



Белорусский Государственный Университет  
Факультет прикладной математики и информатики

# Компьютерная графика

Василевский Константин

Викторович

17 лекций,

17 лабораторных занятий



# Базовые растровые алгоритмы

- Векторная и растровая графика.
- Постановка задачи *растеризации*.
- Изображение отрезка с целочисленными координатами концов.
- Цифровой дифференциальный анализатор.
- Алгоритм Брезенхема.
- Алгоритм Кастла-Питвея.
- Изображение отрезка с нецелочисленными координатами концов.
- Изображение окружностей. Алгоритм Брезенхема.



# Векторная графика и растровая графика

**Векторная графика** — способ представления объектов и изображений, основанный на использовании геометрических примитивов (таких как точки, линии, сплайны, многоугольники), заданных своим математическим описанием.

**Растровая графика** — способ представления объектов, основанный на использовании сетки пикселей.



# Растровая графика

- Эффективно представляет реальные образы.
- Обеспечивает высокую точность передачи градаций цветов и полутонаов.
- Хорошо подходит для устройств вывода (принтеры, мониторы)
- Занимает большое количество памяти.
- Плохо масштабируется.



# Векторная графика

- Занимает меньшее количество памяти.
- Гибкая масштабируемость.
- Возможность легкого редактирования всех частей векторного изображения;
- Простой экспорт векторного рисунка в растровый.
- Отсутствие реалистичности.
- Невозможность использования эффектов, которые можно применять в растровой графике.
- Практически полная невозможность экспорта растрового рисунка в векторный.



# Введение в растеризацию кривых

Возникающая при выводе задача отображения геометрических объектов, заданных их математическим описанием, на растре, называется **растеризацией**.

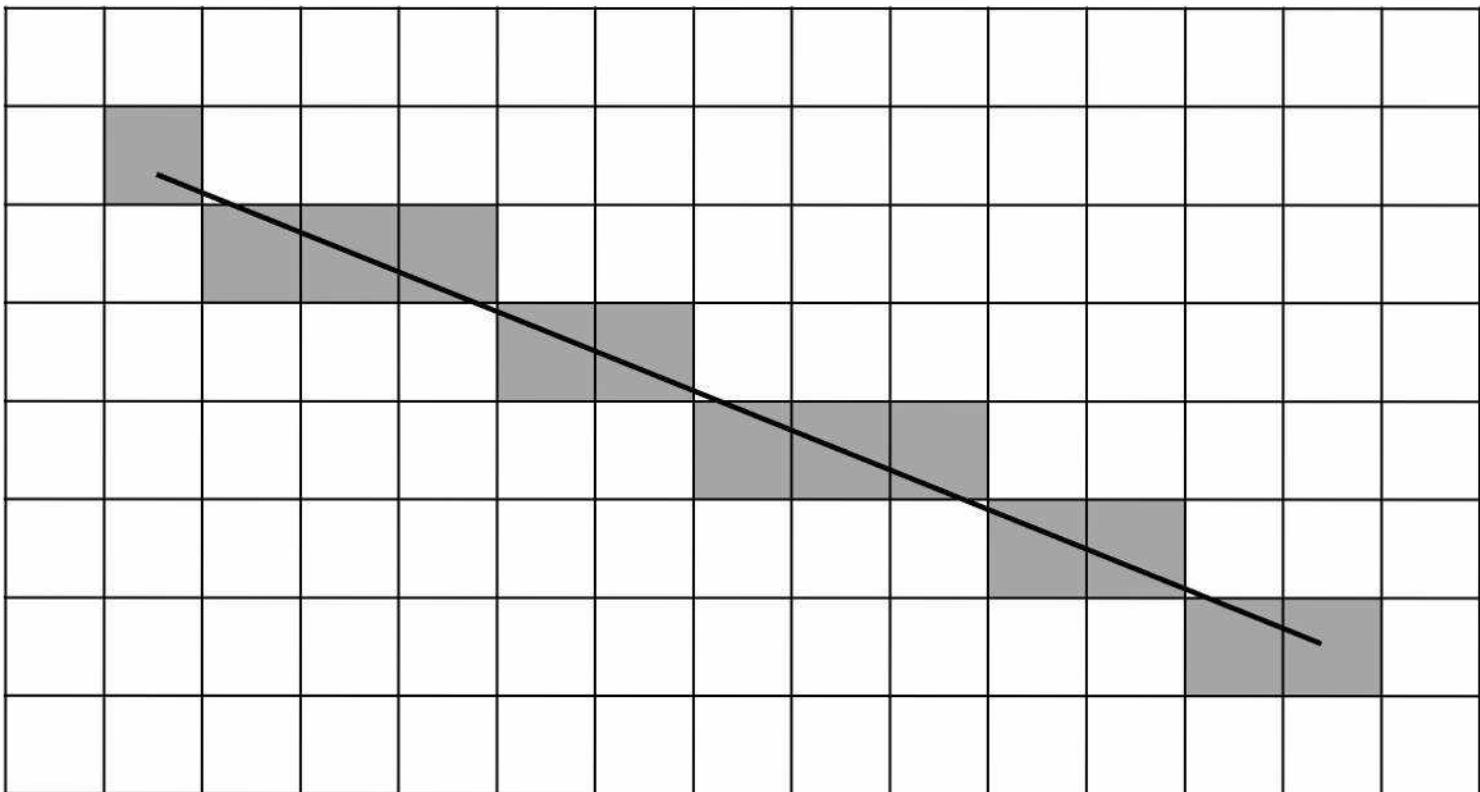
**Растеризация** – процесс определения пикселей, наилучшим образом аппроксимирующих заданный отрезок/кривую.

***Критерии алгоритмов растеризации:***

- качество
- скорость
- простота реализации



# Растеризация





# Понятие линии

В качестве линии на растровой сетке выступает набор пикселей

$$P_1, P_2, \dots, P_n$$

где любые два пикселя  $P_i$  и  $P_{i+1}$  являются соседними.



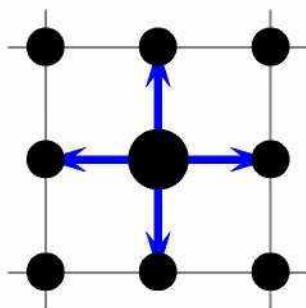
# Понятие связности

Понятие линии базируется на понятии **связности** – возможности соединения двух пикселей растровой линией. При этом возникает вопрос, когда пиксели  $(x_1, y_1)$  и  $(x_2, y_2)$  можно считать соседними. Вводится два понятия связности.



# Понятие связности

**4-связность**, когда пиксели считаются соседними, если либо их x-координаты, либо у-координаты отличаются на единицу

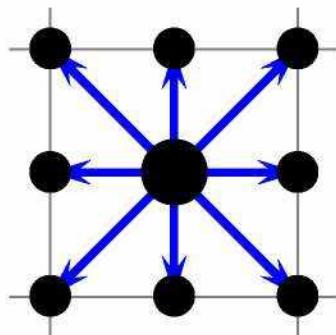


$$|x_1 - x_2| + |y_1 - y_2| \leq 1$$



# Понятие связности

**8-связность**, когда пиксели считаются соседними, если их x- и y-координаты отличаются не более чем на единицу, то есть:



$$|x_1 - x_2| \leq 1, \quad |y_1 - y_2| \leq 1$$

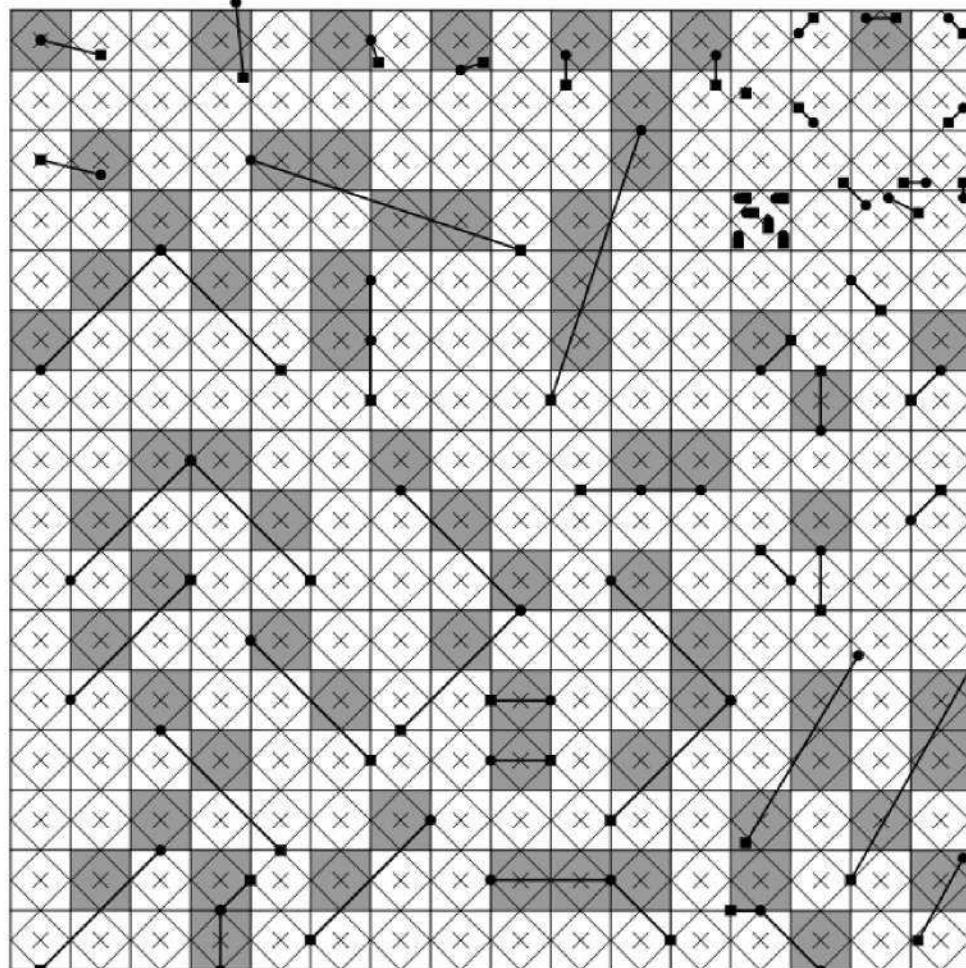


# Понятие связности

Понятие **4-связности** является более сильным, чем **8-связность**: любые два 4-связных пикселя являются и 8-связными, но не наоборот.

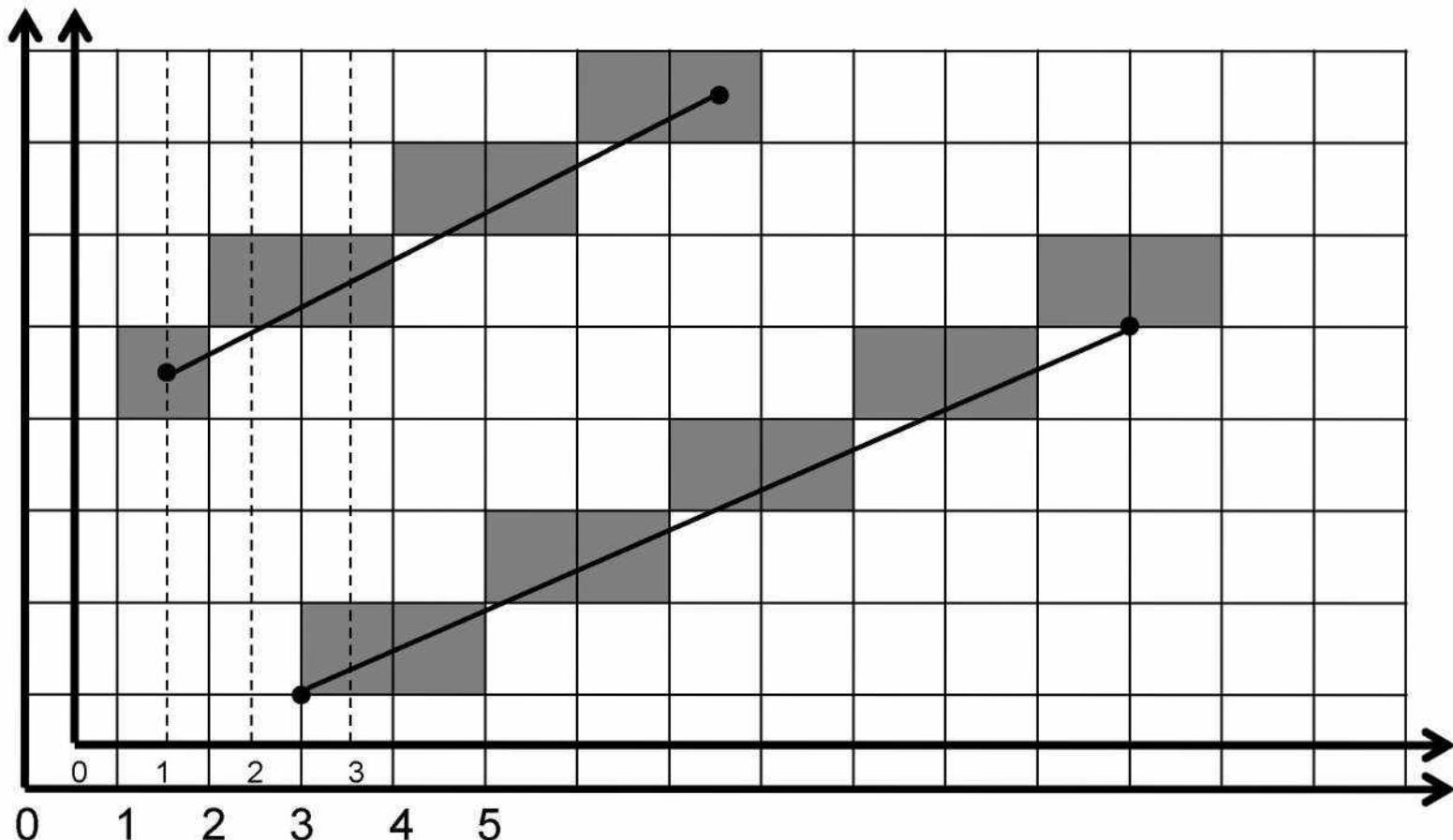


# Изображение отрезка с нечелочисленными координатами концов





# Изображение отрезка с целочисленными координатами концов





# Изображение отрезка с целочисленными координатами концов

- Пошаговый алгоритм
- Алгоритм цифрового дифференциального анализатора
- Алгоритм Брезенхема
- Алгоритм Кастла-Питвея



# Пошаговый алгоритм

- Наиболее очевидный.
- Заключается в постепенном увеличении координаты  $x$ , вычислении координаты  $y=kx+b$  и последующем округлении полученного результата до целого числа.



# Пошаговый алгоритм

Недостатки:

- Работа с вещественными числами.
- Вычисление произведения  $kx$  и последующее округление  
→ затрачивается много времени.



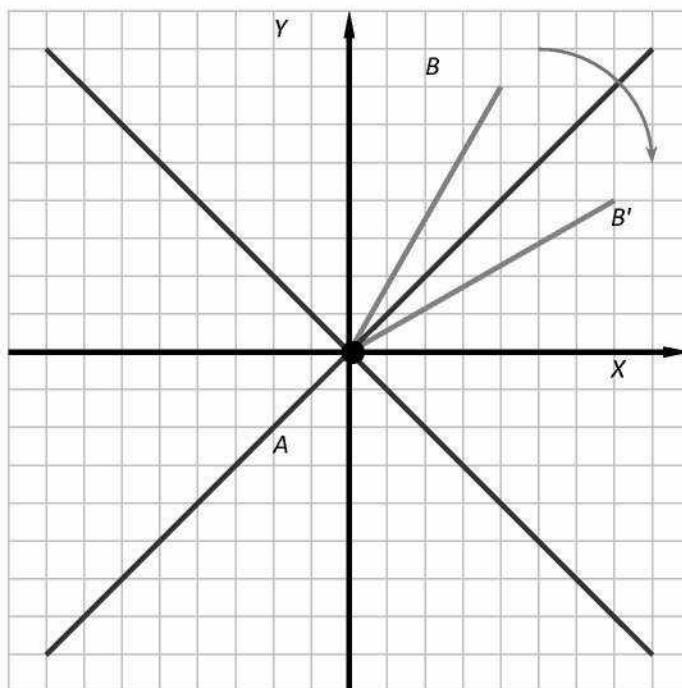
# Алгоритм Брезенхема

- *Bresenham's line algorithm.*
- Строит 8-связную линию.
- Разработан Джеком Е. Брезенхемом в компании IBM в 1962 году.
- Был первоначально разработан для цифровых графопостроителей, однако он в равной степени подходит и для использования растровыми устройствами с ЭЛТ.



# Алгоритм Брезенхема

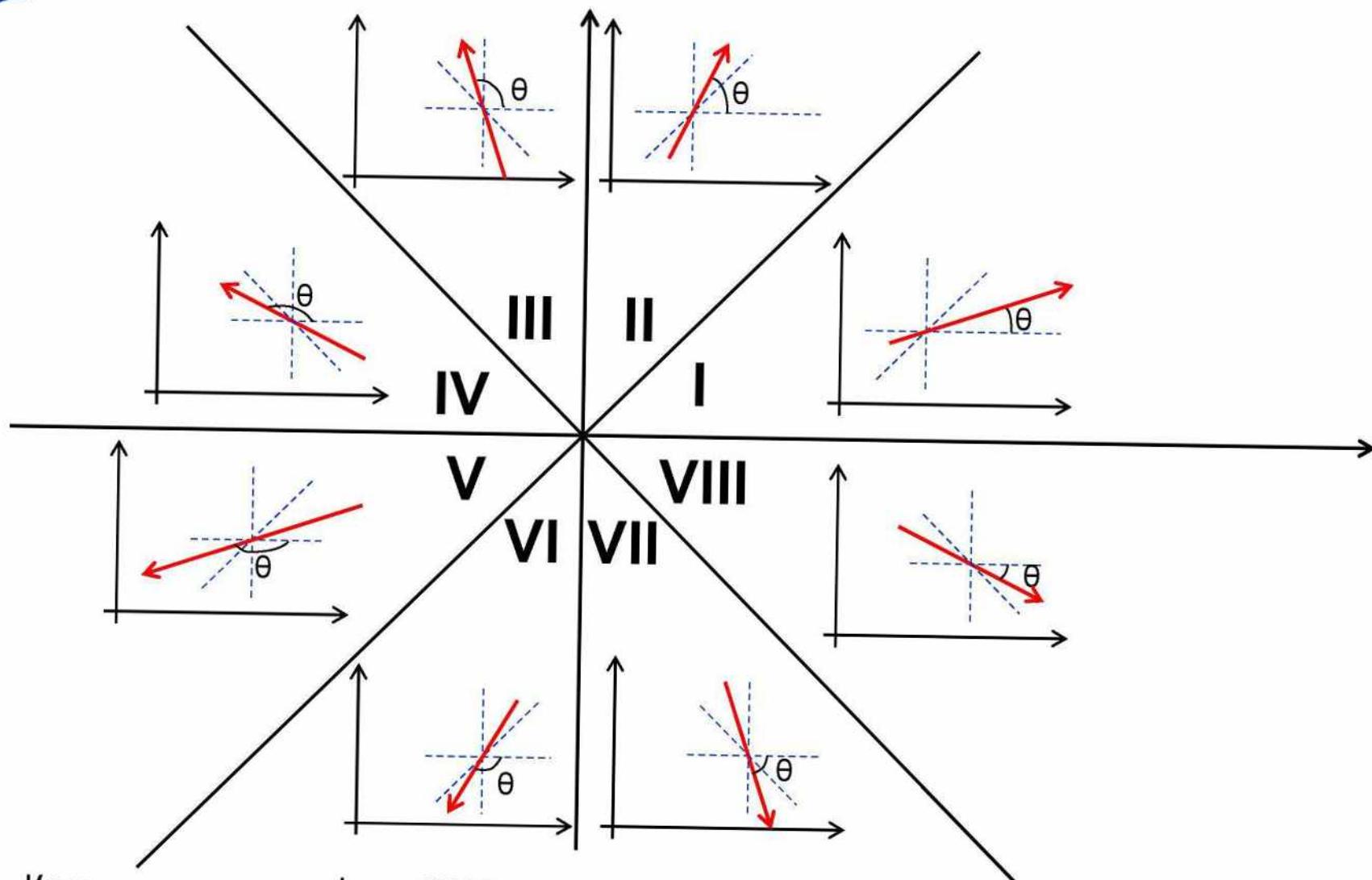
Рассмотрим алгоритм Брезенхема на примере проведения отрезка, лежащего в первом октанте (**канонический случай**).



Отрезок может лежать в любом из 8 октантов, но всегда существуют симметрии относительно осей, разделяющих эти октанты, позволяющие свести задачу к случаю отрезка, лежащего в первом октанте.



# Алгоритм Брезенхема





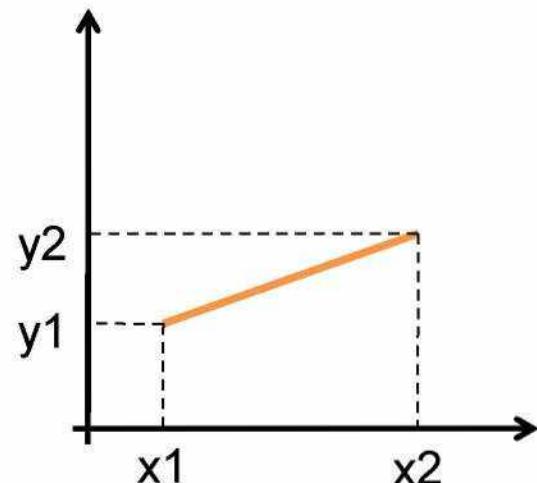
# Алгоритм Брезенхема

Для отрезка, лежащего в первом октанте,  
справедливо следующее:

$$(x_1, y_1), (x_2, y_2)$$

$$x_2 > x_1, \quad y_2 > y_1$$

$$Dx = x_2 - x_1 > Dy = y_2 - y_1$$





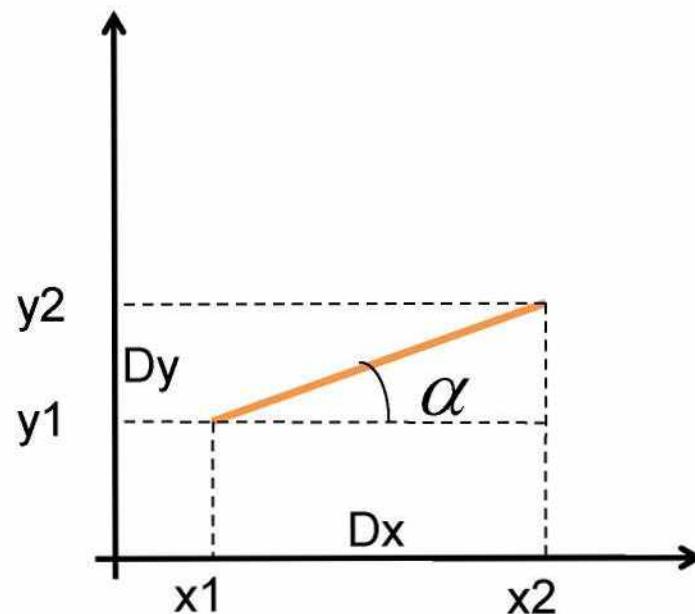
# Алгоритм Брезенхема

$$y = kx + b$$

$$k = \operatorname{tg} \alpha = \frac{Dy}{Dx}$$

$$0 \leq \alpha \leq \frac{\pi}{4}$$

$$0 \leq k \leq 1$$





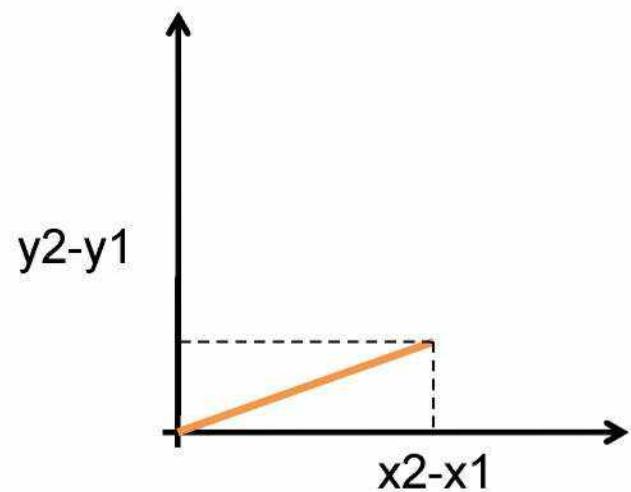
# Алгоритм Брезенхема

Для отрезка, лежащего в первом октанте,  
справедливо следующее:

$$(x_1, y_1), (x_2, y_2)$$

=>

$$(0,0), (x_2 - x_1, y_2 - y_1)$$



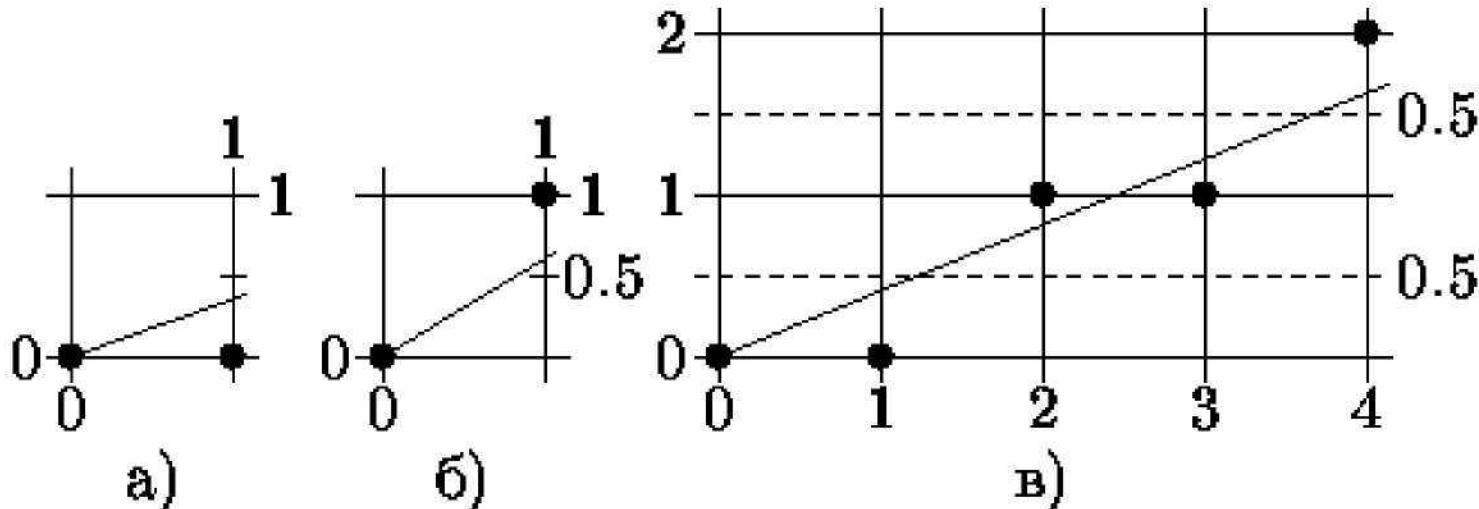
# Алгоритм Брезенхема

- В процессе работы алгоритма координата  $x$  увеличивается на каждом шаге на 1, а координата  $y$  зависит от расстояния между действительным положением отрезка и ближайшими координатами сетки. Такое расстояние называется ошибкой.



# Алгоритм Брезенхема

Если угловой коэффициент прямой  $< 1/2$ , то естественно точку, следующую за точкой  $(0,0)$ , поставить в позицию  $(1,0)$ , а если угловой коэффициент  $\geq 1/2$ , то - в позицию  $(1,1)$ .





# Алгоритм Брезенхема

Первоначально значение ошибки считается равным  $-\frac{1}{2}$ .  
Текущие координаты  $(x,y)=(0,0)$

На следующем шаге к ошибке прибавляется угловой коэффициент и анализируется её значение.

Если ошибка меньше 0, то заполняется ячейка  $(x+1, y)$ , если больше 0, то заполняется ячейка  $(x+1, y+1)$



# Алгоритм Брезенхема

Ошибка для первого шага:

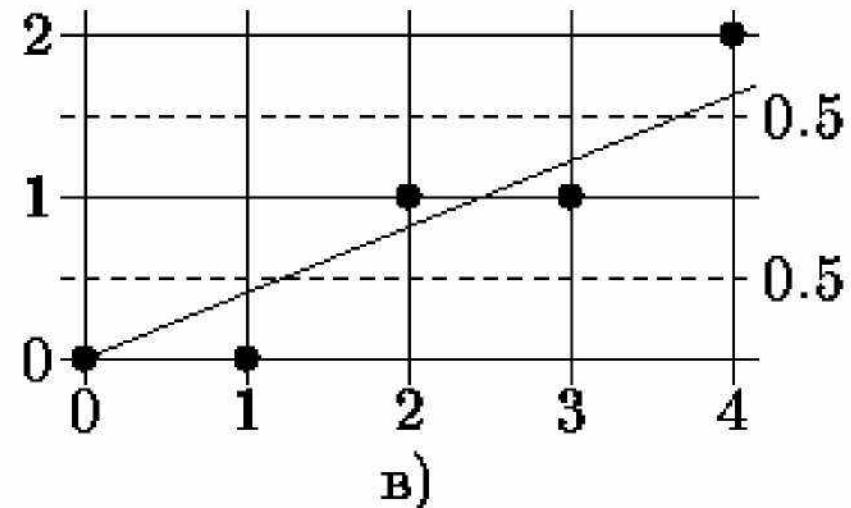
$$E_1 = Dy/Dx - 1/2 < 0$$

→ Для занесения пикселя выбирается точка (1,0)

Ошибка для второго шага вычисляется добавлением приращения Y-координаты для следующей X-позиции

$$E_2 = E_1 + Dy/Dx > 0$$

→ Для занесения пикселя выбирается точка (2,1)





# Алгоритм Брезенхема

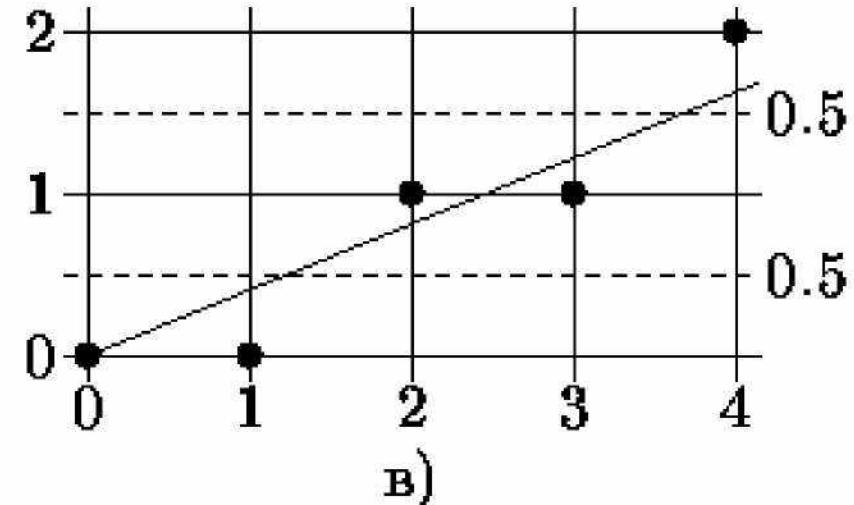
Ошибка для третьего шага:

$$E_3 = E_2 + D_y/D_x$$

Так как отклонение считается от Y-координаты, которая теперь увеличилась на 1, то из накопленного отклонения для вычисления последующих отклонений надо вычесть 1:

$$E_3 = E_2 - 1 < 0$$

→ Для занесения пикселя выбирается точка (3, 1)





# Алгоритм Брезенхема

X= x1;

Y= y1;

Приращение

Dx= x2 - x1;

Dy= y2 - y1;

Инициализация ошибки с  
поправкой на половину  
пикселя

E= Dy/Dx-1/2;

Первая точка вектора

PutPixel(X, Y);

while (X<=X2) {

if (E≥0) {

X= X + 1;

Y= Y + 1;

E= E+Dy/Dx-1;

} else {

X= X + 1;

E= E + Dy/Dx;

}

Очередная точка вектора

PutPixel(X, Y);

}



# Алгоритм Брезенхема

- Работа с вещественными числами
- Есть операция деления
- Разработана целочисленная модификация алгоритма



# Целочисленный алгоритм Брезенхема

Быстродействие алгоритма можно увеличить если использовать только целочисленную арифметику и исключить деление.

Так как важен только знак ошибки, то введем преобразование:

$$E' = E * 2 * D_x$$



# Целочисленный алгоритм Брезенхема

```
X= x1;  
Y= y1;  
Dx= x2 - x1;  
Dy= y2 - y1;  
E'= 2*Dy - Dx;
```

Первая точка вектора

```
PutPixel(X, Y);
```

```
while (X<=X2) {  
    if (E' >=0) {  
        X= X + 1;  
        Y= Y + 1;  
        E'= E' + 2*(Dy - Dx);  
    } else {  
        X= X + 1;  
        E'= E' + 2*Dy;  
    }  
}
```

Очередная точка вектора

```
PutPixel(X, Y);  
}
```



## Целочисленный алгоритм Брезенхема

Таким образом получается алгоритм, в котором используются только целые числа, сложение, вычитание и сдвиг.

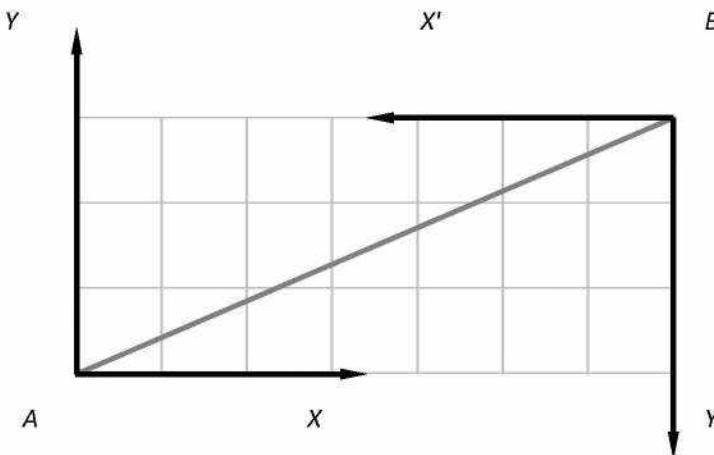
Этот алгоритм пригоден для случая 1го октанта:  $0 < D_y < D_x$ . Для других случаев алгоритм строится аналогичным образом.

```
void BresenhamLine(int x0, int y0, int x1, int y1)
{
    var steep = Math.Abs(y1 - y0) > Math.Abs(x1 - x0); // Проверяем рост отрезка по оси икс и по оси игрек
    // Отражаем линию по диагонали, если угол наклона слишком большой
    if (steep)
    {
        Swap(ref x0, ref y0); // Перетасовка координат вынесена в отдельную функцию для красоты
        Swap(ref x1, ref y1);
    }
    // Если линия растёт не слева направо, то меняем начало и конец отрезка местами
    if (x0 > x1)
    {
        Swap(ref x0, ref x1);
        Swap(ref y0, ref y1);
    }
    int dx = x1 - x0;
    int dy = Math.Abs(y1 - y0);
    int error = dx / 2; // Здесь используется оптимизация с умножением на dx, чтобы избавиться от лишних дробей
    int ystep = (y0 < y1) ? 1 : -1; // Выбираем направление роста координаты у
    int y = y0;
    for (int x = x0; x <= x1; x++)
    {
        DrawPoint(steep ? y : x, steep ? x : y); // Не забываем вернуть координаты на место
        error -= dy;
        if (error < 0)
        {
            y += ystep;
            error += dx;
        }
    }
}
```



# Оптимизация алгоритма Брезенхема

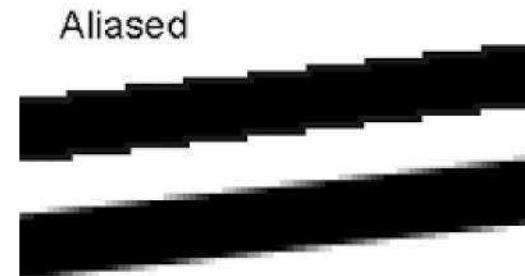
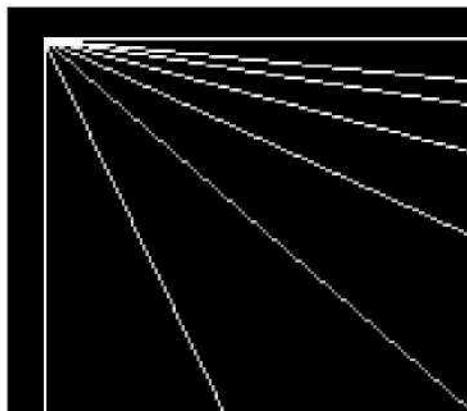
Оптимизация: отрезок симметричен относительно прямой, проходящей перпендикулярно ему через его середину => можно начинать рисовать сразу с двух концов, что сократит время общей работы алгоритма вдвое.





# Рисование сглаженных линий

Сглаживание (anti-aliasing) — технология, используемая для устранения эффекта «зубчатости», возникающего на краях одновременно выводимого на экран множества отдельных друг от друга плоских или объёмных изображений.



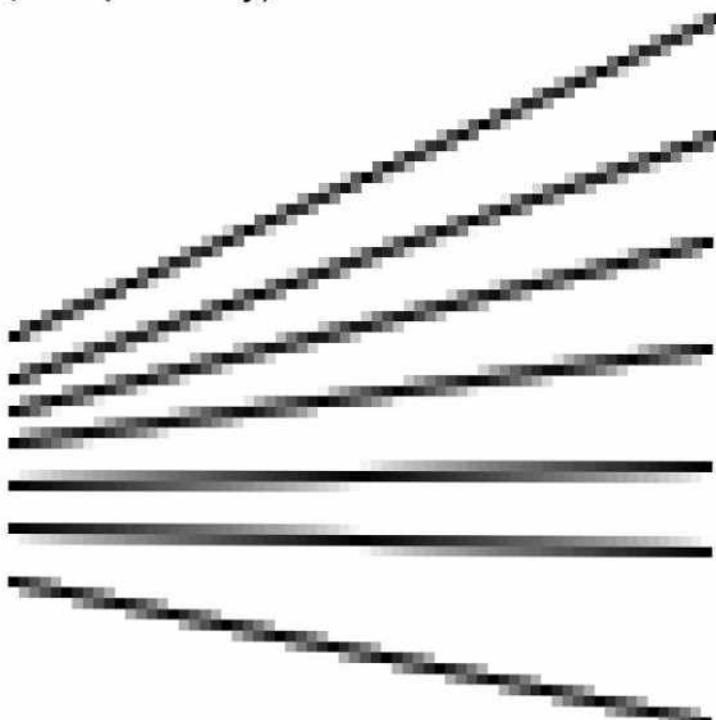
Как реализуется: пиксели, соседние с граничным пикселям изображения, принимают промежуточное значение между цветом изображения и цветом фона, создавая градиент и размывая границу.



# Рисование сглаженных линий

Два подхода к сглаживанию:

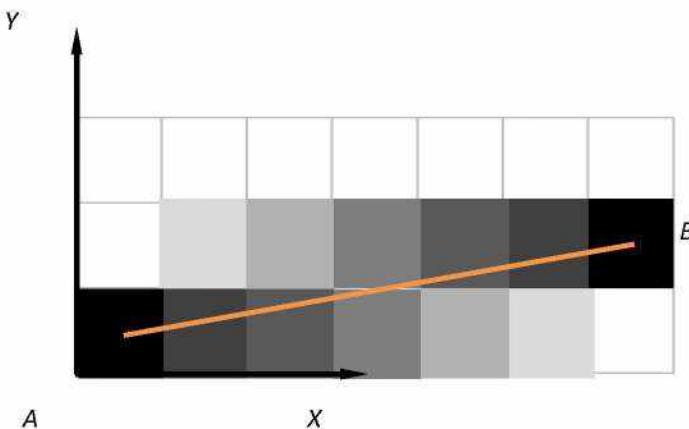
- Растеризация уже сглаженных простейших объектов – линий и кривых (алгоритм Ву)
- Сглаживание уже отрисованного изображения (фильтрация)





# Алгоритм Ву для рисования сглаженных линий (Xiaolin Wu, 1991)

На каждом шаге ведётся расчёт для двух ближайших к прямой пикселей, и они закрашиваются с разной интенсивностью, в зависимости от удалённости. 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 Ву для рисования сглаженных линий

На каждом шаге ведётся расчёт для двух ближайших к прямой пикселей, и они закрашиваются с разной интенсивностью, в зависимости от удалённости. 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.





# Цифровой дифференциальный анализатор (ЦДА)

- DDA - Digital *Differential Analyzer*.
- Строит 8-связную линию.
- Основан на определении дифференциала.
- Формирует дискретную аппроксимацию непрерывного решения дифференциального уравнения отрезка
- В настоящее время практически не применяется.



# Цифровой дифференциальный анализатор (ЦДА)

$(x_1, y_1), (x_2, y_2)$  - конечные точки отрезка

$$\frac{dy}{dx} = \text{const} \Leftrightarrow \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

$$\Delta y = y_i - y_{i-1}$$

$$y_i = y_{i-1} + \Delta y = y_{i-1} + \frac{y_2 - y_1}{x_2 - x_1} \Delta x = y_{i-1} + \frac{Dy}{Dx}$$



# Цифровой дифференциальный анализатор (ЦДА)

Несимметричный алгоритм ЦДА  
(I октант,  $Dy < Dx$ )

Вычислить приращения координат:

$$Dx = x_2 - x_1;$$

$$Dy = y_2 - y_1;$$

Занести начальную точку отрезка

PutPixel ( $x_1, y_1$ );

Сгенерировать отрезок

```
while ( $x_1 < x_2$ ) {  
     $x_1 := x_1 + 1.0$ ;  
     $y_1 := y_1 + Dy/Dx$ ;  
    PutPixel ( $x_1, y_1$ );  
}
```



# Цифровой дифференциальный анализатор (ЦДА)

Несимметричный алгоритм ЦДА ( $Dy < Dx$ )

Вопрос вычисления целочисленной координаты  $u$  на каждой итерации

- округление;
- отбрасывание дробной части;



# Цифровой дифференциальный анализатор (ЦДА)

## Несимметричный алгоритм ЦДА (II октант, $Dx < Dy$ )

Вычислить приращения координат:

$$Dx = x_2 - x_1;$$

$$Dy = y_2 - y_1;$$

Занести начальную точку отрезка

PutPixel ( $x_1, y_1$ );

Сгенерировать отрезок

```
while ( $y_1 < y_2$ ) {  
     $y_1 := y_1 + 1.0$ ;  
     $x_1 := x_1 + Dx/Dy$ ;  
    PutPixel ( $x_1, y_1$ );  
}
```



# Цифровой дифференциальный анализатор (ЦДА)

Симметричный алгоритм ЦДА  
(I четверть,  $x_1 < x_2, y_1 < y_2$ )

Вычислить максимум из приращений координат:

$$Dx = x_2 - x_1; \quad Dy = y_2 - y_1;$$

$$L = \max \{Dx, Dy\};$$

Занести начальную точку отрезка

PutPixel ( $x_1, y_1$ );

Сгенерировать отрезок

$i = 0;$

while ( $i \leq L$ ) {

$y_1 := y_1 + Dy/L;$

$x_1 := x_1 + Dx/L;$

PutPixel ( $x_1, y_1$ );  $++i$ ;

}



# Алгоритм Кастла-Питвея

- Строит 8-связную линию.
- Гораздо менее эффективен с вычислительной точки зрения, чем алгоритм Брезенхема.
- Обладает красивой математической структурой и представляет интерес с чисто алгоритмической точки зрения.
- Основан на идее, схожей с известным алгоритмом Евклида нахождения НОД двух натуральных чисел.



# Алгоритм Кастла-Питвея

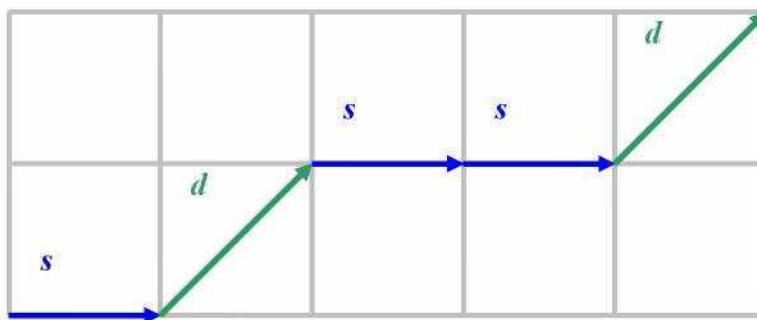
В каноническом случае процесс рисования 8-связной линии можно закодировать последовательностью

вида: **sdssd...**

Где:

s - горизонтальное смещение;

d - диагональное смещение.





# Алгоритм Кастла-Питвея

Рисуем отрезок  $(0,0)-(a,b)$

```
y = b;  
x = a - b;  
m1 = "s"; m2 = "d";  
while( x <>y ) {  
    if( x > y ) {  
        x = x - y;  
        m2 = m1 (+) ~ m2;  
    } else {  
        y = y - x;  
        m1 = m2 (+) ~ m1;  
    }  
}
```

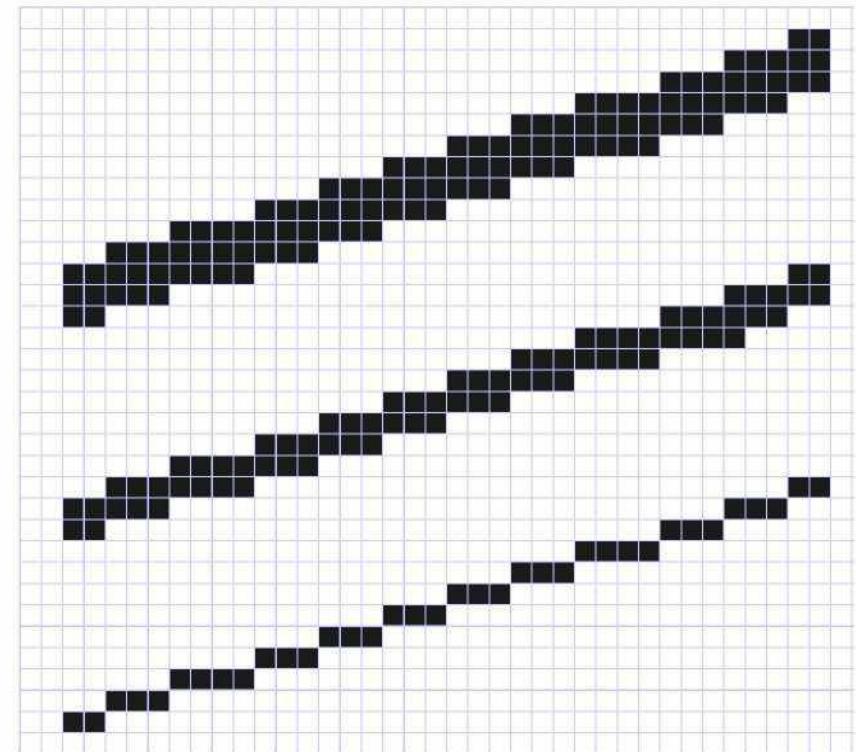
После завершения работы алгоритма **x-кратная**  
последовательность **m2 (+) ~ m1** задает нужный отрезок.



# Изображение отрезков толщиной в более чем один пиксель

Алгоритм:  
использование  
фигуры (например,  
квадрата) или линии в  
качестве элемента  
вывода.

Пример: линия  
толщины 3:  
Закрашивается  
очередной пиксель  
 $(x_i, y_i)$ , а также пиксели  
 $(x_i, y_i+1)$  и  $(x_i, y_i+2)$ .





# Изображение отрезков с нецелочисленными координатами концов

1 вариант:

Округление координат до целочисленных.

Недостатки: могут возникнуть искажения (особенно в случае отрезков небольшой длины).

2 вариант

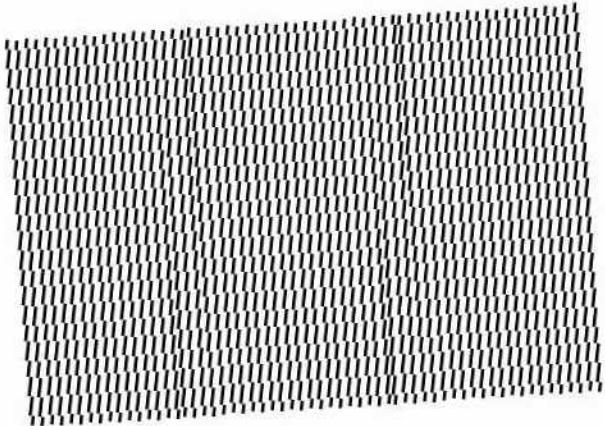
Алгоритм растеризации с субпиксельной точностью  
Величина ошибки вычисляется с поправкой на  
дробную часть координат.



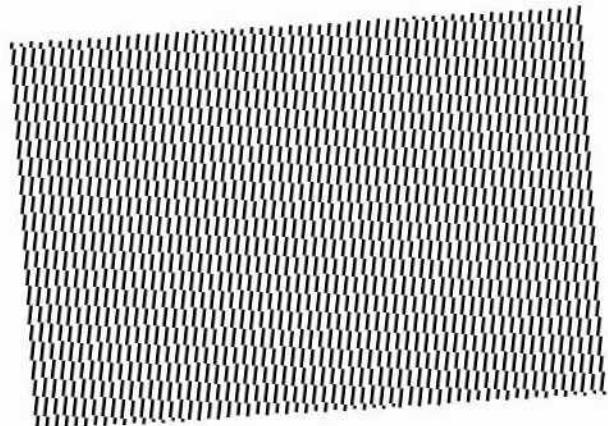
# Изображение отрезков с нецелочисленными координатами концов

Алгоритмы растеризации

а)



б)





# Рисование окружностей и эллипсов

Во многих областях базовыми графическими примитивами, помимо отрезков прямых, являются и конические сечения, т.е. окружности, эллипсы, параболы и гиперболы. Наиболее употребительным примитивом является окружность. Один из наиболее простых и эффективных алгоритмов генерации окружности также разработан Брезенхемом.



# Рисование окружностей и эллипсов

Уравнение окружности радиуса  $R$  с центром в точке  $(x_0, y_0)$

$$(x - x_0)^2 + (y - y_0)^2 = R^2$$

Пошаговое вычисление координат

$$y_i = y_0 + \sqrt{R^2 - (x_i - x_0)^2}$$

$$x_i = x_0 - R, \dots, x_0 + R$$



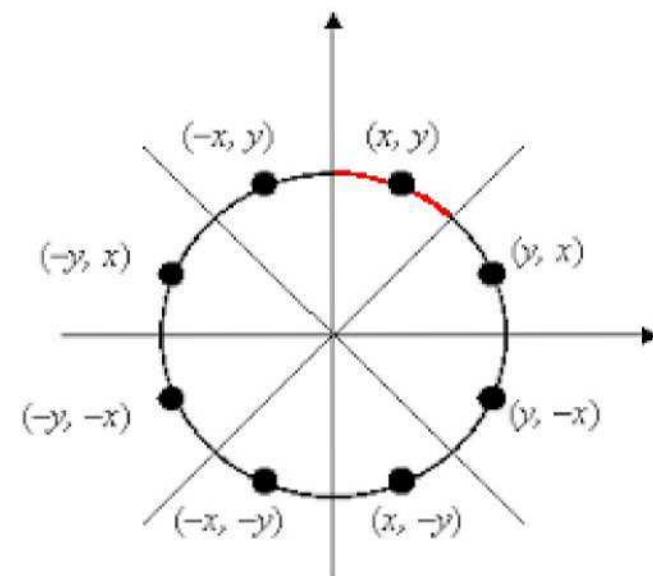
# Алгоритм Брезенхема

для построения окружности

Рассмотрим работу алгоритма на примере построения окружности с центром в начале координат.

Достаточно построить одну дугу окружности, расположенную во втором оваланте. Координаты точек окружности для остальных овалантов получаются симметрично.

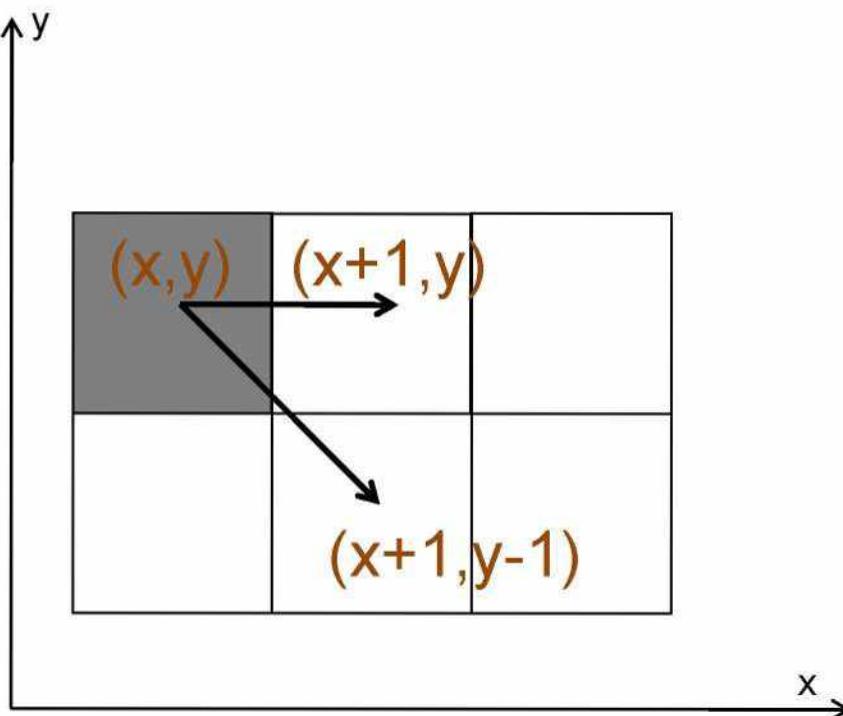
На каждом шаге алгоритма в качестве ошибки выбирается расстояние от окружности до ближайших координат сетки.





# Алгоритм Брезенхема

для построения окружности



Начинаем из точки:  $(0; R)$

Завершаем в точке:  $\left(\frac{R\sqrt{2}}{2}; \frac{R\sqrt{2}}{2}\right)$

На каждой итерации из текущих координат  $(x, y)$  возможны два смещения – в точку  $(x+1, y)$  либо в точку  $(x+1, y-1)$ .

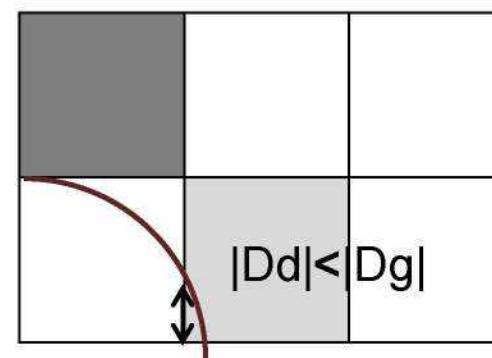
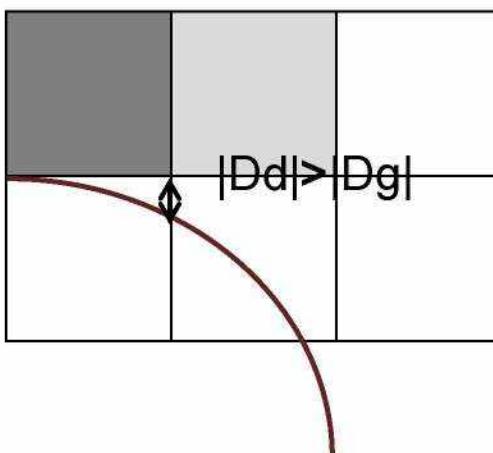


# Алгоритм Брезенхема

для построения окружности

$$Dg = (x+1)^2 + y^2 - R^2$$

$$Dd = (x+1)^2 + (y-1)^2 - R^2$$





# Целочисленный алгоритм Брезенхема для дуги окружности

```
X= 0;  
Y= R;  
E= 3-2*R;
```

Первая точка вектора

```
PutPixel(X, Y);
```

```
while (X<Y) {  
    if (E ≥0) {  
        E= E + 4*(X-Y)+10;  
        X= X+ 1;  
        Y= Y - 1;  
    } else {  
        E= E + 4*X+6;  
        X= X + 1;  
    }  
}
```

Очередная точка вектора

```
PutPixel(X, Y);  
}
```



# Целочисленный алгоритм Брезенхема для целой окружности

```
PutPixel(X, Y);    // отрисовали точку дуги II октанта
PutPixel(X, -Y);   // симметричная точка VII октанта
PutPixel(-X, Y);   // симметричная точка III октанта
PutPixel(-X, -Y);  // симметричная точка V октанта

PutPixel(Y, X);    // симметричная точка I октанта
PutPixel(Y, -X);   // симметричная точка VIII октанта
PutPixel(-Y, X);   // симметричная точка IV октанта
PutPixel(-Y, -X);  // симметричная точка VI октанта
```



## Целочисленный алгоритм Брезенхема для окружности с центром в произвольной точке (X0,Y0)

```
PutPixel(X + X0, Y + Y0);  
PutPixel(X + X0, -Y + Y0);  
PutPixel(-X + X0, Y + Y0);  
PutPixel(-X + X0, -Y + Y0);  
  
PutPixel(Y + X0, X + Y0);  
PutPixel(Y + X0, -X + Y0);  
PutPixel(-Y + X0, X + Y0);  
PutPixel(-Y + X0, -X + Y0);
```



Белорусский Государственный Университет  
Факультет прикладной математики и информатики

# Компьютерная графика

Василевский Константин

Викторович

17 лекций,

17 лабораторных занятий



# Основные алгоритмы вычислительной геометрии. Геометрические задачи визуализации.

- Геометрические преобразования на плоскости
  - способы задания отрезков
  - классификация точки относительно отрезка
  - расстояние от точки до прямой
  - нахождение точки пересечения двух отрезков
  - уравнение нормали
- Отсечение отрезков
- Алгоритм Сазерленда-Коэна
- Алгоритм средней точки
- Алгоритм Лианга-Барски
- Алгоритм Кируса-Бека
- Отсечение многоугольников



# Способы задания отрезка

Отрезок ( $P_1, P_2$ )

$$P_1=(x_1, y_1) \quad P_2=(x_2, y_2)$$

1.  $Ax+By+C=0$  – уравнение прямой на плоскости

$(A, B)$  – вектор нормали

2.  $y=kx+b$  – уравнение прямой на плоскости

$k = (y_2 - y_1) / (x_2 - x_1)$  – угловой коэффициент

3. Параметрическое задание отрезка:

$$P=P+t(P_2-P_1), \quad 0 \leq t \leq 1$$

$$x=x_1+t(x_2-x_1), \quad 0 \leq t \leq 1$$

$$y=y_1+t(y_2-y_1), \quad 0 \leq t \leq 1$$



# Скалярное произведение векторов

Скалярным произведением двух векторов называется число, равное произведению модулей этих векторов на косинус угла между ними.

$$\begin{array}{ll} P_1 = (x_1, y_1) & P_2 = (x_2, y_2) \\ P_3 = (x_3, y_3) & P_4 = (x_4, y_4) \end{array}$$

$$P_1P_2 \cdot P_3P_4 = |P_1P_2| |P_3P_4| \cos(\varphi)$$

$$P_1P_2 \cdot P_3P_4 = (x_2 - x_1)(x_4 - x_3) + (y_2 - y_1)(y_4 - y_3)$$



# Классификация точки относительно отрезка

Одна из важнейших операций заключается в определении положения точки относительно направленного отрезка прямой линии. Результат выполнения операции показывает, лежит ли точка слева или справа от направленного отрезка прямой линии или, если она совпадает с прямой линией, располагается она после конца направленного отрезка этой прямой линии, до начала, а если ни то, ни другое, то не совпадает ли она с началом или с концом или лежит между ними. Направленный отрезок прямой линии четко разделяет плоскость на семь непересекающихся областей.



# Классификация точки относительно отрезка

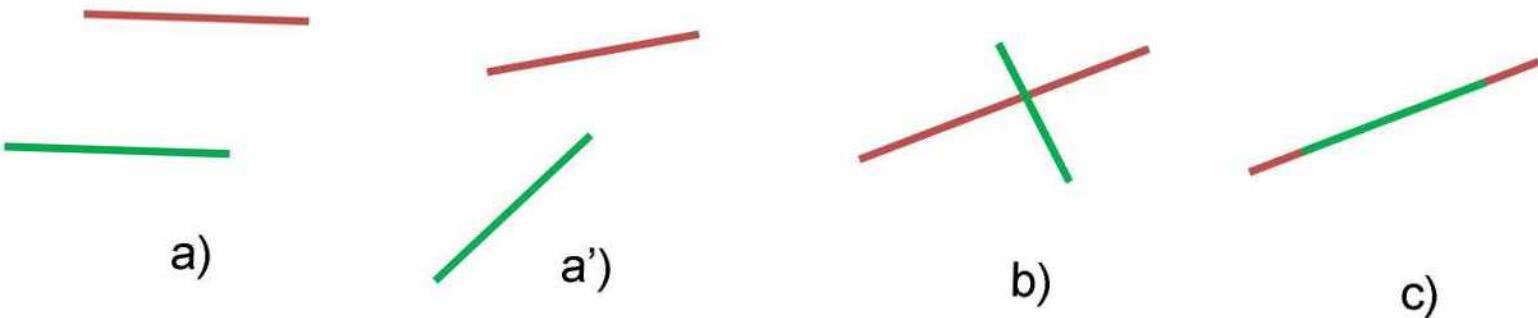




# Нахождение точки пересечения двух отрезков

Пересечение отрезков:

- a) Пустое (не пересекаются)
- b) Одна точка (пересекаются)
- c) Отрезок (накладываются друг на друга)





# Нахождение точки пересечения двух отрезков

1. Точка пересечения двух прямых, заданных в общем виде

$$\begin{cases} A_1x+B_1y+C_1=0 \\ A_2x+B_2y+C_2=0 \end{cases}$$

2. Точка пересечения прямой и отрезка, заданного параметрически.

$A_1x+B_1y+C_1=0$  - прямая

$P_1=(x_1, y_1)$   $P_2=(x_2, y_2)$  - отрезок

$x=x_1+t(x_2-x_1)$ ,  $0 \leq t \leq 1$

$y=y_1+t(y_2-y_1)$ ,  $0 \leq t \leq 1$



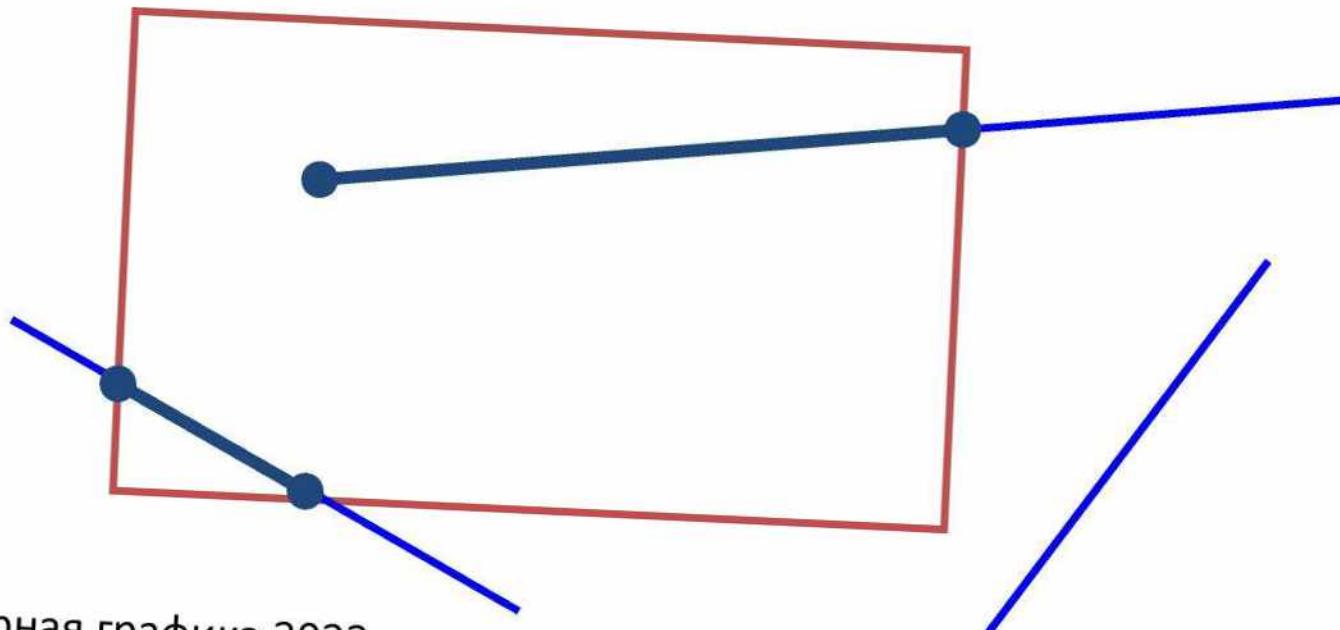
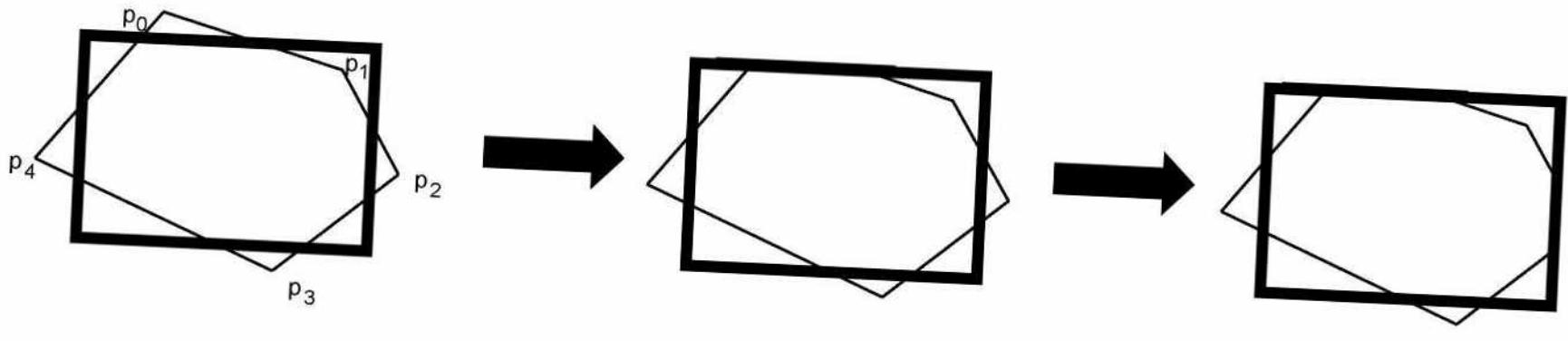
# Отсечение

**Отсечение (clipping)** – это процедура обнаружения и исключения из дальнейшего анализа элементов сцены (линий, поверхностей объектов), которые не попадают в видимую область, отображаемую на экране.

Это операция выполняется одной из первых в процессе визуализации.

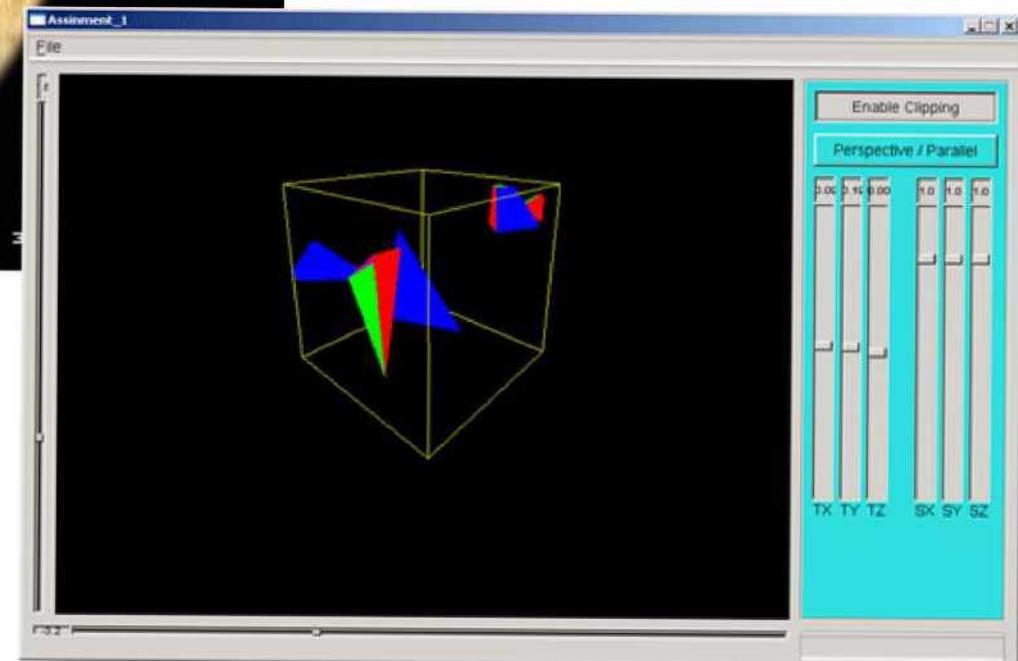
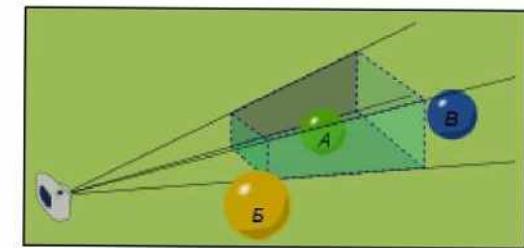


# Отсечение





# Отсечение





# Отсечение

## Преимущества отсечения непосредственно перед растеризацией:

Можно было бы просто проверять, лежит ли пиксель внутри растра непосредственно перед изменением его цвета. К сожалению, такое решение в большинстве случаев слишком неэффективно. Во-первых, подобная проверка сама по себе чрезвычайно замедлит растеризацию. Во-вторых, в тех случаях, когда значительная часть объекта лежит вне растра, растеризация для этой области будет выполняться вхолостую и отсечение подобных областей опять-таки ускорит вычисления.



