Računarska grafika
(2OER7O02)

Device Context (DC)

Auditivne vežbe

Crtanje nezavisno od uređaja

Sofrverska podrška za crtanje nezavisno od uređaja ostvarena je kroz dva DLLa:

- **Gdi.dll** (*graphics device interface*) i
- **drajvera uređaja** (zavisi od konkretnog izlaznog uređaja, npr. Epson9.dll)

Aplikacija mora da obavesti GDI koji drajver treba da učita i da pripremi uređaj za sam proces crtanja (postavlja debljinu i boju linija, četku, paletu, font, *clip*-region, itd.). Ovi zadaci se ostvaruju kreiranjem i upravljanjem **device context**-om (DC).

Device Context

Ono što je slikarima platno, to je za Windows programere *Device Context*, ali je DC i mnogo više od toga.

Device Context (DC) je struktura podataka koja definiše skup grafičkih objekata i njima dodeljenih atributa, kao i grafičke modove koji utiču na prikaz.

GDI obezbeđuje 4 tipa DC-ja:

- *Display* - prikaz na video izlazu (monitoru),
- *Memory* – DC kompatibilan sa nekim drugim uređajem,
- *Information* – daje informacije o nekom uređaju, *i*
- *Printer* - prikaz na hard-copy uređajima (štampači,ploteri,...).

Kako pribaviti DC prozora

- Preuzimanjem *handle*-a na klijentsku površinu prozora (Win32):

  HDC GetDC(HWND *hWnd* // handle to a window **);***

- Preuzimanjem *handle*-a na klijentsku površinu prozora (MFC):

  CDC* CWnd::GetDC( ); *!

* Nakon završetka rada osloboditi sa **ReleaseDC(),** osim u specijalnim slučajevima!

! U MFC-u, funkcije za rad sa DC-jem inkapsulirane su u klasi **CDC**.

U MFC Document-View aplikacijama

```
void CP1View::OnDraw(CDC* pDC)
{
        CP1Doc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);
        // TODO: add draw code for native data here
}
```

Komponente DC-ja

- Atributi
- Grafički modovi
- Grafički objekti

Atributi DC-ja
Boje teksta i pozadine

- **Text color** – boja kojom se ispisuje tekst, ali ne samo to (*foreground color*)
  - **COLORREF CDC::GetTextColor( );**
  - **COLORREF CDC::SetTextColor(COLORREF crColor);**

- **Background color** – boja pozadine (iza teksta, ali i bilo kog drugog objekta)
  - **COLORREF CDC::GetBkColor( );**
  - **COLORREF CDC::SetBkColor(COLORREF crColor);**

Definisanje boje

- COLORREF – 32bit vrednost koja definiše RGB boju u formatu **0x00bbggrr**
- RGB – makro koji olakšava kreiranje boja

COLORREF RGB(

BYTE *bRed*,

BYTE *bGreen*,

BYTE *bBlue*);

RGB(255,0,0) **RGB(0,255,0)**
RGB(0,0,255) **RGB(255,255,0)**
RGB(255,0,255)

```
#define RGB(r, g ,b)
((DWORD) (((BYTE) (r) | ((WORD) (g) << 8)) |  (((DWORD) (BYTE) (b)) << 16)))
```

Grafički modovi

Win32 API podržava 5 grafičkih modova:

- ***background* mod** – definiše način mešanja pozadinske boje sa drugim bojama bitmapa ili teksta,
- ***drawing* mod** - definiše način mešanja *foreground* boje sa drugim bojama olovaka, četki, bitmapa ili teksta,
- ***mapping* mod** – definiše kako se koordinate mapiraju iz logičkih u fizičke,
- ***polygon-fill* mod** – definiše kako se šablon četke koristi za ispunu kompleksnih regiona i
- ***stretching* mod** – definiše kako se boje bitmape mešaju sa drugim bojama u prozoru kada je bitmapa kompresovana ili smanjena

Background mod

- Definiše način mešanja pozadinske boje sa drugim bojama bitmapa ili teksta
- Može biti **OPAQUE** ili **TRANSPARENT**
- Funkcije za proveru i postavku moda:
  - int CDC::GetBkMode( );
  - int CDC::SetBkMode(int *iBkMode*);

  - Obe vraćaju prethodno postavljeni mod

Drawing mod – 1

Definiše način mešanja *foreground* boje sa drugim bojama olovaka, četki, bitmapa ili teksta, i može biti:

- **R2_BLACK** - Pixel je uvek 0.
- **R2_COPYPEN - Pixel je boje olovke.**
- **R2_MASKNOTPEN** - Pixel is a combination of the colors common to both the screen and the inverse of the pen.
- **R2_MASKPEN** - Pixel is a combination of the colors common to both the pen and the screen.
- **R2_MASKPENNOT** - Pixel is a combination of the colors common to both the pen and the inverse of the screen.
- **R2_MERGENOTPEN** - Pixel is a combination of the screen color and the inverse of the pen color.
- **R2_MERGEPEN** - Pixel is a combination of the pen color and the screen color.

Drawing mod – 2

- **R2_MERGEPENNOT** - Pixel is a combination of the pen color and the inverse of the screen color.
- **R2_NOP** - Pixel remains unchanged.
- **R2_NOT** - Pixel is the inverse of the screen color.
- **R2_NOTCOPYPEN** - Pixel is the inverse of the pen color.
- **R2_NOTMASKPEN** -Pixel is the inverse of the R2_MASKPEN color.
- **R2_NOTMERGEPEN** -Pixel is the inverse of the R2_MERGEPEN color.
- **R2_NOTXORPEN** - Pixel is the inverse of the R2_XORPEN color.
- **R2_WHITE** - Pixel is always 1.
- **R2_XORPEN** -Pixel is a combination of the colors in the pen and in the screen, but not in both.

Drawing mod – funkcije

Funkcije za očitavanje trenutno selektovanog moda i postavljanje novog moda crtanja:

- int CDC::GetROP2( );
- int CDC::SetROP2(int *fnDrawMode*);

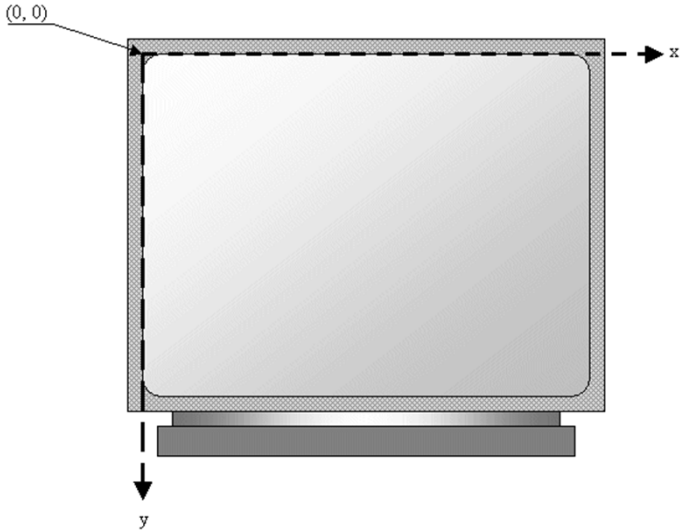
R2_NOTXORPEN

Mapping mod

- Definiše kako se koordinate mapiraju iz logičkih u fizičke.
- U **fizičkom** koordinatnom sistemu jedinice mere su uvek pikseli, y-osa raste naniže, a koordinatni početak je u gornjem levom uglu.
- Sve koordinate koje se prosleđuju funkcijama za crtanje zadaju se u **logičkom** koordinatnom sistemu.
- Inicijalno se fizički i ligički koordinatni sistemi poklapaju.

Mapping mod – MM_TEXT

Podrazumevani mod mapiranja, u kome jednoj logičkoj jedinici odgovara jedan piksel izlaznog uređaja i koordinatne ose fizičkog i logičkog sistema se poklapaju.



Mapping mod – 2

Grupa modova usklađenih sa jedinicama dužine koje se koriste u svakodnevnom životu:

- **MM_TEXT** - 1 logika jedinica mapira se u 1 piksel. X raste na desno, Y naniže.
- **MM_HIENGLISH** – 1 logika jedinica mapira se u 0.001 inča. X raste na desno, Y naviše.
- **MM_HIMETRIC** - 1 logika jedinica mapira se u 0.01 mm. X raste na desno, Y naviše.
- **MM_LOENGLISH** - 1 logika jedinica mapira se u 0.01 inča. X raste na desno, Y naviše.
- **MM_LOMETRIC** - 1 logika jedinica mapira se u 0.1 mm. X raste na desno, Y naviše.
- **MM_TWIPS** - 1 logika jedinica mapira se u 1 **twip** (1/20 tipografske tačke ili 1/1440 inča). X raste na desno, Y naviše.

