Einführung in Visual Basic 6.0

---

1. Allgemeines

Visual Basic ist eine relativ neue Programmiersprache. Sie wurde von Microsoft entwickelt, um Programmierern, und zwar sowohl Anfängern wie auch Profis, ein Werkzeug in die Hand zu geben, mit dem sie schnell und ohne großen Aufwand Programme mit ansprechenden **Benutzeroberflächen** erstellen können. In dem Namen **Visual Basic** steckt dabei zweierlei:

Visual: Optisch orientiert. Aus fertigen Objekten aufgebaut, die sich beliebig aneinanderfügen lassen.

Basic: Nicht alles lässt sich mit den vorgegebenen Objekten allein realisieren. Der dahinter stehende Code ist in der Programmiersprache Basic zu schreiben, die auch heute noch ein Standard ist.

Möglichkeiten und Grenzen von Visual Basic:

Mit Visual Basic lassen sich schnell und mit wenig Programmieraufwand kleine und mittelgroße Programme schreiben, die trotzdem perfekt gestylte Benutzeroberflächen besitzen. Visual Basic ist auch für jeden Access-Anwender ein Muss, da dort dieselben Mechanismen verwendet werden. Und auch in den anderen Office-Produkten (Word, Excel ...) findet man dieses Prinzip (VBA - Visual Basic for Applications).

Die dahinter stehende Programmiersprache Basic ist indes nicht so leistungsfähig wie die „großen" Programmiersprachen. Und sobald man komplexere Datentypen wie Records oder gar Zeiger verwenden möchte, sollte man sich besser auf die Suche nach einem anderen Entwicklungswerkzeug machen. Für die Windows-Programmierung bieten sich in diesem Zusammenhang Delphi, Borland C++ Builder oder MS Visual C++ an. Diese Systeme besitzen jedoch nicht die Einfachheit von Visual Basic und erfordern vom Programmierer mehr Vorkenntnisse.

Eine andere verbreitete Variante besteht darin, nur die **Benutzeroberfläche** in Visual Basic zu erstellen und die dahinter liegenden **Prozeduren** (Berechnungen, Steuerungen, usw.) in einer anderen Sprache (z.B. C++) zu schreiben. Visual Basic ist dabei keine Programmiersprache im herkömmlichen Sinn. Ein VB-Programm ist eine Ansammlung verschiedener Objekte, für die jeweils ein eigener Code geschrieben wird. Entsprechend „zerfahren" wirken VB-Programme manchmal. Gleichwohl ist VB eine Sprache, die dem Programmierer wenige Reglementierungen auferlegt.

2. Visual Basic für Windows

Microsoft Windows verlangte aufgrund seiner graphischen Oberfläche ab Version 3.0 nach Programmiersystemen, die die Erstellung entsprechender Programme ermöglichten. Eines dieser Systeme ist **Visual Basic (VB).**

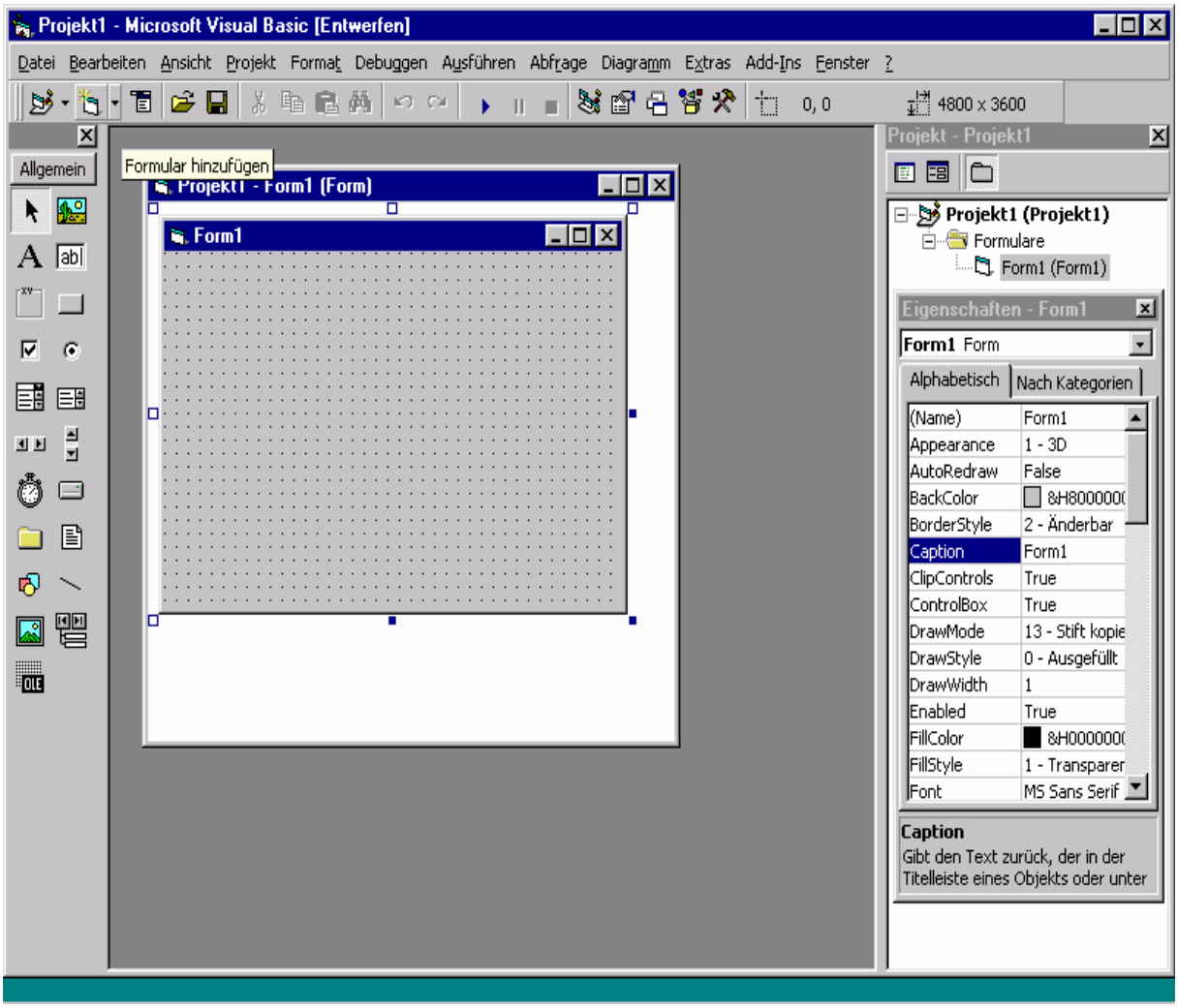
Ereignissteuerung bedeutet hier, dass der Ablauf eines Programms nicht mehr nur vom vorher festgelegten Programmcode abhängt, sondern ganz wesentlich auch von den Aktionen, die der Benutzer zur Laufzeit des Programms setzt.

