CARL HANSER VERLAG

Dieter Orlamünder, Rüdiger Liskowsky, Heinrich Hußmann

Software-Entwicklung mit Delphi

Eine systematische Einführung

3-446-22088-7

www.hanser.de

allen Fahrzeugen außer Mopeds ist die Angabe einer Führerscheinnummer des Halters und des Hubraums notwendig. Bei PKW ist die Zahl der Sitzplätze festzuhalten, bei LKW die zulässige Zuladung. Über Fahrzeughalter sind Name, Adresse und Alter zu speichern. Es sollen Anfragen zu Fahrzeugen (über Kennzeichen) möglich sein, es soll aber auch möglich sein, über den Halternamen alle zugelassenen Fahrzeuge des Halters zu erfragen.

5.4 Delphi-Komponenten

5.4.1 Einordnung

Software-Komponenten sind ganz allgemein Bausteine für Programme, die dafür vorgesehen sind, dass aus ihnen ohne Veränderung des Quellcodes größere Programme zusammengesetzt werden können. Meist werden Komponenten als compilierte Programme ausgeliefert, sodass der Quellcode dem Komponentenanwender (der selbst wieder Programme baut) nicht zugänglich ist. Es gibt derzeit verschiedene Technologien, die es ermöglichen, Softwarebausteine innerhalb eine vorgegebenen Infrastruktur und Entwicklungsumgebung zusammenzusetzen und auf die konkret gewünschten Zwecke anzupassen. Als Technologien von hoher aktueller Bedeutung auf dem Markt sind hier insbesondere zu nennen: Microsoft COM+ (und neuerdings die .NET-Architektur), Java Beans und Enterprise Java Beans. Eine allgemeine Übersicht zum Thema Softwarekomponenten findet sich unter anderem in [35] und in [15]. Das Programmiersystem Delphi stellt seit längerem ebenfalls ein Komponentenkonzept bereit (und ermöglicht darüber hinaus die Integration von Komponenten der Microsoft-Technologien). Im Folgenden soll das originäre Komponentenkonzept von Delphi kurz vorgestellt werden.

Komponenten sind in Delphi spezielle Objekte, die dadurch ausgezeichnet sind, dass sie bereits zur Entwurfszeit eines Projekts aus der Komponentenpalette in ein Formular eingefügt werden können. Weiterhin besteht die Möglichkeit, für diese Komponenten über den Objektinspektor bestimmte Eigenschaften zu setzen (Seite **Eigenschaften**) und Ereignisse zu nutzen (Seite **Ereignisse**). Beispiele sind die bekannten Komponenten **Button**, **Edit**, **Label** usw., die bereits standardmäßig in Delphi zur Verfügung stehen.

Es kann jedoch sinnvoll sein, weitere Komponenten von Fremdherstellern einzubinden, um die Entwicklung von Programmen eines speziellen Anwendungsgebiets zu erleichtern. Ein Beispiel sind hier etwa Komponenten zur Steuerung von Aktoren und Sensoren in einer Gerätesteuerungsanwendung. Ebenfalls kann es oft sinnvoll sein, eigene Komponenten zu entwickeln, wenn bestimmte Standardfunktionalität immer wieder benötigt wird. Im Folgenden wird beschrieben,

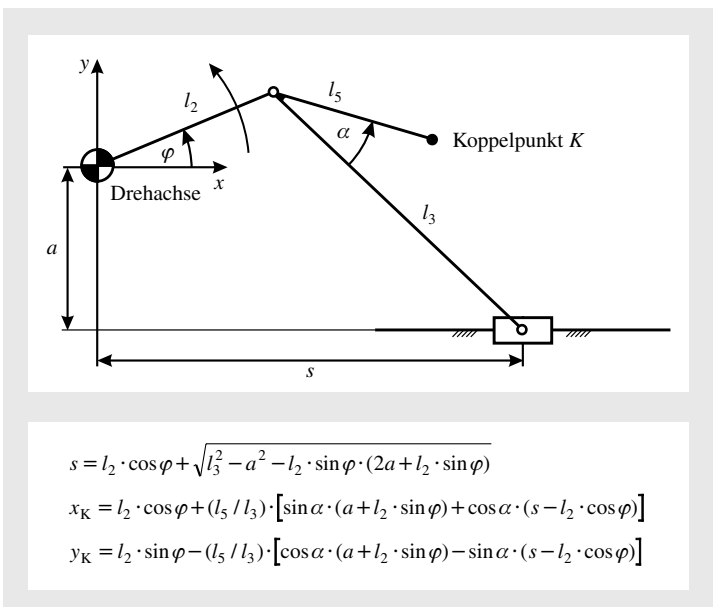
- wie die Installation einer neuen Komponente (ob zugekauft oder selbst geschrieben) in die Delphi-Umgebung geschieht;
- wie Komponenten so zu entwickeln sind, dass sie problemlos in die Komponentenpalette installiert werden können.

