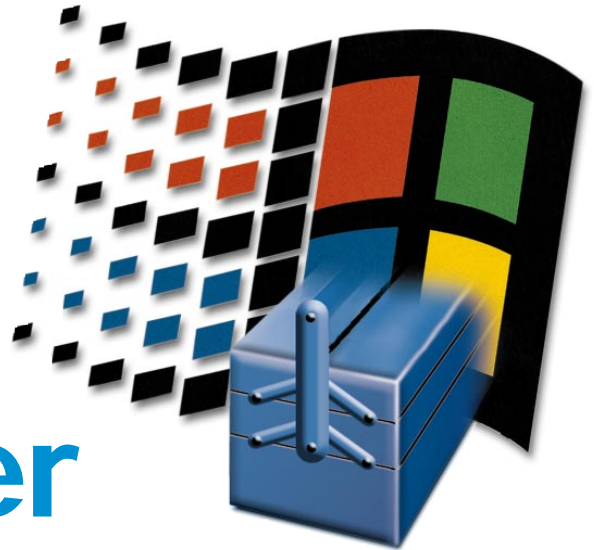




C++ unter Windows -
Schnelleinführung zu Visual C++

Sprung ins kalte Wasser



Das Standardwerkzeug für die Programmierung von Win32-Anwendungen mit C++ ist Microsofts Visual C++. Wer gleich loslegen möchte findet hier einen Schnelleinstieg in das C++-Programmieren mit Visual C++.

THOMAS WÖLFER

Eine Autorenkopie von Microsofts Visual C++ finden Sie auf der Heft-CD. Diese Version ist für Lernzwecke völlig ausreichend – die Software liegt im Wesentlichen in der Form vor, in der auch die kommerzielle Version erhältlich ist.

Beim Einstieg in C++ bietet es sich an, zunächst mit Konsolen-Programmen zu beginnen. (Später in diesem Beitrag erfahren Sie, wie man "echte" Windows-Programme mit VC++ entwickelt.). Ein Konsolen-Programm ist ein textbasiertes Programm, das unter Windows betrieben werden kann. Ein mit VC++ ent-

wickeltes Konsolen-Programm ist dabei sehr wohl 32-bittig. Es handelt sich nicht um ein MS-DOS-Programm, hat aber keine Fensterelemente oder sonstige optische Eigenschaften von Windows-Programmen. Das hat einen klaren Vorteil für das Erlernen von C++: Der heftige Overhead, der durch die "Windowisierung" entsteht, entfällt. Man kann sich also ganz der Sprache widmen.

Zunächst muss Visual C++ installiert werden. Sofern man keine besonderen Vorstellungen für zukünftige Entwicklungen hat, können dabei alle Vorgabewerte des Installationsprogramms übernommen werden. Ist VC++ installiert, kann es einfach aus dem Start-Menü heraus aufgerufen werden.

Visual C++ auf Konsolenprogramm einstellen

Um ein Projekt zu beginnen, verwendet man für ein Konsolen-Programm am besten den passenden Wizard. Der findet sich, indem man im "Datei" (File)-Menü auf "New" klickt und in der daraufhin angezeigten Dialogbox den Reiter "Projects" auswählt. Auf der Dialogbox werden eine ganze Reihe von Projektarten vorgeschlagen – auswählen sollte man hier

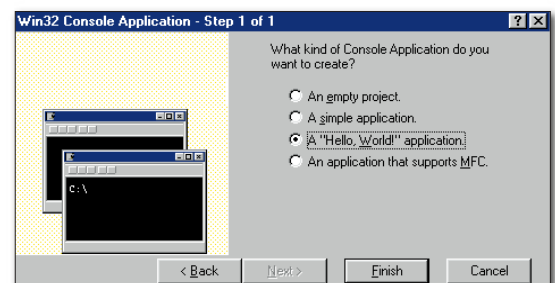
"Windows Console Application". Daraufhin wird der Wizard angezeigt, der einige unterschiedliche Arten von Konsolen-Anwendungen vorschlägt. Für welche man sich entscheidet, ist im Prinzip egal, denn alle erzeugen ein kleines Projekt mit nur wenigen Zeilen Quellcode und dienen eher der Orientierung.

