Examination synopsis

Theoretical questions

1. CONTOURS. Discrete plane geometry. Building of area contour.

2. DRAWING OF SEGMENT. The algorithm of segment creation with digital differential analyzer. The algorithm of Brezenhaim for drawing of segment. Algorithm of Brezenhaim for first quadrant. GENERAL Integer Algorithm of Brezrnhaim for all quadrants.

3. CURVE DRAWING. Algorithm for drawing of circle using regular circle equation. The algorithm of pieces. The algorithm with approximation of pixels. The algorithm which uses symmetry.

4. CUTING TROUTH OF CONTOUR. Erasing and cutting. The algorithm for cutting

5. Generation of fill areas. The raster unwinding of polygons The Algorithm with ordered list of sides. The Algorithm chain filling.

6. TWO- DIMENSIONAL DRAWING. Graphical presentation of experimental data in defined graphical windows

7. THE ECONOMIC GRAPHICS, Histograms and circle diagrams

8. INTERPOLATION AND APPROXIMATION algorithms Classical approach for interpolation and approximation. Polynomial of Lagrang.

9. SPLINE curve. B – spline.

10. CURVE OF BEZIE. The geometric algorithm for building of curve of Bezie. The algorithm of Horner.

11. TWO- DIMENSIONAL TRANSFORMATIONS. Composition.

Program tasks

1. Graphical presentation of experimental data in defined graphical windows

For example: The program have to obtain graphical presentation function: $y=x2+5\sin x$ for $-2 \le x \le 2$ in gr. window. Gr. Window (x0,y0)=(50,400), Rx=300, Py=250

7. THE ECONOMIC GRAPHICS, Histograms and circle diagrams

For example: Input data: 3 real data which correspond with financial result of one firm.
The program has to obtain graphical presentation of financial result a firm for last 3 months like vertical histograms in gr. window: (x0,y0)=(30,420), Ds=30, Dc=40, Py=300.

3. TWO DIMENSIONAL TRANSFORMATIONS. Composition.

For example: Input data: coordinates of 17 points of polygon. The program has to obtain the transformation polygon after next transformations:
- rotation around third point with angle 120°;
- scaling with Sx=2.4 Sy=5 .8 and point of scaling (-120, 160)

INTRODUCTION

The main task of Computer graphics methods is to present numerical results like images. In fact, people understand image information faster then text information. So a lot of systems necessarily include graphical presentation of their results in last 10 years.

There are 2 sciences which occupy whit computer images. They decide two opposed to each other tasks. First task receives images from text and numerical data and second task receives text descriptions of data which discovery in images. First task decides science **Computer Graphics** and second task decides science **Image Recognition**. In our object, we'll pay our attention to main algorithms of Computer Graphics.

In this book are described the main principles of building of computer images like composition two dimension and three dimension objects. The described algorithms treat creation and transformation image objects like real objects.

1. MAIN CONCEPTION IN COMPUTER GRAPHICS (CG)

CG integrates a lot of algorithms for processing of digital information with a purpose to obtain an image according the needs of users.

The main applications in CG are:

- Visualization of scientific experiment results;
- Geometrical design and modeling;
- Training machine;
- Interactive processing before production;
- Art application like films, photos, pictures and advertisement.

1.1. Coordinate systems which are used in computer graphics

For the presentation of a graphical object, we must use some coordinate system. There are two main mode of coordinate system.

Two dimensional systems present 2-D graphical objects in one plain.

In CG the following coordinate systems CS: World CS, Graphical CS are used. The real coordinates from WCS are brought on the coordinates of the graphical devices – screen. In this way it's obtained GCS. This transition is made automatically from program instruments. There is also a CS that is called Normal CS. It has only logical character and offers the graphical information that doesn't dependence on devices. That means that all coordinates are in the scope [0; 1]. The beginning of CS is on the left bottom corner.

Three dimensional systems present 3-D graphical objects in one plain.

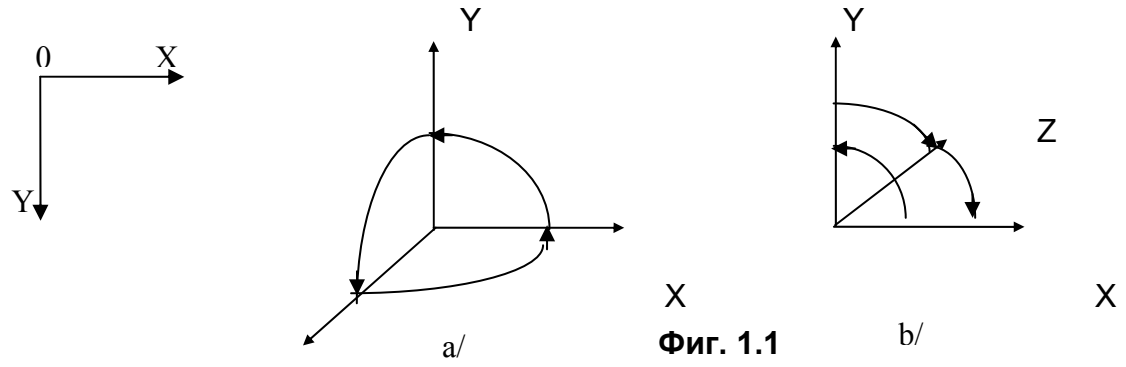
The representation of image objects from 3-D World space on 2-D picture plain is realized by including of several agreements. It' agreed that 3-D objects are projected on the 2-D visible plain by one of these two sorts' projections: **parallel** and **central** projection**.**

- There is the **parallel projection**, if the objects transfer parallel on 2-D visible plain, which is situated parallel of plain x0y from 3-D CS. The main ray which gives the direction of projection is obtained from position of the observer and position of the object.
- There is a **central projection**, if the objects transfer on 2-D visible plain like an observer look at the object from definite place. This place is called center of projection. There the rays of looking pierce the visible plain it's obtained the central projection. For example, Fig. 1.1 shows:

a/ **Right oriented CS,** whose axis transform from one to other by 90^o rotation to the opposite direction of switch;

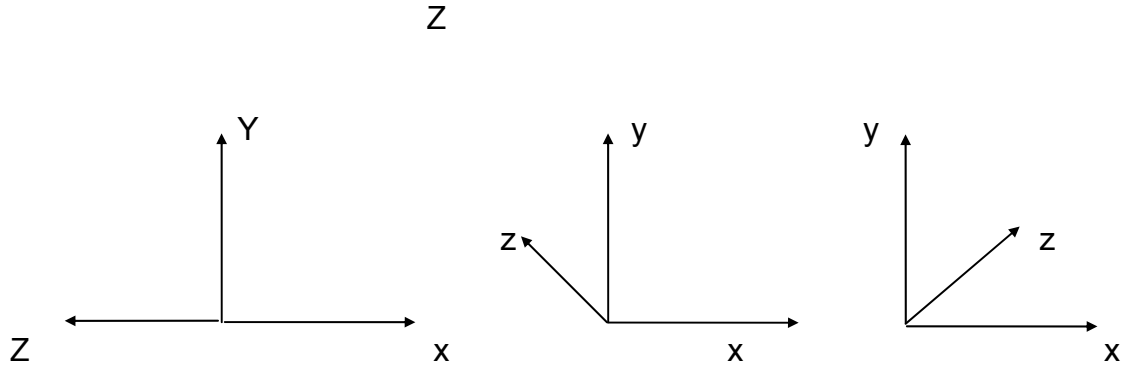
b/ **Left oriented CS**, whose axis transform from one to other by 90^o rotation to the direction of switch;

On the Fig. 1.2 there are some more variants of axis location.



Фиг. 1.1

2-D CS



$X_P>0, Y_P>0, Z_P=0$ $X_P>0, Y_P>0, Z_P<0$ $X_P<0, Y_P>0, Z_P<0$

Fig. 1.2

1.2. Methods of inserting of graphical information

The graphical information is inserted by two devices: a tablet and a scanner.

With first device, the operator inserts by hands a point by point that can be connected or not with lines. The tablets created a vector images. As a result of inserting by tablet it is obtained a vector image description. The tablets don't give a opportunity to insert an image which have a too many points.

The scanners insert graphical information with a pixel by pixel. As a result by inserting with scanner it is obtained a matrix description of the screen raster.

1.3. Methods of outputting of graphical information

The graphical information can be outputted by display or using paper devices.

Display devices are screen.

Every image on the screen are called **raster image** and contains different shining pixels. They receive different colour in different way: vector or raster
The display graphical output devices are vector and raster devices.

The vector devices work in the way of creating of point, segment and polygon by using their coordinates on the screen. The main elements (point, segment and polygon) are called graphical primitives.

The raster devices use all primitives (Pixels) in the definite way.

Using paper devices are plotter and printers.

The plotters are used to output information like a plan or sketch. They are vector devices working in the way of creating of point, segment and polygon by using pen marks of moving of plotter's pen. The plotters use pens with different colors. They are two sorts: tablet and roll. They are necessary for technical documentations in electronics, building, architecture and engineering.

The printers also can to be used for outputting of graphical information. They are raster devices because they present all points of image.
As a conclusion we can summarize that CG is a science for synthesis, analysis and transformation of images. The main differences between ways of creating of images are in the level getting of points and segments.

In our future work we'll discus the obtaining of **display raster image.**

1.4. Graphical files and formats

The images are saved in numerical mode in one end more files, numerical set or Data Bases. The graphical format is called the structure of data saving that describes the image in this file. For digital saving of images it's used **raster** and **vector formats**.

The raster formats describes images by numerical values of separate pixels. This format is used in environment Windows e DIBs (Device Independent Bitmaps), which is realized in BMP (by bit) files or they are included in format RIFF (Resource Interchange File Format) or often used TIFF (Target Interchange File Format). Usually the raster form includes an image header and image matrix. The header describes the main parameters like format, bit number for every pixel and compression matrix. The other famous formats are:: GIF (Computer Server GIF), PCX, RLC, EPS, BIL, PICT, Landsat, BIP, JPEG, JPG, JPE, BSQ, BMP,RLE, DIP, SPOT, WMF, WBM, Sun Raster, ERDAS IMAGINE end etc.

The vector formats of physical presentation of images are used of mathematical image description. The vector formats describe images by contiguous objects description using base (control) points of these objects.

The graphical objects are presented by vector including primitives with its attributes. These are: point, line, segment, circle, arc, polygon and etc. This formats the size of memory is smaller than raster formats but it's necessary additional time for computing of image transformation. This formats are used by systems CAD (Computer Aided Design), GKS(Graphical Kernel System), PHIGS (Programmer's Hierarchical Interactive System). The famous formats are PAL(Microsoft Palette), DXF и DWG (AutoCad), HGL(HP Graphic Language), DGN(MicroStation), VPF, MIF (MapInfo, Arc/Info) и Microsoft SYLK. The metafiles formats allow to keep in a file vector and raster fdata. For example we can give the formats of files of program CorelDRAW- CDR.

1.5. Image Compression

The compression is a process of decreasing of physical memory size which is need for data saving. The image files often have big size and they need from compression. The compression is logical and physical. The compression methods are separated according loss of information, sort of input data, mode of image, and mode of coddling.

There are standards of image compression according (ISO) for statical and dynamical images. JBIG, JPEG, Px64, MPEG -1, 2, 4 [7]. The using of these standards provides an opportunity for direct, fast accesses, fast founding, edition and hardware realization.

1.6. Color models

For a creation of color graphical image in discreet raster space it' used color description. In this case it's used different mathematical systems that are called **Colour Models**. These models include 1, 2, 3 or 4-measure space which components are intensity values of colour representative pixels. There are connections between different color models. The famous colour models:

Gray Scale Model – Fig. 1.3

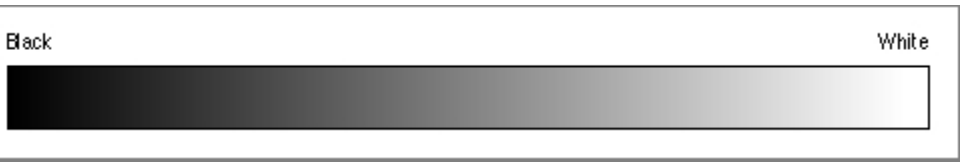


Fig.1.3

This is space that only one component changes from black to white. Usually this space has only one component and it's used on monochrome monitors and printers.

