# Podstawy Programowania PalmOS


O czym należy pamiętać:
- mały ekran (160x160)
- utrudnione wpisywanie tekstu
- synchronizacja z komputerem PC
- ograniczona wielkość pamięci
- szybkość działania

Program jest bazą danych składającą się z zasobów m.in.:
- kod programu
- elementy interfejsu uzytkownika
- łańcuchy tekstowe
- formatki
- ikony

# Elementy interfejsu użytkownika

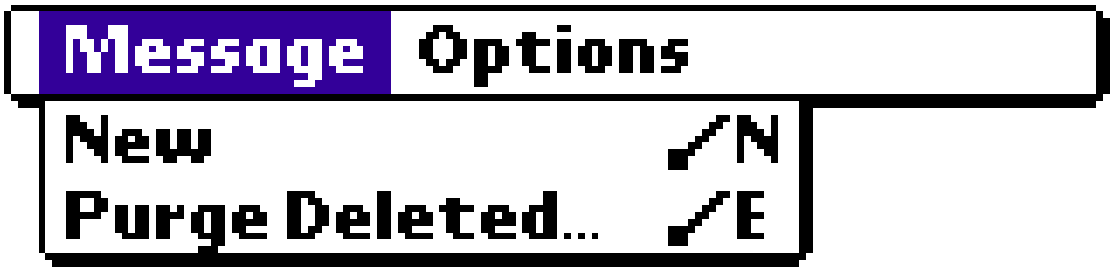
Alarmy:
- informacyjne
- pytanie
- ostrzeżenie
- zatrzymanie

# Elementy interfejsu użytkownika

Formatka

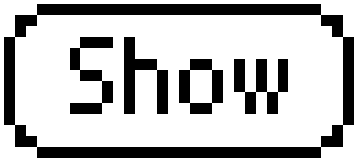

Pole wyboru

Menu



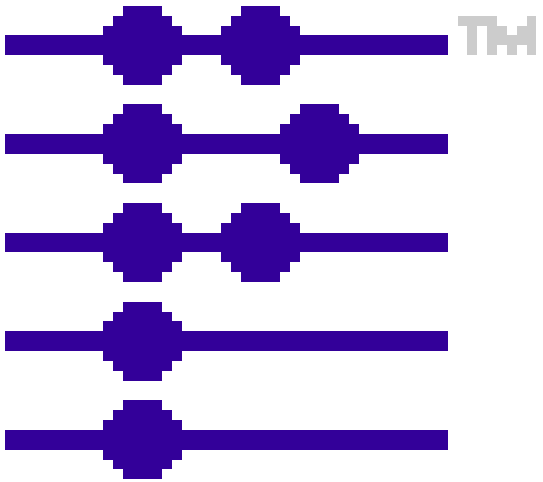

palm®

# Elementy interfejsu użytkownika

Przycisk 


Przełącznik 


Pole 


Mapa bitowa 

# Elementy interfejsu użytkownika

Gadżet

| S | M | T | W | T | F | S |
|---|---|---|---|---|---|---|
| | | | | | | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| 30 | 31 | | | | | |

Znacznik Shift 

Etykieta

Subj:

Body:

# Narzędzia programistyczne

- PalmOS Developer Suite
http://www.palmsource.com/developers

Oparty na środowisku programistycznym ECLIPSE – IBM
Programowanie w C/C++

- Emulator

Wykorzystuje oryginalną zawartość pamięci ROM komputerów Palm.

# Programowanie

Punktem wejściowym programu jest zawsze funkcja PilotMain.

```
UInt32 PilotMain(UInt16 cmd, MemPtr cmdPBP, UInt16 launchFlags)
{
    Err error = errNone;
    switch (cmd) {
        case sysAppLaunchCmdNormalLaunch:
            if ((error = AppStart()) == 0) {
                AppEventLoop();
                AppStop();
            }
            break;
        default:
            break;
    }
    return error;
}
```

# Programowanie

Funkcja AppStart służy do inicjalizacji głównej formatki aplikacji.

```
static Err AppStart(void)
{
    FrmGotoForm(MainForm);
    return errNone;
}
```

W tej funkcji można także załadować preferencje użytkownika, otworzyć bazę danych, zaalokować pamięć.

Funkcja AppStop może być pusta. Aplikacja powinna w niej m.in. zwolnić przydzieloną pamięć.

