Was bleibt noch übrig

Die Applikation ist nun fast fertig, zwei kleine Schönheitsfehler gilt es zu beseitigen. Bei beiden Fehlern durchkämmte ich sämtliche Dokumentationen (u.a. die kompletten MS Developers Library mit ca. 500MByte gepackter Information) um auf eine ordentlich Lösung zu stoßen. Vergeblich, für beide Probleme bedurfte es eines Tricks.

1. Problem:

Sobald unser Formular den Eingabefokus besitzt, ist ein Drucken nicht mehr möglich. Klickt man hingegen auf das Ausgabefenster so steht der Druck- und Seitenansichtsbefehl wieder zur Verfügung. Der Grund liegt in der sonst brauchbaren Kontextsensitivität des MFC Nachrichtenverteilers. Die Menü ID's für den Druck sind in *CMCLabelView* mit internen Bearbeitern verbunden. In *CMCFormView* fehlt diese Verbindung. Es nützt auch nichts, die Verbindung herzustellen, da die Formularansicht nicht weiß wie der Druck auszusehen hat. Die logische Folgerung wäre die Verbindung in den Kontext des MDI Fensters oder des Dokuments zu heben. Dieser Versuch schlägt ebenfalls fehl, da der Bearbeiter nur im Kontext einer Ansichtsklasse verwendet werden darf. Das heißt, wir müssen zunächst in der Formularklasse Bearbeiter einbauen. Ein direkter Aufruf der entsprechenden Funktionen in der *CScrollView* Klasse scheitert aufgrund einer *protected* Implementierung. Zur Lösung müssen wir zwei Funktionen in die *CMCLabelView* Klasse einbauen:

```
void CMCLabelView::Print(){    OnFilePrint();}
CMCLabelView::PrintPreview(){  OnFilePrintPreview();}
```

Für die Bearbeiter in *CMCFormView* verwenden wir folgende Zeilen:

```
void CMCFormView::OnFilePrint()
{
    CMCLabelDoc *pDoc = GetDocument();
    POSITION pos = pDoc->GetFirstViewPosition();
    while (pos != NULL)
    {
        CView* pView = pDoc->GetNextView(pos);
        if ( pView->IsKindOf( RUNTIME_CLASS( CMCLabelView))) {
            ((CMCLabelView *)pView)->Print();
            break;
        }
    }
}
```

Zusätzlich müssen wir noch folgende Zeile in den *MESSAGE_MAP* Abschnitt in "*MCFormview.cpp*" einbauen:

```
    ON_COMMAND(ID_FILE_PRINT_DIRECT, OnFilePrint)
```

Da für ID_FILE_PRINT_DIRECT kein Menüeintrag vorhanden ist, kann dieser Berarbeiter nicht über den *ClassWizard* angelegt werden. Für ID_FILE_PRINT und ID_FILE_PRINT_PREVIEW ist der Class Wizard ohne Probleme verwendbar. Die Nachricht ID_FILE_PRINT_DIRECT wird intern abgesandt wenn z.B. ein Druckauftrag per DDE erfolgt (Für das Drucken aus dem Windows Explorer heraus wird DDE verwendet).

Der Bearbeiter für *CMCFormView::OnFilePrintPreview* ist bis auf den Aufruf von *PrintPreview* identisch. Mit den beiden Befehlen *GetFirstViewPosition* und *GetNextView* durchsuchen wir die Liste der aktiven Views nach der gewünschten Klasse und rufen dort unsere Dummyfunktionen auf. Der Nachteil dieser Lösung ist, daß die gewünschte Ansichtklasse geöffnet sein muss (bei unser immer der Fall). Wünschenswert wäre es, wenn es eine Möglichkeit gäbe einen Standardansicht für diesen Fall zu definieren.

2. Problem:

In jedem neuen MDI Fenster erscheint unser Formular verstümmelt. Man kann zwar beim Erzeugen des Formulars (in *CChildFrame::OnCreateClient*) eine Größe angeben, doch gibt es meines Wissens keine Möglichkeit die richtige Größe zu erfahren. Da *CFormView* von *CScrollView* abgeleitet ist scheint es auch, daß es eine passende Funktion gibt: *ResizeParentToFit*. Die Dokumentation schlägt auch die Verwendung dieser Funktion vor, doch trügt der Namen. Die Funktion verändert nicht das Elternfenster sondern das nächst höher liegende *FrameWindow*. In unserem Fall also *CChildFrame*, die Splitterposition bleibt jedoch unverändert. Der Trick wie man die richtige Größe des Formulars ermittelt führt über die Scroll-Leiste. Die Funktion *GetDeviceScrollSizes* liefert uns die maximale Scrollposition und diese entspricht auch der benötigten Fenstergröße. Wir fügen also der Funktion *CMCFormView::OnInitialUpdate* die Zeile:

```
ResizeParentToFit( FALSE);
```

hinzu und der Funktion *CChildFrame::OnCreateClient* die folgende Zeilen:

```
CFormView *pwnd;
CSize siz1,siz2,siz3;
int idummy;
pwnd = (CFormView *)m_wndSplitter.GetPane(0,0);
pwnd->GetDeviceScrollSizes( idummy, siz1, siz2, siz3);
m_wndSplitter.SetColumnInfo( 0, siz1.cx, 0);
m_wndSplitter.SetRowInfo( 0, siz1.cy, 0);
m_wndSplitter.RecalcLayout();
```

Auch diese Lösung ist nicht ganz Wasserdicht. Die Funktion *ResizeParentToFit* kann dazu führen, daß das MDI Fenster größer als das Hauptfenster ist. Dies wirkt sich aber erst bei Bildschirmauflösungen von 800x600 und darunter aus. Das bei diesen Auflösungen der Bildschirm zu klein wird dürfte sowieso hinlänglich bekannt sein.

Möglicherweise gibt es für die zwei Probleme auch einfachere Lösungen und ich sehe vor lauter Bäumen den Wald nicht mehr. Für Anregungen und Hinweise stehe ich gerne per eMail zur Verfügung.

Trotzalledem sei zu sagen, daß die Vorteile der MFC gegenüber der herkömmlichen C Programmierung überwiegen. Man erspart sich wesentliches an Tipparbeit und das Programm bleibt auch übersichtlicher. Daß es natürlich auch Grenzen gibt an die man früher oder später stößt scheint dabei nur logisch.

Literatur

[1] Marcellus Buchheit: Inside MFC Teil 2, System Journal Mai/Juni 1995, S.90

[2] Marcellus Buchheit: Inside MFC Teil 3, System Journal Juli/August 1995, S.116

[3] Charles Petzold: Programing Windows □


```
micado Smalltalker
==================
   MessageBox message: 'Hello, World'


New Manager
===================
   10 PRINT "HELLO WORLD"
   20 END


Middle Manager
===================
   mail -s "Hello, world." bob@b12
   Bob, could you please write
   me a program that prints
   "Hello, world."?
   I need it by tomorrow.
   ^D
```

```
Senior Manager
==================

   % zmail jim
   I need a "Hello, world."
   program by this afternoon.


Chief Executive
===================

   % letter
   letter: Command not found.
   % mail
   To: ^X ^F ^C
   % help mail
   help: Command not found.
   % damn!
   !: Event unrecognized
   % logout
```

```
Assembler Freak
  ===============

   .MODEL SMALL
   .CODE
   org 100h

   mov ah,9
   mov dx,offset hellostr
   int 21h
   mov ax,4C00h
   int 21h

   hellostr db 'Hello World$'

   tasm /mx hello.asm
   tlink hello /t
```