

**Računarska grafika**  
(2OER7O02)

**Regioni, metafajlovi,  
putanje i transformacije**





**Vežbe**



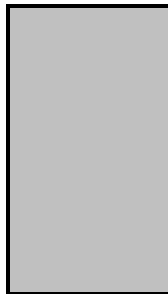
# Region

Region je pravougaonik, mnogougao, elipsa ili kombinacija ovih figura i definiše deo klijentske površine prozora koju je moguće obojiti, invertovati, ispuniti, uokviriti ili iskoristiti za detekciju pozicije kursora (*hit test*).

Tipovi regiona su:

-  *pravougaoni,*
-  *eliptični,*
-  *poligonalni i*
-  *kombinovani.*

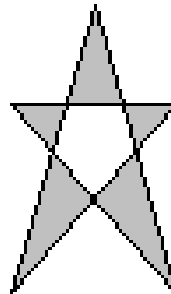
**Rectangular  
region**



**Elliptical  
region**



**Polygonal  
region**



# Kreiranje regiona

MFC klasa **CRgn** (Win32 **HRGN**)

- **BOOL** CreateRectRgn( int *x1*, int *y1*, int *x2*, int *y2*);
- **BOOL** CreateRectRgnIndirect( LPCRECT *lpRect*);
- **BOOL** CreateEllipticRgn( int *x1*, int *y1*, int *x2*, int *y2*);
- **BOOL** CreateEllipticRgnIndirect( LPCRECT *lpRect*);
- **BOOL** CreatePolygonRgn( LPPOINT *lpPoints*, int *nCount*, int *nMode* );
- **BOOL** CreatePolyPolygonRgn( LPPOINT *lpPoints*, LPINT *lpPolyCounts*, int *nCount*, int *nPolyFillMode* );
- **BOOL** CreateRoundRectRgn( int *x1*, int *y1*, int *x2*, int *y2*, int *x3*, int *y3* );
- int CopyRgn( CRgn\* *pRgnSrc* );
- **BOOL** CreateFromPath( CDC\* *pDC* );
- **BOOL** CreateFromData( const XFORM\* *lpXForm*, int *nCount*, const RGNDATA\* *pRgnData* );

# Region oblika mnogougla

- Pravljenje regiona oblika mnogougla

```
BOOL CRgn::CreatePolygonRgn(  
    LPPOINT lpPoints,  
    int nCount,  
    int nMode)
```

- ***lpPoints*** – niz tačaka koje predstavljaju temena mnogougla

- Tačke se mogu definisati nizom objekata klase **CPoint** ili strukture **POINT**

- ***nCount*** – broj temena mnogougla

- ***nMode*** – način ispune (može biti **ALTERNATE** ili **WINDING**)

# Ispuna mnogougla

Alternate mode    Winding mode



## Naizmenični način ispune (**ALTERNATE**)

Da bi se utvrdilo da li je neka tačka **T** unutar mnogougla (piksel treba obojiti), primeniti se sledeći test:

- Povuci zamišljeni zrak, u smeru X-ose, od tačke **T** ka beskonačnosti.
- Svaki put kada zrak preseče okvirnu liniju, inkrementirati brojač.

Tačka **T** **je unutar** mnogougla (piksel treba obojiti) ako je **vrednost brojača neparna**. Prolazak kroz teme se računa 2x.

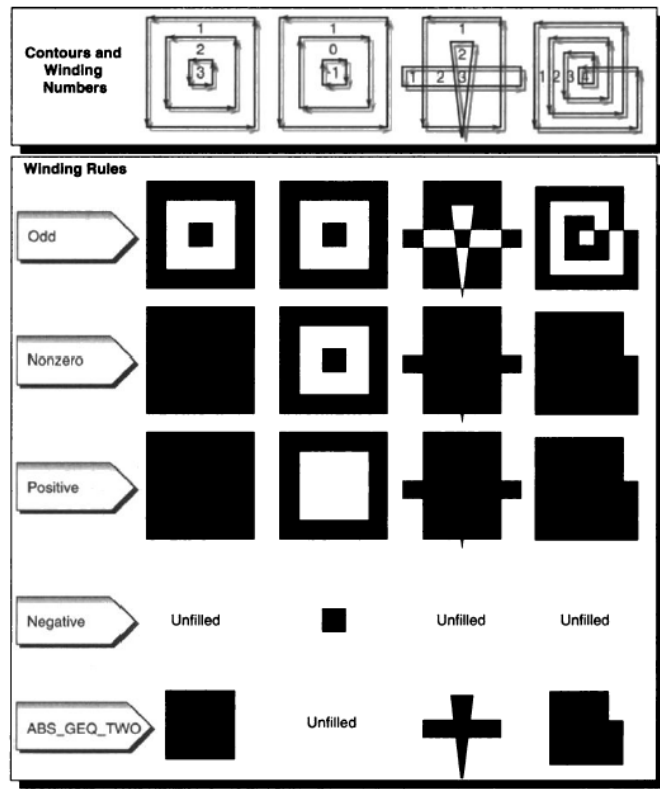
## Način ispune namotavanjem (**WINDING**)

Da bi se utvrdilo da li je neka tačka **T** unutar mnogougla (piksel treba obojiti), primeniti se sledeći test:

- Odaberi smer crtanja okvirne linije i u skladu s tim dodeli smer svakoj stranici na njoj.
- Povuci zamišljeni zrak, u smeru X-ose, od tačke **T** ka beskonačnosti.
- Svaki put kada zrak preseče stranicu čiji smer se pokalpa sa smerom porasta y ose inkrementirati brojač.
- Svaki put kada zrak preseče stranicu čiji smer se pokalpa sa smerom porasta y ose inkrementirati brojač.

Tačka **T** **je unutar** mnogougla (piksel treba obojiti) ako je **vrednost brojača različita od nule**.

# Pravila ispune namotavanjem WINDING



# Kombinovanje regiona

Pravljenje kombinovanog regiona

```
int CRgn::CombineRgn( CRgn* pRgn1, CRgn* pRgn2, int nCombineMode );
```

Kreira region i smešta ga u instancu klase **CRgn** za koju je pozvana ova metoda

- Instanca klase mora da postoji pre poziva metode i ne sme biti njen parametar

Rezultat metoda je tip napravljenog regiona, koji može biti:

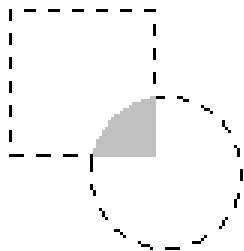
- **COMPLEXREGION** – složen
- **ERROR** – greška (novi region nije kreiran)
- **NULLREGION** – prazan
- **SIMPLEREGION** – jednostavan

Regioni *pRgn1* i *Rgn2* se kombinuju na osnovu parametra *nCombineMode*, koji može imati sledeće vrednosti:

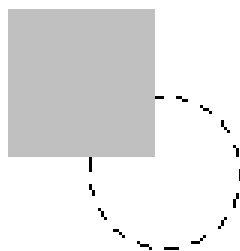
- **RGN\_AND**, **RGN\_COPY**, **RGN\_DIFF**, **RGN\_OR** ili **RGN\_XOR**