# Programowanie

Aplikacja pracuje w pętli wewnątrz funkcji AppEventLoop.

```
static void AppEventLoop(void)
{
    Err error;
    EventType event;
    do {
        EvtGetEvent(&event, evtWaitForever);

        if (SysHandleEvent(&event))
            continue;

        if (MenuHandleEvent(0, &event, &error))
            continue;

        if (AppHandleEvent(&event))
            continue;

        FrmDispatchEvent(&event);

    } while (event.eType != appStopEvent);
}
```

# Programowanie

Funkcja EvtGetEvent
- pobiera zdarzenia z kolejki zdarzeń,
- ważnym jej parametrem jest czas jaki ma czekać na zdarzenie jeśli kolejka jest pusta,
- evtWaitForever – usypia aplikację – oszczędzanie energii,
- evtNoWait – natychmiastowy powrót – komputer pracuje bez przerwy

Funkcja SysHandleEvent
- pierwsza w kolejności,
- obsługa zdarzeń systemowych.

Funkcja MenuHandleEvent
- przetwarza zdarzenia związane z menu.

# Programowanie

Funkcja AppHandeEvent
- standardowa konstrukcja,
- załadowanie formatek,
- przypisanie funkcji obsługi zdarzeń do formatek.

Funkcja FrmDispatchEvent
- reakcja na standardowe polecenia,
- wywołanie funkcji przypisanej do okienka.

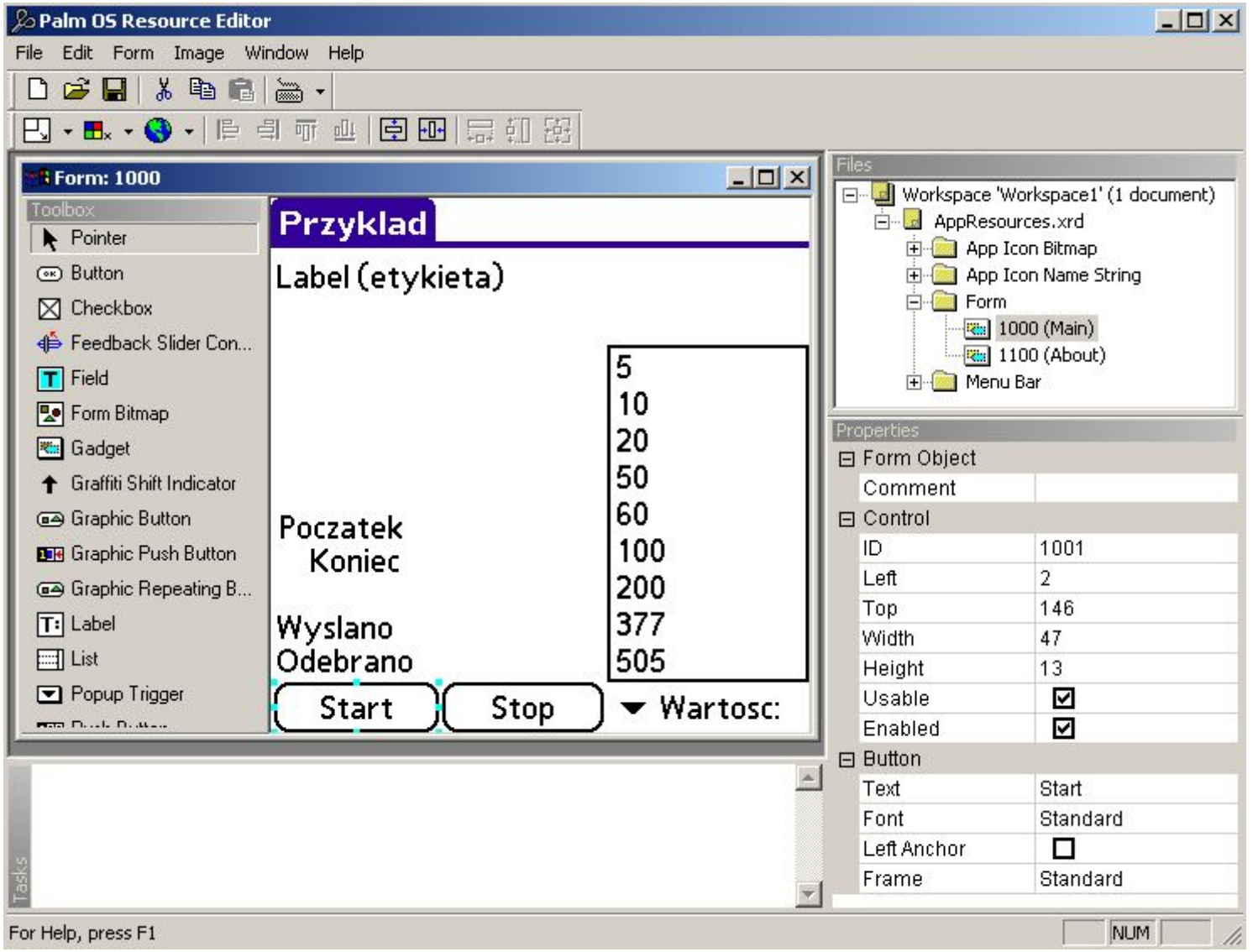
# AppHandleEvent

```
static Boolean AppHandleEvent(EventType* pEvent)
{
    UInt16       formId;
    FormType* pForm;
    Boolean      handled = false;

    if (pEvent->eType == frmLoadEvent) {
       formId = pEvent->data.frmLoad.formID;
       pForm = FrmInitForm(formId);
       FrmSetActiveForm(pForm);
       switch (formId) {
        case MainForm:
            FrmSetEventHandler(pForm, MainFormHandleEvent);
            break;
        default:
            break;
        }
       handled = true;
    }
    return handled;
}
```

