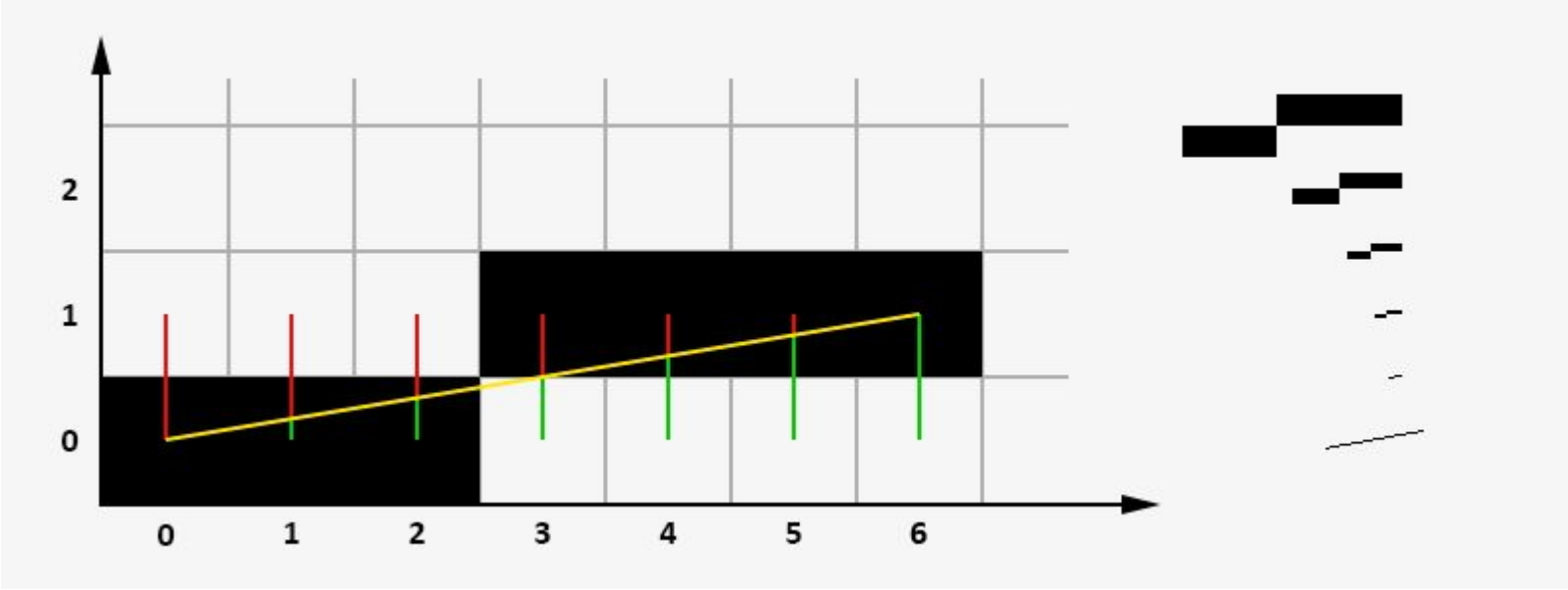
# Операции с изображением на уровне растра

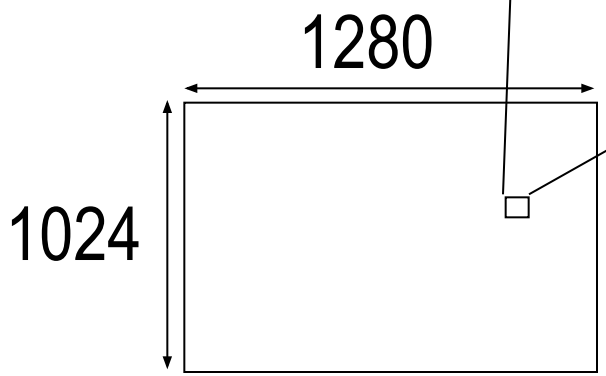


# Содержание

- Растровая развертка
- Принцип работы алгоритма Брезенхема
- Алгоритм Сяоляня
- Брезенхам для окружности
- Растровая развертка эллипса

# Видеоадаптер и буфер кадра

Буфер кадра – прямоугольный массив структур <Red,Green,Blue>



# Растровая и векторная графика. Понятие растра

**Для представления графической информации на двумерной плоскости (например, экране монитора, странице книги и т.п.) в вычислительной технике применяются два основных подхода: растровый и векторный**

**При векторном подходе графическая *информация* описывается как совокупность неких абстрактных геометрических объектов, таких как прямые, отрезки, кривые, прямоугольники и т.п.**

**Растровая *графика* же оперирует изображениями в виде растров. Неформально можно сказать, что растр - это описание изображения на плоскости путем разбиения всей плоскости или ее части на одинаковые квадраты и присвоение каждому квадрату своего цветового (или иного, например, прозрачности, для последующего наложения изображений друг на друга) атрибута.**

**Если таких квадратов имеется конечное число, то получается, что непрерывная цветовая *функция* изображения приближенно представлена конечной совокупностью значений атрибутов.**

# Модель растра

Растр можно рассматривать как кусочно-постоянную аппроксимацию изображения, заданного как цветовая *функция* на плоскости.
Формально, введем следующие определения:
**Растр** (англ. *raster*) - *отображение* вида

$$f : X \times Y \to 2^{\mathbb{R}^2} \times C,$$

где $Y, X \subset \mathbb{Z}$

$2^{\mathbb{R}^2}$ обозначает множество всех подмножеств , $\mathbb{R}^2$

C - множество значений атрибутов (как правило, цвет).
f(i, j) - элемент растра, называемый **пикселем** (англ. *pixel* (от *picture element*)),
f(i, j) = (A(i, j),C(i, j)), где A - область пикселя, C - атрибут пикселя (как правило, цвет).
Чаще всего мы будем пользоваться следующими двумя видами атрибутов:

C(i, j) = I(i, j) - интенсивность (или яркость) пикселя;

C(i, j) = {R(i, j),G(i, j),B(i, j)} - цветовые атрибуты в цветовой модели RGB

# Модели точка и квадрат

Aij может определяться двояко, в зависимости от того, с какой моделью мы хотим работать:

Aij := (i, j) - одна точка. $Aij := (i, i+1) \times (j, j+1)$ - квадрат.

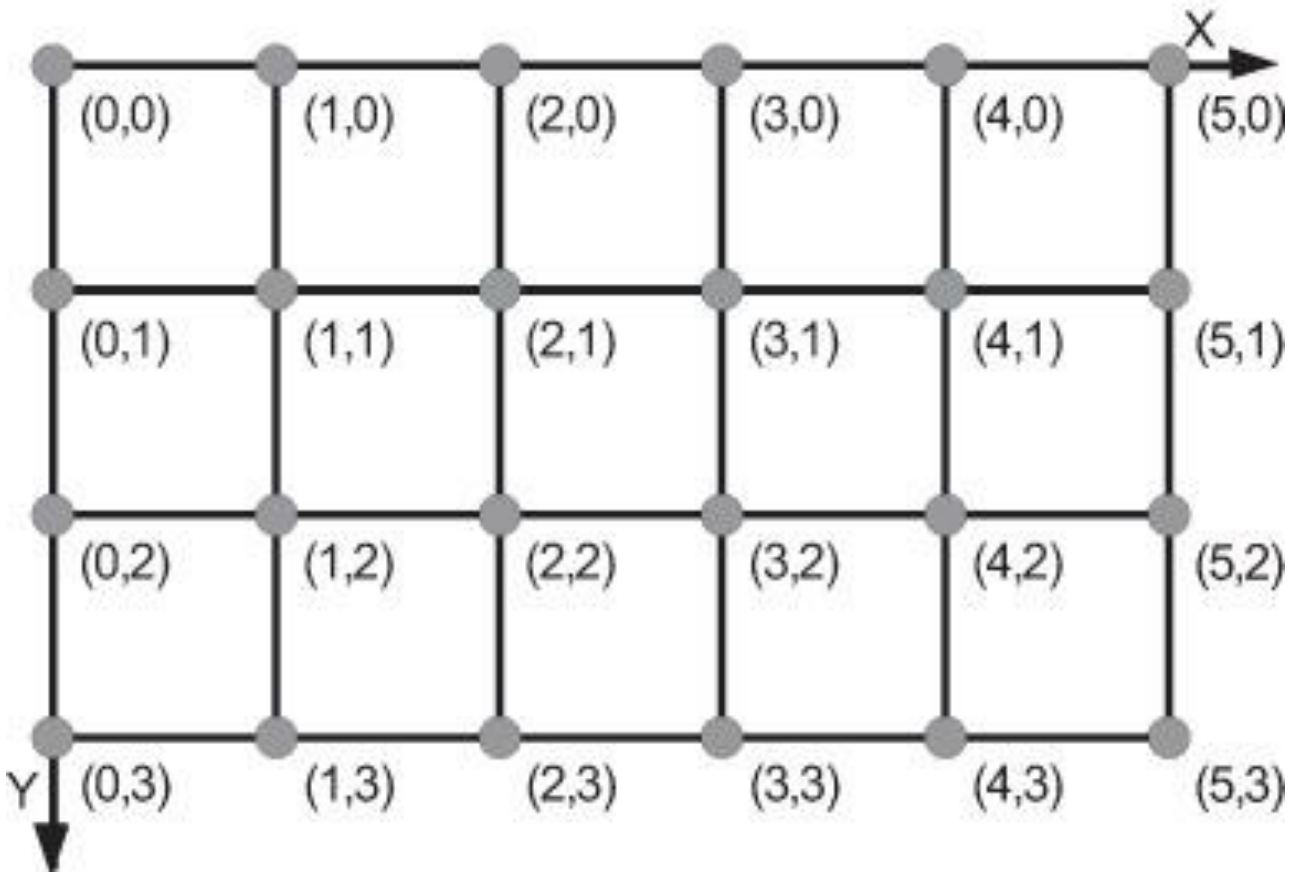
На реальных графических устройствах физически пиксели могут быть прямоугольниками, что иногда порождает дополнительные трудности.