# Kombinovanje regiona

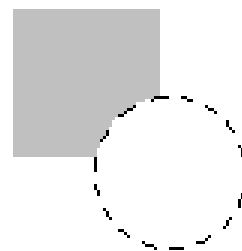
**RGN\_AND**



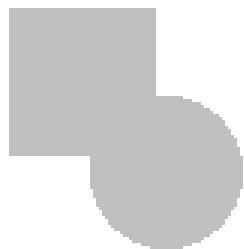
**RGN\_COPY**



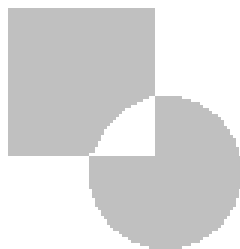
**RGN\_DIFF**



**RGN\_OR**



**RGN\_XOR**





# Odsecanje pomoću regiona

- Selekcija regiona za odsecanje u kontekst uređaja (*Device Context*)

`virtual int CDC::SelectClipRgn( CRgn* pRgn );`

`int CDC::SelectClipRgn( CRgn* pRgn, int nMode );`

- Rezultat metoda je tip seketovanog regiona

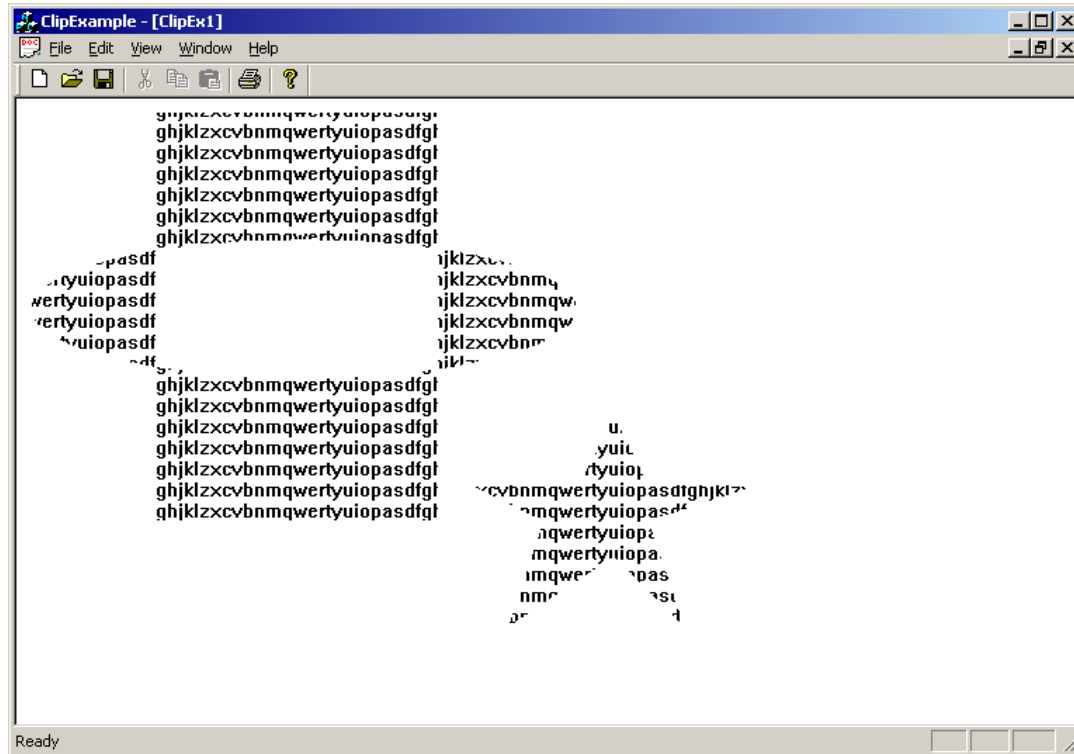
- **COMPLEXREGION, ERROR, NULLREGION, SIMPLEREGION**

- *pRgn* – region koji se selektuje.

- Ako se prosledi NULL, cela klijentska oblast prozora se selektuje (u slučaju druge funkcije, ako se prosleđuje NULL, *nMode* treba postaviti na **RGN\_COPY**)

- *nMode* – način kombinovanja dotadašnjeg sa novoselektovanim regionom

# Primer odsecanja pomoću regiona



## Primer odsecanja pomoću regiona

CString Text =

```
CString("qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbn  
mqwertyuiopasdfghjklzxcvbn");
```

```
CRgn tmpRgn, compoundRgn, rectRgn, ellipseRgn, polyRgn, *oldRgn;  
rectRgn.CreateRectRgn(100, 10, 300, 300);
```

```
ellipseRgn.CreateEllipticRgn(10, 100, 400, 200);
```

```
POINT Poly[] = {{425,225},{400,275},{325,275},{375,300},{350,375},
                {425,325},{475,375},{450,300},{525,275},{450,275}};
```

```
polyRgn.CreatePolygonRgn(Poly, 10, WINDING);
```

```
tmpRgn.CreateRectRgn(0,0,0,0);
```

```
compoundRgn.CreateRectRgn(0,0,0,0);
```

```
tmpRgn.CombineRgn(&rectRgn, &ellipseRgn, RGN_XOR);
```

```
compoundRgn.CombineRgn(&tmpRgn, &polyRgn, RGN_OR);
```

```
CRect rect;
```

```
pDC->GetClipBox(&rect);
```

```
int rgnType=pDC->SelectClipRgn(&compoundRgn);
```

```
pDC->SetTextAlign(TA_LEFT | TA_TOP);
```

```
for(i=0; i<400; i+=15)
```

```
pDC->TextOut(0, i, text);
```

```
pDC->SelectClipRgn(&rect);
```

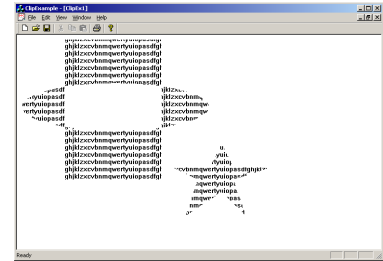
```
compoundRgn.DeleteObject();
```

```
tmpRgn.DeleteObject();
```

```
ellipseRgn.DeleteObject();
```

```
rectRgn.DeleteObject();
```

