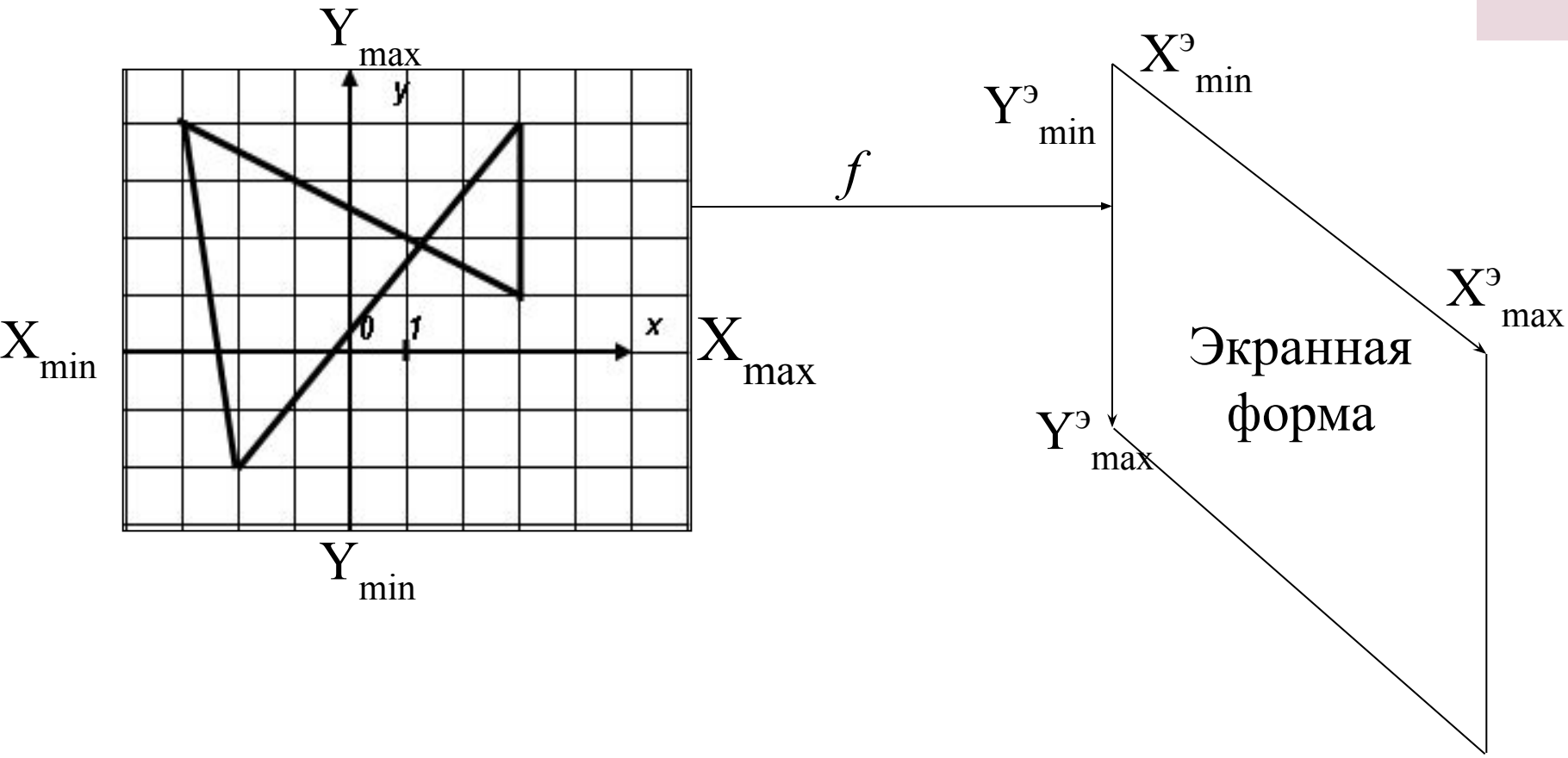
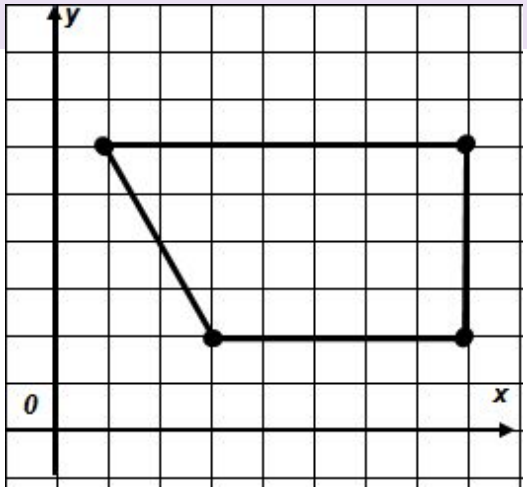
«Базовые вычислительные и растровые алгоритмы»

1. **Область визуализации и функция кадрирования.**
2. **Отсечение.**
3. **Операции с изображением на уровне растра .**

Область визуализации 2D



Пример



$$(x_0, y_0, 1) = (3, 2, 1);$$
$$(x_1, y_1, 1) = (1, 6, 1);$$
$$(x_2, y_2, 1) = (8, 2, 1);$$
$$(x_3, y_3, 1) = (8, 6, 1);$$

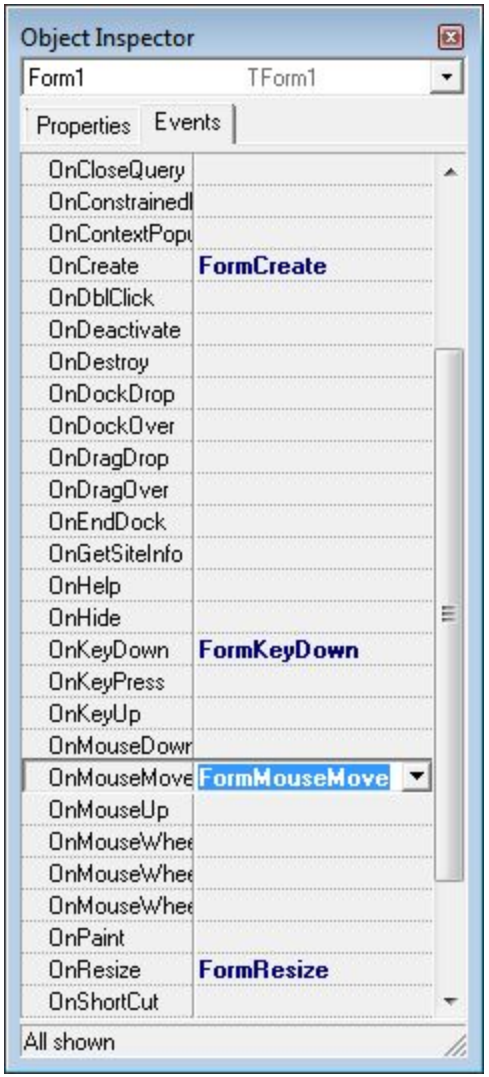
$$[A] = \begin{bmatrix} 3 & 2 & 1 \\ 1 & 6 & 1 \\ 8 & 2 & 1 \\ 8 & 6 & 1 \end{bmatrix}$$

