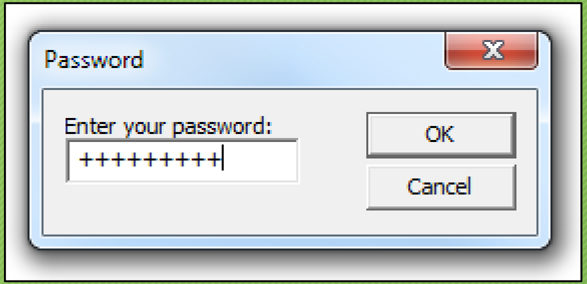
Dialog Boxes

Dialog Box

- Used to get input data from the user or communicate important messages such as an error or a warning
- Usually have owners
- Modal dialog box – blocks until the user returns

- Dialog boxes templates
  - Stored in resources or in memory



Dialog Box

The coordinate space

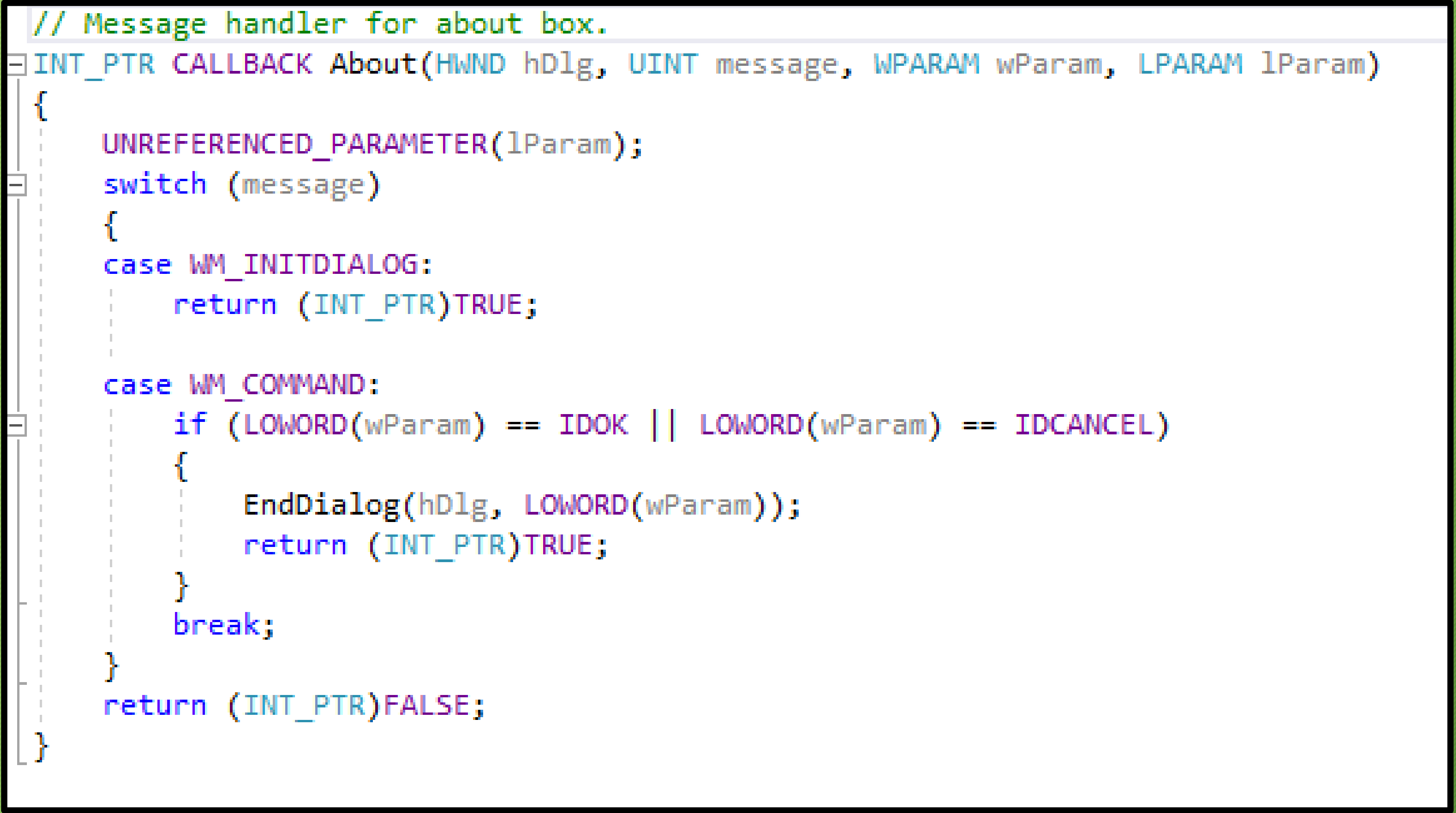
- special dialog box units, independent on display
  - 1/4 of average character width
  - 1/8 of average character height
- MapDialogRect(), GetDialogBaseUnits()

The dialog box procedure

- offers some default functionality
  (e.g. focus management)
- should not call DefWindowProc()
- returns TRUE, if a message was processed

- WM_INITDIALOG (instead of WM_CREATE)

Sample About Dialog Box



```
// Message handler for about box.
INT_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    UNREFERENCED_PARAMETER(lParam);
    switch (message)
    {
    case WM_INITDIALOG:
        return (INT_PTR)TRUE;

    case WM_COMMAND:
        if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)
        {
            EndDialog(hDlg, LOWORD(wParam));
            return (INT_PTR)TRUE;
        }
        break;
    }
    return (INT_PTR)FALSE;
}
```

Dialog Box Definition

```
IDD_DIALOG DIALOGEX 0, 0, 276, 116
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUP | WS_CAPTION |
WS_SYSMENU
CLASS „CustomClass”
CAPTION "Dialog"
FONT 8, "MS Shell Dlg", 400, 0, 0x1
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,151,95,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,219,95,50,14
END
```

Modal Dialog Boxes

- System modal or application modal
- The only window of the application receiving the user’s input
- Standard styles: `WS_POPUP, WS_SYSMENU, WS_CAPTION, DS_MODALFRAME` (never: `WS_CHILD`)
- Creating and displaying
  - DialogBox(), DialogBoxIndirect()
  - DialogBoxParam(), DialogBoxIndirectParam()
- The system creates a new message loop for the dialog box and all the application’s messages are processed here
  - messages not related to this dialog box are forwarded to the owner window
- Closing and destroying: EndDialog()

Modeless Dialog Boxes

- Allow working with other windows of the application at the same time
- Standard styles: `WS_POPUP`, `WS_CAPTION`, `WS_BORDER`, `WS_SYSMENU`
- **Creating**
  - CreateDialog(), CreateDialogIndirect()
  - CreateDialogParam(), CreateDialogIndirectParam()
- **Displaying**
  - ShowWindow() (necessary, if the WS_VISIBLE style is not set)
- **Using**
  - call the IsDialogMessage() function for each message in the message loop
- **Destroying**
  - DestroyWindow()