Im Folgenden werden zwei spezielle Komponenten GETRIEBE und ERZEUGUNGSDATUM vorgestellt. Die Komponente GETRIEBE dient zur Demonstration des ersten Punkts, also der Installation und Nutzung einer zusätzlichen Komponente. Der Quellcode von GETRIEBE ist zu kompliziert, um ihn hier zu erläutern, für Interessierte ist der Code aber über die Webseite verfügbar. Bei ERZEUGUNGSDATUM ist es möglich, den gesamtem

Quellcode leicht zu überschauen. An dieser Komponente werden Entwicklungsprinzipien, Package-Bildung und die Aufnahme in die Komponentenpalette gezeigt. Schließlich erfolgt die Nutzung beider Komponenten im Projekt SCHUBKURBELGETRIEBE.

5.4.2 Komponente GETRIEBE

Zur Demonstration der prinzipiellen Vorgehensweise bei der Entwicklung soll die Komponente GETRIEBE dienen. Das Getriebe ist zusammen mit den grundlegenden Algorithmen für den Koppelpunkt *K* in Bild 5.13 dargestellt. Es gibt eine Anzahl von Parametern, die das Getriebe beschreiben (l_2, l_3, l_5, a und α). Eine Drehung der Schubkurbel bedeutet ein Verschieben des Schlittens auf der horizontalen Achse (d. h. ein bestimmter Zustand der Drehung kann entweder durch einen Wert für den Winkel φ oder durch einen Wert für s angegeben werden). Die erste Formel beschreibt den Zusammenhang zwischen φ und s bei gegebenen Getriebeparametern. Bei einer solchen Drehung beschreibt der Koppelpunkt *K* eine Kurve auf der Fläche (Koppelkurve). Die beiden weiteren Formeln beschreiben die *x*- und *y*-Koordinate von *K* in Abhängigkeit von den Getriebeparametern und dem aktuellen Drehungszustand. Die Komponente GETRIEBE ist in der Lage, diese Kurve auf der Basis der drei Formeln grafisch anzuzeigen.



$$s = l_2 \cdot \cos\varphi + \sqrt{l_3^2 - a^2 - l_2 \cdot \sin\varphi \cdot (2a + l_2 \cdot \sin\varphi)}$$

$$x_K = l_2 \cdot \cos\varphi + (l_5 / l_3) \cdot \left[\sin\alpha \cdot (a + l_2 \cdot \sin\varphi) + \cos\alpha \cdot (s - l_2 \cdot \cos\varphi)\right]$$

$$y_K = l_2 \cdot \sin\varphi - (l_5 / l_3) \cdot \left[\cos\alpha \cdot (a + l_2 \cdot \sin\varphi) - \sin\alpha \cdot (s - l_2 \cdot \cos\varphi)\right]$$

Bild 5.13 Getriebe und Algorithmen

Die Getriebekomponente erfüllt im Detail folgende Aufgaben:

- Im Objektinspektor sind die geometrischen Daten des Schubkurbelgetriebes als Eigenschaften bereits zur Entwurfszeit einstellbar.
- Die Komponente enthält Methoden zur Darstellung, zum Lauf, zum Anhalten des Schubkurbelgetriebes und zur Positionsänderung des Koppelpunktes *K*, die dann in dem Projekt, in das diese Komponente eingefügt wurde, genutzt werden können.

Im Folgenden interessieren wir uns nicht weiter für den Quelltext, der diese Formeln in eine Visualisierung umsetzt (der Programmcode ist auf der Webseite verfügbar), sondern betrachten die Installation der fertigen Komponente.

Seit der Version 3 des Delphi-Systems gibt es so genannte **Pakete** (*packages*), die unter anderem dazu verwendet werden können, Komponenten in kompakter Weise zu verpacken und weiterzugeben. Seit Delphi 4 wird zusätzlich zwischen **Entwurfszeitpackages** und **Laufzeitpackages** unterschieden, wobei erstere bei der Enzwicklung zum Einsatz kommen und letztere dynamisch beim Ablauf des Programms nachgeladen werden. Für unsere Zwecke interessieren wir uns für ein Entwurfszeitpackage, das (möglicherweise zusammen mit vielen anderen Komponenten) die Komponente GETRIEBE enthält. Ein Package tritt als Datei mit der Endung .bpl in Erscheinung. Auf der Webseite zu diesem Buch steht eine solche Package-Datei mit dem Namen **GetriebePk.bpl** zur Verfügung.

