

10 Diagramme und Zeichnungsobjekte (Shapes)

Diagramme stehen im Zentrum vieler Excel-Anwendungen. Dieses Kapitel gibt einen knappen Überblick darüber, welche Diagramme Excel unterstützt, und zeigt anschließend, wie Sie Diagramme programmgesteuert erstellen und ausdrucken können. Ein längeres Beispiel zum Thema Datenprotokollierung demonstriert diverse Programmiertechniken.

Ein weiteres Thema dieses Kapitels sind die seit Excel 97 verfügbaren Zeichnungsobjekte (*Shapes*), mit denen sowohl Diagramme als auch ganz normale Tabellenblätter dekoriert werden können.

Kapitelübersicht

10.1	Umgang mit Diagrammen	502
10.2	Programmierung von Diagrammen	509
10.3	Beispiel – Automatische Datenprotokollierung	517
10.4	Syntaxzusammenfassung Diagramme	531
10.5	Zeichnungsobjekte (Shapes)	532
10.6	Organigramme und andere Diagramme	536

10.1 Umgang mit Diagrammen

Seien Sie unbesorgt – Sie werden an dieser Stelle nicht eine weitere Einführung in den Umgang mit Diagrammen erhalten. Das Thema ist in zahlreichen Excel-Büchern bereits erschöpfend behandelt worden (im wahrsten Sinne des Wortes). Das Ziel dieses Abschnitts besteht vielmehr darin, systematisch und ohne viel Rücksicht auf Bedienungsdetails die Gestaltungsmöglichkeiten von Diagrammen zu beschreiben, die Elemente von Diagrammen zu benennen und deren Funktion zu erklären. Diese Informationen stellen ein elementares Vorwissen für die Programmierung von Diagrammen dar, bei denen es von diversen *ChartXxx*-Objekten nur so wimmelt.

10.1.1 Grundlagen

Diagrammblätter versus eingelagerte Diagramme in Tabellenblättern

In Excel können Sie Diagramme entweder in Tabellen einbetten oder in eigenen Diagrammblättern darstellen. Die erste Variante hat den Vorteil, dass das Diagramm zusammen mit den dazugehörigen Daten ausgedruckt werden kann. Dabei sind auch sehr kleine Diagramme möglich, die nur einen Bruchteil der Seite füllen.

Der Diagrammassistent

Der Weg zu einem neuen Diagramm führt in der Regel über den Diagrammassistenten. Dieser Assistent wird automatisch aufgerufen, sobald Sie ein neues Diagramm erzeugen (mit EINFÜGEN | DIAGRAMM) oder das Symbol DIAGRAMMASSISTENT anklicken. Im ersten Schritt des Diagrammassistenten geben Sie den gewünschten Diagrammtyp, im zweiten Schritt den Datenbereich an. Dabei darf es sich ohne weiteres um einen aus mehreren Teilbereichen zusammengesetzten Zellbereich handeln. In den weiteren Schritten können Sie Optionen zur Gestaltung des Diagramms bestimmen. Der Diagrammassistent kann auch für bereits vorhandene Diagramme aufgerufen werden, um Details der Formatierung zu verändern.

Weiterbearbeitung von Diagrammen

Diagramme, die mit dem Diagrammassistenten erstellt werden, entsprechen oft noch nicht Ihren Vorstellungen. Die eigentliche Arbeit des Feinlayouts findet daher erst nach dem Abschluss des Diagrammassistenten statt: dazu können Sie innerhalb des Diagrammbereichs beinahe alle Diagrammelemente mit der Maus anklicken: die Legende, die Achsen, einzelne Datenreihen (die in Form von Linienzügen, Balken etc. dargestellt werden), den Hintergrund des Diagramms etc. Zu jedem dieser Diagrammelemente existiert ein Kontextmenü, das zumeist reichhaltige Einstellmöglichkeiten eröffnet. In den wichtigsten Einstelldialogen zum jeweiligen Element können Sie noch bequemer durch einen Doppelklick gelangen.

Wenn Sie zum ersten Mal mit Diagrammen arbeiten, werden Sie oft das Problem haben, dass Sie nicht wissen, welches Element Sie anklicken müssen, um eine ganz bestimmte Veränderung zu bewirken. Hier gibt es nur zwei Alternativen: Entweder Sie quälen sich durch das Handbuch oder Sie probieren es einfach aus.

10.1.2 Diagrammtypen

Excel kennt über 70 Diagrammtypen (wobei sich aber viele Typen sehr ähneln). Eine vollständige Referenz finden Sie im Diagrammassistenten (wo die Diagramme nach Gruppen geordnet sind) oder in der VBA-Hilfe zum Schlüsselwort *ChartType*.

Verbunddiagramme

Verbunddiagramme sind Diagramme, in denen mehrere Diagrammtypen vereint sind (beispielsweise ein Linien- und ein Säulendiagramm). Verbunddiagramme können Sie entweder mit Hilfe von benutzerdefinierten Diagrammen erstellen (siehe unten) oder indem Sie den Diagrammtyp einer einzelnen Datenreihe (nicht des ganzen Diagramms) verändern.

Es lassen sich nur solche Diagramme kombinieren, die auf demselben Koordinatensystem aufbauen. Insofern sind die Kombinationsmöglichkeiten relativ gering. 3D-Diagramme lassen sich überhaupt nicht kombinieren.

Pivotdiagramme

Pivotdiagramme sind neu in Excel 2000. Dabei handelt es sich eigentlich nicht um einen neuen Diagrammtyp, sondern um eine neue Form der Datenverbindung zwischen einem Diagramm und einer Pivottable. Die Besonderheit von Pivotdiagrammen besteht darin, dass Kategorien zur Gliederung der Daten dynamisch eingestellt werden können (d.h. durch Listfelder im Diagramm). Das Diagramm wird sofort entsprechend umgebildet. Für das Diagramm an sich stehen fast alle oben erwähnten Diagrammtypen zur Verfügung. Pivotdiagramme werden im Rahmen von Pivottablen in Kapitel 13 beschrieben.

Benutzerdefinierte Diagrammtypen

Die Formatierung von Diagrammen kann auf zwei Weisen erfolgen: Sie können eines der Standardtypen auswählen, oder Sie können so genannte benutzerdefinierte Typen (ehemals Autoformate) verwenden. In diesen Typen sind zahlreiche Formatierungsdetails gespeichert, so dass Sie sehr rasch zu ganz unterschiedlichen Diagrammen gelangen. Die Bezeichnung 'benutzerdefiniert' ist insofern irreführend, als Excel eine ganze Palette vordefinierter (integrierter) Typen kennt.

Die in Excel integrierten benutzerdefinierten Formate geben einen recht guten Überblick, welche Möglichkeiten zur Verfügung stehen. Die Formate sind in OfficeVerzeich-

nis\Office\l\Xl8galry.xls gespeichert. n ist dabei ein Sprachcode (z.B. 1031 für die deutsche Version).

Noch wichtiger ist die Möglichkeit, benutzerdefinierte Formate selbst hinzuzufügen und später wiederzuverwenden. Dazu formatieren Sie ein Diagramm nach Ihren Vorstellungen, öffnen mit der rechten Maustaste den Diagrammtyp-Dialog, wechseln in das Blatt BENUTZERDEFINIERT und wählen die gleichnamige Option. Mit dem Button HINZUFÜGEN können Sie nun Ihre Einstellungen als neuen Diagrammtyp speichern. Die neuen (persönlichen) Diagrammtypen werden in der Datei Benutzerverzeichnis\Anwendungsdaten\Microsoft\Excel\Xlusrgal.xls gespeichert.

10.1.3 Diagrammelemente (Diagrammobjekte) und Formatierungsmöglichkeiten

Sowohl für das Feinlayout von Diagrammen als auch für die Programmierung von Diagrammen ist es erforderlich, dass Sie wissen, zwischen welchen Diagrammobjekten Excel unterscheidet. Eine Hilfestellung beim Experimentieren stellt übrigens die DIAGRAMM-Symbolleiste dar: Dort wird im linken Listenfeld die Bezeichnung des gerade angeklickten Objekts angezeigt, etwa »Achse n «, »Gitternetzlinie n «, » Rn « (gemeint ist eine Datenreihe) etc.

- *Diagramm*: Gemeint ist eigentlich das Objekt *ChartArea*, das für den Hintergrund des gesamten Diagramms zuständig ist (also jene Fläche, die unterhalb der Zeichnungsfläche, der Legende etc. zu sehen ist). Die hier eingestellte Schriftart gilt für alle Texte des Diagramms, bei denen nicht explizit eine andere Schriftart eingestellt wird.
- *Zeichnungsfläche*: Die Zeichnungsfläche (*PlotArea*) stellt ein Rechteck um den grafischen Bereich des Diagramms dar. Die Zeichnungsfläche beinhaltet das eigentliche Diagramm, nicht aber den Titel, die Legende etc. Bei den meisten 2D-Diagrammen zählen nicht einmal die Achsen zur Zeichnungsfläche: Wenn Sie für die Zeichnungsfläche die Hintergrundfarbe grün und für die Diagrammfläche die Hintergrundfarbe rot angeben, wird die Achsenbeschriftung rot unterlegt.
- *Bodenfläche, Wände*: Diese beiden Objekte existieren nur bei 3D-Diagrammen und beschreiben das Aussehen der Bodenfläche und der beiden vertikalen Begrenzungsflächen des Diagramms. Die Zeichnungsfläche betrifft in diesem Fall nur den rechteckigen Raum außerhalb des eigentlichen Diagramms.
- *Ecken*: Auch die Ecken existieren als eigenes Objekt nur bei 3D-Diagrammen. Ecken können zwar nicht formatiert werden, Sie können das Diagramm aber an den Ecken mit der Maus »anfassen« und dreidimensional verdrehen. Das ist oft bequemer als die Einstellung von Blickrichtung und Perspektive über den Dialog 3D-ANSICHT.
- *Datenreihen*: Eine Datenreihe beschreibt eine zusammengehörige Dateneinheit (zumeist die Werte einer Spalte aus der zugrunde liegenden Tabelle; nur wenn Sie im

Diagrammassistenten DATENREIHEN IN ZEILEN auswählen, sind Datenreihen zeilenweise organisiert). Eine Datenreihe wird beispielsweise durch einen Linienzug dargestellt. Die Formatierungsdaten von Datenreihen betreffen die grafische Repräsentierung dieser Datenreihe – also Farbe, Markierungspunkte, Linienstil etc.

- *Datenpunkte*: Die einzelnen Werte einer Datenreihe werden durch Datenpunkte repräsentiert. Normalerweise sind die Formatierungseigenschaften aller Datenpunkte gleich und durch die Eigenschaften der Datenreihe vorgegeben. Sie können aber die Eigenschaften jedes Datenpunkts separat einstellen und so einen einzelnen Datenpunkt einer Reihe hervorheben, getrennt beschriften etc. Bei Kreisdiagrammen können Sie einzelne Segmente aus dem Diagramm herausziehen und dadurch hervorheben – auch das betrifft eine Eigenschaft des Datenpunkts. Vorsicht: Die vertikale Position von Datenpunkten in 2D-Diagrammen kann mit der Maus verändert werden – dadurch verändert sich aber der zugrunde liegende Wert in der Datentabelle!
- *Trendlinien*: Datenreihen von 2D-Diagrammen können Trendlinien zugeordnet werden. Die Trendlinie wird zusätzlich zur normalen Repräsentierung der Daten gezeichnet. Excel kennt dabei zwei Arten von Trendlinien: Näherungskurven (fünf verschiedene Typen) und Ausgleichskurven.
- *Fehlerindikatoren*: Auch Fehlerindikatoren stellen ein Unterelement zu Datenreihen in 2D-Diagrammen dar. Sie markieren den möglichen Abweichungsbereich der Datenpunkte.
- *Koordinatenachsen*: Auch bei den Koordinatenachsen gibt es unzählige Formatierungsdetails, die mit der Skalierung (Minimum, Maximum, linear oder logarithmisch) beginnen und mit der genauen Anordnung der Achsenbeschriftung enden (z.B. jeder zehnte Datenpunkt wird beschriftet, jeder zweite durch einen Teilstrich gekennzeichnet; Teilstriche nach innen oder nach außen gerichtet etc.). Neu in Excel 97 ist die Möglichkeit, Koordinatenachsen auch mit schräger Schrift zu beschriften (Tabellenblatt AUSRICHTUNG).

Es besteht auch die Möglichkeit, ein 2D-Diagramm mit zwei unabhängigen Y-Achsen auszustatten, wobei für einige Datenreihen die eine, für die restlichen Datenreihen die andere Achse gilt. Das ist dann nützlich, wenn Sie zwei inhaltlich verwandte, quantitativ aber unterschiedliche Größen in einem Diagramm darstellen möchten (z. B. eine Spannung und Strom). Voraussetzung für die Darstellung zweier Y-Achsen ist die Trennung der Datenreihen in zwei Gruppen. Am einfachsten erfolgt das durch die Auswahl des Diagrammtyps LINIEN AUF ZWEI ACHSEN aus den benutzerdefinierten Diagrammtypen.

- *Gitternetzlinien*: Die Zeichnungsfläche bei 2D-Diagrammen bzw. die Bodenfläche und die Wände von 3D-Diagrammen können durch Gitternetzlinien überlagert werden. Gitternetzlinien richten sich in ihrer Position an die Teilstriche der Koordinatenachsen. Das Aussehen (Farbe, Linienform) von Haupt- und Hilfsgitterlinien ist individuell einstellbar (allerdings nur bei gewöhnlichen Diagrammen, nicht bei Verbunddiagrammen).