Message Loop Revisited

```
while (GetMessage(&msg, NULL, 0, 0))
{
   if (hDlgModeless == 0 || !IsDialogMessage(hDlgModeless, &msg))
   {
      if (!TranslateAccelerator(hWnd, hAccel, &msg))
      {
         TranslateMessage(&msg);
         DispatchMessage(&msg);
      }
   }
}
```

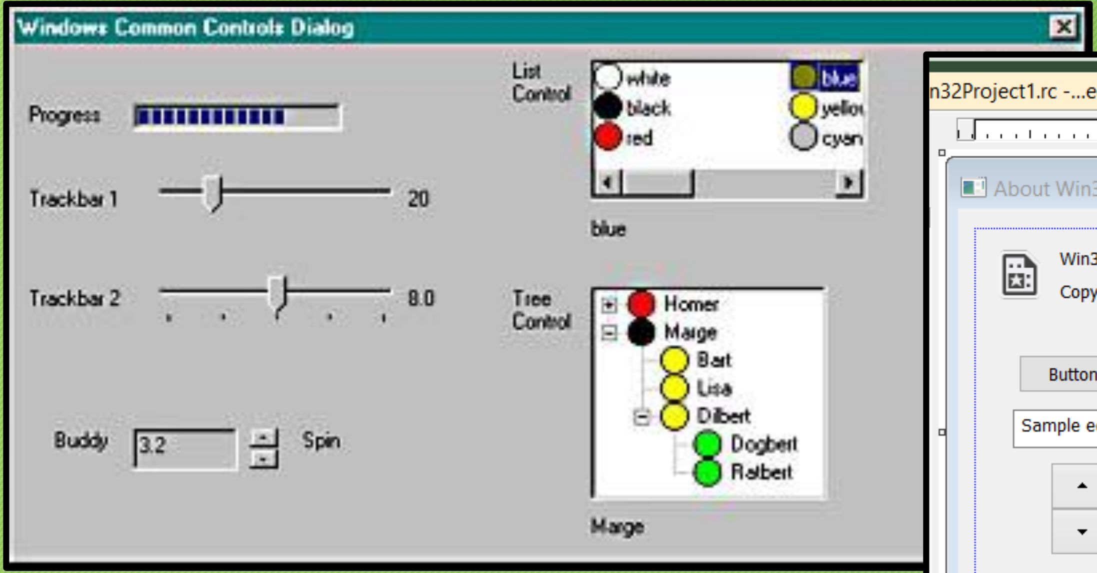
Controls

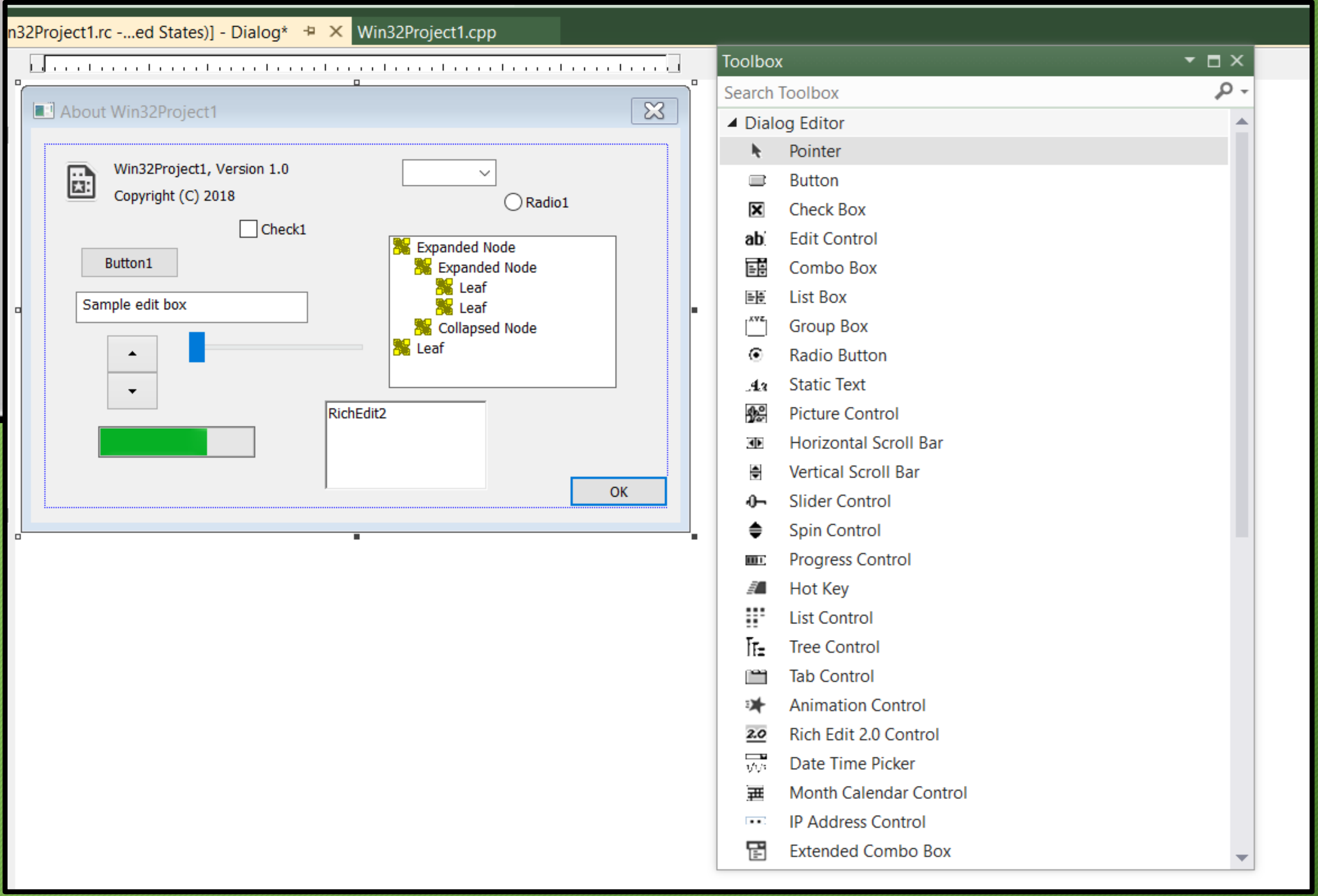
- The dialog box's template defines the position, size, style, identifier, and window class of all controls
- All controls are child windows (WS_CHILD is applied)
- Each control has an unique identifier (except for statics)
- The user's actions on controls generate messages sent to the dialog box
  - `WM_COMMAND`
  - `WM_NOTIFY` - common controls
- To get the control's HWND: GetDlgItem()
- Modifications:
  - SetDlgItemText(), SetDlgItemInt()
  - CheckDlgButton()
  - EnableWindow(), SetFocus()

