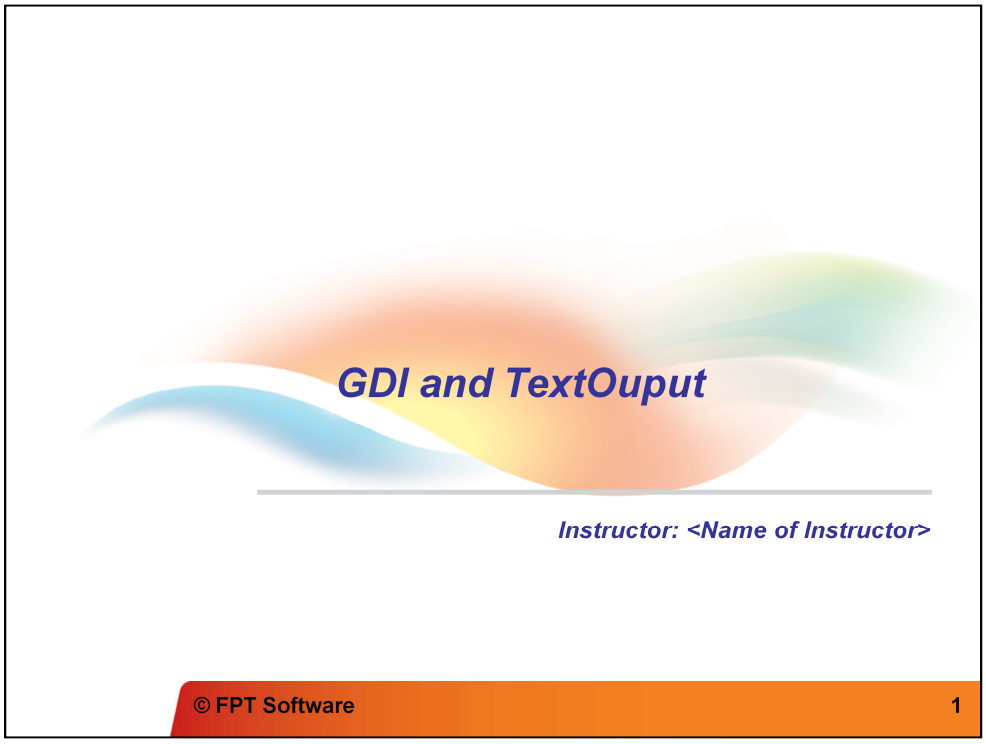


Latest updated by: HanhTT1

Agenda

- Paint and Repaint
- Introduction to GDI
- Scrollbar

The Device Context (1)

- A device context is everything under one name. It is an orchestra, an ensemble of what need in order to draw. It includes the platform you draw on, the dimensioning of the platform, the orientation and other variations of your drawing, the tools you need to draw on the platform, the colors, and various other accessories that can complete your imagination.

The Device Context (2)

- When using a computer, you certainly cannot position tools on the table or desktop for use as needed. To help with drawing on the Windows operating system, Microsoft created the Graphical Device Interface, abbreviated as GDI. It is a set of classes, functions, variables, and constants that group all or most of everything you need to draw on an application. The GDI is provided as a library called Gdi.dll and is already installed on your computer.
- **HDC**: This is the most fundamental class to draw in your applications. It provides all of the primary functions used to perform the basic drawing steps. In order to use this class, first declare a variable from it. Then call the **BeginPaint()** function to initialize the variable using the **PAINSTRUCT** class. Once the variable has been initialized, you can use it to draw. After using the device context call the **EndPaint()** function to terminate the drawing.

The Device Context (3)

- In order to draw using a device context, you must first declare an **HDC** variable. This can be done as follows:

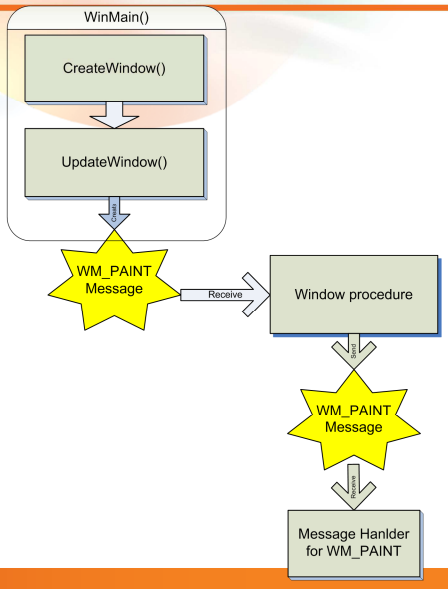
  HDC hDC;
- After declaring this variable, you must prepare the application to paint by initializing it with a call to the **BeginPaint()** function. The syntax of the **BeginPaint()** function is:

  HDC BeginPaint(HWND hWnd, LPPAINTSTRUCT lpPaint);
  - ✓ The *hwnd* argument is a handle to the window on which you will be painting
  - ✓ The *lpPaint* argument is a pointer to the **PAINTSTRUCT** structure. This means that, the **BeginPaint()** function returns two values. It returns a device context as HDC and it returns information about the painting job that was performed. That painting job is stored in a **PAINTSTRUCT** value
- The **PAINTSTRUCT** structure is defined as follows:

```
typedef struct tagPAINTSTRUCT {
        HDC hdc;
        BOOL fErase;
        RECT rcPaint;
        BOOL fRestore;
        BOOL fIncUpdate;
        BYTE rgbReserved[32];
} PAINTSTRUCT, *PPAINTSTRUCT;
```

