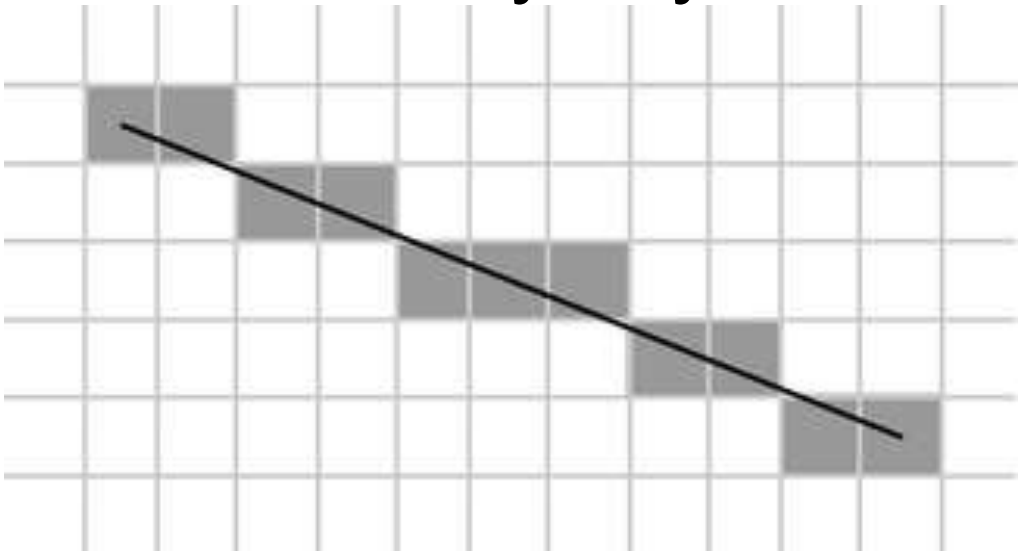
Растровые алгоритмы

Растеризация (Rasterisation)

- **Растеризация** — это перевод двухмерного изображения, описанного векторным форматом в пиксели или точки, для вывода на дисплей или принтер. Процесс, обратный векторизации.
- **Растеризация отрезка** — это задача определения какие точки двумерного растра нужно закрасить, чтобы получить близкое приближение прямой линии между двумя заданными точками.



Связанность

- ***Связность*** – возможность соединения двух пикселей растровой линией, т. е. последовательным набором пикселей.
- 1. Четырехсвязность: пиксели считаются соседними, если либо их *x*-координаты, либо их *y* – координаты отличаются на единицу:
- $|x_1 - x_2| + |y_1 - y_2| \le 1$;
- 2. Восьмисвязность: пиксели считаются соседними, если их *x*-координаты и *y*-координаты отличаются не более чем на единицу:
- $|x_1 - x_2| \le 1, \ |y_1 - y_2| \le 1$.

Растровое представление отрезка

Рассмотрим задачу построения растрового изображения отрезка, соединяющего точки $A(x_a, y_a)$ и $B(x_b, y_b)$. Для простоты будем считать, что

$0 \le y_b - y_a \le x_b - x_a$. Тогда отрезок описывается уравнением:

$y = y_a + \frac{y_b - y_a}{x_b - x_a} (x - x_a)$, $x \in [x_a, x_b]$ или $y = kx + b$.

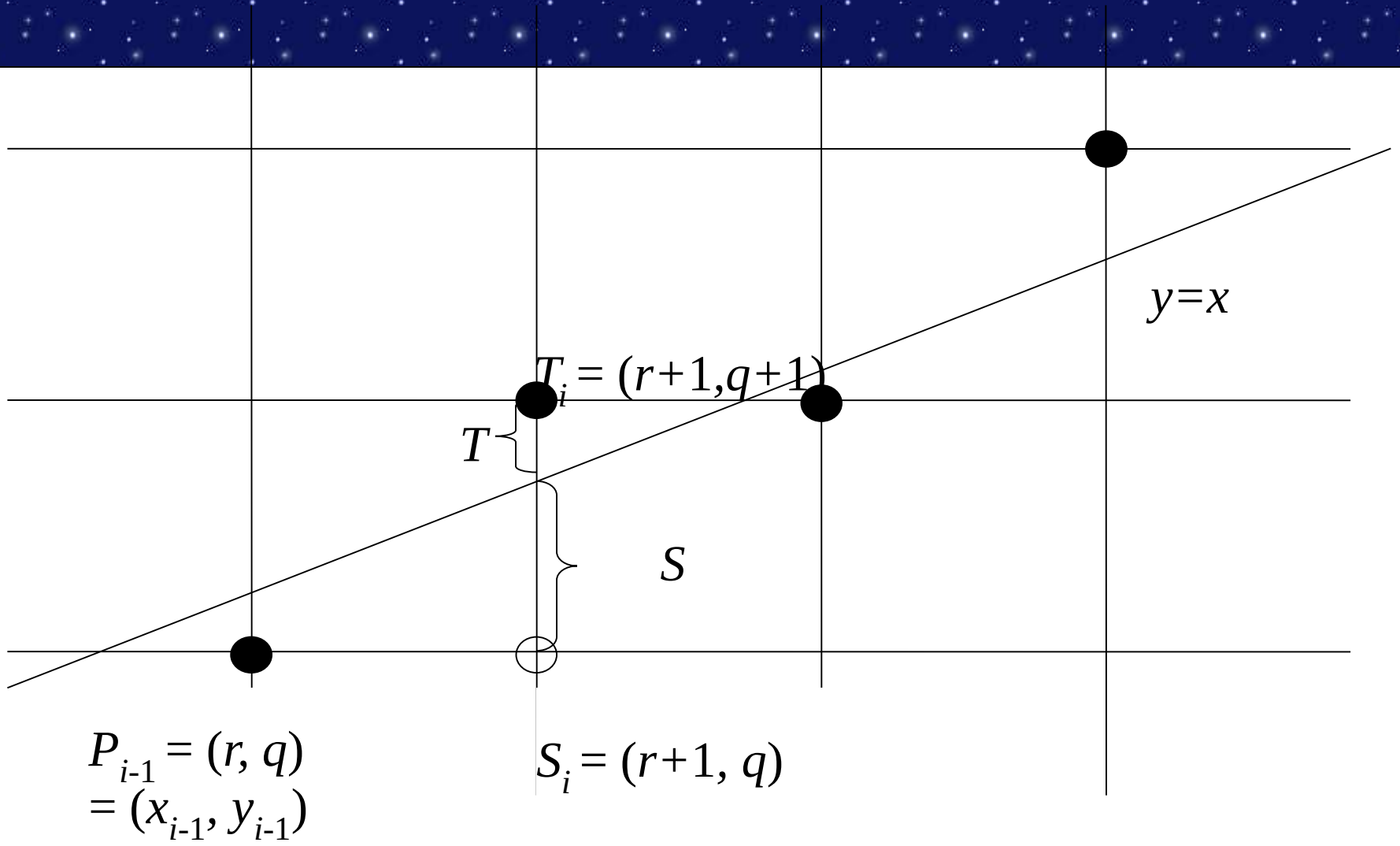
```
void line(int xa, int ya, int xb, int yb, int color)
{double k = ((double)(yb – ya)) / (xb – xa);
double b = ya – k * xa;
for (int x = xa; x <= xb; x++)
putpixel(x, (int)(k * x + b), color);}
```

Растровое представление отрезка

- Вычислений значений функции $y = kx + b$ можно избежать, используя в цикле рекуррентные соотношения, так как при изменении x на 1 значение y меняется на k:

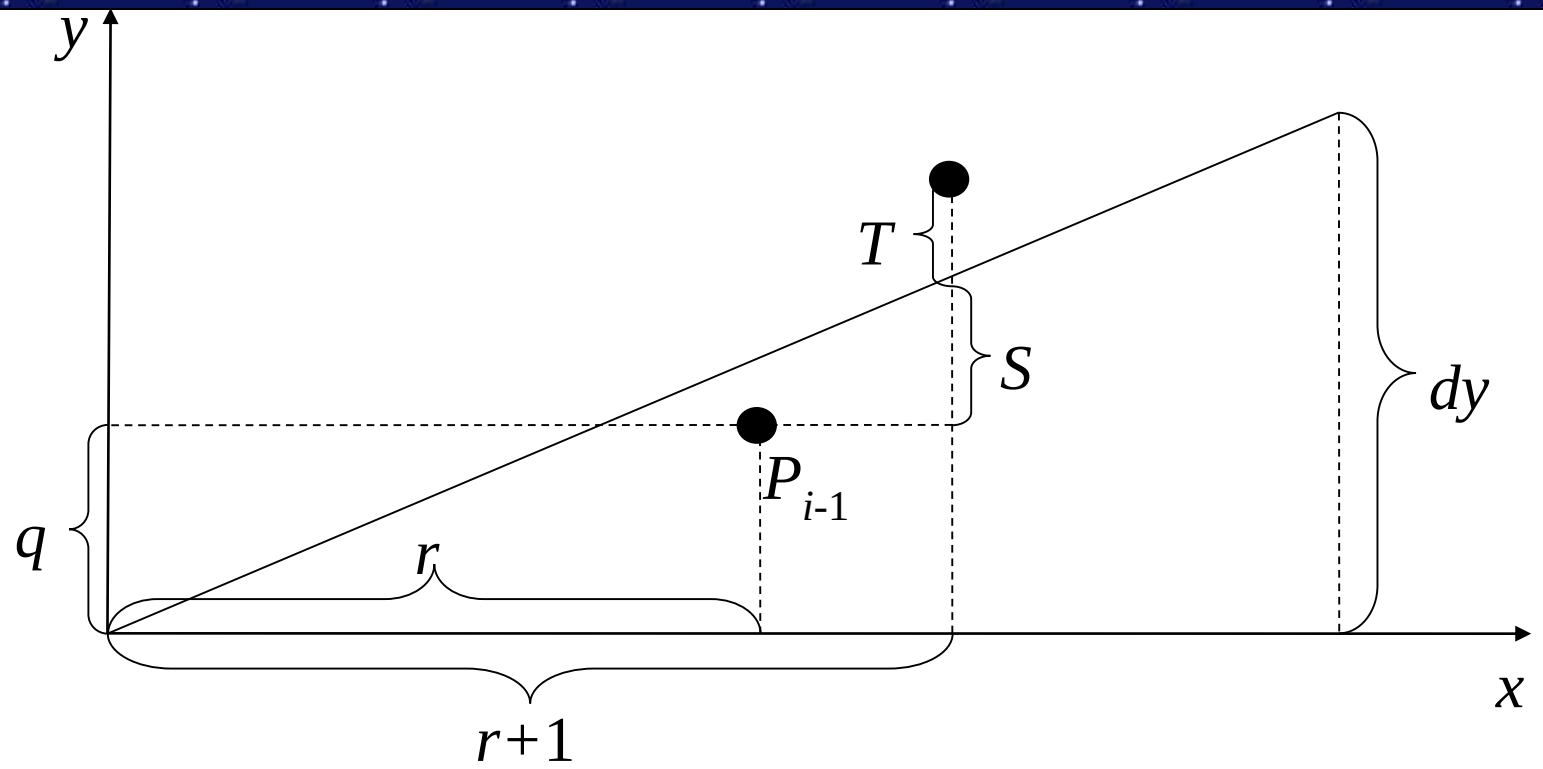
```
void line(int xa, int ya, int xb, int yb, int color)
{double k = ((double)(yb – ya)) / (xb – xa);
double y = ya;
for (int x = xa; x <= xb; x++, y += k)
putpixel(x, (int)y, color);}
```

Алгоритм Брезенхейма



- В алгоритме используется управляющая переменная d_i, которая на каждом шаге пропорциональна разности между S и T. Если $S < T$, то S_i ближе к отрезку, иначе выбирается T_i.

Алгоритм Брезенхейма



Пусть отрезок проходит из точки (x_1, y_1) в точку (x_2, y_2). Перенесем оба конца отрезка с помощью преобразования $T(-x_1, -y_1)$, так чтобы первый конец отрезка совпал с началом координат. Начальной точкой отрезка стала точка (0, 0), конечной точкой – (dx, dy), где $dx = x_2 - x_1$, $dy = y_2 - y_1$

Координаты P_{i-1} после переноса есть (r, q).

Тогда $S_i = (r+1, q)$ и $T_i = (r+1, q+1)$.

Из подобия треугольников можно записать, что $dy/dx=(S+q)/(r+1)$

Алгоритм Брезенхейма

Выразим S:

$$S = \frac{dy}{dx}(r + 1) - q.$$

T можно представить как $T = 1 - S$. Используем предыдущую формулу

$$T = 1 - S = 1 - \frac{dy}{dx}(r + 1) + q.$$

Найдем разницу $S - T$:

$$S - T = \frac{dy}{dx}(r + 1) - q - 1 + \frac{dy}{dx}(r + 1) - q = 2\frac{dy}{dx}(r + 1) - 2q - 1.$$

Помножим левую и правую часть на dx:

$$dx\,(S - T) = 2\,dy\,(r + 1) - 2\,q\,dx - dx = 2\,(r\,dy - q\,dx) + 2\,dy - dx.$$

Величина dx положительная, поэтому неравенство $dx\,(S - T) < 0$ можно использовать в качестве проверки при выборе S_i. Обозначим $d_i = dx\,(S - T)$, тогда

$$d_i = 2\,(r\,dy - q\,dx) + 2\,dy - dx.$$

Алгоритм Брезенхейма

Поскольку $r = x_{i-1}$ и $q = y_{i-1}$, то

$$d_i = 2\,x_{i-1}\,dy - 2\,y_{i-1}\,dx + 2\,dy - dx.$$

Прибавляя 1 к каждому индексу найдем d_{i+1}:

$$d_{i+1} = 2\,x_i\,dy - 2\,y_i\,dx + 2\,dy - dx.$$

Вычитая d_i из d_{i+1} получим

$$d_{i+1} - d_i = 2\,\mathrm{dy}\,(x_i - x_{i-1}) - 2\,\mathrm{dx}\,(y_i - y_{i-1}).$$

Известно, что $x_i - x_{i-1} = 1$, тогда

$$d_{i+1} - d_i = 2\,\mathrm{dy} - 2\,\mathrm{dx}\,(y_i - y_{i-1}).$$

Отсюда выразим d_{i+1}:

$$d_{i+1} = d_i + 2\,\mathrm{dy} - 2\,\mathrm{dx}\,(y_i - y_{i-1}).$$

Алгоритм Брезенхейма

Таким образом, получили итеративную формулу вычисления управляющего коэффициента d_{i+1} по предыдущему значению d_i. С помощью управляющего коэффициента выбирается следующий пиксель – S_i или T_i.

Если $d_i \geq 0$, тогда выбирается T_i и $y_i = y_{i-1} + 1$, $d_{i+1} = d_i + 2\,(dy - dx)$.

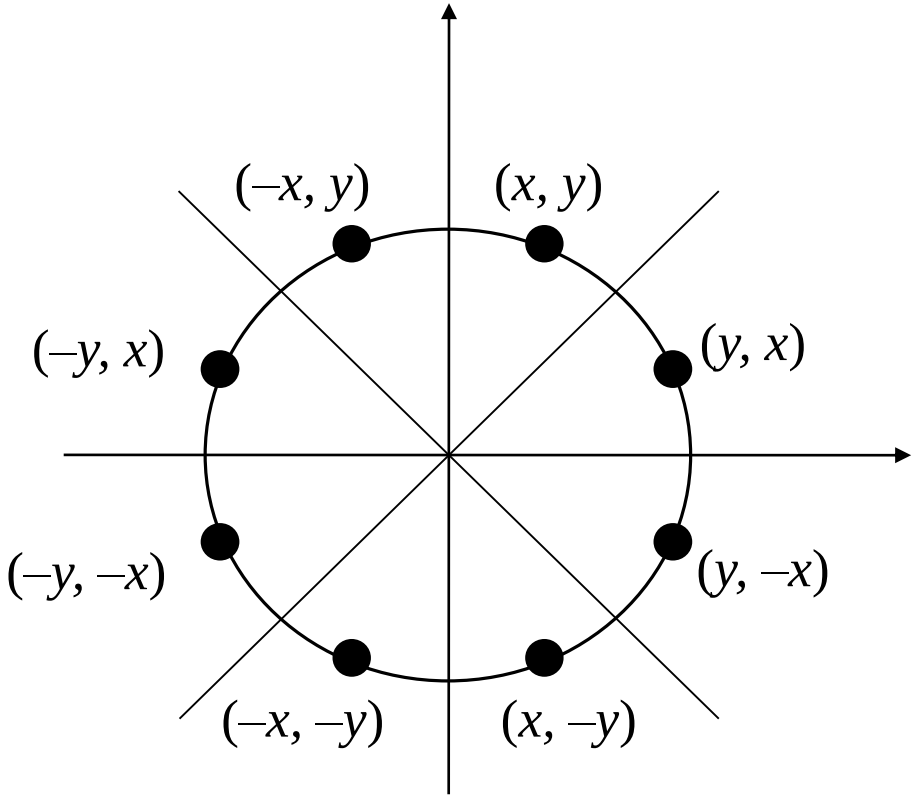
Если $d_i < 0$, тогда выбирается S_i и $y_i = y_{i-1}$ и $d_{i+1} = d_i + 2\,dy$.

Начальные значения d_1 с учетом того, что $(x_0, y_0) = (0, 0)$,

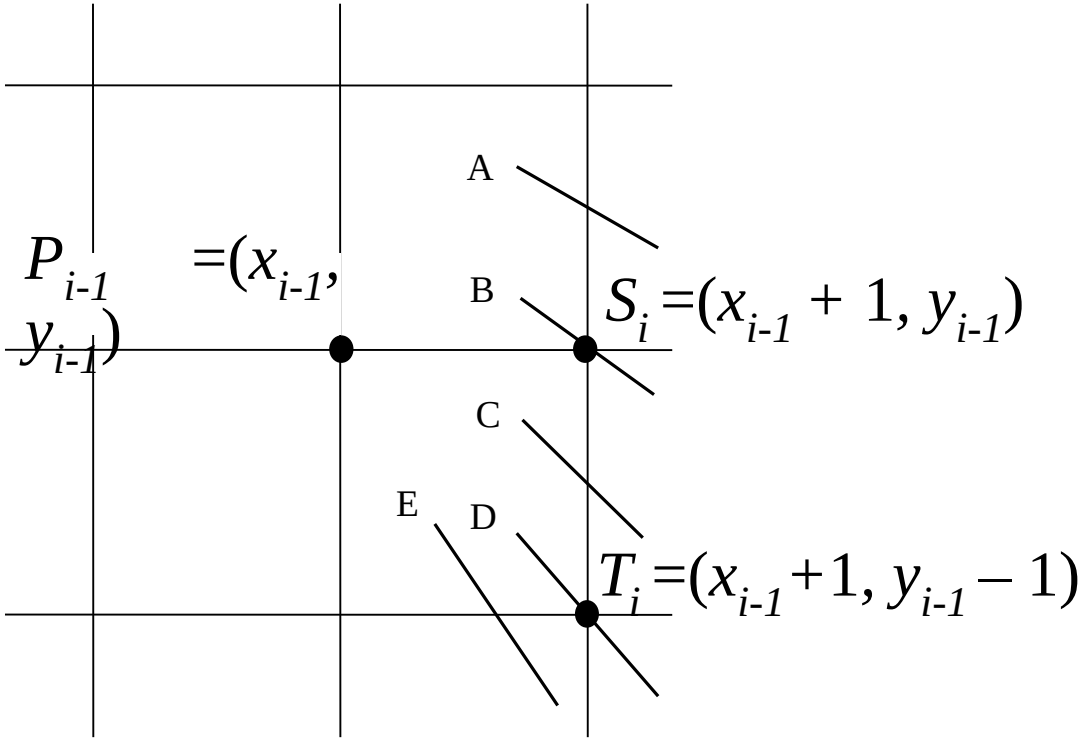
$$d_1 = 2\,dy - dx.$$

Растровая развертка окружности

- $y = \pm\sqrt{R^2 - x^2}$
- $x = R\cos\alpha$, $y = R\sin\alpha$
- Восьмисторонняя симетрия:



Алгоритм Брезенхейма для окружности



$D_1 = 3 - 2R$

Если выбирается S_i (когда $d_i < 0$), то $d_{i+1} = d_i + 4x_{i-1} + 6$

Если выбирается T_i (когда $d_i \geq 0$), то

$d_{i+1} = d_i + 4(x_{i-1} - y_{i-1}) + 10$

Построение кривых

- **Интерполяция** - построение кривой, проходящей через контрольные точки и обладающей некими дополнительными свойствами (часто гладкостью);
- **Аппроксимация** - приближение кривой (не обязательно проходит точно через данные точки, но удовлетворяет некоторому заданному свойству относительно этих точек).
- В настоящее время для задач аппроксимации широко применяются **кривые Безье**. Это связано с их удобством как для аналитического описания, так и для наглядного геометрического построения (это означает, что пользователь может задавать форму кривой интерактивно, т.е. двигая опорные точки курсором на экране).

Кривая Безье (определение 1)

- Кривая Безье — параметрическая кривая, задаваемая выражением

$$\mathbf{B}(t) = \sum_{i=0}^{n} \mathbf{P}_i \mathrm{b}_{i,n}(t), \quad 0 < t < 1$$

где $\boldsymbol{P_i}$ — функция компонент векторов опорных вершин, а $\boldsymbol{b_{i,n}(t)}$— базисные функции кривой Безье, называемые также полиномами Бернштейна

$$\mathrm{b}_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i} \qquad \binom{n}{i} = \frac{n!}{i!(n-i)!}$$

где *n* — степень полинома, *i* — порядковый номер опорной вершины

Кривая Безье (определение 2)

- Кривая Безье – кривая строящаяся методом де Касталье. Данный метод основан на разбиении отрезков, соединяющих исходные точки в отношении t (значение параметра), а затем в рекурсивном повторении этого процесса для полученных отрезков:

$$P_i^j(t) = (1-t)P_i^{j-1}(t) + tP_{i+1}^{j-1}(t). \quad 0 < t < 1$$

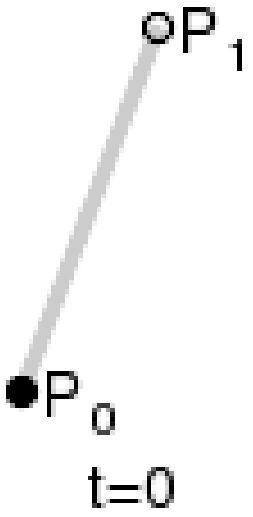
Нижний индекс - номер точки, верхний индекс - уровень разбиения. Уравнение кривой n-ого порядка задается

$$P(t) := P_0^n(t)$$

Линейные кривые

- При ***n*** = 1 кривая представляет собой отрезок прямой линии, опорные точки $\mathbf{P_0}$ и $\mathbf{P_1}$ определяют его начало и конец. Кривая задаётся уравнением:

$$\mathbf{B}(t) = (1-t)\mathbf{P}_0 + t\mathbf{P}_1 \quad t \in [0,1]$$



Параметр *t* в функции, описывающей линейный случай кривой Безье, определяет где именно на расстоянии от $\mathbf{P_0}$ до $\mathbf{P_1}$ находится **B**(*t*). Например, при *t = 0,25* значение функции **B**(*t*) соответствует четверти расстояния между точками $\mathbf{P}_0$ и $\mathbf{P}_1$. Параметр *t* изменяется от 0 до 1, а **B**(*t*) описывает отрезок прямой между точками $\mathbf{P}_0$ и $\mathbf{P}_1$.

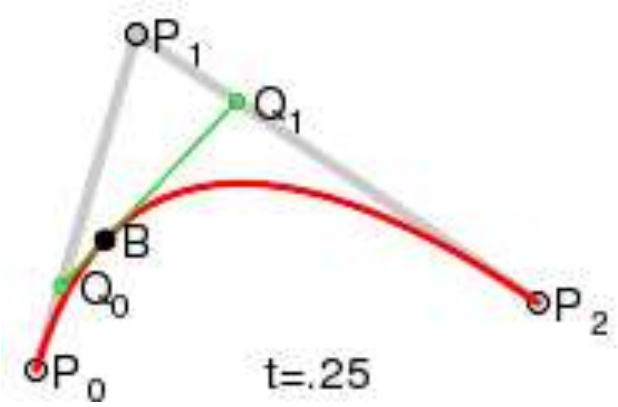
Квадратные кривые

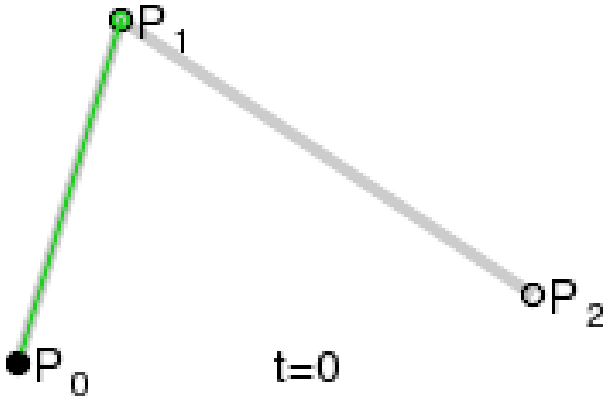
- Квадратная кривая Безье (*n* = 2) задаётся 3-мя опорными точками: $\mathbf{P}_0$, $\mathbf{P}_1$ и $\mathbf{P}_2$.