Standard Controls

- **Predefined window classes for controls:**
  - BUTTON, COMBOBOX, EDIT, LISTBOX, SCROLLBAR, STATIC
  - RichEdit (ver.1.0), RICHEDIT_CLASS (ver.2.0 i 3.0)
- **Common controls - InitCommonControlsEx()**
  - ComboBoxEx, Flat Scroll Bar
  - Date and TimePicker, Month Calendar
  - Progress Bar, Trackbar, Up-Down
  - List View, Tree View, Header, Image List
  - Pager, Property Sheet, Tab
  - Rebar, Toolbar, Status Bar
  - Animation, IP Address, ToolTip, SysLink

Exemplar Common Controls





Non-Standard Controls

- **Nonstandard drawing**
  - owner-drawn
    - static, button, listbox, combobox
    - using special style, e.g. SS_OWNERDRAW
    - WM_MEASUREITEM, WM_DRAWITEM
  - custom draw
    - header, list view, rebar, toolbar, tooltip, trackbar, tree view
    - NM_CUSTOMDRAW
- **Subclassing** – using a custom window procedure
  - Not to confuse with inheritance
  - SetWindowLong() with GWL_WNDPROC
  - CallWindowProc() for standard processing of messages
- **New control** – a new window class and window procedure

Tab Stops

Styles:

- WS_TABSTOP
  - default for most controls that can receive focus
  - automatically transferred to the currently selected radio button within a group
- WS_GROUP
  - default for some controls (e.g. texts)
  - starts new group

Default focused control:

- SetFocus() in WM_INITDIALOG
- first control with WS_TABSTOP style

- **GetNextDlgTabItem(), GetNextDlgGroupItem()**

Standard Dialog Windows

- **MessageBox(), MessageBoxEx()**
- **Standard dialogs:**
  - ChooseColor(), CHOOSECOLOR
  - ChooseFont(), CHOOSEFONT
  - FindText(), FINDREPLACE
  - ReplaceText(), FINDREPLACE
  - GetOpenFileName(), OPENFILENAME
  - GetSaveFileName(), OPENFILENAME
  - PageSetupDlg(), PAGESETUPDLG
  - printing
    - PrintDlg(), PRINTDLG
    - PrintDlgEx(), PRINTDLGEX [2000]

GDI

GDI

- **GDI** – Graphics Device Interface
- A graphical component of Windows
- Uses a lot of system objects – **beware of leaks**
- Allows applications to use graphical devices without any knowledge of their drivers
    - but not all functions equally high-level
    - API within the WinAPI

GDI

- Applications create or get logical graphical objects which can be selected on the device context
- GDI uses existing graphical possibilities of devices or can simulate them if missing
- Applications can use logical coordinates, devices use real coordinates

- In **GDI**, any non-accelereated graphics-related task can be done

HDC

- **HDC** – device context

Types of device contexts:

- Sceen – GetDC(), GetDCEx(), GetWindowDC(), BeginPaint()
- Printer – CreateDC()
- Memory – CreateCompatibleDC()
- Information – CreateIC()

How to release:

- ReleaseDC() – if we got DC by Get…
- DeleteDC() – if we got DC by Create…

HDC

HDC maintains its current state

SelectObject()

- Selects an object and replaces the previous one of the same type

GetCurrentObject()

- pass HDC + type

GetObject()

- HGDIOBJ

currentposition

- **MoveToEx()**

Types of functions in GDI

1. Functions that make something with the Device Context (HDC)
    - get, create, destroy, release, copy, …
2. Functions that set / change / get attributues of the HDC
    - e.g. transformations
3. Functions that retrieve information about devices
4. Functions that operate on GDI objects
    - e.g. pens
5. Functions that draw something

Drawing Window’s Content

- **Background of a window**
  - `WM_ERASEBKGND`
  - `WNDCLASSEX.hbrBackground (HBRUSH)`
- **The client area**
  - `WM_PAINT - BeginPaint(), EndPaint()`
  - `GetDC(), GetDCEx(), ReleaseDC()`
- **The nonclient area**
  - usually managed by the system
  - `WM_NCPAINT, WM_NCACTIVATE`
  - `GetWindowDC(), GetDCEx(), ReleaseDC()`

HDC - modes

- **Modes set on a DC:**
  - background - SetBkMode(), GetBkMode()
    - *background mix mode*
  - drawing - SetROP2(), GetROP2()
    - *foreground mix mode*
  - mapping - SetMapMode(), GetMapMode()
  - polygon fill - SetPolyFillMode(), GetPolyFillMode()
  - stretching - SetStretchBltMode(), GetStretchBltMode()

WM_PAINT

- **One of the most commonly processed message!**

- BeginPaint()

```
PAINTSTRUCT ps;
HDC hdc = BeginPaint(hWnd, &ps);
```

  - gets the DC with a clipping region
  - sends WM_ERASEBKGND
  - sets the clipping region
  - hides the caret if within clipping bounds
  - (call this function only in response to WM_PAINT)
- EndPaint()

Windows redraws only when is necessary

Clipping region – region that can be clipped – to specific parts

Update region

- InvalidateRect(), InvalidateRgn()
- ValidateRect(), ValidateRgn()
- GetUpdateRect(), GetUpdateRgn()
- RectVisible() – to test if a passed rect is in the clipping region
- ExcludeUpdateRgn()
- IntersectClipRect()

- **Immediate redrawing of the client area:**
  - UpdateWindow()
  - RedrawWindow()

From the outside: minimize, alt+TAB, move window,…

Clipping

Automatically set by BeginPaint()

Advanced clipping:

- **Setting the clipping region**
  - SelectClipRgn(), ExtSelectClipRgn(),
  - SelectClipPath()
- **Checking visibility**
  - PtVisible(), RectVisible()
- **Modifying the clipping region**
  - OffsetClipRgn(), ExcludeClipRect(), IntersectClipRect()