- *Titel:* Das Diagramm kann mit mehreren Titeln (für das Diagramm, die Achsen etc.) ausgestattet werden. Position, Schriftart, Ausrichtung etc. sind wiederum frei einstellbar.
- *Legende:* Die Legende ermöglicht eine Zuordnung von den im Diagramm genutzten Farben und Mustern zu den dazugehörigen Datenreihen. Die Beschriftung der Legende wird dem ersten Element der Datenreihe entnommen. Die Legende kann an einer beliebigen Stelle im Diagramm angeordnet werden (sogar über den Daten).

Diagrammoptionen in Extras | Optionen

Über EXTRAS | OPTIONEN | DIAGRAMM sind einige weitere Diagrammoptionen zugänglich. Diese Einstellungen betreffen nur das aktuelle Diagramm (und können auch nur verändert werden, wenn gerade ein Diagramm aktuell ist).

Die Option LEERE ZELLEN bestimmt das Verhalten Excels, wenn es in der Datenreihe auf leere Zellen stößt. In der Voreinstellung WERDEN NICHT ANGEZEIGT tritt im Diagramm an dieser Stelle ein Loch auf (d.h., einige Balken fehlen, eine Linie ist unterbrochen etc.). Die beiden Alternativen lauten ALS NULLWERT ZEICHNEN (dann geht Excel mit leeren Zellen so um, als stünde der Wert 0 darin) oder INTERPOLIEREN (dann versucht Excel, für den fehlenden Bereich selbst passende Daten einzusetzen).

Das Auswahlkästchen NUR SICHTBARE ZELLEN ANZEIGEN bestimmt, wie Excel mit ausgeblendeten Zeilen/Spalten umgehen soll: Wenn das Kästchen aktiviert ist, dann werden Daten aus unsichtbaren Zeilen/Spalten nicht gezeichnet. Im Diagramm tritt an dieser Stelle ein Sprung (kein Loch) auf. Die Einstellung ist vor allem dann von Interesse, wenn die Diagrammdaten aus einer gefilterten Datenbank stammen.

Das Auswahlkästchen DIAGRAMM AN FENSTERGRÖÖE ANPASSEN ist nur für Diagrammblätter von Interesse. Wenn das Kästchen aktiviert ist, wird das Diagramm an die aktuelle Fenstergröße angepasst. Andernfalls wird im Fenster eine Seite des eingestellten Druckers angezeigt. Damit das ganze Diagramm sichtbar ist, muss eventuell der Zoomfaktor verändert werden (ANSICHT | ZOOM).

Trendlinien, Datenglättung

Bei Liniendiagrammen können Sie in den Formatierungseinstellungen zu Datenreihen die Option LINIE GLÄTTEN auswählen. Dadurch wird der sonst oft eckige Verlauf der Kurve ein wenig abgerundet.

Weitergehende Möglichkeiten, durch vorhandenes Datenmaterial eine Näherungs- oder Ausgleichskurve zu legen, bietet das Kommando TRENDLINIE EINFÜGEN. Excel kann eine Datenmenge durch fünf verschiedene Typen von Näherungskurven nähern: durch eine Gerade, eine Polynomkurve (maximal sechster Ordnung), eine logarithmische, exponentielle oder potentielle Kurve. Über die Optionen im Dialog TRENDLINIE FORMATIEREN können Sie angeben, ob und wie weit die Kurve über die vorhandenen Daten hinaus gezeichnet und ob die Formel der Kurve angegeben werden soll.

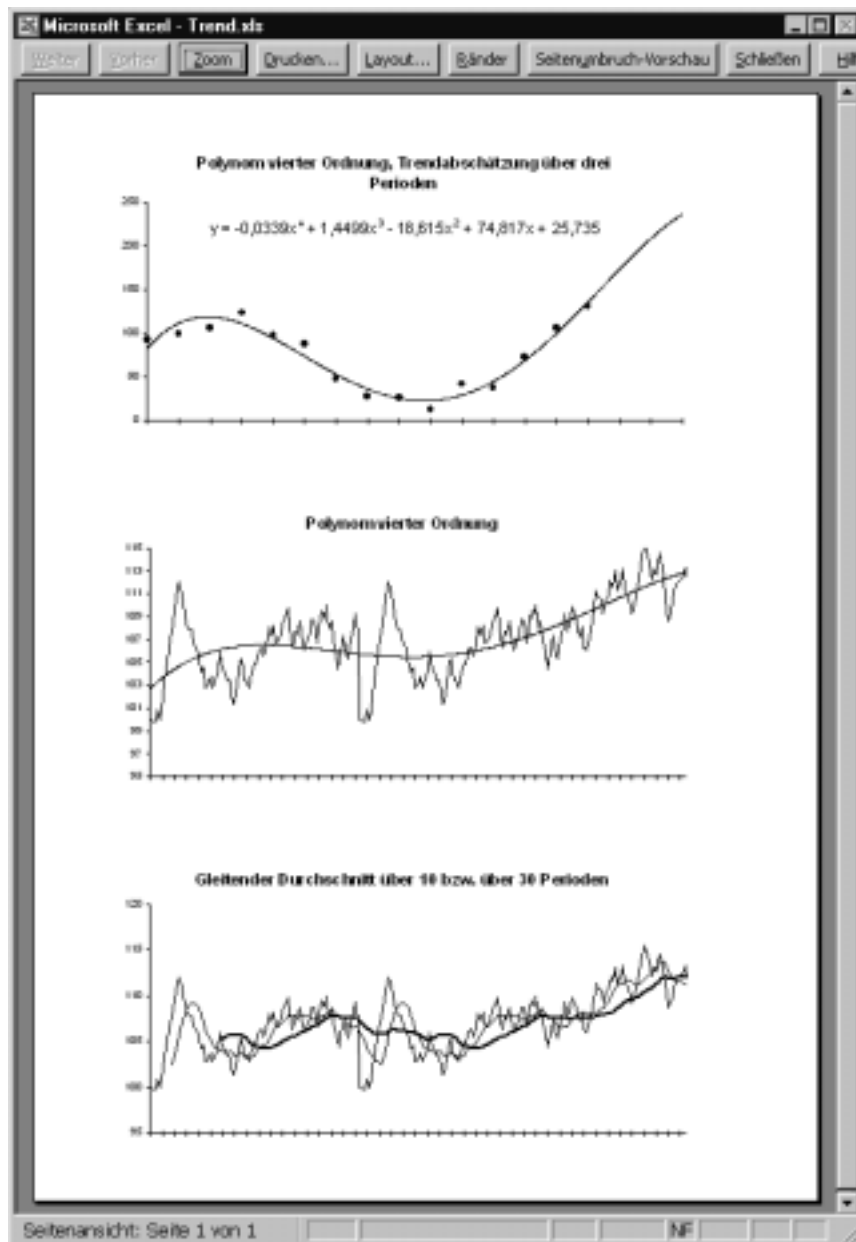


Bild 10.1: Drei Beispiele für Trendlinien

Über den TRENDLINIEN-Dialog kann ein sechster Kurventyp angegeben werden: Ausgleichskurve nach dem Verfahren des gleitenden Durchschnitts. Dabei wird jeder Punkt der Kurve aus dem Durchschnitt der n vorangehenden Datenpunkte berechnet. Eventuelle statistische Messfehler werden dadurch geglättet. Ausgleichskurven kön-

nen aber im Gegensatz zu Näherungskurven nicht über den Datenbereich hinaus gezeichnet werden.

Einige Beispiele für die Anwendung der Trendlinienfunktion zeigt Bild 10.1. Die dazugehörige Beispieldatei 10\Trend.xls finden Sie auf der beiliegenden CD-ROM.

Fehlerkennzeichnung

Datenreihen in 2D-Diagrammen können mit Fehlerindikatoren versehen werden. Dabei handelt es sich um kleine Linien, die jenen Bereich angeben, in dem sich der tatsächliche Wert bei Berücksichtigung eines statistischen Messfehlers befinden kann.

10.1.4 Ausdruck

Beim Ausdruck muss wieder zwischen zwei Varianten unterschieden werden: Wenn das Diagramm in ein Tabellenblatt eingebettet ist, dann erfolgt der Ausdruck im Rahmen des Tabellenausdrucks. Das einzige Problem besteht darin, dass Excel sich wenig Gedanken über den Seitenumbruch macht und ein Diagramm im ungünstigsten Fall in bis zu vier Teile zerlegt. Es schadet also nicht, sich den Ausdruck zuerst in der Seitenansicht anzusehen. Eventuell müssen Sie den Ausdruck durch einige starre Seitenumbrüche (EINFÜGEN | SEITENWECHSEL) manuell optimieren.

Wenn sich das Diagramm dagegen in einem Diagrammblatt befindet, oder wenn Sie ein eingebettetes, aber vorher durch Doppelklick aktiviertes Diagramm ausdrucken möchten, dann existieren im Dialog SEITENEINRICHTUNG einige Optionen speziell für den Diagrammausdruck. Am wichtigsten ist dabei die GEDRUCKTE DIAGRAMMGRÖSSE. In der Standardeinstellung GANZE SEITE VERWENDEN nutzt Excel die gesamte Seitengröße. Wenn das Diagramm nicht zufällig dasselbe Format wie die Seite hat, wird es dabei total verzerrt. Daher ist es zumeist sinnvoller, die Option AN SEITE ANPASSEN auszuwählen. Excel vergrößert das Diagramm jetzt nur so weit, wie dies ohne eine Veränderung des Verhältnisses Länge zu Breite möglich ist. Die dritte Variante BENUTZERDEFINIERT lässt die Größe des Diagramms unverändert.

Mit der Option SCHWARZWEIß-DRUCK können Sie Farbdigramme auch auf Schwarzweiß-Druckern wiedergeben. (Bei den meisten Druckern ist das allerdings auch ohne diese Option möglich.) Egal ob mit oder ohne diese Option, brauchbare Ergebnisse werden Sie mit einem Schwarzweiß-Drucker nur dann erzielen, wenn Sie bereits bei der Formatierung des Diagramms auf Farben verzichten. Nutzen Sie unterschiedliche Linienstärken und -formen zur Unterscheidung mehrerer Datenreihen.

Da die Defaultformate von Excel generell sehr farbenfreudig sind, ist eine entsprechende Schwarzweiß-Formatierung mit einem enormen Aufwand verbunden (grobe Schätzung: 100 Mausklicks für ein typisches Diagramm). Sie sollten daher, wenn diese öfters vorkommen, Schwarzweiß-Diagramme als benutzerdefinierte Diagrammtypen speichern.

10.2 Programmierung von Diagrammen

Die ersten Versuche, Diagramme zu programmieren, stellen sich oft als sehr mühsam heraus. Der Grund: Die Orientierung in der Unmenge von *Chart*-Objekten ist nicht eben einfach, die Objektzuordnung von Eigenschaften und Methode nicht immer einsichtig.

Ein Beispiel gefällig: Die Methode *ClearContents* des *ChartArea*-Objekts löscht die Daten eines Diagramms, nicht aber seine Formatierung. Das ist insofern merkwürdig, als sich das *ChartArea*-Objekt eigentlich nicht auf das Diagramm an sich, sondern nur auf seinen Hintergrund bezieht. Logischer wäre es also gewesen, die Diagrammdata über die *Delete*-Methode des *Chart*-Objekts zu löschen – aber siehe da, diese Methode liefert bei eingebetteten Diagrammen nur eine Fehlermeldung. Offensichtlich ist *Delete* nur zum Löschen von Diagrammblättern geeignet, während die beiden verwandten Methoden *ClearContents* und *ClearFormats* des *ChartArea*-Objekts für die Interna des Diagramms zuständig sind.

Im Gegensatz zum *ChartArea*-Objekt steht übrigens das *PlotArea*-Objekt (»Zeichnungsfläche«): Auch dieses Objekt beschreibt den Hintergrund des Diagramms, diesmal allerdings den Bereich unmittelbar unter den Diagrammlinien, -balken etc.

ANMERKUNG

Auch wenn Sie am Anfang von der Vielfalt der Objekte und ihrer Eigenschaft überwältigt sind, gibt es auch positive Seiten: Sie können wirklich fast alles per Programmcode steuern. Aus Platzgründen ist es hier leider nicht möglich, diese Vielfalt zur Gänze zu beschreiben. In vielen Details werden Sie also auch nach der Lektüre dieses Kapitels auf die Hilfe angewiesen sein.

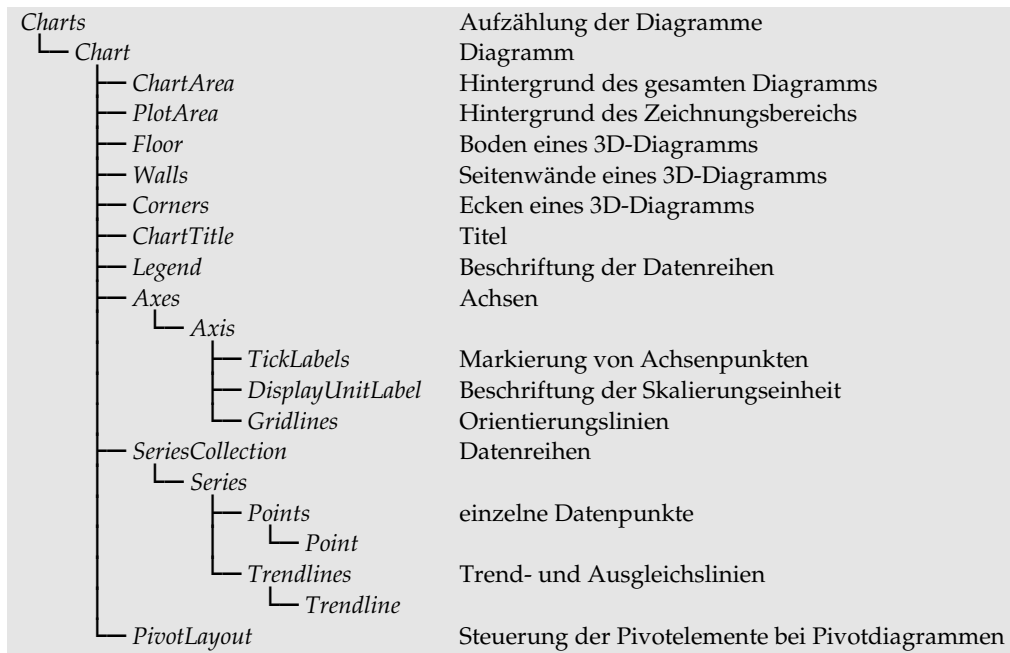
Anstatt lange zu suchen: Nutzen Sie die Makroaufzeichnung!

