

3 Grundlagen der Programmentwicklung

In diesem Kapitel erfahren Sie

- wie das Prinzip der ereignisgesteuerten Programmierung funktioniert
- wie Sie eine Benutzeroberfläche erstellen und welche Steuerelemente es gibt
- welche Eigenschaften Steuerelemente haben und wie Sie diese festlegen
- wie Sie Ereignisprozeduren erstellen und ausführen, und welche Ereignisse es gibt
- wie Sie eine ausführbare Programmdatei erstellen

Voraussetzungen

- ✓ Kenntnisse der Visual Basic-Entwicklungsumgebung


3.1 Mit Visual Basic programmieren

3.1.1 Das Prinzip der ereignisgesteuerten Programmierung

Was sind Ereignisse?

Die Visual Basic-Programmierung wird als ereignisorientierte Programmierung bezeichnet. **Ein Ereignis ist eine Aktion, die den Programmablauf beeinflusst** (vgl. Abschnitt 3.4).

Alle **Aktionen (Tastatureingaben, Mausbewegungen) eines Benutzers**, wie beispielsweise

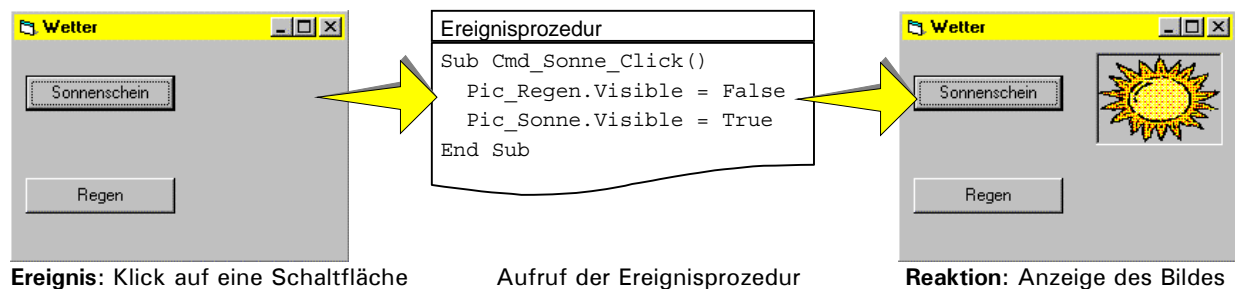
- Klicken oder Doppelklicken einer Schaltfläche
- Verschieben, Öffnen oder Schließen eines Fensters mit der Maus
- Positionieren des Cursors in ein Eingabefeld mit der -Taste

sind **Ereignisse**.

Über Ereignisse können **interne Programmabläufe**, wie beispielsweise

- Berechnungen durchführen
- Öffnen und Schließen eines Fensters (vom Programm gesteuert)
- Ermitteln der aktuellen Uhrzeit und des aktuellen Datums

ausgelöst werden.



Diese Beispiel wird später aufgegriffen und in seinen Einzelheiten erläutert.

Auf Ereignisse reagieren

Mit Visual Basic können Programme entwickelt werden, die durch die grafische Benutzeroberfläche Windows mit Steuerelementen wie beispielsweise Dialogfenstern, Schaltflächen, Eingabefeldern gesteuert werden.

Durch ein **Ereignis** (beispielsweise Klicken, Doppelklicken einer Schaltfläche) kann eine **Reaktion** (beispielsweise Öffnen eines Fensters) hervorgerufen werden. Diese Reaktion wird in Form von Visual Basic-Anweisungen in einer Einheit erstellt, die als **Ereignisprozedur** bezeichnet wird. Eine Ereignisprozedur enthält die Anweisungen, die ausgeführt werden, sobald ein Ereignis eintritt.

Visual Basic kennt auch Prozeduren, die nicht ereignisorientiert sind. Prozeduren werden in weitere Einheiten zusammengefaßt, die als **Module** bezeichnet werden.

Jede Form und jedes Steuerelement (beispielsweise Schaltfläche, Eingabefeld, Listenfeld) besitzt eine Anzahl von möglichen Ereignissen. Wenn eines dieser Ereignisse eintritt, ruft Visual Basic die entsprechende **Ereignisprozedur** auf. Die Anweisungen innerhalb der aufgerufenen Ereignisprozedur werden von oben nach unten abgearbeitet (zum Erstellen von Ereignisprozeduren vgl. Abschnitt 3.4).

3.1.2 Vorgehensweise bei der Erstellung von Anwendungen

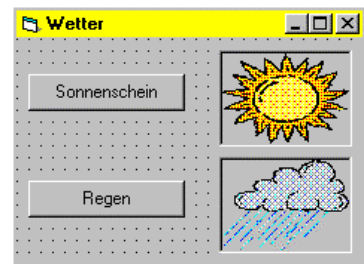
1. Schritt: Problemstellung analysieren

Zu Beginn muß die Aufgabenstellung präzisiert werden: Alle notwendigen Informationen müssen gesammelt werden, die für die Lösung des Problems wichtig sind. Danach können die wichtigsten Objekte (Formen, Steuerelemente und andere Programmelemente, die für den Programmablauf notwendig sind) festgelegt und der Programmablauf strukturiert werden.



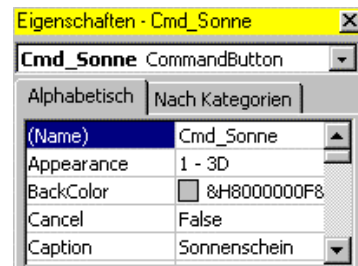
2. Schritt: Benutzeroberfläche erstellen

Durch Anordnen aller notwendigen Steuerelemente (z.B. Eingabefelder, Optionsfelder, Schaltflächen etc.) auf der Form wird die Benutzeroberfläche des Programms erstellt.



