Лекция 2

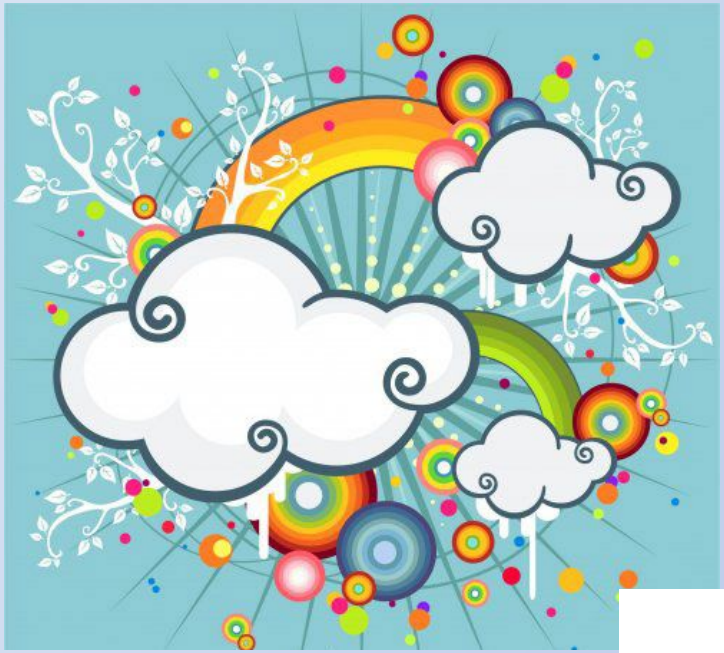
# Растровая графика

Учебные вопросы лекции:

1. Виды компьютерной графики. Общие сведения о растровой графике
2. Цветовое разрешение и цветовые модели
1. Алгоритм Брезенхэма
2. Достоинства и недостатки растровой графики
3. Характеристики растровых изображений
4. Средства для работы с растровой графикой

# Виды компьютерной графики

Векторная графика



Фрактальная графика

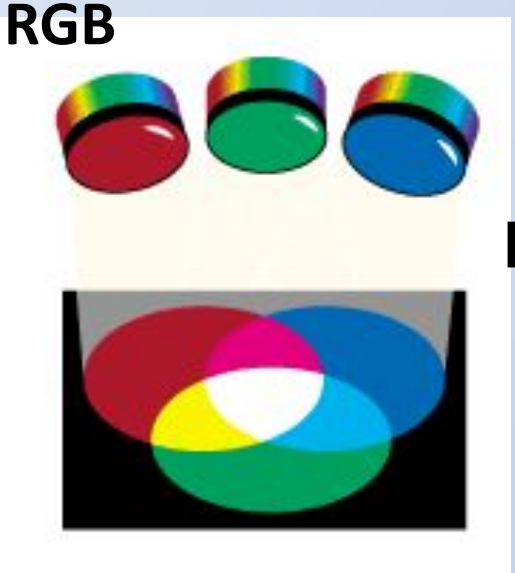
Растровая графика

# Разрешение изображения и его размер

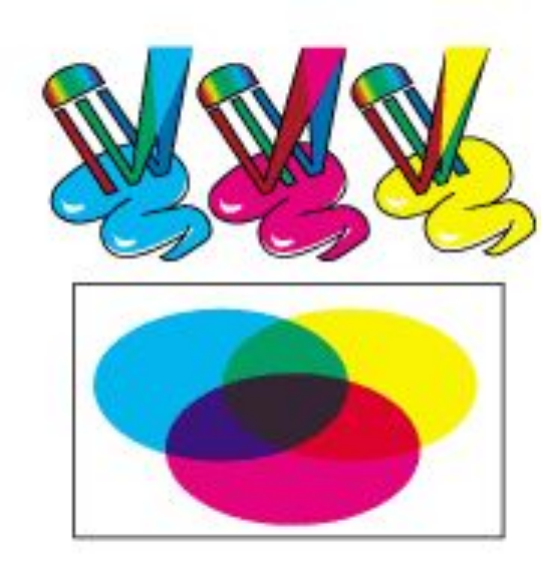
| **Размер иллюстрации** | **75 dpi.** | **150 dpi.** | **300 dpi.** | **600 dpi.** |
|---|---|---|---|---|
| 212×163 | 108×81 | 55×40 | 28×20 | |
| 800×600 | 271×203 | 136×102 | 68×51 | 34×26 |
| 1024×768 | 344×260 | 173×130 | 88×66 | 44×33 |
| 1152×864 | 390×293 | 195×146 | 98×73 | 49×37 |
| 1600×1200 | 542×406 | 271×203 | 136×102 | 68×51 |

