Einführung in Visual Basic
für
Informatik Jg. 9

von
StD Herbert Franke
Albert-Einstein-Gymnasium, Kaarst
Juli 2000

Inhaltsverzeichnis

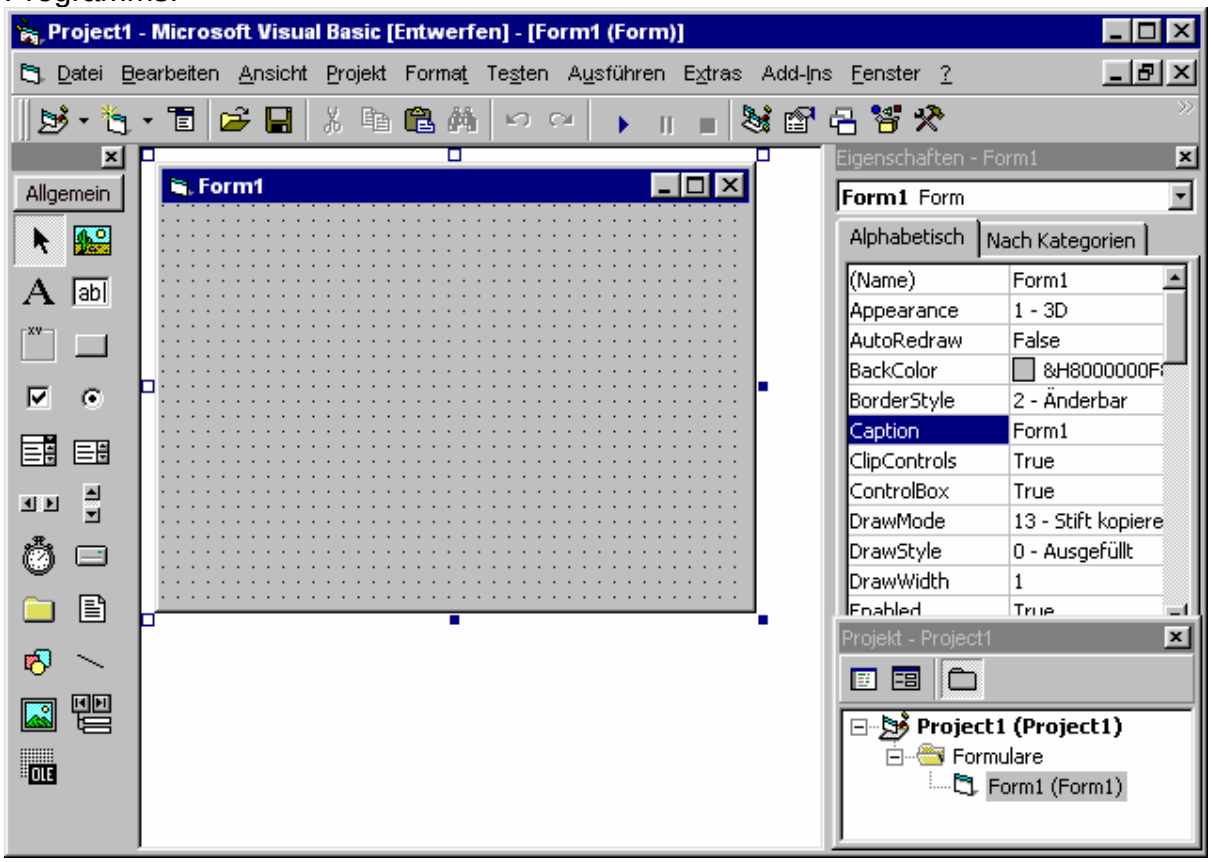
Programmieren mit Visual Basic

Visual Basic ist eine Sprache zur Anwendungsentwicklung unter Windows. Die Arbeitsweise von Windows ist gekennzeichnet durch **Fenster**, **Ereignisse** und **Meldungen**. Windows-Anwendungen sind **ereignisgesteuert**: Die Reihenfolge der Ereignisse bestimmt die Ablauffolge des Programms; der Weg durch den Programm-Code ist bei jeder Ausführung des Programms unterschiedlich.

Interaktive Programmentwicklung

Das herkömmliche Verfahren der Programmentwicklung kann in drei Schritte unterteilt werden: Schreiben, Kompilieren und Testen des Programms. Im Gegensatz zu herkömmlichen Sprachen arbeitet Visual Basic mit einer interaktiven Methode, die eine klare Trennung der drei Schritte unmöglich macht. Visual Basic interpretiert den Programm-Code schon bei der Eingabe, wobei die meisten Syntax- und Schreibfehler „im Vorbeigehen" bemerkt und markiert werden. Es entsteht fast der Eindruck, als ob jemand vom Fach bei der Eingabe des Codes über die Schulter schaut. Aufgrund der interaktiven Arbeitsweise von Visual Basic kannst du deine Anwendungen oft schon während der Entwicklung ausführen.

Die Arbeitsumgebung in Visual Basic wird oft als „**Integrierte Entwicklungs-umgebung"** oder **IDE (Integrated Development Environment)** bezeichnet, weil in ihr viele unterschiedliche Funktionen in einer gemeinsamen Umgebung integriert sind, wie beispielsweise Entwurf, Bearbeitung, Kompilierung und Testen eines Programms.



Den meisten Platz auf dem Bildschirm nimmt eine sogenannte ***Form*** ein, sie hat von Visual Basic den Namen 'Form1' erhalten, dies können wir ändern, aber zunächst ist

uns dieser Name gut genug. In diesem Formfenster ordnen wir die Ein- und Ausgabeelemente so an, wie sie der Benutzer später sehen soll. Einen Vorrat von solchen Elementen siehst du links in der ***Werkzeugleiste***. Wir betrachten zunächst eines dieser Werkzeuge:

Schaltflächen (command button)

Gehe jetzt mit der Maus in die Werkzeugleiste und klicke das Symbol für eine ***Schaltfläche*** (oder auch Befehlsfeld) an:
Dann gehe in die Form, drücke die linke Maustaste und ziehe bei gedrückter Taste die Maus etwas nach rechts unten. Es entsteht eine Schaltfläche, deren Größe du bestimmen kannst. Visual Basic hat der Schaltfläche auch schon eine Aufschrift gegeben, nämlich ‘Command1’. Diese Aufschrift (caption) werden wir später noch ändern, aber zunächst wollen wir einen Befehl mit dieser Schaltfläche verknüpfen. Dazu machst du auf die Schaltfläche einen Doppelklick. Das bewirkt, daß ein Fenster mit dem ***Code-Editor*** aufgemacht wird. Das System hat für uns schon eine Prozedur mit dem Namen ‘Command1_Click’ angelegt. Diese Prozedur wird ausgeführt, wenn der Benutzer die Schaltfläche anklickt. Zur Zeit steht noch nichts drin, aber du kannst dir ja einen Befehl überlegen. Wenn dir nichts Besseres einfällt, dann schreibe in die Prozedur: `PRINT "HALLO !"`
Damit ist unser erstes Programm fertig. Starte es über den Menüpunkt ***Ausführen***. Auf dem Bildschirm erscheint die Form. Sobald du die Schaltfläche anklickst, wird der Befehl, der in der Prozedur Command1_Click steht, ausgeführt. Damit ist das Programm noch nicht beendet! Du kannst mehrmals die Schaltfläche anklicken, jedesmal wird die Prozedur Command1_Click ausgeführt. Das Programm können wir über den Menüpunkt ***Ausführen*** wieder beenden.
Auch Grafikbefehle können wir auf diese Weise ausführen lassen, dazu ist in Visual Basic keine SCREEN-Anweisung nötig, man kann sofort auf die Form zeichnen. Schreibe in die Prozedur

```
FOR r=100 TO 2000 STEP 100
  CIRCLE (r,r),r
NEXT r
```

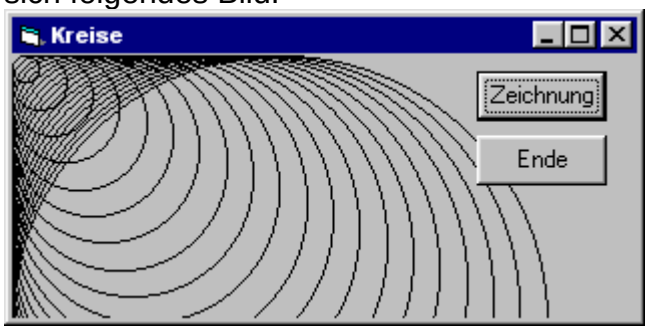
Es stört noch, daß wir das Programm über das Menü beenden müssen. Wir legen deshalb eine neue Schaltfläche an (sie hat zunächst die Aufschrift ‘command2’), durch Doppelklick kommen wir an die Prozedur Command2_Click und fügen ein

```
SUB command2_Click
  END
END SUB
```

Jetzt können wir das Programm beenden, indem wir Command2 anklicken. Allerdings kommt spätestens jetzt der Wunsch auf, die Aufschriften auf den Schaltflächen zu ändern, schließlich sind diese bis jetzt für einen Benutzer nicht sehr hilfreich. Diese Aufschriften sind ***Eigenschaften*** der Schaltflächen. Um sie zu ändern, gehen wir folgendermaßen vor:
- Markieren der Schaltfläche (Anklicken)
- Anklicken der Eigenschaft Caption (Aufschrift) im Eigenschaftsfenster.
- Wert der Eigenschaft Caption ändern und mit RETURN abschließen.

Aufgabe: Ändere die Aufschrift auf der ersten Schaltfläche ab in ‘Zeichnung’, die auf der zweiten Schaltfläche in ‘Ende’. Ändere die Überschrift (Caption) von Form1 in ‘Kreise’.

Wie du bei dieser Gelegenheit siehst, hat eine Schaltfläche sehr viele Eigenschaften, einige sprechen für sich wie *Height* (Höhe), *Width* (Breite) und eben *Caption* (Überschrift, Aufschrift). Eine wichtige Eigenschaft ist noch der *Name*. Dieser lautet weiterhin ‘command1’, auch wenn jetzt die Aufschrift ‘Zeichnung’ auf dem Befehlsfeld steht. Auch die Namen können wir ändern, dadurch wird das Programm lesbarer. In unserem Beispiel geben wir dieselben Namen, wie wir sie als Aufschriften verwendet haben. Wenn du jetzt das Programm ablaufen läßt, ergibt sich folgendes Bild:



Speichern

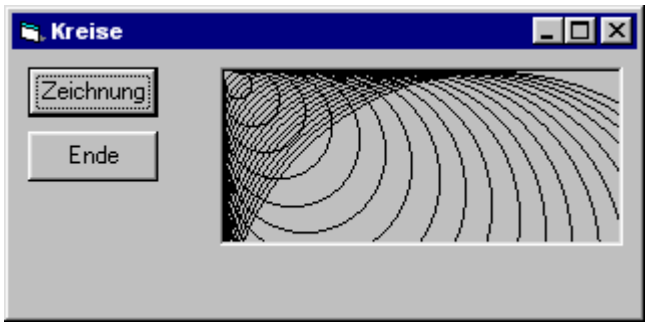
Dieses Programm ist es wert, daß wir es speichern. Im Menüpunkt ‘Datei’ gibt es eine ziemliche Vielfalt an Möglichkeiten. Visual Basic speichert nämlich das Programm (hier heißt es ***Projekt***) nicht als Ganzes ab, sondern jede Form wird einzeln abgespeichert. Falls deine Form noch den Namen ‘Form1’ hat, ändere diesen jetzt ab. Gehe also nochmal zur Form, klicke diese an, und ändere im Eigenschaftsfenster den Namen in ‘Kreise’. Speichere dann das ‘Projekt’ unter dem Namen ‘Kreise’. Dabei wirst du dann gefragt, ob du die Form ‘Kreise’ speichern willst.

Bildfelder (picturebox)

Meistens zeichnet und schreibt man nicht direkt in die Form, sondern verwendet dazu ***Bildfelder (PictureBox).*** Wähle aus der Werkzeugleiste das entsprechende Symbol und plaziere ein Bildfeld auf der Form. Mache das Bildfeld nicht zu klein.
Das Bildfeld erhält vom System den Namen ‘picture1’. Bildfelder haben keine Aufschrift (Caption). Nun sollen die Kreise nicht mehr auf die ganze Form, sondern nur noch in das Bildfeld gezeichnet werden. Das geschieht dadurch, daß man vor den Zeichenbefehlen, durch einen Punkt getrennt, den Namen des Bildfeldes angibt. In unserem Fall müssen wir die Prozedur Zeichnung_Click abändern

```
SUB Zeichnung_Click
  FOR r=100 TO 2000 STEP 100
    picture1.Circle(r,r),r
  NEXT r
END SUB
```

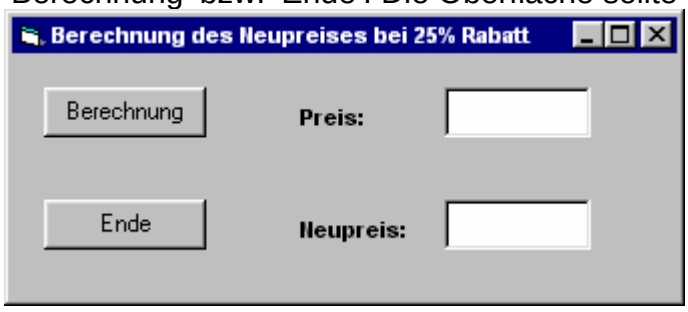
Wenn du das Programm startest und auf ‘Zeichnung’ drückst, sollte sich folgendes Bild ergeben:



Speichere dieses Programm (Visual Basic nennt es Projekt) unter dem Namen ‘Kreise2’. Beachte, daß du der geänderten Form vorher einen anderen Namen z.B. ‘Kreise2’ gibst, da Formen in gesonderten Dateien abgespeichert werden.

Textfelder und Bezeichnungsfelder

In Visual Basic gibt es den INPUT-Befehl nicht! Die Ein- und Ausgabe von Zahlen und Texten erfolgt über ***Textfelder***. Wir betrachten ein kleines Beispiel dazu: Für den Schlussverkauf müssen von den alten Preisen immer 25%-Rabatt abgerechnet werden. Wir benützen dazu zwei Textfelder; eines für den alten Preis, das andere für den neuen.
Gehe in die Werkzeugleiste und klicke das Symbol für ein Textfeld an. Ziehe in der Form zwei Textfelder auf. Textfelder haben keine Aufschrift (Caption). Im Eigenschaftsfenster siehst du, daß die Textfelder die Namen ‘Text1’ und ‘Text2’ haben.
Zu jedem Textfeld gehört ein Text. Im ersten steht ‘text1’ und im zweiten ‘text2’. Das wollen wir gleich ändern. Beide Textfelder sollen als Text zunächst gar nichts enthalten, das erreichst du durch entsprechenden Eintrag ins Eigenschaftenfenster. Als Namen geben wir den Textfeldern gemäß ihrer Bestimmung ‘Preis’ bzw. ‘Neupreis’.
Damit später auch der Benutzer weiß, was in den Textfeldern ist, öffne nun noch zwei ***Bezeichnungsfelder (Label).*** Das machst du ganz entsprechend zu den anderen Feldern, das Symbol in der Werkzeugleiste ist ein ‘A’.
Diese Bezeichnungsfelder tragen zunächst die Aufschrift ‘Label1’ bzw. ‘Label2’. Ändere diese Eigenschaften über das Eigenschaftenfenster in ‘Preis:’ bzw. ‘Neupreis:’. Schließlich zeichnest du noch zwei Befehlsfelder in die Form. Das eine bekommt die Aufschrift ‘Berechnung’, das andere ‘Ende’. Gib ihnen auch die Namen ‘Berechnung’ bzw. ‘Ende’. Die Oberfläche sollte dann etwa so aussehen:



Die Oberfläche ist perfekt, jetzt müssen nur noch die Prozeduren hinter der Oberfläche programmiert werden. Die Zahl, die im Textfeld ‘Neupreis’ steht, muß 75% des Preises betragen. Wir schreiben:

```
Private SUB Berechnung_Click()
  Neupreis.Text = 0.75*Preis.Text
```

```
END SUB
```

Unsere Programmierarbeit bestand nur aus einer Zeile, diese muß allerdings verstanden werden. Wir können sie folgendermaßen lesen:
Der Text im Textfeld ‘Neupreis’ ergibt sich, indem man den Text im Textfeld ‘Preis’ mit 0.75 multipliziert.
Aufgabe: Gestalte die Oberfläche wie angegeben, schreibe die Prozeduren Berechnung_Click und Ende_Click. Starte das Programm. Was passiert, wenn man die Berechnung anklickt und im Textfeld Preis keine Zahl steht?

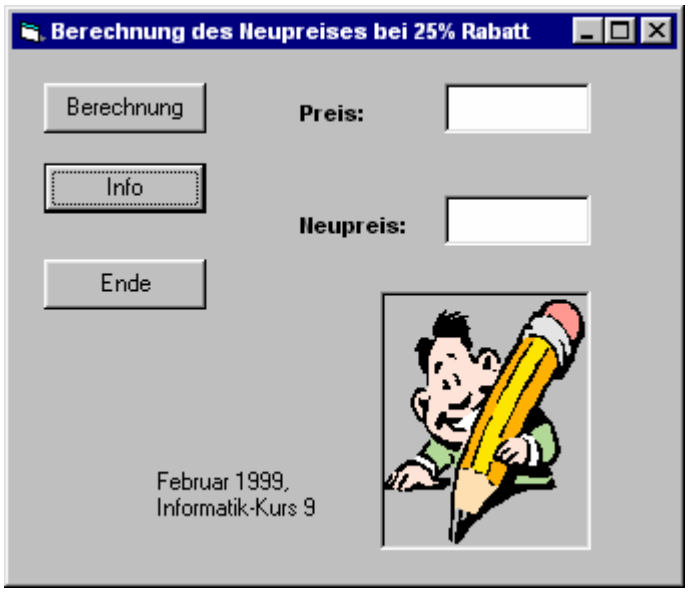
Mit Hilfe der **VAL**-Funktion, die zu einer Zeichenkette die zugehörige Zahl berechnet, können wir diesen Mangel beheben. Falls sich keine Zahl im Textfeld befindet, wollen wir dem Benutzer eine Meldung geben. Dies geht in Visual Basic sehr einfach mit der Anweisung **MsgBox**. Wir ändern die Prozedur Berechnung_Click:

```
Private SUB Berechnung_Click()
  z = VAL(Preis.Text)
  IF z<=0 THEN
    MsgBox ”Gültige Zahl eingeben!”
  ELSE
    Neupreis.Text = 0.75 * z
  END IF
  Preis.SetFocus
END SUB
```

Die letzte Anweisung setzt den Cursor wieder in das Textfeld Preis. Wir wollen außerdem, daß der Neupreis verschwindet, sobald im Textfeld etwas geändert wird. Dies erreichen wir mit dem **Change-Ereignis**. Gib dem Textfeld Preis einen Doppelklick. Es wird die Prozedur Preis_Change angeboten; ergänze die Prozedur mit der VB-Anweisung

```
Neupreis.Text=””
```

und teste dann das Programm.



