VISUAL BASIC, ein Werkzeug für Alle!

Hermann Köberl

1. Einleitung

Mit Windows hat Microsoft ein Produkt geschaffen, das wie kein anderes seinen ungebrochenen Siegeszug durch den gesamten PC-Sektor hält. PCs ohne Windows sind heutzutage nahezu undenkbar. Millionen Anwender haben die Vorzüge der einheitlichen und ausgesprochen benutzerfreundlichen Windows-Oberfläche kennen- und schätzen gelernt. Die neueste Version, Windows 95, stellt erstmals ein echtes 32-Bit-Betriebssystem zur Verfügung, und man darf hoffen, daß einige Probleme der Einführungsphase bald vergessen sind. An W95 und W95-Software führt kein Weg vorbei, dafür wird der Marktführer Microsoft schon sorgen!

Was sind nun die wesentlichen Unterschiede zu den vertrauten DOS-Programmen und welchen Einfluß hat das auf die Programmierung? Vor einigen Jahren war das Entwickeln von Windows-Programmen noch ein nervenaufreibendes Unterfangen, das nur weinigen Profis zugänglich war. Mit Visual Basic sind diese Zeiten endgültig vorbei.

BASIC hat in den letzen Jahren eine starke Weiterentwicklung erfahren und ist mit dem wenig brauchbaren GWBASIC kaum mehr vergleichbar. VISUAL BASIC hat sich mit der Version 3.0 endgültig etabliert und erfüllt mit der professionellen Ausgabe (in englischer Sprache) auch gehobene Ansprüche. Mittlerweile wurden schon tausende von Windowsprogrammen mit diesem Werkzeug entwickelt; mit soviel Erfolg, daß Microsoft VB zur Anwender-Programmiersprache ihrer Standardsoftware wie PROJEKT, EXCEL und ACCESS machte. Das bedeutet, daß Sie mit VB z.B. auch Ihre Tabellenkalkulation maßschneidern können oder eine komplette Datenbankanwendung mit Access .

Die 32-Bit-Version beinhaltet noch etliche Erweiterungen und Verbesserungen und steht in Konkurrenz zu C++ und Borlands DELPHI.

Was ist nun das Faszinierende an diesem Tool?

Sie programmieren nicht mit Code und komplizierten Algorithmen, sondern mit der Maus und ungebremster Kreativität. Viele Problemlösungen können in VB mit nur wenigen Mausklicks programmiert werden, ohne auch nur eine Zeile Code zu schreiben. Selbst Datenbankzugriffe, Grafik und Sound stellen keine Probleme mehr dar. Dabei stehen Ihnen alle Möglichkeiten von Windows offen, angefangen von Schaltflächen über Menüs bis zu kompliziert scheinenden Mechanismen wie Direktzugriff auf Windows-Bibliotheksfunktionen (API) oder selbsterstellte DLL-Bibliotheken sowie OLE- und DDE-Anwendungen. Sie können grafisch ansprechende Applikationen wie mit einem Malprogramm zusammenstellen und die Ergebnisse sofort austesten. So haben Sie immer den Blick frei für das Wesentliche, nämlich für das eigentliche Problem, das Sie mit Ihrem Programm auf eine möglichst benutzerfreundliche Art und Weise lösen möchten. Die Zeit, die Programmierer in anderen Sprachen auf technische Detailfragen verwenden, nutzen Sie, um mit fast spielerischer Leichtigkeit ihre Benutzeroberfläche zu optimieren. Dabei sind Änderungen schnell und intuitiv möglich: Ein Mausklick genügt, um eine günstigere Position für eine Schaltfläche zu finden, ein Fenster ein bißchen zu vergrößern oder Text und Farben zu ändern. Mit Windows-Interna brauchen Sie sich nicht herumzuschlagen. Für Sie ist Windows nichts anderes als eine „Blackbox“. Im Klartext bedeutet dies, daß Sie in Windows Steuerelemente einsetzen können, ohne sich über deren interne Funktion im klaren sein zu müssen. Für Sie sind diese Steuerelemente Schaltflächen, Eingabefelder und so weiter - nichts als „Objekte“, von denen Sie nur die wichtigsten Eigenschaften kennen müssen. Daher nennt man die Programmiertechnik unter Visual Basic auch Objektorientierte (bzw. Ereignisorientierte) Programmierung (OOP), eine Technik, die die Programmierung, wie man sie bisher kannte, komplett umgekrempelt hat.

VISUAL-BASIC ist keine neue Programmiersprache sondern eine neue Philosophie: Auf das erfolgreichste Programmier-Produkt von Microsoft, Quick-Basic, aufbauend, werden sowohl dem Anfänger, als auch dem fortgeschrittenen Programmierer neue Möglichkeiten eröffnet. In konsequenter Weiterentwicklung will Microsoft das eigentliche Programmieren einigen wenigen Spezialisten überlassen, die z.B. in C++ oder Assembler die Komponenten der meisten Anwendungsprogramme entwickeln und als Custom-Controls zum Kauf anbieten. Schon heute gibt es beispielsweise fertige Multimedia-, Kommunikations- und Grafikmodule, die sich der Programmierer als VBX- oder OCX-Datei in sein Projekt laden und mit wenigen Änderungen nach seinen Vorstellungen nutzen kann. Hunderte von solchen Programm-Bausteinen sind bereits auf dem Markt und werden immer häufiger auch für professionelle Applikationen eingesetzt: aus der SOFTWARE wird KOMPONENTWARE .