# Цветовое разрешение и цветовые модели

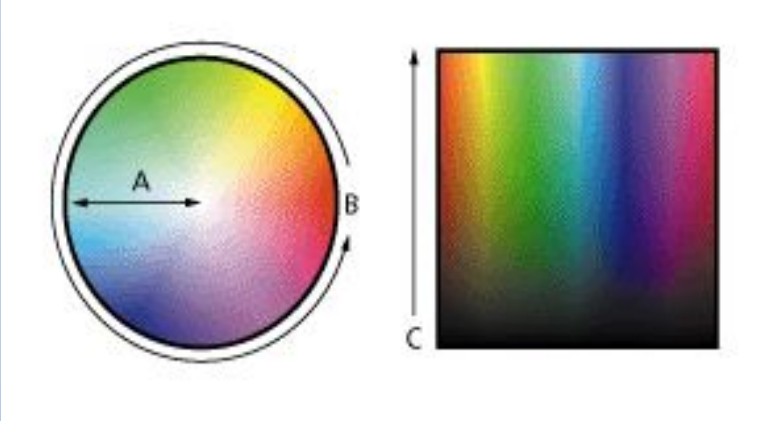
**Цветовая модель RGB**



**Цветовая модель CMYK**



**Цветовая модель HSB**



# Таблица «безопасных» цветов

| | | | | | | |
|---|---|---|---|---|---|---|
| 003333 | 336699 | 3366CC | 003399 | 009999 | 0000CC | 000066 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 006666 | 006699 | 0099CC | 0066CC | 0033CC | 00FFFF | 3333FF | 333399 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 669999 | 009999 | 33CCCC | 00CCFF | 0099FF | 0066FF | 3366FF | 3333CC | 666699 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 339966 | 00CC99 | 00FFCC | 00FFFF | 33CCFF | 3399FF | 6699FF | 6666FF | 6600FF | 6600CC |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 339933 | 00CC66 | 00FF99 | 66FFCC | 66FFFF | 66CCFF | 99FFCC | 9999FF | 9966FF | 9933FF | 9900FF |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 006600 | 00CC00 | 00FF00 | 66FF99 | 99FFCC | CCFFFF | CCCCFF | CC99FF | CC66FF | CC33FF | CC00FF | 99OOCC |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 003300 | 009933 | 33CC33 | 66FF66 | 99FF99 | CCFFCC | FFFFFF | FFCCFF | FF99FF | FF66FF | FF00FF | CC00CC | 660066 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 336600 | 009900 | 66FF33 | 99FF66 | CCFF99 | FFFFCC | FFCCCC | FF99CC | FF66CC | FF33CC | CC0099 | 993399 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 333300 | 669900 | 99FF33 | CCFF66 | FFFF99 | FFCC99 | FF9999 | FF6699 | FF3399 | CC3399 | 990099 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 666633 | 99CC00 | CCFF33 | FFFF66 | FFCC66 | FF9966 | FF6666 | FF0066 | CC6699 | 993366 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 999966 | CCCC00 | FFFF00 | FFCC00 | FF9933 | FF6600 | FF5050 | CC0066 | 660033 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 996633 | CC9900 | FF9900 | CC6600 | FF3300 | FF0000 | CC0000 | 990033 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 663300 | 996600 | CC3300 | 993300 | 990000 | 800000 | 993333 |

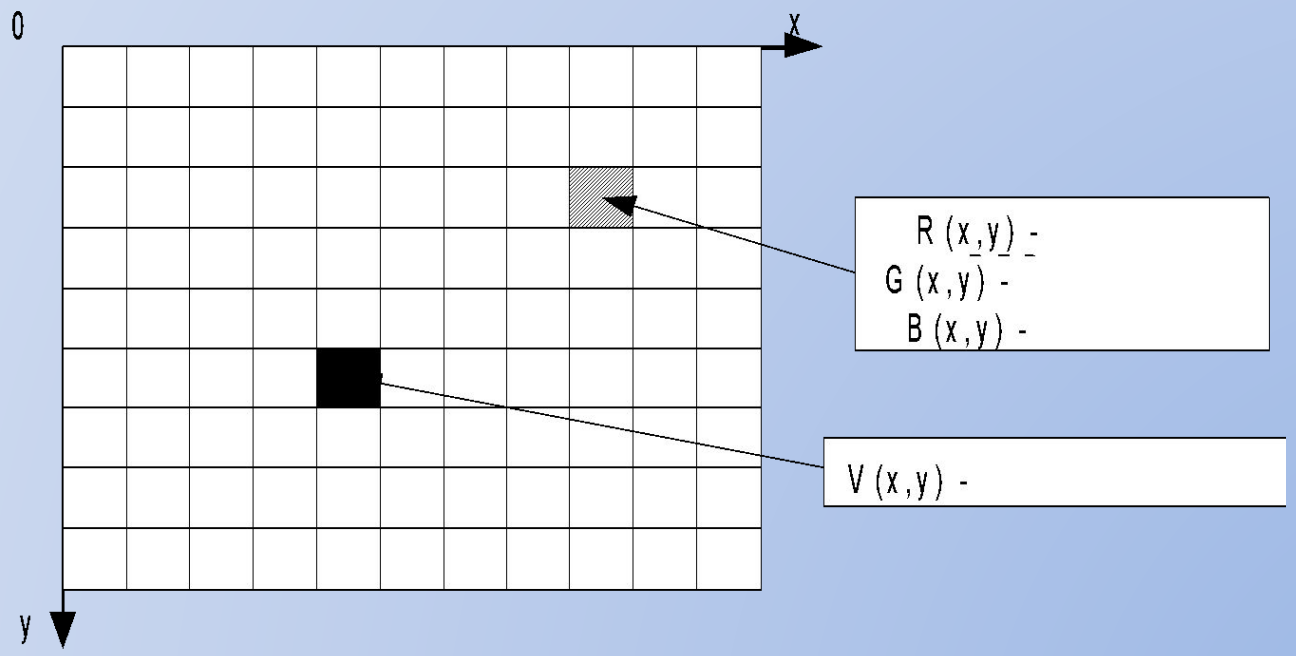
FFFFFF

| | | |
|---|---|---|
| CCCCCC | 999999 | 666666 |
| C0C0C0 | 808080 | 333333 |

000000

# Виды растров

**Растр** – это порядок расположения точек (растровых элементов). На рис. 2. изображен растр, элементами которого являются квадраты, такой растр называется **прямоугольным**, именно такие растры наиболее часто используются.