Модел RGB (Red-Green-Blue)

Colour Space RGB is 3-D. It's components are three main colours: - red, green, blue. The diagonal between beginning of coordinate system and white point presents of gray colour in all its blends (Fig.1. 4).

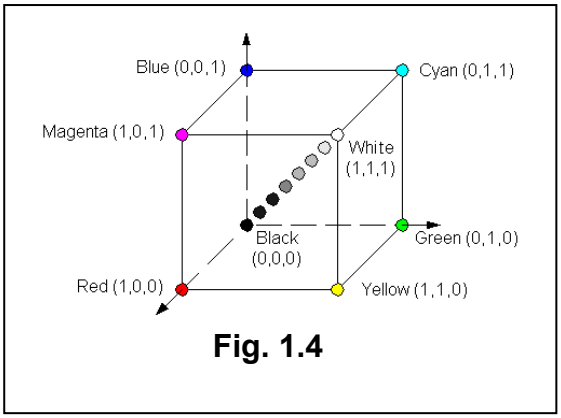


Fig. 1.4

RGB is a full color model**.** The other full colour models can be obtained from it. The transform from RGB to CMY model are applied using printers.

Модел CMY (Cyan, Magenta, Yellow)

This group includes two colour models:

- **CMY**;
- **CMYK**.

The main colours in CMYK are cyan, magenta, yellow and black. The red, green and blue are secondary.(Fig.1.4.).

The connection between RGB and CMYK realizes in this way:

Cyan = 1.0 – Red, *Magenta* = 1.0 – Green, *Yellow* = 1.0 Blue

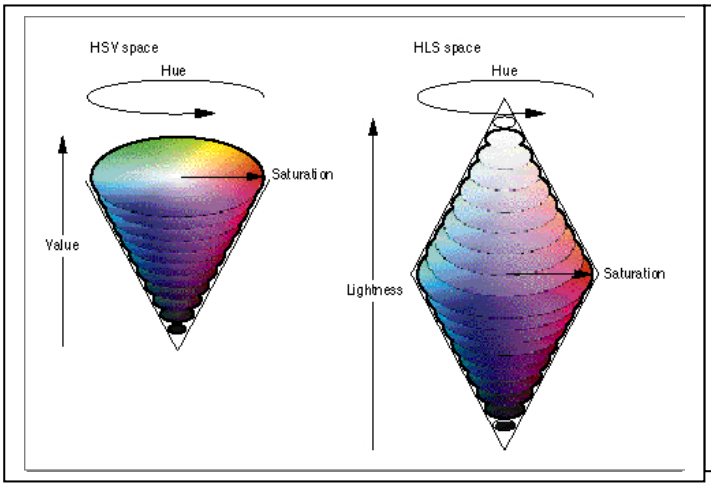


Fig. 1. 5

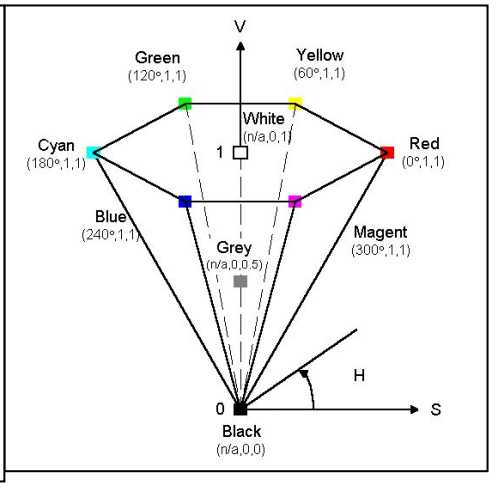


Fig. 1.6

Модели HSV, HLS, HIS

HSV (Hue, Saturation, Value), HLS (Hue, Luminance, Saturation) HIS (Hue, Saturation, Intensity) In HSV, intensity is maximal value 0 when the illuminator is 0 and 1 when the illuminator is 1. In HLS intensity has higher value in plumier 0.5 and value 0 for white and black. In fact HLS is identical with HSV but with different name. (Fig.1.6).

Other models:

Model YIQ, YUV, YcbCr, YCC , XYZ, Yxy, L*u*v* , L*a*b*

Model Munsell

This system has 3 attributes: a colour, a value, a force. This attributes are noted with symbols H,V and C .

Colour (Hue) is attribute which separates one color from another (red from green, blue from yellow etc.). The natural colour order is on Fig. 1. 7.

The attribute Value defines the colour light from 0 (for black) to 10 (for wait).

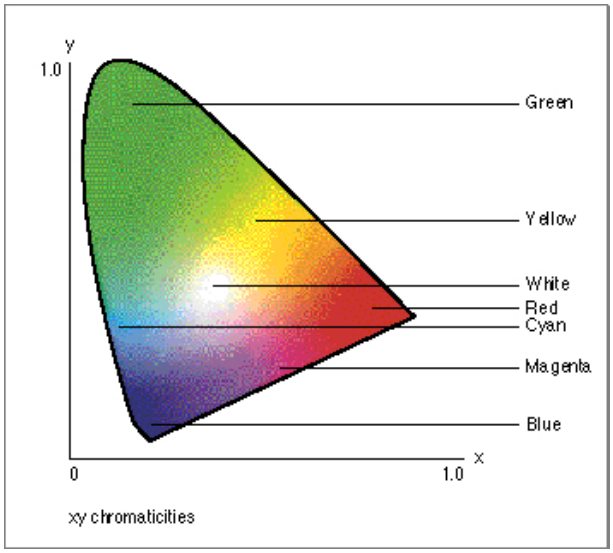


Fig.1.7

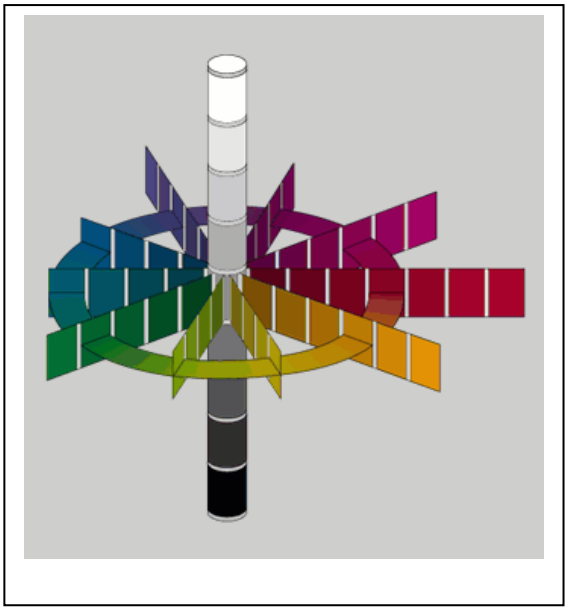


Fig.1.8

The scale of colour force is on Fig. 1.8.

2 GRAPHICAL LIBRARY IN ENVIRONMENT OF BORLAND C++

The program creation of algorithm CG is based on the graphical library of the program language. In C++ this library is in the header file **graphics.h**. In a program with graphical function it has to include this file with other needed files. In creation of main program (**main())** it's to include the header **graphics.h. An example is presented of Example 2.2 It** demonstrates the way of including of graphical application.

Example 2.1

The program text shows:

- loading of graphical driver and mode with maximal windows sizes;
- stopping of picture for looking;
- closing of graphical mode after user's conformation.

```
#include<iostream.h>
#include<graphics.h> // Header file with graphical library
#include<conio.h>
#include<math.h>
#include<dos.h>
void main()
{
int gdriver = 0,gmode;
initgraph(&gdriver,&gmode,"C:\\borlandc\\bgi");
……
……
getche();          //
closegraph();
```

2.1. Graphical Library

There are some groups in graphical library in the file **graphics.h**. The main functions are for:

- ✓ initialization of graphical mode;
- ✓ getting and setting of parameters of graphical presentation;
- ✓ outputting text string in graphical mode;
- ✓ drawing of graphical primitives 2-D and 3-D objects;
- ✓ Saving and restoring of image.

Below the most used functions are looked with description of their names and applications. In this description are used next notes:

color – an integer constant or variable in scope [0;15] which corresponds with number of colour from colour pallet.

pattern – an integer constant or variable in scope [0;11] which corresponds with number of style of pattern;

linestyle – an integer constant or variable in scope [0;3] which corresponds with style of line.

x,y – a couple integer constants or variables which corresponds with coordinates of one point.

Dx,Dy – a couple integer constants or variables which corresponds with coordinates of point moving;

thick={1,3} an integer constant or variable which corresponds with line thick;

page={0,1} an integer constant or variable which corresponds with page number;

font ={0,4} an integer constant or variable which corresponds with text style;l

direction={0,1} an integer constant or variable which corresponds with direction as horizontal or vertical ;

charsize – an integer constant or variable which corresponds with symbol size;

horiz={0,1,2} – an integer constant or variable which corresponds with the way of text horizontal positioning;
vert={0,1,2}– an integer constant or variable which corresponds with the way of text vertical positioning;
string – a constant or variable for string;
num – an integer constant or variable for points number;
arrxy – an array containing coordinates of points;
sangle – an integer constant or variable for start angle;
endangle – an integer constant or variable for end angle;
R – an integer constant or variable for circle radius;
Rx, Ry – an integer constant or variable for ellipse radiuses;

2.2 Initializing of Graphical Mode

initgraph(GraphDriver, GraphMode, "\path\...\...\BGI"**) –** load the graphical driver and sets mode about it. The parameters (GraphDriver, GraphMode) are noted with integer variables which set number of driver and it's mode. If it's used (GraphDriver = DETECT) the setting of driver and is made automatically.
closegrph() – Returns previous mode
graphresult() – Returns as variable code **of operation (initgraph) result.**
grapherrormsg (ErrorCode) - Returns as text of operation code **(ErrorCode) .**

Example 2.2 The program text for :

- loading graphical driver and mode;
- checking of result from this and outputting text massage if the result is different from 0. (the code is 0 if there is no error).

```
void Initialize(void)
{
  int  GraphDriver, GraphMode;
  GraphDriver = DETECT;    /* Automatically setting*/
  initgraph( &GraphDriver, &GraphMode, "" );
  ErrorCode = graphresult();  /* Reading of result from
initialization*/
  if( ErrorCode != grOk ){   /* Checking of error */
    printf(" Graphics System Error: %s\n", grapherrormsg(
ErrorCode ) );
    exit( 1 );
  }
```

2.3. Setting and getting of graphical parameters

setcolor(color) – sets the line colour with a parameter (color).

setbkcolor(color) – sets the background colour with a parameter (color).

setlinestyle(linestyle, pattern, thick) – sets line style (linestyle), pattern style (pattern), и line depth (thick – 1 or 3 pixels).

setfillstyle(pattern, color) – sets pattern style (pattern) and pattern colour (color).

setactivepage(page) – sets an active page.

setvisualpage(page) – sets a view page.

setviewport(x_1,y_1,x_2,y_2) – set a view window with coordinates of top left corner (x_1,y_1) and bottom right corner (x_2,y_2).

getcolor() – returns integer number which corresponds with last setting of line colour.

getbkcolor() – returns integer number which corresponds with last setting of background colour.

getx() – returns integer number which corresponds X axes of graph cursor.

gety() – returns integer number which corresponds X axes of graph cursor.
getmaxx() – returns integer number which corresponds maximal value of X axes.
getmaxy() – returns integer number which corresponds maximal value of Y axes. **getpixel(x,y)** – returns integer number which is colour number of point with coordinates (x,y).