WM_PAINT Message

- Informs to window procedure that window's client area needs be repainting



WM_PAINT Message

- Window procedure receives WM_PAINT message when :
  - A previous hidden area of the window is brought to view
  - Change size of window
  - Use **ScrollWindow()** or **ScrollDC()** function for scroll window's client area
  - Use **InvalidateRect()** or **InvalidateRgn()** function to explicitly generate a WM_PAINT message

Invalid & Valid Rectangle

- Actually, window procedure needs to update only a small are in client area of window. This area is known as “invalidate region” or “update region”
- Window procedure will receive a WM_PAINT message only if part of client area is invalid
- The smallest rectangle that encompasses the invalid region is known as “invalid rectangle”

Getting Device Context

- Method 1 : Use in WM_PAINT event handler
  - PAINTSTRUCT ps;
  - HDC hDC;
  - Case WM_PAINT :
    - hDC = BeginPaint(hWnd, &ps);
    - [Use GDI's functions for drawing to window]
    - EndPaint(hWnd, &ps);
    - Return 0;

Getting Device Context

- Method 2 : Use in other message hanlders
  - HDC hDC;
  - hDC = GetDC (hWnd);
  - [Use GDI's functions]
  - ReleaseDC (hWnd, hDC);

Graphic Drawing Functions (1)

- **COLORREF GetPixel**(**HDC** hDC, **int** nXPos, **int** nYPos);

  Lấy về giá trị màu tại vị trí (nXPos, nYPos) của hDC, trả về -1 nếu điểm này nằm ngoài vùng hiển thị.
- **COLORREF SetPixel**(**HDC** hDC, **int** nXPos, **int** nYPos, **COLORREF** clrRef);

  Vẽ một điểm màu clrRef tại vị trí (nXPos, nYPos) lên hDC. Giá trị trả về là màu của điểm (nXPos, nYPos) hoặc -1 nếu điểm này nằm ngoài vùng hiển thị.
- **DWORD MoveToEx**(**HDC** hDC, **int** x, **int** y);

  Di chuyển bút vẽ đến tọa độ (x, y) trên hDC. Giá trị trả về là tọa độ cũ của bút vẽ, x = LOWORD, y = HIWORD.
- **BOOL LineTo**(**HDC** hDC, **int** xEnd, **int** yEnd);

  Vẽ đoạn thẳng từ vị trí hiện hành đến vị trí (xEnd, yEnd) trên hDC. Hàm trả về TRUE nếu thành công, FALSE nếu thất bại.

Graphic Drawing Functions (2)

- **BOOL Polyline(HDC hdc, CONST POINT *lppt, int cPoints);**

  BOOL PolylineTo(HDC hdc, CONST POINT *lppt, DWORD cCount);

  Draw a series of connected lines

- **BOOL PolyPolyline(HDC hdc, CONST POINT *lppt, CONST DWORD *lpdwPolyPoints, DWORD cCount);**

  draw various polylines in one step

- **BOOL Polygon(HDC *hdc*, CONST POINT **lpPoints*, int *nCount*);**

  A polygon is a closed polyline. In other words, it is a polyline defined so that the end point of the last line is connected to the start point of the first line.

- **BOOL Rectangle(HDC *hdc*, int *nLeftRect*, int *nTopRect*, int *nRightRect*, int *nBottomRect*);**

  Draw a rectangle

TextOut() Function

- Most common GDI function for display text on window
- Prototype :

```
– BOOL TextOut(
    HDC     hdc,          // handle to DC
    int     nXStart,      // x-coordinate of starting position
    int     nYStart,      // y-coordinate of starting position
    LPCTSTR lpString,     // character string
    int     cbString      // number of characters
  );
```

TextOutput

- System Font
- Size of a Character
- Text Metrics
- Formatting Text

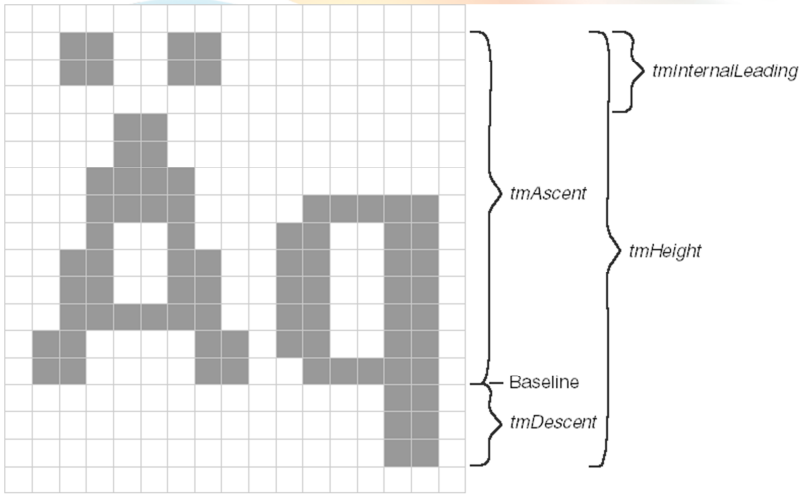
Size of Character

- Information about size of character is stored in struct TEXTMETRIC

  – typedef struct tagTEXTMETRIC {

```
        LONG            tmHeight ;
        LONG            tmAscent ;
        LONG            tmDescent ;
        LONG            tmInternalLeading ;
        LONG            tmExternalLeading ;
        LONG            tmAveCharWidth ;
        LONG            tmMaxCharWidth ;
        [other structure fields]
    } TEXTMETRIC, * PTEXTMETRIC ;
```

Size of Character



TextOut function example

```
HFONT hfnt, hOldFont;
hfnt = GetStockObject(ANSI_VAR_FONT);
if (hOldFont = SelectObject(hdc, hfnt)) {
    TextOut(hdc, 10, 50, “Display sample font”, 18);
    SelectObject(hdc, hOldFont);
}
```


Q&
A