
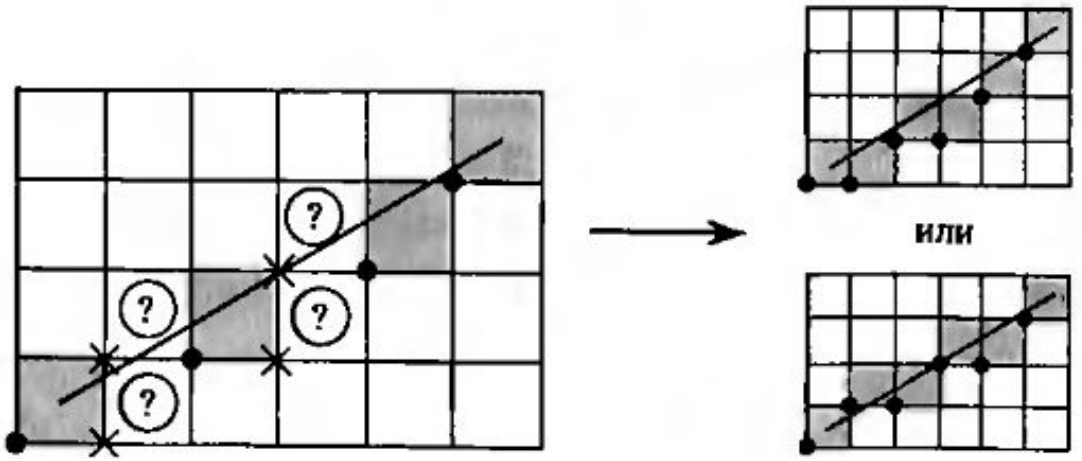
5. Растровая графика

Алгоритмы растровой графики

1. Растеризация графических примитивов.
2. Устранение лестничного эффекта.
3. Заполнение областей.
4. Растеризация сплошных многоугольников.
5. Удаление невидимых линий

Растеризация графических примитивов





Растеризация отрезков

1. Простейший алгоритм
2. Пошаговый алгоритм
3. Алгоритм Брезенхема
4. Алгоритм ЦДА

Уравнение отрезка прямой

(x_1, y_1), (x_2, y_2) – целочисленные координаты начальной и конечной точек отрезка.

$y = ax + b$ – уравнение отрезка прямой.

$y_1 = ax_1 + b$

$y_2 = ax_2 + b$

$a = (y_2 - y_1) / (x_2 - x_1)$

$b = (y_1x_2 - x_1y_2) / (x_2 - x_1)$

Простейший алгоритм

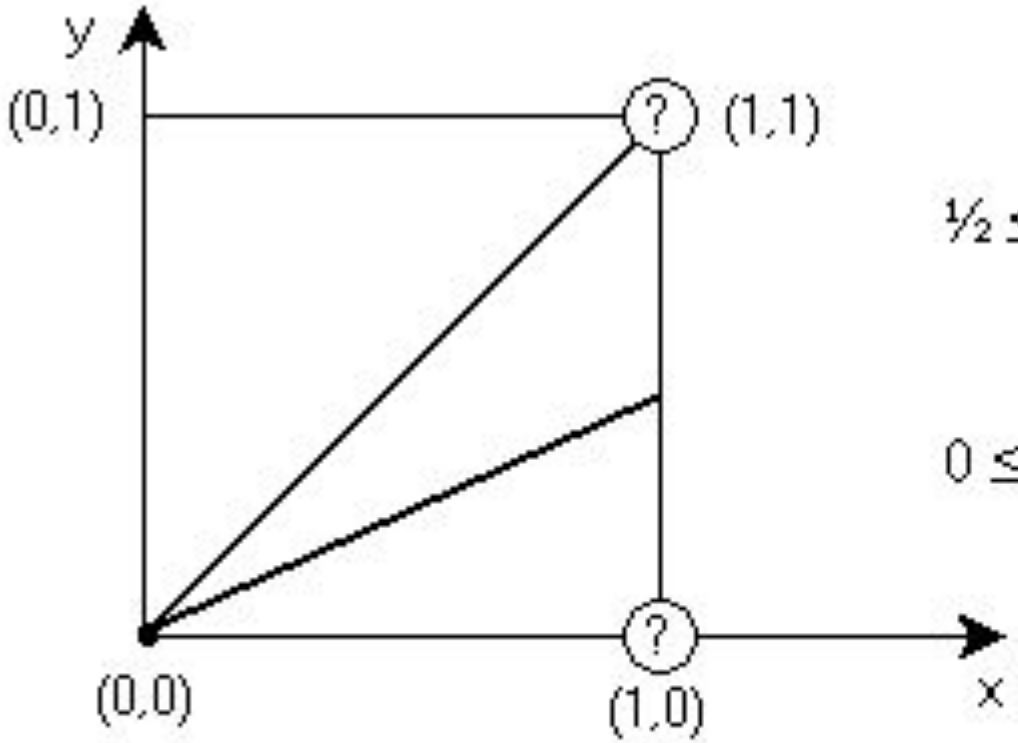
$x = x_1$;
$y = y_1$;
while ($x \le x_2$)
{
  PutPixel(x, y);
  $x = x + 1$;
  y = Round($ax + b$);
}