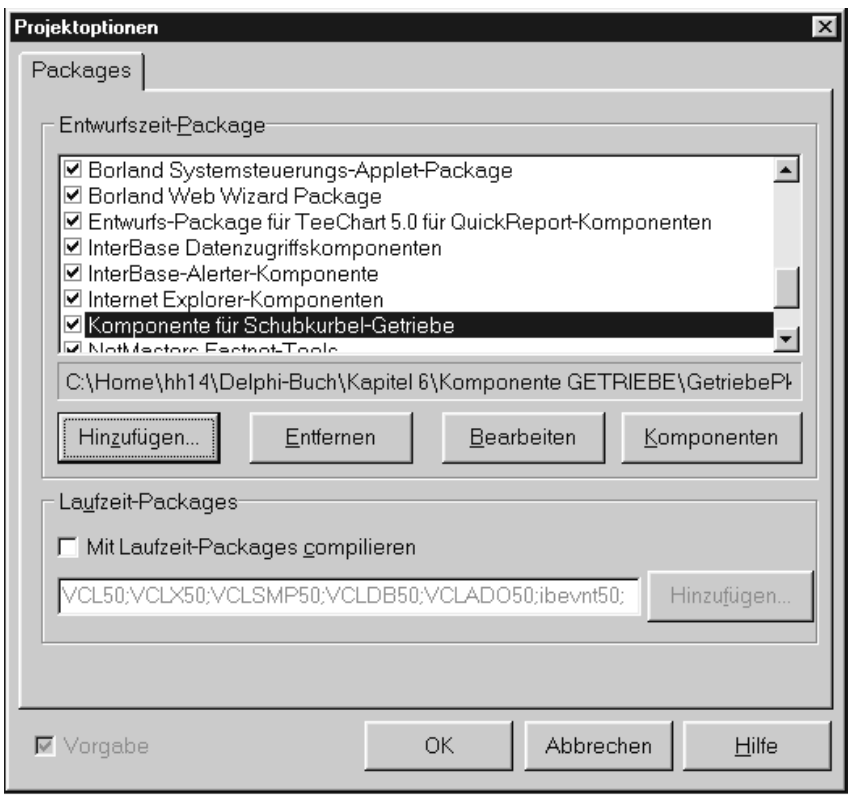


Bild 5.14 Dialog zur Installation einer Komponente

Die Komponente soll nun in die Komponentenpalette eingebunden werden. Jede Komponente enthält die Information, wo sie in der Komponentenpalette erscheinen soll. Standardmäßig erscheinen neue benutzerdefinierte Komponenten unter dem Tabulator **Beispiele** in der Komponentenpalette. Um die neue Komponente einzubinden, verwendet man die Funktion **Packages installieren** aus dem Menü **KOMPONENTE**. Bild 5.14 zeigt das

entsprechende Dialogfenster. In diesem Fenster wird mit **Hinzufügen** die bpl-Datei, in unserem Beispiel **GetriebePK.bpl**, ausdewählt, worauf die Liste mit der neuen Komponente ergänzt wird, wie in Bild 5.14 gezeigt.

Mit dem Klicken von „OK“ ist die Installation bereits abgeschlossen. Die Komponentenpalette enthält nun unter dem Tabulator **Beispiele** ein neues Element (ganz rechts), das der Getriebekomponente entspricht, wie in Bild 5.15 gezeigt.

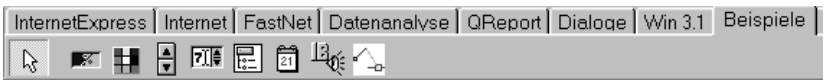


Bild 5.15 Komponentenpalette mit Komponente GETRIEBE

Mit der Installation in der Palette kann die Komponente in gleicher Weise wie Buttons, Menüs etc. auf einem Formular platziert werden. Die folgende Bild 5.16 zeigt den Objektinspektor mit seinen beiden Seiten **Eigenschaften** und **Ereignisse** für eine aktuelle Komponente vom Typ **TGetriebe**.