Функция кадрирования

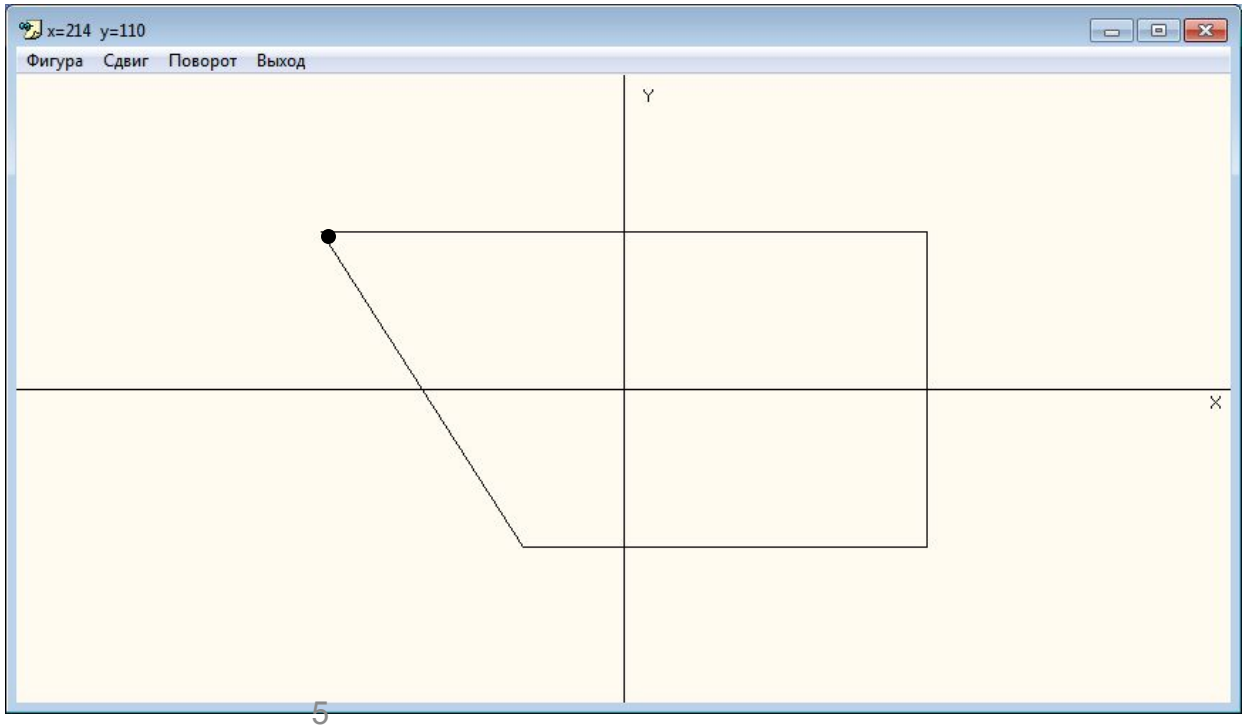
$$f_x = \frac{X^э_{\max} - X^э_{\min}}{X_{\max} - X_{\min}}, \quad f_y = \frac{Y^э_{\max} - Y^э_{\min}}{Y_{\max} - Y_{\min}},$$

$$\begin{cases} X = X^э_{\min} + f_x(x - x_{\min}) \\ Y = Y^э_{\min} + f_y(y - y_{\min}) \end{cases}$$