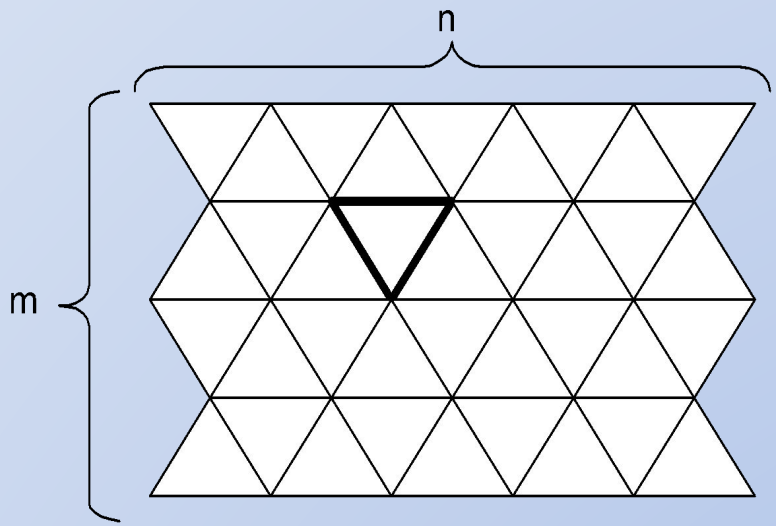



Рис. 3. Треугольный растр

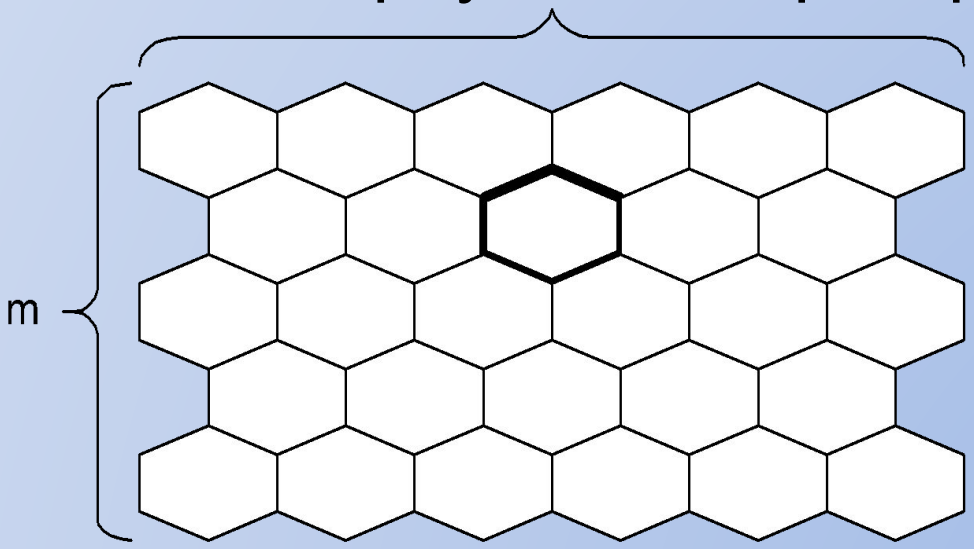



Рис. 4. «Гексагональный растр»

В прямоугольном растре построение линии осуществляется двумя способами:

- Результат – восьмисвязная линия. Соседние пиксели линии могут находится в одном из восьми возможных (см. рис. 5а) положениях. Недостаток – слишком тонкая линия при угле 45°.
- Результат – четырехсвязная линия. Соседние пиксели линии могут находится в одном из четырех возможных положениях. Недостаток – избыточно толстая линия при угле 45°.

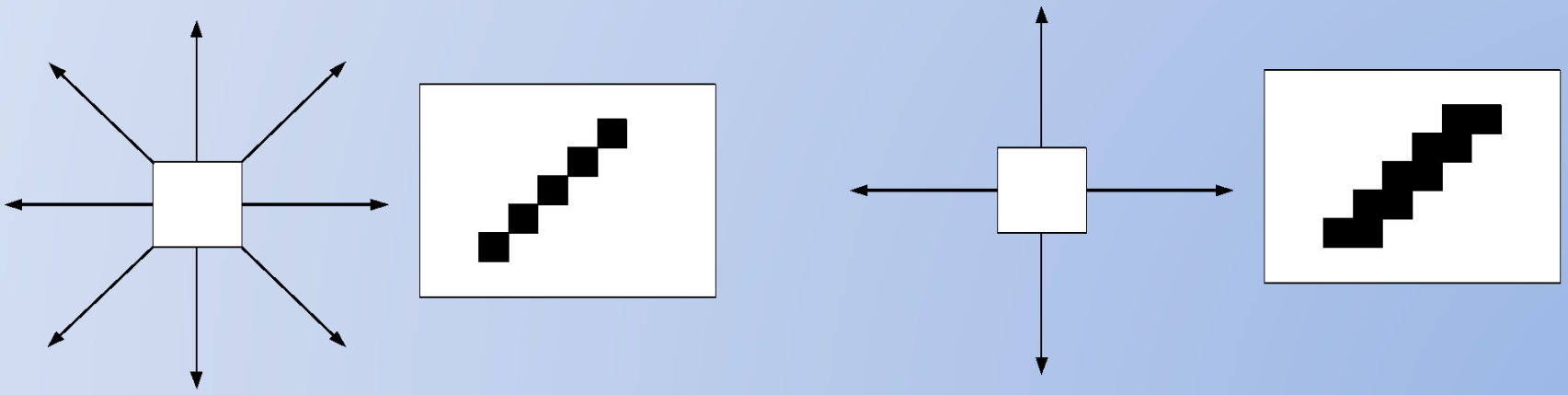
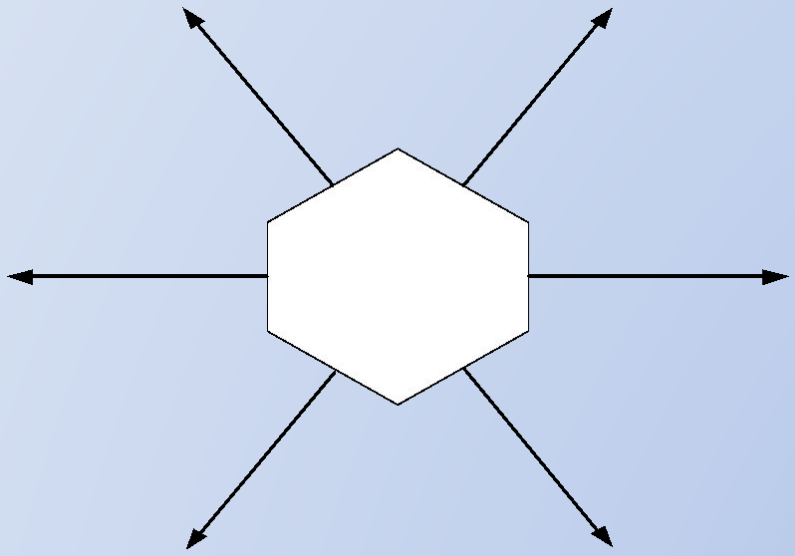



Рис. 5. Построение линии в прямоугольном растре

В гексагональном растре линии шестисвязные такие линии более стабильны по ширине, т.е. дисперсия ширины линии меньше, чем в квадратном растре.