2.4 Work with text in graph mode

settextstyle(font, direction, charsize) – sets text style (font), direction of location and symbol size (charsize)
settextjustify(horiz, vert) – sets justifying of text on horizontal and vertical direction.
outtext(string) – outputs text (string) from current position of graphical cursor. .
outtextxy(x,y,string) – outputs text (string) from point (x,y).
textheight(string) – returns integer number which correspond of text height in pixels
textwidth(string) – returns integer number which correspond of text dept in pixels.

2.5 Main Graphical primitives

putpixel(x, y, color) – draws pixel with coordinates and colour (color).
moveto(x,y) – moves graph cursor to a point (x,y).
moverel(Dx, Dy) – moves graph cursor with distance (Dx,Dy).
line(x_1,y_1,x_2,y_2) – draws line between two points (x_1,y_1) and (x_2,y_2).
lineto(x,y) – draws line between from current position of graph cursor to a point (x,y).

linerel(Dx,Dy) – draws line between from current position of graph cursor to a point which has distance (Dx,Dy).

drawpoly(num, arrxy) – draws polygon which has number (num) of points and which coordinates are given in parameter (arrxy).

filllpoly(num, arrxy) – draws closed polygon which has number (num) of points and which coordinates are given in parameter (arrxy).

floodfill(x, y, border) – fills a full area with current style of pattern beginning from point (x, y) to the border of closed contour which colour is set with parameter (**border**).

rectangle(x_1,y_1,x_2,y_2) – draws rectangle with top left corner (x_1,y_1) and bottom right corner (x_2,y_2).

bar(x_1,y_1,x_2,y_2) – draws bar with top left corner (x_1,y_1) and bottom right corner (x_2,y_2).

bar3d(x_1,y_1,x_2,y_2, d, {topon/ topoff} – draws prism with top left corner (x_1,y_1) and bottom right corner (x_2,y_2), dept d and top or bottom. .

circle(x,y, R) – draws circle with Centrum (x,y) and radius R.

arc(x,y, sangle, endangle, R) – draws circle arc with Centrum (x,y) and radius R, Start Angle (sangle) and end angle (endangle).

pieslice(x,y, sangle, endangle, R) – draws fill circle sector with Centrum (x,y) and radius R, Start Angle (sangle) and end angle (endangle).

ellipse(x,y, sangle, endangle, Rx, Ry) – draws fill ellipse circle ellipse with Centrum (x,y) and radiuses Rx, Ry., Start Angle (sangle) and end angle (endangle).

fillelipse(x,y, Rx, Ry) – draws fill circle ellipse with Centrum (x,y) and radiuses Rx, Ry., Start Angle (sangle) and end angle (endangle).

sector(x,y, sangle, endangle, Rx, Ry) – draws fill ellipse sector with Centrum (x,y) and radiuses Rx, Ry. R, Start Angle (sangle) and end angle (endangle).

Example 2.3 The main function that present a picture of fish using different graphical functions.

```
#include<iostream.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>
#include<dos.h>
void main()
{
int gdriver = 0,gmode;
initgraph(&gdriver,&gmode,"C:\\borlandc\\bgi");
int x0,y0,r,i=300,yt;
setbkcolor(1);
setfillstyle(1,YELLOW);
x0=500;y0=300;r=70;
while(i)
  {
  setcolor(YELLOW); //SET yellow line color
  setfillstyle(1,YELLOW); //sets yellow color of filling
  ellipse(x0,y0,230,310,r,r);//draw fish
  ellipse(x0,y0+107,50,130,r,r);
  floodfill(x0,y0+53,YELLOW); //Filling of fish body
  line(x0+45,y0+53,x0+65,y0+33) ;  //Tell contour
  line(x0+45,y0+53,x0+65,y0+73);
  line(x0+65,y0+33,x0+65,y0+73);
  floodfill(x0+50,y0+53,YELLOW); //Filling of fish tell
    delay(10);
getche();
closegraph();
  }
```

3 COORDINATE SYSTEMS

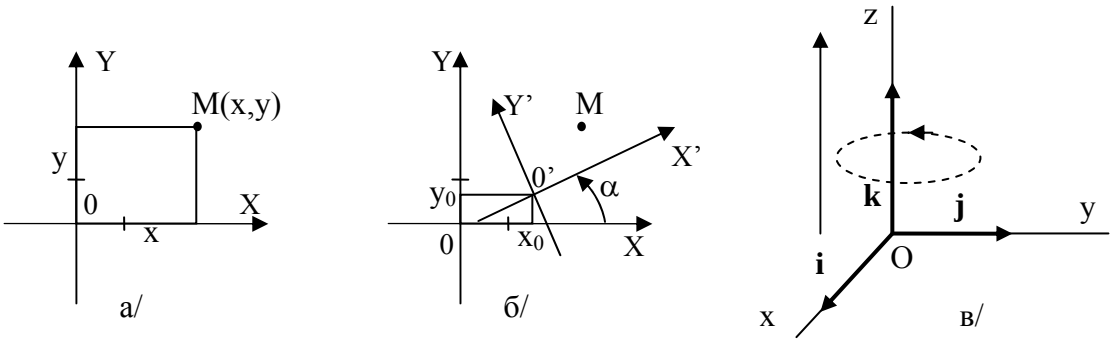


Fig. 3.1 Coordinate systems, mathematical elements and specialities

a/ 2-dimensions CS; b/ Transformations of 2D C S ; c/ Right –oriented 3D CS

2-D Decart CS in the plane is axis cross which obtain from crossing of two perpendicular line. Each on of points in this plane is defined from two numbers (x,y), which is called coordinates. They correspond of algebra measures of projections radius vector of the point in the axes x and y. For example of Right –oriented CS and presentation of point M can be seen on the Fig. 5.2. a/.

3-D CS is begined to give with 3 perpendicular singly vectors. 3-D CS is called Right –oriented (Fig. 5.2 c/), if a vector **i rotates to vector j** by 90° the direction of vector k concurs with movement of a right screw. The beginning of CS is noted with O. Every vector V can be writhed in a kind of linear combination of i, j,k in this way:

V=xi + yj + zk, where x, y, z are coordinates of the point P of vector V. The vector can be writhed in matrix form with equation (3.1).

$$V = [x, y, z] \text{ или } \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad (3.1)$$

HOMOGENOUS COORDINATES

So to escape from the depending of coordinates from size of image it's used homogeneous coordinates. They describe every point with three coordinates (x, y, w), where w is one of measuring of coordinates and it's called scale

coefficient. It's used for inward changing of the scale. The equation aX + bY +c = 0, that describes strait line in 2-d space. If it's changed the point (X, Y) with (x/w, y/w) will be as (5.4):

$$ax + by + ch = 0 \qquad (3.2)$$

This equation is named homogenous and (x, y, w) is called homogenous coordinates of point (X, Y). If w=1 the equation (5.4) describes a plane across the coordinates beginning and set line where $w \neq 0$.

The using of homogenous coordinates allows to present graphical transformations using only matrix's multiplication. If the graphical transformation is presented by [x',y'] = [x y] A, where A is a transformation matrix $A = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix}$

In homogenous coordinates I 2-D space this can be written by:

[x' y' z'] = [x y z] A, where : $A \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix}$

A line between two points $P_1(x_1, y_1, w_1)$, $P_2(x_2, y_2, w_2)$, in homogenous coordinates is defined with the equation (5.8).

$$\det \begin{bmatrix} x & x_1 & x_2 \\ y & y_1 & y_2 \\ w & w_1 & w_2 \end{bmatrix} = 0 \text{ или } x(y_1w_2 - y_2w_1) + y(x_2w_1 - x_1w_2) + w(x_1y_2 - x_2y_1) \qquad (3.3)$$

The point coordinates which are obtained from crossing of two line $L_1(a_1,b_1,c_1)$ and $L_2(a_2,b_2,c_2)$ are set by the members in equation brackets. (5.9).

$a\,(b_1c_2 - c_1b_2) + b\,(c_1a_2 - a_1c_2) + c\,(a_1b_2 - b_1a_2) = 0$ And he

PART II TWO – DIMENSIONAL GRAPHICS

4 CONTOURS

The discrete images use these main notions: contour; closed contour, area and shape of image object.

4.1 Discrete plane geometry

Two pixels are immediate neighbors if they have a common side and indirect neighbors if they have only a common corner. The term N- neighbors are illustrated on Fig 6.1. using even and odd number.

3	2	1
4	PIXEL	0
5	6	7

Fig . 6.1

K-route (contour) is called sequence from pixels a A_1, A_1, …,A_n, that $A_{k-1}$and A_{k+1} are neighbors of A_k. A simple route is this whose pixels are different and closed route is whose first and last pixel are the same.

4.2 Building of area contour

A contour or K-connected multitude from pixels R is named multitude from all pixels which have one or more n-neighbors situated out of R. The traveling over the contour pixels can be realized using a closed route. We present an algorithm in terms of observer who is moving on the pixels choosing always the last right pixel. The initial pixel A can be defined with the method – “Going from top to bottom and from left to right”. The contour building finishes when the last browsing pixel appears initial. For discerning initial pixel from this that we gain after contour building we bring the flag F. The loop from steps 5-8 runs no more than 3 times so to escape from case when the multitude has only 1 pixel. As a result from a multitude of connected points R describing some

area by the algorithm is obtained ouside single contour like sub-multitude of points C_R.

Algorithm for building of contour

Symbols:

A-beginning point of contour set with multitude R;

C – current point;

S- direction for searching presented with codes from Fig6.1.

F - flag with value „true" only in the beginning of building of contour.

D– flag whose value become "true" if it's found the current point of the contour.

Steps:

0. From contour it's chosen the point A, which has less than four neighbors.
1. C=A (Value of point A is assigned to point C), S=6 (direction according Fig.6.1. down) and F="true".
2. **While** (C ≠ A or F="true") **Do** steps from 3 to 9.
   begin
3. D="false".
4. **While** (D="false) **Do** steps from 5 to 8 but no more than 3 times
   begin
5. **if** point B is (S-1)-neighbor of point C and belongs to the multitude R **then**
   C=B, S=S-2, D="true".
6. **Else if** the point B is S-neighbor of point C and belongs to the multitude R **then** C=B, D="true".
7. **Else if** т. B e (S+1)- neighbor of point C and belongs to the multitude R **then** C=B, D="true".
8. **Else** S=S+2 .

End.

9. F=”false”.

End.

If an area has more than one closed contour with this algorithm is obtained every one of contours of area.

DRAWING OF SIMPLE GRAPHICAL PRIMITIVES

5 . DRAWING OF SEGMENT

The display raster can describes like a matrix from pixels. Each one pixel can be lighted with some color. The defining of pixels that belong to line is called drawing of line in the raster. For horizontal, vertical and line with angle 45 degree (Fig. 7.1 a/) the choice of pixels is clean. For every one another orientation (Fig.7.1b) the choice of pixel is more difficult.

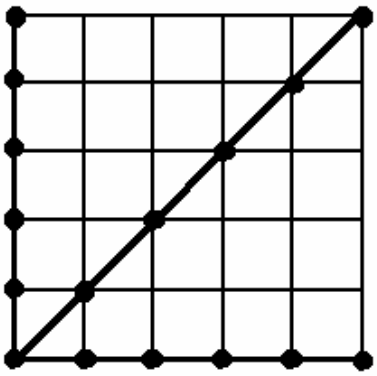


Fig. 5.1 а

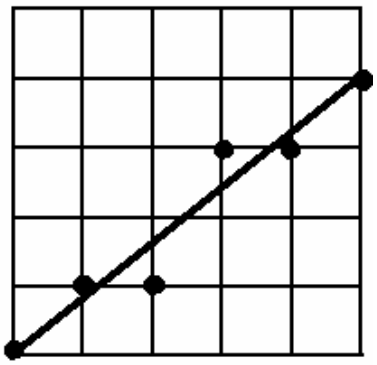


Fig. 5.1 б

The requirements to algorithms for segment drawing are:

- The segments must be look like line as begin and finish with the defined points;
- The briniest in all segment points must be constant ant independent from slope;
- The drawing speed must be maximal.