Wenn Sie wissen möchten, auf welche Weise Sie im Programm eine bestimmte Formatierung erreichen können, verwenden Sie am besten die Makroaufzeichnung als Ratgeber (die Beispiele aus der Hilfe sind praktisch alle aussagegelos). Je kürzer die Aufzeichnung, desto leichter tun Sie sich bei der Interpretation der Ergebnisse. Sie sollten also die Makroaufzeichnung starten, an einem bereits vorhandenen Diagramm nur ein einziges Detail verändern und die Aufzeichnung anschließend sofort wieder stoppen. Wenn Sie am Bildschirm ein Fenster mit dem Programmcode und ein weiteres mit dem Diagramm nebeneinander anordnen, können Sie während der Makroaufzeichnung sogar beobachten, wann die einzelnen Codezeilen erzeugt werden.

Der aus der Makroaufzeichnung resultierende Code funktioniert in der Regel (auf jeden Fall sind während der Arbeit an diesem Kapitel keine gegenteiligen Vorkommnisse aufgetreten), er ist aber selten optimal. Zum Teil sind die darin enthaltenen Anweisungen unnötig umständlich, zum Teil ganz überflüssig. Eine Nachbearbeitung des Codes ist also unumgänglich.

10.2.1 Objekthierarchie

Die folgende Zusammenstellung gibt einen Überblick über die Objekthierarchie bei Diagrammen. Um die Struktur besser verständlich zu machen, wurden nur die wichtigsten Objekte aufgenommen und nur der Fall berücksichtigt, dass das Diagramm in ein Tabellenblatt eingelagert ist (kein Diagrammblatt). Eine vollständige Auflistung aller Diagrammobjekte finden Sie in Abschnitt 16.1.



Kleines Glossar der Diagrammobjekte

Enorme Verwirrung stiften die zahlreichen, oft fast gleich lautenden *Chart*- und *Plot*-Objekte. Bild 10.2 gibt einen ersten Überblick.

Chart: Das eigentliche Diagramm; es besteht aus einigen Datenreihen, die grafisch repräsentiert werden, dem Hintergrund, den Koordinatenachsen, der Legende, dem Titel etc. Auf *Chart*-Objekte kann entweder über die Aufzählung *Charts* zugegriffen werden, wenn sich das Diagramm in einem eigenen Diagrammblatt befindet, oder über *ChartObjects(...).Chart*, wenn das Diagramm in ein Tabellenblatt eingebettet ist.

Der Diagrammtyp wird seit Excel 97 mit der Eigenschaft *ChartType* (ehemals *Type* und *SubType*) eingestellt. Als mögliche Einstellungen sind über 70 Konstanten vordefiniert (siehe Hilfe oder Objektkatalog).

ChartObject: Der äußere Rahmen («Container») eines Diagramms. Das *ChartObject*-Objekt ist nur bei Diagrammen erforderlich, die in Tabellenblätter eingebettet sind. Es steht zwischen dem Tabellenblatt und dem *Chart*-Objekt und bestimmt die Position und die Außenmaße des Diagramms innerhalb des Tabellenblatts. Über die *Worksheet-*

nes Diagrammblatt-Objekt, das vergleichbar mit einem Tabellenblatt wäre. Für das Hauptdiagramm des Blatts entfällt das zwischengelagerte *ChartObject*-Objekt.

Einige weitere Objekte beginnen zwar nicht mit »Chart«, sind aber dennoch interessant:

PlotArea: Der »grafische« Bereich innerhalb des Diagramms. Die Zeichnungsfläche enthält die Koordinatenachsen und die eigentliche Grafik. Die Hauptaufgabe des Objekts besteht darin, die Größe und Position dieses Bereichs innerhalb der Gesamtfläche des Diagramms zu bestimmen. Andere Bereiche im Diagramm sind die Legende (*Legend*-Objekt) und der Titel (*ChartTitle*-Objekt). Bei 3D-Diagrammen werden unabhängig von der *PlotArea* auch die Objekte *Floor* und *Walls* (als Subobjekte von *Chart*) verwaltet. Die beiden Objekte sind für die optische Gestaltung der Begrenzungsflächen des 3D-Diagramms zuständig.

ANMERKUNG

Wenn Sie `PlotArea.Width=n; m=PlotArea.Width` ausführen, dann ist m anschließend deutlich größer als n . Der Grund: `PlotArea.Width` verändert in Wirklichkeit die in Excel 97 eingeführte schreibgeschützte Eigenschaft `InsideWidth`, also den Innenbereich von *PlotArea*. Zu diesem Innenbereich kommt ein Außenbereich hinzu, in dem die Beschriftung der Koordinatenachsen durchgeführt wird. (Die gleichen Probleme treten selbstverständlich auch für `Height/InsideHeight` auf). Für die Einstellung der Größe des Außenbereichs können Sie sich meistens mit folgendem Code behelfen:

```
delta = PlotArea.Width - PlotArea.InsideWidth
PlotArea.Width = n + delta
```

Ganz exakt ist diese Methode auch nicht, weil die Größe des Beschriftungsbereichs nicht konstant ist. Wenn beispielsweise ein Diagramm sehr stark verkleinert wird, verzichtet Excel ganz auf die Achsenbeschriftung, und der Beschriftungsbereich wird 0.

Series, Point: Das *Series*-Objekt verweist auf die Daten einer Datenreihe des Diagramms. Die eigentlichen Zahlenwerte können der `Values`-Eigenschaft des *Series*-Objekts entnommen bzw. über diese Eigenschaft verändert werden. *Series* ist ein Subobjekt zum *Chart*-Objekt. Formatdaten, die nicht die ganze Reihe, sondern nur einen einzelnen Datenpunkt betreffen, werden in *Point*-Objekten verwaltet. Diese stellen wiederum ein Subobjekt zu den *Series*-Objekten dar.

Axis, Gridlines: Das *Axis*-Objekt ist ebenfalls ein Subobjekt zum *Chart*-Objekt. Es beschreibt die Details einer Koordinatenachse. Das *Gridlines*-Objekt stellt ein Subobjekt zum *Axis*-Objekt dar und wird über die Eigenschaften `MajorGridlines` bzw. `MinorGridlines` angesprochen.

Neu in Excel 2000 ist die Möglichkeit, für Koordinatenachsen eine Skalierungseinheit anzugeben (z.B. Millionen). Dazu muss `Axis.DisplayUnit` mit einer der vordefinierten Konstanten eingestellt werden (z.B. `xlMillions`). Das `DisplayUnitLabel`-Objekt gibt an, wie und wo diese Skalierungseinheit (also die Zeichenkette "Millionen") im Diagramm

angezeigt wird. Die Eigenschaft *HasDisplayUnitLabel* gibt an, ob die Achse skaliert ist oder nicht.

Neben den vordefinierten Skalierungseinheiten (10, 100, 1000 bis 1000.000.000) können auch beliebige andere Faktoren verwendet werden. Dazu wird *DisplayUnitCustom* der gewünschte Zahlenwert zugewiesen (der auch kleiner 1 sein darf, etwa 0,001, um Tausendstel anzuzeigen).

```
ActiveChart.Axes(xlValue).DisplayUnitCustom = 0.001
ActiveChart.Axes(xlValue).DisplayUnitLabel.Text = "Tausendstel"
```

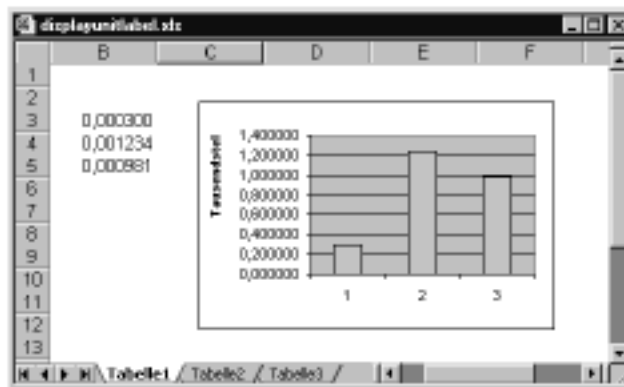


Bild 10.3: Die Y-Achse verwendet Tausendstel als Einheit

Trendline, ErrorBars: *Trendline* und *ErrorBars* sind Subobjekte zum *Series*-Objekt. Sie beschreiben die Details einer Trendlinie zur Datenreihe bzw. das Aussehen der Fehlerindikatoren.

HINWEIS

Die Schlüsselwörter *Gridlines* und *ErrorBars* stehen im Gegensatz zu den meisten anderen Objekten im Plural, obwohl sie nicht auf Aufzählobjekte verweisen.

TIPP

Wenn Sie nicht wissen, welchen Namen ein bestimmtes Objekt im Diagramm hat, können Sie das Objekt (beispielsweise eine Koordinatenachse) anklicken und im Direktfenster *?TypeName(Selection)* ausführen. Als Ergebnis erhalten Sie den Objektname (etwa *Axis*).

10.2.2 Programmieretechniken

Der Diagrammassistent

Die Methode *ChartWizard* stellt in der Regel den schnellsten Weg dar, um zu einem Diagramm zu kommen. Damit die Methode verwendet werden kann, muss zuerst ein

ChartObject-Objekt erzeugt werden. Die Bedeutung der zahlreichen Parameter der Methode können Sie in der Hilfe nachlesen.

```
ActiveSheet.ChartObjects.Add(30, 150, 400, 185).Name = _
    "neues Diagramm"
ActiveSheet.ChartObjects("neues Diagramm").Activate
ActiveChart.ChartWizard tabblatt.[A3:D99], xlLine, 4, xlColumns, 1, 1
```

Diagrammobjekte aktivieren oder auswählen?

Mit den Methoden *Activate* und *Select* hat Microsoft einige Verwirrung gestiftet: Mal muss die eine Methode verwendet werden (Fenster), mal die andere (Arbeitsblätter), mal sind beide erlaubt (Zellbereiche). Bei *ChartObject*-Objekten sind nicht nur beide Methoden erlaubt, sie führen jetzt sogar zu unterschiedlichen Ergebnissen!

Activate entspricht dem einfachen Anklicken des Diagramms mit der Maus. *Selection* verweist nun auf das Objekt *PlotArea* (nicht auf *Chart!*).

Select hat scheinbar diesselbe Wirkung, allerdings zeigt *Selection* nun auf *ChartObject*. *Select* eignet sich also beispielsweise, wenn Sie die Position des Diagramms innerhalb eines Tabellenblatts verändern möchten.

Die Gemeinsamkeit beider Methoden besteht darin, dass *ActiveChart* nach Ihrer Ausführung in jedem Fall auf das *Chart*-Objekt des Diagramms zeigt.

VORSICHT

In Excel 97 bereitet der Zugriff auf Diagramme durch *ActiveChart* zum Teil erhebliche Probleme. Abhilfe: Greifen Sie direkt auf das jeweilige Objekt zu, anstatt es zuerst zu aktivieren und dann via *ActiveChart* zu verändern.

Diagrammobjekte deaktivieren

Die sicherste Methode, um ein aktiviertes Diagramm zu deaktivieren, besteht darin, ein anderes Objekt zu aktivieren, also beispielsweise:

```
Sheets(n).[A1].Select
```

Diagramme löschen, kopieren und einfügen

ChartObject-Objekte können zusammen mit den darin enthaltenen Diagrammen direkt mit *Copy* kopiert und anschließend wieder in das Tabellenblatt eingefügt werden. Nach dem Einfügen verweist die *Selection*-Eigenschaft auf das neue *ChartObject*-Objekt, sodass dieses anschließend benannt werden kann. Wenn Sie ein *ChartObject*-Objekt einfach vervielfältigen möchten, können Sie statt *Copy* und *Paste* direkt die Methode *Duplicate* verwenden. Mit *Delete* können Sie ein *ChartObject*-Objekt samt der darin enthaltenen Daten löschen.

```
ActiveSheet.ChartObjects(1).Copy
ActiveSheet.Paste
```

```
Selection.Name = "Neues Diagramm"
' ...
ActiveSheet.ChartObjects("Neues Diagramm").Delete
```

Ein wenig anders sieht es aus, wenn Sie nur die Diagrammdateien löschen, kopieren oder einfügen möchten, ohne dabei auch das *ChartObject*-Objekt zu verändern. In diesem Fall ist das *ChartArea*-Objekt von zentraler Bedeutung (weil für das *Chart*-Objekt keine *Copy*-Methode definiert ist). Beim Einfügen in ein anderes Diagrammobjekt müssen Sie sich dann aber auf dessen *Chart*-Objekt beziehen.

```
ActiveSheet.ChartObjects(1).Chart.ChartArea.Copy
ActiveSheet.ChartObjects(2).Chart.Paste
```

Auch beim Löschen von Diagrammdateien müssen Sie auf das *ChartArea*-Objekt zugreifen: *Clear* löscht alle Diagrammdateien, *ClearContents* nur die Diagramminhalte (gemeint sind in erster Linie die Datenreihen), *ClearFormats* nur die Formatinformationen.