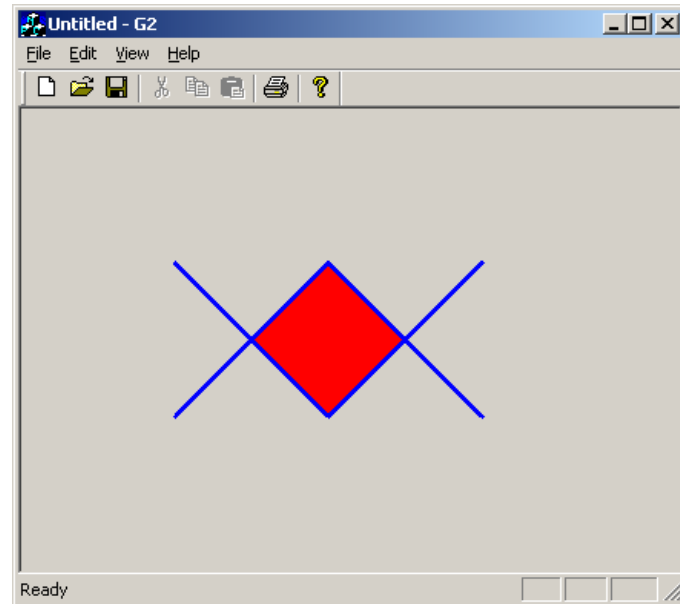
```
polyRgn.DeleteObject();
```



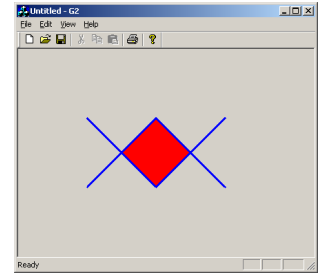
# Ispuna proizvoljne oblasti

**BOOL CDC::FloodFill( int  $x$ , int  $y$ ,  
COLORREF *crColor* );**

- Metoda ispunjava oblast počev od zadate tačke ( $x, y$ ), sve dok ne naiđe na piksele čija je boja ***crColor*** četkom kaja je trenutno izabrana u kontekstu uređaja (*Device Context*).



# Primer ispune proizvoljne oblasti



```
CBrush rbrush(RGB(255,0,0));  
CBrush* pOldBrush = pDC->SelectObject(&rbrush);  
CPen bpen(PS_SOLID, 3, RGB(0,0,255));  
CPen* pOldPen = pDC->SelectObject(&bpen);  
int count =2;  
DWORD num[] = {3,3};  
POINT pts[] = {{100,200},{200,100},{300,200},  
               {100,100},{200,200},{300,100}};
```

```
//pDC->Polyline(&pts[0],3);  
//pDC->Polyline(&pts[3],3);  
pDC->PolyPolyline(pts, num, count);  
pDC->FloodFill( 200, 150,  
               RGB(0,0,255));  
pDC->SelectObject(pOldBrush);  
pDC->SelectObject(pOldPen);  
bpen.DeleteObject();  
rbrush.DeleteObject();
```

# Metafajl

- Metafajl je kolekcija struktura koja čuva vektorsku sliku na način nezavisan od uređaja.
- Vektorska struktura omogućuje veću fleksibilnost od bitmapa (bolje skaliranje), ali se generalno meta fajlovi iscrtavaju sporije od bitmapa.
- Standardni (Windows) meta fajlovi čuvaju se u datotekama sa ekstenzijom WMF, dok unapređeni (Enhanced) format ima ekstenziju EMF.

# Koraci za kreiranje meta fajla

Kreirati **CMetaFileDC** objekat pozivom konstruktora

**CMetaFileDC::CMetaFileDC()**

Pozvati metodu koja pravi Windows Metafile Device Context i dodeljuje ga CMetaFileDC objektu.

**BOOL CMetaFileDC::Create( LPCTSTR lpszFilename = NULL )**

- Ako se prosledi ime, kreira se datoteka, u protivnom samo memorijski CMetaFileDC objekat.

Za napravljeni CMetaFileDC objekat pozivati odgovarajuće metode za crtanje.

- Standardni meta fajlovi ne podržavaju krive, putanje i transformacione funkcije

Kada se završi sa crtanjem, pozvati metodu kojom se zatvara CMetaFileDC objekat i vraća handle na formirani metafile.

**HMETAFILE CMetaFileDC::Close( )**

Na kraju osloboditi CMetaFileDC objekat.

**BOOL CDC::DeleteDC()**

# Učitavanje, kopiranje, crtanje i brisanje meta fajla

- Učitavanje standardnog meta fajla (WMF) iz datoteke čiji se naziv prosleđuje metodi **GetMetaFile** koja vraća *handle* na učitani meta fajl

**HMETAFILE GetMetaFile( LPCTSTR *lpzMetaFile*)**

- Kopiranje meta fajla zadatog pomoću *handle*-a **hmfSrc** u datoteku zadatog naziva i u memorijski meta fajl čiji se *handle*-a vraća

**HMETAFILE CopyMetaFile( HMETAFILE *hmfSrc*, LPCTSTR *lpzFile*)**

- Crtanje meta fajla na uređaju na osnovu *handle*-a meta fajla

**BOOL CDC::PlayMetaFile( HMETAFILE *hmf*)**

- Po završetku korišćenja, meta fajl treba obrisati prosleđivanjem njegovog *handle*-a

**BOOL DeleteMetaFile( HMETAFILE *hmf*)**

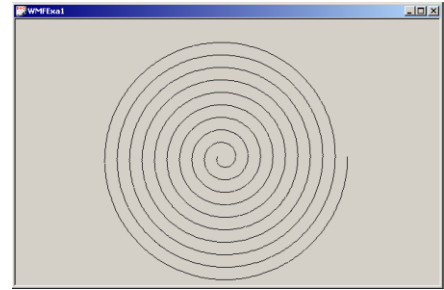


# Primer korišćenja metafajla

```
CString WMFname = CString("metafile.wmf");
double pi = 3.1415926535897932384626433832795;
CMetaFileDC MetaDC;
MetaDC.Create("Proba.wmf");
HMETAFILE MF, NewMF;
int x = 300, y = 200;
MetaDC.MoveTo(x, y);
for (double angle = 0; angle <= 10*2*pi; angle += step) {
    double r = 180 * angle / (10*2*pi);
    double step = 3/(r+1);
    x = (int)(300.0+r*cos(angle+step) +0.5);
    y = (int)(200.0-r*sin(angle+step)+0.5);
    MetaDC.LineTo(x, y);
}
```

```
MF = MetaDC.Close();
NewMF =
CopyMetaFile(MF,WMFname.GetBuffer(WMFname.GetLength()));
DeleteMetaFile(MF);
DeleteMetaFile(NewMF);
MetaDC.DeleteDC();

HMETAFILE Meta =
    GetMetaFile(WMFname.GetBuffer(WMFname.GetLength()));
PlayMetaFile(pDC->m_hDC,Meta);
DeleteMetaFile(Meta);
```



# Koraci za kreiranje unapređenog meta fajla

Kreirati **CMetaFileDC** objekat pozivom konstruktora

**CMetaFileDC::CMetaFileDC()**

Pozvati metodu koja pravi Windows Metafile Device Context i dodeljuje ga CMetaFileDC objektu.

**BOOL CMetaFileDC::CreateEnhanced( CDC\* pDCRef, LPCTSTR lpszFileName,  
LPCRECT lpBounds, LPCTSTR lpszDescription )**

- **DCRef** – referentni DC
- **lpszFileName** – naziv datoteke u koju se smešta (može biti NULL)
- **lpBounds** – okvirni pravougaonik u HIMERIC jedinicama (0.01mm je logicka jedinica)
- **lpszDescription** – opis (naziv slike, aplikacija koja je kreirala i sl.)