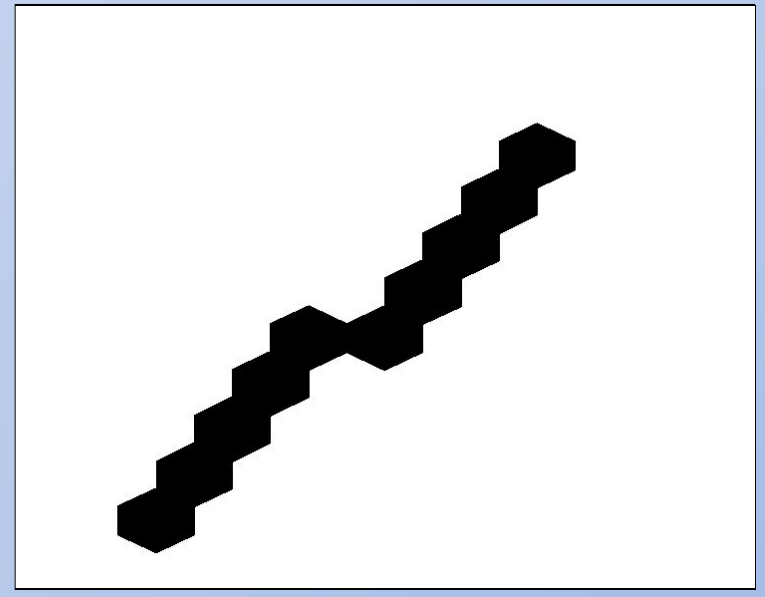
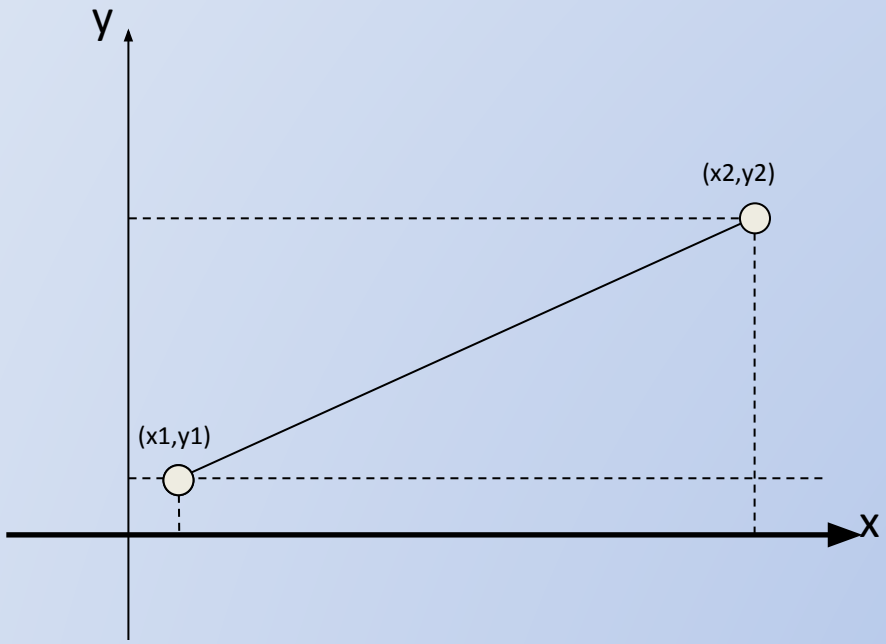
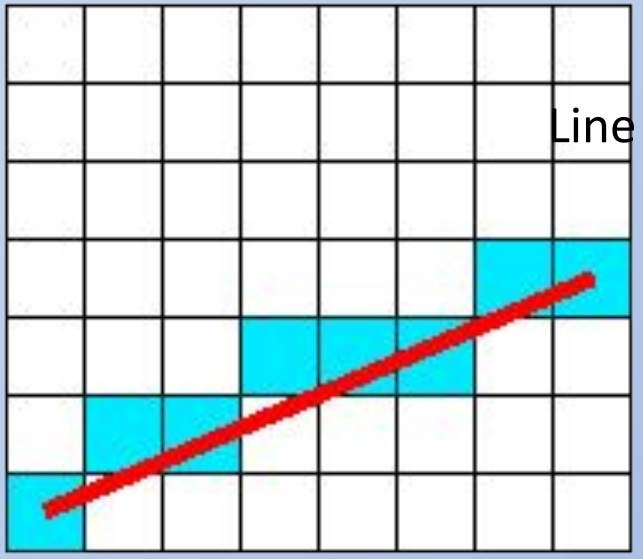


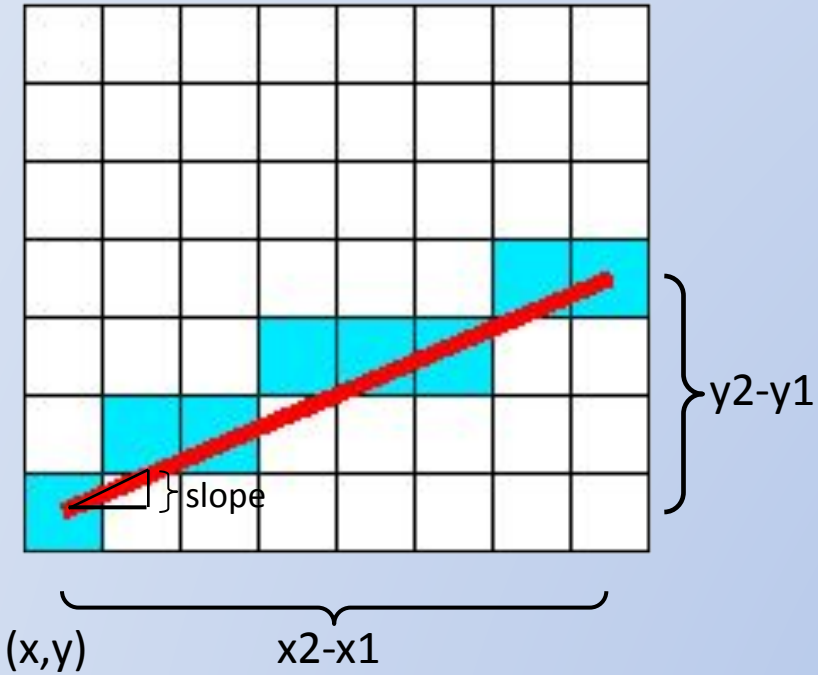
Рис. 6. Построение линии в гексагональном растре

