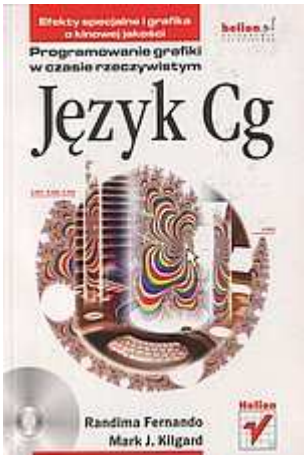
GRAFIKA PROGRAMOWANA



Cg to kompletne środowisko programistyczne do szybkiego tworzenia efektów specjalnych i grafiki o kinowej jakości w czasie rzeczywistym dla wielu platform. Ponieważ język jest niezależny od sprzętu, programiści mogą pisać kod dla interfejsów OpenGL, DirectX oraz systemów Windows, Linux, Mac OS X, a także platform konsolowych, (Xbox) bez potrzeby korzystania z języka asemblerowego. Język Cg powstał w firmie NVIDIA Corporation przy bliskiej współpracy z firmą Microsoft® Corporation i jest kompatybilny z OpenGL API oraz językiem HLSL dla biblioteki DirectX 9.

GRAFIKA PROGRAMOWANA W PASCALU
==========

1. OTWIERANIE I ZAMYKANIE TRYBU GRAFICZNEGO

Do rozpoczęcia pracy w trybie graficznym potrzebne są dwie zmienne typu ***integer***.
Możesz je nazwać jak chcesz, ale najlepiej tak:

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

Zmienna ***ster*** będzie przechowywała informacje o wybranym sterowniku graficznym, a ***tryb*** o wybranym trybie.

Zalecane jest automatyczne wykrycie sterownika, chyba że używasz sterownika specjalnego.

```
ster := Detect;
```

Do pracy w trybie graficznym niezbędne są pliki:

Graph.tpu ← moduł z procedurami graficznymi

EgaVga.bgi ← plik ze sterownikiem

tpu – Turbo Pascal Unit
bgi – Borland Graphic Interface

* Najlepiej jest skopiować te pliki oraz pliki czcionek ***~~~.chr*** do tego samego folderu, w którym zapisujesz swój program.

Jeżeli chcesz zastosować inną rozdzielczość i inną paletę kolorów musisz zainstalować inny plik ***~~~.bgi***.

Otwarcie trybu graficznego

ścieżka dostępu do pliku ze sterownikiem

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

Jeżeli wykonasz *, to możesz napisać pustą ścieżkę (dwa apostrofy bez spacji)

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

Zamknięcie trybu graficznego

```
CloseGraph;
```

Zanim zamkniesz tryb graficzny zastosuj „przytrzymanie ekranu" (readln; readkey; itp.), gdyż inaczej niczego nie zobaczysz.

Wykrycie błędu podczas inicjowania trybu graficznego

Jeżeli chcesz dowiedzieć się co było przyczyną, że grafika nie została otwarta, użyj instrukcji:

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

Chwilowa zmiana trybu graficznego na tekstowy

W trakcie pracy w trybie graficznym możesz przejść do trybu tekstowego bez konieczności zamykania grafiki. Do tego celu służą polecenia:

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

przejście do trybu tekstowego

powrót do trybu graficznego

2. WYBÓR KOLORÓW

Kolor pisaka *(pen)* wybieramy poleceniem:

```
SetColor (kolor);
```

gdzie ***kolor*** jest zmienną liczbową albo predefiniowaną stałą.
0 – czarny, 1 – niebieski, … ,14 – żółty, 15 – biały *(dla trybu standardowego).*

Kolor wypełnienia *(brush)* wybieramy poleceniem:

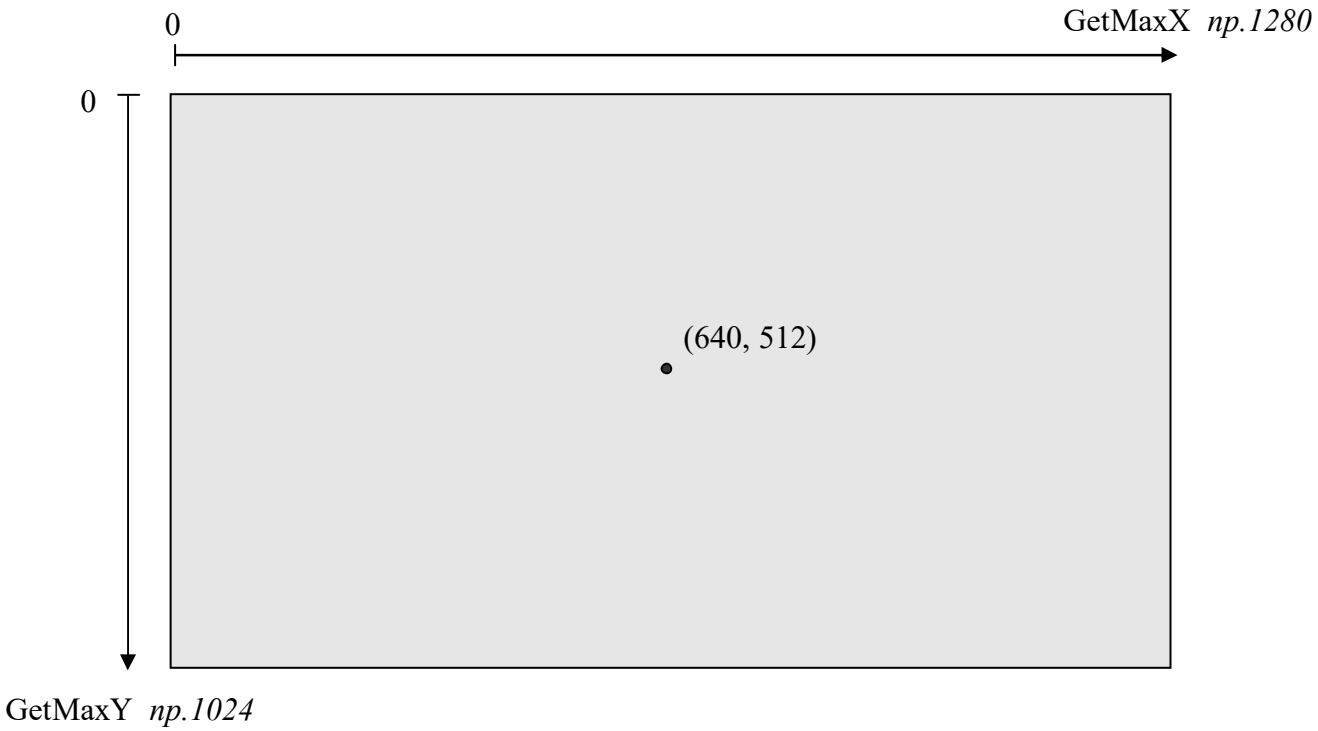
```
SetFillStyle(styl, kolor);
```

gdzie ***kolor*** określa kolor wypełnienia, a ***styl*** sposób wypełnienia np. 1 – wypełnienie jednolite.

Kolor tła *(background)* wybieramy poleceniem:

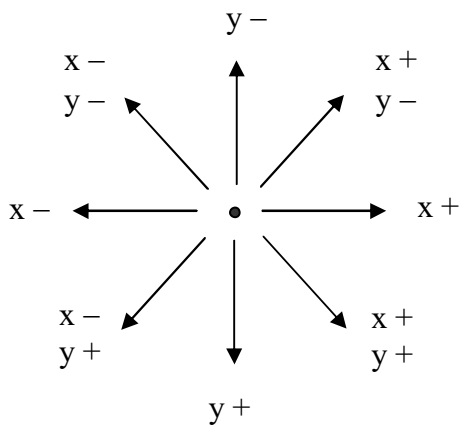
```
SetBkColor (kolor);
```

3. EKRAN W TRYBIE GRAFICZNYM



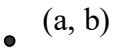
Pamiętaj, że oś Y jest skierowana do dołu (odwrotnie niż w układzie matematycznym).

Zmiany położenia punktu na ekranie



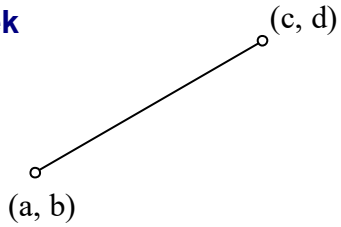
4. PODSTAWOWE PROCEDURY GRAFICZNE

punkt




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

odcinek




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

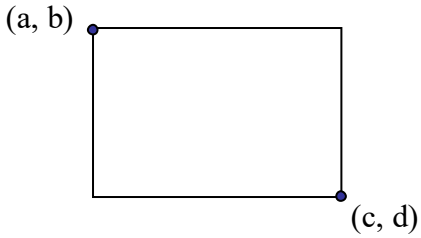
albo

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

albo

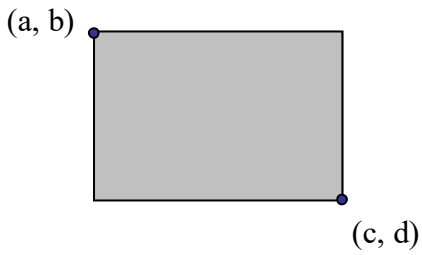
```
MoveTo (a, b);
LineRel (c-a, d-b);
```

Prostokąt *(„pusty”)*




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

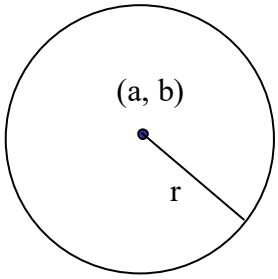
Prostokąt *(„wypełniony”)*



Bar (a, b , c, d);

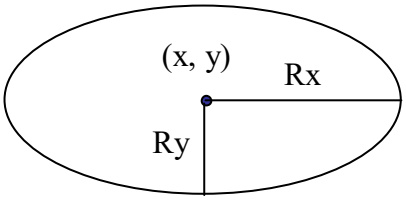
Do ustalenia rodzaju linii służy polecenie ***SetLineStyle(rodzaj, wzór, grubość).***