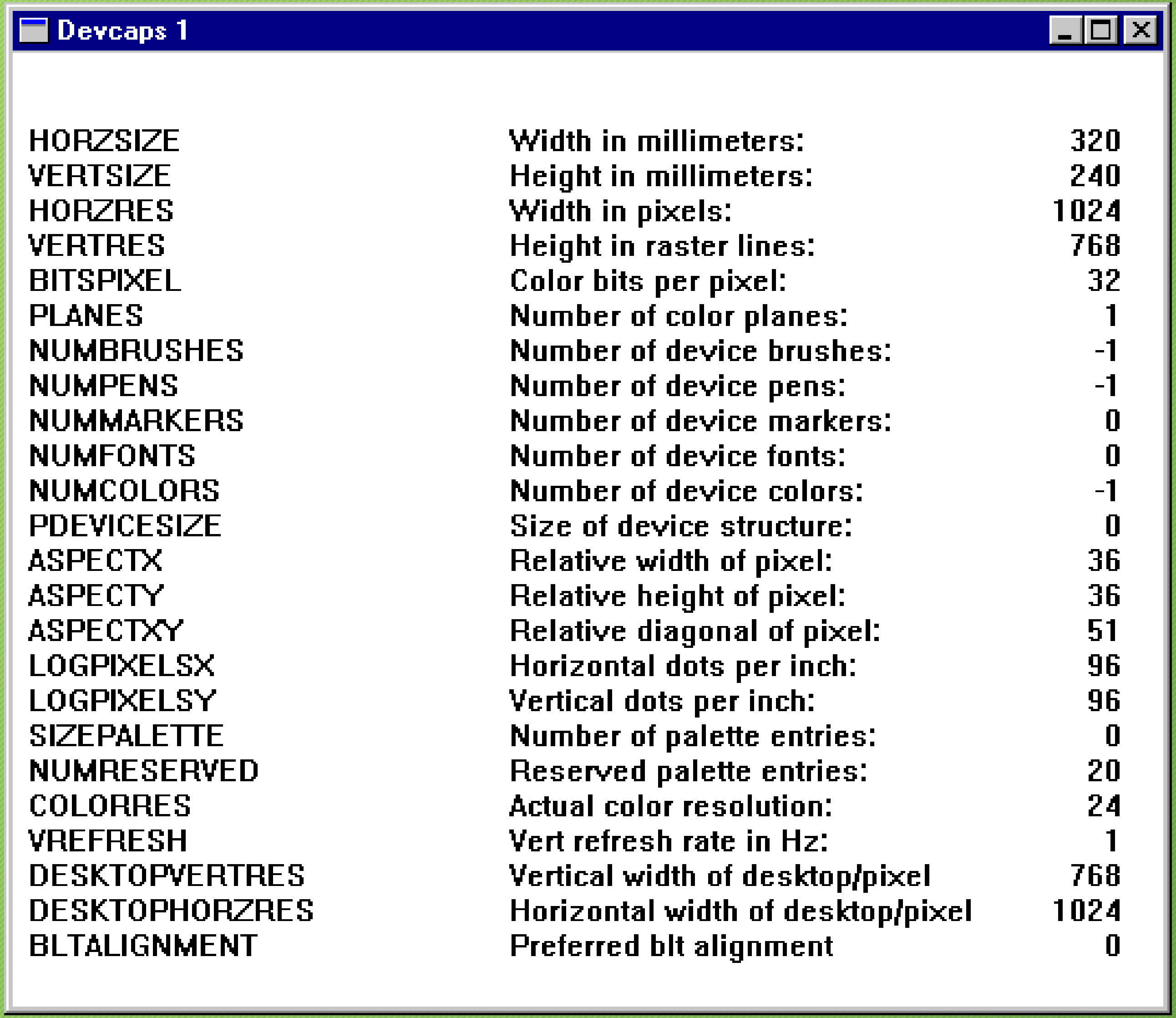
Save / Load on DC

- CS_OWNDC

```
idSaved = SaveDC(hdc);
//some graphical operations
RestoreDC(hdc, idSaved);

//Alternatively:
SaveDC(hdc);
//some graphical operations
RestoreDC(hdc, -1);
```

GetDeviceCaps()



```
HORZSIZE           Width in millimeters:                 320
VERTSIZE           Height in millimeters:                240
HORZRES            Width in pixels:                     1024
VERTRES            Height in raster lines:               768
BITSPIXEL          Color bits per pixel:                  32
PLANES             Number of color planes:                 1
NUMBRUSHES         Number of device brushes:              -1
NUMPENS            Number of device pens:                 -1
NUMMARKERS         Number of device markers:               0
NUMFONTS           Number of device fonts:                 0
NUMCOLORS          Number of device colors:               -1
PDEVICESIZE        Size of device structure:               0
ASPECTX            Relative width of pixel:               36
ASPECTY            Relative height of pixel:              36
ASPECTXY           Relative diagonal of pixel:            51
LOGPIXELSX         Horizontal dots per inch:              96
LOGPIXELSY         Vertical dots per inch:                96
SIZEPALETTE        Number of palette entries:              0
NUMRESERVED        Reserved palette entries:              20
COLORRES           Actual color resolution:               24
VREFRESH           Vert refresh rate in Hz:                1
DESKTOPVERTRES     Vertical width of desktop/pixel       768
DESKTOPHORZRES     Horizontal width of desktop/pixel    1024
BLTALIGNMENT       Preferred blt alignment                 0
```

Colors

COLORREF – structure to store colour

DWORD (32-bit):

A	R	G	B
8-bits	8-bits	8-bits	8-bits

- **RGB macro**
- **GetRValue(), GetGValue(), GetRValue()**

Combining pens and interiors with colors on the screen

- SetROP2(), GetROP2()

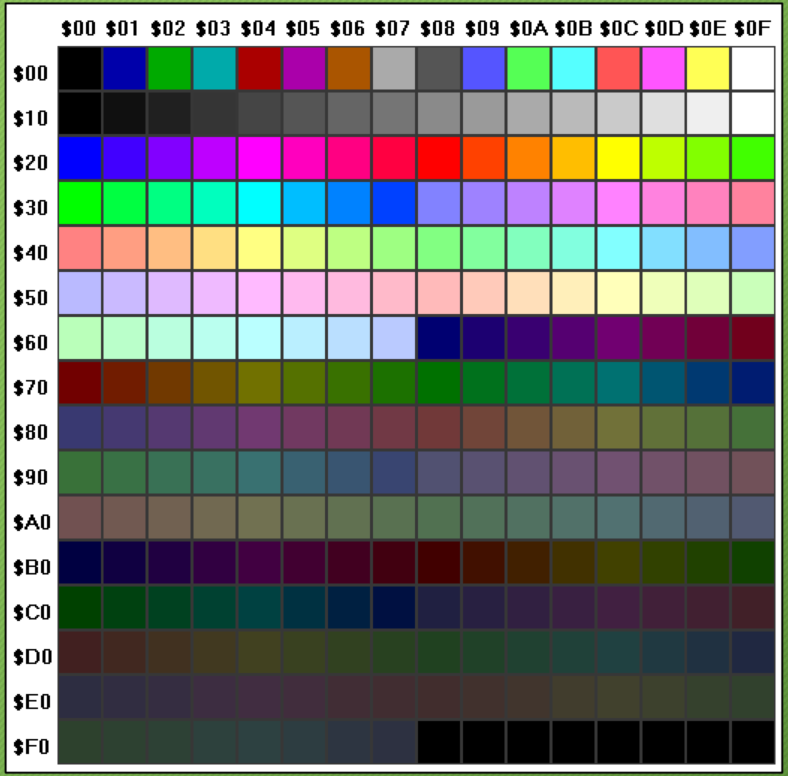
ROP2 - fnDrawMode

R2_BLACK	Pixel is always 0.
R2_COPYPEN	Pixel is the pen color.
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.
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.

