Visual C++ 4.0

oder ein weiterer Schritt zur eierlegenden Wollmilchsau

Heinrich Pommer

Das Setup

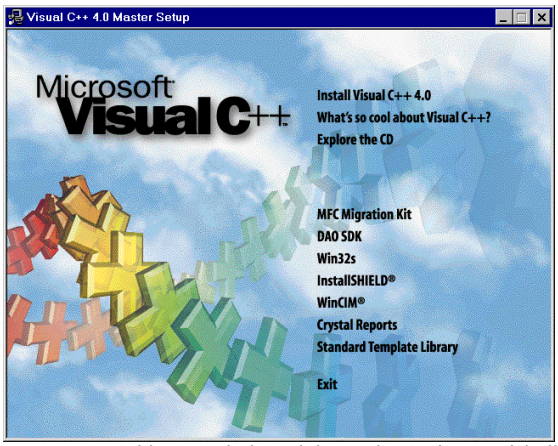


Abbildung 1: Setup Schirm. Nach dem Einlegen der CD lassen sich die Komponenten bequem installieren.

Nach dem Einlegen der CD erscheint, dank des in Windows 95 integrierten CD-Autoplay, ein Auswahlschirm, der es ermöglicht, Visual C++ und die Zusatzprodukte zu installieren. Zum Lieferumfang gehören auch Crystal Reports 4.5, ein Reportgenerator, und Installshield, welches endlich eine brauchbare Möglichkeit zur Erzeugung eigener Installationsprogramme bietet. Die Installation von Visual C++ ist problemlos, solange man über eine genügend große Festplatte verfügt. So belegt eine typische Installation ca. 90 MByte, eine komplette 240 MByte (50 MByte Beispielprogrammme und 90 MByte Online Hilfe). Unter Windows NT 3.51 erfolgt die Installation genauso problemlos. Als weitere Hardwareanforderung sollte noch erwähnt werden, daß ein angenehmes Arbeiten erst ab 20 MByte Ram möglich ist.

Developer Studio

Nach dem Start des Developer Studios merkt man auch, daß der eigene Bildschirm zu klein geraten ist. So würde ich einen 17" Monitor mit 1024*768 dringend als Minimum empfehlen. Auf der linken Seite macht sich das Project Workspace Fenster breit. Es enthält zunächst den Index für Books Online, spielt aber während der Entwicklung eine Schlüsselrolle. Am unteren Rand befindet sich ein allgemeines Ausgabefenster z.B. für die Compilermeldungen während eines Build-Laufs.

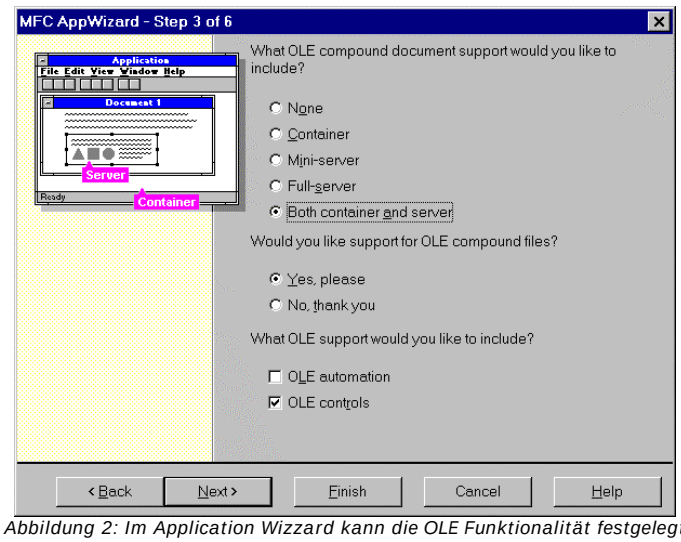


Abbildung 2: Im Application Wizzard kann die OLE Funktionalität festgelegt werden.