Wenn Sie ein leeres *ChartObject*-Objekt in ein Tabellenblatt einfügen möchten (also einen noch leeren Diagrammrahmen), wenden Sie hierfür die *Add*-Methode auf *ChartObjects* an. An die Methode werden Positions- und Größenangaben übergeben (in der Einheit 1/72 Zoll, das sind etwa 0.35 mm). Dem neuen Objekt kann sofort ein Name gegeben werden.

```
ActiveSheet.ChartObjects.Add(0,0,200,100).Name = "neues diagramm"
```

Mehrere Diagramme ausrichten

Wenn Sie in einem Tabellenblatt zwei oder mehrere Diagramme mit der Maus platzieren, werden Sie feststellen, dass es relativ schwierig ist, zwei exakt gleich große, exakt untereinander liegende Diagramme herzustellen. Ein recht gutes Hilfsmittel ist dabei das Menü der ZEICHNEN-Symbolleiste. Mit den darin enthaltenen Einträgen können Sie zuvor markierte Objekte (auch Diagramme) ausrichten. Eine andere Variante besteht darin, einfach auf die *Left*-, *Top*-, *Width*- und *Height*-Eigenschaften der *ChartObject*-Objekte zuzugreifen. Die folgenden Anweisungen für den Direktbereich wurden verwendet, um die fünf Diagramme des Monatsprotokolls (siehe nächsten Abschnitt) horizontal nach Position und Größe des ersten Diagramms auszurichten.

```
set wb = Worksheets("Monatsprotokoll")
For i=2 To 5: wb.ChartObjects(i).Left = _
    wb.ChartObjects(1).Left: Next i
For i=2 To 5: wb.ChartObjects(i).Width = _
    wb.ChartObjects(1).Width: Next i
```

Fertige Diagramme oder benutzerdefinierte Formate verwenden

Die vollständige Erstellung eines Diagramms mit allen Formatierungsdetails per Programmcode ist möglich, aber mühsam und aufwendig zu programmieren. Wenn das Aussehen des Diagramms ohnedies vorgegeben ist (und nicht von den zu verarbeiten-

den Daten abhängt) ist es sinnvoller, das fertige Diagramm in einem Tabellen- oder Diagrammblatt zu speichern und im Programmcode nur noch die Datenzuordnung zu ändern. Die eigentliche Formatierung des Diagramms können Sie dann direkt mit der Maus und ohne Programmieraufwand vornehmen. (Die Prozedur *MonthlyProtocol* im folgenden Abschnitt zeigt dafür ein Beispiel.)

Nicht ganz so minimalistisch in der Programmierung, aber immer noch besser als eine vollkommene Neuprogrammierung, ist der Einsatz von benutzerdefinierten Diagrammtypen. Damit können Sie praktisch alle Formatdaten eines per Programmcode generierten Diagramms mit einer einzigen Anweisung verändern. Anschließend müssen Sie höchstens noch einige Anweisungen zur optimalen Größeneinstellung der einzelnen Diagrammelemente vornehmen. Die Zuweisung eines benutzerdefinierten Diagrammtyps an ein bestehendes Diagramm erfolgt seit Excel 97 über die Methode *ApplyCustomType* (bei früheren Versionen mit *AutoFormat*).

```
ActiveChart.ApplyCustomType ChartType:=xlUserDefined, _
    TypeName:="Tagesprotokoll"
```

Problematisch ist der Einsatz von benutzerdefinierten Diagrammtypen, wenn Sie eine fertige Excel-Anwendung auf einem anderen Rechner installieren möchten. Eigene Diagrammtypen werden in der Datei `Benutzerverzeichnis\Anwendungsdaten\Microsoft\Excel\Xlusrgal.xls` gespeichert. Diese Datei können Sie nicht auf einen anderen Rechner kopieren, weil Sie dadurch die benutzerdefinierten Formate eines anderen Anwenders überschreiben würden! Die Weitergabe eigener Diagrammtypen in einer Datei ist damit ausgeschlossen.

Es gibt aber einen Weg, um diese Einschränkung zu umgehen. Dazu legen Sie in Ihrer Anwendung ein Tabellenblatt an, in dem Sie für jedes benötigte Diagrammformat ein einfaches Beispieldiagramm einbetten. Beim ersten Start des Programms aktivieren Sie der Reihe nach diese Beispieldiagramme und speichern die darin enthaltenen Formatinformationen als benutzerdefinierte Diagrammtypen am jeweiligen Rechner.

```
Application.AddChartAutoFormat Chart:=ActiveChart, _
    Name:= "Neues Diagrammformat", Description:=""
```

Leider besteht keine Möglichkeit festzustellen, welche Diagrammtypen bereits definiert sind.

Diagramme drucken und exportieren

Der Ausdruck von Diagrammen erfolgt über die Methode *PrintOut*, die sowohl auf *Chart*- als auch auf *Workbook*-Objekte angewendet werden kann. Seit Excel 97 können Diagramme zudem mit *Export* in verschiedenen Formaten in eine Grafikdatei exportiert werden.

```
ActiveChart.Export "test.gif", "GIF"
```

Laut Excel-Dokumentation können im zweiten Parameter alle Grafikformate angegeben werden, für die Export-Filter installiert sind. Welche Filter es gibt, wie diese

bezeichnet sind und wie das Programm feststellen kann, ob ein bestimmter Filter installiert ist, verrät die Dokumentation freilich nicht. Sichern Sie entsprechende Prozeduren also mit *On Error* ab! Experimente mit *Export* sind z.B. mit den folgenden Format-Zeichenketten gelungen:

"GIF", "JPEG", "TIF", "TIFF", "PNG"

Nicht unterstützt werden dagegen ausgerechnet "BMP" und "WMF" – also die beiden Microsoft-Standardformate für Bitmaps und für einfache Vektorgrafiken. Wenn Sie Diagramme in diesem Format benötigen, können Sie die Methode *CopyPicture* einsetzen, die das Diagramm in die Zwischenablage kopiert. Leider endet der Export dort, d.h. Excel sieht keine Methode vor, den Inhalt der Zwischenablage in einer Datei zu speichern.

10.3 Beispiel – Automatische Datenprotokollierung

Die Datei 10\Chart.xls demonstriert die Anwendung von Excel zur Protokollierung von Messdaten. Die Notwendigkeit einer Datenprotokollierung ergibt sich immer dann, wenn relativ große Datenmengen über einen längeren Zeitraum dokumentiert und oft auch analysiert werden sollen (müssen). Die Datenherkunft kann ganz unterschiedlich sein: von den automatisch gemessenen Schadstoffwerten technischer Anlagen (etwa einer Kläranlage oder einer Rauchgasentschwefelung) bis zu den Ergebnissen der Qualitätskontrolle einer beliebigen Produktion.

Aufgabe der Protokollierung ist es, aus dem Zahlenfriedhof, der entweder in vielen kleinen oder in einer großen Datei bzw. Datenbank besteht, informative und übersichtliche Ausdrücke zu erzeugen. Es versteht sich von selbst, dass dabei Diagramme zur Visualisierung der Daten eine große Rolle spielen.

Da auf der beiliegenden CD-ROM kein technischer Prozess zur Datenproduktion mitgeliefert werden kann, stellt die Anwendung Chart.xls das Menükommando TESTDATEN ERZEUGEN zur Verfügung. Damit werden Excel-Dateien mit simulierten Messwerten erzeugt. In der Praxis benötigen Sie ein solches Kommando natürlich nur während der Testphase des Programms. In der Regel stehen Ihnen mehr (echte) Messdaten zur Verfügung, als Ihnen lieb ist, und Sie müssen diese Daten nicht gewaltsam durch simulierte Daten vermehren.

10.3.1 Die Bedienung des Beispielprogramms

Nach dem Laden der Datei erscheint ein eigenes Menü. Wenn Sie das Programm einfach schnell ausprobieren möchten, führen Sie der Reihe nach die Kommandos PROTOKOLL|TESTDATEN ERZEUGEN, |TAGESPROTOKOLL und |MONATSPROTOKOLL aus. Die dabei erscheinenden Dialoge zur Datumseingabe bestätigen Sie unverändert mit OK.

Das Programm produziert dann für jeden Tag des laufenden Monats eine Datendatei (Speicherbedarf insgesamt rund 900 kByte, Zeitaufwand ca. eine halbe Minute auf einem einigermaßen modernen Rechner). Anschließend wird das Tagesprotokoll des aktuellen Tags und das Monatsprotokoll des aktuellen Monats in der Seitenansicht präsentiert.



Bild 10.4: Der Dialog zur Eingabe eines Datumsbereichs

Testdaten

Durch das Menükommando PROTOKOLL|TESTDATEN ERZEUGEN werden für jeden Tag Dateien mit dem Namen D_#####.xls erstellt (also etwa D_19991231.xls für den 31.12.1999). Diese Dateien enthalten außer den Messdaten (je 96 Werten in den Datenreihen A1, A2, A3, B und C) auch Sechsstunden-Mittelwerte und -Maxima sowie Tagesmittelwerte und Tagesmaxima (siehe Bild 10.5). Die D_#####.xls-Dateien können Sie nach dem Test dieses Programms natürlich wieder löschen.

Zeit	A1	A2	A3	B	C		Zeit	A1	A2	A3	B	C
00:00	188,2	91,1	119,3	79,7	157,1							
00:15	189,4	98,1	134,9	78,3	158,9							
00:30	190,2	83,4	129,6	79,1	159,9							
00:45	189,7	92,8	132,0	80,1	159,5		00:00 bis 6:00	133,3	113,9	146,1	73,8	147,3
01:00	184,8	82,7	132,5	79,0	158,4		6:00 bis 12:00	67,3	197,5	183,4	139,6	89,2
01:15	179,9	81,8	133,1	78,1	157,7		12:00 bis 18:00	145,8	199,2	149,2	83,3	87,5
01:30	178,0	79,9	135,6	73,6	158,1		18:00 bis 24:00	203,0	76,7	148,3	85,2	118,7
01:45	172,6	79,6	139,4	73,2	159,5							
02:00	168,1	82,5	142,5	73,9	160,9		6-Stunden-Maxima					
02:15	160,8	88,0	143,7	73,0	161,1		00:00 bis 6:00	190,2	174,4	182,7	90,9	161,1
02:30	151,4	93,9	143,8	89,9	159,5		6:00 bis 12:00	116,8	209,0	200,4	163,9	110,3
02:45	141,8	98,7	144,5	86,4	156,5		12:00 bis 18:00	189,2	222,0	181,7	106,9	96,6
03:00	133,8	90,0	147,4	84,7	153,3		18:00 bis 24:00	215,4	128,1	184,8	109,0	146,1
03:15	128,9	90,8	151,4	85,4	150,9							
03:30	119,3	117,0	154,3	88,2	149,4		Tagesmittelwert	137,3	146,8	154,0	90,5	105,7
03:45	110,0	126,6	155,1	85,4	147,8		Tagesmaximum	215,4	222,0	200,4	168,9	161,1
04:00	99,8	135,2	154,9	83,9	145,0							
04:15	90,8	141,6	155,5	84,2	140,5							
04:30	84,4	148,8	153,0	87,7	134,7							

Bild 10.5: Der Aufbau der Tagesdateien für die Messdaten

Bei der Protokollierung der Daten wird angenommen, dass die Datenreihen A1, A2 und A3 inhaltlich zusammengehören. Daher werden A1 bis A3 im Tagesprotokoll in einem einzigen Diagramm dargestellt (siehe Bild 10.6). Im Monatsprotokoll war das aus Gründen der Übersichtlichkeit nicht mehr möglich, weil in den Diagrammen für jede Datenreihe sowohl der Tagesmittelwert als auch das Tagesmaximum in einem eigenen Linienzug dargestellt werden (siehe Bild 10.7).

10.3.2 Programmcode

Überblick über die Komponenten von Chart.xls

Die Excel-Datei Chart.xls besteht aus den folgenden Arbeitsblättern:

- »Intro«: Tabellenblatt mit einigen Informationen zur Bedienung der Anwendung.
- »DailyReport«: Tabellenblatt, in dem das Tagesprotokoll aufgebaut wird. Die darin enthaltenen Diagramme werden für jedes neue Protokoll gelöscht und neu aufgebaut.
- »MonthlyReport«: Tabellenblatt, in dem das Monatsprotokoll aufgebaut wird. Die darin enthaltenen Diagramme sind endgültig, sie werden im Programmcode nicht mehr verändert. Durch den Programmcode wird nur der Inhalt der Zellen B9:M39 verändert.
- »DataTemplate«: Tabellenblatt, das als Vorlage für die Dateien mit den simulierten Testdaten dient.

Der Aufbau der Tabellenblätter darf nicht verändert werden, da im Programmcode direkt auf bestimmte Zellen zugegriffen wird.

Der Programmcode verteilt sich auf folgenden Module:

- »DieseArbeitsmappe«: Menü beim Laden anzeigen, beim Schließen wieder entfernen.
- »FormDateInput«: Dialog zur Eingabe eines Datumsbereichs.
- »MenuEvents«: Ereignisprozeduren zu den Menükommandos.
- »CreateDateFiles«: Prozeduren zur Erzeugung der Testdaten.
- »CreateReports«: Prozeduren zum Aufbau und Ausdruck der Tages- und Monatsprotokolle.

Auf den folgenden Seiten werden die interessantesten Details des Programmcodes beschrieben. Dabei wird dieselbe Reihenfolge wie bei der Bedienung des Programms (Testdaten erzeugen, Tagesprotokoll, Monatsprotokoll) eingehalten. Der Code demonstriert nicht nur verschiedene Möglichkeiten der Diagrammprogrammierung, er zeigt auch, wie Sie Daten aus mehreren Excel-Dateien konsolidieren können, wenn die Excel-Funktion DATEN | KONSOLIDIEREN für Ihre Anwendungen zu wenig flexibel ist.

Testdaten erzeugen

Der Programmteil zur Erzeugung der Testdaten ist insofern von geringem Interesse, als er in einer praktischen Anwendung wegfällt (dort gibt es »echte« Daten). Im vorliegenden Beispiel erzeugt *GenerateDailyWorksheet* eine neue Excel-Datei auf Basis der Mustertabelle im Blatt »DataTemplate«. Dieses Muster enthält nicht nur diverse Formatierungsdaten, sondern auch einige Formeln zur Berechnung von Sechsstundenmittelwerten und -maxima sowie von Tagesmittelwerten und -maxima.

Die simulierten Testdaten werden auf der Basis von sechs überlagerten Sinusschwingungen unterschiedlicher Frequenz errechnet. Die Parameter dieser Funktionen (Amplitude, Frequenz und Phasenverschiebung) werden im globalen Feld *rndmat* gespeichert. Die globale Variable *rndInit* gibt an, ob dieses Feld gültige Werte enthält. Damit wird vermieden, dass die Zufallszahlen für jeden Tag neu ermittelt werden.

Die Zufallszahlen werden in der (hier nicht abgedruckten) Prozedur *InitRandomnumbers* initialisiert. Dabei wird versucht, für die drei Datenreihen A1, A2 und A3 ähnliche Werte zu wählen. Für jeden Tag neu wird die Prozedur *DailyRandomnumbers* aufgerufen. Diese Prozedur verändert die vorhandenen Werte des *zformat*-Felds geringfügig, damit die Daten nicht allzu regelmäßig aussehen.

```
' Datei 10\Chart.xls, Module CreateDataFiles
Dim rndInit As Boolean      'ist Zufallszahlenmatrix initialisiert?
Dim rndmat#(5, 18)         'Zufallszahlenmatrix
Const Pi = 3.1415927
' Arbeitsmappe mit Protokollzahlen für einen Tag erzeugen
Function GenerateDailyWorksheet(dat As Date) As Boolean
    Dim filename$          'Dateiname der neue Arbeitsmappe
    Dim wb As Workbook     'Arbeitsmappe
    Dim ws As Worksheet    'Tabellenblatt in dieser Mappe
    Dim cell As Range      'Verweis auf erste Datenzelle im Blatt
    Dim i%, j%, k%         'Schleifenvariablen
    Dim x#, z As Date      'Zwischenspeicher, Zeit
    filename = ThisWorkbook.Path + "\d_" + _
        Format(dat, "yyyymmdd") + ".xls"
    Application.DisplayAlerts = False
    ' neue Arbeitsmappe erzeugen, 'Messdatenvorlage'-Blatt aus dieser
    ' Arbeitsmappe dorthin kopieren, alle anderen Blätter löschen
    Set wb = Workbooks.Add
    ThisWorkbook.Sheets("DataTemplate").Copy Before:=wb.Sheets(1)
    For i = wb.Sheets.Count To 2 Step -1
        wb.Sheets(i).Delete
    Next i
    wb.Sheets(1).Name = "Tabelle1"
    ' Zufallszahlen in das Tabellenblatt einfügen
    Set ws = wb.Worksheets(1)
```

```

Set cell = ws.[A4]
ws.[a1] = "Messdaten vom " & dat
If Not rndInit Then InitRandomnumbers
DailyRandomnumbers
Application.Calculation = xlManual
For i = 1 To 96                                '00:00 bis 23:45
    z = dat + CDBl(#12:15:00 AM#) * (i - 1)
    cell.Cells(i, 1) = z
    cell.Cells(i, 1).NumberFormat = "hh:mm"
    For j = 1 To 5                              'fünf Datenreihen
        x = rndmat(j, 0)
        For k = 1 To 18 Step 3
            x = x + rndmat(j, k) * (1 + Sin(rndmat(j, k + 1) * z + _
                rndmat(j, k + 2)))
        Next k
        cell.Cells(i, j + 1) = x
    Next j
Next i
Application.Calculation = xlAutomatic
Application.DisplayAlerts = True
On Error Resume Next
' vorhandene Datei löschen
If Dir(filename) <> "" Then Kill filename
wb.SaveAs filename
wb.Close False
If Err = 0 Then
    GenerateDailyWorksheet = True
Else
    MsgBox "Es ist ein Fehler aufgetreten: " & Error
    GenerateDailyWorksheet = False
End If
End Function

```

ANMERKUNG

Es kommt bei automatischen Messprozessen immer wieder vor, dass auf Grund eines Fehlers für einige Zeit (Stunden, eventuell auch Tage) Messdaten ausfallen. In der obigen Prozedur wurde auf das Simulieren von Fehlern verzichtet. Die Protokollierung durch *Daily-* oder *MonthlyReport* funktioniert aber auch dann, wenn Sie aus den erzeugten Dateien einige Zahlenwerte einfach löschen, oder wenn Sie ganze Tagesdateien löschen. Vorsicht ist bei der Berechnung von Mittelwerten geboten: Fehlende Messwerte dürfen nicht als 0-Werte berücksichtigt werden! Die Excel-Tabellenfunktion *MITTELWERT* verhält sich in dieser Beziehung vorbildlich und berücksichtigt nur jene Zellen des angegebenen Bereichs, die nicht leer sind. Nur wenn *alle* Messwerte eines Mittelwertbereichs fehlen, liefert sie als Ergebnis den Fehler »Division durch 0«.

Tagesprotokolle

Das Tagesprotokoll enthält drei Diagramme, in denen der exakte Verlauf der Messwerte eingetragen ist. Dabei werden die Kurven A1, A2 und A3 in einem einzigen Diagramm vereint. Damit Diagramme von mehreren Tagen problemlos miteinander verglichen werden können, ist eine einheitliche Skalierung erforderlich. Aus diesem Grund ist der Y-Bereich starr auf den Wertebereich von 0 bis 300 eingestellt. (Normalerweise ändert Excel die Skalierung automatisch und passt sie an den tatsächlich auftretenden Wertebereich an.) Mit in das Tagesprotokoll integriert wurden eine tabellarische Übersicht der Tagesmittelwerte und der Tagesmaxima der fünf Kurven.

Das Tagesprotokoll zu einem angegebenen Datum wird durch die Prozedur *DailyProtocol* erstellt. Die Diagramme werden dabei vollständig durch den Programmcode erzeugt und in das Tabellenblatt »DailyReport« eingefügt. Bereits vorhandene Diagramme dieses Tabellenblatts (vom letzten Protokoll) werden vorher gelöscht.

Die Prozedur öffnet die Datei mit den Tagesdaten und kopiert daraus einige Eckdaten (Tagesmittelwerte und -maxima) in das Tabellenblatt »DailyReport«. Außerdem wird die Überschrift des Protokolls mit dem jeweiligen Datum ergänzt.

Zur Erzeugung der neuen Diagramme werden zuerst drei leere *ChartObject*-Rahmen im Tabellenblatt platziert. Durch *ChartWizard* wird darin ein Diagramm erzeugt, das den tatsächlichen Ansprüchen einigermaßen entspricht. Die drei *ChartWizard*-Anweisungen unterscheiden sich nur dadurch, dass den Diagrammen unterschiedliche Zellbereiche aus der Tagesdatentabelle zugeordnet werden.

Anschließend beginnt mit der Formatierung der Diagramme die eigentliche Feinarbeit. Die drei Diagramme können dabei in einer Schleife einheitlich bearbeitet werden. Die Prozedur endet damit, dass die Tagesdatendatei geschlossen wird und das Tagesprotokoll mit *PrintOut* ausgedruckt wird. (Wegen der Option *Preview:=True* erfolgt der Ausdruck nur in Form einer Seitenansicht.)

```
' Datei 10\Chart.xls, Module CreateReports
Sub DailyProtocol(dat As Date)
    Dim filename$           'Dateiname der Protokolldatei
    Dim protWBook As Workbook 'Arbeitsmappe der Protokolldatei
    Dim protWSheet As Worksheet 'Tabellenblatt in dieser Mappe
    Dim protRange As Range   'erste Datenzelle in diesem Blatt
    Dim chartWSheet As Worksheet 'Tabellenblatt mit Tagesdiagrammen
    Dim i%, chobj As ChartObject 'Schleifenvariablen
    Application.ScreenUpdating = False
    filename = ThisWorkbook.Path + "\d_" + _
        Format(dat, "yyyymmdd") + ".xls"
    If Dir(filename) = "" Then
        MsgBox "Die Datei " & filename & " existiert nicht. Bitte " & _
            "erzeugen Sie zuerst Testdaten!"
    End If
    Exit Sub
End If
```

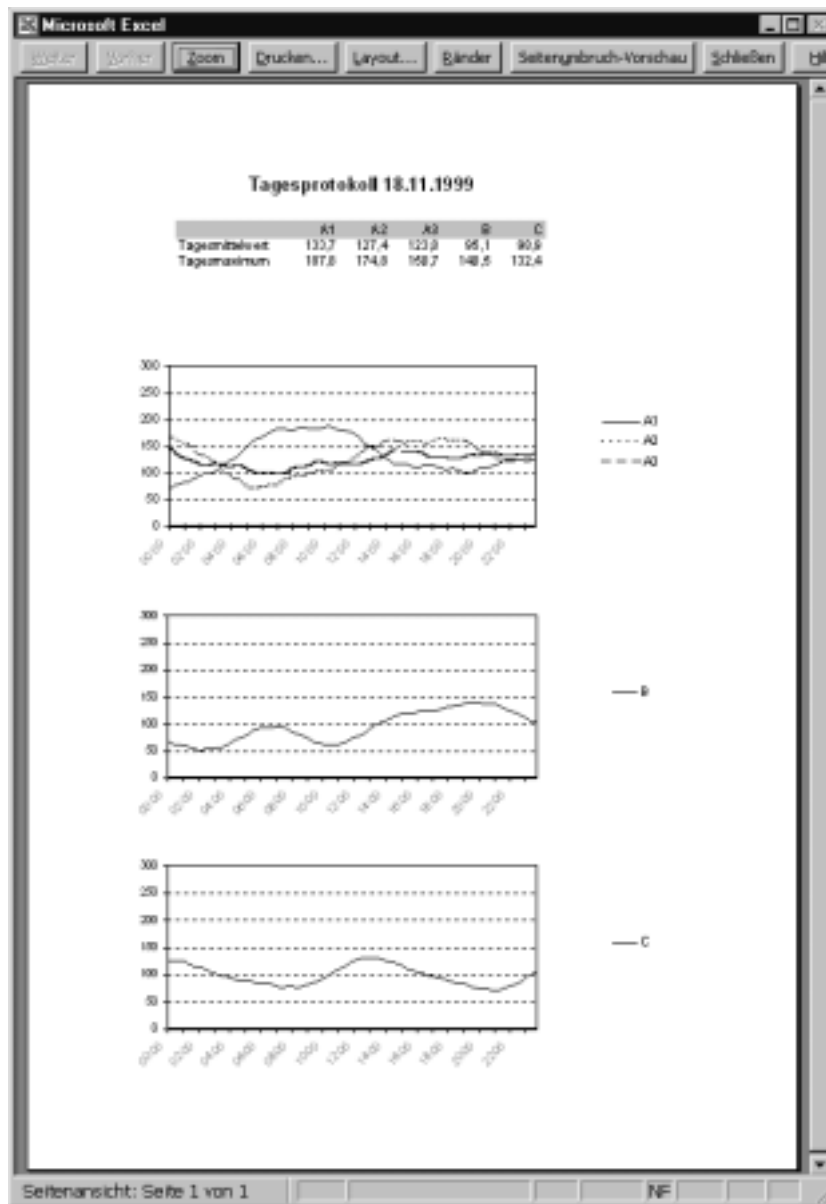


Bild 10.6: Ein Tagesprotokoll

```

Set protWBook = Workbooks.Open(filename)
Set protWSheet = protWBook.Worksheets(1)
Set protRange = protWSheet.[A4]
Set chartWSheet = ThisWorkbook.Worksheets("DailyReport")