# Методы отсечения

- 2D и 3D.
- Отсечение отрезков, многоугольников, сложных объектов.
- Реализованные аппаратно и программно.



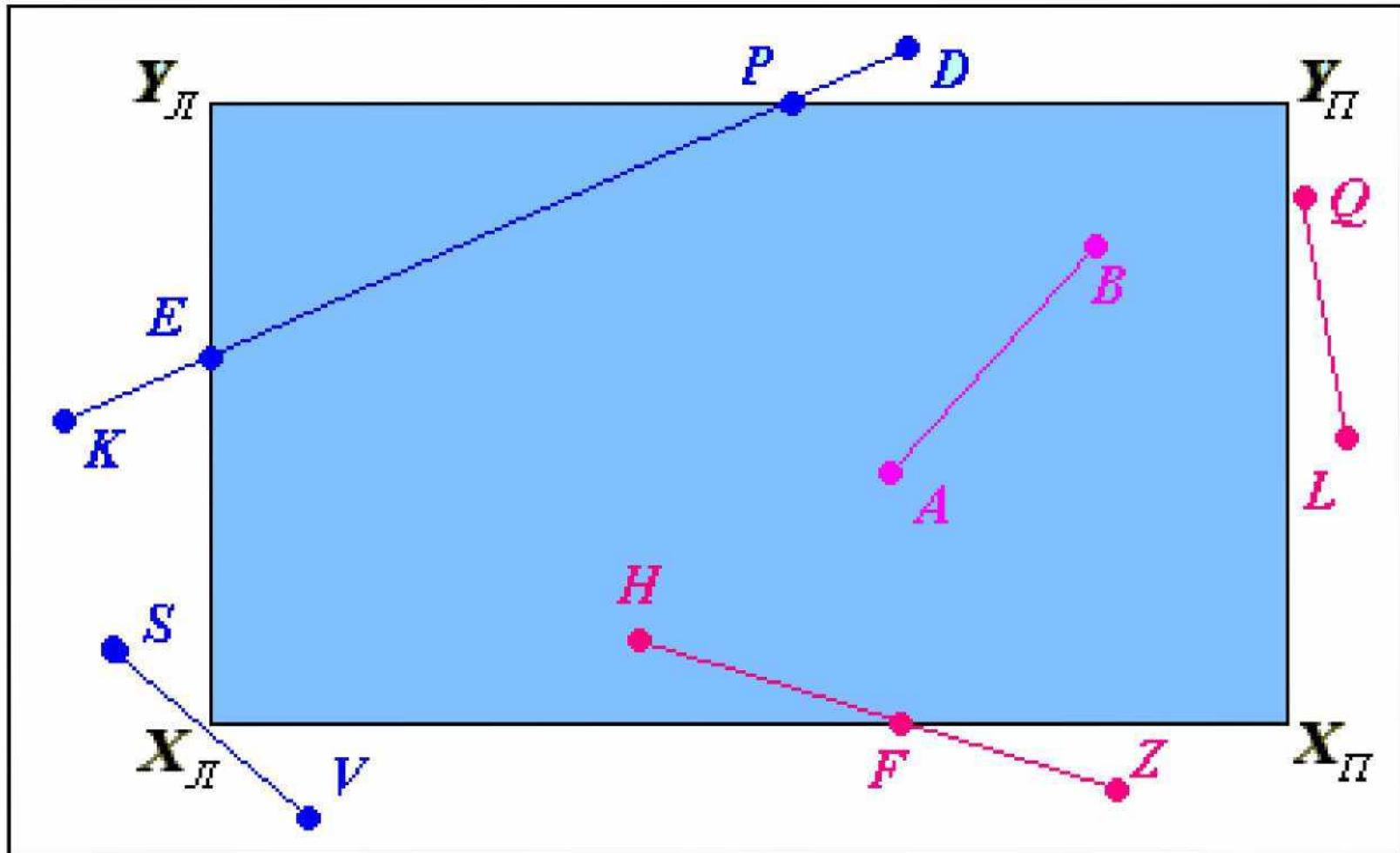
# Отсечение отрезков. Алгоритм Сазерленда-Коэна

Дэн Коэн, Айвен Сазерленд,  
1966-1968.

Алгоритм Сазерленда-Коэна  
осуществляет двумерное  
отсечение отрезка относительно  
прямоугольника со сторонами,  
параллельными координатным  
осям.



# Отсечение отрезков. Алгоритм Сазерленда-Коэна





# Отсечение отрезков. Алгоритм Сазерленда-Коэна

$(x_{\min}, y_{\min}), (x_{\max}, y_{\max})$  - координаты нижнего левого и верхнего правого углов отсекающего прямоугольника.

Точки, лежащие внутри окна:

$$x_{\min} \leq x \leq x_{\max}$$

$$y_{\min} \leq y \leq y_{\max}$$

Если неравенства не выполняются => точка отсекается.

\* Отсечение отдельных точек применяется довольно редко.



# Отсечение отрезков. Алгоритм Сазерленда-Коэна

Плоскость экрана делится на 9 областей, каждой из которых приписывается четырехбитный код – *характеристический код (outcode)* –  $(b_0 b_1 b_2 b_3)$

$$b_0 = \begin{cases} 1, & y > y_{\max} \\ 0, & \text{otherwise} \end{cases}$$

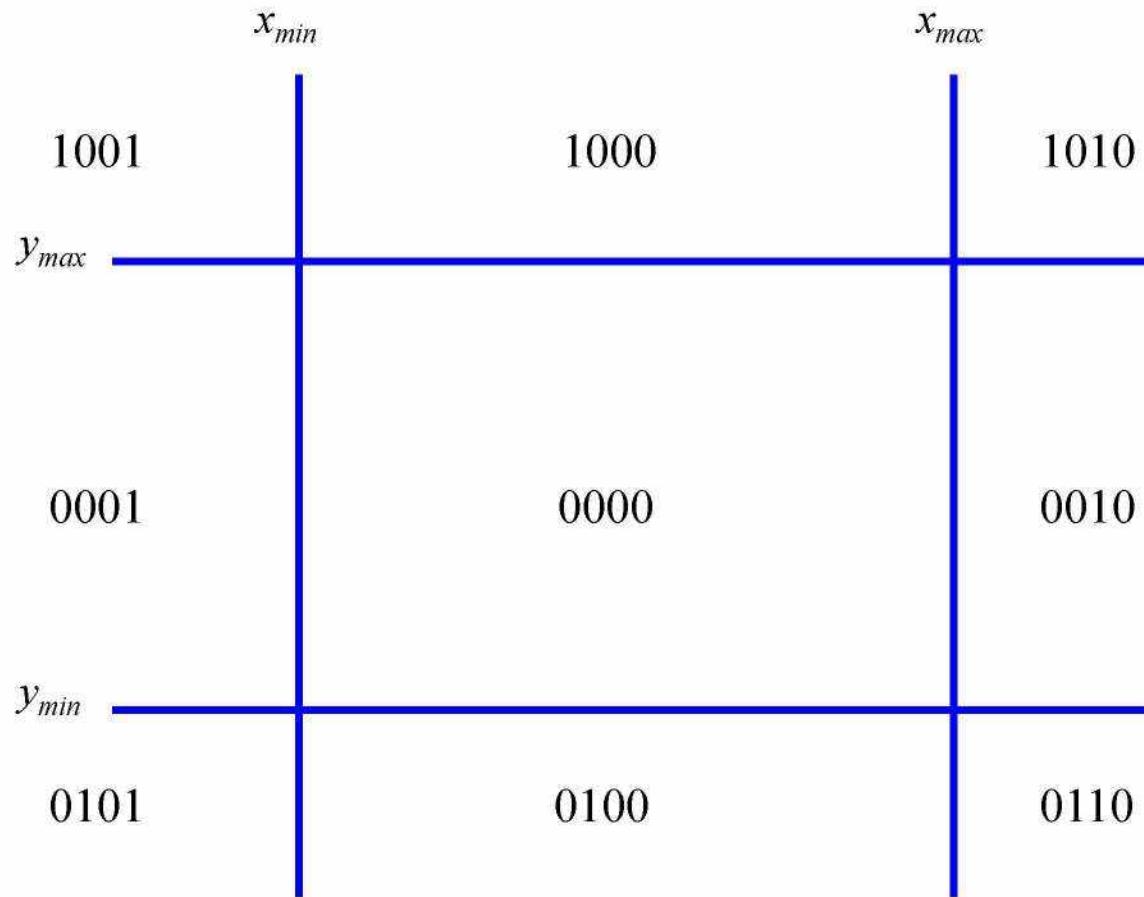
$$b_1 = \begin{cases} 1, & y < y_{\min} \\ 0, & \text{otherwise} \end{cases}$$

$$b_2 = \begin{cases} 1, & x > x_{\max} \\ 0, & \text{otherwise} \end{cases}$$

$$b_3 = \begin{cases} 1, & x < x_{\min} \\ 0, & \text{otherwise} \end{cases}$$



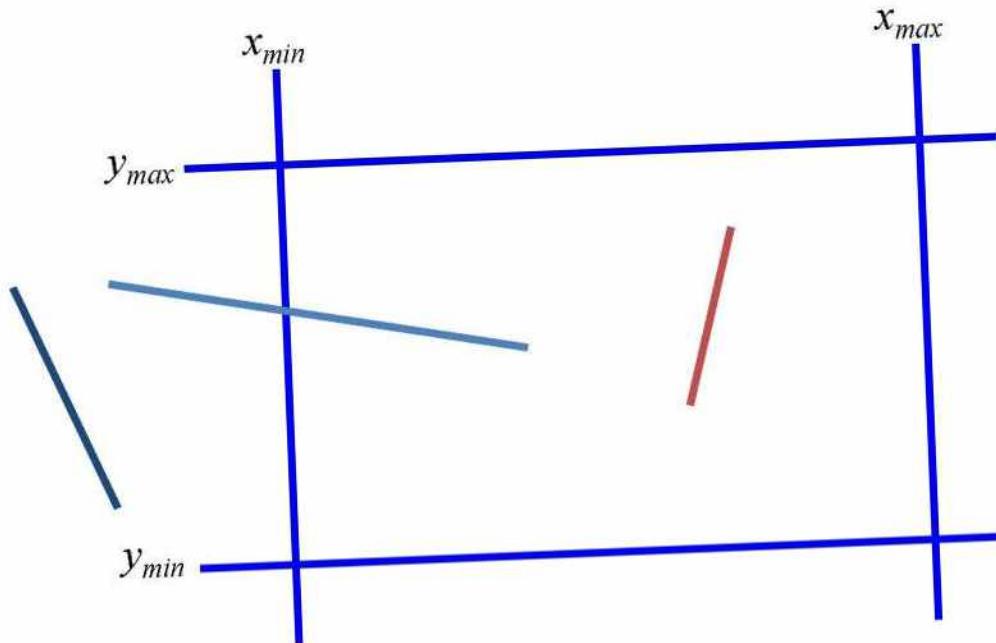
# Отсечение отрезков. Алгоритм Сазерленда-Коэна





# Отсечение отрезков. Алгоритм Сазерленда-Коэна

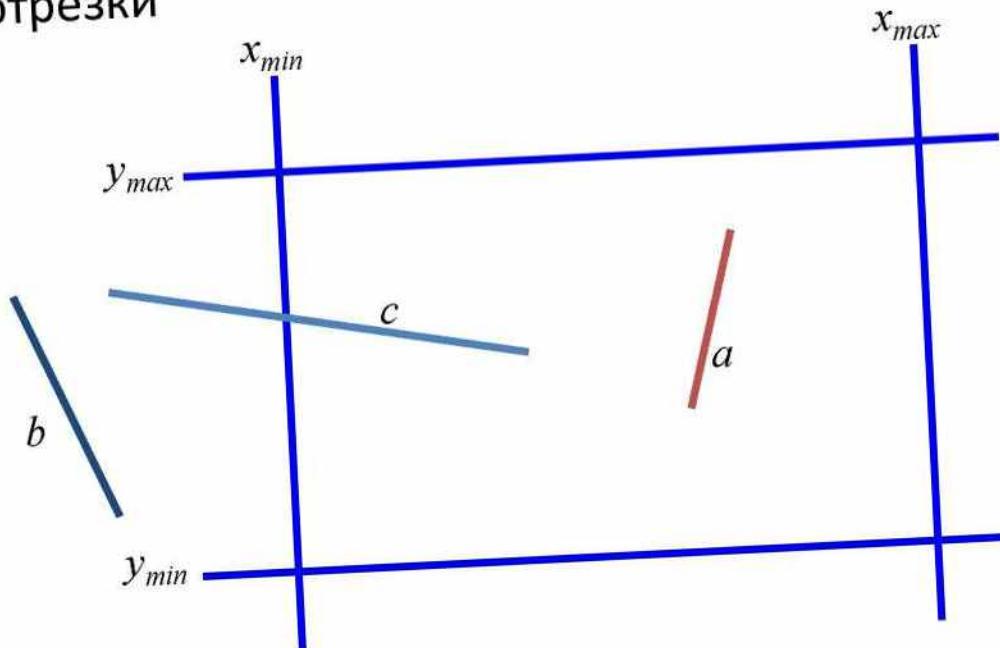
Пусть  $O_1$  и  $O_2$  – характеристические коды конечных точек рассматриваемого отрезка





# Отсечение отрезков. Алгоритм Сазерленда-Коэна

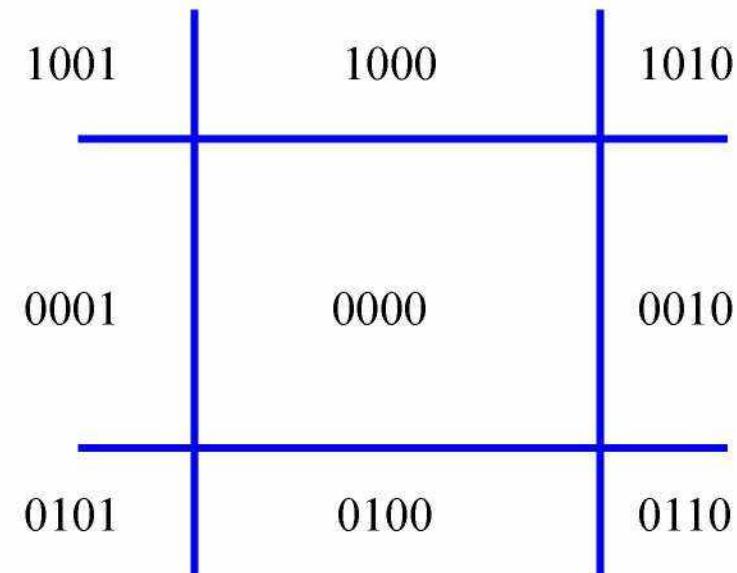
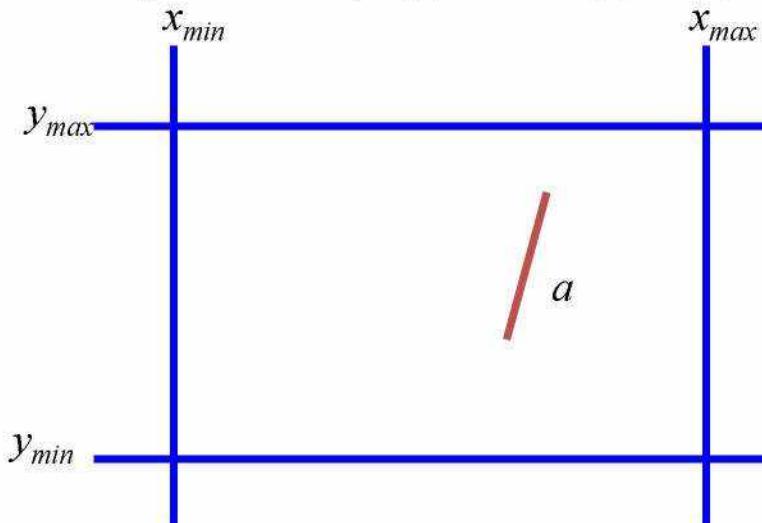
Рассмотрим тривиальные случаи и первым делом отбросим полностью видимые (такие как  $a$ ) и полностью невидимые (такие как  $b$ ) отрезки





# Отсечение отрезков. Алгоритм Сазерленда-Коэна

1. Отрезок лежит целиком внутри отсекающего окна (будет полностью видимым) :  $O_1 = O_2 = 0000$ .  
Отрезок передается для дальнейшей растеризации.

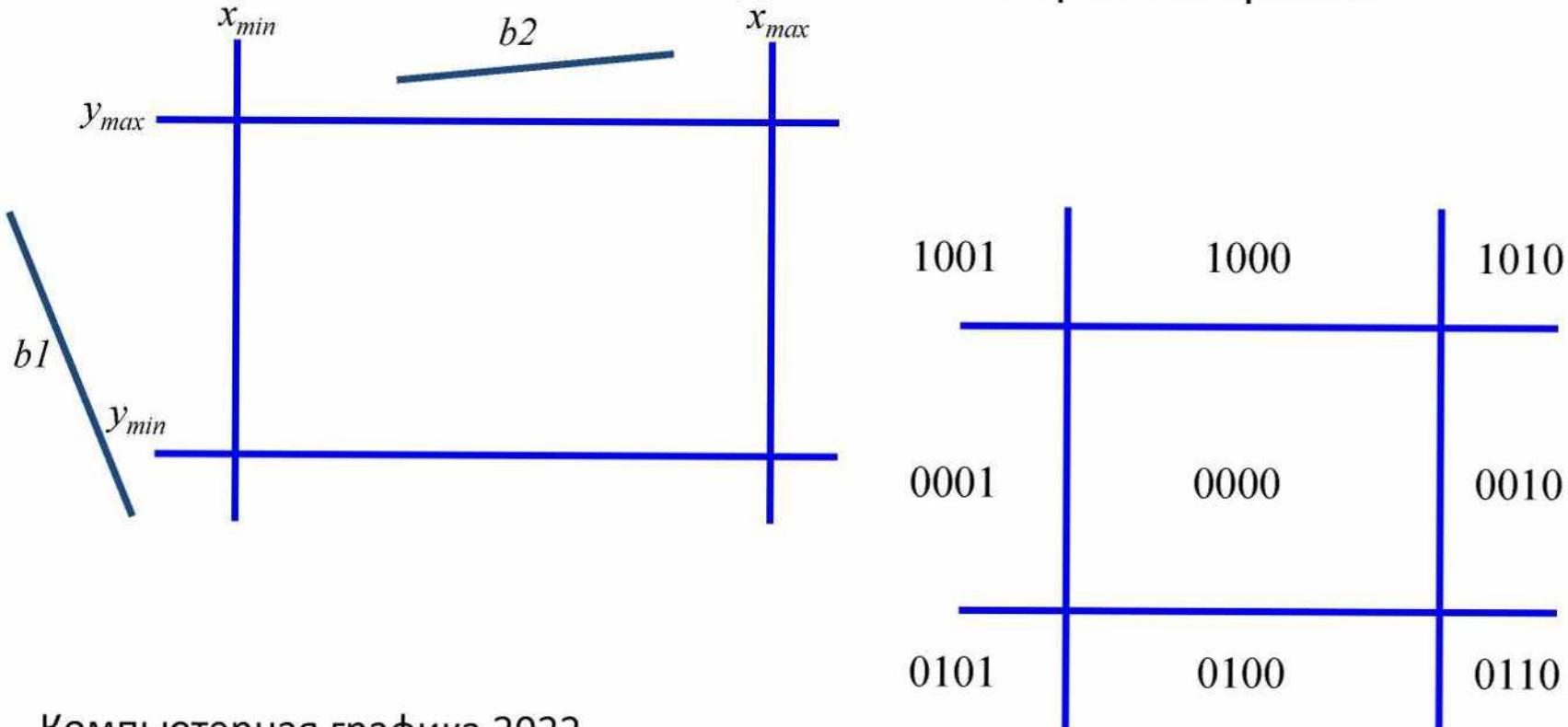




# Отсечение отрезков. Алгоритм Сазерленда-Коэна

2. Обе конечные точки лежат по одну сторону от какой-либо границы отсекающего окна (отрезок будет полностью невидимым)  $\Rightarrow O_1 \& O_2 \neq 0000$

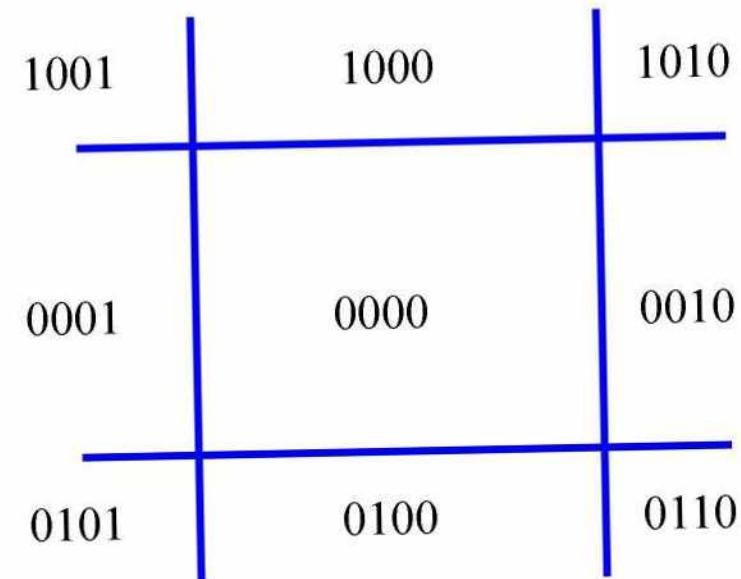
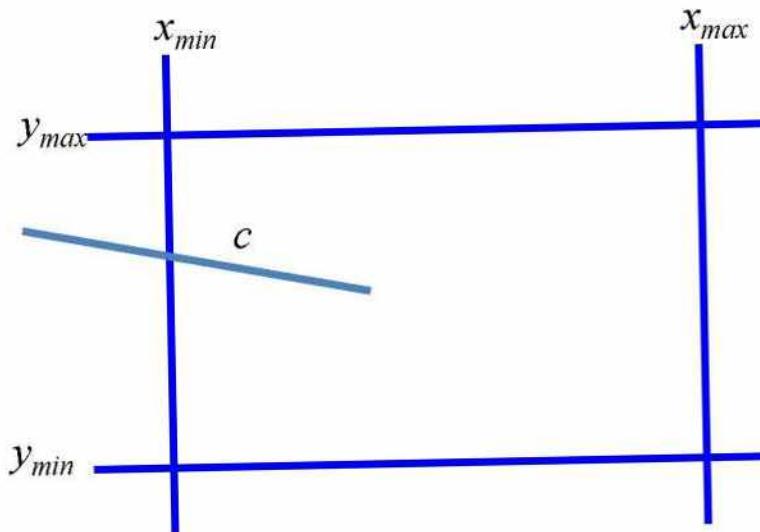
Отрезок исключается из дальнейшего рассмотрения





# Отсечение отрезков. Алгоритм Сазерленда-Коэна

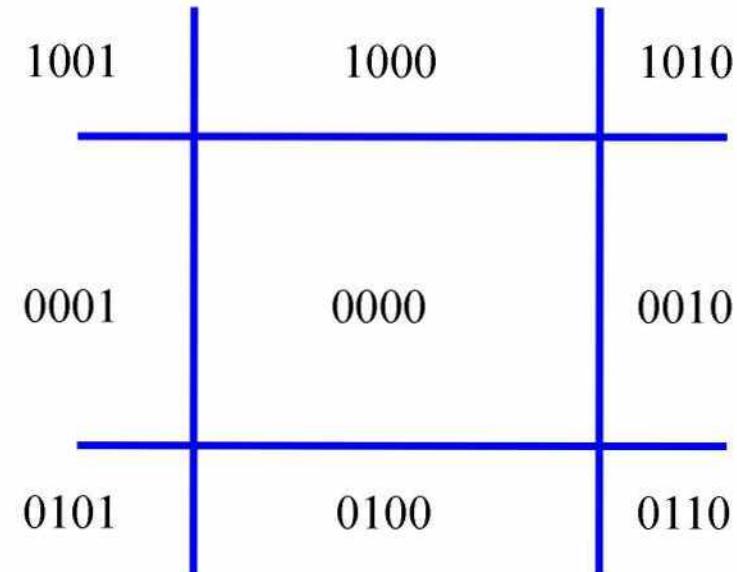
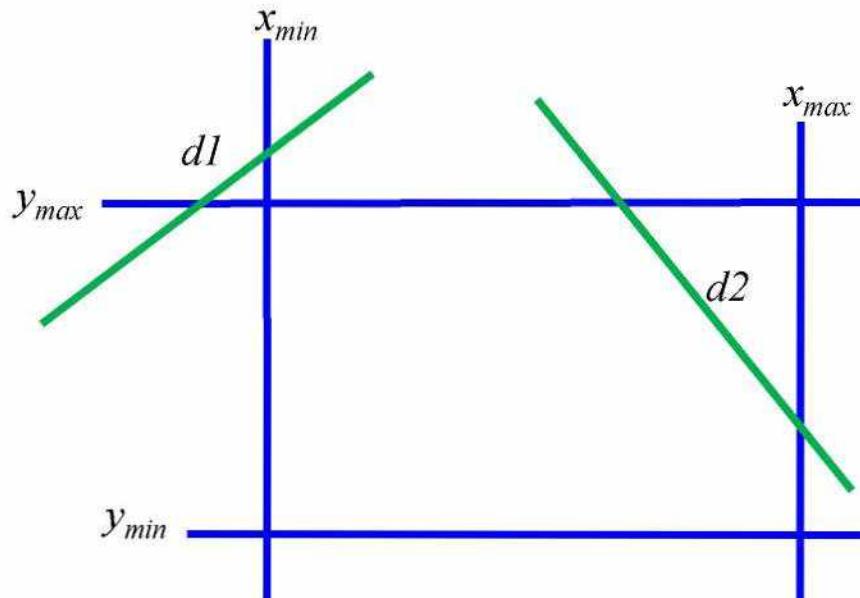
3. Одна из конечных точек лежит внутри отсекающего окна, другая вне его (отрезок будет частично видимым) :  
 $O_1 = 0000$  и  $O_2 \neq 0000$  либо  $O_1 \neq 0000$  и  $O_2 = 0000$ .  
Отрезок передается для дальнейшей обработки.





# Отсечение отрезков. Алгоритм Сазерленда-Коэна

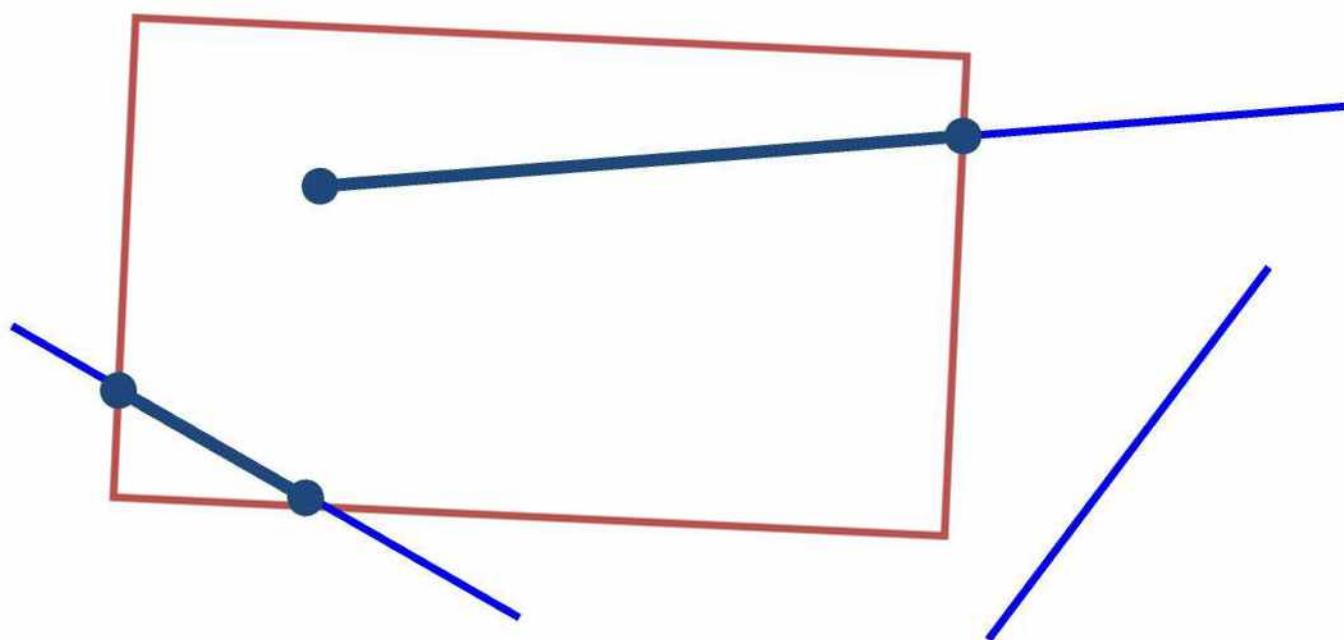
- Обе конечные точки лежат вне рамки отсечения, но по разные стороны от двух ее границ =>  $O_1 \& O_2 = 0000$ . Отрезок передается для дальнейшей обработки.





# Отсечение отрезков. Алгоритм Сазерленда-Коэна

После определения и отбрасывания целиком видимых и целиком невидимых отрезков, решаем задачу вычисления пересечения отрезков со сторонами отсекающего окна.





# Отсечение отрезков. Алгоритм Сазерленда-Коэна

**Обработка частично видимых отрезков.**

1. Если начальная точка  $P_1$  вне окна, то продолжить выполнение, иначе поменять местами  $P_1$  и  $P_2$ .
2. Проанализировать код точки  $P_1$  для определения стороны окна, с которой есть пересечение (по положению единичного бита).
3. Выполнить расчет пересечения:  $P_1^*$
4. Заменить точку  $P_1$  на точку пересечения  $P_1^*$

$$P_1 P_2 \Rightarrow P_1^* P_2$$

5. Если полученный отрезок полностью видимый – передать его для дальнейшей обработки, иначе перейти к пункту 1.



# Отсечение отрезков. Алгоритм Сазерленда-Коэна

$$P_1 = (x_1, y_1) \quad P_2 = (x_2, y_2)$$

$$y = k(x - x_1) + y_1$$

$$y = k(x - x_2) + y_2$$

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





# Отсечение отрезков. Алгоритм Сазерленда-Коэна

*Расчет точки пересечения со сторонами отсекающего окна.*

- Пересечение с верхней горизонтальной границей:  $y=y_{\max}$

$$x = x_1 + (1/k)(y_{\max} - y_1) \quad y = y_{\max}$$

- Пересечение с нижней горизонтальной границей:  $y=y_{\min}$

$$x = x_1 + (1/k)(y_{\min} - y_1) \quad y = y_{\min}$$

- Пересечение с правой вертикальной границей:  $x=x_{\max}$

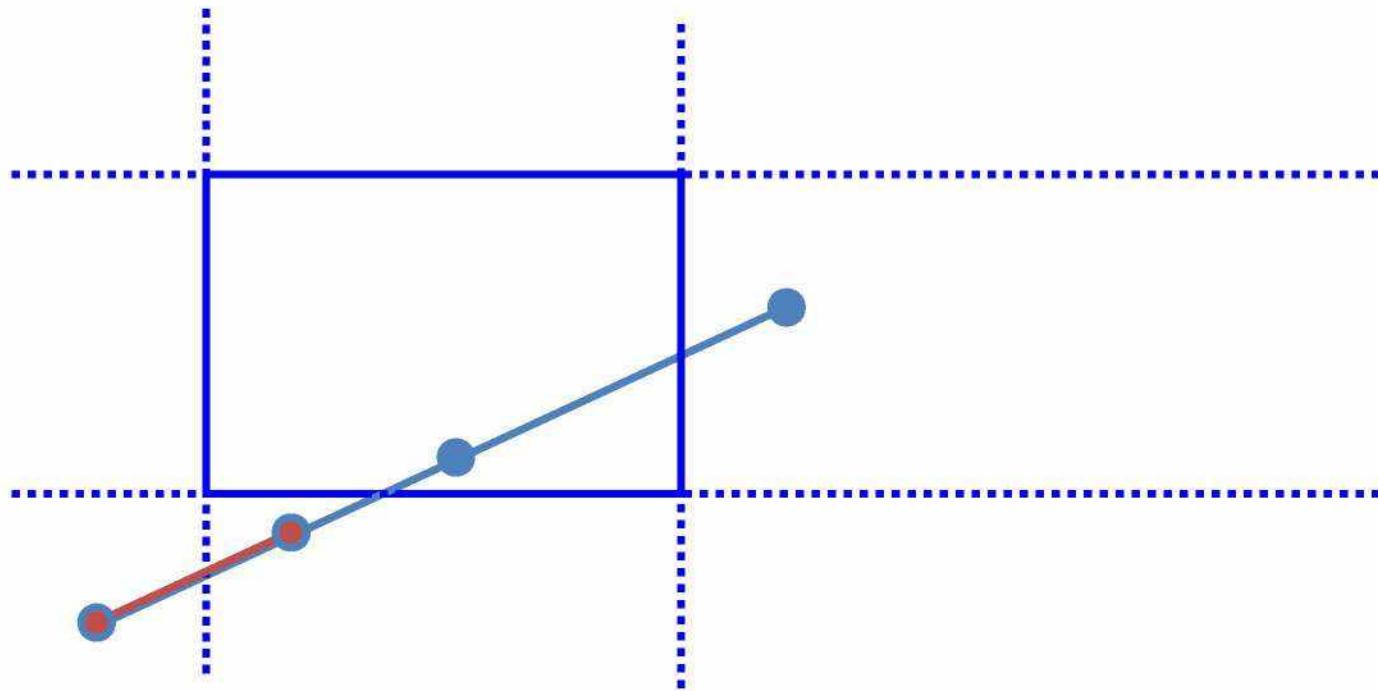
$$y = y_1 + k(x_{\max} - x_1) \quad x = x_{\max}$$

- Пересечение с левой вертикальной границей:  $x=x_{\min}$

$$y = y_1 + k(x_{\min} - x_1) \quad x = x_{\min}$$



# Отсечение отрезков. Алгоритм средней точки





# Отсечение отрезков. Алгоритм средней точки

Отсечь( отрезок АВ )

{

```
if( длина АВ меньше размера пикселя )
    return;
if( АВ лежит вне отсекающего прямоугольника
)
    return;
if( АВ лежит внутри отсекающего
прямоугольника )
    { Отобразить АВ; return; }
```

Отсечь (A, (A+B)/2);

Отсечь ((A+B)/2),B);

}



# Отсечение отрезков. Алгоритм средней точки

Алгоритм средней точки прост, но не очень эффективен на практике, так как требует большой глубины рекурсии. В большинстве случаев алгоритм Сазерленда-Коэна предпочтительнее. Прием с делением отрезка пополам может оказаться эффективным при отсечении относительно сложной непрямоугольной области, для которой проверить принадлежность точки к окну существенно проще, чем найти пересечение отрезка с границей.

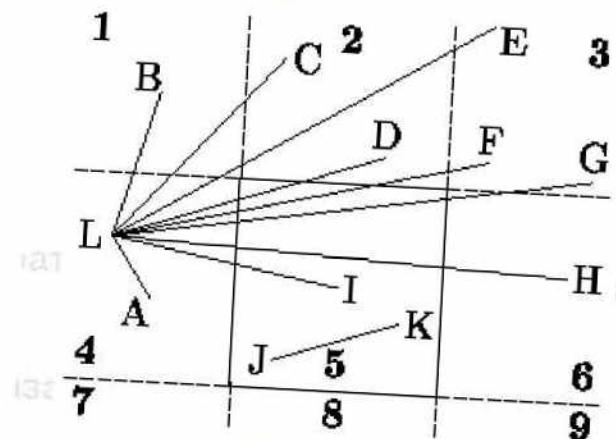
Алгоритм средней точки также обобщается на случай трехмерного пространства тривиальным образом.



# Отсечение отрезков. Двумерный FC-алгоритм

Fast Clipping алгоритм.

- Использует кодирование, но не конечных точек, а линий целиком.
- Возможен 81 вариант расположения отрезка.
- Реализация громоздкая, но сам алгоритм быстрый.





# Отсечение отрезков. Алгоритм Лианга-Барски

Ю-Донг Лианг и Брайан Барски, 1984.

Использует параметрическое представление отрезка.

Отсекающее окно – прямоугольник со сторонами, параллельными осям координат;  $(x_{\min}, y_{\min})$  – нижний левый угол;  $(x_{\max}, y_{\max})$  – верхний правый угол.

На 25-62% быстрее алгоритма Сазерленда-Коэна.



# Отсечение отрезков. Алгоритм Лианга-Барски

- Использует параметрическое представление отрезка.
- Отсекающее окно – прямоугольник со сторонами, параллельными осям координат;  $(x_{\min}, y_{\min})$  – нижний левый угол;  $(x_{\max}, y_{\max})$  – верхний правый угол.
- На 25-62% быстрее алгоритма Сазерленда-Коэна.

$$P_1 = (x_1, y_1) \quad P_2 = (x_2, y_2)$$

$$P(t) = P_1 + t(P_2 - P_1), \quad 0 \leq t \leq 1$$

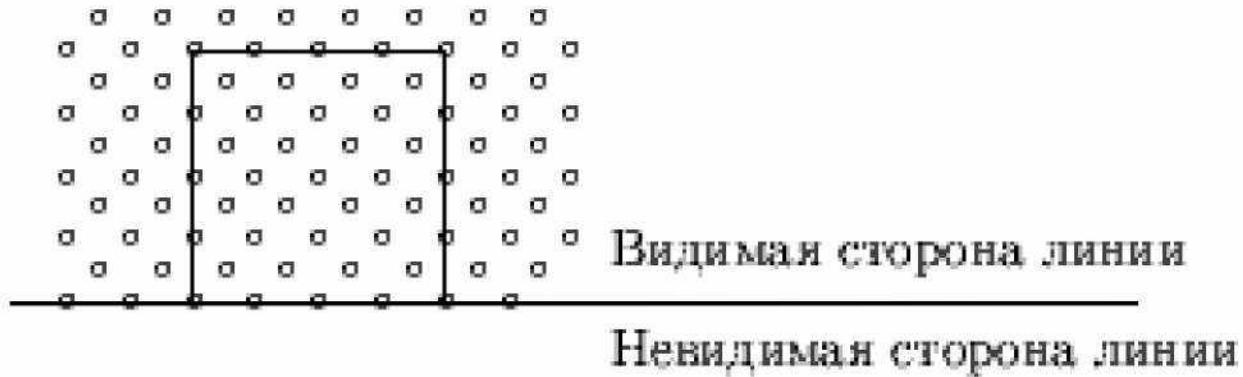
$$x = x_1 + t(x_2 - x_1), \quad 0 \leq t \leq 1$$

$$y = y_1 + t(y_2 - y_1), \quad 0 \leq t \leq 1$$



# Отсечение отрезков. Алгоритм Лианга-Барски

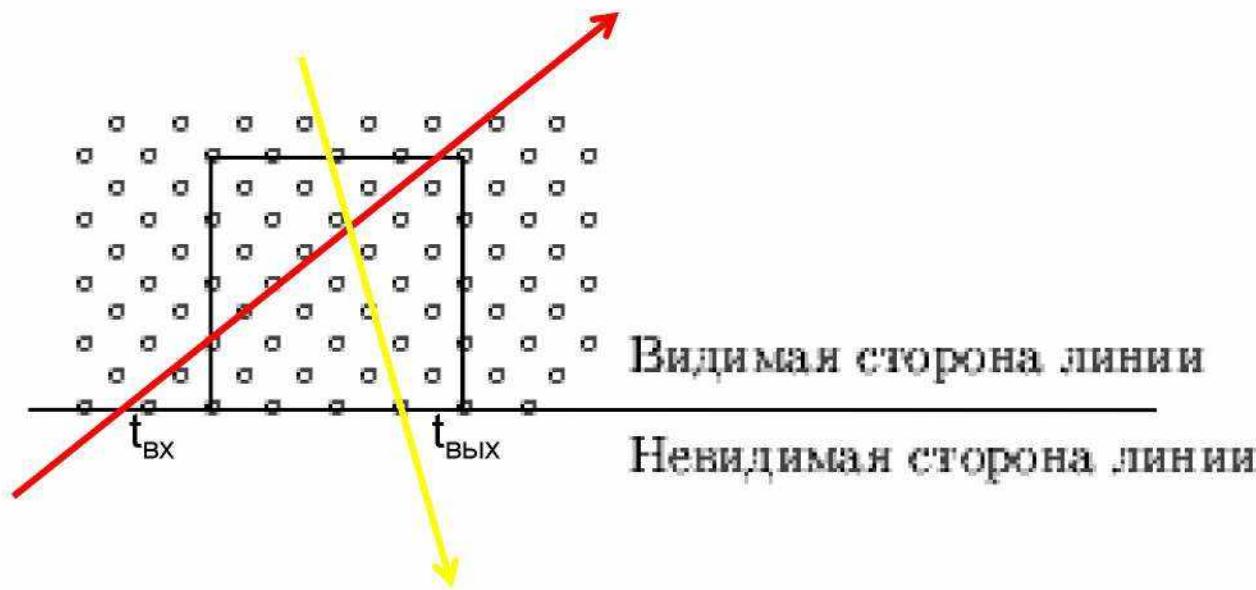
Каждая из прямых, содержащих стороны отсекающего прямоугольника, делит плоскость на 2 области: видимую и невидимую.



Соответственно, точка пересечения со стороной прямоугольника (или с ее продолжением) классифицируется как «входящая», если начальная точка отрезка расположена в «невидимой» относительно данной стороны области, а конечная – в «видимой», и как «выходящая» в противном случае.



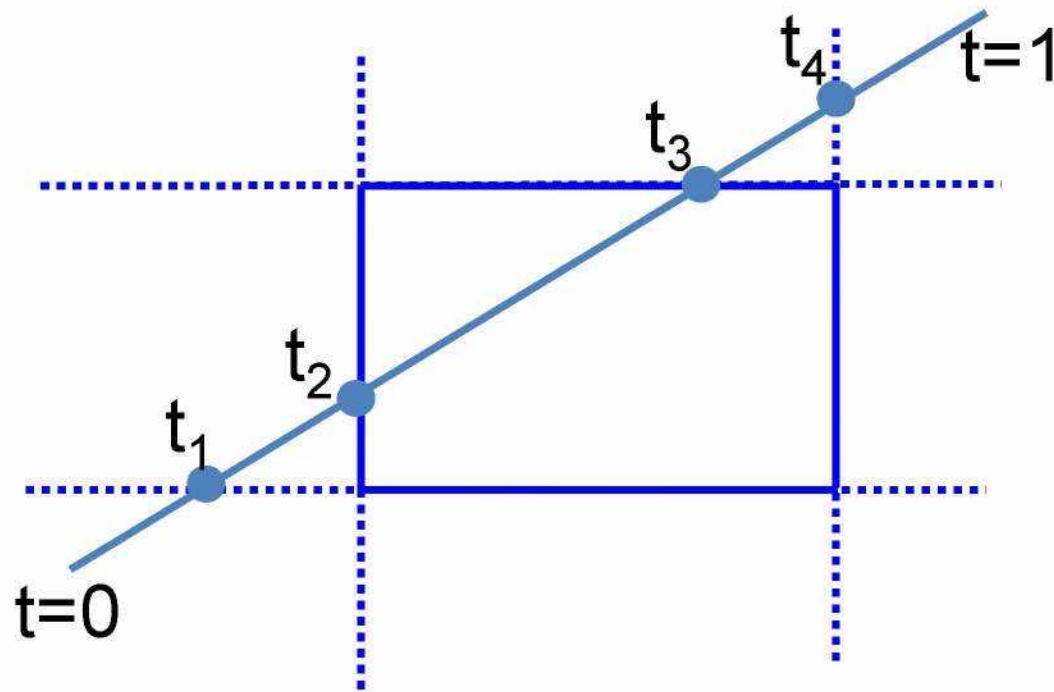
# Отсечение отрезков. Алгоритм Лианга-Барски



Для того, чтобы получить истинные точки пересечения отрезка со сторонами окна, необходимо выбрать максимальное значение параметра  $t$  из всех «входящих», и минимальное значение параметра  $t$  из всех «выходящих».



# Отсечение отрезков. Алгоритм Лианга-Барски





# Отсечение отрезков. Алгоритм Лианга-Барски

- Пересечение с нижней горизонтальной границей:  $y=y_{\min}$

$$t_1 = \frac{y_{\min} - y_1}{y_2 - y_1}$$

- Пересечение с левой вертикальной границей:  $x=x_{\min}$

$$t_2 = \frac{x_{\min} - x_1}{x_2 - x_1}$$

- Пересечение с верхней горизонтальной границей:  $y=y_{\max}$

$$t_3 = \frac{y_{\max} - y_1}{y_2 - y_1}$$

- Пересечение с правой вертикальной границей:  $x=x_{\max}$

$$t_4 = \frac{x_{\max} - x_1}{x_2 - x_1}$$



# Отсечение отрезков. Алгоритм Лианга-Барски

- Следующим этапом является определение того, какие из четырех найденных параметров соответствуют истинным точкам пересечения.
- Значения  $t$  за пределами интервала  $[0;1]$  соответствуют точкам, лежащим вне исходного отрезка – они отбрасываются сразу.
- В итоге мы должны получить ровно 2 значения параметра  $t$ , соответствующих координатам видимой части отрезка.



# Отсечение отрезков. Алгоритм Лианга-Барски

Параметрическое задание отрезка:

$$\begin{aligned}x &= x_1 + t(x_2 - x_1) = x_1 + t \cdot dx, \quad 0 \leq t \leq 1 \\y &= y_1 + t(y_2 - y_1) = y_1 + t \cdot dy, \quad 0 \leq t \leq 1\end{aligned}\tag{1}$$

Неравенства для точек отсекающего окна

$$\begin{aligned}x_{\min} \leq x &\leq x_{\max} \\y_{\min} \leq y &\leq y_{\max}\end{aligned}\tag{2}$$

Подставляя (2) в (1), получаем:

$$\begin{aligned}x_{\min} \leq x_1 + t \cdot dx &\leq x_{\max} \\y_{\min} \leq y_1 + t \cdot dy &\leq y_{\max}\end{aligned}\tag{3}$$



# Отсечение отрезков. Алгоритм Лианга-Барски

$$\begin{aligned}x_{\min} \leq x_1 + t \cdot dx &\leq x_{\max} \\y_{\min} \leq y_1 + t \cdot dy &\leq y_{\max}\end{aligned}\quad (3)$$

$$\begin{aligned}-t \cdot dx \leq x_1 - x_{\min} \\t \cdot dx \leq x_{\max} - x_1 \\-t \cdot dy \leq y_1 - y_{\min} \\t \cdot dy \leq y_{\max} - y_1\end{aligned}\quad (4)$$

$$\begin{aligned}t \cdot S_2 \leq Q_2 \\t \cdot S_4 \leq Q_4 \\t \cdot S_1 \leq Q_1 \\t \cdot S_3 \leq Q_3\end{aligned}\quad (5)$$

$$t \cdot S_i \leq Q_i, i = 1, 2, 3, 4$$

Вводятся  
следующие  
обозначения:

$$\begin{aligned}S_1 = -dy & \quad S_2 = -dx \\S_3 = dy & \quad S_4 = dx\end{aligned}$$

$$\begin{aligned}Q_1 &= y_1 - y_{\min} \\Q_2 &= x_1 - x_{\min} \\Q_3 &= y_{\max} - y_1 \\Q_4 &= x_{\max} - x_1\end{aligned}$$



# Отсечение отрезков. Алгоритм Лианга-Барски

$$t \cdot S_i \leq Q_i, i = 1, 2, 3, 4$$

$$Q_1 = y_1 - y_{\min}$$

$$S_1 = -dy$$

$$Q_2 = x_1 - x_{\min}$$

$$S_2 = -dx$$

$$Q_3 = y_{\max} - y_1$$

$$S_3 = dy$$

$$Q_4 = x_{\max} - x_1$$

$$S_4 = dx$$

Знак  $S_i$  будет показывать ориентацию отрезка относительно соответствующей границы: если  $S_i > 0$ , то отрезок «выходит» из видимой области через границу  $i$ ; если  $S_i < 0$ , то отрезок «входит» в видимую область через границу  $i$ ; если  $S_i = 0$ , то отрезок параллелен данной границе.

Знак  $Q_i$  будет показывать, по какую сторону от соответствующей границы находится начальная точка  $P_1$ .

$Q_i \geq 0 \rightarrow$  начальная точка находится на видимой стороне

$Q_i < 0 \rightarrow$  начальная точка находится на невидимой стороне



# Отсечение отрезков. Алгоритм Лианга-Барски

$$t_{\text{вх}} = 0; t_{\text{вых}} = 1;$$

Для каждого  $i=1,2,3,4$ :

{Если  $S_i > 0$  то

$$t_{\text{вых}} = \min\{Q_i/S_i, t_{\text{вых}}\}$$

Если  $S_i < 0$  то =>

$$t_{\text{вх}} = \max\{Q_i/S_i, t_{\text{вх}}\}$$

Если  $S_i = 0$  то :

если  $Q_i < 0$  то отрезок невидим относительно границы  $i$ , отбрасываем его

если  $Q_i \geq 0$  то отрезок полностью видимый относительно границы  $i$ ,  
переходим к следующему  $i$  }

$P_{\text{вх}} = P_1 + t_{\text{вх}} \cdot (P_2 - P_1)$  и  $P_{\text{вых}} = P_1 + t_{\text{вых}} \cdot (P_2 - P_1)$  – результирующие координаты видимой части отрезка, передаваемые для дальнейшей растеризации



# Отсечение отрезков. Алгоритм Кируса-Бека.

Майкл Кирус и Джей Бек, 1978.

- Обобщение алгоритма Лианга-Барски.
- Использует параметрическое представление отрезка.
- Отсекающее окно – произвольное, выпуклой формы.

$$P(t) = P_1 + t(P_2 - P_1), \quad 0 \leq t \leq 1$$

$$x = x_1 + t(x_2 - x_1), \quad 0 \leq t \leq 1$$

$$y = y_1 + t(y_2 - y_1), \quad 0 \leq t \leq 1$$



# Отсечение отрезков. Алгоритм Кируса-Бека.

Общий принцип работы алгоритма:

1. Рёбра отсекающего многоугольника обходятся против часовой стрелки. Пусть текущее ребро –  $Ei$ ,  $Ni$  – внутренняя нормаль.
2. Для ребра  $Ei$  вычисляем параметр  $t_{Ei}$ , задающий точку пересечения прямой, содержащей ребро  $Ei$ , с прямой, на которой лежит отрезок  $P1P2$ .



# Отсечение отрезков. Алгоритм Кируса-Бека.

Общий принцип работы алгоритма:

3. Вычисляем скалярное произведение  $S$  вектора  $P1P2$  и внутренней нормали  $N$  к прямой, содержащей ребро  $Ei$ .

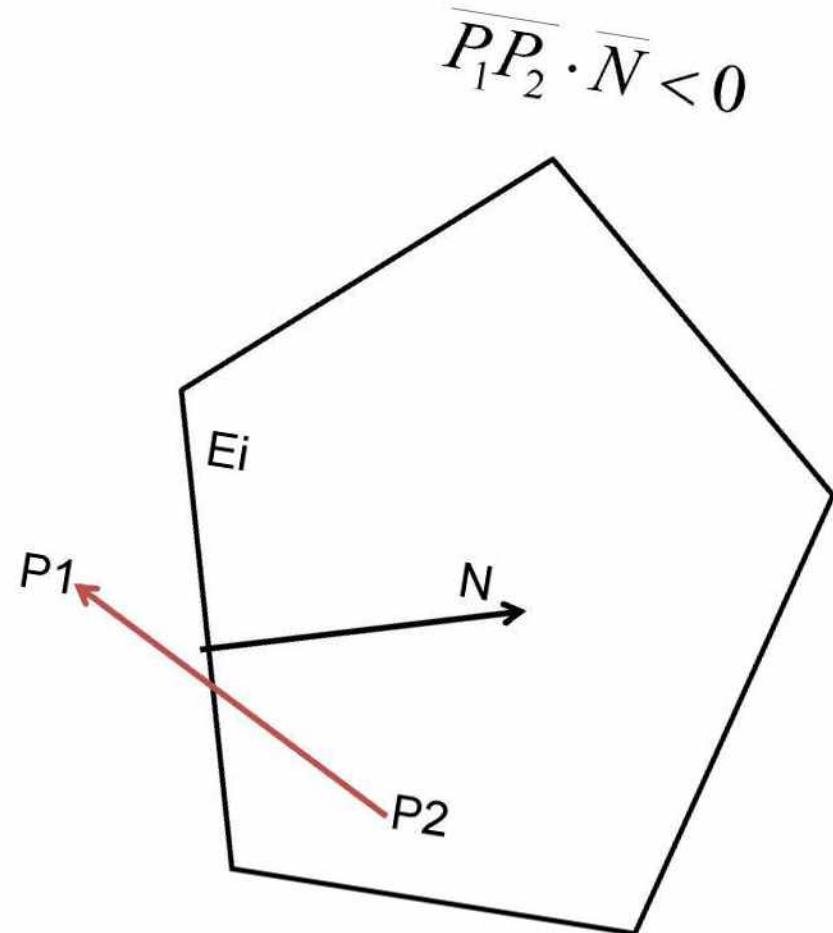
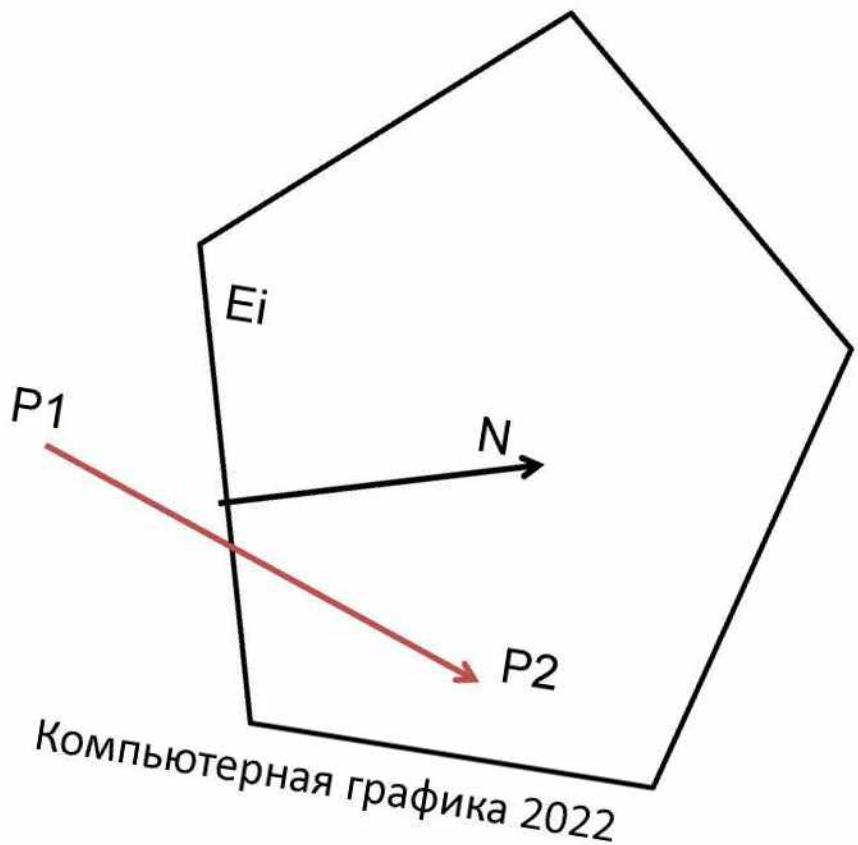
В зависимости от знака произведения возникает одна из следующих ситуаций:

- $S > 0$ : отрезок входит в окно через ребро  $Ei$ ;
- $S < 0$ : отрезок выходит из окна через ребро  $Ei$ ;
- $S = 0$ : отрезок параллелен прямой, содержащей ребро  $Ei$ ;



# Отсечение отрезков. Алгоритм Кируса-Бека.

$$\overrightarrow{P_1 P_2} \cdot \overrightarrow{N} > 0$$





# Отсечение отрезков. Алгоритм Кируса-Бека.

4. Если вычисленный параметр  $t_{Ei}$  лежит вне интервала  $[0;1]$ , то отбрасываем его.

Если вычисленный параметр  $t_{Ei}$  лежит в интервале  $[0;1]$ , а вычисленное скалярное произведение  $S>0$ , то записываем  $t_{Ei}$  в список  $T_{вх}$  потенциальных точек входа в отсекающее окно.

Если вычисленный параметр  $t_{Ei}$  лежит в интервале  $[0;1]$ , а вычисленное скалярное произведение  $S<0$ , то записываем  $t_{Ei}$  в список  $T_{вых}$  потенциальных точек выхода из отсекающего окна.

Если вычисленное скалярное произведение  $S=0$ , то отрезок либо полностью видимый относительно данного ребра, либо полностью невидимый.



# Отсечение отрезков. Алгоритм Кируса-Бека.

5. Ищем максимальное значение параметра в списке  $T_{вх}$  и минимальное значение параметра в списке  $T_{вых}$ .

Найденные значения и будут искомыми точками пересечения.

В случае, если один из списков окажется пустым, то в результате получим одну точку отсечения.

Если пустым оказался список  $T_{вх}$ , то внутри отсекающего окна лежит начальная точка отрезка, если же пуст список  $T_{вых}$ , то внутри отсекающего окна лежит конечная точка.

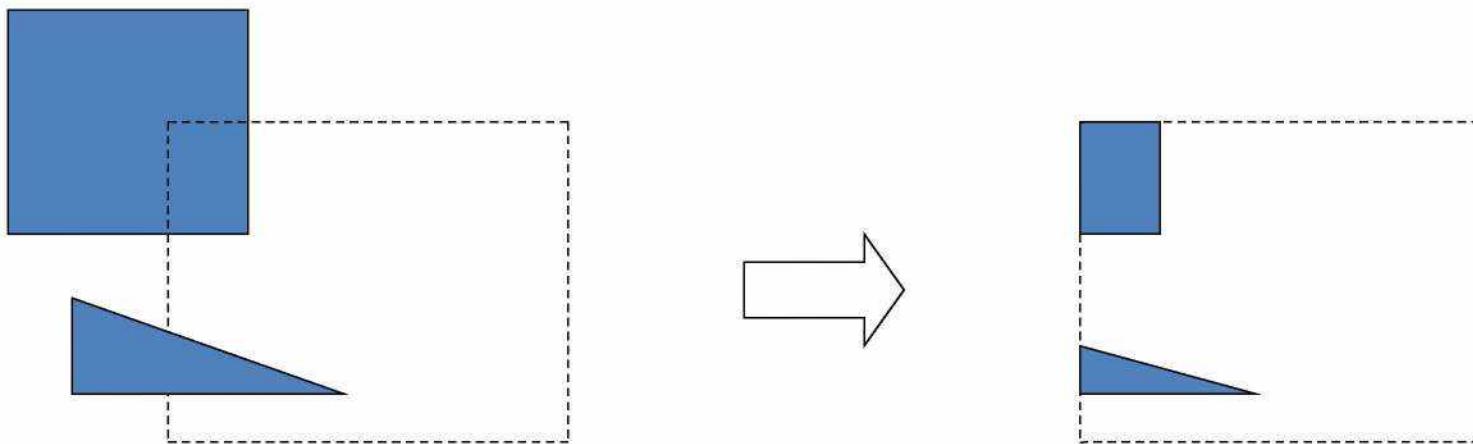
Если оба списка пустые, то отрезок отбрасывается, как полностью невидимый.



# Отсечение многоугольников

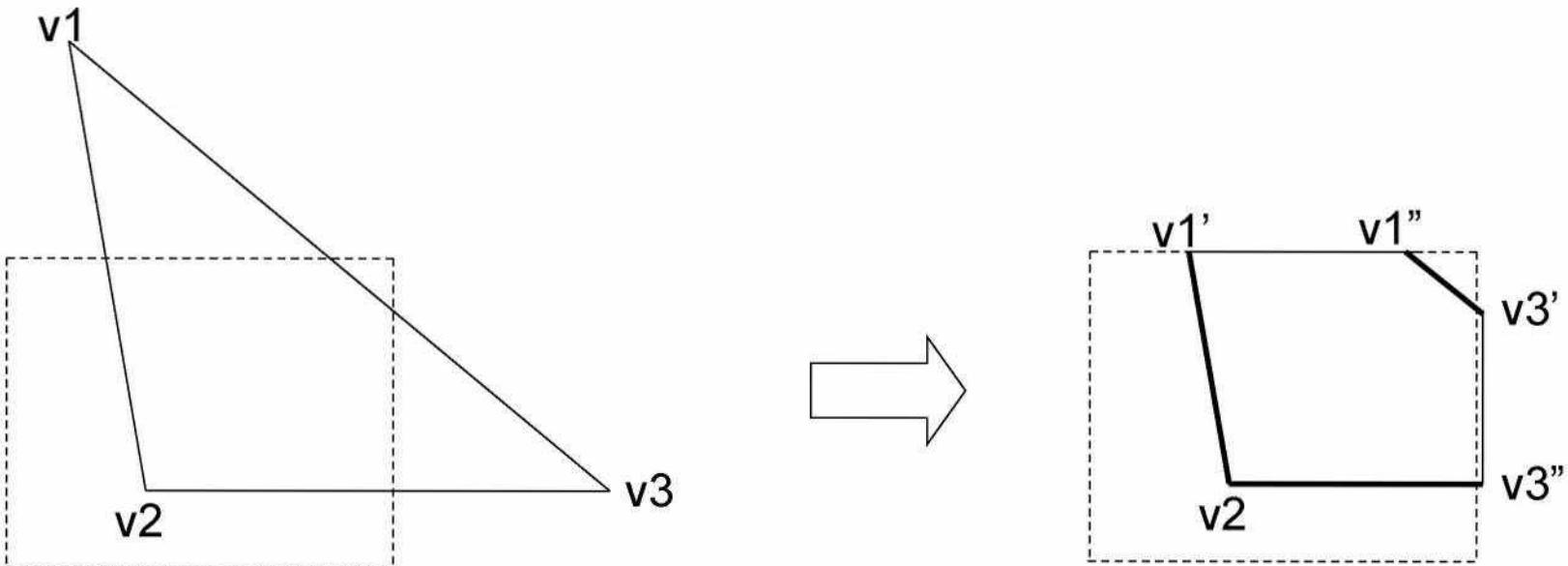
Замкнутый многоугольник можно рассматривать как набор отрезков. Тогда к нему применяются стандартные методы отсечения отрезков. В результате получаем снова набор разрозненных отрезков.

Однако чаще многоугольник рассматривается как сплошная область, поэтому необходимо, чтобы замкнутость сохранилась и у результата.





# Отсечение многоугольников



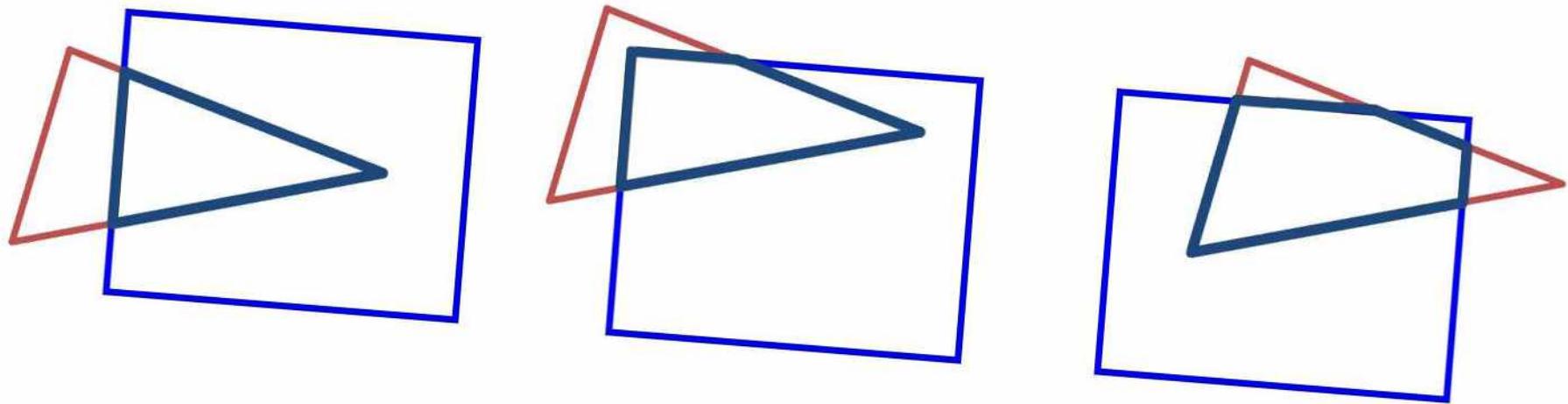
Результат:

либо набор отрезков  $v1'v2, v2v3'', v1''v3'$

либо многоугольник  $v1'v1''v3''v3''v2$

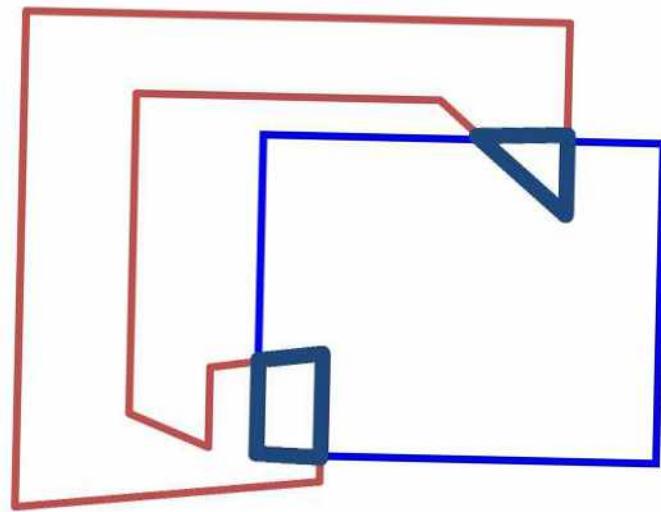


# Отсечение многоугольников





# Отсечение многоугольников





# Отсечение многоугольников. Алгоритм Сазерленда-Ходжмана

Sutherland-Hodgman algorithm, 1974

Айвен Сазерленд

Гарри Ходжман

**Алгоритм отсечения выпуклого многоугольника  
относительно выпуклого окна.**

Основная идея алгоритма состоит в том, что отсечь многоугольник относительно одной прямой достаточно просто. Поэтому алгоритм выполняет последовательное отсечение многоугольника каждой из сторон отсекающего окна, на каждом шаге формируя новый список ребер.



# Отсечение многоугольников. Алгоритм Сазерленда-Ходжмана

Пример работы алгоритма для отсечения произвольного выпуклого многоугольника прямоугольным отсекающим окном.

- Исходный многоугольник задается списком вершин, который порождает список ребер.
- Отсекаем исходный многоугольник поочередно каждой из сторон окна по следующим правилам, на каждом этапе формируя новый список вершин и соответствующий ему промежуточный многоугольник

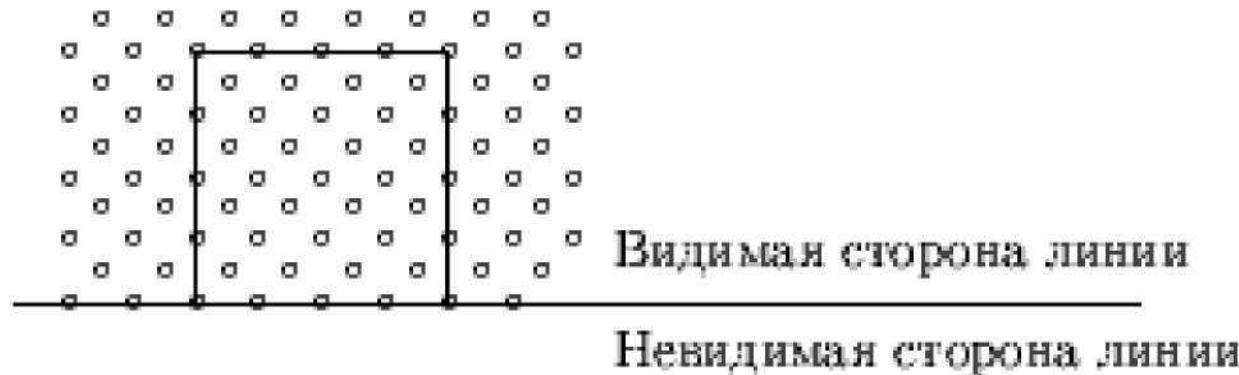


# Отсечение многоугольников. Алгоритм Сазерленда-Ходжмана

Обрабатываем пересечение каждого ребра исходного многоугольника с текущей стороной отсекающего окна.