$$\mathbf{B}(t) = (1-t)^2\mathbf{P}_0 + 2t(1-t)\mathbf{P}_1 + t^2\mathbf{P}_2, \quad t \in [0,1]$$

Для построения квадратных кривых Безье требуется выделение двух промежуточных точек $\mathbf{Q}_0$ и $\mathbf{Q}_1$ из условия чтобы параметр *t* изменялся от 0 до 1:

- Точка $\mathbf{Q}_0$ изменяется от $\mathbf{P}_0$ до $\mathbf{P}_1$ и описывает линейную кривую Безье.
- Точка $\mathbf{Q}_1$ изменяется от $\mathbf{P}_1$ до $\mathbf{P}_2$ и также описывает линейную кривую Безье.
- Точка $\mathbf{B}_0$ изменяется от $\mathbf{Q}_0$ до $\mathbf{Q}_1$ и описывает квадратичную кривую Безье.





Метод де Касталье для кривых второго порядка

Обозначим опорные точки как $P_i, i \in \overline{0,2}$,

начало кривой положим в точке $P_0(t=0)$

а конец - в точке $P_2(t=1)$;

для каждого $t \in [0,1]$ найдем точку $P_0^2(t)$:

$$
\begin{aligned}
&P_0^1(t) = (1-t)P_0 + tP_1, \\
&P_1^1(t) = (1-t)P_1 + tP_2, \\
&P_0^2(t) = (1-t)P_0^1(t) + tP_1^1(t) = \\
&= (1-t)^2 P_0(t) + 2t(1-t)P_1(t) + t^2 P_2(t),
\end{aligned}
$$

таким образом, получим кривую второго порядка.

Кубические кривые

- $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$ и $\mathbf{P}_3$, заданные в 2-х или 3-мерном пространстве определяют форму кривой.

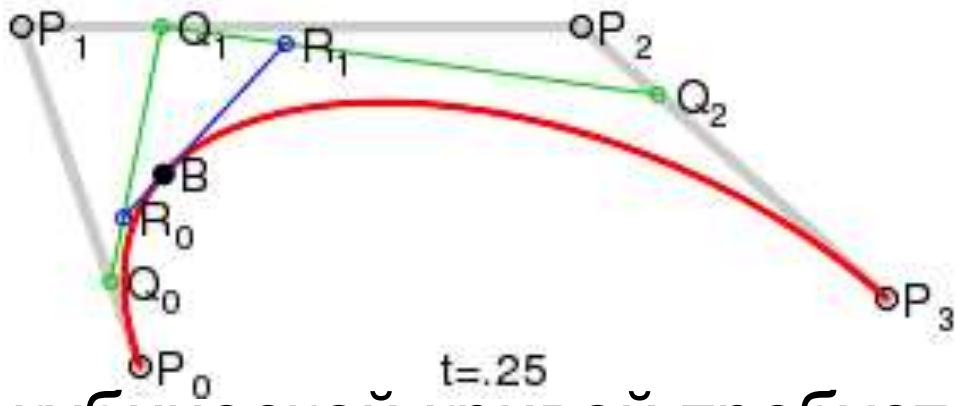
$$\mathbf{B}(t) = (1-t)^3\mathbf{P}_0 + 3t(1-t)^2\mathbf{P}_1 + 3t^2(1-t)\mathbf{P}_2 + t^3\mathbf{P}_3,$$

В матричной форме кубическая кривая Безье записывается следующим образом:

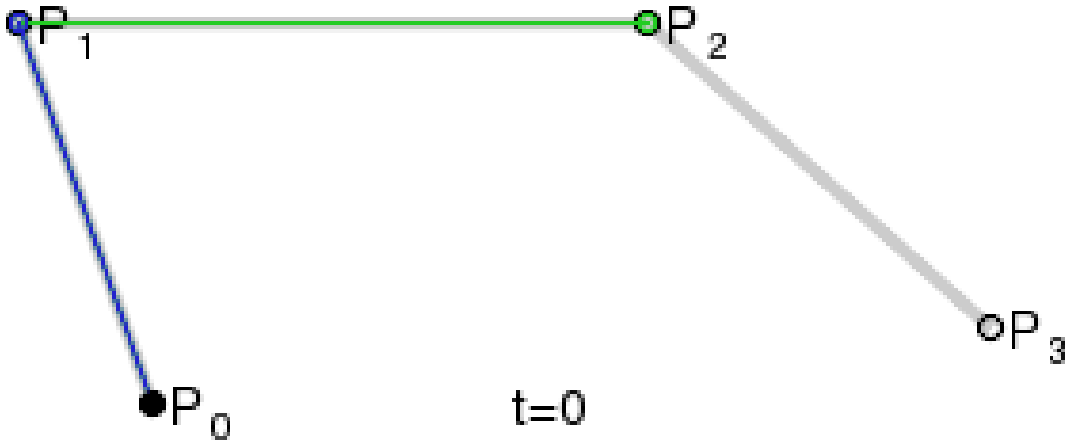
$$\mathbf{B}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \mathrm{M}_B \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} \qquad \mathrm{M}_B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Где $\mathbf{M_B}$ называется базисной матрицей Безье

Построение кубической кривой

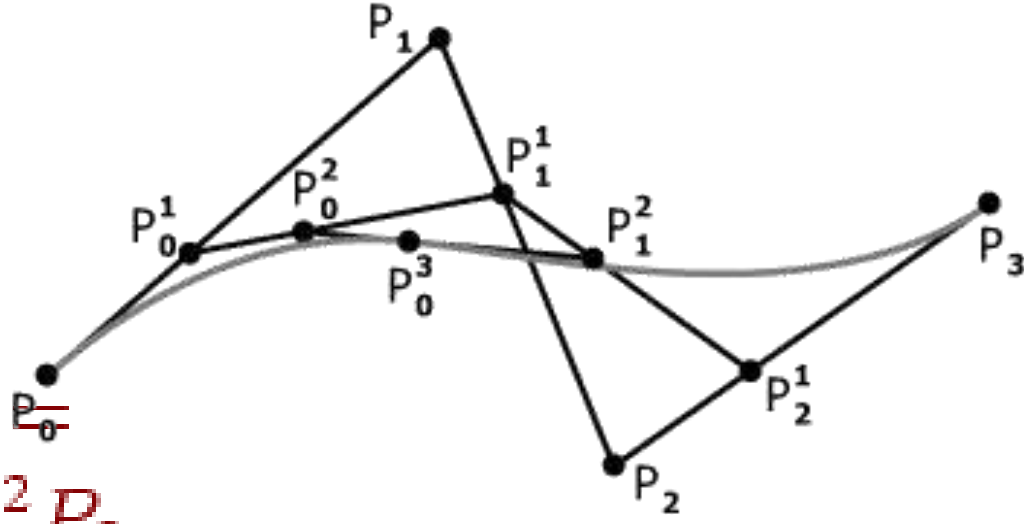


Для построения кубической кривой требуется больше промежуточных точек. Для кубической кривой это промежуточные точки $\mathbf{Q}_0$, $\mathbf{Q}_1$ и $\mathbf{Q}_2$, описывающие линейные кривые, а также точки $\mathbf{R}_0$ и $\mathbf{R}_1$, которые описывают квадратные кривые



