Erfahrungen eines Visual Basic Anfängers

mit Visual-Basic Professional

Stefan Sedlitz

Jetzt ist es soweit, auch ich werde Mitglied der Visual-Basic-Gemeinde. Damit kann ich (zumindest laut Microsoft) am schnellsten Anwendungen für Windows erstellen, die alle Möglichkeiten der grafischen Benutzeroberfläche nutzen. Ich werde produktiver arbeiten als bisher, da mir endlich die geeigneten Hilfsmittel zum Erstellen einer grafischen Benutzeroberfläche zur Verfügung stehen, ...

Erstkontakt

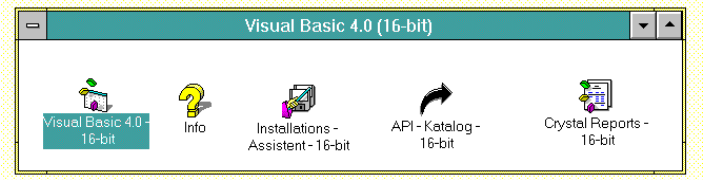
Da ich noch das altmodische Windows 3.1 verwende, benötige ich die 16-Bit Version von Visual Basic. Diese ist in der Standard Edition nicht enthalten, erst die Professional Edition bietet Abhilfe und enthält sowohl die 16-Bit Version als auch die 32-Bit Version.

Das Paket der Professional Edition ist beeindruckend gewichtig. Es enthält zunächst einmal Manuals mit insgesamt ca. 3000 Seiten. Leicht verunsichert (brauche ich das wirklich alles ?) entdecke ich dann noch eine CD-ROM mit Visual Basic und als Draufgabe noch eine CD-ROM von Firma Micro Basic, die laut Aufdruck Tools enthält.

Installation

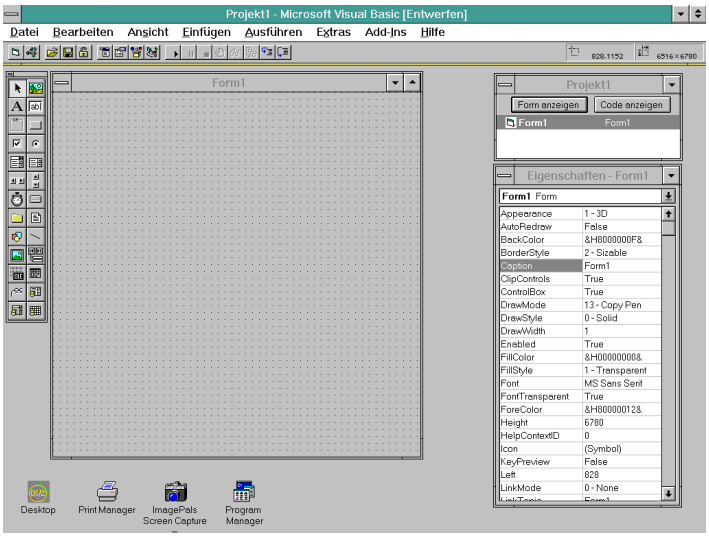
Ein Kompliment an Microsoft, die Installation erfolgt absolut problemlos. Das Install-Programm erkennt, daß nur eine 16-Bit Umgebung verfügbar ist, und bietet auch nur die Installation der 16-Bit Version von Visual Basic an. Dadurch werden auch nur ca. 23 MB meiner Festplatte benötigt (und nicht die für das Gesamtpaket angedrohten 70 MB).

Nach Bestätigung läuft die Installation automatisch ohne weiter Eingriffe ab, und schon bald erscheint eine neue Programmgruppe:



Ich habe es also geschafft. Was nun ? Wie schaut Visual Basic aus ?

Als typischer Anwender verzichte ich fürs erste einmal auf Manuals und klicke auf das neue schöne bunte Icon:



Wie am SreenShot ersichtlich, läuft Visual Basic (übrigens ganz problemlos) bei mir in der Windows Umgebung von OS/2 Warp.

Ganz links ist die Werkzeugsammlung mit den Steuerelementen, dann kommt eine große freie Fläche, auf der die Benutzeroberfläche konstruiert werden kann, ganz rechts sind dann die Windows, um den Steuerelementen auf der Oberfläche bestimmte Eigenschaften zuordnen zu können.