Aufgabe: Ergänze die Form mit einem Befehlsfeld ‘Info’. Wird dieses angeklickt, so erscheint ein Bildfeld mit der Grafikdatei *Manstift.wmf*. Wird dieses Bildfeld angeklickt, so verschwindet es wieder. Beachte dazu die Eigenschaft **Visible**.

Dokumentation zum Einführungsbeispiel: Visual-Basic-Projekt PREIS

Projektdatei: Preis.vbp
Formulardatei: Preis.frm

Steuerelement	Eigenschaft	Wert
Textfeld	Name	*Preis*
	Text	*leer*
Textfeld	Name	*Neupreis*
	Text	*leer*
Bezeichnungsfeld	Name	*label1*
	Caption	*Preis:*
	Font	*Arial*
Bezeichnungsfeld	Name	*label2*
	Caption	*Neupreis:*
	Font	*Arial*
Schaltfläche	Name	*Berechnung*
	Caption	*Berechnung*
Schaltfläche	Name	*Ende*
	Caption	*Ende*

Ereignisprozeduren:

```
Private SUB Berechnung_Click()
     Neupreis.Text = 0.75 * Preis.Text
END SUB

Private SUB Ende_Click()
     END
END SUB

Private SUB Preis_Change()
     Neupreis.Text = ""
END SUB
```

weitere Steuerelemente:

Steuerelement	Eigenschaft	Wert
Schaltfläche	Name	*cmdInfo*
	Caption	*Info*
Bildfeld (Picturebox)	Name	*picInfo*
	Picture	*Manstift.wmf*
	Visible	*False*
Bezeichnungsfeld	Name	*lblInfo*
	Caption	*IF-Kurs 9*
	Visible	*False*

Ereignisprozeduren:

```
Private SUB cmdInfo_Click()
     picInfo.Visible = True
     lblInfo.Visible = True
END SUB
```

```
Private SUB picInfo_Click()
     picInfo.Visible = False
     lblInfo.Visible = False
END SUB
```

Aufgaben:

1. Ergänze im Bereich (Allgemein) des Programm-Codes die Anweisung
   `OPTION EXPLICIT`

Damit übernimmt der/die Programmierer(in) die Verantwortung für alle Namensgebungen.

2. Fehlerhafte Bedienung durch den Benutzer sollte im Programm ausgewertet werden. Dies leistet z.B. die folgende Erweiterung der Prozedur Berechnung_Click:

```
Private SUB Berechnung_Click()
     z = VAL(Preis.Text)
     IF z<=0 THEN
          MsgBox "Gültige Zahl eingeben!"
     ELSE
          Neupreis.Text = 0.75 * z
     END IF
     Preis.SetFocus
END SUB
```

3. Das Beispiel verdeutlicht die Vorgehensweise bei der Entwicklung einer Anwendung mit Visual Basic. Es enthält aber auch eine Reihe von Überlegungen, die bei weiteren Projekten zu Problemen führen.

Lernerfolgskontrolle

1. Visual Basic ist eine Sprache zur Anwendungsentwicklung unter Windows. Die Arbeitsweise von Windows ist gekennzeichnet durch

2. Windows-Anwendungen sind .
   Was bedeutet das?

3. Warum bezeichnet man die Arbeitsumgebung in Visual Basic als „**Integrierte Entwicklungsumgebung" oder IDE (Integrated Development Environment)**?

4. Die Ein- und Ausgabeelemente, die der Programmbenutzer zu sehen bekommt, befinden sich auf .

5. Die ***Werkzeugleiste*** stellt einen dar.

6. Kennzeichne die bisher benutzten Werkzeuge in der nebenstehenden Werkzeugbox und notiere ihre Bezeichnungen (deutsch/englisch).

7. Visual Basic speichert die Entwurfselemente nicht als Ganzes ab, sondern

8. Die verschiedenen Werkzeugelemente besitzen Eigenschaften. Nenne jeweils 3 Eigenschaften zu den bisher benutzten Werkzeugelementen.

9. Der Programm-Code zu den Elementen auf einer Form besteht aus einzelnen .

10. Ein Steuerelement reagiert auf sog. Ereignisse, z.B. einen Klick mit der linken Maustaste. Wie lautet der Rahmen der zugehörigen Prozedur?

Zusammenfassung

Elemente der Integrierten Entwicklungsumgebung:

Hauptmenü, Symbolleiste, Werkzeugsammlung, Projekt-Fenster, Eigenschaften-Fenster, Code-Editor-Fenster, Formular-Layout-Fenster, Form.

Erstellen eines Anwendungsprogramms

1. Erstellen der Benutzeroberfläche
2. Festlegen der Eigenschaften
3. Schreiben des Programm-Codes

Formulare (forms) bilden die Grundlage für das Erstellen der Benutzeroberfläche einer Anwendung. In das Formular werden **Steuerelemente (controls)** aus der Werkzeugsammlung eingefügt. Jedes Steuerelement besitzt eine Reihe von **Eigenschaften**. Zu den Steuerelementen kann man Programm-Code schreiben, also Anweisungen in der Programmiersprache Visual Basic, insbesondere **Ereignisprozeduren (event procedures).** Jedes Anwendungsprogramm ist ereignisgesteuert (event driven).
Die wichtigste Eigenschaft für jedes Steuerelement ist der **Name**. Wenn wir ein Steuerelement benennen, geben wir daher zwei Informationen an: den ***Typ*** des Steuerelements und den ***Zweck***. Dadurch wird die Code-Programmierung besonders einfach.
Folgende Konventionen gelten in der Gemeinde der VB-Programmierer:

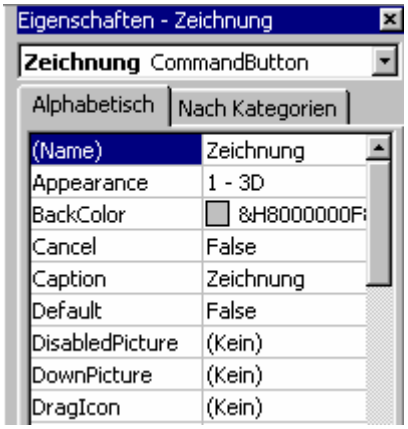
englisch	deutsch	Präfix
control	Steuerelement	
form	Formular	frm
command button	Steuerschaltfläche	cmd
label	Bezeichnungsfeld	lbl
TextBox	Textfeld	txt
PictureBox	Bildfeld	pic
CheckBox	Kontrollkästchen	chk

Ereignisprozeduren haben folgende Syntax

```
SUB SteuerelementName_Ereignis()
   BASIC-Anweisungen
End SUB
```

Die Wertzuweisung für ein Steuerelement lautet generell:

```
Steuerelement.EigenschaftName = EigenschaftWert
```

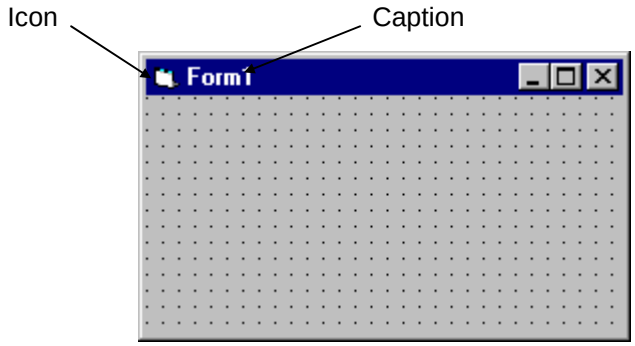


Projektentwurf

Du erstellst im weiteren VisualBasic-Projekte. Bevor du ein Projekt beginnst, ist es gut, wenn du zunächst etwas Zeit darauf verwendest, dir klar zu machen, was du tun willst. Die Überlegungen zu einem geeigneten Projektentwurf sparen viel Zeit und führen zu besseren Ergebnissen. Hauptgedanke dabei ist, ein Projekt zu erstellen, daß einfach zu bedienen, einfach zu verstehen und ohne Fehler ist.

Das Formular (form)

Wir haben gesehen, daß Formulare die zentrale Steuerstelle bei der Entwicklung von VisualBasic-Projekten sind. Wir betrachten einige wichtige Eigenschaften und Ereignisse für Formulare.



Eigenschaften

Hier einige der wichtigsten Eigenschaften:
Name, Caption, Icon, Left, Top, Width, Height, Backcolor, BorderStyle

Aufgabe: Beschreibe die Bedeutung dieser Eigenschaften? Erstelle dazu ein neues Projekt nur mit einem Formular und teste.

Ereignisse

Das Formular dient vorrangig als 'Container' für andere Steuerelemente, aber es unterstützt auch Ereignisse, d.h. es kann auf einige Benutzeraktionen reagieren. Für uns sind nur zwei Ereignisse von Bedeutung:

Ereignis	Beschreibung
Click	Ereignis, das ausgeführt wird, wenn der Benutzer mit der Maus auf das Formular klickt.
Load	Ereignis, das ausgeführt wird, wenn das Formular zum ersten Mal in den Speicher geladen wird. Dies ist ein guter Platz, um Startwerte für verschiedene Eigenschaften und andere Projektwerte einzutragen.

Noch ein Wort zur Namensgebung von Formularen. Zur Erinnerung: Namen von Steuerelementen werden in Ereignisprozeduren benutzt. Dies gilt nicht für Formulare. Alle Formular-Ereignisprozeduren haben das Format:

```
Form_EreignisName
```

D.h., egal wie man die Eigenschaft Name des Formulars wählt, werden Ereignisprozeduren immer unter dem Begriff **Form** aufgelistet.

Steuerschaltflächen (command button)

Steuerschaltflächen dienen dazu, bestimmte Prozesse zu starten, anzuhalten oder zu beenden.

Werkzeug: **Auf dem Formular**:

Eigenschaften

Name	Name, um die Steuerschaltfläche zu identifizieren. Das Präfix ist **cmd**.
Caption	Text, der auf der Steuerschaltfläche erscheint.
Font	Setzt Stil, Größe und Typ des Textes.
Left	Abstand linker Rand zum linken Rand der Form.
Top	Abstand oberer Rand zum oberen Rand der Form.
Width	Breite in twips.
Height	Höhe in twips.
Enabled	Bestimmt, ob die Steuerschaltfläche auf Benutzerereignisse reagiert (bei Ausführung).
Visible	Bestimmt, ob die Steuerschaltfläche auf der Form erscheint (bei Ausführung)

Ereignisse

Nur ein Steuerschaltflächen-Ereignis ist von Bedeutung, aber ein sehr wichtiges:

Ereignis	**Beschreibung**
Click	Ereignis, daß ausgeführt wird, wenn der Benutzer mit der Maus auf die Steuerschaltfläche klickt.

Jede Steuerschaltfläche besitzt eine Ereignisprozedur zum Click-Ereignis.

VISUAL BASIC

Jede Eigenschaft hat einen speziellen Datentyp, abhängig von der Art der Information, die dargestellt wird. Die Wertzuweisung lautet allgemein

```
SteuerelementName.EigenschaftName = EigenschaftWert
```

Als Datentypen kommen vor:
integer, long integer, symbolische Konstanten (sie beginnen immer mit **vb**)**, Boolean, string**.

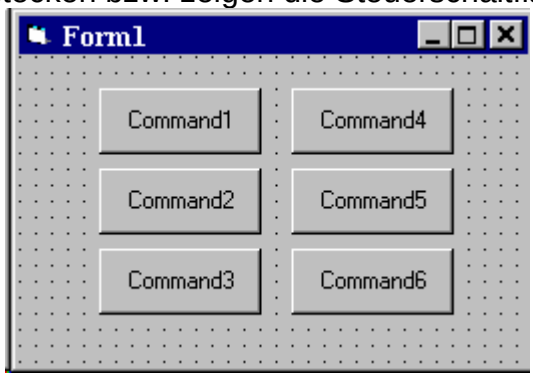
Kommentare

Jeder Programm-Code sollte angemessen kommentiert werden! Um einen Kommentar in einer Prozedur zu schreiben, verwendet man das Kommentar-Symbol: *das Apostroph* (‘).

Projekt - Form Fun

Projektentwurf

Bei diesem Projekt benutzen wir Steuerschaltflächen, um spasseshalber einige Formulareigenschaften zu ändern. Wir ändern die Größe des Formulars, die Farbe des Formulars und verstecken bzw. zeigen die Steuerschaltflächen wieder an.



Festlegung der Eigenschaften

Steuerelement	EigenschaftName	EigenschaftWert
Form1	Name	frmFormFun
	Caption	Form Fun
Command1	Name	cmdKleiner
	Caption	Verkleinern
Command2	Name	cmdGrößer
	Caption	Vergrößern
Command3	Name	cmdVerstecken
	Caption	Verstecken
Command4	Name	cmdRot
	Caption	Rot
Command5	Name	cmdBlau
	Caption	Blau
Command6	Name	cmdZeigen
	Caption	Zeigen
	Visible	False

Die Ereignisprozeduren

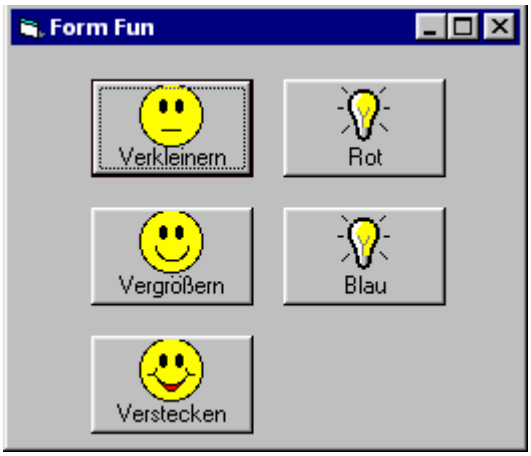
Hier die erforderliche Ereignisprozedur für die Steuerschaltfläche **cmdKleiner:**

```
Private Sub cmdKleiner_Click()
 'Formular verkleinern
 'Höhe um 100 twips verringern
 frmFormFun.Height = frmFormFun.Height - 100
 'Breite um 100 twips verringern
 frmFormFun.Width = frmFormFun.Width - 100
End Sub
```

Aufgaben:

1. Schreibe die anderen Prozeduren für das Click-Ereignis bei den Steuerschaltflächen **cmdGrößer, cmdVerstecken, cmdRot, cmdBlau, cmdZeigen**.

2. Schreibe eine Ereignisprozedur **Form_Click**, mit der das Formular ebenfalls vergrößert werden kann.
3. Ändere den Code für **cmdKleiner** und **cmdGrößer** so ab, daß das Formular auf dem Bildschirm herumbewegt werden kann.
4. Ergänze weitere mögliche Hintergrundfarben für das Formular.
5. Ändere die Steuerschaltfläche **cmdVerstecken** so, daß es die Eigenschaft *Enabled* der Steuerschaltflächen auf *False* setzt, nicht die *Visible* Eigenschaft. Ändere entsprechend auch **cmdZeigen**.
6. Die Steuerschaltflächen kann man mit Bildern verschönern. In deinem Verzeichnis findest du einige Symbole (engl. **Icons** Dateiendung .ico). Die Symbol-Datei wird der **Picture**-Eigenschaft der Steuerschaltfläche zugeordnet. Damit die Grafik auf der Steuerschaltfläche angezeigt wird, muß die **Style**-Eigenschaft der Steuerschaltfläche von "0 - Standard" auf "1 - Grafisch" und die **Appearance**-Eigenschaft auf "1 - 3D" geändert werden.



Bildlaufleisten (scroll bar)

Es gibt zwei Sorten: horizontale und vertikale. Wir können sie verwenden, um Zahlen einzugeben, Lautstärken zu regeln oder um aus einer Optionsliste zu wählen.

horizontale Bildlaufleiste

Werkzeug 

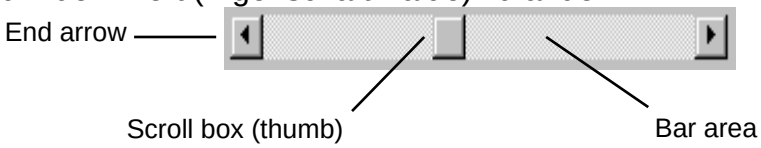
Im Formular (Standard): 

vertikale Bildlaufleiste

Werkzeug 

Im Formular (Standard): 

Beide Typen bestehen aus drei Bereichen, die angeklickt oder gezogen werden können, um den Wert (Eigenschaft Value) zu ändern.



Klick auf **Endpfeile** bewegt die **scroll box** ein kleines Stück, Anklicken der **bar area** bewegt die scroll box ein großes Stück, ziehen der scroll box ermöglicht eine fliessende Bewegung. Mit den Eigenschaften einer Bildlaufleiste kann man ihre Arbeitsweise vollständig festlegen.

Eigenschaften

Eigenschaft	Beschreibung
Name	identifiziert die Bildlaufleiste. Präfix: hsb bzw. vsb.
Value	Die aktuelle Position (ein Integer-Wert) der scroll box.
Max	Der größte Wert. (zwischen -32768 und 32767).
Min	Der kleinste Wert.
SmallChange	Schrittweite, um die sich der Wert ändert, wenn man auf die Endpfeile klickt.
LargeChange	Schrittweite, um die sich der Wert ändert, wenn man auf die bar area klickt
Left, Top, Width, Height, Enabled, Visible	wie bei den anderen Steuerelementen

Ereignisse

Wir benutzen zwei Ereignisse bei Bildlaufleisten:

Ereignis	Beschreibung
Change	Ereignis wird immer dann ausgeführt, wenn der Wert (Eigenschaft Value) sich geändert hat.
Scroll	Ereignis wird ständig ausgeführt, wenn die scroll box bewegt wird.

Das **Change**-Ereignis wird immer dann ausgeführt, wenn sich der Wert ändert. Hier schreiben wir also den Code für die Anzeige des Wertes im begleitenden Steuerelement (üblicherweise ein Bezeichnungsfeld!; die Caption-Eigenschaft erhält

den Wert der Bildlaufleiste). Die Anweisungen gelten natürlich auch für das **Scroll**-Ereignis, weshalb wir sie in die Ereignisprozedur für scroll kopieren.

