Atrybuty kontekstu – obiekty GDI

Obiekt	Zastosowanie	Tworzenie
Pióro (ang. *pen)*	sposób kreślenia linii i krzywych	***CreatePen CreatePenIndirect***
Pędzel (ang. brush)	sposób wypełniania wnętrz	***CreateSolidBrush CreateBrushIndirect CreateHatchBrush***
Czcionka (ang. font)	sposób malowania napisów	***CreateFont CreateFontIndirect***
Region obcinania (ang. clipping region)	sposób obcinania wyjścia funkcji graficznych	***CreateEllipticRgn CreatePolygonRgn CreateRectangleRgn CreateRectangleRgn***
Paleta kolorów	Sposób wyświetlania obrazów w trybach 8-bitowych	***CreatePalette***
Mapa bitowa		***CreateBitmap CreateDIBitmap CreateCompatibleBitmap***

Obiekty GDI – przykład

fragment procedury okna rysujący prostokąt o przerywanych czerwonych krawędziach wypełniony na zielono:

```
case WM_PAINT:
{
        HDC hDC = BeginPaint(hWnd, &ps);

        // utworzenie obiektów GDI (pióra i pędzla)
        HPEN hPen = CreatePen(PS_DOT,0,RGB(255,0,0));
        HBRUSH hBrush = CreateSolidBrush(RGB(0,255,0));

        // wybranie pióra i pędzla
        HPEN hOldPen=(HPEN)SelectObject(hDC,hPen);
        HBRUSH hOldBrush=(HBRUSH)SelectObject(hDC,hBrush);

        Rectangle(hDC,0,0,100,100);

        // Przywrócenie stanu domyślnego
        SelectObject(hDC,hOldPen);    // ponownie wybieramy „stare” pióro
        SelectObject(hDC,hOldBrush); // ponownie wybieramy „stary” pędzel

        DeleteObject(hBrush);          // usuwamy
        DeleteObject(hPen);            // obiekty GDI

        EndPaint(hWnd, &ps);
        break;
}
```

Przekształcenie współrzędnych logicznych na współrzędne urządzenia

xWin = (**xView** - xViewOrg) * (xWinExt/xViewExt)+xWinOrg
yWin = (**yView** - yViewOrg) * (yWinExt/yViewExt)+yWinOrg

(xWin, yWin) – współrzędne urządzenia (domyślnie (0,0))
(xView, yView) – współrzędne logiczne (domyślnie (0,0))

(*xWinOrg, yWinOrg*) - początek okna we współrzędnych urządzenia; zmiana funkcją ***SetWindowOrgEx.***

(*xViewOrg, yViewOrg*) - początek widoku we współrzędnych logicznych; zmiana funkcją ***SetViewportOrgEx.***

Reguła: Punkt (xViewOrg, yViewOrg) zostaje przekształcony w punkt (xWinOrg, yWinOrg)

xWinExt, xViewExt, yViewExt, yWinExt – rozciągłość okna i widoku.

Tryb odwzorowania (ang. *mapping mode*) jest zmieniany funkcją ***SetMapMode.*** Domyślny tryb to MM_TEXT, w którym cztery parametry *Ext są równe 1 (jeden piksel logiczny odpowiada jednemu pikselowi urządzenia).

Inne tryby odwzorowania pozwalają na używanie jednostek fizycznie znaczących np. cali lub centymetrów (użyteczne przy drukowaniu)

Pozostałe atrybuty kontekstu urządzenia

Atrybut	Określa	Zmiana
kolor tła (ang. *backgro und cololor*)	kolor przerw pomiędzy symbolami w tekście w piórach, i pędzlach	***SetBkColor***
tryb tła (ang. *backgro und cololor*)	**OPAQE** - przerwy wypełniane kolorem tła. **TRANSPARENT** - tło nie zmieniane	***SetBkMode***
tryb kreślenia (ang. *drawing mode*)	operacja bitowa wykonywana przy kreśleniu i wypełnianiu.	***SetROP2***
tryb rozciągania mapy bitowej (ang. *stretching mode*)		***SetStretchBltMode***
Tryb wypełniania wielokątów (ang. *polygon filling mode*)		***SetPolyFillMode***
Odstęp pomiędzy znakami (ang. *interspace spacing)*		***SetTextCharacterExtra***

Funkcje zmieniające wartości posiadają odpowiedniki czytające wartości zaczynające się od **Get** (np. ***GetROP2***).