Ereignisgesteuerte Programmierung

Objekte in Visual Basic (Formen oder Steuerelemente) erkennen das Eintreten eines Ereignisses. Wenn zu einem bestimmten Objekt (z.B: Schaltfläche Command1) und einem bestimmten Ereignis (z.B.: Mausklick Click) in der entsprechenden Ereignisprozedur Command1_Click Code existiert, wird dieser automatisch bei Anklicken von Command1 ausgeführt. Visual Basic bietet zu den Objekten und den dazu möglichen Ereignissen bereits vordefinierte Ereignisprozeduren an (mit richtigem Namen und Header). Wenn diese Prozedur benötigt wird, muß sie nurmehr mit dem entsprechenden Code gefüllt werden.

Zum besseren Verständnis ein paar Standardereignisse:

`Change`	Zeigt an, daß sich der Inhalt eines Steuerelements verändert hat.
`Click`	Tritt ein, wenn der Benutzer die Maustaste über einem Steuerelement betätigt (drücken und loslassen).
`DblClick`	Tritt ein, wenn der Benutzer über einem Steuerelement zweimal nacheinander die Maustaste drückt und wieder losläßt.
`DragDrop`	Nach Beendigung einer Drag&Drop Operation, nachdem ein Steuerelement über eine Form oder ein Steuerelement gezogen wurde.
`DragOver`	Tritt während einer Drag&Drop Operation ein, Damit kann angezeigt werden, ob der Mauszeiger über ein zulässiges Ziel gezogen wird, ein solches verläßt oder sich direkt darunter befindet.
`GotFocus`	Tritt ein, wenn ein Objekt den Fokus bekommt (durch Benutzeraktion wie Anklicken, oder wenn durch Code zugewiesen).
`KeyPress`	Tritt ein, wenn der Benutzer eine Taste drückt und wieder losläßt.
`MouseDown`	Tritt ein, wenn der Benutzer eine Maustaste drückt.
`MouseMove`	Tritt ein, wenn der Benutzer die Maus bewegt.
`MouseUp`	ritt ein, wenn der Benutzer eine Maustaste losläßt.

Erste Versuche

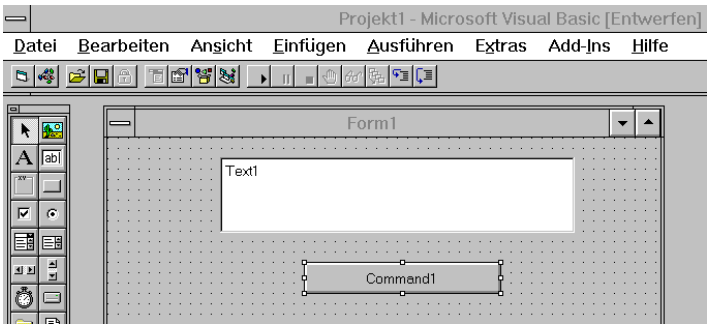
Für den ersten Versuch nehme ich das einfachste Beispiel aus dem Programmierhandbuch.

Es wird in 3 Schritten durchgeführt:

1. Erstellen der Benutzeroberfläche
2. Festlegen der Eigenschaften
3. Schreiben von Code

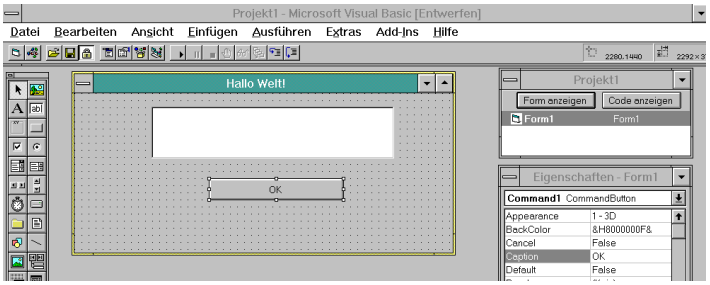
Ich beginne mit der Benutzeroberfläche:

Diese soll aus einem Textfeld und einer Befehlsschaltfläche bestehen. Dazu klickt man auf das entsprechende Werzeug und zieht einfach (wie bei einem Malprogramm) auf der neuen Oberfläche das Objekt zur gewünschen Größe. Die Objekte können jetzt beliebig verschoben, vergrößert oder verkleinert werden. Wenn die Oberfläche das gewünschte Aussehen hat, können die Positionen der Steuerelemente gesperrt werden.