В процессе этой обработки в новый список сохраняются некоторые вершины, находящиеся «внутри» относительно этой стороны и некоторые точки пересечения.

«Внутри» стороны – внутри полуплоскости, отсекаемой этой стороной, то есть на «видимой» области.



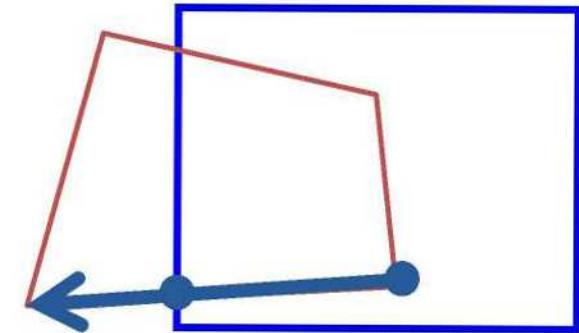
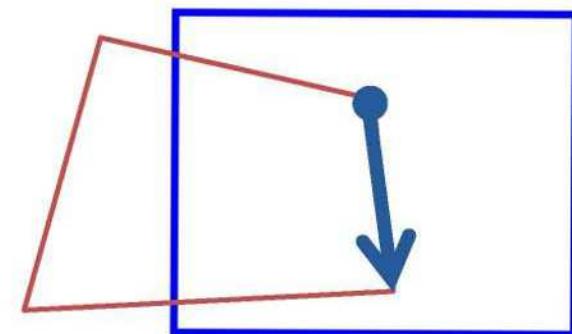


# Отсечение многоугольников.

## Алгоритм Сазерленда-Ходжмана

При такой обработке возможны следующие варианты (пример для пересечения с левой вертикальной границей):

1. Начало и конец ребра многоугольника лежат внутри стороны отсекающего окна; тогда добавляем точку начала к вершинам полигона-результата.
2. Начало лежит внутри стороны отсекающего окна, конец лежит вне; в этом случае вычисляем точку пересечения ребра многоугольника и стороны отсекающего окна; добавляем в список вершин результата начало ребра и вслед за ним точку пересечения.

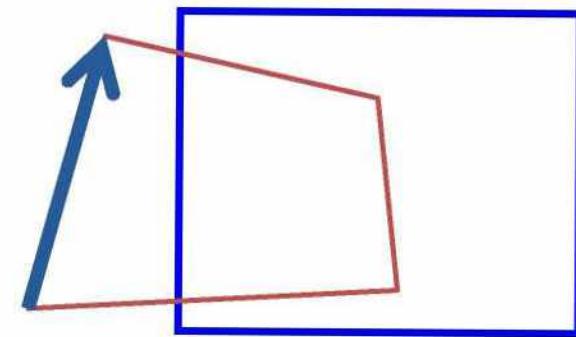




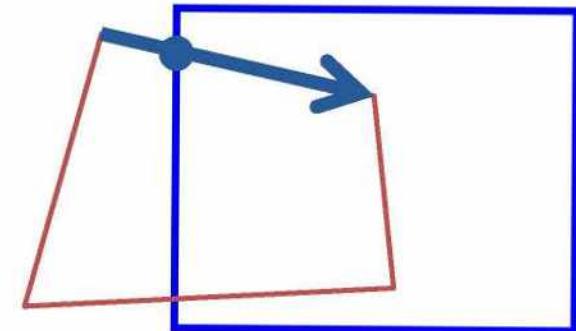
# Отсечение многоугольников.

## Алгоритм Сазерленда-Ходжмана

3. Начало не лежит внутри стороны отсекающего окна, конец тоже; переходим к следующему ребру, никак не изменяя результат.



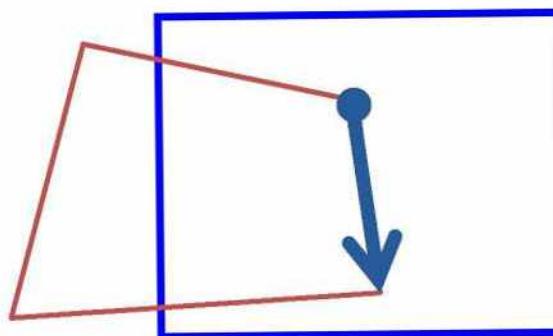
4. Начало не лежит внутри стороны отсекающего окна, конец лежит; считаем точку пересечения и добавляем в новый список только ее.



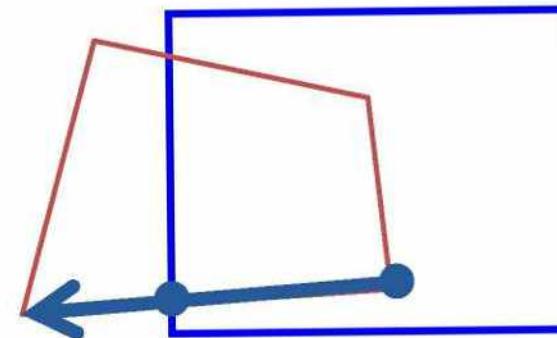


# Отсечение многоугольников. Алгоритм Сазерленда-Ходжмана

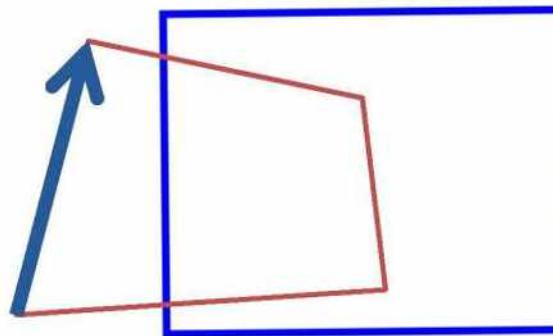
1)



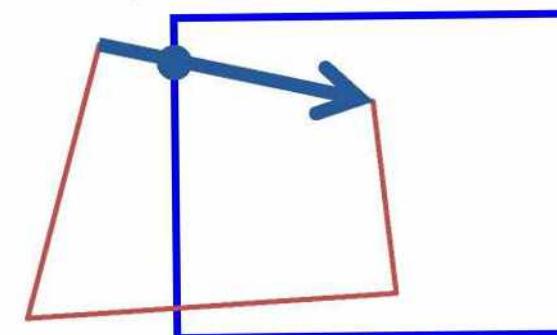
2)



3)



4)





# Отсечение многоугольников. Алгоритм Сазерленда-Ходжмана

Частные случаи.

- Точка, лежащая на границе отсекающего окна, считается внутренней точкой (лежащей внутри окна).
- Отрезок, лежащий на одной из границ окна, считается внутренним (лежащим внутри окна).



# Отсечение многоугольников. Алгоритм Сазерленда-Ходжмана

Особенности реализации.

- Многоугольник задается последовательностью вершин  $\{P_1, \dots, P_n\}$
- Соответствующая последовательность ребер -  $\{P_1 P_2, P_2 P_3, \dots, P_{n-1} P_n, P_n P_1\}$
- Отсекающее окно задается координатами  $(x_{min}, y_{min})$   $(x_{max}, y_{max})$
- Поиск точки пересечения ребра и стороны – просто (рассмотреть все случаи!).
- Определение «видимости» или «невидимости» начальной и конечной точек очередного ребра – по анализу характеристического кода (из алгоритма Сазерленда-Коэна).

Пересечение со стороной  $x=x_{min}$ :

$b_3 = 1 \Rightarrow$  видимая точка, иначе невидимая

Пересечение со стороной  $x=x_{max}$ :

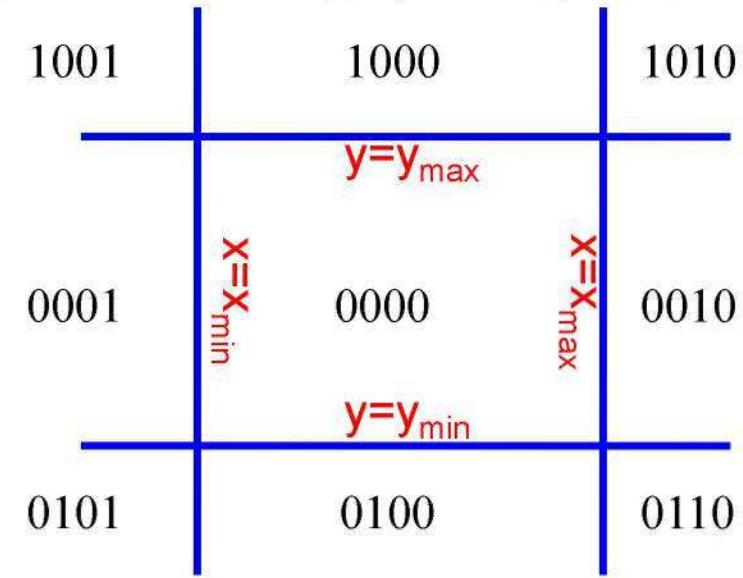
$b_2 = 1 \Rightarrow$  видимая точка, иначе невидимая

Пересечение со стороной  $y=y_{min}$ :

$b_1 = 1 \Rightarrow$  видимая точка, иначе невидимая

Пересечение со стороной  $y=y_{max}$ :

$b_0 = 1 \Rightarrow$  видимая точка, иначе невидимая





# Отсечение многоугольников. Алгоритм Сазерленда-Ходжмана

## Определение видимости точек.

Случай прямоугольного отсекающего окна со  
сторонами, параллельными координатным осям

Пересечение со стороной  $x=x_{\min}$ :

$b_3 = 1 \Rightarrow$  видимая точка, иначе невидимая

Пересечение со стороной  $x=x_{\max}$ :

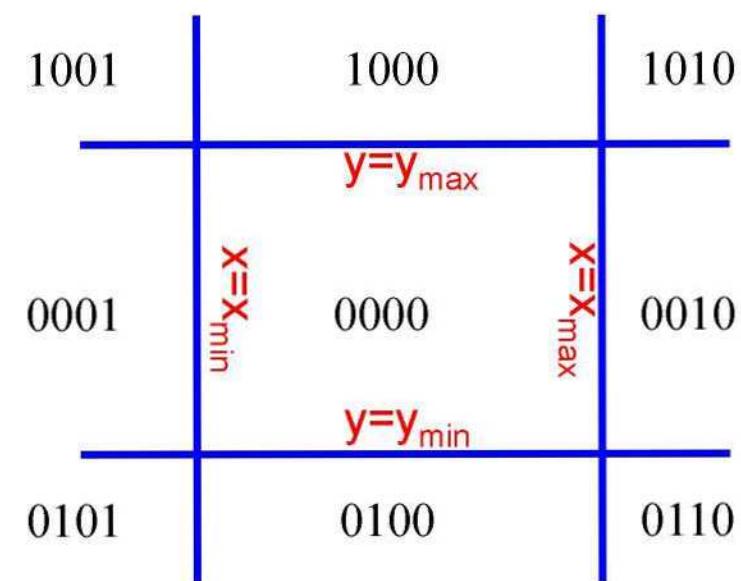
$b_2 = 1 \Rightarrow$  видимая точка, иначе невидимая

Пересечение со стороной  $y=y_{\min}$ :

$b_1 = 1 \Rightarrow$  видимая точка, иначе невидимая

Пересечение со стороной  $y=y_{\max}$ :

$b_0 = 1 \Rightarrow$  видимая точка, иначе невидимая





# **Отсечение многоугольников. Алгоритм Сазерленда-Ходжмана**

**Определение видимости точек.**

Случай произвольного выпуклого отсекающего окна:  
используем нормаль



Белорусский Государственный Университет  
Факультет прикладной математики и информатики

# Компьютерная графика

Василевский Константин

Викторович

17 лекций,

17 лабораторных занятий



# Геометрические преобразования

- Преобразования координат на плоскости.
- Алгоритмы двумерных и трехмерных преобразований.
- Однородные координаты.
- Матрицы аффинных преобразований.
- Перемещение, масштабирование, поворот.
- Положительное вращение вокруг осей.
- Поворот вокруг произвольной оси.
- Задача поворота относительно точки.



# Преобразования координат

Изучение математического аппарата, лежащего в основе машинной графики, начинается с рассмотрения способов вывода и преобразования точек и линий. Эти способы наряду с соответствующими алгоритмами рисования используются для изображения объектов или визуализации графической информации.

Возможность проводить преобразования точек и линий является фундаментом машинной графики. Нарисованный объект может быть представлен в нужном масштабе, повернут, перемещен или модифицирован в соответствии с требованиями решаемой задачи.



# Матрица преобразования на плоскости

$$T = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

матрица преобразования

$$\begin{bmatrix} x & y \end{bmatrix}$$

исходные координаты

Преобразованные координаты:

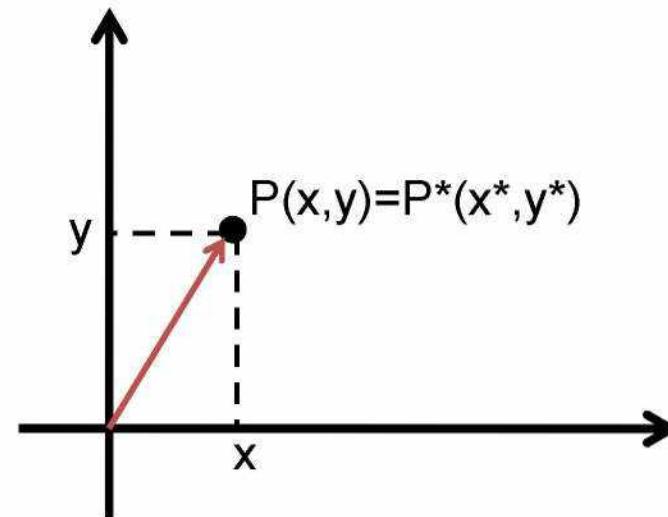
$$\begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} ax + cy & bx + dy \end{bmatrix}$$



# Матрица преобразования на плоскости

Тождественное преобразование

$$E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



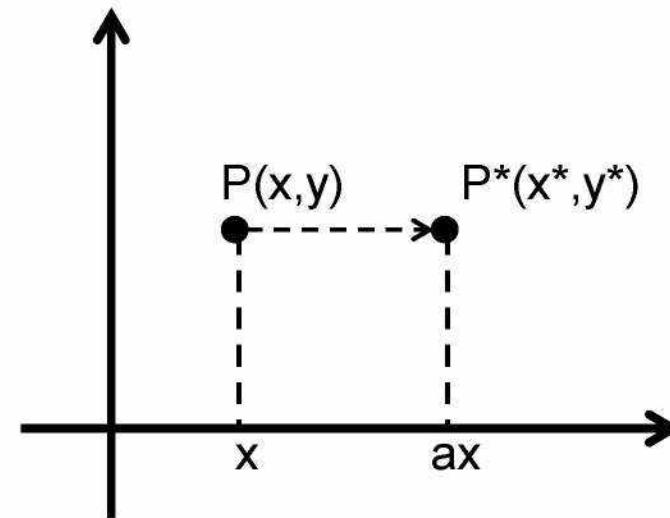
$$\begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix}$$



# Матрица преобразования на плоскости

Масштабирование по координате x

$$T = \begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix}$$



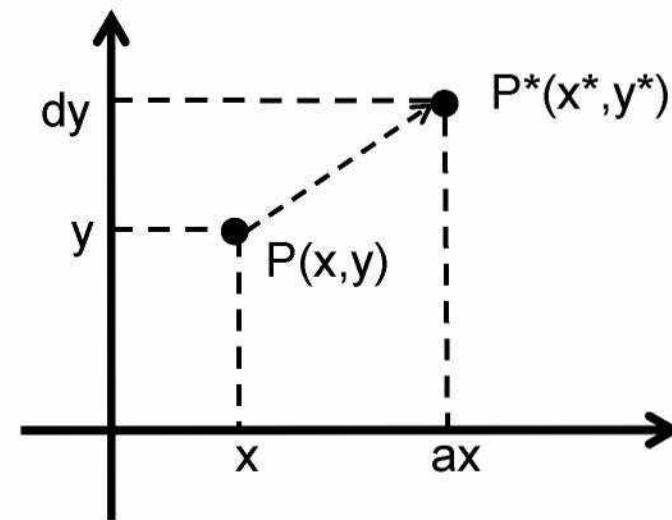
$$\begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} ax & y \end{bmatrix}$$



# Матрица преобразования на плоскости

Масштабирование по координатам x и y

$$T = \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix}$$



$$\begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} = \begin{bmatrix} ax & dy \end{bmatrix}$$

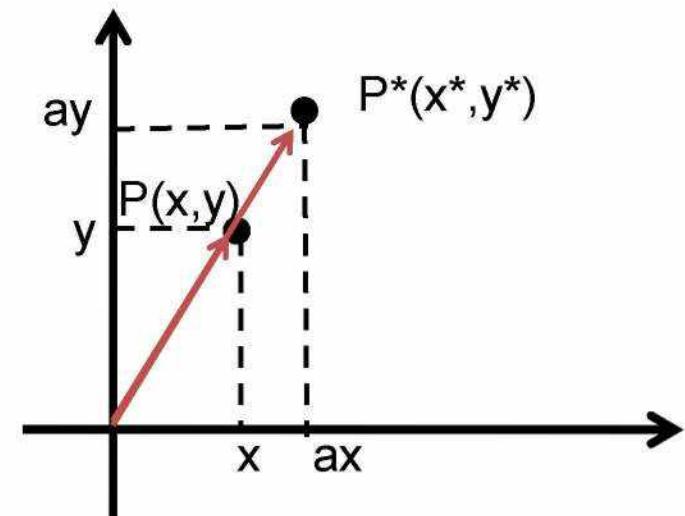


# Матрица преобразования на плоскости

Масштабирование по координатам x и y

$$\begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} = \begin{bmatrix} ax & dy \end{bmatrix}$$

$a=d>1 \Rightarrow$  растяжение вектора P  
 $0 < a=d < 1 \Rightarrow$  сжатие вектора P  
 $a<0$  или  $d<0 \Rightarrow$  отражение

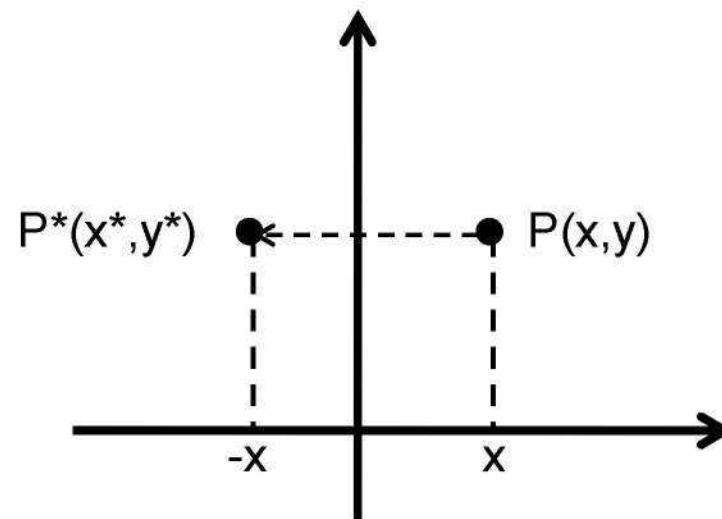




# Матрица преобразования на плоскости

Отражение относительно оси у

$$T = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$



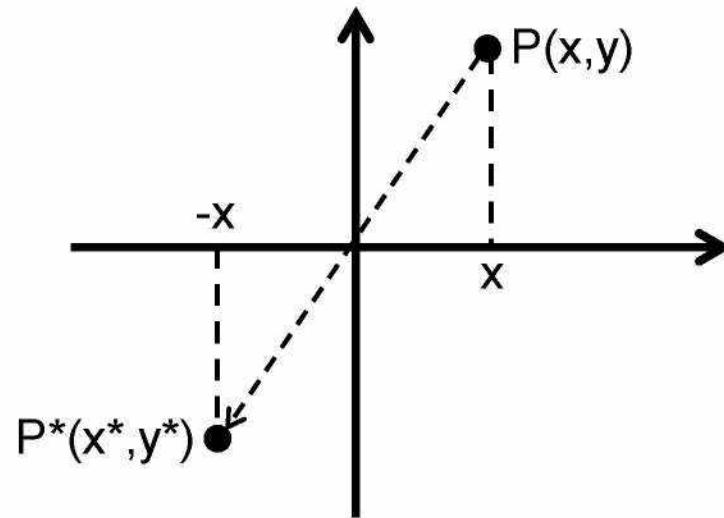
$$\begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -x & y \end{bmatrix}$$



# Матрица преобразования на плоскости

Отражение относительно начала координат

$$T = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$



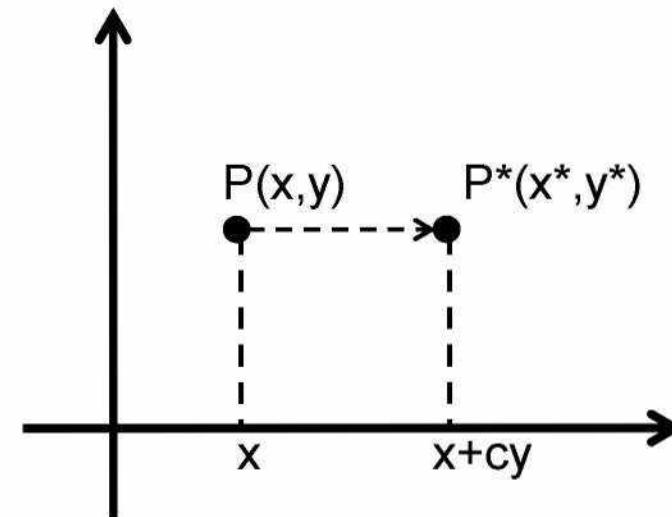
$$\begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} -x & -y \end{bmatrix}$$



# Матрица преобразования на плоскости

Сдвиг

$$T = \begin{bmatrix} 1 & 0 \\ c & 1 \end{bmatrix}$$



$$\begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ c & 1 \end{bmatrix} = \begin{bmatrix} x + cy & y \end{bmatrix}$$



# Матрица преобразования на плоскости

Общий вид

$$T = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} ax + cy & bx + dy \end{bmatrix}$$

Начало координат инвариантно относительно  
преобразования общего вида.



# Матрица преобразования на плоскости. Преобразование прямых линий

$$T = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \overline{AB} \quad A(x_1, y_1) \quad B(x_2, y_2)$$

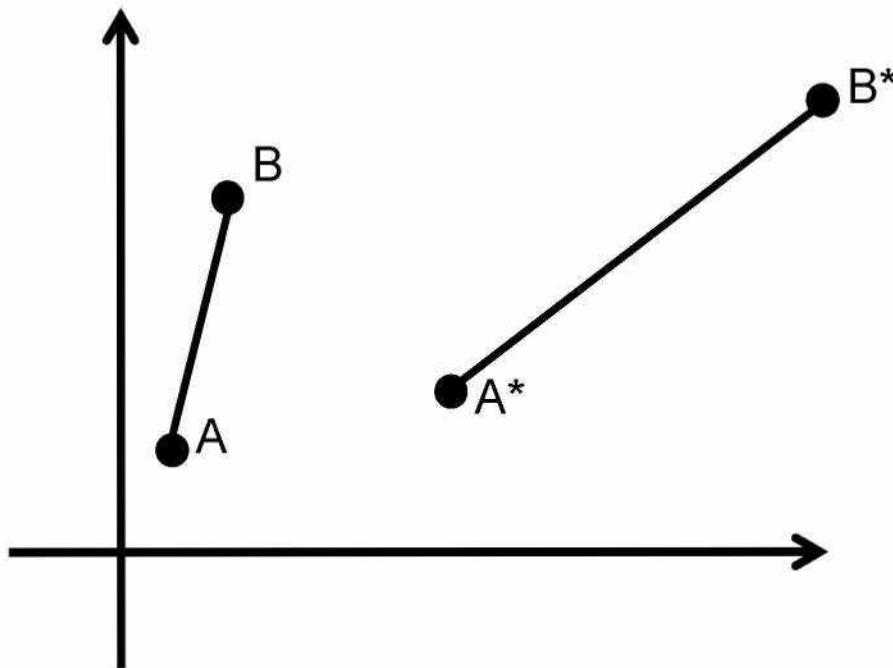
$$\begin{bmatrix} x_1^* & y_1^* \end{bmatrix} = \begin{bmatrix} x_1 & y_1 \end{bmatrix} \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} ax_1 + cy_1 & bx_1 + dy_1 \end{bmatrix}$$

$$\begin{bmatrix} x_2^* & y_2^* \end{bmatrix} = \begin{bmatrix} x_2 & y_2 \end{bmatrix} \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} ax_2 + cy_2 & bx_2 + dy_2 \end{bmatrix}$$

$$\overline{A^*B^*} \quad A^*(x_1^*, y_1^*) \quad B^*(x_2^*, y_2^*)$$

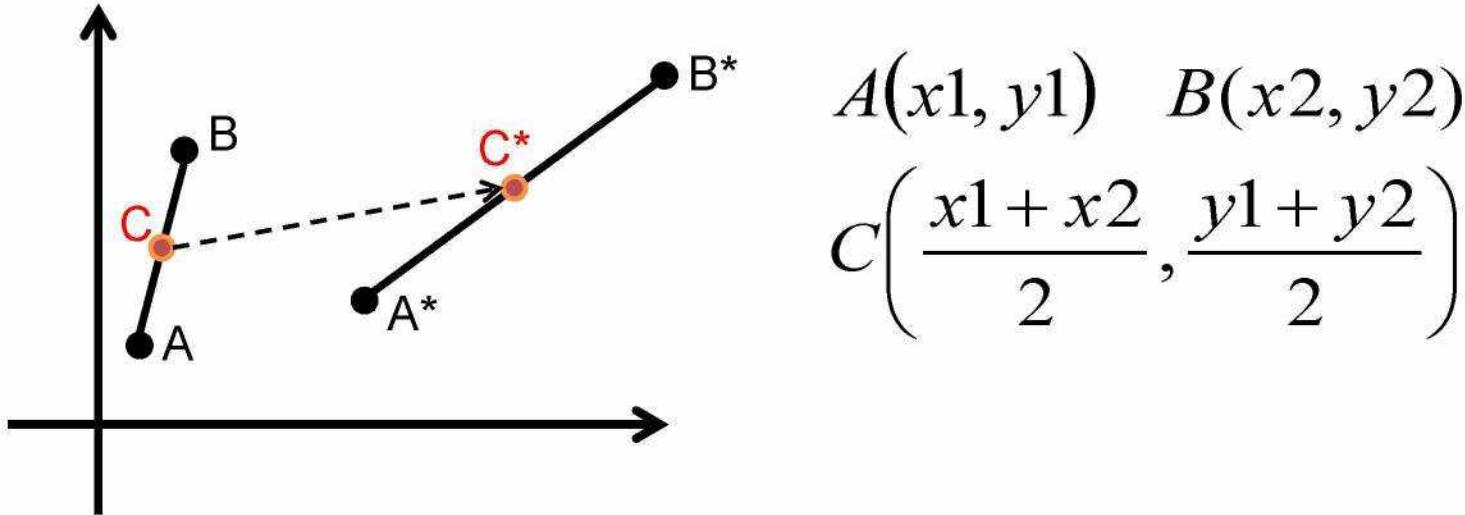


# Матрица преобразования на плоскости. Преобразование прямых линий





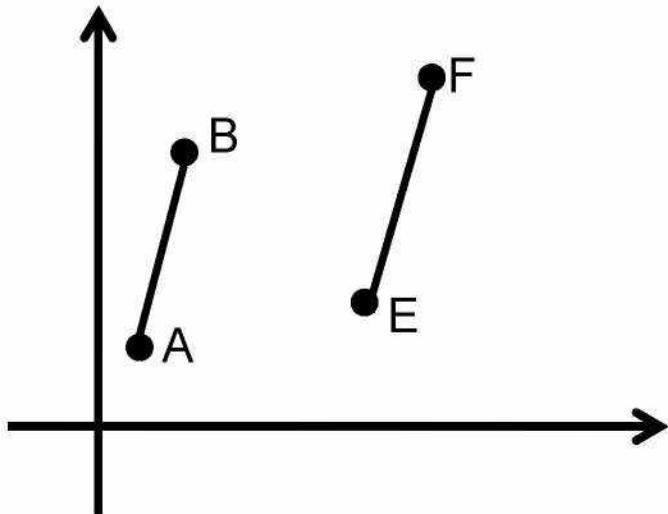
# Матрица преобразования на плоскости. Преобразование средней точки



$$C^* \left( \frac{x1^* + x2^*}{2}, \frac{y1^* + y2^*}{2} \right) =$$
$$\left( \frac{(ax1 + cy1) + (ax2 + cy2)}{2}, \frac{(bx1 + dy1) + (bx2 + dy2)}{2} \right) =$$
$$\left( a \frac{(x1 + x2)}{2} + c \frac{(y1 + y2)}{2}, b \frac{(x1 + x2)}{2} + d \frac{(y1 + y2)}{2} \right)$$



# Матрица преобразования на плоскости. Преобразование параллельных линий

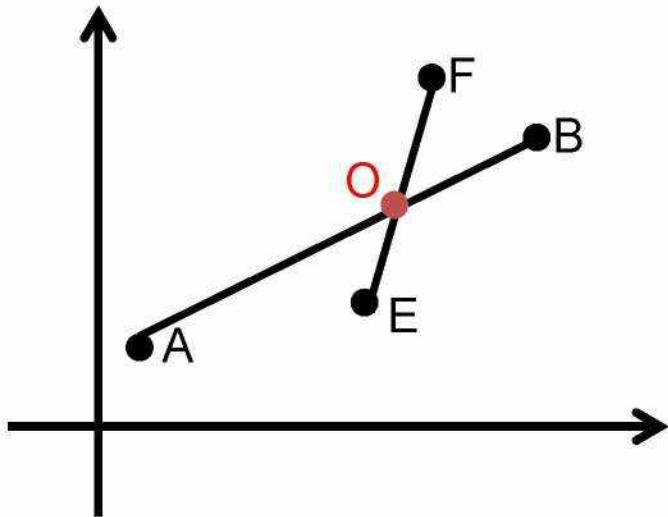


$$A(x_1, y_1) \quad B(x_2, y_2)$$
$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

$$m^* = \frac{y_2^* - y_1^*}{x_2^* - x_1^*} = \frac{(bx_2 + dy_2) - (bx_1 + dy_1)}{(ax_2 + cy_2) - (ax_1 + cy_1)} =$$
$$\frac{b(x_2 - x_1) + d(y_2 - y_1)}{a(x_2 - x_1) + c(y_2 - y_1)} = \frac{b + d\left(\frac{y_2 - y_1}{x_2 - x_1}\right)}{a + c\left(\frac{y_2 - y_1}{x_2 - x_1}\right)} = \frac{b + dm}{a + cm}$$



# Матрица преобразования на плоскости. Преобразование пересекающихся прямых



$$y = m_1 x + b_1$$

$$y = m_2 x + b_2$$

$$O\left(\frac{b_1 - b_2}{m_2 - m_1}, \frac{b_1 m_2 - b_2 m_1}{m_2 - m_1}\right)$$

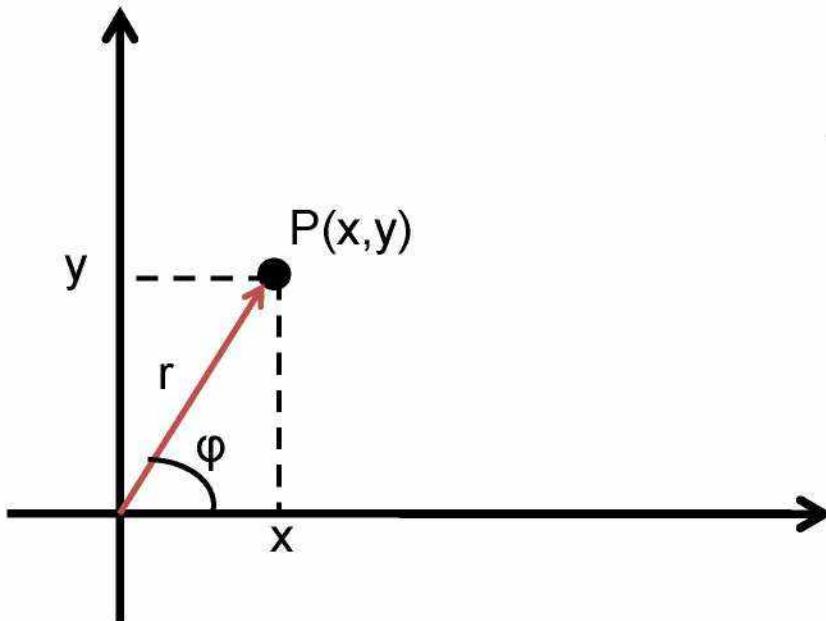
$$m_i^* = \frac{b + dm_i}{a + cm_i}$$

$$O^*\left(\frac{b_1^* - b_2^*}{m_2^* - m_1^*}, \frac{b_1^* m_2^* - b_2^* m_1^*}{m_2^* - m_1^*}\right)$$

$$b_i^* = b_i \frac{ad - bc}{a + cm_i}$$



## Матрица поворота на плоскости



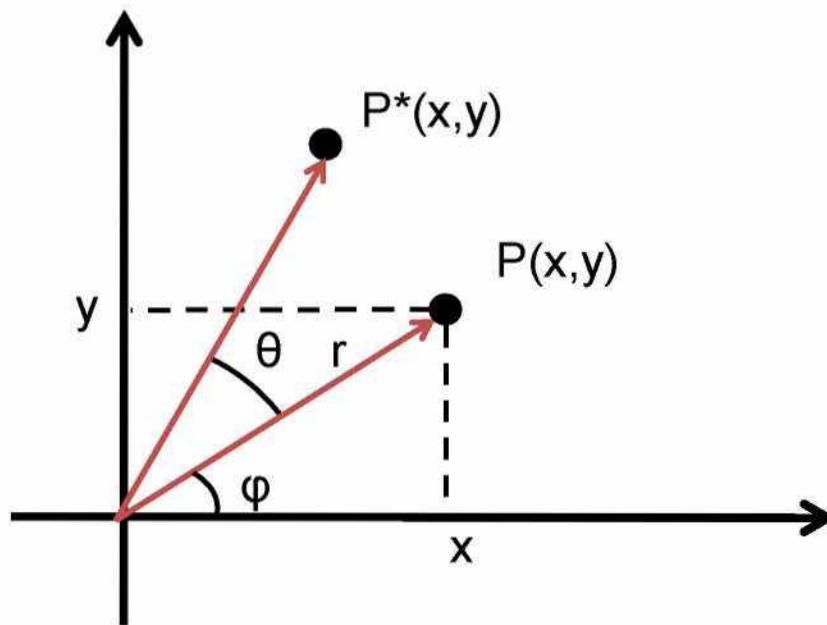
$$P = [x \quad y] = [r \cos \varphi \quad r \sin \varphi]$$

$\theta$  – угол поворота

$$\begin{aligned} P^* &= [x^* \quad y^*] = [r \cos(\varphi + \theta) \quad r \sin(\varphi + \theta)] = \\ &= [r(\cos \varphi \cos \theta - \sin \varphi \sin \theta) \quad r(\sin \varphi \cos \theta + \cos \varphi \sin \theta)] = \\ &= [x \cos \theta - y \sin \theta \quad y \cos \theta + x \sin \theta] \end{aligned}$$



## Матрица поворота на плоскости



$$P = \begin{bmatrix} x & y \end{bmatrix} = \begin{bmatrix} r \cos \varphi & r \sin \varphi \end{bmatrix}$$

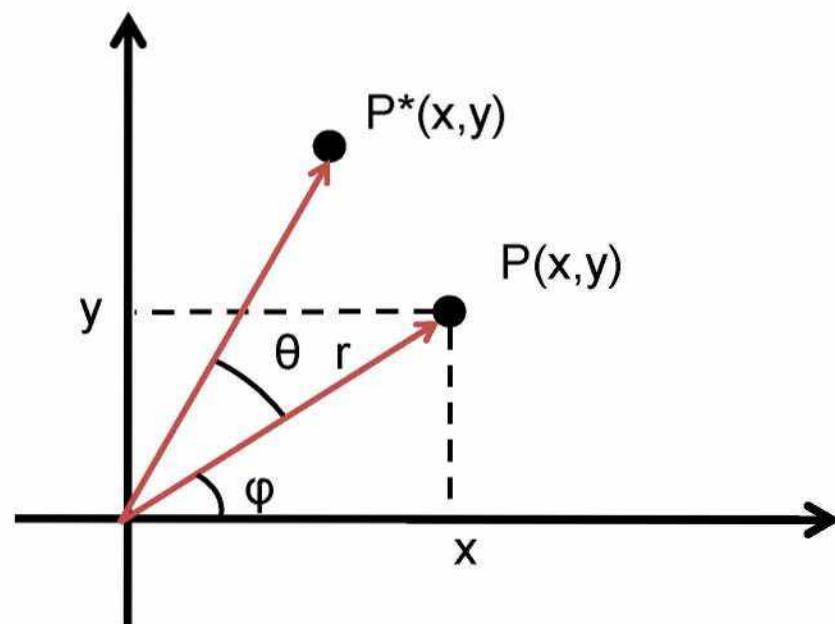
$\theta$  – угол поворота

$$P^* = \begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x \cos \theta - y \sin \theta & y \cos \theta + x \sin \theta \end{bmatrix}$$

$$T = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \quad P^* = P \cdot T$$



## Матрица поворота на плоскости.



$$P = \begin{bmatrix} x & y \end{bmatrix} = \begin{bmatrix} r \cos \varphi & r \sin \varphi \end{bmatrix}$$

θ – угол поворота

$$P^* = \begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x \cos \theta - y \sin \theta & y \cos \theta + x \sin \theta \end{bmatrix}$$

$$T^{-1} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

$$P = P^* \cdot T^{-1}$$



# Матрица перемещения и однородные координаты

Очевидно, что перенос, в отличие от масштабирования и поворота, реализуется с помощью сложения.

$$x^* = ax + cy + M$$

$$y^* = bx + dy + N$$

Вводить константы переноса внутрь структуры общей матрицы размером  $2 \times 2$  не представляется возможным. Эту проблему можно решить за счет введения третьей компоненты в векторы точек и перехода к **однородным координатам**.

$$\begin{bmatrix} x & y \end{bmatrix} \rightarrow \begin{bmatrix} x' & y' & w \end{bmatrix} \rightarrow \begin{bmatrix} x'/w & y'/w & 1 \end{bmatrix} \rightarrow \begin{bmatrix} x & y & 1 \end{bmatrix}$$



## Однородные координаты

Двумерный вектор  $[x \ y]$  в однородных координатах записывается в виде  $[wx \ wy \ w]$ , где  $w \neq 0$ .

Число  $w$  называется масштабным множителем. Для того, чтобы из вектора, записанного в однородных координатах, получить вектор в обычных координатах необходимо разделить первые две координаты на третью.

$$[wx \ wy \ w] \rightarrow [wx / w \ wy / w \ w / w] \rightarrow [x \ y \ 1].$$



# Однородные координаты

Однородные координаты - это математический механизм, связанный с определением положения точек в пространстве. Привычный аппарат декартовых координат не подходит для решения некоторых важных задач.



# Однородные координаты

- В декартовых координатах невозможно описать бесконечно удаленную точку.
- С точки зрения алгебраических операций, декартовы координаты не позволяют провести различия между точками и векторами.
- Невозможно использовать унифицированный механизм работы с матрицами для выражения преобразований точек. С помощью матриц  $2 \times 2$  можно описать поворот и масштабирование, однако описать перенос ( $x' = x + M$ ) нельзя.



# Однородные координаты. Бесконечно удаленные точки

$$\begin{bmatrix} wx & wy & w \end{bmatrix} \rightarrow \begin{bmatrix} wx / w & wy / w & w / w \end{bmatrix} \rightarrow \begin{bmatrix} x & y & 1 \end{bmatrix}.$$

При  $w=1$  эти координаты описывают точку с конечными координатами  $(x,y)$ ,

При  $w=0$  эти координаты описывают точку, бесконечно удаленную в направлении  $(x,y)$ .



# Однородные координаты

Однородными координатами точки

$$P = (x_1, \dots, x_n) \in R^n$$

называются координаты

$$P_{\text{одн}} = (wx_1, \dots, wx_n, w) \in R^{n+1}$$

в которых хотя бы один элемент отличен от нуля.

Преобразование из однородных координат в декартовы однозначно.

Преобразование из декартовых координат в однородные однозначным не является.



# Однородные координаты

Геометрическая интерпретация.

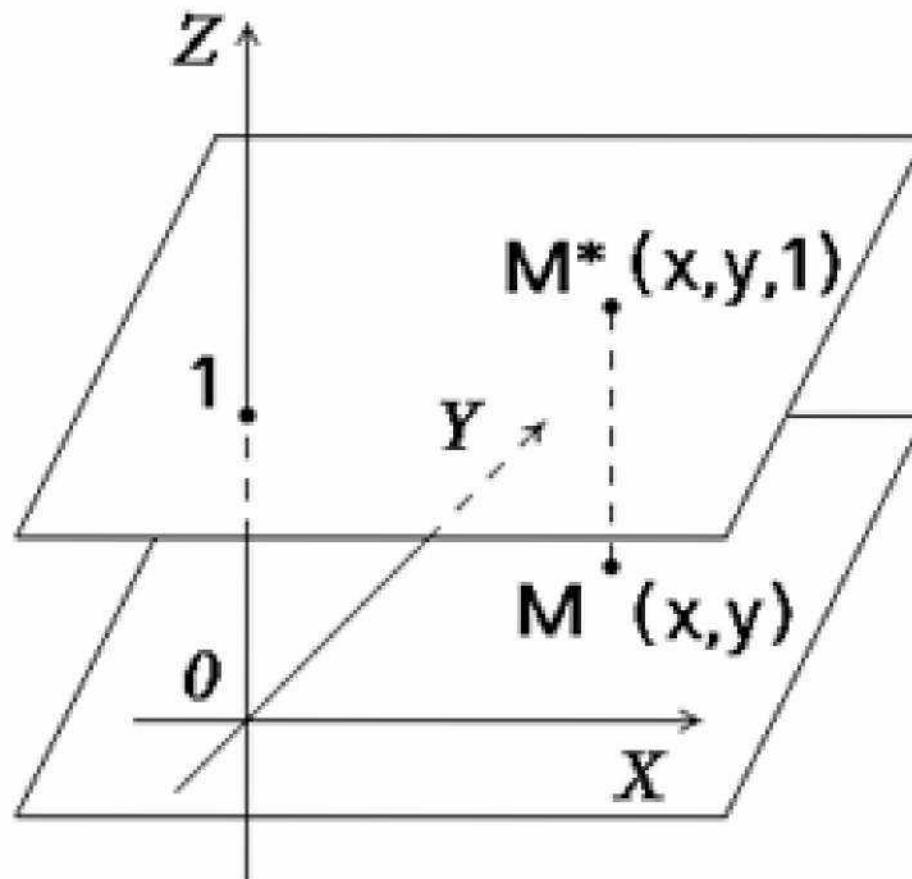
$(wx, wy, w)$  – однородные координаты.

Введение третьей координаты можно трактовать, как переход в трехмерное пространство.

Рассматриваем точку с координатами  $(wx, wy, w)$  в трехмерном пространстве. Прямая, соединяющая эту точку с началом координат, пересечет плоскость  $z=1$  в точке  $(x, y, 1)$ , что эквивалентно проекции на плоскость  $z=1$ .

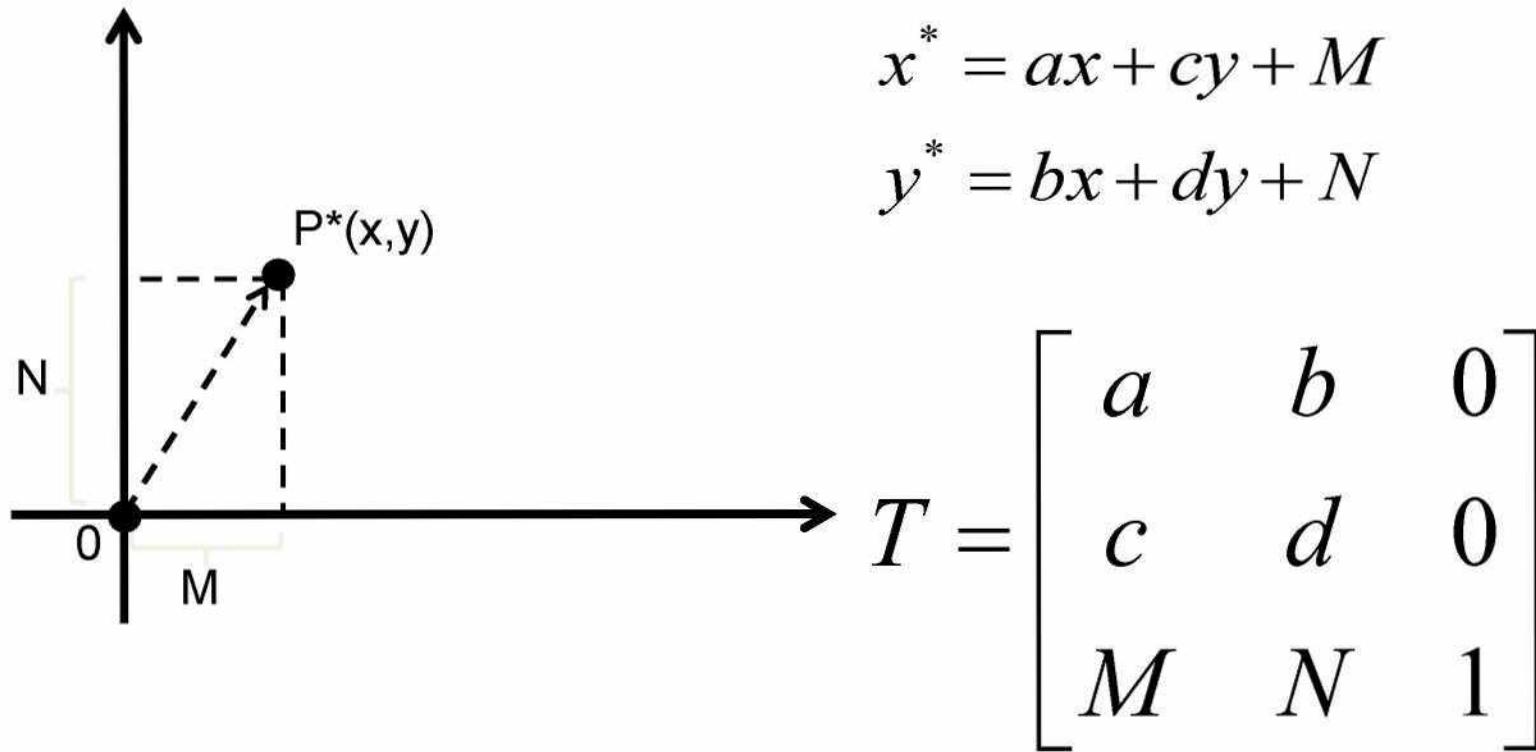


# Однородные координаты





# Матрица перемещения и однородные координаты



$$\begin{bmatrix} x & y \end{bmatrix} \rightarrow \begin{bmatrix} x' & y' & w \end{bmatrix} \rightarrow \begin{bmatrix} x & y & 1 \end{bmatrix}$$



# Матрицы преобразования для однородных координат

$$T_{scale} = \begin{bmatrix} a & 0 & 0 \\ 0 & d & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T_{translation} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ M & N & 1 \end{bmatrix}$$

$$T_{rotate} = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



# Аффинные преобразования на плоскости

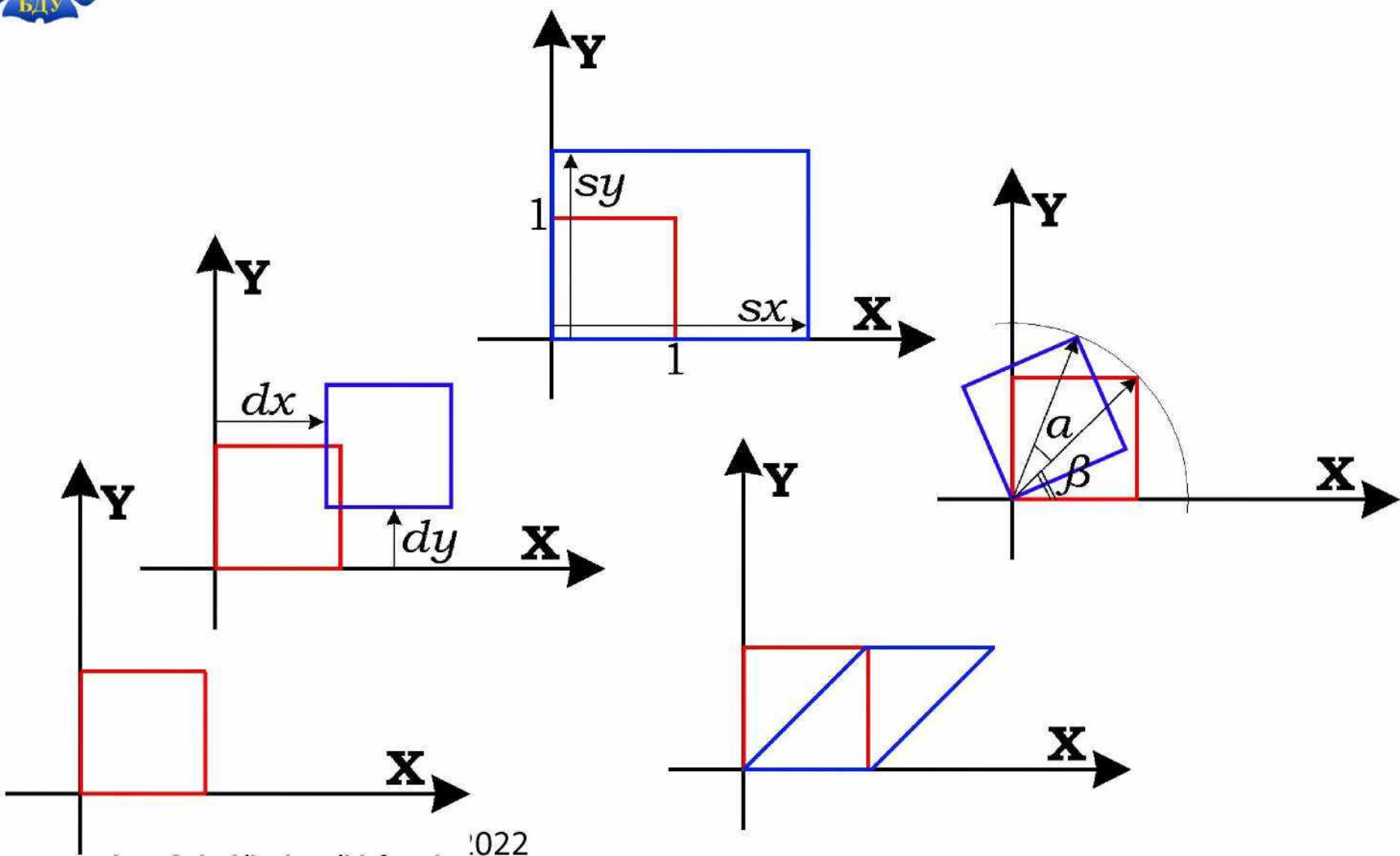
Преобразование плоскости называется **аффинным**, если оно взаимно однозначно и образом любой прямой является прямая. Преобразование называется **взаимно однозначным**, если оно разные точки переводит в разные, и в каждую точку переходит какая-то точка.

Свойства:

1. Любое аффинное преобразование может быть представлено как последовательность операций из числа указанных простейших: перенос, масштабирование и поворот.
2. Сохраняется прямизна линий (линия остается прямой), параллельность прямых, отношение длин отрезков, лежащих на одной прямой и отношение площадей фигур.



# Аффинные преобразования на плоскости





# Аффинные преобразования на плоскости. Примеры

Поворот вокруг произвольной точки D (M,N) на угол а.

- 1) Перемещение точки D в начало координат (а значит и поворачиваемого объекта): преобразование T1.
- 2) Поворот вокруг начала координат на нужный угол: преобразование T2.
- 3) Обратное перемещение в исходный центр вращения: преобразование T3.



# Аффинные преобразования на плоскости. Примеры

$$T1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -M & -N & 1 \end{bmatrix} \quad T2 = \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ M & N & 1 \end{bmatrix} \quad T = T1 \cdot T2 \cdot T3$$



# Аффинные преобразования на плоскости. Примеры

Отражение относительно произвольной прямой.

- 1) Перемещение прямой и объекта таким образом, чтобы прямая прошла через начало координат.
- 2) Поворот прямой и объекта вокруг начала координат до совпадения с одной из осей.
- 3) Отражение относительно координатной оси.
- 4) Обратный поворот вокруг начала координат.
- 5) Обратное перемещение.



# Аффинное преобразование в пространстве

Аффинное преобразование это такое преобразование, которое сохраняет параллельность линий, но не обязательно углы или длины.

$$x' = a_{xx}x + a_{xy}y + a_{xz}z + b_x$$

$$y' = a_{yx}x + a_{yy}y + a_{yz}z + b_y$$

$$z' = a_{zx}x + a_{zy}y + a_{zz}z + b_z$$

Подмножество аффинных преобразований:  
жесткие аффинные преобразования – сохраняют расстояния и углы.



# Матрицы преобразования в пространстве

Геометрические преобразования в трехмерном пространстве осуществляются так же, как и на плоскости. Таким же образом определяются основные преобразования, и матрицы сложных преобразований получаются умножением соответствующих простых матриц.



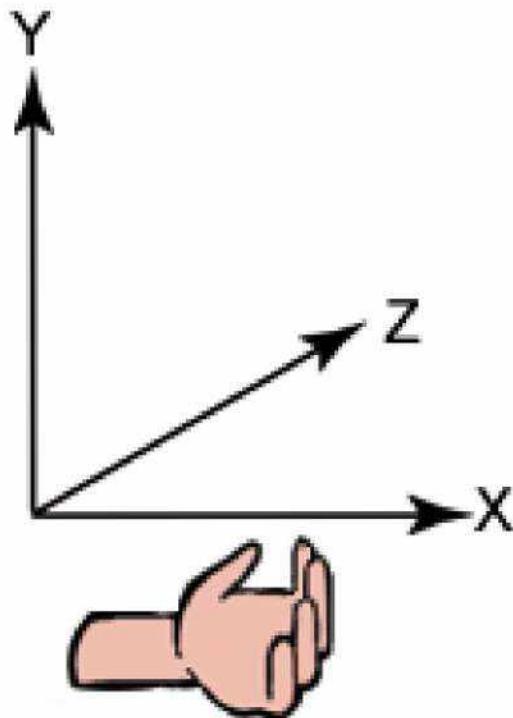
# Матрицы преобразования в пространстве

- преобразование масштабирования;
- преобразование переноса;
- преобразования поворотов (вокруг координатных осей, вокруг произвольных осей).

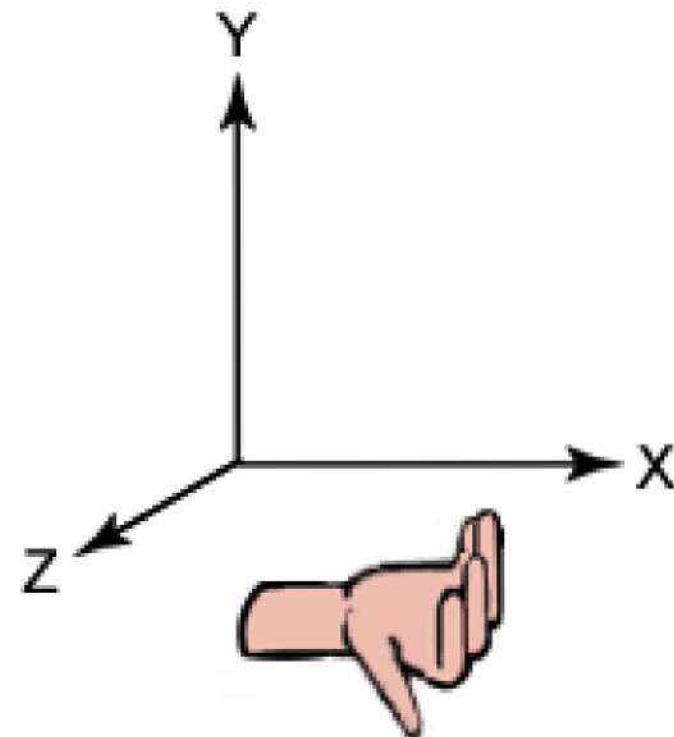


# Системы координат

Левосторонняя система  
координат



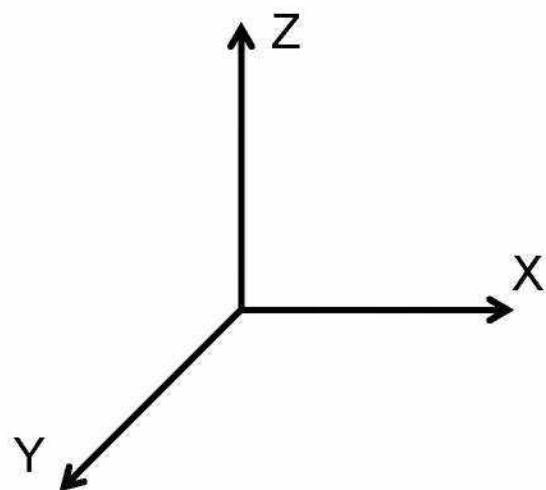
Правосторонняя система  
координат



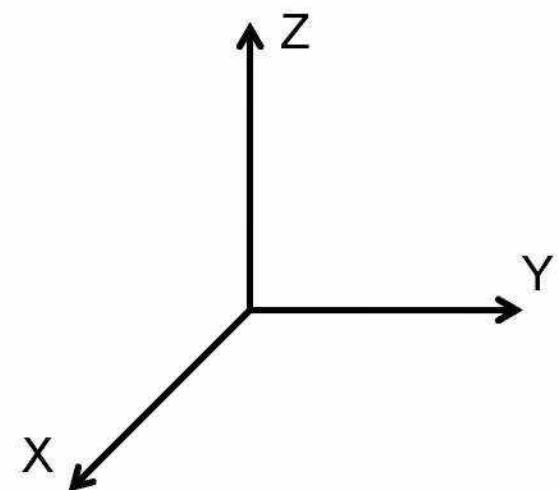


# Системы координат

Левосторонняя система  
координат



Правосторонняя система  
координат





# Матрицы преобразования в пространстве

$[x \quad y \quad z \quad 1]$  - однородные координаты в пространстве

$$T = \begin{bmatrix} a & b & c & p \\ d & e & f & q \\ g & i & j & r \\ l & m & n & s \end{bmatrix} \quad T = \begin{bmatrix} 3 \times 3 & : & 3 \times 1 \\ \dots & : & \dots \\ 1 \times 3 & : & 1 \times 1 \end{bmatrix}$$



# Трехмерное масштабирование

$$X \cdot T = [x \quad y \quad z \quad 1] \cdot \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & e & 0 & 0 \\ 0 & 0 & j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [ax \quad ey \quad jz \quad 1]$$



# Трехмерное общее масштабирование

$$X \cdot T = [x \quad y \quad z \quad 1] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & s \end{bmatrix} = [x \quad y \quad z \quad s]$$

$$[x \quad y \quad z \quad s] \rightarrow [x/s \quad y/s \quad z/s \quad 1]$$



# Трехмерные сдвиги

$$X \cdot T = [x \quad y \quad z \quad 1] \cdot \begin{bmatrix} 1 & b & c & 0 \\ d & 1 & f & 0 \\ g & i & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \\ [x + yd + gz \quad y + bx + iz \quad z + cx + fy \quad 1]$$



# Трехмерный перенос

$$X \cdot T = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix} = \\ \begin{bmatrix} x + t_x & y + t_y & z + t_z & 1 \end{bmatrix}$$



# Трехмерное вращение вокруг координатных осей

$$T_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_z = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



# Трехмерное отражение

$$T_{Oxy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_{Oyz} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{Oxz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



# Композиция преобразований

Более сложные преобразования описываются также, как и для и плоскости. Предположим, например, что требуется выполнить преобразование поворота относительно оси  $x$  на угол  $\alpha$  с последующим переносом на вектор  $T(t_x, t_y, t_z)$ . Матрица полного преобразования равна произведению матриц, его составляющих:

$$M = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha & \sin \alpha & 0 \\ 0 & -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha & \sin \alpha & 0 \\ 0 & -\sin \alpha & \cos \alpha & 0 \\ t_x & t_y & t_z & 1 \end{vmatrix}$$



# Сложные трехмерные преобразования



Белорусский Государственный Университет  
Факультет прикладной математики и информатики

# Компьютерная графика

Василевский Константин  
Викторович

17 лекций,  
17 лабораторных занятий



# Темы лекционных занятий

- Цвет в компьютерной графике. Цветовые модели.
- Формы компьютерного представления двумерных изображений. Файловые форматы.
- Сжатие графических файлов. Основные алгоритмы сжатия.
- Задача растеризации. Базовые растровые алгоритмы.
- Основы обработки цифровых изображений.



# Темы лекционных занятий

- Основные алгоритмы вычислительной геометрии на плоскости.
- Математические основы машинной графики. Преобразование координат на плоскости. Преобразование координат в пространстве. Основы проецирования.
- Модели освещения.
- Визуализация научных данных.
- Векторная графика.
- Геоинформационные технологии и системы.



# Методические материалы, лекции, тесты и условия лабораторных работ

**<https://edufpmi.bsu.by/>**

Факультет прикладной математики и  
информатики >> Компьютерная графика

Кодовое слово: KG2022



# Лекция 1. Цвет в компьютерной графике. Цветовые модели

- Свет и цвет
- Физическая природа света и цвета
- Излученный и отраженный свет
- Особенности восприятия цвета человеком
- Измерение цвета
- Цветовые модели: определение и классификация
- Цветовая модель RGB
- Цветовые модели CMY и CMYK
- Интуитивные цветовые модели
- Цветовая модель CIE XYZ
- Однородные цветовые пространства



# Свет и цвет

БДУ

Свет как физическое явление представляет собой поток электромагнитных волн различной длины и амплитуды.

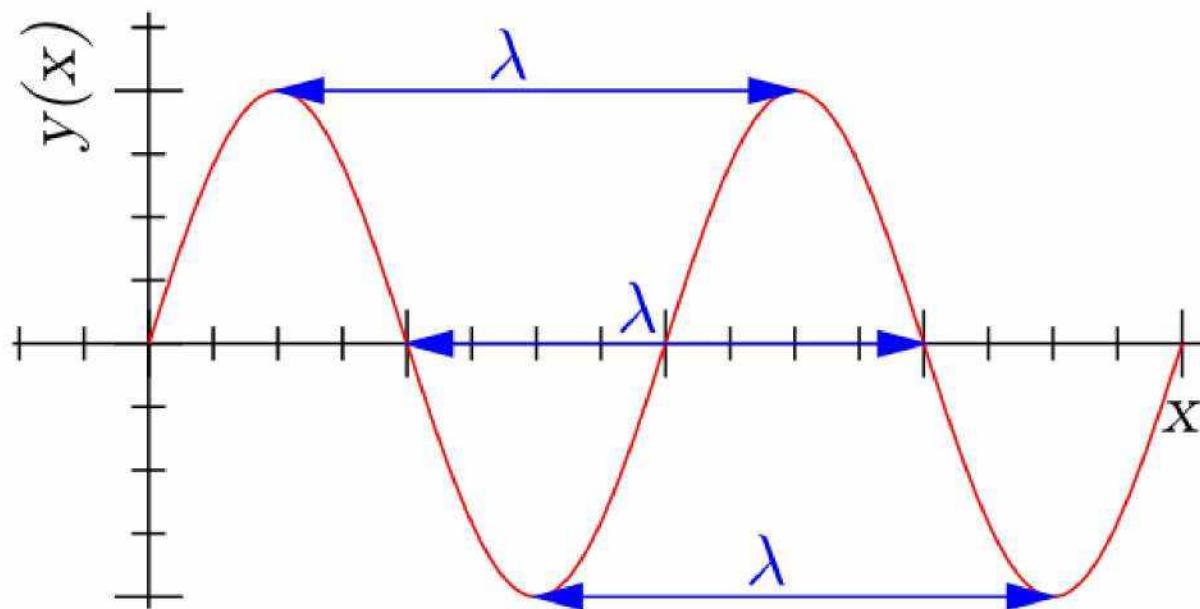
Цвет -- это ощущение, которое возникает в сознании человека при воздействии на его зрительный аппарат электромагнитного излучения с длиной волны в диапазоне от 380 до 740 нм. Эти ощущения могут быть вызваны и другими причинами: болезнь, удар, мысленная ассоциация, галлюцинации, и др.

Цвет – качественная субъективная характеристика электромагнитного излучения оптического диапазона, определяемая на основании возникающего физиологического зрительного ощущения.

Напоминание: Электромагнитное излучение имеет волновую природу, т.е. распространяется в пространстве в виде периодических колебаний (волн), совершаемых им с определенной амплитудой и частотой. Если представить такую волну в виде графика, то получится синусоида. Расстояние между двумя соседними вершинами этой синусоиды называется длиной волны и измеряется в нанометрах (нм) и представляет собой расстояние, на которое распространяется свет за период одного колебания.



Напоминание: Электромагнитное излучение имеет волновую природу, т.е. распространяется в пространстве в виде периодических колебаний (волн), совершаемых им с определенной амплитудой и частотой. Если представить такую волну в виде графика, то получится синусоида.



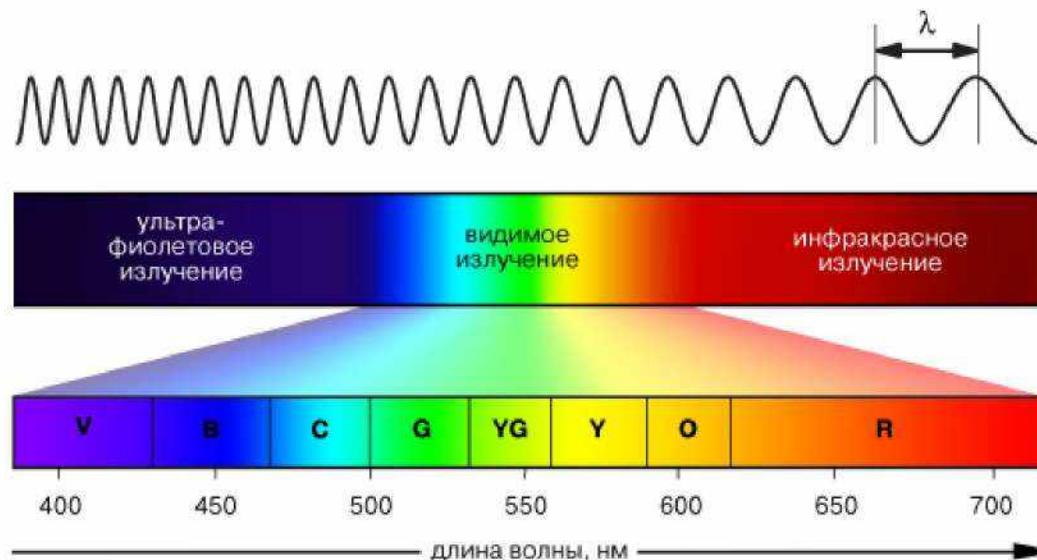


# Видимый свет

Видимый свет состоит из спектрального распределения электромагнитной энергии с длинами волн в диапазоне 380-740 нм.

Монохроматическое излучение – электромагнитное излучение, обладающее очень малым разбросом частот, в идеале – одной частотой (длиной волны).

Цвет излучений, длины волн которых расположены в диапазоне видимого света в определенных интервалах вокруг длины какого-либо монохроматического излучения, называются спектральными цветами.