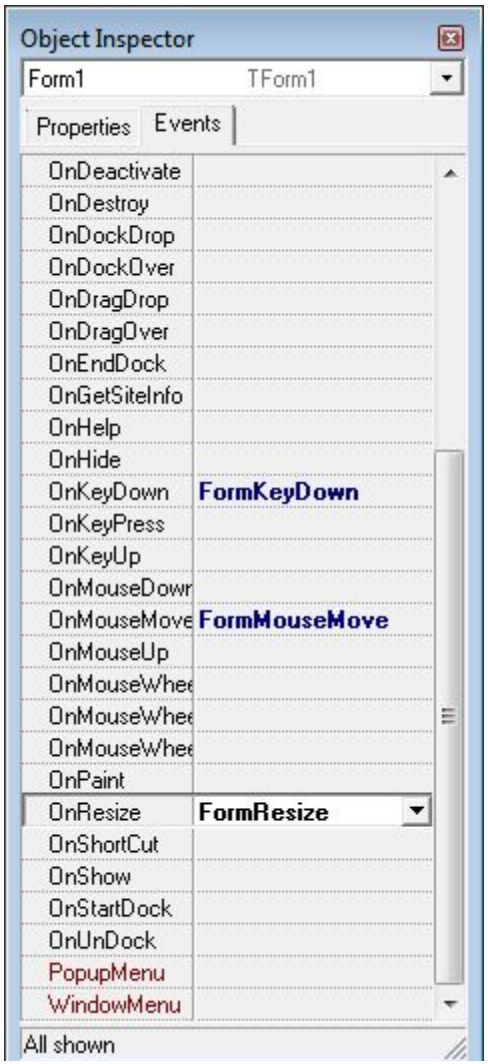
Координаты курсора



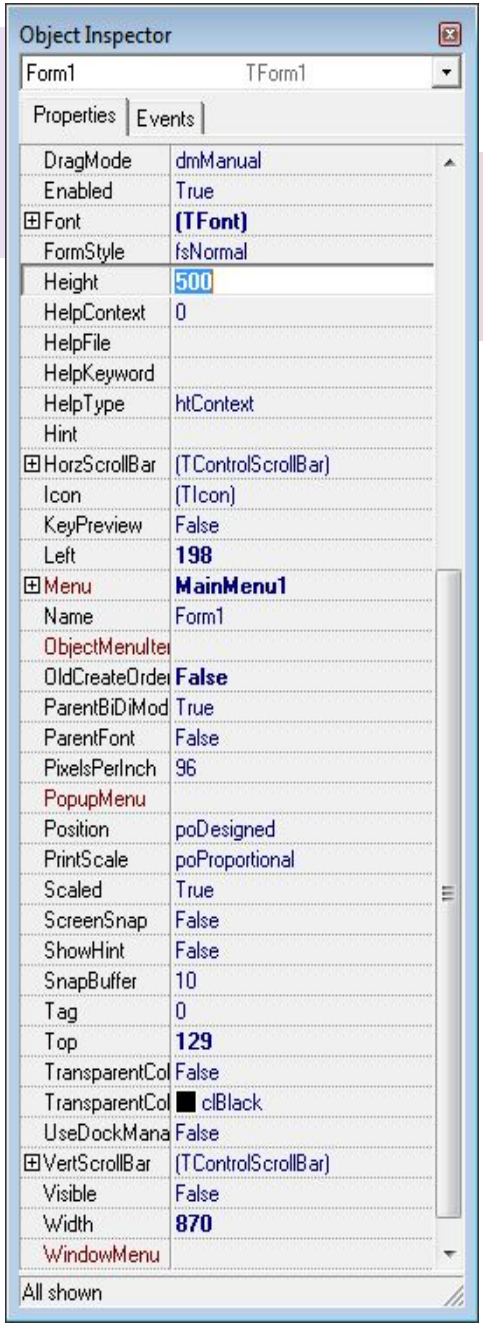

```
procedure TForm1.FormMouseMove(Sender: TObject; Shift:
TShiftState; X,
  Y: Integer);
begin
Caption:= 'x='+ IntToStr(x)+'  y='+IntToStr(y);
end;
```



Экранные координаты

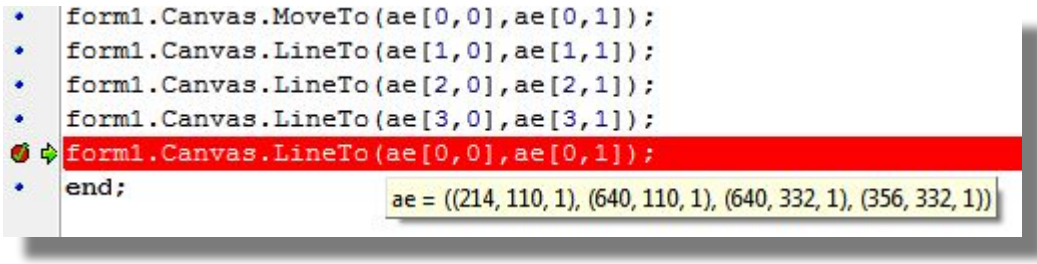



```
procedure TForm1.FormResize(Sender:

TObject);

begin

yemin:= round(Form1.ClientHeight/4);

yemax:= round((3*Form1.ClientHeight)/4);

xemin:= round (Form1.ClientWidth/4);

xemax:= round((3*Form1.ClientWidth)/4);

end;
```



Экранные координаты

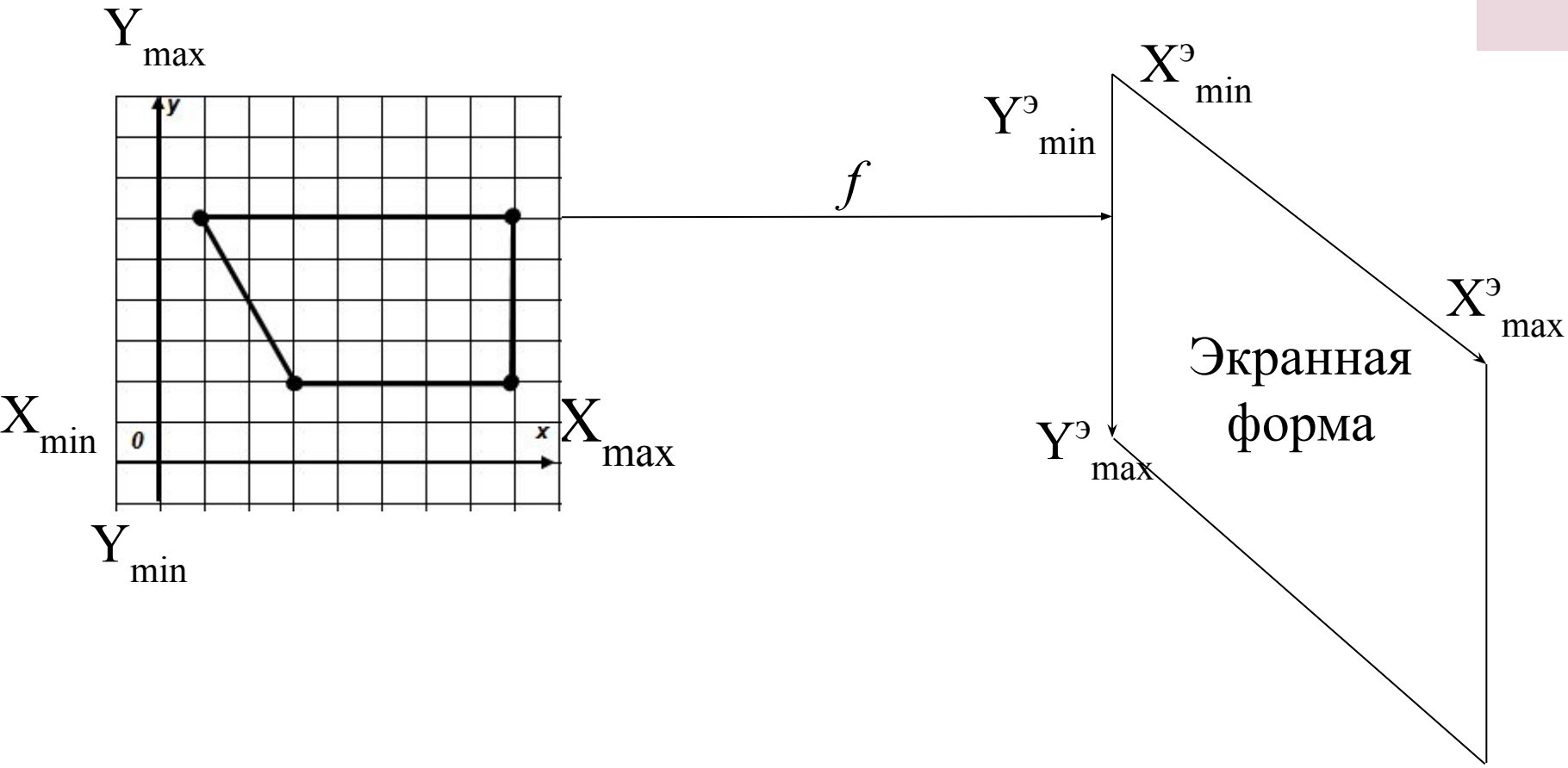
```
function SistCoord(a:vertex):vertex;
Var fx,fy:real; i,j,de:integer;
Begin
  fx:=(xemax-xemin)/(xmax-xmin);
  fy:=(yemax-yemin)/(ymax-ymin);
  de:=round(sqrt(sqr(Form1.ClientWidth)-sqr(Form1.ClientHeight)));
       for i:=0 to 3 do
       begin
       ae[i,0]:=round(xemin+(fx*(a[i,0]-xmin)));
       ae[i,1]:=round(yemax-(fy*(a[i,1]-ymin)));
       ae[i,2]:= 1;
        end;
form1.Canvas.MoveTo(ae[0,0],ae[0,1]);
form1.Canvas.LineTo(ae[1,0],ae[1,1]);
form1.Canvas.LineTo(ae[2,0],ae[2,1]);
form1.Canvas.LineTo(ae[3,0],ae[3,1]);
form1.Canvas.LineTo(ae[0,0],ae[0,1]);
end;
```