VISUAL BASIC

Entscheidungen

```
If Ausdruck Then
   [BASIC-Anweisungen, falls Ausdruck wahr]
Else
   [BASIC-Anweisungen, falls Ausdruck falsch]
End If
If Ausdruck1 Then
   [BASIC-Anweisungen, falls Ausdruck1 wahr]
ElseIf Ausdruck2 Then
   [BASIC-Anweisungen, falls Ausdruck2 wahr]
ElseIf Ausdruck3 Then
   [BASIC-Anweisungen, falls Ausdruck3 wahr]
Else
   [BASIC-Anweisungen, falls Ausdruck1, Ausdruck2 und Ausdruck3 falsch]
End If
```

Zufallszahlen (Rnd-Funktion)

Die Visual-BASIC-Funktion **Rnd** erzeugt eine Zufallszahl zwischen 0 und 1.

```
Zufallszahl = Rnd
```

Mit dem Befehl **Randomize** wird der Zufallsgenerator initialisiert. Dieser Befehl sollte unbedingt in der **Form_Load** Prozedur stehen, also

```
Private Sub Form_Load()
  Randomize
End Sub
```

Um ganzzahlige Zufallszahlen von *klein* bis *gross* zu erhalten, gilt:

```
Zufallszahl = Int(Rnd * (gross - klein + 1)) + klein
```

Dabei wird die BASIC-Funktion **Int** benutzt, die eine Dezimalzahl in eine ganze Zahl umwandelt. Für die Simulation eines Würfels schreibt man:

```
Augenzahl = Int(Rnd * 6) + 1
```

Für ganze Zahlen von 0 bis 100:

```
Zahl = Int(Rnd * 101)
```

Strings (Zeichenketten)

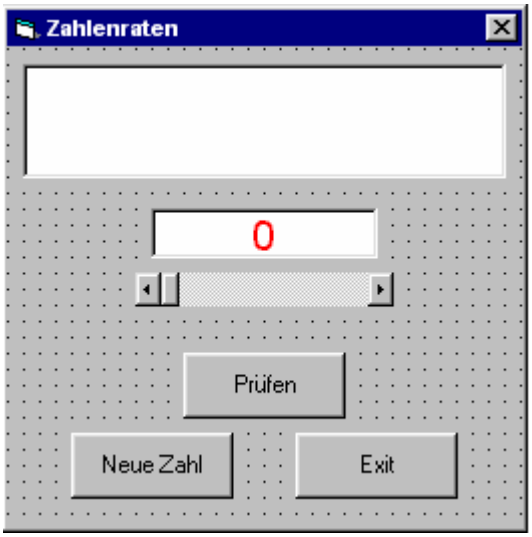
Eine Folge von Zeichen bezeichnet man als **string** (Zeichenkette); sie werden mit Anführungszeichen umrahmt.
Die Caption-Eigenschaft enthält stets einen string. Sollen Zahlen dargestellt werden, so sind diese zunächst in einen string umzuwandeln. Dies erfolgt mit der Basic-Funktion **Str**.
Beispiel:

```
MeineZahl = 71
lblRaten.Caption = Str(MeineZahl)
```

Projekt: Spiel - Zahlenraten

Ich denke mir eine Zahl zwischen 0 und 100. Rate diese Zahl. Als Hilfe sage ich nur, ob die Zahl zu klein, zu groß oder richtig ist.

Die Steuerelemente auf dem Formular
Wir benutzen drei Steuerschaltflächen, zwei Bezeichnungsfelder und eine horizontale Bildlaufleiste.



Festlegung der Eigenschaften

Steuerelement	EigenschaftName	EigenschaftWert
Form1	Name	frmZahlenraten
	Caption	Zahlenraten
	BorderStyle	1-Fixed Single
Label1	Name	lblMitteilung
	Caption	[Blank]
	Font	Arial
	FontSize	16
	Alignment	2-Center
	BackColor	White
	ForeColor	Blue
	BorderStyle	1-Fixed Single
Label2	Name	lblRaten
	Caption	0
	Font	Arial
	FontSize	16
	Alignment	2-Center
	BackColor	White
	ForeColor	Red
	BorderStyle	1-Fixed Single
HScroll1	Name	hsbRaten
	Value	0
	Min	0
	Max	100
	SmallChange	1

	LargeChange	10
	Enabled	False
Command1	Name	cmdPrüfen
	Caption	Prüfen
	Enabled	False
Command2	Name	cmdNeueZahl
	Caption	Neue Zahl
Command3	Name	cmdEnde
	Caption	Exit

Programmierung

Durch einen Klick auf den Button NeueZahl wählt der Computer eine Zufallszahl. Dieses Click-Ereignis aktiviert die Bildlaufleiste und den Prüfen-Button. Geraten wird mit der Bildlaufleiste, wonach der Prüfen-Button angeklickt wird. Der Computer sagt dann im Bezeichnungsfeld **lblMitteilung** etwas zum Tip. Wir brauchen also für jede Steuerschaltfläche eine click-Ereignisprozedur und wir benötigen eine Prozedur für die Anzeige der mittels Bildlaufleiste eingegebenen Zahl.
Ferner benötigen wir zwei Variablen vom Typ **Integer**. Eine Variable **Zahl** speichert die vom Computer erzeugte Zufallszahl, die andere (**MeinTip**) speichert die eingegebene Zahl.
Dies erfolgt im Code-Fenster unter **allgemeine Deklarationen**:

```
Option Explicit
Dim Zahl As Integer
Dim MeinTip As Integer
```

Wenn die Steuerschaltfläche cmdNeueZahl angeklickt wird, muß der Computer folgende Schritte ausführen:

- Bereitstellen einer Zufallszahl zwischen 0 und 100.
- Mitteilung für den Benutzer anzeigen.
- Bildlaufleiste aktivieren, damit geraten werden kann.
- Prüfen-Button aktivieren, um das Raten zu ermöglichen.

Und wir ergänzen noch einen Schritt. Wird der Button NeueZahl einmal angeklickt, hat der Button keine weitere Aufgabe, solange bis die Zahl geraten wurde. Aber wir brauchen einen Button, mit dem wir die Lösung gezeigt bekommen, wenn wir das Spiel vorzeitig aufgeben. Dazu benutzen wir **cmdNeueZahl**. Wir ändern die Caption-Eigenschaft und ergänzen Entscheidungslogik, um den Zustand dieser Steuerschaltfläche zu sehen. Wenn bei einem Klick die Aufschrift "Neue Zahl" lautet, führen wir die obigen Schritte aus. Wenn bei einem Klick die Aufschrift "Lösung zeigen" lautet, wird die Lösungszahl angezeigt und alle Steuerelemente wieder auf ihre Ausgangswerte gesetzt. (Ein Standardverfahren bei Visual Basic.)

Erweiterungen:

1. Ergänze eine Textbox, so daß der Benutzer den Zahlenbereich, in dem die Zahl gesucht werden soll, selbst festlegen kann.
2. Die Mitteilungen an den Benutzer können etwas "genauer" sein. Z.B. wie beim Spiel "Blinde Kuh" die Angaben *eiskalt, kalt, warm, heiß*. Dabei hilft die BASIC-Funktion **Abs**, mit der man den Betrag einer Zahl bzw. den Abstand zweier Zahlen bestimmen kann.
3. Eine weitere Möglichkeit besteht darin, aus dem Projekt ein kleines Mathematik-Spiel zu machen, indem man dem Spieler mitteilt, wie weit seine geratene Zahl von der gedachten Zahl entfernt ist.

4. Noch eine Erweiterungsmöglichkeit: Die Anzahl der benötigten Versuche wird gezählt (Variable Versuche). Ist die gedachte Zahl gefunden worden, erhält der Spieler im Mitteilungsfeld eine Bewertung, z.B.
   a) Versuche < 5 Mitteilung "Das war reiner Zufall!"
   b) 4 < Versuche < 8 Mitteilung "Super!"
   c) Versuche > 7 Mitteilung "Schwach!"
   Für die Bereichsüberprüfung in b) schreibt man in Visual Basic den Ausdruck
   (Versuche > 4) AND (Versuche < 8) .

Programmentwicklung zum Projekt "Zahlenraten"

Algorithmus	**Programm-Code**
Prozedur cmdNeueZahl_Click Zufallszahl ermitteln und in Zahl speichern Mitteilung anzeigen hsbRaten aktivieren cmdPrüfen aktivieren	**Private Sub cmdNeueZahl_Click()** Zahl = Int(Rnd*101) lblMitteilung.Caption="Ich denke mir eine Zahl zwischen 0 und 100" hsbRaten.Enabled=True cmdPrüfen.Enabled=True End Sub

Prozedur cmdPrüfen_Click
WENN MeinTip = Zahl DANN
Mitteilung "Stimmt!" anzeigen
hsbRaten deaktivieren
cmdPrüfen dektivieren
SONST WENN MeinTip < Zahl DANN
Mitteilung "zu klein!"
SONST
Mitteilung "zu gross!"
	Private Sub cmdPrüfen_Click()
	IF MeinTip=Zahl THEN
	lblMitteilung.Caption="Stimmt!"
	hsbRaten.Enabled=False
	cmdPrüfen.Enabled=False
	cmdNeueZahl.Caption="Neue Zahl"
	ELSEIF MeinTip < Zahl THEN
	lblMitteilung.Caption="Zu klein!"
	ELSE
	lblMitteilung.Caption="Zu groß!"
	END IF
	End Sub

Prozedur hsbRaten_Change Eigenschaft Value in MeinTip speichern MeinTip im Label lblRaten anzeigen	**Private Sub hsbRaten_Change()** MeinTip = hsbRaten.Value lblRaten.Caption = STR(meinTip) End Sub
Prozedur hsbRaten_Scroll wie hsbRaten_Change	wie **hsbRaten_Change**

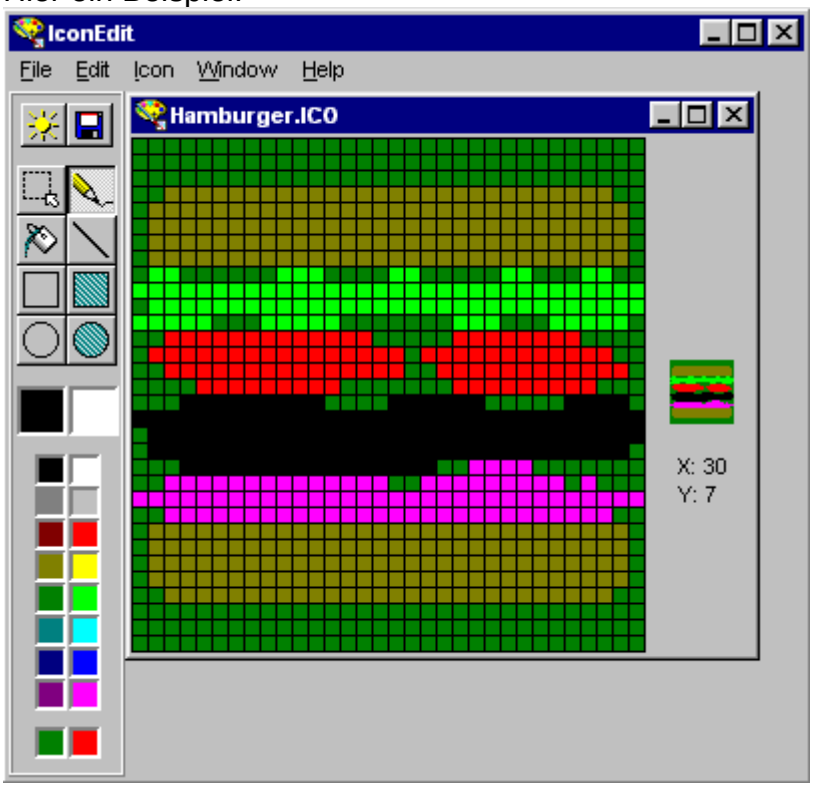
Erweiterungen zu **cmdNeueZahl_Click**
WENN Aufschrift = "Neue Zahl" DANN
wie oben und
Aufschrift ändern in "Lösung zeigen"
SONST
Lösung in lblMitteilung anzeigen
Steuerelemente zurücksetzen auf Anfangswerte
	IF cmdNeueZahl.Caption="Neue Zahl" DANN ...
	cmdNeueZahl.Caption="Lösung zeigen"
	ELSE
	lblMitteilung.Caption="Die gedachte Zahl ist "+STR(Zahl)
	hsbRaten.Enabled=False
	cmdPrüfen.Enabled=False
	cmdNeueZahl.Caption="Neue Zahl"
	END IF

Zufallsgenerator aktivieren	**Private Sub Form_Load()** Randomize End Sub
Variablen bereitstellen	Option Explicit **DIM Zahl AS Integer** **DIM MeinTip AS Integer**

Symbole, Rahmenfelder, Kontrollkästchen, Optionsfelder (und alles zusammen für einen Hamburger)

Symbole (Icon)

Jedes Formular hat links oben ein Symbol (icon), so wie sie auch durchweg in Windows verwendet werden. Mit dem Freeware-Programm **IconEdit** kann man seine eigenen Symbole erstellen und vorhandene ändern. Die Bedienung ist sehr einfach.
Hier ein Beispiel:



Aufgabe: Erstelle selbst ein Hamburger-Symbol.
Mit der Eigenschaft Icon kann man dem Formular dieses Symbol zuweisen.

Das Rahmenfeld (frame)

Ein Rahmen fasst einfach nur andere Steuerelemente zu einer Gruppe zusammen, so daß man sie z.B. gemeinsam aktivieren kann. Ein Rahmenfeld hat keine Ereignisse, sondern nur Eigenschaften.

Werkzeug 

Auf dem Formular: 

Eigenschaften

Eigenschaft	Beschreibung
Name	identifiziert das Rahmenfeld. Präfix ist **fra**.

Caption	Titelangabe über dem Rahmenfeld
Enabled	bestimmt, ob alle Steuerelemente im Rahmenfeld auf Benutzerereignisse reagieren (bei Ausführung).
Visible	bestimmt, ob das Rahmenfeld (und enthaltene Steuerelemente) auf dem Formular erscheinen (bei Ausführung).

Steuerelemente in ein Rahmenfeld plazieren

Steuerelemente müssen aus der Werkzeugsammlung mit einem einfachen Klick ausgewählt werden und dann in das Rahmenfeld gezeichnet werden!

Das Kontrollkästchen (check box)

Mit Hilfe von Kontrollkästchen kann man eine Auswahl vornehmen (ankreuzen) oder verschiedene Zustandsgrößen anzeigen lassen. Kontrollkästchen werden üblicherweise in einem Rahmenfeld angeordnet.

Werkzeug:

Auf dem Formular:

Eigenschaften

Eigenschaft	Beschreibung
Name	identifiziert das Kontrollkästchen. Präfix ist **chk**.
Caption	Text neben dem Kontrollkästchen
Value	der Wert ist **0** bzw. **vbUnchecked** oder **1** bzw. **vbChecked**
Left, Top	bestimmen die Position innerhalb des Rahmenfeldes

Ereignisse

Das einzige interessante Ereignis für ein Kontrollkästchen ist das Click-Ereignis, es wechselt die Value-Eigenschaft des Kontrollkästchens!

Das Optionsfeld (option button)

Mit Optionsfeldern kann man aus einer Gruppe von Möglichkeiten genau eins auswählen. Optionsfelder werden immer in einem Rahmenfeld gruppiert.

Werkzeug:

Auf dem Formular:

Eigenschaften

Eigenschaft	Beschreibung
Name	identifiziert das Optionsfeld. Präfix ist **opt**.

Caption	Text neben dem Optionsfeld
Value	**True** bei ausgewähltem Optionsfeld, sonst **False**
Left, Top	bestimmen die Position innerhalb des Rahmenfeldes

Ein Optionsfeld in einer Gruppe von Optionsfelder sollte stets im Entwurfsmodus den Wert True zugewiesen haben!

Ereignisse

Das einzige interessante Ereignis für ein Optionsfeld ist das Click-Ereignis, es setzt die Value-Eigenschaft des Optionsfeldes auf True und bei den anderen Optionsfelder innerhalb der gleichen Gruppe auf False.

Visual Basic

Entscheidungen - Select Case Anweisung

Die **Select Case** Anweisung wird benutzt, um anhand der Werte einer einfachen Variablen oder eines Ausdrucks Entscheidungen auszuführen.

```
Select Case TestAusdruck
Case Vergleich1
    [BASIC-Code, falls Vergleich1 stimmt]
Case Vergleich2
    [BASIC-Code, falls Vergleich2 stimmt]
Case Vergleich3
    [BASIC-Code, falls Vergleich3 stimmt]
Case Vergleich4
    [BASIC-Code, falls Vergleich4 stimmt]
Case Else
    [BASIC-Code, falls kein Vergleich stimmt]
End Select
```

Als Testausdruck benutzen wir zunächst nur eine Variable vom Typ Integer, deren Wert wir in den Case-Anweisungen mit festgelegten Zahlen vergleichen.

Anm.: Bei der Formulierung der Vergleiche sind auch Bereichsprüfungen möglich, dazu dienen die Schlüsselwörter **Is** und **To**:
z.B. case 1 To 5 case Is > 4 Case 2 To 4, 8 To 12, Is < 2

Projekt - Hamburger Bestellservice

Pizza Taxi ist bekannt, aber warum nicht auch ein Bestellservice für Hamburger? Wir programmieren ein PC-Bestellsystem für Hamburger.

Projektentwurf

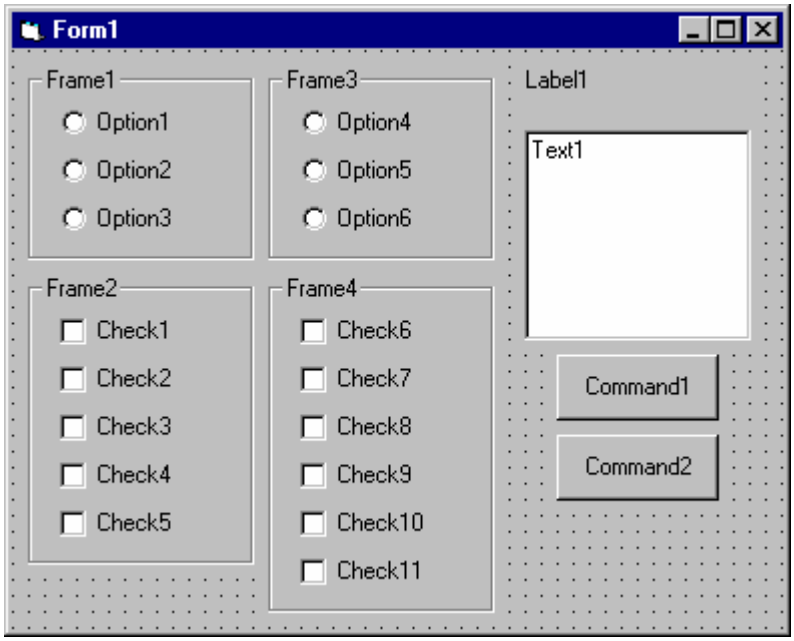
Welche Aufgaben sind zu lösen:

- Drei Brotsorten (eine möglich): Weissbrot, Weizen, Roggen
- Fünf Fleischsorten (mehrere möglich): Rind, Schinken, Puter, Spanferkel, Salami
- Drei Käsesorten (eine möglich): ohne, Gouda, Edamer
- Sechs Beilagen (mehrere möglich): Senf, Mayonnaise, Salat, Tomaten, Zwiebeln, Gurken

Steuerelemente

Wir brauchen eine Gruppe mit drei Otionsfeldern für Brot, eine Gruppe mit fünf Kontrollkästchen für Fleisch, eine Gruppe mit drei Optionsfeldern für Käse und eine Gruppe mit sechs Kontrollkästchen für Beilagen. Wir ergänzen noch eine Steuerschaltfläche zum Löschen der Menüauswahl, eine Steuerschaltfläche für die Zusammenstellung des Hamburgers und ein Textfeld (mit Bezeichnungsfeld), wo die Bestellung ausgegeben wird.