Funkcje graficzne GDI

Kreślenie	Funkcje GDI
linii, łamanych	***MoveToEx, LineTo, PolyLine, PolyLineTo***
tekstów	***TextOut, ExtTextOut, DrawText, DrawTextEx, TabbedTextOut***
wielokątów	***Rectangle, RoundRect, Polygon, PolyPolygon***
łuków, elips i ich fragmentów	***Ellipse, Pie, Chord, Arc, AngleArc, ArcTo***
krzywych Bezier'a	***PolyBezier, PolyBezierTo,***
map bitowych	***BitBlt, StretchBlt***
map DIB	***SetDIBitsToDevice, SetDIBits, StretchDIBits***
wypełnianie wnętrza	***FloodFill, ExtFloodFill***

Do uzyskania informacji o kontekście służy funkcja ***GetDeviceCaps.***

Wykorzystanie pamięciowego kontekstu urządzenia

```
case WM_PAINT:
{
        HDC hDC = BeginPaint(hWnd, &ps);

        HDC hDCMem=CreateCompatibleDC(hDC);
        HBITMAP hOldBitmap=(HBITMAP)SelectObject(hDCMem,hBitmap);

        TextOut(hDCMem,100,100,"Tekst",5); // rysujemy do mapy bitowej !!!

        BitBlt(hDC,0,0,300,300,hDCMem,0,0,SRCCOPY);

        SelectObject(hDCMem,hOldBitmap);

        DeleteDC(hDCMem);
        EndPaint(hWnd, &ps);
        break;
}
```

Obsługa klawiatury

WM_KEYDOWN, **WM_KEYUP** – generowane odpowiednio przy wciśnięciu i puszczeniu klawisza. Jeżeli klawisz przytrzymywany jest dostatecznie **WM_KEYDOWN** generowany jest wielokrotnie.
wParam – kod klawisza, dla klawiszy funkcyjnych zdefiniowane są stałe (VK_SHIFT, VK_ENTER, ...)
lParam – informacje rozszerzone, młodsze 16 bitów zawiera licznik powtórzeń.

GetKeyState – pozwala sprawdzić stan innego klawisza **w chwili wygenerowania** komunikatu.

WM_CHAR –generowany jest dla klawiszy generujących znaki (np. litery, cyfry) z uwzględnieniem stanu klawiszy specjalnych. Za wstawianie tego komunikatu do kolejki odpowiada funkcja ***TranslateMessage*** w pętli komunikatów.
wParam – kod znaku
lParam – informacje rozszerzone, młodsze 16 bitów zawiera licznik powtórzeń.

Obsługa zegara

UINT ***SetTimer***(HWND hWnd, UINT ID, UINT uElapse, TimerProc lpTimer)
BOOL ***KillTimer***(HWND hWnd, UINT ID);

ID – identyfikator zegara (można mieć kilka o różnych interwałach)
uElpase – czas w milisekundach, co jaki powinien być aktywowany zegar (do momentu wywołania ***KillTimer***)

Metoda 1 (Parametr lpTimer==NULL);

Procedura okna otrzymuje komunikat **WM_TIMER** (wParam zawiera identyfikator)

Metoda 2 Parametr lpTimer wskazuje na funkcję o prototypie:

void CALLBACK ***TProc***(HWND hWnd, UINT msg, UINT ID, DWORD Time)

msg == **WM_TIMER**
ID – identyfikator zegara
Time – czas systemowy w milisekundach od momentu startu systemu