Преобразование координат

$$[a] = \begin{bmatrix} 3 & 2 & 1 \\ 1 & 6 & 1 \\ 8 & 2 & 1 \\ 8 & 6 & 1 \end{bmatrix} \xrightarrow{f} [ae] = \begin{bmatrix} 214 & 110 & 1 \\ 640 & 110 & 1 \\ 640 & 332 & 1 \\ 356 & 332 & 1 \end{bmatrix}$$

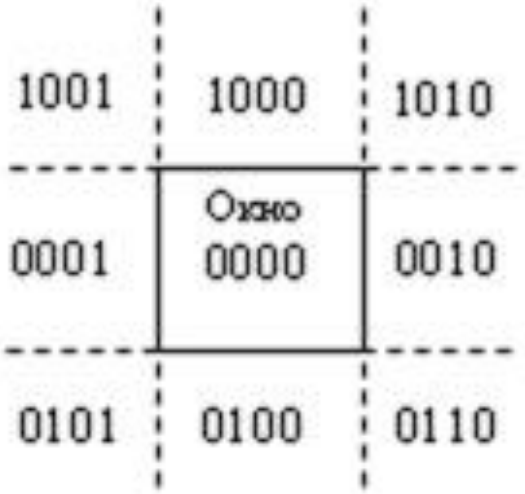
Область визуализации

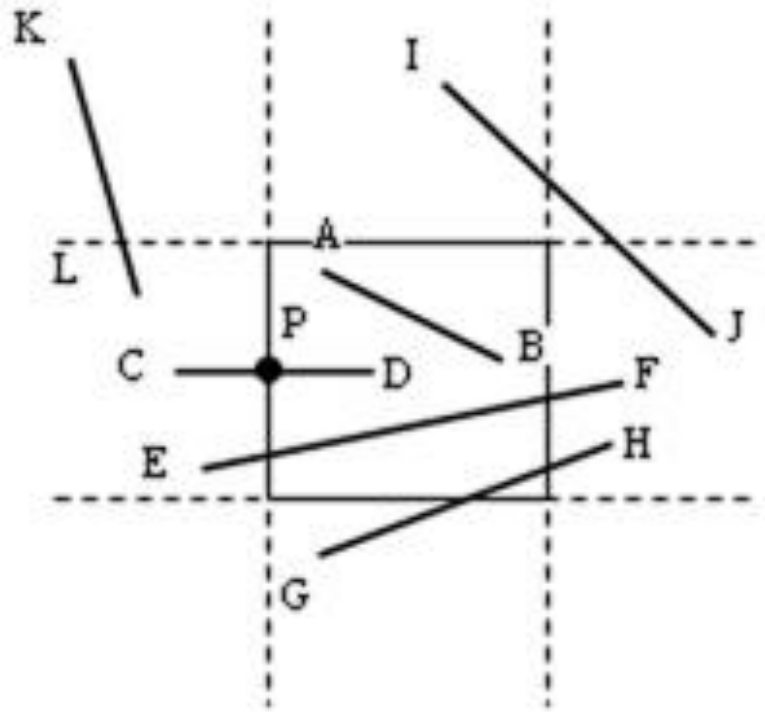


Отсечение

- **алгоритмы, использующие кодирование концов отрезка или всего отрезка (Коэна-Сазерленда);**
- **алгоритмы, использующие параметрическое представление отсекаемых отрезков и окна отсечения (Лианга-Барского).**