# Видимый свет

**Видимый свет** состоит из спектрального распределения электромагнитной энергии с длинами волн в диапазоне 380-740 нм.

Цвет	Диапазон длин волн, нм	Диапазон частот, ТГц
Фиолетовый	380-440	790-680
Синий	440-485	680-620
Голубой	485-500	620-600
Зеленый	500-565	600-530
Желтый	565-590	530-510
Оранжевый	590-625	510-480
Красный	625-740	480-405

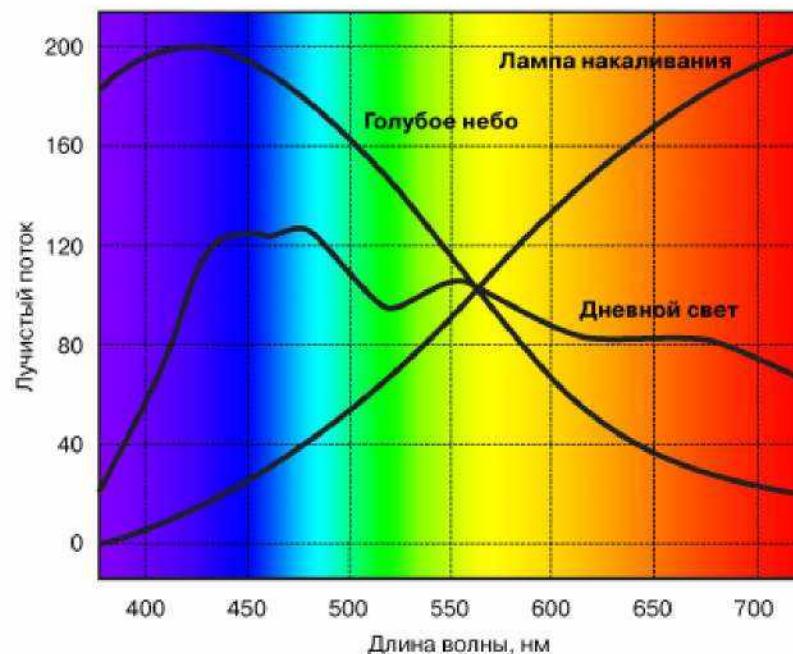


# Спектр

БДУ

В природе излучение от различных источников света либо предметов редко является монохроматичным, т.е. представленным излучением только одной определенной длины волны, и имеет довольно сложный спектральный состав, т.е. в нем присутствуют излучения самых различных длин волн.

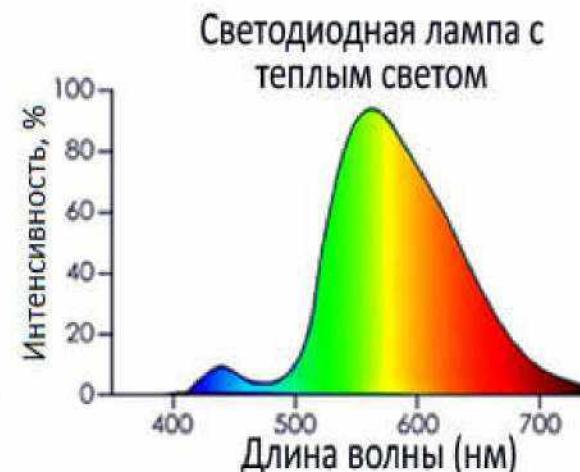
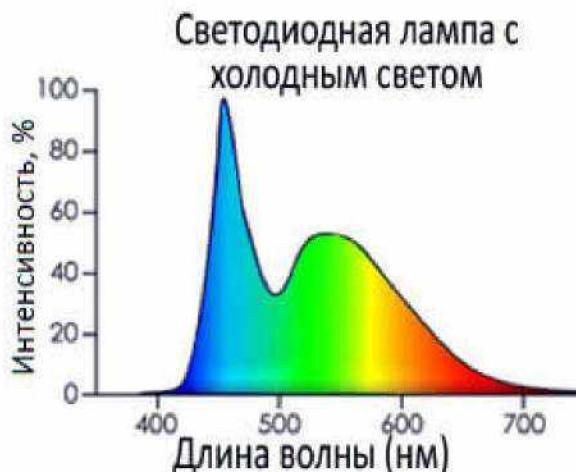
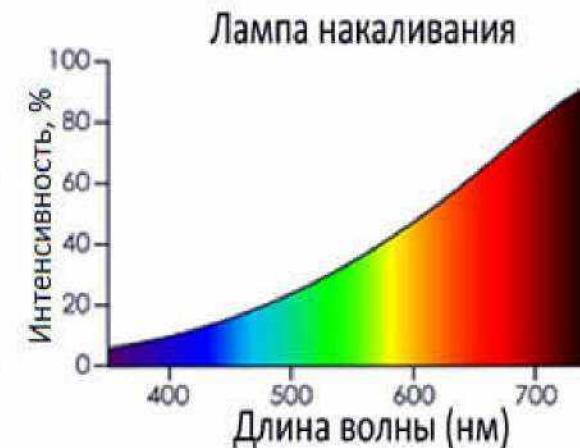
Если представить эту картину в виде графика, где по оси абсцисс будет отложена длина волны, а по оси ординат — интенсивность, то мы получим зависимость, называемую цветовым спектром излучения или просто спектром цвета, или спектральным составом цвета. Интенсивность (или яркость) света представляет собой сумму энергий всех составляющих цветового спектра.





# Спектр

## Спектры различных источников





# Спектр

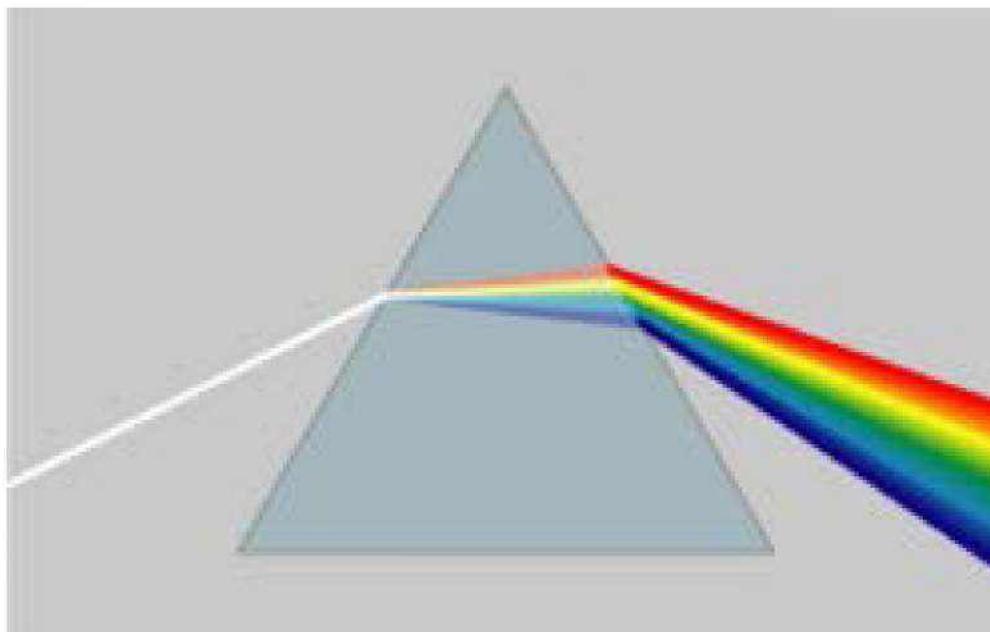
Спектр монохроматического излучения





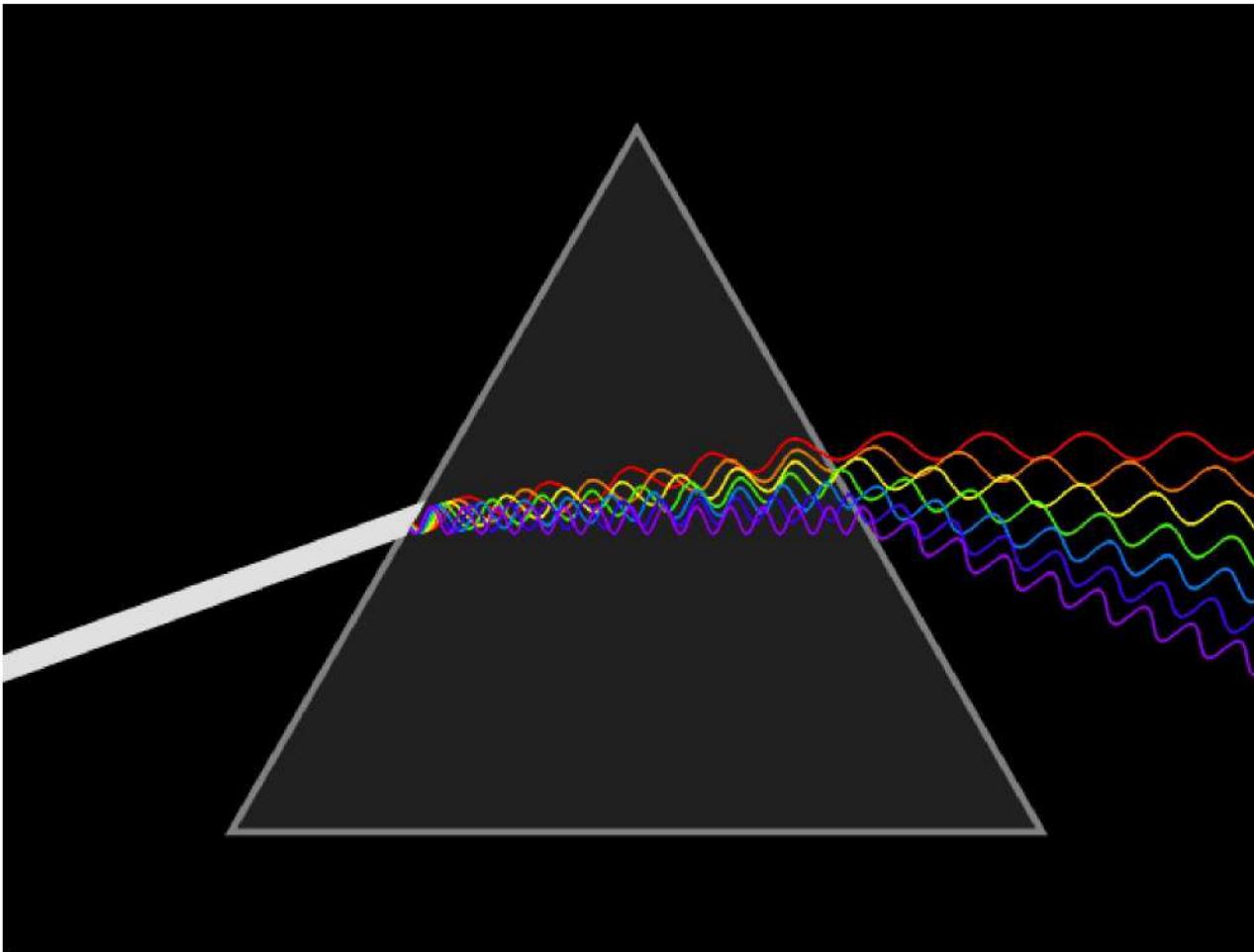
# Свет

Дисперсия света -- зависимость показателя преломления вещества от длины волн  $\lambda$  света.





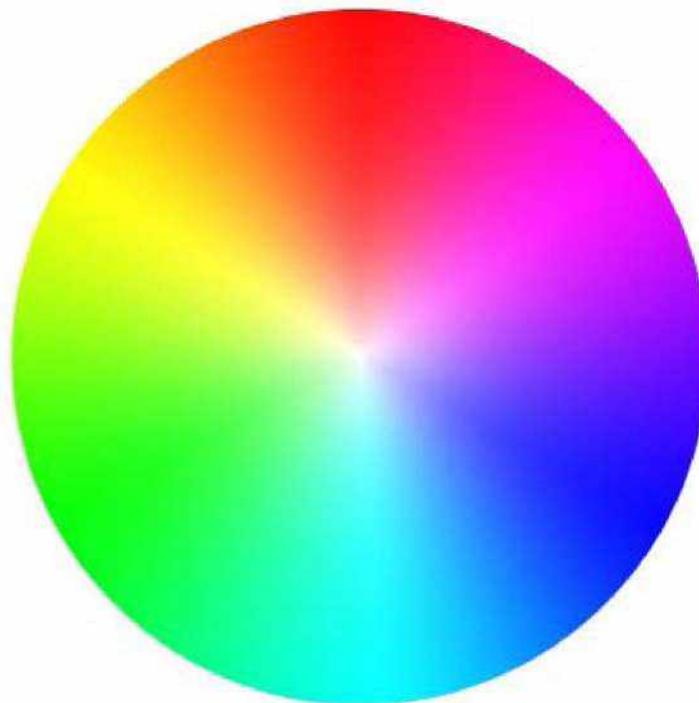
# Дисперсия света





# Цветовой круг

Если замкнуть линию спектра с использование пурпурного цвета, то получится цветовой круг.





# Свет и цвет

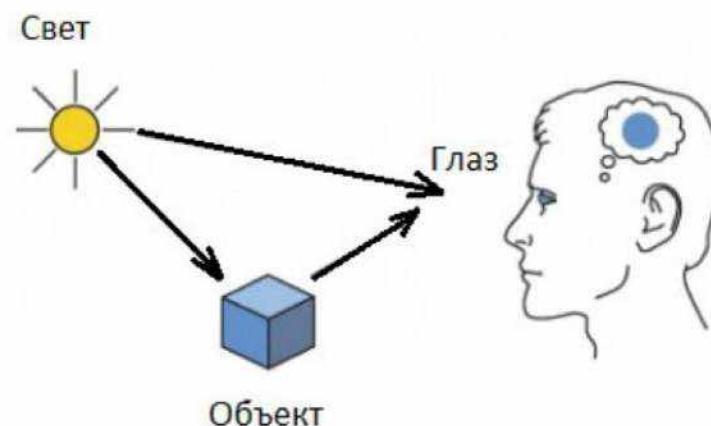
## Излученный и отраженный свет

Все что мы можем увидеть в окружающем мире – это либо излученный свет, либо отраженный свет.

**Излученный свет** – это свет, испускаемый активным источником (Солнце, лампа, экран монитора).

Преобладание в его спектральном составе длин волн определенного диапазона даст нам ощущение доминирующего в нем цвета.

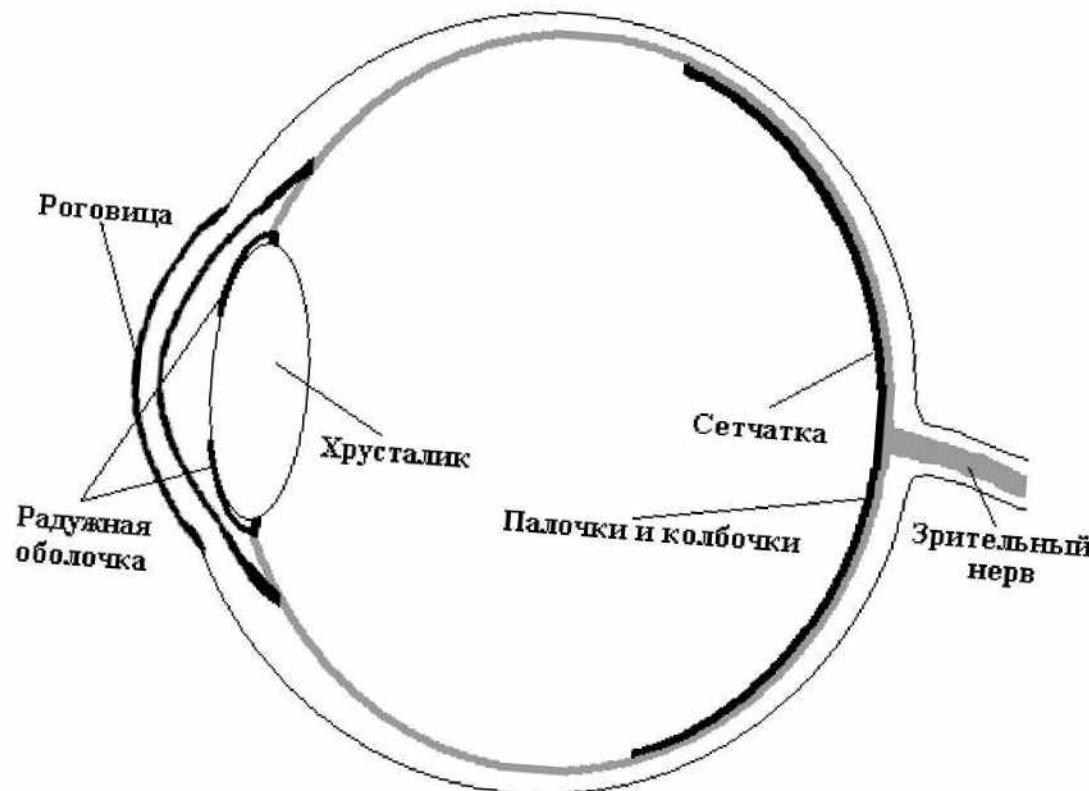
**Отраженный свет** возникает при отражении поверхностью световых волн, падающих на него от источника света.





# Свет и цвет

Восприятие света человеком



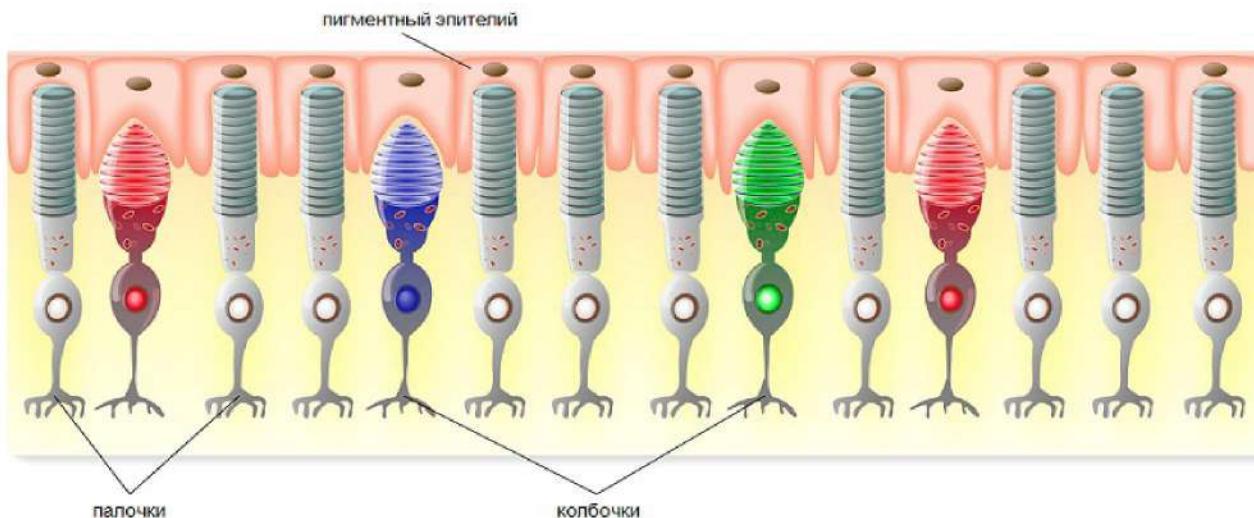


# Свет и цвет

## Восприятие света человеком

Развитие способности к ощущению цвета эволюционно обеспечивалось формированием специальной системы цветового зрения, включающей два типа светочувствительных фоторецепторов: колбочки, сосредоточенные главным образом в центральной ямке и расположенные в основном по периферии сетчатки, и палочки, не обладающие преимущественной чувствительностью к какому-либо спектральному цвету и играющие главную роль в создании ахроматических зрительных образов. В каждом глазу 6 миллионов колбочек и 120 миллионов палочек (т.е. примерно 250 миллионов рецепторов на два глаза).

Палочки действуют в основном при слабом освещении и предоставляют информацию лишь о яркости. Колбочки, эффективно действующие только при достаточно ярком свете, позволяют глазу различать цвета.





# Свет и цвет

## Восприятие света человеком

Колбочки делятся на три типа, которые называют либо как B, G и R, либо как S, M и L. Пики их чувствительности приходятся примерно на 440 нм, 545 нм и 580 нм (для "усредненного" наблюдателя), причем чувствительности разного типа колбочек очень отличаются.



Международная комиссия по освещению (МКО), или  
Commission internationale de l'éclairage (CIE) – международная  
организация, ведущая разработку технических стандартов в  
области света, освещения, цвета и цветовых пространств.  
Создана в 1913 году.



International Commission on Illumination  
Commission Internationale de l'Eclairage  
Internationale Beleuchtungskommission



# Свет и цвет

БДУ

В 1923 году СИЕ провела серию экспериментов по измерению ощущений яркости путем визуального сравнения.

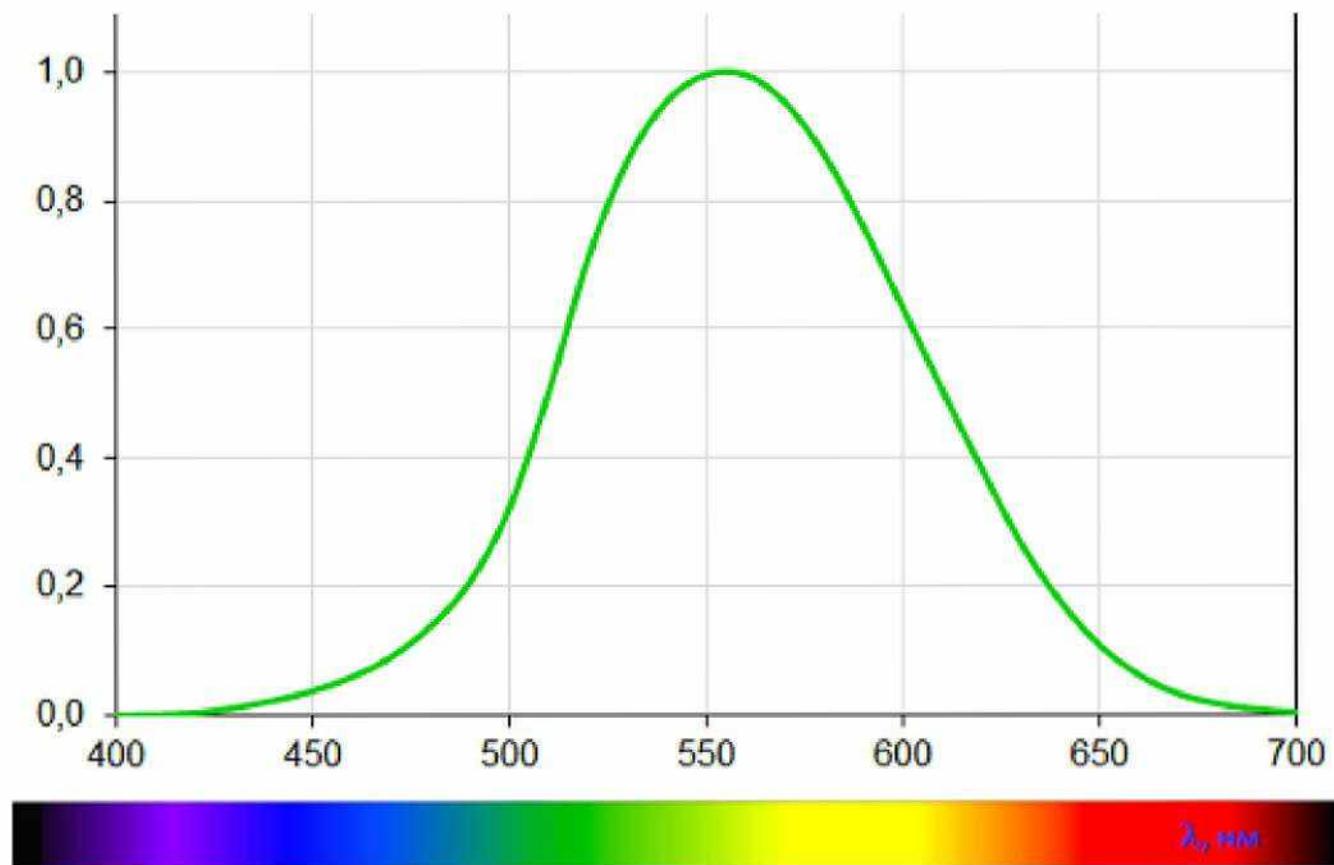
Суть эксперимента: каждый наблюдатель, управляя интенсивностью излучения, должен был уравнять яркости двух монохроматических потоков, измерив при этом их мощности. В результате было выяснено следующее: чтобы уравнять по яркости монохроматическое излучение с длиной волны 555 нм мощностью один ватт, нужно использовать двухватное излучение с длиной волны 512 нм.

Результатом серии таких экспериментов для всего видимого диапазона является кривая спектральной чувствительности глаза (другие названия: кривая спектральной световой эффективности, кривая «видности»).



# Свет и цвет

## Кривая спектральной чувствительности

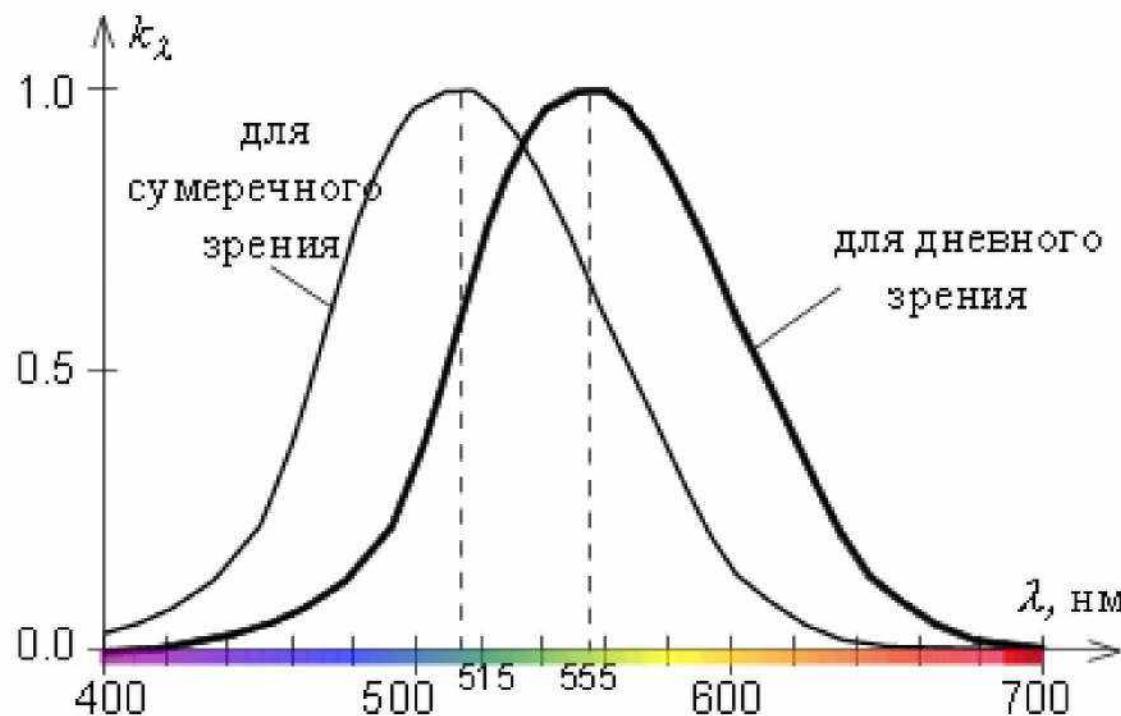




# Свет и цвет

## Кривая спектральной чувствительности

При уменьшении освещенности кривая спектральной чувствительности глаза сдвигается влево, и в сумерках максимум спектральной чувствительности глаза приходится на значение  $\lambda=515$  нм. Это явление называется эффектом Пуркинье.





# Свет и цвет

БДУ

## Как можно «измерить» цвет?

Цвета нам нужно воспроизводить – на бумаге, мониторе или где-нибудь еще, цвета нужно сравнивать и корректировать. Для всего этого цвета нужно уметь строго описывать, т.е. задавать и определять координаты того или иного цвета в некотором пространстве признаков.

Исходя из определения цвета как ощущения, его измерения должны быть измерениями именно цветовых ощущений человека. Любые методы, не основанные на таких измерениях, — бессмысленны. Однако все люди воспринимают цвет немного по-разному. Что же, собственно, измерять?

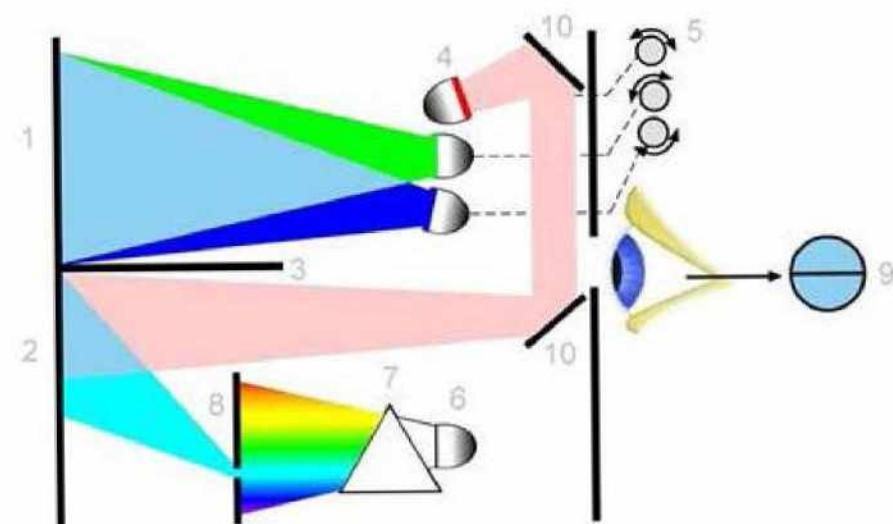
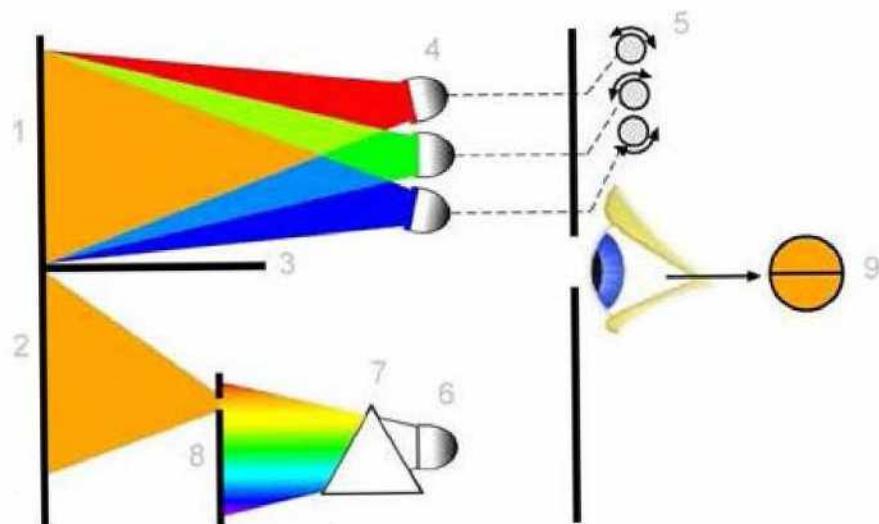
В 20-х годах XX века, независимо друг от друга, учёные Гилд и Райт провели серию экспериментов с целью изучения цветового зрения человека.

В качестве основных цветов выбраны красный, зеленый и синий (длины волн 645.16 нм, 526.32 нм и 444.44 нм)



# Свет и цвет

Как можно «измерить» цвет?

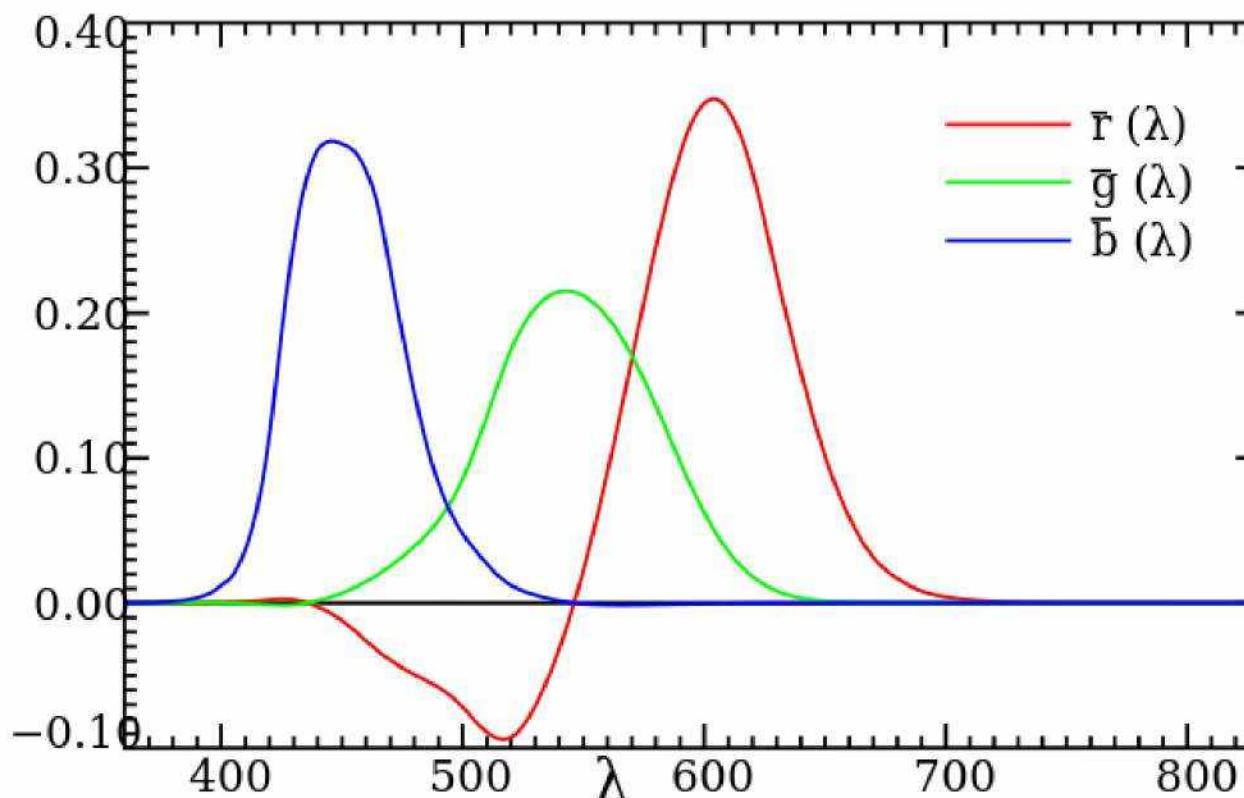




# Свет и цвет

БДУ

## Как можно «измерить» цвет?





# Свет и цвет

БДУ

## Как можно «измерить» цвет?

В экспериментах CIE были получены числовые значения, соответствующие определенным цветовым ощущениям, т.е. цветовые ощущения были измерены.

Измерение цветовых ощущений человека является конечным результатом эксперимента CIE и положено в основу всей современной колориметрии - науки о цветовых измерениях. Физиологическая цветовая координатная система, полученная в результате экспериментов CIE, носит название «CIE RGB». Найти цветовые координаты — это значит найти численное выражение цветового ощущения, то есть измерить цвет.



# Цветовые модели

и цветовые пространства

## Итак, что же такое цветовая модель?

**Цветовая модель** -- математическая модель описания цвета в виде набора чисел, называемых цветовыми координатами.  
(другие названия – цветовая система, цветовая координатная система)

В цветовой модели каждому цвету можно поставить в соответствие строго определенную точку. В этом случае цветовая модель – это просто упрощенное геометрическое представление, основанное на системе координатных осей и принятого масштаба.



# Цветовые модели

По принципу действия цветовые модели можно условно разбить на четыре класса:

- **аддитивные** (RGB), основанные на сложении цветов.

Результирующий цвет является суммой (по яркости) его составляющих, которые добавляются к черному. Так, например, на темный экран светят красным, зеленым и синим лучами, чтобы получить белый. Когда нет ничего, то есть, в основании отсчета, мы видим черный, когда все компоненты находятся на максимуме, мы видим белый.

- **субтрактивные** (CMY, CMYK), основу которых составляет операция вычитания цветов (**субтрактивный синтез**).

Результирующий цвет вычитается (по яркости) из белого. Так, печатая на белом листе красками, мы уменьшаем яркость отраженного света. В этом случае в основании координат мы имеем белый, а в максимуме – черный, складывающийся из цветовых составляющих.

- **перцепционные (интуитивные)** (HSV, HLS), базирующиеся на восприятии.

Здесь принцип формирования цвета основывается не на технической реализации – освещении экрана или печати краской, а на том, как мозг воспринимает цвет. Мы выделяем яркость и цветность, вот эти модели воплощают описанную разницу на практике.

- **искусственные (математические)** (XYZ).



# Цветовые модели

Классификация цветовых моделей по области применения:

- **аппаратно-зависимые** -- модели, используемые в технических средствах ввода-вывода графической информации.

Такие цветовые модели зависят от конкретного вида устройств. Разные мониторы, например, имеют разные границы цветопередачи, то есть, разный цветовой охват и разную яркость и контрастность.

- **аппаратно-независимые** -- модели, не связанные с конкретным воспроизводящим устройством.

Описывают цвет в абстрактных колориметрических терминах. Эти модели предложены международной комиссией по освещению CIE, их широко применяют в теоретических исследованиях и системах управления цветом.



# Цветовые модели. Модель RGB

**RGB** (Red, Green, Blue) -- аддитивная аппаратно-зависимая цветовая модель, описывающая способ кодирования цвета.

При смешении двух лучей основных цветов, результирующий цвет будет светлее составляющих.

В модели RGB обычно используется два формата записи: десятичными числами по компонентам и шестнадцатеричной записью. Например, желтый цвет будет записан так: (255, 255, 0) или #FFFF00.

(255 потому, что обычно изображение записывается в «восьмибитном» формате, то есть, с глубиной цвета 8 бит на канал. На пиксель получится 24 бита).

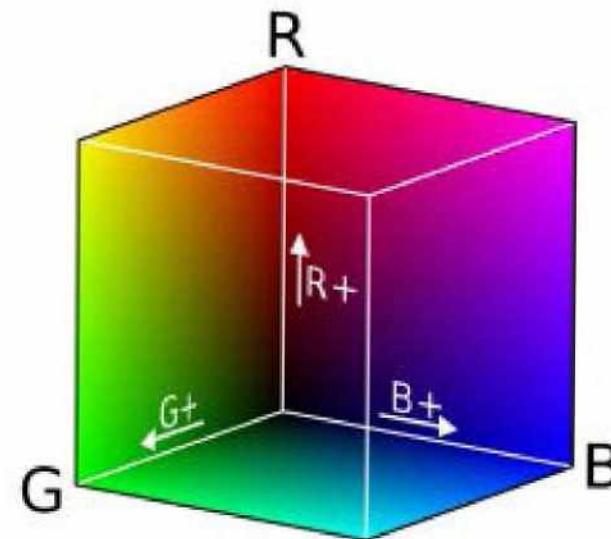
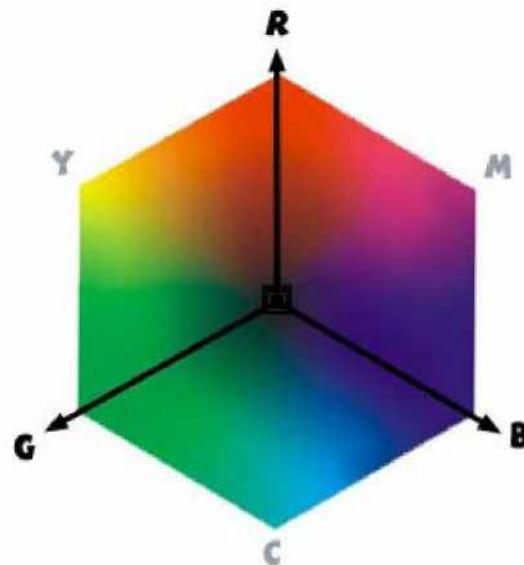
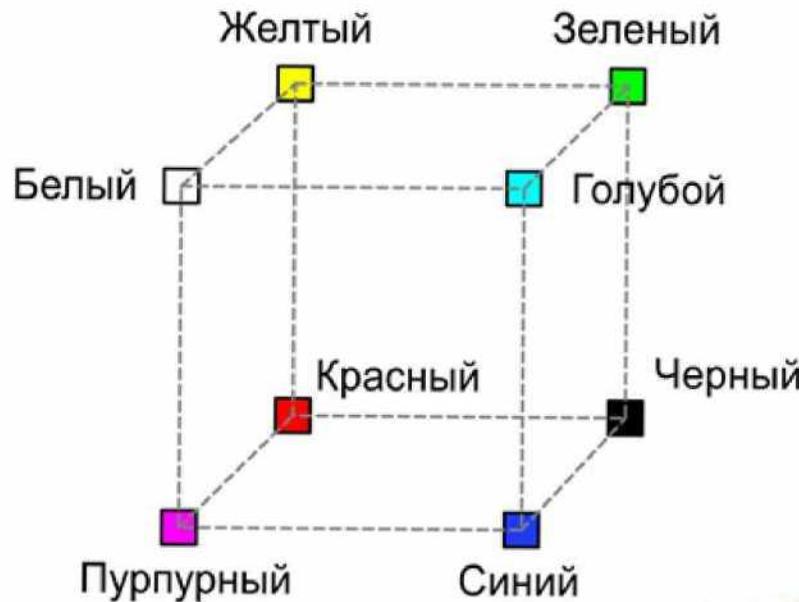
**Область применения:** воспроизведение цвета на мониторах.

В ЭЛТ изображение создается с помощью трех электронных прожекторов, каждый из которых излучает свет своего цвета. На экран нанесен люминофор - вещество, которое светится под воздействием этих прожекторов. Причем люминофор тоже трех видов: один светится от излучения красной пушки, второй - от зеленои, третий - от синей.

Модель является аппаратно- зависимой, так как значения базовых цветов определяются качеством примененного в мониторе люминофора. В результате на разных мониторах одно и то же изображение выглядит неодинаково.



# Цветовые модели. Модель RGB





# Цветовые модели. Модель CMY

**CMY** (Cyan, Magenta, Yellow) -- субтрактивная аппаратно-зависимая цветовая модель, описывающая способ кодирования цвета.

При смешении двух субтрактивных цветов результат затемняется (в модели RGB было наоборот).

В модели CMY каждый из цветов описывается градацией от 0 до 100 или от 0 до 1.

Модель CMY обладает сравнительно с RGB меньшим цветовым охватом.

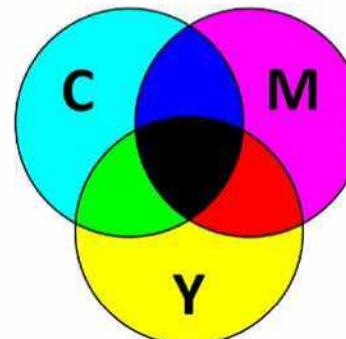
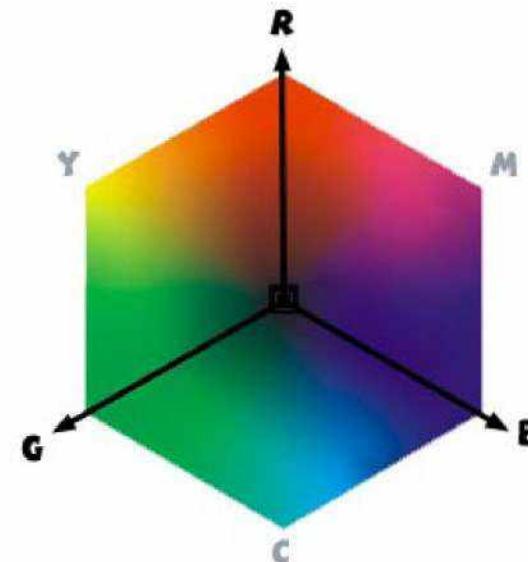
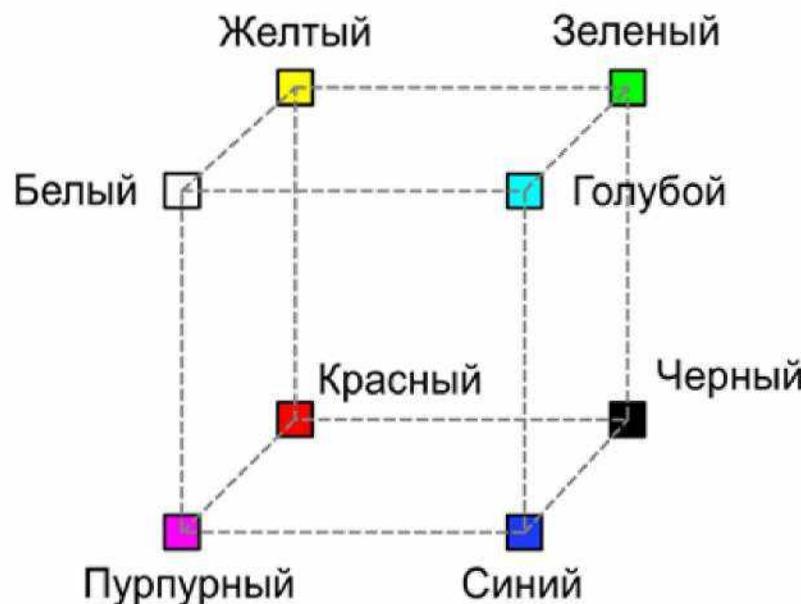
Область применения: полиграфия.

Каждое из чисел, определяющее цвет в CMY, представляет собой процент краски данного цвета, составляющей цветовую комбинацию.

Модель является аппаратно- зависимой, так как на практике реальный цвет будет обусловлен размером точки раstra на фотовыводе, состоянием печатной машины, качеством бумаги и красок.



# Цветовые модели. Модель CMY





# Цветовые модели. Модель CMYK

**CMYK** (Cyan, Magenta, Yellow, Key) -- эта четырехканальная модель является улучшением модели CMY. Компонента **K**, соответствующая черному цвету, добавлена по следующим причинам:

- реальные типографские краски имеют примеси, их цвет не совпадает в точности с теоретически рассчитанными cyan, yellow и magenta; поэтому получить чистый черный цвет достаточно сложно;
- чёрная краска существенно дешевле трех цветных;
- при выводе мелких чёрных деталей изображения возрастает риск неточного совпадения точек нанесения.



# Цветовые модели. Преобразование RGB-CMY

Цвета моделей RGB и CMY являются дополнительными друг к другу (**дополнительный цвет** -- это цвет, результатом смешения которого с данным является белый).

$$C = 1 - R/255$$

$$M = 1 - G/255$$

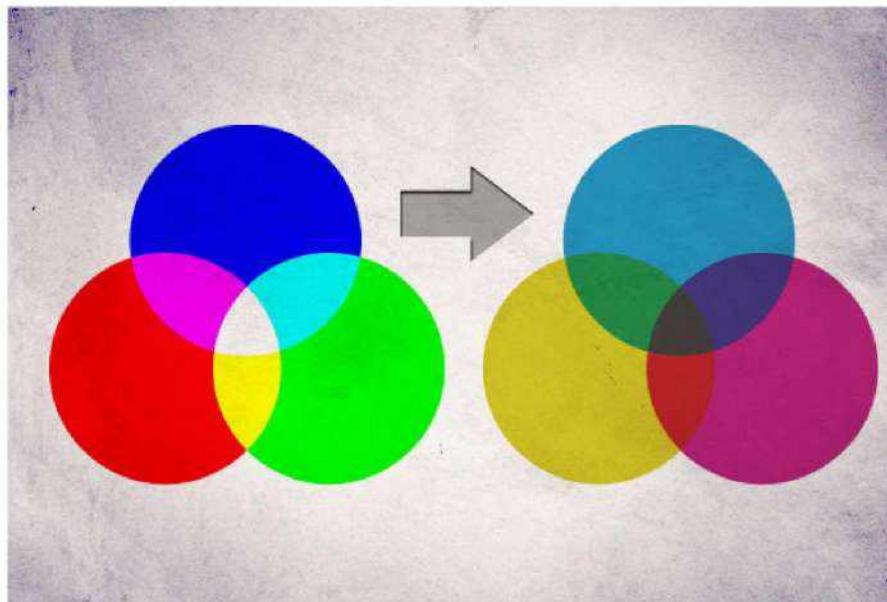
$$Y = 1 - B/255$$

$$\begin{pmatrix} R/255 \\ G/255 \\ B/255 \end{pmatrix} + \begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$



# Цветовые модели. Преобразование RGB-CMY

В связи с различием цветового охвата двух этих моделей при преобразовании изображения RGB в режим CMY цветовые значения RGB, выходящие за пределы цветового охвата CMY корректируются для попадания в этот цветовой охват. В результате этого некоторые данные изображения могут быть утрачены и не восстановятся при обратном переходе из режима CMY в RGB.





# Цветовые модели. Преобразование RGB-CMYK

Если добавлять К, то возникает проблема. CMYK - очень тесно связана с конкретным печатным процессом. В обратном преобразовании не ясно, в какой из компонент (R,G,B) и в какой пропорции учитывать К, так как в модели RGB нет компенсирующего канала (К). Поэтому при переводе в RGB «чистые» чёрные составляющие будут распределены между имеющимися тремя каналами. При обратном переводе контур чёрного будет интерпретирован в зависимости от настроек цветового профиля.

*Преобразование RGB →CMYK:*

$$K = \min(1-R/255, 1-G/255, 1-B/255);$$

$$C=(1-R/255-K)/(1-K); M=(1-G/255-K)/(1-K); Y=(1-B/255-K)/(1-K);$$

*Преобразование CMYK →RGB:*

$$R=255(1-C)(1-K); G=255(1-M)(1-K); B=255(1-Y)(1-K);$$



# Интуитивные цветовые модели.

## Интуитивные цветовые модели

- позволяют обращаться с цветом на интуитивно понятном уровне;
- значительно упрощают проблему согласования цветов, поскольку после установки значения яркости можно заняться настройкой цвета.

Наиболее известные интуитивные цветовые модели:

- HSV (HSB)
- HLS (HIS)
- YUV



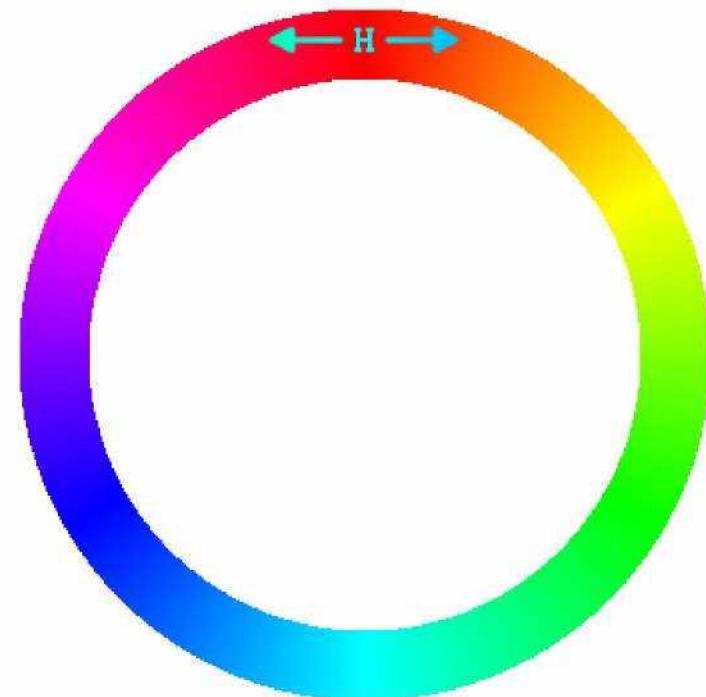
# Интуитивные цветовые модели.

**HSV** (Hue, Saturation, Value --  
тон, насыщенность, значение)

или

**HSB** (Hue, Saturation, Brightness --  
тон, насыщенность, яркость)

**Hue** — цветовой тон,  
(например, красный, зелёный или  
сине-голубой). Варьируется в  
пределах  $0-360^\circ$ , однако иногда  
приводится к диапазону 0-100 или  
0-1.



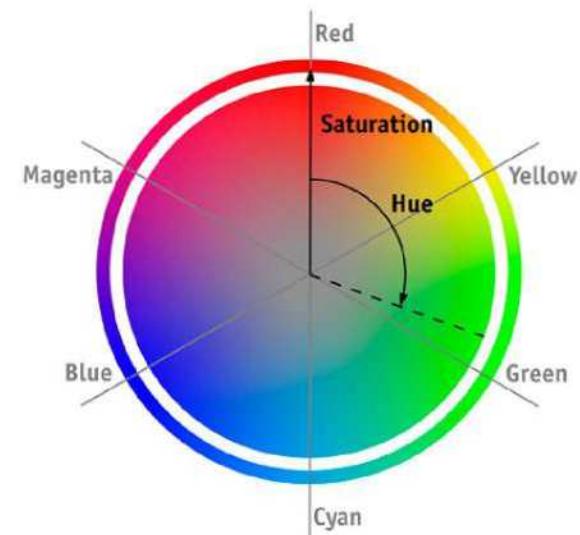
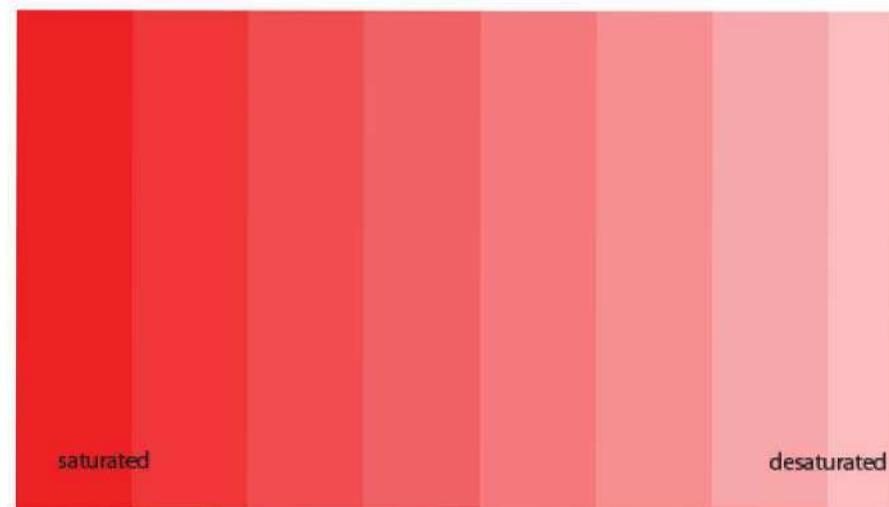


# Интуитивные цветовые модели.

Saturation -- насыщенность.

Насыщенность (Saturation) – это параметр цвета, определяющий его чистоту.

Уменьшение насыщенности цвета означает его разбеливание. Цвет с уменьшением насыщенности становится пастельным, блеклым, размытым. На модели все одинаково насыщенные цвета располагаются на концентрических окружностях, т. е. можно говорить об одинаковой насыщенности, например, зеленого и пурпурного цветов, и чем ближе к центру круга, тем более разбеленные цвета получаются. В самом центре любой цвет максимально разбеливается, проще говоря, становится белым цветом.





# Интуитивные цветовые модели.

## VALUE

Value -- значение цвета, или

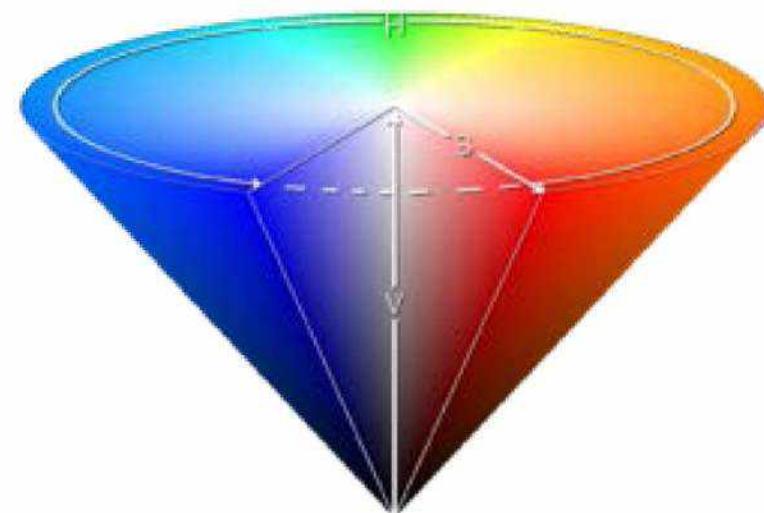
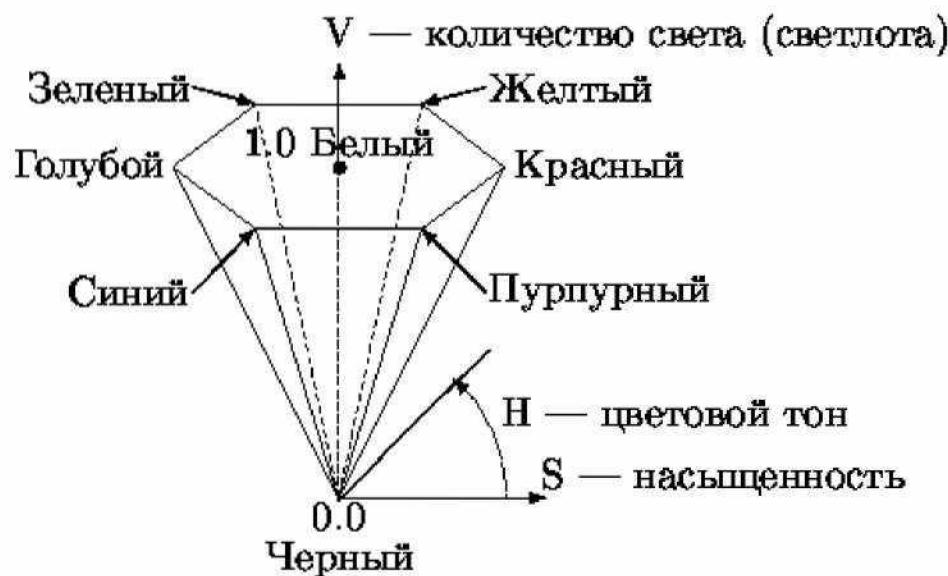
Brightness -- яркость.

Яркость (Brightness) – это параметр цвета, определяющий освещенность или затемненность цвета. Амплитуда (высота) световой волны соответствует этому параметру. Уменьшение яркости цвета означает его зачернение. Работу с яркостью можно характеризовать как добавление в спектральный цвет определенного процента черной краски. Чем больше в цвете содержание черного, тем ниже яркость, тем более темным становится цвет.





# Интуитивные цветовые модели.





# Интуитивные цветовые модели.

Если рассматривать данную модель в соотношении с электромагнитным излучением, то:

- цветовой тон определяется длиной волны, которой соответствует наибольшее значение интенсивности излучения;
- насыщенность – отличием этого значения от среднего по всему диапазону;
- яркость соответствует общей интенсивности излучения.



# Интуитивные цветовые модели.

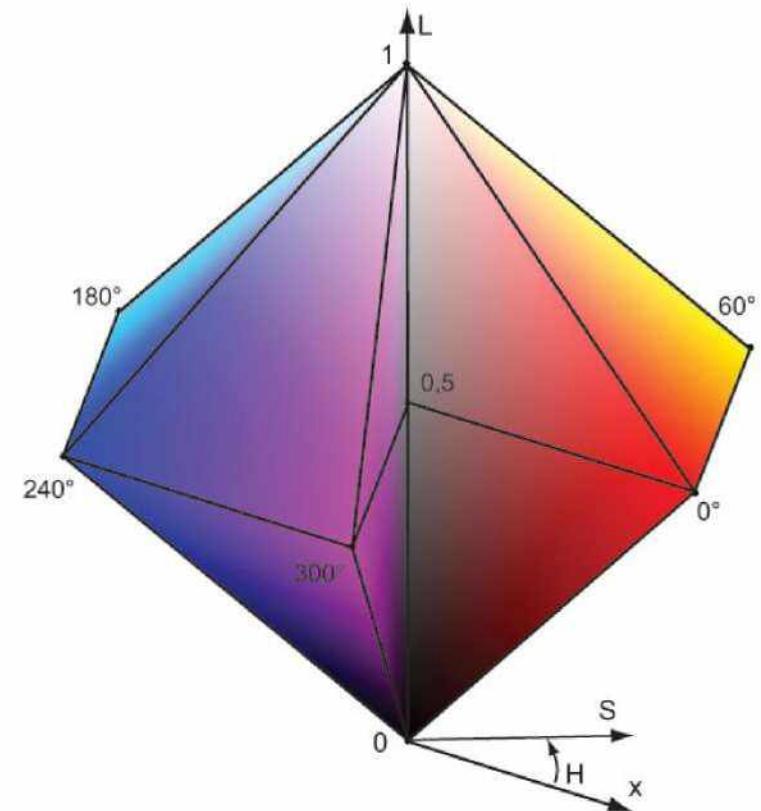
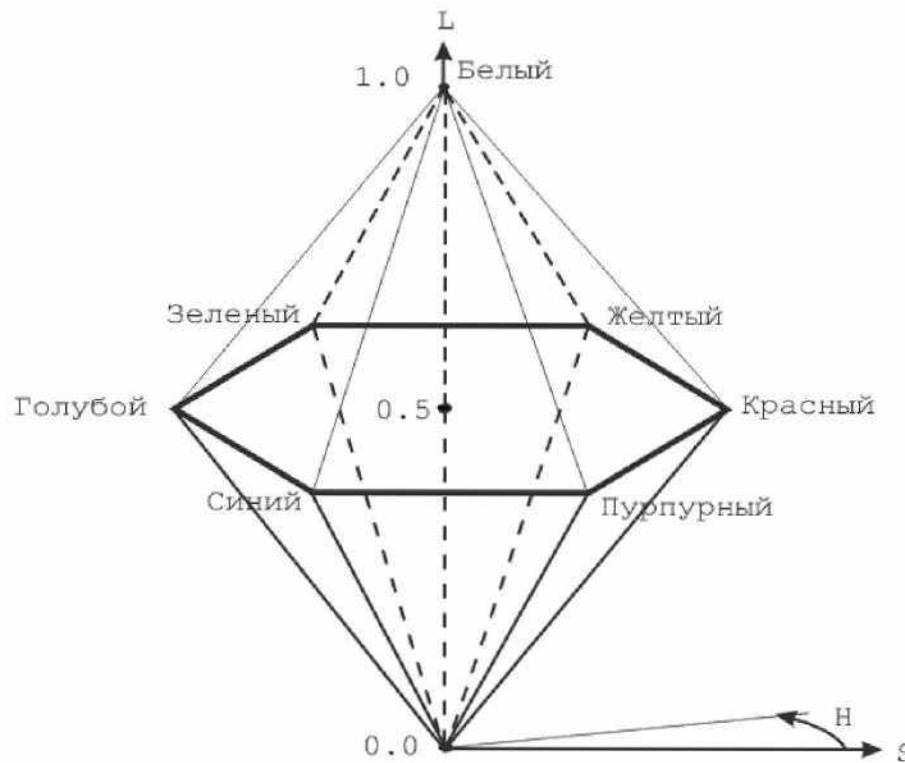
Модель HLS – (Hue, Lightness, Saturation)

Получена из HSV вытягиванием вдоль вертикальной оси. Понятия H и S остались теми же, только по вертикальной оси теперь L вместо V.

Концептуальное различие состоит в том, что в этой модели считается, что движение от чистых цветов (у которых  $L=0,5$ ,  $S = 1$ ) как в направлении белого, так и черного (а не только черного, как в HSV) одинаково приводит к уменьшению информации в H (вплоть до того, что в вершинах H не определено (как впрочем, и на всей вертикальной оси  $S = 0$ )) и сужению диапазона S.



# Интуитивные цветовые модели.



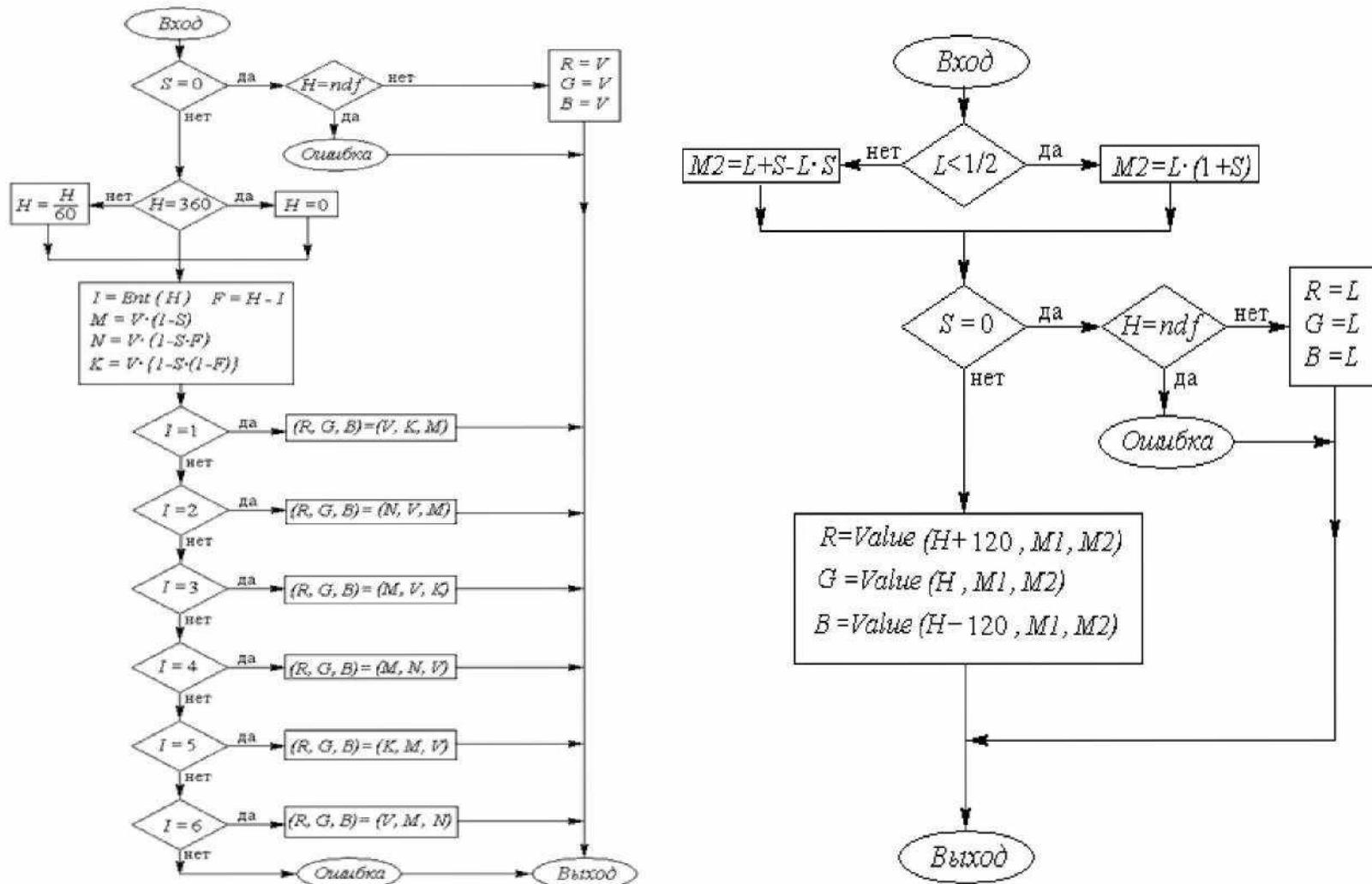


# Интуитивные цветовые модели.

	Интенсивность	Тон	Насыщенность
<i>Уменьшенное значение</i>			
<i>Исходное значение</i>			
<i>Увеличение значение</i>			



# Интуитивные цветовые модели. Преобразования HSV-RGB и HLS-RGB





# Цветовая модель CIE XYZ

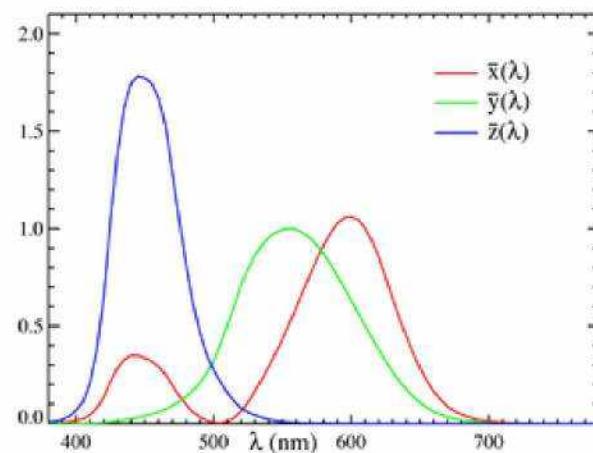
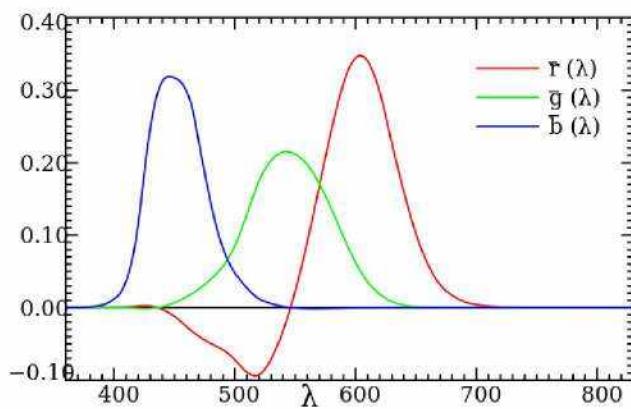
Модель XYZ – это самая первая из используемых моделей. Основная задача этой модели и соответствующего ей цветового пространства – описать границы восприятия человеком цвета. Это эталонная модель, остальные с ней сравниваются. Именно цветовое пространство XYZ охватывает все цвета, которые может увидеть человек.



# Цветовая модель CIE XYZ

Кривые сложения, построенные Гилдом и Райтом, имеют отрицательные участки, и перекрываются для большинства цветов спектра. Для описания независимого цвета это не слишком удобно, причем избавиться от отрицательных значений не удастся ни при какой реальной тройке основных цветов.