Пошаговый алгоритм

$x = x_1$;
$y = y_1$;
while ($x \le x_2$)
{
  PutPixel(x, y);
  $x = x + 1$;
  y = Round($y + a$);
}

Алгоритм Брезенхема

1-й октант: угловой коэффициент лежит в диапазоне от 0 до 1



$$\frac{1}{2} \le \frac{\Delta y}{\Delta x} \le 1 \text{ (ошибка } \ge 0)$$

$$0 \le \frac{\Delta y}{\Delta x} \le \frac{1}{2} \text{ (ошибка } < 0)$$

Инициализировать ошибку в $-\frac{1}{2}$

$$\text{ошибка} = \text{ошибка} + \frac{\Delta y}{\Delta x}$$

Алгоритм Брезенхема (1-й октант)

$x = x_1$; $y = y_1$; $\Delta x = x_2 - x_1$; $\Delta y = y_2 - y_1$; $e = \Delta y / \Delta x - 0.5$;
for ($i = 1$; $i \le \Delta x$; $i = i + 1$)
{
    PutPixel(x, y);
    while (e ≥ 0)
    {
     $y = y + 1$;
     $e = e - 1$;
    }
    $x = x + 1$;
    $e = e + \Delta y / \Delta x$;
}

Целочисленный алгоритм Брезенхема (1-й октант)

$x = x_1$; $y = y_1$; $\Delta x = x_2 - x_1$; $\Delta y = y_2 - y_1$; $e = 2 * \Delta y - \Delta x$;
for ($i = 1$; $i \le \Delta x$; $i = i + 1$)
{
    PutPixel(x, y);
    while ($e \ge 0$)
    {
     $y = y + 1$;
     $e = e - 2 * \Delta x$;
    }
    $x = x + 1$;
    $e = e + 2 * \Delta y$;
}

Общий алгоритм Брезенхема

$x = x_1$; $y = y_1$; $\Delta x = |x_2 - x_1|$; $\Delta y = |y_2 - y_1|$;
$s_1 = \text{Sign}(x_2 - x_1)$; $s_2 = \text{Sign}(y_2 - y_1)$;
$d = 0$;
if ($\Delta y > \Delta x$) { Swap(Δx, Δy); $d = 1$; }
$e = 2 * \Delta y - \Delta x$;
for ($i = 1$; $i \le \Delta x$; $i = i + 1$)
{ PutPixel(x, y);
  while (e ≥ 0)
  {if ($d = 1$) $x = x + s_1$; else $y = y + s_2$;
   $e = e - 2 * \Delta x$;
  }
  if ($d = 1$) $y = y + s_2$; else $x = x + s_1$;
  $e = e + 2 * \Delta y$;
}