Meine Oberfläche „Form1“ besteht nun aus dem Textfeld „Text1“ und der Befehlsschaltfläche „Command1“. Jetzt werden die Eigenschaften und Werte zugeordnet:

Objekt	Eigenschaft	Wert
`Form1`	Caption	Hallo Welt !
`Text1`	Text	(leer)
`Command1`	Caption	OK



Als dritter Schritt fehlt nur noch der Code, der bei Anklicken des OK-Buttons den Text „Hallo Welt!“ im Textfenster ausgibt.

Dazu wird für Command1 das Codefenster geöffnet. Von den angebotenen Prozeduren selektiere ich „Click“. Diese Prozedur wird, wie der Name schon sagt, bei einem Mausklick auf Command1 aktiviert. In die vorgeleistete Prozedurschablone füge ich nurmehr die Zuweisung

```
Text1.Text = ‘Hallo Welt !’
```

ein.

Damit ist das erste Beispiel fertig. Nach dem Starten erscheint ein leeres Textfenster. Nach einem Mausklick auf OK erscheint *„Hallo Welt!“*. Faszinierend.

Weitere Versuche

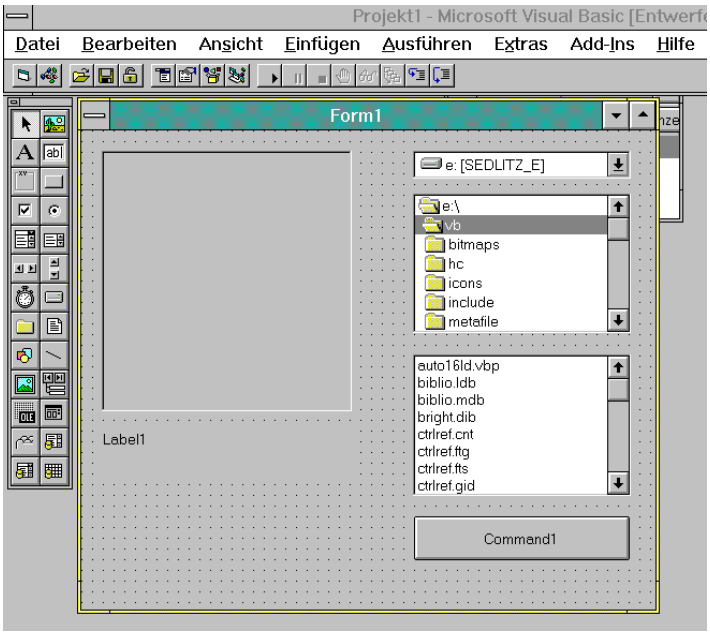
oder

Was bietet Visual Basic noch ?

Funktionen, die bei vielen Anwendeungen benötigt werden, werden bereits durch vordefinierte Steuerelemente unterstützt. Mit wenigen Zeilen Code sind zum Beispiel Datei-Selektions-Funktionen möglich.

Als Beispiel soll ein einfacher Bildbetrachter (für `*.BMP`, `*.WMF` und `*.ICO` Formate) gemacht werden.

Für die Oberfläche werden ein „*Anzeigesteuerelement*“ (in dem das Bild angezeigt werden soll), ein „*Bezeichnungfeld*“ (in dem der Dateiname angezeigt werden soll) und eine „*Befehlsschaltfläche*“ zum Beenden des Programms benötigt. Für die Dateiselektion werden die Steuerelemente „*Laufwerk*“, „*Verzeichnis*“ und „*Datei*“ benötigt. Die damit entworfene Oberfläche hat folgendes Aussehen:



Nach dem Zuordnen von Einstellungen und Werten zu den Eigenschaften der Objekte (z.B. Beschriftung des „*Ende*“-Buttons, Rahmen der Bildfläche, für das Dateilistenfeld werden die angezeigten Dateien mit dem Muster `*.BMP`, `*.WMF`, `*.ICO` auf die von Visual Basic unterstützten Formate eingeschränkt, ...) brauchen nur mehr wenige Zeilen Code geschrieben werden.

```
Private Sub Form_Load()
   Drive1.Drive = App.Path
   Dir1.Path = App.Path
End Sub
```

Damit werden beim Laden der Anwendung Laufwerk und Verzeichnis mit dem aktuellen Pfad initialisiert.

```
Private Sub Command1_Click()
    Unload Me
    End     ' Anwendung beenden
.
End Sub
```

Beenden der Anwendung bei Klick auf den Command Button

```
Private Sub Drive1_Change()
   Dir1.Path = Drive1.Drive
End Sub
```

Bei Änderung des Laufwerks wird Verzeichnisvariable aktualisiert.