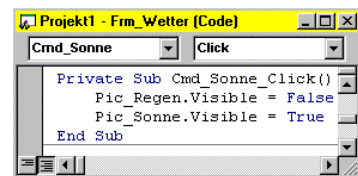
3. Schritt: Eigenschaften festlegen

Für die zuvor erstellten Steuerelemente werden die Eigenschaften festgelegt, beispielsweise Größe, Farbe, Bezeichnung der Schaltflächen, Text- und Eingabefelder.



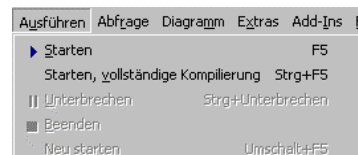
4. Schritt: Ereignisprozeduren codieren

In einer Ereignisprozedur wird in Form von Visual Basic-Anweisungen beschrieben, wie ein bestimmtes Steuerelement auf ein bestimmtes Ereignis (beispielsweise Klicken, Doppelklicken einer Schaltfläche) reagieren soll.



5. Schritt: Ausführen bzw. Testen der Anwendung

Um die Richtigkeit des Programms zu prüfen, muß es an verschiedenen Beispielen getestet werden. Treten während des Tests Fehler auf, müssen die oben genannten Schritte wiederholt werden.



6. Schritt: Ausführbare Programmdatei erstellen

Weist die Anwendung keine Fehler mehr auf, kann eine Programmdatei erstellt werden, die unabhängig von Visual Basic ausgeführt und vom Windows Explorer bzw. Programm-Manager aufgerufen werden kann.



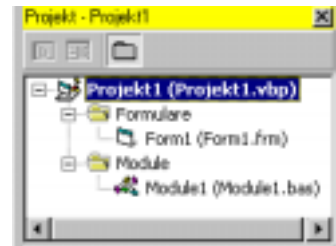
PROGRAMM.EXE

3.1.3 Aufbau einer Visual Basic-Anwendung

Anwendungen werden unter Visual Basic in Form von Projekten verwaltet. Alle Dateien, die zur jeweiligen Anwendung gehören, werden zusammen in einem Projekt gespeichert.

Eine typische Visual Basic-Anwendung besteht aus mindestens zwei Dateien, der Projektdatei und einem Formular.

- ☑ In der Projektdatei (*.vbp) sind alle zum Projekt gehörenden Dateien aufgeführt.
- ☑ Das Formular (in Visual Basic als Form bezeichnet) wird in einer eigenen Datei (*.frm) abgelegt.
- ☑ Haben Sie allgemeine Prozeduren (*.bas) formuliert, werden diese im automatisch angelegten Ordner Module verwaltet.



3.2 Mit Projekten arbeiten

Neues Projekt erstellen

Standardmäßig wird Visual Basic mit einem neuen, leeren Projekt geöffnet. Sie können ein neues Projekt aber auch wie folgt erzeugen:

⇒ Rufen Sie den Menüpunkt DATEI - NEUES PROJEKT auf.

Alternativen: **STRG N** oder

Ist bereits ein Projekt geöffnet, wird dieses geschlossen. Sofern Sie im aktuellen Projekt Änderungen vorgenommen und diese noch nicht gespeichert haben, werden Sie in mehreren Dialogfenstern aufgefordert, das aktuelle Projekt zu speichern.

⇒ Klicken Sie doppelt auf das Symbol STANDARD-EXE.

Anschließend wird ein neues Projekt begonnen.



Projekt speichern

⇒ Wählen Sie den Menüpunkt DATEI - PROJEKT SPEICHERN.

Alternative:

Visual Basic fordert Sie beim ersten Speichervorgang auf, die gerade bearbeitete Form bzw. das neu erstellte Modul zu benennen.

⇒ Geben Sie den Form- bzw. Modulnamen ein.

Visual Basic vergibt für Formdateien automatisch die Dateierweiterung FRM. Module erhalten die Dateierweiterung BAS.

Danach fordert Sie Visual Basic auf, das gesamte Projekt zu speichern.

⇒ Geben Sie den Projektnamen ein.

Visual Basic vergibt für den Projektnamen automatisch die Dateierweiterung VBP.

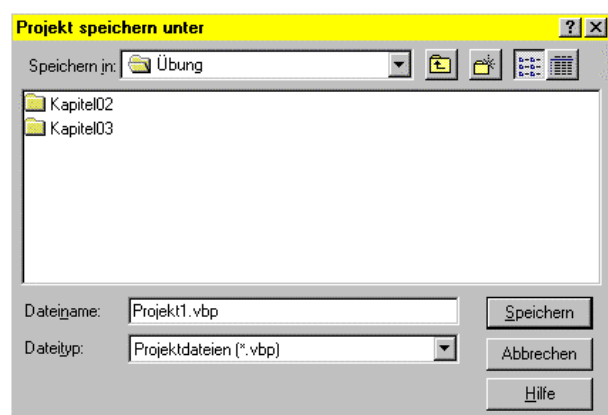


Abb. 3.1 Projekt speichern

Sie sollten beim Entwerfen von Dialogen und Formen bestimmte Regeln einhalten, um beispielsweise den Dialogen ein einheitliches Erscheinungsbild zu geben, die Dialoge benutzerfreundlich zu gestalten und dem Benutzer ein unnötiges Suchen nach Informationen zu ersparen.








Allgemeine Hinweise zur Gestaltung von Dialogen finden Sie beispielsweise unter Visual Basic-Dokumentation/Arbeiten mit Visual Basic/Programmierhandbuch/Teil 2: Einsatzmöglichkeiten für Visual Basic/Erstellen einer Benutzeroberfläche/Entwerfen von Anwendungen im Hinblick auf den Benutzer/Grundlagen des Oberflächenentwurfs.


