Der Wizard soll bei Konsolen-Projekten nicht, fertigen, wiederverwendbaren Quellcode erzeugen (bei anderen Anwendungstypen ist dies ein Grund, den Wizard zu verwenden), sondern die "richtigen" Compiler- und Linker-Einstellungen für das Projekt vornehmen. Diese Einstellungen sind von Projektart zu Projektart unterschiedlich. Abhängig davon, welches Projekt man welcher Zielformat anlegen will, müssen unterschiedliche #defines gesetzt, Libraries mitgelinkt und sonstige Schritte unternommen werden. Der Wizard nimmt es einem ab, darüber genau Bescheid zu wissen – und für die ersten Schritte mit der Entwicklungsumgebung ist das angebracht.

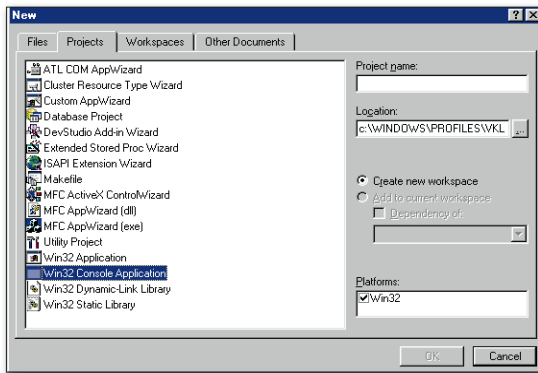
Dennoch kann es nicht schaden, die Projekteinstellungen zu überfliegen und den einen oder anderen Text aus der Online-Hilfe zu lesen. Die Projekteinstellungen findet man, indem man zunächst

STEP BY STEP

- ▶ **Visual C++ auf Konsolenprogramm einstellen**
Schreiben Sie erst einmal eine Zeile C
- ▶ **Den Debugger nutzen**
Und dann Debuggen Sie
- ▶ **Ab in die Praxis**
Ein bißchen Programmieren tut gut
- ▶ **Die Microsoft Foundation Classes nutzen**
Hinein in C++ mit Microsoft
- ▶ **Eine Reaktion auf ein Ereignis programmieren**
"Hello World" auf den Button gebracht



MIT DIESEM WIZARD erzeugt man eine Konsolen-Anwendung.



EINE C-ANWENDUNG können Sie entweder als 'Win32 Application' oder als 'Win32 Console Application' erstellen. Unser erstes Beispiel soll eine Console Application werden.

im "View"-Menü den "Workspace" aktiviert, sofern das noch nicht der Fall sein sollte. Visual C++ blendet ein Fenster ein, das am unteren Rand zwei oder drei Reiter hat und bei dem mit Hilfe des Reiters "Files" alle am Projekt beteiligten Dateien eingeblendet werden.

Im Falle eines Konsolen-Projekts sind es drei Dateien, wobei man die beiden namens "stdafx.*" zu Anfang ignorieren kann. Ist die Dateiansicht aktiviert, so kann man mit der rechten Maustaste hineinklicken und den Befehl "Settings" auswählen. Dieser Befehl blendet den "Projekt"-Dialog ein. Hier finden sich alle Compiler-, Präprozessor und Linker-Einstellungen sowie alle anderen Einstellungen, die für den Bau des Zielprogramms notwendig sind. Dabei können viele Einstellungen auch für einzelne Dateien innerhalb des Projekts vorgenommen werden. Dazu muss die betroffene Datei angeklickt werden – der rechte Teil des Projekt-Einstellungsdialogs verändert sich dann so, dass nur die für diese Datei möglichen Einstellungen übrigbleiben.

■ Den Debugger nutzen

Soviel zur Projektverwaltung. Wichtiger ist der Debugger. Er ist für das Lernen der Sprache ein ungeheuer hilfreiches Werkzeug. Mit wird ein Programm in einzelnen Schritten, von Quellcode-Zeile zu Quellcode-Zeile ausgeführt. Nach jedem Schritt können der Inhalt von Variablen, der Zustand der Anzeige und auch beliebige Speicheradressen angesehen werden. Es ist möglich in der Entwicklungsumgebung sogenannte "Breakpoints" zu setzen. Ein Breakpoint wird dabei innerhalb des Quellcode-Editors platziert und im Quellcode farblich markiert. Lässt man das

Programm laufen, bleibt es automatisch an der Zeile stehen, in der der Breakpoint gesetzt wurde. Sobald das Programm steht, können alle oben geschilderten Untersuchungen vorgenommen werden. Ist man mit diesen fertig, kann man das Programm einfach weiterlaufen lassen, oder man läuft zeilenweise durch die nächsten Anweisungen. Lässt man das Programm nicht im Einzelschritt-Modus laufen, so läuft es entweder bis zum nächsten Breakpoint oder bis das Ende des Programms erreicht wurde.

