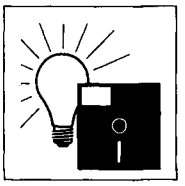


Visuelle Entwicklungsumgebungen und KnowledgePro Windows (Teil 1)

"To use a computer must always be easier than to not use a computer."
Ted Nelson

Peter Ebenhoch

Gliederung Teil 1

Gliederung Teil 2 (im nächsten Heft)

1 Einleitung

Der kommerzielle Durchbruch grafischer Benutzeroberflächen, den Microsoft Windows 3.0 1990 eingeläutet hat, führte auch im Bereich der Intel-basierten[1] Personal Computer zur Entwicklung von neuartigen, einfachen Programmierwerkzeugen, die treffend als visuelle Entwicklungsumgebungen bezeichnet werden können[2]. Diese "Visual Development Environments" (VDE) erleichtern die Entwicklung von GUI[3]-basierten-Applikationen ungemein und sollen auch Nichtprogrammierern die Erstellung von Windows-Anwendungen ermöglichen.

Eingängige Konzepte und Metaphern

Dem Benützer wird durch eingängige Konzepte und Metaphern der Zugang zur Erstellung von Applikationen erleichtert bzw. überhaupt erst ermöglicht, dafür müssen aber auch – verglichen mit traditionellen low-level Sprachen wie Pascal oder C/C++ – entsprechende Einschränkungen der Programmierfreiheit und Abstriche bezüglich der Qualität der Ergebnisse[4] in Kauf genommen werden. Nach Toolbook[5] [6] wird im folgenden Beitrag KnowledgePro Windows (KPWin)[7] von KnowledgeGarden vorgestellt.

2 Überblick: Visuelle Entwicklungsumgebungen

Nach Petzold[2] kombiniert ein VDE eine integrierte Entwicklungsumgebung (IDE), die Borland 1983 erstmals mit Turbo Pascal einführte, mit zusätzlichen Möglichkeiten vereinfachter, interaktiver Interfacegestaltung und ermöglicht dadurch die Windows-Programmierung ohne das Windows-API[8] mit seinen insgesamt ca. 1000 Funktionen[9] bemü-

Peter Ebenhoch. Studium der Rechtswissenschaft in Wien. Gegenwärtig Arbeit auf den Gebieten Rechtstheorie und der Rechtsinformatik.

1 Vorreiter für den Apple MacIntosh: Hypercard. Dazu: Brian L. Dear, HyperCard. What Is It?, BYTE 8/1988, Macintosh Supplement, S. 71–74

2 Charles Petzold, Visual Development Environments, PC Magazine, 11/1992, S. 195–217

3 GUI: graphical user interface, graphische Benutzeroberfläche

4 Die erstellten Anwendungen sind oft langsam und ressourcenfressend.

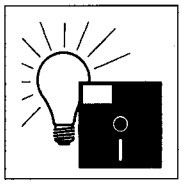
5 Dieser letzte Punkt hat unter anderem durch die technische Entwicklung ("Multimedia") in letzter Zeit an Bedeutung gewonnen: Manche Autorensysteme wie MacroMind Director oder MediaBlitz (siehe Fußnote 11) bieten ganze Misch- und Schneidepulte zur Audio- und Videomanipulation an. vgl Steinbrink, c't 10/1993, S. 174

6 Clemens Tobias Steins, Vor- und Nachteile von ToolBook – Ein Erfahrungsbericht, jur-pc 8/93, S. 2246

7 Der Beitrag beschäftigt sich mit KnowledgePro Windows Gold, Version 2.3; KPWin Gold ist bei BoldBuyer$, Schwertstraße 1–2, 71065 Sindelfingen (Tel.: 07031-811095), erhältlich und kostet DM 939,– (für den Hochschulbereich DM 728,–); in der Schweiz und Österreich bei: MIK, Ekkehardstraße 8, CH-8006 Zürich.

8 API: Application Programming Interface, Programmierschnittstelle von Windows

9 Charles Petzold, Programmierung unter Windows 3.1, Microsoft Press, S. 27



ToolBook, Compel, HyperCard, ObjectVison, IconAuthor

hen zu müssen. Entscheidend für den Charakter und damit für das Einsatzgebiet eines VDE's ist deshalb, welche Funktionen das VDE (gewissermassen als Subset des API) zur Verfügung stellt.
So ermöglichen ToolBook, Compel, HyperCard, ObjectVison[10] oder IconAuthor[11] entsprechend der verwendeten Metapher (Buch, Dia, Karte, Formular, Ikon) das visuelle Gestalten von ganzen Anwendungen, die bei einigen Systemen wie HyperCard und ToolBook anschliessend mit einer erweiterten Makrosprache (HyperTalk, OpenScript) verfeinert werden können bzw. verfeinert werden müssen[12].