Для того, чтобы избежать отрицательных значений в кривых сложения, они были подвергнуты линейному математическому преобразованию, в результате чего были получены новые кривые сложения, называемые кривыми сложения цветов стандартного колориметрического наблюдателя CIE 1931.





# Цветовая модель CIE XYZ

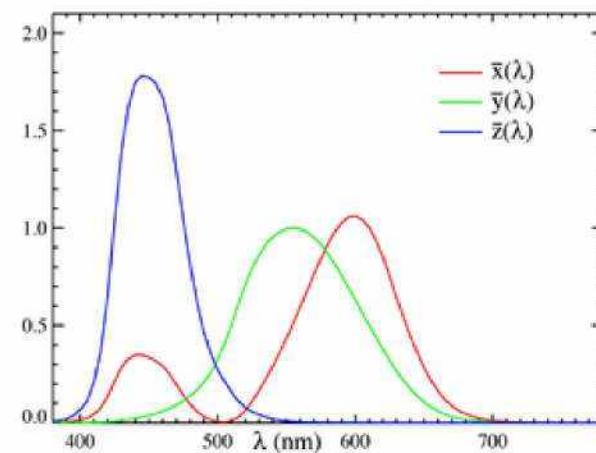
Новые кривые сложения цветов были получены путем перенесения системы цветовых координат, основанных на использовании трех основных цветов  $R=700\text{nm}$ ,  $G=546.1\text{nm}$ ,  $B=435.8\text{nm}$ , в систему координат, основанную на трех «воображаемых» цветах  $X$ ,  $Y$ ,  $Z$ .

Сами  $X$ ,  $Y$  и  $Z$  не являются реальными цветами, спектральный состав этих цветов на некоторых длинах волн отрицательный. Но подобраны они так, что любой реальный цвет представим в виде их линейной комбинации с неотрицательными коэффициентами. Так выглядят функции подбора для XYZ:

$$X=0.49000R+0.31000G+0.20000B$$

$$Y=0.17697R+0.81240G+0.01063B$$

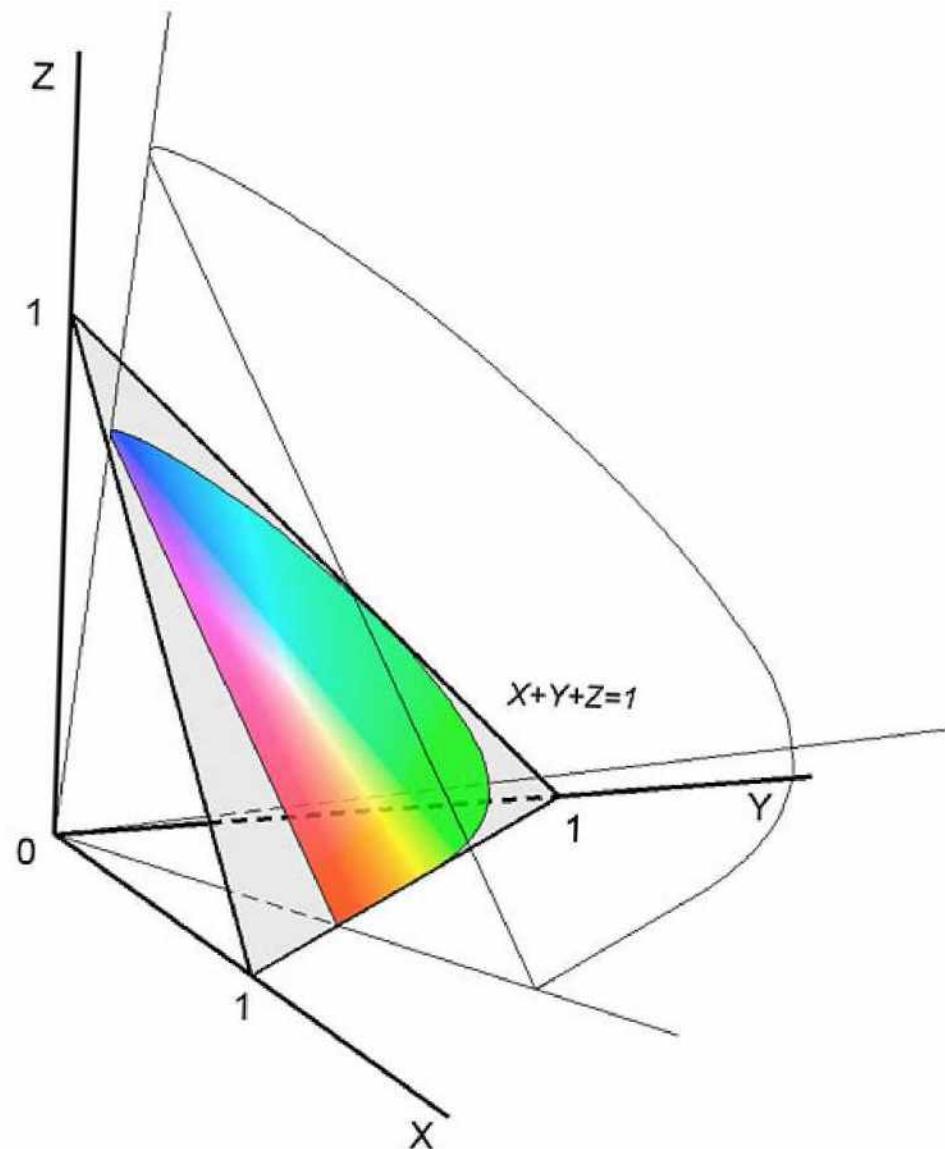
$$Z=0.00000R+0.01000G+0.99000B$$





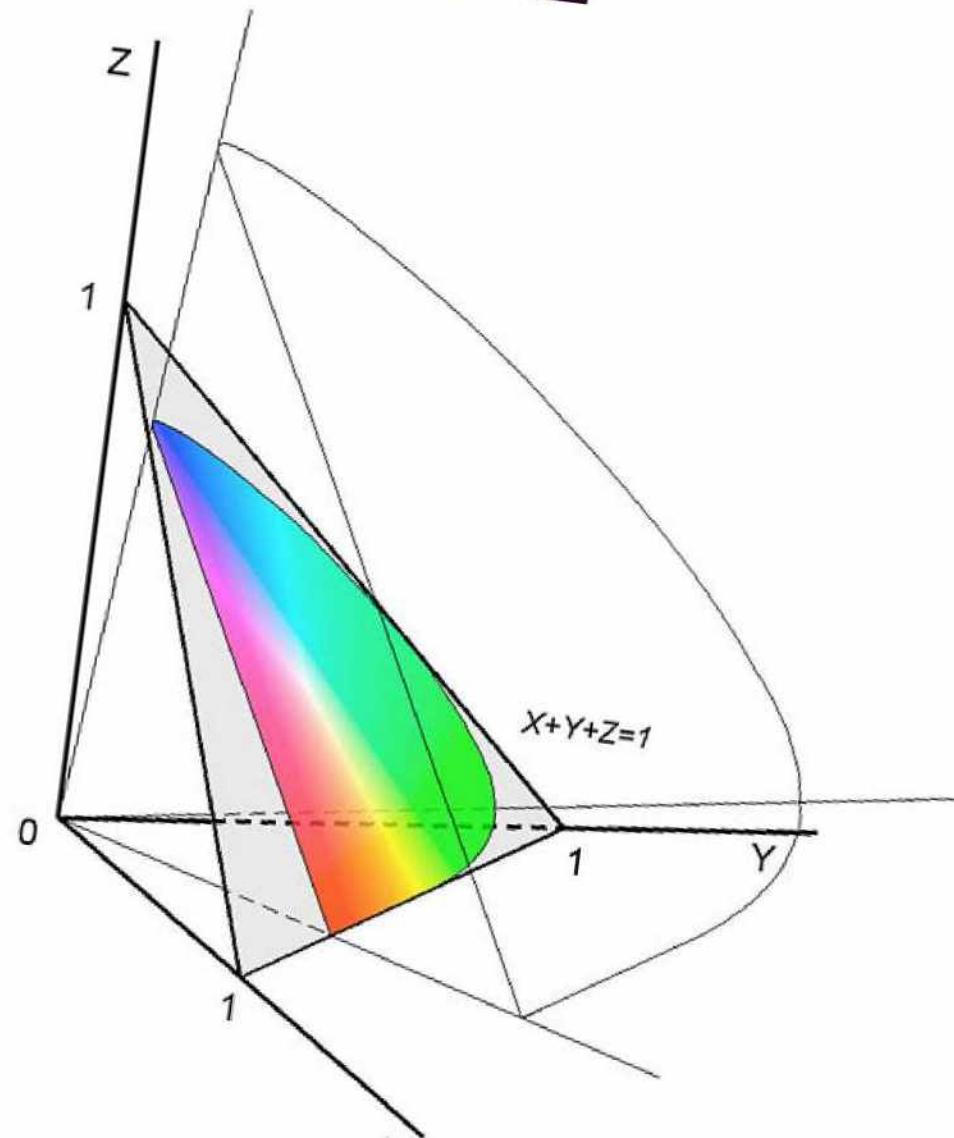
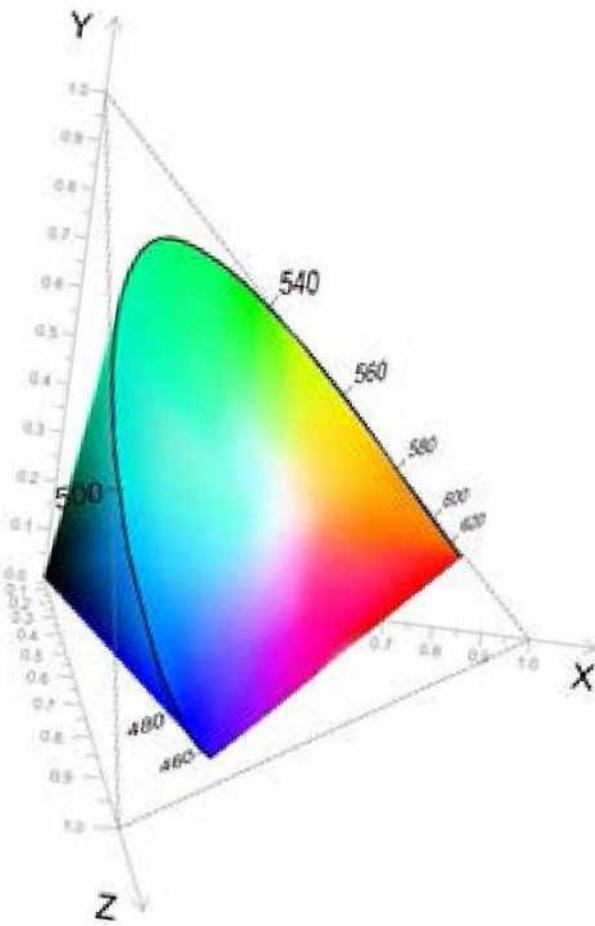
# Цветовая модель CIE XYZ

Любой реальный цвет можно представить точкой в пространстве XYZ (CIE XYZ), но далеко не любая точка в этом пространстве представляет реальный цвет. В прямоугольной системе координат XYZ область видимых человеческим глазом цветов представляет собой конус со сложным основанием.





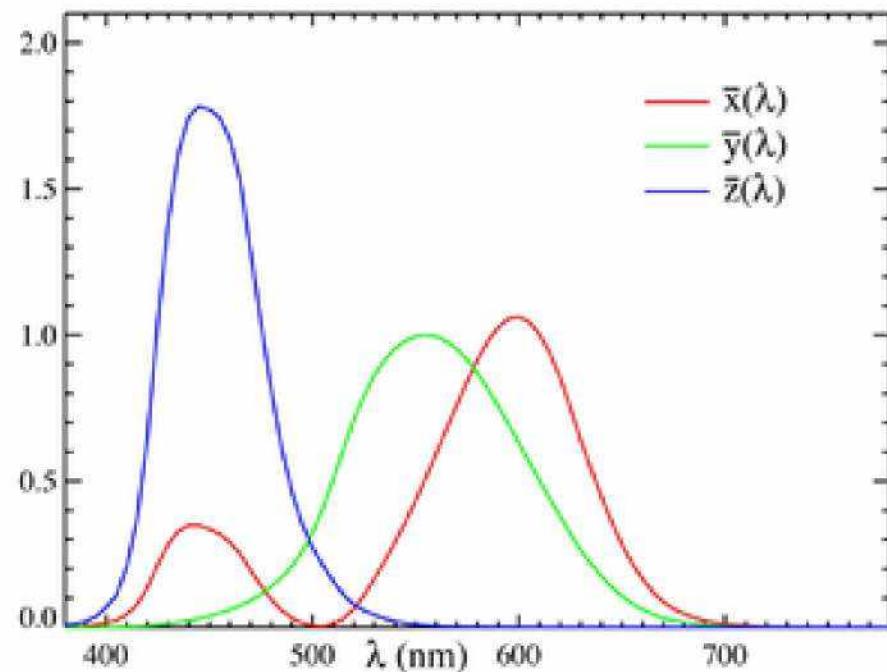
# Цветовая модель CIE XYZ





# Цветовая модель CIE XYZ

Кривая  $y(\lambda)$  практически совпадает с кривой спектральной чувствительности. Это позволяет сделать вывод о том, что компонента Y в модели XYZ отвечает за яркостную составляющую.

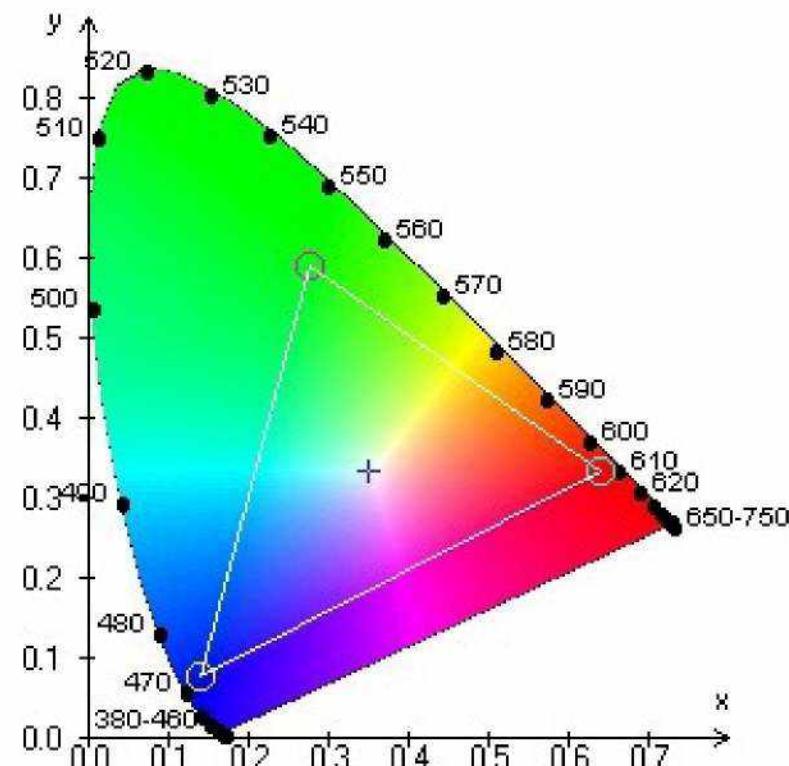




# Цветовая модель CIE XYZ

Крайне удобным производным от XYZ пространством является CIE xy, получаемое сечением XYZ плоскостью  $X+Y+Z=1$ . На плоскости вводится система координат с началом в точке пересечения плоскости с осью Z и координатами x и y, равными единицам в точках пересечения плоскости с осями X и Y соответственно.

Цвета в пределах плоскости xy отличаются по цветовому тону и насыщенности, в то время как яркость остается за пределами описания. Иногда для полного описания цвета используют пространство xyY, рассматривая в качестве яркости значение Y.





# Преобразование RGB-CIE XYZ

$$Rn = g[R / 255] * 100$$

$$Gn = g[G / 255] * 100$$

$$Bn = g[B / 255] * 100$$

$$g(x) = \begin{cases} \left( \frac{x + 0.055}{1.055} \right)^{2.4}, & x \geq 0.04045 \\ \frac{x}{12.92}, & \text{иначе} \end{cases}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{pmatrix} \begin{pmatrix} Rn \\ Gn \\ Bn \end{pmatrix}$$

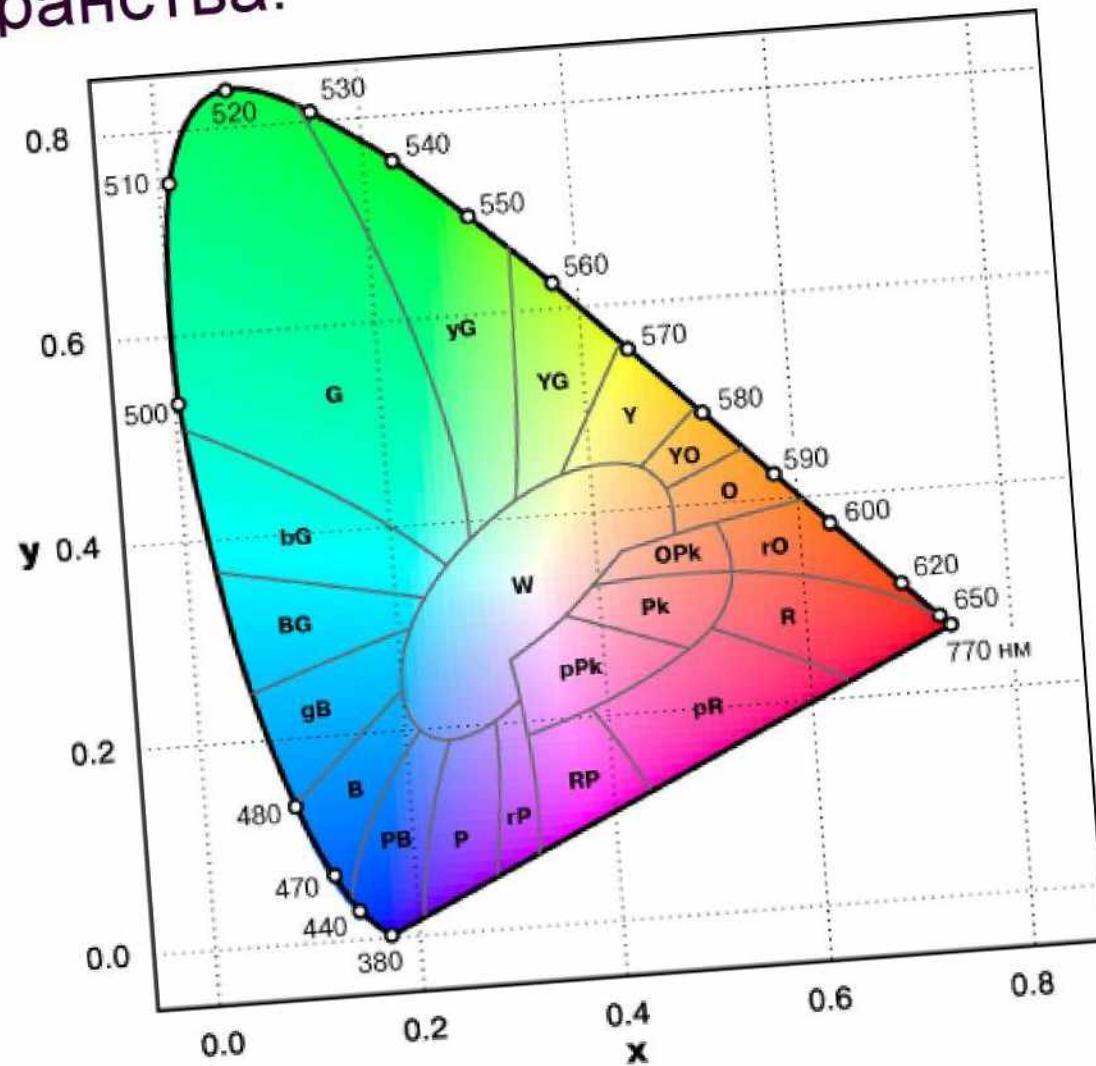


## Равноконтрастные цветовые пространства.

Одним из существенных недостатков цветового пространства XYZ является то обстоятельство, что одинаковым изменениям численных координат цвета не соответствуют равнозначные изменения цветовых ощущений. Иначе говоря, разница между двумя соседними цветами находящимися в одной области графика будет не так ощутима, как разница между двумя соседними цветами, находящимися в другой области графика.



# Равноконтрастные цветовые пространства.





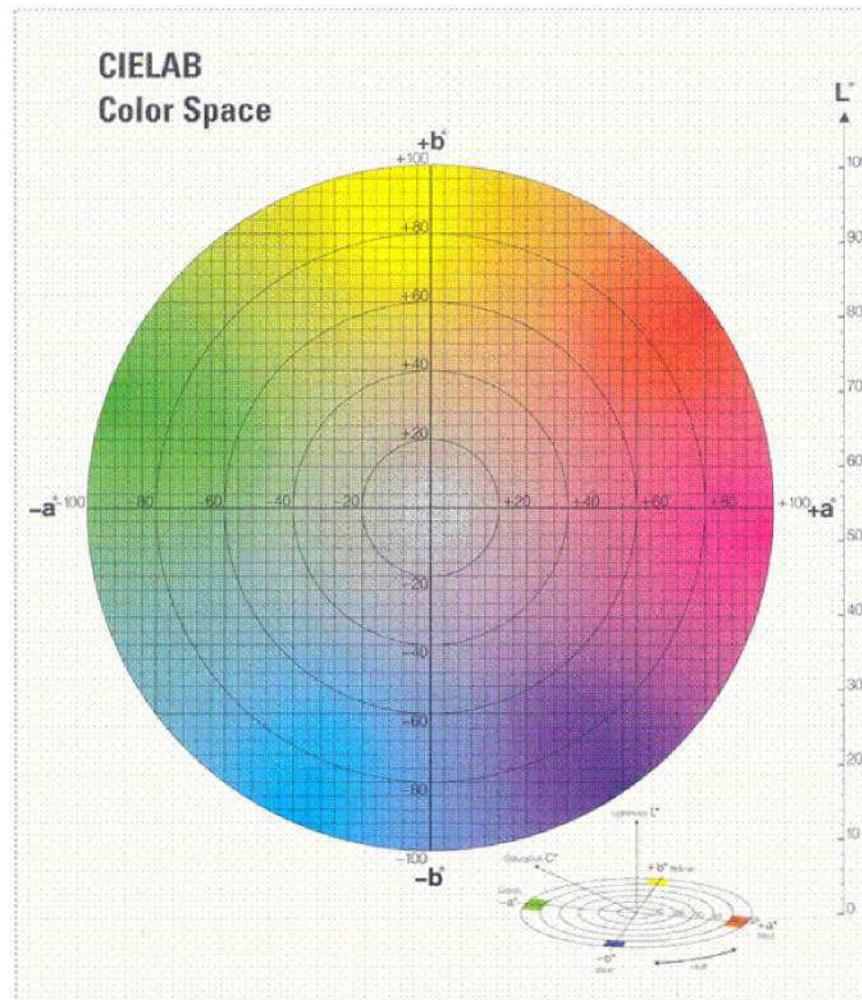
# Равноконтрастные цветовые пространства. Lab.

Цветовая модель Lab (CIE L\*a\*b\*) была разработана Международной комиссией по освещению с целью создания цветового пространства, изменение цвета в котором было бы более линейным с точки зрения человеческого восприятия, то есть чтобы одинаковое изменение значений координат цвета в разных областях цветового пространства производило одинаковое ощущение изменения цвета. В связи с этим модель Lab позволяет математически корректировать нелинейность восприятия цвета человеком, т.е. она является аппарато-независимой.

Цветовая модель Lab позволяет отдельно воздействовать на яркость, контраст и цвет изображения, что в значительной степени ускоряет обработку и уменьшает количество шума цифровых изображений.



# Равноконтрастные цветовые пространства. Lab.





# Равноконтрастные цветовые пространства. Lab.

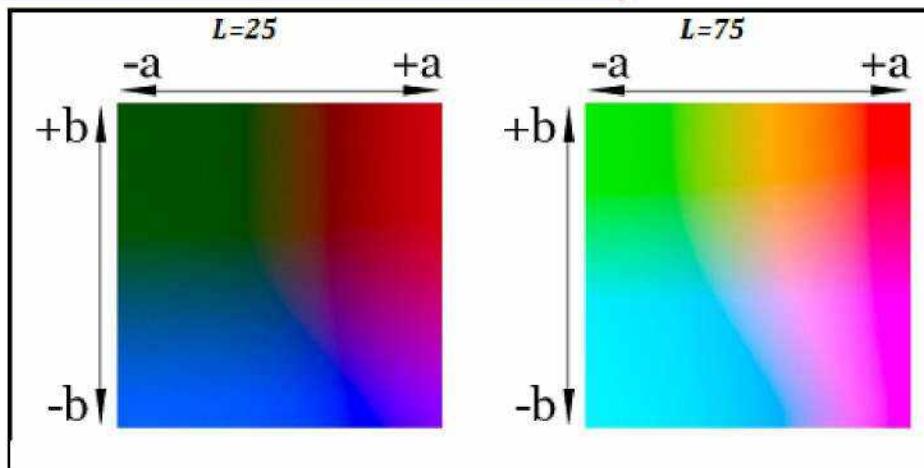
В системе Lab координата L означает светлоту (в диапазоне от 0 до 100).

$$L = 116 \cdot F(Y/Y_w) - 16$$

$$\begin{aligned} a &= 500 \cdot (F(X/X_w) - F(Y/Y_w)) \\ b &= 200 \cdot (F(Y/Y_w) - F(Z/Z_w)) \end{aligned}$$

$$F(x) = \begin{cases} \sqrt[3]{x}, & x \geq 0.008856 \\ 7.787x + \frac{16}{116}, & 0 \leq x < 0.008856 \end{cases}$$

$(X_w, Y_w, Z_w)$  – координаты точки белого





# Равноконтрастные цветовые пространства. Lab. Координаты точки белого в пространстве XYZ

<i>Illuminant</i>	X	Y	Z	
A	109.850	100.000	35.585	Incandescent/tungsten
B	99.0927	100.000	85.313	Old direct sunlight at noon
C	98.074	100.000	118.232	Old daylight
D50	96.422	100.000	82.521	ICC profile PCS
D55	95.682	100.000	92.149	Mid-morning daylight
D65	95.047	100.000	108.883	Daylight, sRGB, Adobe-RGB
D75	94.972	100.000	122.638	North sky daylight
E	100.000	100.000	100.000	Equal energy



# Равноконтрастные цветовые пространства. Luv.

$$u' = 4X / (X + 15Y + 3Z)$$

$$v' = 9Y / (X + 15Y + 3Z)$$

$$L^* = 116 \cdot F(Y/Y_w) - 16$$

$$u^* = 13 \cdot L^* (u' - u'_w)$$

$$v^* = 13 \cdot L^* (v' - v'_w)$$

В обратную сторону:

$$u' = u^* / (13L^*) + u'_w$$

$$v' = v^* / (13L^*) + v'_w$$

$$Y = F^{-1}((L^* + 16) / 116)Y_w$$

$$X = 9 \cdot Y \cdot u' / (4 \cdot v')$$

$$Z = (4 \cdot X - 15 \cdot v' \cdot Y - v' \cdot X) / (3 \cdot u')$$



# Телевизионные цветоразностные цветовые системы

В цветовых системах YIQ и YUV информация о цвете представляется в виде сигнала яркости (Y) и двух цветоразностных сигналов (IQ и UV соответственно).

$$Y = 0.299R + 0.587G + 0.114B$$

$$I = 0.596R - 0.274G - 0.321B$$

$$Q = 0.211R - 0.526G + 0.311B$$

$$Y = 0.299R + 0.587G + 0.114B$$

$$U = 0.492(B - Y)$$

$$V = 0.877(R - Y)$$



# Законы Грассмана (1853)

- 1. Закон трехмерности.** Для описания любого цвета необходимо и достаточно трех компонентов. Любые четыре цвета находятся в линейной зависимости, хотя существует неограниченное число линейно независимых совокупностей из трех цветов.
- 2. Закон аддитивности.** Цвет смеси зависит только от цветов смешируемых компонент и не зависит от их спектральных составов.
- 3. Закон непрерывности.** Если в смеси трех цветовых компонент одна меняется непрерывно, а две другие остаются постоянными, то цвет смеси также будет изменяться непрерывно.



Белорусский Государственный Университет  
Факультет прикладной математики и информатики

# Компьютерная графика

Василевский Константин

Викторович

17 лекций,

17 лабораторных занятий



# Формы компьютерного представления двумерных изображений. Растворные и векторные форматы.

- История вопроса. Основные формы представления графической информации.
- Растворная форма представления изображений. Форматы файлов: BMP, PNG, GIF, JPEG.
- Векторная форма представления изображения. Форматы файлов: WMF, EPS, DXF, CDR и др.



# Векторная графика и растровая графика

**Векторная графика** -- способ представления изображений, основанный на использовании объектов и геометрических примитивов (таких как точки, линии, сплайны, многоугольники), заданных своим математическим описанием.

Форматы векторной графики: *WMF, EPS, DXF, CDR*.

**Растровая графика** -- способ представления объектов, основанный на использовании сетки пикселей.

Форматы растровой графики: *BMP, PNG, GIF, JPEG*.

## **Фрактальная графика**



# Понятие раstra и пикселя

**Растр** -- это множество  $M$ , такое, что существует взаимно однозначное (биективное) отображение

$$F : X \times Y \rightarrow M$$

где  $X \subset \mathbb{Z}$        $Y \subset \mathbb{Z}$

**Пиксель  $F(i,j)$**  -- элемент раstra.

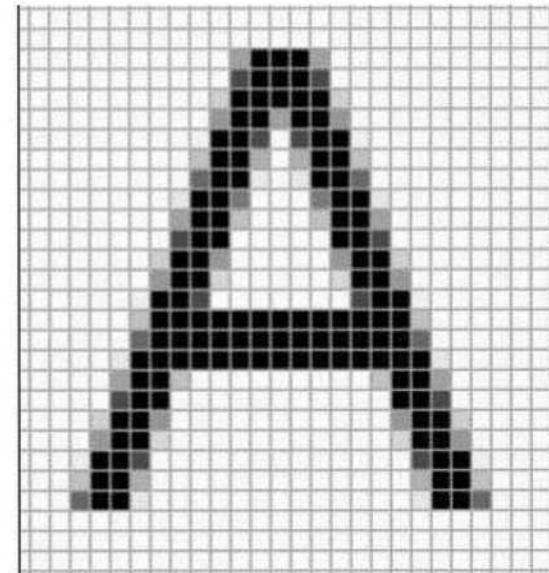
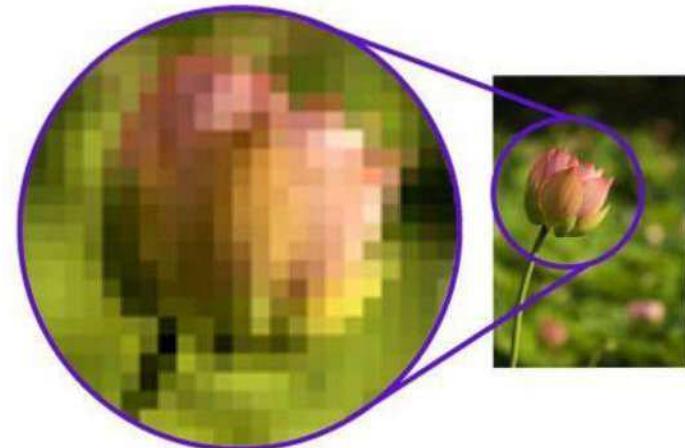
**M** -- множество атрибутов (как правило, это цвет).



# Растровая графика



coloradocolorist.ru





# Достоинства растровой графики

- Эффективно представляет реальные образы.
- Обеспечивает высокую точность передачи градаций цветов и полутонов.
- Хорошо подходит для устройств вывода (принтеры, мониторы).



# Недостатки растровой графики

- Занимает большое количество памяти.
- Геометрические преобразования значительно ухудшают качество.

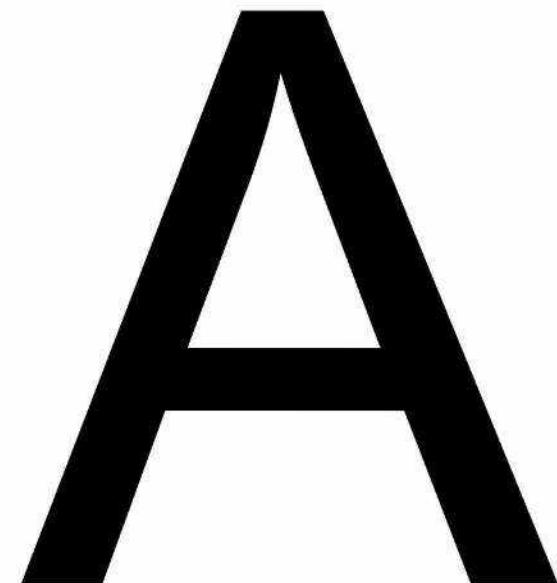
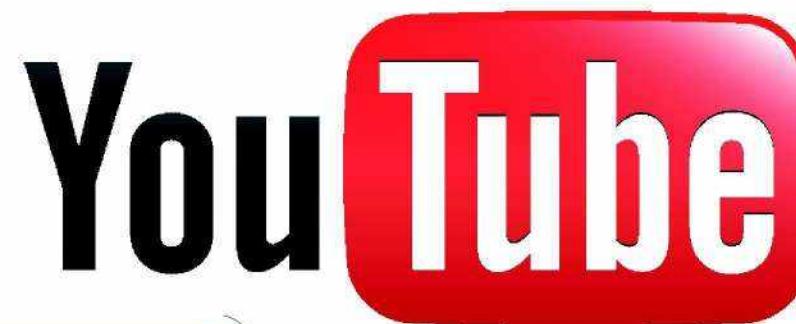


# Векторная графика

Векторные изображения формируются из простейших объектов - графических примитивов (точка, линия, окружность, прямоугольник и т. д.), которые хранятся в памяти компьютера в виде описывающих их математических формул и атрибутов (цвет, прозрачность и т.д).



# Векторная графика





# Достоинства векторной графики

- Занимает меньшее количество памяти.
- Гибкая масштабируемость.
- Возможность легкого редактирования всех частей векторного изображения;
- Простой экспорт векторного рисунка в растровый



# Недостатки векторной графики

- Отсутствие реалистичности.
- Невозможность использования эффектов, которые можно применять в растровой графике.
- Практически полная невозможность экспорта растрового рисунка в векторный.



# Фрактальная графика

Математической основой фрактальной графики является фрактальная геометрия. Здесь в основу метода построения изображений положен принцип наследования от, так называемых, «родителей» геометрических свойств объектов-наследников.

**Понятия фрактал, фрактальная геометрия и фрактальная графика**, появившиеся в конце 70-х, сегодня прочно вошли в обиход математиков и компьютерных художников. Слово фрактал образовано от латинского *fractus* и в переводе означает «состоящий из фрагментов». Оно было предложено математиком **Бенуа Мандель-Бротом** в 1975 году для обозначения нерегулярных, но самоподобных структур, которыми он занимался.

**Фракталом** называется структура, состоящая из частей, которые в каком-то смысле подобны целому. Одним из основных свойств фракталов является **самоподобие**. Объект называют **самоподобным**, когда увеличенные части объекта походят на сам объект и друг на друга.



# Фрактальная графика



В центре фрактальной фигуры находится её простейший элемент — **равносторонний треугольник**, который получил название «**фрактальный**». Затем, на среднем отрезке сторон строятся равносторонние треугольники со стороной, равной  $1/3$  от стороны исходного фрактального треугольника. В свою очередь, на средних отрезках сторон полученных треугольников, являющихся объектами-наследниками первого поколения, выстраиваются треугольники-наследники второго поколения со стороной  $1/9$  от стороны исходного треугольника.

Таким образом, мелкие элементы фрактального объекта повторяют свойства всего объекта. Полученный объект носит название «**фрактальной фигуры**». Процесс наследования можно продолжать до бесконечности. Таким образом можно описать и такой графический элемент как прямая.



# Фрактальная графика

Изменяя и комбинируя окраску фрактальных фигур, можно моделировать образы живой и неживой природы (например, ветви дерева или снежинки), а также составлять из полученных фигур «фрактальную композицию». Фрактальная графика, так же как векторная и трёхмерная, является вычисляемой. Её главное отличие в том, что **изображение строится по уравнению или системе уравнений**. Поэтому в памяти компьютера для выполнения всех вычислений ничего, кроме формулы, хранить не требуется.

Только изменив коэффициенты уравнения, можно получить совершенно другое изображение. С помощью нескольких математических коэффициентов можно задать линии и поверхности очень сложной формы.

Итак, базовым понятием для фрактальной компьютерной графики являются **«Фрактальный треугольник»**. Затем идет **«Фрактальная фигура»**, **«Фрактальный объект»**, **«Фрактальная прямая»**, **«Фрактальная композиция»**, **«Объект-родитель»** и **«Объект наследник»**.



# Растровая и векторная графика. Сравнение.

	Растровая графика	Векторная графика
Основные элементы	Точка (пиксель)	Линии и кривые
Занимаемый объем памяти	Большой	Маленький
Геометрические преобразования (масштабирование, повороты и т.д.)	Приводят к потере информации и искажению изображения	Не изменяют хранимую информацию
Фотореалистичность и естественная цветопередача	Присутствует	Отсутствует
Области применения	Обработка фотографий, работа со сканированными изображениями	Компьютерный дизайн, моделирование, проектирование.



# Растровые изображения. Характеристики.

Размер изображения -- количество пикселей по ширине и по высоте ( $800 \times 600\text{px}$ ,  $1024 \times 768\text{px}$ ,  $1600 \times 1200\text{px}$  и т. д.)

Разрешающая способность или разрешение изображения -- величина, определяющая количество элементов изображения на единицу длины (иначе говоря, плотность пикселей).

Единицы измерения: точки на дюйм (dpi);  
в одном дюйме изображения, например при разрешении в 72 dpi,  
содержится 72 единичных элемента (пикселя, точек).

Глубина цвета (или битовая глубина) -- это количество градаций, допустимых для каждого пикселя. Указывается в количестве бит на пиксель или бит на канал для данной цветовой модели.



# Файловые форматы

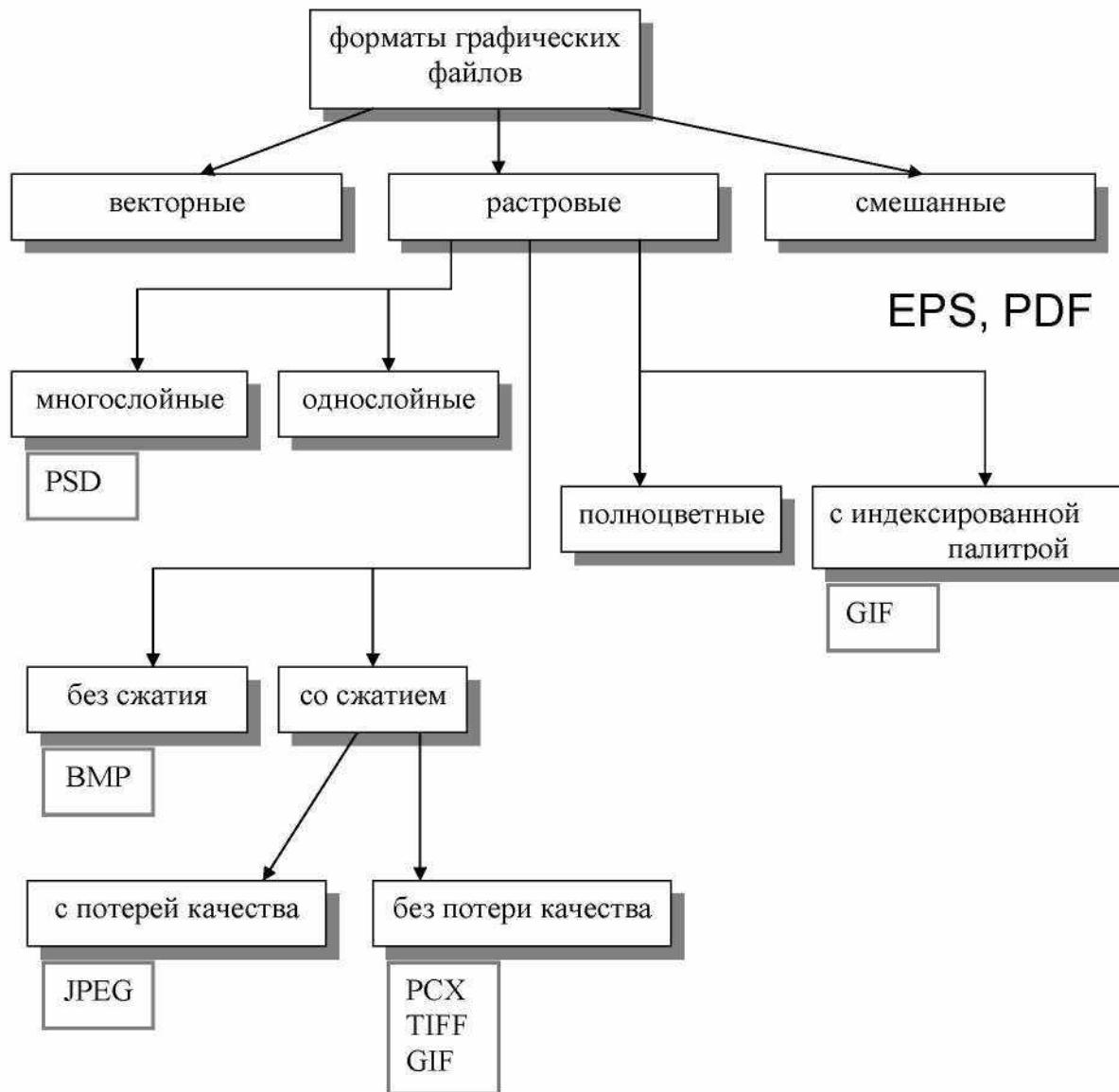
Формат графического файла -- способ представления и расположения графических данных на электронных носителях.

Различают *растровые* и *векторные* форматы.

Большинство векторных форматов могут так же содержать внедренные в файл растровые объекты или ссылку на растровый файл.



# Растровые файловые форматы





# Растровые файловые форматы

- BMP
- GIF
- PNG
- JPEG
- TIFF
- PCX
- TGA
- IMG

Чем отличаются:

- структура файла
- глубина поддерживаемой палитры
- тип сжатия
- поддержка различных цветовых моделей
- поддержка прозрачности



# Растровые файловые форматы

## BMP (Bitmap Picture)

- Формат хранения растровых изображений, разработанный компанией Microsoft. Его поддержка интегрирована в операционные системы Windows и OS/2.
- Файл состоит из заголовка фиксированной длины, каталога информации об изображениях и непосредственно изображения.
- Файлы формата *BMP* могут иметь расширения .bmp, .dib и .rle.
- Поддерживается только цветовая модель RGB.
- Глубина цвета в данном формате может быть 1, 2, 4, 8, 16, 24, 32, 48 бит на пиксель, но глубина 2 бита на пиксель официально не поддерживается.
- Поддерживает RLE-сжатие без потери качества, но этот вариант используется редко из-за потенциальных проблем несовместимости.
- Не поддерживает анимацию, прозрачность и черезстрочное отображение.



# Растровые файловые форматы

## GIF (Graphic Interchage Format)

- Один из самых первых форматов в Интернете.
- Глубина цвета – 8 бит на пиксель.
- Целесообразно использовать этот формат в Web, для изображений без плавных цветовых переходов (логотипы, баннеры, надписи, схемы).
- Поддерживает сжатие без потерь (LZW-компрессия).
- Поддерживает анимацию, прозрачность и черезстрочное отображение.



# Растровые файловые форматы

## PNG (Portable network graphics)

- Поддерживает глубину цвета до 48 бит.
- Сжатие без потерь по алгоритму Deflate.
- Поддерживает прозрачность (альфа-канал) и черезстрочное отображение; отсутствует поддержка анимации.



# Растровые файловые форматы

## JPEG (Joint Photographic Experts Group)

- Используется для хранения и передачи фотоизображений.
- Поддерживает 24-битную палитру.
- Сжатие с потерями по алгоритму JPEG.
- Не поддерживает анимацию, прозрачность.



# Растровые файловые форматы

## TIFF (Tagged Image File Format)

- Используется в полиграфии.
- Поддерживает глубину цвета до 48 бит и различные цветовые модели.
- Поддержка различных алгоритмов сжатия (ZIP, LZW, JPEG).
- Поддержка прозрачности.



# Векторные файловые форматы

- WMF
- EMF
- EPS
- CDR
- PDF
- AI

Чем отличаются:

- совместимость с различными программными комплексами



# Векторные файловые форматы

## WMF (Windows Metafile)

Векторный формат WMF использует графический язык Windows и, можно сказать, является ее родным форматом. Служит для передачи векторов через буфер обмена (Clipboard). Понимается практически всеми программами Windows, так или иначе связанными с векторной графикой. Однако, несмотря на кажущуюся простоту и универсальность, пользоваться форматом WMF стоит только в крайних случаях для передачи «голых» векторов. WMF искажает цвет, не может сохранять ряд параметров, которые могут быть присвоены объектам в различных векторных редакторах, изначально не мог содержать растровые объекты, в других ОС несовместим со многими программами.



# Векторные файловые форматы

## EPS (Encapsulated PostScript)

Формат Encapsulated PostScript можно назвать самым надежным и универсальным способом сохранения данных. EPS предназначен для передачи векторов и растра в издательские системы, создается почти всеми программами, работающими с графикой. EPS поддерживает все необходимые для печати цветовые модели. EPS имеет много разновидностей, что зависит от программы-создателя.



# Векторные файловые форматы

## PDF (Portable Document Format)

PDF – межплатформенный открытый формат электронных документов, изначально разработанный фирмой Adobe Systems с использованием ряда возможностей языка PostScript, используемый в полиграфии и для отображения документов, презентаций. Открывается на любых устройствах с любыми операционными системами ровно в том виде, в котором был создан.

Средство просмотра PDF формата, Adobe Acrobat Reader – бесплатное.

Редакторы PDF-файлов – платные.



Белорусский Государственный Университет  
Факультет прикладной математики и информатики

# Компьютерная графика

Василевский Константин  
Викторович  
17 лекций,  
17 лабораторных занятий



# Алгоритмы сжатия графической информации

- Сжатие графической информации.  
Избыточность данных.
- Критерии сравнения алгоритмов сжатия.
- Алгоритм кодирования длины повторения (RLE).
- Словарные алгоритмы: алгоритм LZ77, алгоритм LZW.
- Алгоритмы статистического кодирования:  
Алгоритм Хаффмана. Арифметическое кодирование.
- Необходимость сжатия с потерями. Оценка потерь. Алгоритм сжатия изображений JPEG.



# Растровые графические форматы

BMP GIF PNG JPEG TIFF PCX TGA IMG

Чем отличаются:

- структура файла
- глубина поддерживаемой палитры
- тип сжатия
- поддержка различных цветовых моделей
- поддержка прозрачности
- поддержка анимации
- и т.д.



# Глубина цвета

**Глубина цвета (или битовая глубина)** – это количество градаций, используемых для представления каждого пикселя растрового изображения. Указывается в количестве бит на пиксель или бит на канал для данной цветовой модели.

Классификация изображений в зависимости от глубины цвета:

- Бинарные изображения (двуцветные): на каждый пиксель отводится один бит; обычно это черно-белые изображения.





# Глубина цвета

- Grayscale-изображения (в градациях серого). Каждый пиксель имеет яркость в диапазоне от 0 до 255.
- Индексированные 8-битные изображения.
- Полноцветные изображения:  
Цвет каждого пикселя задается 3 каналами, каждый по одному байту (например, (R,G,B)).





# Сжатие изображений

**Сжатие данных** – алгоритмическое преобразование данных, производимое с целью уменьшения занимаемого ими объёма.

Под **сжатием изображений** понимают применение алгоритмов сжатия данных к изображениям, хранящимся в цифровом виде.

Сжатие изображений обусловлено тремя особенностями:

- Большой объем памяти, занимаемой растровыми изображениями.
- Особенность человеческого восприятия: человеческое зрение при анализе изображения оперирует контурами, общим переходом цветов и сравнительно нечувствительно к малым изменениям в изображении.
- Особенности структуры изображения: изображение, в отличие, например, от текста, обладает избыточностью в двухмерном измерении.



# Избыточность данных

В растровых изображениях есть два типа избыточности:

- В случайном изображении некоторые цвета преобладают, а другие встречаются редко.
- Избыточность в результате корреляции пикселей.



# Задача выбора алгоритма сжатия

Пример классификация изображений с целью выбора алгоритма сжатия:

**Класс 1. Изображения с небольшим количеством цветов (4-16) и большими областями, заполненными одним цветом.** Плавные переходы цветов отсутствуют. Примеры: деловая графика — гистограммы, диаграммы, графики и т.п.

**Класс 2. Изображения, с плавными переходами цветов,** построенные на компьютере. Примеры: графика презентаций, эскизные модели в САПР.

**Класс 3. Фотореалистичные изображения.** Пример: отсканированные фотографии.

**Класс 4. Фотореалистичные изображения с наложением деловой графики.** Пример: реклама.

Достаточно сложной и интересной задачей является поиск наилучшего алгоритма для конкретного класса изображений.



## Критерии сравнения алгоритмов сжатия

- Худший, средний и лучший коэффициенты сжатия.

$$\text{коэффициент сжатия} = \frac{\text{размер выходного файла}}{\text{размер входного файла}}$$

- Класс изображений, на который ориентирован алгоритм.
- Симметричность (соотношение ресурсоёмкости процесса сжатия и обратного процесса восстановления).
- Потеря качества.
- Характерные особенности алгоритма и изображений, к которым его применяют.



# Сжатие с потерями и без потерь

**A** называется алгоритмом сжатия без потерь (англ. *Lossless compression*), если существует алгоритм  $A^{-1}$  (обратный к  $A$ ) такой, что для любого изображения  $I$  верно:

$$A(I) = I_1 \text{ и } A^{-1}(I_1) = I$$

**A** называется алгоритмом сжатия с потерями (англ. *Lossy compression*), если он не обеспечивает возможность точного восстановления исходного изображения. Если парный к  $A$  алгоритм, обеспечивающий примерное восстановление, обозначить как  $A^*$ , тогда для изображения  $I$  будет верно:

$$A(I) = I_1, A^*(I_1) = I_2$$

и при этом полученное восстановленное изображение  $I_2$  не обязательно точно совпадает с  $I$ . Пара  $A, A^*$  подбирается так, чтобы обеспечить большие коэффициенты сжатия и тем не менее сохранить визуальное качество, т.е. добиться минимальной разницы в восприятии между  $I$  и  $I_2$ .



## Сжатие изображений. Основные определения.

**Кодер(компрессор)** – программа, которая сжимает исходный файл и создает на выходе файл со сжатыми данными, в которых мало избыточности.

**Декодер (декомпрессор)** – работает в обратном направлении.

**Симметричное сжатие** – сжатие, при котором и кодер, и декодер используют один и тот же базовый алгоритм, но используют его в противоположных направлениях.



# Алгоритм RLE

**RLE** (*run-length encoding* – кодирование длин серий, или кодирование повторов) – алгоритм сжатия данных, заменяющий повторяющиеся символы (серии) на один символ и число его повторов.

Существует 2 варианта работы алгоритма – на битовом уровне и байтовом уровне.

Используется в следующих растровых форматах:

BMP (по желанию)

PCX (всегда)

TGA (по желанию)

IMG (всегда)

TIFF (по желанию)

Алгоритм рассчитан на деловую графику – изображения с большими областями повторяющегося цвета.



# Алгоритм RLE

RLE - битовый уровень.

Данные представляются на уровне последовательности битов (например, черно-белое изображение).

Например: последовательности

000011111000111111111111

соответствует набор чисел количеств повторений

4 5 3 11



# Алгоритм RLE

RLE - битовый уровень.

Однако, учитывая то, что числа, обозначающие количество повторений, также надо кодировать битами, считается, что каждое число повторений изменяется от 0 до 7 (т.е. можно закодировать ровно тремя битами), и они чередуются.

Например, последовательности

11111111111

будет сопоставлена сжатая последовательность

111 000 100

Последовательность, состоящая из

21 единицы, 21 нуля, 3 единиц и 1 ноль

закодируется так

111 000 111 000 111 111 000 111 000 111 011 001



# Алгоритм RLE

RLE - байтовый уровень.

Данные представляются на уровне последовательности байтов (например, grayscale изображение).

Изображение вытягивается в цепочку байт по строкам раstra и далее этот входной поток разбивается на байты (буквы) и повторяющиеся байты кодируются в двухбайтную пару (счетчик, буква).

**Пример: AABBBCCDAA кодируем (2A) (3B) (1C) (1D) (2A)**

Признаком счетчика служат, например, единицы в двух верхних битах считанного файла; соответственно оставшиеся 6 бит расходуются на счетчик, который может принимать значения от 1 до 64. Таким образом, строка из 64 повторяющихся байтов превращается в два байта, т.е. сжимается в 32 раза.



# Алгоритм RLE

Модификация алгоритма.

Для улучшения результатов работы для входных данных используется предварительная обработка.

**Преобразование Барроуза-Уилера** (англ. *Burrows-Wheeler transform - BWT*) – алгоритм, используемый для предварительной обработки данных перед сжатием, разработанный для улучшения эффективности последующего кодирования. Преобразование Барроуза-Уилера меняет порядок символов во входной строке таким образом, что повторяющиеся подстроки образуют на выходе идущие подряд последовательности одинаковых символов.



# Алгоритм RLE

Преобразование Барроуза-Уилера.

Преобразование выполняется в три этапа:

1. Составляется таблица всех циклических перестановок входной строки.
2. Производится лексикографическая (в алфавитном порядке) сортировка строк таблицы.
3. В качестве выходной строки выбирается последний столбец таблицы преобразования и номер строки, совпадающей с исходной.



# Алгоритм RLE

Преобразование Барроуза-Уилера. Пример.

Трансформация			
Вход	Все Перестановки	Сортировка Строк	Выход
.BANANA.	.BANANA. .BANANA A..BANAN NA..BANA ANA..BAN NANA..BA ANANA..B BANANA..	ANANA..B ANA..BAN A..BANAN BANANA.. NANA..BA NA..BANA .BANANA.. .BANANA	BNN.AA.A



# Алгоритм RLE

## Преимущества:

простота и скорость работы (как скорость кодирования, так и скорость декодирования)

## Недостатки:

неэффективность на неповторяющихся наборах символов.  
Использование специальных перестановок повышает  
эффективность алгоритма, но при этом сильно увеличивает  
время работы.



# Характеристики алгоритма RLE

## **Коэффициенты сжатия:**

Первый вариант: 32, 2, 0,5. Второй вариант: 64, 3, 128/129  
(Лучший, средний, худший коэффициенты)

**Класс изображений:** Ориентирован алгоритм на изображения с небольшим количеством повторяющихся цветов: деловую и научную графику.

**Симметричность:** симметричен.

**Характерные особенности:** К положительным сторонам алгоритма можно отнести только то, что он не требует дополнительной памяти при архивации и разархивации, а также быстро работает. Интересная особенность группового кодирования состоит в том, что степень архивации для некоторых изображений может быть существенно повышена всего лишь за счет изменения порядка цветов в палитре изображения.



# Алгоритм LZ77

Словарный метод сжатия информации.

Разработан в 1970 году и опубликован в 1977 году учеными Абрахамом Лемпелем и Якобом Зивом.

Основная идея состоит в использовании ранее прочитанной последовательности входных данных в качестве динамического словаря.



# Алгоритм LZ77

Для входного потока данных кодер создает скользящее окно, содержащее последовательность символов – буфер, или словарь.

Буфер разбивается на 2 части – слева буфер поиска и справа буфер предварительного просмотра.

Буфер поиска содержит символы, которые поступили недавно и уже были закодированы (длина – несколько тысяч байт).

В буфере предварительного просмотра содержится текст, который надо закодировать (длина – несколько десятков байт).



# Алгоритм LZ77

Алгоритм работы кодера:

1. Кодер просматривает буфер поиска справа налево и ищет в нем первое появление текущего символа из буфера предварительного просмотра.
2. Кодер определяет, сколько совпадающих символов следует за найденным символом и текущим, и запоминает длину  $L$  найденной совпадающей подпоследовательности и смещение  $S$  найденного символа относительно текущего.
3. Кодер продолжает поиск по буферу поиска, пытаясь найти более длинные совпадающие подпоследовательности, обновляя  $L$  и  $S$ .
4. Если найдены 2 совпадающие подпоследовательности одинаковой длины, то кодер выбирает наиболее удаленную из них.
5. В выходной поток записывается метка  $< S; L; \text{очередной необработанный символ в буфере предварительного просмотра} >$
6. Буфер сдвигается вправо на число позиций, равное  $L+1$ .
7. Если обратный поиск не обнаружил ни одной совпадающей подпоследовательности, то записывается метка со смещением 0, длиной 0 и текущим символом.



# Алгоритм LZ77

Пример работы.

Входная последовательность «абракадабра»

Размер буфера поиска: 20. Размер буфера предварительного просмотра: 4

Шаг	Буфер	Макс.совпавшая подпосл-ть	Метка
1	«»+« <b>а</b> бра»	-	<0,0,a>
2	«а»+« <b>бр</b> ак»	-	<0,0,б>
3	«аб»+« <b>рак</b> а»	-	<0,0,p>
4	«абр»+« <b>акад</b> »	а	<3,1,k>
5	«абрак»+« <b>адаб</b> »	а	<5,1,d>
6	«абракад»+« <b>абра</b> »	абра	<7,4,->



# Алгоритм LZ77

Алгоритм работы декодера:

1. Декодер последовательно строит свой буфер поиска
2. Декодер считывает метку, находит совпадение в своем буфере, записывает совпадающую подпоследовательность, а затем символ из третьего поля.

**<0,0,a>**

→ «а»

**<0,0,б>**

→ «аб»

**<0,0,p>**

→ «абр»

**<3,1,к>**

→ «абрак»

**<5,1,д>**

→ «абракад»

**<7,4,->**

→ «абракадабра»



# Алгоритм LZ77

## Преимущества:

не вносит искажений в исходные данные;  
не требует хранения дополнительной информации;

## Недостатки:

алгоритм основан на предположении, что похожие образцы сживаемых данных находятся недалеко друг от друга; если содержимое файла этому условию не удовлетворяет, то он будет сжиматься плохо;  
ограниченные размеры буфера; Увеличение размеров буфера могло бы, с одной стороны, улучшить коэффициент сжатия, но с другой стороны, увеличивает время поиска.



# Алгоритм LZ78

Словарный метод сжатия информации.

Идея: алгоритм в явном виде использует словарный подход, генерируя временный словарь во время кодирования и декодирования.

Отличие от LZ77: LZ78 не использует скользящее окно, он хранит словарь из уже просмотренных фраз.



# Алгоритм LZ78

Изначально словарь пуст, а алгоритм пытается закодировать первый символ. На каждой итерации алгоритм ищет в словаре максимальную совпадающую подпоследовательность. Метки этого алгоритма будут состоять уже из двух полей – индекса в словаре самой длинной найденной подпоследовательности и символа, который идет за этой подпоследовательностью в обрабатываемых данных. При этом после кодирования такой пары подпоследовательность с приписанным символом добавляется в словарь, а алгоритм продолжает кодирование со следующего символа.



# Алгоритм LZ78

Одна из главных проблем LZ78 — размер словаря. Он ничем не ограничен и может достигать огромных размеров на больших объемах входных данных. В различных модификациях установлены лимиты на размер словаря: при достижении определенного лимита словарь может «замораживаться» (в него перестают добавляться новые слова), из словаря могут удалять менее используемые или самые старые слова и так далее.

Хорошой структурой для организации словаря является дерево.



# Алгоритм LZW

Словарный метод сжатия информации.

Модификация алгоритма LZ78.

Разработан в 1984 году Терри Уелчем.

Используется в следующих растровых форматах:

GIF (всегда)

TIFF (по желанию)



# Сжатие изображений. LZW

На самом начальном этапе словарь заполняется всеми символами входного алфавита.

В процессе сжатия отыскивается наиболее длинная цепочка, уже записанная в словарь.

Каждый раз, когда новая цепочка элементов не найдена в словаре, она добавляется в словарь; при этом записывается код цепочки, для которой есть совпадение со словарем.

Процесс сжатия выглядит достаточно просто. Мы считываем последовательно символы входного потока и проверяем, есть ли в созданной нами таблице строк такая строка. Если строка есть, то мы считываем следующий символ, а если строки нет, то мы заносим в поток код для предыдущей найденной строки, заносим строку в таблицу и начинаем поиск снова.

Особенность LZW заключается в том, что для декомпрессии нам не надо сохранять таблицу строк в файл для распаковки. Алгоритм построен таким образом, что мы в состоянии восстановить таблицу строк, пользуясь только потоком кодов. Мы знаем, что для каждого кода надо добавлять в таблицу строку, состоящую из уже присутствующей там строки и символа, в которого начинается следующая строка в потоке.



# Характеристики алгоритма LZW

**Коэффициенты сжатия:** ~ 1000, 4, 5/7 (лучший, средний, худший коэффициенты).

(Сжатие в 1000 раз достигается только на одноцветных изображениях размером кратным примерно 7 Мб)

**Класс изображений:** Ориентирован на 8-битные изображения.

**Симметричность:** Почти симметричен, при условии оптимальной реализации операции поиска строки в таблице.

**Характерные особенности:** Ситуация, когда алгоритм увеличивает изображение, встречается крайне редко. LZW универсален – именно его варианты используются в обычных архиваторах.



# Алгоритм Хаффмана

Один из классических алгоритмов, известных с 60-х годов. Относится к классу статистических методов: для кодирования использует статистические свойства сжимаемых данных.

Алгоритм Хаффмана использует особый вид представления элементов – префиксный код. Префиксный код – это код переменной длины, обладающий свойством префикса: менее короткие коды не совпадают с префиксом (начальной частью) более длинных. Такой код позволяет осуществлять взаимно-однозначное кодирование.

Идея алгоритма: использовать частоту появления одинаковых байт в изображении. Сопоставляет символам входного потока, которые встречаются большее число раз, цепочку бит меньшей длины. И, напротив, встречающимся редко – цепочку большей длины. Для сбора статистики необходим дополнительный проход по изображению.

Декодирование осуществляется прямой заменой кода на соответствующий элемент. Алгоритм требует записи в файл таблицы соответствия кодируемых символов и кодирующих цепочек.



# Алгоритм Хаффмана

Алгоритм работы кодера:

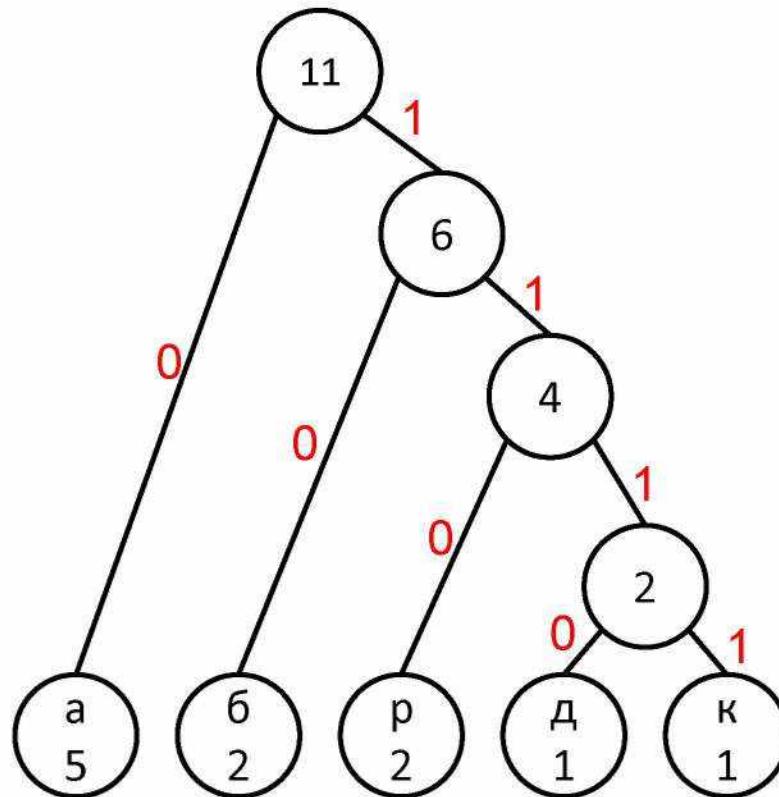
1. Составляется список кодируемых символов, при этом один символ рассматривается как узлы двоичного дерева, с весом, равным частоте появления символа во входном потоке.
2. Символы упорядочиваются по убыванию частот.
3. Из списка выбираются два символа с наименьшими весами.
4. Формируется новый узел, с весом, равным сумме весов двух выбранных узлов, и становится их предком.
5. Если свободных узлов более одного, то переходим к пункту 3.
6. Код соответствующего символа – путь от соответствующего узла с символом к вершине.



# Алгоритм Хаффмана

Пример работы кодера:

1. Входной поток: «абракадабра»
2. Список входных символов: {а, б, д, к, р}. Частоты: {5, 2, 1, 1, 2}.





# Характеристики алгоритма Хаффмана

**Коэффициенты компрессии:** 8, 1.5, 1

(Лучший, средний, худший коэффициенты).

**Класс изображений:** Практически не применяется к изображениям в чистом виде. Обычно используется как один из этапов компрессии в более сложных схемах.

**Симметричность:** 2 (за счет того, что требует двух проходов по массиву сжимаемых данных).

**Характерные особенности:** Единственный алгоритм, который не увеличивает размера исходных данных в худшем случае (если не считать необходимости хранить таблицу перекодировки вместе с файлом).



## Алгоритм Хаффмана с фиксированной таблицей CCITT Group 3

Используется при сжатии черно-белых изображений (один бит на пиксель). Полное название данного алгоритма CCITT Group 3. Это означает, что данный алгоритм был предложен третьей группой по стандартизации Международного Консультационного Комитета по Телеграфии и Телефонии (Consultative Committee International Telegraph and Telephone). Последовательности подряд идущих черных и белых точек в нем заменяются числом, равным их количеству. А этот ряд, уже в свою очередь, сжимается по Хаффману с фиксированной таблицей.



## Характеристики алгоритма CCITT Group 3

**Коэффициенты сжатия:** лучший коэффициент стремится в пределе к 213.(3), средний 2, в худшем случае увеличивает файл в 5 раз.

**Класс изображений:** Двуцветные черно-белые изображения, в которых преобладают большие пространства, заполненные белым цветом.

**Симметричность:** почти симметричен.

**Характерные особенности:** Данный алгоритм чрезвычайно прост в реализации, быстр и может быть легко реализован аппаратно.



# Сжатие изображений с потерями

Многие изображения практически сжимаются стандартными алгоритмами, хотя, на первый взгляд, обладали явной избыточностью. Это привело к созданию нового типа алгоритмов – сжимающих с потерей информации.

Как правило, коэффициент сжатия и, следовательно, степень потерь качества в них можно задавать. При этом достигается компромисс между размером и качеством изображений.



# Сжатие изображений с потерями

Со временем стандартные алгоритмы сжатия, рассчитанные на текстовую информацию, перестали удовлетворять предъявляемым требованиям.

Отличия между текстовой и графической информацией.

1. Текст одномер, а изображения – двухмерны.
2. Огромное количество возможных значений точек полноцветного изображения.
3. Существуют методы, которые автоматически удаляют неважную информацию из растрового изображения.



# Сжатие изображений с потерями

## Оценка потерь

- root mean square – RMS

$$d(x,y) = \sqrt{\frac{\sum_{i=1, j=1}^{n,n} (x_{ij} - y_{ij})^2}{n^2}}$$

- peak-to-peak signal-to-noise ratio – PSNR

$$d(x,y) = 10 \cdot \log_{10} \frac{255^2 \cdot n^2}{\sum_{i=1, j=1}^{n,n} (x_{ij} - y_{ij})^2}$$

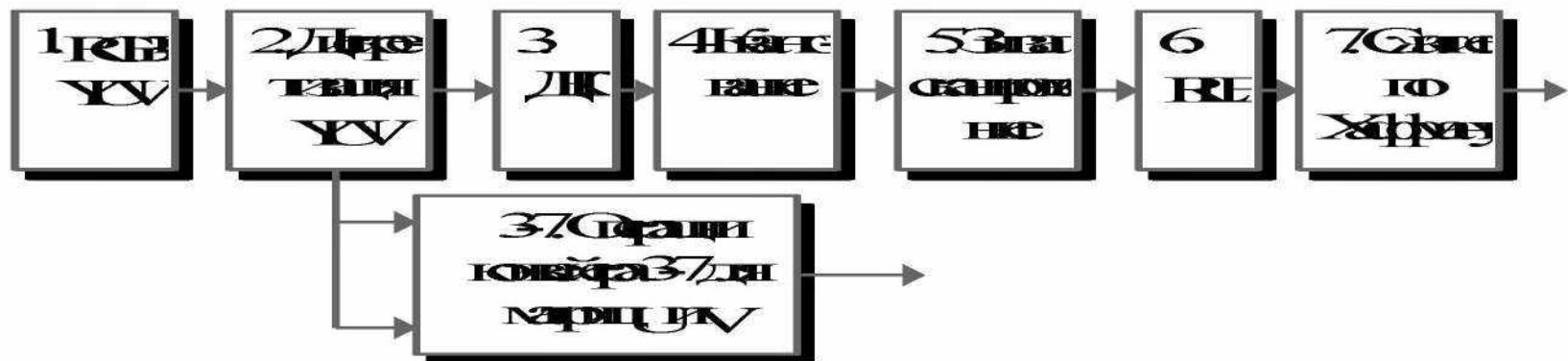
- самой важной «мерой» оценки потерь является мнение наблюдателя, то есть визуальная оценка



# Сжатие изображений с потерями

JPEG (Joint Photographic Expert Group) – один из самых новых и мощных алгоритмов сжатия с потерями.