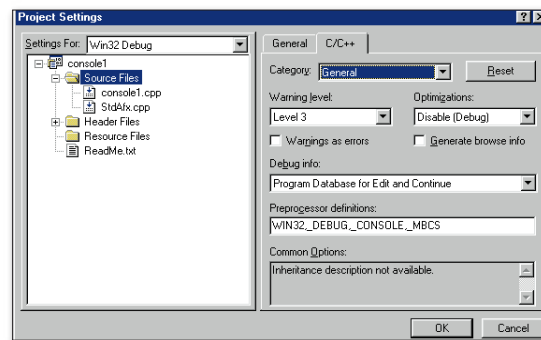
■ Ab in die Praxis

Nun zur Praxis. Als sinnvolles Übungsbeispiel dient der Quellcode "Bsp2" aus dem Beitrag "Einführung in C++" in

fehl betätigt wurde, öffnet VC++ ein weiteres Fenster, das die Meldungen des Compilers und Linkers enthält. Nachdem der fertige und problemlos übersetzbare Beispielquellcode verwendet wurde, bestehen diese Meldungen ausschließlich aus solchen positiver Natur. Wenn später beim Übersetzen einmal Fehler auftreten, so ist dieses Fenster von besonderer Wichtigkeit. Man kann darin auf eine Fehlermeldung doppelklicken. Dieser Doppelklick öffnet ein Fenster im Quellcode-Editor, innerhalb dessen der Cursor auf der Zeile steht, die den Fehler enthält. (Man kann den Cursor ebenso mit der Maus auf die Fehlernummer platzieren und F1 drücken – so erhält man am schnellsten eine Erklärung zum aufgetretenen Fehler.)

■ Debuggen muss sein

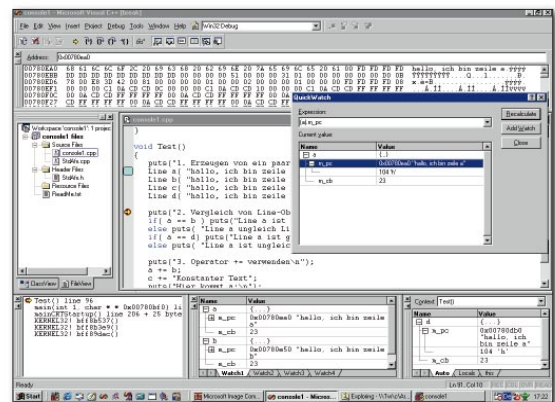
Nun liegt ein fertig übersetztes C++-Programm vor und das Debuggen kann beginnen. Zu diesem Zweck setzt man am einfachsten einen Breakpoint auf eine der ersten Zeilen im Programm (d.h. auf eine Zeile innerhalb von main()) – das geht am schnellsten mit dem Hotkey F9. (Ein weiterer Druck mit F9 auf der gleichen Zeile schaltet den Breakpoint wieder aus.) Damit wird ein Breakpoint auf dieser Zeile gesetzt – der Debugger hält das Programm an und übernimmt die Kontrolle, sobald das Programm diese Zeile erreicht hat. Es gibt noch eine ganze Reihe anderer Breakpoints, die sich erst nach einer vorgegebenen Anzahl von Durchläufen bemerkbar machen. Außerdem kann man



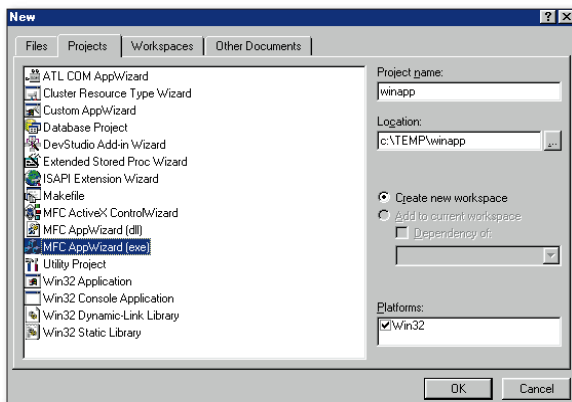
MIT DEN EINSTELLUNGEN für ein Projekt hat man zunächst wenig zu tun. Benötigt man sie dennoch einmal ist der übersichtliche Projektmanager von VC++ sehr hilfreich.

diesem Heft. Öffnen Sie im Quellcode-Editor eine relevante C++-Datei und ersetzen Sie deren kompletten Inhalt durch den der angegebenen Beispieldatei. Sie finden sie zum Download bei www.pc-magazin.de/download/spezial_21/crashkurs/, denn wir hatten wenig Platz auf der CD. Bitte entpacken Sie diese Datei vorher in ein geeignetes Verzeichnis.