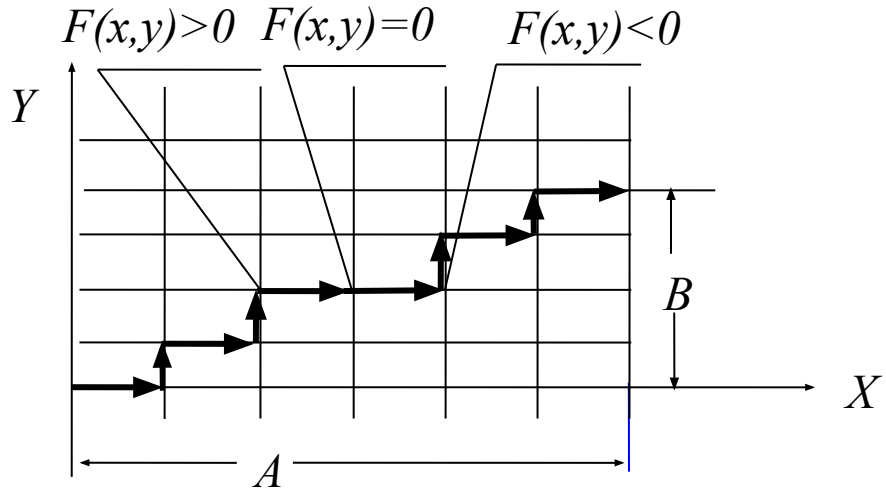
Алгоритм ЦДА

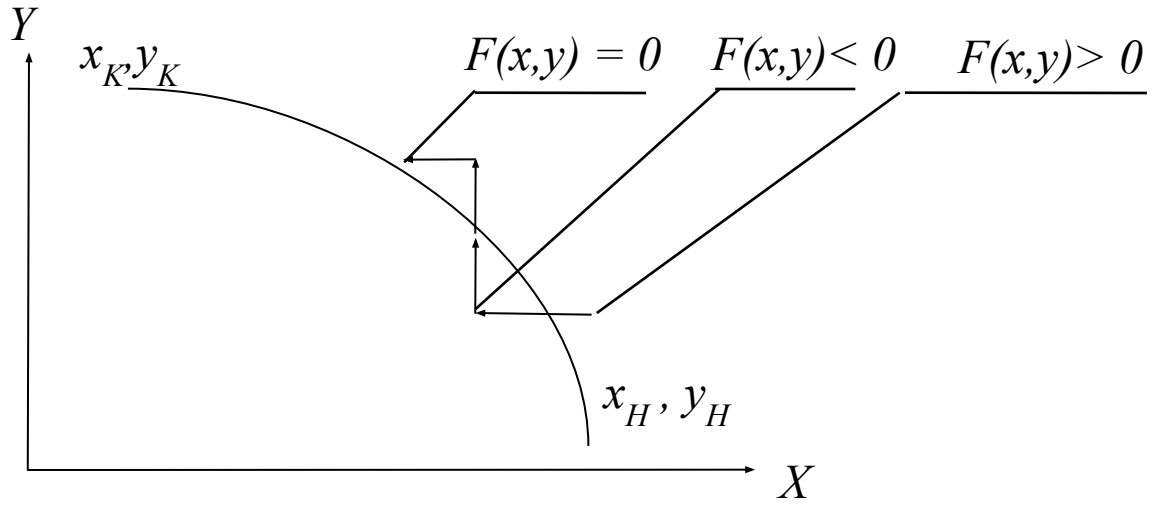
$L = \mathrm{Max}(|x_2 - x_1|, |y_2 - y_1|);$
$\Delta x = (x_2 - x_1) / L;\ \Delta y = (y_2 - y_1) / L;$
$x = x_1 + 0.5 * \mathrm{Sign}(\Delta x);$
$y = y_1 + 0.5 * \mathrm{Sign}(\Delta y);$
for ($i = 1;\ i \le L;\ i = i + 1$)
{
    PutPixel(Round(x), Round(y));
    $x = x + \Delta x;$
    $y = y + \Delta y;$
}

Методы растеризации графических примитивов

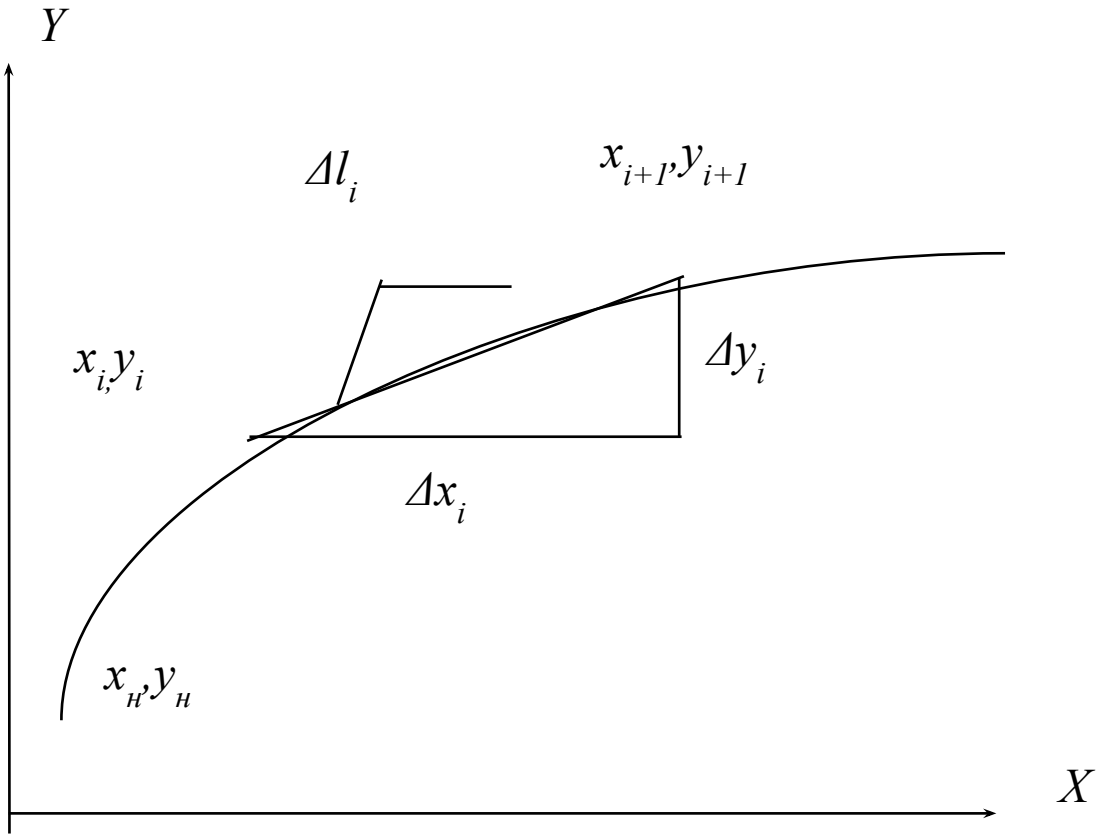
1. Метод оценочной функции
2. Метод цифровых дифференциальных анализаторов

