Programowanie niskopoziomowe

Laboratorium 09 – Materiały uzupełniające

„Tryb graficzny”

Cześć materiałów autorstwa prof. A. Timofiejewa (fragmenty książki)

7.7. Programowanie aplikacji graficznej

7.7.1. Współdziałanie aplikacji graficznej z systemem Windows

Aplikacja graficzna zawsze ma okno. Każdy graficzny element okna jest rozpatrywany też jako okno.

Komponenty systemy Windows współdziałają za pomocą komunikatów o zdarzeniach. Aplikacja analizuje i przerabia komunikat w tzw. *procedurze okna*.

System Windows otrzymuje wskaźnik na procedurę okna aplikacji w strukturze `WNDCLASS`. Aplikacja musi przygotować na początku programu tą strukturę i zarejestrować ją za pomocą funkcji `RegisterClass`. Na bazie struktury `WNDCLASS` system Windows tworzy okno, gdy aplikacja wywołuje funkcję `CreateWindowsEx`. System Windows wyświetla okno po wywołaniu funkcji `ShowWindow`.

System Windows kieruje komunikaty o zdarzeniach do procedury okna, a wykorzystanie komunikatu zależy od zawartości tej procedury. Komunikaty nie są wysyłane od razu do procedury okna. Aby aplikacja otrzymywała tylko „swoi” komunikaty, w aplikacji musi działać tzw. *pętla obsługi komunikatów*.

Wewnątrz pętli aplikacja wywołuje funkcję `GetMessage`, która pobiera bieżący komunikat z kolejki komunikatów Windows. Wyjście z pętli obsługi komunikatów musi nastąpić, gdy funkcja `GetMessage` zwróci wartość zerową.

W pętle komunikatów aplikacja wywołuje funkcję `DispatchMessage` i przez nią przekazuje komunikat do systemu Windows, który kieruje komunikat do procedury okna. Przed wywołaniem funkcji `DispatchMessage` zwykle ma miejsce wywołanie funkcji `TranslateMessage`, która przekształca część komunikatów od klawiatury.

A więc, aplikacja graficzna dla Windows ma następującą strukturę:

- zapełnienie pól struktury `WNDCLASS`, w tym wpisywanie adresu procedury okna,
- rejestracja struktury za pomocą funkcji `RegisterClass`,
- kreacja okna za pomocą funkcji `CreateWindowsEx`,
- wyświetlenie okna za pomocą funkcji `ShowWindow`,
- pętla komunikatów – pobranie, analiza na komunikat zakończenia, przekształcenie opcjonalne i wysyłanie komunikatów z powrotem do systemu Windows,
- zamykanie procesu za pomocą funkcji `ExitProcess`.

Dodatkowo należy napisać procedurę okna w postaci podprogramu z tradycyjną nazwą `WndProc`. Faktycznie w tej procedurze (funkcji) jest schowana specyfika aplikacji.

Przykład 7.5. Aplikacja graficzna z tworzeniem okna (program w języku asemblera MASM32)

```
.586P
.MODEL flat, STDCALL
;--- pliki ---------
include .\include\grafika.inc
;--- biblioteki ---------
includelib .\lib\user32.lib
includelib .\lib\kernel32.lib
includelib .\lib\gdi32.lib
;--- stałe ---
CSstyle EQU CS_HREDRAW + CS_VREDRAW + CS_GLOBALCLASS
WNDstyle EQU WS_OVERLAPPEDWINDOW
;--- sekcja danych ------
_DATA SEGMENT
 hwnd        DD  0
 hinst   DD  0
```

```
 ;---
 msg MSGSTRUCT <?>
 wndc WNDCLASS <?>
 ;---
 tytul   DB "Aplikacja graficzna",0
 ALIGN 4
 cname   DB "MainClass", 0
 ALIGN 4
 nagl        DB  "Komunikat", 0
 ALIGN 4
 terr1   DB  "Błąd 1!", 0
 ALIGN 4
 terr2   DB  "Błąd 2!", 0
 ALIGN 4
 terr3   DB  "Błąd 3!", 0
_DATA ENDS
;--- sekcja kodu ---------
_TEXT SEGMENT
;;;;;;;;;;;;;;;;;;;;;;
WndProc PROC
;--- procedura okna ---
(jest przytoczona niżej)
WndProc  ENDP
;--- start programu ---
Start:
;--- deskryptor aplikacji ----
 INVOKE GetModuleHandleA, 0
 mov hinst, EAX
;--- wypełnienie struktury okna WNDCLASS
 mov EAX, [hinst]
 mov [wndc.clsHInstance], EAX
 mov [wndc.clsStyle], CSstyle
 mov [wndc.clsLpfnWndProc], OFFSET WndProc
 mov [wndc.clsCbClsExtra], 0
 mov [wndc.clsCbWndExtra], 0
 INVOKE LoadIconA, 0, IDI_APPLICATION  ; ikona
 mov [wndc.clsHIcon], EAX
 INVOKE LoadCursorA, 0,   IDC_ARROW ; kursor
 mov [wndc.clsHCursor], EAX
 INVOKE GetStockObject, WHITE_BRUSH    ; tło
 mov [wndc.clsHbrBackground], EAX
 mov [wndc.clsLpszMenuName], 0
 mov DWORD PTR [wndc.clsLpszClassName], OFFSET cname
;--- rejestracja okna ---
 INVOKE RegisterClassA, OFFSET wndc
 cmp EAX, 0   ; analiza na błąd
 jne @F
 jmp err1
@@:
;--- utworzenie okna ---
 INVOKE CreateWindowExA, 0, OFFSET cname, OFFSET tytul, \
WNDstyle, 100, 100, 400, 400, 0, 0, hinst, 0
 cmp EAX, 0   ; analiza na błąd
 jne @F
 jmp err2
@@:
 mov hwnd, EAX
 INVOKE ShowWindow, hwnd, SW_SHOWNORMAL
 INVOKE UpdateWindow, hwnd
;--- pętla obsługi komunikatów ---
```

```
msgloop:
;--- pobranie komunikatu ---
 INVOKE GetMessageA, OFFSET msg, 0, 0, 0
 cmp EAX, 0  ; analiza na zakończenie procesu
 jne @F
 jmp etkon
@@:
 cmp EAX, -1  ; analiza na błąd
 jne @F
 jmp err3
@@:
;---przekształcenie komunikatu: ---
 INVOKE TranslateMessage, OFFSET msg
;---przekazywanie komunikatu do Windows: ---
 INVOKE DispatchMessageA, OFFSET msg
 jmp msgloop
;--- obsługa błędów ---------
err1:
;--- okno z komunikatem o błędzie----
 INVOKE MessageBoxA,0,OFFSET terr1,OFFSET nagl,0
 jmp etkon
err2:
;--- okno z komunikatem o błędzie----
 INVOKE MessageBoxA,0,OFFSET terr2,OFFSET nagl,0
 jmp etkon
err3:
;--- okno z komunikatem o błędzie----
 INVOKE MessageBoxA,0,OFFSET terr3,OFFSET nagl,0
 jmp etkon
;--- zakończenie procesu ---------
etkon:
 INVOKE ExitProcess, [msg.msWPARAM]
_TEXT   ENDS
END Start
```