Windows-Messages stehen hier in engem Zusammenhang mit der Ereignissteuerung. Windows überwacht ständig alle Tätigkeiten an Tastatur und Maus und meldet diese Ereignisse weiter (also z.B. das Drücken oder Loslassen einer Taste, eine Mausbewegung, das Drücken oder Loslassen einer Maustaste usw.).

Diese "Botschaften" werden den Programmen übermittelt und können von diesen ausgewertet werden. Der Programmierer muss nur noch festlegen, was passieren soll, wenn beispielsweise die linke Maustaste über einem bestimmten Befehls-Icon gedrückt wird. des Befehlsknopfes oder überhaupt das Feststellen, welches Fenster im Vordergrund liegt, erledigt Windows für ihn.

3. Der Aufbau von VB

Visual Basic ist ein Windows-Programm mit folgender graphischer Arbeitsoberfläche bzw. -umgebung:



Das zentrale Element ist das **Formular** (Bild oben Mitte), eine Art Vorlage für das spätere Windows-Fenster, in dem sich das Programm abspielt. Es enthält so genannte **Steuerelemente,** die der so genannten „Toolbox“ (Bild oben links) entnommen und in die Entwurfansicht des Formulars bzw. Fensters eingebaut werden können.

Ein VB-Programm kann dabei aus mehreren Formularen oder Modulen (Basic-Codes) bestehen. Die Verwaltung all dieser Teile erfolgt über das **Projekt-Fenster**.

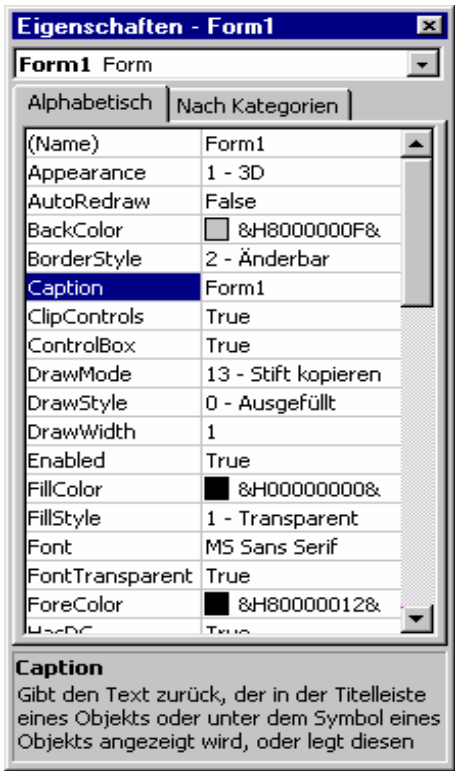
Wie fast jedes Windows-Programm besitzt auch Visual Basic die bekannte Microsoft-Menüleiste, die den Zugang zu zahlreichen Optionen ermöglicht. Die Verwaltung der Tools erfolgt über die so genannte **Eigenschaften** im Eigenschaftsfenster (Bild oben rechts). Nachfolgend die der Toolbox Element entnehmbaren Steuerelemente:

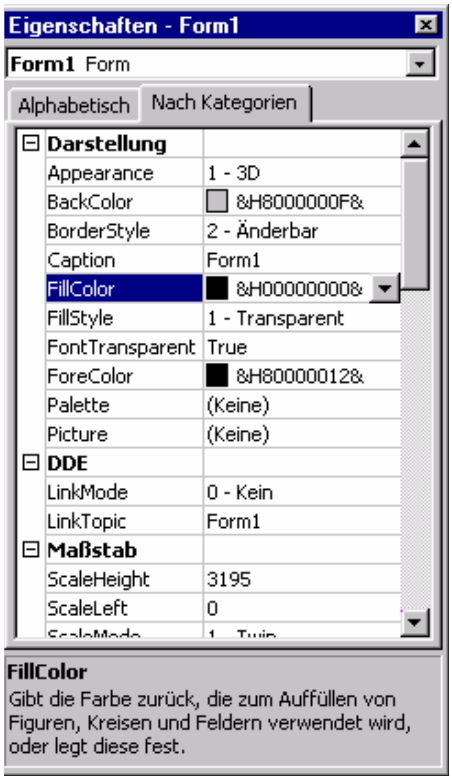
Zeiger		Bildfeld (Picture Box) [pic]
Bezeichnungsfeld (Label) [lbl]		Textfeld (Text Box) [txt]
Rahmen (Frame) [fra]		Befehlsschaltfläche (Command Button) [cmd]
Kontrollkästchen (Check Box) [chk]		Optionsfeld (Option Button) [opt]
Kombinationsfeld (Combo Box) [cbo]		Listenfeld (List Box) [lst]
Hor. Schieberegler (Hor. Scroll Bar) [hsb]		Vertik. Schieberegler (Vertical Scroll Bar) [vsb]
Zeitgeber (Timer) [tim]		Laufwerksfeld (Drive List Box) [drv]
Verzeichnisliste (Directory List Box) [dir]		Dateiliste (File List box) [fil]
Figur (Shape) [shp]		Linie (Line) [lin]
Anzeigefeld (Image) [img]		Datenbank (Data) [dat]
OLE-Steuerelement (OLE) [ole]		Dialog (Common Dialog) [dia]
Tabellen-Dialog (SSTab) [sst]		Datenbank-Liste (DB List) [dbl]
DB-Kombifeld (DB Combo Box) [dbc]		Datenbank-Tabelle (DB Grid) [dbg]

4. Steuerelemente (Teil 1)

Achtung!
Alle Steuerelemente und Formulare erhalten nach ihrer Auswahl SOFORT den „richtigen“ **Namen**. Es ist sinnvoll, den Namen so zu wählen, dass daraus die Art des Elementes abgeleitet werden kann. Allen selbst gewählten Namen wird eine so genannte Namenskonvention vorangestellt. Beispiel: Ein Fenster vom Typ „Label“, das für die Ausgabe eines kurzen Textes bestimmt ist, könnte *lblAusgabe* heißen. „lbl“ ist die **Namenskonvention** für „Label“ (s. Steuerelemente).