Метод оценочной функции



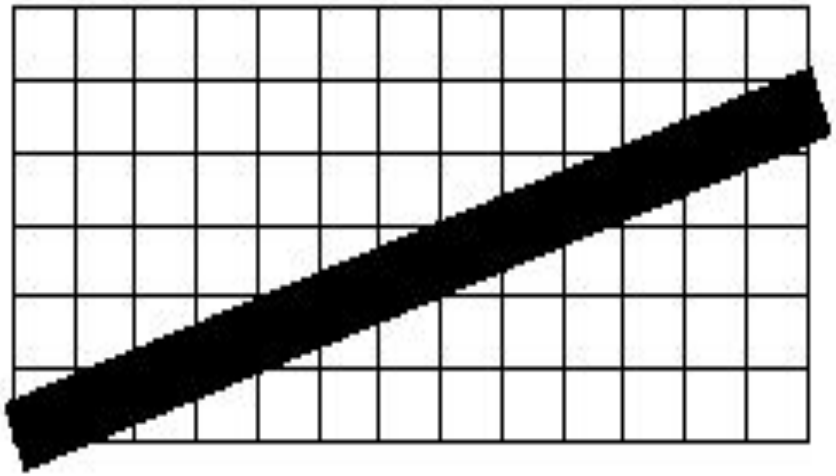


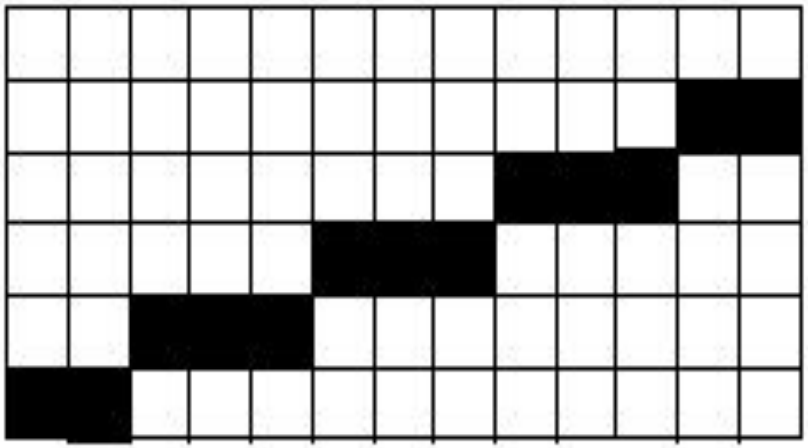
Метод цифровых дифференциальных анализаторов