Okrąg



Circle (a, b , r);

Elipsa *(„pusta”)*



Ellipse (x, y, α, β, Rx, Ry);

(x, y) - środek elipsy
α - kąt początkowy
β - kąt końcowy
Rx - promień poziomy
Ry - promień pionowy

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

Elipsa *(„zamalowana”)*

FillEllipse (x, y, Rx, Ry);

Łuk okręgu

Arc (x, y, α, β, r);

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

Wielokąt *(na przykładzie trójkąta)*

```
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);

„pusty”

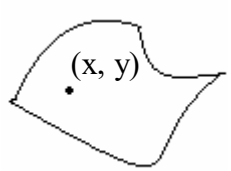
FillPoly(4, trjkt);

„zamalowany”

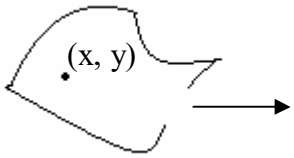
4 – bo pierwszy wierzchołek musi być podany też na końcu, żeby łamana się zamknęła.

5. „WYLEWANIE FARBY”

Figura, którą chcemy zamalować musi mieć zamknięty brzeg!



DOBRZE



farba „wypłynie przez dziurę”

ŹLE

FloodFill (x, y , kolor_brzegu);

Informacja o kolorze punktu

GetPixel (x, y);

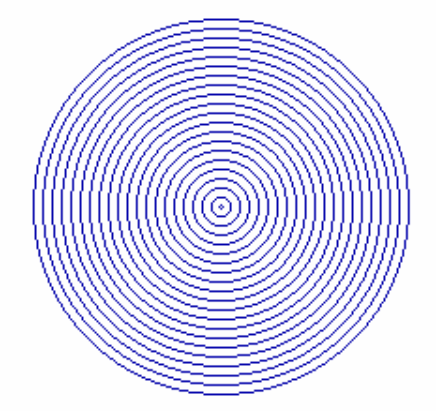
6. TEKSTY W TRYBIE GRAFICZNYM

SetTextStyle(krój, kierunek, rozmiar);	OutTextXY (x, y, tekst);

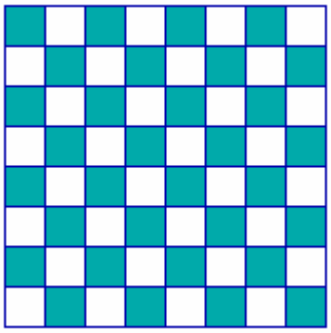
np. ***SetTextStyle(1, 0, 4)*** krój – rodzaj czcionki, kierunek – kierunek pisania (*poziomo/pionowo)*

7. PRZYKŁADY

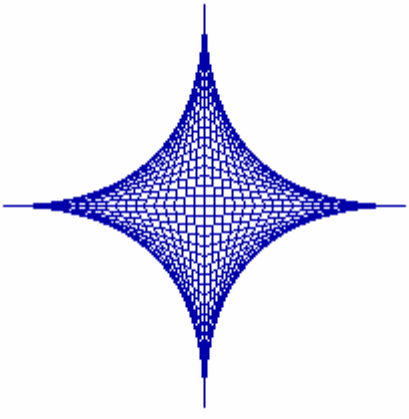
```
setcolor(1);
for k:=0 to 50 do circle(200,200,k*5);
```



```
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);
```

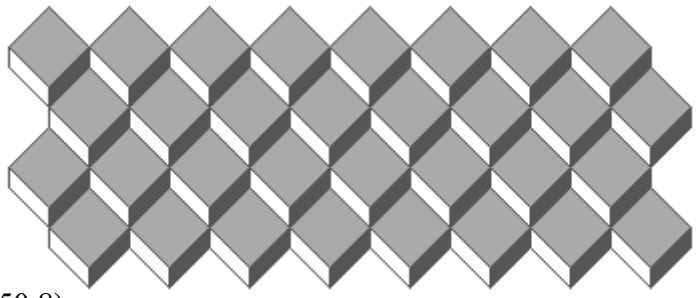


```
for k:=0 to 20 do
begin
line(200,50+k*5,200+k*5,150);
line(200,50+k*5,200-k*5,150);
line(200,150+k*5,300-k*5,150);
line(200,150+k*5,100+k*5,150);
end;
```

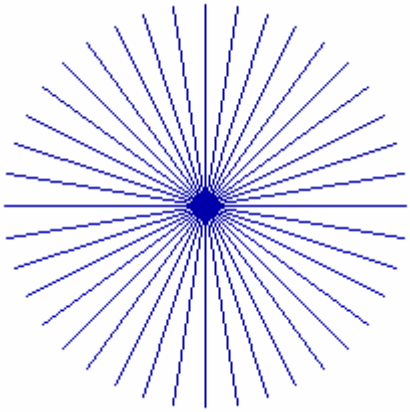


```
settextstyle(3,0,20);
for k:=0 to 5 do for j:=0 to 5 do
begin
if (k>0) and (j>0) then setcolor(1) else setcolor(11);
outtextxy(20+k,50+j,'ABC');
end;
```

```
 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;
```



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

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