В реальности, как правило, X и Y - ограниченные наборы неотрицательных целых чисел; такой растр называется **прямоугольным**.

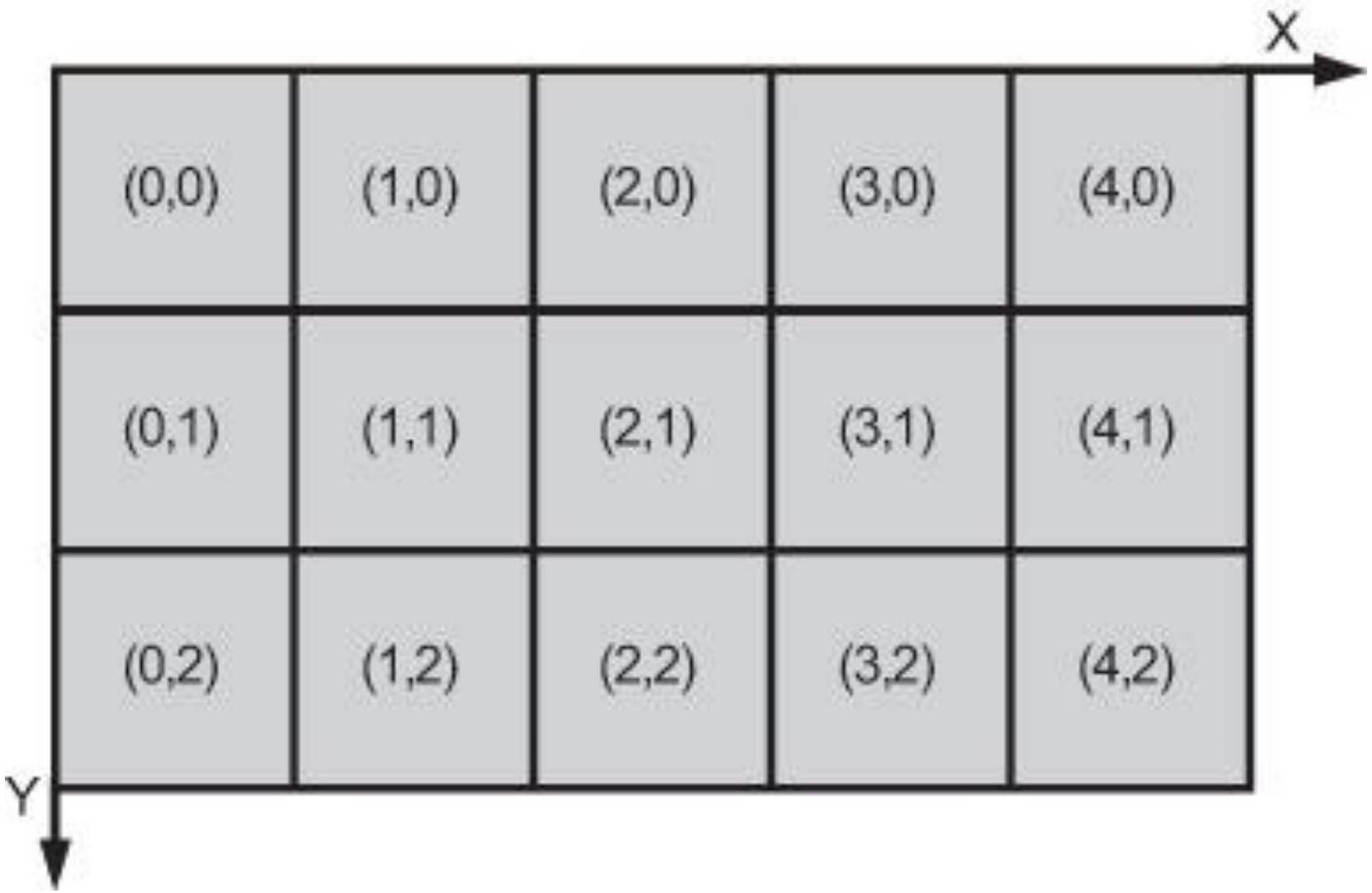
Для него применимо понятие **Аспектовое отношение** (англ. *aspect ratio*) - *отношение* ширины к высоте растра (|X|/|Y|).

Чаще всего такое понятие употребляется в связи с физическими растрами (дисплеями, ПЗС-матрицами фотоаппаратов и т.д.) и записывается в виде простой дроби с ":", например "4:3".

# Модель растра первого типа.



# Модель растра 2 типа



# Растровая развертка

Экран растрового дисплея можно рассматривать как матрицу дискретных элементов, или пикселей.

Процесс определения пикселей, наилучшим образом аппроксимирующих некоторую геометрическую фигуру, называется разложением в растр, или построением растрового образа фигуры.

Построчная *визуализация* растрового образа называется **растровой разверткой** данной фигуры.

# Алгоритм Брезенхема растровой дискретизации отрезка

При построении растрового образа отрезка необходимо, прежде всего, установить критерии "хорошей" аппроксимации.

Первое требование состоит в том, что *отрезок* должен начинаться и кончаться в заданных точках и при этом выглядеть сплошным и прямым (при достаточно высоком разрешении дисплея этого можно добиться).

Кроме того, яркость вдоль отрезка должна быть одинаковой и не зависеть от наклона отрезка и его длины.

*Алгоритм* должен работать быстро. Для этого необходимо по возможности исключить *операции* с вещественными числами.

С целью ускорения работы алгоритма можно также реализовать его на аппаратном уровне.

В большинстве алгоритмов используется пошаговый метод изображения, т.е. для нахождения координат очередной точки растрового образа наращивается *значение* одной из координат на единицу растра и вычисляется *приращение* другой *координаты*.

# Цепочный код

Для представления на экране растрового дисплея любой кривой единичной толщины необходимо найти координаты близких к линии смежных точек на целочисленной сетке .

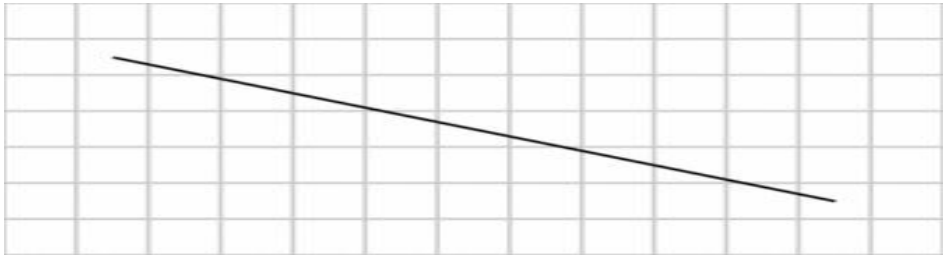
В дальнейшем будем считать, что на плоскости имеется квадратная сетка с шагом 1, причем узлы целочисленной решетки являются центрами соответствующих квадратных ячеек сетки.

Другими словами, узлы растра окружены «единичными» квадратными окрестностями «радиуса» 1/2.

