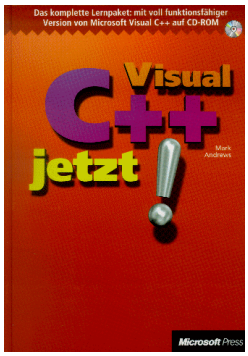
Buchbesprechung: „Visual C++ jetzt!“

Autor: Mark Andrews; Titel: Visual C++ jetzt!; Verlag: Microsoft Press; Untertitel: Das komplette Lernpaket: mit voll funktionsfähiger Version von Microsoft Visual C++; Seitenzahl: 429; Preis: ÖS 577,-; ISBN-Nummer: 3-86063-020-2; mit CD:

Grete Peschek DSK540//scramble



- Sie wollen Windows-Applikationen programmieren?
- Sie können in der Programmiersprache C oder besser noch in C++ programmieren?
- Sie wollen sich nicht noch zusätzlich das neueste Programmpaket von Visual C++ besorgen?

Wenn obige Punkte zutreffen, sind Sie mit den Buch „Visual C++ jetzt!“ sicherlich gut beraten. Ist der Einstieg einmal geschafft, können Sie dann noch immer auf die neueste Compilerversion umsteigen.

Das Buch ist in insgesamt zehn Kapiteln unterteilt. Die ersten vier Kapiteln stellen eine gelungene Kombination von Theorie und Beispielen über C++, Objekte und Methoden und Windows Programmierung dar. Viele Begriffe der Objektorientierten Programmierung werden präzise und einfach erklärt. Anschließend werden die Werkzeuge von Visual C++ und die MFC-Bibliothek, die das generische C++ und das Windows-API erweitert, anhand konkreter Beispiele eingeführt. (Siehe auch Beispielprogramm) Die letzten vier Kapiteln behandeln verschiedene Problemkreise, wie Mausereignisse für die Interaktion mit dem Benutzer, Dialogfelder, Datenverwaltung und Windows-Grafik und Animation. Für jede Anwendungsgruppe wird zunächst der theoretische Hintergrund erklärt und anschließend ein komplettes Beispiel entwickelt

Die Begleit CD-ROM enthält

- den Microsoft Visual C++Compiler Version 1.0
- den Visual C++ Linker
- die vollständige Entwicklungsumgebung von Visual C++
  - Visual C++ Editor
  - Quellcode-Browser
  - Quellcode- und Assemblersprachen-Debugger
  - Online Hilfedateien
  - eine Sammlung von Werkzeugen für das Erstellung und die Verwaltung von Ressourcen und C++-Klassen (Wizards)
- Sammlung von Anmerkungen mit Informationen über Visual C++ und den Visual C++-Compiler Version 1.0

Obige Compilerversion verwendet die MFC 2.0. Compiler und MFC Versionen sind zwar nur für 16-Bit Anwendungen ausgelegt und bieten nicht die Unterstützung wie etwa Visual C++ 4.0 mit der MFC 4.0, sind dafür aber fast gratis und reichen sicherlich für eine Einblick in die Programmierung von Windows-Applikationen aus.

Da der Umgang mit der MFC und der Wizards relativ einfach ist, kann in kürzester Zeit ein lauffähiges Programm erstellt werden.

Folgendes Beispielprogramm soll zeigen, wie unter Ausnützung der Programmierumgebung von Visual C++ und wenigen selbst einzufügenden Befehlen eine einfache Windowsapplikation entstehen kann.

Das in Kapitel 5 vorgestellte Programm SCRAMBLE stellt eine MDI-Anwendung dar, die eine Bitmap-Ressource darstellt. Man kann eine beliebige Anzahl von untergeordneten Fenstern öffnen, die aber immer das gleich Bitmap enthalten. Für den Einbau einer separaten Kontrolle der untergeordneten Fenster ist allerdings das Studium eines weiteren Kapitels notwendig.

Das Buch beschreibt die genaue Vorgangsweise, um ein Programmgerüst für die Erstellung einer grafisch orientierten Anwendung zu erhalten.

Nachdem nahezu automatisch alle notwendigen Klasse mittels AppWizard erzeugt werden, müssen nur noch relativ wenige Programmzeilen für die fertige Applikation hinzugefügt werden. Im untenstehenden Programmlisting sind die hinzugefügten Anweisungen entsprechend markiert.