Verwenden Sie...	
einprägsame Namen	Vergeben Sie einen einprägsamen, aussagekräftigen Namen für den Dialogtitel. Der Benutzer sollte immer wissen, wozu der Dialog dient.
Gruppierungen	Gruppieren Sie zusammengehörende Steuerelemente, und versehen Sie diese mit einer Beschriftung.
die Schaltflächen OK und ABBRECHEN	Diese Standard-Schaltflächen sollten auf jedem Dialog enthalten und rechts oben platziert sein, um dem Anwender unnötiges Suchen zu ersparen.
weitere Schaltflächen	Sollen auf Ihrem Dialog weitere Schaltflächen enthalten sein, platzieren Sie diese leicht abgesetzt unter den Schaltflächen OK und ABBRECHEN. Achten Sie auch auf Einheitlichkeit der Schaltflächen in Größe und Ausrichtung (vgl. Abschnitt 4.8).
die Schaltfläche HILFE	Enthält Ihr Dialog die Schaltfläche HILFE, sollte diese rechts unten auf dem Dialog platziert sein.
nicht zu viele Steuerelemente auf einem Dialog	Vermeiden Sie "überladene" Dialoge. Ihr Dialog sollte immer übersichtlich sein. Der Anwender sollte nach Informationen nicht lange suchen müssen. Verwenden Sie gegebenenfalls Register oder Erweiterungen, z.B. über eine Schaltfläche OPTIONEN.

3.3.2 Überblick über die Steuerelemente

In der Werkzeugsammlung befinden sich zahlreiche Steuerelemente, die zur Gestaltung einer Form bzw. für die Anwendung benötigt werden.

 Cursor	Der Cursor ist selbst kein Steuerelement. Ist diese Schaltfläche aktiviert, können Steuerelemente auf der Form verschoben oder in ihrer Größe verändert werden.
 PictureBox (Bildfeld)	Zur Anzeige von Text, Bildern (Bitmap, Icon bzw. Symbol, Metadatei) und Grafiken (vgl. Image)
 Label (Bezeichnungsfeld)	Zur Anzeige von erläuterndem Text; sie geben beispielsweise Informationen über Steuerelementinhalte. Der Text kann vom Anwender nicht bearbeitet werden.
 TextBox (Eingabefeld)	Zur Abfrage von Benutzereingaben (Texteingaben über die Tastatur)
 Frame (Rahmen)	Steuerelemente, beispielsweise Options- oder Kontrollfelder, können in einem Rahmen gruppiert werden.
 CommandButton (Schaltfläche)	Schaltflächen werden eingesetzt, um Befehle bzw. Ereignisprozeduren beispielsweise durch Klicken oder Doppelklicken zu aktivieren.
 CheckBox (Kontrollfeld)	Um aus einer Liste von Optionen keine , mehrere oder alle auszuwählen; Kontrollfelder können eingeschaltet und ausgeschaltet werden.

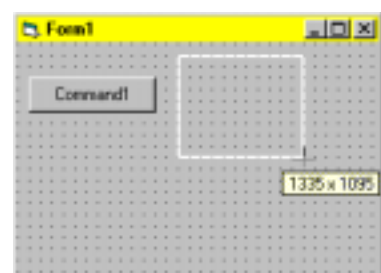
Haben Sie beim Starten von Visual Basic das Symbol VB EINSTEIGER EDITION-STEUERELEMENTE gewählt (im Dialogfenster NEUES PROJEKT, Register NEU), werden Ihnen in der Werkzeugsammlung weitere Steuerelemente zur Verfügung gestellt.

 SSTab (Register)	Zum Entwerfen von Registerdialogen; der Registerdialog erscheint bereits zur Entwurfszeit so, wie er zur Laufzeit aussieht.
 MSFlexGrid (Datengebundene Tabelle)	Zur Anzeige von Datentabellen (in Verbindung mit dem Datensteuerelement); eine Bearbeitung der Daten ist nicht möglich.
 CommonDialog (Standarddialog)	Stellt die gemeinsamen Windows-Dialogfenster zur Verfügung, wie beispielsweise Dialogfenster zum Speichern oder Drucken
 TabStrip (Register)	Zum Entwerfen von Registerdialogen; der Registerdialog erscheint erst zur Laufzeit korrekt.
 ToolBar (Symbolleiste)	Zum Erstellen von Windows-Symbolleisten
 StatusBar (Statuszeile)	Zum Erstellen einer Windows-Statuszeile
 ProgressBar (Fortschrittsleiste)	Zur Anzeige des Fortschritts eines länger anhaltenden Prozesses, beispielsweise eines Speichervorgangs
 TreeView (Hierarchisches Listenfeld)	Zur hierarchischen Anzeige von Elementen einer Liste, beispielsweise zum Auflisten von Verzeichnissen und Dateien
 ListView (Listenansicht)	Zur Anzeige von Elementen in einer Liste; ähnlich dem Windows-Explorer können verschiedene Ansichten definiert werden (Elemente als Text oder als Symbole).
 ImageList (Abbildungsliste)	Zum Erstellen von Bilderlisten; auf jedes Bild kann durch einen Index zugegriffen werden. Das Steuerelement wird meist dazu verwendet, andere Steuerelemente (beispielsweise Symbolleisten) mit Abbildungen zu beliefern.
 Slider (Schieberegler)	Zur Abfrage eines Wertes aus einem festgelegten Wertebereich mit Hilfe eines Schiebereglers
 ImageCombo	Zur Auswahl eines Elementes aus einem Listenfeld und/oder zur Eingabe eines Textes in ein Eingabefeld; zusätzlich kann neben jedem Listeneintrag ein Bitmap angezeigt werden.
 DataGrid (Datengebundene Tabelle)	Zur Anzeige von Datentabellen (in Verbindung mit dem Datensteuerelement); eine Bearbeitung der Daten (ändern, einfügen, löschen) ist möglich.
 Adodc (Datenbankfeld)	Dient als Datenquelle für die damit verbundenen Steuerelemente; unterstützt beim Bewegen durch die Datensätze
 MSHFlexGrid (Datengebundene Tabelle)	Erweiterte Funktion des MSFlexGrid; Bearbeitung der Daten ist ebenfalls nicht möglich.