Visual Basic

Bei Visual Basic hingegen kann zunächst die Oberfläche visuell gestaltet werden; die Funktionalität wird den Bildschirmelementen ("Forms und Controls") anschliessend durch die Auswahl aus vorgegebenen Optionen ("Properties") und mittels herkömmlichem, geschriebenen Code beigebracht.
Von der anderen Seite her betrachtet, stellen sich diese Erleichterungen aber immer wieder als Einschränkungen dar: Bei Visual Basic ist es nicht vorgesehen und auch nicht möglich, auch die Oberfläche direkt in Codeform zu erstellen oder zu bearbeiten; der Programmierer bleibt somit auf die – natürlich äußerst üppige! – Auswahl an vorgegebenen "Objekten"[13] und deren mögliche Einstellungen angewiesen, kann selber aber keine neuen schaffen. Object Vision verlangt die "Programmierung" in der Form von Entscheidungsbäumen, das Schreiben von Programmanweisungen ist überhaupt ausgeschlossen. Bei ToolBook kann sich ua. die der verwendeten Buchmetapher entspringende Seitenstruktur zum Hemmnis wandeln[14].

3 KnowledgePro Windows

No "fatware"

KnowledgePro Windows 2.3 präsentiert sich entgegen dem Trend zur "Fatware"[15] sehr schlank: Es wird auf einer einzigen High-Density-Diskette ausgeliefert und belegt voll installiert weniger als drei Megabyte auf der Festplatte.
Das KPWin-Hauptfenster stellt sich zunächst als Windows typischer ASCII-Editor mit Multi-Document-Interface[16], Search-/Replace-Funktion und Clipboard-Integration dar. Der erstellte Code wird mit der Extension "kb" (steht für Knowledge-Base) als einfacher ASCII-Text abgespeichert.

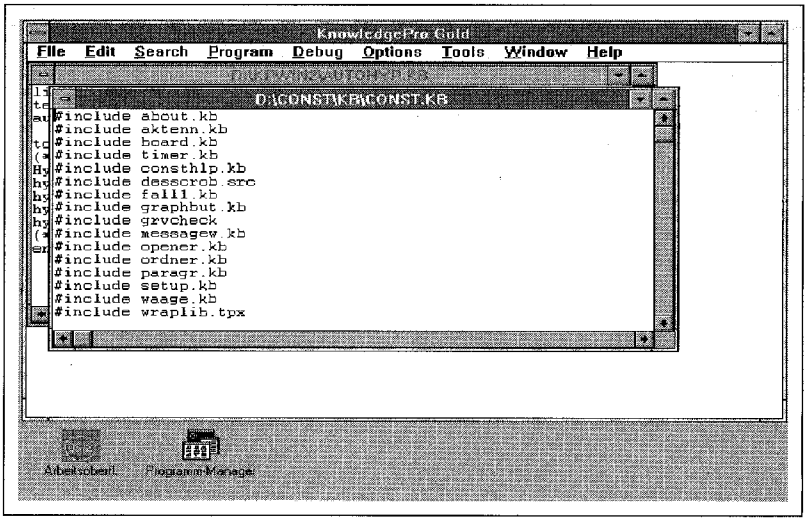


Abb. 1: Das KPWin Arbeitsfenster als einfacher ASCII-Editor.

---

10 dazu: Matthias Kraft, ObjectVision: Ein Formular mit dem man rechnen kann ..., jur-pc 12/92, S. 1880-1882

11 Zum neu erschienen Compel und zu IconAuthor im Vergleich zu weiteren Multimedia Autorensystemen: Bernd Steinbrink, Multimedia Regisseure, Autorensysteme und -sprachen im Vergleich, c't 10/1993, S. 168-179

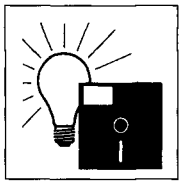
12 Steins, a. a. O., S. 2246

13 Da diese Objekte vorgegeben sind und keine Modifikation durch Vererbung der Eigenschaften möglich ist, handelt es sich hier nicht um Objekte im engeren Sinn der objektorientierten Programmierung; siehe dazu 3.2.6.