Palletes

Palettes (useful, when only 256 colours can be used)

- CreatePalette(), DeleteObject()
- SelectPalette(), RealizePalette(), UnrealizeObject(), ResizePalette()
- GetPaletteEntries()
- GetNearestPaletteIndex()
- GetSystemPaletteEntries()
- GetSystemPaletteUse()


$00 $01 $02 $03 $04 $05 $06 $07 $08 $09 $0A $0B $0C $0D $0E $0F
$00
$10
$20
$30
$40
$50
$60
$70
$80
$90
$A0
$B0
$C0
$D0
$E0
$F0

Availability of colors

- GetNearestColor()

Let’s draw something

GDI objects

• Pens	HPEN
• Brushes	HBRUSH
• Fonts	HFONT
• Bitmaps	HBITMAP
• Palette	HPALETTE
• Regions	HRGN

• Enhanced Meta Files	HDC

- Different types of device contexts (as mentioned)

GDI objects

- **Creating and destroying**
  - Create...(), e.g. CreatePen(), CreateSolidBrush(), CreateFont ()
  - **all created objects must be destroyed**
  - DeleteObject()
- **SelectObject()**
  - sets the object as active in the DC
  - **objects selected as current must not be destroyed**
  - If you want to delete a selected object, select a different one (preferably the previous one or the system default one)
- **Stock objects**
  - GetStockObject()

Using GDI objects

```
hNewPen = CreatePen(PS_SOLID, 1, RGB(255, 0, 0));
if (hNewPen)
  hOldPen = SelectObject(hDC, hNewPen);
else
{
  //handle error
  return;
}

//all drawing operations using the new pen

if (hOldPen)
  SelectObject(hDC, hOldPen); //deselect hNewPen
if (hNewPen)
  DeleteObject(hDC, hNewPen); //delete the pen
                                  //if created
```

Pens

- **Cosmetic pens**
  - available properties: width, style, and colour
  - they always have fixed width (no scaling)
  - CreatePen(), CreatePenIndirect(),
  - ExtCreatePen(), GetStockObject()
- **Geometric pens**
  - available properties: width, style, colour, pattern, optional hatch, end style and join style
  - can be scaled – width is set using logical coordinates
  - ExtCreatePen()

Example – drawing a Sine Wave (from Petzold)

```
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR szCmdLine, int iCmdShow) {
     static TCHAR szAppName[] = TEXT ("SineWave") ;
     HWND         hwnd ;
     MSG          msg ;
     WNDCLASS     wndclass ;

     wndclass.style         = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc   = WndProc ;
     wndclass.cbClsExtra    = 0 ;
     wndclass.cbWndExtra    = 0 ;
     wndclass.hInstance     = hInstance ;
     wndclass.hIcon         = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName  = NULL ;
     wndclass.lpszClassName = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("Eror!"), szAppName, MB_ICONERROR);
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Sine Wave Using Polyline"), WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}
```

Example – drawing a Sine Wave (from Petzold)

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static int  cxClient, cyClient ;
     HDC         hdc ;
     int         i ;
     PAINTSTRUCT ps ;
     POINT       apt [NUM] ;

     switch (message)
     {
     case WM_SIZE:
          cxClient = LOWORD (lParam) ;
          cyClient = HIWORD (lParam) ;
          return 0 ;
     case WM_PAINT:
          hdc = BeginPaint (hwnd, &ps) ;

          MoveToEx (hdc, 0,        cyClient / 2, NULL) ;
          LineTo   (hdc, cxClient, cyClient / 2) ;

          for (i = 0 ; i < NUM ; i++)
          {
               apt[i].x = i * cxClient / NUM ;
               apt[i].y = (int) (cyClient / 2 * (1 - sin (TWOPI * i / NUM))) ;
          }

          Polyline (hdc, apt, NUM) ;
          return 0 ;

     case WM_DESTROY:
          PostQuitMessage (0) ;
          return 0 ;
     }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

Brushes

- **Types of brushes:**
  - solid - CreateSolidBrush()
  - hatch - CreateHatchBrush()
  - pattern - CreatePatternBrush(), CreateDIBPatternBrushPt()
- **Stock brushes**
  - GetStockObject()
- **Patterns**
  - PatBlt()
  - brush origin: SetBrushOrgEx(), GetBrushOrgEx()