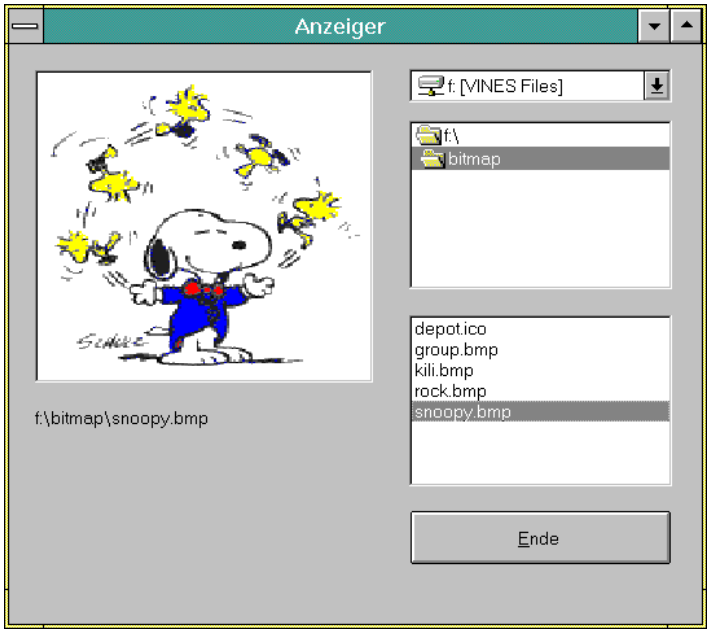
```
Private Sub Dir1_Change()
   File1.Path = Dir1.Path
End Sub
```

Bei Verzeichnisänderung wird Pfadvariable für Dateiliste aktualisiert.

```
Private Sub File1_DblClick()
   If Right(File1.Path, 1) <> "\" Then
      Label1.Caption = File1.Path & "\" & File1.filename
   Else
      Label1.Caption = File1.Path & File1.filename
   End If
   Form1.Image1.Picture = LoadPicture(Label1.Caption)
End Sub
```

Bei Doppelklick auf den Dateinamen wird zuerst dem „Bezeichnungfeld“ (in dem der Dateiname angezeigt werden soll) der vollständige Dateiname zugewiesen (Check, ob Hauptverzeichnis: wenn der Pfad das Hauptverzeichnis ist, wird kein \ zugesetzt). Dann wird mit der `LoadPicture` Funktion die selektierte Datei ins „*Anzeigesteuerelement*“ geladen (und damit angezeigt).

Das Endresultat



Weitere Features von Visual Basic

Die weiteren Features von Visual Basic möchte ich nur kurz vorstellen, damit zumindest ein Eindruck von der Mächtigkeit entsteht. Eine ausführliche Beschreibung bzw. Beispiele würden den Rahmen dieses Artikels (erster Einstieg in Visual Basic) sprengen.

Sprachmittel

Visual Basic bietet die üblichen Kontrollstrukturen einer Programmiersprache (If Then Else, Select Case, Schleifen, etc), Variablen, Prozeduren. Dazu kommen noch objektorientierte Konstrukte wie Objekte, Methoden und Klassen.

Menü-Steuerelement

Das Erstellen von Menüs (Menü-Steuerelement) wird durch einen eigenen Menü-Editor unterstützt. Es ist damit genauso einfach zu entwerfen und mit Eigenschaften und Funktionen zu versehen, wie die am Anfang beschrieben Oberflächen.

Tabelle-Steuerelement

Damit wird das Anzeigen von Informationen in Zeilen und Spalten erleichtert. Mit dem Tabelle-Steuerelement können Zeilen und Spalten in der Größe verändert werden, Grafiken verwendet werden, Zellen ausgewählt und markiert werden und Zeilen hinzugefügt und gelöscht werden.

MDI-Anwendungen

MDI heißt Multiple-Document Interface und bedeutet, daß eine Anwendung mehrere Formen innerhalb einer umgebenden Form (Container-Form) unterstützt. MDI wird zum Beispiel von Excel oder Word for Windows unterstützt. In einer MDI-Anwendung kann der Benutzer gleichzeitig mehrere Dokumente anzeigen, jedes Dokument in einem eigenen Fenster. Diese Fenster sind Teil eines übergeordneten Fensters, das den Arbeitsbereich zur Verfügung stellt.