Инициализации точки растра с координатами (*i, j) соответствует закраска ка*ким-либо цветом ее квадратной окрестности

Всякая точка на плоскости имеет четырех непосредственных соседей и восемь косвенных соседей.

Если точки являются непосредственными соседями, то их квадратные окрестности имеют общую сторону. Квадратные окрестности косвенных соседей имеют общую сторону или общую вершину.



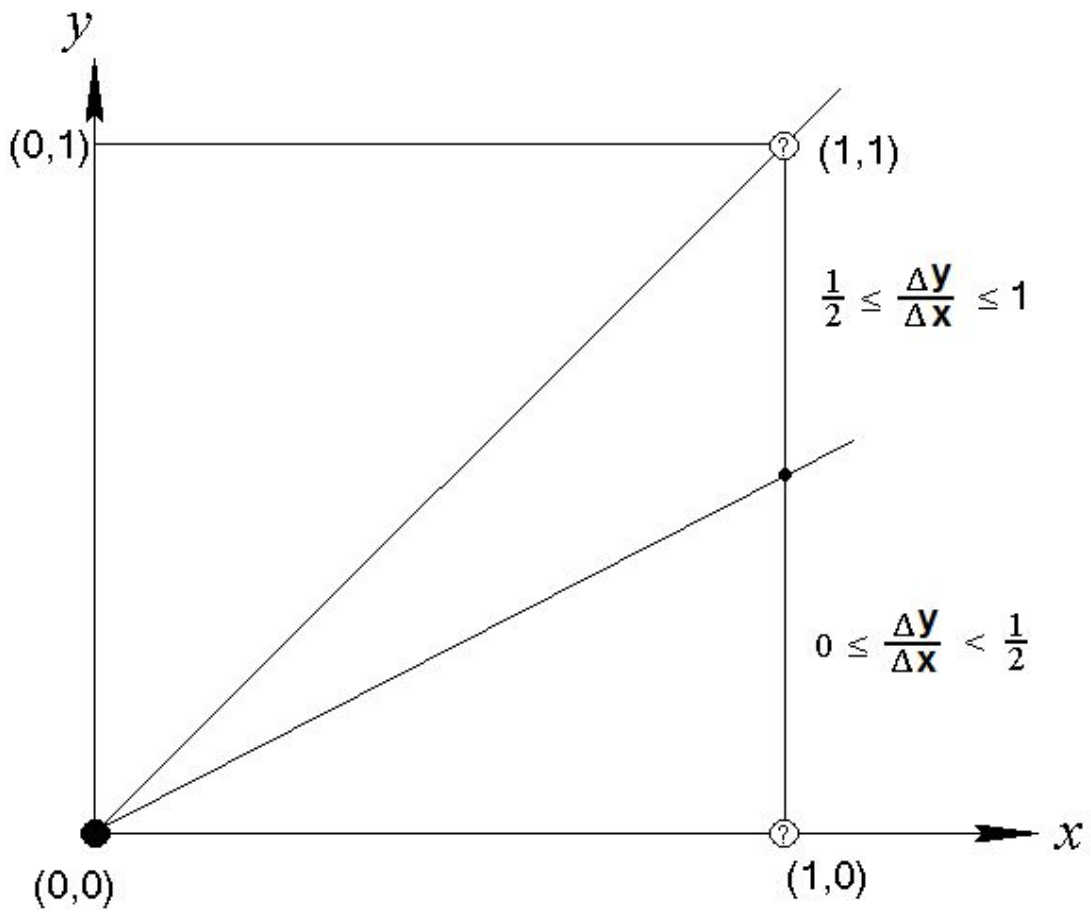
# Принцип работы алгоритма Брезенхема

Берётся отрезок и его начальная координата *x*.

К иксу в цикле прибавляем по единичке в сторону конца отрезка.

На каждом шаге вычисляется ошибка — расстояние между реальной координатой *y* в этом месте и ближайшей ячейкой сетки.

Если ошибка не превышает половину высоты ячейки, то она заполняется.

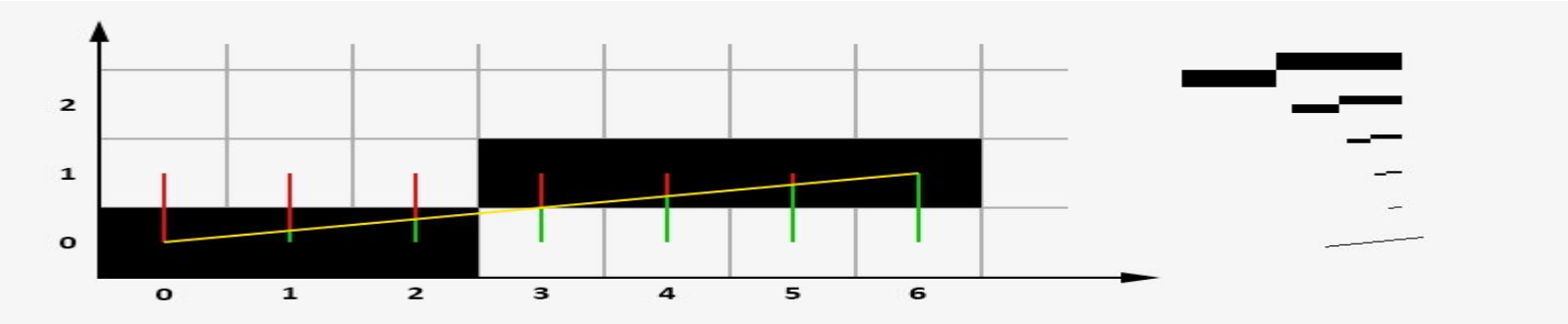


# Простое объяснение

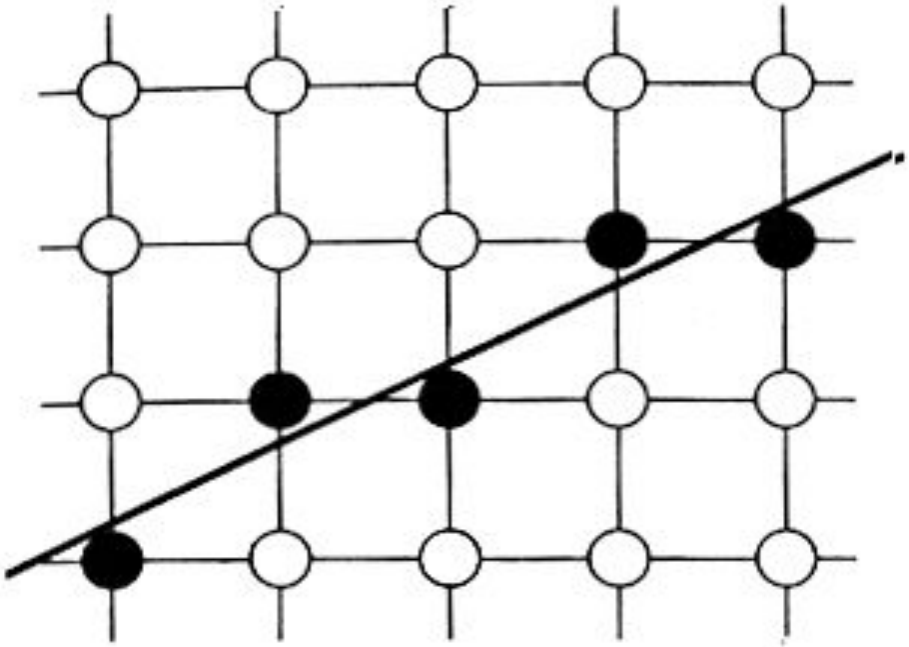
Сначала вычисляется угловой коэффициент *(y1 — y0)/(x1 — x0)*. Значение ошибки в начальной точке отрезка *(0,0)* принимается равным нулю и первая ячейка заполняется.

На следующем шаге к ошибке прибавляется угловой коэффициент и анализируется её значение, если ошибка меньше *0.5*, то заполняется ячейка *(x0+1, y0)*, если больше, то заполняется ячейка *(x0+1, y0+1)* и из значения ошибки вычитается единица.

На картинке ниже жёлтым цветом показана линия до растеризации, зелёным и красным — расстояние до ближайших ячеек. Угловой коэффициент равняется одной шестой, на первом шаге ошибка становится равной угловому коэффициенту, что меньше *0.5*, а значит ордината остаётся прежней..