Plik grafika.inc zawiera deklaracje stałych oraz deklaracje zastosowanych funkcji API.

7.7.2. Procedura okna

Procedura okna musi być napisana według reguł wywołania `STDCALL`. Nazwa procedury może być każda, ale tradycyjnie jest stosowana nazwa `WndProc`. Argumenty procedury w kolejności z lewej strony na prawo:

- deskryptor okna (typ `DWORD`),
- identyfikator komunikatu (typ `DWORD`),
- parametr `WPARAM` komunikatu (typ `DWORD`),
- parametr `LPARAM` komunikatu (typ `DWORD`).

Komunikaty nieobsługiwane przez procedurę okna muszą być przekazane funkcji `DefWindowsProc`, która ma argumenty podobne do argumentów procedury okna. Ta funkcja może zmienić zawartość rejestrów `EBX, ESI, EDI`. Dlatego w procedurze okna te rejestry są odkładane na stos.

Procedura okna musi oczyścić stos. Uwzględniając wyżej wymienione powody należy stosować następującą strukturę procedury okna:

```
WndProc  PROC
     push EBP ; standardowy prolog
     mov  EBP, ESP ; standardowy prolog
;--- odkładanie na stos
     push EBX
     push ESI
     push EDI
;--- analiza komunikatu
     cmp  DWORD PTR [EBP+0Ch], WM_CREATE
     jne  @F
```

```
    jmp wmcreate
@@: cmp DWORD PTR [EBP+0Ch], WM_DESTROY
    jne @F
    jmp wmdestroy
@@: ; ...
;--- komunikaty nieobsługiwane ---
    push DWORD PTR [EBP+14h] ; parametr LPARAM komunikatu
    push DWORD PTR [EBP+10h] ; parametr WPARAM komunikatu
    push DWORD PTR [EBP+0Ch] ;identyfikator komunikatu
    push DWORD PTR [EBP+08h] ;deskryptor okna
    call DefWindowProc
    jmp kon
wmcreate:
    mov EAX, 0
    jmp kon
wmdestroy:
    push 0
    call PostQuitMessage  ; wysyłanie WM_QUIT
    mov EAX, 0
    jmp kon
;--- zdejmowanie ze stosu
kon: pop EDI
    pop ESI
    pop EBX
    mov ESP,EBP ; standardowy epilog
     pop EBP ; standardowy epilog
    ret 16  ; zwolnienie komórek stosu
WndProc ENDP
```

Przykład procedury okna dla programu przykładu 7.5:

```
WndProc PROC
;--- procedura okna ---
; DWORD PTR [EBP+14Ch] - parametr LPARAM komunikatu
; DWORD PTR [EBP+10h] - parametr WPARAM komunikatu
; DWORD PTR [EBP+0Ch] - ID = identyfikator komunikatu
; DWORD PTR [EBP+08h] - HWND = deskryptor okna
 ;---------------------------------
 push   EBP ; standardowy prolog
 mov EBP, ESP ; standardowy prolog
;--- odkładanie na stos
 push   EBX
 push   ESI
 push   EDI
;---
 cmp DWORD PTR [EBP+0Ch], WM_CREATE
 jne @F
 jmp wmCREATE
@@:
 cmp DWORD PTR [EBP+0Ch], WM_DESTROY
 jne @F
 jmp wmDESTROY
@@:
 cmp DWORD PTR [EBP+0CH], WM_COMMAND
 jne @F
 jmp wmCOMMAND
@@:
 cmp DWORD PTR [EBP+0CH], WM_LBUTTONDOWN
 jne @F
 jmp wmLBUTTON
@@:
 cmp DWORD PTR [EBP+0CH], WM_RBUTTONDOWN
```

```
 jne @F
 jmp wmRBUTTON
@@:
;--- komunikaty nieobsługiwane ---
 INVOKE DefWindowProcA, DWORD PTR [EBP+08h], \
DWORD PTR [EBP+0Ch], DWORD PTR [EBP+10h], DWORD PTR [EBP+14h]
 jmp konWNDPROC
wmCREATE:
 jmp konWNDPROC
wmDESTROY:
 INVOKE PostQuitMessage, 0    ; wysyłanie WM_QUIT
 mov EAX, 0
 jmp konWNDPROC
wmCOMMAND:
... ; działania po naciśnięciu klawisza klawiatury
 mov EAX,0
 jmp konWNDPROC
wmRBUTTON:
... ; działania po naciśnięciu prawego przycisku myszy
 mov EAX,0
 jmp konWNDPROC
wmLBUTTON:
... ; działania po naciśnięciu lewego przycisku myszy
 mov EAX,0
 jmp konWNDPROC
;--- zdejmowanie ze stosu
konWNDPROC:
 pop EDI
 pop ESI
 pop EBX
 mov ESP,EBP  ; standardowy epilog
 pop EBP ; standardowy epilog
 ret 16  ; zwolnienie komórek stosu z argumentami
WndProc  ENDP
```