Anordnung der Steuerelemente



Festlegung der Eigenschaften

Steuerelement	Eigenschaft	Wert
Form1	Name	frmHamburger
	Caption	Hamburger Service
	Borderstyle	1 - Fixed Single
	Icon	Hamburger.ico

Frame1	Name	fraBrot
	Caption	Brot
	FontSize	10
	FontStyle	Bold
Option1	Name	optWeiss
	Caption	Weiss
	Value	True
Option2	Name	optWeizen
	Caption	Weizen
Option3	Name	optRoggen
	Caption	Roggen

Frame2	Name	fraFleisch
	Caption	Fleisch
	FontSize	10
	FontStyle	Bold
Check1	Name	chkRind
	Caption	Rindfleisch
Check2, Check3, Check4, Check5 entsprechend		

Entsprechend erfolgt die Festlegung der Eigenschaften für das 3. und 4. Rahmenfeld und die darin enthaltenen Steuerelemente.
Ferner ergänzen wir die Eigenschaften für das Bezeichnungsfeld **lblBestellung** und das Textfeld.

Text1	Name	txtBestellung
	MultiLine	True
	ScrollBars	2-Vertical
	Text	[kein]

Abschließend noch die beiden Steuerschaltflächen

Command1	Name	cmdBestellung
	Caption	Bestellung
Command2	Name	cmdLöschen
	Caption	Löschen

Programmierung

In diesem Projekt werden die Kontrollkästchen und Optionsfelder in den verschiedenen Rahmenfeldern dazu benutzt, die Zusammenstellung des Hamburgers festzulegen. Wenn die Auswahl erfolgt ist, wird **Bestellung** angekickt

und die gewünschte Zusammenstellung im Textfeld aufgelistet. Mit **Löschen** wird die Menütafel wieder auf ihre Ausgangswerte gesetzt.
Wir benötigen noch zwei Integer-Variablen: **BrotWahl** speichert die ausgewählte Brotsorte, **KäseWahl** speichert die ausgewählte Käsesorte. Dazu legen wir folgende Zuordnung fest:

Brotwahl		**Käsewahl**	
Wert	Brotsorte	Wert	Käsesorte
1	Weiss	o	kein
2	Weizen	1	Gouda
3	Roggen	2	Edamer

Die Verwendung von Variablen zum Speichern der Wahl bei Optionsfeldern ist gängig. Wir ergänzen noch zwei Variablen (**AnzahlFleisch** und **AnzahlBeilagen**), um zu zählen, wieviele Sorten jeweils ausgewählt wurden. Also

```
Option Explicit
Dim BrotWahl As Integer
Dim KäseWahl As Integer
Dim AnzahlFleisch as Integer
Dim AnzahlBeilagen As Integer
```

BrotWahl und KäseWahl müssen initialisiert werden, und zwar passend zu den Ausgangswerten der Optionsfelder: **optWeiss** (Weissbrot) hat den Wert True und **optKein** (kein Käse) hat den Wert True. Damit ergibt sich mit obiger Zuordnungstabelle für die Initialisierung mittels der **Form_Load** Prozedur

```
Private Sub Form_Load()
  'Initialiert Brot- und Käseauswahl
  BrotWahl = 1
  KäseWahl = 0
End Sub
```

Nachdem sämtliche Auswahlen an der Menütafel erfolgt sind, wird **Bestellung** angeklickt. An dieser Stelle muß der Computer nun Folgendes tun:

- Entscheiden, welche Brotsorte gewählt wurde.
- Entscheiden, ob und welche Fleischsorten gewählt wurden.
- Entscheiden, ob und welche Käsesorte gewählt wurde.
- Entscheiden, ob und welche Beilagen gewählt wurden.
- Bestellung in das Textfeld schreiben.

Dazu ein Beispiel:

```
Private Sub optWeiss_Click()
 'Weissbrot ausgewählt
 BreadChoice = 1
End Sub
```

Ähnlich ist die Codierung für die anderen Brotsorten und für die Käsewahl.
Damit ist die Auswahl an Brot und Käse bekannt, wenn **Bestellung** angeklickt wird. Die Auswahl an Fleisch und Beilagen ist jedoch nicht bekannt. Der einzige Weg, diese Auswahlen festzustellen, besteht darin, jeden einzelnen checkbox-Wert zu überprüfen, ob er gesetzt (**vbchecked**) wurde oder nicht (**vbunchecked**). Dies erfolgt in der Ereignisprozedur **cmdBestellung_Click**. Dort schreiben wir auch den Code für das Ausfüllen des Textfeldes mit der Bestellung. Diese Ausgabe erfolgt

dabei mittels der Text-Eigenschaft, und zwar durch schrittweises Zusammenbauen einer Zeichenkette mit allen Daten:

- Brotsorte in Text-Eigenschaft eintragen (mittels Select Case).
- Ersetze die Text-Eigenschaft durch den vorherigen Text plus einer Fleischsorte (mittels If/End If für jede Fleischsorte).
- Ersetze die Text-Eigenschaft durch den vorherigen Text plus eine Käsesorte (benutze Select Case).
- Ersetze die Text-Eigenschaft durch den vorherigen Text plus eine Beilage (mittels If/End If für jede Beilage).
- Text-Eigenschaft komplett.

Alle bestellten Bestandteile des Hamburgers sollen in separaten Zeilen aufgeführt werden, dazu können wir die symbolische Konstante **vbCrLf** benutzen, womit eine neue Zeile begonnen wird. Beispielweise:

```
txtBestellung.Text = "Hamburger Service:" & vbCrLf
txtBestellung.Text = txtBestellung.Text & "Weissbrot" & vbCrLf
```

Bleibt noch das **cmdLöschen_Click** Ereignis. Hier ein Auszug:

```
Private Sub cmdLöschen_Click()
'Weissbrot
BrotWahl = 1
optWeiss.Value = True
'Fleischauswahl löschen
chkRind.Value = vbUnchecked
...
'kein Käse
KäseWahl = 0
optkein.Value = True
'Beilagen löschen
chkSenf.Value = vbUnchecked
...
'Textfeld löschen
txtBestellung = ""
End Sub
```

Erweiterungen

1. Ergänze das Formular um eine Steuerschaltfläche Ende.
2. Ergänze das Programm so, daß in der Bestellung neben den ausgewählten Teilen auch der Preis sowie der Preis für den Hamburger ausgegeben wird. Dazu ein Tip: die formatierte Textausgabe mit der **Format-**Funktion von Visual Basic, z.B.

   ```
   txtBestellung.Text = txtBestellung.Text & Format(preis,
   "#.00DM")
   ```

Bildfelder, Mausereignisse und Farben

Das Bildfeld (picture box)

Das Bildfeld ist ein sehr vielseitig verwendbares Steuerelement. Damit kann man Bilder darzustellen, Steuerelemente einbetten, Grafikbefehle benutzen und Animationen ausführen.

Werkzeug	Auf dem Formular
	

Eigenschaften

Name	identifiziert das Bildfeld. Präfix ist **pic**.
Picture	legt die Grafikdatei fest, die im Bildfeld dargestellt wird.
AutoSize	Damit kann das Bildfeld der Grafik angepasst werden.
BackColor	Hintergrundfarbe des Bildfeldes
ForeColor	Farbe der Bildfeld-Grafik
DrawStyle	Linientyp für die Grafikmethoden zum Zeichnen
DrawWidth	Linienbreite (in twips)
FillColor	Farbe zum Ausfüllen von Figuren
FillStyle	Muster zum Ausfüllen von Figuren
BorderStyle	bestimmt den Randtyp des Bildfeldes
ScaleWidth	Breite des Bildfeldes für die Grafikmethoden
ScaleHeight	Höhe des Bildfeldes für die Grafikmethoden
Enabled	bestimmt, ob das Bildfeld und seine Inhalte auf Ereignisse reagieren können (bei Programmausführung)
Visible	bestimmt, ob das Bildfeld auf dem Formular sichtbar ist

Die Eigenschaft Picture

Das Bildfeld kann Bilddateien in verschiedenen Grafikformaten anzeigen:

Bezeichnung	Endung	Bemerkungen
Bitmap	bmp	Pixelformat
Icon	ico	Bitmap-Dateien; max. Größe: 32 x 32
Metafile	wmf	Vektorgrafik-Datei

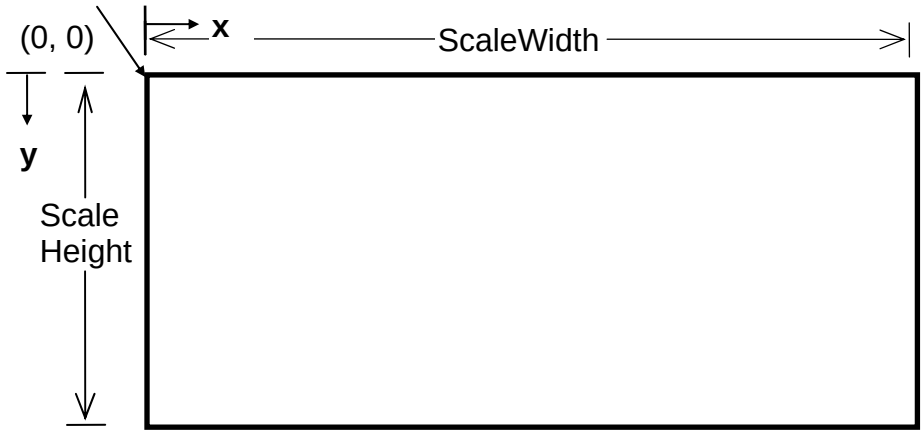
JPEG	jpg	komprimiertes Bitmap-Format (bis 16M. Farben)
GIF	gif	komprimiertes Bitmap-Format (bis 256 Farben)

Aufgabe: Erstelle ein Bildfeld und teste die Darstellung der verschiedenen Grafikformate. Du findest dazu verschiedene Grafikdateien in deinem Arbeitsverzeichnis. Beachte dabei die Unterschiede!
Bildfelder haben die gleiche Container-Eigenschaft wie Rahmenfelder, d.h. erstellt man Steuerelemente innerhalb eines Bildfeldes, so sind sie mit diesem Bildfeld fest verbunden und bilden eine Gruppe.
Welcher Unterschied besteht eigentlich zwischen den beiden Eigenschaften **Height** und **ScaleHeight**? Die Eigenschaft Height ist die Höhe des Bildfeldes (in twips); ScaleHeight ist die Höhe des Zeichenbereichs innerhalb des Bildfeldes. Entsprechendes gilt für Width und ScaleWidth.

Grafikmethoden
Bildfelder bieten Grafikmethoden. Eine Methode ist eine Prozedur oder Funktion, die einem Steuerelement Leben verleiht. In einem Bildfeld werden Grafikmethoden benutzt, um etwas zu zeichnen. Methoden können nur im Ausführungsmodus benutzt werden. In BASIC wird eine Methode wie folgt angesprochen:

```
SteuerelementName.MethodenName [optionale Argumente]
```

Alle Graphikmethoden verwenden ein Standard-Bildfeld Koordinatensystem.



Die PSet-Methode

PSet setzt einen Punkt mit den Koordinaten x, y in der gewünschten Farbe.

```
BildfeldName.PSet (x, y), Farbe
```

Die Line-Methode

Mit der Line-Methode kann man Linien und Rechtecke in vielen Varianten zeichnen. Untersuche dazu Beispiele mit folgender Syntax!

```
a) BildfeldName.Line (x1, y1) - (x2, y2), Farbe
b) BildfeldName.Line - (x3, y3), Farbe
c) BildfeldName.Line (x1, y1) - (x2, y2), Farbe, B
d) BildfeldName.Line (x1, y1) - (x2, y2), ,B
```

Die Circle-Methode

Mit der Circle-Methode kann man Kreise, Ellipsen, Bögen und Kreissegmente zeichnen.

```
BildfeldName.Circle (x, y), r, Farbe
```

Die Cls-Methode

Mit der Cls-Methode wird die gesamte Zeichnung aus dem Bildfeld gelöscht.

```
picBildfeldName.Cls
```

Im folgenden Projekt erstellen wir ein Malprogramm in Visual Basic. Dazu benötigen wir noch einige Basic-Kenntnisse im Umgang mit der Maus und einige Details zu Farben.

VISUAL BASIC

Mausereignisse

Die Maus ist das bevorzugte Eingabegerät zur Erzeugung von Grafiken in Visual Basic.

Das MouseDown-Ereignis

Das **MouseDown**-Ereignis wird immer dann ausgelöst, wenn eine Maustaste gedrückt wird während sich der Mauszeiger über einem Steuerelement befindet. Die Prozedur hat folgenden Aufbau:

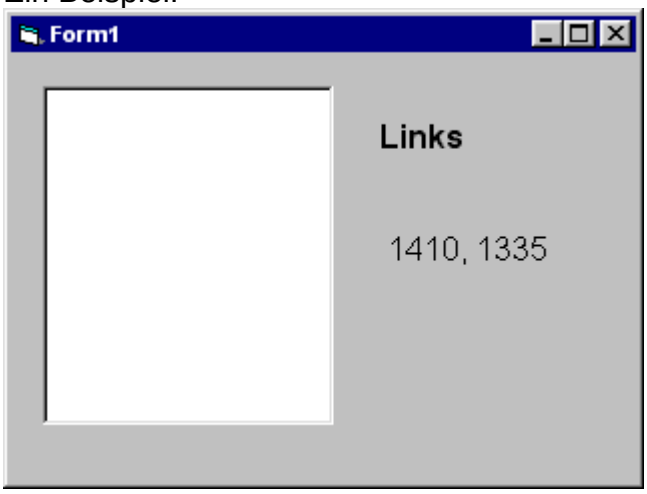
```
Private Sub SteuerelementName_MouseDown(Button As
Integer, Shift As Integer, X As Single, Y As Single)
   [BASIC-Code für das MouseDown-Ereignis]
End Sub
```

Button, **Shift**, **X** und **Y** sind Argumente mit folgender Bedeutung. Button ist ein Integer-Wert, der die Maustaste kennzeichnet, die das Ereignis verursachte. Die Werte sind als symbolische Konstanten verfügbar**: vbLeftButton, vbMiddleButton, vBRightButton**.

Shift ist ebenfalls ein Integer-Wert, der den Zustand der Tasten Shift, Strg und Alt angibt.

Die Werte X und Y (Datentyp single) geben die Koordinaten des Mauszeigers im Steuerelement an, als es angeklickt wurde.

Ein Beispiel:



Wir erstellen ein Formular mit dem Bildfeld **picBeispiel** und zwei Bezeichnungsfelder mit den Namen **lblMaustaste** und **lblKoordinaten**, in denen die benutzte Maustaste und die Koordinaten beim Mausklick angezeigt werden.

```
Private Sub picBeispiel_MouseDown(Button As Integer,
Shift As Integer, X As Single, Y As Single)
 Select Case Button
  Case vbLeftButton
   lblMaustaste.Caption = "Links"
  Case vbMiddleButton
   lblMaustaste.Caption = "Mitte"
  Case vbRightButton
   lblMaustaste.Caption = "Rechts"
 End Select
 lblKoordinaten.Caption = Str(X) & "," & Str(Y)
End Sub
```

Das MouseUp-Ereignis

Das MouseUp-Ereignis ist das Gegenstück zum MouseDown-Ereignis. Es wird immer dann ausgelöst, wenn eine zuvor gedrückte Maustaste losgelassen wird.

```
Private Sub SteuerelementName_MouseUp(Button As Integer,
Shift As Integer, X As Single, Y As Single)
  [BASIC-Code für MouseUp-Ereignis]
End Sub
```

Das MouseMove-Ereignis

Das MouseMove-Ereignis wird kontinuierlich ausgelöst, wenn die Maus bewegt wird.

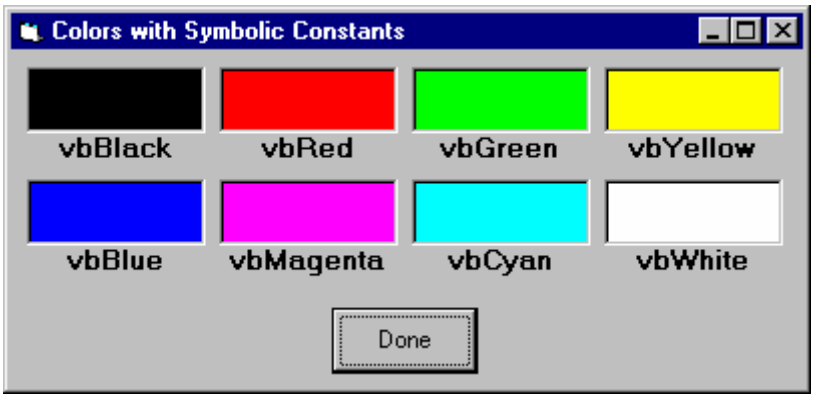
```
Private Sub SteuerelementName_MouseMove(Button As
Integer, Shift As Integer, X As Single, Y As Single)
   [BASIC-Code für MouseMove-Ereignis]
End Sub
```

Farben

Visual Basic bietet uns drei Wege an, Farbwerte anzugeben: symbolische Konstanten, die QBColor-Funktion und die RGB-Funktion

symbolische Konstanten

vbBlack	vbRed	vbGreen	vbYellow	vbBlue	vbMagenta	vbCyan	vbWhite
Schwarz	Rot	Grün	Gelb	Blau	Magenta	Cyan	Weiss



QBColor-Funktion

Die QBColor-Funktion liefert 16 Farben. Dies sind die Standardfarben einer VGA-Grafikkarte, wie sie auch bei QBasic unter DOS benutzt werden.