# Алгоритм Брезенхэма для рисования линии



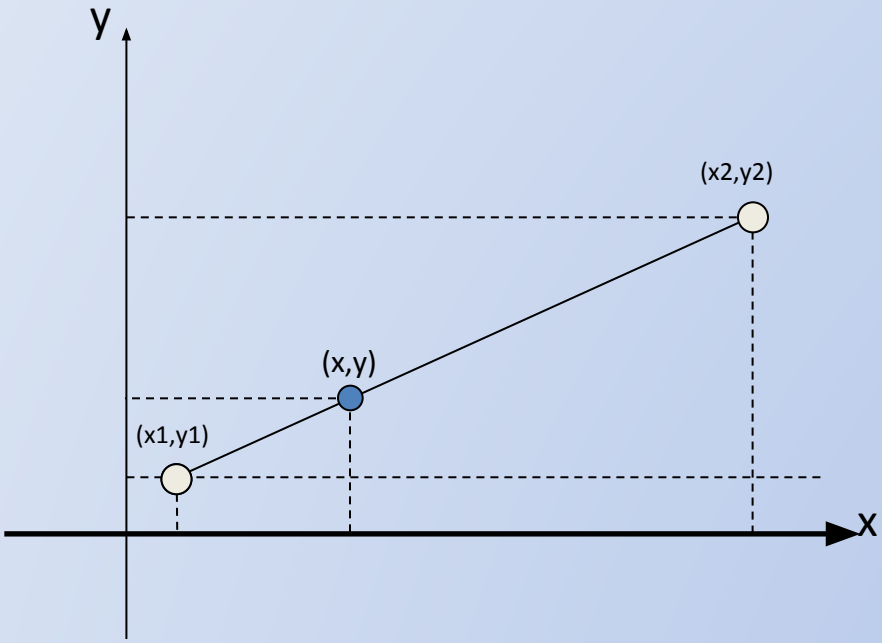



# Line: Digital Differential Analyzer (DDA)




```
int x;
float
  y,
  slope = (y2 - y1) / (x2 - x1);

x = x1; y = y1 + 0.5;

while (x <= x2)
{
  SetPixel(x, (int)y);
  y = y + slope;
  x = x + 1;
}
```

# Line: Алгоритм Брезенхема (метод центральной точки)




$$\frac{x - x1}{x2 - x1} = \frac{y - y1}{y2 - y1}$$

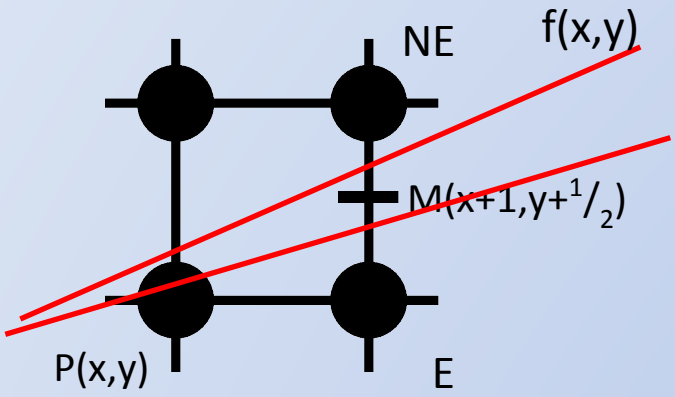
$$(x - x1) \cdot (y2 - y1) - (y - y1) \cdot (x2 - x1) = 0$$

$$dy \cdot x - dx \cdot y - (x1 \cdot dy - y1 \cdot dx) = 0$$

$$f(x, y) = dy \cdot x - dx \cdot y - (x1 \cdot dy - y1 \cdot dx)$$

$$\begin{cases} f(x, y) > 0 & \text{точка } (x,y) \text{ «ниже» прямой} \\ f(x, y) = 0 & \text{точка } (x,y) \text{ «лежит» на прямой} \\ f(x, y) < 0 & \text{точка } (x,y) \text{ «выше» прямой} \end{cases}$$

# Line: Алгоритм Брезенхема (метод центральной точки)

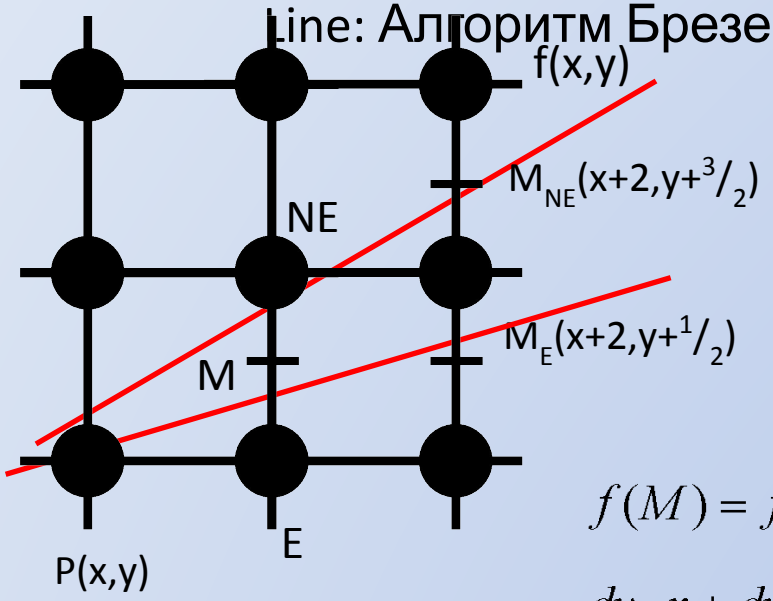



Подставляем точку M в функцию f:

- если f(M) > 0 выбираем точку NE
- если f(M) <= 0 выбираем точку E

```
int x, y;

x = x1; y = y1;

SetPixel(x, y);
count = dx;
while (count > 0)
{
  count = count - 1;

  if (f(x + 1, y + 0.5) > 0)
    y = y + 1;
  x = x + 1;
  SetPixel(x, y);
}

/* f(x, y) =
 *    dy * x - dx * y - (x1 * dy - y1 * dx)
 * dx = x2 - x1; dy = y2 - y1;
 */
```

# Line: Алгоритм Брезенхема (метод центральной точки)




Подставляем точку M в функцию f:

- если f(M) > 0 выбираем точку NE
- если f(M) <= 0 выбираем точку E

Изменения значения f(M) при переходе к новым точкам (E или NE):

$$f(M) = f(x+1, y+{}^1\!/_2) = dy\cdot(x+1) - dx\cdot(y+{}^1\!/_2) - C =$$

$$dy\cdot x + dy - dx\cdot y - {}^{dx}\!/_2 - C$$

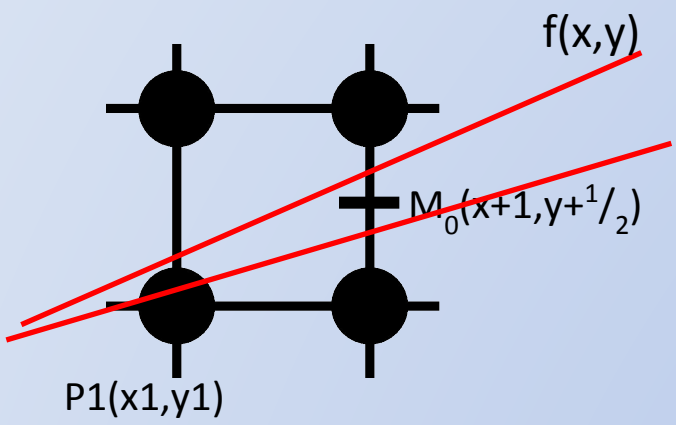
$$f(M_E) = f(x+2, y+{}^1\!/_2) = dy\cdot(x+2) - dx\cdot(y+{}^1\!/_2) - C =$$

$$dy\cdot x + 2\cdot dy - dx\cdot y - {}^{dx}\!/_2 - C = \underline{f(M) + dy}$$

$$f(M_{NE}) = f(x+2, y+{}^3\!/_2) = dy\cdot(x+2) - dx\cdot(y+{}^3\!/_2) - C =$$

$$dy\cdot x + 2\cdot dy - dx\cdot y - {}^{3\cdot dx}\!/_2 - C = \underline{f(M) + dy - dx}$$

# Line: Алгоритм Брезенхема (метод центральной точки)




Известны приращения f.

Найдем первоначальное значение для точки (x1,y1)

$$f(M_0) = f(x1+1, y1+\tfrac{1}{2}) =$$

$$dy \cdot (x1+1) - dx \cdot (y1+\tfrac{1}{2}) - (x1 \cdot dy - y1 \cdot dx) =$$

$$dy - \tfrac{dx}{2}$$

# Line: Алгоритм Брезенхема (метод центральной точки)

Сохранились вещественные числа.

Сделаем замену: 2f = e

Тогда помеченные строки изменяться на:

```
e = 2 * dy - dx;
```

```
e > 0
```

```
e = e + 2 * dy - 2 *dx;
```

```
e = e + 2 * dy
```

и e – целое число.

```
int x, y, dx, dy;
float f;

dx = x2 - x1;
dy = y2 - y1;
f = dy - dx / 2.0;

x = x1; y = y1;
SetPixel(x, y);
count = dx;
while (count > 0)
{
  count = count - 1;

  if (f > 0)
  {
    y = y + 1;
    f = f + (dy - dx);
  }
  else
    f = f + dy;
  x = x + 1;
  SetPixel(x, y);
}
```

# Line: Алгоритм Брезенхема (метод центральной точки)

```
int x, y, dx, dy, incrE, incrNE, e;

dx = x2 - x1;
dy = y2 - y1;
e = 2 * dy - dx;
incrE = 2 * dy;
incrNE = 2 * dy - 2 * dx;

x = x1; y = y1;
SetPixel(x, y);
count = dx;
while (count > 0)
{
  count = count - 1;

  if (f > 0)
  {
    y = y + 1;
    f = f + incrNE;
  }
  else
    f = f + incrE;
  x = x + 1;
  SetPixel(x, y);
}
```

# Line: Алгоритм с использованием Fixed Point (DDA)

Fixed Point – вещественные числа с фиксированной точкой.

Рассмотрим 4-байтное целое:

| 2b целая часть | 2b дробная часть |
|---|---|

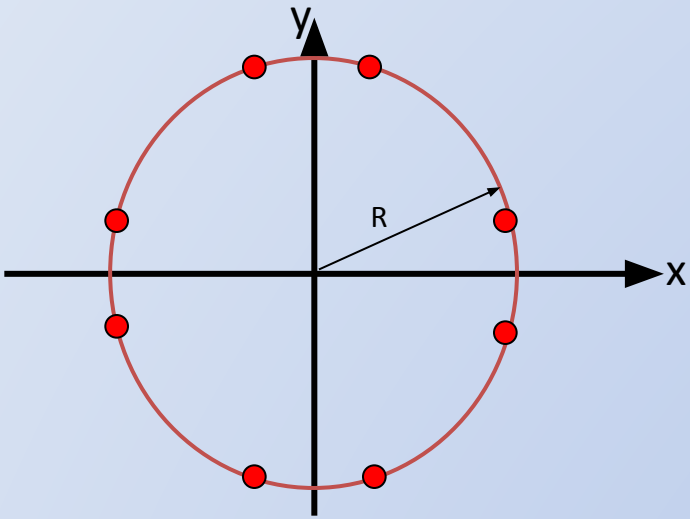
Точность $^{1}/_{65536}$

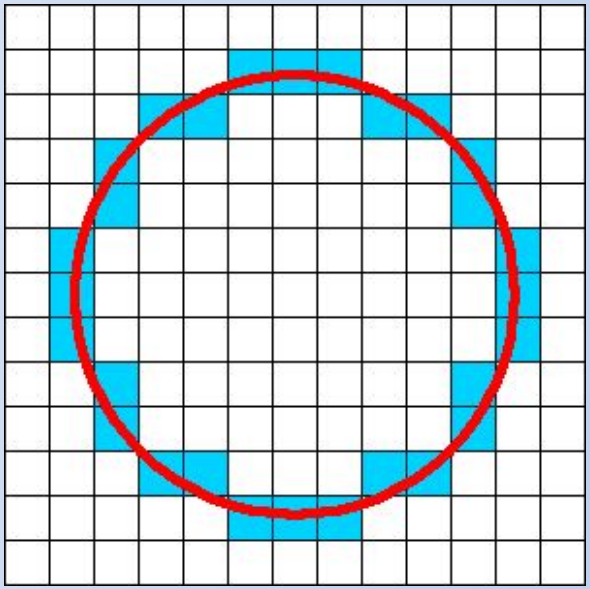
Если x и y fixed point, то

- сложение не изменяется (x+y)
- вычитание не изменяется (x-y)
- целая часть – «двоичный сдвиг» вправо на 16 бит (x >> 16)
- из целого: x = a << 16

```
int x;
long
  y,
  slope = ((y2 - y1) << 16) / (x2 - x1);

x = x1; y = y1 << 16;

SetPixel(x1, y1);
count = x2 - x1;
while (count > 0)
{
  count = count - 1;

  y = y + slope;
  x = x + 1;
  SetPixel(x, y >> 16);
}
```

