Grundlagen in Visual Studio
MFC und .net

- Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
- Hochschule Harz
- FB Automatisierung und Informatik
- mwilhelm@hs-harz.de
- Raum 2.202
- Tel. 03943 / 659 338

Inhalt

Grafikobjekte in Windows

- Grafikobjekte (Linien, etc.) benötigen einen Gerätekontext zum Zeichnen

- Ein MFC-Programm hat verschiedene Möglichkeiten an einem Gerätekontext zu gelangen
  - Aufruf von CWnd::GetDC
  - Aufruf in einer On-Paint-Methode
  - Metafile erzeugen

Grafikkontexte

CPaintDC-Objekte bilden die Basis für das Zeichnen

1) Aufruf der Funktion **BeginPaint**,
2) dem anschließenden Anzeigen im Gerätekontext
3) Aufruf der Funktion **EndPaint**.

Der Konstruktor **CPaintDC** ruft automatisch **BeginPaint auf.**
Der Destruktor ruft automatisch **EndPaint** auf.

Alle Draw-Funktionen in der MFC erhalten automatisch ein CpaintDC-Objekt. Er wird automatisch wieder freigegeben.

Grafikkontexte

- **CClientDC**-Objekte umfassen die Arbeit mit einem Gerätekontext, der nur den Innenbereich eines Fensters repräsentiert. Der Konstruktor **CClientDC** ruft die Funktion **GetDC** und der Destruktor die Funktion **ReleaseDC** auf.

- **CWindowDC**-Objekte umfassen einen Gerätekontext, der das gesamte Fenster, einschließlich des Rahmens, repräsentiert.

Grafikobjekte in Windows

- **Dialogfenster:**

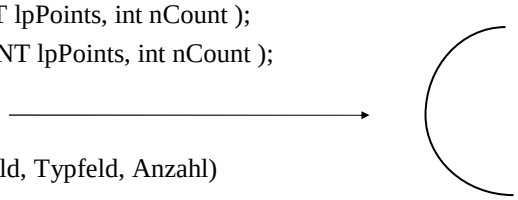
```
void CGrafik1Dlg::OnPaint() {
    CPaintDC dc(this); // Gerätekontext für Zeichnen
    dc.MoveTo (int x, int y);
    dc.LineTo (int x, int y);
}
```

- **SDI Fenster:**

```
void CLabor1View::OnDraw(CDC* pDC) {
    // dynamischer Gerätekonzept
    pDC->MoveTo (int x, int y);
    pDC->LineTo (int x, int y);
}
```

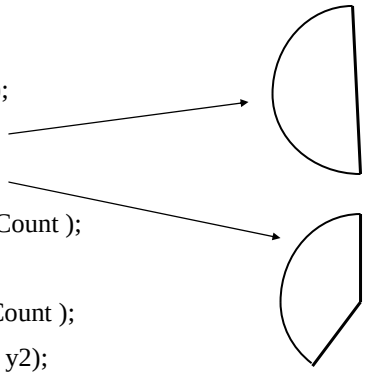
Grafikfunktionen

- Linien
  - MoveTo(int x, int y);
  - LineTo (int x, int y);
  - Polyline (LPPOINT lpPoints, int nCount);
  - PolylineTo(LPPOINT lpPoints, int nCount);
  - Arc
  - ArcTo
  - PolyDraw (Punktfeld, Typfeld, Anzahl)
  - PolyBezier
  - PolyBezierTo
  - AngleArc
  - Rectangle (int x1, int y1, int x2, int y2);
  - RoundRect (int x1, int y1, int x2, int y2);



Grafikfunktionen

- Flächen (Linien und Muster)
  - Ellipse(int x1, int y1, int x2, int y2);
  - Chord (P_1 bis P_4); (siehe Arc)
  - Pie(P_1 bis P_4); (siehe Arc)
  - Polygon (LPPOINT lpPoints, int nCount);
  - PolyPolygon
  - Polyline(LPPOINT lpPoints, int nCount);
  - Rectangle (int x1, int y1, int x2, int y2);
  - RoundRect (int x1, int y1, int x2, int y2);
  - FillRect(CRect * rect; CBrush brush);



Polyline

- BOOL Polyline(LPPOINT lpPoints, int nCount);
- lpPoints: Points to an array of POINT structures or CPoint objects to be connected.
- nCount: Specifies the number of points in the array. This value must be at least 2.