`QBColor(Index)`

Index	Farbe	Index	Farbe
0	Schwarz	8	Grau
1	Blau	9	Hellblau
2	Grün	10	Hellgrün
3	Cyan	11	Hellcyan
4	Rot	12	Hellrot
5	Magenta	13	Hellmagenta
6	Braun	14	Gelb
7	Weiss	15	Weiss

Beispiel:

Die RGB-Funktion

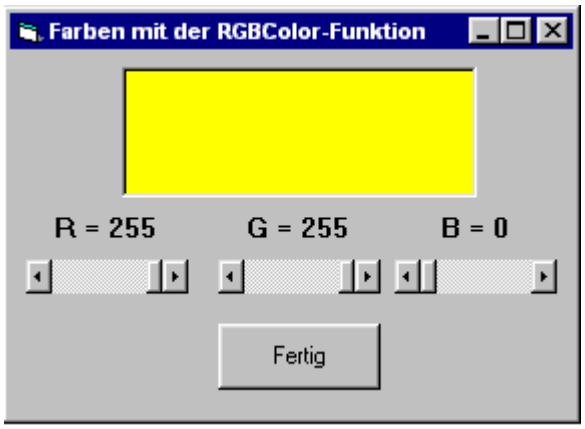
Mit der RGB-Funktion kann man über 16 Millionen Farben erzeugen. Die Syntax:

`RGB(Rot, Grün, Blau)`

wobei **Rot**, **Grün** und **Blau** Zahlenwerte von 0 bis 255 sind, mit denen die Intensität der Farbe festgelegt wird.

Anm.: 256 x 256 x 256 = 16.777.216

Beispiel: Untersuche diese Farbfunktion mit einem kleinen Testprojekt.

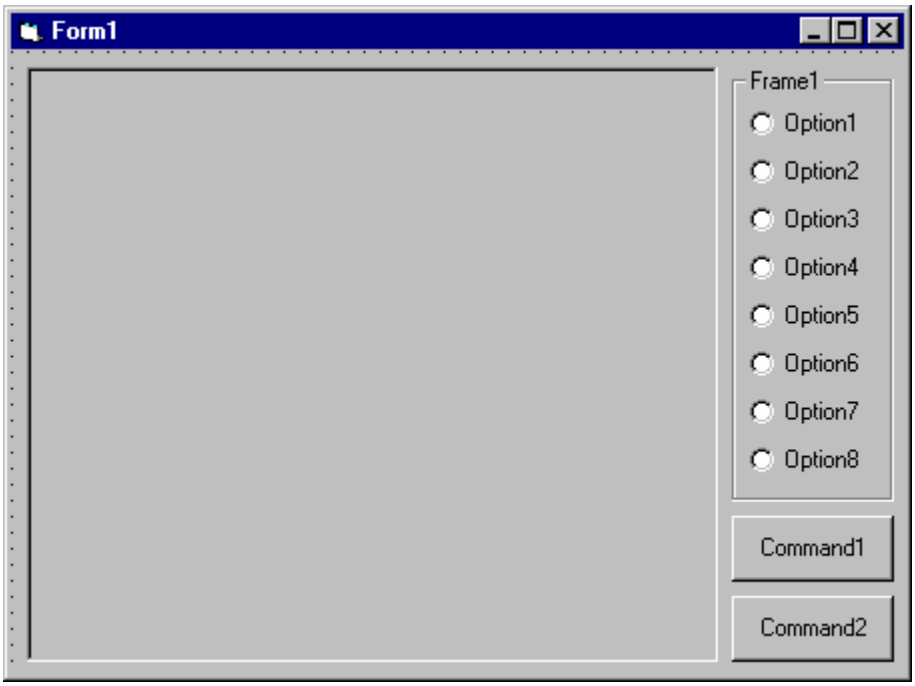


Projekt - Spass an der Wandtafel

Projektentwurf

Der Entwurf dieses Projekts ist einfach: mit der Maus zeichnen wir farbige Linien auf einer Computer-Wandtafel. Ein Bildfeld stellt die Tafel dar. Mittels Optionsfelder wählen wir die Kreidefarbe aus. Das Malen wird mittels Mausereignisse gesteurt. Zwei Steuerschaltflächen werden noch benutzt: einen zum Auswischen der Tafel und eine, um das Programm zu beenden.

Anordnung der Steuerelemente auf dem Formular



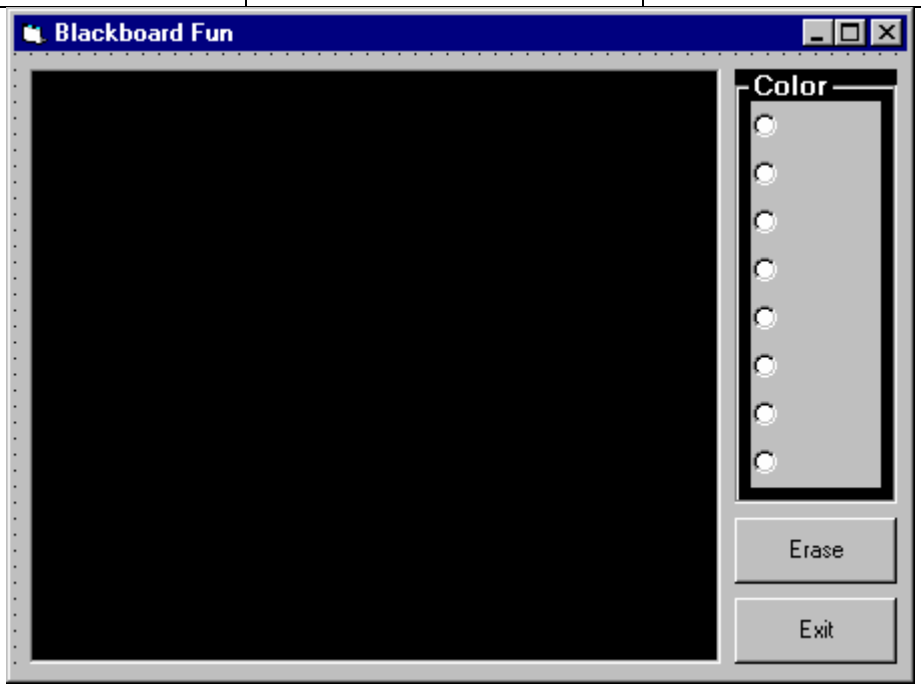
Eigenschaften der Steuerelemente

Form1	Eigenschaftsname	EigenschaftWert
	Name	frmTafel

	Caption	Spass an der Tafel
	BorderStyle	1-Fixed Single
Picture1	Name	picTafel
	BackColor	Black
Frame1	Name	fraColor
	Caption	Farbe
	BackColor	Black
	ForeColor	White
	FontSize	10
	FontStyle	Bold

Option1	Name	optGrau
	Caption	[kein]
Option2	Name	optBlau
	Caption	[kein]
und so weiter für die anderen 6 Farben		

Command1	Name	cmdLöschen
	Caption	Löschen
Command2	Name	cmdEnde
	Caption	Ende



Variablen und Ereignisprozeduren

Variablendeklaration:
MousePress (Datentyp Boolean): ihr Wert ist True, wenn die linke Maustaste gedrückt wird.
DrawColor (Datentyp Integer) speichert die Nummer der aktuellen **QBColor**.

```
Option Explicit
Dim MousePress As Boolean
Dim DrawColor As Integer
```

Bei der Farbwahl machen wir uns die **BackColor**-Eigenschaft der Optionsfelder zunutze, so daß der Benutzer die ausgewählte Farbe im Optionsfeld sehen kann. Die Startfarbe ist Hellgrau, daher setzen wir **DrawColor** auf **8** und den Wert von **optGrau** auf **True**. Dies erfolgt in der **Form_Load** Prozedur:

```
Private Sub Form_Load()
 'Initialisiere die acht Farb-Optionsfelder
 optGrau.BackColor = QBColor(8)
 optBlau.BackColor = QBColor(9)
 optGrün.BackColor = QBColor(10)
 optCyan.BackColor = QBColor(11)
 optRot.BackColor = QBColor(12)
 optMagenta.BackColor = QBColor(13)
 optGelb.BackColor = QBColor(14)
 optWeiss.BackColor = QBColor(15)
 'Initialisiere die Zeichenfarbe auf Grau
 optGrau.Value = True
 DrawColor = 8
End Sub
```

Jedes Otptionsfeld benötigt eine Click-Ereignis Prozedur, um den Wert von **DrawColor** entsprechend zu setzen. Also z.B.

```
Private Sub optBlau_Click()
 'Blau QBColor
 DrawColor = 9
End Sub
```

Die beiden Ereignisprozeduren für die Steuerschaltflächen:

```
Private Sub cmdLöschen_Click()
 'Löscht die Tafel
 picTafel.Cls
End Sub
Private Sub cmdEnde_Click()
 'Beendet das Programm
 End
End Sub
```

Für den Ablauf des eigentlichen Malens sind drei Ereignisse zu betrachten

- Klick auf die linke Maustaste - startet das Malen
- Ziehen der Maus bei gedrückter linker Maustaste - Malen fortsetzen
- Loslassen der linken Maustaste - beendet das Malen

```
Private SUB picTafel_MouseDown(Button As Integer, Shift
As Integer, X As Single, Y As Single)
'Startet das Malen, wenn Mausklick links
IF Button = vbLeftButton THEN
  MousePress = True
  picTafel.PSet (X, Y), QBColor(DrawColor)
End IF
End SUB

Private SUB picTafel_MouseMove(Button As Integer, Shift
As Integer, X As Single, Y As Single)
'Malen fortsetzen
IF MousePress = True THEN
  picTafel.Line -(X, Y), QBColor(DrawColor)
End IF
End SUB

Private SUB picTafel_MouseUp(Button As Integer, Shift As
Integer, X As Single, Y As Single)
'Stoppt das Malen
IF Button = vbLeftButton THEN
  MousePress = False
End IF
End SUB
```

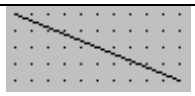
Ergänzungen

1. Ergänze ein Optionsfeld, um die **DrawWidth** Eigenschaft des Bildfeldes zu setzen. Die Eigenschaft **DrawStyle** lohnt sich auch.
2. Ergänze die Möglichkeit, ausgefüllte Rechtecke zu zeichnen. Benutze dazu das **MouseDown**-Ereignis, um die linke obere Ecke zu speichern, und entsprechend **MouseUp** für die rechte untere Ecke. Oder nutze das **MouseMove**-Ereignis! Lass dem Benutzer die Auswahl zwischen **FillColor** and **FillStyle**.
3. Man kann auch Effekte einbauen. Deklariere zwei Variablen **Xs** und **Ys**. Ergänze dann in der Prozedur **picTafel_MouseDown** vor der PSet-Methode:
   `Xs = X und Ys = Y`
   Ersetze noch den Code für das Zeichnen einer Linie in **picTafel_MouseMove** durch:
   ```
   picTafel.Line (Xs, Ys) -(X, Y), QBColor(DrawColor)
   ```

Linien, Figuren, Anzeige, Felder

Linienelement (line control)

Das Steuerelement **line** erzeugt eine einfache gerade Linie mit einstellbarer Breite und Farbe.

Werkzeug	Auf dem Formular:
	

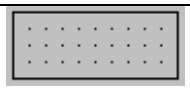
Eigenschaften

Name	identifiziert das Linienelement. Präfix: **lin**
BorderColor	setzt die Linienfarbe
BorderStyle	legt den Linientyp fest
BorderWidth	Breite in twips
X1,Y1,X2,Y2	Positionierung auf dem Formular
Visible	bestimmt, ob die Linie sichtbar ist

Das Linienelement dient nur zur optischen Aufteilung, z.B. um ein Tic-Tac-Toe Spiel oder ein Schachspiel auf dem Formular aufzubauen.

Figurenelement (shape control)

Figuren (**shapes**) sind Kreise, Ovale, Rechtecke, abgerundete Rechtecke und Quadrate, die farbig gestaltet werden können, und einfach nur als Hintergrund auf dem Formular erscheinen.

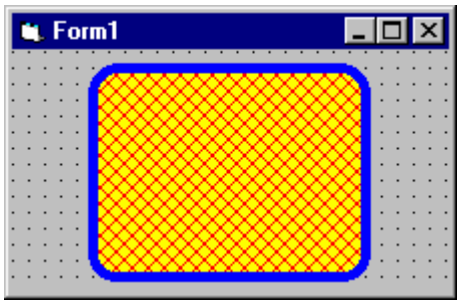
Werkzeug	Auf dem Formular
	

Eigenschaften

Name	identifiziert die Figur. Präfix: **shp**
BackColor	Hintergrundfarbe der Figur
BackStyle	Hintergrund transparent oder opaque
BorderColor	Randfarbe der Figur
BorderStyle	Randtyp: transparent, solid, dashed, dotted
BorderWidth	Breite des Figurrandes (in twips)
FillColor	Füllfarbe der Figur
FillStyle	Füllmuster

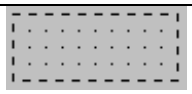
Shape	bestimmt den Figurtyp
Left	Abstand linker Rand zum linken Formularrand
Top	entsprechend
Width	Breite der Figur in twips
Height	Höhe der Figur in twips
Visible	bestimmt, ob die Figur sichtbar ist

Übung:



Die Anzeige (image control)

Die Anzeige (**image**) macht nur Eines: sie stellt Grafikdateien dar.

Werkzeug	Auf dem Formular:
	

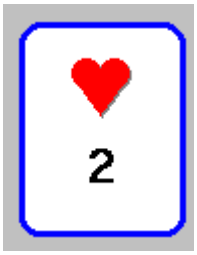
Eigenschaften

Name	identifiziert die Anzeige. Präfix: **img**
Picture	angezeigte Grafikdatei
Stretch	Größenanpassung (**True**) an die Anzeige, oder nicht (**False**)
BorderStyle	Rahmentyp der Anzeige
Left, Top, Height, Enabled, Visible wie sonst	

Ereignisse

Die wichtigsten Ereignisse für die Anzeige sind **Click, MouseDown, MouseUp**.

Noch ein Tip für die Ausgestaltung von Formularen: Formulare sind mit der **Layertechnik** ausgestattet wie sie von Zeichenprogrammen her bekannt ist, d.h. Steuerelemente können in Schichten übereinander angeordnet werden. Damit kann man gute Darstellungen erzielen, z.B. diese Spielkarte:



Visual BASIC

Felder (arrays)

BASIC bietet die Möglichkeit, eine große Zahl an Variablen unter dem gleichen Namen zu speichern - Feldvariablen (array). Die Deklaration ist einfach:

```
Dim Schüler(300) As String
Dim Punkte(10) As Integer
```

Die Zahl in Klammern nennt man die Feld**dimension**.
Jedes **Element** in einer Feldvariablen wird durch den Feldnamen und einen Index angesprochen.

```
Schüler(150) = "Micro Soft"
Punkte(2) = 100
Summe = Punkte(1) + Punkte(2) + Punkte(3)
```

For-Next-Schleife

Die For-Next-Schleife ist hervorragend geeignet für die Arbeit mit Feldern.

```
For Zähler = StartWert To EndWert Step Schrittweite
  [BASIC-Code, der ausgeführt werden soll]
Next Zähler
```

Lokale Variablen einer Prozedur

Variablen, die im allgemeinen Deklarationsteil aufgeführt werden, besitzen globale Gültigkeit für alle Prozeduren (**globale Variablen**). Variablen, die unmittelbar nach der Kopfzeile einer Prozedur deklariert werden, sind nur innerhalb dieser Prozedur gültig (**lokale Variablen**).

Ein Mischverfahren

Wir werden im Folgenden einen Stapel von forlaufend nummerierten Karten mischen, d.h. eine zufällige Anordnung erzeugen. Jeder kennt ein Verfahren, um Karten mit der Hand zu mischen. Aber wie machen wir das in Basic?
Wir benutzen ein Verfahren, daß sich einfach programmieren läßt: Wir ziehen zufällig eine Karte aus dem Stapel und legen sie als letzte Karte in den Stapel. Mit den verbleibenden Karten im Stapel verfahren wir genauso, bis alle Karten bearbeitet sind. So geht's bei N Zahlen bzw. Karten:

- Starte mit einem Feld von N aufeinanderfolgenden Zahlen ganzen Zahlen.
- Wähle zufällig ein Element aus dem Feld. Vertausche dieses mit dem Letzten im Feld. Damit hast du noch ein Restfeld mit N-1 Elementen zu bearbeiten.

- Wähle zufällig ein Element aus dem Restfeld. Vertausche es mit dem letzten Element im Restfeld. Wieder hat sich die Zahl der verbliebenen Elemente um eins verringert.
- Wiederhole das Entfernen eines Elements und Vertauschen mit dem Letzten des Restfeldes bis keine Elemente mehr da sind. Dann enthält das Feld alle Zahlen in zufälliger Anordnung.

1	**2**	**3**	**4**	**5**	Startliste
1	**2**	**5**	**4**	**3**	3 zufällig gezogen
1	**2**	**5**	**4**	**3**	4 zufällig gezogen
5	**2**	**1**	**4**	**3**	1 zufällig gezogen
2	**5**	**1**	**4**	**3**	5 zufällig gezogen

In Basic kann man den Wert von zwei Variablen mit dem sog. Ringtausch wie folgt vertauschen:

```
Hilfsvariable = Variable1
Vriable1 = Variable2
Variable2 = Hilfsvariable
```

Nun brauchen wir nur noch die Mischroutine, um aus einer sortierten Liste von Zahlen eine zufällige Anordnung zu erzeugen.
globale Variablen:

```
Dim ZahlenFeld(52) As Integer
Dim ElementAnzahl As Integer
```

Hier die Basic-Codierung:

```
'Variablendeklaration (zu Beginn der Prozedur)
Dim TempWert As Integer
Dim Zaehler As Integer
Dim FeldElement As Integer
Dim Rest As Integer
'Initialisiere Zahlenfeld
For Zaehler = 1 to ElementAnzahl
  Zahlenfeld(Zaehler) = Zaehler
Next Zaehler
'restliche Werte abarbeiten
'starte bei ElementAnzahl und vertausche einen Wert
'bei jedem Durchlauf
'nach jedem Durchlauf ist Rest um 1 kleiner
For Rest = ElementAnzahl to 2 Step -1
  'FeldElement zufällig auswählen
  FeldElement = Int(Rnd * Rest) + 1
  'FeldElement und letztes Element tauschen
  TempWert = Zahlenfeld(Rest)
  Zahlenfeld(Rest) = Zahlenfeld(FeldElement)
  Zahlenfeld(FeldElement) = TempWert
Next Rest
```

Projekt - Kartenduell

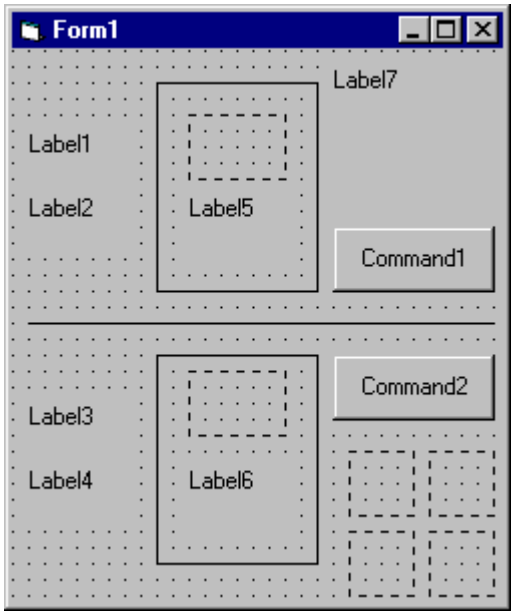
Kartenduell

Bei diesem Projekt spielst du ein Kartenspiel gegen den Computer. Jeder bekommt den halben Kartenstapel (26 Karten). Jeder deckt eine Karte auf; derjenige mit der höherwertigen Karte gewinnt die Karte des Gegners. Gewonnen hat derjenige, der am Ende die meisten Karten hat.