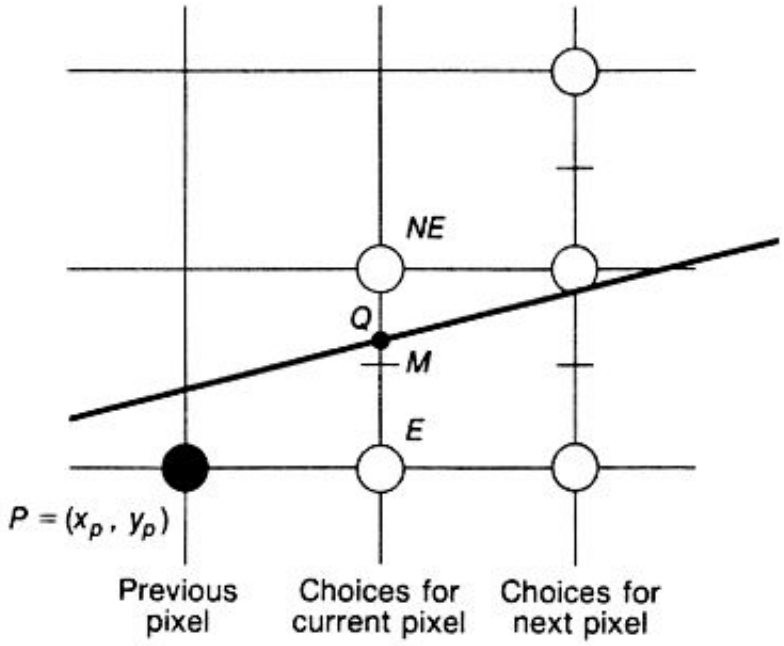
# Алгоритм Брезенхама (1/4)



Отрезок, соединяющий
P(x1, y1) и Q(x2, y2)

$$y = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1), x \in [x_1, x_2]$$

# Алгоритм Брезенхама (2/4)




$$F(x,y) = (x - x_1)dy - (y - y_1)dx$$

$$dx = x_2 - x_1$$

$$dy = y_2 - y_1$$

$F(x,y) = 0$ -- точка на отрезке

$F(x,y) < 0$ -- точка выше

$F(x,y) > 0$ -- точка ниже

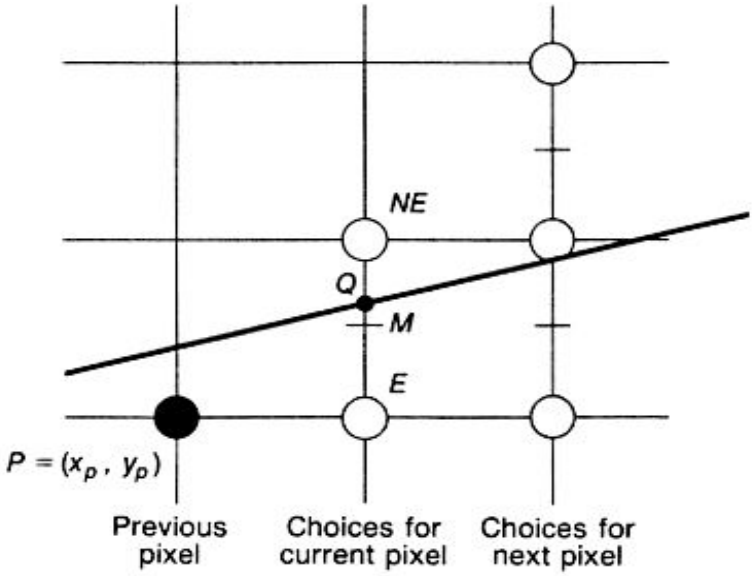
Точка P определена, тогда координаты срединной точки

$$(x_p + 1, y_p + 1/2)$$

и значение функции в этой точке

$$d = F(x_p + 1, y_p + 1/2)$$

# Алгоритм Брезенхама (3/4)




Если $d < 0$, то выбирается $E$ и

$$d_{new} = F(x_p + 2, y_p + \tfrac{1}{2})$$

$$d_{new} - d_{old} = F(x_p + 2, y_p + \frac{1}{2}) - F(x_p + 1, y_p + \frac{1}{2})$$

$$\Delta_E = d_{new} - d_{old} = dy = y2 - y1$$

Если $d \geq 0$, то выбирается $NE$

$$d_{new} = F(x_p + 2, y_p + \frac{3}{2})$$

$$\Delta_{NE} = dy - dx = (y_2 - y_1) - (x_2 - x_1)$$

В начальной точке

$$d_{start} = F(x_1 + 1, y_1 + 1/2) =$$

$$(x_1 + 1 - x_1)dy - (y_1 + 1/2 - y_1)dx =$$

$$= dy - dx/2$$

# Алгоритм Брезенхама (4/4)

Одна неприятность -- деление на 2

Чтобы избежать вещественной арифметики, сделаем преобразование =>

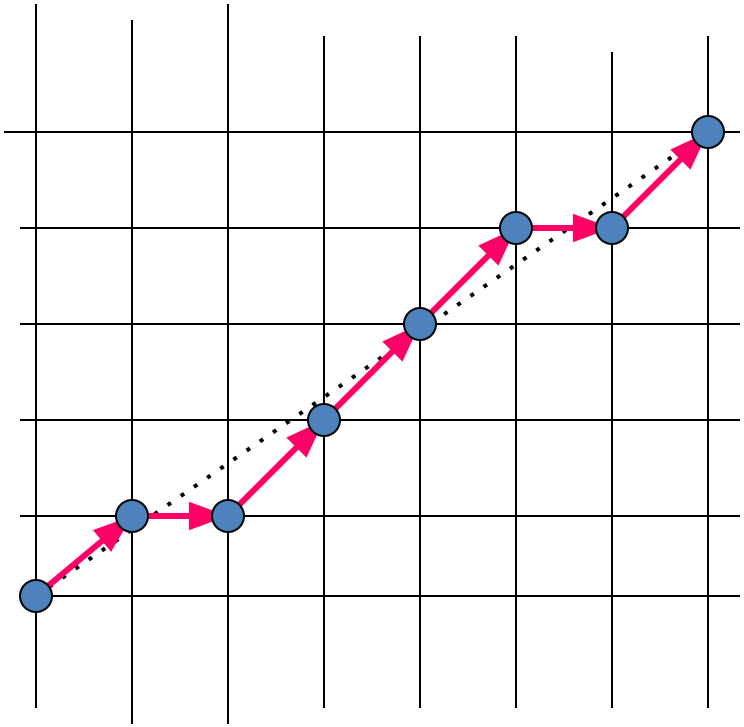
$$F'(x, y) = 2F(x, y)$$

$$d' = 2d$$

$$d_{start} = 2dy - dx$$

$$\Delta d' = 2\Delta d$$

$$d_{start} = 2dy - dx = 3;\ d_{NE} = -4;\ d_E = 10$$



$d_0 = 10 - 7 = 3 > 0$ (NE)

$d_1 = 3 - 4 = -1 < 0$ (E)

$d_2 = -1 + 10 = 9$ (NE)

$d_3 = 9 - 4 = 5$ (NE)

$d_4 = 5 - 4 = 1$ (NE)

$d_5 = 1 - 4 = -3$ (E)

$d_6 = -3 + 10 = 7$ (NE)

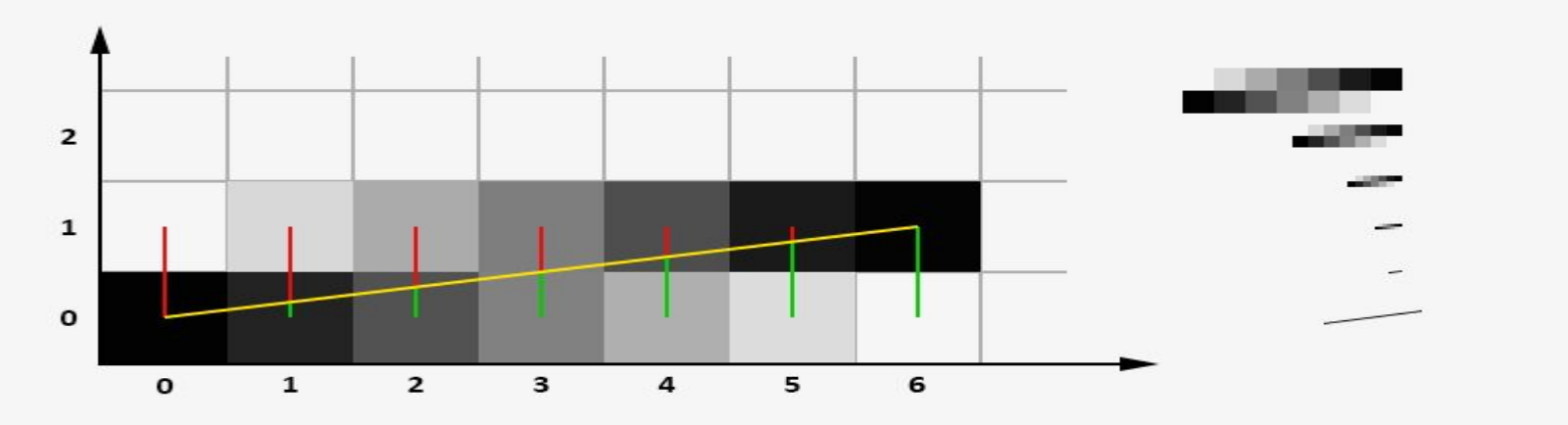
# Алгоритм У Сяолиня

Для рисования сглаженных линий.