2. Unterschiede zwischen DOS- und WINDOWS-Programmierung

Für viele PC-Freaks, die seit Jahren in Pascal, Quick-Basic oder C programmieren, stellt sich nun die Frage:

Was ist bei Windows-Programmen anders?

Der für den Programmersteller wichtigste Unterschied ist wahrscheinlich die unterschiedliche Kommunikation zwischen Programm und Benutzer:

In DOS-Programmen sind alle Tastatur- und Mauseingaben voneinander getrennt und werden durch zyklische Abfragen der entsprechenden Interrupts gesteuert. Das Programm bestimmt somit den Dialog, die Art der Eingabe und damit den gesamten Ablauf. Windows-Programme fangen alle Benutzeraktivitäten selbständig ab und wandeln sie in Nachrichten (Messages) um. Diese stellen somit das zentrale Kommunikationsmedium zwischen der Applikation und dem Betriebssystem dar. Alle Tasks werden über einen sogenannten Message-Dispatcher gesteuert, der ähnlich einer Postzentrale alle Nachrichten empfängt und weiterleitet.

Jede Anwendung läuft in einem Fenster (entspricht der Form in VB) ab und ist praktisch die Hauptprozedur dieses Fensters. Alle an ein solches ‘Window’ weitergeleiteten Nachrichten werden von der jeweiligen Prozedur geprüft und lösen entweder eine Reaktion aus oder werden einfach an eine Standardprozedur weitergeleitet. Dadurch wird das gesamte System fast völlig vom Benutzer und dessen Eingaben gesteuert. Alle Anwendungen benutzen die gleichen Funktionen (Windows DLLs) und kommunizieren mit dem System über genormte Schnittstellen (Nachrichten, Strukturen). Jede Windows-Applikation entspricht sowohl hinsichtlich ihrer Obefläche als auch ihres Verhaltens einem genormten Interface.

Eine ähnliche Funktion wie bei der Eingabe erfüllt Windows auch bei allen Ausgabeoptionen: Unter DOS sind alle Ausgaben an ein bestimmtes physisches Medium (Bildschirm, Drucker etc.) gerichtet - unter Windows erfolgt der gesamte Output über die gleichen Ausgabefunktionen, ganz gleich welches Ausgabemedium angesprochen werden soll. Bildschirmausgaben erfolgen über Fenster.

Während DOS sehr wohl zwischen Grafik- und Textausgaben unterscheidet, erfolgen unter Windows alle Ausgaben im Grafikmodus (Text als spezielle Form der Grafik). Die genauen Koordinaten werden dabei entsprechend der Auflösung sowie der Größe und des Standortes des jeweiligen Applikationsfensters vom Bildschirmtreiber errechnet. Texte und Bilder können fast nach Belieben kombiniert werden, wobei Windows einen Großteil der Routinearbeiten übernimmt.

Die weiteren Unterschiede, wie Multitasking und die verbesserte Speicherverwaltung sind für den Windows-Programmierer weniger von Bedeutung, jedoch sollen die Begriffe DLL, API, DDE und OLE kurz erläutert werden:

Dynamic Link Libraries (DLL's)

Mehrfach geladene Programme sind in der Lage, sich Code und Daten zu teilen. Werden zum Beispiel Funktionen einer bestimmten Bibliothek von mehreren parallel laufenden Anwendungen genutzt, so muß diese Bibliothek lediglich einmal geladen werden. Die Programme können die Funktionen danach dynamisch aufrufen. Die Bibliotheksfunk-

tionen müssen also nicht fester Bestandteil der jeweiligen Anwendung sein, sondern werden erst zur Laufzeit mit dem Programm verbunden.

Laufzeitbibliotheken mit den genannten Eigenschaften heißen DLLs und stellen die Grundbausteine von Windows dar. Die aufrufbaren Funktionen werden (**A**)pplication-(**P**)rogramm-(**I**)nterface genannt. Die Laufzeitbibliothek von VB3 befindet sich im Verzeichnis. `\WINDOWS\SYSTEM` als `VBRUN300.DLL` und ist ziemlich umfangreich.

Dynamic Data Exchange (DDE)

Dynamic Data Exchange (DDE) bedeutet, daß Windows-Programme untereinander Daten über eine genormte Schnittstelle austauschen können , ähnlich , wie das auch über das Clip-Board geschieht.

Eine interessante Möglichkeit, Objekte aus anderen Anwendungen mit einem Windows-Programm zu veknüpfen (link) bzw. es vollständig einzubauen (embedd) wird durch die **OLE-Fähigkeit** realisiert. Man kannz.B. direkt Excel-Tabellen oder Paintbrush-Grafiken in das laufende Programm einbauen oder die „Quellenprogramme“ als **OLE-Server** benutzen.