Projektentwurf

Wir verwenden Figurenelemente, um die Umrisse der Spielerkarten darzustellen. Ein Bezeichnungsfeld zeigt den Kartenwert und eine Anzeige zeigt die Kartenfarbe. Eine Steuerschaltfläche steuert den Beginn eines neuen Spiels oder das Ziehen einer neuen Karte, abhängig von Spielsituation. Eine weitere Steuerschaltfläche dient zum Beenden eines Spiels während ein Spiel läuft oder zum Beenden des Programms, wenn ein Spiel beendet ist. Der aktuelle Punktestand wird in einem Bezeichnungsfeld angezeigt.

Aufbau des Formulars



Eigenschaften der Steuerelemente

Form1	Name	frmKartenduell
	Caption	Kartenduell
	BorderStyle	1-Fixed Single

Shape1	Name	shpSpieler
	BackColor	White
	BackStyle	1-Opaque
	BorderColor	Blue
	BorderWidth	3
	Shape	4-Rounded Rectangle
Shape2	Name	shpComputer

	weitere Eigenschaften wie bei Shape1

Image1	Name	imgSpieler
	Stretch	True
Image2	Name	imgComputer
	Stretch	True
Image3	Name	imgHerz
	Picture	Heart.ico
	Stretch	True
	Visible	False
Image4	Name	imgKaro
	Picture	Diamond.ico
	Stretch	True
	Visible	False
Image5	Name	imgKreuz
	Picture	Club.ico
	Stretch	True
	Visible	False
Image6	Name	imgPik
	Picture	Spade.ico
	Stretch	True
	Visible	False

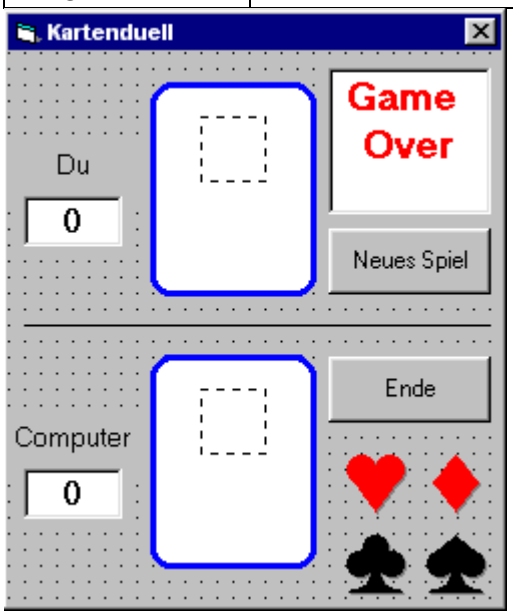
Die Eigenschaft Visible ist mit Absicht **False**. Die Bilder sollen zur Laufzeit nicht auf dem Formular erscheinen, sondern die Bilder werden hier nur als Vorrat für die Darstellung auf den Spielkarten abgelegt.

Label1	Name	lblDu
	Caption	Du
	FontSize	10
	Alignment	2-Center
Label2	Name	lblDeinePunkte
	Caption	0

	FontSize	12
	FontStyle	Bold
	Alignment	2-Center
	BackColor	White
	BorderStyle	1-Fixed Single
Label5	Name	lblSpieler
	Caption	[Blank]
	FontSize	18
	FontStyle	Bold
	Alignment	2-Center
	BackColor	White

Label3 (lblComp), **Label4** (lblCompPunkte), **Label6** (lblComputer) entsprechend für den Computer.

Label7	Name	lblGameOver
	Caption	Game Over
	FontSize	14
	FontStyle	Bold
	Alignment	2-Center
	BackColor	White
	ForeColor	Red
	BorderStyle	1-Fixed Single
Command1	Name	cmdNeu
	Caption	Neues Spiel
Command2	Name	cmdEnde
	Caption	Ende
Line1	Name	linKarte



Programmierung

Die Idee des Spiels ist einfach. Zum Start klickst du auf **Neues Spiel**. Die Karten werden gemischt, die Punkte auf Null gesetzt und die Aufschrift des Startbuttons wechselt zu "Nächste Karte". Der Ende-Button ändert auch seine Beschriftung in "Stopp". Eine Karte für dich und eine für den Computer werden angezeigt und ihre Werte verglichen. Der Spieler mit der höheren Karte bekommt zwei Punkte. Bei Unentschieden bekommt jeder Spieler einen Punkt. Die Punktstände werden angezeigt. **Nächste Karte** anklicken. Für jeden Spieler werden die nächsten Karten angezeigt und die Punktstände aktualisiert. Und so weiter, bis jeder Spieler 26 Karten erhalten hat. Nun wird die Mitteilung "Game over" angezeigt und die Werte der Steuerschaltflächen werden auf ihre ursprünglichen Werte zurückgesetzt. Anhand der Spielstände kann der Gewinner abgelesen werden. Das Spiel läßt sich vorzeitig durch einen Klick auf **Stopp** beenden.
Wir benötigen nur zwei Ereignisprozeduren: **cmdNeu_Click** und **cmdEnde_Click** und folgende globalen Variablen.

```
Option Explicit
Dim KartenNummer(52) As Integer
Dim KartenIndex As Integer
```

Natürlich müssen wir für das Spiel den Zufallsgenerator initialisieren:

```
Private Sub Form_Load()
Randomize
End Sub
```

Nun zu den Schritten, die durch die Ereignisprozedur **cmdNeu_Click** erledigt werden. Die Steuerschaltfläche hat zwei Aufgaben: sie startet ein neues Spiel oder holt eine neue Karte. Das Klick-Ereignis besteht also aus zwei Abschnitten.
Schritte, wenn "Neues Spiel" angezeigt wird:

- ‘Game Over’-Hinweis verstecken
- cmdNeu Caption auf “Nächste Karte”
- cmdEnde Caption auf “Stopp”
- Punktstände auf 0
- Karten mischen
- KartenIndex auf 1 setzen
- erste Karte für jeden Spieler anzeigen
- Karten vergleichen - Punktestände aktualisieren

Schritte, wenn "Nächste Karte" angezeigt wird:

- Karten anzeigen
- Karten vergleichen - Punktestände aktualisieren
- KartenIndex um 1 erhöhen
- Wenn keine Karten mehr da, Spiel beenden, ‘Game Over’ Mitteilung, Caption-Eigenschaft wechseln. Andernfalls, warte auf Klick des "Nächste Karte"-Buttons.

Die meisten Schritte lassen sich einfach programmieren. Der einzige knifflige Schritt ist die Darstellung und der Vergleich der Karten. Dazu benötigen wir eine Zuordnungstabelle, die jeder Karte eindeutig eine Zahl, die Kartennummer, zuordnet.

Zuordnung von Kartennummern

Herz	Karo	Kreuz	Pik	Anzeige	Vergleichswert
1	14	27	40	2	1
2	15	28	41	3	2
3	16	29	42	4	3
4	17	30	43	5	4
5	18	31	44	6	5
6	19	32	45	7	6
7	20	33	46	8	7
8	21	34	47	9	8
9	22	35	48	10	9
10	23	36	49	Bube (J)	10
11	24	37	50	Dame (Q)	11
12	25	38	51	König (K)	12
13	26	39	52	As (A)	13

Die Kartenvergleiche müssen anhand dieser Kartennummern erfolgen!

Der BASIC-Code zur Ereignisprozedur **cmdNeu_Click** :

```
Private Sub cmdNeu_Click()
'lokale Variablen
Dim TempWert As Integer
Dim Zaehler As Integer
Dim Zufallszahl As Integer
Dim Rest As Integer
Dim DeineZahl As Integer ' Deine Kartennummer
Dim ComputerZahl As Integer ' Computer Kartennummer
If cmdNeu.Caption = "Neues Spiel" Then
  'Neues Spiel angeclickt
  lblGameOver.Visible = False
  cmdNeu.Caption = "Nächste Karte"
  cmdEnde.Caption = "Stopp"
  'Punktzahlen auf Null
  lblDeinePunkte.Caption = "0"
  lblCompPunkte.Caption = "0"
  'Karten mischen
  'Initialisiere KartenNummer
  For Zaehler = 1 To 52
    KartenNummer(Zaehler) = Zaehler
  Next Zaehler
  'restliche Werte durcharbeiten
  'Beginne bei 52 und überspringe einen Wert
  'bei jedem For/Next-Durchlauf
  'Nach jedem Durchlauf ist Rest um 1 kleiner
  For Rest = 52 To 2 Step -1
    'zufällig eins herausgreifen
    Zufallszahl = Int(Rnd * Rest) + 1
    'Vertausche dies zugehörige Karte mit der untersten
    TempWert = KartenNummer(Rest)
    KartenNummer(Rest) = KartenNummer(Zufallszahl)
    KartenNummer(Zufallszahl) = TempWert
```

```
  Next Rest
  'Setze KartenIndex auf 1
  KartenIndex = 1
End If
'Zeige Karten, falls weitergespielt wird
If KartenIndex <= 26 Then
  'Zeige deine Kartenfarbe
  'Bestimme deine Kartennummer für den Vergleich
  Select Case KartenNummer(KartenIndex)
  Case 1 To 13
    imgSpieler.Picture = imgHerz.Picture
    DeineZahl = KartenNummer(KartenIndex)
  Case 14 To 26
    imgSpieler.Picture = imgKaro.Picture
    DeineZahl = KartenNummer(KartenIndex) - 13
  Case 27 To 39
    imgSpieler.Picture = imgKreuz.Picture
    DeineZahl = KartenNummer(KartenIndex) - 26
  Case 40 To 52
    imgSpieler.Picture = imgPik.Picture
    DeineZahl = KartenNummer(KartenIndex) - 39
  End Select
  'Zeige den Wert deiner Karte
  Select Case DeineZahl
  Case 1 To 9
    lblSpieler.Caption = Str(DeineZahl + 1) + " "
  Case 10
    lblSpieler.Caption = "J"
  Case 11
    lblSpieler.Caption = "Q"
  Case 12
    lblSpieler.Caption = "K"
  Case 13
    lblSpieler.Caption = "A"
  End Select
  'Farbe der Computerkarte anzeigen
  'Nummer dazu ermitteln für den Vergleich
  Select Case KartenNummer(KartenIndex + 26)
  Case 1 To 13
    imgComputer.Picture = imgHerz.Picture
    ComputerZahl = KartenNummer(KartenIndex + 26)
  Case 14 To 26
    imgComputer.Picture = imgKaro.Picture
    ComputerZahl = KartenNummer(KartenIndex + 26) - 13
  Case 27 To 39
    imgComputer.Picture = imgKreuz.Picture
    ComputerZahl = KartenNummer(KartenIndex + 26) - 26
  Case 40 To 52
    imgComputer.Picture = imgPik.Picture
    ComputerZahl = KartenNummer(KartenIndex + 26) - 39
  End Select
  'Kartenfarbe für ComputerKarte anzeigen
  Select Case ComputerZahl
  Case 1 To 9
    lblComputer.Caption = Str(ComputerZahl + 1) + " "
  Case 10
    lblComputer.Caption = "J"
  Case 11
    lblComputer.Caption = "Q"
  Case 12
    lblComputer.Caption = "K"
  Case 13
    lblComputer.Caption = "A"
  End Select
  'Vergleiche die angezeigten Karten
  If DeineZahl > ComputerZahl Then
```

```
        'Du gewinnst
        lblDeinePunkte.Caption = Str(Val(lblDeinePunkte.Caption) + 2)
      ElseIf ComputerZahl > DeineZahl Then
        'Computer gewinnt
        lblCompPunkte.Caption = Str(Val(lblCompPunkte.Caption) + 2)
      Else
        'Unentschieden
        lblDeinePunkte.Caption = Str(Val(lblDeinePunkte.Caption) + 1)
        lblCompPunkte.Caption = Str(Val(lblCompPunkte.Caption) + 1)
      End If
      KartenIndex = KartenIndex + 1
    Else
      'Game over
      lblGameOver.Visible = True
      cmdNeu.Caption = "Neues Spiel"
      cmdEnde.Caption = "Ende"
    End If
    End Sub
```

Die Ereignisprozedur **cmdEnde_Click** ist dagegen einfach:

```
Private Sub cmdEnde_Click()
If cmdEnde.Caption = "Ende" Then
  'Programmende
  End
Else
  'Spiel beenden
  lblGameOver.Visible = True
  cmdEnde.Caption = "Ende"
  cmdNeu.Caption = "Neues Spiel"
End If
End Sub
```

Erweiterungen

1. Mögliche Erweiterungen des Spiels sind naheliegend, aber nicht ganz einfach.
2. Es können mehr als zwei Spieler teilnehmen.
3. Nach einer Spielrunde werden die gewonnenen Karten für ein neues Spiel benutzt. Das Spiel geht so lange bis ein Spieler keine Karten mehr hat.
4. Bei gleichwertigen Karten (unentschieden) kommt es zum Duell: jeder Spieler zieht drei weitere Karten, die nicht gezeigt werden. Dann wird eine weitere Karte aufgedeckt und verglichen, der Gewinner bekommt alle 10 Karten des Duells.

Ergänzung:

Grobgliederung der Prozedur cmdNeu_Click()

Wenn das Spiel neu begonnen wird (Caption = "Neues Spiel"), sind folgende Vorbereitungen zu treffen:

- Caption auf "Nächste Karte" umstellen
- 'Game Over'-Hinweis nicht sichtbar
- Caption von cmdEnde auf "Stop" umstellen"
- Punktestände auf 0 setzen
- Karten mischen
- Kartenzähler auf 1 setzen

Wenn das Spiel weitergeführt wird (Caption = "Nächste Karte"), ist folgender Ablauf gegeben:

- Sind noch Karten da?
  Wenn ja, dann
  - Karten für beide aufdecken
  - Karten vergleichen
  - Punktestände aktualisieren
  - Kartenzähler um 1 erhöhen

  Wenn nein, dann
  - Spiel beenden
  - 'Game Over'-Mitteilung
  - Caption ändern

Weitere Überlegungen:

1. Jede Karte wird im Kartenfeld durch eine Zahl von 1 bis 52 nummeriert.
2. Mittels der Zuordnungstabelle wird die Farbe und der Wert einer Karte identifiziert.
3. Die Karten, d.h. das Kartenfeld, werden gemischt.
4. Die Karten werden anhand eines Vergleichswertes miteinander verglichen.

Spieler						Computer						
1.	2.	3.	4.			27.	28.	29.	30.			
47	5	20	31			4	37	50	23			

Dabei ergibt sich für die ersten drei Spielrunden

	Spieler			Computer	
Kartennummer	47			4	
Anzeige		**9**			**5**
Vergleichswert	47-39=8		>	4	
Kartennummer	5			37	
Anzeige		**6**			**Q**
Vergleichswert	5		<	37-26=11	
Kartennummer	20			50	
Anzeige		**8**			**Q**
Vergleichswert	20-13=7		<	50-39=11	

Zeitgeber, Animation, Tastaturereignisse

Der Zeitgeber (timer control)

Der Zeitgeber ist ein interessantes Steuerelement: es erzeugt Ereignisse ohne Benutzereingaben. Es arbeitet im Hintergrund und erzeugt in einstellbaren Zeitintervallen Ereignisse. Das ist besonders praktisch für Grafikanimationen, da hierbei die Bildschirmdarstellung in regelmäßigen Abständen aktualisiert werden muß.

Werkzeug: 

Auf dem Formular: 

Eigenschaften

Name	identifiziert den Zeitgeber. Präfix: **tim**
Interval	Millisekunden zwischen den Zeitgeber-Ereignissen
Enabled	schaltet den Zeitgeber ein oder aus

Ereignisse

Timer	Ereignisprozedur, die nach jedem Zeitintervall ausgeführt wird, sofern Enabled auf True gesetzt ist

Beispiel: Für einen Zeitgeber mit Namen **timBeispiel** lautet die Ereignisprozedur einfach:

```
Private Sub timBeispiel_Timer()
[BASIC-Code, der nach jedem Intervall ausgeführt wird]
End Sub
```

Übung1:
Erstelle ein Formular mit einem Zeitgeber **timZeit1** und einer Steuerschaltfläche **cmdButton**. Wir erstellen dann folgende Ereignisprozeduren:

```
Private Sub timZeit1_Timer()
Beep
Form1.BackColor = QBColor(Int(Rnd * 16))
End Sub


Private Sub cmdButton_Click()
If timZeit1.Enabled = True Then
  timZeit1.Enabled = False
Else
  timZeit1.Enabled = True
End If
End Sub
```

Untersuche die Funktionsweise!

Übung2: Nun soll der Computer jede Sekunde piepsen, die Farbe des Formulars aber viermal je Sekunde wechseln.

Wir löschen den Farbwechsel beim **timZeit1_Timer** Ereignis und ergänzen einen zweiten Zeitgeber (**timZeit2**) zum Formular. Setze dabei die Eigenschaft Enabled auf False und Interval auf 250. Erstelle die Ereignisprozedur:

```
Private Sub timZeit2_Timer()
Form1.BackColor = QBColor(Int(Rnd * 16))
End Sub
```

Für die **cmdButton_Click** Ereignisprozedur können wir uns Folgendes überlegen:

```
Private Sub cmdButton_Click()
timZeit1.Enabled = Not (timZeit1.Enabled)
timZeit2.Enabled = Not (timZeit2.Enabled)
End Sub
```

Untersuche auch hierfür die Funktionsweise.