5.1. The algorithm of segment creation with digital differential analyzer

This method uses a differential equation whose decision is a line.

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

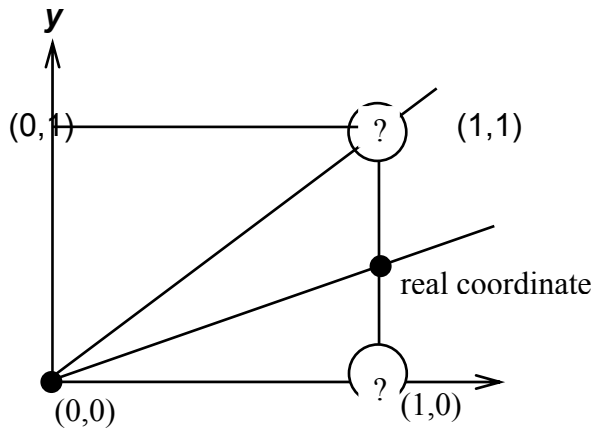
The decision is: $y_{i+1} = y_i + \Delta y$ $y_{i+1} = y_i + \frac{y_2 - y_1}{x_2 - x_1}\Delta x$

Where: (x_1,y_1) and (x_2,y_2) are start and end point of segment;

y_i - current coordinate of last lighted point. This method are named Digital differential analyzer .

5.2. The algorithm of Brezenhaim for drawing of segment

This algorithm is basic and very important in computer graphic. Brezenhaim offered to choose “optimal” coordinates of segment pixels. During the segment drawing one coordinate *(x or y)* changes with 1. That depends on angle coefficient of slope of segment. Another coordinate changes with *0 or 1 according to distance* between real coordinate and nearest coordinate of raster grid. This distance is named “raster error” end mark ε. The algorithm of Bresenhaim take a decision for every drawing pixel whether coordinate change with 1 or 0 using the control of raster error sign. The fig.2.2 illustrates a segment in first octant (angle coefficient of slope of segment $\in$[0,1] .



1/2 ≤ Δy/Δx ≤ 1 (raster error ≥ 0)

0≤ Δy/Δx < 1/2 (raster error <0)

Fig. 5.3

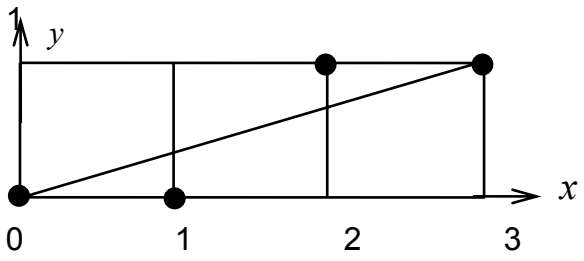
If the angle coefficient of slope of segment in point (0,0) is more than ½ the algorithm will choose pixel (1,1). If the angle coefficient of slope of segment in point (0,0) is less than ½ the algorithm will choose pixel (1,0). On the next figure 7.3 is presented case with the angle coefficient of slope of segment (m=1/4). The segment has 4 points. The raster error for points according algorithm is computed by formula $\varepsilon_i = \varepsilon_{i-1} + m$, where $\varepsilon_0 = -1/2$

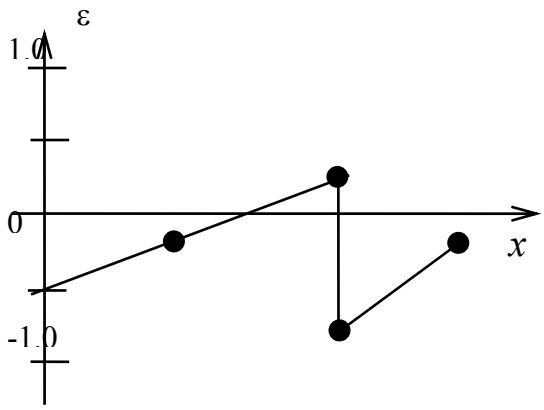
For example below: $\varepsilon_1 = -1/2+1/3=-1/6$ <1/2 -> y1=y0

$\varepsilon_2 = -1/6+1/3=1/6$ <1/2 -> y2=y1

$\varepsilon_3 = +1/6+1/3=3/6$ =1/2 -> y3=y2+1 -> $\varepsilon_3' = \varepsilon_3 - 1 = -1/2$

$\varepsilon_4 = -1/2+1/3=-1/6$ < ½ -> y4=y3





Algorithm of Brezenhaim* for first quadrants *(0 ≤ Δy ≤ Δx)

Symbols: x, y, Δx, Δy ***–integer , ε-real number;***

Initialization of variables:

$x = x_1$**;** $y = y_1$**;** $\Delta x = x_2 - x_1$**;** $\Delta y = y_2 - y_1$

$\varepsilon = \Delta y / \Delta x - 1/2$

// main loop

for i=1 to Δx

begin

Plot (x, y)

while *($\varepsilon \geq 0$)*

begin

y=y+1

$\varepsilon = \varepsilon - 1$

end while

x=x+1

$\varepsilon = \varepsilon + \Delta y \ / \ \Delta x$

next *i*

end for

finish

For full realization of algorithm of Brezenhaim it's necessary to process the segments from all quadrants.

<u>GENERAL Integer Algorithm of Brezrnhaim for all</u>* <u>quadrants</u>*.

Symbols: x, y, Δx, Δy ***– integer ,*** ε ***- real number;***

Initialization of variables:

$x = x_1$

$y = y_1$

$\Delta x = abs(x_2 - x_1)$

$\Delta y = abs(y_2 - y_1)$

$s_1 = Sign(x_2 - x_1)$

$s_2 = Sign(y_2 - y_1)$

if $\Delta y > \Delta x$ ***then***

Temp=Δx

$\Delta x = \Delta y$

$\Delta y = Temp$

Obmen=1

else

Obmen=0

endif

$\bar{\varepsilon} = 2 * \Delta y - \Delta x$

// The main loop

for *i=1 to* Δx

begin

Plot *(x, y)*

while *($\bar{\varepsilon} \geq 0$)*

begin

if *Obmen = 1* ***then***

$x = x + s_1$

else

$y = y + s_2$

endif

$\bar{\varepsilon} = \bar{\varepsilon} - 2 * \Delta x$

end while

if *Obmen=1* ***then***

$y = y + s_2$

else

$x = x + s_1$

endif

$\bar{\varepsilon} = \bar{\varepsilon} + 2 * \Delta y$

next i

end for

finish

6. DRAWING CURVE LINE. DRAWING OF CIRCLE

Кривите линии се строят с помощта на стандартни методи, а именно The main principle of drawing curves is using of line approximation between closed points. Indeed the raster line always is consisted from straight line but when they are more it create illusion for curve.

6.1. Algorithm for drawing of circle using regular circle equation.

This curve is the most used. It's basic component for drawing of pictures , circle diagrams and many another complicated images.

Input data for drawing of circle: center coordinates (x_c, y_c) and radius r.

The equation of circle can be writhed by some different ways.

Coordinates of circle points (x,y) can be defined depending on angle α that is measured in clock direction from x-axis.

$$x = x_C + r.cos(\alpha)$$
$$y = y_C + r.sin(\alpha)$$

6.2. The algorithm of pieces

Input data: x_C, y_C, r and number of approximation pieces n.

For α ***from*** *0* ***to*** *2*π* ***with step*** *2*π / n* ***do***

begin

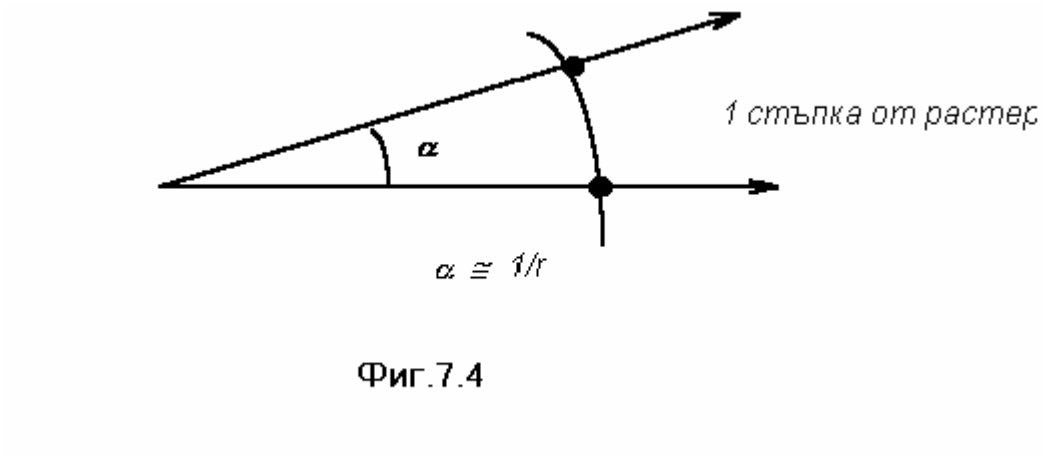
it's computed new circle point (*x* ***and*** *y)* ***using equation above.***

It's drew a segment from the last point to new circle point.

end

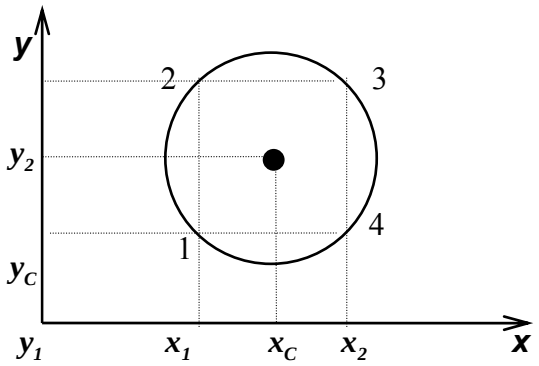
6.3. The algorithm with approximation of pixels

The speed of circle drawing can be raised as it lights pixels witout connected lines. The best result it obtains when the distance between neighbour pixels is 1 pixel. This optimal case it can be obtained if it's used step for angle around 1/r ($\Delta\alpha \cong 1/r$ **).**

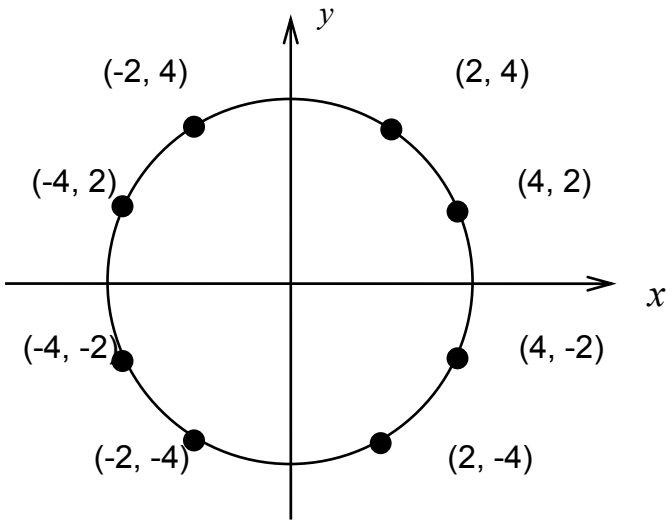


Фиг.7.4

When we draw the circle, we can use its features of symmetry. Its top half is symmetry of bottom half and left half is symmetry of right half. That is presented with dependences between its coordinates like:



If we know coordinates of one point (for example 1 on the figure above) we can easy compute the coordinate of his symmetry three points according circle center (2, 3 and 4 on the figure above). This way can be continued like such way for 8 , 16 e.t.c. For example, see the figure below.