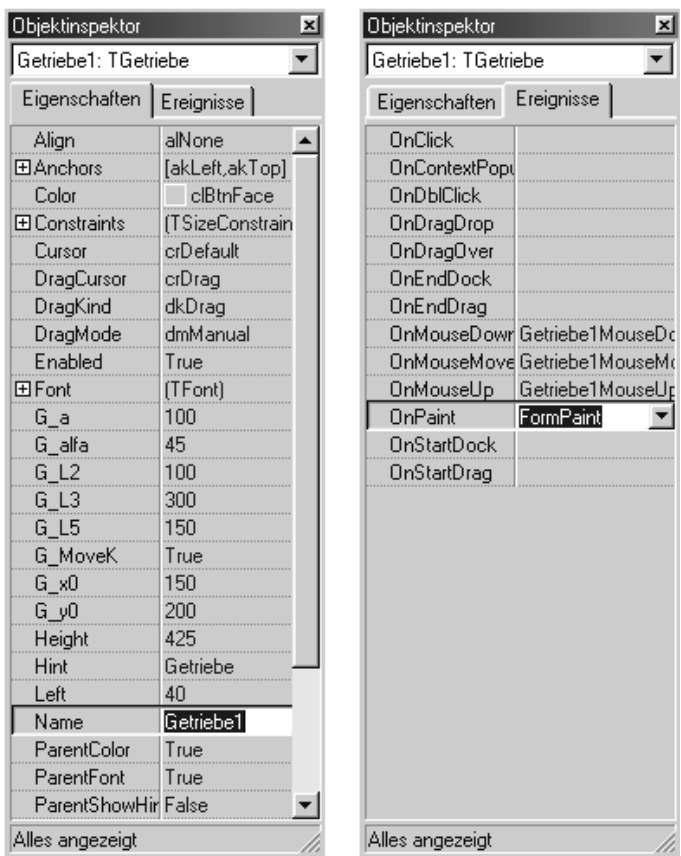


Bild 5.16 Objektinspektor für eine aktuelle Komponente vom Typ **TGetriebe**

5.4.3 Komponente ERZEUGUNGSDATUM

Möchte man selbst eine eigene Komponente mit Delphi entwickeln, die wie oben gezeigt installiert und verwendet werden kann, dann ist im Wesentlichen eine Klasse mit speziellen Eigenschaften zu schreiben. Im Folgenden wird dies an einer sehr einfachen Komponente demonstriert, die nur das Erzeugungsdatum (Übersetzungszeit) des Programms ausgibt. Darüber hinaus soll die Komponente zur Entwurfszeit (mit dem Objektinspektor) so konfigurierbar sein, dass man einen beliebigen Text als Vorspann zu dieser Datumsinformation definieren kann. Eine solche Komponente kann an manchen Stellen sinnvoll sein, z. B. in einem Informationsdialog über ein Programm.

Um eine Klasse als Komponente einsetzen zu können, müssen bei deren Entwicklung die folgenden Bedingungen erfüllt werden:

1. Die Komponente muss direkt oder indirekt von der Klasse **TComponent** abgeleitet werden.
2. Die entwickelte Komponente muss zur Aufnahme in die Komponentenpalette durch Delphi registriert werden. Dazu existiert die Prozedur **Register**.
3. Alle Eigenschaften und Ereignisse, die im Objektinspektor sichtbar sein sollen, müssen das Zugriffsrecht ***published*** besitzen, sofern nicht schon ihre Vorfahren dieses Zugriffsrecht haben.

Die Entwicklung und Installation eigener Komponenten erfolgt prinzipiell in folgenden Schritten:

1. Programmierung einer Unit in Object Pascal, die die Komponentenklasse und die Registrierungs-Prozedur enthalten muss. Das Delphi-System stellt zur Unterstützung dieses Schritts einen so genannten **Komponentenexperten** bereit, der ein Rahmenprogramm für diese Unit erstellt, das dann noch weiter auszufüllen ist.
2. Erstellen einer Bilddatei, die die Ikone der neuen Komponente definiert. Es handelt sich hier um eine Ressourcendatei (*.dcr*) in Form einer Bitmap. Das Delphi-System bietet hierzu einen Bildeditor.
3. Erzeugen eines (Entwurfszeit-)Pakets und übersetzen in die installierbare Form. Hierbei wird man durch das Delphi-Werkzeug **Package Editor** unterstützt.
4. Schließlich Installation in der Komponentenpalette wie oben bereits gezeigt.

Schritte 3 und 4 können noch stärker automatisiert werden, indem man die Funktion **Komponente installieren** aus dem Menü **Komponente** verwendet. Der Übersichtlichkeit halber werden die Schritte im Folgenden aber getrennt gezeigt.