3. Allgemeiner Aufbau eines Windows Programms

Normalerweise läuft jedes Windows-Programm in einem Fenster mit einer gewissen Eigendynamik ab: Das Programm filtert aus dem „Nachrichtenstrom“ Informationen und reagiert darauf. Allen Anwendungen liegt ein Konzept zugrunde, das so einfach zu verstehen ist, daß Sie es anwenden können, ohne es zu kennen: Das Konzept der **Objektorientierten Programmierung** (OOP).

Windows unterstützt die unterschiedlichsten Hardware-Systeme, Sprachen und Anwendungsbereiche. Ein Programm-Entwickler für Windows müßte all diese verschiedenen Möglichkeiten berücksichtigen.

Ein solcher Aufwand wäre kaum gerechtfertigt. Niemand würde Anwendungen für Windows programmieren, wenn er derart viele Eventualitäten beachten müßte. Microsoft hat folglich ein anderes Konzept gewählt, und dieses Konzept heißt Objektorientierte Programmierung (OOP). Kernpunkt dieses Konzepts ist das Black-Box-Prinzip. Das heißt, daß der Programmierer Funktionen von Windows nutzen kann, ohne zu wissen, wie sie eigentlich funktionieren. Wie aber funktioniert die Objektorientierte Programmierung konkret? Nun, das läßt sich an einem Beispiel aus dem täglichen Leben recht anschaulich verdeutlichen:

Objekte des täglichen Lebens

Stellen Sie sich ein Telefon vor. Ein Telefon stellt für die meisten Menschen eine Blackbox wie oben angesprochen dar. Sie benutzen es, wissen aber nicht genau, wie es funktioniert. Das brauchen Sie auch nicht zu wissen, Hauptsache, Sie können damit telefonieren. Eine solche Blackbox, deren Innenleben man bewußt nicht kennt, nennen wir hier **„Objekt“.**

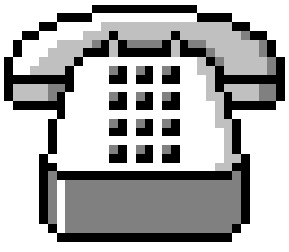


Abbildung 1

Wie kann man nun ein solches Objekt und den Umgang damit treffend beschreiben?

Zunächst ist ein Objekt immer Träger von **„Eigenschaften“:**

Farbe:	grün, rot, grau, gelb, weiß...
Zustand:	neu, alt
Tasten:	ja, nein

In der Notation von Visual Basic werden Eigenschaften wie folgt geschrieben:

```
Objektname.Eigenschaft = Eigenschaftswert
```

Was hier kompliziert klingt**,** wird logischer, wenn man sich ein paar Beispiele ansieht:

```
Telefon.Farbe = grün

Telefon.Tasten = ja

Telefon.Zustand = neu
```

Eigenschaften kann man sowohl auslesen, um zum Beispiel die Farbe eines Telefons festzustellen, als auch festlegen **-** die Eigenschaft „angeschlossen“ zum Beispiel, kann durch Ziehen des Steckers von „ja“ auf „nein“ verändert werden.

Eine weitere wichtige Möglichkeit, ein Objekt wie das Telefon zu charakterisieren, ist die Frage, was man damit machen kann. Man spricht in diesem Zusammenhang von **"Methoden".** Folgende Methoden sind bei unserem Telefon-Objekt prinzipiell denkbar:

```
abnehmen, auflegen, wählen.
```

In Visual Basic werden diese Methoden folgendermaßen notiert:

```
Objektname.Methode
```

Hier wieder ein paar Beispiele zur Verdeutlichung:

```
Telefon.abnehmen

Telefon.auflegen

Telefon.wählen  112
```

Ein Telefon ist deshalb ein schönes Beispiel für ein Objekt, weil es ein gewisses „Eigenleben“ hat. Stellen Sie sich vor, Sie haben sich in einem Unternehmen beworben und warten auf den Rückruf des Personalchefs. Dann ist das Klingeln des Telefons mit Sicherheit ein "Ereignis". Dieser Begriff wird auch in der Objektorientierten Programmierung benutzt und in Visual Basic wie folgt notiert:

```
Objektname_Ereignis
```

Ein Beispiel: `Telefon_klingeln`

Beachten Sie bitte, daß Methoden und Ereignisse zwei völlig verschiedene Dinge sind! Methoden wenden Sie als Benutzer des Telefons auf das Telefon an, Ereignisse - wie zum Beispiel das Klingeln - löst das Telefon aus.

Für jedes Objekt, das Sie ausgewählt haben, wird vom VB-Programmiersystem eine sogenannte Ereignisprozedur - zumindest der Beginn und das Ende- automatisch breitgestellt. Es bleibt dem Programmierer überlassen, ob das Programm auf ein bestimmtes Ereignis reagieren soll, dann füllt er die entsprechende Prozedur einfach mit Programm-Code aus.

Windows-Objekte

Welche Auswirkungen haben diese Erkenntnisse nun auf das Programmieren unter Windows? Erinnern Sie sich noch an das Black-Box-Prinzip? Der Programmierer soll Elemente von Windows nutzen können, ohne deren inneren Aufbau zu verstehen. Welche Elemente (oder besser gesagt: Objekte) wären das denn?