14 Zu dieser und zu weiteren Einschränkungen: Steins, a. a. O., S. 2247 und 2248; Ebeling, a. a. O. S. 1849

15 "Fighting Fatware", BYTE, April 1993, S. 98-108,

16 Multi-Document-Interface (MDI): Das Hauptfenster einer Anwendung enthält mehrere untergeordnete Fenster, die simultanes Arbeiten ermöglichen.



Ein solches Programm kann direkt von dieser Entwicklungsumgebung aus mit "Go" ausgeführt werden. Zur Fehlersuche stehen zahlreiche Debugging-Werkzeuge bereit (siehe 3.2.7). Nach Fertigstellung kann das Programm "kompiliert" werden, wobei aber keine selbständig ausführbare Exe-Datei ensteht[17], sondern lediglich der Code für die lizenzfreie Weitergabe als "Compiled Knowledge Base" (*.ckb) mit dem beiliegenden Runtime-Modul vorbereitet und – da die Syntaxprüfung entfallen kann – das Laden beschleunigt wird. Von dieser Arbeitsumgebung aus können nun die integrierten visuellen Werkzeuge gestartet werden.

3.1 Visuelle Werkzeuge – der KPWin Designer

Das wichtigste visuelle Tool ist der KPWin-Designer, der die visuelle Gestaltung der Programmoberfläche ermöglicht. Der KPWin-Designer ermöglicht das Zeichnen der Oberfläche, Anordnen der Elemente wie Buttons, Radio-Buttons, Popup-Fenster etc., sowie das Integrieren von Icöns und Bitmaps inklusive dem Definieren von Hyper-Regionen (siehe 3.2.5.2).

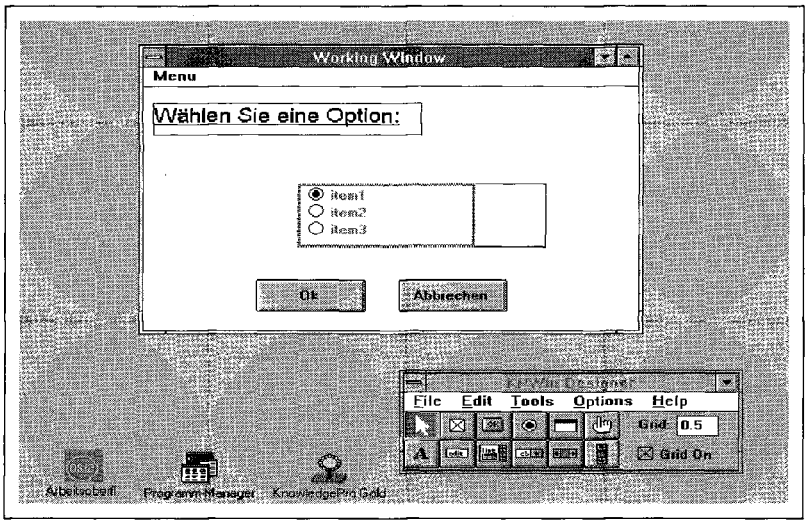


Abb. 2: Gestalten der Programmoberfläche mit dem KPWin Designer.

Im Unterschied zu anderen VDE's erzeugt der KPWin-Designer aber ebenfalls Programm-Code, der wahlweise über die Zwischenablage oder als Datei mit dem Editor weiter bearbeitet und abschließend entweder direkt in das Hauptprogramm übernommen oder in der externen Datei belassen und mit der Interpreter Anweisung `include` eingebunden werden kann. Der KPWin-Designer ermöglicht aber auch das Zurückladen des manuell editierten Codes (zur Überprüfung oder weiteren Bearbeitung).
Die Funktionalität der Oberfläche kann mit dem KPWin-Designer durch die Zuordnung von Topics (siehe unten 3.2.2) vorbereitet werden, die Ausarbeitung erfordert aber jedenfalls die Verwendung der KPWin-Programmiersprache, die trotz ihrer Einfachheit äußerst mächtig ist[18].
Weitere visuelle Tools erlauben die Auswahl von Fonts und Farben (Font-Designer, Color-Designer) und liefern einen Überblick über vorhandene Topics (Library-Browser).