Schritt 1: Programmierung der Komponente

Für die Entwicklung einer Komponente ist es zweckmäßig, den Komponenten-Experten zu benutzen, der über **Neue Komponente** im Menü **Komponente** oder über die allgemeine **Neu**-Funktion im Menü **Datei** aufgerufen wird, siehe Bild 5.17. Für das Beispiel wurden der Klassenname der Komponente mit **TErzeugungsdatum** gewählt und als Vorfahr die Klasse **TCustomLabel**, die sich besonders zur Darstellung von Texten auf einem Formular eignet. Diese Auswahl sollte wohlbedacht werden, da die neue Komponente von ihrem Vorfahr eine größere Anzahl an sinnvollen Eigenschaften erbt, in unserem Beispiel etwa zur Größenveränderung und zur Einstellung des gewünschten Zeichensatzes. Für die Komponente GETRIEBE von oben wurde als Vorfahr **TPaintBox** gewählt, da diese einen

eigenen Canvas, ein eigenes Koordinatensystem und Mechanismen zur Positionierung und Größeneinstellung besitzt. Der Name der Unit bestimmt den Dateinamen (*.pas*-Datei); hier kann auch das Verzeichnis eingestellt werden, wo die Datei abgelegt wird. Mit der voreingestellten und hier nicht geänderten Angabe zur Palettenseite (Beispiele) wird bestimmt, wo sich die Komponente nach ihrer Installation befindet.

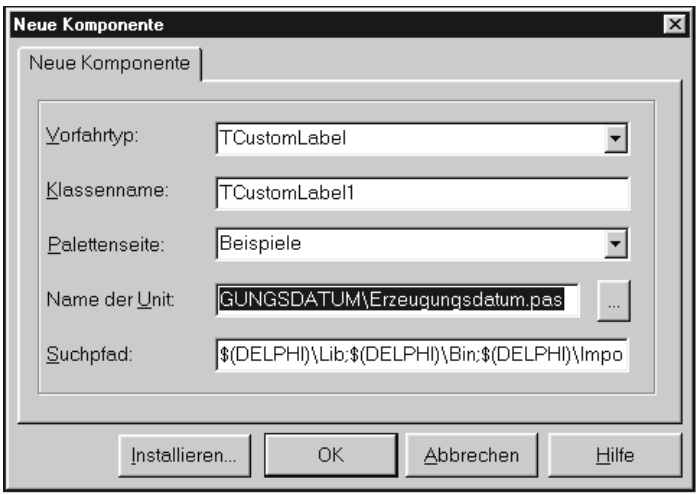


Bild 5.17 Komponenten-Experte

Nach Bestätigung mit „OK“ wird das folgende Gerüst für die Komponenten-Unit erzeugt, welches dann entsprechend den inhaltlichen Forderungen an die Unit auszufüllen ist.

Beispiel:

```
unit Erzeugungsdatum;
interface
uses
   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
   Dialogs, StdCtrls;

type
   TErzeugungsdatum = class(TCustomLabel)
   private
      { Private-Deklarationen }
   protected
      { Protected-Deklarationen }
   public
      { Public-Deklarationen }
   published
      { Published-Deklarationen }
   end;

procedure Register;

implementation

procedure Register;
begin
```

```
    RegisterComponents('Beispiele', [TErzeugungsdatum]);
end;

end.
```

■

Besonders zu beachten ist hier die Prozedur **Register**, die letztlich für die Installation der Komponente in der Komponentenpalette sorgt.

Als Hilfsmittel bei der Vervollständigung des Programmtexts dienen die folgenden Systemfunktionen (nähere Informationen dazu finden sich in der Online-Hilfe von Delphi):

- **ParamStr(0)** zur Ermittlung von Pfad- und Dateinamen des ausgeführten Programms
- **FileAge(...)** für Datums- und Zeitinformationen zur angegebenen Datei
- **FileDateToDateTime(...)** und **FormatDateTime(...)** zur Konvertierung von Datums- und Zeitwerten.

Das Kernstück des Programms ist natürlich der Aufruf dieser Systemfunktionen. Hierzu deklarieren wir im privaten Abschnitt eine Methode (**SetLabelText**) und eine **String**-Variable, die den auszugebenden Vorspann vor dem Erzeugungsdatum aufnehmen soll.

```
private
    { Private-Deklarationen}
    Prefix: string;
    procedure SetLabelText;
```

Im Implementierungsteil ruft nun die Prozedure **SetLabelText** die Systemfunktionen auf:

```
procedure TErzeugungsdatum.SetLabelText;
    var Datum: TDateTime;
begin
    Datum:=FileDateToDateTime(FileAge(ParamStr(0)));
    Caption:=Prefix+FormatDateTime('dd.mm.yyyy',Datum);
end {TErzeugungsdatum.SetLabelText};
```