Он отличается тем, что на каждом шаге ведётся расчёт для двух ближайших к прямой пикселей, и они закрашиваются с разной интенсивностью, в зависимости от удаленности.

Точное пересечение середины пикселя даёт 100% интенсивности, если пиксель находится на расстоянии в 0.9 пикселя, то интенсивность будет 10%.

Иными словами, сто процентов интенсивности делится между пикселями, которые ограничивают векторную линию с двух сторон.



# *Алгоритм Брезенхема для генерации окружностей*

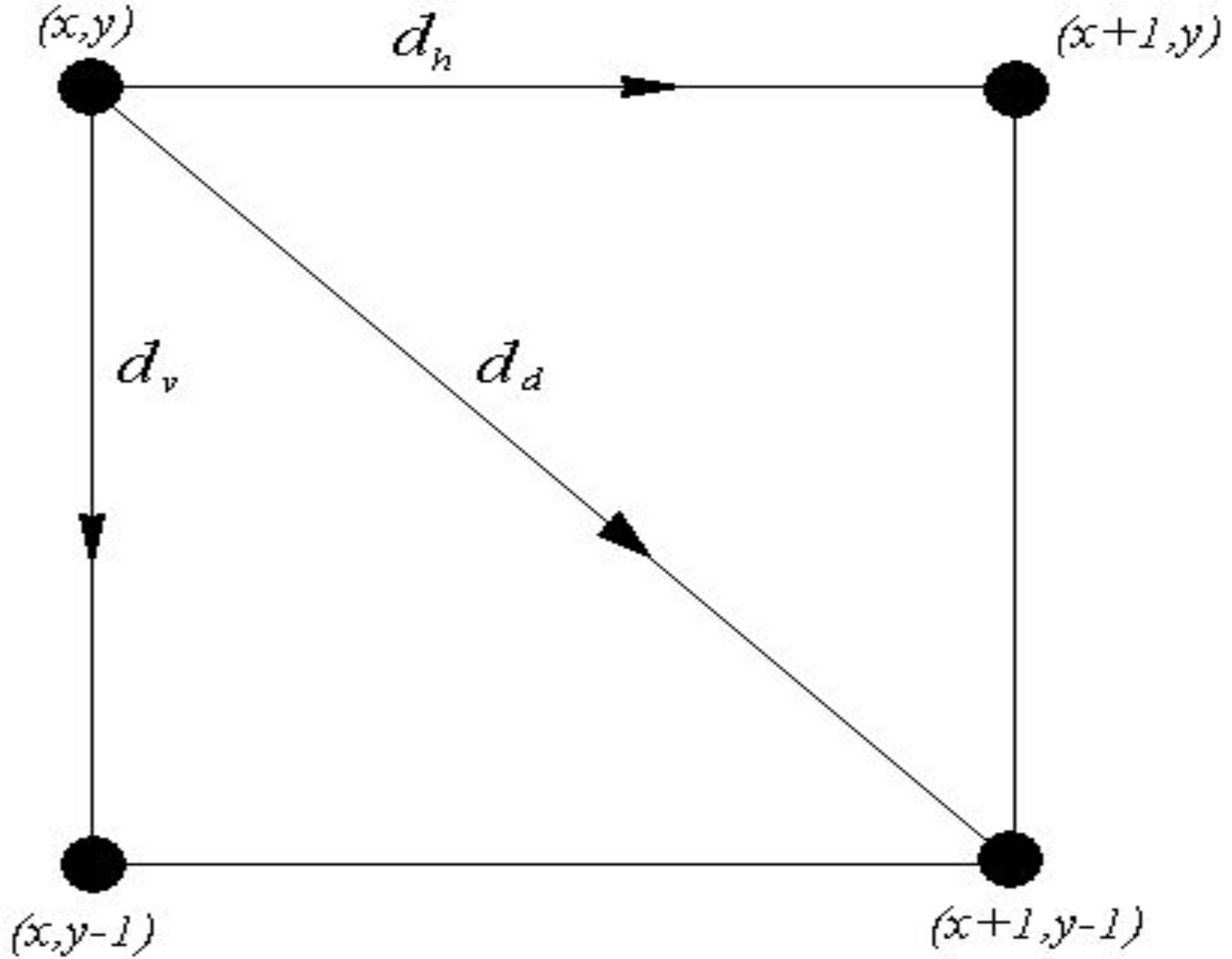
**Выбирается генерация по часовой стрелке с началом в точке x = 0, y = R.**

**Для любой заданной точки на окружности при генерации по часовой стрелке существует только три возможности выбрать следующий пиксел, наилучшим образом приближающий окружность: горизонтально вправо, по диагонали вниз и вправо, вертикально вниз. Эти направления обозначены соответственно $m_H$, $m_D$, $m_V$.**

**Алгоритм выбирает пиксел, для которого минимален квадрат расстояния между одним из этих пикселов и окружностью, т. е. минимум из** $d_h = |s_h|, \quad d_v = |s_v|, \quad d_d = |s_d|,$