3.3.3 Steuerelemente einer Form hinzufügen

Die Form ist in ein Raster eingeteilt, um die Steuerelemente genau anordnen zu können. Zur Laufzeit ist dieses nicht sichtbar. Einige Steuerelemente sind nur in der Entwurfsansicht sichtbar, beispielsweise der Zeitgeber.

- ⇒ Klicken Sie auf das Steuerelement in der Werkzeugsammlung, das Sie der Form hinzufügen möchten.
Es erscheint nun aktiviert.
- ⇒ Ziehen Sie in der Form bei gedrückter linker Maustaste das Steuerelement auf die gewünschte Größe.
- ⇒ Wenn Sie die Maustaste wieder loslassen, erscheint das Steuerelement auf der Form.



Überblick wichtiger Eigenschaften

Alignment	Text kann rechtsbündig, linksbündig oder zentriert ausgegeben werden.	Height	Die Höhe des Objektes wird festgelegt.
BackColor	Die Hintergrundfarbe des Objektes kann verändert werden.	HelpContextID	Einstellen der Hilfetextnummer, die mit dem Objekt verknüpft ist
BorderStyle	Die Art der Objekt-Umrahmung kann bestimmt werden.	Interval	Der Abstand, wann ein Zeitgeber aktiviert wird, wird festgelegt.
Cancel	Legt fest, ob es sich um die Schaltfläche ABBRECHEN handelt	Left	Die Position (X) der linken oberen Ecke des Objektes wird festgelegt.
Caption	Stellt die Beschriftung bzw. Überschrift eines Objektes dar	MousePointer	Das Aussehen des Mauszeigers kann bestimmt werden (Pfeil, Kreuz etc.).
DataField	Daten können mit einem Daten-Steurelement verknüpft werden.	Name	Gibt den Namen bzw. die interne Bezeichnung des Objektes an
Default	Eine Schaltfläche ist ausgewählt und kann mit  bestätigt werden.	Picture	Ein Bildfeld kann mit einem Inhalt versehen werden.
DragIcon	Festlegen des Mauszeigers beim Ziehen eines Objektes	TabStop	Ein Steurelement kann mit der  -Taste angesprungen werden.
DragMode	Wechseln zwischen automatischem/manuellem Ziehen eines Objektes	Text	Enthält den Text des Steurelementes
Enabled	Aktiviert das Objekt, um beispielsweise auf Ereignisse zu reagieren	Top	Die Position (Y) der linken oberen Ecke kann eingestellt werden.
Font	Textformatierung wie Schriftart, Schriftschnitt, Schriftgröße etc.	Visible	Objekte können während des Programmlaufs sichtbar/unsichtbar sein.
ForeColor	Legt die Schriftfarbe des Textes fest	Width	Die Breite eines Objektes wird festgelegt.

Manche Eigenschaften haben für verschiedene Objekte unterschiedliche Bedeutung. Damit stehen Ihnen für jedes Objekt unterschiedliche Werte als Einstellung zur Verfügung.



BEISPIEL

Steuerelement	Bedeutung der Eigenschaft <code>Value</code>	Einstellungsmöglichkeiten
Kontrollfeld	Ein Kontrollfeld ist aktiviert oder nicht aktiviert.	0 - nicht ausgewählt 1 - ausgewählt 2 - abgeblendet
Optionsfeld	Ein Optionsfeld ist aktiviert oder nicht aktiviert.	<i>True</i> (wahr) - ausgewählt <i>False</i> (falsch) - nicht ausgewählt
Schaltfläche	Eine Schaltfläche wird gedrückt oder nicht gedrückt.	<i>True</i> - gedrückt <i>False</i> - nicht gedrückt
Horizontale und vertikale Bildlaufleisten	Die Position des Bildlaufeldes einer Bildlaufleiste wird eingestellt.	Werte zwischen -32768 und 32767 bzw. der Eigenschaften-Einstellungen <i>Min</i> und <i>Max</i>

3.3.5 Beispiel zur Verwendung von Steuerelementen

Sie wollen ein Programm *Wetter* erstellen, das die beiden Schaltflächen *Sonnenschein* und *Regen* sowie zwei Bildfelder enthält. Bei Klick auf eine der Schaltflächen soll im jeweiligen Bildfeld die entsprechende Grafik angezeigt werden.

3.4 Ereignisprozeduren erstellen

Jedes Steuerelement und jede Form kann auf verschiedene Ereignisse (beispielsweise Klicken einer Schaltfläche) reagieren. Dazu müssen Sie sogenannte Ereignisprozeduren programmieren. Die Ereignisse, die einem Objekt zugeordnet sind, sind vom jeweiligen Objekttyp abhängig.