Конвейер операций, используемый в алгоритме JPEG.

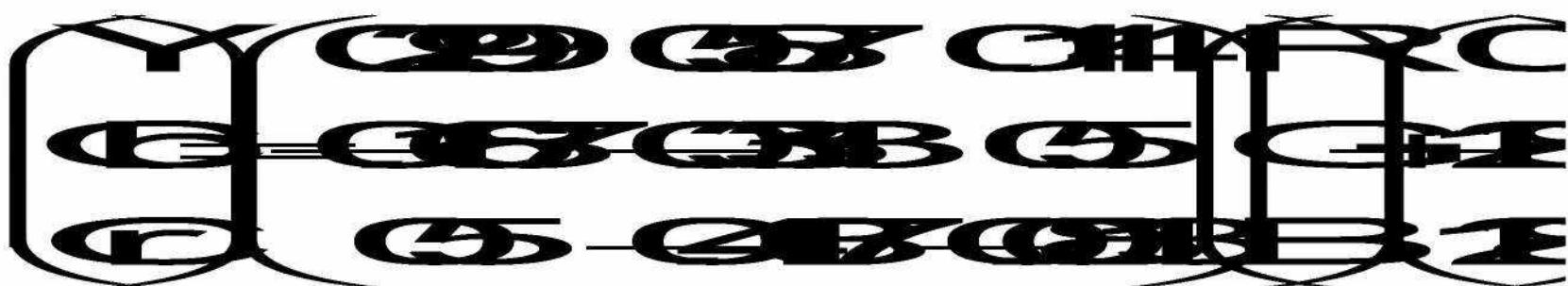




# JPEG

## Шаг 1.

Изображение переводится из цветового пространства RGB в цветовое пространство YCrCb (YUV).

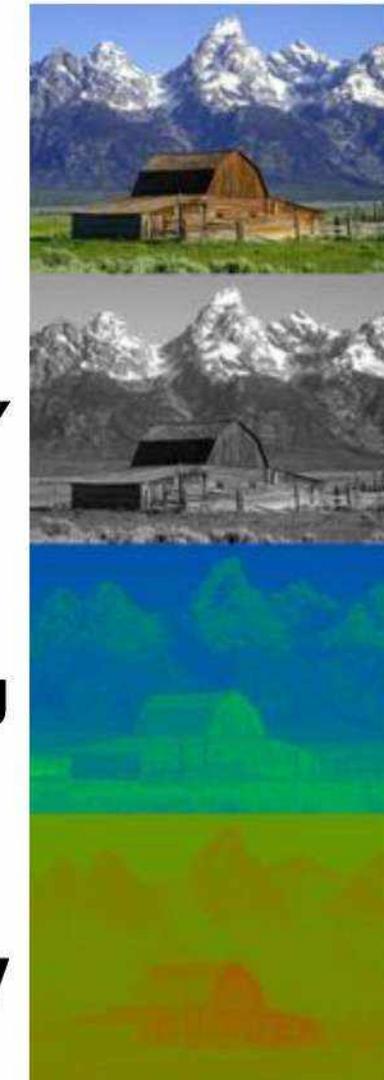




# JPEG

## Шаг 1.

Изображение переводится из цветового пространства RGB в цветовое пространство YCrCb (иногда называют YUV).





# JPEG

## Шаг 2.

### Субдискретизация компонент цветности.

Большая часть визуальной информации, к которой наиболее чувствительны глаза человека, состоит из высокочастотных, полутоновых компонентов яркости (Y) цветового пространства YCbCr. Две других составляющих цветности (Cb и Cr) содержат высокочастотную цветовую информацию, к которой глаз человека менее чувствителен. Следовательно, определенная ее часть может быть отброшена. Например, в изображении размером 1000x1000 пикселей можно использовать яркости всех 1000x1000 пикселей, но только 500x500 пикселей для каждого компонента цветности. При таком представлении каждый пиксель цветности будет охватывать ту же область, что и (для яркости). В результате мы сохраним для каждого блока 2x2 всего блок 2x2 пикселя 6 пиксельных значений (4 значения яркости и по 1 значению для каждого из двух каналов цветности) вместо того, чтобы использовать 12 значений при обычном описании. Практика показала, что уменьшение объема данных на 50% почти незаметно отражается на качестве большинства изображений.



# JPEG

БДУ

## Шаг 2.

### Субдискретизация компонент цветности.

Стандарт JPEG предлагает несколько различных вариантов определения коэффициентов субдискретизации. Канал яркости Y всегда остается с полным разрешением. Каналы Cr и Cb дублируются на соседние пиксели (так называемое «укрупнение» пикселей).

Структура субдискретизации описывается как X:a:b.

(X – ширина блока; a - число выборок в горизонтальном направлении в первой строке;

b - число выборок во второй строке)

4:4:4

(без изменений)

1	2	3	4
1	2	3	4
1	2	3	4

4:2:2

1	2	3	4
1	1	2	
1	1	2	

4:2:0

1	2	3	4
1		2	
1		2	

4:1:1

1	2	3	4
1			
1			

Компонента светимости Y остаётся неизменной!



# JPEG

## Шаг 3.

### Применение дискретного косинус-преобразования (ДКП).

Пиксели каждой цветной компоненты собираются в блоки  $8 \times 8$ , которые называются единицами данных. Далее к каждой единице данных применяется ДКП, описываемое формулой:

$$G_{ij} = \frac{1}{\sqrt{2n}} C_i C_j \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} p_{xy} \cos\left(\frac{(2y+1)j\pi}{2n}\right) \cos\left(\frac{(2x+1)i\pi}{2n}\right), \quad i, j = \overline{1, n}$$

$$C_i = \begin{cases} \frac{1}{\sqrt{2}}, & i = 0 \\ 1, & i > 0 \end{cases}$$

ДКП применяется для каждой единицы данных размера  $n \times n = 8 \times 8$ .

В результате из каждой единицы данных получаем новую матрицу  $8 \times 8$ , которая содержит частотные коэффициенты; коэффициенты в левом верхнем углу соответствуют низкочастотной составляющей изображения, а в правом нижнем — высокочастотной.

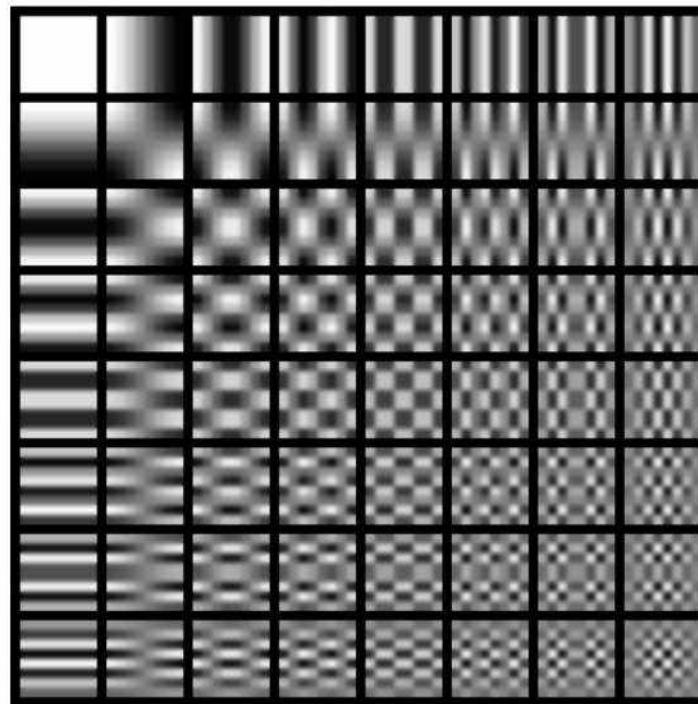


# JPEG

БДУ

Шаг 3.

Применение дискретного косинус-преобразования (ДКП).  
Базовые функции.





# JPEG

## Шаг 4.

### Квантование

Термин «квантование» при использовании в сжатии данных означает округление вещественных чисел до целых или преобразование целых чисел в меньшие целые.

Квантование осуществляется с помощью деления матрицы коэффициентов ДКП на так называемую матрицу квантования (МК).

$$Gq_{ij} = \text{IntegerRound}\left(\frac{G_{ij}}{q_{ij}}\right)$$

В стандарт JPEG включены рекомендованные МК, построенные опытным путем. Матрицы для большего или меньшего коэффициентов сжатия получают путем умножения исходной матрицы на некоторое число *gamma*.



# JPEG

## Шаг 4.

### Квантование. Рекомендуемая JPEG матрица квантования

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99



# JPEG

## Шаг 5.

### Зигзаг-упорядочивание.

Матрица 8x8 переводится в 64-элементный вектор при помощи «зигзаг»-сканирования, т.е. берутся элементы с индексами (0,0), (0,1), (1,0), (2,0)...

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$	$a_{0,6}$	$a_{0,7}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$	$a_{1,6}$	$a_{1,7}$
$a_{1,8}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{3,0}$			
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$				
$a_{4,0}$	$a_{4,1}$	$a_{4,2}$					
$a_{5,0}$	$a_{5,1}$						
$a_{6,0}$	$a_{6,1}$						
$a_{7,0}$	$a_{7,1}$						



# JPEG

БДУ

## Шаг 6.

### **Сжатие методом RLE**

Полученный после зигзаг-сканирования вектор сжимается с помощью алгоритма группового кодирования (RLE).

## Шаг 7.

Получившиеся пары сжимаются кодированием по Хаффману с фиксированной таблицей.



# Характеристики алгоритма JPEG

**Коэффициенты сжатия:** 2-200 (Задается пользователем).

**Класс изображений:** Полноцветные 24 битные изображения или изображения в градациях серого без резких переходов цветов (фотографии).

**Симметричность:** 1.

**Характерные особенности:** В некоторых случаях, алгоритм создает «ореол» вокруг резких горизонтальных и вертикальных границ в изображении (эффект Гиббса). Кроме того, при высокой степени сжатия изображение распадается на блоки 8x8 пикселей.



Белорусский Государственный Университет  
Факультет прикладной математики и информатики

# Компьютерная графика

Василевский Константин

Викторович

17 лекций,

17 лабораторных занятий



# Основы обработки цифровых изображений.

- Введение в компьютерную обработку изображений.
- Представление изображения.
- Поэлементные операции над изображениями.
- Линейные фильтры: определение, сглаживающие фильтры, контрастоповышающие фильтры, разностные фильтры.
- Нелинейные фильтры: примеры нелинейных фильтров, морфологические операторы.



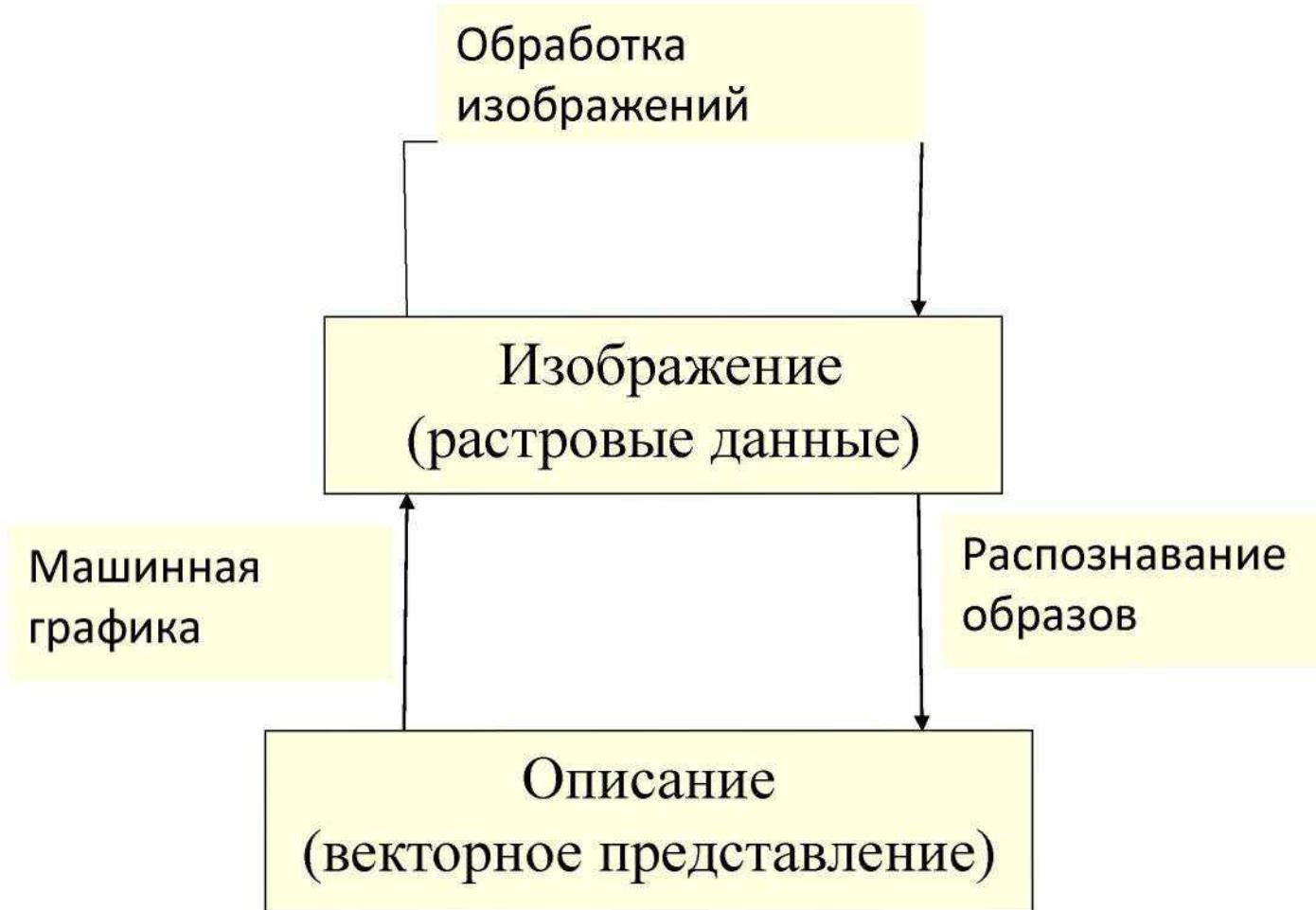
# Введение

Направления:

- машинная графика (computer graphics): визуализация (создание) изображения
- обработка изображений (image processing): преобразование изображения
- распознавание изображений (computer vision): получение описания объектов на изображении



# Введение





# Обработка изображений

Области применения:

- повышение качества изображений для улучшения визуального восприятия человеком;
- обработка изображений для их хранения, передачи и представления в автономных системах машинного зрения (сжатие видеоданных, а также анализ, распознавание и интерпретация зрительных образов для принятия решений и управления поведением технических систем).



# Подходы к обработке изображений

Классификация 1:

- методы обработки в пространственной области (*пространственные методы*);
- методы обработки в частотной области (*частотные методы*).

Классификация 2:

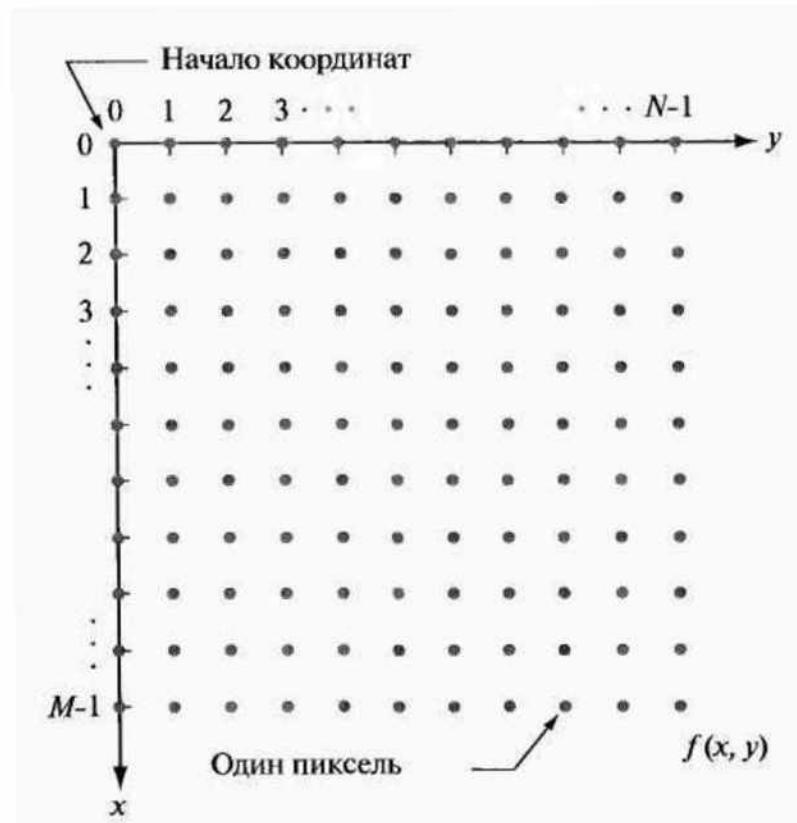
- *поэлементные операции*: перекодировка, арифметические и логические операции, преобразование яркости;
- *локальные операции*: операции клеточной логики, фильтрация, сглаживание, выделение контуров, операции математической морфологии;
- *глобальные операции*: геометрические преобразования (поворот, масштабирование, нелинейные преобразования координат), спектральные преобразования (Фурье, Адамара и т.п.).



# Представление изображения

Будем рассматривать  
полутоновые изображения  
(изображения в оттенках  
серого)

- $f(m, n)$
- $m=0, \dots, M-1$
- $n=0, \dots, N-1$





# Представление изображения

$$f = \begin{bmatrix} f(0, 0) & f(0, 1) & \dots & \dots & f(0, N-1) \\ f(1, 0) & f(1, 1) & \dots & \dots & f(1, N-1) \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ f(M-1, 0) & f(M-1, 1) & \dots & \dots & f(M-1, N-1) \end{bmatrix}$$

$$f(m, n) \in [0; 255]$$



# Поэлементные преобразования над изображениями

- модифицируется только одно значение (например уровень яркости) всех отдельных пикселей изображения, при этом геометрические характеристики изображения не изменяются;
- если для преобразования используется монотонная функция (возрастающая или убывающая), то всегда существует обратная операция, позволяющая получить исходное изображение из преобразованного. Если же используемая функция не является монотонной, то определенная часть информации может быть безвозвратно утрачена;
- данные операции служат главным образом для улучшения восприятия и не вносят никакой новой информации в изображение.



# Поэлементные преобразования над изображениями

$$f'(m, n) = \Psi(f(m, n))$$

Операция является линейной ,если:

$$\Psi(af + bg) = a \Psi(f) + b \Psi(g)$$

Функция  $\Psi$  называется функцией градационного преобразования  
*(преобразования интенсивностей)*



# Поэлементные преобразования над изображениями

1. Добавление к изображению  $f(m, n)$  целочисленной константы  $\chi$  (положительной или отрицательной).

$$f'(m, n) = f(m, n) \pm \chi \quad \chi > 0, \chi < 0$$

2. Преобразование изображения в негатив.

$$f_{neg}'(m, n) = 255 - f(m, n)$$

3. Умножение изображения на константу  $\alpha$ .

$$f'(m, n) = \alpha f(m, n) \quad \alpha > 1, \alpha < 1$$



# Поэлементные преобразования над изображениями

4. Степенные преобразования.

$$f'(m, n) = (f(m, n))^{\gamma} \quad \gamma > 1, \gamma < 1$$

$$f'(m, n) = 255 \left( \frac{f(m, n)}{f_{\max}} \right)^{\gamma}$$

5. Логарифмическое преобразование

$$f'(m, n) = \log(1 + f(m, n))$$

$$f'(m, n) = 255 \frac{\log(1 + f(m, n))}{\log(1 + f_{\max})}$$



# Поэлементные преобразования над изображениями

$$f'(m, n) = \alpha_1 f_1(m, n) + \alpha_2 f_2(m, n) + \dots + \alpha_N f_N(m, n)$$

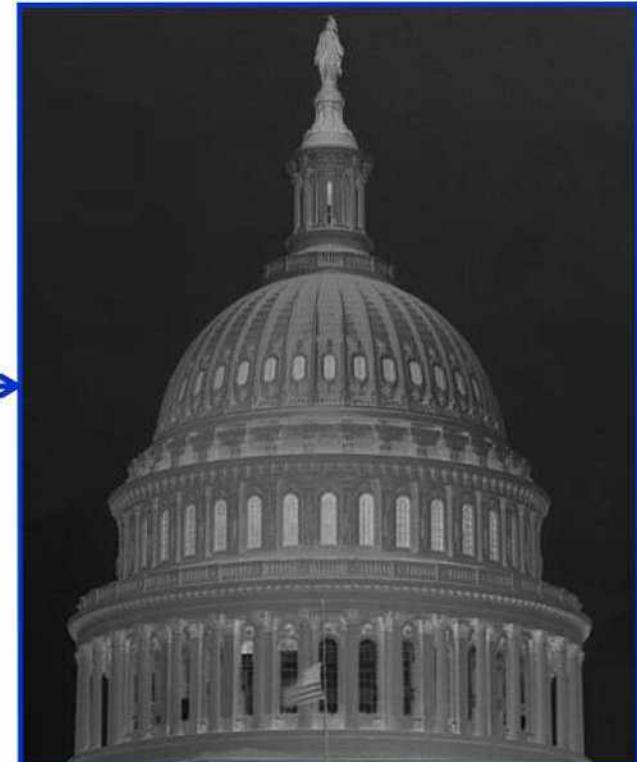
- сложение изображений;
- вычитание изображений;
- умножение изображений;
- деление изображений;
- линейная комбинация изображений.



# Поэлементные преобразования над изображениями



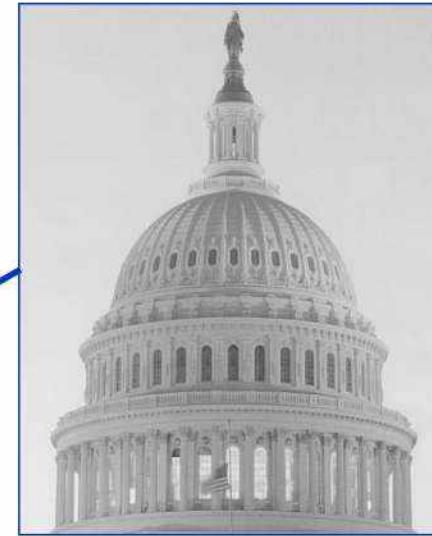
негатив





# Поэлементные преобразования над изображениями

+ константа  $>0$



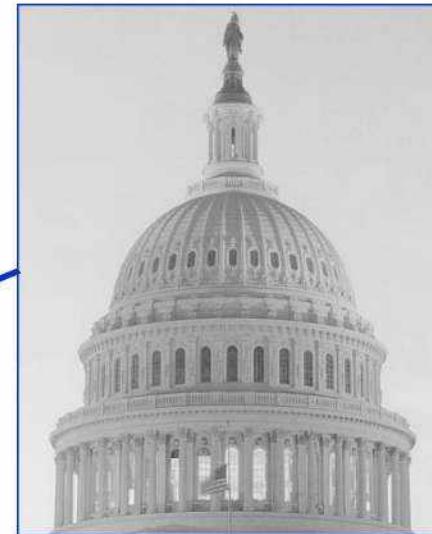
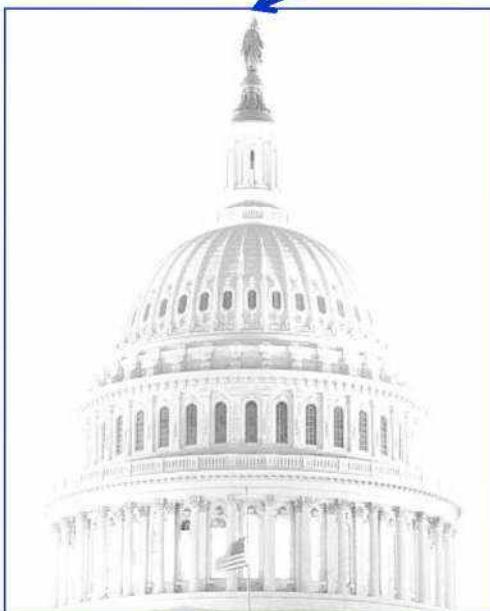
+ константа  $<0$





# Поэлементные преобразования над изображениями

\*константа  $>1$

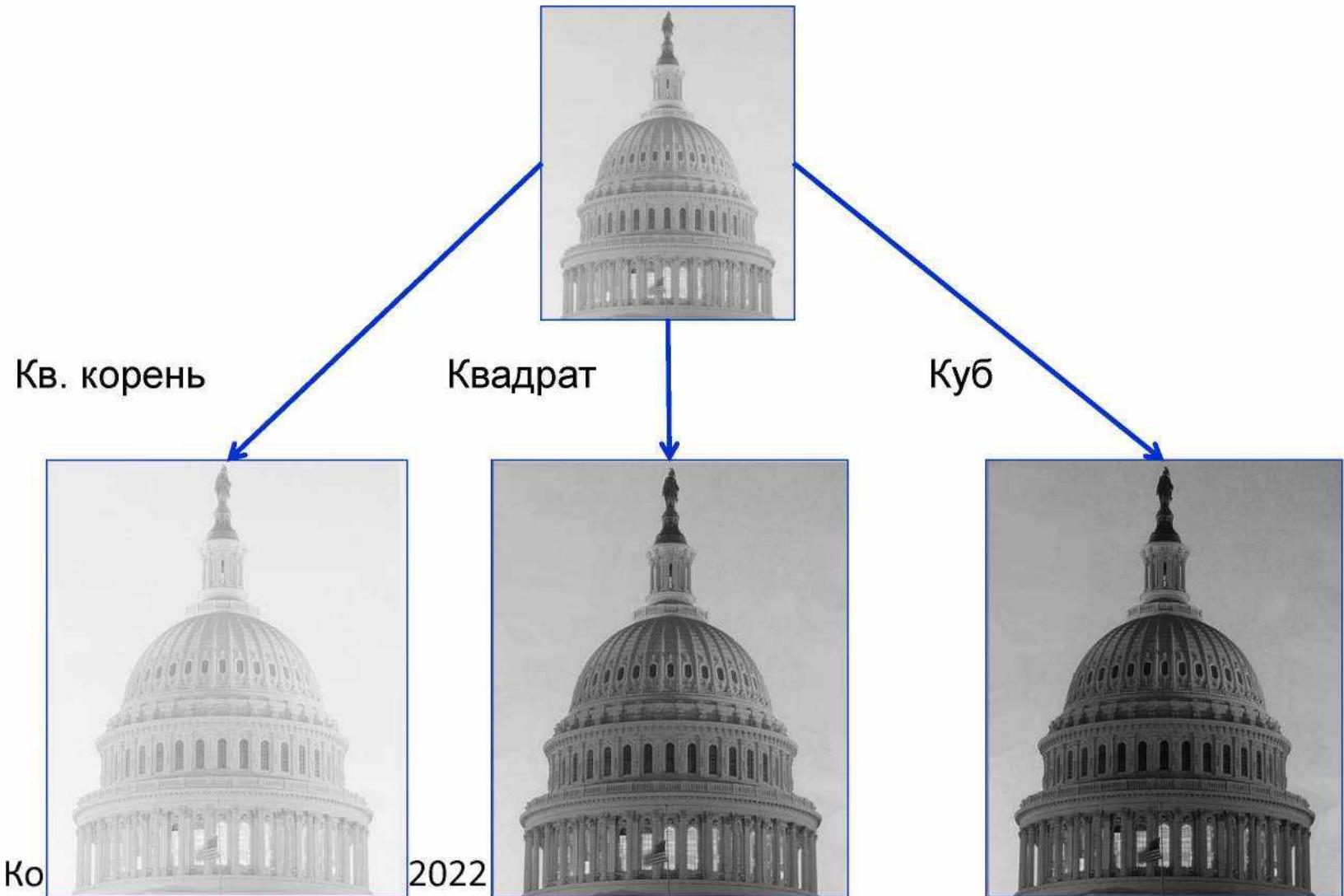


\*константа  $<1$



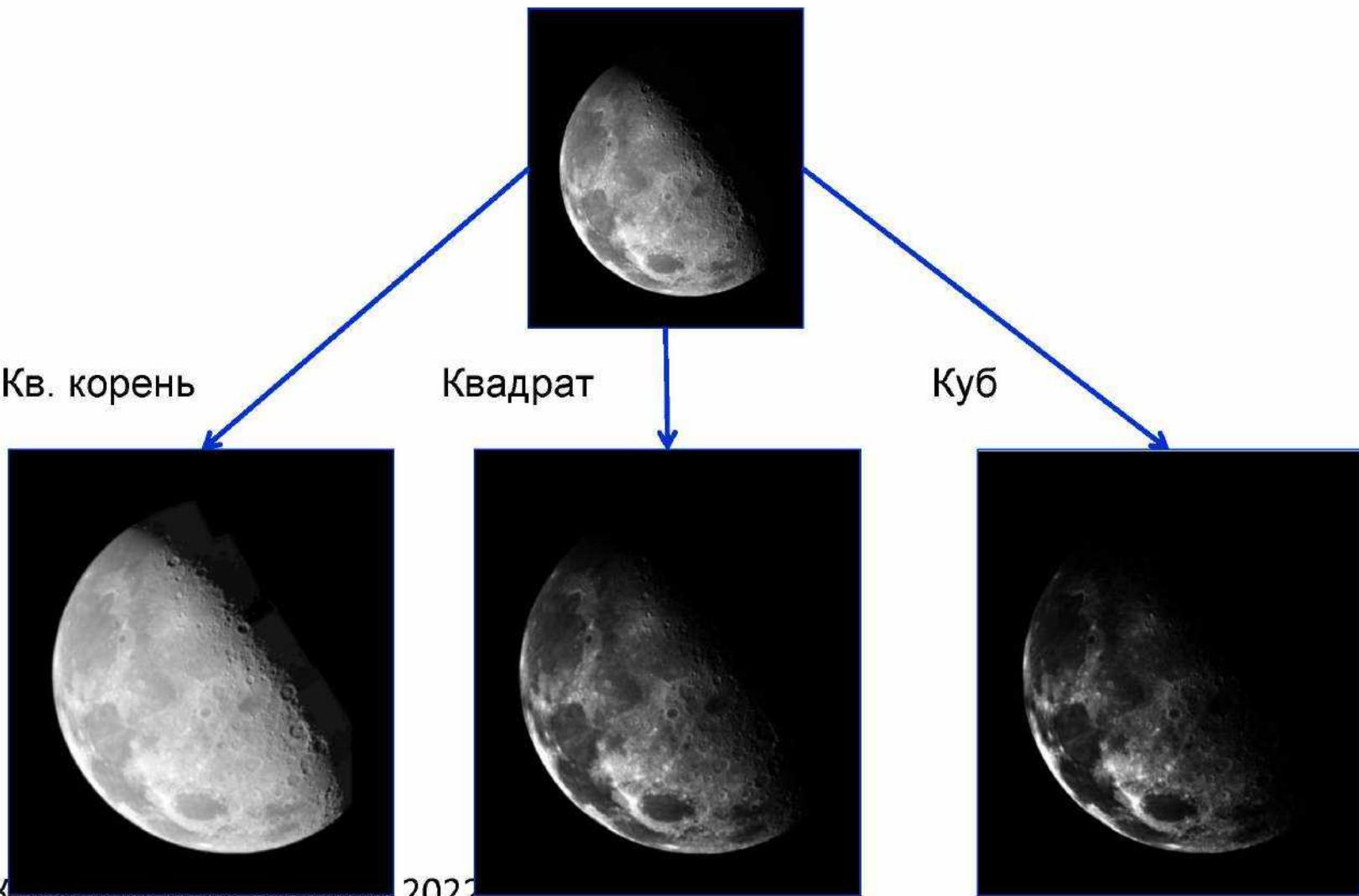


# Поэлементные преобразования над изображениями



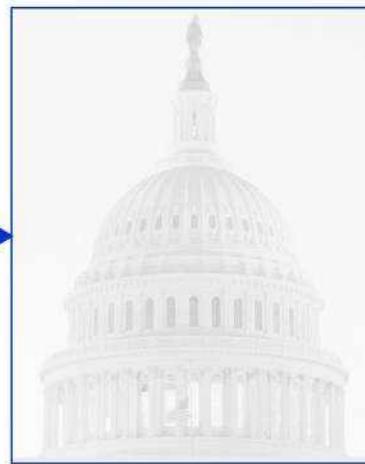


# Поэлементные преобразования над изображениями





# Поэлементные преобразования над изображениями



Логарифм





# Таблицы преобразований

LUT (*look-up table*)

- переход к табличному заданию функции преобразования;
- высокое быстродействие;
- гибкость процедуры обработки;

0	...	255
$f(0)$	...	$f(255)$



# Линейное контрастирование

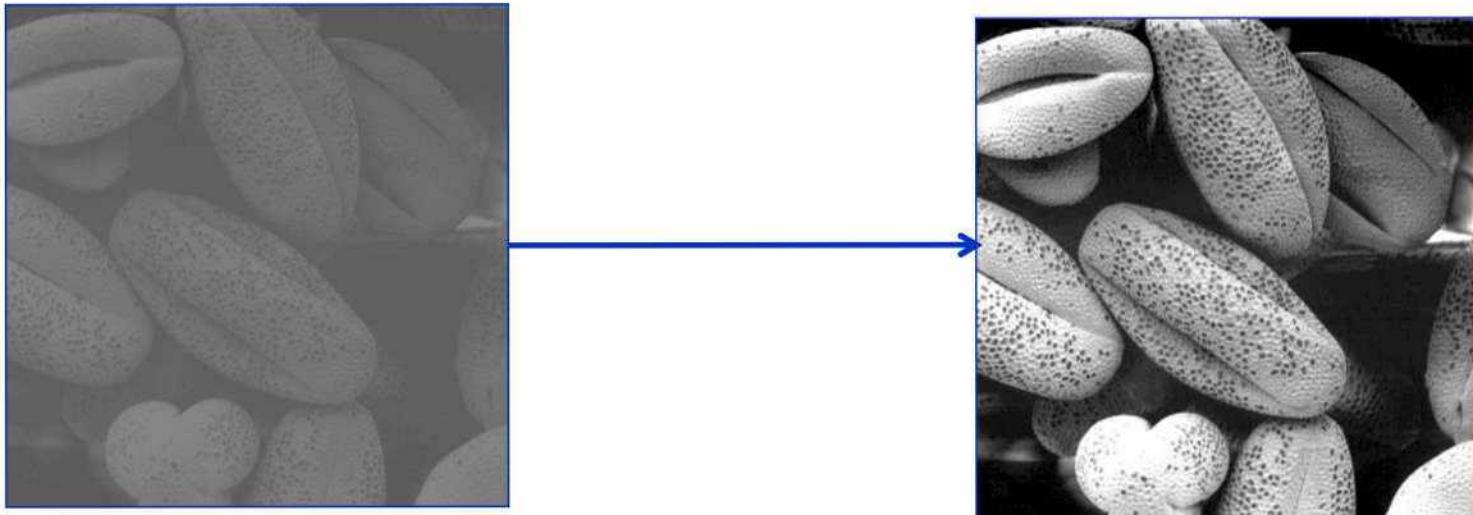
- Расширяет диапазон яркостей
  - Применяется к малоконтрастным изображениям
- 
- $[f_{min}, f_{max}]$  - реальный диапазон яркостей исходного изображения
  - $[f_{MIN}, f_{MAX}]$  - допустимый диапазон для данного типа изображений

$$f'(m, n) = \frac{f(m, n) - f_{min}}{f_{max} - f_{min}} (f_{MAX} - f_{MIN}) + f_{MIN}$$

$$f'(m, n) = \frac{255}{f_{max} - f_{min}} (f(m, n) - f_{min})$$



# Линейное контрастирование





# Основы фильтрации в пространственной области. Цифровые фильтры

- Подавление на изображении нежелательного шума.
- Улучшение резкости изображения, подчеркивание мелких деталей на изображении.
- Устранение на изображении определенных дефектов.
- Улучшение изображений плохого качества.
- Восстановление изображений.



# Основы фильтрации в пространственной области. Цифровые фильтры

$$f'(m, n) = T(f(m, n))$$

$f(m, n)$  – входное изображение

$f'(m, n)$  – обработанное изображение

$T$  – оператор над  $f$ , определенный в некоторой окрестности точки  $(m, n)$

*Цифровые фильтры бывают:*

- линейные

$$T(af + bg) = a T(f) + b T(g)$$

- нелинейные



# Основы фильтрации в пространственной области. Цифровые фильтры

Дискретная свертка  $w$ :

$$f'(m, n) = (w \times f)(m, n) = \frac{1}{W} \sum_{i, j \in K} f(m + i, n + j) w(i, j)$$

$K$  - окрестность рассматриваемой точки изображения  $(m, n)$ ,

$w(i, j)$  – веса пикселей в окрестности  $K$

$W$  – сумма весов



# Основы фильтрации в пространственной области. Цифровые фильтры

$$\begin{bmatrix} w(-1, -1) & w(0, -1) & w(-1, 1) \\ w(0, -1) & w(0, 0) & w(0, 1) \\ w(1, -1) & w(1, 0) & w(1, 1) \end{bmatrix} \quad \text{- маска } 3 \times 3 \text{ (фильтр, шаблон)}$$

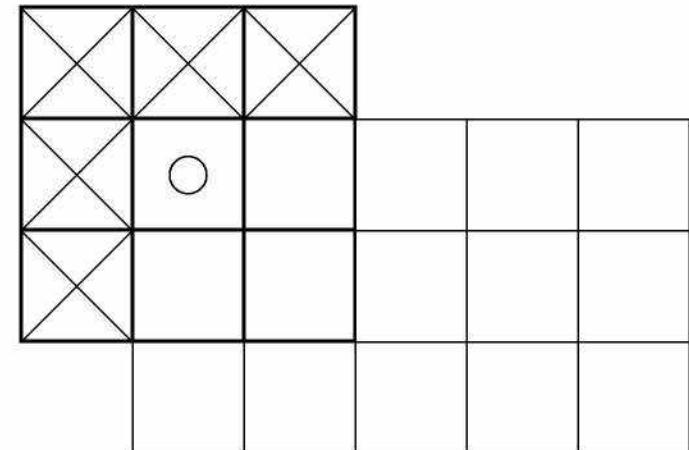
$$f'(m, n) = \frac{1}{W} \sum_{i=-1}^1 \sum_{j=-1}^1 f(m+i, n+j) w(i, j)$$

$$W = \sum_{i=-1}^1 \sum_{j=-1}^1 w(i, j)$$



# Основы фильтрации в пространственной области. Цифровые фильтры

- Ограничение перемещения центра маски по изображению.
- Используется только та часть маски, которая полностью находится внутри изображения.
- Расширение изображения за его границы добавлением строк и столбцов из нулей (или других постоянных значений).





# Основы фильтрации в пространственной области. Цифровые фильтры

- Низкочастотные пространственные фильтры,  
*(усредняющие или сглаживающие фильтры)*
- Высокочастотные пространственные фильтры  
*(пространственными фильтрами повышения резкости).*



# Низкочастотные (сглаживающие фильтры)

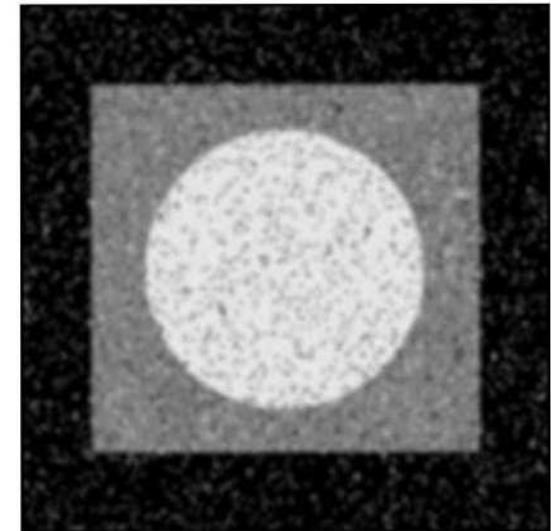
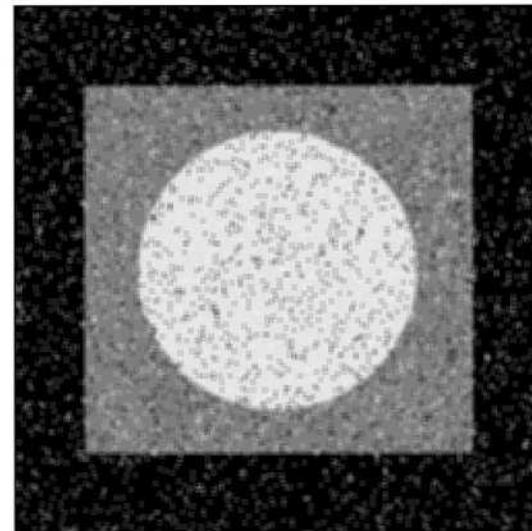
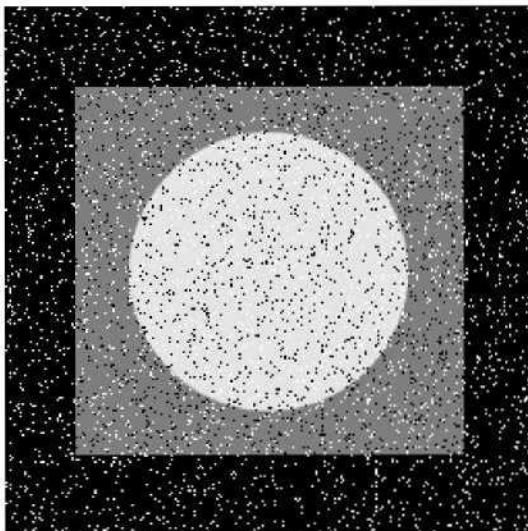
Однородный усредняющий фильтр

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad f'(m, n) = \frac{1}{W} \sum_{i=-1}^1 \sum_{j=-1}^1 f(m+i, n+j) w(i, j)$$
$$W = 9$$



# Низкочастотные (сглаживающие) фильтры

Однородный усредняющий фильтр





# Низкочастотные (сглаживающие фильтры)

Однородный усредняющий фильтр:  
модификация

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & a & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad f'(m, n) = \frac{1}{W} \sum_{i=-1}^1 \sum_{j=-1}^1 f(m+i, n+j) w(i, j)$$
$$W = 8 + a \quad a = 0, 1, 2, 4, 12$$



# Низкочастотные (сглаживающие фильтры)

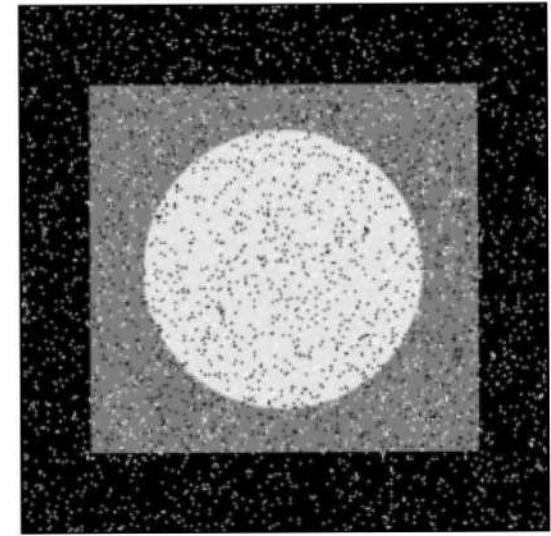
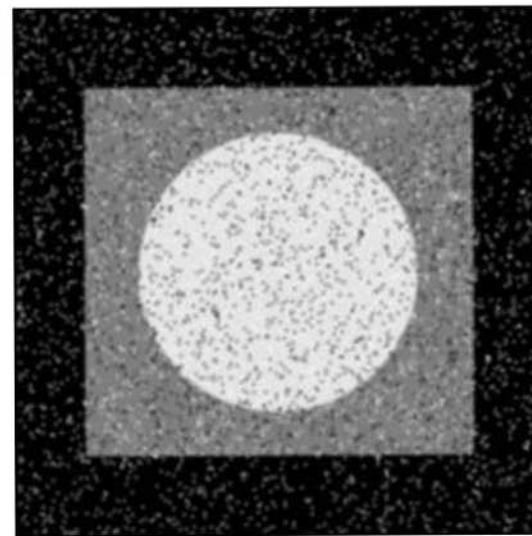
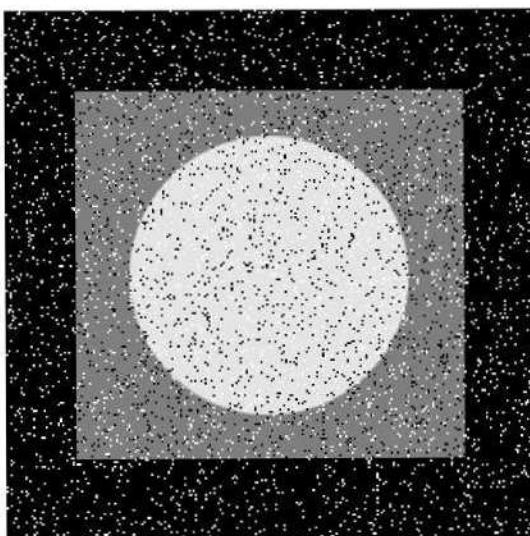
Фильтры Гаусса:

$$\begin{bmatrix} 1 & b & 1 \\ b & b^2 & b \\ 1 & b & 1 \end{bmatrix} \quad f'(m, n) = \frac{1}{W} \sum_{i=-1}^1 \sum_{j=-1}^1 f(m+i, n+j) w(i, j)$$
$$W = (2+b)^2 \quad b = 1, 2, 3, 4$$



# Низкочастотные (сглаживающие) фильтры

Фильтры Гаусса:  $b = 2, 4$





# Низкочастотные (сглаживающие фильтры)

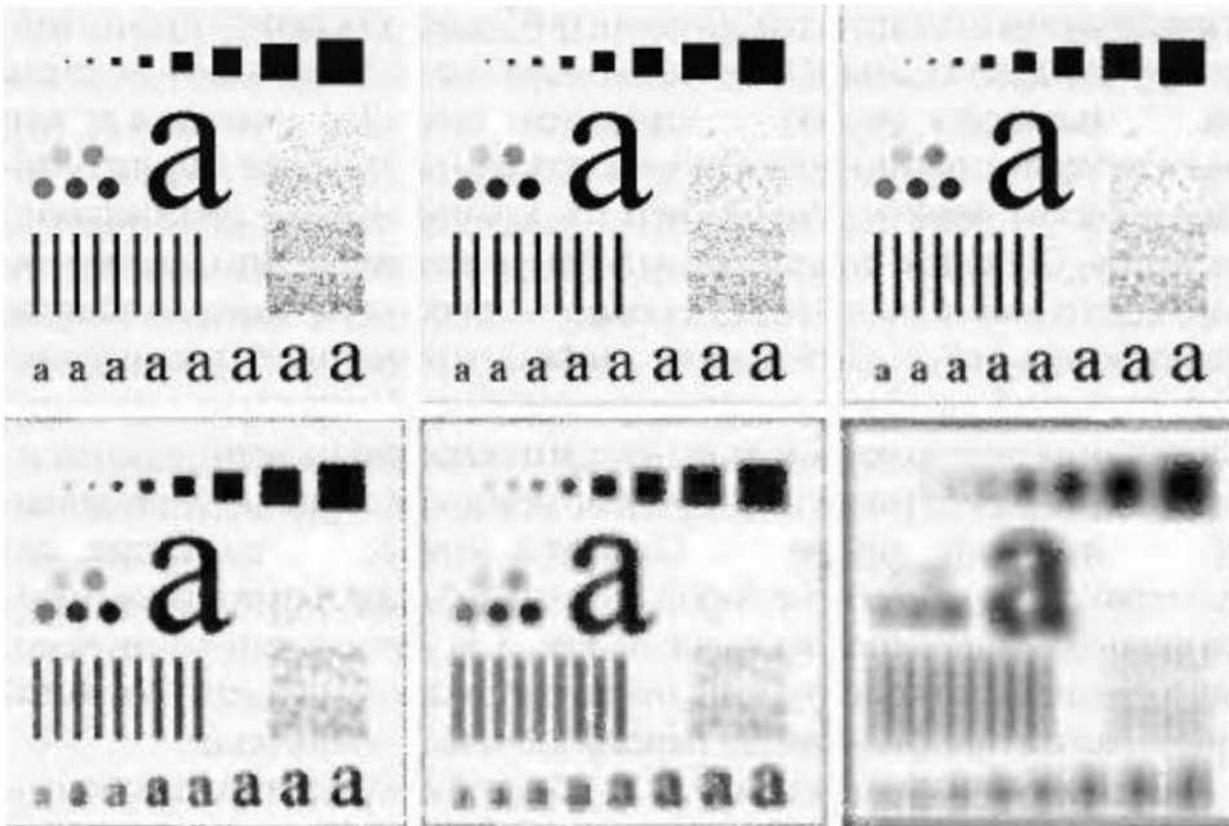
Разновидности фильтров Гаусса:

$$\begin{bmatrix} 1 & 4 & 1 \\ 4 & 12 & 4 \\ 1 & 4 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 & 1 \\ 3 & 16 & 3 \\ 1 & 3 & 1 \end{bmatrix}$$

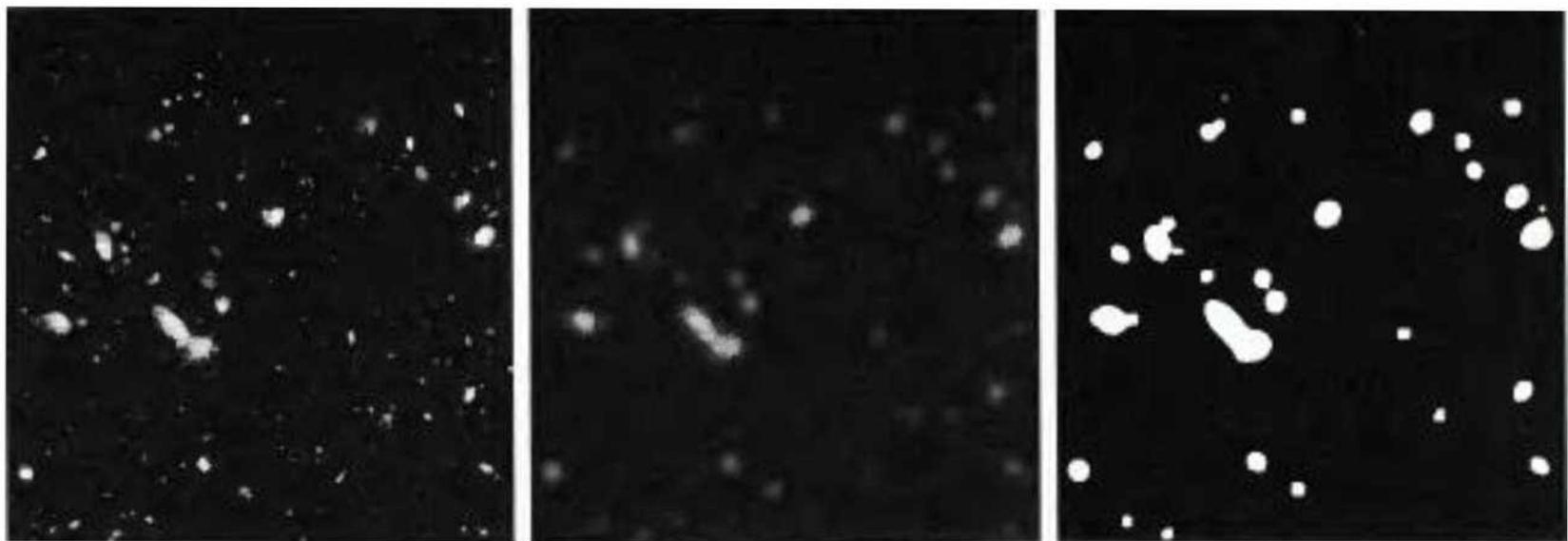


# Низкочастотные (сглаживающие фильтры)





# Низкочастотные (сглаживающие) фильтры





# Высокочастотные фильтры (фильтры повышения резкости)

Фильтры, которые сохраняют без изменений или усиливают высокочастотные составляющие спектра сигнала.

Основаны на операции дифференцирования изображения.



# Высокочастотные фильтры (фильтры повышения резкости)

Первая производная определяется градиентом.

*Двухмерный градиент изображения  $f(x, y)$  в точке  $(x, y)$  – это вектор:*

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$G_x = \frac{\partial f}{\partial x} = f(x+1, y) - f(x, y);$$
$$G_y = \frac{\partial f}{\partial y} = f(x, y+1) - f(x, y).$$



# Высокочастотные фильтры (фильтры повышения резкости)

Модуль вектора градиента:

$$\nabla f = |\nabla \mathbf{f}| = \sqrt{G_x^2 + G_y^2}$$

Это значение максимальной скорости изменения функции  $f$  в точке  $(x, y)$ , причем максимум достигается в направлении вектора  $\nabla \mathbf{f}$ .

Направление вектора градиента определяется углом:

$$\alpha(x, y) = \arctg \left( \frac{G_x}{G_y} \right)$$



# Высокочастотные фильтры (фильтры повышения резкости)

Для упрощения вычислений:

Вместо:

$$\nabla f = \sqrt{G_x^2 + G_y^2}$$

Используем:

$$\nabla f = |G_x| + |G_y|$$



# Высокочастотные фильтры (фильтры повышения резкости)

Вторая производная определяется лапласианом изображения  $f(x, y)$  в точке  $(x, y)$ :

$$L(f) = \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\begin{aligned}\frac{\partial^2 f}{\partial x^2} &= \frac{\partial f}{\partial x} \Big|_{(x+1)} - \frac{\partial f}{\partial x} \Big|_x = [f(x+1, y) - f(x, y)] - [f(x, y) - f(x-1, y)] = \\ &= f(x+1, y) + f(x-1, y) - 2f(x, y)\end{aligned}$$

$$\begin{aligned}\frac{\partial^2 f}{\partial y^2} &= [f(x, y+1) - f(x, y)] - [f(x, y) - f(x, y-1)] = \\ &= f(x, y+1) + f(x, y-1) - 2f(x, y).\end{aligned}$$

$$L(f(x, y)) = [f(x-1, y) + f(x+1, y) + f(x, y-1) + f(x, y+1)] - 4f(x, y)$$



# Высокочастотные фильтры (фильтры повышения резкости)

Вторая производная определяется лапласианом изображения  $f(x, y)$  в точке  $(x, y)$ :

$$L(f(x, y)) = [f(x-1, y) + f(x+1, y) + f(x, y-1) + f(x, y+1)] - 4f(x, y)$$

$$\Rightarrow \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



# Высокочастотные фильтры (фильтры повышения резкости)

Фильтры повышения резкости:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 20 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



# Высокочастотные фильтры (фильтры повышения резкости)

Лапласиан гауссиана *LoG* (*Laplacian of Gauss*)

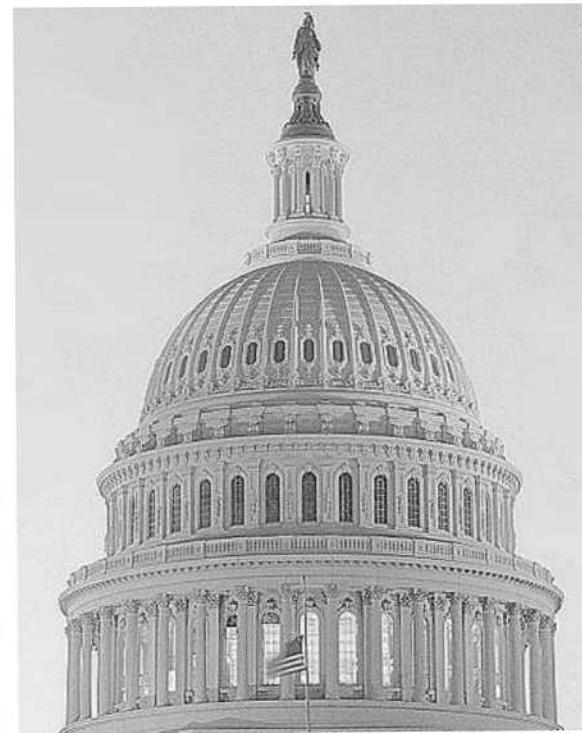
$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$



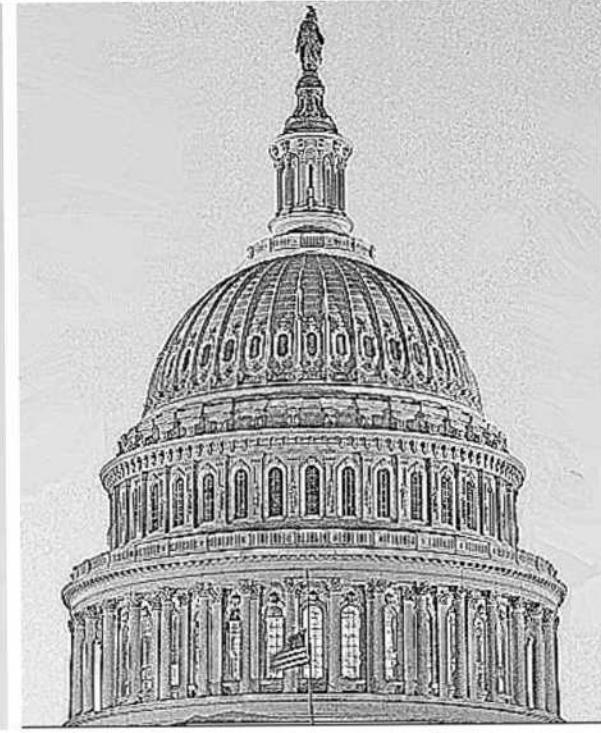
# Высокочастотные фильтры (фильтры повышения резкости)



Исходное



Лапласиан



LOG



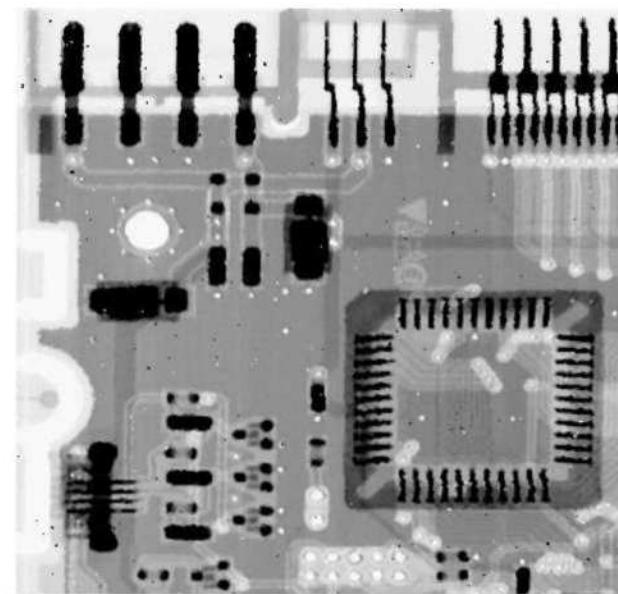
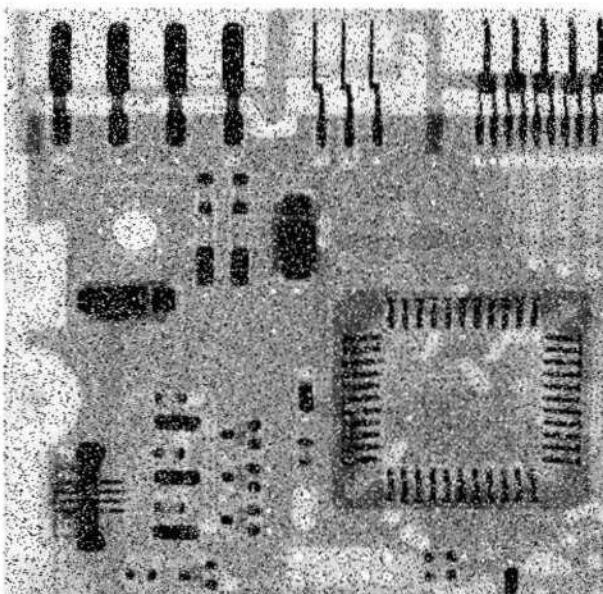
# Нелинейные фильтры, основанные на порядковых статистиках

Отклик такого фильтра определяется предварительным упорядочиванием (ранжированием) значений пикселей, покрываемых маской фильтра, и последующим выбором значения, находящегося на определенной позиции упорядоченной последовательности (т.е. имеющего определенный ранг). Собственно фильтрация сводится к замещению исходного значения пикселя (в центре маски) на полученное значение отклика фильтра.