$$s_h = (x+1)^2 + y^2 - r^2, \quad s_v = x^2 + (y-1)^2 - r^2,$$

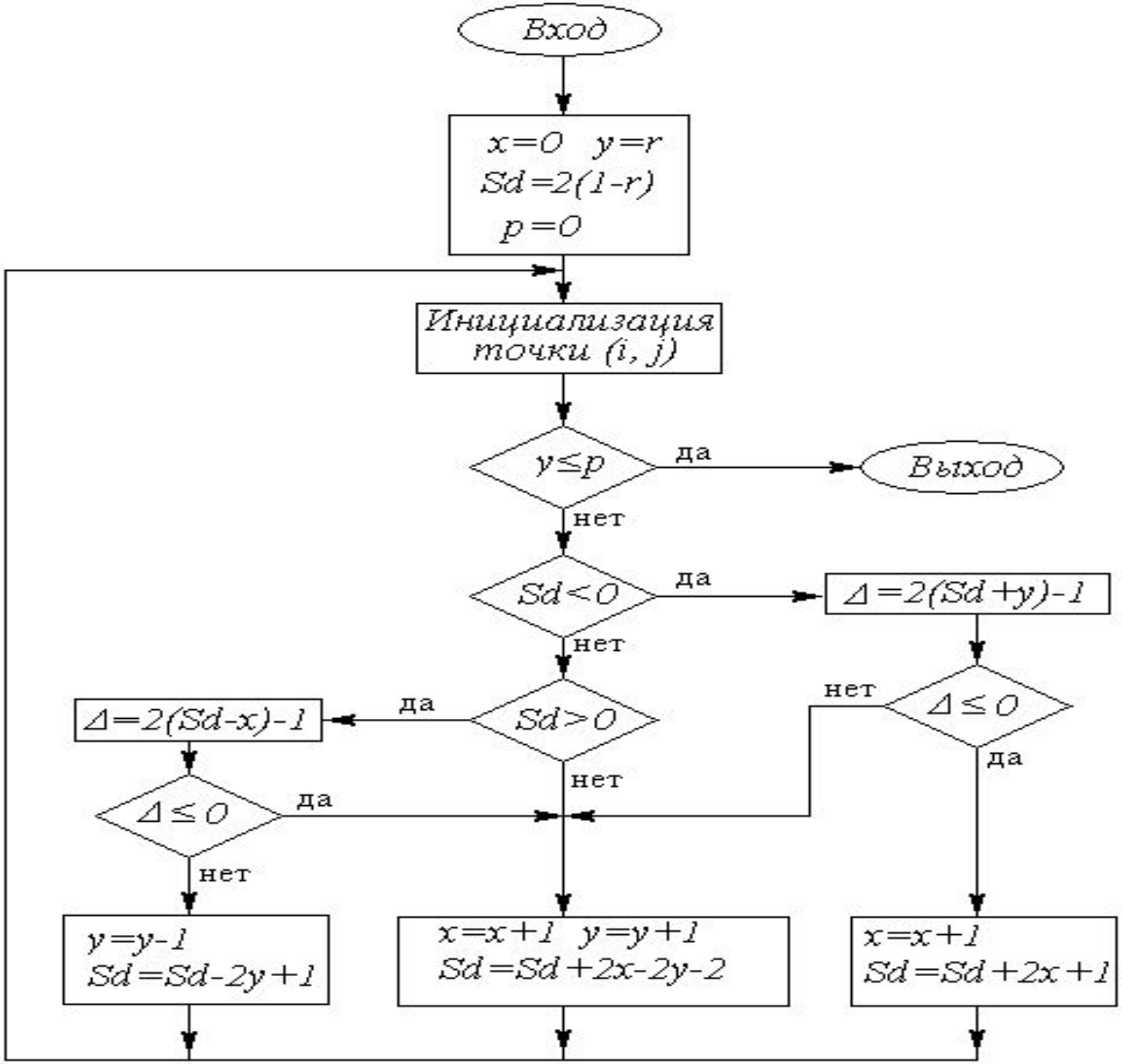
$$s_d = (x+1)^2 + (y-1)^2 - r^2.$$


(x,y)
$d_h$
(x+1,y)
$d_v$
$d_d$
(x,y-1)
(x+1,y-1)

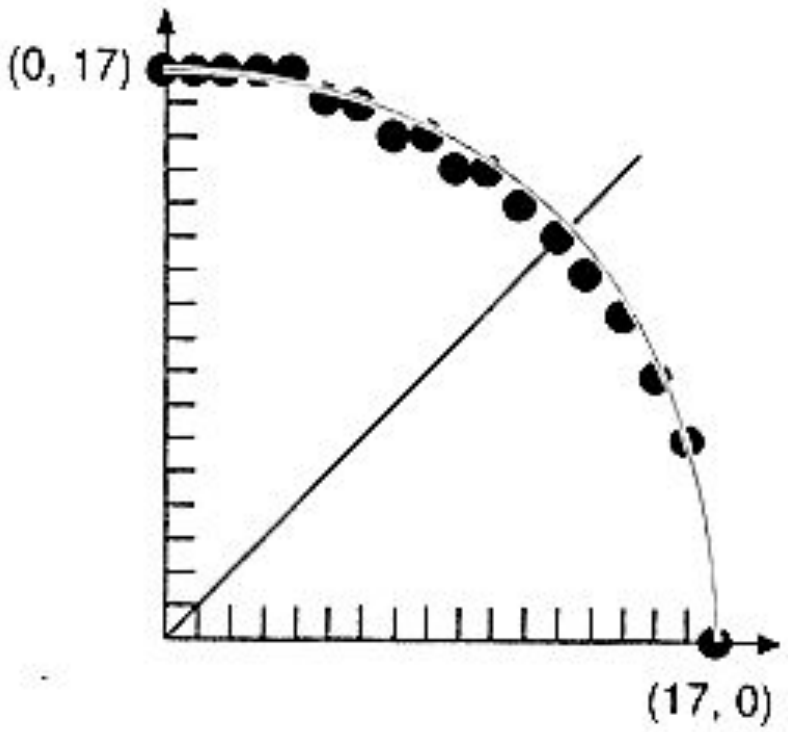
# Описание алгоритма

*Алгоритм* можно упростить, перейдя к анализу знаков величин $s_h, s_v, s_d$. При $s_d < 0$ диагональная точка лежит внутри окружности, поэтому ближайшими точками могут быть только диагональная и правая. Теперь достаточно проанализировать знак выражения $\Delta = d_v - d_d$. Если $\Delta \leq 0$, выбираем горизонтальный шаг, в противном случае - диагональный. Если же $s_d > 0$, то определяем знак $\Delta^1 = d_d - d_v$, и если $\Delta^1 \leq 0$, выбираем диагональный шаг, в противном случае - вертикальный. Затем вычисляется новое *значение* $s_d$, причем желательно минимизировать вычисления не только этой величины, но и величин $\Delta, \Delta^1$ на каждом шаге алгоритма. Путем несложных преобразований можно получить для первого шага алгоритма, что $\Delta = 2(s_d + y) - 1, \quad \Delta^1 = 2(s_d + x) - 1$.

После перехода в точку $(x', y'),\ x' = x + 1,\ y' = y + 1$ по диагонали новое *значение* $s_d$ вычисляется по формуле $s'_d = s_d + 2x' - 2y' + 2$, при горизонтальном переходе $(x' = x + 1, y' = y) \quad s'_d = s_d + 2x' + 1$, при вертикальном $(x' = x, y' = y - 1)$. $s'_d = s_d - 2y' + 1$.


Вход
x=0 y=r
Sd=2(1-r)
p=0
Инициализация точки (i, j)
y≤p
да
Выход
нет
Sd<0
да
Δ=2(Sd+y)-1
нет
Sd>0
да
Δ=2(Sd-x)-1
нет
Δ≤0
да
Δ≤0
да
нет
нет
y=y-1
Sd=Sd-2y+1
x=x+1 y=y+1
Sd=Sd+2x-2y-2
x=x+1
Sd=Sd+2x+1