Nach diesem Vorgang liegt ein komplett lauffähiges Programm vor, welches zunächst noch übersetzt werden muss. In der VC++-Entwicklungsumgebung erfolgt dies mit dem Befehl "Build" aus dem "Build" Menü. Nachdem dieser Be-



VC++-DEBUGGER IN AKTION: Bevor man ernsthaft anfängt, mit VC++ zu programmieren, sollte man sich mit den Fähigkeiten des Debuggers intensiv vertraut machen.



ALS ERSTES WINDOWS-Projekt sollte man eine MFC-Anwendung (exe) erzeugen.

Breakpoints definieren, die von Variablen-Inhalten abhängig sind. (Programm anhalten, wenn die Variable "N" einen bestimmten Wert annimmt.) Alle Möglichkeiten zum Setzen von Breakpoints befinden sich im Edit (Bearbeiten)-Menü unter "Edit Breakpoints".

Erster Start mit F5

Ist der Breakpoint gesetzt, kann das Programm gestartet werden. Natürlich nicht von der Kommandozeile aus (dies geht prinzipiell auch, bringt jedoch fürs Debugging nichts), sondern von innerhalb der Entwicklungsumgebung. Der benötigte Hotkey ist F5. Hätte man diese Taste vor dem Übersetzen betätigt – hätte VC++ die Übersetzung automatisch gestartet. Zu Demonstrationszwecken ist es sinnvoller und übersichtlicher, dies in einzelnen Schritten zu tun.

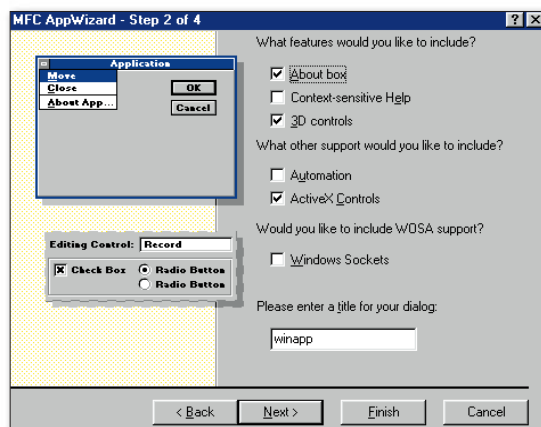
Wurde das Programm gestartet, wird es geladen und bleibt am Breakpoint stehen. Der Debugger wartet auf Eingaben. Es stehen verschiedene Möglichkeiten zur Verfügung. So können die verschiedenen Debug-Ansichten aus dem "View"-Menü ausprobiert werden. Mit diesen kann man das Programm im aktuellen Zustand aus verschiedenen Gesichtspunkten betrachten. Es können die Inhalte von Variablen betrachtet werden. Die Debugging-Fenster von Visual C++ unterstützen übrigens in gleicher Form Drag&Drop wie auch alle anderen Fenster: Man kann einfach eine Variable innerhalb des Quellcode-Editors markieren und dann per Drag&Drop in eine der Debug-Ansichten ziehen. Je nach

Ansicht werden die entsprechenden Informationen angezeigt.

Schritt für Schritt

Nachdem sich das Programm unter der Kontrolle des Debuggers befindet, kann dieser dazu benutzt werden, schrittweise durch den Quellcode zu gehen. Dazu dienen die verschiedenen Befehle des "Debug"-Menüs. Die wichtigsten sind "Step into (F11)" und "Step over (F10)". Mit diesen Funktionen kann am einfachsten durch den Quellcode gelaufen werden. Das Programm arbeitet dabei die Zeilen, so dass die Auswirkungen der Instruktionen im Ausgabefenster sichtbar werden.

Das schrittweise Abarbeiten des



DIE VERSCHIEDENEN OPTIONEN einer Dialogbox-Anwendung haben zunächst keine praktische Bedeutung. Man kann die vorgeschlagenen Einstellungen übernehmen.