Als erster muss dem neu geöffneten Formular ein passender Namen gegeben werden. Das Formular wird angeklickt, um es zu aktivieren. Im Eigenschaftenfenster bieten sich u.a. zwei Felder an, die dazu dienen, die Eigenschaften des Formulars "Form1" zu bestimmen:





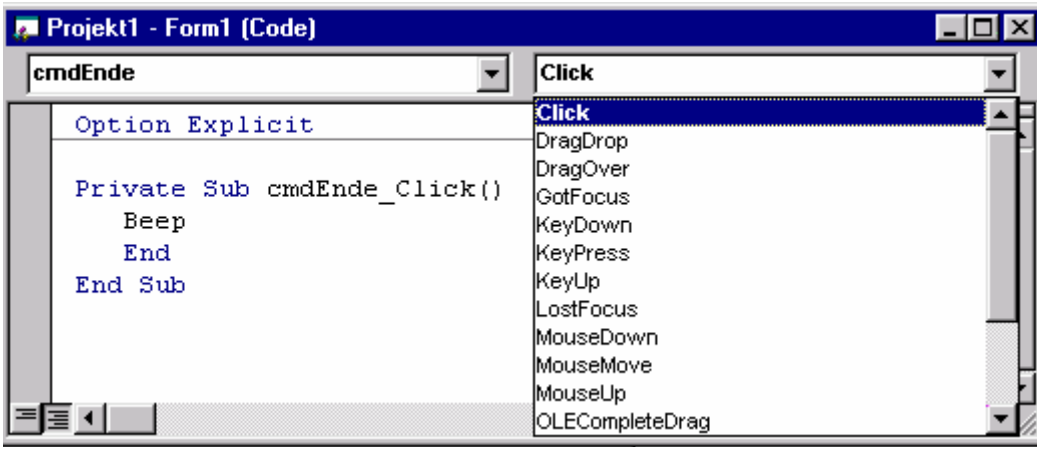
Je nach Bedarf können noch andere Eigenschaften des Formulars geändert werden (z.B. die Hintergrundfarbe, Größe usw.).

Danach kann z.B. ein Bezeichnungsfeld (Label) auf dem Formular platziert werden, das einen bestimmten Text enthält. Auch dieses Feld erhält sofort den „richtigen“ Namen und die anderen gewünschten Eigenschaften (Caption, Font, ...).

Anschließend wird noch einen Befehlsknopf (command button) zum Beenden des Programms auf dem Formular platziert und mit dem richtigen Namen (cmdEnde) und der richtigen Caption (z.B. „Beenden“) versehen.

Durch Doppelklick auf den Beenden-Button wird das Code-Fenster geöffnet, in dem die gewünschten Befehle zwischen *Private Sub cmdEnde_Click()* und *End Sub* einzutragen sind: z.B. „Beep“ und „End“. Die auf diese Weise programmierte Prozedur heißt **Ereignisprozedur**, weil sie in einer ganz bestimmten Situation, nämlich beim Anklicken der Befehlsschaltfläche, ausgeführt wird. Im Code selbst wird dies mit der Formel *cmdEnde_Click()* notiert, den das Programm automatisch generiert, sofern der Name des Steuerelementes wie in diesem Fall *cmdEnde* betitelt wurde.

Der entsprechende Programmcode, der die Aktion "Beenden" in Verbindung mit einem zu hörenden Piepton ausführt, sieht aus wie folgt:



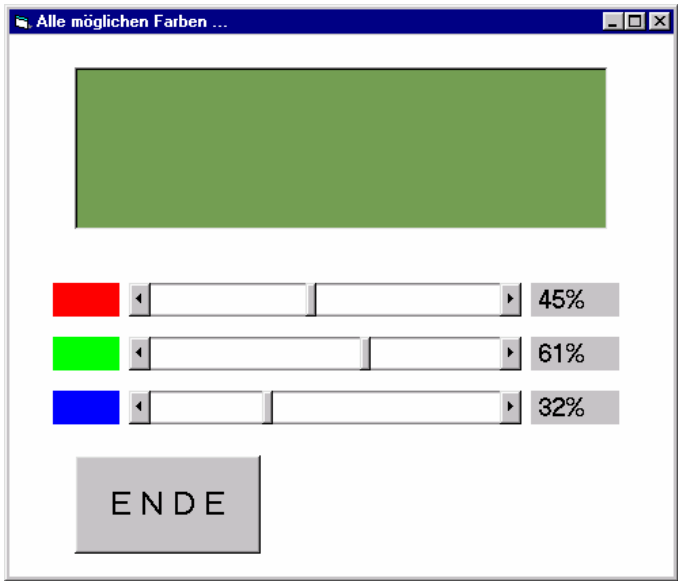
5. Prozeduren, Funktionen und Module

Um mit Hilfe von Steuerelementen irgendwelche Veränderungen bewirken zu können, muss für die gewünschten Aktionen ein ereignisabhängiger **Programmcode** formuliert werden. Dies kann z.B. für folgende Steuerelemente der Toolbox geschehen:

Steuerelement	Aktion	Anmerkung
Befehlsschaltfläche (cmd)	Click	einmaliges Anklicken
Optionsknopf (opt)	Click	einmaliges Anklicken
Vert. Schieberegler (vsb)	Change	Verändern
	Scroll	Bewegen

Aufgabe 1
Zu erstellen ist ein Programm, das mit Hilfe von Schiebereglern die stufenlose Mischung der drei Grundfarben Rot, Grün und Blau ermöglicht.

Bei jeder Bewegung eines der drei Schieberegler (entweder mit den Pfeiltasten am Ende oder mit dem Schiebebalken) soll der prozentuale Anteil einer jeden Farbe stets neu ermittelt und im großen Farbfeld entsprechend mit den anderen beiden Farben vermischt werden:



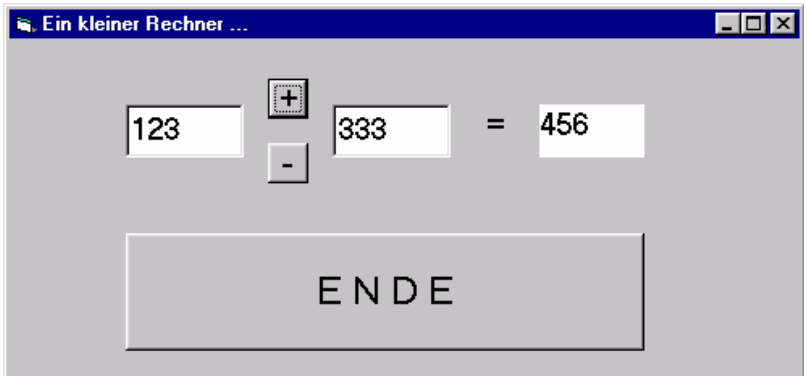
VB stellt für die stufenlose Farbmischung folgende Funktion zur Verfügung: rgb (rot, grün, blau) mit den Parameterwerten von 0 - 255. Für die Schieberegler ist jeweils das Ereignis Change und das Ereignis Scroll abzufangen. Es ist sinnvoll eine Ereignisprozedur (Public Sub) zu schreiben, die die Regler ausliest, die %-Werte schreibt und die Farben entsprechend mischt. Hier ein Link zur RGB-Farbtabelle: http://gucky.uni-muenster.de/cgi-bin/rgbtab