Lines & Curves

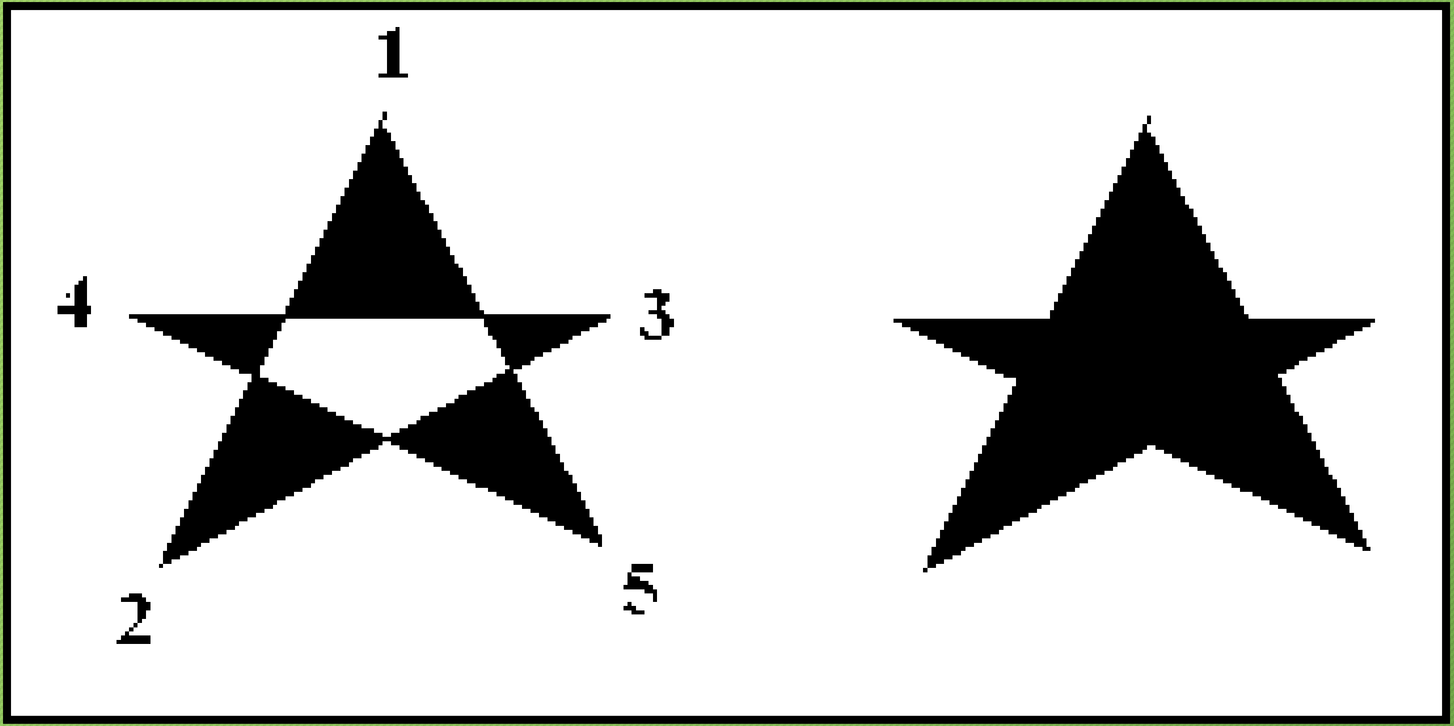
- **Always drawn using the current pen**
- **Current position:**
  - MoveToEx()
- **Lines:**
  - LineTo()
  - PolylineTo(), Polyline(), PolyPolyline()
- **Curves**
  - Arc(), SetArcDirection(), GetArcDirection()
  - PolyBezier()
- **Lines and curves**
  - AngleArc(), PolyDraw()

Filled figures

- **The outline is drawn using the current pen**
- **The interior is filled using the current brush**
  - use GetStockObject(NULL_BRUSH) to draw outline only
- **Rectangles:**
  - Rectangle(), RoundRect()
  - FillRect(), FrameRect(), InvertRect()
- **Other figures:**
  - Ellipse()
  - Chord()
  - Pie()
  - Polygon()

Polygons

- SetPolyFillMode(), GetPolyFillMode()



Rectangles

RECT, *PRECT

- **Operations**
  - SetRect()
  - SetRectEmpty(), IsRectEmpty()
  - EqualRect(), CopyRect()
  - InflateRect(), OffsetRect()
  - PtInRect()
  - IntersectRect(), UnionRect()

Regions

- **HRGN**
- **Creating**
  - CreateRectRgn(), CreateRoundRectRgn(), CreateEllipticRgn(), CreatePolygonRgn()
- **Selecting**
  - SelectObject
- **Filling**
  - FillRgn()
  - SetPolyFillMode(), GetPolyFillMode()
- **Drawing**
  - PaintRgn()

Region operations

- **Combining**
    - CombineRgn()
- **Inversion**
    - InvertRgn()
- **Moving**
    - OffsetRgn()
- **Checking if a point is inside the region**
    - PtInRegion()
- **Getting the bounding rectangle (AABB)**
    - GetRgnBox()

Paths (https://msdn.microsoft.com/en-us/library/windows/desktop/dd145181(v=vs.85).aspx

- **Creating**
  1. BeginPath()
  2. drawing using GDI functions (not all functions are supported)
  3. EndPath()
- **Stroking**
  - StrokePath(), StrokeAndFillPath()
- **Filling**
  - FillPath(), SetPolyFillMode(), GetPolyFillMode()
- **Clipping**
  - SelectClipPath()
- **Converting a path to a region**
  - PathToRegion()

Bitmaps

What is a bitmap?

- Memory representation of an image 😆

„Bitmap term" may differ:

- traditional bitmap or just a way to say image (PNG, JPEG, GIF)
- metadata
- different encodings
- different devices

Device-independent pictures – enhanced-format metafiles.

- CreateEnhMetaFile(), DeleteEnhMetaFile(),
- PlayEnhMetaFile(), CopyEnhMetaFile(), EnumEnhMetaFile(),
- GetEnhMetaFileHeader(), GetEnhMetaFileDescription()

Bitmaps

- Traditional bitmap – a 2D array of pixels, no compression
- Each cell represents a colour

Creating bitmaps:

  - CreateBitmap(),
  - CreateBitmapIndirect() – pass a BITMAP as parameter
  - CreateCompatibleBitmap() – compatible with a device (HDC)
  - DeleteObject()

Getting and setting pixels:

  - GetPixel(), SetPixel()
  - COLORREF

Bitmaps

- **Rotating**
  - PlgBlt()
- **Scaling**
  - StretchBlt(),
  - SetStretchBltMode()
- **Copying:**
  - BitBlt (Bit Block transfer)
- **Using a mask**
  - MaskBlt()

BitBlt – dwRop parameter

Value	Meaning
BLACKNESS	Fills the destination rectangle using the color associated with index 0 in the physical palette. (This color is black for the default physical palette.)
CAPTUREBLT	Includes any windows that are layered on top of your window in the resulting image. By default, the image only contains your window. Note that this generally cannot be used for printing device contexts.
DSTINVERT	Inverts the destination rectangle.
MERGECOPY	Merges the colors of the source rectangle with the brush currently selected in hdcDest, by using the Boolean AND operator.
MERGEPAINT	Merges the colors of the inverted source rectangle with the colors of the destination rectangle by using the Boolean OR operator.
NOMIRRORBITMAP	Prevents the bitmap from being mirrored.
NOTSRCCOPY	Copies the inverted source rectangle to the destination.
NOTSRCERASE	Combines the colors of the source and destination rectangles by using the Boolean OR operator and then inverts the resultant color.
PATCOPY	Copies the brush currently selected in hdcDest, into the destination bitmap.
PATINVERT	Combines the colors of the brush currently selected in hdcDest, with the colors of the destination rectangle by using the Boolean XOR operator.
PATPAINT	Combines the colors of the brush currently selected in hdcDest, with the colors of the inverted source rectangle by using the Boolean OR operator. The result of this operation is combined with the colors of the destination rectangle by using the Boolean OR operator.
SRCAND	Combines the colors of the source and destination rectangles by using the Boolean AND operator.
SRCCOPY	Copies the source rectangle directly to the destination rectangle.
SRCERASE	Combines the inverted colors of the destination rectangle with the colors of the source rectangle by using the Boolean AND operator.
SRCINVERT	Combines the colors of the source and destination rectangles by using the Boolean XOR operator.
SRCPAINT	Combines the colors of the source and destination rectangles by using the Boolean OR operator.
WHITENESS	Fills the destination rectangle using the color associated with index 1 in the physical palette. (This color is white for the default physical palette.)