Quellcodes im Debugger ist nicht nur ein probates Hilfsmittel beim Lernen, sondern eine unerlässliche Funktion beim ganz normalen Programmieren. Nur mit einem Debugger kann man die richtige Funktionsweise eines Programms sicherstellen. Weitere wichtige Funktionen finden sich ebenfalls im Debug-Menü. So ist es möglich, den Instruction Pointer (den Zeiger, der auf die aktuelle Instruktion zeigt) an eine andere Stelle zu verschieben und so in den Ablauf des Programms manuell einzugreifen.

Die Microsoft Foundation Classes nutzen

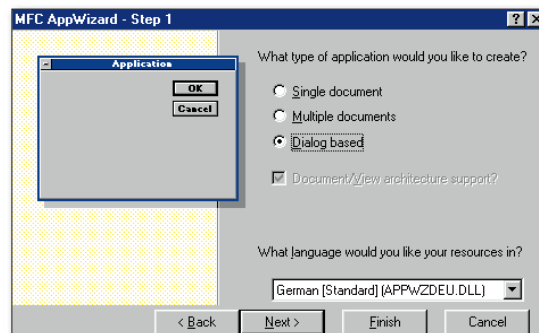
Nach diesen elementaren Grundlagen folgt ein Schnelleinstieg in die Entwicklung von Windows-Programmen mit MFC unter Visual C++.

Dieser kann ein ausführliches Tutorial nicht ersetzen. Er versetzt Sie jedoch in die Lage ein eigenes Windows-Programm zu schreiben, das man als Ausgangspunkt für die weitere Erforschung verwenden kann. Sobald man sich in der Entwicklungsumgebung von VC++ einigermaßen heimisch fühlt, sei das "Scribble"-Tutorial, das in Form von Online-Dokumentation zu VC++ gehört, empfohlen. Anders als das im folgenden vorgestellte Programm ist "Scribble" ein Programm mit multiplen Fenstern, Werkzeugleisten und "echter" Interaktionsmöglichkeit. Allerdings ist es auch entsprechend komplizierter als das folgende Beispiel. Mit diesem wird lediglich ein Programm generiert, das im wesentlichen aus einer Dialogbox mit einem anklickbaren Button besteht.

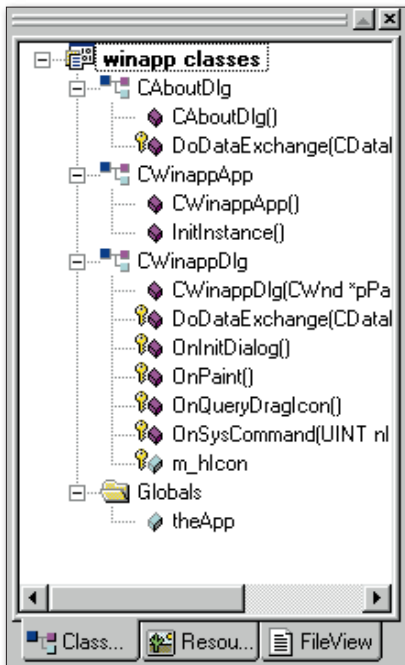
Ähnlich wie beim Konsolen-Programm verwendet man zum Anlegen eines neuen Projekts für Windows-Programme am besten den Wizard, der über "Datei-Neu" (File-New) erreicht werden kann. Als Projekt-Typ wird für dieses Beispiel "MFC AppWizard (exe)" ausgewählt.

Zwar gibt es andere Methoden, mit denen man Windows-Anwendungen erzeugen kann, die MFC (Microsoft Foundation Classes) basierten Projekte jedoch sind zum Einstieg am einfachsten zu handhaben. Die MFC-Bibliothek übernimmt sehr viel Handarbeit, so dass man sich um die eigentliche Problematik kümmern kann.

Für das Projekt müssen ein Name und ein Pfad auf der Festplatte angegeben



DIE EINFACHSTE FORM einer MFC-Windows Anwendung ist eine Dialogbox basierte. Damit fällt das Lernen leicht.



DIE "CLASSVIEW" zeigt es: Eine echte Windows-Anwendung hat von Haus aus deutlich mehr Inhalt im Grundgerüst, als für eine Konsolen-Anwendung notwendig ist.

werden, dann gelangt man zum nächsten Schritt des Wizards. Hier wird festgelegt, welche MFC-Anwendung erzeugt werden soll.