Za napravljeni CMetaFileDC objekat pozivati odgovarajuće metode za crtanje.

Kada se završi sa crtanjem, pozvati metodu kojom se zatvara CMetaFileDC objekat i vraća *handle* na formirani metafile.

**HENHMETAFILE CMetaFileDC::CloseEnhanced( )**

Na kraju osloboditi CMetaFileDC objekat.

**BOOL CDC::DeleteDC()**

# Učitavanje, kopiranje, crtanje i brisanje unapređenog meta fajla

- Učitavanje unapređenog meta fajla (WMF) iz datoteke čiji se naziv prosleđuje metodi `GetMetaFile` koja vraća *handle* na učitani meta fajl

`HMETAFILE GetEnhMetaFile( LPCTSTR lpaszMetaFile)`

- Kopiranje meta fajla zadatog pomoću *handle*-a *hmfSrc* u datoteku zadatog naziva i u memorijski meta fajl čiji se *handle*-a vraća

`HMETAFILE CopyEnhMetaFile(HENHMETAFILE hmfSrc, LPCTSTR lpaszFile)`

- Crtanje unapređenog meta fajla na uređaju na osnovu *handle*-a meta fajla i okvirnog pravougaonika u kome treba iscrtati meta fajl

`BOOL CDC::PlayMetaFile(HENHMETAFILE hMF, LPCRECT lpBounds)`

- Po završetku korišćenja, meta fajl treba obrisati prosleđivanjem njegovog *handle*-a

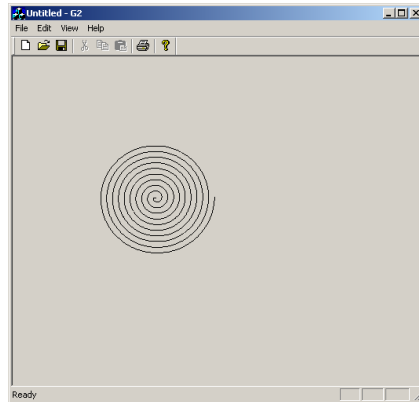
`BOOL DeleteEnhMetaFile(HENHMETAFILE hmf)`

# Primer korišćenja unapređenog meta fajla

```
CMetaFileDC MetaDC;  
BOOL b = MetaDC.CreateEnhanced( pDC, "EhProba.emf",  
    CRect(0,0,15000,15000), "AksPaint" );  
  
ENHMETAFILE MF;  
int x = 200, y = 200;  
MetaDC.MoveTo(x, y);  
for (double angle = 0; angle <= 10*2*pi; angle += step) {  
    double r = 180 * angle / (10*2*pi);  
    double step = 3/(r+1);  
    x = (int)(300.0+r*cos(angle+step) +0.5);  
    y = (int)(200.0-r*sin(angle+step)+0.5);  
    MetaDC.LineTo(x, y);  
}
```

```
EMF = MetaDC.CloseEnhanced();  
MetaDC.DeleteDC();
```

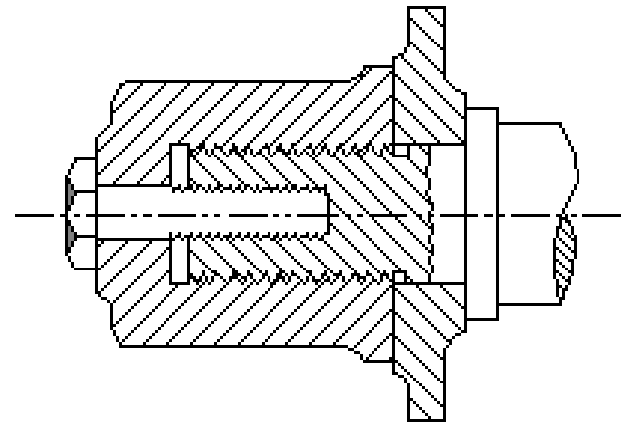
```
ENHMETAHEADER emfHeader;  
GetEnhMetaFileHeader(MF, sizeof(ENHMETAHEADER), &emfHeader);  
int nWidth = emfHeader.rcIBounds.right - emfHeader.rcIBounds.left;  
int nHeight = emfHeader.rcIBounds.bottom - emfHeader.rcIBounds.top;  
int factor = 2;  
PlayEnhMetaFile(pDC->m_hDC,MF,  
    CRect(100, 100, 100+nWidth/factor, 100+nHeight/factor));  
DeleteEnhMetaFile(MF);
```



# Putanja

Putanju čine jedna ili više figura koje mogu biti:

- ispunjene,
- uokvirene ili
- i ispunjene i uokvirene



Imaju široku primenu jer dozvoljavaju kombinovanje krivih i pravih segmenata, kao i primenu stilova na spojeve linijskih segmenata.

# Karakteristike putanje

- Putanja je jedan od tipova GDI objekata koji se može selektovati u DC.
- Za razliku od nekih drugih tipova GDI objekata (olovaka, četki i fontova), koji se formiraju zajedno sa svakim novim DC-jem (podrazumevani objekti), ne postoji podrazumevana putanja.
- Da bi se kreirala putanja i selektovala u DC, najpre je potrebno pozvati primitive (funkcije) koje je čine. One se navode u delu koda između poziva funkcija **BeginPath** i **EndPath**.
- Putanja ne može postojati nezavisno od DC i postoji samo jedna putanja
  - Poziv **BeginPath** odbacuje prethodnu putanju iz DC-a
  - Poziv **EndPath** selektuje putanju u DC
- Dozvoljene primitive (funkcije) za pravljanje putanje su:
  - AngleArc, LineTo, Polyline, Arc, MoveToEx, PolylineTo, ArcTo, Pie, PolyPolygon, Chord, PolyBezier, PolyPolyline, CloseFigure, PolyBezierTo, Rectangle, Ellipse, PolyDraw, RoundRect, ExtTextOut, Polygon, TextOut

# Karakteristike putanje

## ■ Definisanje putanje

- početak: **BOOL CDC::BeginPath( )**
- crtanje primitiva: **Arc, LineTo, Pie, ...**
- završetak: **BOOL CDC::EndPath( );**

## ■ Iscrtavanje

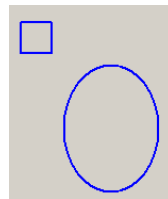
- okvirom: **BOOL CDC::StrokePath( )**
- ispunom: **BOOL CDC::FillPath( )**
- okvirom i ispunom: **BOOL CDC::StrokeAndFillPath( )**

# Primer korišćenja putanje

```
pDC->BeginPath();  
    pDC->Rectangle(20,20,50,50);  
    pDC->Ellipse(CRect(60,60,150,180));  
pDC->EndPath();  
  
CPen pen;  
pen.CreatePen(PS_SOLID, 2, RGB(0,0,255));  
CPen* pOldPen = pDC->SelectObject(&pen);  
CBrush brush;  
brush.CreateHatchBrush(HS_BDIAGONAL, RGB(255,0,255));  
CBrush* pOldBrush = pDC->SelectObject(&brush);
```

```
// pDC->StrokePath();  
// pDC->FillPath();  
pDC->StrokeAndFillPath();
```

