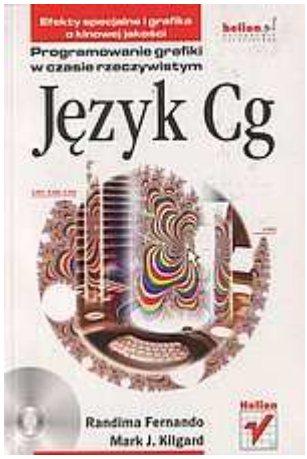
PROGRAMMABLE GRAPHICS



Cg is a complete development environment for fast creation of special effects and cinema-quality graphics in real time for multiple platforms. Because the language is hardware independent, developers can write code for OpenGL, DirectX and Windows, Linux, Mac OS X, and console platforms (Xbox) without the need for an assemble language. Cg was developed by NVIDIA Corporation in close collaboration with Microsoft® Corporation and is compatible with the OpenGL API and HLSL language for DirectX 9 library.

PROGRAMMABLE GRAPHICS IN PASCAL

==========

1. OPENING AND CLOSING THE GRAPHICAL MODE

Two integer variables are required to start working in graphical mode.
You can name them as you like, but best of all yes:

```
var ster, tryb: integer;
```

The ***ster*** variable will store information about the selected graphic driver and the ***tryb*** about the selected mode.

It is recommended to automatically detect the driver unless you are using a special driver.

```
ster := Detect;
```

Files are required to work in graphic mode:

Graph.tpu	a module with graphic procedures
EgaVga.bgi	driver file

tpu – Turbo Pascal Unit
bgi – Borland Graphic Interface

* It is best to copy these files and font files ~~~~.chr to the same folder where you save them. your programme.

If you want to use a different resolution and color palette you need to install a different file ***~~~.bgi***.

Opening the graphical mode:

path to the driver file

```
ster  := Detect;
InitGraph (ster, tryb, 'C:\TP\BGI');
```

If you do, you can write *
an empty path (two apostrophes without spaces)

```
ster  := Detect;
InitGraph (ster, tryb, '');
```

Close graphical mode:

```
CloseGraph;
```

Before you close the graphical mode, use "holding down the screen". (readln; readkey; etc.), otherwise nothing You will not see.

Detection of an error when initiating graphical mode.

If you want to find out what was the reason for the graphics not being opened, use the instructions:

```
blad  := GraphResult;
if blad<>0 then
writeln ('kod=', GraphErrorMsg(blad));
```

Momentary change of graphic mode to text mode

When working in graphical mode, you can switch to text mode without having to close the graphics. Commands are used for this purpose:

```
RestoreCrtMode;
SetGraphMode (tryb);
```

switch to text mode

return to graphical mode

2. COLOR SELECTORS

The colour of the pen should be selected by means of a command:

```
SetColor (kolor);
```

Where ***kolor*** is a numeric variable or a predefined constant.
0 - black, 1 - blue, ..., 14 - yellow, 15 - white (for standard mode).

The colour of the filling (brush) is selected with the command:

```
SetFillStyle(styl, kolor);
```

Where ***kolor*** determines the colour of the filling and ***styl*** of the filling method, e.g. 1 - uniform filling.

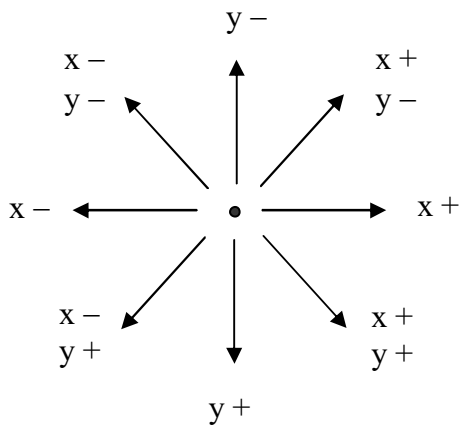
Select the background color with the command:

```
SetBkColor (kolor);
```

3. THE SCREEN IN GRAPHIC MODE

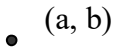
Remember that the Y-axis is pointing downwards (in contrast to the mathematical system).