Bitmaps

- **Transparency**
  - AlphaBlend()
  - TransparentBlt()
- **Gradient**
  - GradientFill()
- **Filling**
  - PatBlt()
  - FloodFill()

Types of bitmaps

- **Device dependent**
  - contains a table of colours
  - BITMAPINFO, BITMAPINFOHEADER, RGBQUAD
  - GetObject()
  - GetDeviceCaps()
- **Device Independent Bitmap - DIB**
  - BITMAP – no table of colours
- **.BMP files**
  - there is no function to save a bitmap to a file, it can be saved manually using structures: BITMAPINFO, BITMAPINFOHEADER, RGBQUAD
  - LoadBitmap() – to read a bitmap from resources
  - LoadImage() – to read a bitmap from resources or a file

Flicker-Free Drawing

- **DoubleBuffering – usually a flag/property in higher level APIs**
- Turn off drawing of the background
- Do nothing in response to the WM_ERASEBKGND message
- Use memory (offline) device context

```
//Create memory DC, bitmap, and select the bitmap
HDC hMemDC = CreateCompatibleDC(hDC);
HBITMAP hBmp = CreateCompatibleBitmap(hDC, nWidth, nHeight);
HBITMAP hOldBmp = (HBITMAP)SelectObject(hMemDC, hBmp);

… //Draw on hMemDC (thus: the new bitmap)

//Copy from hMemDC to real DC
BitBlt(hDC, 0, 0, nWidth, nHeight, hMemDC, 0, 0, SRCCOPY);

//Clean up
SelectObject(hMemDC, hOldBmp);
DeleteObject(hBmp);
DeleteDC(hMemDC);
```

Drawing Transparent Bitmaps

```
//Load bitmap (from file or resource):
//HBITMAP hBmp = (HBITMAP)LoadImage(NULL, filePath,
//                     IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE);
HBITMAP hBmp = LoadBitmap(hInst, MAKEINTRESOURCE(IDB_SARG_MOVE_01));
//Create DC and select the bitmap
BITMAP bmpInfo;
GetObject(hBmp, sizeof(BITMAP), &bmpInfo);
HDC hTmpDC = CreateCompatibleDC(hdc);
HBITMAP hOldBmp = (HBITMAP)SelectObject(hTmpDC, hBmp);

//Copy to destination DC with transparent color set
COLORREF transparentColor = GetPixel(hTmpDC, 0, 0);
TransparentBlt(hdc, 0, 0,
        bmpInfo.bmWidth, bmpInfo.bmHeight, hTmpDC, 0, 0,
        bmpInfo.bmWidth, bmpInfo.bmHeight, transparentColor);

//Clean up
SelectObject(hTmpDC, hOldBmp);
DeleteDC(hTmpDC);
```

Coordinate Space

- **Transformations**
  - SetWorldTransform(), ModifyWorldTransform()
- **Mapping modes**
  - SetMapMode(), GetMapMode()
  - `MM_TEXT, MM_TWIPS`
  - `MM_ANISOTROPIC, MM_ISOTROPIC,`
  - `MM_HIENGLISH, MM_LOENGLISH,` `MM_HIMETRIC, MM_LOMETRIC`
- **Custom coordinate space**
  - SetWindowOrgEx(), SetWindowExtEx()
  - SetViewportOrgEx(), SetViewportExtEx()
- **Converting device points to logical points and vice versa**
  - DPtoLP(), // device points to logical points
  - LPtoDP() // logical points to device points

SetWorldTransform

```
BOOL SetWorldTransform(HDC   hdc, const XFORM *lpXform);
```

```
| eM11 eM12 0 |
| eM21 eM22 0 |
| eDx  eDy   1 |
```

- 2D linear transformations **that can be combined** via matrix multiplication require 3D matrices

Options we have:

- **Translation**
- **Rotation**
- **Reflection**
- **Scaling**
- **Shearing**

Fonts and Text

Fonts

Properties:

- typeface, style, size

Font families:

- decorative, dontcare, modern, roman, script, swiss

Types of fonts:

- raster, vector, TrueType, OpenType

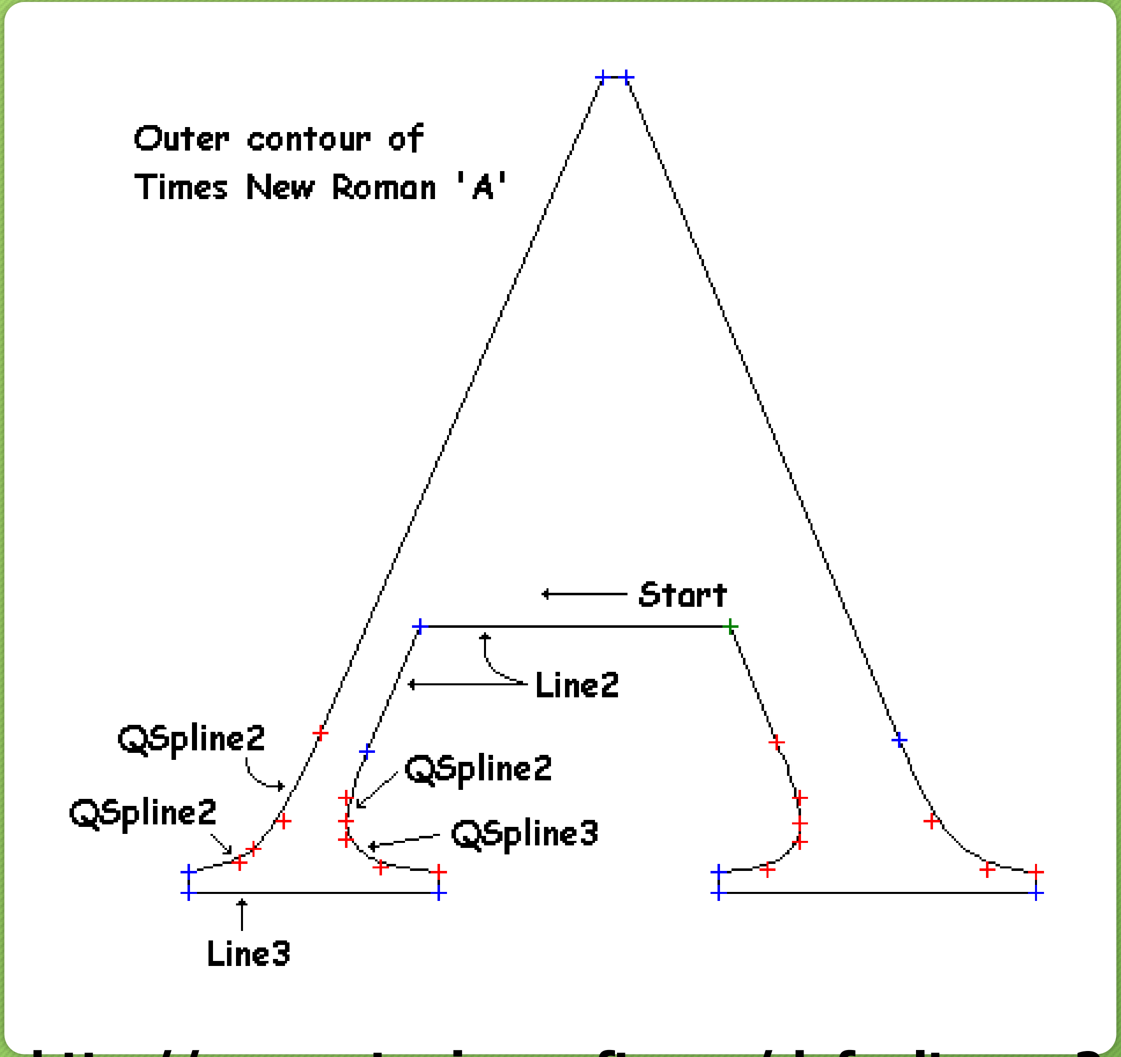
Character Set:

- Windows, Unicode, OEM, symbol

Names:

- Times New Roman, Arial, Verdana

Glyph Outline

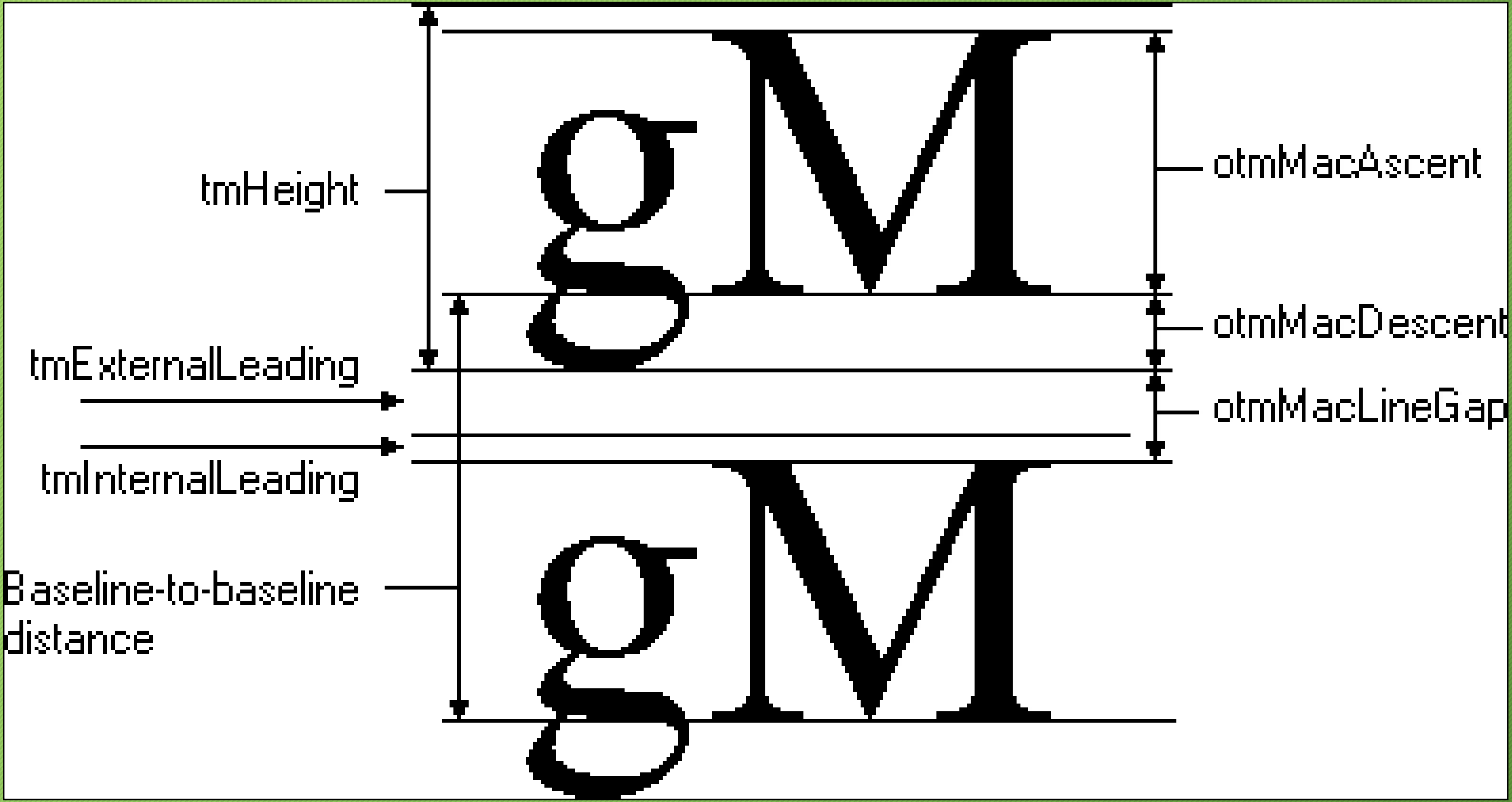


http://support.microsoft.com/default.aspx?scid=kb;en-us;243285
http://my.execpc.com/~dg/tutorial/Glyph/Glyph.html

Using Fonts

- **Creating a font**
  - CreateFont(), CreateFontIndirect()
  - ChooseFont()
- **Selecting a font on a DC as current**
  - SelectObject()
- **Enumerating fonts installed in the system**
  - EnumFonts(), EnumFontFamiliesEx()
- **Getting information about a font**
  - GetFontData(), GetOutlineTextMetrics(), GetGlyphOutline()
- **Installing a font in the system**
  - AddFontResource(), AddFontResourceEx()
  - RemoveFontResource()
  - WM_FONTCHANGE

GetOutlineTextMetrics()



Text

- **Formatting**
  - SetBkColor(), SetBkMode(), SetTextColor()
  - SetTextAlign(), SetTextCharacterExtra()
  - SetTextJustification()
- **Getting size of the drawn text**
  - GetTextExtendPoint32(), GetTabbedTextExtend()
  - GetCharWidth32(), GetCharWidthFloat()
  - GetCharABCWidths(), GetCharABCWidthsFloat()
  - GetTextMetrics(), GetOutlineTextMetrics()
- **Drawing**
  - DrawText(), DrawTextEx()
  - TextOut(), ExtTextOut(), PolyTextOut(), TabbedTextOut()