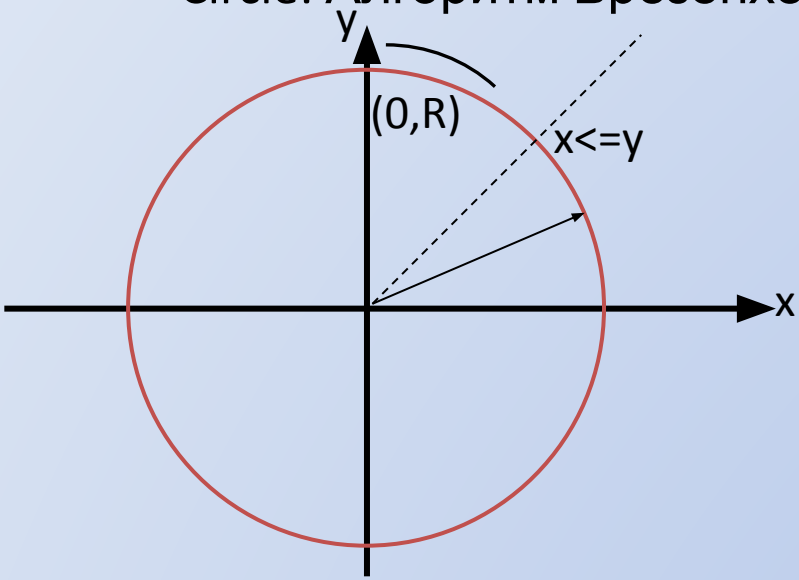
# Circle



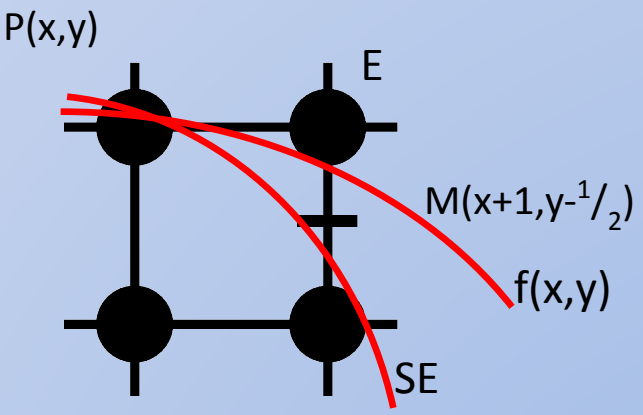
```
SetPixel4(x, y):
   SetPixel(Cx, Cy + R);
   SetPixel(Cx, Cy - R);
   SetPixel(Cx + R, Cy);
   SetPixel(Cx - R, Cy);

SetPixel8(x, y):
   SetPixel(Cx + x, Cy + y);
   SetPixel(Cx - x, Cy + y);
   SetPixel(Cx + x, Cy - y);
   SetPixel(Cx - x, Cy - y);
   SetPixel(Cx + y, Cy + x);
   SetPixel(Cx - y, Cy + x);
   SetPixel(Cx + y, Cy - x);
   SetPixel(Cx - y, Cy - x);
```

# Circle: Алгоритм Брезенхема (метод центральной точки)




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

$$\begin{cases} f(x,y) > 0 & \text{точка } (x,y) \text{ вне круга} \\ f(x,y) = 0 & \text{точка } (x,y) \text{ на окружности} \\ f(x,y) < 0 & \text{точка } (x,y) \text{ внутри круга} \end{cases}$$

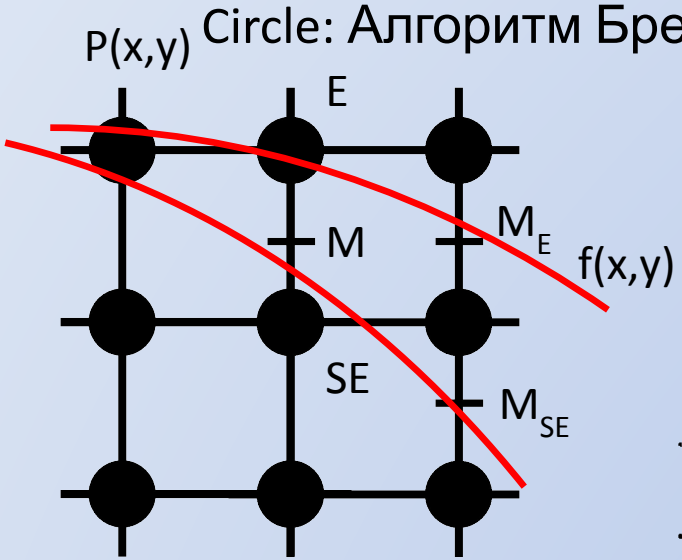



```
int x, y;

x = 0; y = R;

SetPixel4(x, y);
while (x <= y)
{
  if (f(x + 1, y - 0.5) < 0)
    y = y - 1;
  x = x + 1;
  SetPixel8(x, y);
}
```

Подставляем точку M в функцию f:

- если f(M) >= 0 выбираем точку SE
- если f(M) < 0 выбираем точку E

# Circle: Алгоритм Брезенхема (метод центральной точки)




Изменения значения f(M) при переходе к новым точкам (E или SE):

$$f(M) = f(x+1, y-\tfrac{1}{2}) = (x+1)^2 + (y-\tfrac{1}{2})^2 - R^2 =$$

$$x^2 + 2x + 1 + y^2 - y + \tfrac{1}{4} - R^2$$

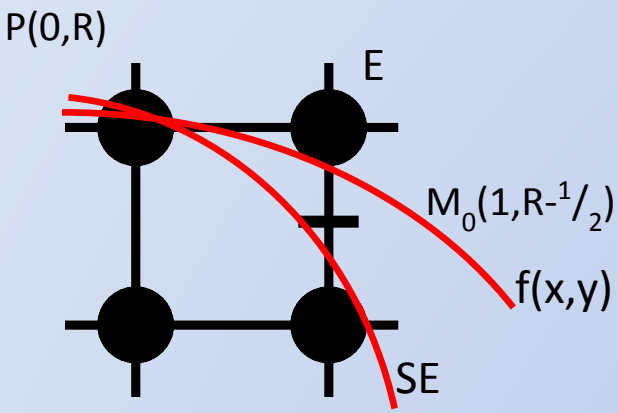
$$f(M_E) = f(x+2, y-\tfrac{1}{2}) = (x+2)^2 + (y-\tfrac{1}{2})^2 - R^2 =$$