Zur Auswahl stehen "Single Document", "Multiple Document" und "Dialog based". Der einfachste Anwendungstyp ist "Dialog based". Dabei wird eine Anwendung erzeugt, die im wesentlichen aus einer Dialogbox besteht. Das ist einfach und übersichtlich, und wird deshalb für dieses Beispiel verwendet.

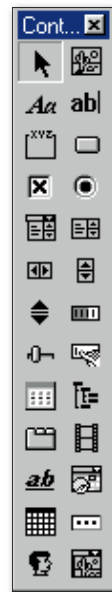
Allerdings besitzt auch eine dialogboxbasierte Anwendung eine ganze Reihe von Optionen, die im Zug mehrerer Schritte mit dem Wizard eingestellt werden können.

Für dieses Beispiel sollen die einzelnen Optionen nicht erläutert werden, man findet dazu eine Erläuterung in der Online-Hilfe zum Wizard. Für das Beispielprogramm können die vorgeschlagenen Vorgabewerte ruhig übernommen werden, denn auf den tatsächlichen Aufbau des Beispielprogramms haben sie keine Auswirkungen: Nur Detailfragen werden hier eingestellt. Für das spätere Programmieren von Anwendungen ist es sicherlich wichtig, ob ein Programm über Support für ActiveX verfügt oder nicht – für dieses Beispiel spielt das keine Rolle.

Ist der Wizard schließlich abgearbeitet, hat VC++ ein neues Projekt angelegt. Anders als bei der Konsolen-Anwendung enthält das neue Projekt bereits eine ganze Reihe von Dateien und Klassen – es lohnt sich auf jeden Fall, einen Blick in das "Workspace"-Fenster zu werfen und die erzeugten Klassen und deren Methoden zu begutachten. Man findet dabei sehr schnell heraus, dass das bisher generierte Programm durchaus einiges an Komplexität zu bieten hat. Dabei wurde noch nicht eine einzige Zeile programmiert!

Interessant dabei: Der generierte Code kann jetzt schon übersetzt und gestartet werden (F5). Es liegt also ein vollkommen funktionstüchtiges Windows-Programm vor. Deshalb sollten Sie das Programm tatsächlich einmal übersetzen und ein wenig damit herumspielen. Die angezeigte Dialogbox verfügt zwar noch nicht über viele Funktionen, hat aber bereits zwei Buttons, mit der sie geschlossen werden kann. Und auch einen "About" (Info über)-Dialog, der über das System-Menü des Programms angezeigt werden kann. Und alles, ohne dass auch nur eine einzige Zeile programmiert wurde.

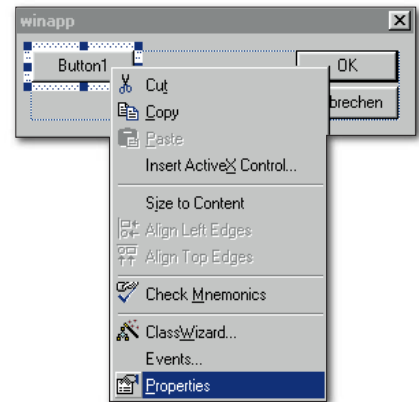
Um die Euphorie an dieser Stelle etwas zu bremsen: Der Rest des Weges zum Windows-Programmierer ist nicht ganz so einfach, denn die Automatismen enden an dieser Stelle. Von nun an gibt es zwar noch den einen oder anderen Wizard, der einem weiterhilft, aber die zu implementierende Logik des Programms muss im weiteren Verlauf dann doch vom Programmierer entwickelt und implementiert werden.



Das bisherige Programm wird nun um einen Button erweitert. Klickt man auf diesen, soll eine weitere Dialogbox mit der bekannten "Hello World"-Nachricht angezeigt werden. Und das geht so:

Zunächst muss die Dialogbox mit dem zusätzlichen Button ausgestattet werden – dazu benötigt man den Dialogbox-Edi-

DIE DIALOGBOX-Kontrollelemente können aus dieser Werkzeugleiste per Drag&Drop auf den Dialog geschoben werden.



MIT EINEM RECHTEN Mausklick erreicht man die Eigenschaften des Buttons am schnellsten.