Übung 3:
Erstelle ein neues Projekt mit einem Bildfeld (**picBild**), einer Steuerschaltfläche (**cmdButton**) und einem Zeitgeber(**timZeit1**) [Enabled: False, Interval: 100]. Deklariere ferner eine globale Integer-Variable **X**. Schreibe dann folgende Ereignisprozeduren:

```
Private Sub cmdButton_Click()
timZeit1.Enabled = Not (timZeit1.Enabled)
End Sub

Private Sub timZeit1_Timer()
picBild.Circle (0.5 * picBild.ScaleWidth, 0.5 *
picBild.ScaleHeight), X, QBColor(Int(Rnd * 16))
X = X + 25
If X > 0.5 * picBild.ScaleWidth Then
  X = 0
End If
End Sub
```

Untersuche die Funktionsweise!

Visual BASIC

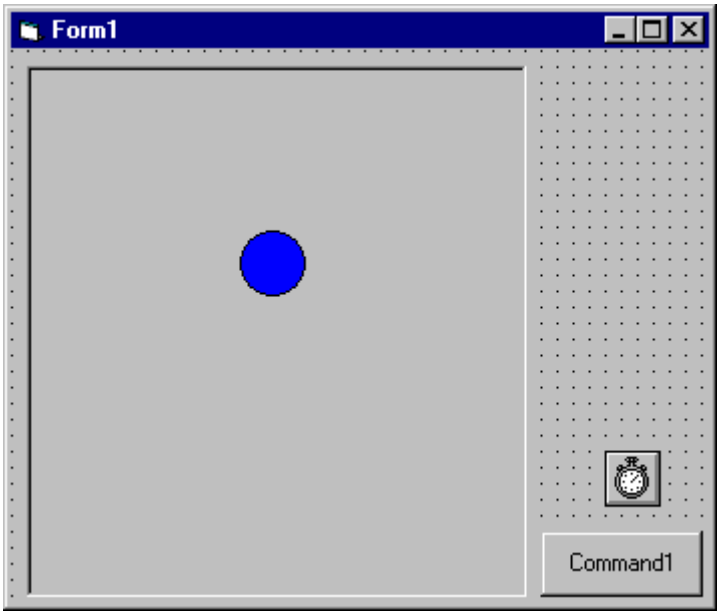
Die Move-Methode

Wir erzeugen die Animation von Objekten, d.h. Steuerelementen wie Linien- und Figuerenelemente oder Anzeigen. Alle Bewegungen erfolgen in einem Bildfeld. (Das Formular unterstützt auch Animationen.)
Die Bewegung von Steuerelementen in einem Bildfeld ist einfach. Zunächst zeichnen wir alle Steuerelemente in das Bildfeld, die **Move**-Methode wird dann benutzt, um Steuerelemente zu bewegen.

```
SteuerelementName.Move  NeuLinks, NeuOben
```

Diese Methode bewegt das Steuerelement von seiner aktuellen Postion nach NeuLinks,NeuOben. Dies sind dann die neuen Werte der beiden Eigenschaften Left und Top des Steuerelements. Jedes zu bewegende Steuerelement braucht einen ihm zugeordneten Zeitgeber! Die Move-Anweisung wird in der zugehörigen Zeitgeber-Ereignisprozedur eingetragen. Die so erzeugte periodische Bewegung ist eine **Animation**.

Übung 4:
Erstelle ein neues Projekt mit einem Bildfeld (**picBild**). Zeichne ein Figurenelement (**shpBall**) in das Bildfeld und setze einen Zeitgeber (**timZeit1**; Enabled: false, Interval: 100) auf das Formular.



Dazu erstellen wir folgende Ereignisprozeduren:

```
Private Sub cmdButton_Click()
timZeit1.Enabled = Not (timZeit1.Enabled)
shpBall.Left = 0.5 * (picBild.ScaleWidth - shpBall.Width)
shpBall.Top = 0
End Sub

Private Sub timZeit1_Timer()
shpBall.Move shpBall.Left, shpBall.Top +
picBild.ScaleHeight / 40
End Sub
```

Untersuche die Funktionsweise.

Verschwinden von Steuerelementen

Wenn Steuerelemente sich durch ein Bildfeld bewegen, benötigen wir darüber Kenntnis, wann sie sich aus dem Bildfeld bewegen. Dazu müssen wir grundsätzlich verschiedene Positionen und Dimensionen vergleichen.
Für unser Beispiel mit dem fallenden Ball sieht das so aus:

```
Private Sub timZeit1_Timer()
shpBall.Move shpBall.Left, shpBall.Top +
picBild.ScaleHeight / 40
If shpBall.Top > picBild.ScaleHeight Then
  shpBall.Top = -shpBall.Height
End If
End Sub
```

Gegen die Wand und zurück

Wir ändern nun unser Beispiel so, dass der Ball zurückprallt, wenn er den Boden des Bildfeldes erreicht.

```
Option Explicit
Dim BallDir As Integer
```

BallDir wird benutzt, um die Bewegungsrichtung des Balles zu beschreiben. BallDir=1 bedeutet abwärts, BallDir = -1 bedeutet aufwärts. Unsere Prozeduren cmdButton_Click und timZeit1_Timer:

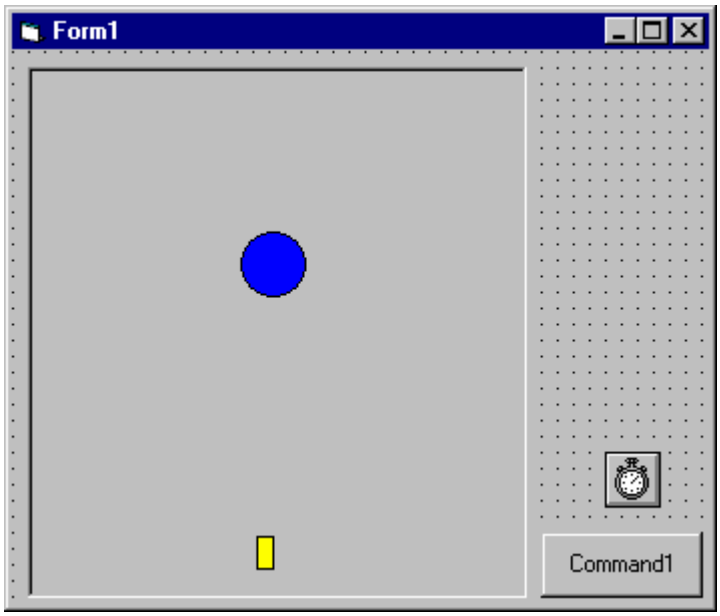
```
Private Sub cmdButton_Click()
timZeit1.Enabled = Not (timZeit1.Enabled)
shpBall.Left = 0.5 * (picBild.ScaleWidth - shpBall.Width)
shpBall.Top = 0
BallDir = 1
End Sub

Private Sub timZeit1_Timer()
shpBall.Move shpBall.Left, shpBall.Top + BallDir *
picBild.ScaleHeight / 40
If (shpBall.Top + shpBall.Height) > picBild.ScaleHeight
Then
  shpBall.Top = picBild.ScaleHeight - shpBall.Height
  BallDir = -1
  Beep
ElseIf shpBall.Top < 0 Then
  shpBall.Top = 0
  BallDir = 1
  Beep
End If
End Sub
```

Zusammenstöße feststellen

Jedes Steuerelement wird durch ein Rechteck beschrieben. Ein Zusammenstoß tritt also auf, wenn sich die beiden rechteckigen Bereiche überschneiden.
Vier Bedingungen müssen dazu gleichzeitig erfüllt sein.

```
If (Control1.Left + Control1.Width) > Control2.Left Then
  If  Control1.Left < (Control2.Left + Control2.Width)
Then
    If (Control1.Top + Control1.Height) > Control2.Top
Then
      If Control1.Top < (Control2.Top + Control2.Height)
Then
           [BASIC code for overlap, or collision]
      End If
    End If
  End If
End If
```

Übung 5:
Ergänze ein kleines Figurenelement (**shpKlotz**) nahe am Boden des Bildfeldes. Wir wollen prüfen, ob der Ball mit der Figur kollidiert und abprallt.

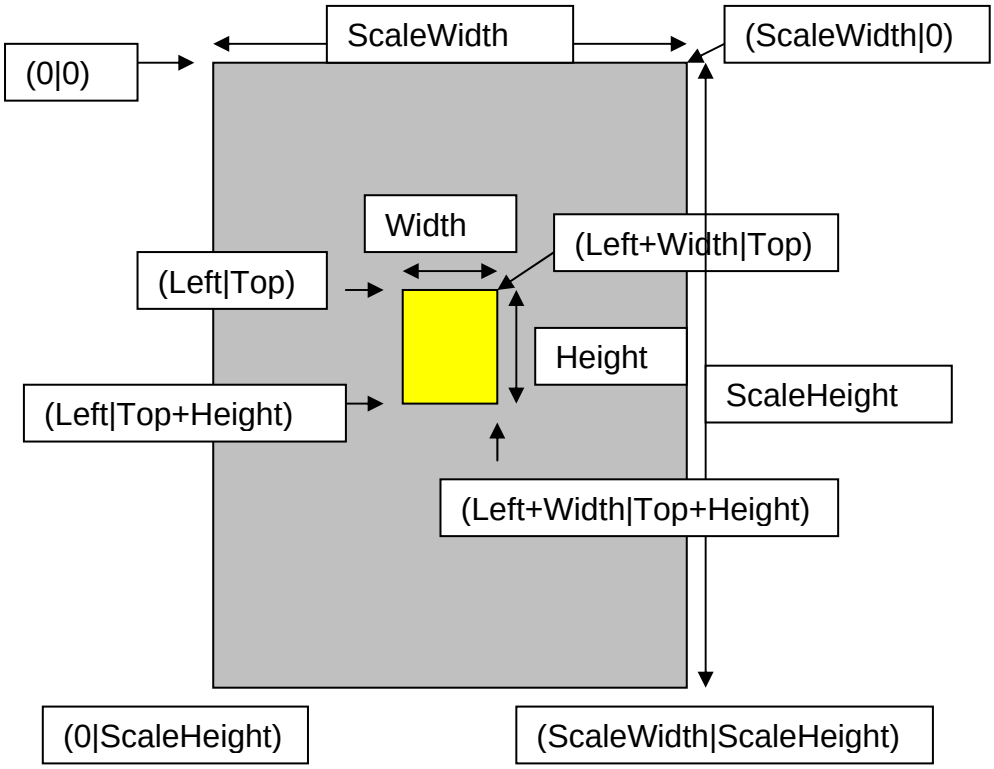


Wir ändern die Zeitgeber-Ereignisprozedur:

```
Private Sub timZeit1_Timer()
Dim Kollision As Boolean
shpBall.Move shpBall.Left, shpBall.Top + BallDir *
picBild.ScaleHeight / 40
Kollision = False
If (shpBall.Left + shpBall.Width) > shpKlotz.Left Then
  If shpBall.Left < (shpKlotz.Left + shpKlotz.Width) Then
    If (shpBall.Top + shpBall.Height) > shpKlotz.Top Then
      If shpBall.Top < (shpKlotz.Top + shpKlotz.Height)
Then
        Kollision = True
      End If
    End If
  End If
End If
If Kollision = True Then
  shpBall.Top = shpKlotz.Top - shpBall.Height
  BallDir = -1
  Beep
ElseIf shpBall.Top < 0 Then
  shpBall.Top = 0
  BallDir = 1
  Beep
End If
End Sub
```

Animationen mit der MOVE-Methode

Bei einer Animation ändert sich die Position eines Objekts mit der Zeit. Wir benötigen dazu die Koordinaten des Objekts sowie das umgebende Koordinatensystem.
Das Objekt besitzt die Eigenschaften LEFT, TOP, WIDTH und HEIGHT.
Das Bildfeld liefert das umgebende Koordinatensystem anhand der Eigenschaften SCALEWIDTH und SCALEHEIGHT.



1. Wir bewegen eine Figur (shpBall) in einem Bildfeld (picBild) von oben nach unten.

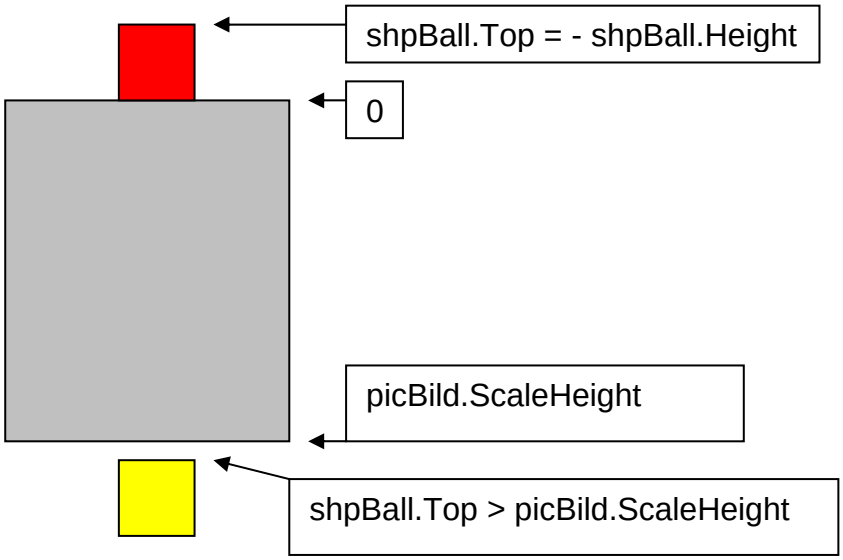
Bei einem Click auf die Schaltfläche cmdButton wird die Timereigenschaft *enabled* umgeschaltet und das Objekt an seine Startposition Mitte/Oben gesetzt. Mit der Methode MOVE verändern wir bei jedem Timerereignis die vertikale Position der Figur um einen Teilbetrag der Bildfeldhöhe.

```
timzeit1.Enabled = NOT (timZeit1.Enabled)
shpBall.Left = 0.5 * picBild.ScaleWidth - shpBall.Width)
shpBall.Top = 0
shpBall.Move shpBall.Left, shpBall.Top + picBild.ScaleHeight / 40
```

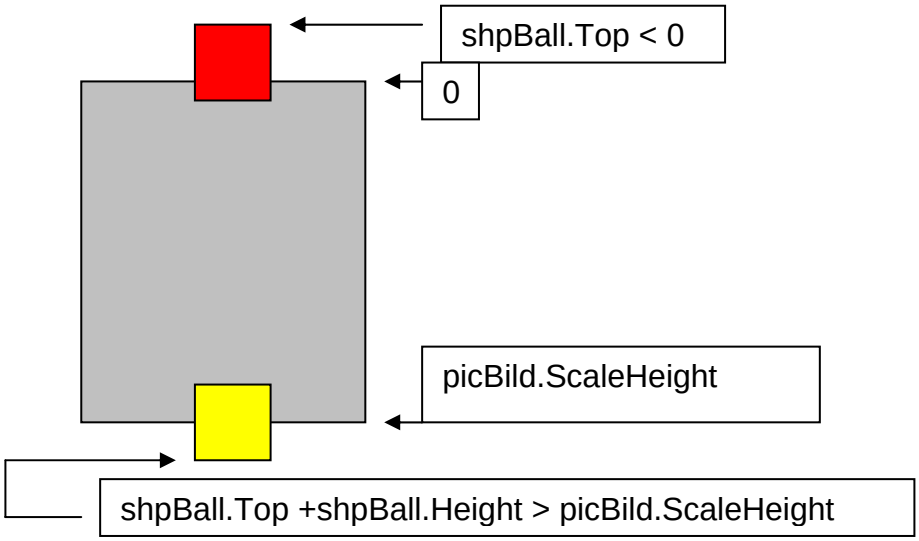
2. Verschwinden der Figur aus dem Bildfeld

Wenn die Figur das Bildfeld unten verlässt, wird sie vor den oberen Bildrand gesetzt.

```
shpBall.Top > picBild.ScaleHeight
shpBall.Top = - shpBall.Height
```

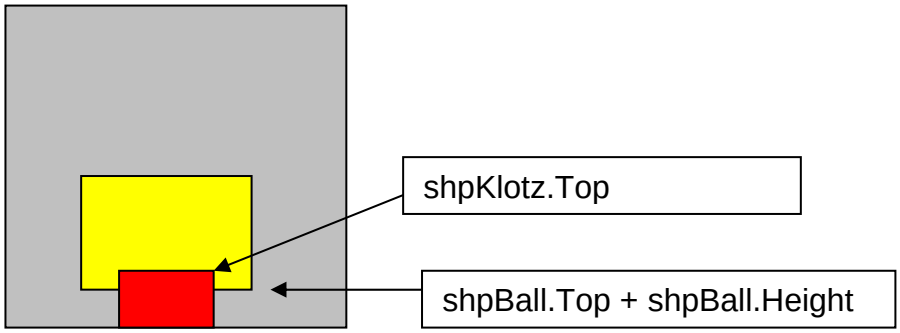


3. **Gegen die Wand und zurück (Bewegungen runter und rauf)**
Bei Bewegungen nach unten wird die Eignschaft Top vergrößert, bei Bewegungen nach oben wird sie verkleinert. Abhängig von der Bewegungsrichtung versehen wir daher die Berechnung der Position mit einem Faktor +1 bzw. -1. (Variable: BallDir)
shpBall.Top < 0 : obere Wand getroffen
shpBall.Top + shpBall.Height > picBild.ScaleHeight : untere Wand getroffen
shpBall.Move shpBall.Left, shpBall.Top + BallDir * picBild.ScaleHeight / 40



4. Zusammenstösse feststellen
Wie kann man feststellen, ob die bewegte Figur auf ein Hindernis (shpKlotz) trifft? Ein ausgefülltes Rechteck wird am unteren Rand des Bildfeldes platziert. Wenn der Ball auf den Klotz trifft, kommt es zu einer Überlappung der beiden rechteckigen Rahmenbereiche der Figuren.

shpBall.Top + shpBall.Height > shpKlotz.Top



Aufgaben:

1. Ersetze das Figurenelement shpBall durch eine Anzeige imgBall, in der eine Bilddatei, z.B. ein Smiley, dargestellt wird.
2. Führe waagerechte Bewegungen von links nach rechts aus.
3. Simuliere für die waagerechten Bewegungen Zusammenstösse mit zwei Rechtecken, die sich an den Rändern befinden.

Tastatur-Ereignisse

Mehrere Steuerelemente können Tastatur-Ereignisse erkennen: das Formular, Textfelder und Bildfelder. Dennoch, nur dasjenige Steuerelement, das den Fokus (**focus**) hat, kann Tastatur-Ereignisse empfangen. Mit der **SetFocus**-Methode kann man den Fokus einem Steuerelement zuweisen.

```
SteuerelementName.SetFocus
```

Dieser Befehl weist SteuerelementName den Fokus zu und macht es zum aktiven Steuerelement.