$$x^2 + 4x + 4 + y^2 - y + \tfrac{1}{4} - R^2 = f(M) + 2x + 3$$

$$f(M_{NE}) = f(x+2, y-\tfrac{3}{2}) = (x+2)^2 + (y-\tfrac{3}{2})^2 - R^2 =$$

$$x^2 + 4x + 4 + y^2 - 3y + \tfrac{9}{4} - R^2 = f(M) + 2x - 2y + 5$$

# Circle: Алгоритм Брезенхема (метод центральной точки)




Определили приращения f.
Найдем первоначальное значение для точки (x1,y1)

$$f(M_0) = f(1, R - \tfrac{1}{2}) = 1^2 + (R - \tfrac{1}{2})^2 - R^2 =$$

$$1 + R^2 - R + \tfrac{1}{4} - R^2 = \tfrac{5}{4} - R$$

```
int x, y, f = 1 - R;

x = 0; y = R;

SetPixel4(x, y);
while (x <= y)
{
  if (f > 0)
  {
    y = y - 1;
    f = f + 2 * (x - y) + 5;
  }
  else
    f = f + 2 * x + 3;
  x = x + 1;
  SetPixel8(x, y);
}
```

Все приращения - целые. Сравнение f с 0 строгое: ‘<‘.
Поэтому из первоначального f можно вычесть $^1/_4$.

# Методы сжатия растровых изображений:

RLE (Run Length Encoding) – метод сжатия, заключающийся в поиске последовательностей одинаковых пикселей в сточках растрового изображения («красный, красный, ..., красный» записывается как «N красных»).

LZW (Lempel–Ziv–Welch) – более сложный метод, ищет повторяющиеся фразы – одинаковые последовательности пикселей разного цвета. Каждой фразе ставится в соответствие некоторый код, при расшифровке файла код замещается исходной фразой.

## Геометрические характеристики растра

Для растровых изображений, состоящих из точек, особую важность имеет понятие *разрешения,* выражающее количество точек, приходящихся на единицу длины. При этом следует различать:

- разрешение оригинала;
- разрешение экранного изображения;
- разрешение печатного изображения.

**Разрешение оригинала.** Разрешение оригинала измеряется в *точках на дюйм (dots per inch – dpi)* и зависит от требований к качеству изображения и размеру файла, способу оцифровки и создания исходной иллюстрации, избранному формату файла и другим параметрам. В общем случае действует правило: чем выше требование к качеству, тем выше должно быть разрешение оригинала.

**Разрешение экранного изображения.** Для экранных копий изображения элементарную точку растра принято называть *пикселом.* Размер пиксела варьируется в зависимости от выбранного *экранного разрешения* (из диапазона стандартных значений), *разрешение оригинала* и масштаб отображения.

Мониторы для обработки изображений с диагональю 20–21 дюйм (профессионального класса), как правило, обеспечивают стандартные экранные разрешения 640x480, 800x600, 1024x768,1280x1024,1600x1200,1600x1280, 1920x1200, 1920x1600 точек. Расстояние между соседними точками у качественного монитора составляет 0,22–0,25 мм.

**Разрешение печатного изображения и понятие линиатуры.** Размер точки растрового изображения как на твердой копии (бумага, пленка и т. д.), так и на экране зависит от примененного метода и параметров *растрирования* оригинала. При растрировании на оригинал как бы накладывается сетка линий, ячейки которой образуют *элемент растра.* Частота сетки растра измеряется числом *линий на дюйм (lines per inch – lpi)* и называется *линиатурой.*

Размер точки растра рассчитывается для каждого элемента и зависит от интенсивности тона в данной ячейке. Чем больше интенсивность, тем плотнее заполняется элемент растра.

**Размер** растра обычно измеряется количеством пикселов по горизонтали и вертикали. Можно сказать, что для компьютерной графики зачастую наиболее удобен растр с одинаковым шагом для обеих осей, то есть dpiX = dpiУ. Это удобно для многих алгоритмов вывода графических объектов. Иначе – проблемы. Например, при рисовании окружности на экране дисплея EGA (устаревшая модель компьютерной видеосистемы, ее растр– прямоугольный, пикселы растянуты по высоте, поэтому для изображения окружности необходимо генерировать эллипс).

**Динамический диапазон.** Качество воспроизведения тоновых изображений принято оценивать *динамическим диапазоном (D).* Это *оптическая плотность,* численно равная десятичному логарифму величины, обратной *коэффициенту пропускания*

Для оптических сред, пропускающих свет, динамический диапазон лежит в пределах от 0 до 4. Для поверхностей, отражающих свет, значение динамического диапазона составляет от 0 до 2. Чем выше динамический диапазон, тем большее число полутонов присутствует в изображении и тем лучше качество его восприятия.

## Форматы растровых графических файлов

| Формат | Макс. число бит/пиксел | Макс. число цветов | Макс. размер изображения, пиксел | Методы сжатия | Кодирование нескольких изображений |
|---|---|---|---|---|---|
| BMP | 24 | 16 777 216 | 65535 x 65535 | RLE | – |
| GIF | 8 | 256 | 65535 x 65535 | LZW | + |
| JPEG | 24 | 16 777 216 | 65535 x 65535 | JPEG | – |
| PCX | 24 | 16 777 216 | 65535 x 65535 | RLE | – |
| PNG | 48 | 281 474 976 710 656 | 2 147 483 647 x 2 147 483 647 | Deflation (вариант LZ77) | – |
| TIFF | 24 | 16 777 216 | всего 4 294 967 295 | LZW, RLE и другие | + |

## Факторы, влияющие на количество памяти, занимаемой растровым изображением

Файлы растровой графики занимают большое количество памяти компьютера. Некоторые картинки занимают большой объем памяти из–за большого количества пикселов, любой из которых занимает некоторую часть памяти. Наибольшее влияние на количество памяти занимаемой растровым изображением оказывают три факта:

- размер изображения;
- битовая глубина цвета;
- формат файла, используемого для хранения изображения.

## Достоинства:

- Растровая графика эффективно представляет реальные образы.
- Устройства вывода, такие как лазерные принтеры, для создания изображений используют наборы точек. Растровые изображения могут быть очень легко распечатаны на таких принтерах, потому что компьютерам легко управлять устройством вывода для представления отдельных пикселов с помощью точек.

## Недостатки:

Растровые изображения занимают большое количество памяти. Существует так же проблема редактирования растровых изображений, так как большие растровые изображения занимают значительные массивы памяти.