Komunikaty, które nie są obsługiwane procedurą okna, muszą być przekazany do funkcji DefWindowProc. Jeśli procedura okna obrabia komunikat, ona musi zwrócić zero w rejestrze EAX.

7.7.3. Standardowe obiekty graficzne

Każdy obiekt graficzny jest rozpatrywany jako okno (okienko) systemu *Windows* i jest charakteryzowany przez swoją procedurę okna. Dla standardowych obiektów graficznych procedury okienne są przechowywane w bibliotekach, na przykład w bibliotece `user32.lib`.

Dla standardowego obiektu graficznego nie jest potrzebna rejestracja klasy, co oznacza, ze nie jest potrzebne zapełnienie struktury `WNDCLASS` i wywołanie funkcji `RegisterClass`.

System Windows zawiera następujące klasy standardowych obiektów graficznych:

`BUTTON` - prostokąt, który zmienia wygląd, jeśli użytkownik naciśnie myszą na pole prostokąta,

`COMBOBOX` – kontrolka zawierająca listę rozwijaną (ang. list box) i pole edycyjne (ang. edit control),

`EDIT` - prostokątne pole edycyjne, które przyjmuje znaki z klawiatury,

`LISTBOX` - lista rozwijana, w której można zaznaczyć element (linijkę). Prostokąt z listą może zawierać poziomowy oraz pionowy suwak,

`MDICLIENT` - okno potomne (ang. MDI client window) umieszczone w polu okna-rodzica, do którego są przekazywane komunikaty o zdarzeniach,.

`SCROLLBAR` – suwak,

`STATIC` – pole, które nie produkuje zdarzeń i komunikatów.

Każde z tych okien może mieć następujący styl lub kombinację stylów:

WS_BORDER – kreacja okna (kontrolki) z wąziutką otoczką,

WS_CAPTION – kreacja paska tytułowego,

WS_DLGFRAME – kreacja okna z otoczką podobną do otoczki okna dialogowego (okno nie może mieć paska tytułowego),

WS_CHILD (lub WS_CHILDWINDOW) – kreacja okna potomnego,

WS_CLIPCHILDREN – wyeliminuje obszar zajmowany oknem potomnym przed przerysowaniem, gdy jest rysowane okno-rodzic,

WS_CLIPSIBLINGS – wyeliminuje obszar zajmowany oknem potomnym przed przerysowaniem, gdy są rysowane inne okna potomne,

WS_DISABLED – kreacja okna (kontrolki), które nie produkuje komunikatów o zdarzeniach,

WS_GROUP – razem z WS_TABSTOP zaznacza kontrolkę pierwszą w grupie; następne kontrolki należą do grupy,

WS_HSCROLL – kreacja okna (kontrolki) zawierającego suwak poziomowy,

WS_VSCROLL – kreacja okna (kontrolki) zawierającego suwak pionowy,

WS_MINIMIZE (lub WS_ICONIC) kreacja okna zminimalizowanego,

WS_MAXIMIZE - kreacja okna maksymalizowanego,

WS_MAXIMIZEBOX - kreacja okna mającego przycisk maksymalizacji,

WS_MINIMIZEBOX - kreacja okna mającego przycisk minimalizacji,

WS_OVERLAPPED (lub WS_TILED) - kreacja okna mającego pasek tytułowy i otoczkę,

WS_OVERLAPPEDWINDOW (lub WS_TILEDWINDOW) - kreacja okna z opcjami WS_OVERLAPPED, WS_CAPTION, WS_SYSMENU, WS_THICKFRAME, WS_MINIMIZEBOX i WS_MAXIMIZEBOX,

WS_POPUP - kreacja okna „pop-up”,

WS_POPUPWINDOW - kreacja okna „pop-up” z opcjami WS_BORDER, WS_POPUP i WS_SYSMENU

WS_SIZEBOX (lub WS_THICKFRAME) - kreacja okna, którego rozmiar można zmieniać,

WS_SYSMENU - kreacja okna mającego menu,

WS_TABSTOP - kreacja okna (kontrolki), które może otrzymywać ognisko, gdy użytkownik naciśnie klawisz TAB,

WS_VISIBLE - kreacja okna (kontrolki), które jest widoczne.