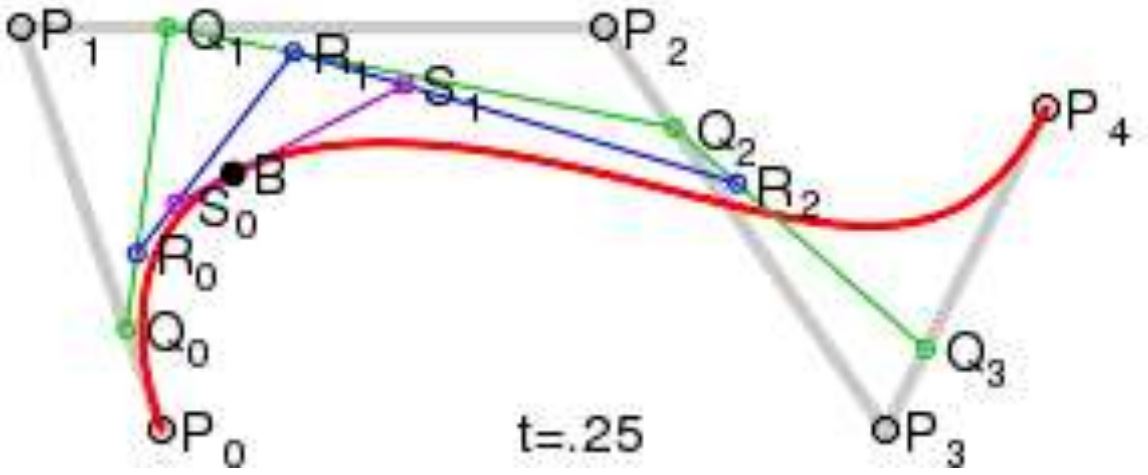
Метод де Касталье для кубических кривых

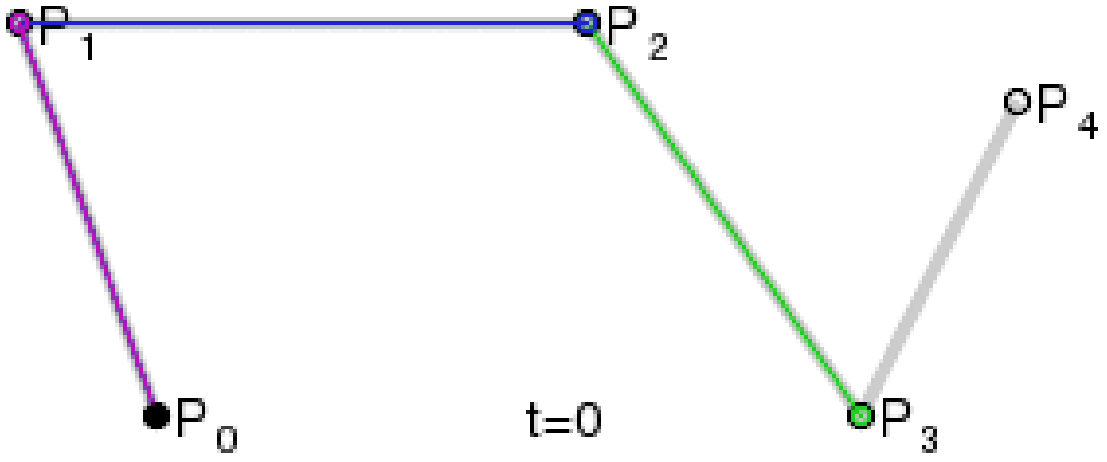
$$P_0^1(t) = (1-t)P_0 + tP_1,$$
$$P_1^1(t) = (1-t)P_1 + tP_2,$$
$$P_2^1(t) = (1-t)P_2 + tP_3,$$



$$P_0^2(t) = (1-t)P_0^1(t) + tP_1^1(t) =$$
$$= (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2,$$
$$P_1^2(t) = (1-t)P_1^1(t) + tP_2^1(t) =$$
$$= (1-t)^2 P_1 + 2t(1-t)P_2 + t^2 P_3,$$
$$P_0^3(t) = (1-t)P_0^2(t) + tP_1^2(t) =$$
$$= (1-t)^2 P_0^1(t) + 2t(1-t)P_1^1(t) + t^2 P_2^1(t) =$$
$$= (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3.$$

Построение кривой четвертой степени





Программная реализация

```
while t<1 do
  begin
    b.X := Round((1-t)*(1-t)*(1-t)*P[0].X +
          3*t*(1-t)*(1-t)*P[1].X +
          3*t*t*(1-t)*p[2].X + t*t*t*P[3].X);
    b.Y := Round((1-t)*(1-t)*(1-t)*P[0].Y +
          3*t*(1-t)*(1-t)*P[1].Y +
          3*t*t*(1-t)*p[2].Y +
          t*t*t*P[3].Y);
    Form1.Canvas.Pixels[B.x,B.y] := clRed;
    t:=t + dt;
  end;
```

Свойства кривых Безье

- кривая всегда располагается внутри фигуры, образованной линиями, соединяющими контрольные точки;
- прямая линия образуется при коллинеарном (на одной прямой) размещении управляющих точек;
- кривая Безье симметрична, то есть обмен местами между начальной и конечной точками (изменение направления траектории) не влияет на форму кривой;
- масштабирование и изменение пропорций кривой Безье не нарушает ее стабильности;
- любой частичный отрезок кривой Безье также является кривой Безье;
- окружность не может быть описана параметрическим уравнением кривой Безье.

Сплайн Безье

- Для построения сложных по форме линий отдельные кривые Безье могут быть последовательно соединены друг с другом в ***сплайн* Безье***.
- Для того, чтобы обеспечить гладкость линии в месте соединения двух кривых, три смежные опорные точки обеих кривых должны лежать на одной прямой.

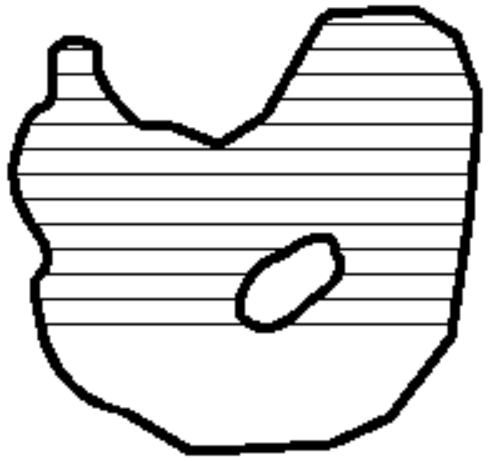
*Сплайн (гибкое лекало, полоса металла, используемая для черчения кривых линий) — функция, область определения которой разбита на конечное число отрезков, на каждом из которых сплайн совпадает с некоторым полиномом.

Закраска области, заданной цветом границы

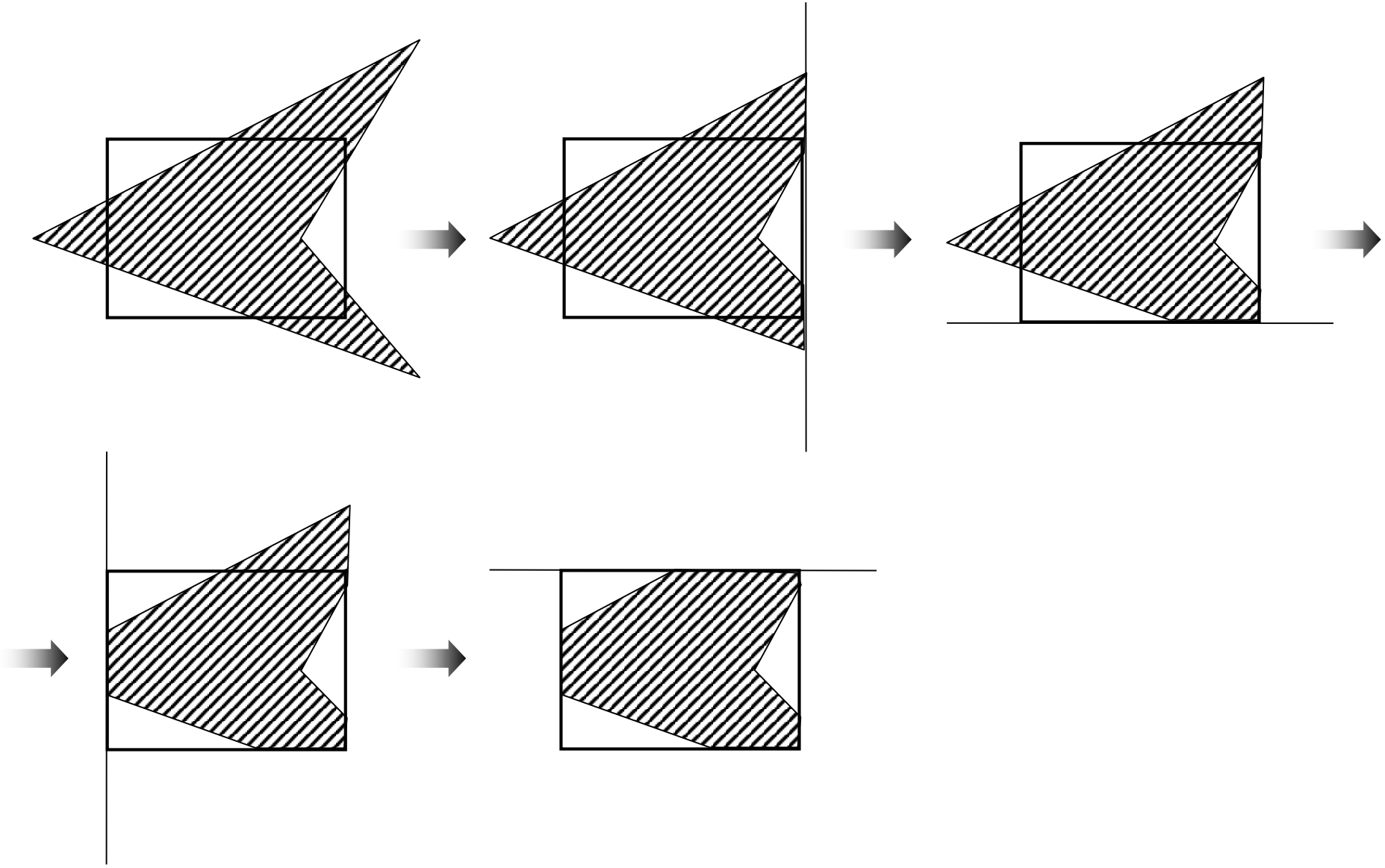
```
void PixelFill(int x, int y, int border_color, int color)
{int c = getpixel(x, y);
if ((c != border_color) && (c != color))
{putpixel(x, y, color);
PixelFill(x – 1, y, border_color, color);
PixelFill(x + 1, y, border_color, color);
PixelFill(x, y – 1, border_color, color);
PixelFill(x, y + 1, border_color, color);}}
```