Wann aber wird diese Prozedur aufgerufen? Selbstverständlich bereits beim Erzeugen der Komponente. Also deklarieren wir einen öffentlichen Konstruktor, der SetLabelText aufruft. Außerdem sollte er die Initialisierung der Vorfahr-Klasse unverändert durchführen.

```
constructor TErzeugungsdatum.Create(AOwner: TComponent);
begin
    inherited Create(AOwner);
    SetLabelText;
end;
```

Um den Vorspanntext im Objektinspektor einstellbar zu machen, deklarieren wir eine veröffentlichte (*published*) Eigenschaft **PrefixText**.

```
published
    { Published-Deklarationen }
    property PrefixText: string read Prefix
                                 write SetPrefixText;
```

Hier kommen nun die Mechanismen der Eigenschaften (siehe Abschnitt 5.3.4) zum Einsatz. Beim Lesen der Eigenschaft wird einfach der Text der lokalen (privaten) Va-

riable **Prefix** übernommen. Aber beim Schreiben, d. h. der Veränderung des Textes im Objektinspektor (oder auch zur Laufzeit durch andere Objekte), muss unbedingt der angezeigte Text angepasst werden. Deshalb wird in diesem Fall eine Methode **SetPrefixText** aufgerufen, die dies übernimmt und gleichzeitig den Wert von **Prefix** ändert.

```
procedure TErzeugungsdatum.SetPrefixText(Text: string);
begin
   Prefix:=Text;
   SetLabelText;
end {TErzeugungsdatum.SetPrefixText};
```

Der vollständige Quelltext der Komponenten-Unit ist wie immer auf der Webseite zu finden.

Schritt 2: Erstellen einer Bilddatei für die Ikone

Soll die neu entwickelte Komponente durch eine eigene Ikone symbolisiert werden, so ist ein Bild mithilfe des **Bildeditors** zu erstellen. Dieser ist über das Menü **Tools | Bildeditor** erreichbar. Die Ikone wird aus 24×24 Pixeln generiert. Bild 5.18 zeigt den Bildeditor und die Ikonen für beide hier behandelten Komponenten.

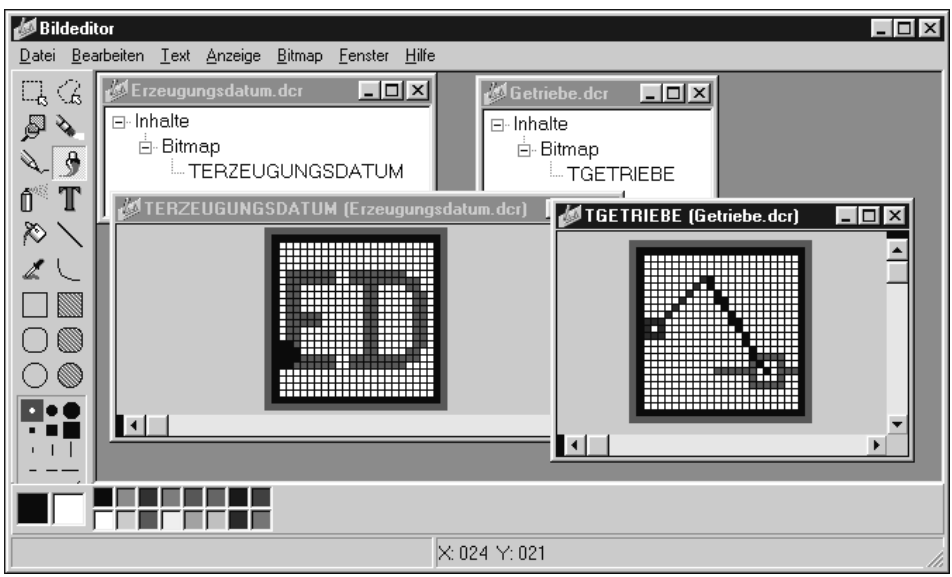


Bild 5.18 Bildeditor mit Ikonen

Im Bildeditor muss die Ikone entsprechend der zugehörigen Komponentenklasse benannt werden. Damit es im Rahmen der anschließenden Installation in der Komponentenpalette sichtbar wird, muss die Ressourcen-Datei im gleichen Verzeichnis wie die Komponente unter deren Namen mit der Erweiterung *.dcr* gespeichert werden. Bei falscher Verfahrensweise oder wenn keine eigene Ikone erzeugt wurde, wird automatisch die der Vorfahrklasse verwendet, bei der Komponente GETRIEBE also die **TPaintBox**-Ikone und bei ERZEUGUNGSDATUM die von **TCustomLabelText**.

Schritt 3: Erzeugen eines Pakets

Um eine Komponente in ein installierbares Paket zu überführen, benutzt man am besten den **Package Editor** von Delphi. Dieser Editor hilft vor allem bei der Erstellung einer Beschreibungsdatei (*.dpk*), die alle zum Paket gehörigen Bestandteile zusammenfasst. Außerdem wird, so noch nicht geschehen, der Compiler aufgerufen, um das Programm der Komponente zu übersetzen. Selbstverständlich kann dieser Schritt erst abgeschlossen werden, wenn keine Übersetzungsfehler auftreten.

Der **Package Editor** ist (unter anderem) erreichbar über das Menü **Neu** mit anschließender Auswahl von **Package**. Bild 5.19 zeigt das Dialogfenster, nachdem die Komponente ERZEUGUNGSDATUM (genauer ihre Quelltext-Datei) mit **Hinzufügen** ausgewählt wurde. Es werden zwei Dateien angezeigt; die zweite ist die Bitmap-Datei für die Ikone. Mit **Speichern unter** wird der Name für das Paket gewählt, hier **ErzDatum**. Dieser Name muss verschieden von denen aller enthaltenen Komponenten sein!

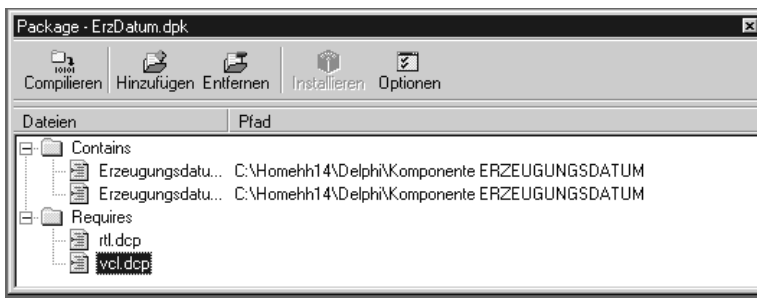


Bild 5.19
Package Editor

Hinweis:

1. Die Bild 5.19 ist mit Delphi-Version 6 erstellt, bei älteren Versionen erscheinen andere Namen für die benutzten Pakete (bei „Requires“).
2. Das erzeugte Paket (in diesem Fall eine Datei **ErzDatum.bpl**) findet sich an einer vom System festgelegten Stelle wieder. Dies wird meist ein Verzeichnis „Projects/BPL“ im Delphi-Verzeichnis sein. Dieser Ort kann über das Menü **Tools** | **Umgebungsoptionen** | **Bibliothek** eingestellt werden, aber auch über den Dialog der Projektoptionen für das einzelne Paket verändert werden.

Schritt 4: Installation der Komponente

Dieser Schritt kann genau analog zu dem in Abschnitt 5.4.2 beschriebenen Verfahren erfolgen. Der **Package Editor** bietet jedoch auch eine Kurzfassung für diesen Schritt durch einfaches Klicken auf **Installieren**.

Nutzung der Komponente

Nach der Installation kann die Komponente ERZEUGUNGSDATUM in einem beliebigen Anwendungsprogramm eingesetzt werden, indem sie über die Palette ausgewählt, auf dem Formular platziert und mit dem Objektinspektor konfiguriert wurde. Bild 5.20 zeigt eine solche Instanz der Komponente, nachdem sie auf das Formular platziert wurde, der Zeichensatz vergrößert und als Prefix-Wert der Text „Erzeugt am“ eingegeben wurde. Interessant zu sehen ist, dass die Komponente sofort bei ihrer Platzierung den Konstruktor ausführt und deshalb ein Erzeugungsdatum anzeigt – und zwar das Datum, zu dem das

aufrufende Programm compiliert wurde, also das Delphi-Entwicklungssystem! Sobald das Programm übersetzt und gestartet wird, zeigt die Komponente zur Laufzeit das Übersetzungsdatum des Anwendungsprogramms an.