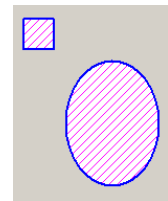
```
pDC->SelectObject(pOldBrush);  
pDC->SelectObject(pOldPen);
```



StrokePath



FillPath



StrokeAndFillPath



## Odsecanje pomoću putanja

- Postavljanje trenutno selektovane putanje u DC

**BOOL CDC::SelectClipPath( int *nMode* )**

- ***nMode*** – definiše način kombinovanja trenutnog selektovanog regiona i trenutnog selektovane putanje:

- **RGN\_AND, RGN\_COPY, RGN\_DIFF, RGN\_OR, RGN\_XOR**

**Clip Path** **Clip Path**

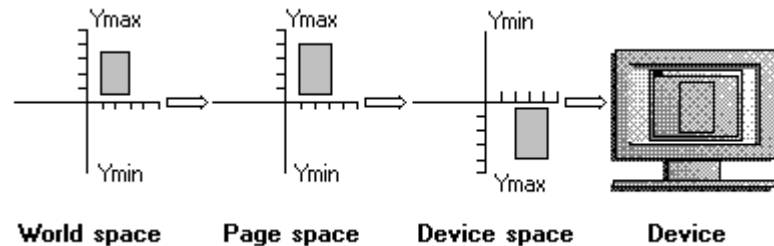
# Koordinatni prostori i transformacije

- Koordinatni prostori i transformacije koriste se za skaliranje, rotaciju, translaciju, ...
- Koordinatni prostor je ravan prostor u kome se koristi Dekartov koordinatni sistem.
- Postoje 4 koordinatna prostora:
  - world
  - page
  - device i
  - physical device
- Transformacija je algoritam po kome se menja veličina, orijentacija, položaj i oblik objekata.

# Transformacija prostora

■ Koordinatni prostor predstavlja sredstvo da se specificira lokacija svake tačke u ravni. Koriste se:

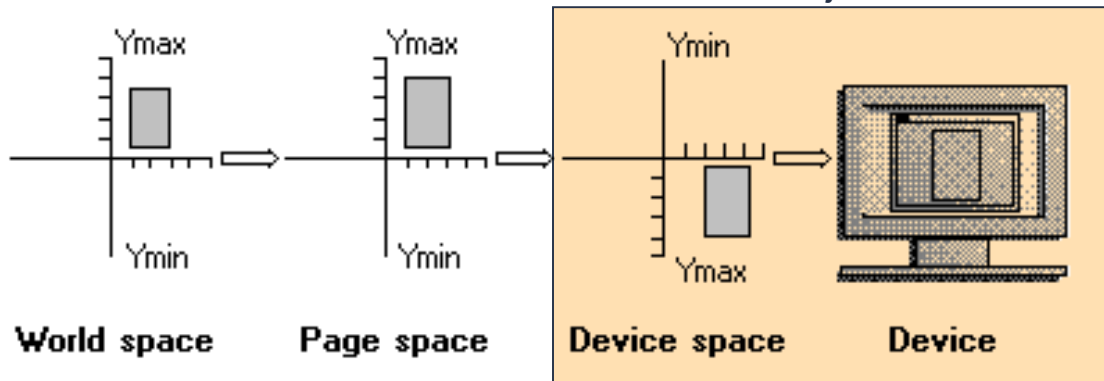
- World – dimenzija  $2^{32} \times 2^{32}$
- Page – dimenzija  $2^{32} \times 2^{32}$
- Device – dimenzija  $2^{27} \times 2^{27}$
- Physical device – dimenzije zavise od uređaja (ekran, deo prozora, papir štampača i sl.)



■ Transformacija određuje način preslikavanja piksela iz jednog prostora u drugi.

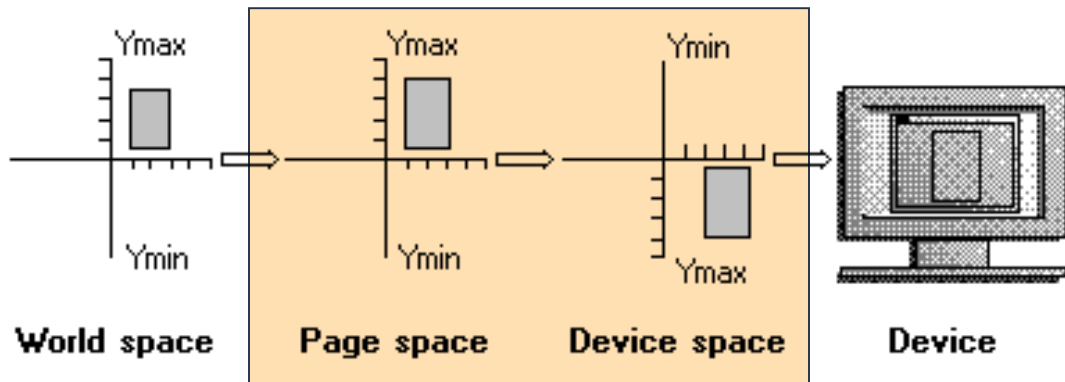
# Preslikavanje *Device* u *Physical Device* prostor

- Izvršava se samo translacija
- Ovom transformacijom upravlja USER komponenta Windows-a za upravljanje prozorima
- Jedina uloga ove transformacije je da se početak *device* prostora mapira na početak (tj. odgovarajući piksel) *physical device* prostora
- Programski se ne može uticati na ovu transformaciju



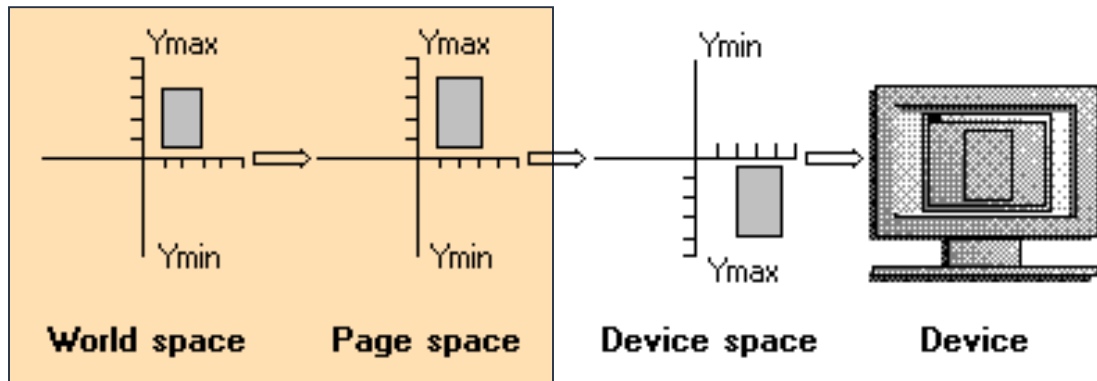
# Preslikavanje *Page* prostora u *Device* prostor