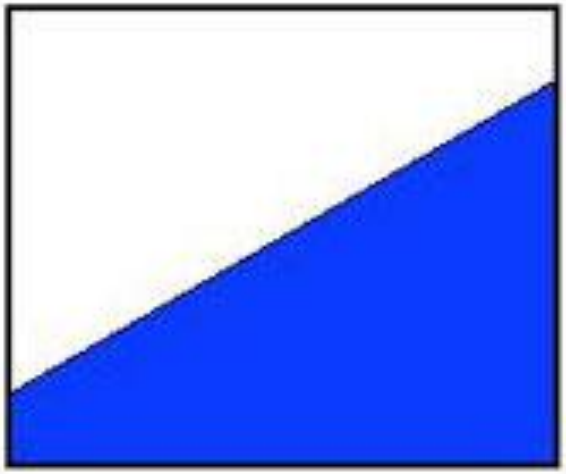


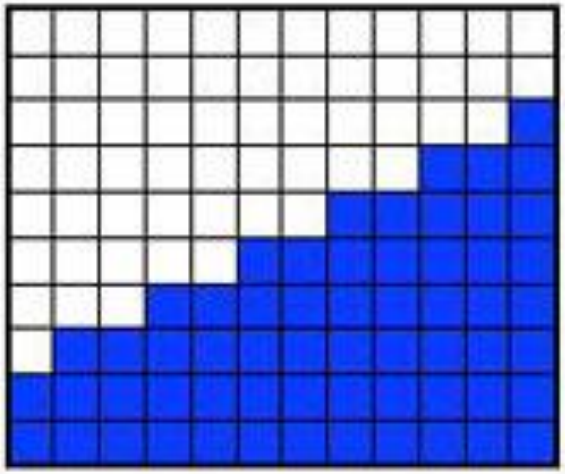
$$\sqrt{(\Delta x_i)^2 + (\Delta y_i)^2} <= 1 \qquad \Delta x_i = \frac{F_y^/}{\sqrt{(F_x^/)^2 + (F_y^/)^2}} \qquad \Delta y_i = -\frac{F_x^/}{\sqrt{(F_x^/)^2 + (F_y^/)^2}}$$

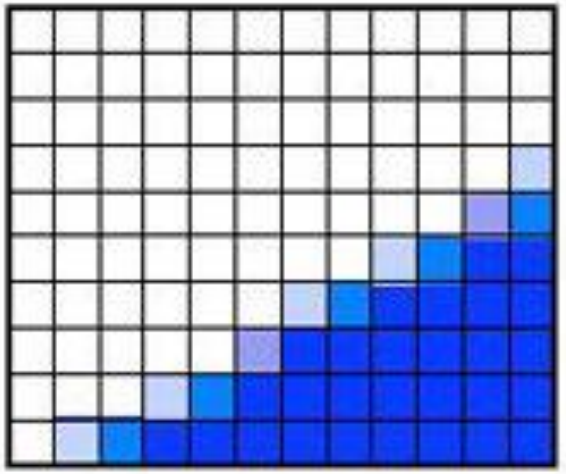
Лестничный эффект











Устранение лестничного эффекта

1. Выравнивание – каждый пиксел высвечивается с яркостью, пропорциональной площади пиксела, которую занимает отрезок.
2. Изменение разрешения – подготовка изображения высокого разрешения (кратного реальному) с последующим масштабированием и использованием сглаживающего фильтра.
3. Учет наклона отрезка – изменение яркости в зависимости от наклона отрезка (максимум у вертикального отрезка – 1, минимум у горизонтального отрезка – 0.707).

Алгоритм Ву

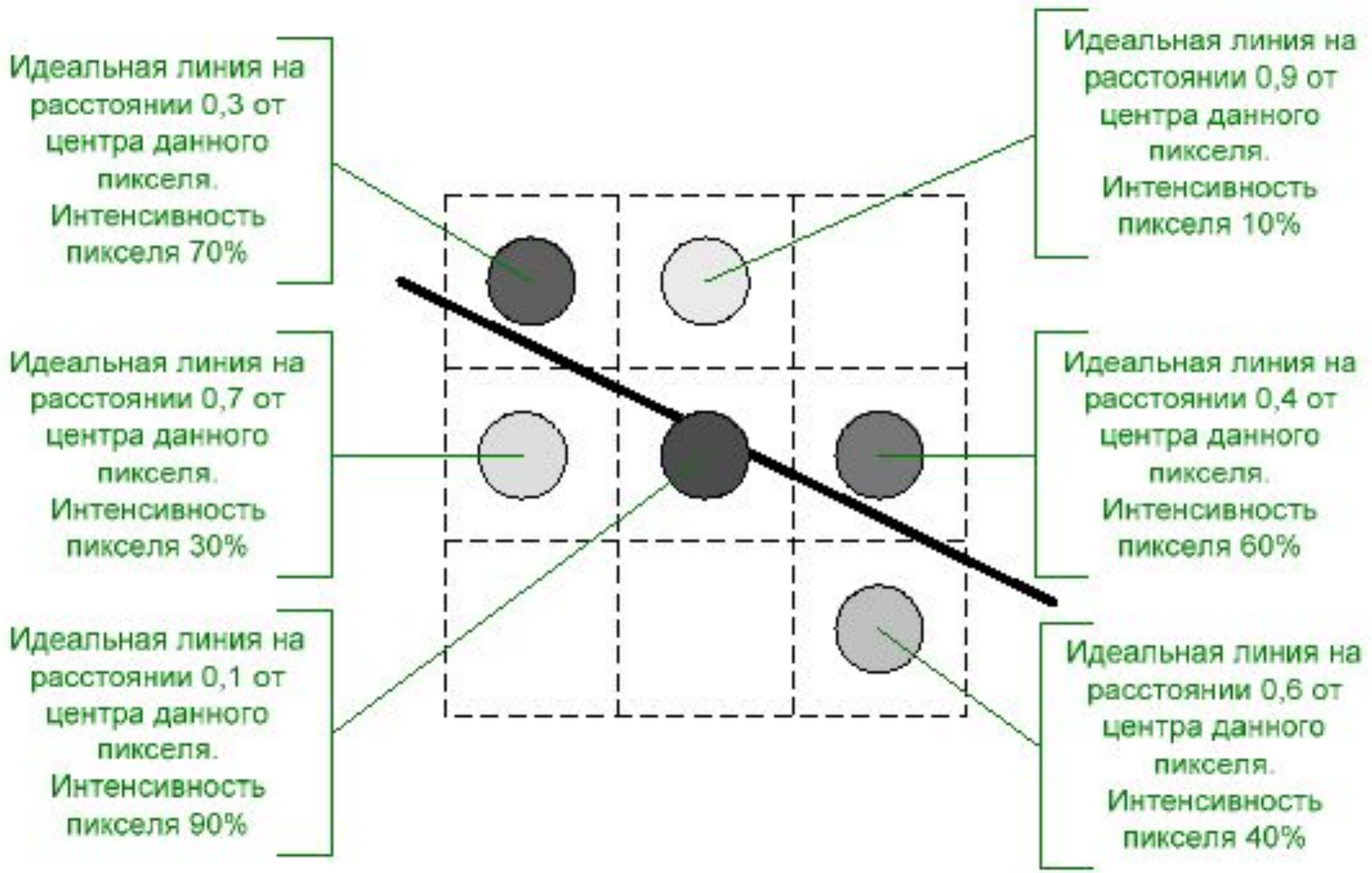
При рисовании линий обычным образом с каждым шагом по основной оси высвечиваются два пиксела по неосновной оси.