AppWizzard

Um ein Projekt zu beginnen, führt kein Weg am AppWizard vorbei, bietet er doch die Möglichkeit, für jede Aufgabenstellung einen Grundstein zu legen. Sei es mit einer simplen Make Datei für eine statische Bibliothek oder allen nötigen Dateien zur Erstellung eines eigenen AppWizards. Die stärkste Seite zeigt er jedoch bei der Generierung einer MFC (Microsoft Foundation Classes) Applikation. Auf sechs Dialogseiten werden alle zukünftigen Eigenschaften des Programms ausgewählt, beginnend beim Datenbanksupport weiter zum OLE Support bis zu den Kleinigkeiten wie kontextsensitive Hilfe und MAPI Support. Ein Großteil der MFC Funktionalität spiegelt sich in diesen Einstellungen wieder, doch dazu später. Es läßt sich auch die Sprache für die Applikation wählen, neben Englisch werden auch die Sprachen Deutsch, Spanisch, Französisch und Italienisch unterstützt. Die Unterstützung bezieht sich auf alle Menüs, Dialogfelder und Stringresourcen.

Der Hexenmeister erzeugt mehrere Dateien welche bereits ein compilierbares und lauffähiges Programm darstellen. Das Programm besitzt eine Statuszeile, eine Werkzeugleiste, Dialogboxen für Dateiöffnen und -schließen, Seitenvorschau und Druckerunterstützung. Alles Dinge, deren Implementierung von Hand mühsam und zeitaufwendig ist.

Project Workspace

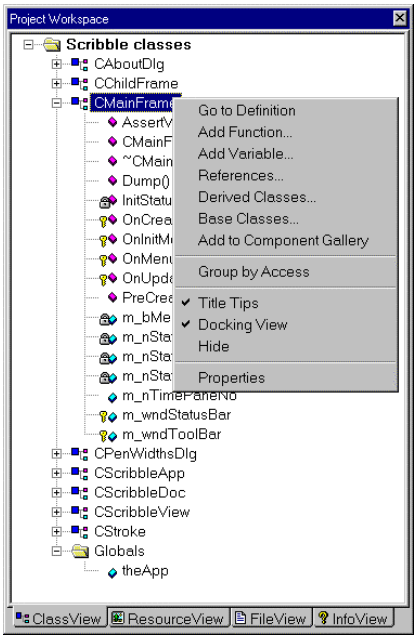


Abbildung 3: Im Project Workspace Window können neue Funktionen leicht hinzugefügt werden.

Das Project Workspace-Fenster zeigt nun, wozu es fähig ist. Es enthält eine Baumdarstellung der Klassen, Memberfunktionen und -variablen. Durch einen Klick mit der rechten Maustaste kann man zur entsprechenden Stelle im Quellcode springen oder neue Variablen bzw. Memberfunktionen anlegen. Beim Anlegen von Mitgliedsfunktionen wird der Funktionsrumpf und die entsprechende Deklaration in der Header-Datei kreiert. Bei der Entwicklung stehen auch noch der Wizard Bar und der Class Wizard zur Verfügung (Microsoft beweist wieder einmal seinen Hang zu Hexenkünstlern).

Class Wizzard

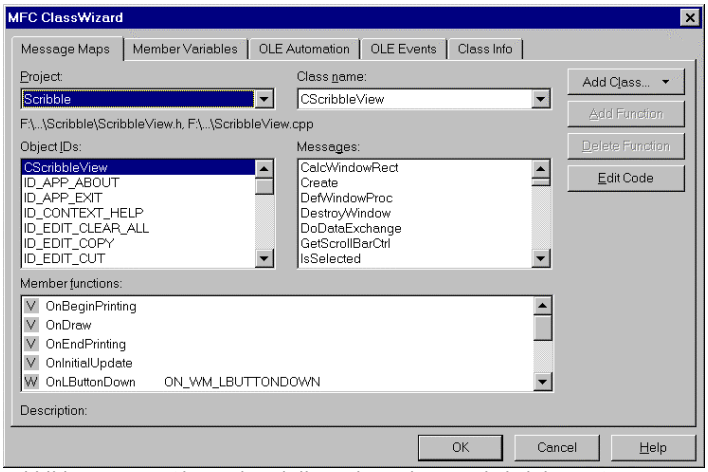


Abbildung 4: Der Class Wizard dient als Regiezentrale bei der MFC Programmierung.

Der Wizard Bar befindet sich am oberen Rand des Quellcodeeditors und erlaubt eine schnelle Navigation und Erstellung von virtuellen Mitgliedsfunktionen. Er kennt alle Funktionen der MFC-Basisklassen, welche überschreibbar sind. Mit dem Class Wizard können alle Feinheiten der Klassen verwaltet werden: die Verbindung der Mitgliedsvariablen mit entsprechenden Elementen des User Interface genauso wie die Verwaltung von *Events* und *Automation* der OLE Klassen.

Das Project Workspace-Fenster stellt außer der Class View auch noch ein File View und ein Resource View bereit. Im File View werden fast alle zum Projekt gehörenden Dateien dargestellt, auch Text- und Hilfedateien. Es werden nur jene Dateien versteckt gehalten, welche nicht zur direkten Bearbeitung gedacht sind. Zum Beispiel `resource.h` welche alle ID-Definitionen der Resourcen enthält und von der Entwicklungsumgebung verwaltet wird. Die Resource View-Ansicht regelt den Zugriff auf alle Resourcen des Projekts. Integriert sind auch alle Editoren für die Standardelemente. Nützlich erweist sich der Toolbar Editor.

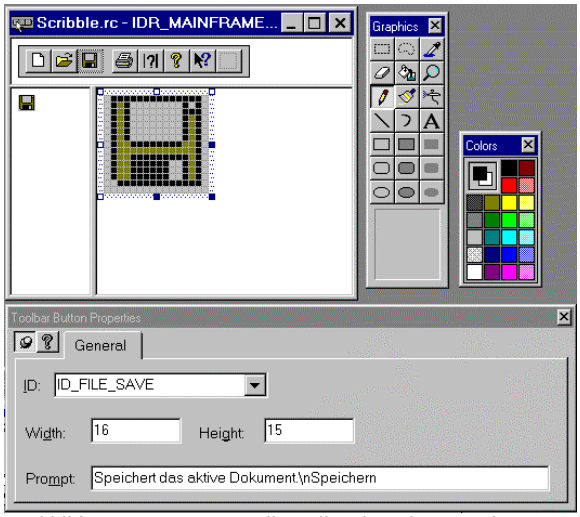


Abbildung 5: Der neue Toolbareditor ist sehr gut gelungen.

Abgesehen von der Schaltfläche kann auch gleichzeitig der Statuszeilentext und *ToolTip* Text verändert werden. Diese Texte werden automatisch als Stringresource abgespeichert. Man braucht sich auch überhaupt nicht um die Zuordnung der ID's zu kümmern, das erledigt die Entwicklungsumgebung ohne Probleme. Der Dialogboxeditor kommt nun auch mit OCX Controls zurecht. Leider trüben auch Fehler das Bild. So ist es zum Beispiel nicht möglich, einen farbigen Mauscursor zu erstellen. Erst nachdem ein Farbcursor importiert wurde, kann dieser auch verändert werden. Die Unterstützung für animierte Mauszeiger fehlt leider völlig. Spartanisch wirkt auch der Versionsresourceneditor, die recht komplizierte Materie dieser Einstellungen wird nicht sehr vereinfacht. Positiv ist wieder, daß sich Resourcen in verschiedenen Sprachen verwalten lassen.

Component Gallery

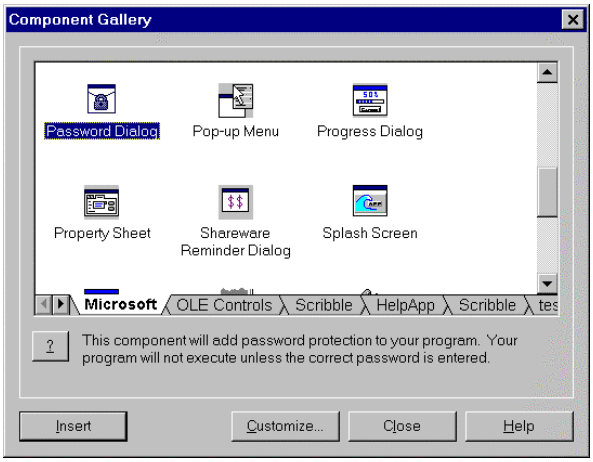


Abbildung 6: Die Component Gallery enthält häufig benötigte Programmfeatures.

Völlig neue Wege beschreitet Microsoft mit der Component Gallery. Dieser Ansatz zur Verwaltung wiederverwendbaren Codes zeigt, wie ein Teil der Programmierung in Zukunft aussehen wird. So implementierte Microsoft eine Reihe von Standardeigenschaften moderner Programme in die Gallery. Es stehen unter anderem ein Splash Screen (Informationsfenster während des Programmstarts), ein Shareware Reminder Dialog und ein Tip of the Day Dialog zur Verfügung. Um die aktuelle Zeit in der Statuszeile anzuzeigen, sind nur mehr sechs Maus-clicks notwendig. Zur nachträglichen Anpassung der Komponenten ist jedoch Handarbeit gefragt, welche sich auf Grund der dürftigen Dokumentation der einzelnen Komponenten als recht schwierig erweisen kann. Auch gibt es keine Dokumentation zur Erstellung eigener intelligenter Teile. Es werden zwar alle selbst erstellten Klassen automatisch in die Component Gallery eingefügt, doch können diese keinen Sourcecode verändern, sondern nur komplette Klassendefinitionen hinzufügen. Hat man einige Projekte erstellt, so wird die Gallery schnell unübersichtlich, so daß eine nachträgliche Verwaltungsarbeit notwendig wird. OCX Komponenten müssen ebenfalls erst über die Gallery hinzugefügt werden, um im Resourceeditor zur Verfügung zu stehen. Die Nützlichkeit der Gallery wird sich in der Praxis zeigen.

Fortran Powerstation, Visual Source Safe, Visual Test

Drei weitere, separat zu erwerbende Produkte können in das Developerstudio integriert werden: Fortran Powerstation (Fortran Entwicklungswerkzeuge), Visual Source Safe (Versionsverwaltung für Teamentwicklungen), Visual Test (automatisierbare Testtools) und Developer Library (Entwickler Datenbank). Die Einbindung der Fortran Powerstation ist sicherlich nützlich, obwohl ich niemanden kenne, der gemischt Fortran und C/C++ programmiert, die Integration eines kompletten Assembler Systems wäre mir lieber gewesen. Der Zugang zu Visual Source Safe und Visual Test war längst überfällig. Da die Developer Library auch über Dokumentation von Visual C++ 4.0 verfügt, erleichtert deren Integration die Suche nach Informationen erheblich.

Standard Template Library

Auch der Compiler mußte einige Neuerungen über sich ergehen lassen. Als letzter der bekannten Compilerhersteller akzeptierte Microsoft mit der Version 2.0 die Existenz von Templates und Exceptions im C++ Standard. Dafür werden nun die Standard Template Library (STL) von HP mitgeliefert. In der STL findet man einige Vorlagen für häufig benötigte Problemstellungen, z.B. verschiedene Sortieralgorithmen. Die Dokumentation der STL beschränkt sich auf ein Word Dokument. Ein Eintrag im Hilfe System beschreibt, welcher Änderungen an der STL durchzuführen sind, um sie problemlos mit der MFC verwenden zu können. Neu ist die Unterstützung für Namespaces und Runtime Type Information (RTTI). Ein Namespace definiert einen zusätzlichen Namen, um Konflikte bei Verwendung verschiedener Bibliotheken zu vermeiden. Mit der RTTI ist es möglich, den Typ von Objekten zur Laufzeit zu bestimmen.

Optionen

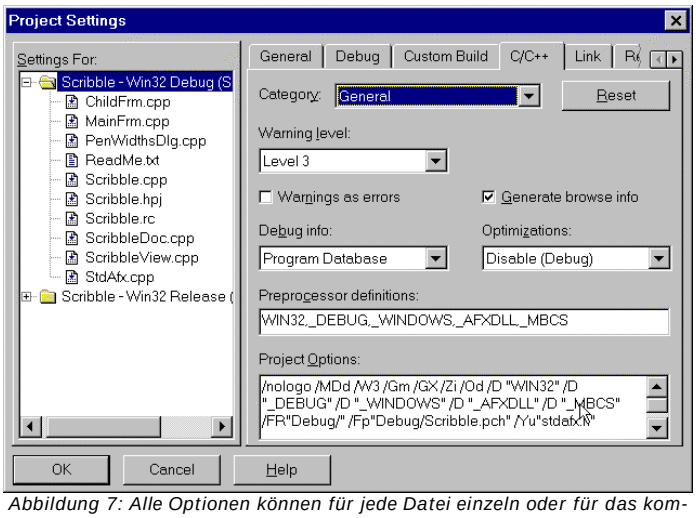


Abbildung 7: Alle Optionen können für jede Datei einzeln oder für das komplette Projekt verändert werden (Build Settings).

Die Turnaround Zeiten werden durch incremental Compiling und incremental Linking erheblich reduziert. Bei diesen Einstellungen werden nur jene Funktionen einer Quellcodedatei übersetzt und gelinkt, welche auch wirklich verändert wurden. Die Compilereinstellungen können für ein komplettes Projekt oder für jede Datei extra verändert werden. Die Entwicklungsumgebung erlaubt es auch, Subprojekte zu definieren. So kann ein großes Projekt mit seinen DLL's in einem Workspace verwaltet werden.

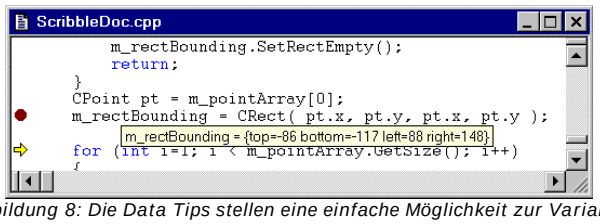


Abbildung 8: Die Data Tips stellen eine einfache Möglichkeit zur Variableninspektion dar.

Debugging

Die Möglichkeiten der Fehlersuche wurden stark überarbeitet. Tritt in einem laufenden Programm ein Zugriffsfehler auf, so kann der Debugger nachträglich geladen werden und ermöglicht danach Zugriff auf alle Variablen und den CallStack. Durch dieses *Just in Time* Debugging genannte Feature ist es nicht mehr nötig, ein Programm im Debugger zu starten und danach die einzelnen Funktionen zu tracen. Gut gelungen sind auch die DataTips. Ähnlich wie bei den Tooltips für Werkzeugleisten braucht der Mauszeiger nur über einer Variablen im Quelltext ruhen, und es erscheint ein kleines Fenster mit dem Inhalt. Auch ganze Ausdrücke können so dargestellt werden, indem sie einfach mit der Maus selektiert werden. Zusätzlich stehen ein Variable Window und ein Watch Window zur Verfügung.

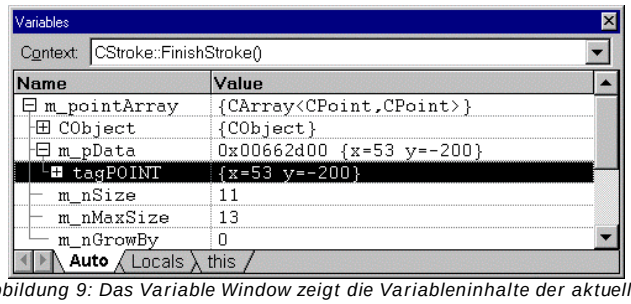


Abbildung 9: Das Variable Window zeigt die Variableninhalte der aktuellen und vorhergehenden Codezeile.

Das dreiteiligen Variable Window bietet Zugriff auf die Variablen der derzeitigen und vorhergegangen Codezeile, den lokalen Variablen und dem "this" Pointer. Im Watch Window können beliebige Ausdrücke durch Drag&Drop aus dem Quelltext, oder durch Eingabe eines eigenen Ausdrucks dargestellt werden. Eine Änderung der Variableninhalte für non-const Variablen ist dem Watch Window vorbehalten. Wem das Watch Window zu klein wird, dem wird durch vier Tabs, für vier individuelle Watch Gruppen, geholfen. Um Platz in den Ausgabefenstern zu sparen, werden die wichtigsten Elemente bekannter MFC Objekte in der ersten Zeile dargestellt. Für ein CString Objekt sieht die Darstellung z.B. folgendermaßen aus: *CString myString = { "Mein Text" }*. Das lästige Expandieren größere Objekte wird dadurch oftmals überflüssig. Ein *AutoDowncast* genanntes Feature erledigt automatisch einen Downcast auf die richtige Basisklasse eines Zeigers. Unabhängig von der Art des Zeigers steht der Zugriff auf die richtige Objektstruktur offen.

Foundation Classes

Nun zum Kern der zukünftigen Windowsentwicklung, den Microsoft Foundation Classes. War der Übersichtsplan der ersten Versionen noch ein locker bedrucktes A4 Blatt, so benötigt man bei der aktuellen 4.0 Version bereits eine Lupe, um die 8 Punkt Schrift entziffern zu können. Grundaufgabe der MFC ist natürlich die Kapselung eines Großteils des komplexen Windows API. Die eingeführte Zwischenschicht ist in den meisten Fällen sehr dünn, sodaß der Overhead gering bleibt. An anderer Stelle erweitert die MFC das Standard API erheblich. Beginnend bei kleineren Erweiterungen, wie einer Height Funktion für CRect Objekte bis zur kompletten Implementierung des komplexen OLE Themas, wird vieles der sonst notwendigen Codierarbeit von Microsoft übernommen. Nahezu unmöglich erscheint es mir, eine OLE-fähige Applikation in reinem C Code zu schreiben. Die vorhandenen Klassen für OLE-Dokumente, -Controls und -Server, inklusive *Inplace Activation* und anderen Feinheiten, reduzieren die eigene Arbeit auf ein erträgliches Maß. Statuszeilen und Toolbars wurden von der MFC schon bereitgestellt, lange bevor mit Windows 95 entsprechende Controls eingeführt wurden. Die MFC verwendet nun intern die neu vorhandenen Controls und bleibt trotzdem kompatibel zu Vorgängerversionen, eines der besten Beispiele für die Hintergedanken von C++ und objektorientierter Programmierung. Abseits des Window APIs inkludiert die MFC-Klassen für die Verwaltung von indizierten Listen, verketteten Listen und Hash Tabellen. Die Algorithmen sind durch die Verwendung von Templates typunabhängig. Eine Klasse für AVL-Bäume wäre noch wünschenswert. Neu ist die Einführung der Multithreadfähigkeit und diverser Synchronisationsmechanismen. In einem Punkt zieht nun endlich Visual C++ dem Leistungsumfang von Visual Basic nach, in der Datenbankunterstützung. Der Datenbankkern (Jet Engine) von Access 95 und Visual Basic 4.0 wird nun auch über die MFC DAO (Data Access Objects) Klassen ansprechbar. Access- und Visual Basic Programmierer finden die bekannten Datenbankobjekte schnell wieder. Neulinge der Datenbankprogrammierung haben einiges an Lernabeit vor sich. Die Verbindung mit dem mitgelieferten Tool Crystal Reports läßt jedoch noch zu wünschen übrig.

Leider ist nicht alles Gold was glänzt, so fehlen der MFC wichtige API's wie DirectX, DCI und OpenGL, sodaß man wieder auf die Standard C Windows Programmierung zurückgreifen muß. Auch stellt sich die Frage, ob der durch die MFC erzeugte Overhead zeitkritische bzw. schnelle Anwendungen zuläßt. Als größtes Manko der MFC galt seit Einführung die schwache und unvollständige Dokumentation. Diese wurde nun zwar deutlich verbessert, doch ist die Einarbeitungszeit selbst für C Profis sehr hoch.

Schwierig wird es vor allem dann, wenn man die, durch die MFC vorgegebene, Applikationsstruktur verändern muß; daß dies jedoch möglich ist, zeigt die Entwicklungsumgebung selbst, die nach Aussage von Microsoft vollständig mit Visual C++ und der MFC entwickelt wurde.

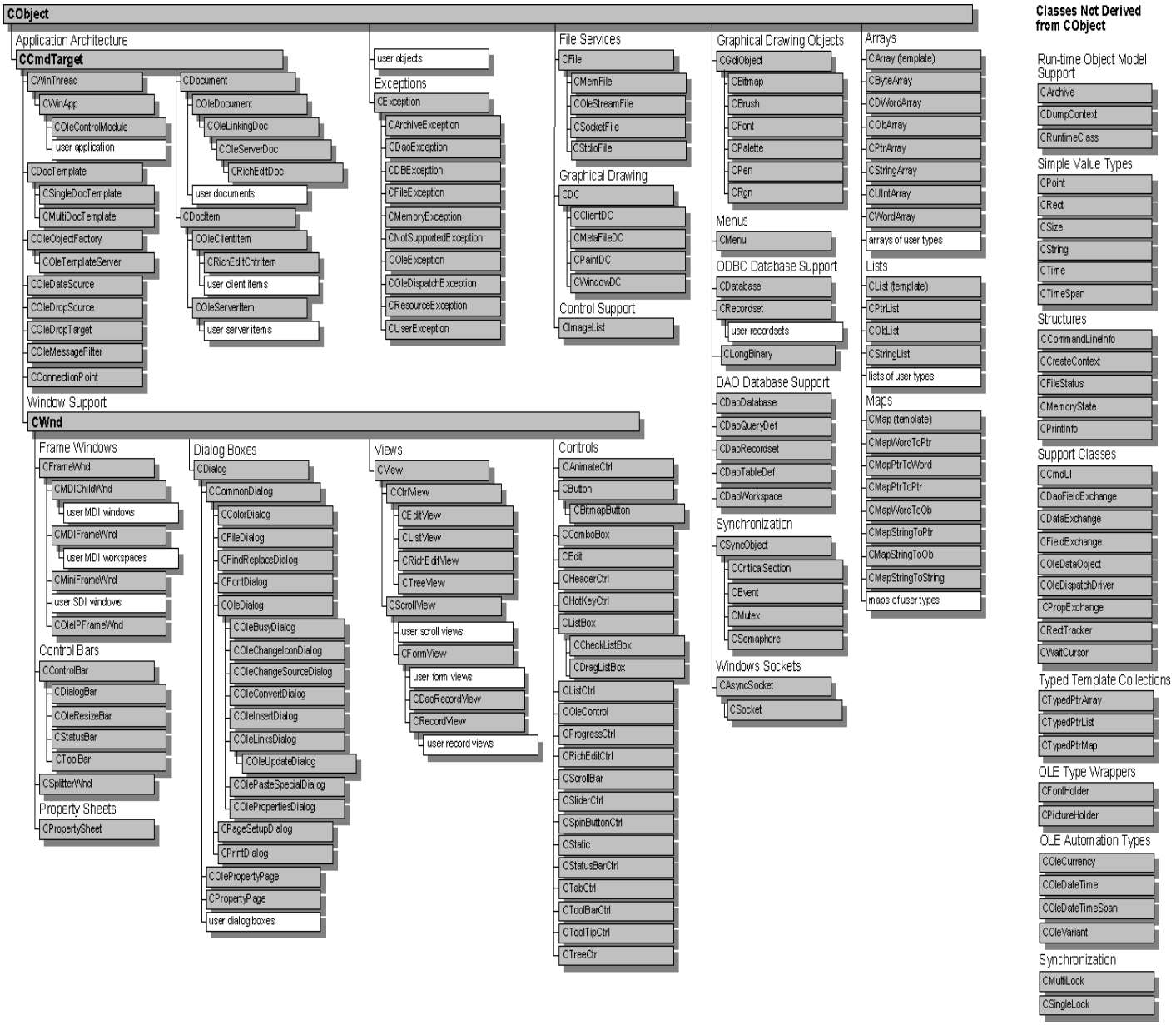
Fazit

Der bei Microsoft inzwischen zur Regel gewordene, größere Versionssprung scheint bei Visual C++ das erste Mal gerechtfertigt zu sein, und die Konkurrenz hat wieder einmal ein neues Ziel vor Augen. Wie bei neuen Produkten leider üblich, gibt es auch einige Bugs, sodaß bereits ein Patch File verfügbar ist.

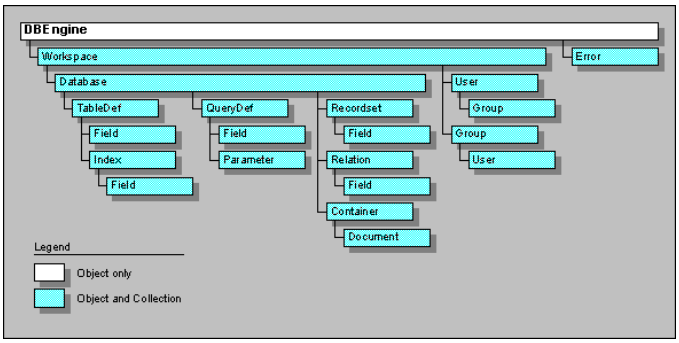
Nicht vergessen werden darf die Tatsache, daß Visual C++ neben den durch NT abgedeckten Plattformen auch als Cross Platform Development Kit for Macintosh Systems erhältlich ist. Dadurch ist es möglich, MFC-Programme für Apple Computer zu erstellen. Die gesteigerten Macintosh-Aktivitäten durch Microsoft läßt mir noch Platz für die Ge-

rüchteküche:
Plant Microsoft den Kauf eines Computerherstellers ;-) ? □

Foundation Classes Version 4.0



Data Access Object Hierarchy Chart



Glossar *(Im Text nicht erklärte Begriffe):*

API (Application Programming Interface): Von einem System zur Verfügung gestellte Funktion zur Erstellung von Applikationen.

DirectX Bestehend aus DirectDraw, -Sound, -Input, -Video und -3D wurde dieses API entwickelt um das langsame Hardwareunabhängige Windows API zu umgehen. Mit diesen API's kann die Hardware auf direkterem und schnellerem Wege angesprochen werden (spezielle Treiber sind erforderlich). DirectVideo und Direct3D befanden sich zur Zeit der Drucklegung noch im Entwicklungsstadium.

MAPI (Messaging API): API zur Unterstützung von Mail Services.

OCX (OLE Control): Weiterentwicklung der VBX Controls. Für 16Bit und 32Bit Windows verfügbar. Basiert auf der OLE Schnittstelle.

OLE (Object Linking and Embedding): Standard um Objekte zwischen Applikationen auszutauschen.

OLE Automation Möglichkeit zur Manipulation eines Objekts außerhalb Applikation. Meist über eine eigene Script Sprache.

OLE Events Von OLE Controls ausgelöste Ereignisse.

OpenGL (Open Graphics Library): Von Silicon Graphics entwickelte Grafikschnittstelle. Ursprünglich für den Workstation Bereich gedachtes Interface mit besonderem Schwerpunkt in der 3D-Grafik.

VBX (Visual Basic Control): Für Visual Basic entwickelter Standard zur Einbindung eigener Steuerelemente (Für 32Bit Windows nicht mehr verfügbar).