Mapping mod – 3
Proizvoljne veličine logičkih jedinica

- **MM_ISOTROPIC** - logičke jedinice mapiraju se u proizvoljne fizičke, ali sa ograničenjem da je jednako skaliranje po obe ose. (Kada se postavi *windows extent*, *viewport* automatski podešava da zadrži izotropnost- ne poštuju se u potpunosti postavke *viewport* i *window extent*-a).
- **MM_ANISOTROPIC** – logičke jedinice mapiraju se u proizvoljne fizičke.

Mapping mod – postavljanje moda

Funkcije za očitavanje trenutno selektovanog moda i postavljanje novog moda mapiranja:

- int CDC::GetMapMode( );
- int CDC::SetMapMode(int *fnMapMode*);

Mapping mod – funkcije preslikavanja

Postavljanje mapiranja fizičkih u logičke jedinice. Koriste se samo kod MM_ISOTROPIC i MM_ANISOTROPIC modova, uvek u paru, i definišu koliko se logičkih jedinica preslikava u jednu fizičku i obrnuto.

- BOOL CDC::SetWindowExt(
  int *nXExtentW*, // new horizontal window extent
  int *nYExtentW*, // new vertical window extent);
- BOOL CDC:: SetViewportExt(
  int *nXExtentV*, // new horizontal viewport extent
  int *nYExtentV*, // new vertical viewport extent);

Značenje: ***nXExtentV*** **fizičkih** jedinica (piksela) odgovara ***nXExtentW*** **logičkih** jedinica po X-osi, a ako se znaci razlikuju X-osa raste ulevo. Isto važi i za Y-osi (ako se razlikuju znaci, Y-osa raste naviše).

Mapping mod – funkcije postavljanja koordinatnog početka

Koje će logičke koordinate imati tačka sa fizičkim koordinatama (0,0), zadaje se pomoću funkcije:

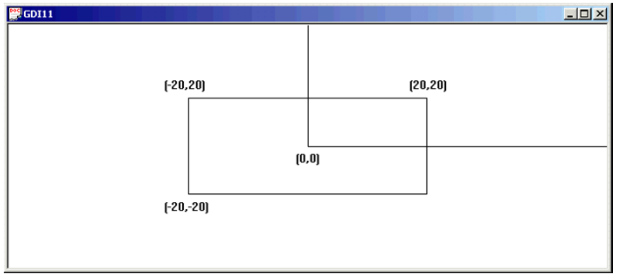
- BOOL CDC::SetWindowOrg(
  int *X*, // new logical x-coordinate of window origin
  int *Y* // new logical y-coordinate of window origin);

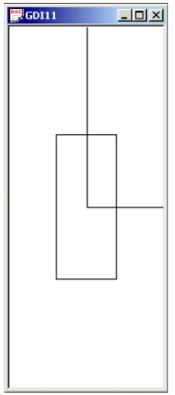
Koje će fizičke koordinate imati tačka sa logičkim koordinatama (0,0), tj. položaj u odnosu na gornji levi ugao, zadaje se pomoću funkcije:

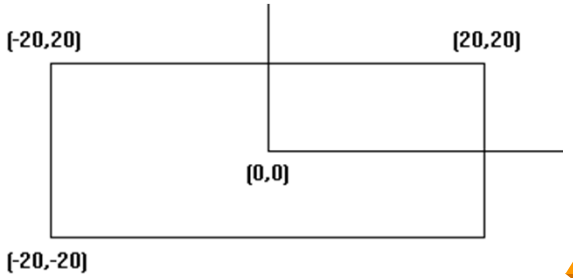
- BOOL CDC::SetViewportOrg(
  int *X*, // new device x-coordinate of viewport origin
  int *Y* // new device y-coordinate of viewport origin);

Primer

```
CRect rect;
GetClientRect(&rect);
pDC->SetMapMode(MM_ANISOTROPIC);
pDC->SetWindowExt(100,-100);
pDC->SetViewportExt(rect.right,rect.bottom);
pDC->SetWindowOrg(-50,50);
pDC->Rectangle(-20,20,20,-20);
```



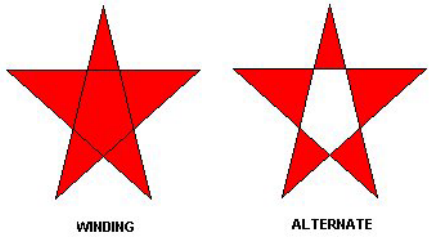




Polygon-fill mod

Definiše kako se šablon četke koristi za ispunu kompleksnih regiona, i može biti:

- **ALTERNATE** – ispunjava oblasti između neparnog i parnog broja ivica koje preseca scan-linija.
- **WINDING** – ispunjava svaki region sa nenultom *winding* vrednošću.
- Funkcije:
  - int CDC::GetPolyFillMode( );
  - int CDC::SetPolyFillMode(int *iPolyFillMode*);



Stretching mod

Definiše kako sistem kombinuje piksele bitmape sa postojećim pikselima na podlozi prilikom poziva funkcije StretchBlt, i može imati sledeće vrednosti:

- **BLACKONWHITE** - Performs a Boolean AND operation using the color values for the eliminated and existing pixels. If the bitmap is a monochrome bitmap, this mode preserves black pixels at the expense of white pixels.
- **COLORONCOLOR** - Deletes the pixels. This mode deletes all eliminated lines of pixels without trying to preserve their information.
- **HALFTONE** - Maps pixels from the source rectangle into blocks of pixels in the destination rectangle. The average color over the destination block of pixels approximates the color of the source pixels. After setting the HALFTONE stretching mode, an application must call the **SetBrushOrgEx** function to set the brush origin. If it fails to do so, brush misalignment occurs.
- **STRETCH_ANDSCANS** - Same as **BLACKONWHITE**.
- **STRETCH_DELETESCANS** - Same as **COLORONCOLOR**.
- **STRETCH_HALFTONE** - Same as **HALFTONE**.
- **STRETCH_ORSCANS** - Same as **WHITEONBLACK**.
- **WHITEONBLACK** - Performs a Boolean OR operation using the color values for the eliminated and existing pixels. If the bitmap is a monochrome bitmap, this mode preserves white pixels at the expense of black pixels.

Stretching mod

Funkcije za očitavanje trenutno postavljenog i postavljenje novom *stretching* moda:

- int CDC::GetStretchBltMode( );
- int CDC::SetStretchBltMode(int *iStretchMode*);

BLACKONWHITE i **WHITEONBLACK** modovi se koriste da očuvaju *foreground* piksele u monohromatskim bitmapama

COLORONCOLOR mod se koristi da očuva boju u kolor bitmapama (brz je ali je slika krzava ako nije preslikavanje 1:1)

HALFTONE mod umekšava prikaz bitmapa kada preslikavanje nije 1:1

Grafički objekti

- **Olovke** (*Pen*) – koristi se za iscrtavanje linija i krivih i
- **Četke** (*Brushes*) – za ispunu unutrašnjosti poligona, elipsi i putanja (*paths*),
- **Bitmape** *(Bitmaps)* – za kreiranje, manipulaciju i skladištenje slika na disku, za kopiranje i skrolovanje delova ekrana,
- **Fontovi** (*Fonts*) – za ispis teksta na ekranu i drugim izlaznim uređajima,
- **Logička paleta** (*Logical Palette*) – paleta boja koju aplikacija kreira i dodeljuje je datom DC-ju, definiše skup raspoloživih boja,
- **Putanje** (*Paths*) – jedan ili više oblika koji mogu biti ispunjeni, uokvireni ili i jedno i drugo, i
- **Regioni** (*Regions*) – pravougaonik, poligon ili elipsa (ili kombinacija više oblika), koji se može ispuniti, obojiti, invertovati, uokviriti, koristiti za ispitivanje položaja kursora (*hit-testing*), ili odsecanje *(clipping).*

Atributi grafičkih objekata

Grafički objekat	Atributi
Bitmapa	Veličina [B], dimenzije [pix], color-format, kompresiona šema, itd.
Četka	Stil, boja, šablon i početak
Paleta	Veličina i boje
Font	Ime, širina, visina, debljina, karakter-set, itd.
Putanja	Oblik
Olovka	Stil, debljina i boja
Region	Pozicija i dimenzije

Promena atributa objekata

Kada aplikacija kreira DC, sistem automatski postavlja podrazumevane objekte i njihove atribute (osim bitmape i putanje).