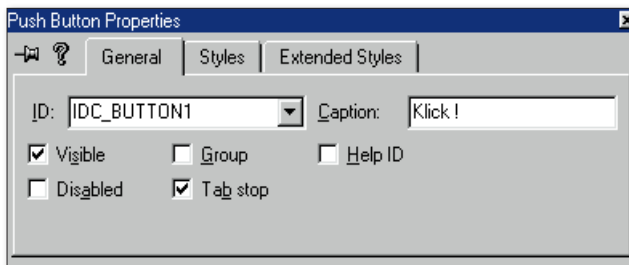
tor. Dieser öffnet sich automatisch, sobald eine Dialogbox-Ressource in der Entwicklungsumgebung geöffnet wird. Um dies zu tun, wechselt man in die "Ressource"-Ansicht des Workspaces, öffnet dort den Ast "Dialogs" und doppelklickt auf den gewünschten Dialog. Dieser wird in der Entwicklungsumgebung geöffnet und dort in etwa so dargestellt, wie das auch beim laufenden Programm der Fall ist. Zusätzlich werden Hilfsraster, Hilfslinien und einige Werkzeuge eingeblendet. Eines der Werkzeuge ist die "Controls"-Werkzeugleiste.

Diese Werkzeugleiste enthält alle Elemente, die auf einer Dialogbox untergebracht werden können, wie Buttons, Listboxen, Textelemente und so weiter. Um einen Button auf dem Dialog zu platzieren, klickt man auf das "Button"-Symbol in der "Controls"-Leiste, hält den Mausknopf gedrückt und zieht das Symbol auf den Dialog. Lässt man die Maustaste los, ist der Button platziert. Stimmt die Position nicht, kann man den Button nachträglich leicht mit der Maus verschieben.

Der Dialog hat jetzt einen neuen Knopf. Dieser trägt per Vorgabe den Namen (und den Text) "Button 1".

Das ist unschön. Besser wäre es, wenn der Button mit dem Text "Klick!" beschriftet wäre. Das ist die Veränderung, die im nächsten Arbeitsschritt vorgenommen wird.

Für die Vereinfachung der Arbeit mit dem VC++ ist es gut zu wissen, dass man auf nahezu jedes Element mit der rechten Maustaste klicken kann, um an ein "Eigenschaften"-Menü zu gelangen. Dies gilt auch für den soeben platzierten Button. Es wird nun innerhalb des Editors daraufgeklickt und dann der Befehl "Ei-



Ein Button hat deutlich mehr Eigenschaften, als man zunächst vermuten würde.

igenschaften" (Properties) ausgewählt. Daraufhin öffnet sich ein Fenster, mit dem die Eigenschaften des Buttons verändert werden können.

Ein Button hat zahlreiche Eigenschaften. Die Auswirkung von Veränderungen bei diesen Eigenschaften sind leicht auszuprobieren: einfach ändern, das Programm mit F5 neu übersetzen und starten. Die meisten Eigenschaften sind selbsterklärend oder aber beim Ausprobieren ersichtlich. Einzig die "ID", die per Vorgabe den Wert IDC_BUTTON_1 besitzt, ist erklärungsbedürftig. Bei dieser ID handelt es sich um das Symbol, unter dem mit dem Button später im Programm gearbeitet werden kann. Soll zur Laufzeit die Beschriftung des Buttons verändert werden, geschieht dies mit der Funktion SetDlgItemText(). Diese Funktion erhält zwei Parameter, und zwar den neuen Text und die ID des zu verändernden Elements. Das verwenden wir nicht, statt dessen wird der Text "Button 1" in den Text "Klick!" verändert. Dafür tippt man einfach den neuen Text im Eigenschaften-Dialog ein.

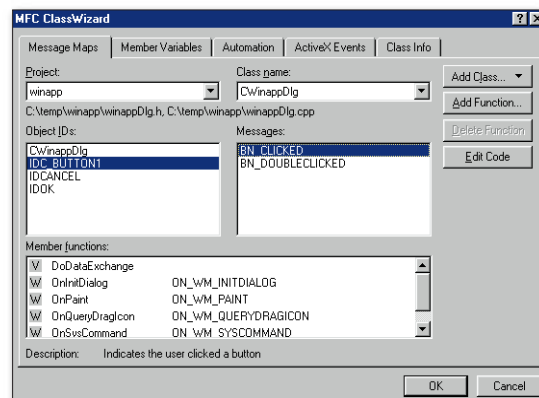
Übersetzt und startet man das Programm, sieht man bereits die erste Veränderung: Der Dialog hat einen neuen Button, und zwar einen, der nicht vom Wizard erzeugt wurde – einschließlich einer von Hand veränderten Beschriftung in "Klick!".