Алгоритм Коэна-Сазерленда

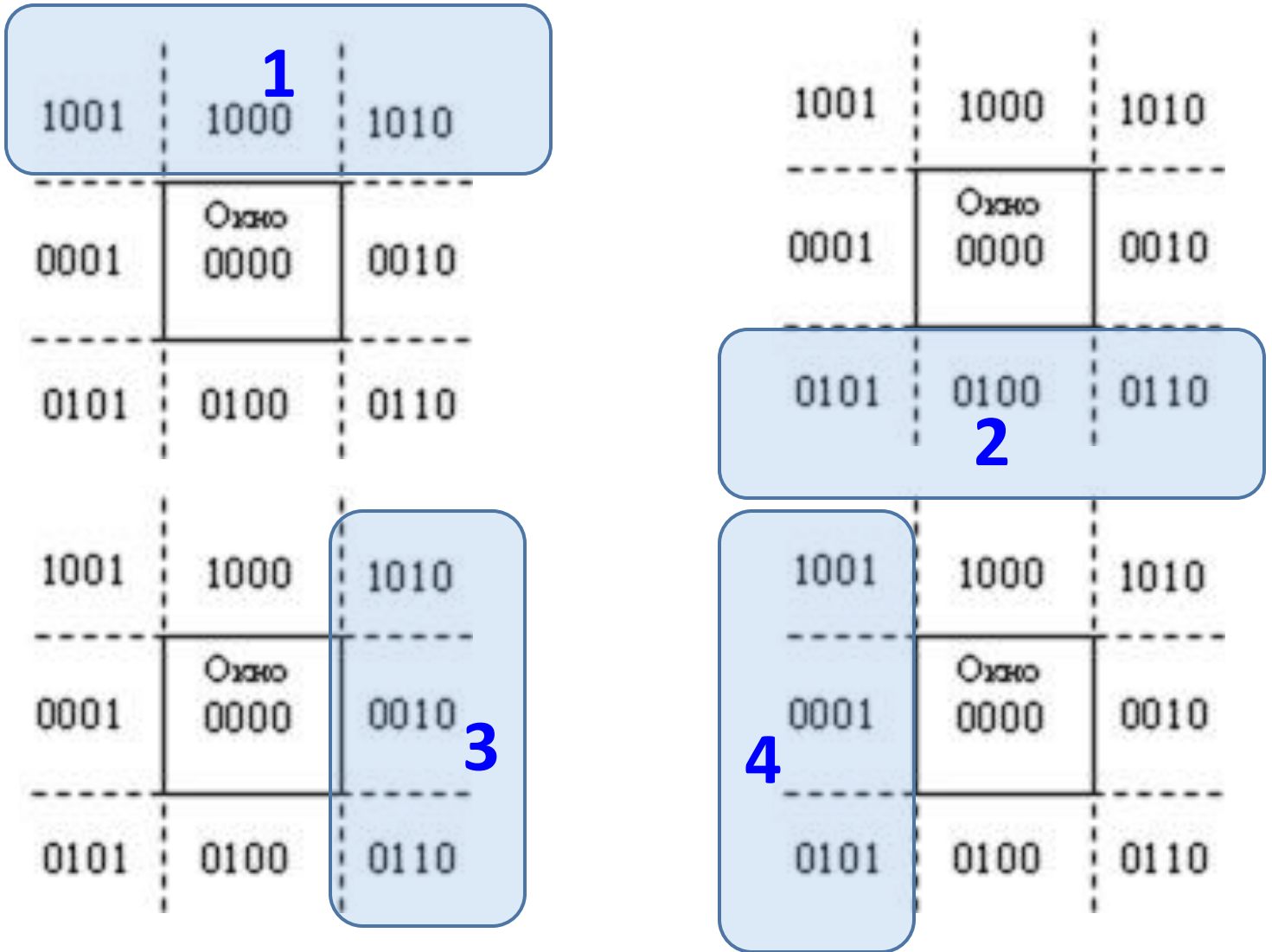




Алгоритм Коэна-Сазерленда

- **Две конечные точки отрезка получают 4-х разрядные коды, соответствующие областям, в которые они попали:**
- **1 рр = 1 - точка над верхним краем окна;**
- **2 рр = 1 - точка под нижним краем окна;**
- **3 рр = 1 - точка справа от правого края окна;**
- **4 рр = 1 - точка слева от левого края окна.**

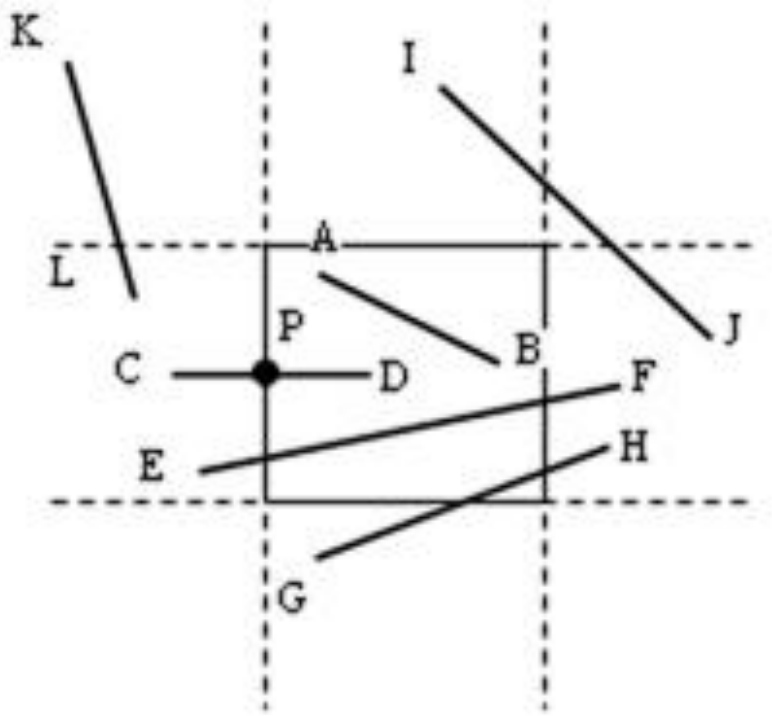
Битовый код



Битовый код

- A=0000;
- B=0000;
- C=0001;
- D=0000;
- E=0001;
- F=0010;
- G=0101;
- H=0010;
- I=1000;
- J=0010;
- K=1001;
- L=0001.

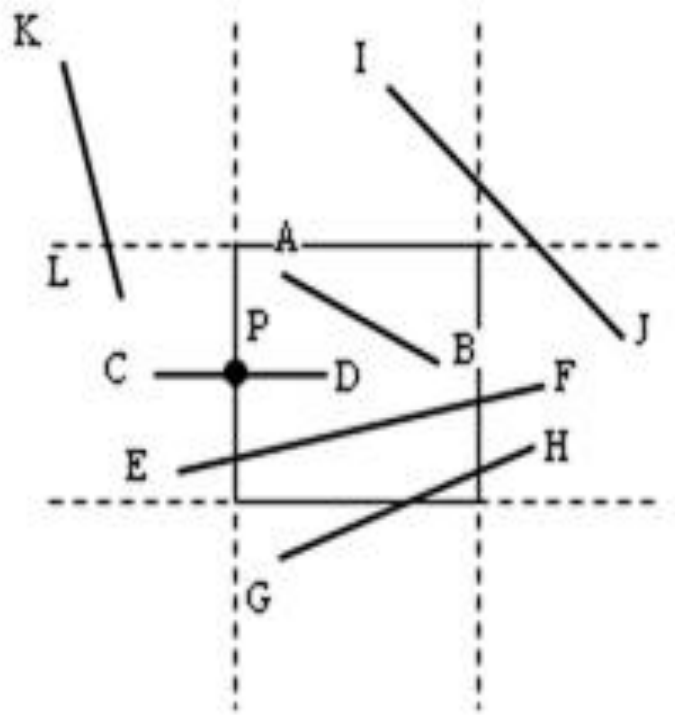
Алгоритм Коэна-Сазерленда



Пусть X - код точки-начала отрезка, Y - код точки-конца отрезка, тогда возможны три случая:

1. X = Y = 0000. Этот случай означает, что обе точки лежат внутри прямоугольника (т. е. отсечение не требуется).
2. X and Y ≠ 0. В этом случае точки лежат по одну сторону от какой-либо отсекающей линии (с внешней ее стороны). Следовательно, отрезок полностью лежит вне окна.
3. Если не выполнены условия 1 или 2, то необходимо находить точки пересечения с некоторыми из отсекающих прямых. Для этого разбивают отрезок найденными точками пересечения и затем применяют тот же анализ кодов концов для полученных новых отрезков.