Changing the position of a point on the screen



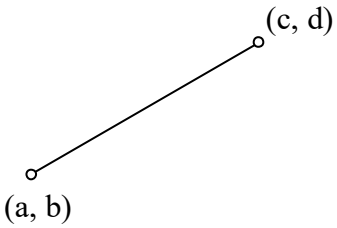
4. BASIC GRAPHIC PROCEDURES

pixel




```
PutPixel (a, b, kolor);
```

line




```
Line (a, b , c, d);
```

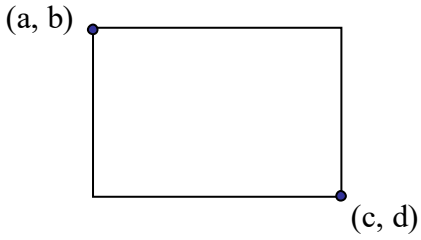
or

```
MoveTo (a, b);
LineTo (c, d);
```

or

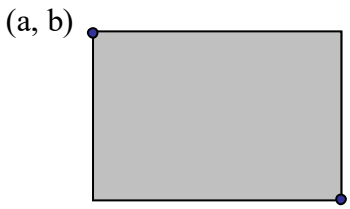
```
MoveTo (a, b);
LineRel (c-a, d-b);
```

rectangle *(„empty")*




```
RectAngle (a, b , c, d);
```

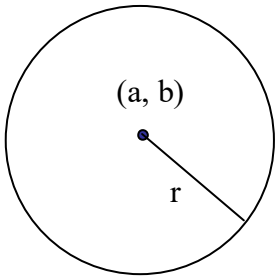
rectangle *(„fill")*




```
Bar (a, b , c, d);
```

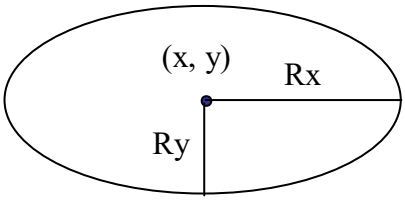
Use the SetLineStyle command to determine the line type (type, pattern, thickness).

circle



Circle (a, b , r);

Ellipse *(„empty")*



Ellipse (x, y, α, β, Rx, Ry);

(x, y) - ellipse centre
α - starting angle
β - end angle
Rx - horizontal radius
Ry - vertical radius

e.g **Ellipse (300, 400, 0, 360, 100, 980);**

Ellipse *(„fill")*

FillEllipse (x, y, Rx, Ry);

Arc

Arc (x, y, α, β, r);

e.g. **Arc(300, 400, 0, 90, 100);**

Polygon *(using the example of a triangle)*

```
const  trjkt  : array  [1..4] of PointType =
                    ((x:10; y:10),
                    (x:15; y:100),
                    (x:150; y:50),
                    (x:10; y:10));
```

DrawPoly(4, trjkt);

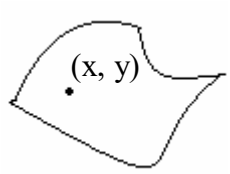
„empty"

FillPoly(4, trjkt);

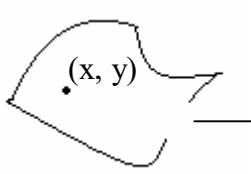
„fill"

5. FLOODFILL

The figure we want to paint must have a closed edge!



GOODS



BAD

paint "will flow through a hole".

FloodFill (x, y , kolor_brzegu);

Information on the colour of the point

GetPixel (x, y);

6. TEXTS IN GRAPHIC MODE

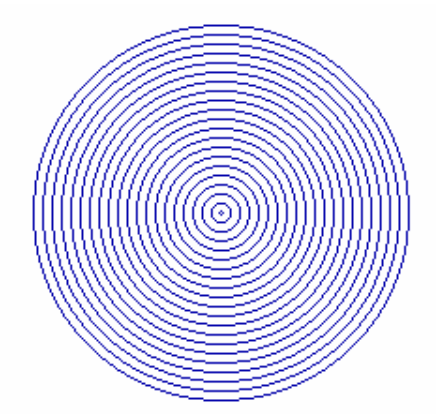
SetTextStyle(krój, kierunek, rozmiar);

OutTextXY (x, y, tekst);