Allerdings: Man kann so oft auf den "Klick!"-Button klicken, wie man will, es tut sich nichts. Das ist auch logisch, denn es wurde noch keine Reaktion auf einen Klick programmiert. Das geschieht im folgenden und letzten Schritt dieses Beispiels.

Eine Reaktion auf ein Ereignis programmieren

Sofern das Programm noch läuft, sollte es jetzt beendet werden. Sie wenden sich nun wieder der Entwicklungsumgebung zu. Um den Button dazu zu bewegen, auf ein Ereignis (den Klick)

zu reagieren, verwendet man am besten ein weiteres Hilfsmittel von Visual C++: den Class Wizard. Mit ihm kann man Code an Ereignisse binden, Controls Variablen zuweisen und noch eine ganze Menge anderer interessanter Dinge tun. Alle diese Dinge sind zwar auch von Hand zu erledigen, der Wizard macht sie aber deutlich schneller. Trotzdem sollte man zu einem späteren Zeitpunkt herausfinden, was der Wizard eigentlich exakt macht bzw. welche Veränderungen er am Quellcode vornimmt. Dazu sei erneut auf das "Scribble"-Tutorial verwiesen.



Mit dem Class Wizard verbindet man Ressourcen mit Programmcode.

Um den Class Wizard zu starten, klickt man erneut mit der rechten Maustaste auf den Button und wählt den Befehl "Class Wizard". Dieser wird geöffnet und präsentiert sich in Form einer relativ umfangreichen Dialogbox.

Auf dem Reiter "Message Maps" finden sich jede Menge Kontrollelemente, darunter zwei große Listen mit dem Titel "Object-IDs" und "Messages". Bei den Object-IDs tauchen alle IDs auf, die innerhalb des Programms verwendet werden, unter anderem die Button-ID IDC_BUTTON_1. Wird auf das Objekt mit dieser ID, also auf den Button geklickt, soll eine Aktion erfolgen. Für diese muss eine Funktion implementiert werden. Deshalb wird zunächst die Button-ID

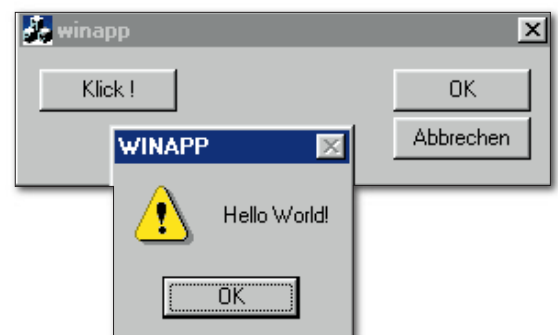
unter "Object IDs" ausgewählt, dann wählt man unter "Messages" die Nachricht mit der Bezeichnung BN_CLICKED aus. BN_CLICKED ist erneut eine ID – diese wird zur Laufzeit an den Button geschickt, wenn er gedrückt wurde. Um für diesen Fall eine Funktion zu schreiben, wird auf den Button "Add Function" geklickt und dann der vorgeschlagene Funktionsname mit OK akzeptiert. Der Wizard hat sich bereits darum gekümmert, dass eine Funktion aufgerufen wird, wenn der Button gedrückt wird. Jetzt muss diese Funktion noch mit Inhalt gefüllt werden.

Dafür klickt man im Class Wizard auf "Edit Code" und gelangt dadurch zurück in den Quelltext-Editor. Praktischerweise an genau die richtige Stelle innerhalb der zuvor angelegten Funktion. Für dieses Beispiel wird die Funktion nun um eine einfache Zeile erweitert:

```
AfxMessageBox("Hello World!");
```

Daraufhin kann das Programm erneut übersetzt und gestartet werden – ein Klick auf den "Klick!"-Button öffnet ab sofort eine weitere Dialogbox mit dem Text "Hello World!".

Qn: Mit diesem Beitrag haben Sie einen Schnelleinstieg in die Windows-Programmierung mit VC++ erhalten. Das Programmieren von Windows-Programmen ist allerdings aufwendig. Mit VC++ haben Sie jedoch ein ungemein leistungsfähiges Werkzeug an der Hand, um auch aufwendigste Probleme zu lösen. Lassen Sie sich von anfänglichen Schwierigkeiten nicht abhalten, das nächste fertige Programm wird Sie für Ihre Mühen entlohnen. v UR



DAS ERSTE EIGENE Windows-Programm.