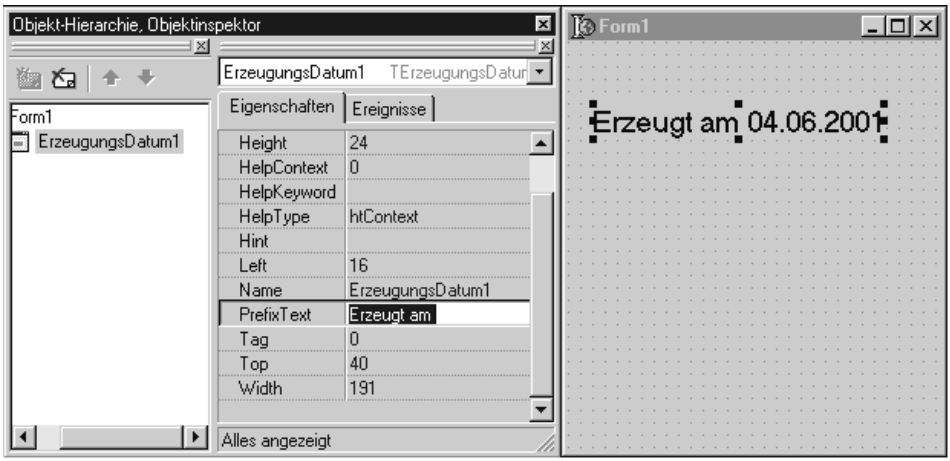


Bild 5.20 Nutzung der Komponente ERZEUGUNGSDATUM

Die Komponente ERZEUGUNGSDATUM wurde auch im Projekt SCHUBKURBELGETRIEBE genutzt, siehe den nächsten Abschnitt.

5.4.4 Projekt SCHUBKURBELGETRIEBE

Das nachstehende Projekt ist ein Beispiel für die Anwendung vieler in diesem Kapitel gezeigten Programmiertechniken. Es sind beide oben diskutierten Komponenten eingebunden und es werden Techniken der interaktiven Grafikprogrammierung verwendet. Das

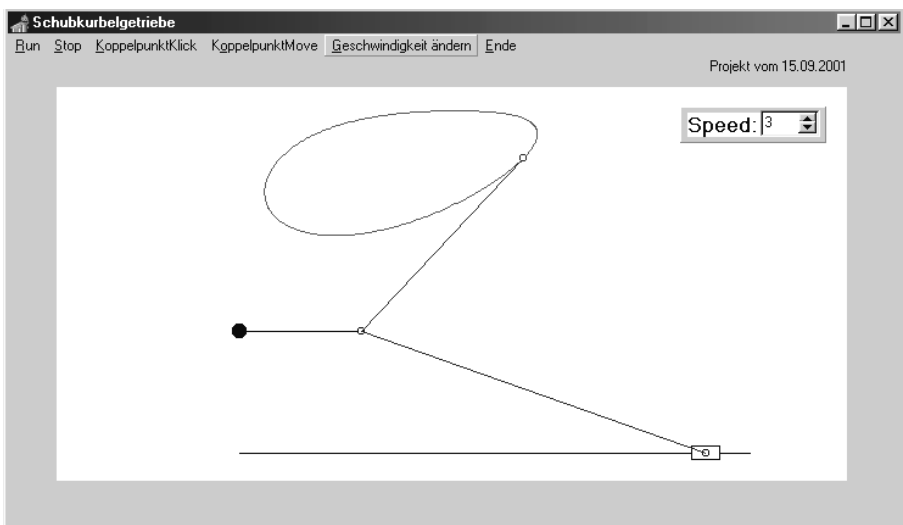


Bild 5.21 Oberfläche des Projekts Schubkurbelgetriebe.dpr

Programm erlaubt es, die Simulation des Getriebes interaktiv zu starten und anzuhalten, die Geschwindigkeit der Simulation einzustellen und einige der Parameter interaktiv zu verändern. Die Lage des Koppelpunktes K kann durch Mausklick oder Ziehen mit der Maus neu festgelegt werden. Damit werden indirekt die Parameter l_5 und α aus Bild 5.13 neu bestimmt; die anderen Parameter sind in diesem Programm fest vorgegeben (wären aber leicht durch Modifikation der Eigenschaften der Getriebekomponente und Neuübersetzung anzupassen). Bild 5.21 zeigt ein Ausgabefenster der Anwendung.

5.4.5 Übungen

5.8 Ü-Projekt SCHUBKURBELGETRIEBE2

Modifizieren Sie das Projekt SCHUBKURBELGETRIEBE so, dass es möglich wird, die restlichen Parameter des Getriebes interaktiv einzustellen, z. B. durch ein Dialogfenster. ***Hinweis:*** Hierzu sind keinerlei Kenntnisse der Komponentenentwicklung notwendig! Die verwendeten Komponenten sollen unverändert bleiben.

5.9 Ü-Projekt WECKER

Erstellen Sie eine Delphi-Komponente für eine digitale Weckuhr, bei der über den Objektkonfigurator eingestellt werden kann

- ob Sekunden-, Minuten- und Stundenfelder angezeigt werden
- welche Trennzeichen für die Felder verwendet werden (z. B. „ : “)
- mit welcher Häufigkeit die Uhranzeige aktualisiert werden soll
- welcher Zeitpunkt eine Aktion auslösen soll und was diese Aktion ist

Die Verwendung der Komponente ist in einem Beispielprojekt zu demonstrieren.