Beispiel:
POINT p[5] = { 100,100, 300,100, 300,300, 100,300, 100,100};
dc.Polyline(p,5);

PolyDraw

- BOOL PolyDraw(const POINT* lpPoints, const BYTE* lpTypes, int nCount);
- lpPoints: Points to an array of POINT data structures that contains the endpoints for each line segment and the endpoints and control points for each Bézier spline.
- lpTypes: Points to an array that specifies how each point in the lpPoints array is used. Values can be one of the following:

Konstante der Typdefinitionen:

- PT_MOVETO
- PT_LINETO
- PT_BEZIERTO
- PT_CLOSEFIGURE

Arc (Bogen)

- BOOL Arc(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4);
- BOOL Arc(LPCRECT lpRect, POINT ptStart, POINT ptEnd);

- x1: LEFT
- y1: TOP
- x2: RIGHT
- y2: BOTTOM
- x3: Specifies the x-coordinate of the point that defines the arc's starting point (in logical units). This point does not have to lie exactly on the arc.
- y3: Specifies the y-coordinate of the point that defines the arc's starting point (in logical units). This point does not have to lie exactly on the arc.
- x4: Specifies the x-coordinate of the point that defines the arc's endpoint (in logical units). This point does not have to lie exactly on the arc.
- y4: Specifies the y-coordinate of the point that defines the arc's endpoint (in logical units). This point does not have to lie exactly on the arc.

Linienmuster

- PS_SOLID
- PS_DASH
- PS_DASHDOT
- PS_DASHDOTDOT
- PS_NULL
- PS_INSIDEFRAME

Beispiel:

```
CPen cpen(PS_DOT, 3,
    RGB(0,255,255) );
dc.SelectObject(&cpen);
```

Flächenmuster

- HS_BDIAGONAL
- HS_FDIAGONAL
- HS_CROSS
- HS_HORIZONTAL
- HS_DIAGCROSS
- HS_VERTICAL

Beispiel:

```
CBrush brush1( RGB(255,0,255) );
dc.SelectObject(&brush1);

CBrush brush2( HS_DIAGCROSS, RGB(255,0,255) );
dc.SelectObject(&brush2);

CBrush brush3;
brush3.CreateSolidBrush( RGB(255,255,255));
dc.SelectObject(&brush3);
```

GDI-Zeichenmodi

Bit-Operator

R2_NOP	color = color
R2_NOT	color = NOT color
R2_BLACK	color = BLACK
R2_WHITE	color = WHITE
R2_COPYPEN	color = NewColor
R2_NOTCOPYPEN	color = NOT NewColor
R2_MERGEPENNOT	color = (NOT color) OR NewColor
R2_MASKPENNOT	color = (NOT color) AND NewColor
R2_MERGENOTPEN	color = (NOT NewColor) OR color
R2_MASKNOTPEN	color = (NOT NewColor) AND color
R2_MERGEPEN	color = color OR NewColor
R2_NOTMERGEPEN	color = NOT (color OR NewColor)
R2_MASKPEN	color = color AND NewColor
R2_NTMASKPEN	color = NOT (color AND NewColor)
R2_XORPEN	color = color XOR NewColor
R2_NOTXORPEN	color = NOT (color XOR NewColor)

Beispiele

```
dc.SetROP2(R2_NOT);
dc.MoveTo(100,100);
dc.LineTo(200,300);
```

```
CPen cpen1(PS_DOT, 1,   RGB(255,0,0) );
dc.SelectObject(&cpen1);
dc.SetROP2(COPYPEN);
dc.Rectangle(100,100,  300,300);

CPen cpen2(PS_SOLID, 1,   RGB(0,0,0) );
dc.SelectObject(&cpen);
dc.SetROP2(COPYPEN);
dc.Rectangle(100,100,  300,300);
```

Ausgabe von Texten

Methoden zum Zeichnen eines Textes:

- DrawText
- TextOut
- ExTextOut
- TabbedTextOut
- GrayString

Methoden zum Verwalten von Schriften:

- GetFontData
- GetCharWidth
- GetCharABCWidths
- GetOutlineTextMetrics
- Rasterfont vs. TrueTypefonts

DrawText

- **virtual int DrawText(LPCTSTR** *lpszString*, **int** *nCount*, **LPRECT** *lpRect*, **UINT** *nFormat* **);**
- **int DrawText(const CString&** *str*, **LPRECT** *lpRect*, **UINT** *nFormat* **);**

- *lpszString:* Points to the string to be drawn. If *nCount* is –1, the string must be null-terminated.
- *nCount:* Specifies the number of chars in the string. If *nCount* is –1, then *lpszString* is assumed to be a long pointer to a null-terminated string and **DrawText** computes the character count automatically.
- *lpRect: Points to a RECT-Structure*
- *str:* A Cstring-object that contains the specified characters to be drawn.
- *nFormat:* Specifies the method of formatting the text. It can be any combination of the following values (combine using the bitwise OR operator):