# Медианный фильтр

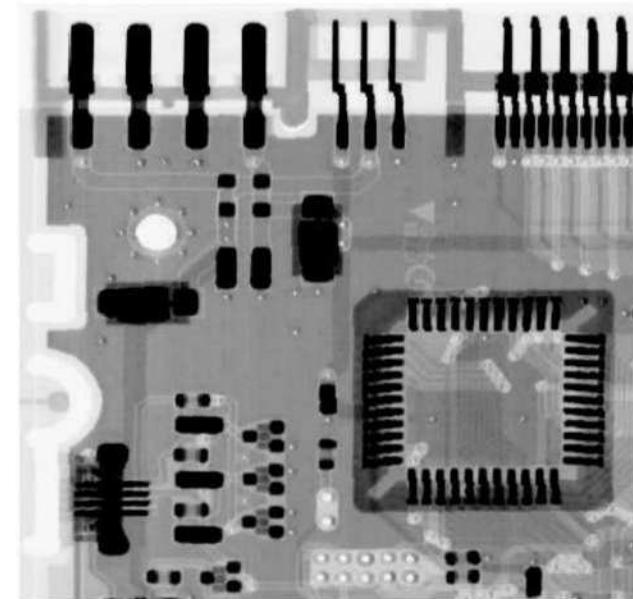
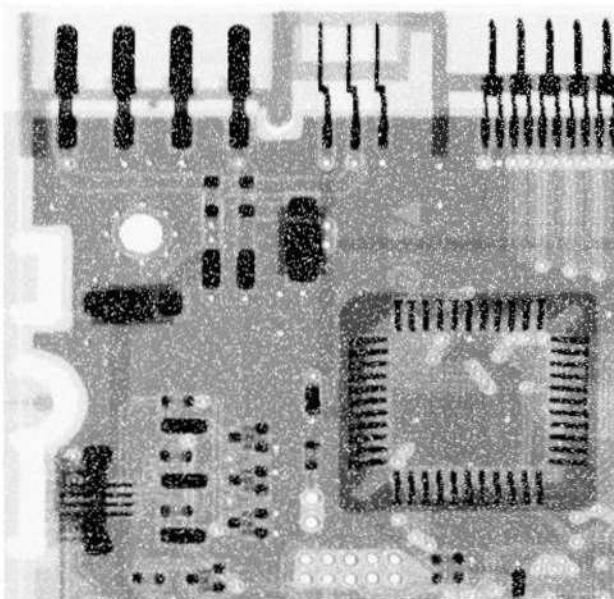
Заменяет значение яркости пикселя на значение медианы распределения яркостей всех пикселей в окрестности





## Фильтр минимума

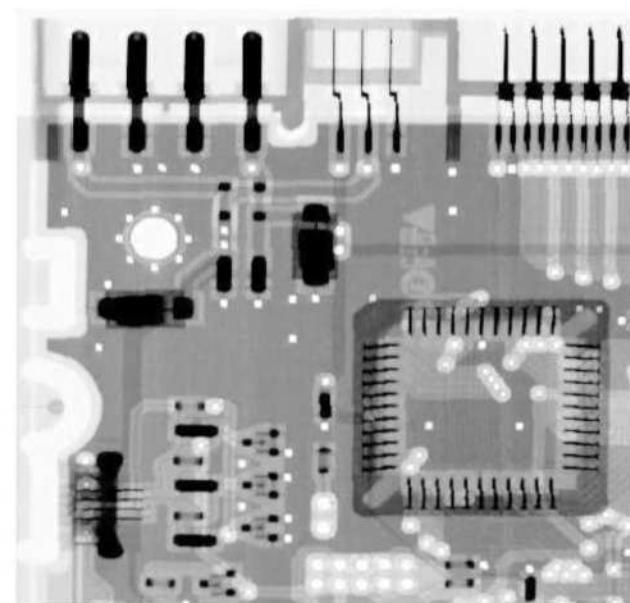
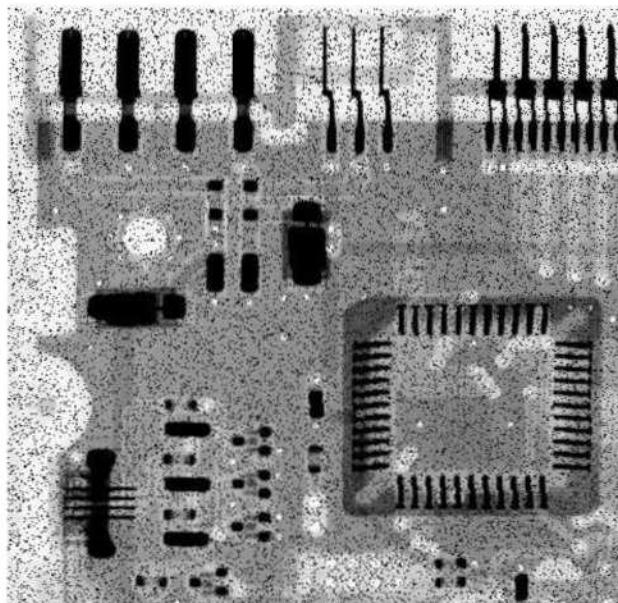
Заменяет значение яркости пикселя на значение минимальной яркости в окрестности





## Фильтр максимума

Заменяет значение яркости пикселя на значение максимальной яркости в окрестности





# Основы обработки цветных изображений

Два подхода:

- каждая цветовая компонента изображения обрабатывается отдельно, а затем результирующее цветное изображение составляется из компонент, обработанных по отдельности;
- непосредственная работа с цветными пикселями; значение пикселя изображения рассматривается как вектор



# Основы обработки цветных изображений

$$g(x, y) = T[f(x, y)] \quad f(x, y) = \begin{bmatrix} R_f(x, y) \\ G_f(x, y) \\ B_f(x, y) \end{bmatrix}$$

Первый подход:

$$g'(x, y) = \begin{bmatrix} R_g(x, y) \\ G_g(x, y) \\ B_g(x, y) \end{bmatrix} = \begin{bmatrix} T[R_f(x, y)] \\ T[G_f(x, y)] \\ T[B_f(x, y)] \end{bmatrix}$$



# Основы обработки цветных изображений

$$g(x, y) = T[f(x, y)] \quad f(x, y) = \begin{bmatrix} R_f(x, y) \\ G_f(x, y) \\ B_f(x, y) \end{bmatrix}$$

Второй подход:

$$g(x, y) = \begin{bmatrix} R_g(x, y) \\ G_g(x, y) \\ B_g(x, y) \end{bmatrix} = \begin{bmatrix} T_1[R_f(x, y), G_f(x, y), B_f(x, y)] \\ T_2[R_f(x, y), G_f(x, y), B_f(x, y)] \\ T_3[R_f(x, y), G_f(x, y), B_f(x, y)] \end{bmatrix}$$



# Основы обработки цветных изображений

Сглаживающие фильтры: результат один

$$\begin{bmatrix} R_g(x, y) \\ G_g(x, y) \\ B_g(x, y) \end{bmatrix} = g(x, y) = T[f(x, y)] = \frac{1}{K} \sum_{(x, y) \in K} f(x, y) = \begin{bmatrix} \frac{1}{K} \sum_{(x, y) \in K} R_f(x, y) \\ \frac{1}{K} \sum_{(x, y) \in K} G_f(x, y) \\ \frac{1}{K} \sum_{(x, y) \in K} B_f(x, y) \end{bmatrix}$$

Медианный фильтр: результат различен



Белорусский Государственный Университет  
Факультет прикладной математики и информатики

# Компьютерная графика

Василевский Константин

Викторович

17 лекций,

17 лабораторных занятий



# Основы обработки цифровых изображений.

- Гистограмма изображения
- Сегментация изображений
- Пороговая обработка
- Бинаризация
- Морфологическая обработка
- Библиотека OpenCV



# Гистограмма изображения

**Гистограмма** показывает распределение на изображении пикселей с различными уровнями яркости из заданного диапазона, т.е. представляет собой дискретную функцию, несущую информацию о числе элементов изображения, имеющих заданный уровень яркости.

$$h(j_i) = h_i, \quad i = 0, 1, \dots, L - 1$$

$h_i$  – число элементов изображения, имеющих яркость  $j_i$



# Гистограмма полутонового изображения

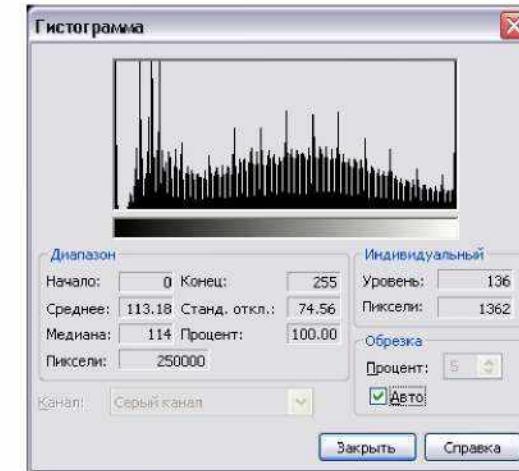
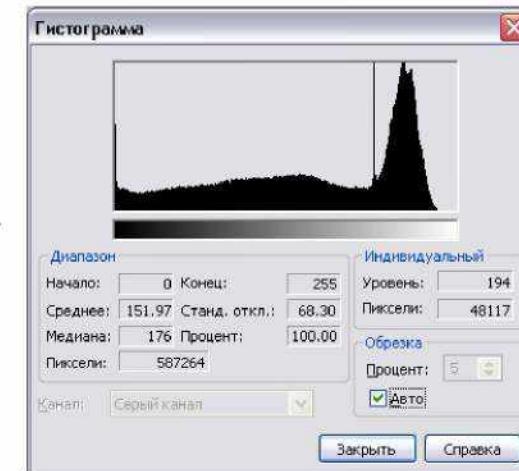
$$h(i) = h_i, \quad i = 0, 1, \dots, 255 \quad (1)$$

$$h_i = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g_i(m, n), \quad i = 0, 1, \dots, 255 \quad (2)$$

$$g_i(m, n) = \begin{cases} 1, & \text{если } f(m, n) = i, \\ 0 & \text{в противном случае.} \end{cases} \quad (3)$$

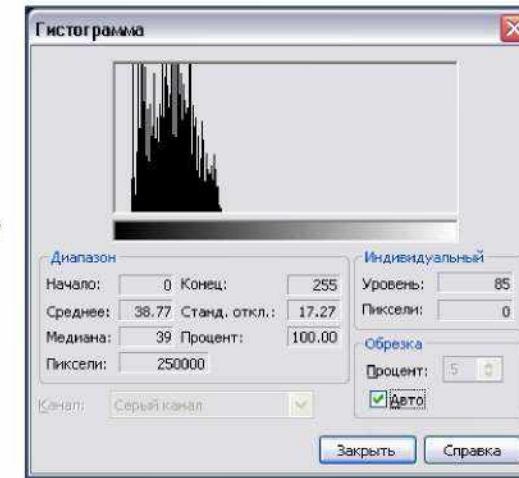
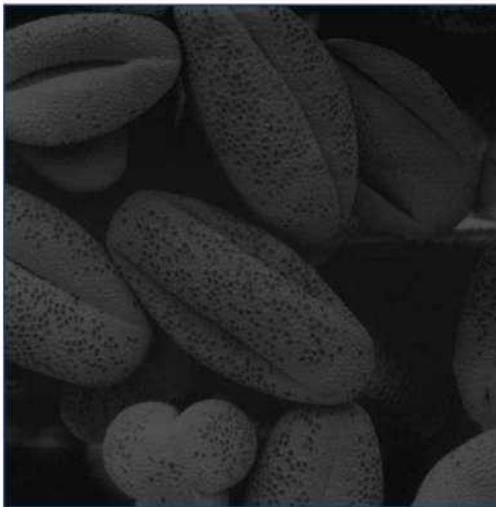
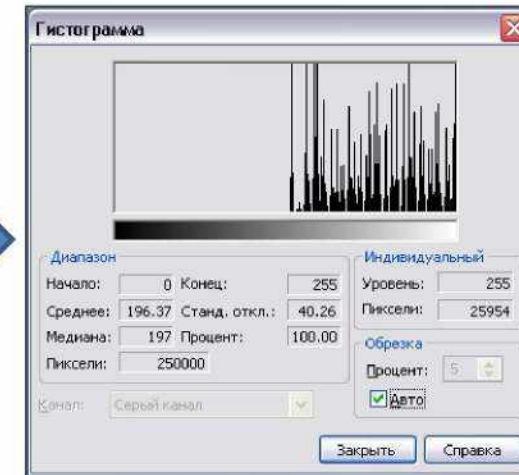
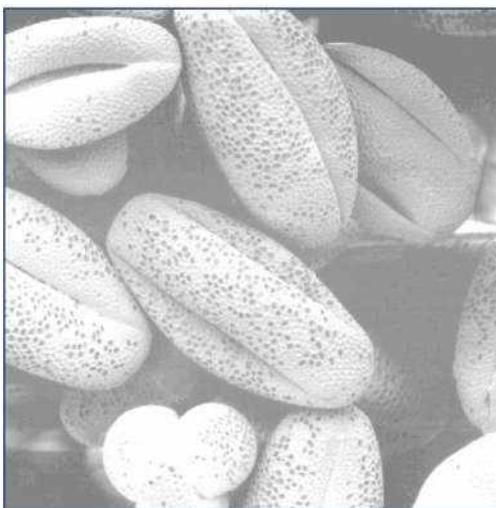


# Гистограмма изображения



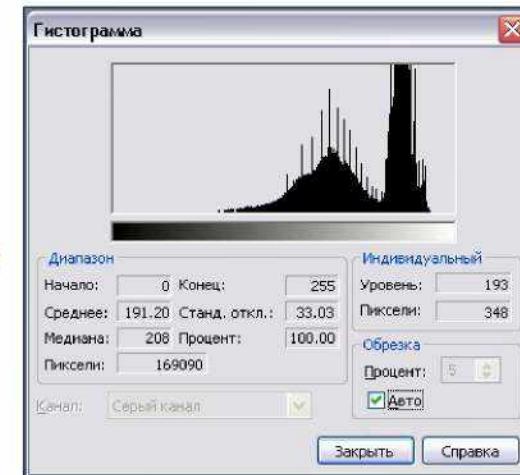
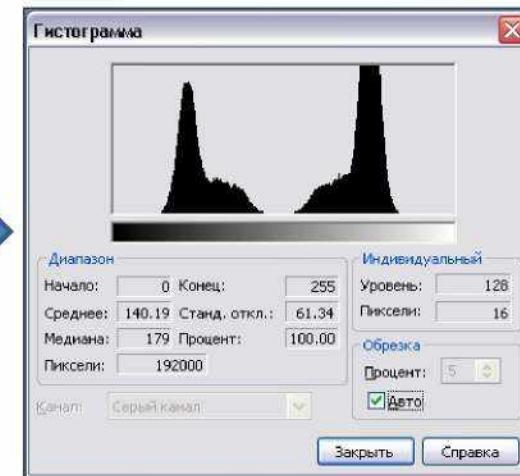


# Гистограмма изображения





# Гистограмма изображения





# Нормализация гистограммы изображения

$$h'(i) = \frac{h_i}{H}$$

где  $h_i$  – количество пикселей с яркостью  $i$

$$H = \sum_{i=0}^{255} h_i$$
 – общее число пикселей на изображении

$h'(i)$  – вероятность появления пикселя с яркостью  $i$

$$0 \leq h'(i) \leq 1$$

$$\sum_i h'(i) = 1$$



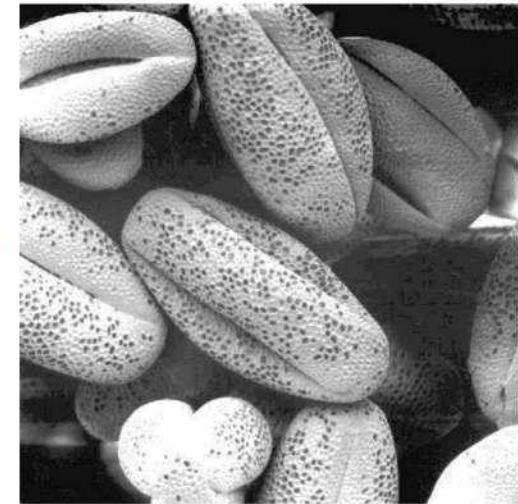
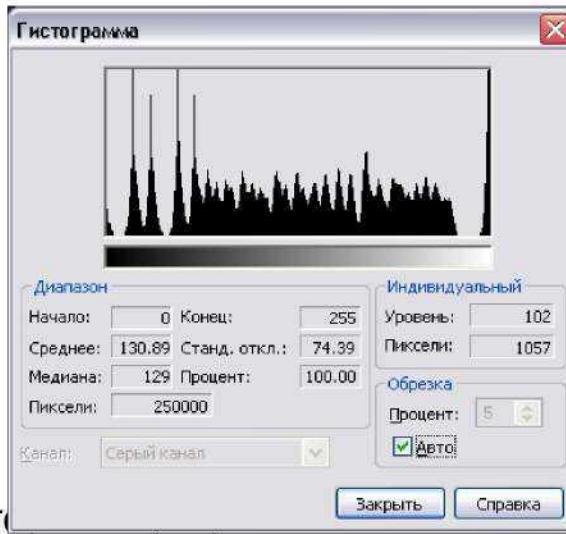
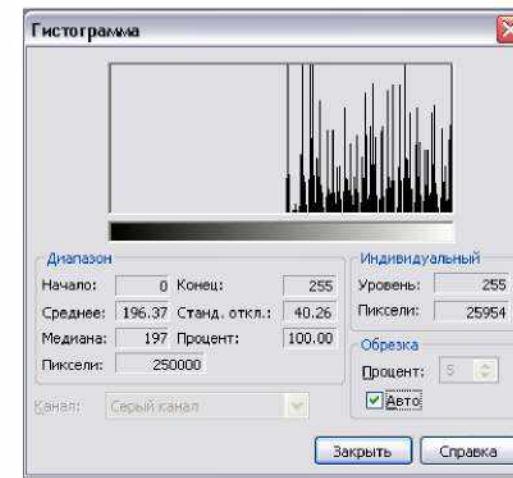
# Эквализация гистограммы изображения

Процесс **эквалайзации (выравнивания)** гистограммы предполагает, что идеальная гистограмма должна иметь форму, близкую к прямоугольнику, т.е. ни один уровень яркости не должен преобладать над другими.

Данный метод формирует на выходе изображение, уровни яркости которого равномерно покрывают весь диапазон возможных значений, и является полностью «автоматическим».



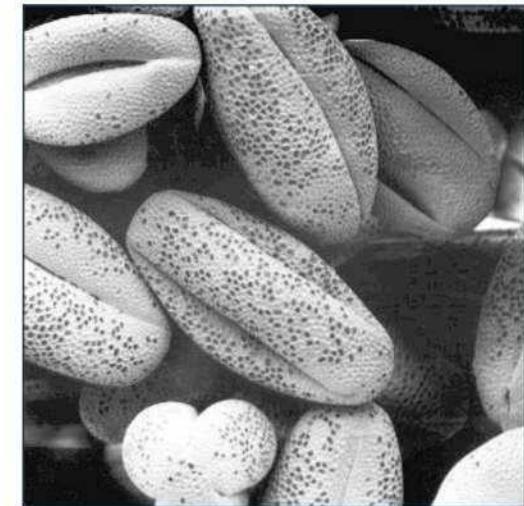
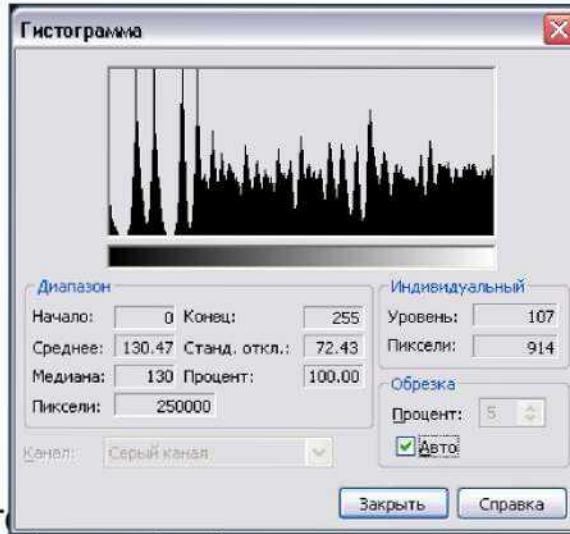
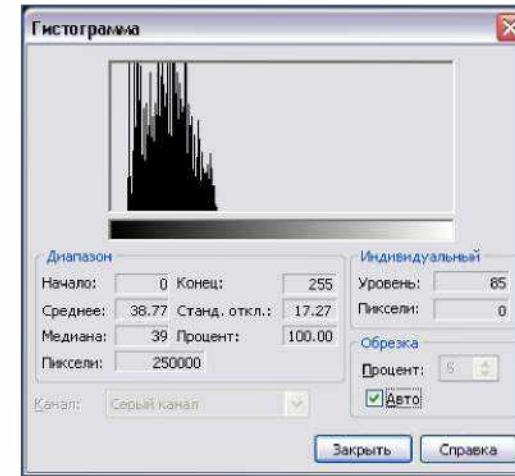
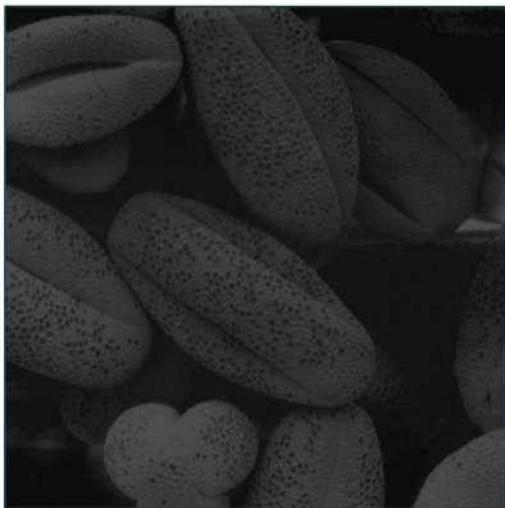
# Эквализация гистограммы



Компьютер



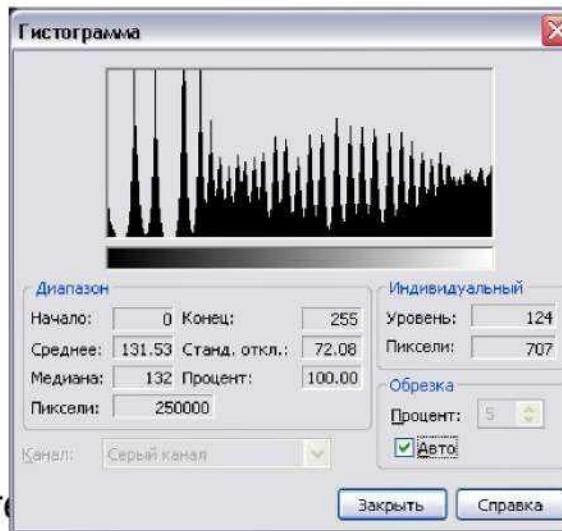
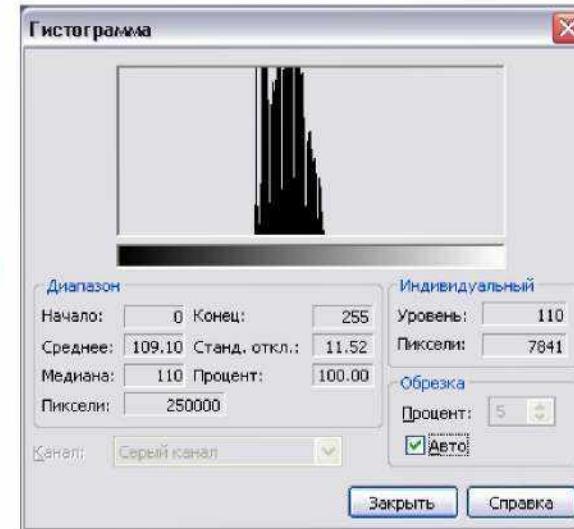
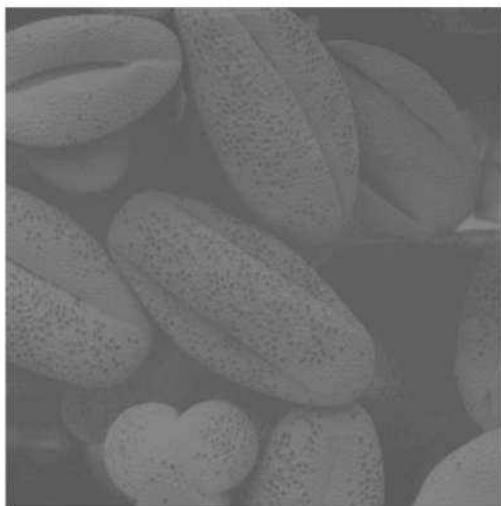
# Эквализация гистограммы



Компьютер



# Эквализация гистограммы



Компьюте...



# Эквализация (линейное выравнивание) гистограммы изображения

$f(m,n)$  – исходное полутоновое изображение размера  $M \times N$

1. Построение гистограммы исходного изображения

$$h(i) = h_i, i = 0, 1, \dots, 255$$

2. Нормализация гистограммы

$$H = \sum_{i=0}^{255} h_i \quad h'(i) = \frac{h_i}{H}, i = 0, 1, \dots, 255$$

3. Построение кумулятивной гистограммы

$$Sh(i) = \sum_{j=1}^i h'(j), \quad i = 0, 1, \dots, 255$$

4. Вычисление новых значений яркости каждого пикселя изображения

$$f'(m, n) = 255 \cdot Sh(f(m, n))$$



# Эквализация (нелинейное выравнивание) гистограммы изображения

$H = M \times N$  – площадь изображения (т.е. количество пикселей)

$L$  – число допустимых уровней яркости (256 в нашем случае).

Необходимо, чтобы на каждый уровень яркости приходилось по  $H/L$  пикселей.

Если значение яркости некоторого уровня  $i$  в  $k$  раз больше среднего  $H/L$ , то этот уровень следует отобразить в  $k$  отдельных уровнях.

Если значение яркости некоторого уровня  $i$  в  $k$  раз меньше среднего  $H/L$ , то этот уровень следует объединить с  $k$  соседними уровнями с примерно равным значением.



# Эквализация (выравнивание) гистограммы изображения

Процедура преобразования гистограммы может применяться как к изображению в целом, так и к отдельным его фрагментам. Последнее может быть полезным при обработке нестационарных изображений, содержание которых существенно различается по своим характеристикам на различных участках. В этом случае лучшего эффекта можно добиться, применяя гистограммную обработку к отдельным участкам.



# Методы повышения контраста

- Линейное контрастирование

$$f'(m, n) = \frac{255}{f_{\max} - f_{\min}} (f(m, n) - f_{\min})$$

- Эквализация гистограммы

Задание для л/р: реализовать оба метода повышения контраста для полуточновых изображений. Сравнить результат.



# Эквализация гистограммы цветного изображения

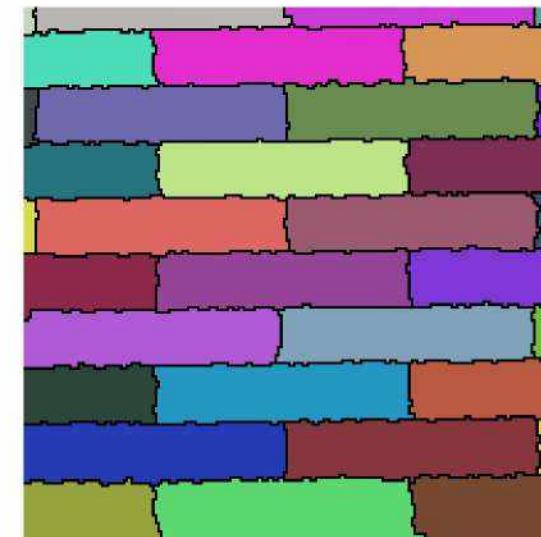
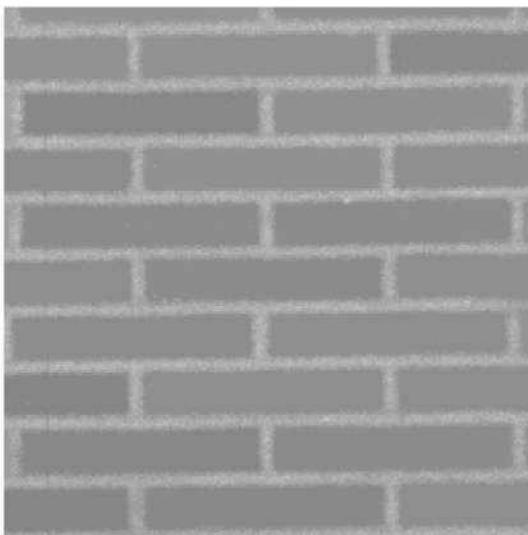
- Работа с цветом в пространстве RGB.  
Процедура эквализации гистограммы для каждой цветовой компоненты в отдельности.
- Переход к цветовому пространству HSV.  
Применение процедуры эквализации только для компоненты яркости (V).

Задание для л/р: реализовать оба варианта эквализации гистограммы цветного изображения. Сравнить результат.



# Сегментация изображений

**Сегментация** изображений – процесс разделения изображения на однородные в определенном смысле области.





# Пороговая обработка

Пороговые преобразования занимают центральное место в прикладных задачах сегментации изображений.

Пороговая обработка – самый простой метод сегментации, ориентированный на обработку изображений, отдельные однородные участки которых различаются средней яркостью.



# Пороговая обработка

## Бинаризация

**Бинаризация** - операция порогового разделения, которая в результате дает бинарное изображение. Целью операции бинаризации является уменьшение количества информации, содержащейся на изображении. В процессе бинаризации исходное полутоновое изображение, имеющее  $L$  уровней яркости, преобразуется в черно-белое изображение, пиксели которого имеют только два значения – 0 и 1. Поскольку количество информации в бинарном изображении почти на порядок меньше, чем в совпадающим с ним по размерам полутоновом изображении, то бинарное изображение легче обрабатывать, хранить и пересылать.



# Пороговая обработка Бинаризация

$f(x,y)$  – изображение  $(f(m,n))$

$T$  – порог

$f(x,y) > T$  - точка объекта

$f(x,y) < T$  - точка фона

В общем случае:

$T = T(x,y,p(x,y),f)$

- глобальная – значение Т одинаково для всех точек изображения, т.е.  $T=T(f)$
- локальная - значение Т зависит от пространственных координат x и y, т.е.  $T=T(x,y,f)$
- адаптивная - значение Т зависит от локальных характеристик окрестности рассматриваемой точки, т.е.  $T=T(x,y,f, p(x,y))$



# Бинаризация

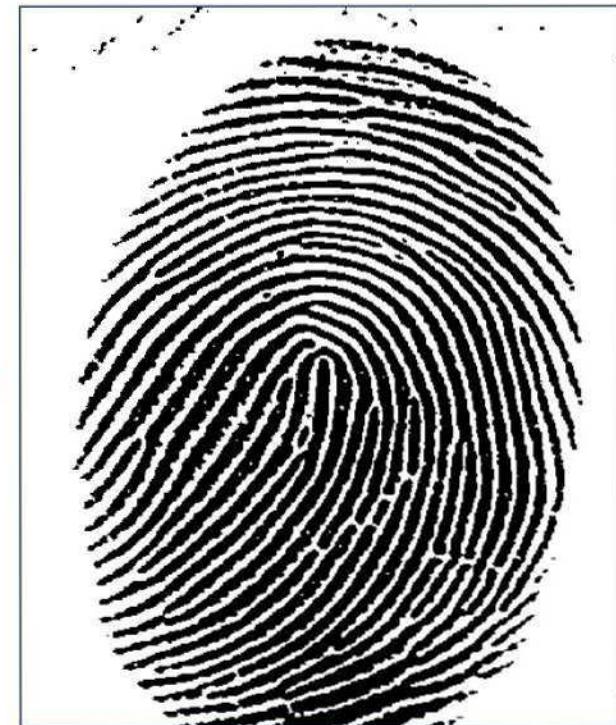
- Бинаризация с нижним порогом
- Бинаризация с верхним порогом
- Бинаризация с двойным ограничением
- Бинаризация по условию\*
- Неполная пороговая обработка\*
- Многоуровневое пороговое преобразование\*

\* - Данная операция формирует изображение, не являющееся бинарным, но состоящее из сегментов с различной яркостью



# Бинаризация с нижним порогом

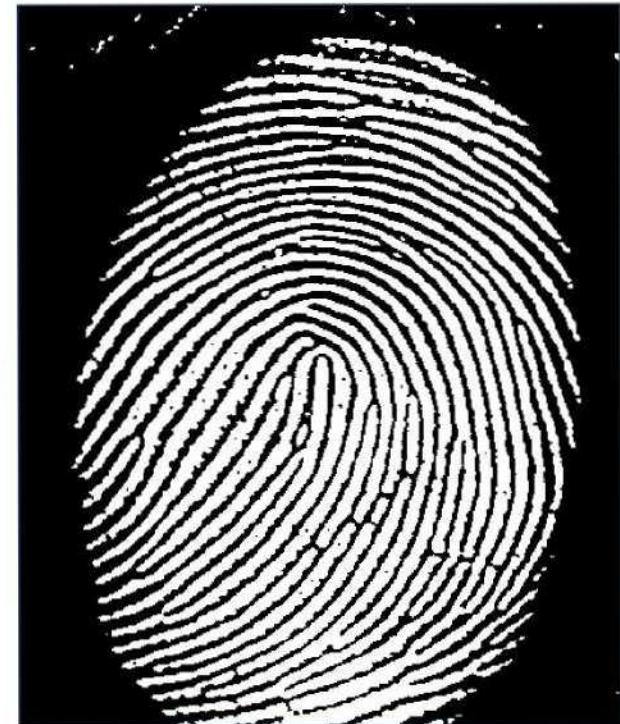
$$f'(m, n) = \begin{cases} 0, & f(m, n) \leq t; \\ 1, & f(m, n) > t, \end{cases}$$





# Бинаризация с верхним порогом

$$f'(m, n) = \begin{cases} 0, & f(m, n) \geq t; \\ 1, & f(m, n) < t. \end{cases}$$





# Бинаризация с двойным ограничением

$$f'(m, n) = \begin{cases} 0, & f(m, n) \leq t_1; \\ 1, & t_1 < f(m, n) \leq t_2; \\ 0, & f(m, n) > t_2, \end{cases} \quad t_1 < t_2$$



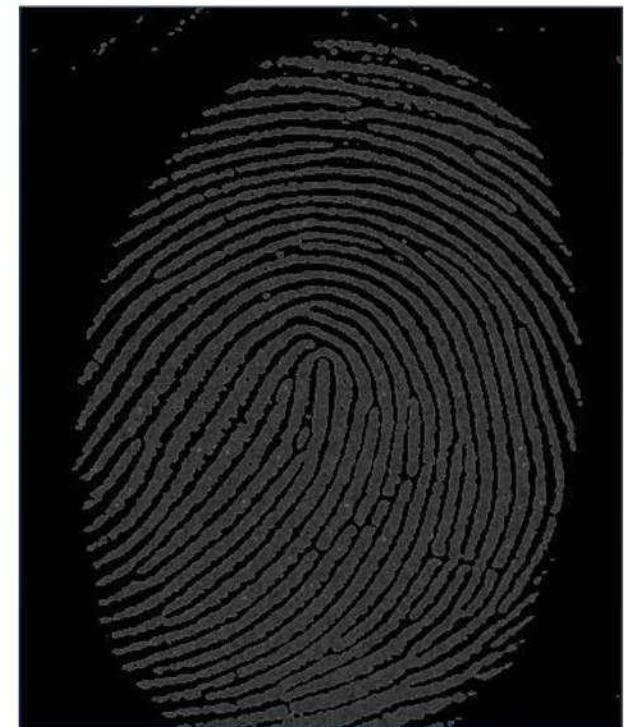
# Бинаризация по условию

$$f'(m, n) = \begin{cases} 0, & f(m, n) \leq t_1; \\ s, & t_1 < f(m, n) \leq t_2; \\ 0, & f(m, n) > t_2, \end{cases}$$



# Неполная пороговая обработка

$$f'(m, n) = \begin{cases} f(m, n), & f(m, n) > t; \\ 0, & f(m, n) \leq t. \end{cases} \quad f'(m, n) = \begin{cases} f(m, n), & f(m, n) < t; \\ 255, & f(m, n) \geq t. \end{cases}$$





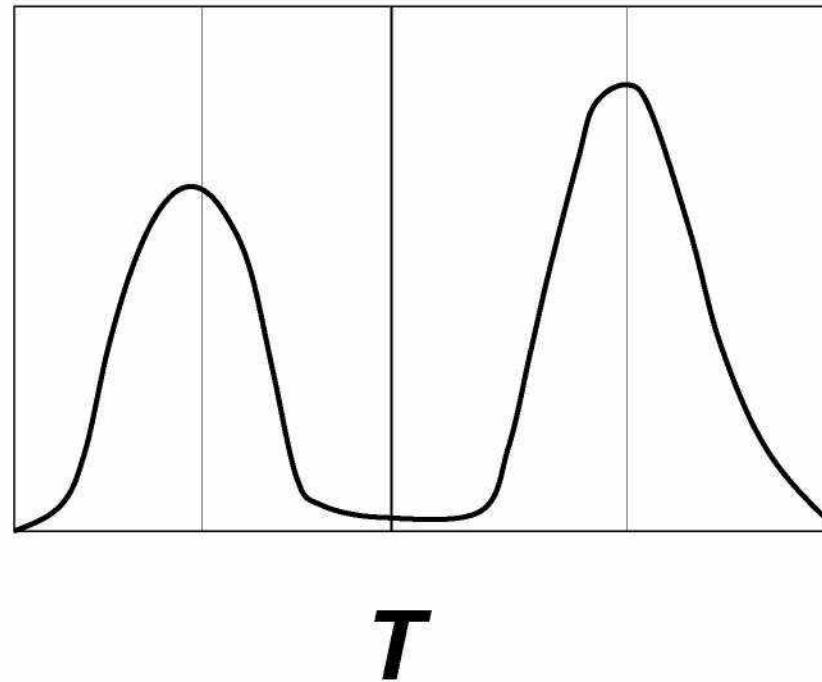
# Глобальная пороговая обработка. Выбор порога.

- Использование гистограммы изображения
- Определение порога на основе градиента яркости изображения
- Метод Отсю (определение оптимального порога)
- Метод использования энтропии гистограммы



# Глобальная пороговая обработка. Выбор порога с использованием гистограммы.

Использование гистограммы.





# Глобальная пороговая обработка. Выбор порога с использованием гистограммы.

## Использование гистограммы.

- 1 Выбирается некоторая начальная оценка значения порога  $t$ .
- 2 Выполняется сегментация изображения с помощью порога  $t$ . В результате образуются две группы пикселей:  $G_1$ , состоящая из пикселей с яркостью больше  $t$ , и  $G_2$ , состоящая из пикселей с яркостью меньше или равной  $t$ .
- 3 Вычисляются средние значения  $\mu_1$  и  $\mu_2$  яркостей пикселей по областям  $G_1$  и  $G_2$  соответственно.
- 4 Вычисляется новое значение порога

$$t = \frac{\mu_1 + \mu_2}{2}$$

- 5 Повторяются шаги со 2-го по 4-й до тех пор, пока разница значений порога  $t$  в соседних итерациях не окажется наперед меньше заданного параметра  $\varepsilon$ .



# Глобальная пороговая обработка. Выбор порога на основе градиента.

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad G_x = \frac{\partial f}{\partial x} = f(x+1, y) - f(x, y);$$
$$G_y = \frac{\partial f}{\partial y} = f(x, y+1) - f(x, y).$$

$$G(x, y) = \max \{|G_x|, |G_y|\}$$

$$t = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) G(x, y)}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} G(x, y)}$$



# Глобальная пороговая обработка. Метод Отсу

С помощью данного метода вычисляется порог  $T$ , который минимизирует внутриклассовую дисперсию (или максимизирует межклассовую дисперсию). Т.е., вычисленный порог минимизирует среднюю ошибку сегментации, т.е. среднюю ошибку от принятия решения о принадлежности пикселей изображения объекту или фону.



# Глобальная пороговая обработка. Метод Отсу

1. Построение нормализованной гистограммы изображения

$$h(i) = h_i, \quad i = 0, 1, \dots, 255, \quad \sum_{i=0}^{255} h_i = 1$$

2. Разбиение на два класса пикселей  $G_1$  и  $G_2$  с помощью некоторого значения порога  $T$ . Вычисление межклассовой дисперсии:

$$S(T) = P_1(T) \cdot P_2(T) \cdot (M_1(T) - M_2(T))^2$$

где  $P_1$  и  $P_2$  – вероятности двух классов,  
 $M_1$  и  $M_2$  – средние значения яркости

$$P_1(T) = \sum_{i=0}^T h(i), \quad P_2(T) = 1 - P_1(T)$$

$$M_1(T) = \sum_{i=0}^T i \cdot h(i) / P_1(T), \quad M_2(T) = \sum_{i=T+1}^{255} i \cdot h(i) / P_2(T)$$

3. Максимизация  $S(T)$ :  $T_{opt} = \arg \left\{ \max_{0 \leq T \leq 255} (S(T)) \right\}$



# Глобальная пороговая обработка. Метод Отсу

`t=graythresh(F) =150; im2bw(F,t)`

(обработка с помощью глобального порога, вычисленного по методу  
Отсу в системе Matlab)





# Глобальная пороговая обработка. Метод Отсу

Бинаризация изображения с помощью глобального порога не всегда даёт удовлетворительные результаты.

`t=graythresh(F) =105  
im2bw(F,t)`  
(обработка с помощью глобального порога, вычисленного по методу Отсу в системе Matlab)

ponents or broken connection paths. There is no pollution past the level of detail required to identify those.

Segmentation of nontrivial images is one of the most important steps in computer vision processing. Segmentation accuracy determines the effectiveness of computerized analysis procedures. For this reason, considerable attention must be taken to improve the probability of rugged segmentation. In such applications as industrial inspection, at least some degree of noise in the environment is possible at times. The experienced designer invariably pays considerable attention to such

problems of broken connection paths. There is no pollution past the level of detail required to identify those.

Segmentation of nontrivial images is one of the most important steps in computer vision processing. Segmentation accuracy determines the effectiveness of computerized analysis procedures. For this reason, considerable attention must be taken to improve the probability of rugged segmentation. In such applications as industrial inspection, at least some degree of noise in the environment is possible at times. The experienced designer invariably pays considerable attention to such



# Локальная пороговая обработка

Локальная пороговая обработка характеризуется тем, что изображение разбивается на подобласти, в каждой из которых для сегментации используется свое значение порога.

Основные проблемы:

- как разбить исходное изображение;
- как оценить порог для каждой полученной области.

Методы:

- Метод Бернсена.
- Метод Чоу и Канеко.
- Метод Эйквила.
- Метод Ниблацка.



# Локальная пороговая обработка. Метод Бернсена.

Все изображение делится на квадраты  $r \times r$  ( $r$  – нечетное) с центром в точке  $(m, n)$ . Для каждого пикселя изображения в пределах квадрата используется порог, вычисляемый по формуле:

$$t(m, n) = \frac{j_{low} + j_{high}}{2}$$

где  $j_{low}$  и  $j_{high}$  являются соответственно наименьшим и наибольшим уровнем яркости в квадрате.

Если в принятой области выполняется условие

$$C(m, n) = (j_{high} - j_{low}) \leq \varepsilon$$

где  $\varepsilon$  – заданная пороговая величина, то тогда исследуемый квадрат содержит объекты только одного класса: объекты или фон.

Исследования показывают, что наилучшие результаты достигаются для значений  $\varepsilon = 15$  и  $r = 15$ .



# Локальная пороговая обработка. Метод Ниблацка.

Для каждого пикселя изображения используется свое значение порога. Определение величины порога происходит на основе вычисления локального среднего и локального среднеквадратического отклонения. Значение порога в точке с координатами  $(m, n)$  вычисляется в соответствии с формулой:

$$t(m, n) = \mu(m, n) + k \sigma(m, n)$$

где  $\mu(m, n)$  – среднее, а  $\sigma(m, n)$  – среднеквадратическое отклонение в локальной окрестности точки изображения  $(m, n)$ .

Размер окрестности должен выбираться таким образом, чтобы, с одной стороны, при обработке сохранялись локальные детали, а с другой – чтобы можно было устраниить шумы. Экспериментальные исследования показывают, что для окна  $r = 15 \times 15$  и коэффициента  $k = -0.2$  удается получить на изображении хорошо разделенные объекты и фон.



# Адаптивная пороговая обработка

Для зашумленных изображений, имеющих низкое качество, например, изображений, имеющих неоднородные или изменяющиеся яркости фона, или же изображений, характеризующихся сложной текстурой фона, большой проблемой является то обстоятельство, что ни один из алгоритмов пороговой обработки не может дать удовлетворительных результатов в случае присутствия на изображении различных видов помех.



# Адаптивная пороговая обработка

$$f'(m, n) = \begin{cases} 1, & |\hat{P}_l - f(m, n)| > t, \quad l = 0, \dots, 7; \\ 0, & \text{иначе} \end{cases}$$

$$\begin{bmatrix} P_1(m_1, n_1) & P_2(m_2, n_2) & P_3(m_3, n_3) \\ P_0(m_0, n_0) & f_{mn} & P_4(m_4, n_4) \\ P_7(m_7, n_7) & P_6(m_6, n_6) & P_5(m_5, n_5) \end{bmatrix}$$

$$\hat{P}_l = \frac{1}{(2K+1)^2} \sum_{-K \leq i, j \leq K} f(m_l + i, n_l + j)$$

средняя локальная яркость

$2K+1$  – размер окна



# Адаптивная пороговая обработка

Вычисление порога  $t$ :

1. Вычисляем

$$f_{\max} = \max_{-K \leq i, j \leq K} f(m+i, n+j) \quad f_{\min} = \min_{-K \leq i, j \leq K} f(m+i, n+j)$$

$$\hat{P} = \frac{1}{(2K+1)^2} \sum_{-K \leq i, j \leq K} f(m+i, n+j)$$

$$\Delta f_{\max} = | f_{\max} - \hat{P} | \quad \Delta f_{\min} = | f_{\min} - \hat{P} |$$



# Адаптивная пороговая обработка

Вычисление порога  $t$ :

2. Если  $\Delta f_{\max} > \Delta f_{\min}$

то 
$$t = \alpha \left( \frac{2}{3} f_{\min} + \frac{1}{3} \hat{P} \right)$$

3. Если  $\Delta f_{\max} < \Delta f_{\min}$

то 
$$t = \alpha \left( \frac{1}{3} f_{\min} + \frac{2}{3} \hat{P} \right)$$



# Адаптивная пороговая обработка

Вычисление порога  $t$ :

4. Если  $\Delta f_{\max} = \Delta f_{\min}$  и  $f_{\max} \neq f_{\min}$

То увеличиваем размер окна на 1 ( $K=K+1$ ) и повторяем шаги 1-3.

Если при этом  $f_{\max} = f_{\min}$ , то  $t = \alpha \hat{P}$

$\alpha$  - константа из диапазона  $[0.3, \dots, 0.8]$ .

Для изображений очень плохого качества с большим шумом и низким контрастом следует использовать значение  $\alpha = 1/3$ . В большинстве же случаев достаточным является значение  $\alpha = 2/3$



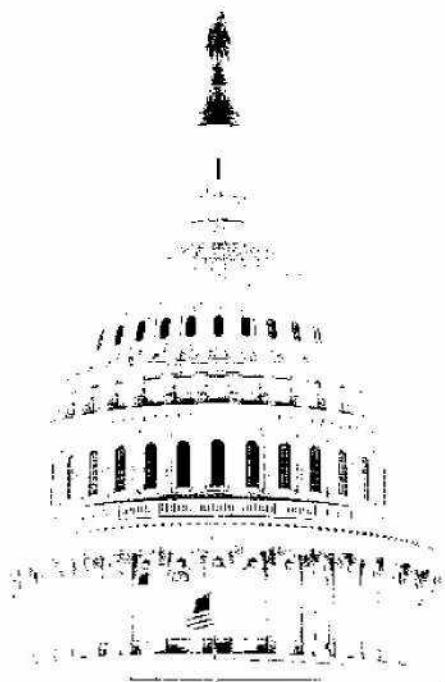
# Адаптивная пороговая обработка

Пример: бинаризация с нижним порогом.

исходное



$t=128$



$t=190$





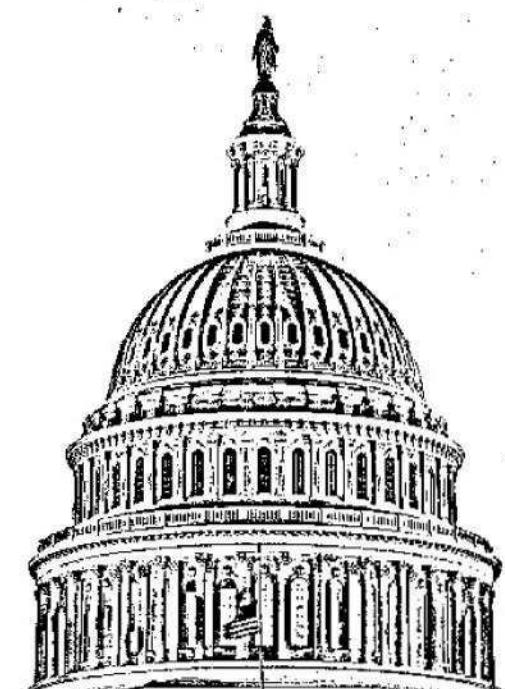
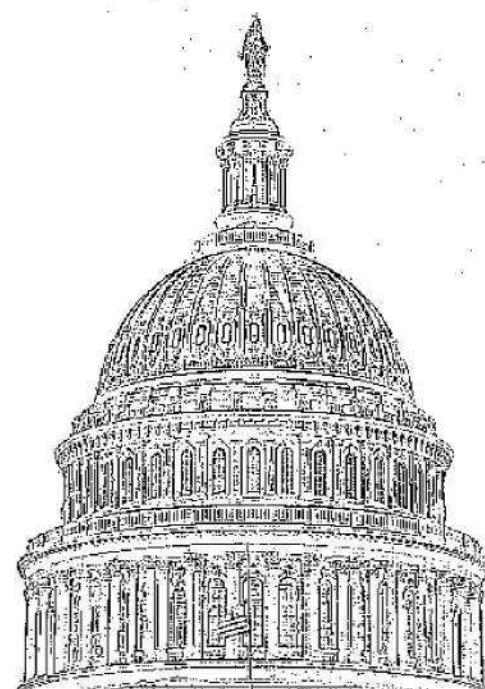
# Адаптивная пороговая обработка

Пример: адаптивная пороговая обработка.

Исходное



CV\_ADAPTIVE\_THRESH\_MEAN\_C  
block\_size=3                            block\_size=11





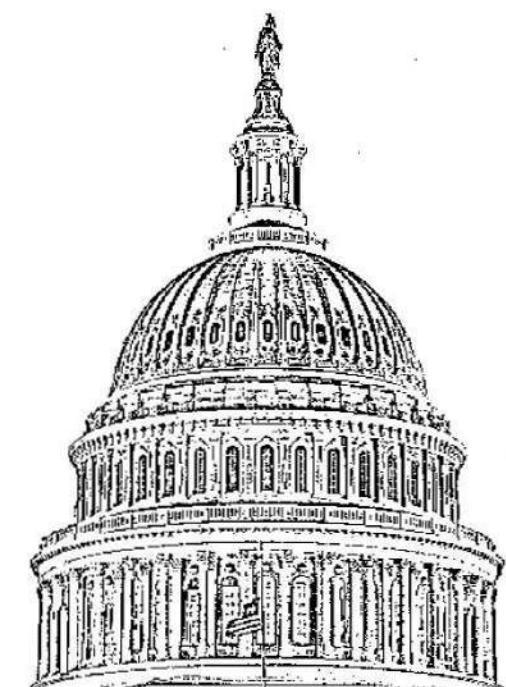
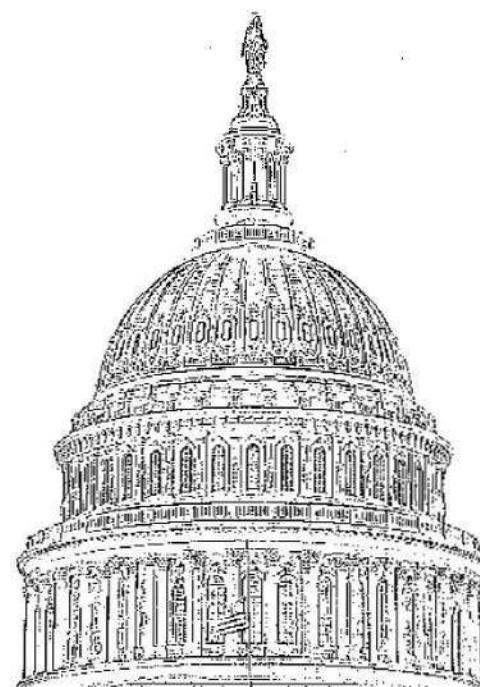
# Адаптивная пороговая обработка

Пример: адаптивная пороговая обработка.

Исходное

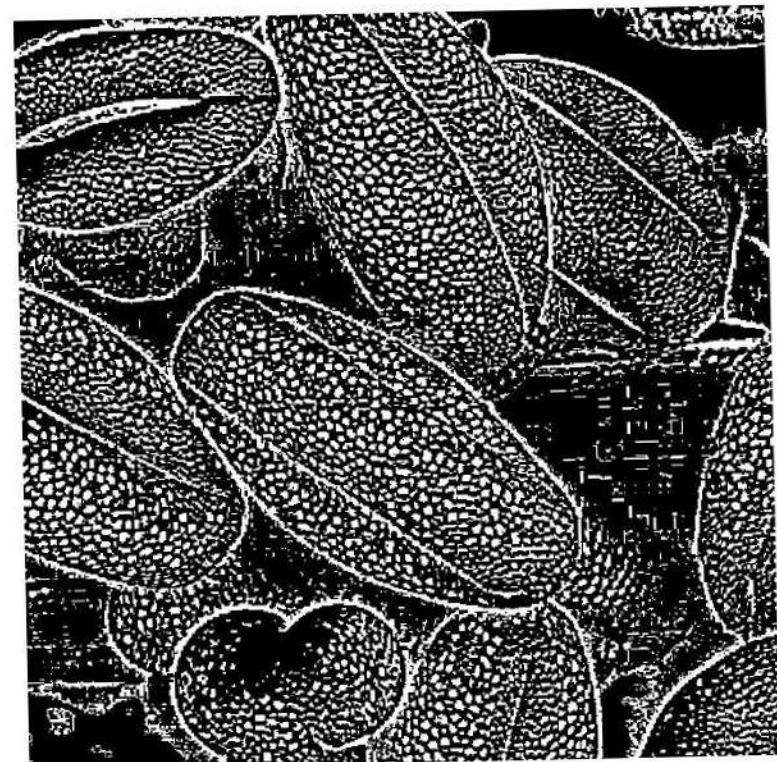
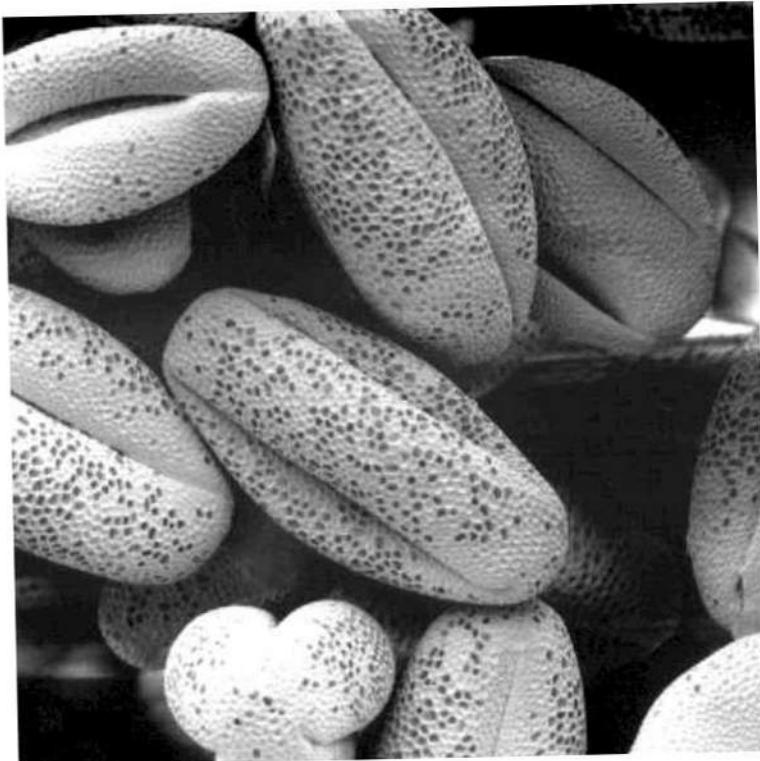


CV\_ADAPTIVE\_THRESH\_GAUSSIAN\_C  
block\_size=3                                   block\_size=11





# Адаптивная пороговая обработка



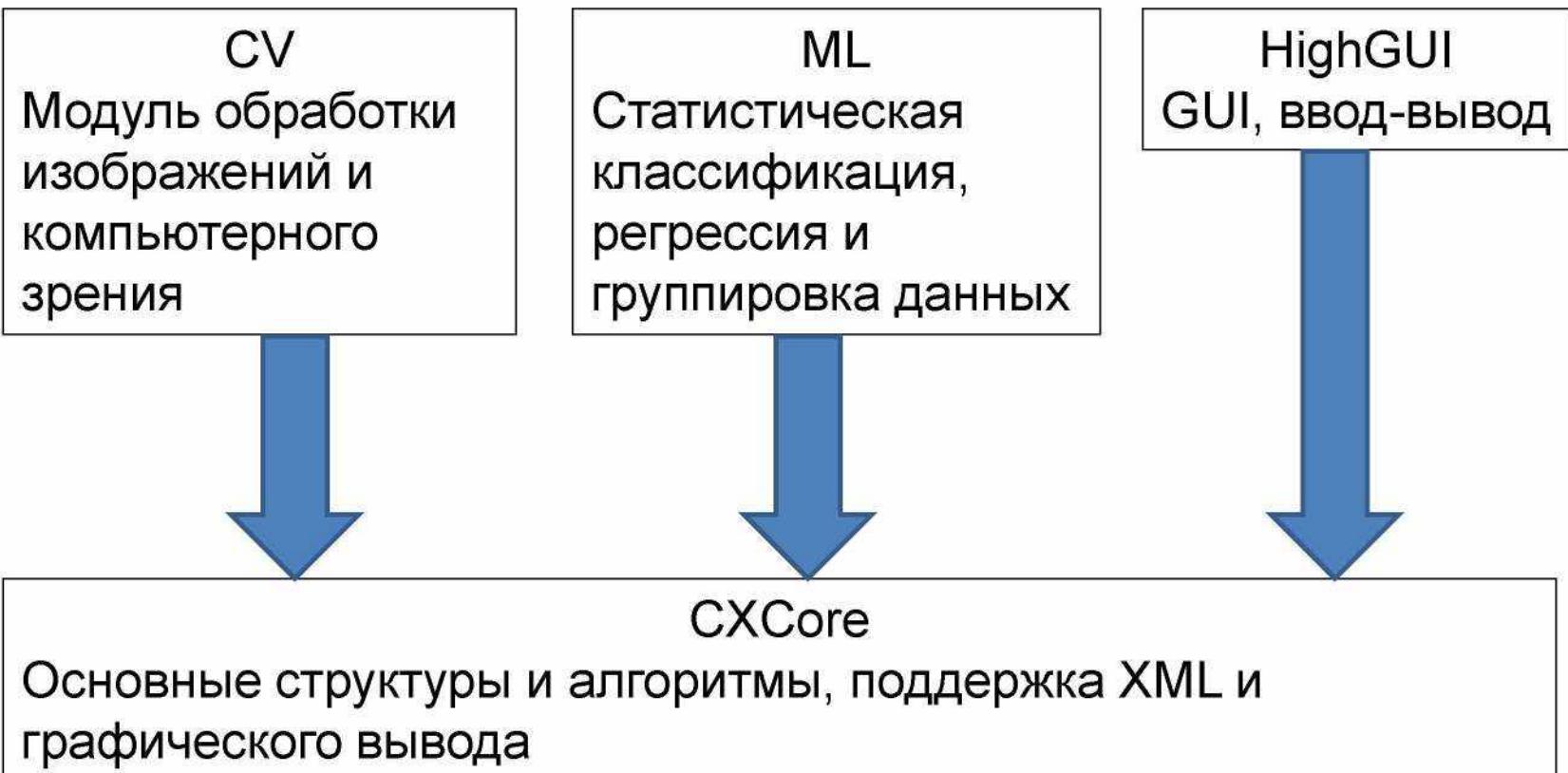


# OpenCV

**OpenCV** (англ. *Open Source Computer Vision Library*, библиотека компьютерного зрения с открытым исходным кодом) — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других языков. Может свободно использоваться в академических и коммерческих целях.



# OpenCV





# OpenCV

- ***opencv\_core*** — основная функциональность.  
Включает в себя базовые структуры, вычисления(математические функции, генераторы случайных чисел) и линейную алгебру, и т. д.
- ***opencv\_imgproc*** — обработка изображений (фильтрация, геометрические преобразования, преобразование цветовых пространств и т. д.).
- ***opencv\_highgui*** — простой UI, ввод/вывод изображений и видео.
- ***opencv\_ml*** — модели машинного обучения (SVM, деревья решений, обучение со стимулированием и т. д.).



# OpenCV

- ***opencv\_features2d*** — распознавание и описание плоских примитивов.
- ***opencv\_video*** — анализ движения и отслеживание объектов (оптический поток, шаблоны движения, устранение фона).
- ***opencv\_objdetect*** — обнаружение объектов на изображении (нахождение лиц с помощью алгоритма Виолы-Джонса, распознавание людей HOG и т. д.).
- ***opencv\_calib3d*** — калибровка камеры, поиск стерео-соответствия и элементы обработки трехмерных данных.



# OpenCV

Где используется:

- Google Maps, Google street view, Google Earth.
- Научные и промышленные исследования.
- Медицинская визуализация.
- Системы обеспечения безопасности.
- Поиск изображений.
- Видео-поиск.
- Мониторинг конвейерной продукции в производственных системах.
- Робототехника.



Белорусский Государственный Университет  
Факультет прикладной математики и информатики

# Компьютерная графика

Василевский Константин

Викторович

17 лекций,

17 лабораторных занятий



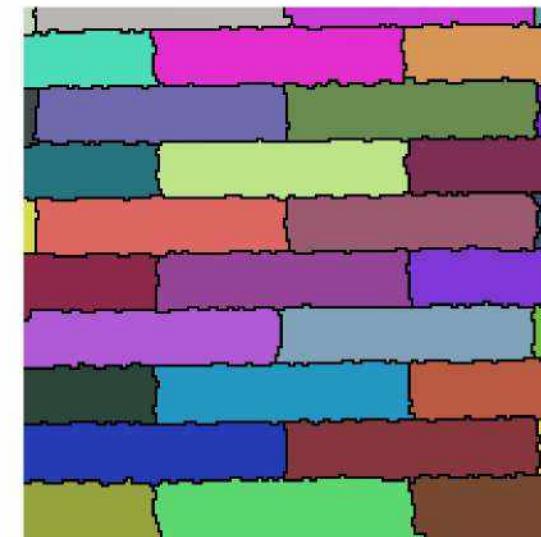
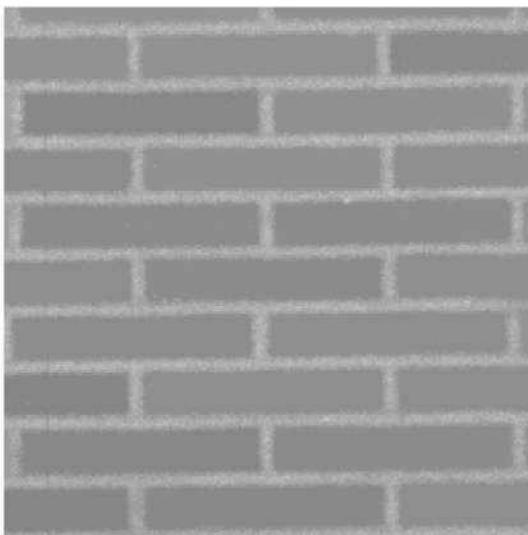
# Основы обработки цифровых изображений.

- Гистограмма изображения
- Сегментация изображений
- Пороговая обработка
- Бинаризация
- Морфологическая обработка
- Библиотека OpenCV



# Сегментация изображений

**Сегментация** изображений – процесс разделения изображения на однородные в определенном смысле области.





# Сегментация изображений

- разбиение изображения на области, однородные по некоторому критерию
- разбиение изображения на области на основании резких изменений сигнала, таких как перепады яркости



# Сегментация изображений

- пороговая обработка (пороговая сегментация) – самый простой метод сегментации, ориентированный на обработку изображений, отдельные однородные участки которых различаются средней яркостью
- сегментация на основе поиска разрывов яркости
- сегментация по морфологическим водоразделам
- текстурная сегментация



# Сегментация изображений (обнаружение разрывов яркости)

Три основных типа разрывов яркости:

- *точки*
- *линии*
- *перепады яркости (края)*



# Сегментация изображений

**Напоминание:**

Дискретная свертка  $w$ :

$$f'(m, n) = (w \times f)(m, n) = \frac{1}{W} \sum_{i, j \in K} f(m + i, n + j) w(i, j)$$

$K$  - окрестность рассматриваемой точки изображения  $(m, n)$ ,

$w(i, j)$  – веса пикселей в окрестности  $K$

$W$  – сумма весов



# Сегментация изображений

$$\begin{bmatrix} w(-1, -1) & w(0, -1) & w(-1, 1) \\ w(0, -1) & w(0, 0) & w(0, 1) \\ w(1, -1) & w(1, 0) & w(1, 1) \end{bmatrix} \quad \text{- маска } 3 \times 3 \text{ (фильтр, шаблон)}$$

$$f'(m, n) = \frac{1}{W} \sum_{i=-1}^1 \sum_{j=-1}^1 f(m+i, n+j) w(i, j)$$

$$W = \sum_{i=-1}^1 \sum_{j=-1}^1 w(i, j)$$



# Сегментация изображений. Обнаружение точек.

$$w = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$R(m, n) = \sum_{i=-1}^1 \sum_{j=-1}^1 f(m+i, n+j) w(i, j)$  - отклик фильтра

$T$  – неотрицательный порог

$|R| > T \Rightarrow$  в пикселе, соответствующем центру маски, обнаружена точка

$R=0$  на областях постоянной яркости



# Сегментация изображений. Обнаружение линий.

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

горизонтальная      вертикальная       $+45^\circ$        $-45^\circ$

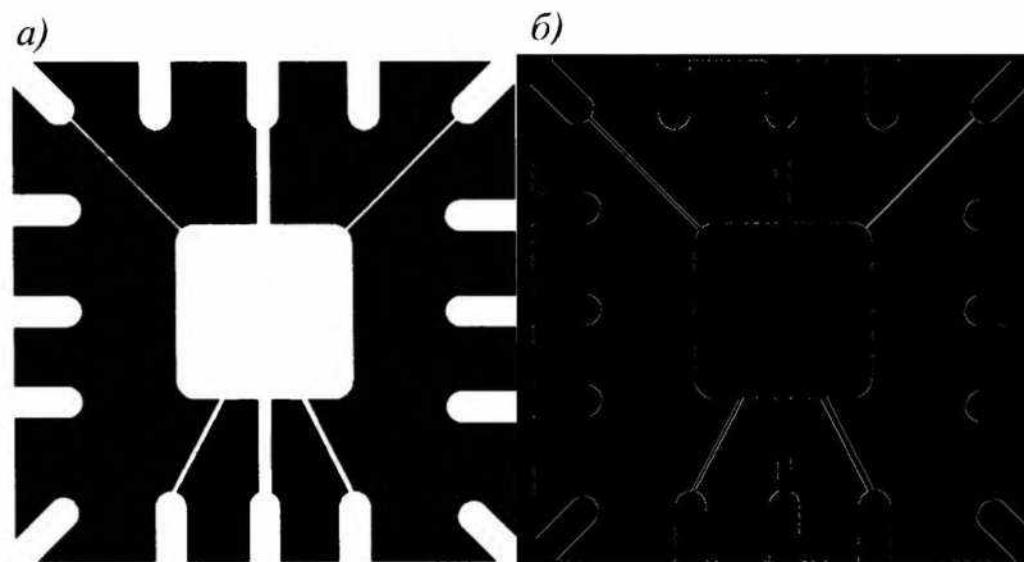
$R_1, R_2, R_3$ , и  $R_4$  - отклики каждой из масок.

Если  $|R_i| > |R_j|$  при  $j \neq i$ , то точка, соответствующая центру маски, считается принадлежащей линии, ориентированной вдоль направления маски  $i$ .



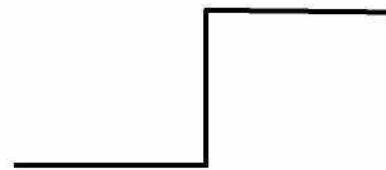
# Сегментация изображений. Обнаружение линий.

- а) исходное бинарное изображение шаблона для изготовления выводов кристалла интегральной микросхемы
- б) модуль откликов на диагональную маску

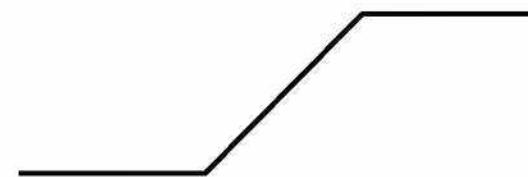




# Сегментация изображений. Обнаружение границ.



модель идеального перепада



модель наклонного перепада

Перепад яркости – это связное множество пикселей, лежащих на границе между двумя областями.

Первая производная – для обнаружения перепада.

Вторая производная – для обнаружения типа перепада (от темного к светлому либо от светлого к темному).



# Сегментация изображений. Обнаружение границ.

Первая производная.

$$G_x = \frac{\partial f}{\partial x} = f(x+1, y) - f(x, y);$$

$$G_y = \frac{\partial f}{\partial y} = f(x, y+1) - f(x, y).$$

$$G_x : \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad G_y : \begin{bmatrix} 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$G_x : \begin{bmatrix} -1 & 1 \\ 0 & 0 \end{bmatrix} \quad G_y : \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix}$$



# Сегментация изображений. Обнаружение границ.

Другое определение первая производной:  
*перекрестные градиентные операторы Робертса*

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

$$G = f(x+1, y+1) - f(x, y);$$

$$G = f(x-1, y-1) - f(x, y).$$



# Сегментация изображений. Обнаружение границ.

Маски оператора Превитта:

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix}$$

$$G_x = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$



# Сегментация изображений. Обнаружение границ.

Маски оператора Собеля:

$$\begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$$

$$G_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$



# Сегментация изображений. Обнаружение границ.

Рассмотренные маски применяются для получения составляющих градиента  $G_x$  и  $G_y$ .

Для вычисления величины градиента используется формула:

$$\nabla f = |\nabla \mathbf{f}| = \sqrt{G_x^2 + G_y^2}$$

Или приближенное значение:

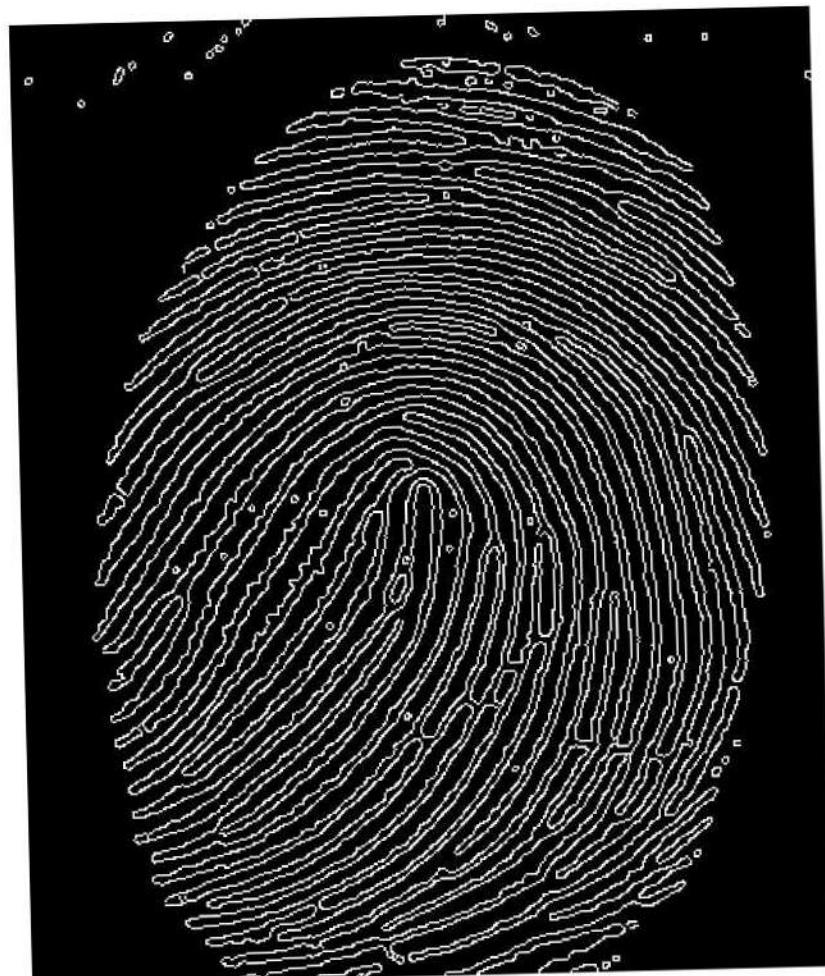
$$\nabla f = |G_x| + |G_y|$$

Результирующее изображение – вычисленный градиент.  
(светлые границы на темном фоне)



# Сегментация изображений.

## Обнаружение границ. Оператор Собеля





## Сегментация изображений.

Для сравнения: глобальная пороговая обработка





# Сегментация изображений. Обнаружение границ.

Другие операторы:

- Робинсона
- Кирша
- Моравеца

Другие подходы:

Детектор границ Канни -  
многоступенчатый алгоритм для обнаружения  
широкого спектра границ в изображениях.



# Морфологическая обработка

*Математическая морфология –  
подход к обработке изображений,  
суть которого заключается в том,  
что исходное изображение  
рассматривается как множество, и  
к нему применяются теоретико-  
множественные операции.*



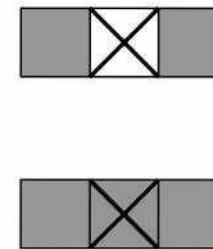
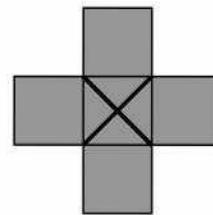
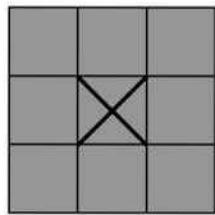
# Морфологическая обработка

Наи важнейшим понятием морфологических преобразований есть понятие *структурообразующего элемента* (*структурообразующего множества* или *примитива*), который представляет собой определенную совокупность элементов изображения с одним выделенным (*центральным*) элементом.



# Морфологическая обработка

## Типовые структурирующие элементы





# Морфологическая обработка

Морфологические преобразования:

- Эрозия
- Дилатация
- Размыкание
- Замыкание

$f(m,n)$  – бинарное изображение

$$A, B \subset \mathbb{Z}^2$$

$A$  – некоторая область

$B$  – структурирующий элемент



# Морфологическая обработка.

## Операция эрозии

Операция эрозии  $A \ominus B$ .

Результат операции эрозии представляет собой множество, состоящее из центральных пикселей всех структурирующих элементов, которые целиком помещаются внутри области  $A$ . Меру степени эрозии определяет размер структурирующего элемента. Чем он больше, тем большая часть края подлежащей эрозии области будет удалена.



# Морфологическая обработка. Операция эрозии.

Операция эрозии  $A \ominus B$ .

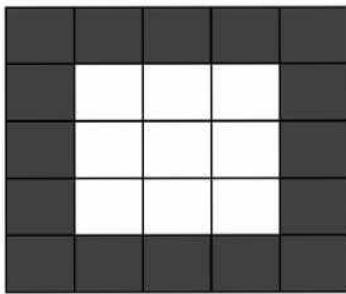
Эрозию с математической точки зрения можно рассматривать как *фильтр минимума*.