# Аппроксимация окружности




Неявное и явное представление

$$x^2 + y^2 = R^2$$

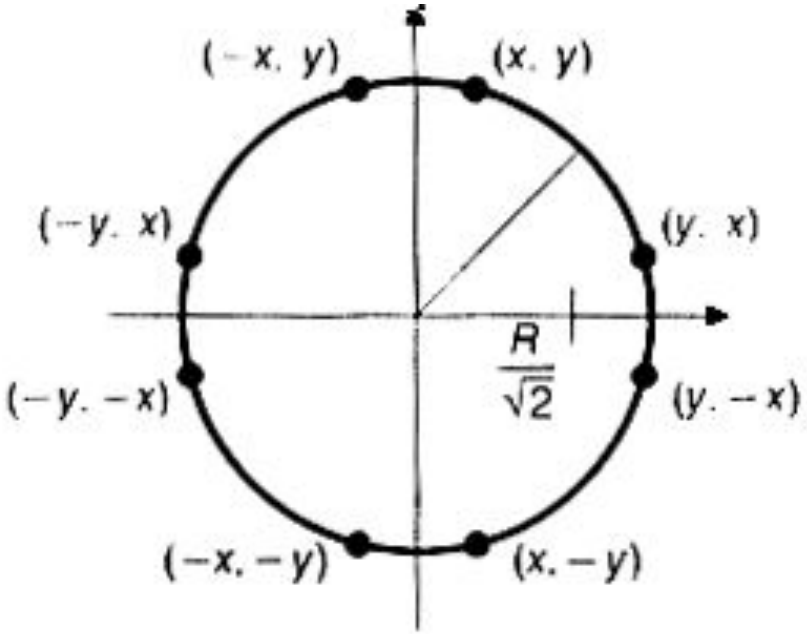
$$y = \pm\sqrt{R^2 - x^2}$$

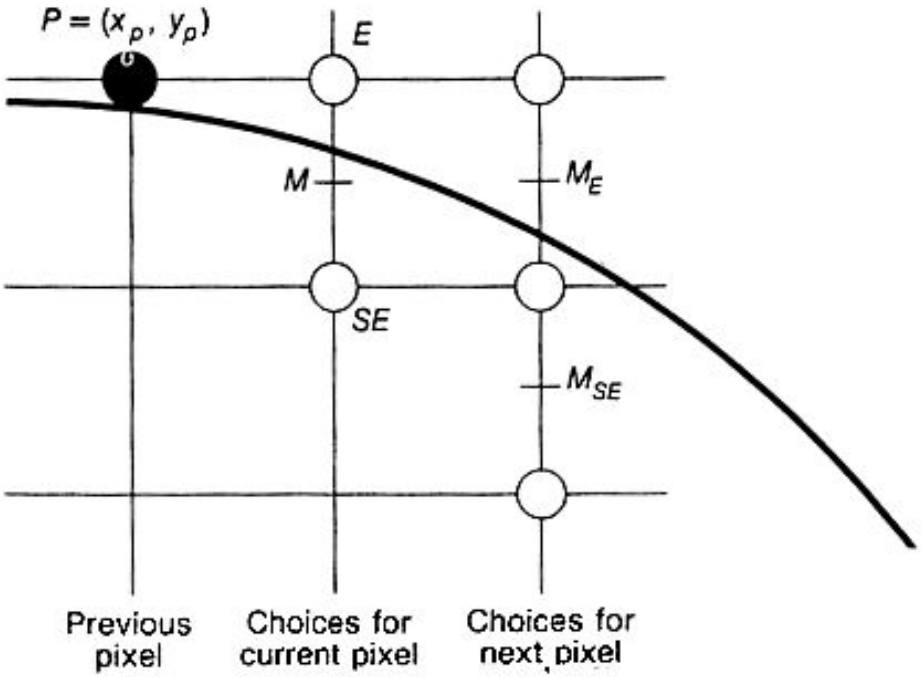
Параметрическое представление

$$x = R\cos\theta$$

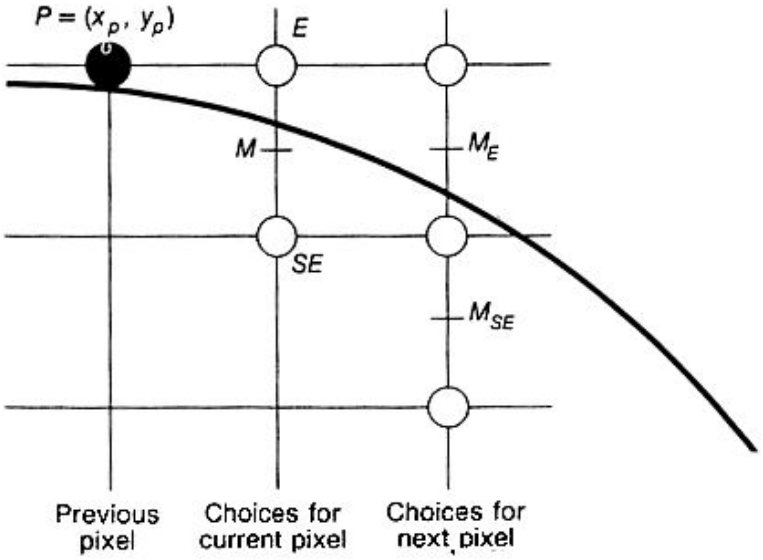
$$y = R\sin\theta$$

# Брезенхам для окружности (1/3)



# Брезенхам для окружности (2/3)




$$F(x, y) = x^2 + y^2 - R^2$$

Для точки P с коорд. $(x_p, y_p)$

$$d_{old} = F(x_p + 1, y_p - \frac{1}{2})$$

Для пиксела E:

$$d_{new} = f(x_p + 2, y_p - \frac{1}{2}) = d_{old} + (2x_p + 3)$$

$$\Delta d_E = (2x_p + 3)$$

Для пиксела SE:

$$d_{new} = d_{old} + (2x_p - 2y_p + 5)$$

$$\Delta d_{SE} = (2x_p - 2y_p + 5)$$

# Брезенхам для окружности (3/3)

В начальной точке (0, R)

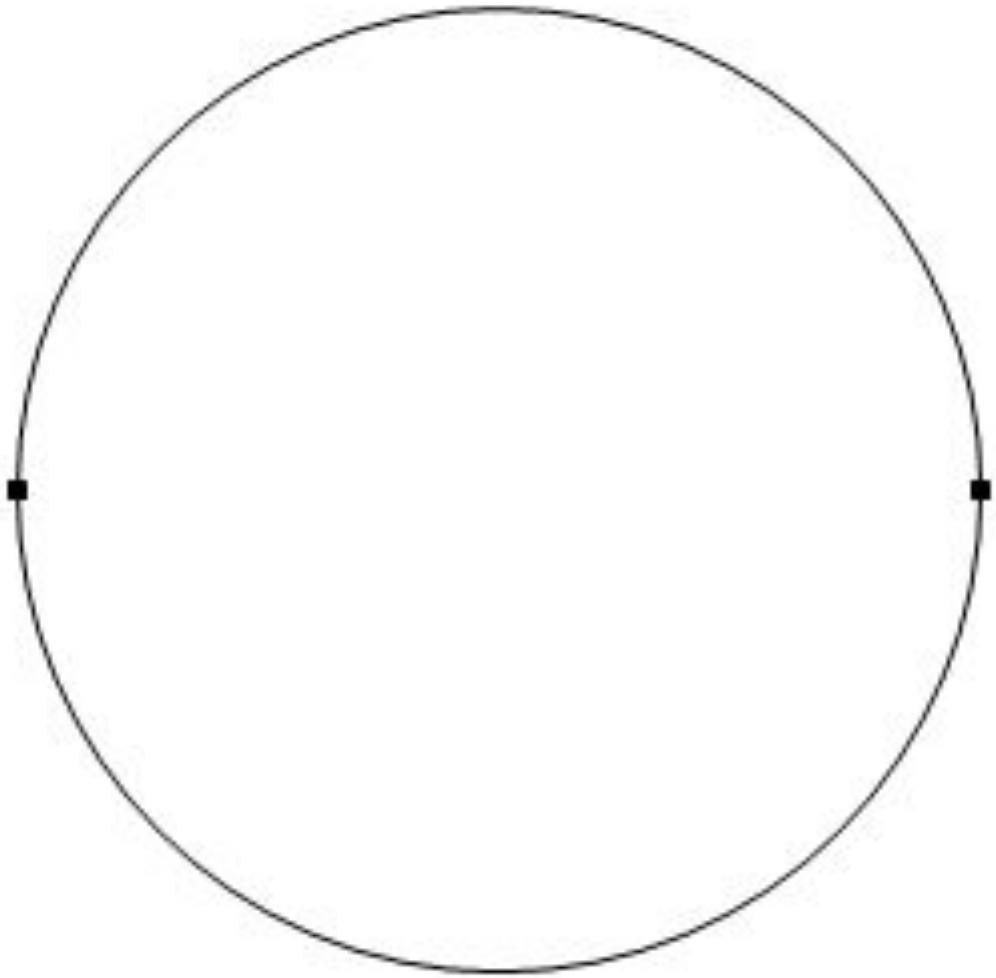
$$F(1, R-\frac{1}{2}) = 1 + (R^2 - R + \frac{1}{4}) - R^2 = \frac{5}{4} - R$$

$$d_0 = \frac{5}{4} - R$$

И опять нужно исключить вещественные операции.

Сделав замену $h = d - 1/4$, получим $h = 1 - R$.

Тогда необходимо сравнивать $h$ с $-1/4$, но так как приращения $d$ – целые числа, то сравнивать можно с нулем.



# Растровая развертка Эллипса

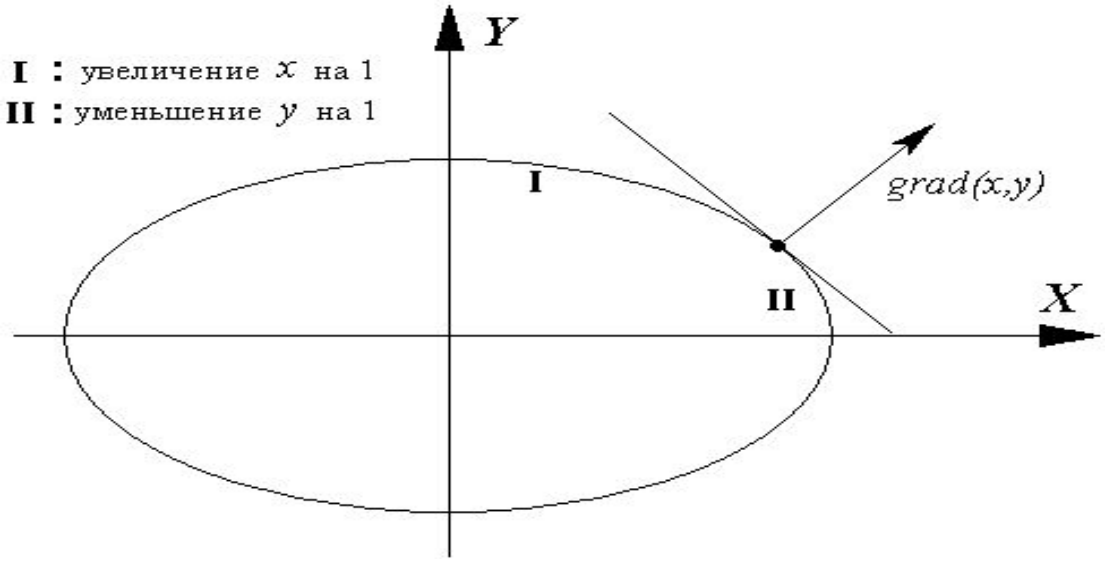
Для построения растровой развертки эллипса с осями, параллельными осям координат, и радиусами воспользуемся каноническим уравнением

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1,$$

которое перепишем в виде

$$f(x, y) \equiv b^2x^2 + a^2y^2 - a^2b^2 = 0.$$

В отличие от окружности, для которой было достаточно построить одну восьмую ее часть, а затем воспользоваться свойствами симметрии, *эллипс* имеет только две оси симметрии, поэтому придется строить одну четверть всей фигуры. За основу возьмем дугу, лежащую между точками в первом квадранте координатной плоскости.