KeyDown-Ereignis

Das **KeyDown**-Ereignis hat die Fähigkeit, das Betätigen **jeder** Taste auf der Tastatur zu erkennen:

- Spezielle Kombinationen mit Shift-, Strg- und Alt-Tasten
- Einfg, Entf, Pos1, Bild↑ und Bild↓
- Cursor-Steuertasten
- Tasten auf dem Nummernblock
- Funktionstasten

Das KeyDown-Ereignis für ein Steuerelement *SteuerelementName* wird immer dann aufgerufen, wenn das Steuerelement den Fokus hat und eine Taste gedrückt wird.

```
Private Sub SteuerelementName_KeyDown(KeyCode As Integer,
Shift As Integer)
  [BASIC-Code für das KeyDown-Ereignis]
End Sub
```

Das Argument Shift informiert über den Zustand der Shift-, Strg- und Alt-Tasten.

Jede Taste der Tastatur besitzt einen sog. **Tastenschlüssel (KeyCode)**, den man im Programm auswertet. Einige Tasten werden auch durch symbolische Konstanten identifiziert, z.B. die Pfeiltasten zur Cursorsteuerung:

vbKeyLeft	←
vbKeyUp	↑
vbKeyRight	→
vbKeyDown	↓

Übung 6:
Starte ein neues Projekt mit einem Textfeld (**txtTest1**). Benutze folgendes **txtTest1_KeyDown** Ereignis:

```
Private Sub txtTest1_KeyDown(KeyCode As Integer, Shift As
Integer)
Select Case KeyCode
Case vbKeyLeft
  txtTest1.Text = "Left arrow"
Case vbKeyRight
  txtTest1.Text = "Right arrow"
Case vbKeyDown
  txtTest1.Text = "Down arrow"
Case vbKeyUp
  txtTest1.Text = "Up arrow"
End Select
End Sub
```

Schreibe in das Textfeld und benutze die vier Pfeiltasten.
Ergänze das Formular noch um ein weiteres Textfeld (**txtTest2**). Starte das Projekt, klicke in dieses neue Textfeld und schreibe etwas Text. Drücke auch die Pfeiltasten. Alles klar? Klicke einmal mit der Maus in Textfeld **txtTest1**.

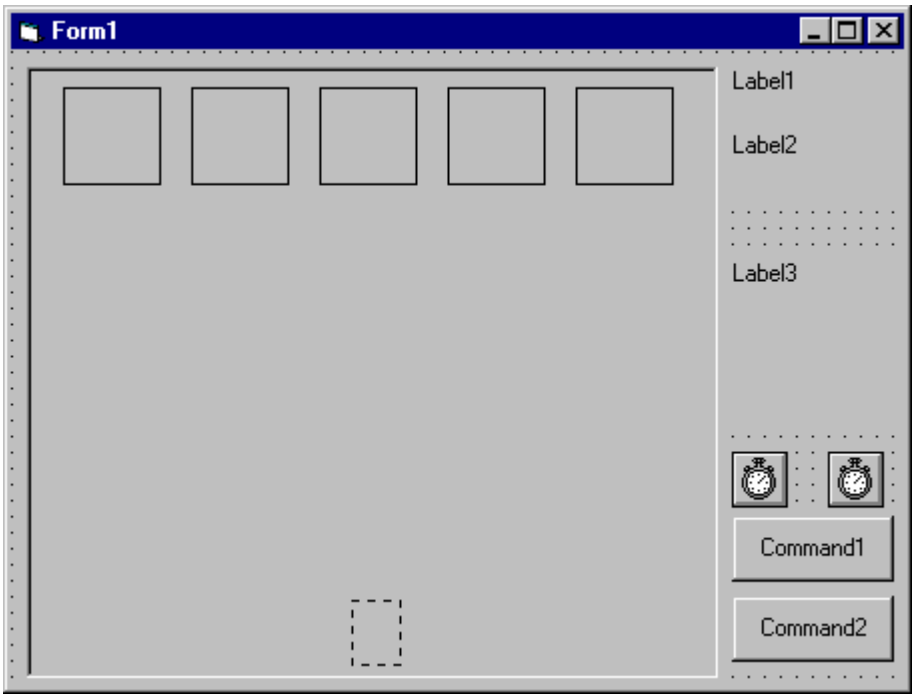
Neben dem **KeyDown-Ereignis** wird noch das **KeyPress-Ereignis** häufig verwendet. In unserem folgenden Projekt Ballons benutzen wir es nicht, daher findest Du dazu Erläuterungen und Beispiele im Anschluss an das Projekt.

Projekt - Ballons

In diesem Projekt erstellen wir ein kleines Videospiel. Bunte Luftballons fallen vom Himmel. Mit einer Spitze kannst du die Luftballons zerplatzen und Punkte sammeln. Du versuchst, so viele Luftballons wie möglich in einer Minute zu zerplatzen. Die Spitze kann mit der linken (←) und rechten (→) Pfeiltaste gesteuert werden.

Projektentwurf
Die gesamte Spielaktion erfolgt in einem Bildfeld. Es gibt fünf mögliche Ballons, jeder wird durch ein Figurenelement dargestellt. Die Spitze wird in einer Anzeige gehalten. Diese Anzeige wird mit der linken und rechten Cursortaste bewegt. Eine Steuerschaltfläche steuert Beginn und Ende des Spiels. Eine weitere Steuerschaltfläche beendet das Programm. Der Spielstand wird in einem Bezeichnungsfeld angezeigt.

Die Steuerelemente auf dem Formular



Eigenschaften der Steuerelemente

Form1	Name	frmBallons
	Caption	Ballons
	BorderStyle	1-Fest einfach
Picture 1	Name	picBallons
	BackColor	hellgelb

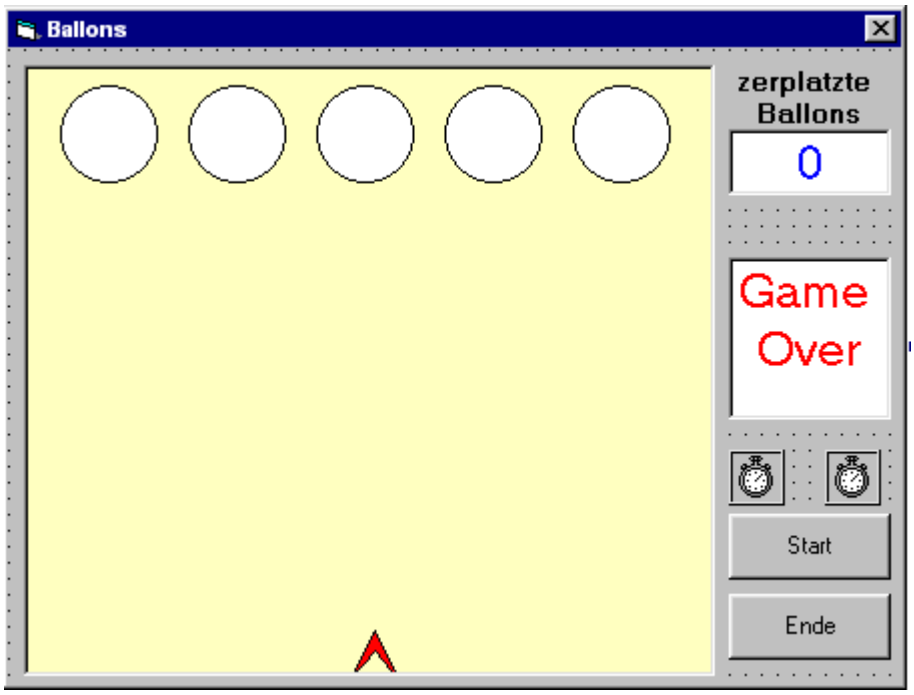
Shape1	Name	shpBall1
	BackStyle	1-undurchsichtig
	Shape	3-Kreis
	Visible	False
entsprechend für **Shape2, Shape3, Shape4, Shape5**		

Image1	Name	imgSpitze
	Picture	arrow.wmf
	Stretch	True
Label1	Name	lblKopf
	Caption	zerplatzte Ballons
	FontSize	10
	FontStyle	fett
	Alignment	2-Zentriert

Label2	Name	lblPunkte
	Caption	0
	FontSize	18
	Alignment	2-Zentriert
	BackColor	Weiss
	ForeColor	Blau
	BorderStyle	1-Fest einfach
Label3	Name	lblOver

	Caption	Game Over
	FontSize	18
	Alignment	2-zentriert
	BackColor	Weiss
	ForeColor	Rot
	BorderStyle	1-Fixed Single

Command1	Name	cmdStart
	Caption	Start
Command2	Name	cmdEnde
	Caption	Ende
Timer1	Name	timBallons
	Enabled	False
	Interval	100
Timer2	Name	timSpiel
	Enabled	False
	Interval	60000



Ereignisprozeduren

Das Spiel ist von der Idee her einfach. Zum Spielen klickt man den Start-Button. Die fünf Ballons fallen im Bildfeld herunter, jeder unterschiedlich schnell. Die Spitze wird mit der linken und rechten Pfeiltaste gesteuert. Ist die Spitze unter einem Ball bei einer Kollision, platzt der Ballon und du bekommst einen Punkt. Nach dem Platzen oder Auftreffen auf dem Boden erscheinen neue Ballons wieder oben. Du versuchst, so viele Ballons wie möglich in 60 Sekunden zu zerplatzen. Nach dieser Zeit erscheint die Meldung "Game over". Du kannst wieder auf Start klicken oder das Programm mit Ende beenden.

Wir verwenden ein Feld, um die verschiedenen Ballgeschwindigkeiten zu erfassen.

```
    Option Explicit
```

```
Dim BallSpeed(5) As Integer
```

Die Werte werden später wie folgt berechnet:

```
Int(Rnd * 50) +50
```

In der Prozedur **Form_Load** müssen wir natürlich **Randomize** eintragen.

Für die Bewegung der Spitze brauchen wir die Ereignisprozedur

```
Private Sub picBallons_KeyDown(KeyCode As Integer, Shift
As Integer)
'Prüfe auf Pfeiltasten
If KeyCode = vbKeyLeft Then
  imgSpitze.Move imgSpitze.Left - 50, imgSpitze.Top
ElseIf KeyCode = vbKeyRight Then
  imgSpitze.Move imgSpitze.Left + 50, imgSpitze.Top
End If
End Sub
```

Die **cmdEnde_Click** Prozedur ist so wie immer.

Hier nun die Schritte, die beim **cmdStart_Click** Ereignis ausgeführt werden. Da die Steuerschaltfläche zwei Aufgaben hat, teilen wir das Ganze in zwei Abschnitte:
Wenn Caption "Start" ist:

- Mitteilung ‘Game Over’ verstecken
- cmdStart Caption auf “Stopp” setzen
- cmdEnde-Button deaktivieren
- Punktestand auf 0
- Initialisiere für jeden Ballon Position, Farbe und Geschwindigkeit
- Fokus an das Bildfeld geben (für KeyDown!)
- Zeitgeber starten

Wenn Caption "Stopp" ist:

- Mitteilung ‘Game Over’ anzeigen
- cmdStart Caption auf “Start”
- cmdEnde-Button aktivieren
- Zeitgeber stoppen

Betrachten wir nun die Ereignisprozedur **cmdStart_Click** in bezug auf die oben genannten Schritte.

```
Private Sub cmdStart_Click()
If cmdStart.Caption = "Start" Then
  'Neues Spiel
  lblOver.Visible = False
  cmdStart.Caption = "Stop"
  cmdEnde.Enabled = False
  lblPunke.Caption = "0"
  imgSpitze.Left = 0.5 * (picBallons.ScaleWidth -
imgSpitze.Width)
  'Position, Farbe und Geschw.für jeden Ballon
  shpBall1.Top = -shpBall1.Height
  shpBall1.BackColor = QBColor(Int(Rnd * 6) + 9)
  shpBall1.Visible = True
```

```
      BallSpeed(1) = Int(Rnd * 50) + 50
      diese Zeilen entsprechend für die vier anderen Ballons
      ...
      'Fokus auf das Bildfeld
      picBallons.SetFocus
    Else
      'Spiel beendet
      lblOver.Visible = True
      cmdStart.Caption = "Start"
      cmdEnde.Enabled = True
    End If
    'Zeitgeber umschalten
    timBallons.Enabled = Not (timBallons.Enabled)
    timSpiel.Enabled = Not (timSpiel.Enabled)
    End Sub
```

Die **timSpiel_Timer** Ereignisprozedur sollte so aussehen:

```
    Private Sub timSpiel_Timer()
    '60 Sekunden sind um - Spiel beenden
    timBallons.Enabled = False
    timSpiel.Enabled = False
    lblOver.Visible = True
    cmdStart.Caption = "Start"
    cmdEnde.Enabled = True
    End Sub
```

Nun fehlt noch das Herzstück der Programmierung: das **timBallons_Timer** Ereignis. Die erforderlichen Schritte für einen Ballon:

- Bewege den Ballon
- Prüfe, ob der Ballon den Boden berührt hat. Wenn ja, starte einen neuen Ballon mit neuer Farbe und neuer Geschwindigkeit.
- Prüfe, ob der Ballon zerplatzt wurde. Wenn ja, kurzes Tonsignal, erhöhe den Punktestand und entferne den Ballon.

Hier die vollständige Codierung für den ersten Ballon:

```
'Bewege Ballon 1
shpBall1.Move shpBall1.Left, shpBall1.Top + BallSpeed(1)
If (shpBall1.Top + shpBall1.Height) > picBallons.ScaleHeight
Then
  'Ballon erreicht den Boden
  shpBall1.Top = -shpBall1.Height
  shpBall1.BackColor = QBColor(Int(Rnd * 6) + 9)
  shpBall1.Visible = True
  BallSpeed(1) = Int(Rnd * 50) + 50
ElseIf (shpBall1.Top + shpBall1.Height) >
(picBallons.ScaleHeight - imgSpitze.Height) Then
  'Ballon nicht zerplatzt, prüfe auf Treffer
  If shpBall1.Visible = True Then
    If shpBall1.Left < imgSpitze.Left Then
      If (shpBall1.Left + shpBall1.Width) > (imgSpitze.Left +
imgSpitze.Width) Then
        'Ballon zerplatzt
```

```
        'Punkt geben - Ballon entfernen
        Beep
        lblPunkte.Caption = Str(Val(lblPunkte.Caption) + 1)
        shpBall1.Visible = False
      End If
    End If
  End If
End If
```

Füge diese Anweisungen in die Ereignisprozedur **timBallons_Timer** ein. Mittels *Copy and Paste* kannst du nun auch die Anweisungen für die vier anderen Ballons einbauen. Ändere dabei einfach nur die Ballon-Nummer.

Erweiterungen

Bei der Programmierung fällt auf, dass die Anweisungen für die Behandlung der fünf Ballons bis auf die Nummerierung identisch sind. Erstellt man für die fünf Shapes ein Datenfeld, so kann man über eine FOR-NEXT-Schleife eine erheblich kürzere Programmierung erreichen. Das VB-Projekt Ballons2 zeigt dir diese vereinfachte Programmierung, wobei das Feld mit den Shapes shpBall(0)...shpBall(4) bereits beim Formularaufbau erstellt wurde. [Beachte: Der Feldindex beginnt bei VB stets mit 0!]
Natürlich kann man unser kleines Spiel nun vielfältig erweitern. Aber irgendwann ist die Unterrichtszeit auch einmal zu Ende.

Ergänzung zum Keypress-Ereignis

Das KeyPress-Ereignis

Das KeyPress-Ereignis erkennt alle Tasten, denen eine **ASCII**-Nummer (Zahlen von 0 bis 255) zugeordnet sind. Die KeyPress Ereignisprozedur für ein Steuerelement mit Namen *SteuerelementName* lautet

```
Private Sub SteuerelementName_KeyPress(KeyAscii As
Integer)
  [BASIC-Code für das KeyPress-Ereignis]
End Sub
```

Das Argument **KeyAscii** ist die ASCII-Nummer der gedrückten Taste. Eine ASCII-Codetabelle findet ihr in der Online-Hilfe von Visual Basic. Es gibt aber zwei sehr nützliche Funktionen:
Die **Chr**-Funktion wandelt eine ASCII-Nummer in das Zeichen um.

```
Zeichen = Chr(KeyAscii)
```

Die **Asc**-Funktion liefert zu einem Zeichen die ASCII-Nummer.

```
KeyAscii = Asc(Zeichen)
```

Übung 7:

Starte ein neues Projekt und füge ein Bezeichnungsfeld (**lblAnzeige**) und ein Textfeld (**txtEingabe**) hinzu. Schreibe folgende **txtEingabe_KeyPress** Ereignisprozedur:

```
Private Sub txtEingabe_KeyPress(KeyAscii As Integer)
lblAnzeige.Caption = Chr(KeyAscii) + Str(KeyAscii)
End Sub
```

Untersuche die Funktionsweise.

Übung 8:

Die Asc-Funktion wird sehr viel mit dem KeyPress-Ereignis benutzt. Wir können sie dazu benutzen, um sicherzustellen, daß der Benutzer nur bestimmte, gewünschte Zeichen eingibt. Ändere im vorherigen Beispiel **txtEingabe_KeyPress** so:

```
Private Sub txtEingabe_KeyPress(KeyAscii As Integer)
If KeyAscii < Asc("0") Or KeyAscii > Asc("9") Then
  KeyAscii = 0
End If
lblAnzeige.Caption = Chr(KeyAscii) + Str(KeyAscii)
End Sub
```

Untersuche die Funktionsweise. Versuche, alphanumerische Werte einzugeben - im Bezeichnungsfeld wird nichts dargestellt (KeyAscii = 0) -. Diesen Prozess der Erkennung und Änderung unerwünschter Tasteneingaben nennt man Tastenfalle (**key trapping**)

Übung 9:

Hier eine weitere Anwendung für dieses key trapping:

```
Private Sub txtEingabe_KeyPress(KeyAscii As Integer)
If KeyAscii >= Asc("a") And KeyAscii <= Asc("z") Then
  KeyAscii = KeyAscii - 32
End If
If KeyAscii < Asc("A") Or KeyAscii > Asc("Z") Then
  KeyAscii = 0
End If
lblAnzeige.Caption = Chr(KeyAscii) + Str(KeyAscii)
End Sub
```

Untersuche die Funktionsweise.

Übung 10:

Nehmen wir noch ein Beispiel, hoffentlich blickst du da durch!

```
Private Sub txtEingabe_KeyPress(KeyAscii As Integer)
KeyAscii = Int(Rnd * 96) + 32
lblAnzeige.Caption = Chr(KeyAscii) + Str(KeyAscii)
End Sub
```