# MainFormHandleEvent

```
static Boolean MainFormHandleEvent(EventType* pEvent)
{
    Boolean       handled = false;
    FormType* pForm;
    Err       err;

    switch (pEvent->eType) {
        case menuEvent:
            return MainFormDoCommand(pEvent->data.menu.itemID);

        case frmOpenEvent:
            pForm = FrmGetActiveForm();
            FrmDrawForm(pForm);
            handled = true;
            break;
         break;

        default:
        break;
    }

    return handled;
}
```

# Edytor zasobów




Przykład edycji parametrów przycisku. Przycisk otrzymał identyfikator ID.

Identyfikator wykorzystujemy w programie.

# Wykorzystanie zasobów

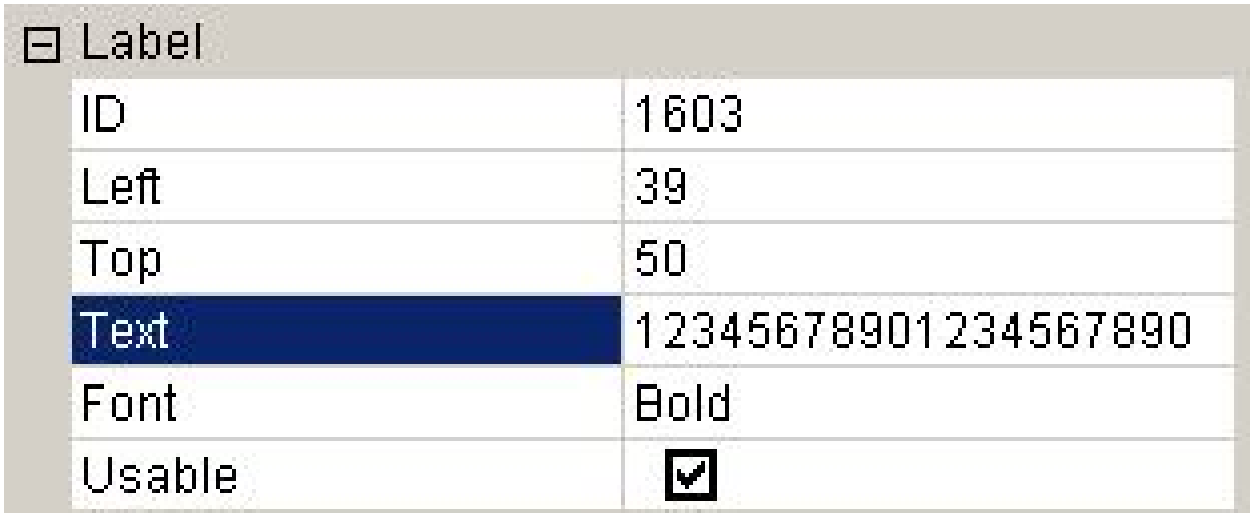
Wykorzystanie listy wyboru oraz przycisku w programie.

```
switch (pEvent->eType) {

        case popSelectEvent:
        if (pEvent->data.ctlEnter.controlID == 1004) {
            switch(pEvent->data.popSelect.selection) {
              case 0:
                step_size = 5;
                break;
              case 1:
                step_size = 10;
                break;
           ..................
              }
           }; //if
            break;

        case ctlSelectEvent:
        if (pEvent->data.ctlSelect.controlID == 1001) {
                start_counting();
           }; //if
            break;
        } //switch pEvent->eType
```

# Wykorzystanie zasobów

Wykorzystanie etykiety jako elementu wyjściowego.

```
static Boolean MainFormHandleEvent(EventType* pEvent)
{
Boolean     handled = false;
FormType*   pForm;
Char        lancuszek[]="                    ";        //20 znakow

....

case ctlSelectEvent:
if (pEvent->data.ctlSelect.controlID == 1000) {
             pForm = FrmGetActiveForm();
             StrCopy(lancuszek, "Wcisnales klawisz   ");
             FrmCopyLabel (pForm, 1603, lancuszek);
       };
       break;

....
```



A teraz Rodacy do pracy !!