Eine solche **Ereignisprozedur** stellt der VB-Programmcode dar. Man aktiviert das Code-Fenster und kann dann mit <Extras> <Prozedur hinzufügen> ein neues Unterprogramm initialisieren. Der entsprechende Programmcode lautet (**auskommentiert** wird mit einem pro Zeile vorangestellten Apostroph):

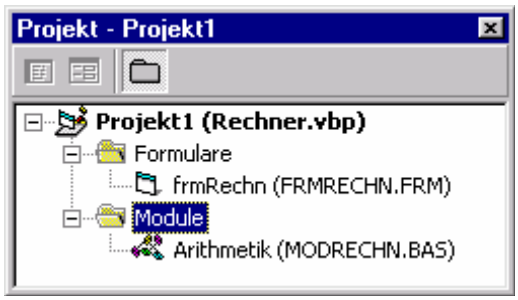
```
Public Sub NeueFarbe()

            ' Dieses Unterprogramm liest die 3 Schieberegler aus,
            ' aktualisiert die Wert-Anzeigen der einzelnen Regler
            ' und mischt die Farben neu.
            ' Hier werden die %-Angaben neu gesetzt:

   lblValRot.Caption = Str(Int(hsbRot.Value / 255 * 100)) + "%"
   lblValGrün.Caption = Str(Int(hsbGrün.Value / 255 * 100)) + "%"
   lblValBlau.Caption = Str(Int(hsbBlau.Value / 255 * 100)) + "%"

            ' Und hier wird die Farbe neu gemischt

   lblFarbe.BackColor = RGB(hsbRot.Value, hsbGrün.Value, hsbBlau.Value)

End Sub
```

Aufgabe 2

Zu erstellen ist ein Programm, das die Addition oder Subtraktion zweier Zahlen ermöglicht. Dies kann in der bisherigen Art und Weise erfolgen, indem man für die Schaltflächen „+“ und „-* eigene Ereignisprozeduren festlegt.

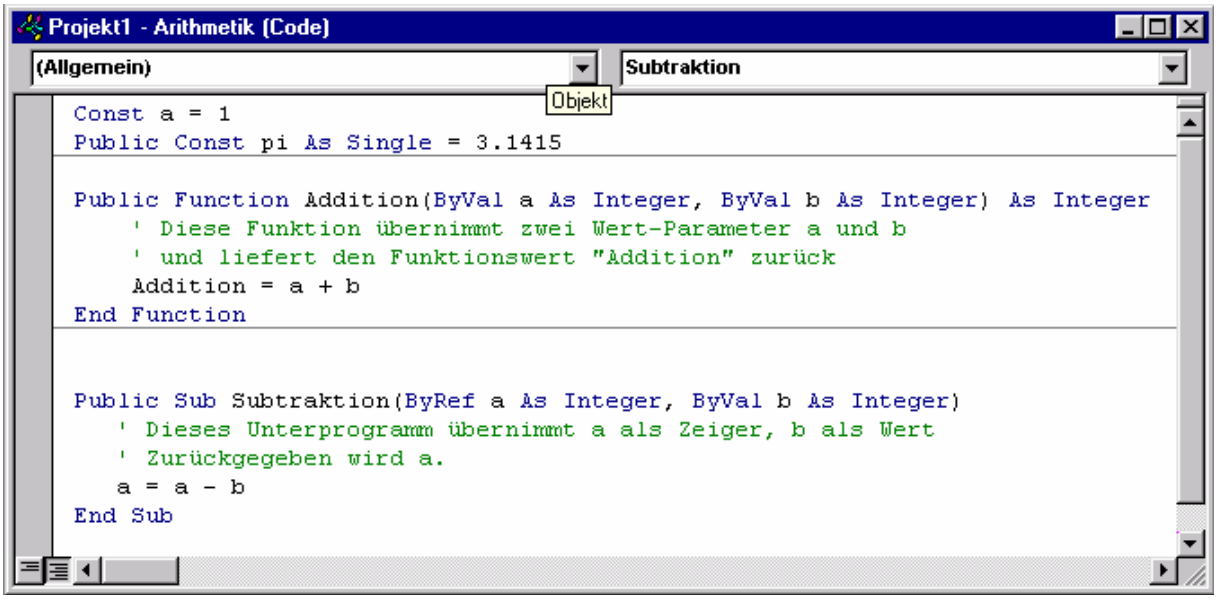


Ein anderer Weg sind die so genannten **Module**.



Module enthalten einen Code und werden getrennt von den Formularen gespeichert. Sie können auch in andere Projekte eingebunden werden. Wie auch in anderen Programmiersprachen gibt es in Visual Basic die Unterscheidung zwischen Funktionen (die einen Funktionswert zurückliefern) und Prozeduren (in VB: Sub), die keinen Funktionswert an das aufrufende Programm zurückgeben. Trotzdem kann natürlich durch Parameterübergabe ByRef (d.h. als Zeiger) statt ByVal (als expliziter Wert) eine beliebige Anzahl von Werten auch aus einer Prozedur übernommen werden.

Beispiel 1: Hier erhält die Funktion „Addition“ vom aufrufenden Programmteil zwei Werte übergeben (By Value - ByVal). Das Ergebnis wird zunächst als Variable „Addition“ berechnet und bei der Beendigung der Funktion an den aufrufenden Programmteil übergeben. Der Aufruf einer Funktion erfolgt mit ihrem Namen, z.B.: Ergebnis = Addition (x,y)

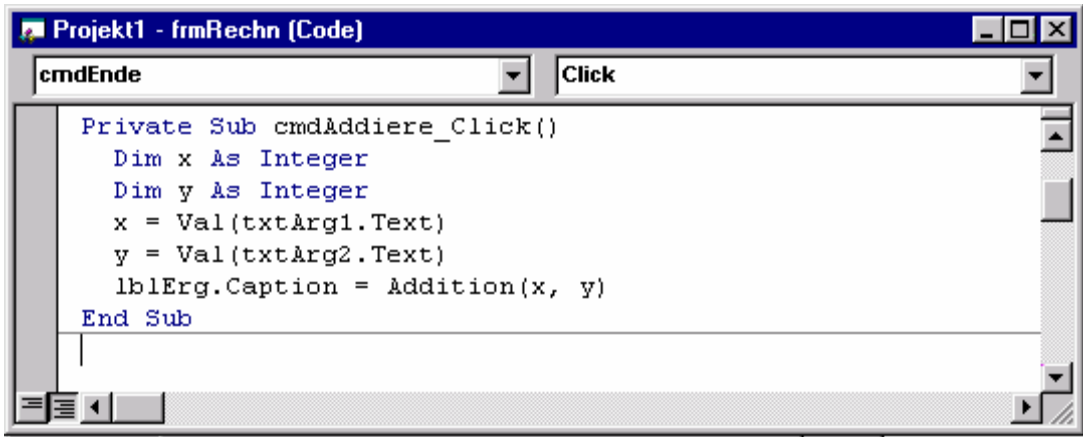


Module werden häufig auch verwendet, um bestimmte, immer wieder auftretende Konstante zu deklarieren. Dies erfolgt im allgemeinen Teil des Moduls unter Deklarationen und sinnvollerweise als „public“. Beispiel:

```
Public Const Pi as single = 3.1415
```

Die Angabe des Datentyps ist nicht notwendig, aber sinnvoll. Wenn nicht explizit „Public“ angegeben wurde, gelten Konstante als „Private“ für den entsprechenden Programmteil (Funktion, Modul, ...).