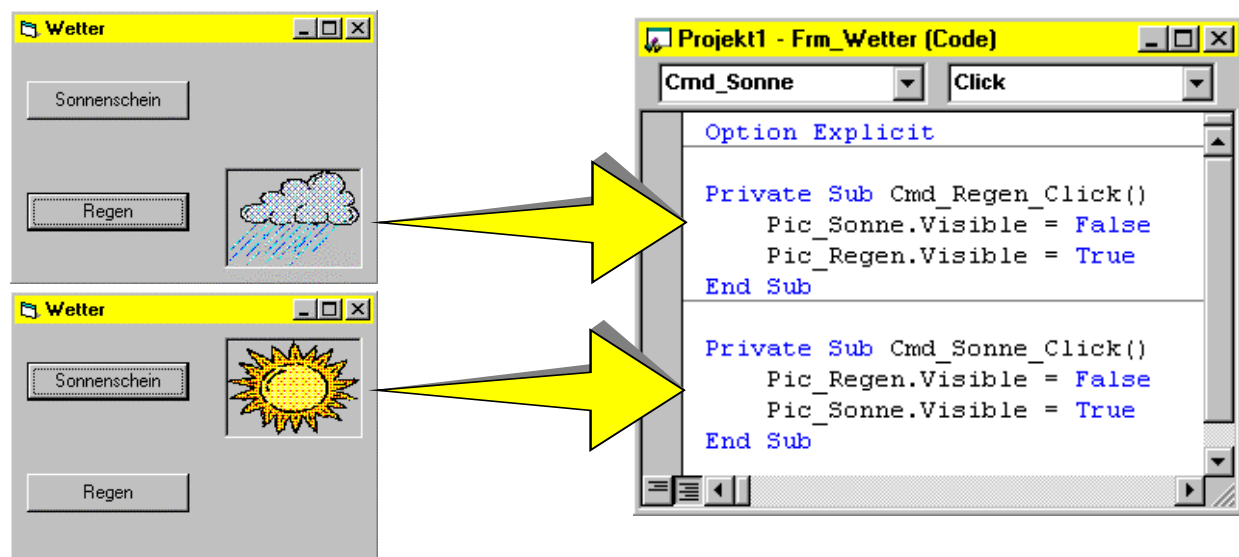
Prozeduren werden in **Ereignisprozeduren** und **nicht ereignisorientierte Prozeduren** unterteilt (vgl. Kapitel 10).

Der Programmcode wird in Visual Basic in Blöcken geschrieben, die als Prozeduren bezeichnet werden. Tritt das Ereignis nicht ein, wird die Ereignisprozedur auch nicht ausgeführt.



Das Programm *Wetter* besitzt die Schaltflächen *Sonnenschein* und *Regen*. Folgende Ereignisse und Reaktionen sollen möglich sein:

- Die Schaltfläche *Regen* wird angeklickt. Als Reaktion wird die Ereignisprozedur *Cmd_Regen_Click* abgearbeitet.
- Die Schaltfläche *Sonnenschein* wird angeklickt. Als Reaktion wird die Ereignisprozedur *Cmd_Sonne_Click* abgearbeitet.



Wenn keine der beiden Schaltflächen angeklickt wird, wird auch keine Ereignisprozedur ausgeführt.



Falls Sie nachträglich den Namen eines Objektes ändern, zu dem Sie schon Ereignisprozeduren programmiert haben, müssen Sie die Namen der Prozeduren anpassen.

Überblick wichtiger Ereignisse

Activate	Eine Form wird aktiviert, beispielsweise wenn sie angeklickt wird.
Change	Der Inhalt eines Eingabefeldes oder die Position eines Bildlaufes in einer Bildlaufleiste ändert sich.
Click	Eine Schaltfläche wird angeklickt oder über die Leertaste ein Optionsfeld aktiviert.
DbClick	Auf einem Element in einem Listenfeld wird ein Doppelklick ausgeführt.
Deactivate	Eine Form wird deaktiviert, z.B. wenn ein anderes Objekt der Anwendung angeklickt wird.
DragDrop	Ein Rahmen wird gezogen (drag) und losgelassen (drop), z.B. um ihn zu verschieben.
DropDown	Ein Listenfeld wird aufgeschlagen.
Error	Es wird beispielsweise versucht, auf ein Datenfeld zuzugreifen, das nicht existiert.
GotFocus	Eine Form wird z.B. durch einen Klick mit der Maustaste in den Vordergrund gestellt.
KeyDown	Eine beliebige Taste wird gedrückt.

3.5 Anweisungen eingeben und bearbeiten

Anweisungen eingeben

- ⇒ Klicken Sie im Codefenster an die Stelle, an der Sie die Anweisung eingeben wollen (immer zwischen `Sub Prozedurname` und `End Sub`), und geben Sie die Anweisungen ein.
- ⇒ Mit **RETURN** wird eine Anweisung beendet und in eine neue Codezeile gesprungen.

Mehrere Anweisungen	In einer Zeile können mehrere Anweisungen nacheinander eingegeben werden, wenn sie mit ":" getrennt werden.
Zeilenumbruch	Der Code einer Zeile wird in der nächsten Zeile fortgesetzt, wenn nach einem Leerzeichen ein Unterstrich <code>_</code> folgt. Beispiel: <code>Meldung = "In die Variable Meldung wird ein Text geschrieben und noch einer "</code> <code>& "in der nächsten Zeile mit _ und dem kaufmännischen UND angehängt"</code>
Sicherung von Codeteilen	Die Ereignis-Prozeduren gelöschter Steuerelemente werden im <i>Allgemeinen Deklarationssteil</i> als <i>benutzerdefinierte</i> Prozeduren gespeichert.
Syntax Checking	Jedesmal, wenn Sie die aktuelle Anweisungszeile mit der RETURN -Taste, der Maus oder über eine der Pfeiltasten verlassen, wird die Zeile sofort auf Syntaxfehler überprüft. Sie erhalten im Falle einer fehlerhaften Anweisung ein Dialogfenster mit einem Hinweis auf das gefundene Problem.



Codezeilen einrücken

Damit Ihre Anweisungen übersichtlich und gut lesbar angeordnet sind, können Sie einzelne Zeilen oder Bereiche der Prozedur einrücken:

- ⇒ Positionieren Sie den Cursor an den Anfang der Codezeile, und betätigen Sie die **↵**-Taste. *oder* Rufen Sie den Menüpunkt **BEARBEITEN - EINZUG VERGRÖßERN** auf.

Wechseln Sie mit **RETURN** in die nächste Zeile, wird diese ebenfalls eingezogen. Um den Einzug zu entfernen, betätigen Sie **← KORREKTUR**, oder wählen Sie den Menüpunkt **BEARBEITEN - EINZUG VERKLEINERN**.