```

```

' alle schon vorhandenen Diagramme dieses Blatts löschen
For Each chobj In chartWSheet.ChartObjects
    chobj.Delete
Next chobj
' Überschrift, Tagesmittelwerte und -maxima in Tabelle übertragen
chartWSheet.[ReportLabel] = "Tagesprotokoll " & dat
protWSheet.[I19:M19].Copy
chartWSheet.[DailyAverage].PasteSpecial xlValues
protWSheet.[I21:M21].Copy
chartWSheet.[DailyMax].PasteSpecial xlValues
' die drei Diagramme erstellen
For i = 1 To 3
    chartWSheet.ChartObjects.Add(30, 150 + 200 * (i - 1), 400, 185). _
        Name = "Tagesdaten " & i
    chartWSheet.ChartObjects("Tagesdaten " & i).Activate
    If i = 1 Then
        ActiveChart.ChartWizard protWSheet.[A3:D99], _
            xlLine, 4, xlColumns, 1, 1
    ElseIf i = 2 Then
        ActiveChart.ChartWizard protWSheet.[A3:A99,E3:E99], _
            xlLine, 4, xlColumns, 1, 1
    ElseIf i = 3 Then
        ActiveChart.ChartWizard protWSheet.[A3:A99,F3:F99], _
            xlLine, 4, xlColumns, 1, 1
    End If
Next i
' die Diagramme formatieren
For Each chobj In chartWSheet.ChartObjects
    chobj.Border.LineStyle = xlNone           'keine Umrahmung
    With chobj.Chart
        .HasTitle = False                   'kein Titel
        .PlotArea.Border.LineStyle = xlAutomatic 'Umrandung
        .PlotArea.Interior.ColorIndex = xlNone 'innen kein Muster
        .Axes(xlCategory).TickLabelSpacing = 8
        .Axes(xlCategory).TickMarkSpacing = 4 'Beschriftung
        .Axes(xlValue).MinimumScale = 0     ' und Skalierung
        .Axes(xlValue).MaximumScale = 300  ' der X-Achse
        .Axes(xlCategory).TickLabels.Orientation = 45 '45 Grad-Text
    End With
    For i = 1 To .SeriesCollection.Count    'Datenlinien
        .SeriesCollection(i).Border.ColorIndex = 1
        .SeriesCollection(i).Border.Weight = xlThin
        .SeriesCollection(i).Border.LineStyle = xlContinuous
        .SeriesCollection(i).MarkerStyle = xlNone
    Next i
Next i

```

```

If .SeriesCollection.Count > 2 Then           '2. und 3.
    .SeriesCollection(2).Border.LineStyle = xlDot 'Linienzug
    .SeriesCollection(3).Border.LineStyle = xlDash 'optisch
End If                                         'unterscheiden
' Größe des eigentlichen Diagramms und der Legende
.PlotArea.Left = 5
.PlotArea.Top = 5
.PlotArea.Width = 290
.PlotArea.Height = 140
.Legend.Left = 340
.Legend.Width = 50
.Legend.Border.LineStyle = xlNone
End With
Next chobj
ActiveWindow.Visible = False 'Diagramm deaktivieren
protWBook.Close
chartWSheet.PrintOut Preview:=True
End Sub

```

Monatsprotokolle

Die Monatsprotokolle sind etwas aufwendiger als die Tagesprotokolle gestaltet und beanspruchen insgesamt drei Seiten. Die erste Seite besteht aus einer Übersicht aller Tagesmittelwerte und -maxima sowie aus den daraus resultierenden Monatsmittelwerten und -maxima. Die beiden folgenden Seiten enthalten drei bzw. zwei Diagramme mit dem Verlauf der Mittelwerte bzw. der Maxima. Die Kurvenzüge für die Mittelwerte sind dabei geglättet (Kurvenzug anklicken, Kontextmenü DATENREIHEN FORMATIEREN | MUSTER, Option LINIE GLÄTTEN). Bild 10.7 zeigt die zweite Seite des Monatsprotokolls mit den Kurvenzügen für die Messwerte A1 bis A3.

Zur der Erzeugung des Monatsprotokolls wurde eine vollkommen andere Vorgehensweise als beim Tagesprotokoll gewählt. Die Diagramme wurden (per Maus) im Tabellenblatt »MonthlyReport« angelegt und werden von der Prozedur *MonthlyProtocol* überhaupt nicht angerührt. *MonthlyProtocol* verändert lediglich jene Datenzellen, auf die die fertigen Diagramme zugreifen.

Diese Vorgehensweise hat Vor- und Nachteile: Der Vorteil besteht darin, dass der Programmieraufwand viel geringer ist. Sie können also auch mit minimaler Erfahrung bei der Diagrammprogrammierung zu guten Ergebnissen gelangen. Den Nachteil bemerken Sie dann, wenn Sie versuchen, fünf gleiche Diagramme per Mausklick zu erzeugen. Das ist beinahe ebenso mühsam wie die Programmierung! (Selbst dann, wenn Sie zuerst ein Diagramm erstellen, dieses dann kopieren und nur noch die Zellbereiche der Datenreihen verändern.) Außerdem ist diese Vorgehensweise natürlich nur dann möglich, wenn das Diagramm wie im vorliegenden Beispiel von den Daten weitgehend unabhängig ist. Wenn dagegen die Anzahl der Datenreihen, die Anzahl

der Datenpunkte, der Wertebereich der Datenreihen etc. variieren, dann führt kein Weg an einem »echten« Programm vorbei.

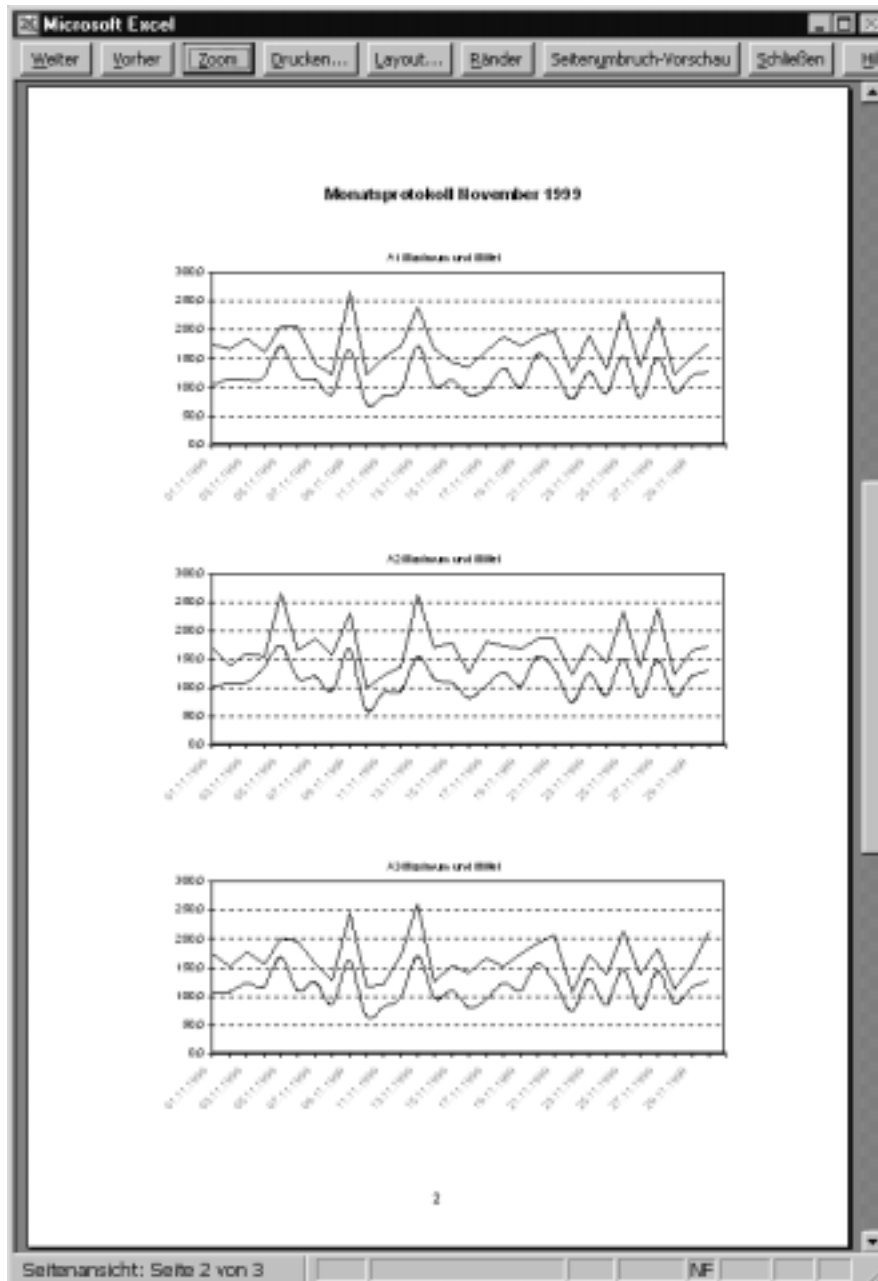


Bild 10.7: Eine Seite aus dem dreiseitigen Monatsprotokoll

Die Diagramme gehen von 31 Tagen aus. Bei den Monaten mit weniger Tagen bleiben am rechten Diagrammrand ein bis drei Datenpunkte leer. Dadurch wird zwar der zur Verfügung stehende Raum nicht ganz optimal genutzt, dafür ergibt sich aber ein wesentlicher Vorteil: Die Skalierung der X-Achse ist unabhängig von der Anzahl der Tage pro Monat. Die Diagramme sind damit besser vergleichbar.

Nun zum Programmcode, der aus den oben genannten Gründen keine einzige Zeile mit diagrammtypischen Anweisungen enthält. Die Prozedur ist eher ein Beispiel dafür, wie Daten aus bis zu 31 Dateien in einer einzigen Tabelle konsolidiert werden können. Die einzelnen Dateien werden dazu nicht geöffnet, vielmehr wird in Formeln der Art = 'C:\Eigene Dateien\Test\[D_970101.XLS]Tabelle1'!\$L\$19 direkt auf einzelne Zellen anderer Tabellen zugegriffen. Diese Form des Datenzugriffs geht überraschend schnell vor sich. Die Erstellung des Monatsprotokolls dauert kaum länger als jene des Tagesprotokolls.

Der komplizierteste Teil der Prozedur betrifft die Erzeugung dieser Formeln, die durch die Veränderung der *FormulaR1C1*-Eigenschaft der jeweiligen Zellen in die Tabelle eingetragen werden. Die Formeln müssen relativ mühsam als Zeichenketten erstellt werden. Das R1C1-Format ist für solche Aufgaben besser geeignet, weil so zumindest die Umwandlung von Spaltennummern in Buchstaben entfällt.

```
Sub MonthlyProtocol(dat As Date)
    Dim sdat As Date, edat As Date 'Start- und Enddatum
    Dim nrdays As Integer         'Anzahl der Tage
    Dim chartWSheet As Worksheet 'Tabellenblatt mit Monatsdiagrammen
    Dim chartRange As Range       'erste Datenzelle in diesem Blatt
    Dim z As Date, i%, j%         'Schleifenvariablen
    Dim filename As String

    sdat = DateSerial(Year(dat), Month(dat), 1)
    nrdays = DateSerial(Year(dat), Month(dat) + 1, 1) - _
        DateSerial(Year(dat), Month(dat), 1)
    edat = dat + nrdays - 1
    ThisWorkbook.Activate
    Set chartWSheet = ThisWorkbook.Worksheets("MonthlyReport")
    chartWSheet.Activate
    chartWSheet.[a1].Select
    Set chartRange = chartWSheet.[B9]
    ' Monatstabelle aufstellen
    Application.Calculation = xlManual
    chartWSheet.[B1] = "Monatsprotokoll " & Format(dat, "mmmm yyyy")
    For i = 1 To nrdays
        z = dat + i - 1
        chartRange.Cells(i, 1) = z
        filename = ThisWorkbook.Path + "\d_" + _
            Format(z, "yyyymmdd") + ".xls"
    
```

```

If Dir(filename) = "" Then
    For j = 1 To 5
        chartRange.Cells(i, 1 + j).FormulaR1C1 = ""
        chartRange.Cells(i, 7 + j).FormulaR1C1 = ""
    Next j
Else
    filename = "=" & ThisWorkbook.Path + _
        "\[d_" + Format(z, "yyyymmdd") & ".xls]Tabelle1'"
    For j = 1 To 5
        chartRange.Cells(i, 1 + j).FormulaR1C1 = _
            filename & "!R19C" & 8 + j
        chartRange.Cells(i, 7 + j).FormulaR1C1 = _
            filename & "!R21C" & 8 + j
    Next j
End If
Next i
If nrdays < 31 Then
    For i = nrdays + 1 To 31
        For j = 1 To 12
            chartRange.Cells(i, j).ClearContents
        Next j
    Next i
End If
Application.Calculate
chartWSheet.Range("B9:M39").Copy
chartWSheet.Range("B9:M39").PasteSpecial Paste:=xlValues
Application.CutCopyMode = False
chartWSheet.PrintOut Preview:=True
Application.Calculation = xlAutomatic
End Sub

```

Nachdem alle Verweise in die Tabelle eingetragen und die Tabelle auf dieser Basis neu berechnet wurde, wird der gesamte Zellbereich in die Zwischenablage kopiert. Mit *PasteSpecial* werden anschließend nur die Zahlenwerte (anstatt der Formeln) eingetragen. Dieser Vorgang spart Speicher und erhöht die weitere Verarbeitungsgeschwindigkeit. Außerdem kommt Excel nicht auf die Idee, bei der nächsten Gelegenheit zu fragen, ob es die vorhandenen Verweise aktualisieren soll.

Die Prozedur endet wie *DailyProtocol* mit dem Ausdruck der Tabelle samt den fünf darin enthaltenen Diagrammen. Bei der Layout-Gestaltung der Tabelle mit DATEI|SEITE EINRICHTEN wurde übrigens als Kopfzeile »keine« und als Fußzeile eine Seitennummer eingestellt (weil das Protokoll ja immerhin drei Seiten lang ist).

Menüverwaltung

Die Menüverwaltung liefert gegenüber den Beispielen der vorangegangenen Kapitel keine neuen Informationen mehr, weswegen auf den Abdruck der Ereignisprozeduren verzichtet wird. Das Menü ist als eigenes *CommandBar*-Objekt realisiert. Es wird beim Laden von *Chart.xls* in *Workbook_Open* sichtbar gemacht und in *Workbook_BeforeClose* wieder verborgen.