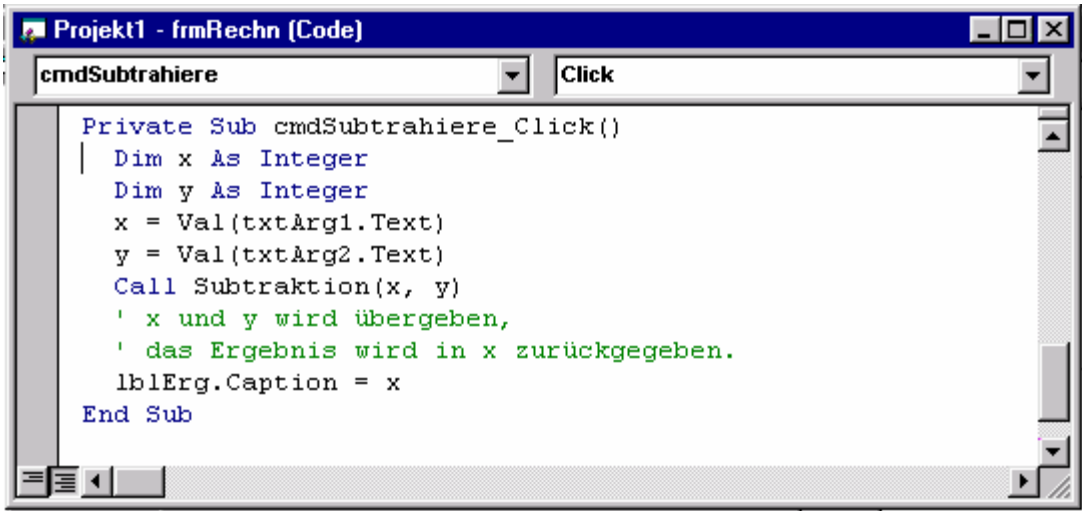
Wie im unteren Beispiel, dem Funktionsaufruf in der Ereignisprozedur für den Additions-Knopf, ersichtlich, kann der Funktionswert aber auch gleich einer Steuerelement-Eigenschaft zugewiesen werden:



In diesem Beispiel werden zwei Hilfsvariablen x und y deklariert, die die Lesbarkeit des Programms verbessern. Diese Variablen sind lokal für die Ereignisprozedur. Visual Basic stellt verschiedene Datentypen zur Verfügung.

Die gebräuchlichsten **Datentypen** sind Byte: 1 Byte, Boolean: 2 Byte, Short Integer: 2 Byte, Long Integer: 4 Byte, Single: 4 Byte, Double: 8 Byte, String/Variant: variabel Zahl oder Zeichen.

Beispiel 2: Hier handelt es sich um eine „echte“ Prozedur, d.h. es wird kein Funktionswert ermittelt und zurückgegeben. In diesem Fall erfolgt die Rückgabe des berechneten Wertes über die Variable a, die nicht als Wert-Parameter, sondern als Zeiger (d.h. By Reference - ByRef) übergeben wurde.



Der Aufruf erfolgt in der Ereignisprozedur für die Subtraktion. Beide Unterprogramme, also sowohl die Funktion für die Addition als auch die Prozedur für die Subtraktion, sind als „Public“ deklariert, d.h. sie stehen allen Programmteilen (insbesondere auch solchen außerhalb des eigenen Moduls) zur Verfügung. Im Gegensatz dazu ist die Gültigkeit von „Private“-Prozeduren auf die eigene Form bzw. das eigene Modul beschränkt.

6. Steuerstrukturen, Datentypen, Variable

Auch in Visual Basic gibt es die heute in allen Programmiersprachen in ähnlicher Form vorhandenen Steuerstrukturen. Die folgende Auflistung stellt die VB-spezifischen Eigenheiten dar:

a) Verzweigungen

Einfach-Verzweigung

```
If  <Bedingung> Then
        <Anweisung>
        <Anweisung>
Else
        <Anweisung>
        <Anweisung>
End If
```

Mehrfach-Verzweigung

```
Select Case <Ausdruck>
Case <Wert>
        <Anweisung>
        <Anweisung>
Case <Wert>
        <Anweisung>
        <Anweisung>
Case <Wert>
        <Anweisung>
        <Anweisung>
End Select
```

---

b) Wiederholungsanweisungen

Schleife mit Vorausabfrage (vorzeitiges Verlassen der Schleife möglich mit *Exit Do*)

```
<Initialisierung>
Do While <Bedingung>
        <Anweisung>
        <Anweisung>
.Loop
```

Schleife mit Endabfrage (oder: Loop While <Bedingung>)

```
<Initialisierung>
Do
        <Anweisung>
        <Anweisung>
Loop Until <Bedingung>
```

Zählschleife (vorzeitiges Verlassen der Schleife möglich mit **Exit For**)

```
For <Anfangswert> To <Endwert> Step <Schrittweite>
<Anweisung>
<Anweisung>
Next
```

Datentypen
Die auch für VB 6.0 bis hin zu Visual Basic 2010 Express gebräuchlichen Datentypen sind:

Datentyp	Speicherbedarf	Wertebereich
Byte	1 Byte	0 bis 255
Boolean	2 Bytes	**True** oder **False**
Integer	2 Bytes	-32.768 bis 32.767
Long (lange Ganzzahl)	4 Bytes	-2.147.483.648 bis 2.147.483.647
Single (Gleitkommazahl mit einfacher Genauigkeit)	4 Bytes	-3,402823E38 bis -1,401298E-45 für negative Werte; 1,401298E-45 bis 3,402823E38 für positive Werte.
Double (Gleitkommazahl mit doppelter Genauigkeit)	8 Bytes	-1,79769313486232E308 bis -4,94065645841247E-324 für negative Werte; 4,94065645841247E-324 bis 1,79769313486232E308 für positive Werte.
Currency (skalierte Ganzzahl)	8 Bytes	-922.337.203.685.477,5808 bis 922.337.203.685.477,5807
Decimal	14 Bytes	+/-79.228.162.514.264.337.593.543.950.335 ohne Dezimalzeichen; +/-7,9228162514264337593543950335 mit 28 Nachkommastellen; die kleinste Zahl ungleich Null ist +/-0,0000000000000000000000000001.
Date	8 Bytes	1. Januar 100 bis 31. Dezember 9999.
Object	4 Bytes	Beliebiger Verweis auf ein Objekt vom Typ **Object**.
String (variable Länge)	10 Bytes plus Zeichenfolgenlänge	0 bis ca. 2 Milliarden.
String (feste Länge)	Zeichenfolgenlänge	1 bis ca. 65.400
Variant (mit Zahlen)	16 Bytes	Numerische Werte im Bereich des Datentyps **Double**.
Variant (mit Zeichen)	22 Bytes plus Zeichenfolgenlänge	Wie bei **String** mit variabler Länge.
Benutzerdefiniert (mit **Type**)	Zahl ist von Elementen abhängig	Der Bereich für jedes Element entspricht dem Bereich des zugehörigen Datentyps.

7. Steuerelemente (Teil 2)

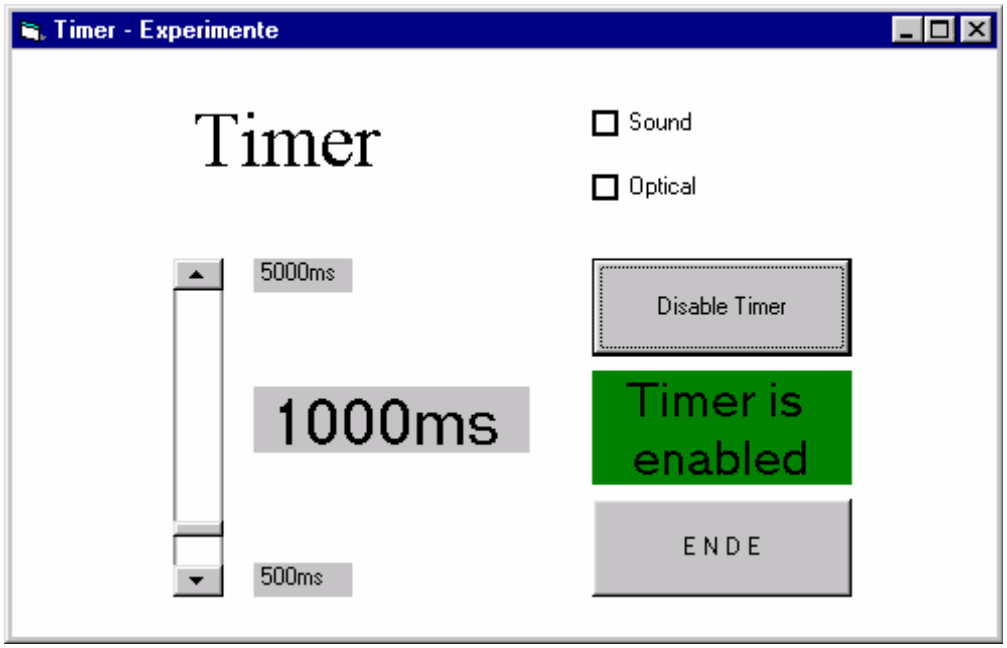
Bisher wurden folgende Steuerelemente verwendet:

- Formular (frm)
- Bezeichnungsfeld (lbl)
- Textfeld (txt)
- Schieberegler (vsb, hsb)
- Rahmen (fra)
- Optionsfeld (opt)
- Befehlsschaltfläche (cmd)

Die folgenden beiden Programme machen die Verwendung von 4 weiteren Steuerelementen deutlich:

Steuerelement	Aktion	Anmerkung
Timer (tim)	Interval	Zeitgeber
Laufwerksfeld (drv)		Laufwerk
Verzeichnisliste (dir)		Verzeichnis
Dateiliste (fil)		Dateien

Aufgabe 3:
Erstelle ein Programm, das in (einstellbaren) Abständen einen Signalton oder ein optisches Signal erzeugt. Das Steuerelement Timer besitzt relativ wenige Eigenschaften. Die wichtigsten davon sind Enabled (true/false) und Interval (zur Einstellung des Zeitintervalls in Millisekunden).
Wenn ein Timer "enabled" ist, löst er in zeitlichen Abständen, die über die Eigenschaft Intervall eingestellt werden, das Timer-Ereignis aus (übrigens das einzige Ereignis, für das ein Timer programmiert werden kann):



Mit dem Schieberegler sind die Timer-Intervalle einstellbar. Mit dem oberen Befehlsknopf ist der Timer ein- oder abschaltbar. Die beiden Kontrollkästchen rechts oben ermöglichen die (getrennte) Aktivierung von akustischen und optischen Signalen.

Aufgabe 4:
Erstelle ein Programm, das die Auswahl einer Graphik (z.B. Bitmap) über Laufwerk, Verzeichnis und Datei ermöglicht und die Graphik dann in einem Bildfeld darstellt. In diesem Programm werden die 3 neuen Steuerelemente zum Datei-Handling über Laufwerk, Verzeichnis und Dateiliste verwendet.

Das Laufwerksfeld (*drvLW*) liest die hardwaremäßig vorhandenen Laufwerke ein und ermöglicht mittels Pull-Down-Menü eine Auswahl. Diese Wahl steht in der Eigenschaft *drvLW.Drive* zur Verfügung (aber nur zur Laufzeit des Programms, nicht in der Entwurfsphase!).

Die Verzeichnisliste (*dirVerzeichnis*) übernimmt zunächst von Windows ein Standard-Laufwerk. Wenn man ein anderes Laufwerk haben möchte, muss man die Eigenschaft *dirVerzeichnis.Path* setzen (ebenfalls nur zur Laufzeit möglich): *dirVerzeichnis.Path = drvLW.Drive*

Wenn im Verzeichnislistenfeld ein anderes Verzeichnis gewählt wird, ändert sich die Path-Eigenschaft entsprechend. Die Dateiliste (*filDateien*) übernimmt zunächst von Windows ein Standard-Verzeichnis. Wenn man ein anderes Verzeichnis haben möchte, muss man die Eigenschaft *filDateien.Path* setzen (ebenfalls nur zur Laufzeit möglich): *filDateien.Path = dirVerzeichnis.Path*

Außerdem kann man für jedes Dateilistenfeld ein Suchmuster (Pattern) festlegen. Wenn die Eigenschaft *filDateien.Pattern* entsprechend gesetzt wird, werden nur die gewünschten Dateien aufgelistet.