3.2 KnowledgePro als Programmiersprache

Das Kernstück von KnowledgePro ist aber ohne Zweifel die Programmiersprache: Vorteile dieser Sprachorientierung sind die sehr kompakten Endergebnisse[19] [20] und vor allem, wie zu zeigen ist, die Sprache selber.
Diese ist typenlos[21]. Nicht nur deshalb fallen Parallelen zu anderen Sprachen der künstlichen Intelligenz ins Auge: *KI-Parallelen*

---

17 Eine solche Exe-Datei läßt sich mit dem zusätzlich angebotenen Tool KPWin++ erstellen: Aus der KB-Datei wird dabei direkt C++-Quellcode generiert, der sich anschließend mit einem C++-Compiler in eine Exe-Datei kompilieren läßt (siehe dazu unten 5.2).

18 Dies zeigt nicht zuletzt der Designer selber: er ist nämlich ein in die Arbeitsumgebung integriertes KPWin-Programm, dessen Quellcode (wie auch der der anderen Werkzeuge) ebenfalls mitgeliefert wird.

19 Ein umfangreicher Prototyp einer interaktiven Lernumgebung für das österreichische Verfassungsrecht ("Constitutor") umfaßt zB. 100 Kilobyte; dazu kommt allerdings noch das KnowledgePro-Runtime-Modul mit ca. 650 Kilobyte.

20 vgl. den einprägsamen Vergleich von Ebeling, a. a. O., S. 1853

21 genauer betrachtet weist sie einen vordefinierten Datentyp auf: die Liste; siehe 3.2.3.

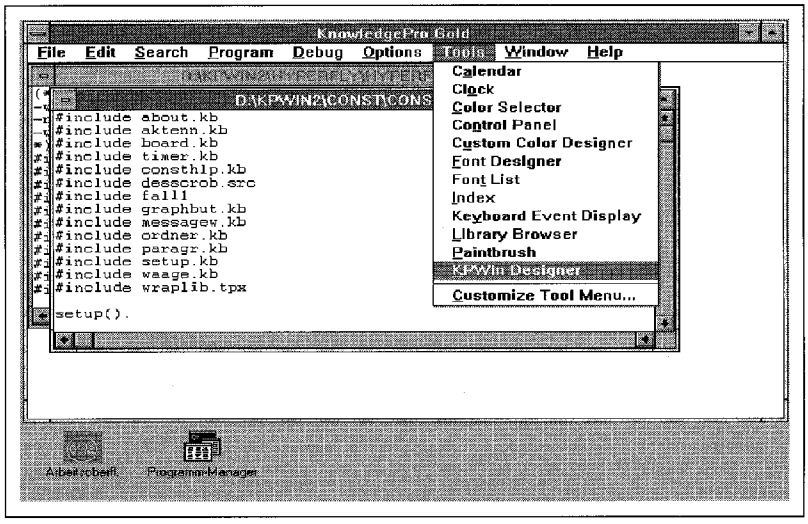


Abb. 3: Die KPWin-Arbeitsumgebung mit dem Menü der verfügbaren visuellen Tools.

Die (möglichen) verschachtelten Klammerstrukturen und die zahlreichen Befehle zur Listenverarbeitung erinnern ebenso an LISP[22] wie die Möglichkeit des Backward-Chaining an PROLOG[23] und die objektorientierten Eigenschaften an SMALLTALK; die Speicherverwaltung erfolgt automatisch (garbage collection).

3.2.1 einfach, aber nicht primitiv

Auf die Komplexität und Schwerfälligkeit obiger KI-Sprachen haben die Entwickler *Bev* und *Bill Thompson* hingegen verzichtet. Die Befehle sind (Grundkenntnisse der englischen Sprache vorausgesetzt) sehr einfach und leicht lesbar:
`say` und `text` bewirken Textausgaben, `ask` ermittelt eine Benutzereingabe, `where` sucht die Position eines Listenelements, `wait` stoppt den Progammablauf, usw.