- Definiše mod mapiranja za sve grafičke primitive određenog DC-ja
- Podržane su sledeće transformacije: translacija, skaliranje i refleksija (promena orijentacije osa)
- Definiše se postavljanjem odgovarajućeg moda mapiranja i koordinatnog početka (videti slajdove sa prvog termina)

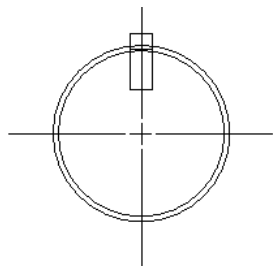


# Preslikavanje *World* prostora u *Page* prostor

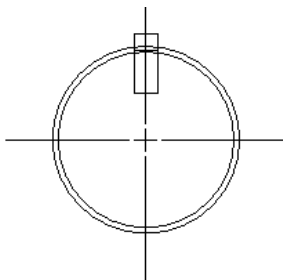
- Omogućava ono što se u drugim API-jima naziva preslikavanje lokalnih u svetske koordinate
- Omogućuje sledeće transformacije:
  - translaciju, skaliranje, rotaciju, smicanje, refleksiju



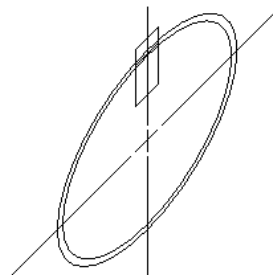
# Osnovne transformacije



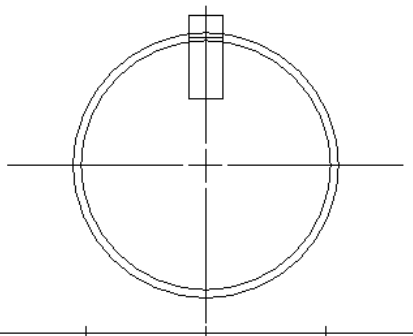
**Original View**



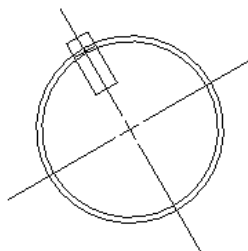
**Translated View**



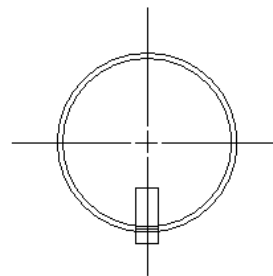
**Sheared View**



**Scaled View**

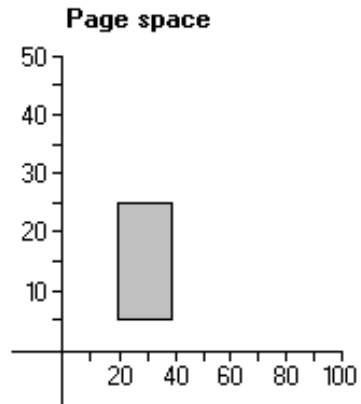
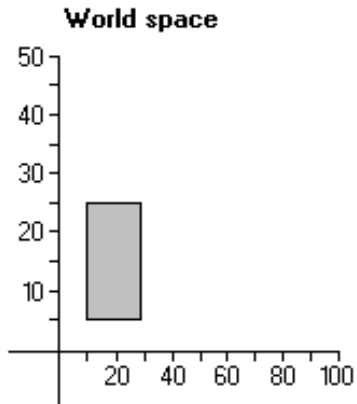


**Rotated View**



**Reflection**

# Translacija



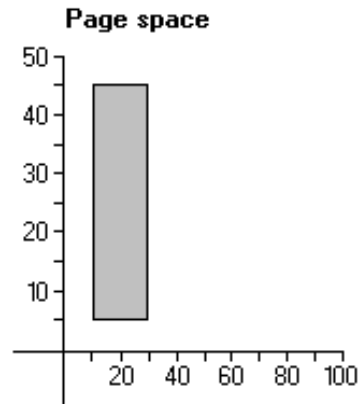
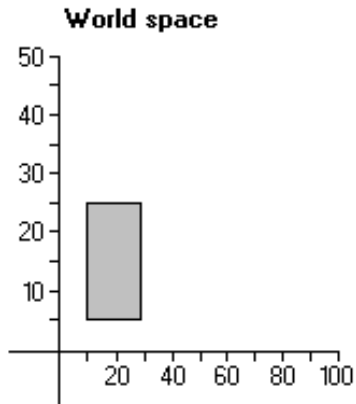
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} Dx \\ Dy \\ 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$x' = x + Dx$$

$$y' = y + Dy$$



# Skaliranje

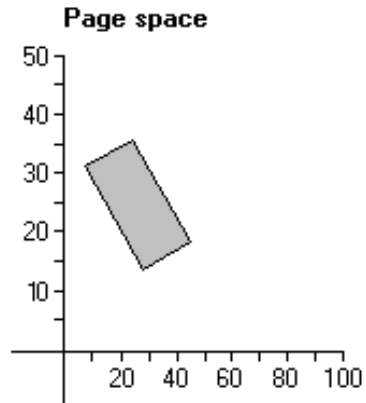
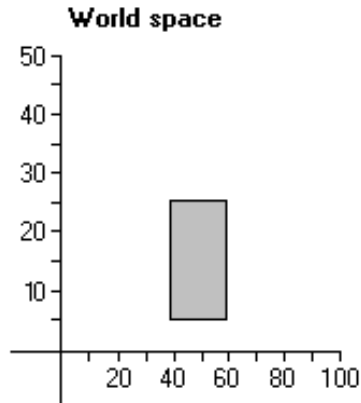


$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$x' = Sx \cdot x$$

$$y' = Sy \cdot y$$

# Rotacija

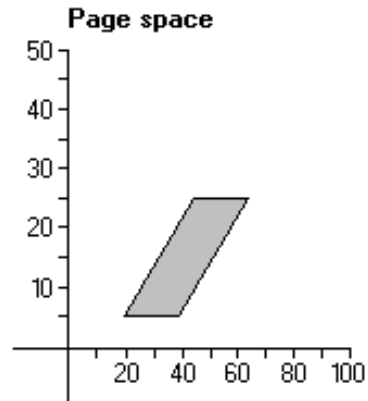
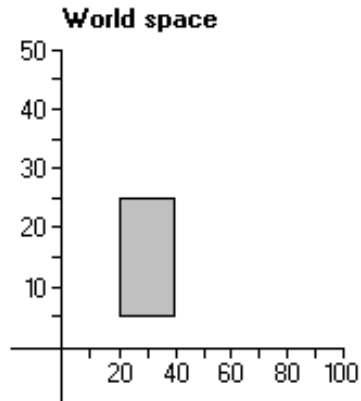