Dialogverwaltung

Der Dialog *FormDateInput* wird universell für die drei Kommandos PROTOKOLL | TEST-DATEN ERZEUGEN, ... | TAGESPROTOKOLL und ... | MONATSPROTOKOLL verwendet. Je nach Verwendungszweck wird der Text im Textfeld *lblInfo* geändert. Durch die Prozeduren *ProtocolMenu_GenerateNewFiles*, *_DailyProtocol* und *_MonthlyProtocol*, von denen hier nur eine abgedruckt ist, werden außerdem die Texte in den Textfeldern *txtFrom* und *txtTo* voreingestellt.

Die beiden Daten können durch Drehpfeile vergrößert bzw. verkleinert werden. Dazu werden die Drehwerte auf 0 voreingestellt. Der zulässige Wertebereich geht von -1000 bis 1000, Sie können also das Datum theoretisch um plus/minus 1000 Tage verändern. (Theoretisch deswegen, weil Sie kaum so viel Geduld aufbringen werden. Viel schneller geht es, wenn Sie das Datum einfach per Tastatur eingeben.)

```
' Datei 10\Chart.xls, Module MenuEvents
' Menükommando, um Monatsprotokoll erzeugen
Sub ChartSampleMenu_MonthlyProtocol()
    Dim dat As Date, lastmonth As Integer
    lastmonth = -1
    With FormDateInput
        .dat1 = DateSerial(Year(Now), Month(Now), 1)
        .dat2 = DateSerial(Year(Now), Month(Now), _
            DateSerial(Year(Now), Month(Now) + 1, 1) - _
            DateSerial(Year(Now), Month(Now), 1))
        .txtFrom = CStr(.dat1)
        .txtTo = CStr(.dat2)
        .spinTo = 0: .spinFrom = 0
        .lblInfo = "Geben Sie an, für welchen Datumsbereich Sie " & _
            "Monatsprotokolle erstellen und ausdrucken möchten."
        .Show
    If .result = False Then Exit Sub
    ' Testdaten erzeugen
    Application.ScreenUpdating = False
    Application.DisplayStatusBar = True
```

```

For dat = CDate(.txtFrom) To CDate(.txtTo)
  If lastmonth <> Month(dat) Then
    Application.StatusBar = "Monatsprotokoll vom " & _
      Format(dat, "mmm yy") & " erzeugen"
    MonthlyProtocol CDate(dat)
    lastmonth = Month(dat)
  End If
Next dat
Application.StatusBar = False
Application.DisplayStatusBar = False
End With
End Sub

```

Sofern die Eingabe mit OK abgeschlossen und in *btnOK_Click* kein Eingabefehler entdeckt wurde, werden in einer Schleife alle Tage des Datumbereichs durchlaufen. Jedes Mal, wenn sich dabei der Monat ändert, wird *MonthlyProtocol* aufgerufen. Zugegebenermaßen ist dieser Algorithmus nicht übermäßig raffiniert programmiert – es ist aber sicherlich die einfachste Lösung, die für beliebige Zeitbereiche funktioniert (auch für mehr als 12 Monate). Eine Berechnung des Monatsersten jedes neuen Monats würde vermutlich mehr Zeit beanspruchen als ein einfaches Durchlaufen aller Tage. Auf jeden Fall hätte es ein bisschen mehr Denkaufwand beim Programmieren bedeutet, und Programmierer sind bekanntermaßen nicht immer zum Denken aufgelegt ...

Die eigentlichen Dialogereignisprozeduren fallen vergleichsweise kurz und trivial aus. Beachten Sie bitte, dass das Drehfeld nicht synchronisiert wird, wenn im Textfeld ein neues Datum per Tastatur eingegeben wird. Aus diesem Grund ist es nicht möglich, ein über die Tastatur eingegebenes Datum anschließend mit dem Drehfeld zu verändern.

```

' Ereignisprozeduren zum Formular für die Datumseingabe
Option Explicit
Public result As Boolean, dat1 As Date, dat2 As Date
Private Sub btnCancel_Click()
  result = False
  Hide
End Sub

Private Sub btnOK_Click()
  If IsDate(txtFrom) And IsDate(txtTo) Then
    result = True
    Hide
  Else
    MsgBox "Ungültige Datumseingabe!"
  End If
End Sub

```

```

Private Sub spinFrom_Change()
    txtFrom = CStr(dat1 + spinFrom)
End Sub

Private Sub spinTo_Change()
    txtTo = CStr(dat2 + spinTo)
End Sub

```

10.4 Syntaxzusammenfassung Diagramme

Dieser Abschnitt fasst wirklich nur die allerwichtigsten Objekte, Methoden und Eigenschaften zusammen. Eine Zusammenfassung der Objekthierarchie aller Diagrammobjekte finden Sie in Kapitel 16. Dort sind auch alle Objekte (in alphabetischer Reihenfolge) kurz beschrieben. In den folgenden Syntaxboxen steht *wb* als Abkürzung für ein *Workbook*-Objekt, *ws* für ein *Worksheet*-Objekt, *chobj* für ein *ChartObject*-Objekt und *ch* für ein *Chart*-Objekt.

Diagrammobjekte

<i>ws.ChartObjects(..)</i>	eingebettetes Diagrammobjekt auswählen
<i>ws.ChartObjects.Add ..</i>	neuen (leeren) Diagrammrahmen
<i>chobj.Select</i>	entspricht einfachem Mausklick
<i>chobj.Activate</i>	entspricht doppeltem Mausklick
<i>ActiveWindow.Visible = False</i>	deaktivieren
<i>chobj.Chart</i>	verweist auf Diagrammobjekt
<i>chobj.Copy</i>	Diagrammobjekt samt Diagramm kopieren
<i>ws.Paste: Selection.Name = ".."</i>	Diagrammobjekt samt Diagramm einfügen
<i>chobj.Duplicate.Name = ".."</i>	vorhandenes Diagrammobjekt duplizieren
<i>chobj.Delete</i>	Diagrammobjekt samt Diagramm löschen

Diagramme

<i>ActiveChart</i>	verweist auf aktives Diagramm
<i>wb.Charts(..).Select</i>	wählt Diagrammblatt aus
<i>ch.ChartArea.Copy</i>	kopiert Diagramminhalt
<i>ch.Paste</i>	fügt Diagramminhalt ein
<i>ch.ChartArea.Clear</i>	löscht gesamtes Diagramm
<i>ch.ChartArea.ClearContents</i>	löscht nur die Daten
<i>ch.ChartArea.ClearFormats</i>	löscht nur die Formate
<i>ch.ChartWizard ...</i>	Diagramm mit Assistenten erzeugen
<i>ch.ApplyCustomType ...</i>	benutzerdefiniertes Format verwenden
<i>Application.AddChartAutoFormat ...</i>	neues benutzerdefiniertes Format speichern

<i>ch.CopyPicture</i>	kopiert Diagramm als Grafik oder Bitmap in die Zwischenablage
<i>ch.Export</i> <i>ch.PrintOut</i>	speichert das Diagramm in einer Grafikdatei druckt das Diagramm aus
<i>ch.ChartArea</i>	verweist auf Gesamthintergrund
<i>ch.PlotArea</i>	verweist auf Hintergrund der Grafik
<i>ch.Floor, ch.Walls</i>	verweist auf Boden und Wände (3D-Diagramme)
<i>ch.ChartTitle</i>	verweist auf Diagrammtitel
<i>ch.Legend</i>	verweist auf Legende
<i>ch.Axes(..)</i>	verweist auf Achsen
<i>ch.SeriesCollection(..)</i>	verweist auf Datenreihen

10.5 Zeichnungsobjekte (Shapes)

Überblick

Das *Shape*-Objekt dient primär zur Darstellung von AutoFormen (Linien, Rechtecke, Pfeile, Sterne etc. – siehe Symbolleiste ZEICHNEN). Es löst damit die diversen Zeichnungsobjekte aus Excel 5/7 ab. Verwirrung stiftet allerdings die große Anzahl verwandter Objekte.

Hierarchie der Shape-Objekte	
Worksheet/Chart	
└─ Shapes	alle <i>Shape</i> -Objekte innerhalb des Blatts
└─ Shape	ein <i>Shape</i> -Objekt
├─ ConnectorFormat	Verbindung zu anderen Objekten
├─ ControlFormat	zusätzliche Eigenschaften für Steuerelemente
├─ FillFormat	Hintergrundmuster (via <i>Fill</i> -Eigenschaft)
├─ GroupShapes	Einzelobjekte (via <i>GroupItems</i> , wenn <i>Type=msoGroup</i>)
└─ Shape	
├─ HyperLink	Querverweise und Internetlinks
├─ LineFormat	Linieneigenschaften (via <i>Line</i>)
├─ LinkFormat	zusätzliche Eigenschaften für OLE-Objekte
├─ OLEFormat	noch mehr Eigenschaften für OLE-Objekte
├─ PictureFormat	Eigenschaften für Bildobjekte
├─ Range	Verankerungszellen (via <i>TopLeft-/BottomRightCell</i>)
├─ Shadow	Eigenschaften für den Schatten
├─ ShapeNodes	Liniensegmente (via <i>Nodes</i> , wenn <i>Type=msoFreeform</i>)
└─ ShapeNode	
├─ ShapeRange	Einzelobjekte bei Mehrfachbearbeitung (via <i>Range</i>)
└─ Shape	
├─ TextEffectFormat	Eigenschaften für WordArt-Objekt
├─ TextFrame	Textbox innerhalb eines AutoForm-Objekts
└─ ThreeDFormat	3D-Effekte (via <i>ThreeD</i>)

Die *Shapes*-Aufzählung ermöglicht den Zugriff auf alle *Shape*-Objekte eines Tabellen- oder Diagrammblatts. Zum Einfügen neuer Zeichnungsobjekte stehen eine ganze Reihe von Methoden zur Verfügung – *AddShape* für AutoFormen, *AddLine* für Linien und Pfeile etc.

ShapeRange ermöglicht die gemeinsame Bearbeitung mehrerer *Shape*-Objekte (in gleicher Weise, als wären diese Objekte mit Shift und Maus markiert).

Freihandformen (also etwa frei gezeichnete Linienzüge) stellen eine Sonderform von *Shape*-Objekten dar. In diesem Fall verweist die Eigenschaft *ShapeNodes* auf eine gleichnamige Auflistung von *ShapeNode*-Objekten. Diese Objekte enthalten unter anderem die Koordinatenpunkte der einzelnen Liniensegmente.

Ein *Shape*-Objekt wird auch zur Verwaltung einer so genannten Gruppe verwendet (im interaktiven Betrieb: Kontextmenükommandos GRUPPIERUNG). In diesem Fall führt die Eigenschaft *GroupItems* zu einem *GroupShape*-Objekt, das seinerseits die Verwaltung der Gruppenelemente übernimmt. Als Gruppenelemente kommen nicht nur *Shape*-Objekte in Frage, sondern auch Diagramme, OLE-Objekte etc.

Shape wird schließlich zur Verwaltung vollkommen fremder Objekte verwendet – etwa für MS-Forms-Steuererelemente (*Type=msoOLEControlObject*). In diesem Fall steht *Shape* zwischen dem Tabellen- und Diagrammblatt und dem eigentlichen Objekt. *Shape* kümmert sich dann unter anderem um die Positionierung des Steuererelements. Zur Kommunikation zwischen Blatt und Steuererelement wird das *ControlFormat*-Objekt verwendet, das über die gleichnamige Eigenschaft von *Shape* angesprochen wird. *ControlFormat* ist zumeist transparent, weil dessen Eigenschaften im Eigenschaftsfenster des Steuererelements auftauchen und wie Steuererelementeigenschaften verwendet werden können.

Shape-Eigenschaften

Objektyp: Die zwei wichtigsten Eigenschaften sind sicherlich *Type* und *AutoShapeType*. Wenn für *Type=msoAutoShape* eingestellt wird, dann kann mit *AutoShapeType* einer der zahllosen AutoForm-Typen angegeben werden (es gibt über 130!). Wenn durch das *Shape*-Objekt dagegen keine AutoForm repräsentiert wird, wird der Objektyp durch die *msoShapeType*-Konstanten angegeben. Elemente wie *msoChart*, *msoComment*, *msoEmbeddedOLEObject*, *msoFreeForm*, *msoGroup*, *msoOLEControlObject* oder *msoText-Box* beweisen, dass Excel-intern jedes Objekt, das sich außerhalb einer Zelle befindet, durch *Shape*-Objekte verwaltet wird.

Positionierung: Zu jedem Objekt wird der linke obere Eckpunkt (*Left* und *Top*) sowie Breite und Höhe (*Width* und *Height*) gespeichert. Diese Koordinaten beziehen sich auf das linke obere Eck des Dialogs bzw. Tabellenblatts. *TopLeftCell* und *BottomRightCell* geben darüber hinaus die Zellen unter dem linken oberen bzw. unter dem rechten unteren Eck an. *Placement* bestimmt, wie sich das Steuererelement bei einer Veränderung der Tabelle verhalten soll (*xlMoveAndSize*, *xlMove* oder *xlFreeFloating*).