Их интенсивность подбирается пропорционально расстоянию от центра пиксела до идеальной линии – чем дальше пиксел, тем меньше его интенсивность. Значения интенсивности двух пикселов дают в сумме 100 %, т.е. интенсивность одного пиксела, точно попавшего на идеальную линию.

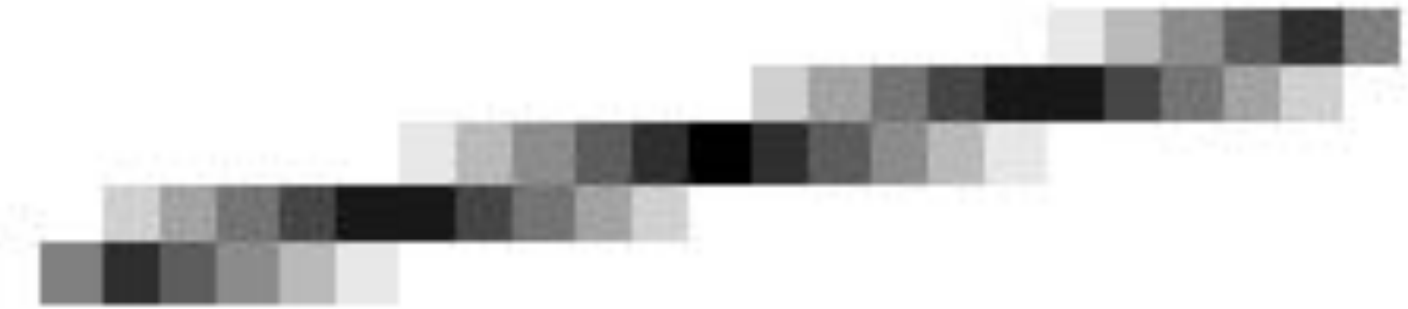
Горизонтальные, вертикальные и диагональные линии не сглаживаются.

Т.о. учитываются особенности человеческой зрительной системы.

Алгоритм Ву


Идеальная линия на расстоянии 0,3 от центра данного пикселя. Интенсивность пикселя 70%
Идеальная линия на расстоянии 0,9 от центра данного пикселя. Интенсивность пикселя 10%
Идеальная линия на расстоянии 0,7 от центра данного пикселя. Интенсивность пикселя 30%
Идеальная линия на расстоянии 0,4 от центра данного пикселя. Интенсивность пикселя 60%
Идеальная линия на расстоянии 0,1 от центра данного пикселя. Интенсивность пикселя 90%
Идеальная линия на расстоянии 0,6 от центра данного пикселя. Интенсивность пикселя 40%