Durch Klick auf die Schaltfläche *Sonnenschein* soll eine Ereignisprozedur aufgerufen werden, die das Bild *Sonne* ein- und das Bild *Regen* ausblendet.

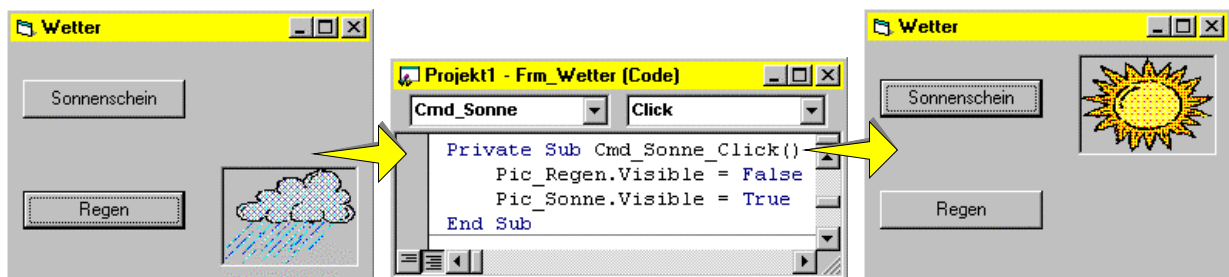


Abb. 3.5 Erstellen einer Ereignisprozedur

Aktionen rückgängig machen

Die zuletzt durchgeführte Eingabe oder der zuletzt ausgeführte Befehl kann im Visual Basic-Editor in den meisten Fällen wieder rückgängig gemacht werden.



- Im Codefenster können Sie die letzte oder mehrere codierte Anweisung(en) rückgängig machen.
- Die letzte Löschung oder der letzte Kopiervorgang kann rückgängig gemacht werden.
- Im Formfenster können Sie Steuerelemente wiederherstellen, die Sie gerade gelöscht haben.

- ⇒ Wählen Sie den Menüpunkt **BEARBEITEN - RÜCKGÄNGIG**, um eine Aktion zurückzunehmen.



Sind in einem Projekt mehrere Formen enthalten, muß festgelegt werden, welche Form beim Starten des Programmes zuerst angezeigt wird (vgl. Abschnitt 9.5).

- ⇒ Rufen Sie den Menüpunkt **PROJEKT - EIGENSCHAFTEN VON NAME** auf.
- ⇒ Wählen Sie im Listenfeld ① die gewünschte Form aus.

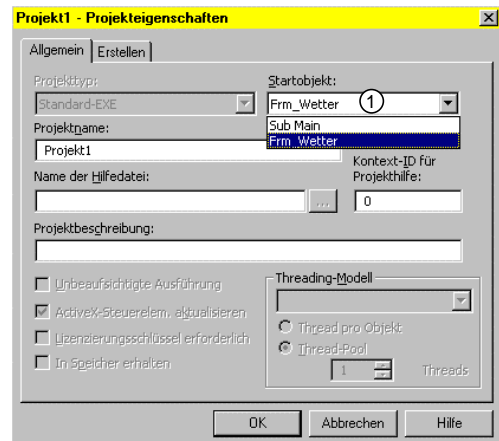


Abb. 3.6 Startform festlegen

Erfolgreiches Übersetzen

Nach erfolgreicher Übersetzung (keine syntaktischen Fehler) erscheint die Form bzw. die Benutzeroberfläche, die Sie zuvor erstellt haben, als ausführbares Programm am Bildschirm.

Fehlerhaftes Übersetzen

Falls Ihr Programm bzw. Ihre Ereignisprozedur noch formale Fehler enthält, werden diese vom Compiler festgestellt. Der Übersetzungslauf wird in diesem Fall abgebrochen, das Codefenster automatisch geöffnet und die Stelle des aufgetretenen Fehlers markiert.

Zusätzlich erscheint eine erläuternde Fehlermeldung. Führen Sie nach der Fehlerbehebung einen erneuten Übersetzungsvorgang durch. (Zu einer ausführlichen Beschreibung, wie Sie Fehler beheben können, vgl. Kapitel 11).

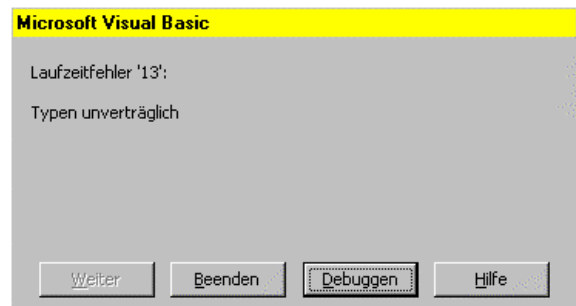


Abb. 3.7 Fehlermeldung

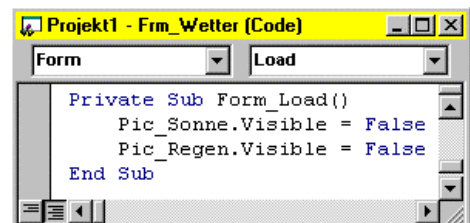
Testen der Ereignisprozeduren

Das System testet Ihr Programm auf formale Korrektheit. Die inhaltliche Korrektheit müssen Sie selbst prüfen. Um Ihre programmierten Ereignisprozeduren zu testen, führen Sie die Ereignisse an den jeweiligen Steuerelementen durch (beispielsweise Klicken auf die Schaltfläche), und beobachten genau die Reaktionen. Entsprechen die eingetretenen Reaktionen nicht den von Ihnen gewünschten, müssen Sie die Anweisungen der Ereignisprozeduren ändern und erneut testen.