Objekte von Windows sind zum Beispiel Schaltflächen, Eingabefelder, Listen und Ähnliches. All diese Elemente, mit denen der Benutzer Eingaben tätigen kann, oder mit denen ihm Informationen angezeigt werden, nennt man Steuerelemente. Objekte von Windows sind aber auch Fenster, Menüs, der Maus-Zeiger, der Drucker, der Bildschirm und so weiter. Generell kann man fast alles unter Windows als Objekt ansprechen.

Dabei haben Windows-Objekte meist ganz spezielle Eigenschaften. Nehmen wir als Beispiel das Objekt *Schaltfläche.* Welche Eigenschaften hat dieses Objekt? Zunächst einmal die Beschriftung. In Visual Basic benutzt man für Objekteigenschaften stets englischsprachige Schlüsselworte. Das Schlüsselwort für Beschriftung lautet `Caption`. Weitere Eigenschaften einer Schaltfläche sind deren Breite *(*`Width`*),* deren Höhe *(*`Height`*),* mit welcher Schriftart Sie beschriftet ist *(*`Font-`

Name) und in welcher Schriftgröße (FontSize). Eine weitere Eigenschaft ist die Farbe (BackColour) und so weiter.

Welche Methoden kann man auf ein Objekt anwenden? Man kann es zum Beispiel anzeigen lassen, dazu benutzt man die Refresh-*Methode.* Oder man kann es bewegen, dazu verwendet man die Move-*Methode.* Bei manchen anderen Objektarten kann man eine Reihe ganz spezieller Methoden anwenden, wie zum Beispiel beim Objekt CD-ROM die Methode „Laufwerkschublade öffnen" (OpenDoor) oder bei einem Drucker die Methode „Seite auswerfen" (FormFeed). Sie sehen: Mit Methoden läßt sich eine ganze Menge anstellen.

Wie sieht es nun aber mit den Ereignissen aus, die solche Objekte auslösen können? Hat eine Schaltfläche eine Art Eigenleben, wie das vorhin genannte Telefon? Auf den ersten Blick nicht, dennoch können Schaltflächen ausgesprochen wichtige Ereignisse auslösen. Klickt beispielsweise ein Benutzer eine Schaltfläche an, wird damit das Ereignis: Click ausgelöst.

Ist dies wirklich ein Ereignis, oder handelt es sich bei Click nicht eher um eine Methode? Aus der Sicht des Benutzers ist Click mit Sicherheit eine Methode, wir betrachten das Programm jedoch aus der Sicht des Programmierers. Und aus diesem Blickwinkel ist Click ein Ereignis, denn wir wissen ja nicht, wann und wie der Benutzer auf die Schaltfläche klicken wird.

Eigenschaften:
- Breite (Width)
- Höhe (Height)
- Beschriftung (Caption)
- Zeichensatz (FontName)
- Farbe (BackColor)
- Aktiv (Enabled)

Methoden:
- anzeigen(Refresh)
- bewegen(Move)

Ereignisse:
- Maus losgelassen (MouseUp)
- Taste drücken (KeyPress)
- Maus gedrückt (MouseDown)
- Maus klicken (MouseClick)

Abbildung 2

Beispiele:

```
Knopf.Caption= "ENDE"
Knopf.Move 0,0,100,100
Knopf_Click
```

Nach den Grundlagen soll nun die praktische VB-Programmierung kurz erörtert werden.

4. Wie entsteht ein Visual-Basic-Projekt?

Ein Projekt besteht meist aus mehreren zusammengehörigen Dateien (Formen, Basic-Module, Custom-Controls), welche im sogenannten Projekt-Fenster sichtbar und in der Projektdatei aufgelistet sind.

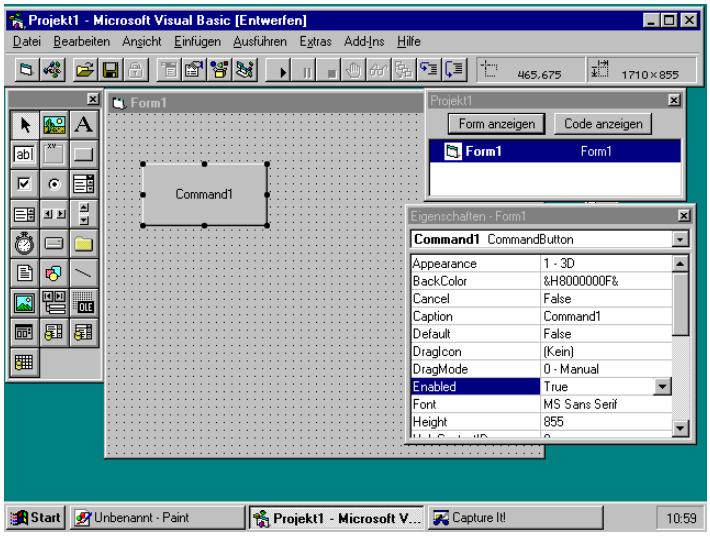


Abbildung 3

Die oben dargestellte **Visual-Basic Benutzeroberfläche** besteht aus mehreren Bereichen:

- **Form-Fenster:** Jede von VB erstellte Anwendung benötigt mindestens eine Form. Diese kann Steuerelemente enthalten, sowie Text oder Grafik ausgeben. In diesem Fenster läuft das eigentliche Programm ab.
- **Projekt-Fenster:** (rechts oben) **.** Hier sind sämtliche Dateien aufgelistet, die zu einer VB-Anwendung gehören.
- **Werkzeugsammlung:** (am linken Rand der Abbildung) Durch Ziehen oder Anklicken können hier die gewünschten Steuerelemente ausgewählt und auf die Form gebracht werden.
- **Eigenschaften-Fenster:** (rechts unten) Jedes Steuerelement besitzt bestimmte Eigenschaften, die hier nach Bedarf interaktiv verändert werden können.
- **Menüleiste:** Dient zum Aufrufen einzelner Pull-Down-Menüs, welche die eigentlichen Funktionen der Benutzeroberfläche zum Inhalt haben.
- **Werkzeugleiste:** Die wichtigsten Menüfunktionen sind hier noch einmal als Icons verfügbar.

Weitere Fenster, die nicht in der Abbildung enthalten sind:

- **Menüentwurfsfenster:** Dient zur Erstellung eigener Pull-Down-Menüs.
- **Code-Fenster:** Zu jedem Objekt wird automatisch, für jedes mögliche Ereignis eine Art 'Programmhülse' bereitgestellt, die der Programmierer wahlweise mit Code füllen kann. Wenn das Objekt auf bestimmte Ereignisse nicht reagieren soll, bleibt die zugehörige Prozedur eben leer.
- **Test-Fenster:** Es wird nur dann eingeblendet, wenn die Programmausführung unterbrochen oder schrittweise ausgeführt wird.Dann lassen sich hier Zwischenergebnisse auslesen oder Direktbefehle eingeben.

Das **Entwickeln von VB-Programmen** geschieht in folgenden Schritten:

- Laden der Form
- Plazieren von Steuerelementen auf das Formfenster
- Interaktives Ändern bestimmter Eigenschaften in Form und Steuerelementen
- Bestimmen der Ereignisse, auf die jedes Steuerelement reagieren soll
- Schreiben des Programmcodes (QBasic-Code, VB-Methoden, VB-Eigenschaften) in die automatisch erstellten Ereignis-Prozeduren.
- Ergänzungen durch zusätzliche Prozeduren, Deklarationen , VBX-Module etc.
- Testen des Programmes und Umwandlung in eine EXE-Datei.

VB4 gibt es in drei (deutschen) Varianten: Standard, Profi- und Enterprise-Ausgabe. Diese unterscheiden sich nicht nur im Preis, sondern auch durch die unterschiedlichen Wekzeugangebote , die bei der professionellen Version kaum Wünsche offen lassen. Für den Anfänger jedoch etwas verwirrend, reicht sicherlich die Standardversion aus, welche im Vergleich zur Version 3 einige Zusätze und Verbesserungen enthält.

Aben nun genug Theorie. Ein unterhaltsames Projekt soll den praktischen Einsatz von VB demonstrieren.

5. Projekt : Casino

Projektziel

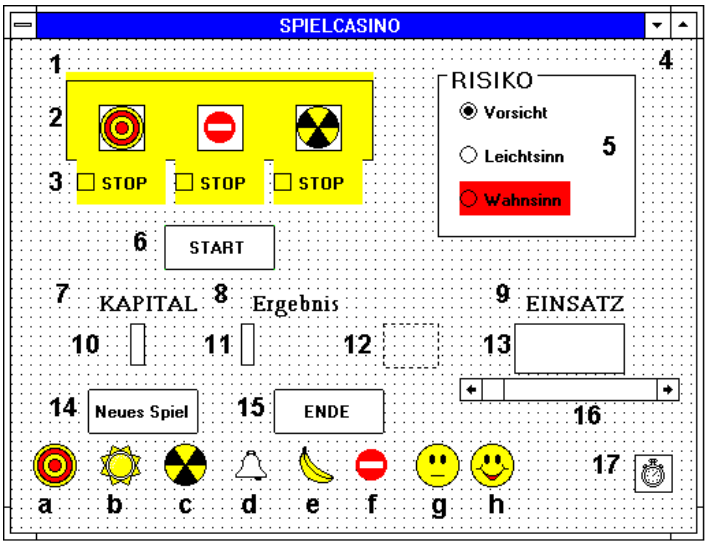
Am Bildschirm soll ein „einarmiger Bandit“ nachgebildet werden:Dieser besteht aus einem Eingabefeld bzw. einer Bildlaufleiste für den Einsatz, drei Bildfeldern für die rotierenden Symbole mit je einem Stoppknopf, einer Optionsfeldergruppe zur Auswahl des Risikos, sowie Anzeigefelder für Ergebnis und vorhandenes Kapital.Weiters sind 2 Befehlsschaltflächen für Start,Neues Spiel und Ende vorgesehen. Gewinn oder Verlust soll durch eine lachendes oder trauriges Iconengesicht angezeigt werden.