# Нормаль и точка деления дуги

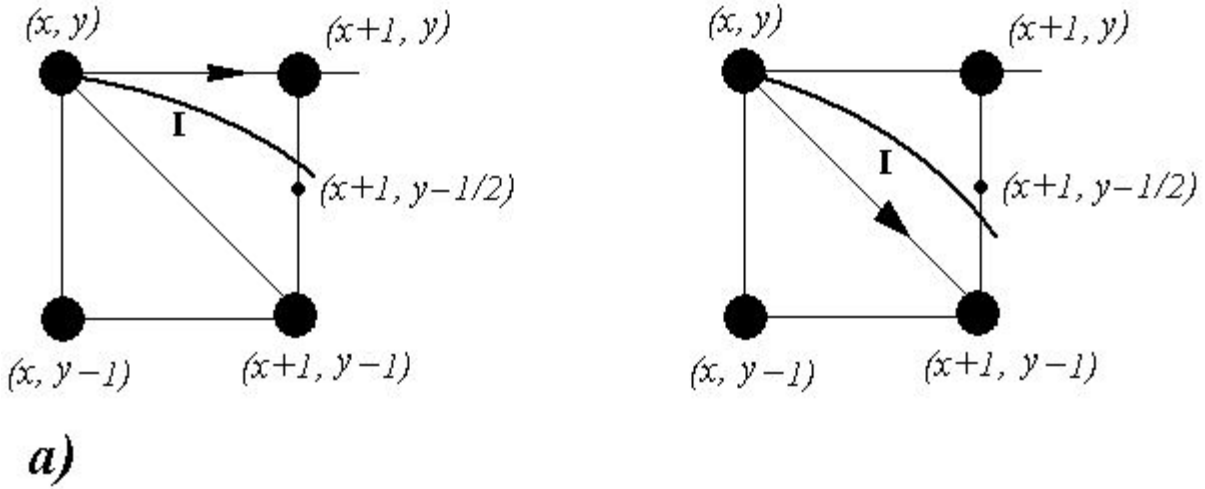
В каждой точке $(x, y)$ эллипса существует *вектор* нормали, задаваемый *градиентом функции* $f$. Дугу разобьем на две части: первая - с углом между нормалью и горизонтальной осью больше 45 $\deg$ (тангенс больше 1) и вторая - с углом, меньшим 45 $\deg$ (рис. 8.9). Движение вдоль дуги будем осуществлять в направлении по часовой стрелке, начиная с точки $(0, b)$. Вдоль всей дуги координата является *монотонно убывающей* функцией от $x$, но в первой части она убывает медленнее, чем растет *аргумент*, а во второй - быстрее. Поэтому при построении растрового образа в первой части будем увеличивать $x$ на единицу и искать соответствующее *значение* $y$, а во второй - сначала уменьшать *значение* $y$ на единицу и определять соответствующее *значение* $x$.

Направление нормали соответствует вектору

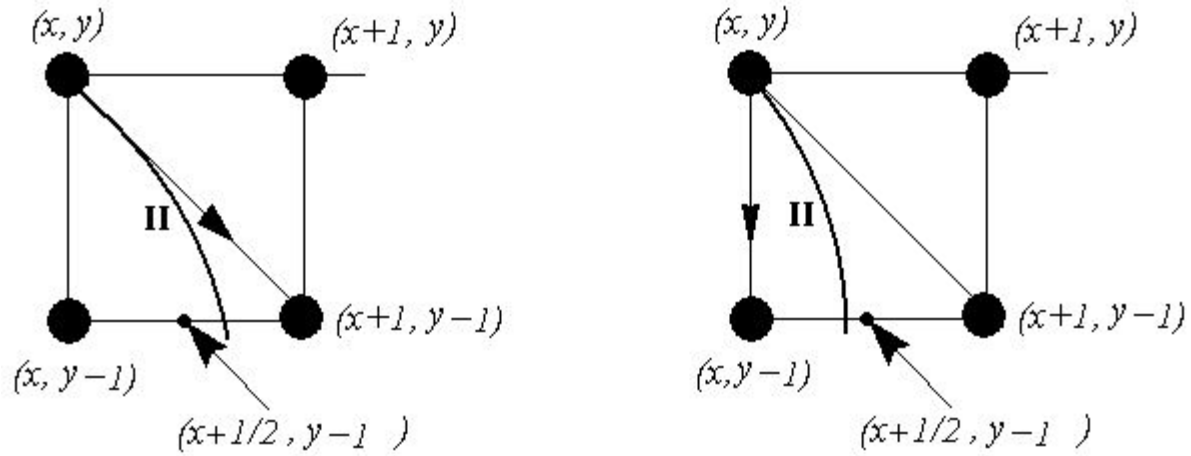
$$grad(x, y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) = (2b^2x, 2a^2y).$$

Отсюда находим тангенс угла наклона вектора нормали: $t = a^2y/b^2x$. Приравнивая его единице, получаем, что *координаты* точки деления дуги на вышеуказанные части удовлетворяют равенству $b^2x = a^2y$. Поэтому критерием того, что мы переходим ко второй области в целочисленных координатах, будет соотношение $a^2(y - 1/2) \leq b^2(x + 1)$, или, переходя к целочисленным операциям, $a^2(2y - 1) \leq 2b^2(x + 1)$.

# Схема перехода в первой и второй областях дуги эллипса




*а)*




*б)*

# Вычисление параметров

При перемещении вдоль первого участка дуги мы из каждой точки переходим либо по горизонтали, либо по диагонали, и критерий такого перехода напоминает тот, который использовался при построении растрового образа окружности. Находясь в точке $(x, y)$, мы будем вычислять *значение* $\Delta = f(x+1, y-\frac{1}{2})$. Если это *значение* меньше нуля, то дополнительная точка $(x+1, y-\frac{1}{2})$ лежит внутри эллипса, следовательно, ближайшая точка растра есть $(x+1, y)$, в противном случае это точка $(x+1, y-1)$ (рис. 8.10а).

На втором участке дуги возможен переход либо по диагонали, либо по вертикали, поэтому здесь сначала *значение координаты* y уменьшается на единицу, затем вычисляется $\Delta = f(x+\frac{1}{2}, y-1)$ и направление перехода выбирается аналогично предыдущему случаю (рис. 8.10б).

Остается оптимизировать *вычисление* параметра $\Delta$, умножив его на 4 и представив в виде функции координат точки. Тогда для первой половины дуги имеем

$$\widetilde{\Delta}(x,y) \equiv 4\Delta(x,y) = 4b^2(x+1)^2 + a^2(2y-1)^2 - 4a^2b^2,$$
$$\widetilde{\Delta}(x+1,y) = \widetilde{\Delta}(x,y) + 4b^2(2x+3),$$
$$\widetilde{\Delta}(x+1,y-1) = \widetilde{\Delta}(x,y) + 4b^2(2x+3) - 8a^2(y-1).$$

Для второй половины дуги получим

$$\widetilde{\Delta}(x,y) \equiv 4\Delta(x,y) = b^2(2x+1)^2 + 4a^2(y-1)^2 - 4a^2b^2,$$
$$\widetilde{\Delta}(x+1,y) = \widetilde{\Delta}(x,y) + 8b^2(x+1),$$
$$\widetilde{\Delta}(x+1,y-1) = \widetilde{\Delta}(x,y) + 8b^2(x+1) - 4a^2(2y+3).$$

# Литература

https://www.intuit.ru/studies/courses/70/70/lecture/2106?page=3
https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%91%D1%80%D0%B5%D0%B7%D0%B5%D0%BD%D1%85%D1%8D%D0%BC%D0%B0
https://habr.com/ru/post/185086/
Перемитина О.Н. Компьютерная Графика ТУСУР 2012