Алгоритм By

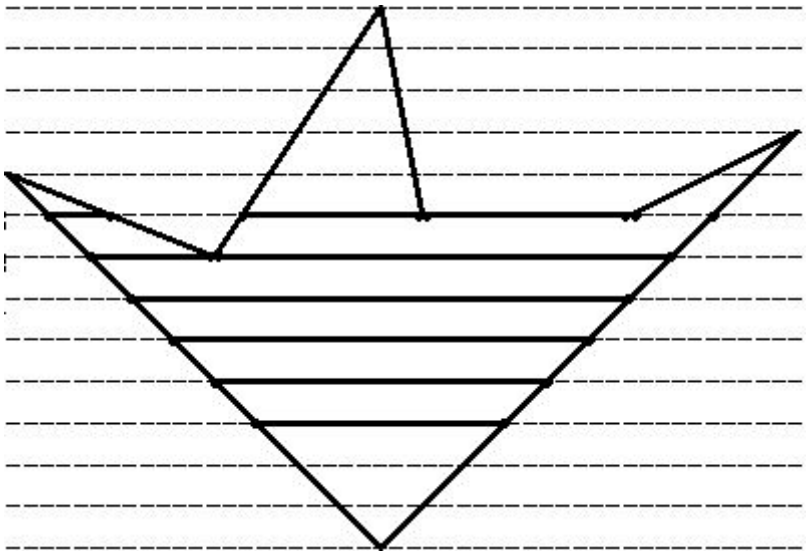


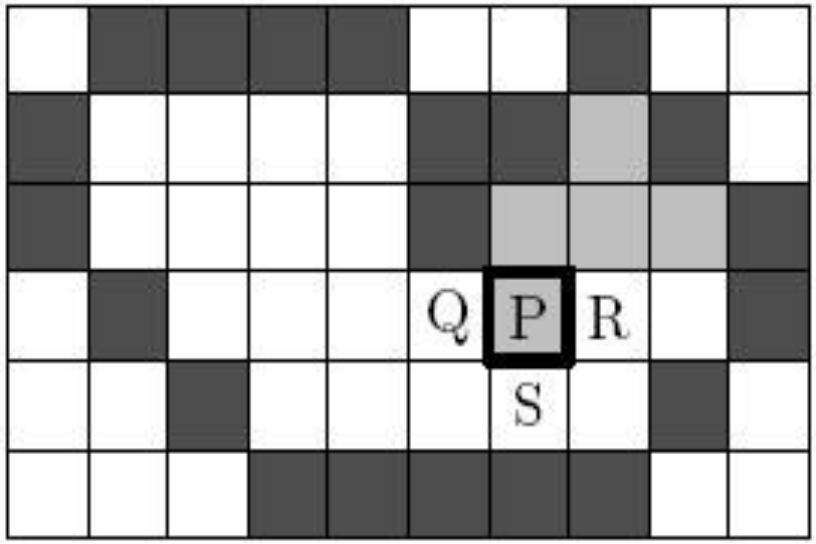
Целочисленный алгоритм Ву

```
// Координаты концов отрезка - (0, 0) и (a, b), a > 0, b > 0, b < a
// plot(x,y,I) закрашивает пиксель (x, y) с интенсивностью I
// I0 - максимальная интенсивность (2 ^ m - 1)
x0 = 0; x1 = a; y0 = 0; y1 = b;
plot(x0,y0,I0); plot(x1,y1,I0);
D = 0;
d = floor( (b / a) * 2 ^ n + 0.5 ); // меньшее целое
while ( x0 < x1 )
{     D = D + d;
      if( произошло переполнение D )  { y0++; y1--; }
      I1 = D / 2 ^ (n - m); // битовый сдвиг вправо на n - m
      I2 = двоичное_дополнение( I1 );
      plot(x0, y0, I1); plot(x0, y0+1, I2);
      plot(x1, y1, I1); plot(x1, y1-1, I2);
      x0++; x1--;
}
```

Заполнение областей

1. Построчное сканирование
2. Заполнение с затравкой





Построчное сканирование

1. Имеется область, граница которой разложена в растр.
2. Внутри задана точка.
3. Заданы значения: *a* – граничных пикселов, *b* – внутренних пикселов до заполнения, *c* – внутренних пикселов после заполнения.
4. Объект заключается в прямоугольную оболочку.
5. Проводится построчное сканирование прямоугольной оболочки: в строке находится пиксел со значением *a*, затем пиксел, следующий за ним и имеющий значение *b*, которое меняется на *c*, и так до тех пор, пока не будет встречен еще один пиксел со значением *a*, после чего осуществляется переход на следующую строку.