Obiekt (przycisk) typu BUTTON może mieć następujące opcje do ustawienia właściwości:

BS_3STATE – kreacja przyciska podobnego do okienka wyboru (ang. check box), które może być w stanie niezdefiniowanego wyboru zaznaczanym kolorem szarym,

BS_AUTO3STATE – kreacja przyciska podobnego do przyciska z opcją BS_3STATE, które różni się automatyczną zmianą w kolejności stanów: „wybrany”, „niezdefiniowany”, „niewybrany”,

BS_AUTOCHECKBOX – kreacja przyciska, który automatycznie zmienia stan w kolejności: „wybrany”, „niewybrany”,

BS_AUTORADIOBUTTON – kreacja przyciska, którego wybór zmienia stan innych przycisków grupy na stan „niewybrany”,

BS_CHECKBOX – kreacja prostokątnego przyciska wybory (ang. check box) razem z tekstem,

BS_DEFPUSHBUTTON – kreacja przyciska, który produkuje zdarzenie podobne do naciśnięcia klawiszu ENTER,

BS_GROUPBOX – kreacja prostokąta, wewnątrz którego znajdują się kontrolki grupy,

BS_LEFTTEXT (lub BS_RIGHTBUTTON) – opcja do umieszczenia tekstu z lewej strony od przycisku wyboru,

BS_OWNERDRAW (lub BS_USERBUTTON w 16-bitowych Windows) – kreacja przyciska, który jest rysowany po otrzymaniu komunikatu WM_MEASUREITEM lub przerysowany po otrzymaniu komunikatu WM_DRAWITEM,

BS_PUSHBUTTON – kreacja przyciska, który wysyła komunikat WM_COMMAND w przypadku wyboru przyciska,

BS_RADIOBUTTON – kreacja okrągłego przyciska wybory (ang. check box) razem z tekstem,

BS_BITMAP – kreacja przyciska, na którym jest rysowana bitmapa,

BS_ICON – kreacja przyciska, na którym jest rysowana ikona,

BS_TEXT – kreacja przyciska z tekstem,

BS_BOTTOM, BS_LEFT, BS_RIGHT, BS_TOP, BS_CENTER – opcje do umieszczenia tekstu na dole, z lewej strony, z prawej strony, na górze lub w środku prostokąta przyciska,

BS_VCENTER – opcja do umieszczenia tekstu pionowo w środku prostokąta przyciska,

BS_MULTILINE – opcja do umieszczenia tekstu na wiele linijkach,

BS_NOTIFY – zezwolenie na wysyłanie komunikatów BN_DBLCLK, BN_KILLFOCUS i BN_SETFOCUS,

BS_PUSHLIKE – przekształca przycisk wyboru, przycisk na trzy stany lub przycisk „radio button” na „zwykły” przycisk.

Obiekty typu COMBOBOX można zdefiniować z następującymi opcjami stylu:

CBS_AUTOHSCROLL – automatyczne przesuwanie tekstu przy wprowadzeniu symbolu na końcu,

CBS_DISABLENOSCROLL – wyświetlenie suwaka pionowego niezależnie od ilości pokazanych elementów,
CBS_DROPDOWN – rozwiniecie listy przy naciśnięciu na pole edytowania,
CBS_DROPDOWNLIST – rozwiniecie listy przy naciśnięciu na pole edytowania oraz wyświetlenie w tym polu wybranej linijki,
CBS_HASSTRINGS – informuje, że każdy element listy jest wierszem (aby otrzymać tekst elementu należy zastosować komunikat CB_GETLBTEXT ze strony aplikacji),
CBS_LOWERCASE, CBS_UPPERCASE – konwersja na małe lub duże litery,
CBS_NOINTEGRALHEIGHT – rozmiar prostokąta nie jest zmieniany w zależności od ilości elementów,
CBS_OEMCONVERT – konwersja znaków na znaki ze zbioru symboli OEM (wprowadzono dla list rozwijanych z nazwami plików),
CBS_OWNERDRAWFIXED – wskazuje, że właściciel listy rysuje elementy (właściciel okienka otrzymuje komunikat WM_MEASUREITEM przy kreacji obiektu oraz komunikat WM_DRAWITEM przy każdej wizualnej zmianie obiektu),
CBS_OWNERDRAWVARIABLE – wskazuje, że właściciel listy rysuje elementy oraz wysokość linijek może być zmienna (właściciel okienka otrzymuje komunikat WM_MEASUREITEM przy kreacji obiektu oraz komunikat WM_DRAWITEM przy każdej wizualnej zmianie obiektu),
CBS_SIMPLE – lista jest pokazywana na stałe,
CBS_SORT – sortowanie wierszy w kolejności alfabetyczną,

Do obiektów typu EDIT mają zastosowanie następujące opcji:
ES_AUTOHSCROLL – automatyczne przesuwanie tekstu o 10 znaków przy wprowadzeniu symbolu na końcu (naciśnięcie klawiszu ENTER powoduje powrót tekstu do pozycji początkowej),
ES_AUTOVSCROLL – automatyczne przesuwanie tekstu o jedną stronę przy naciśnięciu klawiszu ENTER na ostatniej linijkę,
ES_CENTER – środkowanie tekstu w polu obiektu,
ES_LEFT, ES_RIGHT – rozmieszczenie tekstu z lewej lub prawej strony pola obiektu,
ES_LOWERCASE, ES_UPPERCASE – konwersja na małe lub duże litery,
ES_MULTILINE – umieszczenie tekstu na wiele linijkach (aby klawisz ENTER wprowadzał nowy wiersz musi być ustawiona opcja ES_WANTRETURN),
ES_NOHIDESEL – zostawia zaznaczenie znaków, jeśli obiekt traci ognisko,
ES_NUMBER – obiekt przyjmuje tylko cyfry,
ES_OEMCONVERT – konwersja znaków na znaki ze zbioru symboli OEM (wprowadzono dla pól edycyjnych z nazwami plików),
ES_PASSWORD - wyświetla znak „*” zamiast wprowadzonego znaku (komunikat EM_SETPASSWORDCHAR zamienia znak „*” na inny),
ES_READONLY – chroni tekst od zmiany,
ES_WANTRETURN – naciśnięcie klawiszu ENTER wprowadza nowy wiersz.