Das oben aufgeführte Programm enthält auch noch ein Bildfeld (Picture Box), das dann zur Laufzeit das gewählte Bild darstellen soll (VB6 erkennt die gängigen Graphik-Formate). Das Laden eines Bildes ist offenbar etwas aufwendiger, sodass es nicht genügt, die Eigenschaft *Picture* einfach auf den gewünschten Dateinamen zu setzen (wie dies in der Entwurfsphase möglich ist), sondern man benötigt die Funktion *LoadPicture (<Pfad\Dateiname>)*, um das gewünschte Bild zu laden. VB als objektorientierte Programmiersprache kennt also:

a) **Eigenschaften,** die Objekte (Steuerelement) betreffen, von denen jedes eine Anzahl von Eigenschaften besitzt, die zur Entwurfszeit oder Laufzeit benutzt bzw, verändert werden können.
Beispiel: *lblAusgabe.Caption = ''Hallo''*

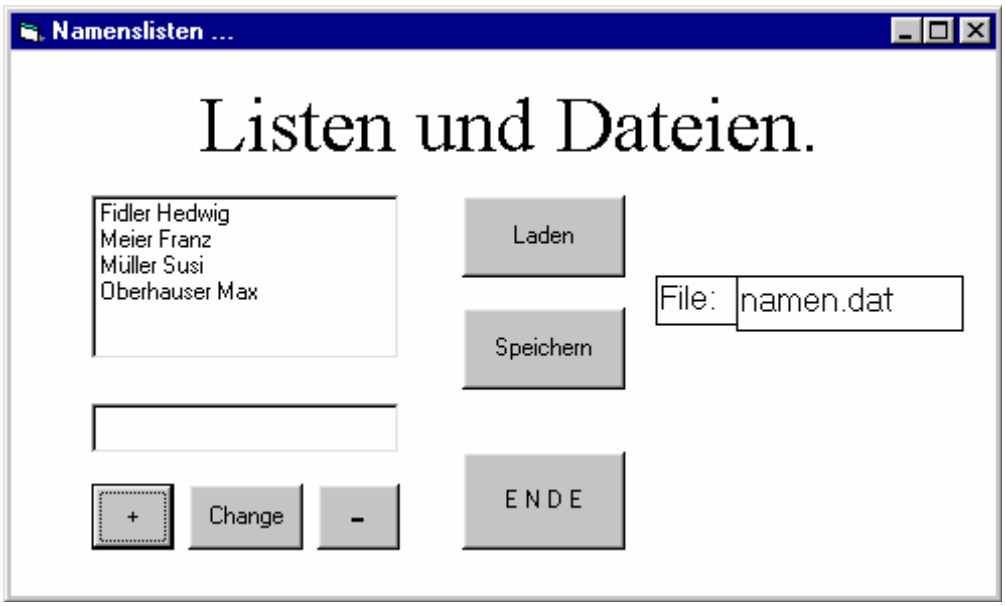
b) **Methoden** als vordefinierte Prozeduren, die etwas tun, aber keinen Wert zurückgeben.
Beispiel: *picBild.Print ''Hallo''*
Druckt den Text ''Hallo'' in das Bildfeld.

c) **Funktionen** sind ebenfalls vordefinierte Routinen, die aber einen Wert zurückliefern, also in einer Zuweisung stehen können.
Beispiel: *picBild.Picture = LoadPicture (lblDateiname.Caption)*

8. Steuerelemente (Teil 3)

An dieser Stelle geht es um die Verwaltung von Listen und um Möglichkeiten, Daten auf externe Datenträger zu speichern oder zu laden.

Aufgabe 5:
Erstelle ein Programm, das eine Liste (beispielsweise eine Namensliste) verwaltet. Das Hinzufügen, Löschen oder Ändern von Einträgen soll möglich sein. Außerdem soll die gesamte Liste auf einen externen Datenträger gespeichert werden können und auch wieder geladen werden können:



Zur Listenverwaltung stellt Visual Basic zwei Steuerelemente zur Verfügung:

- Listenfeld (List Box), [lst], und
- Kombinationsfeld (Combo Box), [cbo].

Die beiden Steuerelemente sind sich recht ähnlich. Das Kombinationsfeld stellt eine Kombination zwischen einem Textfeld und einem Listenfeld dar (Pull-Down).

---

9. Standard-Befehle

Für das Kombinationsfeld gelten sinngemäß dieselben Eigenschaften und Methoden. Zusätzlich steht die Eigenschaft Text (wie bei einem Textfeld) zur Verfügung.

Da bei Listenfeldern häufig größere Datenmengen auftreten, besteht der berechtigte Wunsch, diese Daten auch abspeichern zu können.
Visual Basic bietet in diesem Zusammenhang unter anderem folgende Standard-Basic-Befehle:

- Open ... Öffnen einer Datei
- Close ... Schließen einer Datei
- Input # ... Einlesen von einer Datei
- Write # ... Schreiben auf eine Datei (unformatiert)
- Print # ... Schreiben auf eine Datei (formatiert)
- FreeFile ... freie Dateinummer suchen/wählen

Beispiel 1:

Einlesen einer Liste von einer Datei:

```
Private Sub cmdLaden_Click()
Dim x As Integer
Dim s As String
Dim nr As Byte
nr = FreeFile
        Open txtFile.Text For Input As #nr
                Do
                Input #nr, s
                lstNamen.AddItem s
                Loop Until EOF(nr)
        Close #nr
End Sub
```

Beispiel 2:

Schreiben einer Liste in eine Datei:

```
Private Sub cmdSpeichern_Click()
Dim x As Integer
Dim nr As Byte
nr = FreeFile
        Open txtFile.Text For Output As #nr
                For x = 0 To lstNamen.ListCount - 1
                Write #nr, lstNamen.List(x)
                Next x
        Close #nr
End Sub
```

---

10. Dateinamenerweiterungen

Ein VB-Projekt ist ein in sich zusammengehörendes Programm, dessen einzelne Elemente im Projektordner anhand folgender Dateinamenerweiterungen zu erkennen ist:

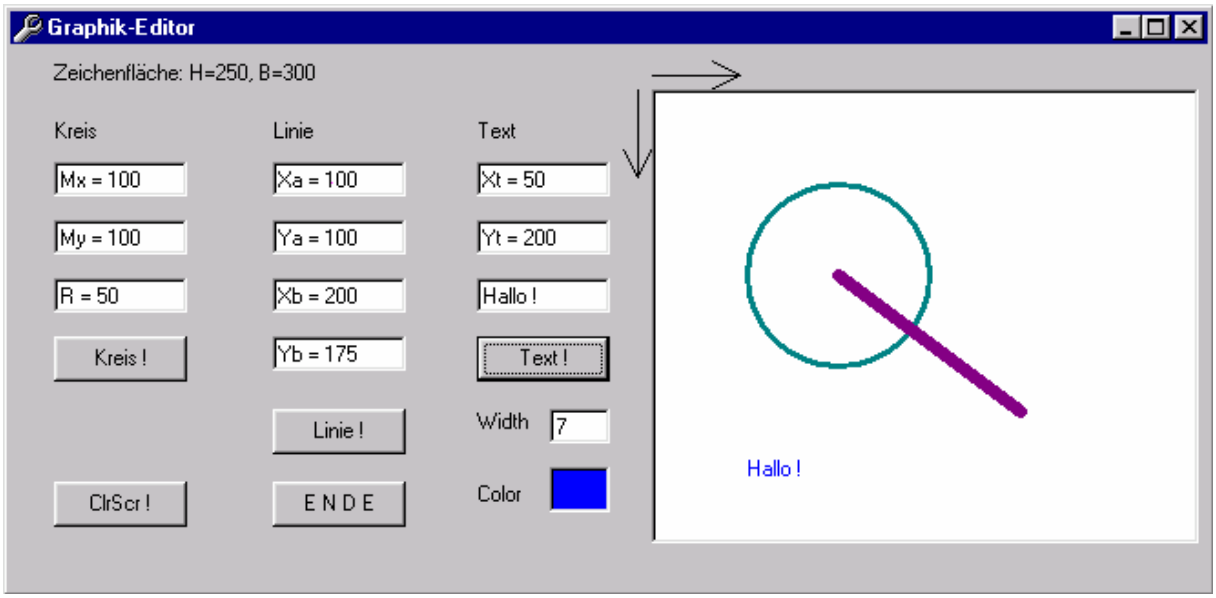
* einer Projektdatei (*.VBP), die Informationen zu allen Komponenten enthält.
* einer Datei pro Formular (*.FRM).
* einer *.VBW-Datei, in der Bildschirm-Positionen gespeichert werden.
* evtl. einer zusätzlichen Datei (*.FRX) pro Form, wenn in der Form Steuerelemente mit Binärdaten enthalten sind.
* einer Datei für jedes Standardmodul (*.BAS).
* einer Datei für jedes Klassenmodul (*.CLS).
* evtl. Dateien mit Zusatzsteuerelementen (*.VBX, *.OCX).
* evtl. einer Resource-Datei (*.RES).

---

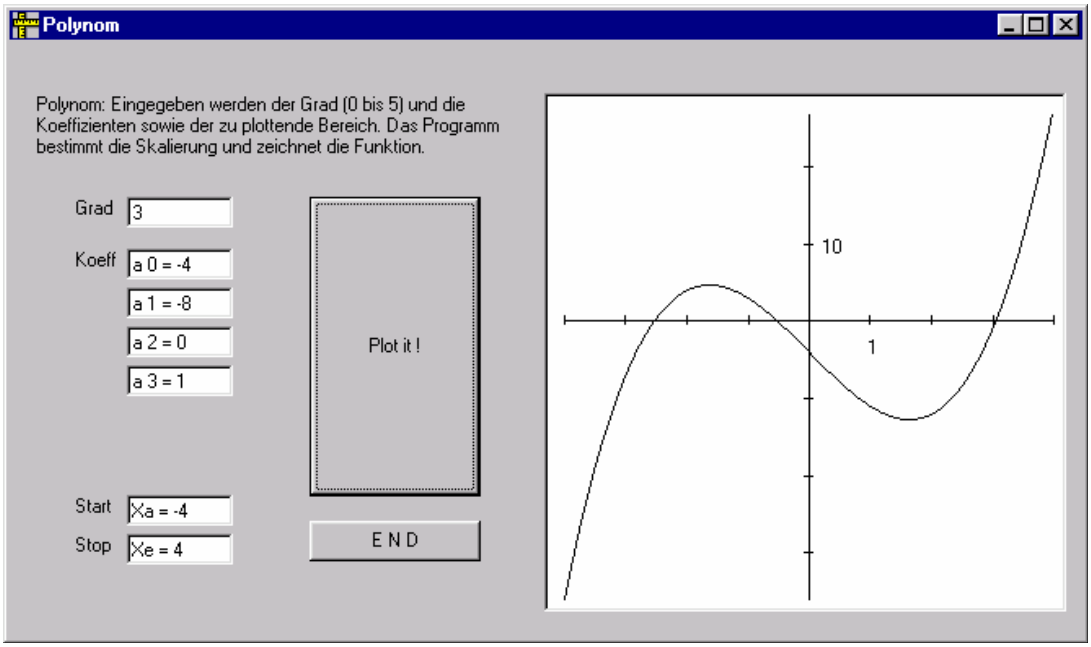
11. Graphikprogramme

Im Bereich technischer Programme ergibt sich oft die Notwendigkeit, bestimmte Sachverhalte (z.B. Messreihen) graphisch darzustellen. VB bietet einfache und effiziente Grundfunktionen zum Erstellen von Graphiken, die im Wesentlichen auf den Steuerelementen LINE und SHAPE sowie dem Bildfeld-Steuerelement beruhen.

Die folgenden beiden Beispiele mit Formularen und zugehörigen Programmcodes sollen dies anschaulich machen:




```
Private Sub cmdClrScr_Click()
        'Löschen der Zeichenebene
        picP1.Cls
End Sub

Private Sub lblColor_Click()
        'Wahl der Zeichenfarbe
        aktuelleFarbe = aktuelleFarbe + 1
                If aktuelleFarbe = 15 Then
                aktuelleFarbe = 0
                End If
        picP1.ForeColor = QBColor(aktuelleFarbe)
        lblColor.BackColor = picP1.ForeColor
End Sub

Private Sub txtWidth_Change()
        'Wahl der Strichstärke
        picP1.DrawWidth = Val(txtWidth.Text)
End Sub
```



Bei Programmen dieser Art ist die sinnvolle Wahl des Koordinatensystems bzw. eine entsprechende (lineare, logarithmische, ...) Koordinatentransformation von großer Bedeutung (Koordinatenursprung, Achsen, Skalierungen):

```
Public Function xt(ByVal x As Double) As Double
        'Koordinaten-Transformation
        ' xy-Zeichenebene auf Bildfläche 300x300
        xt = 300 / (xstopp - xstart) * (x - xstart) + 10
End Function

Public Function yt(ByVal y As Double) As Double
        'Koordinaten-Transformation
        ' xy-Zeichenebene auf Bildfläche 300x300
        yt = 300 - 300 / (maxi - mini) * (y - mini) + 10
End Function
```