Grafik

Grafik wird durch grafische Steuerelemente und durch Grafikmethoden unterstützt. Eine Vielzahl von Funktionen (Größenänderungen, Verschieben, Ändern der Form, Farbunterstützung, Grafikformate BMP, ICO und WMF, automatisches Skalieren von Bildern, Standardfiguren wie Linie, Kreis, etc.) steht zur Verfügung.

DDE und OLE

Visual Basic unterstützt DDE (Dynamischer Datenaustausch) und OLE (Object Linking and Embedding).

Damit werden der Datenaustausch mit anderen DDE-Quellen und das Verknüpfen oder Einbetten von Objekten ermöglicht. Daten von anderen Anwendungen (am häufigsten wahrscheinlich von MS Excel, MS Word für Windows, MS Project, MS Access und MS FoxPro für Windows) können so komfortabel in die eigene Visual Basic Anwendung übernommen und verarbeitet werden.

Datenbank Zugriffe

Mit dem Daten-Steuerelement können Anwendungen bestehende Datenbanken (wie MS Access, dBase, MX FoxPro, Paradox etc.) verarbeiten (zugreifen, bearbeiten, aktualisieren). Zusätzlich kann auch auf MS Excel-, Lotus 1-2-3- und ASCII-Text-Dateien zugegriffen werden. Es sind ebenso Zugriffe und Veränderungen in ODBC-Datenbanken (Open DataBase Connectivity) im Netzwerk (wie bei MS SQL Server oder Oracle) möglich.

Der Datenzugriff erfolgt mit dem MS Jet-Datenbankmodul (wird auch in MS Access eingesetzt).

Client/Server Anwendungen

Wie schon bei den Datenbank-Zugriffen erwähnt, können mit Visual Basic nach Installation des ODBC-Treibers Client/Server-Anwendungen erstellt werden (mit SQL-Abfragen, Transaktionen, Mehrbenutzerbetrieb, Rollback, etc.).

Installationsassistent

Für die Weitergabe oder Vertrieb einer fertigen Visual Basic Anwendung kann mit dem Installationsassistenten ein entsprechendes Installationsprogramm erstellt werden. Ebenso wird das Erstellen der fertigen Auslieferungsdisketten unterstützt.

Zusatzsteuerelemente

Zusatzsteuerelemente sind Erweiterungen der Werkzeugsammlung von Visual Basic. Diese stehen in einer speziellen Form einer DLL (Dynamic Link Library) zur Verfügung. Es können sowohl selbstentwickelte Zusatzsteuerelemente, als auch beliebige Zusatzsteuerelemente von Drittanbietern verwendet werden. Die wichtigsten der von Visual Basic Professional Edition angebotenen Zusatzsteuerelemente sind:

- 3-D Elemente: Für Schaltflächen, Optionsfelder, Kontrollfelder etc. gibt es zu den Standardversionen noch Versionen im 3-D Look.
- Animierte Schaltfläche: Zeigt eine Folge von Bitmaps an, die einen Animationseffekt erzeugen können (wenn der User auf das Steuerelement klickt).
- Kommunikation: Unterstützt die Kommunikation und Senden/Empfangen von Daten über die serielle Schnittstelle.
- MAPI-Sitzung/Nachricht: Für Anwendungen, die E-Mail unterstützen und MAPI-Funktionen verwenden.
- Multimedia-MCI: Verwaltet das Aufzeichnen und Abspielen von Mulimedia-Dateien auf MCI-Geräten (Media Control Interface).
- Hierarchie (TreeView): Kann eine Liste von Elementen hierarchisch darstellen. Das heißt, untergeordnete Elemente können geöffnet (ausgeklappt) oder geschlossen (eingeklappt) werden.
- Fortschrittleiste: Damit kann der Fortschritt einer längeren Aktion grafisch angezeigt werden. Ein Rechteck wird während der Operation von links nach rechts mit Blöcken gefüllt.
- RTF-Box: Unterstützt das Eingeben und Bearbeiten von Texten nicht nur im Standard-ASCII-format, sondern auch im Rich Text Format.
- Schieberegler: Mit dem Schieberegler kann man per Maus einen Wert auf der Skala des Schiebereglers auswählen.
- Symbolleiste (Toolbar): Damit wird das Erstellen eigener Symbolleisten (mit Button-Objekten) unterstützt. □