Starten Sie das Programm *Wetter*, und klicken Sie auf die Schaltflächen. Arbeitet das Programm korrekt, wird bei jedem Klick die entsprechende Grafik angezeigt. Sie werden allerdings feststellen, daß beim Start des Programmes zunächst beide Grafiken sichtbar sind. Durch eine zusätzliche Prozedur können Sie dies verhindern.

- ⇒ Klicken Sie doppelt auf die Form.
Es wird eine Prozedur für das Standardereignis `Load` der Form eingefügt.
- ⇒ Setzen Sie für beide Bildfelder die Eigenschaft `Visible` auf `False`, so daß beim Starten keines der Bilder angezeigt wird.



3.7 Ausführbare Programmdatei erstellen

Ist Ihr Programm fehlerfrei und vollständig, können Sie eine ausführbare Programmdatei erstellen. Mit dieser Programmdatei kann Ihr Programm unter Windows auch ohne Visual Basic gestartet werden. Dabei werden alle Formen, Module und Zusatzsteuerelemente Ihres Projektes zu einer einzelnen ausführbaren Programmdatei zusammengefügt, die die Dateierweiterung .EXE erhält.

- ⇒ Speichern Sie Ihr fehlerfreies Projekt über den Menüpunkt DATEI - PROJEKT SPEICHERN.
- ⇒ Wählen Sie den Menüpunkt DATEI - Name.EXE ERSTELLEN.
- ⇒ Geben Sie im Eingabefeld einen Namen ① für die Datei ein.
- ⇒ Bestätigen Sie mit OK oder .

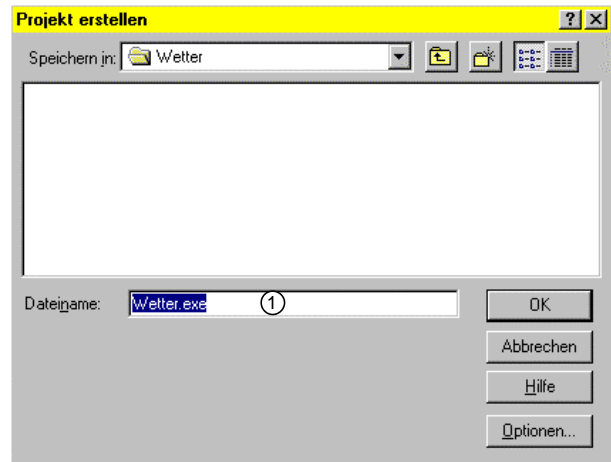


Abb. 3.8 Eine EXE-Datei erstellen

Programm aus dem Explorer von Windows starten

- ⇒ Öffnen Sie den Explorer von Windows, indem Sie beispielsweise im Startmenü auf den Eintrag PROGRAMME - WINDOWS-EXPLORER klicken.
- ⇒ Wechseln Sie zu dem Verzeichnis, in dem Ihre Programmdatei gespeichert ist.
- ⇒ Klicken Sie doppelt auf den Dateinamen des Programms (Typ ANWENDUNG).


Paket- und Weitergabe-Assistent

Zur Ausführung einer Anwendung benötigt Windows zahlreiche weitere Dateien (z.B. DLL-Dateien); die EXE-Datei allein genügt nicht. Auf Ihrem Computer sind diese Dateien durch die Installation der Visual Basic-Entwicklungsumgebung automatisch vorhanden.

Wollen Sie Ihre Anwendung auf einen fremden Rechner transferieren, müssen Sie auch alle weiteren von Visual Basic benötigten Dateien mitliefern. Bei der Zusammenstellung dieser Dateien ist Ihnen der Paket- und Weitergabe-Assistent behilflich.

- ⇒ Speichern und kompilieren Sie Ihr Projekt.
- ⇒ Rufen Sie den Menüpunkt ADD-INS - ADD-IN-MANAGER auf.

Wenn Sie den Paket- und Weitergabe-Assistenten das erste Mal starten, erhalten Sie nebenstehendes Dialogfenster.

- ⇒ Markieren Sie den Eintrag ①, und aktivieren Sie das Kontrollfeld ②.
- oder Klicken Sie doppelt auf den Eintrag ①.
Das Menü ADD-INS erhält ein zusätzliches Menüelement  Paket- und Weitergabe-Assistent..., über das Sie den Assistenten starten können.

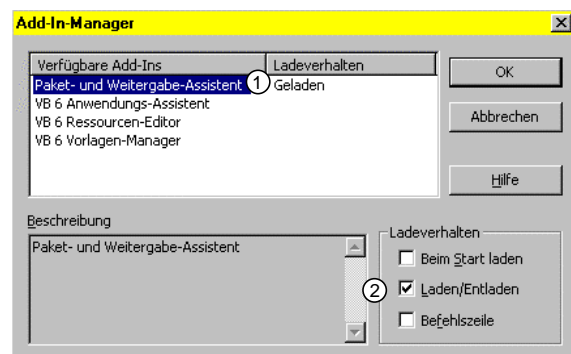


Abb. 3.9 Erstmaliges Starten

Der Paket- und Weitergabe-Assistent wird automatisch auf das gerade geladene Projekt angewandt. Ist kein Projekt geladen, erhalten Sie eine entsprechende Meldung.

Im folgenden Dialogfenster können Sie aus drei Funktionen auswählen. Die Funktion VERPACKEN ist die am häufigsten verwendete. Diese Funktion erstellt aus Ihrem Projekt ein Paket, das alle zur Ausführung Ihrer Anwendung benötigten Dateien enthält.