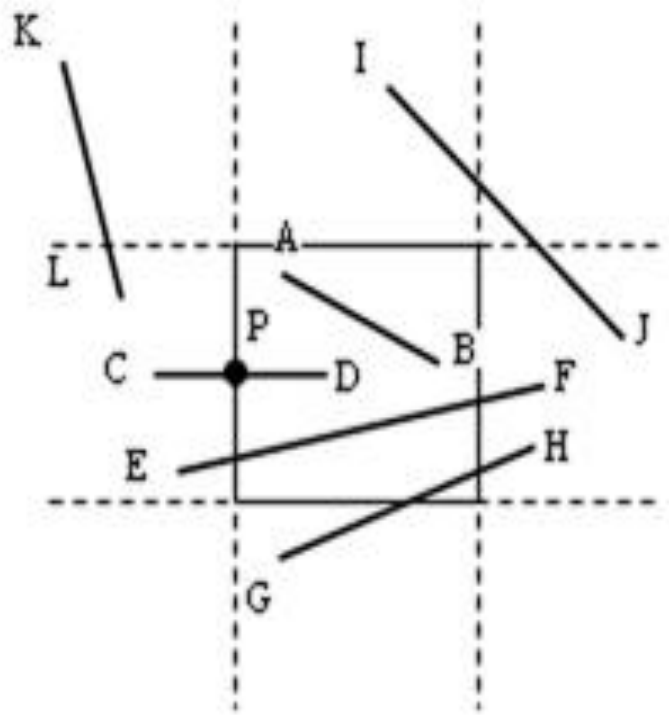
Алгоритм Коэна-Сазерленда



- X = Y=0:
- Отрезок AB

(0000)and(0000)

Алгоритм Коэна-Сазерленда



- X and Y≠0:
- Отрезок KL

(1001)and(0001)

Алгоритм Коэна-Сазерленда

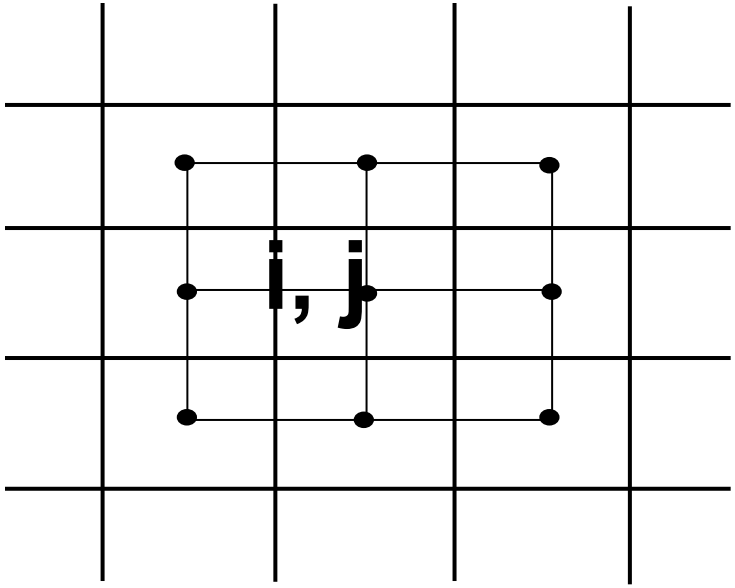
Отрезки	Коды концов	Результаты логического умножения	Примечание
AB	0000 0000	0000	Целиком видим
CD	0001 0000	0000	
EF	0001 0010	0000	
GH	0101 0010	0000	
IJ	1000 0010	0000	
KL	1001 0001	0001	Целиком невидим

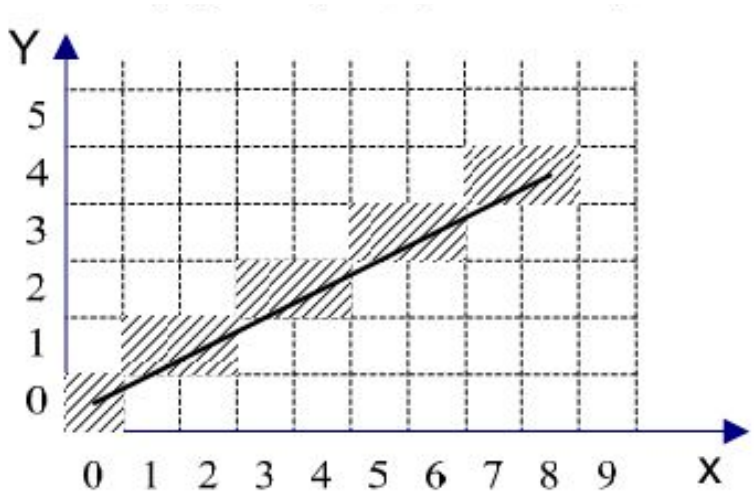
Задание

- Используя алгоритм Коэна-Сазерленда, выявить видимые, невидимые и отсекаемые отрезки AB, CD, EF, GH, IJ, KL.
- Если известны: A(0000), B(0010), C(0001), D(1001), E (0110), F(0010), G(0000), H(0000), I(1000), J(0010), K(1001), L(1010).
- Отобразите расположение отрезков относительно окна отсечения.

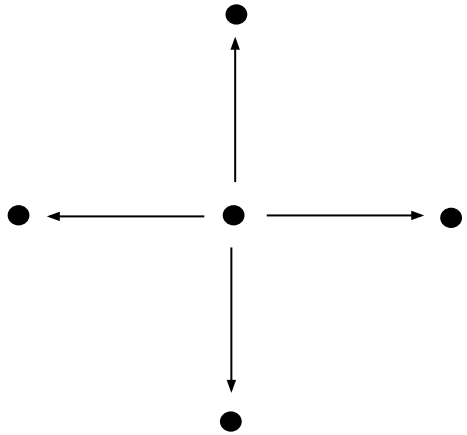
Операции с изображением на уровне растра