DrawText: Optionen

- **DT_BOTTOM:** + **DT_SINGLELINE**.
- **DT_CALCRECT:** DrawText returns the height of the formatted text, but does not draw the text.
- **DT_BOTTOM:**
- **DT_CENTER:**
- **DT_LEFT**
- **DT_NOCLIP**
- **DT_NOPREFIX** (&& vs. &)
- **DT_RIGHT**
- **DT_TABSTOP**
- **DT_TOP**
- **DT_VCENTER**
- **DT_WORKBREAK** (Trennung)

TextOut

- **virtual BOOL TextOut(int** *x*, **int** *y*, **LPCTSTR** *lpszString*, **int** *nCount* **);**
- **BOOL TextOut(int** *x*, **int** *y*, **const CString&** *str*);

- *x:* Specifies the logical x-coordinate of the starting point of the text.
- *y:* Specifies the logical y-coordinate of the starting point of the text.
- *lpszString:* Points to the character string to be drawn.
- *nCount:* Specifies the number of bytes in the string.
- *Str:* A **CString** object that contains the characters to be drawn.

TextOut: Textausrichtungen

pDC->SetTextAlign(Optionen);

- **TA_LEFT**
- **TA_CENTER**
- **TA_RIGHT**

- **TA_BOTTOM**
- **TA_BASELINE**
- **TA_TOP**

- **TA_NOUPDATECP**
- **TA_UPDATECP**

ExtTextOut

- BOOL ExtTextOut(int x, int y, UINT nOptions, LPCRECT lpRect, const CString& str, LPINT lpDxWidths);
- x: The logical x-coordinate of the character cell for the first character
- y: The logical y-coordinate of the top of the character cell for the first character
- nOptions: Specifies the rectangle type:
  - ETO_CLIPPED Specifies that text is clipped to the rectangle.
  - ETO_OPAQUE Specifies that the current background color fills the rectangle.
- lpRect: Points to a RECT structure that determines the dimensions of the rectangle. This parameter can be NULL. You can also pass a CRect object for this parameter.
- lpszString: Points to the specified character string to be drawn. nCount: Specifies the number of characters in the string.
- lpDxWidths: Points to an array of values that indicate the distance between origins of adjacent character cells. For instance, lpDxWidths[i] logical units will separate the origins of character cell i and character cell i + 1. If lpDxWidths is NULL, ExtTextOut uses the default spacing between characters.
- str: A CString object that contains the specified characters to be drawn.

Bedeutung

- DT_BOTTOM

Der Text wird an der Grundlinie ausgerichtet. Zusätzlich muss die Konstante DT_SINGLELINE mit angegeben werden.

- DT_CALCRECT

Dient zur Berechnung der Breite und Höhe des Rechtecks, in dem der Text vollständig dargestellt werden kann. Enthält der Text mehrere Zeilen, so wird die durch lpRect vorgegebene Breite für das Ausgabefeld beibehalten und die Höhe entsprechend berechnet. Enthält der Text nur eine Zeile, so wird die Breite des Ausgaberechtecks berechnet. Der Text selbst wird in keinem Fall ausgegeben.

- DT_CENTER Gibt den Text zentriert im Ausgabefeld aus.
- DT_EXPANDTABS

Expandiert Tabulatoren im Text. Standardmäßig werden Tabulatoren auf die 8-fache mittlere Zeichenbreite gesetzt.

- DT_EXTERNALLEADING

Bindet für die Berechnung der Zeilenhöhe das Datum 'external leading' mit ein. (Siehe dazu auch Grafik am Anfang der nächsten Lektion).

Bedeutung

- DT_LEFT Richtet den Text linksbündig aus.
- DT_NOCLIP Verwendet für die Textausgabe kein Clipping.

- DT_NOPREFIX

Standardmäßig wird das nach einem &-Zeichen folgende Zeichen unterstrichen ausgegeben. Durch Angabe von DT_NOPREFIX kann dies ausgeschaltet werden. Soll im Standardfall ein &-Zeichen dargestellt werden, so ist dieses Zeichen zu verdoppeln (&&).

- DT_RIGHT Richtet den Text rechtsbündig aus.

- DT_SINGLELINE

Der Text soll unabhängig von CR und LF immer in einer Zeile ausgegeben werden.