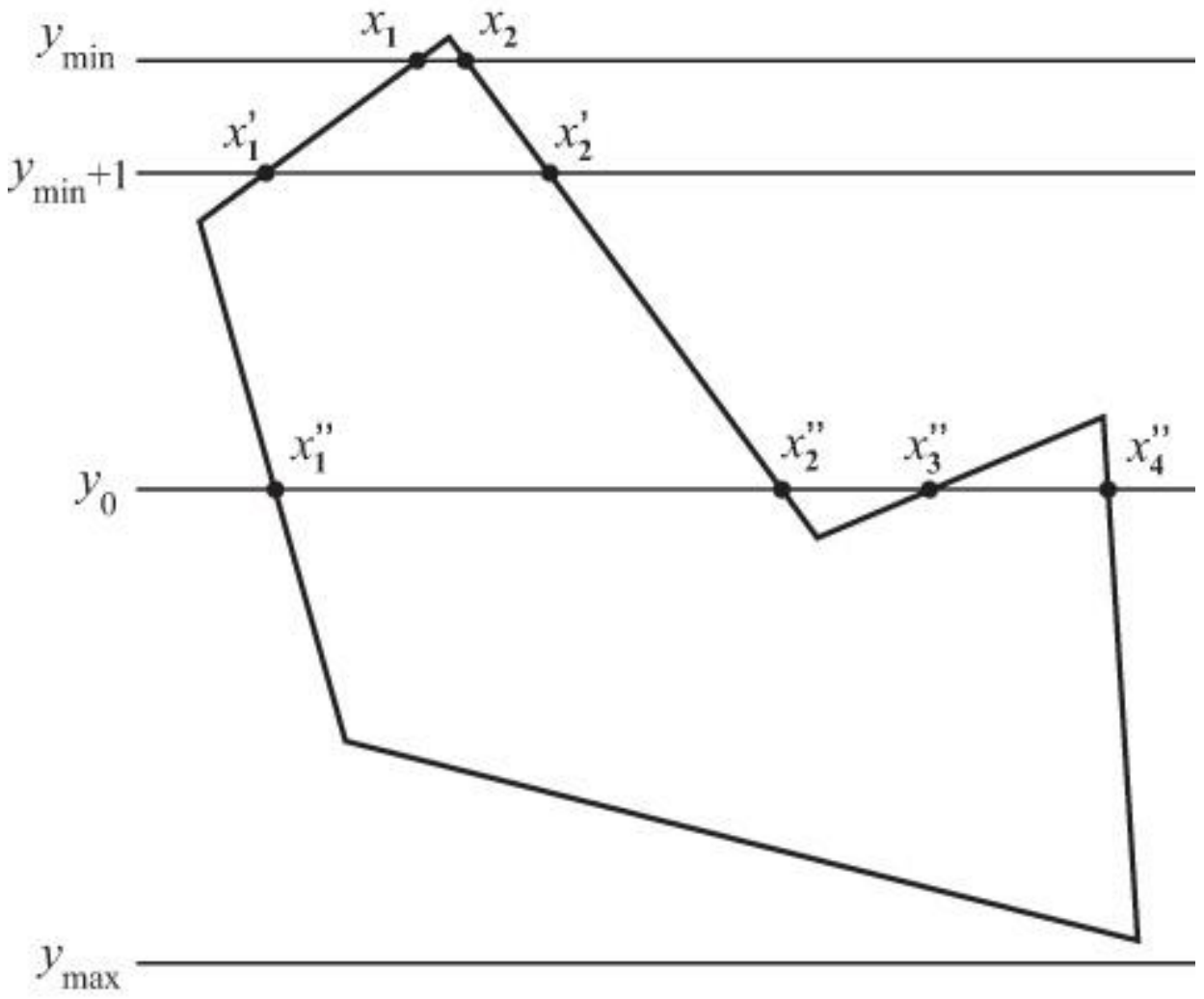
Заполнение с затравкой

1. Указать затравочный пиксел внутри контура.
2. Поместить затравочный пиксел в стек.
3. Пока стек не пуст:
   1) извлечь пиксел из стека;
   2) присвоить пикселу требуемое значение;
   3) для каждого из соседних четырехсвязных пикселов проверить:
      - Является ли он граничным;
      - Не присвоено ли ему требуемое значение;
   4) проигнорировать пиксел в любом из этих двух случаев, иначе поместить пиксел в стек.

Растеризация сплошных многоугольников

1. Растеризация всех негоризонтальных ребер многоугольника.
2. Все точки помещаются в списки.
3. Для каждой координаты y_{min}, y_2, …, y_{max} сопоставляется список *x*-координат всех пикселов, закрашенных при растеризации ребер, которые находятся на горизонтали *y*.

Растеризация сплошных многоугольников



Удаление невидимых линий

1. Алгоритмы, работающие в объектном пространстве.
2. Алгоритмы, работающие в пространстве изображения.

Алгоритм Z-буфера

1. Заполнить буфер кадра фоновым значением интенсивности или цвета.
2. Заполнить *z*-буфер минимальным значением *z*.
3. Преобразовать каждый многоугольник в растровую форму в произвольном порядке.
4. Для каждого пиксела с координатами (*x, y*) в многоугольнике вычислить его глубину *z*(*x, y*).
5. Сравнить глубину *z*(*x, y*) со значением *a*, хранящимся в *z*-буфере в этой же позиции.
6. Если *z*(*x,y*) > *a*, то записать атрибут этого многоугольника (интенсивность, цвет и т. п.) в буфер кадра и заменить значение *a* в *z*-буфере на *z*(*x,y*). В противном случае никаких действий не производить.