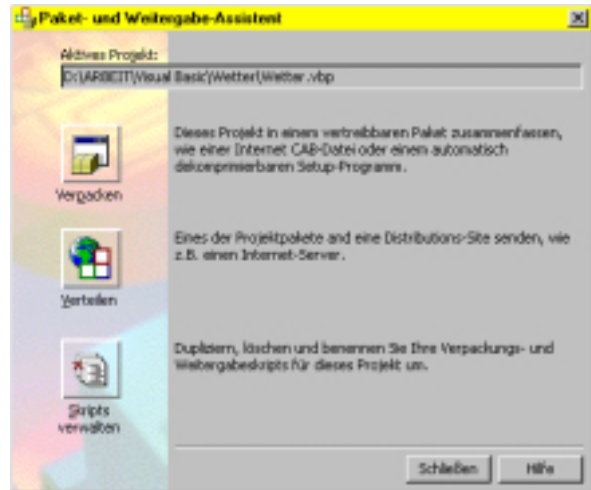


Abb. 3.10 Funktion auswählen

3.8 Schnellübersicht

Sie möchten...	Lösung	Alternative
ein neues Projekt erstellen	DATEI - NEUES PROJEKT	
ein Projekt öffnen	DATEI - PROJEKT ÖFFNEN	
ein Projekt speichern	DATEI - PROJEKT SPEICHERN	
ein Element eines Projektes unter einem anderen Namen speichern	DATEI - SPEICHERN VON <i>Name</i> UNTER	--
ein Steuerelement einer Form hinzufügen	Steuerelement in der Werkzeugsammlung anklicken und auf der Form aufziehen	--
Objekteigenschaften festlegen bzw. ändern	Objekt markieren, ANSICHT - EIGENSCHAFTENFENSTER	
eine Ereignisprozedur für ein Objekt schreiben	Objekt markieren, ANSICHT - CODE	
eine Aktion rückgängig machen	BEARBEITEN - RÜCKGÄNGIG	
ein Programm starten	AUSFÜHREN - STARTEN	
ein Programm beenden	AUSFÜHREN - BEENDEN	
eine ausführbare Programmdatei erstellen	DATEI - <i>Name</i> .EXE ERSTELLEN	--
Ihr Projekt für die Installation auf einem anderen Rechner bereitstellen	ADD-INS - PAKET- UND WEITERGABE-ASSISTENT	--

3.9 Übung

Übung 1 - Erstellen einer Ereignisprozedur

Beim Klicken auf eine Schaltfläche soll in einer Textbox ein festgelegter Text erscheinen.

- ① Fügen Sie einer leeren Form eine Schaltfläche und eine Textbox hinzu, und positionieren Sie beide Steuerelemente horizontal zentriert auf der Form.
- ② Vergeben Sie für die Objekte des Projektes folgende Eigenschaftswerte:

Eigenschaft	Neuer Wert	Wirkung nach Start
Visible	True bzw. False	Die Form ist sichtbar bzw. unsichtbar.
Enabled	True bzw. False	Die Form ist aktiv bzw. inaktiv, d.h. ein Klick auf die Formfläche erzeugt einen Warnpiepser.
BackColor	Auswahl aus der Farbpalette	Die gewählte Farbe wird als Formfarbe angezeigt.
Controlbox (sowie auch Min-, Maxbutton)	True bzw. False	Das Systemmenüfeld wird angezeigt bzw. ausgeblendet.
ShowInTaskbar	True bzw. False	Nach Minimieren der Form wird diese in der Task-Leiste angezeigt bzw. nicht angezeigt.

- ④ Zeichnen Sie in die Form ein Eingabefeld (TextBox) ein.
- ⑤ Starten Sie das Projekt (achten Sie darauf, daß Sie die Form-Eigenschaften vorher wieder zurückstellen!).
- ⑥ Geben Sie in das Eingabefeld Text ein, und bearbeiten Sie ihn.
- ⑦ Beenden Sie den Programmablauf.
- ⑧ Markieren Sie das Eingabefeld, um im Eigenschaftenfenster dessen Eigenschaften anzuzeigen.
- ⑨ Geben Sie bei (Name) *Txt_Meldung* ein, legen Sie folgende Eigenschaften fest, und starten Sie jeweils das Projekt:

Eigenschaft	Neuer Wert	Wirkung
Text	Das ist meine erste TextBox.	Anzeige des Textes
Enabled	True bzw. False	Im Eingabefeld erscheint der Cursor, und es kann Text eingegeben werden, bzw. es ist keine Eingabe möglich (Fehlerpiepser).
Visible	True bzw. False	Sichtbar bzw. unsichtbar
Font	Wählen Sie im Dialogfenster die Schriftart etc.	Anzeige der Schrift gemäß gewählter Einstellung

- ⑩ Zeichnen Sie eine Schaltfläche mit dem Namen *Cmd_Meldung* in die Form ein.
- ⑪ Testen Sie die Wirkung der oben veränderten Eigenschaften. Beachten Sie, daß Sie statt der Eigenschaft `.Text` die Eigenschaft `.Caption` verändern müssen.

Übung 3 - Das Codefenster und die Ereignisprozeduren

Nach Klick auf *Cmd_Meldung* soll ein Dialogfenster mit dem Text "Hello World" erscheinen.

- ① Markieren Sie die Schaltfläche *Cmd_Meldung*, und öffnen Sie das Codefenster durch Doppelklick auf die Schaltfläche.

Es wird die Standardereignisprozedur angezeigt. Im Falle der Schaltfläche ist das der einfache Mausklick.

- ② Geben Sie den folgenden Code ein:

```
MsgBox "Hello World"
```

- ③ Starten Sie das Projekt.

Nach Klick auf die Schaltfläche erscheint das rechts angezeigte Dialogfenster, das nach Klick auf OK wieder verschwindet.



Nach Klick auf *Cmd_Meldung* soll "Hello World" in dem Eingabefeld *Txt_Meldung* erscheinen.

- ④ Gehen Sie wieder in das Click-Ereignis von *Cmd_Meldung*, und kommentieren Sie die obere Zeile aus, d.h. setzen Sie ein Hochkomma ' vor die Zeile mit dem `MsgBox`-Befehl.