Spielregeln

Vor dem Starten des Spieles muß ein Einsatzbetrag vorhanden sein.Die rotierenden Symbole stoppen nach einer bestimmten Zeit von selbst, wenn sie nicht durch die Stopp-Knöpfe angehalten wurden.

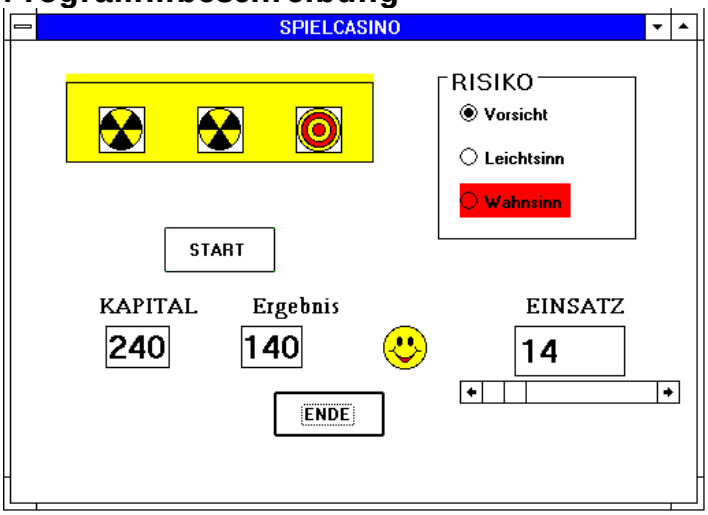
Bei 2 oder 3 gleichen Iconen wird der Einsatz vervielfacht, je nach dem eingestellten Risiko. Erscheint jedoch in einem der 3 Bildfeldern ein Einbahnzeichen, wird der Gewinn negativ.Die Wahrscheinlichkeit zu verlieren ist bei hohem Risiko größer: Es steigt die Anzahl der ladbaren Bilder und die Häufigkeit der Einbahnzeichen wird größer. Wird das Kapital Null oder negativ, muß das Spiel beendet werden.

Benutzeroberfläche



Nr	*Objekt*	*Name*	*Eigenschaften*	*Ereignisse/Methoden/Funktion*
1	Rahmen	`Rahmen2`		
2	Bildfeld	`Bild1(0-2)`	Picture	
3	KontrollKästchen	`Stopp(0-2)`	Caption,Visible,Value	
4	Rahmen	`Rahmen1`	Caption,Font...	
5	Optionsfeld	`Risiko(0-2)`	Caption,Value	
6	Befehlsschaltfläche	`Start`	Caption,Value,Font..,Enabled	Click
7,8,9	Bezeichnungsfeld	`Bezeichnung`	Caption,Font..	
10	Bezeichnungsfeld	`Kapitalfeld`	Autosize,Backcolor,Alignment,Font..	
11	Bezeichnungsfeld	`Ergebnisfeld`	Autosize,Backcolor,Alignment,Font..	
12	Anzeige	`Anzeige1`	Visible,Picture	
13	Textfeld	`Einsatzfeld`	Alignment,Font..,Text	Change,Setfocus
14	Befehlsschaltfläche	`NeuSpiel`	Caption,Font..,Visible	Click,Setfocus
15	Befehlsschaltfläche	`Ende`	Caption,Font..,Visible	Click
16	HBildlauf	`HBildlauf1`	Min,Max,Value,Small-,LargeChange	Change
17	Zeitgeber	`Zeit1`	Enabled,Interval	Timer
a..h	Anzeigefelder	`Anzeige2(0-7)`	Iconen aus VB\ICONS\.....	

Programmbeschreibung



Anfangs sind die globalen Variablen zu definieren und beim Laden der Startform die Anfangswerte für Laufzeit und Bildlaufleiste zu setzen.Das Anfangskapital ist hier mit 100 angenommen.

Als Nächstes ist die Eingabe des Einsatzes zu Programmieren. Dies geschieht entweder über das Textfeld über das Ereignis `Einsatzfeld_Change` oder durch Betätigung der Bildlaufleiste. Beide Arten sind zu synchronisieren, der Einsatzbetrag der globalen Variablen Einsatz zuzuweisen und im Einsatzfeld formatiert auszugeben. Nun kann der Startknopf betätigt werden: Einleitend muß überprüft werden, ob noch Spielkapital vorhanden ist, ob ein Einsatz vorhanden ist und ob letzterer nicht größer als das Spielkapital ist. Ist alles o.k. dann werden die Anhalteknöpfe aktiviert, die Einsatzeingabe gesperrt, der aktuelle Risikofaktor in Abhängigkeit vom aktiven Optionsfeld bestimmt und die Wahlfläche für neues Spiel unsichtbar. Das Rotieren der Bilder wird schließlich durch Setzen des Zeitintervalles der Uhr gestartet. Die Prozedur `Zeit1_Timer` wird danach alle 100 Millisekunden ausgeführt, solange bis `Zeit1.Interval` auf Null gesetzt ist. In dieser Prozedur wird für jedes noch nicht gestoppte Bildfeld entweder eine Icone oder das Einbahnzeichen geladen. Die Bildauswahl erfolgt zufällig aus einer bestimmten Bilderzahl, welche vom Risikofaktor abhängt: höheres Risiko = mehr Bilder zur Auswahl.

Auch die Wahrscheinlichkeit, daß eine Einbahnstraße geladen wird, wird vom Risikofaktor gesteuert.

Die Restlaufzeit wird immer um das Timer-Intervall verkützt, sodaß nach einiger Zeit entweder alle 3 Bilder durch Stoppknöpfe oder nach Ablauf der Restzeit zum Stillstand kommen. Ist dies der Fall, wird ein Unterprogramm `ERGEBNIS` zur Ermittlung des Gewinnes oder Verlustes aufgerufen. Dieses Programm stellt keine Ereignisprozedur dar, sondern ist mit dem Menüpunkt Ansicht/Neue Prozedur zu erstellen und wird bei den allgemeinen Programmteilen zusammen mit den Deklarationen abgelegt.

Das Spielergebnis wird durch Multiplizieren des Einsatzes mit einem Faktor berechnet, welcher vom Risikofaktor und von der Anzahl der gleichen Bilder abhängt. Eine geladene Einbahnstraße macht den Gewinn negativ. Gewinn (Verlust) und aktuelles Kapital werden zusammen mit einem lachenden oder weinendem Gesicht angezeigt. Der Spieler kann nun das Spiel beenden oder weiter sein Glück versuchen.

Ausbauvorschläge: Eingabe des Spielernamens und Abspeichern des „Scores". Auf Wunsch sollen die 10 besten Spieler aufgelistet werden.