Da bi se promenile podrazumevane vrednosti atributa, mora se:

1. **Napraviti novi objekat odgovarajućeg tipa**
2. **Sačuvati prethodni objekat tog tipa**
3. **Selektovati novi objekat u DC**

Po završetku korišćenja novog objekta, mora se:

1. **Selektovati prethodni objekat u DC**
2. **Obrisati kreirani (novi) objekat**

Primer selektovanja objekta

```
CPen newPen ( PS_SOLID, 0, RGB(0,0,255) );
CPen* oldPen = pDC->SelectObject( &newPen );
...
pDC->SelectObject( oldPen );
newPen.DeleteObject();
```

Snimalje/vraćanje stanja DC-ja

- virtual int CDC:: SaveDC( );
  - svi selektovani objekti i postavljeni grafički modovi snimaju se u *context* steku

- BOOL CDC::RestoreDC(int *nSavedDC*);
  - *nSavedDC* određuje instancu DC koja se vraća iz setka:
    - pozitivan – odgovarajuća instanca (prosleđuje se vrednost koju je vratio prethodni poziv funkcije SaveDC),
    - negativan – relativno u odnosu na vrh steka (-1 poslednje snimljeno stanje)
  - Sve instace DC-ja između izabranog indeksa i vrha steka se brišu!

GDI+

Šta je potrebno za GDI+

- gdiplus.dll – već je uključen u distribuciju WinXP, ali ga stariji OS ne sadrže
- GdiPlus.lib – statička biblioteka
- Include datoteke – GdiPlus.h, GdiPlusBase.h, GdiPlusBitmap.h ...

StdAfx.h ili View.cpp

- #include <gdiplus.h>
- #pragma comment(lib, "\\GDIPlus\\GdiPlus.lib")
- using namespace Gdiplus;

Ako nije definisan ULONG_PTR dodati i sledeće

- #ifndef ULONG_PTR
- #define ULONG_PTR unsigned long*
- #endif

Application.h / .cpp

ULONG_PTR m_gdiplusToken;

CApp::InitInstance()

- Gdiplus::GdiplusStartupInput gdiplusStartupInput;
- Gdiplus::GdiplusStartup(&m_gdiplusToken, &gdiplusStartupInput, NULL);

CApp::ExitInstance()

- Gdiplus::GdiplusShutdown(m_gdiplusToken);

Formiranje Graphics objekta

- Ono što je DC za GDI, to je **Graphics** za GDI+
- Može se kreirati na više načina, a koristićemo sledeća 2:
  - **Graphics(HDC *hdc*);**
  - **Graphics(Image* *image*);**

Primer:

```
void CGDIPlusView::OnDraw(CDC* pDC)
{
        Graphics graphics(pDC->m_hDC);
}
```

GDI+ nema koncept selekcije objekata

Objekti se direktno prosleđuju funkcijama za iscrtavanja, npr:

```
Pen pen(Color(255, 0, 0, 0));
graphics.DrawLine(&pen, 20, 10, 300, 100);
```

Qt

Formiranje QPainter objekta

- Ono što je DC za GDI, to je **QPainter** za Qt
- Kreira se na osnovu QPaintDevice-a:
  - QPainter::QPainter (QPaintDevice * device)
- Trenutno podržani “uređaji” su:
  - QWidget, QImage, QPixmap, QGLPixelBuffer, QPicture i QPrinter.

Primer:

```
void SimpleExampleWidget::paintEvent(QPaintEvent *) {
        QPainter painter(this);
}
```

Selektovanje objekata

setPen
setBrush
setFont
setClipPath
setClipRect
setClipRegion
setBackground
setBackgroundMode
setBrushOrigin
setCompositionMode
setLayoutDirection
setOpacity
setRenderHint
setRenderHints
setTransform
setViewTransformEnabled
setViewport
setWindow
setWorldMatrixEnabled
setWorldTransform

Dodatne aktivnosti

- Proveriti i napisati koji su parametri (atributi i modovi) DC-ja postavljeni inicijalno prilikom poziva OnDraw metoda MFC aplikacije.
- Koje parametre treba proslediti funkciji *SetViewportOrg* (umesto poziva f-je *SetWindowOrg*) u primeru na slaju 20, da se prikaz ne bi promenio?
- Kako treba postaviti parametre u funkcijama SetWindowExt i SetViewportExt da bi slika na ekranu bila 2x uvećana (umanjena) u odnosu na sliku definisanu logičkim koordinatama?
- Koji *drawing* mode treba postaviti da bi svako drugo iscrtavanje linije (preko postojece linije) poništilo (izbrisalo) prethodno iscrtanu liniju, bez obzira na boju pozadine.

Dodatne aktivnosti

- Šta je *handle*?
- Kako se može pribaviti DC tekućeg prozora, a kako čitavog ekrana?
- Kako se mogu pribaviti podaci o uređaju na kome se crta (rezolucija, veličina prozora,...)?
- Otkriti šta rade funkcije:
  - LPtoDP i
  - DPtoLP.

Pitanja