Операции с изображением на уровне растра



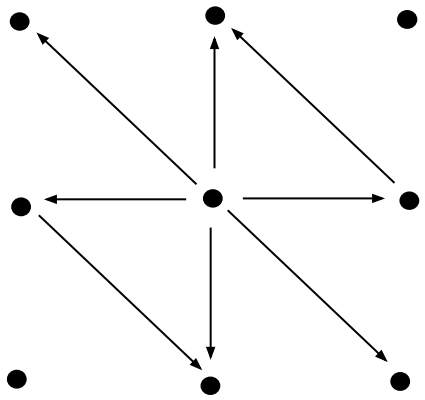


Непосредственные соседи



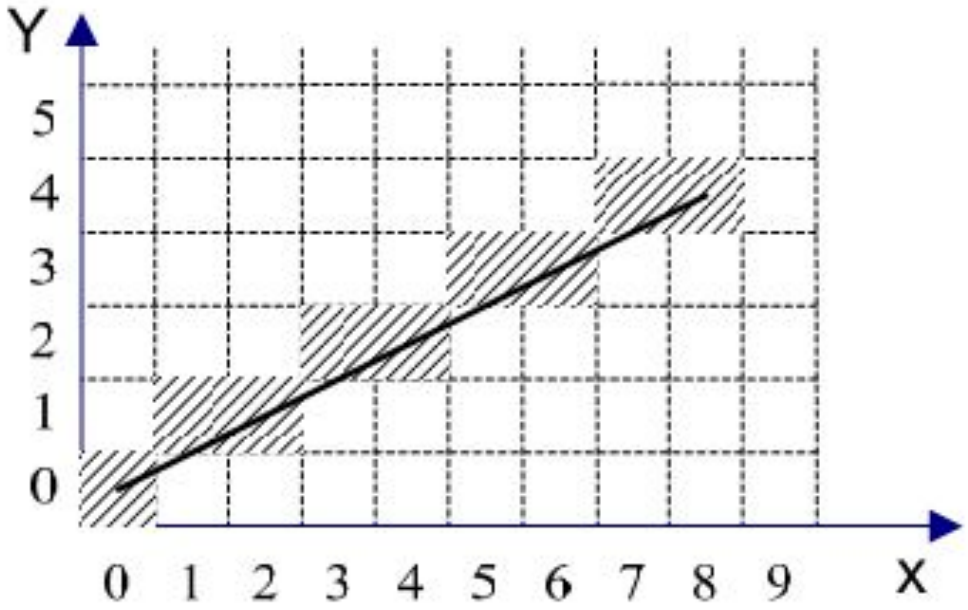
Точки на плоскости называются *непосредственными соседями* (4-соседями) если у них отличаются только x-координаты или только y-координаты, причем только на 1.

Косвенные соседи



Точки на плоскости называются *косвенными соседями* (8-соседями) если у них отличаются x-координаты или y-координаты, но не более чем на 1.

Прямое вычисление координат



$$y = k \cdot x + b,$$

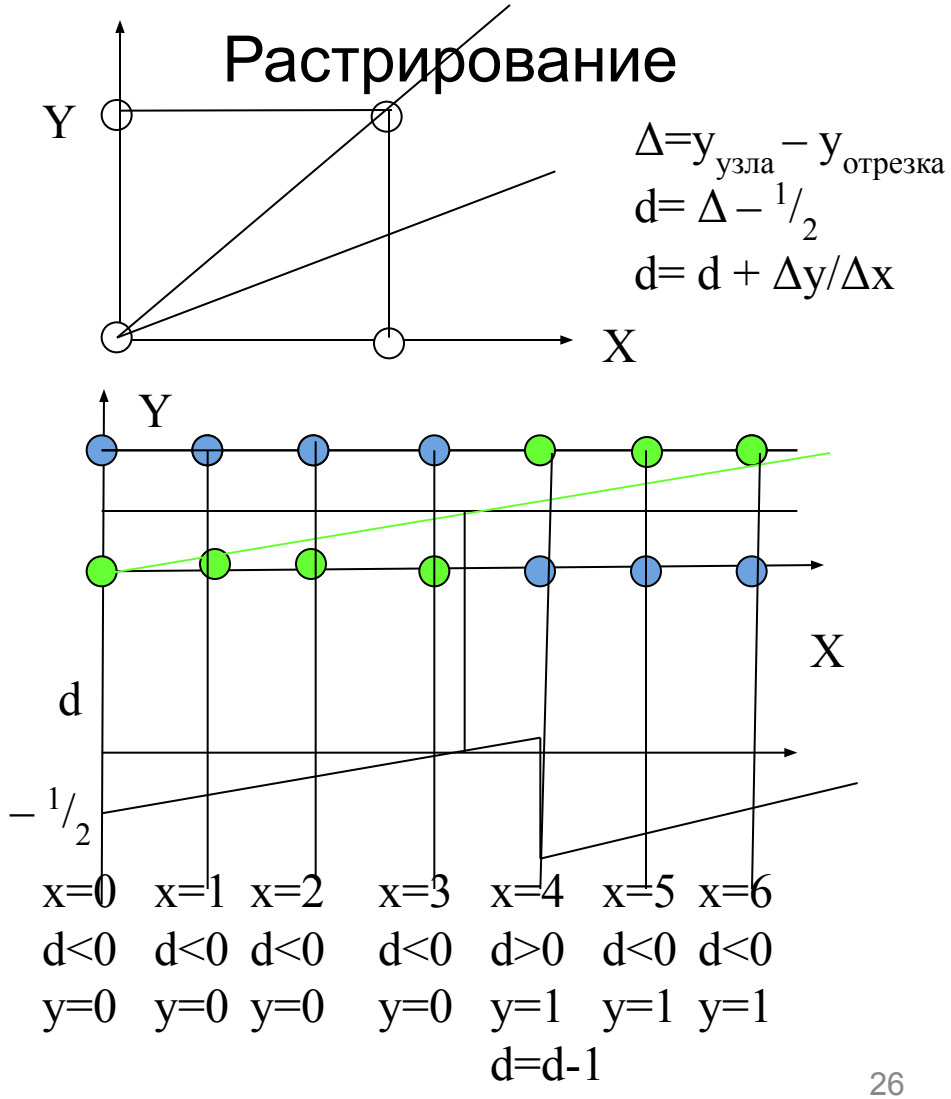
$$k = \frac{(y_2 - y_1)}{(x_2 - x_1)};$$

$$b = y_1 - k \cdot x_1.$$

Характеристика «прямых» алгоритмов

- Работа с вещественными числами.
- Наличие сложных операций (деление, умножение, тригонометрические функции).
- Возможно накопление ошибки.
- Проблема выбора шага приращения.