The computing time can be decreased if it's ised equations witouts functions sin and cos. Every circle point is computed using the coordinates of previous circle point. If the point (x_1y_1) belongs to the circle the coordinates of next point (x_2y_2) can be computed with:

$$x_2 = x_C + (x_1 - x_C).ca + (y_1 - y_C).sa \qquad y_2 = y_C + (y_1 - y_C).ca - (x_1 - x_C).sa$$

where: *ca* and *sa* are constants, whose values depend on the angle step in this way: $ca=\cos(\Delta\alpha)$ $sa=\sin(\Delta\alpha)$

The start point has coordinates: $x_1 = x_C$, $y_1 = y_C + r$. The computations are stopped when $x_2 - x_C > y_2 - y_C$.

7 . CUTING TROUTH OF CONTOUR

Erasing and cutting

Sometimes it's required excluding a part of the current image. When it's used methods for separating of areas from image. Theses separated areas can be erased or cut. In first case there's erasing of image area. In second case there's cutting of image area

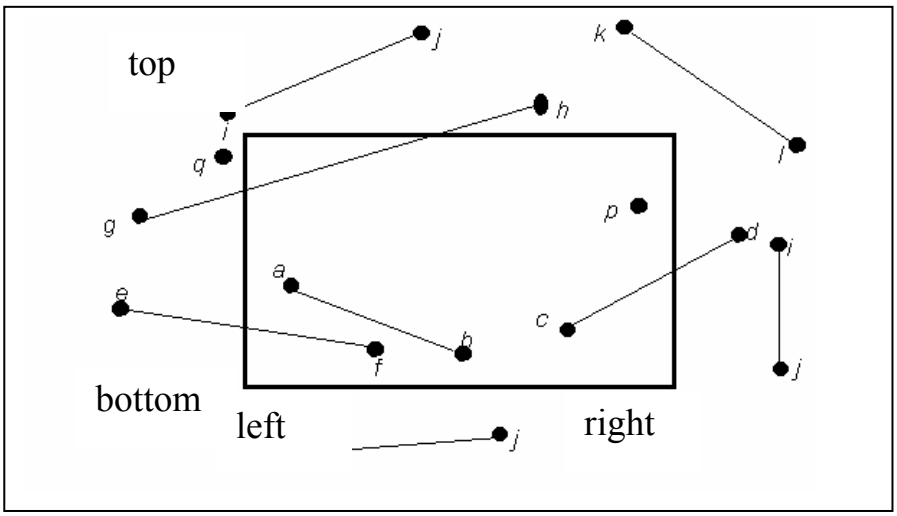
7.1. Erasing

Usually the separating methods use windows with shape of circle or rectangle.

The shape of rectangle is easiest to apply. The algorithm includes drawing segment into rectangle with color of background. By analogy in circle, it draws line (radiuses) insaide of circle.

7.2. CUTTING

We talk about cutting when some application requires keeping somepart of image and erasing another part. Usually for cutting is used window with shape of rectangle. This is named graphical window. The algorithm of cutting has to find all points and line parts which are in the graphical window.



On the figure it shows an window cutting some different situated segments. The rectangle has 4 faces (top, bottom, left right). These faces are parallel to coordinate axes. The purpose of algorithm is to find points and segment parts that are situated inside the window's rectangle. These founded points rest in the image and another will be erased.

There are three main cases of location of segments according window:

- Segments are fully on outside of the rectangle. For example on the figure above, those are segments ij, q. Their points answer the inequalities:

$$x \le x_{left} \text{ or } x \ge x_{right} \text{ or } y \le y_{bottom} \text{ or } y_{top} \le y$$

- Segments are fully on inside of the rectangle. For example on the figure above that is segment ab, p. Their points answer the inequalities. .

$$x_{left} \le x \le x_{right} \text{ and } y_{bottom} \le y \le y_{top}$$

- Segments are part inside or outside of the rectangle. For example on the figure, above segments gh, ef, cd. There's an example for segment kl , which is outside, but whose coordinates isn't answer the inequalities.

The algorithm defining visible an invisible segments.

a, b ***– start and end points of segment.***

st1. For every segment it's verified fully visibility.

(If one coordinates from these of point a and b it follows that the segment is part visible.)

if $x_a<x_{left}$ ***or*** $x_a>x_{right}$ ***then 1 endif***

if $x_b<x_{left}$ ***or*** $x_b<x_{right}$ ***then 1 endif***

if $y_a<_{bottom}$ ***or*** $y_a>y_{top}$ ***then 1 endif***

if $y_b<y_{bottom}$ ***or*** $y_b> y_{top}$ ***then 1 endif***

the segment is fully visible -> go to label3

st2. For every segment it's verified fully invisibility.

label 1: ***if*** $x_a<x_{left}$ ***and*** $x_b<x_{left}$ ***then 2 endif***

if $x_a>x_{right}$ ***and*** $x_b>x_{right}$ ***then 2 endif***

if $y_a<y_{bottom}$ ***and*** $y_b<y_{bottom}$ ***then 2 endif***

if $y_a>y_{top}$ ***and*** $y_b>y_{top}$ ***then 2 endif***

***st3.3* Case with part visible segment or suspect for this.**

The locate the point of crossing of segment with the window.

label 2: ***the segment is invisible***

label 3: ***go to the next point of segment.***

Test for full visibility and invisibility by method of Koen and Souserland.

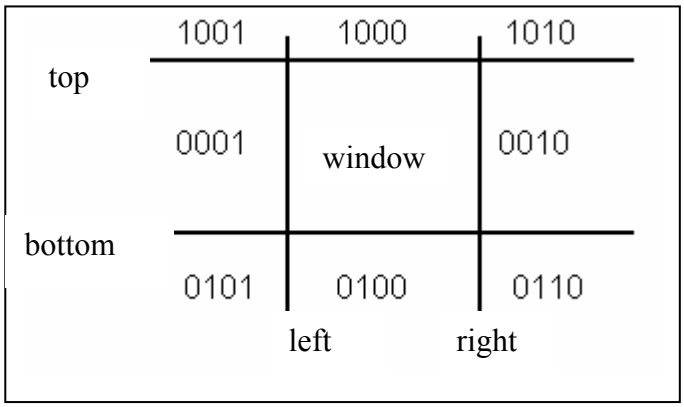
In this method, it's used 4-bits code. The codes for all possible area of location are on the figure 8.2. The right code bit if first. In every bit it's put 1 if the next conditions are executed: for first bit – if the point is on the left of window;

for second bit – if the point is on the right of window;

for third bit – if the point is on the bottom of window;

for forth bit – if the point is on the top of window;

Otherwise in bit it's put 0;



If codes for start and end points of segment are 0 it follows that two points lie into the window and the segment is visible. This codes can be used for refusing of invisible segments. If bit logical multiplication of two codes (of end segment points) isn't equal to 0 the segments is fully invisible. If bit logical multiplication of two codes is equal to 0, the segment can be fully or part visible. For full visibility it's necessary to verify individually codes of points. For part visible segments it's used sub-program that compute point of crossing of window and the segment. ***The computing of points of crossing can use the equal of crossing of two line.*** If the segment is from $P_1(x_1, y_1)$ to

$P_2(x_2, y_2)$. Let the slope of line is: $m = (y_2 - y_1)/(x_2 - x_1)$. The points of crossing with the sides of window rectangle and the segment have coordinates:

with left: $x_{left}, y = m(x_{left} - x_1) + y_1$ $m \neq \infty$
with right: $x_{right}, y = m(x_{right} - x_1) + y_1$ $m \neq \infty$
with top: $y_{top}, x = x_1 + (1/m)(y_{top} - y_1)$ $m \neq 0$
with bottom: $y_{bottom}, x = x_1 + (1/m)(y_{bottom} - y_1)$ $m \neq 0$

IF the slope is $m = \infty$ the segment is parallel to vertical sides of window and on the analogy if the slope is $m = 0 = \infty$ the segment is parallel to horizontal sides of window.

8. GENERATION OF FILL AREAS

The opportunity in raster image to present “thick, fill ” area is very used. For creation fill areas there are the methods in two modes: the **raster unwind** and **chain filling.** In first methods during the time of scanning there’s the verification whether the point into the area contour. The second method requires setting of one point into the area contour. The algorithm examines the neighbor pixels whether they are into area contour.

A lot of closed contours are simple polygons. The contours always are approximated with fitting polygons. The easiest method is consisted in verification for belonging of raster pixels of the inside of polygon. But in this case the computing time is maximal. For decreasing of computing time in generation of fill area there are next algorithms.

8.1. The raster unwinding of polygons

The color characteristics of raster pixels change only in the place where the sides of polygon cross raster rows. On fig. 9.1. the row 2 crosses the polygon in points with x=1 and x=8. It’s obtained 3 areas.

$x<1$ - out of polygon
$1 \le x \le 8$ - in polygon
$x>0$ - out of polygon

the row 4 divides polygon to 5 areas:

$x<1$ - out of polygon

$1 \le x \le 4$ - in polygon
$4 < x < 6$ - out of polygon
$6 \le x \le 8$ - in polygon
$x > 8$ - out of polygon

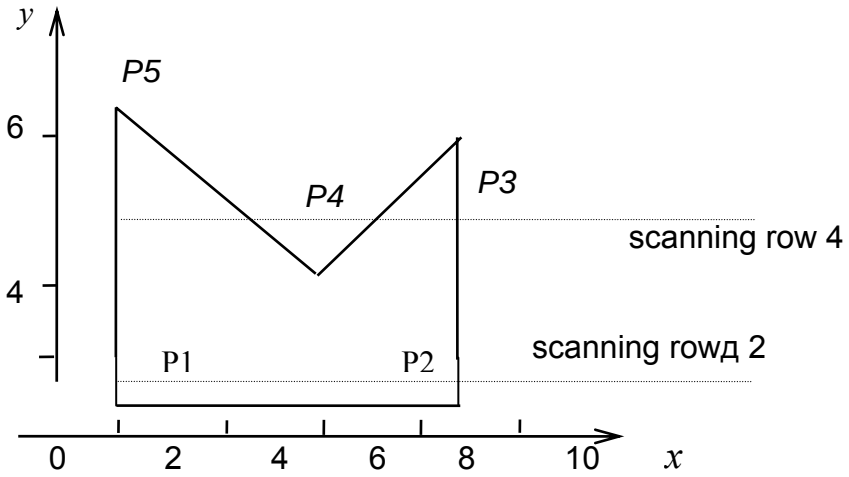


Fig. 9. 1

On the figure 9.1 the row 4 with color of background will be draw pixels: from 0 to 1; from 4 to 6, from 8 to 10. The pixels from 1 to 4 and from 6 to 8 will be drew with color filling.

8.2. The Algorithm with ordered list of sides.

Preparation of data:

It's obtained for every side of polygon the points of crossing with scanning rows. The horizontal sides are ignored. All points of crossing *(x, y+1/2)* are set in the list. The list is sorted in order to increasing of x: (x_1,y_1) *before* (x_2,y_2) , if $y_1 > y_2$ or $y_1 = y_2$ and $x_1 \le x_2$ **.**

The transformation of data in raster mode:

It's separated from sorted list couples elements (x_1,y_1) and (x_2,y_2) **.** The structure of list guarantees that $y = y_1 = y_2$ and $x_1 \le x_2$. It's activated in scanning row y pixels for integer value of x like : $x_1 \le x+1/2 \le x_2$ **.**

8.3. The Algorithm chain filling

In this algorithm requires setting of one point into the area contour. The algorithm examines the neighbor pixels whether they are into area contour. The algorithm tries to find all pixels into the polygon and to light them. The color of pixels of contour are different from color.

If the area is limited all pixels of contour have the same color. The pixels of outside have different color obligatory. The limited area can be 4 or 8 - connected. If the area is 4- connected every pixel in it can be gain with movement in 4 directions- left, right, up, down. For the 8 - connected area every pixel in it can be gain with combination with 2 horizontal, 2 vertical and 4 diagonal directions.