Закраска области с использованием сканирующих строк

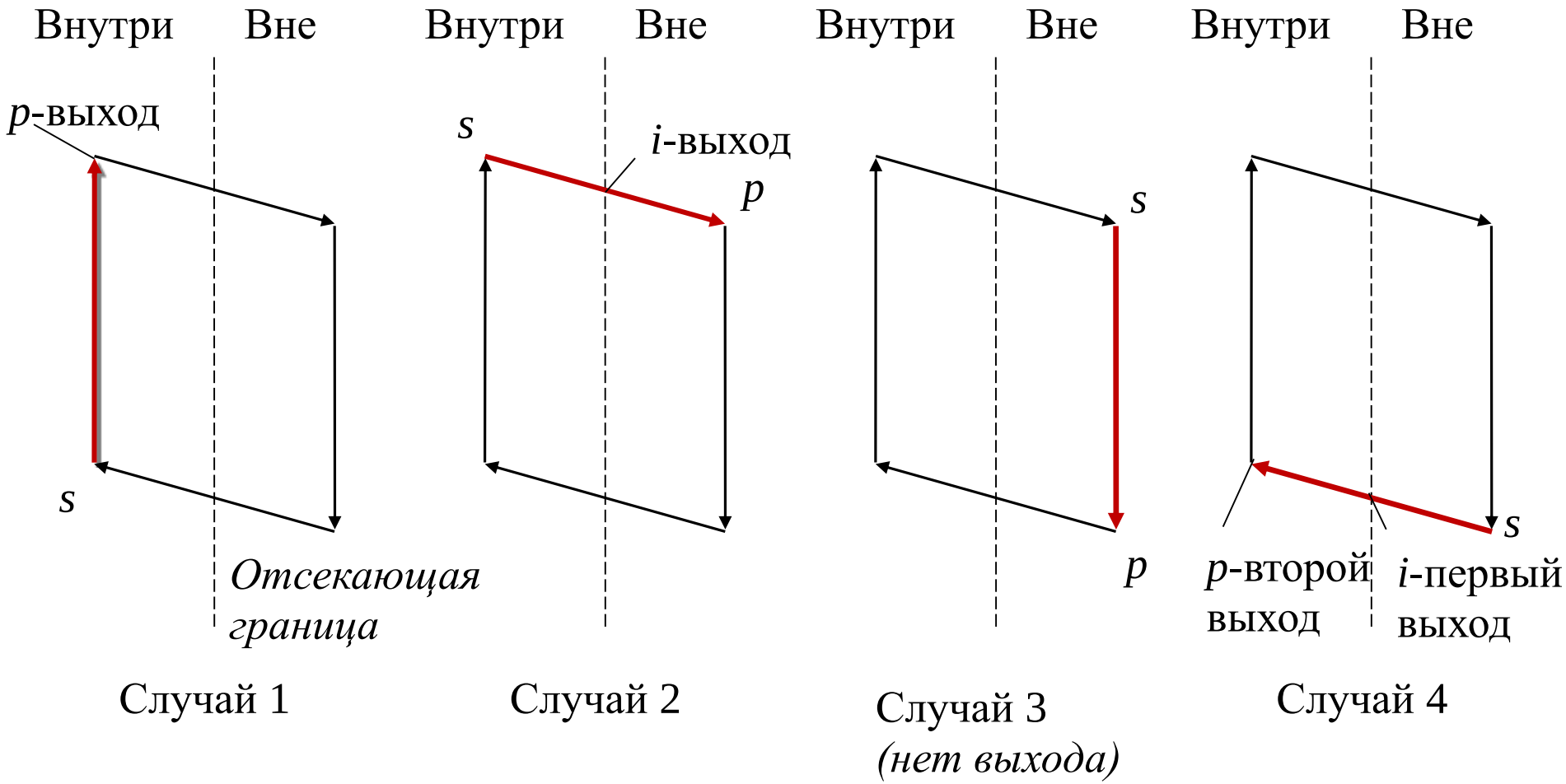
1. Поместим затравочную точку в стек.
2. Извлекаем координаты точки с вершины стека в переменные *(x,y)*
3. Заполняем максимально возможный интервал, в котором находится точка, вправо и влево вплоть до достижения граничных точек.
4. Запоминаем крайнюю левую x_l и крайнюю правую x_r абсциссы заполненного интервала.
5. В соседних строках над и под интервалом (x_l,x_r) находим незаполненные к настоящему моменту внутренние точки области, которые объединены в интервалы, а в правый конец каждого такого интервала помещаем на стек.
6. Если стек не пуст, то переходим к пункту 3.



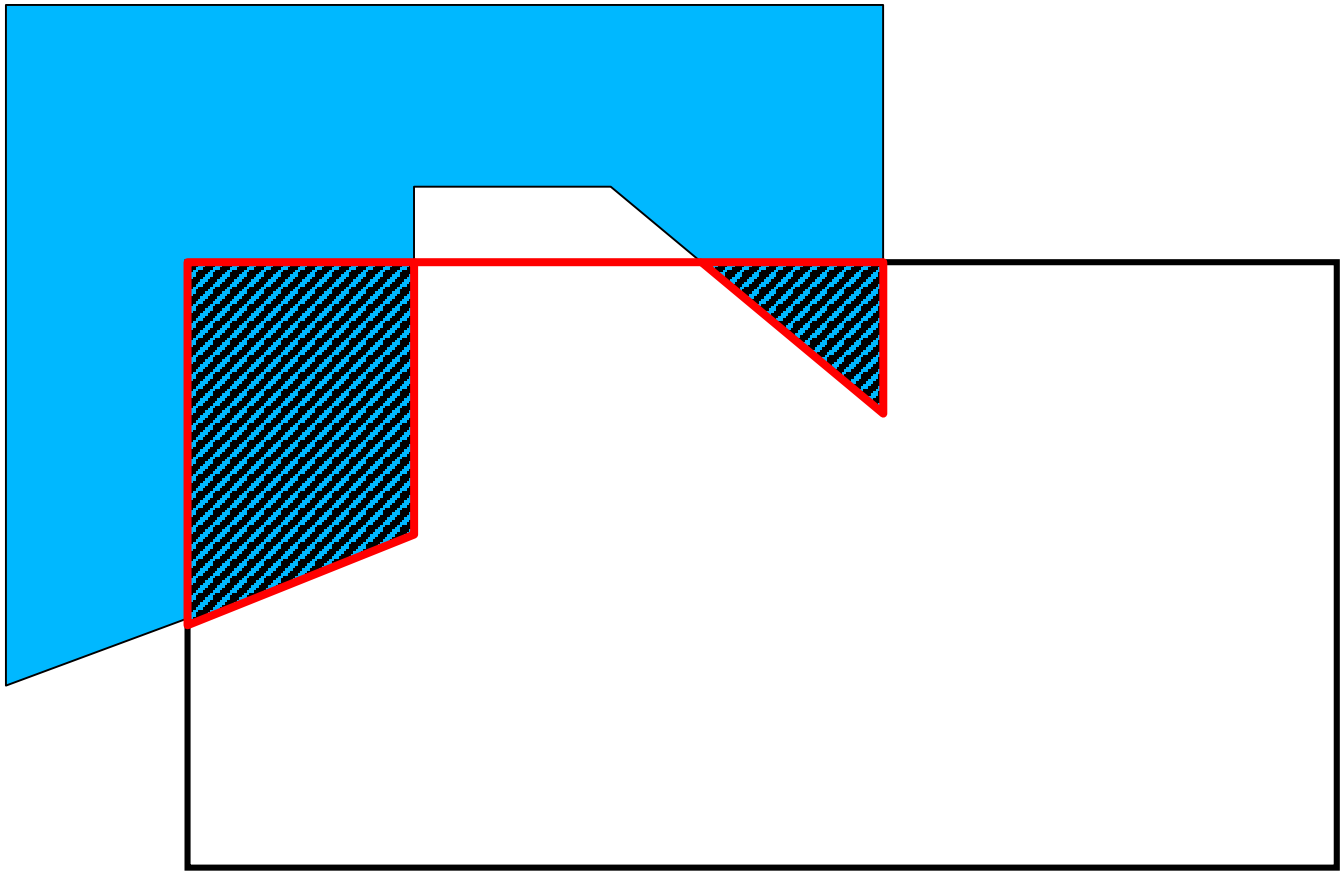
Отсечение многоугольников (алгоритм Сазерленда-Ходгмана