$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$x' = \cos(\alpha) \cdot x - \sin(\alpha) \cdot y$$

$$y' = \sin(\alpha) \cdot x + \cos(\alpha) \cdot y$$

# Smicanje



$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & Sx & 0 \\ Sy & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$x' = x + Sx \cdot y$$

$$y' = Sy \cdot x + y$$

# Refleksija

Horizontalna refleksija

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$x' = -x$$

$$y' = y$$

Vertikalna refleksija

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$x' = x$$

$$y' = -y$$

# Struktura matrice transformacije

- XFORM struktura
- Definiše transformaciju koja se može koristiti pri preslikavanju iz world prostora u page prostor

```
typedef struct _XFORM {  
    FLOAT eM11;  
    FLOAT eM12;  
    FLOAT eM21;  
    FLOAT eM22;  
    FLOAT eDx;  
    FLOAT eDy;  
} XFORM;
```

## XFORM atributi

operacija	eM11	eM12	eM21	eM22
rotacija	cos	sin	-sin	cos
skaliranje	horizontalno skaliranje	0	0	vertikalno skaliranje
iskišenje	1	horizontalna konstanta proporcionalnosti	vertikalna konstanta proporcionalnosti	1
refleksija	horizontalna refleksiona komponenta	0	0	vertikalna refleksiona komponenta

eDx – horizontalna komponenta translacije

eDy – vertikalna komponenta translacije

# Metoda SetWorldTransform

■ Postavljanje transformacije

`BOOL SetWorldTransform( HDC hdc, CONST XFORM *lpXform )`

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} x & y & 1 \end{pmatrix} \begin{pmatrix} eM11 & eM12 & 0 \\ eM21 & eM22 & 0 \\ eDx & eDy & 1 \end{pmatrix} \quad \begin{aligned} x' &= x * eM11 + y * eM21 + eDx \\ y' &= x * eM12 + y * eM22 + eDy \end{aligned}$$

■ Pre poziva ove funkcije treba postaviti grafički mod na **GM\_ADVANCED**

`SetGraphicsMode(pDC->m_hDC, GM_ADVANCED)`

# Primer postavljanja transformacije

```
int prevMode = SetGraphicsMode( pDC->m_hDC, GM_ADVANCED);  
DWORD dw = GetLastError();
```

```
XFORM Xform, XformOld;
```

```
BOOL b = GetWorldTransform(pDC->m_hDC, &XformOld);
```

```
b = SetWorldTransform(pDC->m_hDC, &Xform);
```

```
dw = GetLastError();
```

```
// Iscrtavanje
```

```
b = SetWorldTransform(pDC->m_hDC, &XformOld);
```

```
SetGraphicsMode(pDC->m_hDC, prevMode);
```

Xform.eM11 = (FLOAT) 1.0;

Xform.eM12 = (FLOAT) 0.0;

Xform.eM21 = (FLOAT) 0.0;

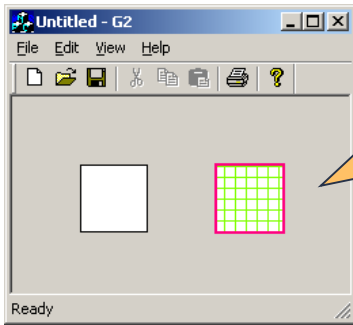
Xform.eM22 = (FLOAT) 1.0;

Xform.eDx = (FLOAT) 50.0;

Xform.eDy = (FLOAT) 0.0;

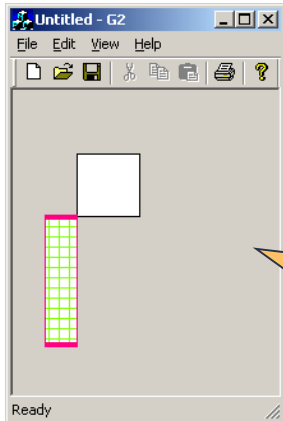


# Primer translacije i skaliranja



## Translacija

Xform.eM11 = 1.0;  
Xform.eM12 = 0.0;  
Xform.eM21 = 0.0;  
Xform.eM22 = 1.0;  
Xform.eDx = **100.0**;  
Xform.eDy = 0.0;

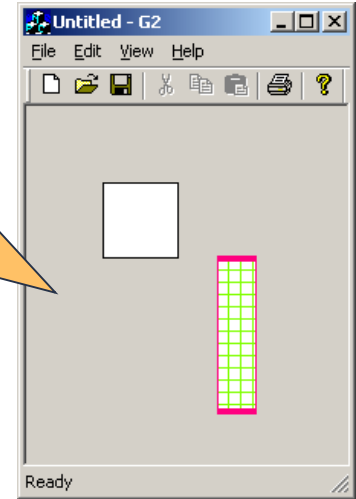


## Skaliranje

Xform.eM11 = **0.5**;  
Xform.eM12 = 0.0;  
Xform.eM21 = 0.0;  
Xform.eM22 = **2.0**;  
Xform.eDx = 0.0;  
Xform.eDy = 0.0;

## Skaliranje i Translacija

Xform.eM11 = **0.5**;  
Xform.eM12 = 0.0;  
Xform.eM21 = 0.0;  
Xform.eM22 = **2.0**;  
Xform.eDx = **100.0**;  
Xform.eDy = 0.0;



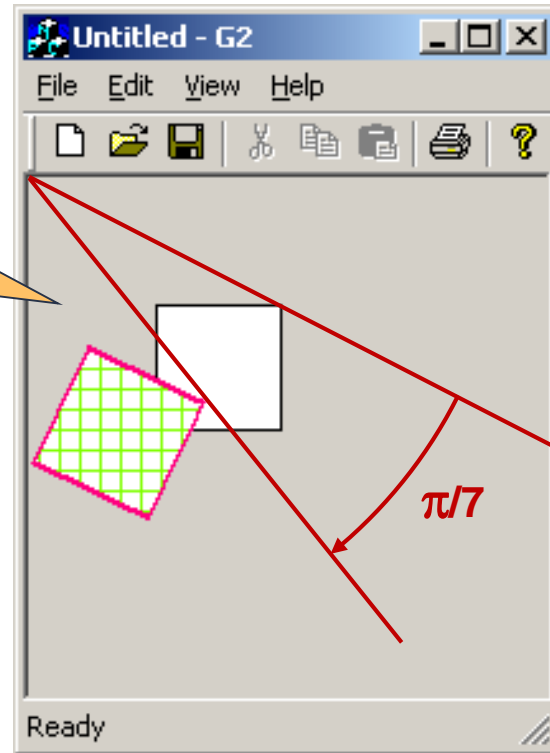
pDC->Rectangle(50,50,100,100);

# Primer rotacije

## Rotacija

Xform.eM11 = **cos(pi/7.0)**;  
Xform.eM12 = **sin(pi/7.0)**;  
Xform.eM21 = **-sin(pi/7.0)**;  
Xform.eM22 = **cos(pi/7.0)**;  
Xform.eDx = 0.0;  
Xform.eDy = 0.0;

pDC->Rectangle(50,50,100,100);



# Izmena transformacije

Metod za izmenu trenutne transformacije DC-a

```
BOOL ModifyWorldTransform( HDC hdc,  
                           CONST XFORM *lpXform, DWORD iMode);
```