SCRAMDOC.H

```
// scramdoc.h : interface of the CScrambleDoc class

class CScrambleDoc : public CDocument
{
private:
    CBitmap* m_pArches;
    CBitmap* m_pBackground;
public:
    CBitmap* GetBackground() { return m_pBackground; }
    void LoadBackground(CBitmap*);
    void UnloadBackground();

protected: // create from serialization only
    CScrambleDoc();
    DECLARE_DYNCREATE(CScrambleDoc)

// attributes
public:

// operations
public:

// implementation
public:
    virtual ~CScrambleDoc();
    virtual void Serialize(CArchive& ar);    // overridden for
                    // document i/o
#ifdef _DEBUG
    virtual    void AssertValid() const;
    virtual    void Dump(CDumpContext& dc) const;
#endif
protected:
    virtual    BOOL    OnNewDocument();

// generated message map functions
protected:
    //{{AFX_MSG(CScrambleDoc)
    afx_msg void OnBackgroundArches();
    afx_msg void OnBackgroundClear();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```

SCRAMDOC.CPP

```
// scramdoc.cpp : implementation of the CScrambleDoc class
//

#include "stdafx.h"
#include "scramble.h"

#include "scramdoc.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

// CScrambleDoc

IMPLEMENT_DYNCREATE(CScrambleDoc, CDocument)

BEGIN_MESSAGE_MAP(CScrambleDoc, CDocument)
    //{{AFX_MSG_MAP(CScrambleDoc)
    ON_COMMAND(ID_BACKGROUND_ARCHES, OnBackgroundArches)
    ON_COMMAND(ID_BACKGROUND_CLEAR, OnBackgroundClear)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

// CScrambleDoc construction/destruction

CScrambleDoc::CScrambleDoc()
{
    m_pArches = new CBitmap;
    if (m_pArches)
    {
        m_pArches->LoadBitmap(IDB_BITMAP1);
    }
    m_pBackground=m_pArches;
}

CScrambleDoc::~CScrambleDoc()
{
    if (m_pArches)
    {
```

```
        delete m_pArches;
        m_pArches=NULL;
    }
}

BOOL CScrambleDoc::OnNewDocument()
{
    if (!CDocument::OnNewDocument())
        return FALSE;
    // TODO: add reinitialization code here
    // (SDI documents will reuse this document)
    return TRUE;
}

// CScrambleDoc serialization

void CScrambleDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
        // TODO: add storing code here
    }
    else
    {
        // TODO: add loading code here
    }
}


// CScrambleDoc diagnostics

#ifdef _DEBUG
void CScrambleDoc::AssertValid() const
{
    CDocument::AssertValid();
}

void CScrambleDoc::Dump(CDumpContext& dc) const
{
    CDocument::Dump(dc);
}

#endif //_DEBUG

// CScrambleDoc commands
void CScrambleDoc::LoadBackground(CBitmap* pBackground)
{
    m_pBackground=pBackground;
}

void CScrambleDoc::UnloadBackground()
{
    m_pBackground=NULL;
}

void CScrambleDoc::OnBackgroundArches()
{
    LoadBackground(m_pArches);
    UpdateAllViews(NULL);
}

void CScrambleDoc::OnBackgroundClear()
{
    UnloadBackground();
    UpdateAllViews(NULL);
}
```

SCRAMVW.CPP
// scramvw.cpp : implementation of the CScrambleView class

```
//

#include "stdafx.h"
#include "scramble.h"

#include "scramdoc.h"
#include "scramvw.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

// CScrambleView

IMPLEMENT_DYNCREATE(CScrambleView, CView)

BEGIN_MESSAGE_MAP(CScrambleView, CView)
    //{{AFX_MSG_MAP(CScrambleView)
    // NOTE - the ClassWizard will add and remove mapping macros here.
    // DO NOT EDIT what you see in these blocks of generated code !
     //}}AFX_MSG_MAP
     // Standard printing commands
     ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
     ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
END_MESSAGE_MAP()

// CScrambleView construction/destruction

CScrambleView::CScrambleView()
{
    // TODO: add construction code here
}

CScrambleView::~CScrambleView()
{
}

// CScrambleView drawing

void CScrambleView::OnDraw(CDC* pDC)
{
    CScrambleDoc* pDoc = GetDocument();
    CBitmap* pBitmap;
    BITMAP Bitmap;
    CDC dc;

    pBitmap = pDoc->GetBackground();
    if (pBitmap)
    {
        dc.CreateCompatibleDC(pDC);
        CBitmap* pOldBitmap = dc.SelectObject(pBitmap);

        pBitmap->GetObject(sizeof(Bitmap), &Bitmap);
        pDC->BitBlt(0,0,    Bitmap.bmWidth,
                Bitmap.bmHeight,
                &dc,
                0,0,
                SRCCOPY);
        dc.SelectObject(pOldBitmap);
    }
}

// CScrambleView printing

BOOL CScrambleView::OnPreparePrinting(CPrintInfo* pInfo)
{
    // default preparation
    return DoPreparePrinting(pInfo);
}

void CScrambleView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add extra initialization before printing
}

void CScrambleView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add cleanup after printing
}

// CScrambleView diagnostics

#ifdef _DEBUG
void CScrambleView::AssertValid() const
{
    CView::AssertValid();
}

void CScrambleView::Dump(CDumpContext& dc) const
{
    CView::Dump(dc);
}

CScrambleDoc* CScrambleView::GetDocument() // nondebug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CScrambleDoc)));
    return (CScrambleDoc*) m_pDocument;
}

#endif
//_DEBUG

// CScrambleView message handlers
```

Bei Ausführung obigen Programms kann folgendes am Bildschirm erzeugt werden: