wdv-notes 234
Stand: 15.OKT.1993 (4.)
Wiss.Datenverarbeitung
FREIE UNIVERSITÄT BERLIN

PC-Software: Microsoft *VISUAL C++* V 1.0.

© 1993–1994 *Karl-Heinz Dittberner*

PC

Alle grafischen Benutzeroberflächen (GUIs) sind dafür verschrien, daß für sie die Entwicklung von Anwendungs-Programmen äußerst schwierig sei. Solche Schwierigkeiten basieren aber hauptsächlich auf der Tatsache, daß es sich bei allen GUI-Systemen (z. B. MacOS, X-Windows, Windows) um reine *ereignisgesteuerte* (event-driven) Software handelt, die eine völlig andere Vorgehensweise beim Programmieren erfordert.

In den GUI-Systemen werden vom Anwendungsprogrammierer quasi nur Programm-Versatzstücke zur Verfügung gestellt. Erst der Benutzer des Computers verbindet diese Programmstücke durch seine Handlungen (Mausklicks, Eingaben usw.) – nach seinen jeweiligen Wünschen – zum Gesamtprogramm. Der Programmierer muß sich daher bei der Entwicklung sehr in all die möglichen Vorgehensweisen von Benutzern hineindenken. Nur das ist die „Schwierigkeit"!

Die GUI-Systeme beruhen also darauf, daß der Informationsfluß zu den Menüs, Rollbalken und anderen Objekten durch die Triggerung von Geräten (z. B. durch den Mausklick, durch das Bewegen der Maus, durch den Tastendruck) irgendwann, nicht im Programm vorher bestimmbar, also absolut *asynchron*, ausgelöst wird. Dieses Prinzip ermöglicht es, eine wirklich benutzerfreundliche Software zu entwickeln, die ein vollkommen intuitives Arbeiten mit einem Computer erlaubt. Erst dadurch wird der Computer für den Menschen zu einem richtigen Werkzeug.

Die programmtechnische Behandlung derartiger asynchroner Systeme ist bereits seit den 60er-Jahren (!), vor allem durch die Arbeiten des Finnen Dijkstra [z. B. 1], theoretisch geklärt (Stichworte: *Prozeß-Synchronisierung, Semaphoren, Prozeß-Scheduling, Message/Mail-Boxen, quasi-parallele Prozesse, dynamische Datenstrukturen*) [2]. Diese wichtigen Ideen wurden auch bei der Behandlung von Programm-Interrupts durch Peripheriegeräten in Gerätetreibern (Handlern) von Betriebssystemen verwendet. Es ist aber das große Verdienst eines Steve Jobs all diese Prinzipien erstmalig und konsequent bei einem kommerziellen Produkt – dem Apple Macintosh Computer – zum Nutzen aller Computer-Anwender eingeführt zu haben [3]. Heute gehören solche grafischen Benutzeroberflächen zum allgemeinen Standard der gesamten Computer-Industrie.

Für eine einfachere Programmentwicklung für GUIs werden von der Industrie heute viele objektorientierte Compiler und Softwarewerkzeuge angeboten, auch für den PC [4]. Hierzu gehört auch das ANSI-konforme ***VISUAL C++*** der Firma Microsoft, mit dem Programme für ***Windows*** *und* für das neue ***Windows NT*** [5] – auch Portierungen – entwickelt werden können.

In diesem Merkblatt wird das neue ***VISUAL C++*** vorgestellt. Diese Beschreibung basiert auf der 32-Bit „Preliminary Edition" der NonUSA-Fassung vom ***VISUAL C++*** Version 1.0, die seit Anfang Oktober 1993 offiziell vorliegt.

1. Zur Position von *VISUAL C++*:

Die grafischen Benutzeroberflächen (GUIs) bilden eine Welt von Objekten und Ereignissen, die man durch eine objektorientierte Hochsprache, wie z. B. C++, in den Griff bekommen kann. Eine Assemblerprogrammierung wäre viel zu fehlerträchtig und daher zu teuer. Für eine Programmierung im wegweisenden Oberon ist es noch viel zu früh.

Die wichtigsten C++-Compiler für PCs mit 80x86-Chips wurden kürzlich von der BYTE vergleichend getestet [4]. Die Compiler teilen sich in zwei Klassen. Ein visuelles Programmieren erlauben: Visual C/C++ V 1.0 von Microsoft, C++ V 3.1 von Borland, C/C++ V 6.0 von Symantec. Nur mit einem klassischen Befehlsinterface versehen sind: MetaWare's High C/C++ V 3.1 und das C/C++32 von Watcom.

Zum Lieferumfang der Compiler von Microsoft, Borland und Symantec gehören jeweils ein umfangreicher Satz von Softwarewerkzeugen für eine angenehme Programmentwicklung für *Windows* und *Windows NT*. Die Compiler von MetaWare und Watcom sind da bescheidener ausgestattet, sie erzeugen aber „schnelleren" Maschinencode. Der Compiler von Watcom unterstützt auch die Programmentwicklung für IBM's *OS/2*–V 2.1.

Mit 195,– $ ist *VISUAL C++* von Microsoft die preiswerteste und umfangreichste Entwicklungsumgebung. Die Profi-Ausrüstung von *VISUAL C++* kostet 495,– $ und kann auch für MS-DOS eingesetzt werden.

2. Eigenschaften von *VISUAL C++*:

Durch die Verwendung der integrierten Entwicklungsumgebung *VISUAL C++* wird eine Programmentwicklung für *Windows* (*NT*) tatsächlich deutlich vereinfacht und produktiver; noch einfacher wäre aber für die Zukunft wünschenswert.

Mit dem *AppWizard* beginnt man jede Programmentwicklung. Mit ihm lassen sich alle für die neue Applikation benötigten Objekt-Klassen und deren Parameter skizzieren. Der zugehörige Code wird dann mit den anderen Werkzeugen erzeugt.

Alle Ressourcen eines Anwendungsprogramms – insbesondere die Objekte der Benutzerschnittstelle (Dialogboxen und Menüs) – werden mit dem Ressourcen Editor *AppStudio* (Apples ResEdit läßt grüßen!) angefertigt. Das *AppStudio* ist mit dem *ClassWizard* gekoppelt, mit dem der Objekt-Code angefertigt wird.

3. Zur Arbeit mit *VISUAL C++*:

Wie fertigt man nun konkret eine Benutzerschnittstelle an? Im *ClassWizard* wird zunächst die gewünschte Objekt-Klasse ausgewählt (z. B. eine Dialogbox). Vom *AppStudio* wird darauf der Katalog der verfügbaren Objekte angezeigt, aus dem per Mausklick das gewünschte Objekt ausgewählt wird. Aus einer Liste kann nun die benötigte Message-Art ausgewählt werden. Nach dem Klicken auf die Taste »Add-Function« wird ein Name für die neue Funktion eingeben. Der *ClassWizard* fügt diese Deklaration automatisch der Objekt-Klasse zu, schreibt einen (noch) leeren Funktionskopf in die Quellendatei und schaltet auf den Text-Editor um. Mit diesem wird nun der Code eingegeben.

Im Hintergrund sorgen automatisch die *Microsoft Foundation Classes* (MFC) für die richtigen Verknüpfungen bezüglich des Event-Handlings. Das *MFC* stellt für alle Objekte von *Windows* (*NT*) die Deklarationen der Objekt-Klassen zur Verfügung. Der Programmierer muß sich also um das eigentliche Event-Handling nicht selbst kümmern. Das ist eine wesentliche Erleichterung und die geeignete Basis zur Entwicklung von komplexen Applikationen.

4. Die Installation von *VISUAL C++*:

Die Installation von *VISUAL C++* auf einem PC ist sehr leicht. Der PC muß dazu mit einem CD-ROM Laufwerk ausgestattet sein, denn das *VISUAL C++* wird ausschließlich zusammen mit der Dokumentation auf einer CD-ROM geliefert.

Der PC muß mit mindestens 6 MByte RAM-Speicher ausgestattet sein. Auf der Platte belegt das VISUAL C++ etwa 52 MByte.

5. Die Zusatz-Werkzeuge:

Zum *VISUAL C++* gehören alle Quellenprogramme der Klassenbibliothek *MFC* 2.0 mit Browser, vorcompilierte Header, ein symbolischer Debugger, ein Profiler (nur im Profi-Kit) und speziell für die Windows-Umgebung: Ein Heap Walker, der Message Spy, ein solider Crash Analysator, ein Ressource Editor und ein Compiler für die bequeme Anfertigung von Online-Hilfen für die Anwendungsprogramme.

6. Sonstige Hinweise:

■ Das *VISUAL C++* kann zur Zeit noch keinen *speziellen* Code für Intels Pentium (80586) erzeugen.

7. Literatur:

[1] Dijkstra, E. W.: Cooperating Sequential Processes. *In:* Genuys, E. (Ed.): Programming Languages. London: Academic Press 1968, pp 43–112.

[2] Dittberner, K.-H.: IfPScheduler – Verwaltung von (quasi-)parallelen Prozessen. FU Berlin (IfP): wdv-notes Nr. 215, 1988–1993.

[3] Dittberner, K.-H. (Ed.): Wird die PC-Welt veräppelt? FU Berlin (IfP): wdv-notes Nr. 202, 1991–1993.

[4] Grehan, R.: C++ Does Windows. The BYTE Lab tests five C++ compilers to find out if C++ really simplifies Windows development. BYTE 18, No. 10, 130–136 (Sep. 1993).

[5] Dittberner, K.-H.: Windows NT: Die System-Applikationen. FU Berlin (IfP): wdv-notes Nr. 225, 1992–1993.

© khd-research • 2004 • Eine Publikation von http://www.khd-research.net/ • File: Notes/wdv-notes_234.pdf