8.4 The algorithm of filling of limited contour using stack

Set the chain pixel in stack.

WHILE *the stack is not empty do*

begin

Get the pixel from stack.

Assign to pixel of necessary value.

Every one from neighbor pixels of the current *4-connected pixel is verified for belonging of contour or belonging of contour inside. If we has first case the pixel is ignored otherwise the pixel is set in stack.*

end

This algorithm can be modified for 8-connected area if we brose 8-connected pixels.

8.5. The algorithm of filling

The point *(x, y)* ***sets the chain pixel***

Push ***– procedure which puts the pixel in stack***

*Pop****- procedure which gets the pixel from stack***

Pixel(x, y)= Point(x, y)

Initialisation of stack

Push Pixel(x, y)

while (Stack isn't empty)

Get pixel from stack

Pop Pixel(x, y)

if Pixel*(x, y)<> New_Value* ***then***

Pixel*(x, y)=New_Value*

endif

Verification neighbor pixels whether they have to set in stack. if *Pixel(x+1, y)<>New_value****. and. Pixel****(x+1, y)<>Limit_value* ***then***

Push Pixel(x+1, y) ***endif***

if *Pixel(x, y+1)<>New_Value* ***.and.*** *Pixel(x, y+1)<>Limit_value* ***then***

Push Pixel(x, y+1) ***endif***

if *Pixel(x-1, y)<>New_value****. and.*** *Pixel(x-1, y)<>Limit_value* ***then***

Push Pixel(x-1, y) ***endif***

if *Pixel(x,y-1)<>New_value****. and.****Pixel(x, y-1)<>Limit_value* ***then***

Push Pixel(x, y-1) ***endif***

end while

finish i

In this algorithm are verified and set in stack 4-connected pixels beginning with right current pixel. The direction of visitation of pixels is opposite to the clock arrow.

9. CURVES IN DISCRET GRID

In general mathematical description of curve in set plane can be presented with next parametrical equation: $x = X(t), y = Y(t)$, where $\alpha \le t \le \beta$, (11.1)

The curve can be presented with equation with different mode as: $f(x, y) = 0$ (11.2)

For presentation of curves, the first mode is more convenient like parameter t is taken the sequences $t_1,...,t_i,...,t_n$ and it's computed corresponding pixels.

For their coordinates it's used round values: $X(t_i)$ and $Y(t_i)$. The image result consists from single pixels and connecting lines. The most important component in this case is the number of computed points.

9.1. GRAPHICAL PRESENTATION OF EXPERIMENTAL DATA IN DEFINED GRAPHICAL WINDOWS

Input data:

Let there are *n number of couple of real data* { $(x_i y_i)$ $i = 1,...,n$ ***This data can be obtained from experiments or computing.***

Result: an image in the graphical window that shows points corresponded with couples of real data.

On the Fig. 4.11 is presented graphical image of experimental data.

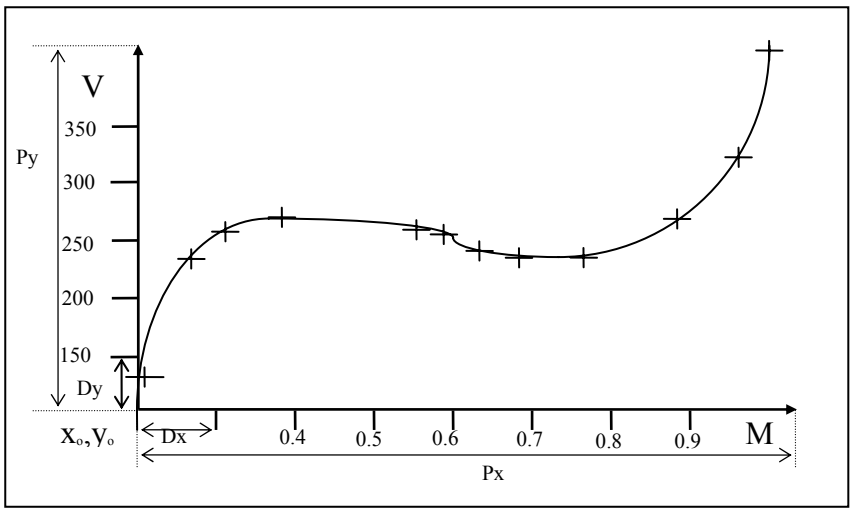


Fig. 11. 1

Steps of algorithm:

The obtaining of image of data $\{(x_i, y_i)\ i = 1,...,n\}$ ***(from experiment or computing) has next steps:***

1. Setting the requirements to image.

- The place of graphical window. In this case it's used the coordinates of left bottom corner x_0, y_0***.***

- The size of graphical window in horizontal direction P_x ***and in vertical direction*** P_y ***in pixels.***

- A kind of axes and size of their grid D_x, D_y ***, axes text.***

- Colors and number of dependences for drawing and additional texts.

2. <u>Computing of scopes of data</u>

This process includes computing of minimal and maximal data that will be presented in horizontal and vertical direction.

$$X\min = \min\{x_1,...,x_n\}, \quad X\max = \max\{x_1,...,x_n\}$$

$$Y\,min = min\{y_1,...,y_n\}, Y\,max = max\{y_1,...,y_n\}.$$

There is possible to extend of scope according some reasons.

<u>3. Computing of scale coefficients</u>

The scale coefficients are depending on scopes of real data and size of graphical window.

<u>scale coefficient in horizontal direction:</u> $S_x = \frac{X\,max - X\,min}{P_x}$

<u>scale coefficient in vertical direction:</u> $S_y = \frac{Y\,max - Y\,min}{P_y}$

where: Px, Py are the size of window.

S_x **-** scale coefficient in horizontal direction;

S_y **-** scale coefficient in horizontal direction .

<u>4. Drawing of axes</u>

- Beginning and end of axes:

Horizontal axis - (x_0, y_0), $(x_0 + P_x, y_0)$

Vertical axis - (x_0, y_0), $(x_0, y_0 - P_y)$

- Number of grid segments:

Horizontal axis - $Ip = |P_x / D_x|$

Vertical axis - $Jp = |P_y / D_y|$

- First and last point of i-st grid segment:

Horizontal axis - $(x_0 + i.D_x, y_0)$, $(x_0 + i.D_x, y_0 + 3)$, $i = 1,...I_p$

Vertical axis - $(x_0, y_0 - j.D_y)$ $(x_0 - 3, y_0 - j.D_y)$, $j = 1,...J_p$

-The value which corresponds with i-st grid segment:

Horizontal axis - $(X\,min + i.D_x.S_x)$, $i = 1,...I_p$

Vertical axis - $(Y\min + j.D_y.S_y)$, $j = 1,...J_p$

<u>5. Computing of coordinates of points which correspond with couple of real data:</u>

$$x'_i = x_0 + \frac{x_i - X\,min}{S_x}$$

$$y'_i = y_0 - \frac{y_i - Y\,min}{S_y}$$

Where: x'_i, y'_i ***- coordinates of point;***

(x_i, y_i)***. couple real data;***

Sx, Sy – scale coefficients;

Xmin, Ymin – bottom borders of data scopes.

10 THE ECONOMIC GRAPHICS
HISTOGRAMS AND CIRCLE DIAGRAMS

Graphical images are used about visualization of evaluations and analysis of economical, statistical and sociological information. The using of presentation of this information with histograms and circle diagram is very useful and developed. This way of presentation allows users to understand presented information very quickly.

The input data are couples *of data* ***{ $(t_i\ a_i)$ i =1,...,n This data can be obtained by different ways.***

10.1. Histograms

Histograms are images which present the data like vertical or horizontal bars. Usually they are fill with different colors. They have no more than 7 or 10. Fig 5.1. shows an example for vertical histogram.

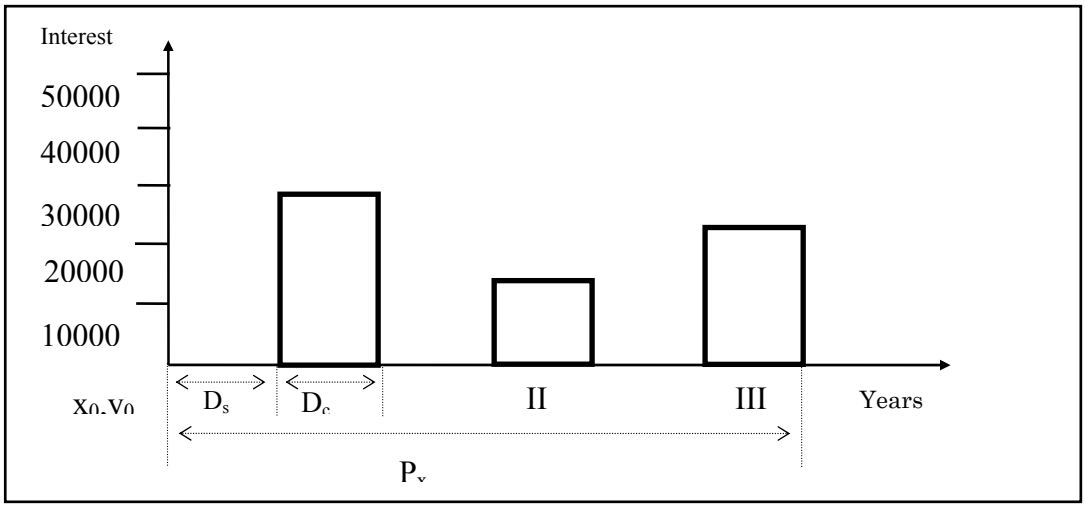


Fig. 10.1

Steps of algorithm for building of histograms:

The obtaining of image of data { $(t_i\ a_i)$ i =1,...,n *) i =1,...,N}* ***(from experiment or computing) has next steps:***

1. Setting the requirements to image.

- The numbers of bars N and their directions;

- The place of graphical window. In this case it's used the coordinates of left bottom corner x_0, y_0***;***

- The size of graphical window in horizontal direction P_x ***and in vertical direction*** P_y ***in pixels.***

- A kind of value axis and size of their grid D.

- For text Axis distance between bars Ds ***and bar's width*** Dc ***bars.***

- Colors and number of dependences for drawing and additional texts.

3. <u>Computing of scopes of data</u>

This process includes computing of minimal and maximal data that will be presented in horizontal and vertical direction.

$$A\min = \min\{a_1,...,a_N\}, \quad A\max = \max\{a_1,...,a_N\}$$

There is possible to extend of scope according some reasons.

The main cases are:

a/ If the Amin>0 then Amin=0 (the scope will be (0;Amax);

b/ If the Amax<0 then Amax=0 (the scope will be (Amin; 0).

<u>3. Computing of scale coefficients</u>

The scale coefficients are depending on scopes of real data and size of graphical window.

<u>scale coefficient for value axis :</u> $S = \frac{A\max - A\min}{P}$

where: P is the size of window, ***S-*** scale coefficient for value axis.

<u>4. Drawing of axes</u>

4.1 Value Axis

- Beginning and end of axes:

Horizontal axis - (x_0, y_0), $(x_0 + P_x, y_0)$

Vertical axis - (x_0, y_0), $(x_0, y_0 - P_y)$

- Number of grid segments:

Horizontal axis - $Ip = |P_x / D_x|$

Vertical axis - $Jp = |P_y / D_y|$

- First and last point of i-st grid segment:

Horizontal axis - $(x_0 + i.D_x, y_0)$, $(x_0 + i.D_x, y_0 + 3)$, $i = 1,...I_p$

Vertical axis - $(x_0, y_0 - j.D_y)$ $(x_0 - 3, y_0 - j.D_y)$, $j = 1,...J_p$

-The value which corresponds with i-st grid segment:

Horizontal axis - ($X\min + i.D_x.S_x$***),*** $i = 1,...I_p$

Vertical axis - ($Y\min + j.D_y.S_y$***),*** $j = 1,...J_p$

4.2. Text Axis

- Number of texts below bars N.

- First and last point of i-st grid segment of text axis:

- Horizontal axis $(x_0 + i.(D_s + D_c) - D_c, y_0)$, $(x_0 + i.(D_s + D_c), y_0)$, $i = 1,...N$

- Vertical axis $(x_0, y_0 - i.(D_s + D_c)) + D_c)$, $(x_0, y_0 - i.(D_s + D_c) - D_c)$, $i = 1,...N$

5. Computing of coordinates of points which definite bars according data $(a_i, i=1...N)$

5.1. Coordinates of i-st vertical bar:

- coordinates of top left corner $(x_0 + i*(D_s + D_c) - D_c, y_0 - \frac{(a_i - A_{\min})}{S})$;

- coordinates of bottom right corner $(x_0 + i*(D_s + D_c), y_0)$.

5.2. Coordinates of i-st horizontal bars:

- coordinates of top left corner $(x_0, y_0 - i*(D_s + D_c))$;

- coordinates of bottom right corner $(x_0 + \frac{(a_i - A_{min})}{S}, y_0 - i*(D_s + D_c) - D_c)$.

10.2. CIRCLE DIAGRAM

Circle diagrams are image that present data like part from one circle , ellipse or cylinder. Often they are used to present percents data. An example is on Fig. 11.3.

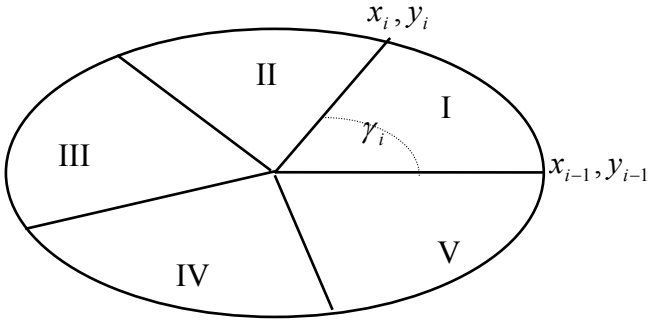


Fig 10.2

The sectors of circle or ellipse are draught with carefully chosen colours and fills. The circle diagram is clear if the sectors are less than 7.

Algorithm for building of circle diagram.

The obtaining of image of data { (t_i a_i) i =1,...,n) i =1,...,N} (from experiment or computing) has next steps:

1. Setting the requirements to image.

- ***The numbers of sectors N and colour fills for every one sector;***
- ***The center*** x_0, y_0 ***and radiuses of circle(R) or ellipse(Rx, Ry);***
- ***Texts for every one sector and additional texts.***

2. The computing of angle of sector that correspond to data.

- ***summarizing of all data*** $A = \sum_{i=1}^{N} a_i$ ***;***
- ***central angle that correspond with i-st data a_i is*** $\gamma_i = \frac{2.\pi.a_i}{A}$, $i = 1,...N$.
- ***coordinates of points that definite of i-st sector are :***

for circle:

initial point (X_{i-1}, Y_{i-1}) ***, end point*** (X_i, Y_i) ***, where:***

$$X_i = O_x + R.\cos \sum_{1}^{N} \gamma_i \text{ ,}$$

$$Y_i = O_y - R.\sin \sum_{1}^{N} \gamma_i \text{ , } i = 1,...N \text{ .}$$

for ellipse:

initial point - (X_{i-1}, Y_{i-1}) ***, end point*** (X_i, Y_i) ***, where:***

$$X_i = O_x + R._x \cos \sum_{j=1}^{i} \gamma_j \text{ ,}$$

$$Y_i = O_y - R_y.\sin \sum_{j=1}^{i} \gamma_j \text{ , } i = 1,...N \text{ .}$$

The Algorithm which uses the graphical functions in C++

```
Sum=0
for i=1 to N    Sum=Sum+a(i)
start_angle=0
For i=1 to N
begin
    gama=2*Pi*a(i)/Sum
    end_angle=start_angle+d=gama
    Set color and fill style.
    draw sector with parameters:  start angle  and end_angle
end
```

11 INTERPOLATION AND APPROXIMATION ALGORITHMS

11.1. Classical approach for interpolation and approximation

Let it is given totalities real data $x_1,...,x_n$ and $y_1,...,y_n$, where $x_1 < x_2 < ... < x_n$ and y_i is some observed or computed number. The main task of one-dimensional interpolation is the building of function for which is valid $f(x_i) = y_i$ about every one i and the function f(x) take "reasonable" value for every one x, $x_1 < x < x_n$. The criteria for "reasonable" can change and can never to be understood right. If the point (x_i, y_i) are very precise experimental data it can be rationally to interpolate they with smooth function. If the data are crude might turn out that we need with different requirements of task. If it allows the value $f(x_i)$ to be different from y_i, the result can present the character of data more right. In this way the task become the task of approximation. - $f(x_i) \approx y_i$ for every i and $f(x)$ to take "reasonable" value for $x, x_1 < x < x_n$.

The searching of a curve that crosses exactly across multitude of points is the task of **interpolation.** The searching of a curve that crosses near by

multitude of points is the task of **approximation.** For bought we ca use the term “Building curve from set points. The essence of task is: From infinite number curves that we can build to choose one with a suitable criterion. Most interpolation and approximation algorithm are built using the linear combination from elemental functions. When the elemental function is polynomial the kind of algorithm is definite from kind of polynomial.

Polynomials

Interpolation

The function which interpolates data : multitude of points (x_i, y_i), has a mode:

$$y(x) = C_1F_1(x) + C_2F_2(x) + ... + C_nF_n(x), \qquad (11.1)$$

where $F_n(x)$ is monomial like these:

$$F_n(x) = x^n,\ F_n(x) = \sin(nx),\ F_n(x) = e^{\lambda_n x}$$

Theorem: It exists only one polynomial from power n or less which interpolates n+1 data. The polynomial is obtained after a solution of system linear equations. This leads to problems with roundness errors that rise with the power of polynomial.

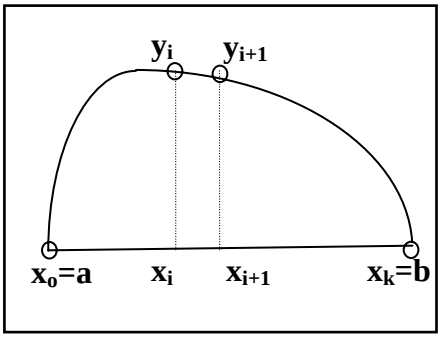
Approximation task is definite with:

$$y(x) \approx C_1F_1(x) + C_2F_2(x) + ... + C_nF_n(x), \qquad (11.2)$$

where $F_n(x)$ is polynomial.

11.2 Spline curve

A spline curve is function that is obtained with different partial functions. In general aspect splines are presented below.



$$P(x) = P_i(x),\ x_i \le x \le x_{i+1}\ ,\ i = 0,1,...,k-1$$

$$P_i^{(j)}(x_i) = P_{i+1}^{(j)}(x_i),\ i = 1,...,k-1,\ j = 0,1,...,r-1$$

where :

x_0, x_k - start- and end-point if segment [a, b] , $x_0 = a, x_k = b$;

$x_1,...,x_{k-1}$ - points which divide segment [a, b] and obtain sub-segments. Every segment is definite from stick points (x_i , x_{i+1}) and corresponding curve points with names "**knots**" (y_i , y_{i+1})

i - sub-segment number;

$k-1$ - total number of sub-segments;

$P_i(x)$ - polynomial with power to m;

$P_i^{(j)}(x)$ - j-st derivate of $P_i(x)$ in point x_i .

The conditions for continuity are set with equations where *j>0.* Where the limitation drop out then *r=0*. If *r=1* there is continues function under whose derivates is not put limitations. If *r=m+1*, the segment [a, b] is covered with a polynomial and therefore if *r=m* there is maximal number of limitations raised from partial function $P_i(x)$.

We have Simple spline if *r=m*. The terms: linear, quadratic and cubic spline refer for ***m=1, m=2, m=3*** respectively. The simple spline have ***k+m*** power of freedom. Regular spline is called partial polynomial function if *r<m*. The sort of polynomial $P_i(x)$ determinates sort of spline.

11.2.1. Interpolation curve of Katmul-Rom

For area of points P_0, P_1, P_2, P_3 the spline curve of Katmul-Rom is definite with the equatin:

$$R(t) = \frac{1}{2}\left(-t(1-t)^2 P_0 + (2-5t^2+3t^3)P_1 + (1+4t-3t^2)P_2 - t^2(1-t)P_3\right), \quad 0 \le t \le 1.$$

This curve has following features: crosses points exactly; the function has continuity and definiteness.

11.2.2. Elemental beta-spline curve

For area of points P_0, P_1, P_2, P_3 бета – spline curve are determinate with equation:

$$R(t) = b_0(t)P_0 + b_1(t)P_1 + b_2(t)P_2 + b_3(t)P_3, \quad 0 \le t \le 1.$$

Function coefficients $b_0(t)$, $b_1(t)$, $b_2(t)$, $b_3(t)$ are put with next formulas:

$$b_0(t) = \frac{2\beta_1^3}{\delta}(1-t)^3; \qquad b_3(t) = \frac{2t^3}{\delta};$$

$$b_1(t) = \frac{1}{\delta}(2\beta_1^3 t(t^2 - 3t + 3) + 2\beta_1^2(t^3 - 3t + 2) + 2\beta_1(t^3 - 3t + 2) + \beta_2(2t^3 - 3t + 1));$$

$$b_2(t) = \frac{1}{\delta}(2\beta_1^2 t^2(-t + 3) + 2\beta_1 t(-t^2 + 3) + 2\beta_2 t^2(-2t + 3) + 2(-t^3 + 1));$$

where $\beta_1 > 0$ and $\beta_2 \geq 0$ and $\delta = 2\beta_1^3 + 4\beta_1^2 + 4\beta_1 + \beta_2 + 2$.

The parameters β_1 and β_2 are called format parameters of beta –spline curve. The features of this curve are following: comes inside to jut enveloping curve. The shape of curve can be regulated by parameters β_1 and β_2 .

12.2.3. The spline curve of Bezie

Let is put area of **points of support** P_0, P_1, P_2, P_3 . The spline curve of Besie is definite with equation:

$$R(t) = (((1-t)P_0 + 3tP_1)(1-t) + 3t^2 P_2)(1-t) + t^3 P_3, \quad 0 \leq t \leq 1.$$

The curve begins with a point P_0 and ends with a point P_3, touching to segments P_0P_1 and P_0P_1. This curve has following features: comes inside to jut enveloping curve; the function has continuity and definiteness.

11.2.4. B – spline

B – spline with constant value of i-st sub-segment is put by expression:

$$N_{i,0}(x) = \begin{cases} 1, x_i \leq x \leq x_{i+1} \\ 0, x \notin [x_i, x_{i+1}] \end{cases}$$

B-spline with power m for segment $[x_i, x_{i+m+1}]$ is definite by expression:

$$N_{i,m}(x) = \frac{x - x_i}{x_{i+m} - x_i} N_{i,m-1}(x) + \frac{x_{i+m+1} - x}{x_{i+m+1} - x_{i+1}} N_{i+1,m-1}(x)$$

Every polynomial $N_{i,m}$ is obtained from one or two another polynomials with one power below $N_{i,m-1}$ $N_{i+1,m-1}$.

The partial polynomial has mode:

$$P(x) = \sum_{i=-m}^{k-1} a_i . N_{i,m}(x) \qquad (12.3)$$

The coefficients a_i are definite from limitation equations and the number of unknown values depends on the choice of knots.

11.3. Polynomial of Lagrang

This algorithm uses polynomial associated with the multitudes $\{x_i\}$, $l_j(x)$ $l_j(x)$ possessing following features: $l_j(x_i) = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases}$

$$l_j(x) = \frac{(x - x_0).....(x - x_{j-1})(x - x_{j+1}).....(x - x_n)}{(x_j - x_0)(x_j - x_1).....(x_j - x_{j-1})(x_j - x_{j+1}).....(x_j - x_n)}$$

The founded function has mode:

$$y = \sum_{j=0}^{n} l_j(x).y_j \qquad (11.4)$$

The main defect of this curve is the considerable hesitations that curve can have between two neighbor points.

12.4. Polinomial of Bezie

The algorithm uses the multitude of **points of support**. The polynomial is definite with parameter mode:

$$X = P_x(t),\ X = P_x(t) \quad when \quad t \quad isaparameter \quad and \quad 0 \le t \le 1$$

If $(x_0, y_0), (x_1, y_1).....(x_m, y_m)$ are points of support the polynomial of Bezie is definite like this:

$$P_x(t) = \sum_{i=0}^{m} C_m^i.t^i.(1-t)^{m-i}.x_i \qquad (11.5а)$$

$$P_y(t) = \sum_{i=0}^{m} C_m^i.t^i.(1-t)^{m-i}.y_i \qquad (11.5б)$$

Where C_m^i is combination from m objects in i - class, m- number of poits of support.

$$C_m^i = \frac{m!}{i!.(m-i)!}$$

In vector form the polynomial is presented below:

$$P(t) = \begin{bmatrix} P_x(t) \\ P_y(t) \end{bmatrix},\ P_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix},\ P_x(t) = \sum_{i=0}^{m} C_m^i.t^i.(1-t)^{m-i}.x_i$$

The scope of t is from 0 to 1.

that points of support can be right definite by only very competent user. This curve has following features: comes inside to jut enveloping curve; the function has continuity and definiteness. This algorithm is very convenient for 3-dimensional cases.

The geometric algorithm for building of curve of Bezie

Symbols:

P_i- Initial points of support; R_i - old points of support; Q_i- new points of support.

For every t from 0 to 1 with step dt do

begin

for i from 0 to m $R_i = P_i$;

n=m;

While n>0 do

begin

For i from 0 to n-1 $Q_i = R_i + t \cdot (R_{i+1} - R_i)$;

n=n-1;

For i from 0 to n $R_i = Q_i$;

end;

P(t)= R_0;

end.

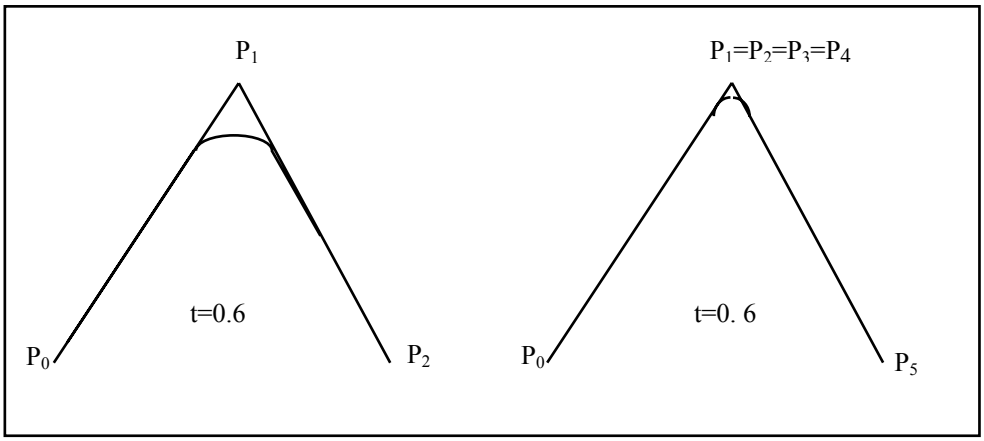


Fig. 11.2

If $m < 5$ is convenient to use geometrical algorithm for building of curve of Bezie because of number of operations is less. However if m>5 is preferred the algorithm of Horner.

The algorithm of Horner

Symbols:

P_i - Initial points of support; R_i - old points of support; Q_i - new points of support.

For every t from 0 to 0,5 with step dt do

begin

computing $(1-t)^m$;

$Q_0=P_m$;

for i from 1 to m do

begin

$Q_i=(t(1-t))Q_{i-1}+ C_m^{m-i}.P_{m-i}$;

end;

$P(t)= (1-t)^m.Q_m$;

end;

For every t from 0,5 to 1 with step dt do

begin

computing t^m;

$Q_0 =P_0$;

for i om 1 ∂o m do

begin

$Q_i=((1-t)/t).Q_{i-1} + C^i_m.P_i$;

end;

$P(t)= t^m.Q_m$;

end.

The closeness of curve to points of support can be regulated with repeated using of one point of support like example of Fig 6.2. Using of (P_1=P_2=P_3=P_4) leads the more closeness curve to point P_1.

12 TWO DIMENSIONAL TRANSFORMATIONS

There are four main basic graphical transformations: translation, scaling, rotation and reflection. They correspond to geometrical concept for identity, similarity and axis symmetry. All possible image transformation can be presented with composition from these basic transformations.

The each contour consists of a lot of pixel points. All these point must to be transformed in the same way. The transformations are presented below with equations for one point, but they must be apply for each one point from contour. It's used symbols:

P (x,y) –for point before transformation and name (old point);

P' (x',y') –for point after transformation and name (new point).

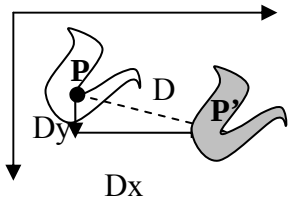
12.1 Translation

For transformation TRANSLATION it's necessary to definite the vector of translation. D=||dx dy||, where dx is a X-component and dy is a Y-component.

Let P=(x,y) is a point from contour before the transformation (old point). Let P'=(x',y') is a point from contour after the transformation (new point).

The equations which describe dependence between them are:

$$x' = x + dx,$$
$$y' = y + dy$$



Translation

In matrix mode the equations can be written like: P'=P+D, where

$$P = \begin{Vmatrix} x \\ y \end{Vmatrix} - old\ \ point \quad P' = \begin{Vmatrix} x' \\ y' \end{Vmatrix} - new \quad point \quad D = \begin{Vmatrix} dx \\ dy \end{Vmatrix} - translation\ vector$$

There is another matrix equation which use multiplication instead addition.

*[x'y'1]=[xy1]*T(dX, dY)* , where

$$T(dX,dY)=\begin{Vmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ dX & dY & 1\end{Vmatrix}$$

The Using of homogeneous coordinates allows to present translation dependence with third matrix equations.

$$[u_1 \quad v_1 \quad 1]=[x \quad y \quad 1]T',\ \textit{където}\ T'=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ -x_0 & -y_0 & 1\end{bmatrix}$$

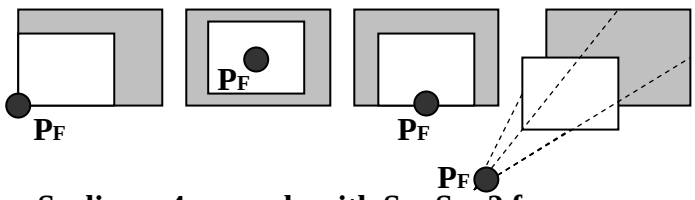
12.2. Scaling

For Scaling it's necessary to definite the matrix with scaling coefficients Sx, Sy and point of scaling P_F=|| x_F y_F ||. On the figure below it can see four example with scaling with Sx=Sy=2 and different place of point P_F. Actually it scales the distance between P_F and P with scale coefficients Sx, Sy.

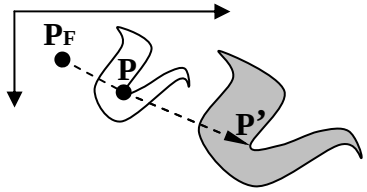
Let P=(x,y) is a point from contour before the scaling (old point). Let P'=(x',y') is a point from contour after the scaling (new point).

The equations which describe dependence between them are:

$$x'=x_F+(x-x_F)S_x$$
$$y'=y_F+(y-y_F)S_y$$



Scaling – 4 example with Sx=Sy=2 for different point P_F



Scaling – regular case

In matrix mode the equations can be written like: P'=P_F+ (P_F-P)S, where

$$P=\begin{Vmatrix}x\\ y\end{Vmatrix}-old\ p.\ P'=\begin{Vmatrix}x'\\ y'\end{Vmatrix}-new\ p.,P_F=\begin{Vmatrix}x_F\\ y_F\end{Vmatrix}-point\ of\ scaling,\quad S=\begin{Vmatrix}S_x & 0\\ 0 & S_y\end{Vmatrix}-scaling\ matrix$$

There is another matrix equation which uses only multiplication.

$[x'y'1] = [x_F y_F 1] + ([xy1] - [x_F y_F 1]) * S(S_X, S_Y)$, *where:* $S = \begin{vmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{vmatrix}$

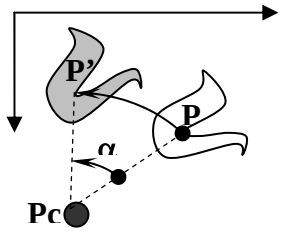
12.3. Rotation

For rotation it's necessary to definite the centre P_R=|| x_R y_R ||. of rotation and angle of rotation α. The object turns around P_R with angle α.

Let P=(x,y) is a point from contour before the transformation (old point). Let P'=(x',y') is a point from contour after the transformation (new point).

The equations which describe dependence between them are:

$$x' = x_R + ((x - x_R).\cos(\alpha)) - (y - y_R).\sin(\alpha))$$
$$y' = y_R + ((y - y_R).\cos(\alpha)) + (x - x_R).\sin(\alpha))$$



Rotation

In matrix mode the equations can be written like: $P' = P_R + (P_R - P)R$, where

$$P = \left\| \begin{matrix} x \\ y \end{matrix} \right\| - old\ p.\ P' = \left\| \begin{matrix} x' \\ y' \end{matrix} \right\| - new\ p., P_R = \left\| \begin{matrix} x_F \\ y_F \end{matrix} \right\| - centre\ of\ rotation, \quad R = \left\| \begin{matrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{matrix} \right\| - rotation\ matrix$$

There is another matrix equation which uses only multiplication.

$[x'y'1] = [x_R y_R 1] + ([xy1] - [x_R y_R 1]) * R$, **where** $R = \left\| \begin{matrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{matrix} \right\|$ **-** matrix of rotation**.**

12.4. Transformation composition

Let't is set the composition: Rotation around P_R with angle 45°, scaling to P_F with Sx=2, Sy=3 and moving with dx=30, dy=-40. Common record of composition will be consisted in this way: The result from first transformation will be input data for second transformation. It's result will be input data for third transformation. The composition record is:

$$P' = \left\{ P_F - \left[(P_R - (P - P_R) \begin{Vmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{Vmatrix} - P_F \right] \begin{Vmatrix} 2 & 0 \\ 0 & 3 \end{Vmatrix} \right\} + \begin{Vmatrix} 30 \\ 40 \end{Vmatrix}$$

12.5. Image Movement

The graphical transformation compositions are used to make image movement. The way is following:

1. The image object is draw with color different from background color.
2. The image object is draw with the same color as background color.
3. The image object is transformed.
4. Do step 1 and 2 again.

Computer Graphics Bibliography

1. Lode's Computer Graphics Tutorial, http://student.kuleuven.be/~m0216922/CG/index.html,

2. Computer graphics Cornell University Program of Computer Graphics, 3. http://www.graphics.cornell.edu/online/tutorial/

3. Wikipedia, the free encyclopedia, chapter Computer graphics, http://en.wikipedia.org/wiki/Computer_graphics

4. 3D Computer Graphics (3rd Edition) (Hardcover) by Alan H. Watt