Отсечение многоугольников (алгоритм Сазерленда-Ходгмана

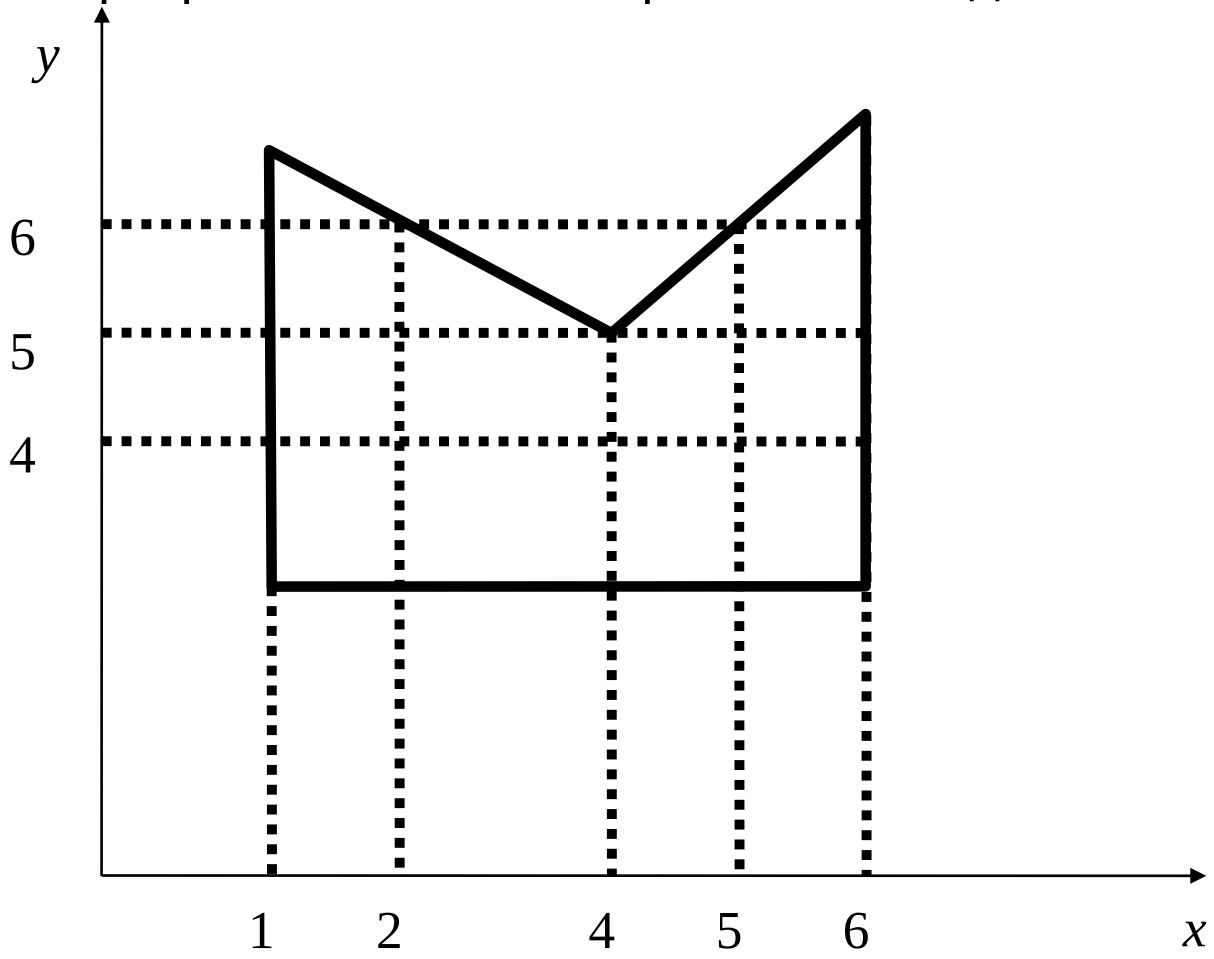


Проблема отсечения



Заполнение многоугольников

Алгоритмы построчного заполнения, основаны на том, что соседние пикселы в строке скорее всего одинаковы и меняются только там где строка пересекается с ребром. Это называется ***когерентностью растровых строк***. При этом достаточно определить X-координаты пересечений строк сканирования с ребрами. Пары отсортированных точек пересечения задают интервалы заливки.



Оптимизация алгоритма заполнения многоугольника

1) Если какие-либо ребра пересекались *i*-й строкой, то они скорее всего будут пересекаться также и строкой *i+1*. Это называется ***когерентностью ребер***. При переходе к новой строке легко вычислить новую *X*-координату точки пересечения ребра, используя *X*-координату старой точки пересечения и тангенс угла наклона ребра:

$$X_{i+1} = X_i + 1/k$$

(тангенс угла наклона ребра - $k = dy/dx$, так как $dy = 1$, то $1/k = dx$)

2) Для ускорения работы алгоритма используется ***список активных ребер*** (САР). Этот список содержит те ребра многоугольника, которые пересекают сканирующую строку. При пересечении очередной сканирующей строки вершины многоугольника, из САР удаляются ребра, которые находятся выше, и добавляются концы, которые пересекает сканирующая строка. При работе алгоритма находятся пересечения сканирующей строки только с ребрами из САР.