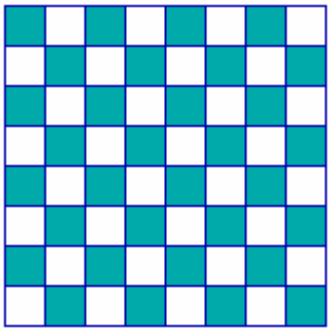
e.g. ***SetTextStyle(1, 0, 4)*** typeface - font type, direction - writing direction *(horizontal/vertical)*

7. EXAMPLES

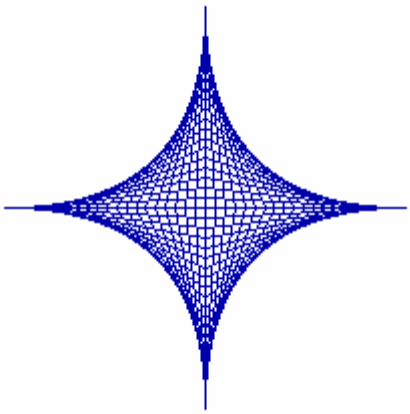
```
setcolor(1);
for k:=0 to 50 do circle(200,200,k*5);
```



```
setbkcolor(15);
cleardevice;
for k:=0 to 7 do
for j:=0 to 7 do
begin
if (k mod 2=0) then
if (j mod 2=0) then setfillstyle(1,3) else  setfillstyle(1,15)
else
if (j mod 2=1) then setfillstyle(1,3) else  setfillstyle(1,15);
bar(50+k*20,50+j*20,70+k*20,70+j*20);
end;
setcolor(1);
for k:=0 to 7 do
for j:=0 to 7 do
rectangle(50+k*20,50+j*20,70+k*20,70+j*20);
```

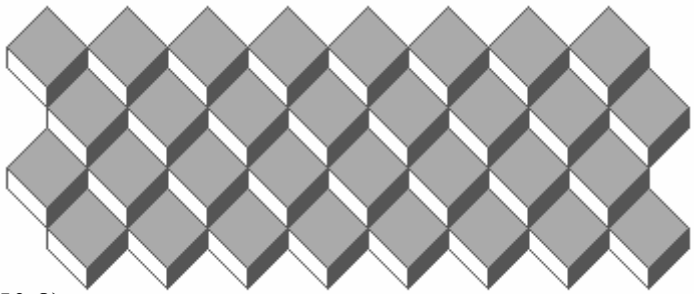


```
for k:=0 to 20 do
begin
line(200,50+k*5,200+k*5,150);
line(200,50+k*5,200-k*5,150);
line(200,150+k*5,300-k*5,150);
line(200,150+k*5,100+k*5,150);
end;
```

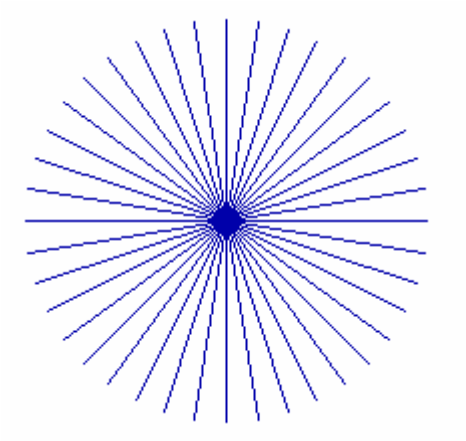


```
settextstyle(3,0,20);
for k:=0 to 5 do for j:=0 to 5 do
begin
if (k>0) and (j>0) then setcolor(1) else setcolor(11);
outtextxy(20+k,50+j,'ABC');
end;
```

```
for k:=0 to 7 do for j:=0 to 3 do
begin
moveto(50+k*40,50+j*50);
linerel(-20,20); linerel(0,10);
linerel(20,20); linerel(20,-20);
linerel(0,-10); linerel(-20,-20);
linerel(-20,20); linerel(20,20);
 linerel(0,10); linerel(0,-10);
linerel(20,-20);
setfillstyle(1,7);floodfill(50+k*40,60+j*50,8);
setfillstyle(1,8);floodfill(60+k*40,85+j*50,8);
 end;
```



```
kat:=pi/20;
for k:=0 to 19 do
begin
x:=100*cos(kat*k);
y:=100*sin(kat*k);
line(300,300,300+round(x),300+round(y));
end;
```



//////////////////////////////////////

\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\