Opcje obiektów typu LISTBOX:
LBS_DISABLENOSCROLL – wyświetlenie suwaka pionowego niezależnie od ilości pokazanych elementów,
LBS_EXTENDEDSEL – zezwolenie na zaznaczanie wiele linijek (przy naciśniętym klawiszu SHIFT),
LBS_HASSTRINGS – informuje, że każdy element listy jest wierszem (aby otrzymać tekst elementu należy zastosować komunikat CB_GETTEXT ze strony aplikacji),
LBS_MULTICOLUMN – ustawia tryb „wiele kolumn” (komunikat LB_SETCOLUMNWIDTH ustawia szerokość kolumn),
LBS_MULTIPLESEL – przełącza selekcję przy każdym naciśnięciu,
LBS_NODATA – definiuje obiekt bez pokazywania danych, jeśli ilość elementów listy przekroczy 1000,
LBS_NOINTEGRALHEIGHT – rozmiar prostokąta obiektu nie jest zmieniany w zależności od ilości elementów,
LBS_NOREDRAW - ustawia, że lista nie będzie przerysowana po zmianach (ten styl można zmienić wysyłając do listy komunikat WM_SETREDRAW),
LBS_NOSEL – elementy listy można tylko widzieć, ale nie można wybierać,
LBS_NOTIFY – zezwolenie na wysyłanie komunikatów o naciśnięciu w obszarze obiektu,
LBS_OWNERDRAWFIXED – wskazuje, że właściciel listy rysuje elementy (właściciel okienka otrzymuje komunikat WM_MEASUREITEM przy kreacji obiektu oraz komunikat WM_DRAWITEM przy każdej wizualnej zmianie obiektu),

LBS_OWNERDRAWVARIABLE – wskazuje, że właściciel listy rysuje elementy oraz wysokość linijek może być zmienna (właściciel okienka otrzymuje komunikat WM_MEASUREITEM przy kreacji obiektu oraz komunikat WM_DRAWITEM przy każdej wizualnej zmianie obiektu),
LBS_SORT – sortowanie elementów w kolejności alfabetyczną,
LBS_STANDARD – włączone są opcje LBS_SORT i LBS_NOTIFY oraz obiekt ma otoczkę,
LBS_USETABSTOPS – zezwolenie na wprowadzenie znaków tabulacji,
LBS_WANTKEYBOARDINPUT – zezwolenie na otrzymanie przez właściciela komunikatu WM_VKEYTOITEM o naciśnięciu klawiszu.

Suwaki jako obiekty typu SCROLLBAR mogą mieć właściwości ustawiane przez następujące opcje stylu:
SBS_BOTTOMALIGN – dolna granica suwaka znajduje się na granicę prostokąta obiektu (zwykle stosuje się razem z opcją SBS_HORZ),
SBS_TOPALIGN – górna granica suwaka znajduje się na granicę prostokąta obiektu (zwykle stosuje się razem z opcją SBS_HORZ),
SBS_LEFTALIGN – lewa granica suwaka znajduje się na granicę prostokąta obiektu (zwykle stosuje się razem z opcją SBS_VERT),
SBS_RIGHTALIGN – prawa granica suwaka znajduje się na granicę prostokąta obiektu (zwykle stosuje się razem z opcją SBS_VERT),
SBS_HORZ - definiuje suwak poziomowy,
SBS_VERT - definiuje suwak pionowy,
SBS_SIZEBOX - definiuje rozmiar suwaka,
SBS_SIZEBOXBOTTOMRIGHTALIGN – prawy dolny róg suwaka znajduje się w rogu prostokąta obiektu (zwykle stosuje się razem z opcją SBS_SIZEBOX),
SBS_SIZEBOXTOPLEFTALIGN – lewy górny róg suwaka znajduje się w rogu prostokąta obiektu (zwykle stosuje się razem z opcją SBS_SIZEBOX),
SBS_SIZEGRIP - definiuje rozmiar suwaka oraz wypukłą otoczkę.