Методы устранения ступенчатости

Метод увеличения частоты выборки

+	+
+	+

Увеличение разрешения в два раза

+	+	+	+
+	+	+	+
+	+	+	+
+	+	+	+

Увеличение разрешения в четыре раза

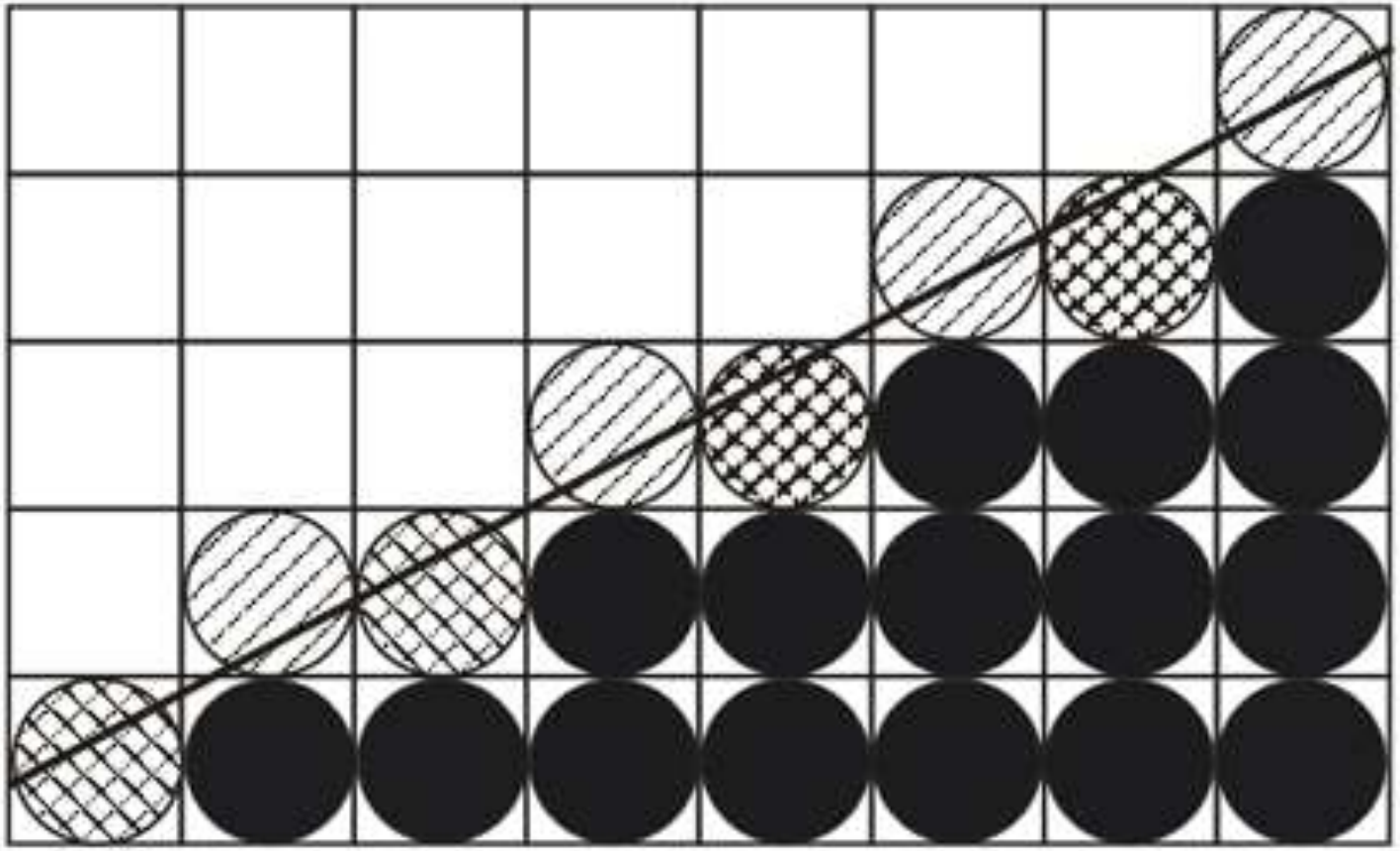
В некоторой степени можно получить лучшие результаты, если рассматривать больше подпикселов и учитывать их влияние с помощью весов при определении атрибутов.

1	2	1
2	4	2
1	2	1

1	2	3	4	3	2	1
2	4	6	8	6	4	2
3	6	9	12	9	6	3
4	8	12	16	12	8	4
3	6	9	12	9	6	3
2	4	6	8	6	4	2
1	2	3	4	3	2	1

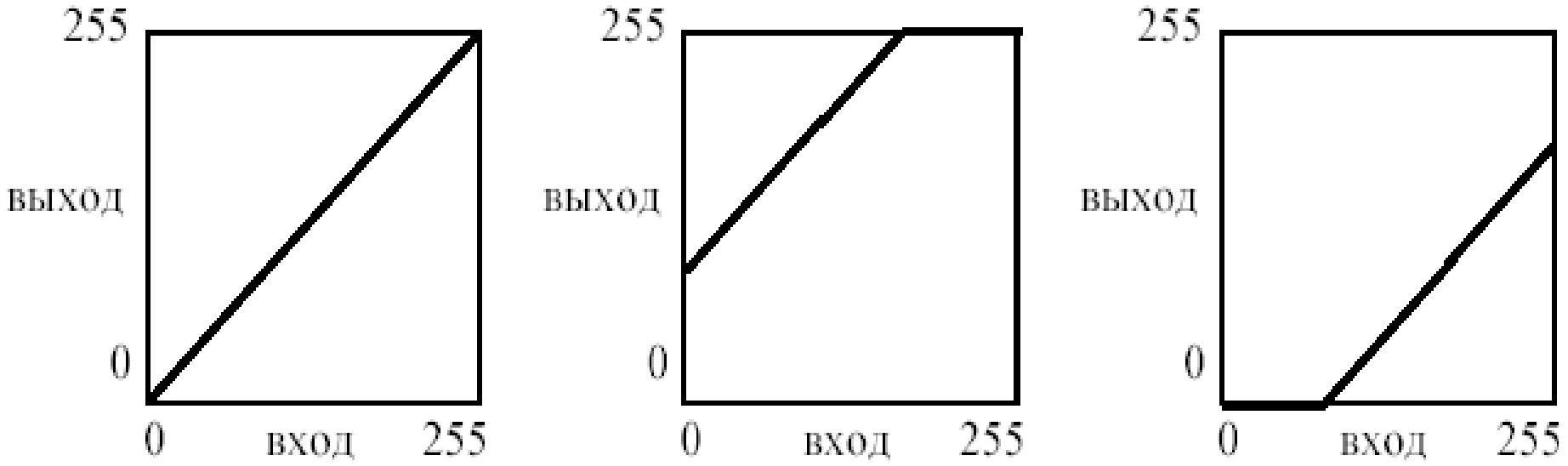
Методы устранения ступенчатости

Метод, основанный на использовании полутонов



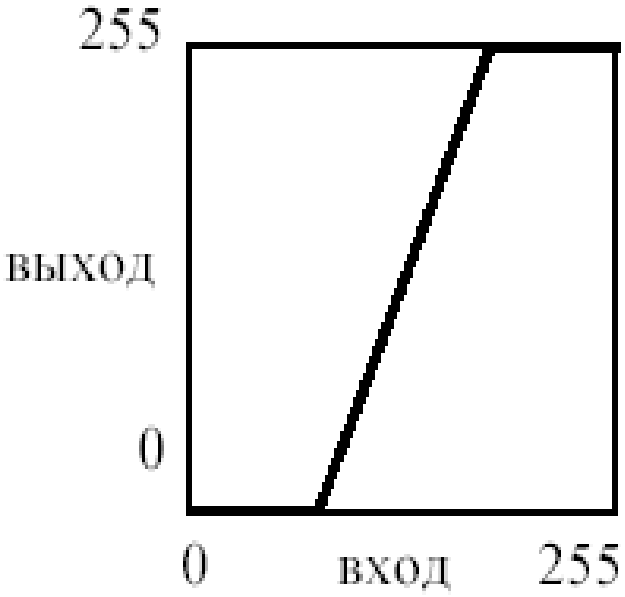
Яркость

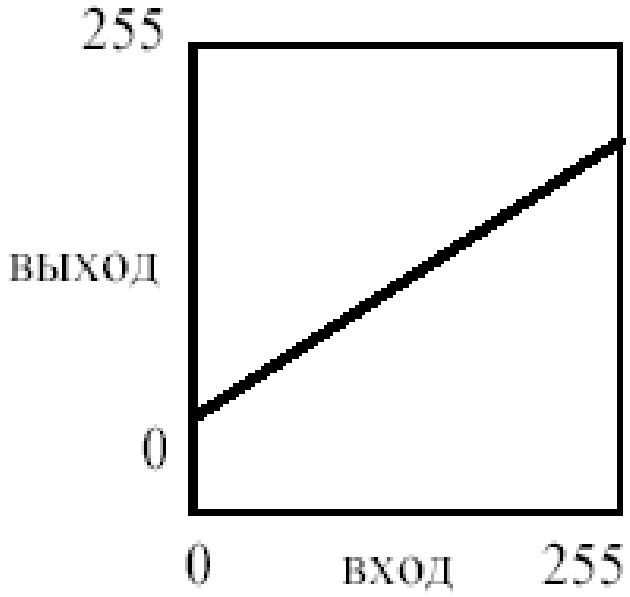
- Яркость представляет собой характеристику, определяющую то, на сколько сильно цвета пикселей отличаются от чёрного цвета.



Контраст

- Контраст представляет собой характеристику того, насколько большой разброс имеют цвета пикселей изображения.





Гистограмма

- **Гистограмма** (в фотографии) — это график распределения полутонов изображения, в котором по горизонтальной оси представлена Яркость, а по вертикали — относительное число пикселов с данным значением яркости.