Алгоритм Брезенхема



Алгоритм обладает следующими достоинствами:

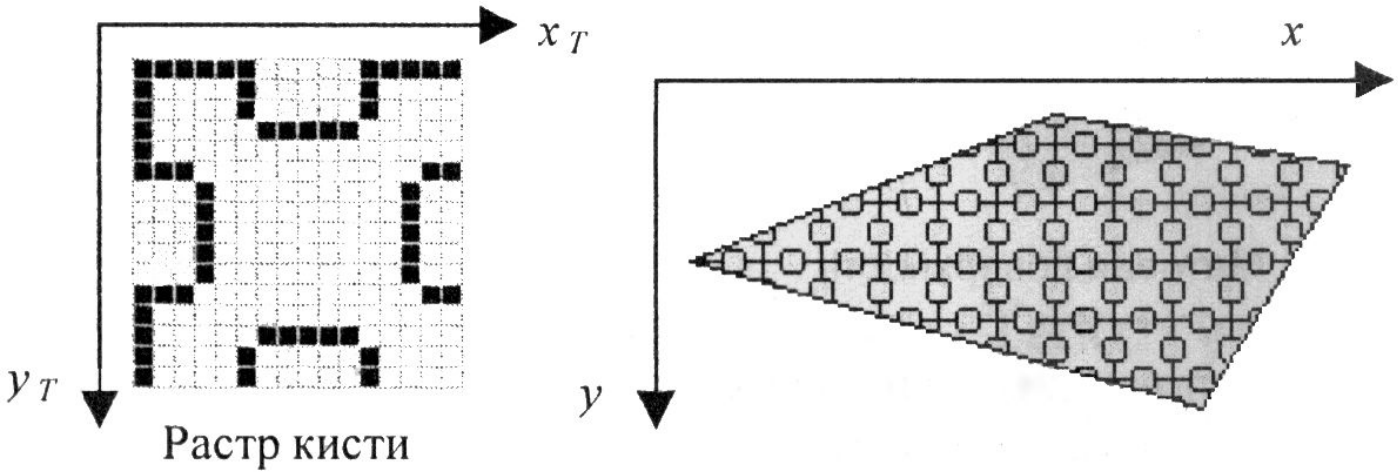
- **целочисленная арифметика;**
- **простые операции (сложение сдвиг);**
- **малый объем вычислений;**
- **отсутствие накопления ошибки**

Алгоритм Брезенхема

```
 Bresenham(x1,y1,x2,y2,Color: integer);
Var  dx,dy,incr1,incr2,d,x,y,xend: integer;
begin
   dx:= ABS(x2-x1);  dy:= Abs(y2-y1);
  d:=2*dy-dx; {начальное значение для d}
incr1:=2*dy; {приращение для d<0}
incr2:=2*(dy-dx); {приращение для d>=0}
if x1>x2 then {начинаем с точки с
меньшим знач. x}
    begin
       x:=x2; y:=y2; xend:=x1;
    end
 else
    begin
        x:=x1; y:=y1; xend:=x2;
    end;
```

```
PutPixel(x,y,Color); {первая точка
отрезка}
While x<xend do
  begin
    x:=x+1;
    if d<0 then
    d:=d+incr1 {выбираем нижнюю точку}
  else
    begin
      y:=y+1;
      d:=d+incr2; {выбираем верхнюю
            точку, y-возрастает}
     end;
  PutPixel(x,y,Color);
  end;{while}
end;{procedure}
```

Текстура



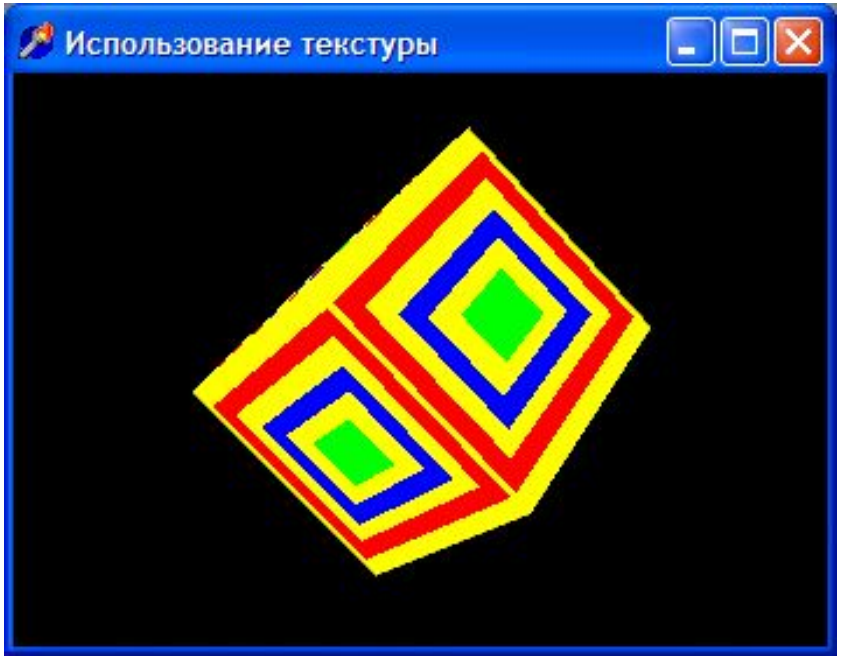
$$x_T = x \cdot \mathrm{mod}\, m, \qquad x_T = 0....m-1,$$

$$y_T = y \cdot \mathrm{mod}\, n, \qquad y_T = 0...n-1$$

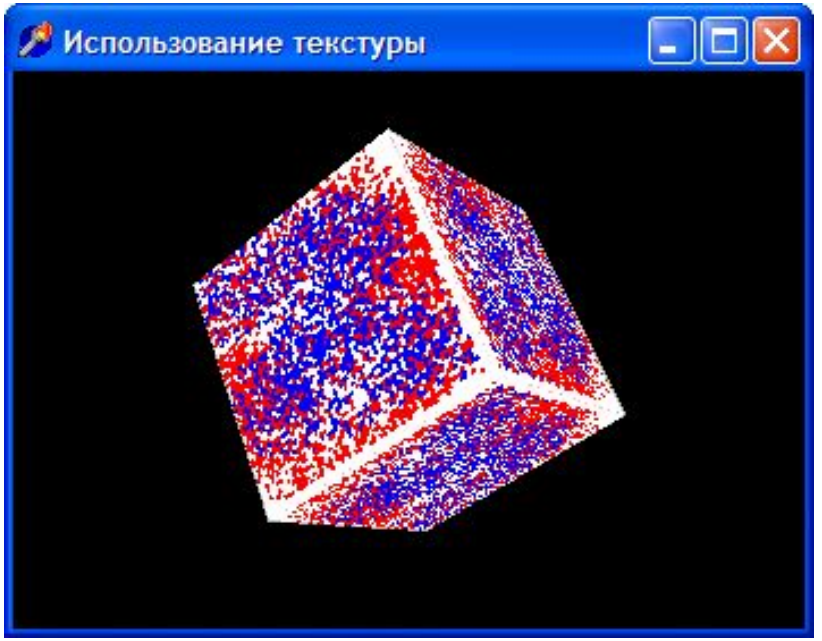
***m, n* — размеры растра кисти по горизонтали и вертикали**

Текстура

1. Упорядоченная



2.Стохастическая



Спасибо за внимание!