Obiekty typu STATIC można definiować z następującymi opcjami:
SS_BITMAP = 0Eh - definiuje bitmapę, przy czym nazwa okna jest nazwą bitmapy w pliku resursów,
SS_ICON = 3h - definiuje ikonę, przy czym nazwa okna jest nazwą ikony w pliku resursów,
SS_METAPICT - definiuje obraz w postaci metapliku, przy czym nazwa okna jest nazwą metapliku w pliku resursów,
SS_BLACKFRAME = 7h – definiuje prostokąt z otoczką z tym kolorem, co kolor otoczki okien Windows, który domyślnie jest czarnym,
SS_GRAYFRAME = 8h – definiuje prostokąt z otoczką z tym kolorem, co kolor ekranu Windows, który domyślnie jest szarym,
SS_WHITEFRAME = 9h – definiuje prostokąt z otoczką z tym kolorem, co kolor tła okien Windows, który domyślnie jest białym,
SS_BLACKRECT = 4h – definiuje prostokąt z tym kolorem, co kolor okien Windows, który domyślnie jest czarnym,
SS_GRAYRECT = 5h – definiuje prostokąt z tym kolorem, co kolor ekranu Windows, który domyślnie jest szarym,
SS_WHITERECT = 6h – definiuje prostokąt z tym kolorem, co kolor tła okien Windows, który domyślnie jest białym,
SS_CENTER = 1h – definiuje prostokąt, w środku którego jest umieszczany tekst – nazwa okna,
SS_CENTERIMAGE – centrowanie obiektu zdefiniowanego opcjami SS_BITMAP lub SS_ICON,
SS_RIGHTIMAGE – przywiązanie obiektu zdefiniowanego opcjami SS_BITMAP lub SS_ICON do prawego dolnego rogu prostokąta,
SS_LEFT = 0h, SS_RIGHT = 2h – definiuje prostokąt, od lewej lub prawej strony którego jest umieszczany tekst – nazwa okna,
SS_LEFTNOWORDWRAP – definiuje prostokąt, od lewej strony którego jest umieszczany tekst – nazwa okna, przy czym tekst nie jest zawrócony,
SS_NOPREFIX – ochrania od interpretacji znaku „&” jako wskaźnika na akcelerator (kombinacja przyśpieszająca),
SS_NOTIFY – zezwolenie na wysyłanie komunikatów STN_CLICKED i STN_DBLCLK,
SS_SIMPLE – definiuje prostokąt, od lewej strony którego jest umieszczany krótki (jedna linijka) tekst – nazwa okna,

Okna dialogowe mogą mieć następujące styli:

DS_3DLOOK – wygląd trzywymiarowy (system nie starszy niż Windows 95 i Windows NT 4.0 automatycznie produkuje taki wygląd),

DS_ABSALIGN – informuje, że współrzędne okna dialogowego są współrzędnymi ekranowymi (a nie współrzędnymi obszaru okna),

DS_CENTER – centruje okno dialogowe,

DS_CENTERMOUSE – centruje pozycję kursora myszy,

DS_CONTEXTHELP – dodaje przycisk-znak zapytania na pasku tytułu. Naciśnięcie znaku produkuje komunikat WM_HELP; procedura okna musi wywołać funkcję WinHelp przez wysyłanie komunikatu HELP_WM_HELP; przy kreacji okna system dodaje opcję WS_EX_CONTEXTHELP do stylu okna dialogowego,

DS_CONTROL – kreacja okna dialogowego jako potomka drugiego okna dialogowego, co daje możliwość przechodzić z okna do drugiego okna naciskając klawisz tabulacji,

DS_FIXEDSYS – wyznacza SYSTEM_FIXED_FONT zamiast SYSTEM_FONT,

DS_LOCALEDIT – powoduje rozmieszczenie struktur dla kontrolek okna typu pól edycyjnych w sekcji danych aplikacji (można zastosować tylko do 16-bitowych aplikacji),

DS_MODALFRAME – kreacja okna, które może mieć nagłówek i menu (opcje WS_CAPTION i WS_SYSMENU),

DS_NOFAILCREATE – kreacja okna, gdybyś nawet miał miejsce błąd,

DS_NOIDLEMSG – tłumi komunikat WM_ENTERIDLE, który powoduje umieszczenie okna na planie pierwszym,

DS_RECURSE – wyznacza styl dla okien dialogowych podobnych do kontrolek,

DS_SETFONT – informuje, że w strukturze DLGTEMPLATE znajdują się dwa dodatkowych składnika zawierających nazwę i rozmiar czcionki,

DS_SETFOREGROUND - przesuwa okno dialogowe, aby ono było widoczne,

DS_SYSMODAL - kreacja okna modalnego, tj. okna, bez zamknięcia którego nie można przejść do innego okna.

7.7.4. Programowanie okna z przyciskiem i oknem edycyjnym

Rozpatrzmy programowanie okna z obiektami graficznymi, na przykład, z przyciskiem i oknem edycyjnym.

Jeżeli znamy nazwę klasy standardowego obiektu graficznego, to można przystąpić do kreacji obiektu za pomocą funkcji `CreateWindowEx`. Instrukcje dotyczące kreacji obiektu należy umieścić w procedurze okna-rodzica we fragmencie, który jest wykonywany po otrzymaniu komunikatu WM_CREATE.

W niżej przytoczonym przykładzie ma miejsce kreacja wewnątrz głównego okna dwóch obiektów graficznych: przyciska i pola edycyjnego.