*lpXform* – struktura sa transformacionom matricom kojom treba izmeniti trenutnu transformaciju DC-a

*iMode* – način na koji treba izmeniti trenutnu transformaciju DC-a

- **MWT\_IDENTITY** – resetuje transformaciju (učitava se jedinična matrica i ignoriše se prosleđena transformaciona matrica)
- **MWT\_LEFTMULTIPLY** – množi trenutnu transformacionu matricu sa prosleđenom sa leve strane (prosleđena matrica je levi operand u množenju)
- **MWT\_RIGHTMULTIPLY** – množi trenutnu transformacionu matricu sa prosleđenom sa desne strane (prosleđena matrica je desni operand u množenju)

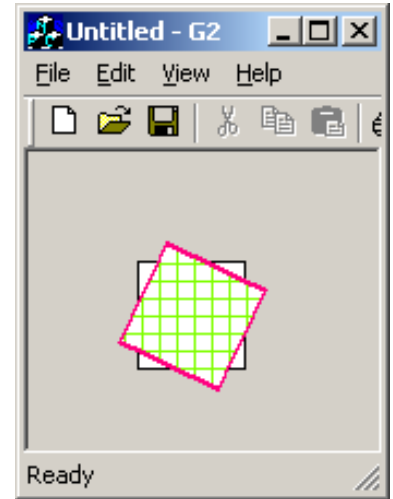
# Primer izmene transformacija

```
Xform.eM11 = 1.0;  
Xform.eM12 = 0.0;  
Xform.eM21 = 0.0;  
Xform.eM22 = 1.0;  
Xform.eDx = -75.0;  
Xform.eDy = -75.0;
```

```
b = SetWorldTransform(pDC->m_hDC,  
&Xform);
```

```
Xform.eM11 = cos (pi/7.0);  
Xform.eM12 = sin (pi/7.0);  
Xform.eM21 = -sin (pi/7.0);  
Xform.eM22 = cos (pi/7.0);  
Xform.eDx = 75.0;  
Xform.eDy = 75.0;
```

```
b = ModifyWorldTransform(pDC->m_hDC,  
&Xform, MWT_RIGHTMULTIPLY);  
pDC->Rectangle(50,50,100,100);
```



# Kombinovanje transformacija

- Metod za kombinovanje matrica transformacije

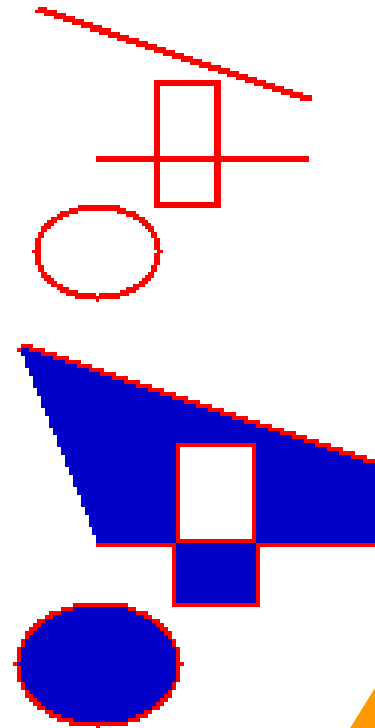
```
BOOL CombineTransform( LPXFORM lpxformResult,  
                        CONST XFORM *lpxform1,  
                        CONST XFORM *lpxform2 )
```

- lpxformResult* – pokazivač na strukturu sa transformacionom matricom koja prihvata kombinovanu transformaciju (rezultujuća matrica)
- lpxform1* – pokazivač na strukturu sa transformacionom matricom prve transformacije (leva matrica)
- lpxform2* – pokazivač na strukturu sa transformacionom matricom druge transformacije (desna matrica)

**GDI+**

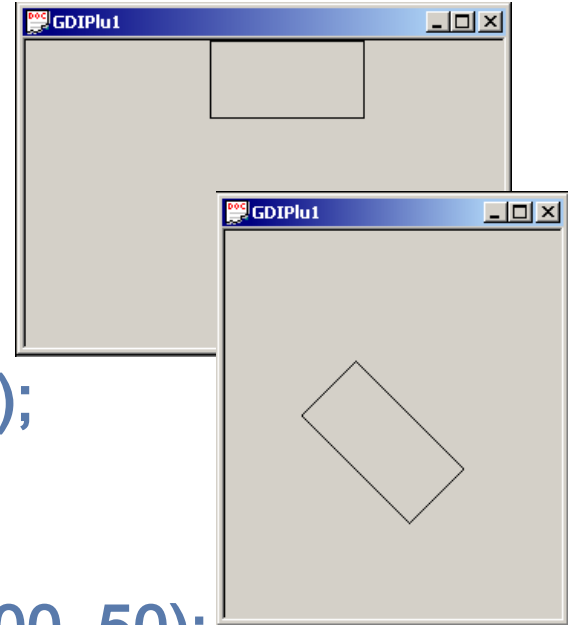
# Putanja

```
GraphicsPath path;  
Pen pen(Color(255, 255, 0, 0), 2);  
SolidBrush brush(Color(255, 0, 0, 200));  
path.AddLine(10, 10, 100, 40);  
path.AddLine(100, 60, 30, 60);  
path.AddRectangle(Rect(50, 35, 20, 40));  
path.AddEllipse(10, 75, 40, 30);  
graphics.DrawPath(&pen, &path);  
graphics.FillPath(&brush, &path);
```



# Kombinovanje transformacije

```
Matrix transformMatrix;  
transformMatrix.Rotate(45.0f);  
graphics.SetTransform(&transformMatrix);  
Pen pen(Color(255, 0, 0, 0));  
graphics.DrawRectangle(&pen, 120, 0, 100, 50);
```



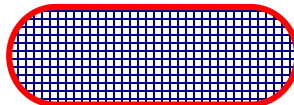


# Kombinovanje transformacije

```
Matrix matrix;  
matrix.Reset(); // LoadIdentity  
matrix.Translate(-X, -Y, MatrixOrderAppend);  
                // MatrixOrderAppend, MatrixOrderPrepend  
matrix.Rotate(alpha, MatrixOrderAppend);  
matrix.Translate(X, Y, MatrixOrderAppend);  
graphics.SetTransform(&matrix);
```

# Domaći zadaci

- Nacrtati proizvoljan simbol (npr. simbol PTT-a, treba da sadrži i neki tekst) i snimiti ga kao unapređeni metafajl (EMF).
- Učitati EMF nacrtan u prethodnoj tački, kao i proizvoljan EMF nacrtan u CorelDraw-u, odrediti njihove aspekte (odnos visine i širine) i iscrtati ih proporcionalno, sa visinom od 50 piksela i zarotirane za 450.
- Napraviti clip-region dobijen kao unija elipse i kvadrata koji se delimično preklapaju i ispuniti ga uniformnom žutom bojom.
- Nacrtati putanju prikazanu na slici (sastavljenu od dva luka i dva prava linijska segmenta) i ispuniti ga proizvoljnom šrafurom.



# Pitanja