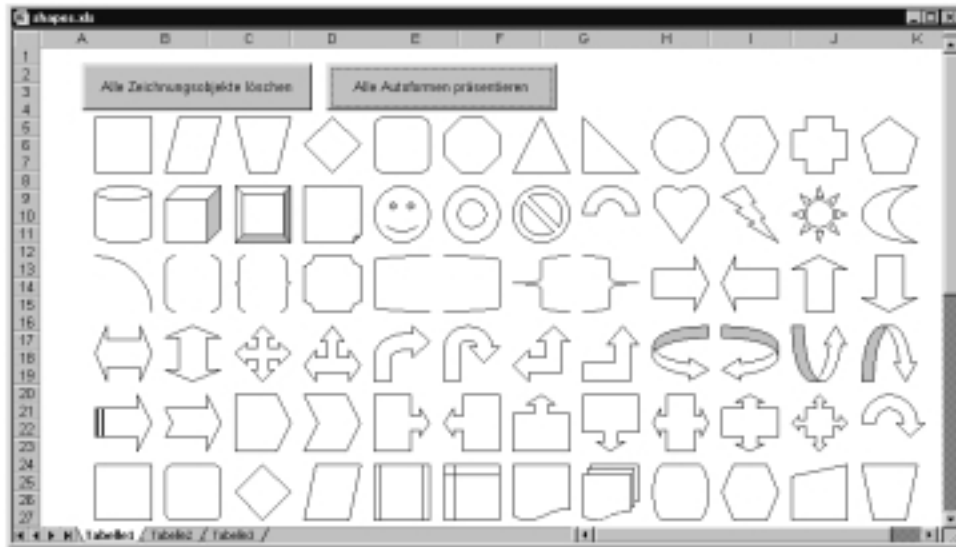


Bild 10.8: Ein Teil der vordefinierten AutoFormen

Format: Die Möglichkeiten zur optischen Gestaltung sind beinahe grenzenlos. Jede der folgenden Eigenschaften führt auf ein eigenes Objekt (dessen Name in Klammern angegeben wird, wenn er vom Eigenschaftsnamen abweicht): *Adjustments*, *Callout* (*CalloutFormat*), *Fill* (*FillFormat*), *Hyperlink*, *Line* (*LineFormat*), *PictureFormat*, *Shadow* (*ShadowFormat*), *TextEffect* (*TextEffectFormat*), *TextFrame* und *ThreeD* (*ThreeDFormat*). Ob dieser Überfluss an Objekten nicht zu viel des Guten ist?

Sonstiges: Je nachdem, welche Objekte durch *Shape* repräsentiert werden, stehen weitere Eigenschaften zur Verfügung: *ConnectorFormat* (wenn das Objekt mit anderen Objekten verbunden ist), *ControlFormat* (bei Steuerelementen), *GroupItems* (bei Objektgruppen), *Nodes* (bei Freihandobjekten) sowie *LinkFormat* und *OLEFormat* (bei OLE-Objekten).

HINWEIS

Beachten Sie, dass die *Shape*-Objekte zwar in der Excel-Bibliothek definiert sind, die zugeordneten Konstanten allerdings in der Office-Bibliothek. Nach dem Laden alter Excel-5-/-7-Dateien ist die Office-Bibliothek normalerweise nicht aktiviert – dies muss mit EXTRAS | VERWEISE erfolgen.

Beispiel

Die Zeichnungsobjekte aus Bild 10.8 wurden mit der Schleife in *btnShowAllAutoShapes_Click* erzeugt. Kurz zur Syntax von *AddShape*: der erste Parameter gibt den AutoForm-Typ an (1 bis 137), die vier folgenden Parameter bestimmen Ort (*Left/Top*) und Größe (*Width/Height*) des Objekts. Das Koordinatensystem beginnt in der linken oberen Ecke des Tabellenblatts.

```
' Datei 10\Shapes.xls, Tabelle1
Private Sub btnShowAllAutoShapes_Click()
    Dim i&
    For i = 0 To 136
        ActiveSheet.Shapes.AddShape i + 1, _
            40 + 50 * (i Mod 12), 50 + 50 * (i \ 12), 40, 40
    Next
End Sub
```

Um die Zeichnungsobjekte wieder zu löschen, dient die folgende Prozedur. Entscheidend ist dabei der *Type*-Test: ohne ihn würden auch die Buttons aus dem Tabellenblatt gelöscht!

```
Private Sub btnDeleteShapes_Click()
    Dim s As Shape
    For Each s In ActiveSheet.Shapes
        If s.Type = msoAutoShape Or s.Type = msoLine Then s.Delete
    Next
End Sub
```

Die Prozedur *btnStar_Click* zeichnet einen Stern aus bunten Pfeilen. Beachten Sie, dass Pfeile nicht zu den AutoFormen zählen, sondern eine eigene *Shape*-Kategorie bilden. Aus diesem Grund muss *AddLine* statt *AddShape* eingesetzt werden. *ForeColor* verweist auf ein *ColorFormat*-Objekt, mit dem die Farbe des Objekts eingestellt werden kann.

HINWEIS

Der Programmcode lässt vermuten, dass in Excel unbeschränkt viele Farben zur Verfügung stehen. Leider ist das nicht der Fall. Vielmehr steht nur eine Palette von 56 Farben zur Verfügung (offenbar ein Relikt von frühen Excel-Versionen). Daher bewirkt die Zuweisung einer *RGB*-Farbe nur, dass die am ehesten passende Farbe aus dieser Palette ausgewählt wird.

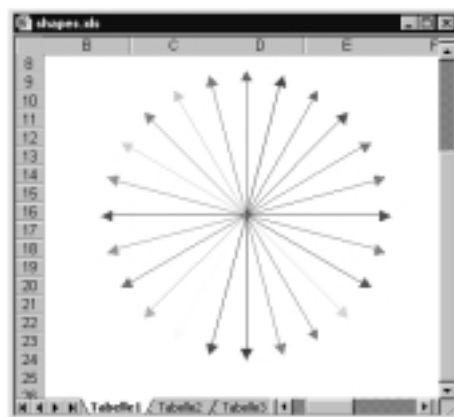


Bild 10.9: Ein Stern aus bunten Pfeilen

```

Private Sub btnStar_Click()
    Dim degree#
    Dim s As Shape
    Const Pi = 3.1415927
    Randomize
    For degree = 0 To 2 * Pi Step Pi / 12
        Set s = ActiveSheet.Shapes.AddLine(200, 200, _
            200 + 100 * Sin(degree), 200 + 100 * Cos(degree))
        s.Line.EndArrowheadStyle = msoArrowheadTriangle
        s.Line.EndArrowheadLength = msoArrowheadLengthMedium
        s.Line.EndArrowheadWidth = msoArrowheadWidthMedium
        s.Line.ForeColor.RGB = RGB(Rnd * 255, Rnd * 255, Rnd * 255)
    Next
End Sub

```

10.6 Organigramme und andere Diagramme

Beginnend mit Excel 2002 können Sie mit EINFÜGEN|SCHEMATISCHE DARSTELLUNG Organigramme sowie fünf weitere Diagrammtypen in ein Excel-Tabellenblatt einfügen (Zyklusdiagramm, Radialdiagramm, Pyramidendiagramm, Venn-Diagramm und Ziel-diagramm). Diese Diagramme erscheinen in einer Grundform und können dann durch eigene Texte, Formatierungen und zusätzliche Subobjekte erweitert werden. Bei der Bearbeitung der Diagramme helfen die ORGANIGRAMM-Symbolleiste (nur für Organigramme) bzw. die DIAGRAMM-Symbolleiste (bei den anderen fünf Diagrammtypen).

Um Diagramme per VBA-Programmierung zu erzeugen bzw. zu verändern, müssen Sie die neuen *DiagramXxx*-Objekte einsetzen, die im Mittelpunkt dieses Abschnitts stehen: *Diagram* beschreibt ein gesamtes Diagramm, *DiagramNode* eines seiner Elemente. Die Aufzählungen *DiagramNodes* und *DiagramNodeChildrens* helfen bei der Verwaltung der Diagrammelemente.

HINWEIS

Bevor Sie sich auf das Abenteuer der *Diagram*-Programmierung einlassen, einige Anmerkungen:

- Die Makroaufzeichnung funktioniert weder beim Erzeugen noch beim Verändern von Geschäftsdiagrammen. Das macht die Erforschung der *Diagram*-Objekte zu einem mühsamen Unterfangen.
- Auch die *Diagram*-Objekte selbst sind offensichtlich unausgereift. Ein besonders augenscheinlicher Mangel: Selbst erzeugte Diagramme können nicht beschriftet werden, die Beschriftung vorhandener Diagramme kann nicht geändert werden.
- Speichern Sie Ihr Projekt regelmäßig! Ich habe beim Experimentieren gleich eine ganze Reihe von Abstürzen erlebt.

Diagramm erzeugen

Um ein neues Geschäftsdiagramm zu erzeugen, verwenden Sie die Methode *AddDiagram* des *Shapes*-Objekts. Dabei müssen Sie den gewünschten Diagrammtyp (*msoDiagramXxx*-Konstante) sowie Position und Größe angeben. Als Resultat erhalten Sie ein *Shape*-Objekt, dessen Eigenschaft *Diagram* auf das gleichnamige *Diagram*-Objekt verweist.

```
Dim s As Shape
Dim d As Diagram
Dim ws As Worksheet
Set ws = Worksheets(1)
Set s = ws.Shapes.AddDiagram(msoDiagramRadial, 10, 10, 200, 100)
Set d = s.Diagram
```

Diagrammelemente hinzufügen

Ein neues Geschäftsdiagramm ist – unabhängig von seinem Typ – vorerst leer. Der nächste Schritt besteht also darin, das Diagramm mit Elementen (mit *DiagramNode*-Objekten) zu füllen. Bei ersten Experimenten irritiert, dass das *Diagram*-Objekt zwar mit der Eigenschaft *Nodes* auf eine *DiagramNodes*-Aufzählung zeigt, diese Aufzählung aber nicht (wie sonst üblich) über eine *Add*-Methode verfügt.

Die Beispielprogramme in der Hilfe zeigen schließlich den einzig zielführenden (aber vollkommen unlogischen) Weg: Wenn Sie mit *AddDiagram* ein neues Diagramm erzeugen, erhalten Sie ein *Shape*-Objekt zurück (siehe oben): Für dieses Objekt gibt es die Eigenschaft *DiagramNode*, die auf ein Objekt dieses Typs verweist. Offensichtlich wird zusammen mit jedem *Diagram*-Objekt ein unsichtbares und gewissermaßen virtuelles *DiagramNode*-Objekt erzeugt, das als Startpunkt für das Hinzufügen weiterer Elemente dient. (*Diagram.Nodes.Count* liefert allerdings 0.)

Die weitere Vorgehensweise hängt vom Diagrammtyp ab: Bei Organigrammen und Radialdiagrammen muss zuerst genau ein Wurzelobjekt erzeugt werden. Alle weiteren Objekte werden als untergeordnete Objekte (*Children*) zu diesem Objekt hinzugefügt. Die folgenden Zeilen erzeugen ein Radialdiagramm mit einem Kreis in der Mitte (*root*) und drei damit verbundenen Kreisen rundherum (*child1* bis *child3*).

```
' ws verweist auf ein Worksheet-Objekt
' für msoDiagramRadial und msoDiagramOrgChart
Dim s As Shape
Dim root As DiagramNode, child1 As DiagramNode, _
    child2 As DiagramNode, child3 As DiagramNode
Set s = ws.Shapes.AddDiagram(msoDiagramRadial, 10, 10, 200, 100)
Set startnode = s.DiagramNode
Set root = startnode.Children.AddNode
Set child1 = root.Children.AddNode
Set child2 = root.Children.AddNode
Set child3 = root.Children.AddNode
```

Bei den anderen Diagrammtypen befinden sich dagegen alle Diagrammobjekte auf gleicher Ebene. Hier sieht die Vorgehensweise wie im folgenden Beispiel aus, dass ein vierteiliges Pyramidendiagramm erzeugt.

```
' msoDiagramPyramid, msoDiagramCycle, msoDiagramTarget, msoDiagramVenn
Dim s As Shape
Dim child1 As DiagramNode, child2 As DiagramNode, _
    child3 As DiagramNode, child4 As DiagramNode
Set s = ws.Shapes.AddDiagram(msoDiagramPyramid, 10, 10, 200, 100)
Set startnode = s.DiagramNode
Set child1 = startnode.Children.AddNode
Set child2 = child1.AddNode
Set child3 = child1.AddNode
Set child4 = child1.AddNode
```

Diagrammelemente beschriften

Der Text eines Diagrammelements wird über ein *TextFrame*-Objekt verwaltet (siehe den vorigen Abschnitt). Ausgehend von einem *DiagramNode*-Objekt führt die folgende Eigenschaftsliste zur Texteigenschaft: *child1.TextShape.TextFrame.Characters.Text*.

Das große Problem besteht nun darin, dass eine Veränderung von *Text* in Excel 2002 schlicht unmöglich ist. (Auch die Verwendung von der alternativen Eigenschaft *Caption* oder der Methode *Insert* hilft nichts.) Dass der Fehler ein drei viertel Jahr nach dem Erscheinen von Excel 2002 noch nicht einmal dokumentiert ist (geschweige denn behoben), lässt vermuten, dass die neuen *DiagramXxx*-Objekte auf eher geringes Interesse von Seiten der Programmierer gestoßen sind.

Ohne eine Möglichkeit, die Diagrammelemente zu beschriften, ist natürlich jede weitere Programmierung sinnlos. Es bleibt nur die Hoffnung, dass die vielen Ungereimtheiten der *Diagram*-Objekte in künftigen Excel-Versionen korrigiert werden.

Diagramme löschen

Es gibt keine *Delete*-Methode für *Diagram*-Objekte. Stattdessen müssen Sie das zugrunde liegende *Shape*-Objekt löschen. Über die Eigenschaft *HasShape* können Sie testen, ob das *Shape*-Objekt zur Darstellung eines Diagramms oder für andere Zwecke verwendet wird. Die folgende Schleife löscht alle Geschäftsdiagramme im ersten Tabellenblatt der Datei.

```
Dim s As Shape
Dim ws As Worksheet
Set ws = Worksheets(1)
For Each s In ws.Shapes
    If s.HasDiagram Then s.Delete
Next
```