Przykład 7.6. Aplikacja graficzna z tworzeniem obiektów graficznych wewnątrz okna (program w języku asemblera MASM32)

```
.586P
.MODEL flat, STDCALL
;--- pliki ---------
include .\include\grafika.inc
;--- biblioteki ---------
includelib .\lib\user32.lib
includelib .\lib\kernel32.lib
includelib .\lib\gdi32.lib
;--- stałe ---
CSstyle EQU CS_HREDRAW+CS_VREDRAW+CS_GLOBALCLASS
BSstyle EQU BS_PUSHBUTTON+WS_VISIBLE+WS_CHILD+WS_TABSTOP
WNDstyle EQU WS_CLIPCHILDREN OR WS_OVERLAPPEDWINDOW OR \
             WS_HSCROLL OR WS_VSCROLL
EDTstyle EQU WS_VISIBLE+WS_CHILD+WS_TABSTOP+WS_BORDER
kolor EQU 000000FFh ; czerwony  ; kolory: G B R
;--- sekcja danych ------
_DATA        SEGMENT
 hwnd        DD  0
 hinst   DD  0
 hdc        DD   0
 hbutt   DD  0
 hedt    DD  0
```

```
 hbrush    DD    0
 holdbrush       DD    0
 ;---
 msg MSGSTRUCT <?>
 wndc WNDCLASS <?>
 ;---
 naglow  DB  "Autor Jan Masztalski.",0
 rozmN   DD  $ - naglow   ;ilość znaków w tablicy
 ALIGN   4   ; przesuniecie do adresu podzielnego na 4
 puste   DB  "_____________________"   ; spacje
 rozmP   DD  $ - puste    ;ilość znaków w tablicy
 ALIGN   4   ; przesuniecie do adresu podzielnego na 4
 tytul   DB "Aplikacja graficzna",0
 ALIGN 4
 cname   DB "MainClass", 0
 ALIGN 4
 MES1       DB "Lewy, myszy",0
 ALIGN 4
 tbutt   DB  "BUTTON", 0
 ALIGN 4
 tstart  DB  "Start", 0
 ALIGN 4
 tedt          DB "EDIT", 0
 ALIGN 4
 tnazwaedt   DB  " ", 0
 ALIGN 4
 ttekst  DB  "tekst", 0
 ALIGN 4
 nagl        DB  "Komunikat", 0
 ALIGN 4
 terr        DB  "Błąd!", 0
 ALIGN 4
 terr2   DB  "Błąd 2!", 0
 ALIGN 4
 bufor   DB  128 dup(' ') ; bufor ze spacjami
 rbuf        DD  128 ; rozmiar buforu
 znaczn     DD    0
 rt   RECT  <120,50,210,90>
 rr   DD    0
_DATA        ENDS
;--- sekcja kodu ---------
_TEXT        SEGMENT
;;;;;;;;;;;;;;;;;;;;;;;
WndProc PROC
;--- procedura okna ---
; DWORD PTR [EBP+14h] - parametr LPARAM komunikatu
; DWORD PTR [EBP+10h] - parametr WPARAM komunikatu
; DWORD PTR [EBP+0Ch] - ID = identyfikator komunikatu
; DWORD PTR [EBP+08h] - HWND = deskryptor okna
 ;-----------------------------------
 push    EBP ; standardowy prolog
 mov EBP, ESP ; standardowy prolog
;--- odkładanie na stos
 push    EBX
 push    ESI
 push    EDI
 cmp DWORD PTR [EBP+0Ch], WM_CREATE
 jne @F
 jmp wmCREATE
@@:
```

```
 cmp DWORD PTR [EBP+0Ch], WM_DESTROY
 jne @F
 jmp wmDESTROY
@@:
 cmp DWORD PTR [EBP+0CH], WM_COMMAND
 jne @F
 jmp wmCOMMAND
@@:
 cmp DWORD PTR [EBP+0CH], WM_LBUTTONDOWN
 jne @F
 jmp wmLBUTTON
@@:
 cmp DWORD PTR [EBP+0CH], WM_RBUTTONDOWN
 jne @F
 jmp wmRBUTTON
@@:
;--- komunikaty nieobsługiwane ---
 INVOKE DefWindowProcA, DWORD PTR [EBP+08h], \
     DWORD PTR [EBP+0Ch], DWORD PTR [EBP+10h], \
     DWORD PTR [EBP+14h]
 jmp konWNDPROC
wmCREATE:
;--- utworzenie klawisza ---
 INVOKE CreateWindowExA, 0, OFFSET tbutt, OFFSET tstart, \
  BSstyle, 10, 50, 100, 40, DWORD PTR [EBP+08h], 0, hinst, 0
 mov hbutt, EAX
;--- utworzenie okna edycyjnego ---
 INVOKE CreateWindowExA, 0, OFFSET tedt, OFFSET tnazwaedt, \ EDTstyle, 10,
100, 100, 40, DWORD PTR [EBP+08h], 0, hinst, 0
 mov hedt, EAX
 INVOKE SendMessageA, hedt,WM_SETTEXT,0,OFFSET ttekst
 INVOKE SetFocus,hedt
 ;---
 INVOKE CreateSolidBrush,kolor
 mov hbrush,EAX
;---
 mov EAX,0
 jmp konWNDPROC
wmDESTROY:
 INVOKE DeleteObject, hbrush
 INVOKE PostQuitMessage, 0    ; wysyłanie WM_QUIT
 mov EAX, 0
 jmp konWNDPROC
wmCOMMAND:
 mov EAX,hbutt
 cmp EAX,DWORD PTR [EBP+14h] ;czy LPARAM komunikatu \
                              WM_COMMAND = hbutt?
 je @F
 jmp et1
@@:
 cmp znaczn,1
 je @F
 mov znaczn,1
 INVOKE SelectObject,hDC,hbrush
 mov holdbrush,EAX
 INVOKE FillRect,hDC,OFFSET rt,hbrush
 mov EAX, 0
 jmp konWNDPROC
@@:
 mov znaczn,0
```

```
 INVOKE SelectObject,hDC,holdbrush
 mov hbrush,EAX
 INVOKE FillRect,hDC,OFFSET rt,holdbrush
 mov EAX, 0
 jmp konWNDPROC
et1:
 jmp konWNDPROC
wmRBUTTON:
 jmp wmDESTROY
wmLBUTTON:
 INVOKE SendMessageA,hedt,WM_GETTEXT,128,OFFSET bufor
 mov rr,EAX
 INVOKE TextOutA,hDC,110,120,OFFSET bufor,rr
 mov EAX,0
 jmp konWNDPROC
;--- zdejmowanie ze stosu
konWNDPROC:
 pop EDI
 pop ESI
 pop EBX
 mov ESP,EBP  ; standardowy epilog
 pop EBP ; standardowy epilog
 ret 16  ; zwolnienie komórek stosu
WndProc  ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;--- start programu ---
Start:
;--- deskryptor aplikacji ----
 INVOKE GetModuleHandleA, 0
 mov hinst, EAX
;--- wypełnienie struktury okna WNDCLASS
 mov EAX, hinst
 mov [wndc.clsHInstance], EAX
 mov [wndc.clsStyle], CSstyle
 mov [wndc.clsLpfnWndProc], OFFSET WndProc
 mov [wndc.clsCbClsExtra], 0
 mov [wndc.clsCbWndExtra], 0
 INVOKE LoadIconA, 0, IDI_APPLICATION  ; ikona
 mov [wndc.clsHIcon], EAX
 INVOKE LoadCursorA, 0,   IDC_ARROW ; kursor
 mov [wndc.clsHCursor], EAX
 INVOKE GetStockObject, WHITE_BRUSH    ; tło
 mov [wndc.clsHbrBackground], EAX
 mov [wndc.clsLpszMenuName], 0
 mov DWORD PTR [wndc.clsLpszClassName], OFFSET cname
;--- rejestracja okna ---
 INVOKE RegisterClassA, OFFSET wndc
 cmp EAX, 0
 jne @F
 jmp err0
@@:
;--- utworzenie okna głównego ---
 INVOKE CreateWindowExA, 0, OFFSET cname, OFFSET tytul, \
       WNDstyle, 50, 50, 600, 400, 0, 0, hinst, 0
 cmp EAX, 0
 jne @F
 jmp err2
@@:
 mov hwnd, EAX
 INVOKE ShowWindow, hwnd, SW_SHOWNORMAL
```

```
 INVOKE GetDC,hwnd
 mov hdc,EAX
 INVOKE lstrlenA,OFFSET naglow
 mov rozmN,EAX
 INVOKE TextOutA,hDC,10,20,OFFSET naglow,rozmN
 INVOKE UpdateWindow, hwnd
;--- pętla obsługi komunikatów
msgloop:
 INVOKE GetMessageA, OFFSET msg, 0, 0, 0
 cmp EAX, 0
 jne @F
 jmp etkon
@@:
 cmp EAX, -1
 jne @F
 jmp err0
@@:
 INVOKE TranslateMessage, OFFSET msg
 INVOKE DispatchMessageA, OFFSET msg
 jmp msgloop
;--- obsługa błędów ---------
err0:
;--- okno z komunikatem o błędzie----
 INVOKE MessageBoxA,0,OFFSET terr,OFFSET nagl,0
 jmp etkon
err2:
;--- okno z komunikatem o błędzie----
 INVOKE MessageBoxA,0,OFFSET terr2,OFFSET nagl,0
 jmp etkon
;--- zakończenie procesu ---------
etkon:
 INVOKE ExitProcess, [msg.msWPARAM]
_TEXT        ENDS
END Start
```