PROGRAMM

```
'Globale Variable definieren
   Dim Laufzeit As Integer           'Laufzeit für rotierende Iconen
   Dim Riskfaktor As Integer         'ausgewähltes Risiko
   Dim Kapital As Integer            'Spielkapital
   Dim Bildindex(0 To 2) As Integer  'Iconen Nummer
   Dim Einsatz As Integer            'Eingesetzter Betrag
   Dim Gewinn As Integer             'Gewonnener Betrag

Sub Form_Load ()
   Kapital = 100            'Anfangswerte
   Einsatz = 0
   Gewinn = 0
   KapitalFeld.Caption = Format$(Kapital, "#####")
   HBildlauf1.Min = 0          'Eigenschaften der Bildlaufleiste
   HBildlauf1.Max = Kapital    'Größter Einsatz=Kapital
   HBildlauf1.Value = 0#
   HBildlauf1.LargeChange = 10
   HBildlauf1.SmallChange = 1
End Sub

Sub Einsatzfeld_Change ()
   'Anpassen der Bildlaufleistenstellung je nach dem
   'im Einsatzfeld eingegebenen Betrag
   If Val(Einsatzfeld.Text) <= Kapital Then
        HBildlauf1.Value = Val(Einsatzfeld.Text)
   End If
   Einsatz = Val(Einsatzfeld.Text)  ' Einsatzbetrag
End Sub

Sub HBildlauf1_Change ()
   'Bildlaufleiste wird für Einsatz benutzt
   Einsatz = HBildlauf1.Value
   'Übertragen ins Textfeld
   Einsatzfeld.Text = Format$(Einsatz, "#####")
End Sub

Sub Start_Click ()
   'Das Spiel beginnt
   If Kapital <= 0 Then
      'Ohne Geld ka Musi!
      MsgBox "Leider! Das Spiel ist aus!"
      Ende.Visible = True    'Spielende möglich
      NeuSpiel.Visible = True  'Oder ein neues Spiel
      NeuSpiel.SetFocus   'Diese Schaltfläche erhält Focus
      Exit Sub     'Ausstieg aus Unterprogramm
   End If
   If Einsatz <= 0 Then
      'Für Vergeßliche und Schwindler
      MsgBox "Dein Einsatz bitte!" 'Höfliche Mitteilung
      Start.Value = False    'Startknopf wird deaktiviert
      Einsatzfeld.SetFocus   'Eingabe im Einsatzfeld wird erwartet
      Exit Sub
   End If
   If Einsatz > Kapital Then
      'No Credit
      MsgBox "Einsatz > Kapital!"
      Einsatzfeld.SetFocus
      Exit Sub
   End If
   Stopp(0).Visible = True 'Stoppknöpfe werden sichtbar
   Stopp(0).Value = False  ' und auf AUS gestellt
   Stopp(1).Visible = True
   Stopp(1).Value = False
   Stopp(2).Visible = True
   Stopp(2).Value = False
   Einsatzfeld.Enabled = False  ' Nichts geht mehr!
   HBildlauf1.Enabled = False   ' Keine Einsatzänderung möglich
   Rem Risikofaktor je nach gewählter Risiko-Option
   If Risiko(0).Value = True Then
        Riskfaktor = 3
   ElseIf Risiko(0).Value = True Then
        Riskfaktor = 4
   Else
        Riskfaktor = 5
   End If
   'Wahlfläche für neues Spiel unsichtbar
   NeuSpiel.Visible = False
   Laufzeit = 4000         ' Laufzeit der Iconen ohne Stopps
   Zeit1.Interval = 100    ' Zeitnehmer Intervall auf 100 Millisekuden,
Timer-Start
End Sub

Sub Zeit1_Timer ()
    'Dieses Programm wird je nach dem Wert für Zeit1.Interval
    'regelmäßig aufgerufen
    Randomize Timer      'Anfangswert für Zufallszahlen nach PC-Uhr
    'Laden zufälliger Iconen in die 3 Bildfelder
    For bild% = 0 To 2
          r% = 0    'Schalter für Einbahnstraßen-Icone
          t1% = Int(Rnd * 100) + 1     'Zufallszahl zw. 1 und 100
          'Wahrscheinlichkeit für Einbahnstraße je nach Risiko
          Select Case Riskfaktor
             Case 3
                 If t1% < 5 Then r% = 1   'Nur wenn Zufallszahl < 5
             Case 4
                 If t1% < 10 Then r% = 1  'wenn Zufallszahl < 10
             Case 5
                 If t1% < 15 Then r% = 1  'wenn Zufallszahl < 1
          End Select
          ' Anzahl der möglichen Iconen je nach Risiko
          t% = Int(Rnd * Riskfaktor)  'Icone zum Laden
          'Laden nur wenn Stoppknopf noch nicht eingeschaltet
          If Stopp(bild%).Value = 0 Then
           If r% = 0 Then
              'Bild wird geleden
              Bild1(bild%).Picture = Anzeige2(t%).Picture
              Bildindex(bild%) = t%
           Else
              'Einbahnstraße wird geladen
              Bild1(bild%).Picture = Anzeige2(5).Picture
              Bildindex(bild%) = 5    'Einbahnstraße
           End If
         End If
     Next bild%

     'Restlaufzeit wird um Timer-Intervall verkürzt
     'Ausschalten des Timers wenn alle 3 Bilder stillstehen
     Laufzeit = Laufzeit - Zeit1.Interval
     If Stopp(0).Value = 1 And Stopp(1).Value = 1 And \
        Stopp(2).Value = 1 Or Laufzeit <= 0 Then
         Zeit1.Interval = 0      'Zeitnehmer aus
         Stopp(0).Visible Stopp(0).Visible= False  'Stoppknöpfe unsichtbar
         Stopp(1).Visible = False
         Stopp(2).Visible = False
         Start.Value = False       'Startknopf inaktivieren
         Einsatzfeld.Enabled = True  'Einsatzfeld aktivieren
         HBildlauf1.Enabled = True  'Bildlaufleiste ebenfalls
         Call Ergebnis     'Ergebnisberechnung
     End If
End Sub

Sub Ergebnis ()
    Rem Ergebnis berechnen und ausgeben
    Select Case Riskfaktor      'Gewinnfaktor je nach Risikowahl
       Case 3
          faktor% = 2
       Case 4
          faktor% = 4
       Case 5
          faktor% = 8
    End Select

    'Gibt es gleiche Bilder?
    If Bildindex(0) = Bildindex(1) And Bildindex(0) = Bildindex(2) Then
        Gewinn = Einsatz * faktor% * 10 ' 3 gleiche
    ElseIf Bildindex(0) = Bildindex(1) Or \
           Bildindex(0) = Bildindex(2) Or Bildindex(1) = Bildindex(2) Then
        Gewinn = Einsatz * faktor% * 5  ' 2 gleiche
    Else
        Gewinn = 0 ' alles verschieden
    End If
    'Einbahnstraße macht Gewinn zu Verlust!
    If Bildindex(0) = 5 Or Bildindex(1) = 5 Or Bildindex(2) = 5 Then
         Gewinn = Gewinn * -.5
    End If
     ' Ausgabe im Ergebnisfeld und Kapitalfeld
    ErgebnisFeld.Caption = Format$(Gewinn, "#####")
    Kapital = Kapital + Gewinn
    KapitalFeld.Caption = Format$(Kapital, "#####")
    HBildlauf1.Max = Kapital
    'Lachendes oder weinendes Gesicht
    If Gewinn > 0 Then
         Anzeige1.Picture = Anzeige2(7).Picture
    Else
         Anzeige1.Picture = Anzeige2(6).Picture
    End If
         Anzeige1.Visible = True
    Start.SetFocus    'Aktuelle Eingabe auf Startknopf
End Sub

Sub NeuSpiel_Click ()
   'Intialisierung der Startvariablen
   Kapital = 100
   Einsatz = 0
   Gewinn = 0
   KapitalFeld.Caption = Format$(Kapital, "#####")
   HBildlauf1.Min = 0
   HBildlauf1.Max = Kapital
   HBildlauf1.Value = 0#
   HBildlauf1.LargeChange = 10
   HBildlauf1.SmallChange = 1
   ErgebnisFeld.Caption = Format$(Gewinn, "#####")
   Risiko(0).Value = 1
End Sub

Sub Ende_Click
        End     'Beenden des Programmes
End Sub
```

Prof. Dipl.Ing.
Dr. Hermann Köberl
HTL, Ettenreichgasse 54, 1100 Wien
Lorenzgasse 23, 2700 Wr. Neustadt □