С точки зрения компьютерной обработки эрозия базируется на удалении всех тех элементов изображения со значением 1 (белых точек), которые имеют хоть одного соседа со значением 0 (черная точка)

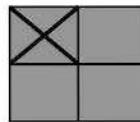


# Морфологическая обработка. Операция эрозии

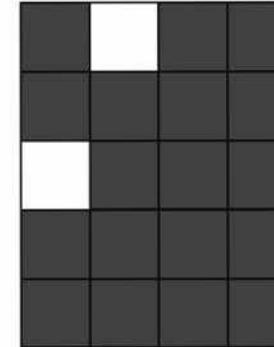
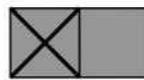
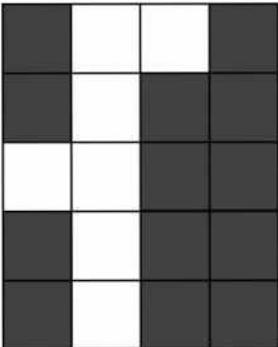
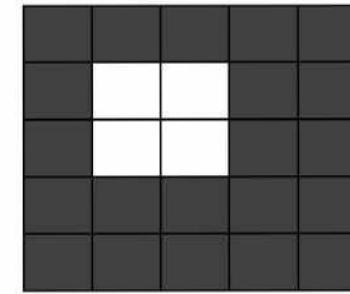
*A*



*B*



$A \ominus B$





# Морфологическая обработка. Операци эрозии

Эрозия приводит к устраниению на бинарном изображении несущественных (по их размерам) деталей и сглаживанию краев объектов.





# Морфологическая обработка. Операция дилатации

Операция дилатации  $A \oplus B$ .

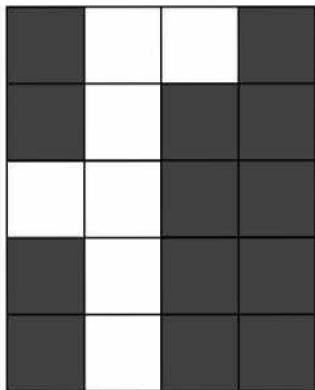
Результат операции дилатации представляет собой множество, состоящее из центральных пикселей всех структурирующих элементов, у которых хотя бы одна точка лежит внутри области  $A$ . Меру степени дилатации определяет размер структурирующего элемента.

С математической точки зрения дилатацию можно рассматривать как *фильтр максимума*.



# Морфологическая обработка. Операция дилатации

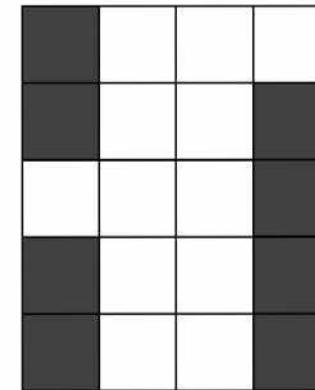
$A$



$B$



$A \oplus B$





# Морфологическая обработка. Операция дилатации

Применение дилатации для:

- устранения небольших разрывов линий путем их перекрытия;
- объединения фрагментов изображения, лежащих рядом.





# Морфологическая обработка.

## Операции замыкания и размыкания

- размыкание сглаживает контуры объекта, обрывает узкие перешейки и ликвидирует выступы небольшой ширины;
- замыкание также проявляет тенденцию к сглаживанию участков контуров, но в отличии от размыкания, в общем случае «заливает» узкие разрывы и длинные углубления малой ширины, а также ликвидирует небольшие отверстия и заполняет промежутки контура.

размыкание = эрозия + дилатация,  
замыкание = дилатация + эрозия.



# Морфологическая обработка. Операция замыкания





# Морфологическая обработка. Операция размыкания





# Морфологическая обработка. Выделение границ.

Морфологические операции можно также использовать для выделения границ бинарного объекта.

Легко заметить, что граничные точки имеют как минимум один фоновый пиксел в своей окрестности. Таким образом, применив оператор эрозии со структурным элементом, содержащим все возможные соседние элементы, мы удалим все граничные точки.

Тогда граница получится с помощью операции разности множеств между исходным изображением и изображением, полученным в результате эрозии.



# Морфологическая обработка. Полутоновые изображения

## Дилатация

$$f \oplus b(m, n) = \max\{f(m - x, n - y) + b(x, y) \mid (m - x, n - y) \in D_f, (x, y) \in D_b\}$$

## Эрозия

$$f \ominus b(m, n) = \min\{f(m - x, n - y) + b(x, y) \mid (m + x, n + y) \in D_f, (x, y) \in D_b\}$$

$D_b$  – структурирующий элемент



# Частотные методы улучшения изображения.

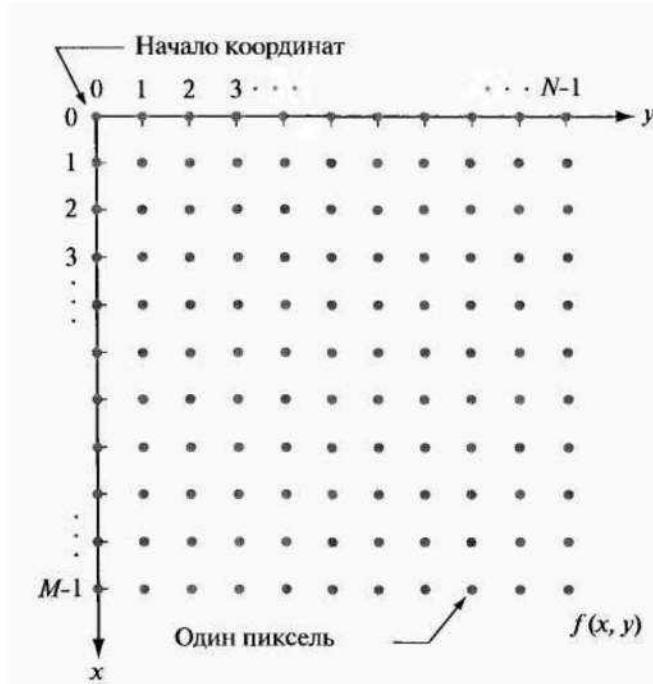
- Преобразование Фурье и частотная область.
- Фильтрация в частотной области.
- Соответствие между фильтрацией в пространственной области и фильтрацией в частотной области.



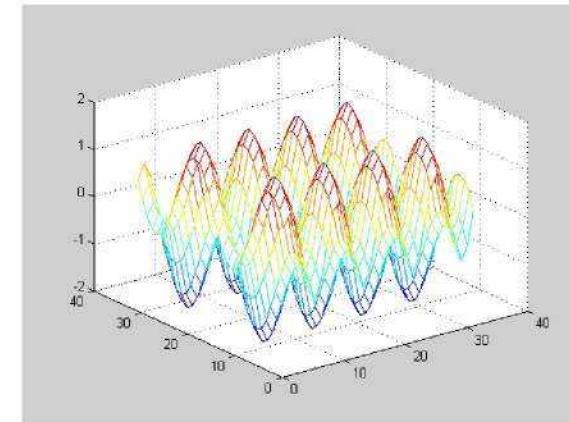
# Пространственная область и частотная область

Методы обработки в пространственной области основаны на прямом манипулировании пикселями изображения .

$$f'(m, n) = T(f(m, n))$$



Методы обработки в частотной области основаны на модификации сигнала, формируемого путем применения к изображению преобразования Фурье





# Преобразование Фурье

1. Любая функция, периодически воспроизводящая свои значения, может быть представлена в виде суммы синусов и/или косинусов определенных частот, умноженных на некоторые коэффициенты (ряд Фурье).
2. Когда функция не является периодической (но площадь под ее графиком конечна), она может быть представлена в виде интеграла от синусов и/или косинусов, умноженных на некоторую весовую функцию (преобразование Фурье).
3. Дискретное преобразование Фурье переводит конечную последовательность вещественных чисел в конечную последовательность коэффициентов Фурье.

Любая функция, заданная как рядом, так и преобразованием Фурье, может быть полностью восстановлена при помощи некоторой процедуры обращения.

Преобразование Фурье позволяет выявить некие периодические компоненты в исходной функции и оценить их вклад в ее форму.



## Дискретное двумерное преобразование Фурье

Дискретное двумерное преобразование Фурье для изображения  $f(x,y)$ ,  $x=0,\dots,M-1$ ,  $y=0,\dots,N-1$

$$F(u,v) = \frac{1}{M \cdot N} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \cdot e^{-i2\pi(ux/M+vy/N)},$$
$$u = 0,1,\dots,M-1, \quad v = 0,1,\dots,N-1$$

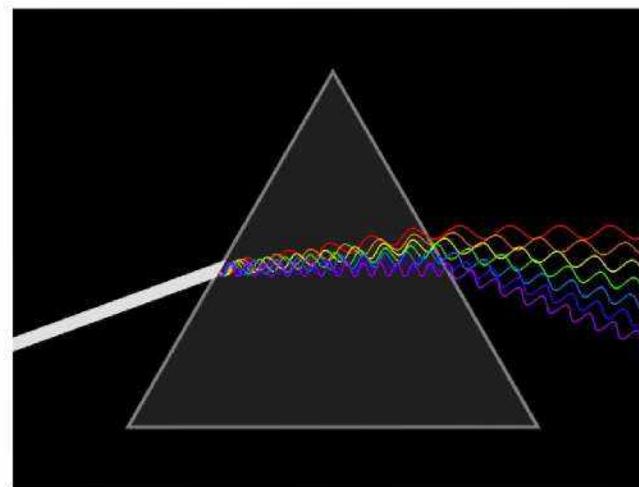
Обратное преобразование Фурье

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) \cdot e^{i2\pi(ux/M+vy/N)},$$
$$x = 0,1,\dots,M-1, \quad y = 0,1,\dots,N-1$$



## Дискретное двумерное преобразование Фурье

Дискретное двумерное преобразование Фурье можно рассматривать как своего рода «математическую» призму, которая раскладывает функцию на различные составляющие в зависимости от ее «частотного содержания»





# Частотные составляющие изображения

Частоты в Фурье-преобразовании связаны с вариацией яркости на изображении. Наиболее медленно меняющаяся (постоянная) частотная составляющая ( $u=v=0$ , начало координат) совпадает со средней яркостью изображения.

$$F(0,0) = \frac{1}{M \cdot N} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)$$



# Частотные составляющие изображения

**Низкие частоты** (отвечают точкам вблизи начала координат Фурье-преобразования):

- соответствуют медленно меняющимся компонентам изображения и плавным изменениям яркости;
- определяют основное содержание изображения – фон и крупноразмерные объекты

**Высокие частоты** (отвечают точкам, удаленным от начала координат Фурье-преобразования):

- соответствуют все более и более быстрым изменениям яркости, границам объектов и другим деталям изображения, характеризуемым резкими изменениями яркости, таким как шум.
- определяют мелкоразмерные объекты, мелкие детали и шумовую компоненту.



# Частотные методы улучшения изображения.

Спектр Фурье задается формулой:

$$|F(u, v)| = \sqrt{R^2(u, v) + I^2(u, v)}$$

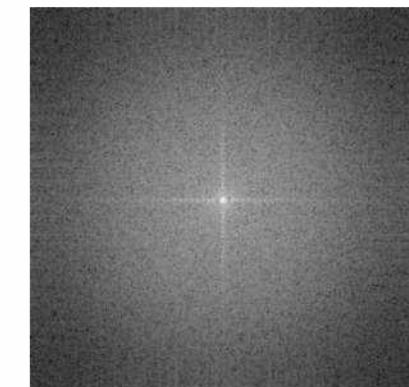
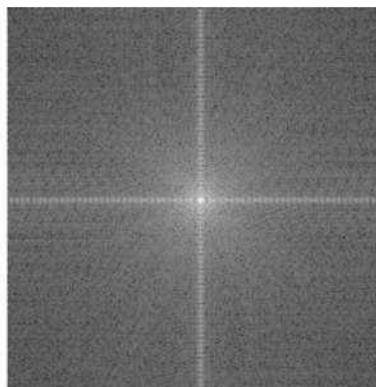
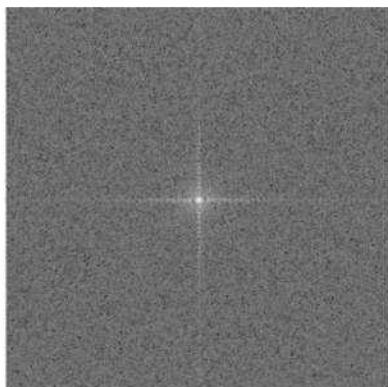
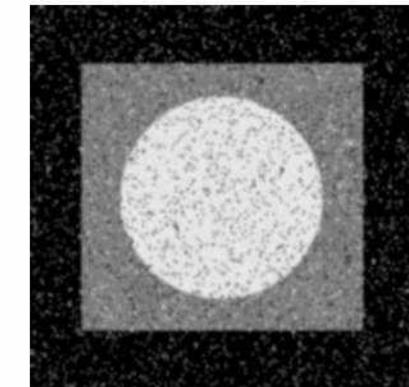
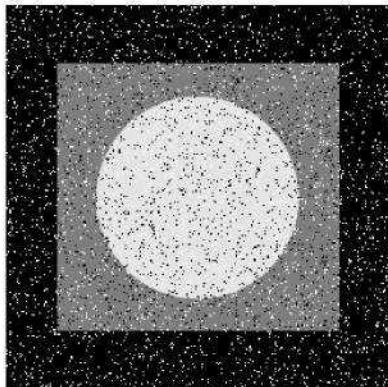
Где  $R(u, v)$  и  $I(u, v)$  – вещественная и мнимая компоненты  $F(u, v)$

Для улучшения зрительного восприятия при визуализации спектр центрируется и подвергается логарифмическому преобразованию.



# Частотные методы улучшения изображения.

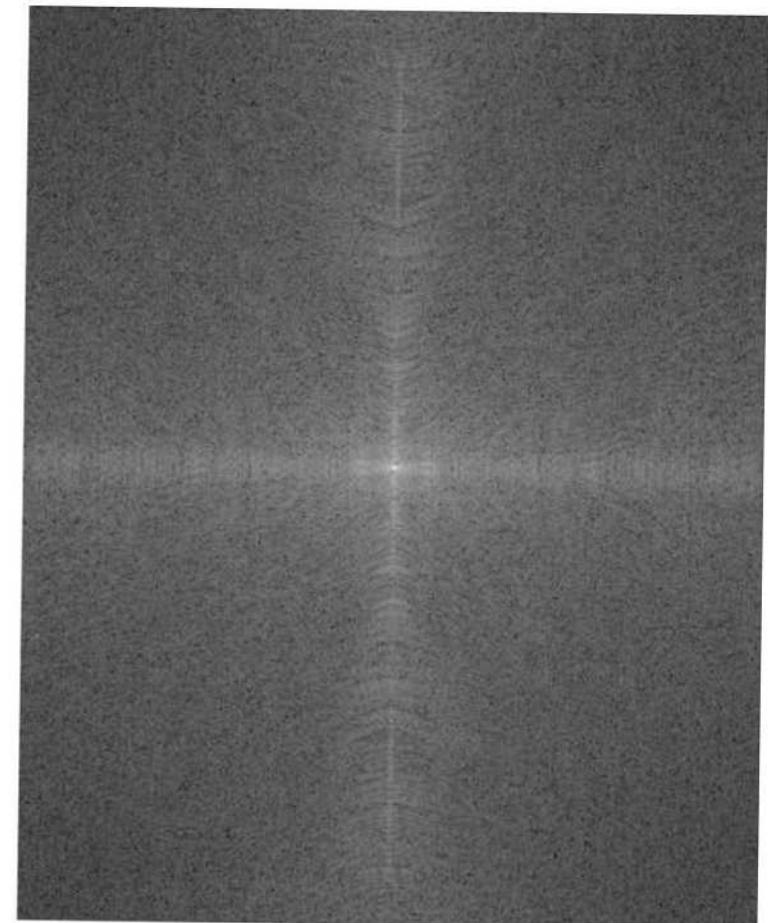
Примеры спектров изображений





# Частотные методы улучшения изображения.

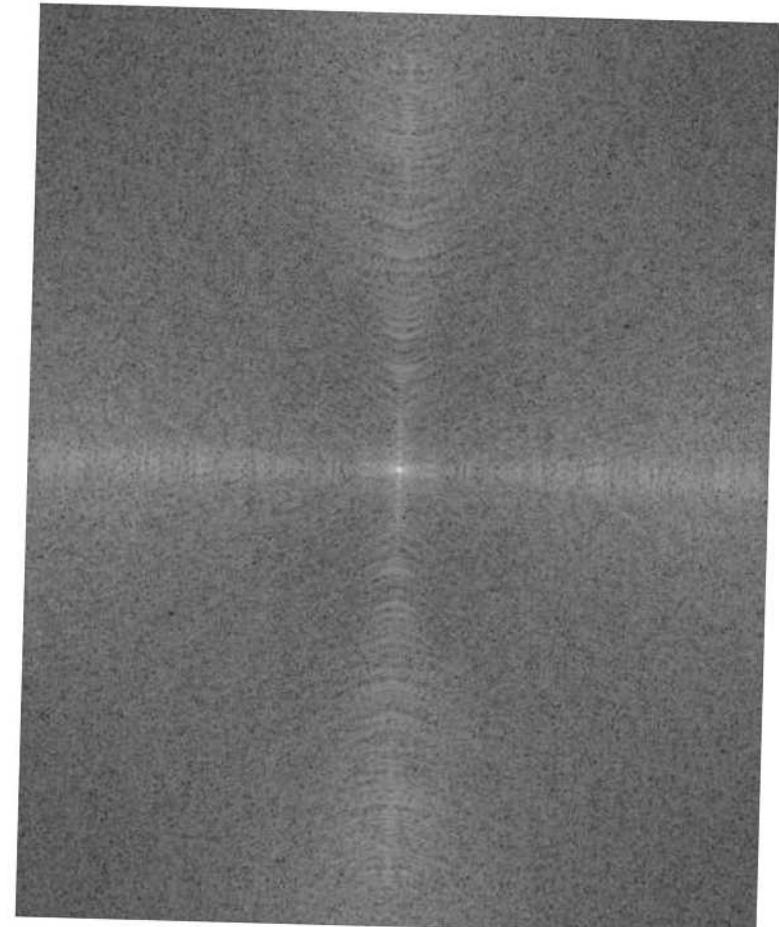
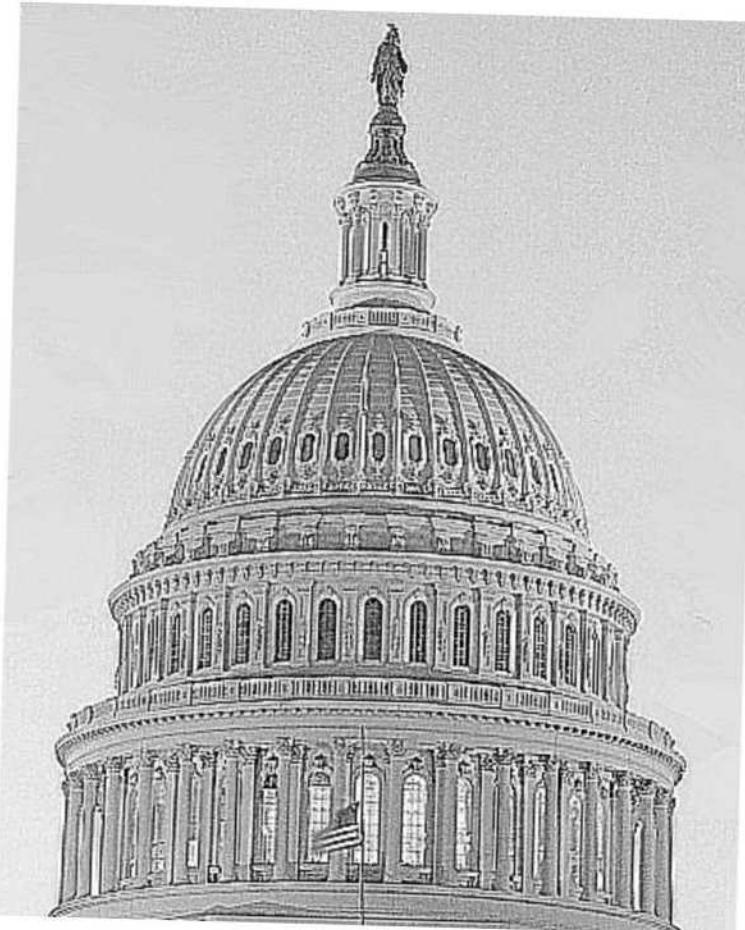
Примеры спектров изображений





# Частотные методы улучшения изображения.

Примеры спектров изображений



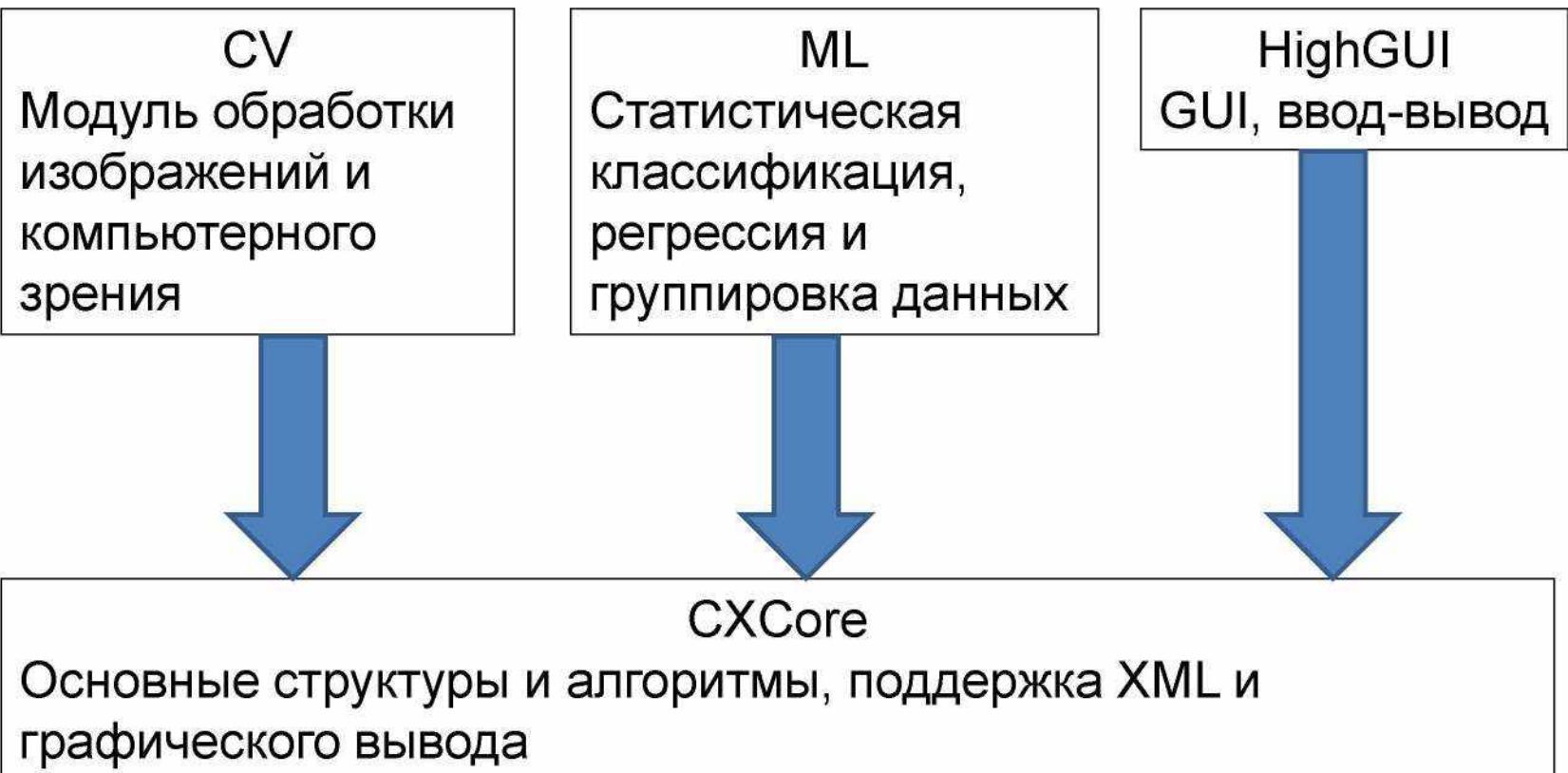


# OpenCV

**OpenCV** (англ. *Open Source Computer Vision Library*, библиотека компьютерного зрения с открытым исходным кодом) — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других языков. Может свободно использоваться в академических и коммерческих целях.



# OpenCV





# OpenCV

- ***opencv\_core*** — основная функциональность.  
Включает в себя базовые структуры, вычисления(математические функции, генераторы случайных чисел) и линейную алгебру, и т. д.
- ***opencv\_imgproc*** — обработка изображений (фильтрация, геометрические преобразования, преобразование цветовых пространств и т. д.).
- ***opencv\_highgui*** — простой UI, ввод/вывод изображений и видео.
- ***opencv\_ml*** — модели машинного обучения (SVM, деревья решений, обучение со стимулированием и т. д.).



# OpenCV

- ***opencv\_features2d*** — распознавание и описание плоских примитивов.
- ***opencv\_video*** — анализ движения и отслеживание объектов (оптический поток, шаблоны движения, устранение фона).
- ***opencv\_objdetect*** — обнаружение объектов на изображении (нахождение лиц с помощью алгоритма Виолы-Джонса, распознавание людей HOG и т. д.).
- ***opencv\_calib3d*** — калибровка камеры, поиск стерео-соответствия и элементы обработки трехмерных данных.



# OpenCV

Где используется:

- Google Maps, Google street view, Google Earth.
- Научные и промышленные исследования.
- Медицинская визуализация.
- Системы обеспечения безопасности.
- Поиск изображений.
- Видео-поиск.
- Мониторинг конвейерной продукции в производственных системах.
- Робототехника.



Белорусский Государственный Университет  
Факультет прикладной математики и информатики

# Компьютерная графика

Василевский Константин

Викторович

17 лекций,

17 лабораторных занятий



- Сложные трехмерные преобразования
- Проекции
- Освещение в компьютерной графике
- Введение в OpenGL



# Трехмерные преобразования координат

- преобразование масштабирования;
- преобразование переноса;
- преобразования поворотов (вокруг координатных осей, вокруг произвольных осей).



# Трехмерные преобразования координат. Однородные координаты

Однородные координаты - это математический механизм, связанный с определением положения точек в пространстве. Точка в трехмерном пространстве представляется четырехмерным вектором.

$$\begin{bmatrix} x & y & z & s \end{bmatrix} \rightarrow \begin{bmatrix} x/s & y/s & z/s & 1 \end{bmatrix}$$

Введение однородных координат позволяет решать более сложные задачи:

- описать бесконечно удаленную точку;
- провести различие между точкой и вектором;
- ввести преобразование переноса.



# Трехмерные преобразования координат. Масштабирование

$$X \cdot T = [x \ y \ z \ 1] \cdot \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & e & 0 & 0 \\ 0 & 0 & j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [ax \ ey \ jz \ 1]$$



# Трехмерные преобразования координат. Общее масштабирование

$$X \cdot T = [x \ y \ z \ 1] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & s \end{bmatrix} = [x \ y \ z \ s]$$

$$[x \ y \ z \ s] \rightarrow [x/s \ y/s \ z/s \ 1]$$



# Трехмерные преобразования координат. Перенос

$$X \cdot T = [x \quad y \quad z \quad 1] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix} = [x + t_x \quad y + t_y \quad z + t_z \quad 1]$$



# Трехмерные преобразования координат. Вращение

$$T_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_z = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



# Сложные трехмерные преобразования координат

Последовательные преобразования могут быть скомбинированы или объединены в одно преобразование, дающее тот же самый результат. Так как умножение матриц является некоммутативной операцией, то в общем случае важен порядок ее выполнения. Матрица, ближайшая к матрице координатного вектора, задает первое преобразование, а последняя - последнее преобразование.

$$X \cdot T = X \cdot T_1 \cdot T_2 \cdot \dots \cdot T_n$$



## Сложные трехмерные преобразования. Поворот вокруг оси, параллельной одной из координатных осей

- переместить тело так, чтобы локальная ось совпала с координатной;
- повернуть вокруг указанной оси;
- переместить преобразованное тело в исходное положение.

$$X^* = X \cdot T = X \cdot T_r \cdot R_x \cdot T_r^{-1}$$



## Сложные трехмерные преобразования. Поворот вокруг произвольной оси в пространстве

Поворот вокруг произвольной оси в пространстве выполняется с помощью переноса и простых поворотов вокруг координатных осей. Так как метод поворота вокруг координатной оси известен, то основная идея заключается в том, чтобы совместить произвольную ось вращения с одной из координатных осей.



## Сложные трехмерные преобразования. Поворот вокруг произвольной оси в пространстве

Пусть произвольная ось в пространстве проходит через точку  $(x_0, y_0, z_0)$  с направляющим вектором  $(cx, cy, cz)$ . Поворот вокруг этой оси на некоторый угол  $\alpha$  будет выполняться по следующему правилу:

- выполнить перенос так, чтобы точка  $(x_0, y_0, z_0)$  находилась в начале системы координат – Т1;
- выполнить соответствующие повороты так, чтобы ось вращения совпала, например, с осью Oz – Т2\*T3;
- выполнить поворот на угол  $\alpha$  вокруг оси Oz – Т4;
- выполнить преобразование, обратное тому, что позволило совместить ось вращения с осью Oz;
- выполнить обратный перенос.



## Сложные трехмерные преобразования. Поворот вокруг произвольной оси в пространстве

$$T_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_0 & -y_0 & -z_0 & 1 \end{bmatrix}$$

$$T_4 = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_z/d & c_y/d & 0 \\ 0 & -c_y/d & c_z/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3 = \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} d & 0 & c_x & 0 \\ 0 & 1 & 0 & 0 \\ -c_x & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad d = \sqrt{c_y^2 + c_z^2}$$



# *Геометрические проекции*

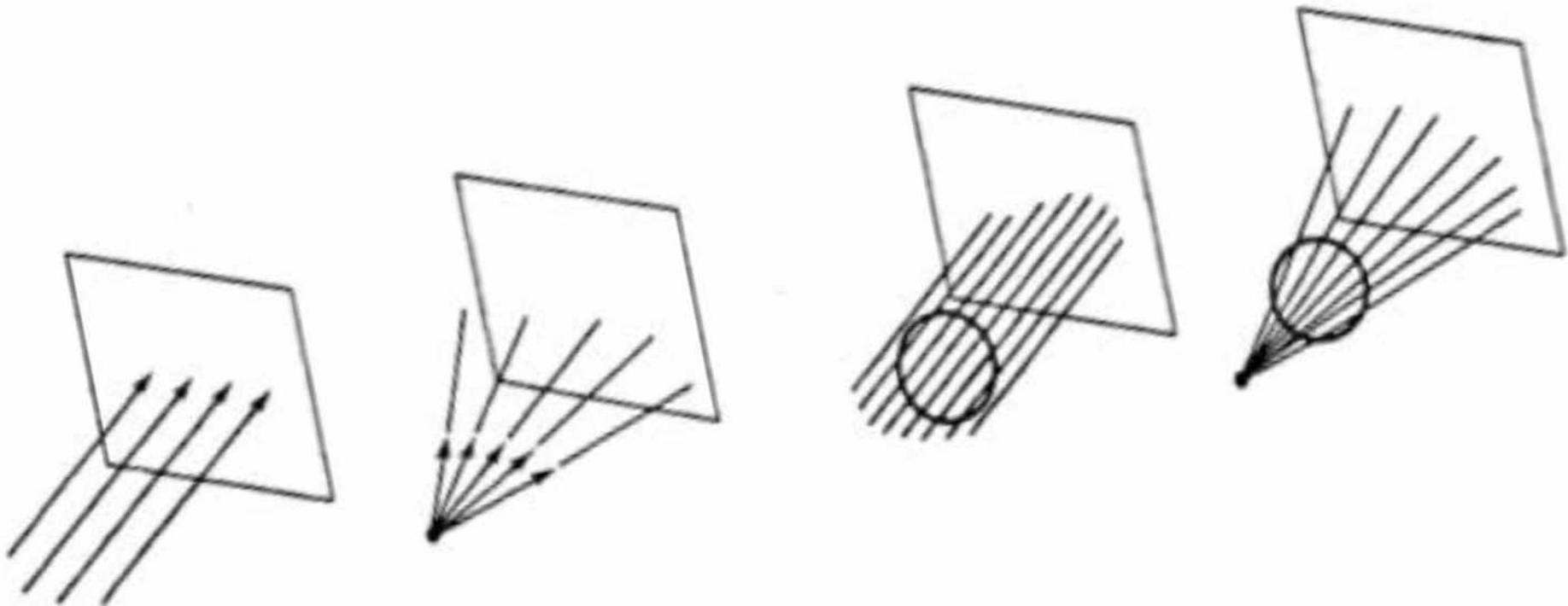


# Плоские геометрические проекции

Для получения проекции объекта на некоторую плоскость (эта плоскость называется картинной) необходимо провести через каждую его точку прямую из заданного проектирующего пучка (эти прямые называются проекторами), и затем найти координаты точки пересечения этой прямой с картинной плоскостью. В случае центрального проектирования все прямые исходят из одной точки, называемой центром проецирования. При параллельном проектировании центр считается лежащим в бесконечности.



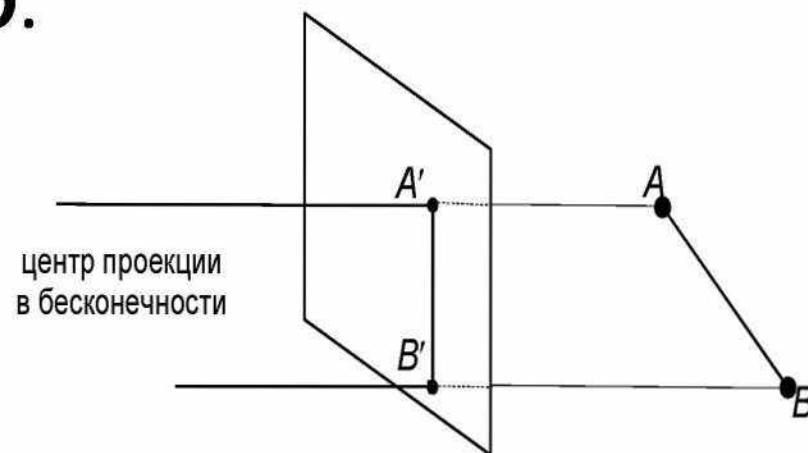
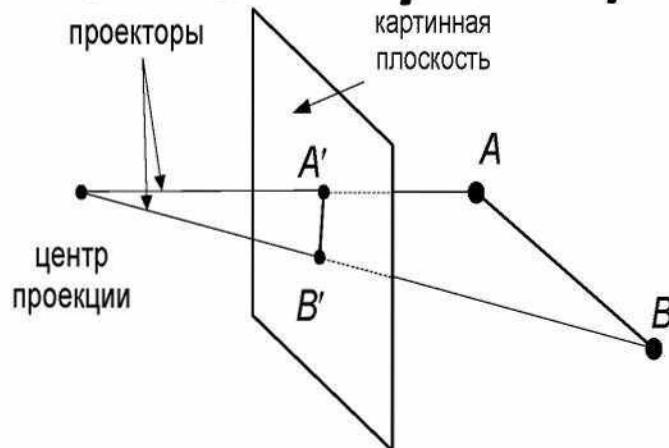
# Плоские геометрические проекции





# Геометрические проекции

- Проекция трехмерного объекта строится при помощи проекционных лучей, которые проходят через каждую точку объекта и, пересекая **картинную плоскость**, образуют **проекцию**.



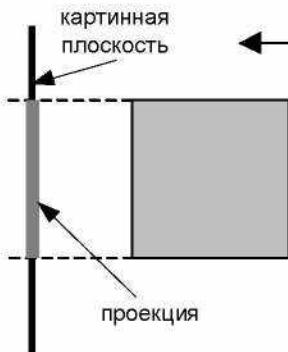
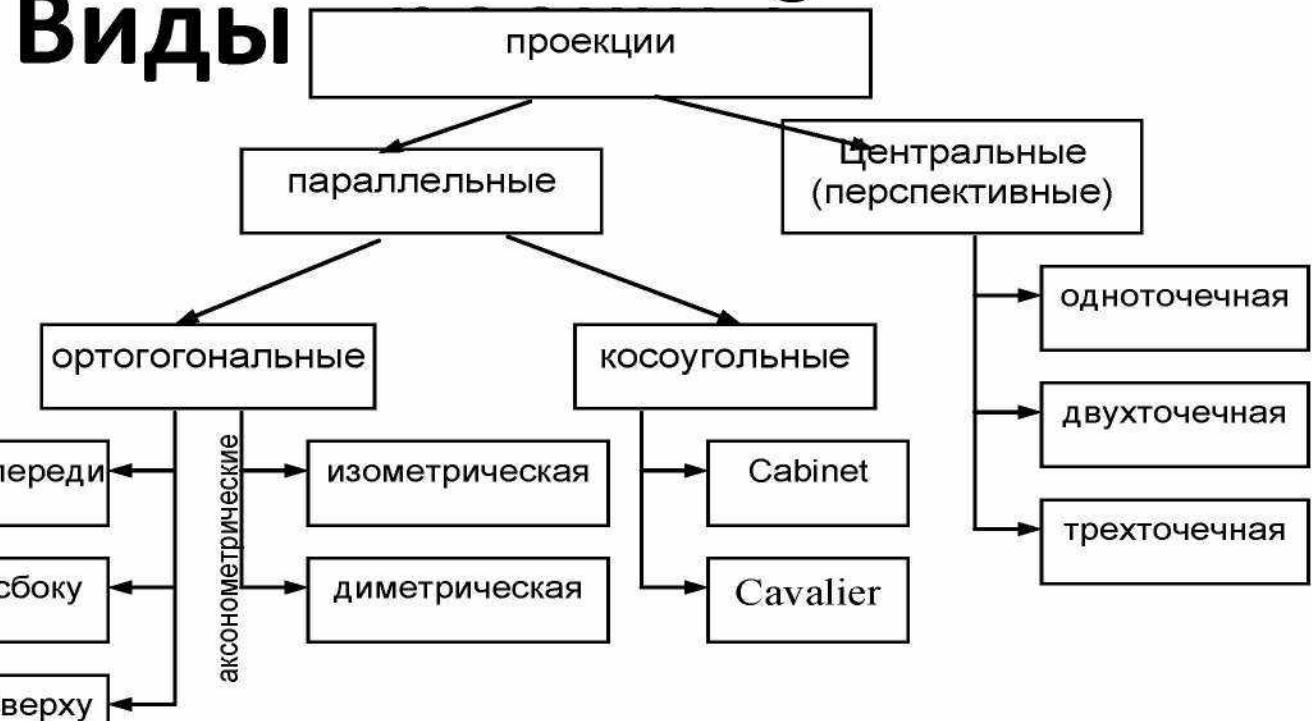


# Геометрические проекции. Иерархия

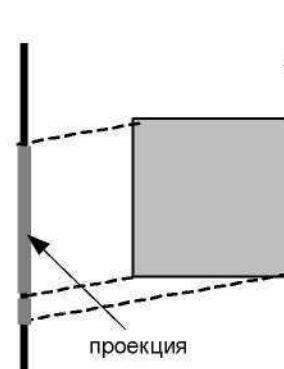




# Виды



ортографические проекции



косоугольные проекции





# Геометрические проекции

**Плоские геометрические проекции** объектов образуются пересечением прямых, называемых проекторами, с плоскостью, называемой плоскостью проекции. Проекторы - это прямые, проходящие через произвольную точку, называемую центром проекции, и каждую точку объекта. Если центр проекции расположен в конечной точке трехмерного пространства, получается **центральная (перспективная)** проекция. Если центр расположен в бесконечности, то все проекторы параллельны и результат является **параллельной** проекцией.



# Геометрические проекции

Для разработки различных преобразований можно использовать два разных подхода. В первом предполагается, что центр проекции или точка зрения фиксирована, а плоскость проекции перпендикулярна каждому проектору. Для получения требуемого вида манипулируют объектом. Во втором подходе предполагается, что объект фиксирован, центр проекции может как угодно перемещаться в трехмерном пространстве, а плоскость проекции не обязательно перпендикулярна направлению взгляда. Оба подхода математически эквивалентны.



# Параллельные проекции

**Параллельные** проекции делятся на три типа в зависимости от соотношения между направлением проецирования и нормалью к проекционной плоскости:

- 1) **ортографические** – направления совпадают, т. е. направление проецирования перпендикулярно к проекционной плоскости; проекторы направлены вдоль одной из координатных осей.
- 2) **аксонометрические** – образуются манипулированием объекта с помощью поворотов и перемещений таким образом, чтобы в его последующей ортографической проекции были видны по крайней мере три соседние грани.
- 3) **косоугольные** – направление проецирования и нормаль к проекционной плоскости не совпадают.



# Ортографические проекции

$$P_{Oxy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad P_{Oxz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_{Oyz} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



# Аксонометрические проекции

Аксонометрическая проекция образуется манипулированием объекта с помощью поворотов и перемещений таким образом, чтобы были видны по крайней мере три соседние грани. Результат затем проецируется с центром проекции, расположенным в бесконечности, на одну из координатных плоскостей, обычно на плоскость  $z=0$ . Если грань не параллельна плоскости проекции, то аксонометрическая проекция не показывает истинную форму этой грани. Однако остаются постоянными относительные длины параллельных в исходном пространстве линий, т. е. параллельные линии одинаково укорачиваются (искажаются).

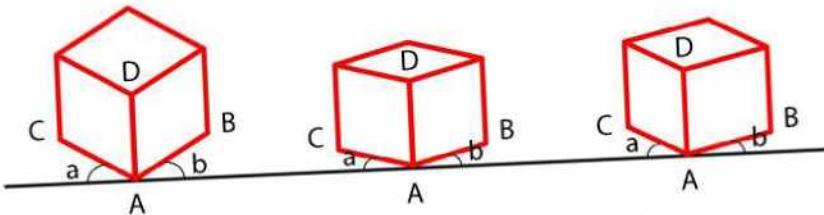


# Аксонометрические проекции

**Коэффициент искажения** есть отношение длины проекции отрезка к его истинной длине. Представляют интерес три аксонометрические проекции: **тритиометрическая, диметрическая и изометрическая**. В тритиометрической проекции меньше всего ограничений, а в изометрической - больше всего. Изометрическая проекция есть частный случай диметрической, а диметрическая проекция есть частный случай тритиометрической.



# Аксонометрические проекции



$a = b = 30$   
 $AB = AC = AD$

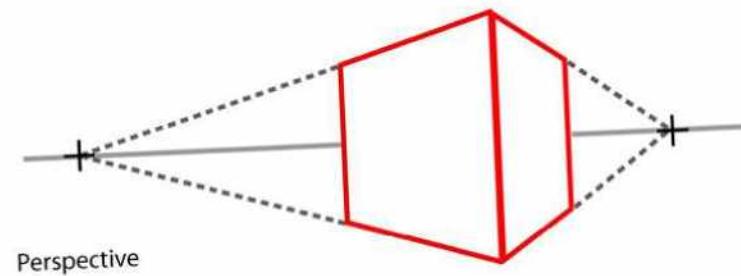
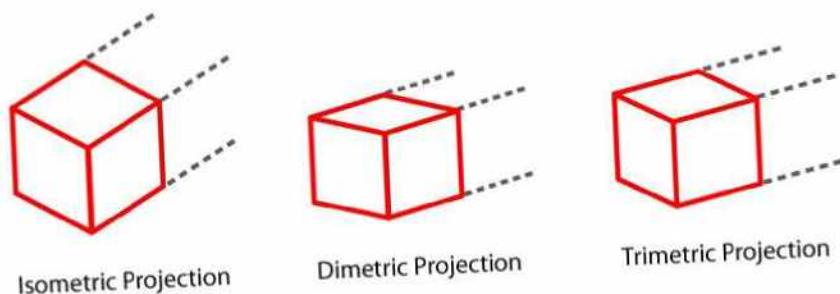
Isometric Projection

$a = b$   
 $AB = AC \neq AD$

Dimetric Projection

$a \neq b$   
 $AB \neq AC \neq AD$

Trimetric Projection



Perspective



# Аксонометрические проекции. Триметрическая проекция

Триметрическая проекция строится произвольными поворотами вокруг любых координатных осей, совершаемыми в произвольном порядке, с последующим проецированием на плоскость  $z=0$ .

В общем случае для триметрической проекции коэффициенты искажения по каждой из проецируемых осей не равны друг другу.



# Аксонометрические проекции. Триметрическая проекция

Коэффициенты искажения  
( $T$  – общая матрица преобразования):

$$K = UT = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \cdot T = \begin{bmatrix} x_x^* & y_x^* & 0 & 1 \\ x_y^* & y_y^* & 0 & 1 \\ x_z^* & y_z^* & 1 & 1 \end{bmatrix}$$



# Аксонометрические проекции. Триметрическая проекция

Коэффициенты искажения:

$$f_x = \sqrt{x_x^{*2} + y_x^{*2}}$$

$$f_y = \sqrt{x_y^{*2} + y_y^{*2}}$$

$$f_z = \sqrt{x_z^{*2} + y_z^{*2}}$$



# Аксонометрические проекции. Диметрическая проекция

Диметрическая проекция - это триметрическая проекция с двумя одинаковыми коэффициентами искажения, третий коэффициент может иметь любое значение.

Диметрическая проекция строится с помощью поворота вокруг оси  $Oy$ , затем поворота вокруг оси  $Ox$  и проецирования на плоскость  $z=0$ .

Углы поворотов выбираются таким образом, чтобы коэффициенты искажения по осям  $Ox$  и  $Oy$  были равны друг другу.



# Аксонометрические проекции. Диметрическая проекция

Коэффициенты искажения  
(Т – общая матрица преобразования):

$$T = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\delta & \sin\delta & 0 \\ 0 & -\sin\delta & \cos\delta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$K = UT = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \cdot T = \begin{bmatrix} \cos\theta & \sin\theta \sin\delta & 0 & 1 \\ 0 & \cos\delta & 0 & 1 \\ \sin\theta & -\cos\theta \sin\delta & 0 & 1 \end{bmatrix}$$



# Аксонометрические проекции. Диметрическая проекция

Коэффициенты искажения

$$f_x = \sqrt{\cos^2 \varphi + \sin^2 \varphi \sin^2 \theta}$$

$$f_y = \sqrt{\cos^2 \theta}$$

$$f_z = \sqrt{\sin^2 \varphi + \cos^2 \varphi \sin^2 \theta}$$



# Аксонометрические проекции. Изометрическая проекция

Изометрическая проекция - это триметрическая проекция, в которой все три коэффициента искажения равны.

$$f_x = f_y = f_z$$

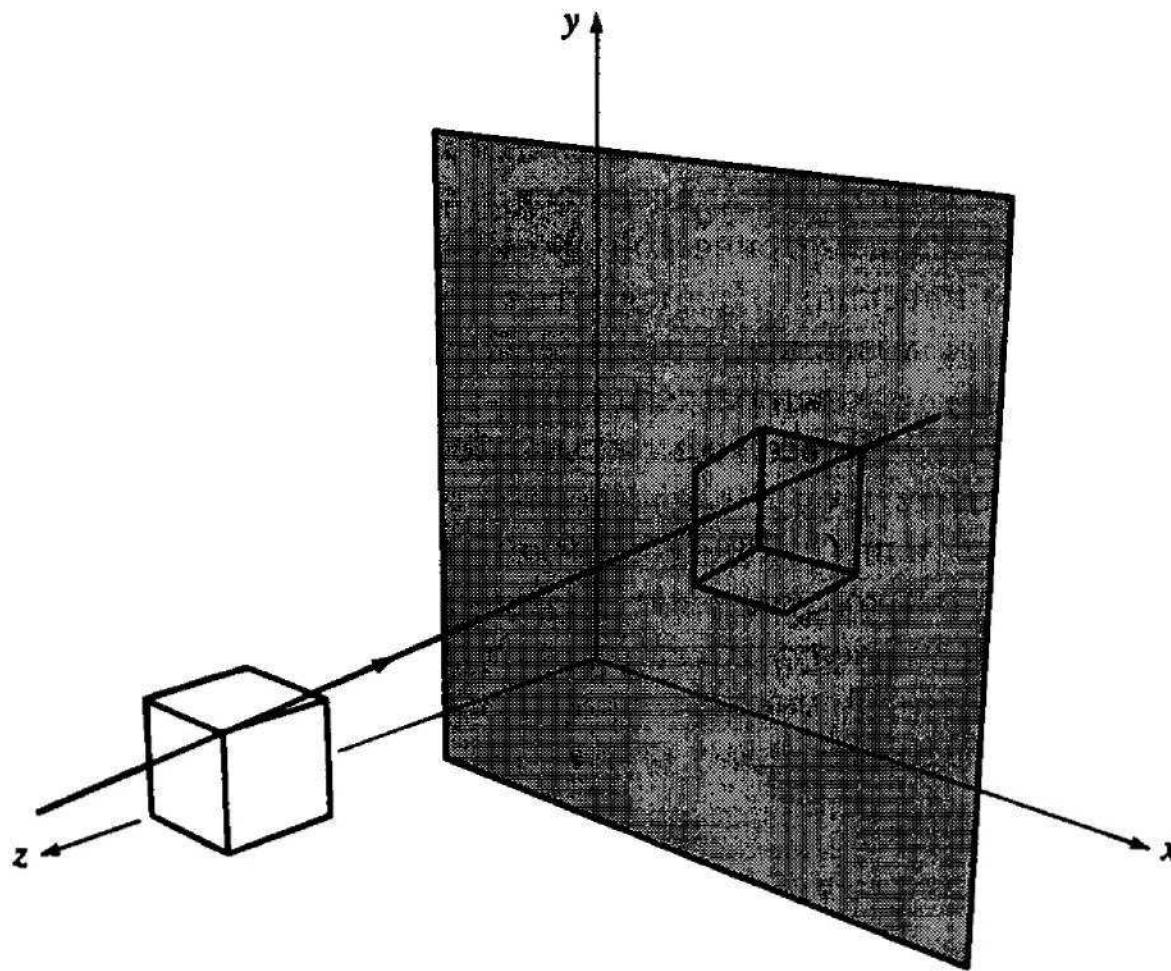


# Косоугольные проекции

В противоположность ортографическим и аксонометрическим проекциям, для которых проекторы перпендикулярны плоскости проекции, косоугольная проекция формируется параллельными проекторами с центром, лежащим в бесконечности, и расположенными под косым углом к плоскости проекции.



# Косоугольные проекции





# Косоугольные проекции

Особый интерес представляют две косоугольные проекции - **кавалье** и **кабине**. Проекция **кавалье** получается, когда угол между проекторами и плоскостью проекции составляет  $45^\circ$ . В этой проекции коэффициенты искажения для всех трех главных направлений одинаковы. Результат этой проекции выглядит неестественно утолщенным.



# Косоугольные проекции

Проекцией **кабине** называется такая косоугольная проекция, у которой коэффициент искажения для ребер, перпендикулярных плоскости проекции, равен  $1/2$ .

Для проекции кабине угол между проекторами и плоскостью проекции составляет  $\arctg(1/2)=63.43^\circ$



# Косоугольные проекции

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -f \cos \alpha & -f \sin \alpha & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



# Перспективные преобразования

Перспективное преобразование имеет место, когда не равен нулю любой из первых трех элементов четвертого столбца обобщенной матрицы преобразования однородных координат. В отличие параллельных преобразований происходит неоднородное искажение линий объекта, зависящее от ориентации и расстояния от объекта до центра проекции. Все это помогает нашему восприятию глубины, но не сохраняет форму объекта.



# Перспективные преобразования

Одноточечное перспективное преобразование:

$$X \cdot T = [x \quad y \quad z \quad 1] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$
$$[x \quad y \quad z \quad rz + 1] \rightarrow \left[ \frac{x}{rz + 1} \quad \frac{y}{rz + 1} \quad \frac{z}{rz + 1} \quad 1 \right]$$



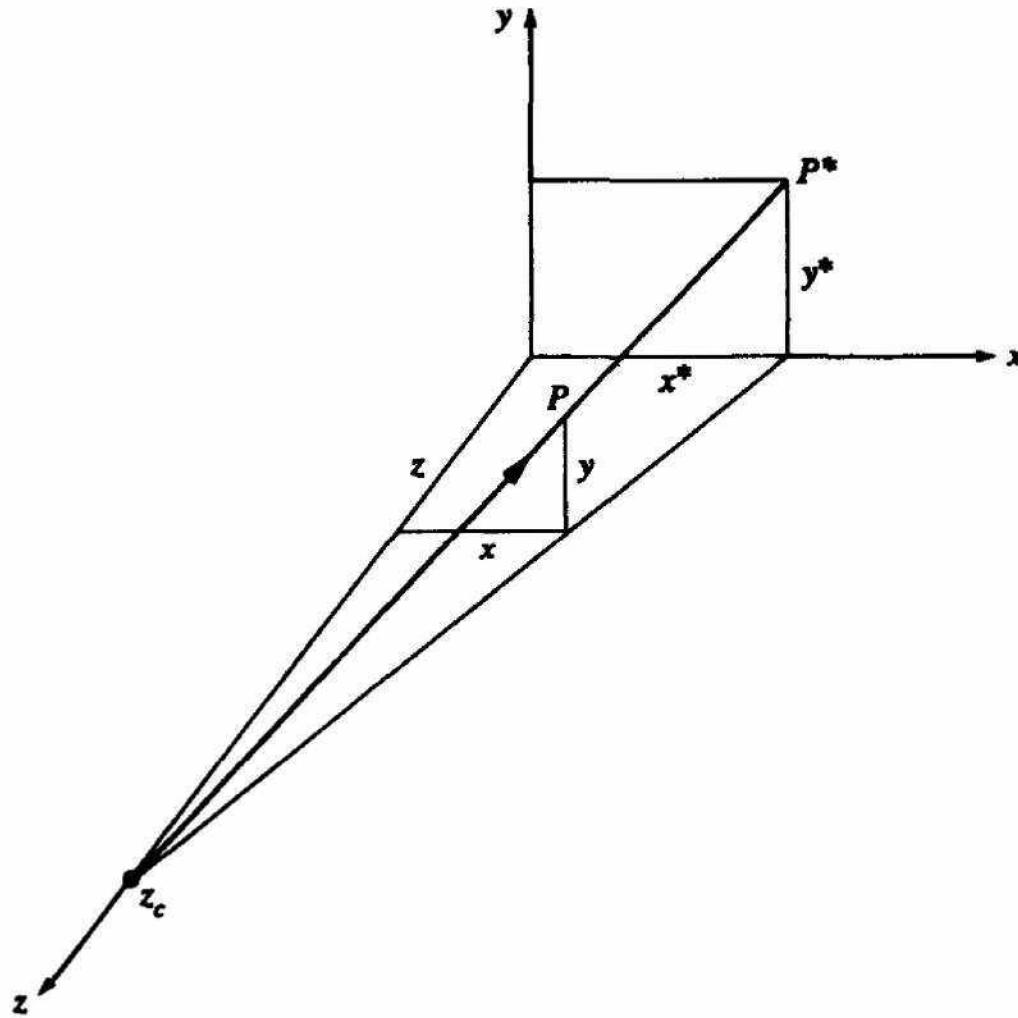
# Перспективные преобразования

Перспективное проецирование на некоторую двумерную видовую плоскость можно получить, объединив ортографическую проекцию с перспективным преобразованием. Например, перспективное проецирование на плоскость  $z=0$  выполняется с помощью преобразований:

$$X \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = X \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & r \\ 0 & 0 & 0 & 1 \end{bmatrix} = \\ = [x \quad y \quad 0 \quad rz+1] \rightarrow \begin{bmatrix} \frac{x}{rz+1} & \frac{y}{rz+1} & 0 & 1 \end{bmatrix}$$



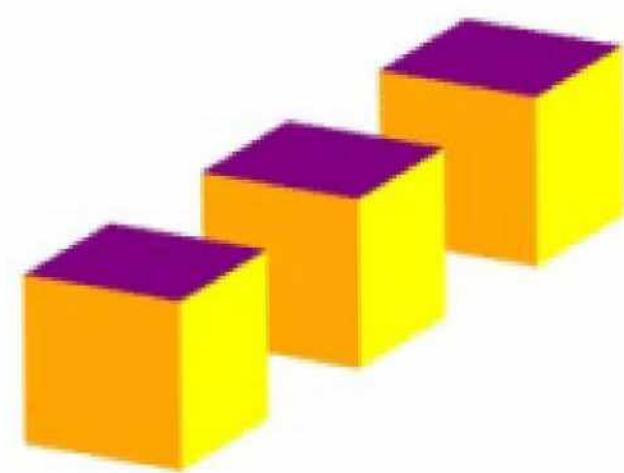
# Перспективные преобразования



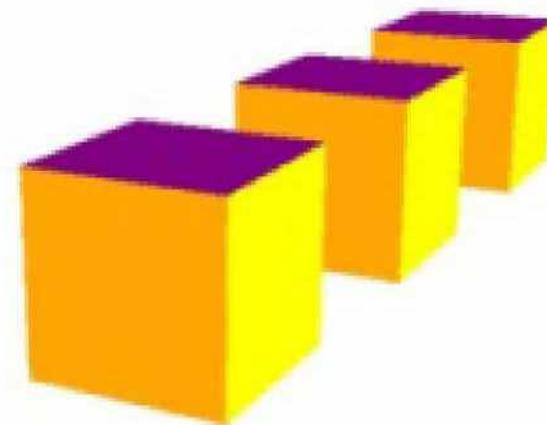


# Перспективные преобразования

Orthographic Projection



Perspective Projection





# **Освещение в компьютерной графике**



# Освещение в компьютерной графике

Световая энергия, падающая на поверхность, может быть поглощена, отражена или пропущена. Объект можно увидеть, только если он отражает или пропускает свет; если же объект поглощает весь падающий свет, то он невидим и называется абсолютно черным телом.

Свойства отраженного света зависят от строения, направления и формы источника света, от ориентации и свойств поверхности. Отраженный от объекта свет может также быть диффузным или зеркальным. **Диффузное** отражение света происходит, когда свет как бы проникает под поверхность объекта, поглощается, а затем вновь испускается. При этом положение наблюдателя не имеет значения, так как диффузно отраженный свет рассеивается равномерно по всем направлениям. **Зеркальное** отражение происходит от внешней поверхности объекта.



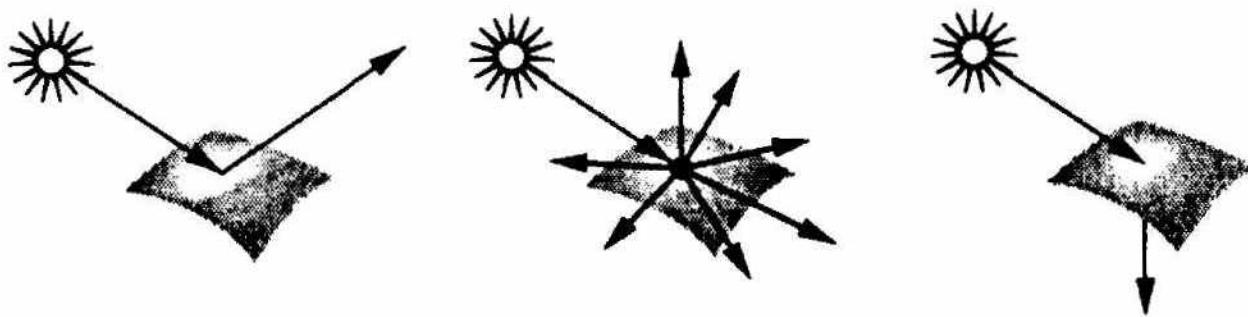
# Освещение в компьютерной графике

Три основных типа взаимодействия света и материала поверхности:

- **Зеркальное отражение.** Поверхности выглядят блестящими, поскольку хотя небольшая часть энергии и поглощается, остальной свет отражается под одним углом, равным углу падения.
- **Диффузное отражение.** При диффузном отражении падающий свет рассеивается в разных направлениях. Такой тип взаимодействия характерен для равномерно окрашенной поверхности.
- **Преломление.** Отражается часть падающего света.



# Освещение в компьютерной графике





# Освещение в компьютерной графике

Как правило, источником света мы считаем излучающие объекты.

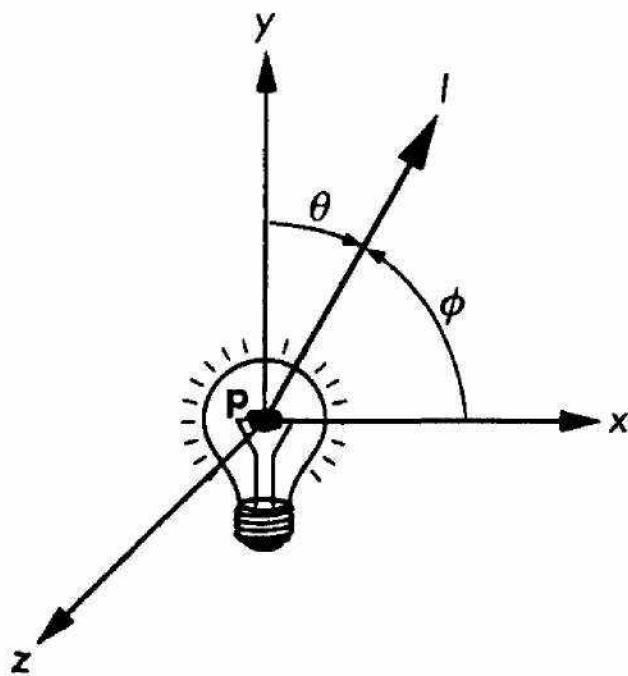
Идеальный точечный источник света (point source) излучает свет одинаково во всех направлениях.

Элементарный источник света в общем случае характеризуется **функцией излучения** (illumination function),  $I(x, y, z, \theta, \varphi, \lambda)$ , зависящей от шести параметров.

$$\text{В общем случае } I = \begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix}$$



# Освещение в компьютерной графике





# Освещение в компьютерной графике

Четыре основных типа источников света:

- фоновое освещение (ambient lighting);
- точечные источники (point sources);
- прожекторы (spotlights);
- удаленные источники света (distant light).



# Отражение фонового цвета

Интенсивность падающего на поверхность фонового света  $I_a$  одинакова во всех точках этой поверхности. Частично энергия этого света поглощается материалом поверхности, а частично — отражается. Отражение фонового света характеризуется коэффициентом  $R=k_a (0 \leq k_a \leq 1)$ . Следовательно, получим:

$$I_a^* = I_a k_a \quad (1)$$

$$k_a = [k_{ar}, k_{ag}, k_{ab}]$$



# Диффузное отражение. Закон Ламберта

Свет точечного источника отражается от идеального рассеивателя по закону косинусов Ламберта: интенсивность отраженного света пропорциональна косинусу угла между направлением света и нормалью к поверхности:

$$I_d^* = I_d k_d \cos\theta \quad (2)$$

$k_d$  - коэффициент диффузного отражения ( $0 \leq k_d \leq 1$ )  
 $\theta$  - угол между направлением света и нормалью к поверхности ( $0 \leq \theta \leq \pi/2$ )



# Диффузное отражение. Закон Ламберта

В это соотношение можно ввести и множитель, зависящий от расстояния, который позволит учесть ослабление светового потока по мере удаления источника от анализируемой точки поверхности

$$I_d^* = I_d k_d \cos\theta / (K + D) \quad (3)$$

D – расстояние до источника

K – произвольная постоянная



# Зеркальное отражение

Интенсивность зеркально отраженного света зависит от угла падения, длины волны падающего света и свойств вещества. Угол отражения от идеальной отражающей поверхности (зеркала) равен углу падения, в любом другом положении наблюдатель не видит зеркально отраженный свет.

$$I_s^* = I_s k_s \cos^\alpha \phi \quad (4)$$

$k_s$  - коэффициент зеркального отражения ( $0 \leq k_s \leq 1$ )

$\alpha$  - коэффициент резкости бликов (shininess coefficient)

$\phi$  – угол между вектором, характеризующим направление идеального зеркального отражения, и вектором, направленным к наблюдателю



# Модель Фонга

$$I^* = I_a k_a + \frac{I_d k_d \cos \theta + I_s k_s \cos^\alpha \varphi}{K + D}$$

$k_a$  - коэффициент диффузного отражения рассеянного света

$k_d$  - коэффициент диффузного отражения

$k_s$  - коэффициент зеркального отражения

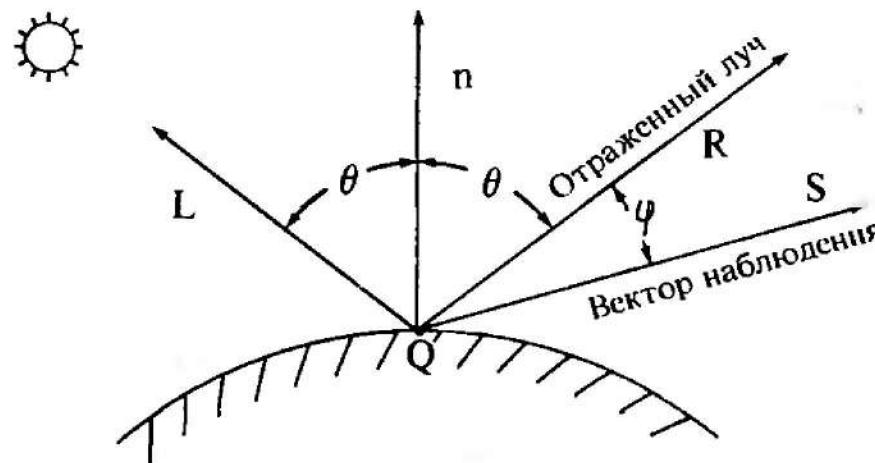
$\theta$  - угол между направлением света и нормалью к поверхность

$\varphi$  – угол между вектором, характеризующим направление

идеального зеркального отражения, и вектором, направленным к наблюдателю

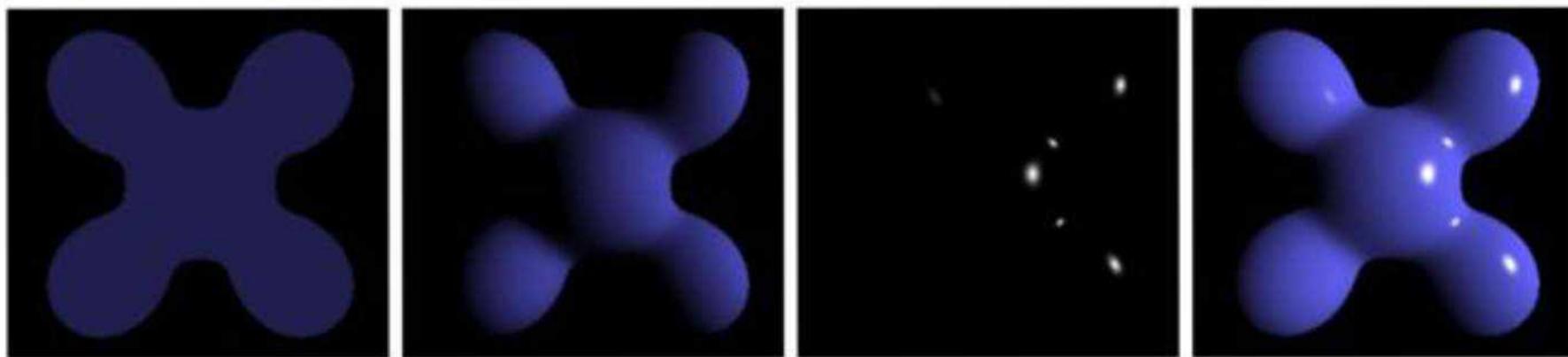


# Модель Фонга

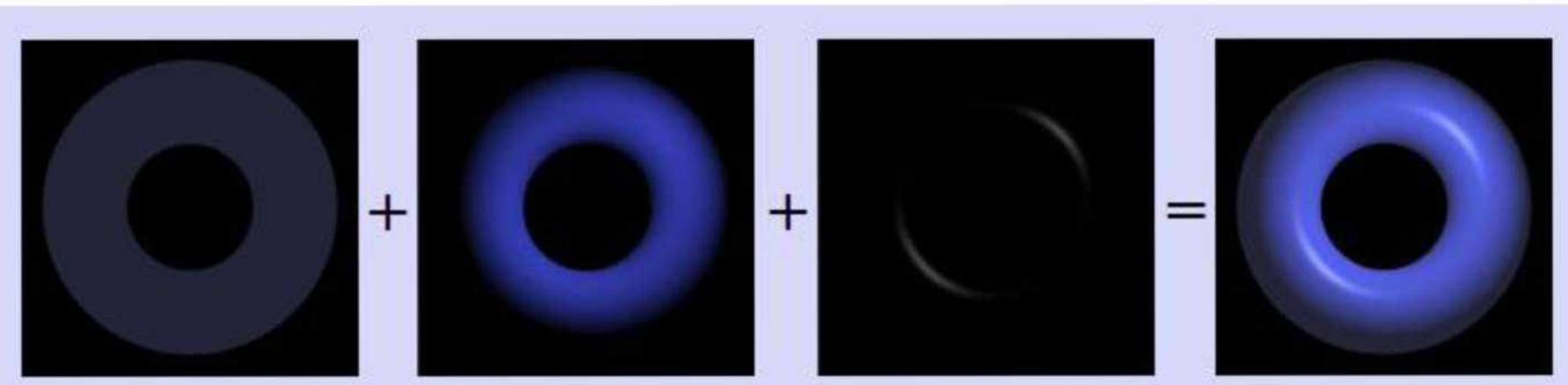




# Модель Фонга



Ambient      +      Diffuse      +      Specular      =      Phong Reflection



Фоновая составляющая

Рассеянная составляющая

Зеркальная составляющая

Суммарное освещение