Aby sterować obiektem graficznym ze strony aplikacji należy wysyłać do obiektu komunikaty. W przykładzie 7.6 do pola edycyjnego typu EDIT jest wysyłany komunikat WM_SETTEXT:

```
 INVOKE SendMessageA, hedt,WM_SETTEXT,0,OFFSET ttekst
```

Za pomocą tego komunikatu w pole edycyjne jest wpisywany tekst.

7.7.5. Kontekst urządzenia

Rysowanie i wyprowadzenie tekstu w trybie graficznym jest niemożliwe bez użycia tzw. *kontekstu urządzenia* (ang. *device context*). Kontekst urządzenia jest obiektem programowym ze złożonej strukturą. Ten obiekt jest miejscem do przechowywania wyświetlanych danych oraz atrybutów sterujących procesem wyświetlenia. Deskryptor kontekstu urządzenia to 32-bitowy adres wspomnianego obiektu.

W stosunku do ekranu kontekst urządzenia nierzadko jest nazywany *kontekstem okna*. W operacjach drukowania też jest stosowany kontekst urządzenia, który w tym przypadku jest nazywany *kontekstem drukarki*.

Do operacji na kontekście okna i kontekście drukarki służą jednakowe funkcje (podprogramy). Na pewnie podobizna operacji wyświetlenia i drukowania oraz podobizna potrzebnych struktur programowych spowodowali stosowanie pojęcia „kontekst urządzenia”.

W programie przykładu 7.6 kontekst urządzenia jest wykorzystany przy wyprowadzeniu tekstu za pomocą funkcji TextOut oraz przy rysowaniu kolorowego prostokąta za pomocą funkcji FillRect.

Deskryptor kontekstu urządzenia zwraca funkcja GetDC, do której w przypadku ekranu należy przekazać deskryptor okna. Na końcu programu należy zwolnić kontekst urządzenia wywołując funkcję ReleaseDC.

Przy wyprowadzeniu tekstu oraz przy rysowaniu są potrzebne obiekty: czcionka, pióro, pędzel. Do kontekstu urządzenia są podłączone domyślne czcionka, pióro, pędzel. Przełączenie kontekstu urządzenia na nowy obiekt (czcionka, pióro, pędzel) wykonuje funkcja SelectObject, która zwraca deskryptor zamienianego obiektu. Na zakończenie programu należy za pomocą tej funkcji przywrócić domyślne czcionkę, pióro, pędzel.

W przykładzie 7.6 funkcja CreateSolidBrush produkuje nowy pędzel z deskryptorem hbrush. Później za pomocą funkcji SelectObject ma miejsce przełączenie pędzla, a w końcu programu – podłączenie poprzedniego pędzla.