- DT_TABSTOP

Legt fest, dass im High-Byte der Format-Spezifikation die Anzahl der Zeichen pro Tab-Stop definiert ist.

Bedeutung

- DT_TOP

Der Text wird an der Oberkante ausgerichtet. Zusätzlich muss die Konstante DT_SINGLELINE mit angegeben werden.

- DT_VCENTER

Der Text wird vertikal zentriert ausgerichtet. Zusätzlich muss die Konstante DT_SINGLELINE mit angegeben werden.

- DT_WORDBREAK

Der Text wird automatisch umgebrochen wenn das nächste Wort über die rechte Grenze des Ausgabebereichs hinausgehen würde. Eine CR/LF Sequenz erzeugt ebenfalls eine neue Zeile.

Beispiele:

```
CFont font;
font.CreatePointFont(140,"Times New Roman");
CFont* oldfont = dc.SelectObject( &font);
dc.SetBkMode(TRANSPARENT);

CRect rect;
GetClientRect (&rect);
dc.DrawText ( "Mein erster Text", -1, &rect,
        DT_SINGLELINE | DT_CENTER | DT_VCENTER);

dc.SetTextColor( RGB(255,0,0) );
dc.TextOut(100,550,"DrawText");

CRect rect1(0,0,200,100);
dc.DrawText ( "ABCDEFGHIJKLMNOPQRSTUVWXYZ",
-1, &rect1, DT_SINGLELINE | DT_LEFT);
```

Koordinatensysteme

- **Logische Koordinaten**, sind Daten, die einer CDC-Methode übergeben werden, um etwas zu zeichnen.
- **Physikalische Koordinaten** bezeichnen die konkreten Bildpunkte des Ausgabegerätes
  - Bildschirm
  - Drucker
  - Metafile
  - Speicher
- **Abbildungmodi**:
  - MM_TEXT 1 Bildpunkt +x / +y
  - MM_LOMETRIC 0,1 mm +x / -y
  - MM_LOENGLISH 0,01 Zoll +x / -y
  - MM_HIENGLISH 0,001 Zoll +x / -y
  - MM_TWIPS 1/440 Zoll +x / -y
  - MM_ISOTROPIC benutzerdefiniert x/y gleichskaliert
  - MM_ANISOTROPIC benutzerdefiniert x/y-Skalierung unabhängig
- dc.SetMapMode(MM_LOMETRIC);

Fertige GDI-Objekte im Stock

- **NULL_PEN** Stift, der nicht zeichnet
- **BLACK_PEN** Stift, schwarz, solid, 1 Pixel
- **WHITE_PEN** Stift, weiß, solid, 1 Pixel
- **NULL_BRUSH** Pinsel, der nicht zeichnet
- **HOLLOW_BRUSH** Pinsel, der nicht zeichnet
- **BLACK_BRUSH** Pinsel, schwarz
- **DKGRAY_BRUSH** Pinsel, dunkelgrau
- **GRAY_BRUSH** Pinsel, grau
- **LTGRAY_BRUSH** Pinsel, hellgrau
- **WHITE_BRUSH** Pinsel, weiß
- **ANSI_FIXED_FONT** Ansi-Schrift, nicht proportional (à la Courier)
- **ANSI_VAR_FONT** Ansi-Schrift, proportional (à la Helvetica, Garamond)
- **SYSTEM_FONT** System-Schrift proportional
- **SYSTEM_FIXED_FONT** System-Schrift nicht proportional (FixedSys)

Beispiele:

```
CPen pen2(PS_NULL, 0, RGB(0,0,0));
CPen pen;
pen.CreateStockObject(NULL_PEN);
dc.SelectObject(&pen);

Cbrush brush;
brush.CreateStockObject( GRAY_BRUSH);
dc.SelectObject(&brush);
dc.Ellipse(0,0,100,200);
```

```
dc.SelectStockObject(NULL_PEN);
dc.SelectStockObject(LTGRAY_BRUSH);
dc.Ellipse(0,0,100,200);
```


CDC Class Members: Übersicht

- Data Members
- Construction/Destruction
- Initialization
- Device-Context Functions
- Drawing-Tool Functions
- Type-Safe Selection. Helpers
- Color and Color Palette Functions
- Drawing-Attribute Functions
- Mapping Functions
- Coordinate Functions
- Region Functions
- Clipping Functions
- Line-Output Functions
- Simple Drawing Functions
- Ellipse and Polygon Functions
- Bitmap Functions
- Text Functions
- Font Functions
- Printer Escape Functions
- Scrolling Functions
- Metafile Functions
- Path Functions