Befehlsaufbau

Jeder Befehl besteht aus diesem Schlüsselwort und den dazugehörigen Parametern (in runden Klammern) und wird durch einen Punkt abgeschlossen. Werden keine Parameter angegeben, verwendet KPWin umfangreiche Voreinstellungen (Defaults), bei grafischen Objekten wie Buttons oder Fenster sowohl für deren Größe als auch für die Reaktionen auf Nachrichten (Events), die ja für Windows kennzeichnend sind[24]. Dank dieser Defaults kommen viele Befehle ohne oder nur mit wenigen Parametern aus, wodurch die Programmerstellung beschleunigt und das Ausprobieren und Erlernen der Sprache erleichtert wird.
Beispiele:

```
(*Kommentare sind in solchen Klammern!*)
window ().
(*Öffnet ein Fenster.*)
say ('Hallo!').
(*Öffnet ein Fenster und schreibt 'Hallo!' hinein.*)
Dateiinhalt is read ('datei.txt').
(*Liest den Text aus DATEI.TXT und ordnet es der Variablen "Dateiinhalt"
zu.*)
```

Für viele Befehle bestehen zudem mehrere alternative Schreibmöglichkeiten:
`?Var` ist gleichbedeutend mit `value_of (Var)` und ermittelt den Wert der Variablen Var;
`Var = 5.` oder `Var is 5.` oder: `make (Var, 5).` ordnet der Variablen Var den Wert 5 zu.

Das "hello world"-Beispiel

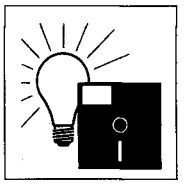
Das folgende Beispiel zeigt Text und einen OK-Button in einem Fenster an und wartet (`wait`) auf das Anklicken des Ok-Buttons, der die Unterbrechung der Programmausführung aufhebt (`continue`) so daß das Fenster wieder geschlossen wird (`close_window`):

---

22 "Since lists are the only predefined data type, you could view KnowledgePro as just another version of Lisp. But its object-oriented environment is much easier to use than any Lisp I've seen.", John McCormick, Government Computer News, 26/1992, S. 39–41

23 zur Ablaufsteuerung bei Prolog: Doris Elke Altenkrüger, Wissensdarstellung und Expertensysteme, 1987, S. 116

24 "Beim Versuch, Windows mit zwei Worten zu beschreiben, bleiben nur diese beiden übrig: Fenster und Nachrichten." Thomas Hornschuh, Klassen für Massen, C++-Klassenbibliotheken für Windows, c't 8/93, S. 207



```
window ().
text ('Hallo Welt').
button (Ok, continue). (*der Button wird unter den Text gesetzt*)
wait ().
close_window ().
(*schließt das geöffnete Fenster nach Anklicken des Buttons wieder*)
```

Genügen die Default-Einstellungen nicht (soll bei obigem Beispiel zB. ein größeres Fenster mit Titel erscheinen), so müssen die fehlenden Parameter in der Klammer ergänzt werden. Die vollständige Syntax, die für jeden Befehl im Referenzhandbuch und in der On-line Hilfe dargestellt wird (und aus dieser über die Zwischenablage in den Editor übernommen werden kann), lautet für ein Fenster:

`window (EVENT_TOPIC, COLUMN, ROW, WIDTH, HEIGHT, TITLE, STYLE, PARENT, TEXTCOLOR, BACKCOLOR, EVENT_LIST).`

`EVENT_LIST` kann die Events aufnehmen, auf die das Fenster reagieren soll. Wie es regiert, hängt vom Inhalt des `EVENT_TOPICs` ab, das wiederum beliebig definiert werden kann. `COLUMN, ROW, WIDTH, HEIGHT` bestimmen die Größe und Ausrichtung des Fensters. `STYLE` kann eine Liste mit den Stil-Eigenschaften aufnehmen (`Child, Popup, Overlapped`, etc). `PARENT` ist das übergeordnete "Eltern-Fenster".

3.2.2 Topics

Grundstruktur jedes KPWin-Programmes ist das *Topic*. Ein Topic beginnt mit dem Schlüsselwort `topic Topicname (Parameter)` wobei dieser `Topicname` beliebig gewählt werden kann: Wird der Text in Anführungszeichen eingeschlossen, können sogar Leerzeichen enthalten sein und so zur leichteren Verständlichkeit auch prägnante Satzteile (oder Sätze) bis zu 80 Zeichen als Topicbezeichnung verwendet werden; dies erleichtert die Erstellung und erhöht die Lesbarkeit eines Programmes (vgl. auch die Topic-Namen bei Expertensystemen 3.4, sowie die Idee des literarischen Programmierens unter 4.1). Abgeschlossen wird ein Topic mit `end`.

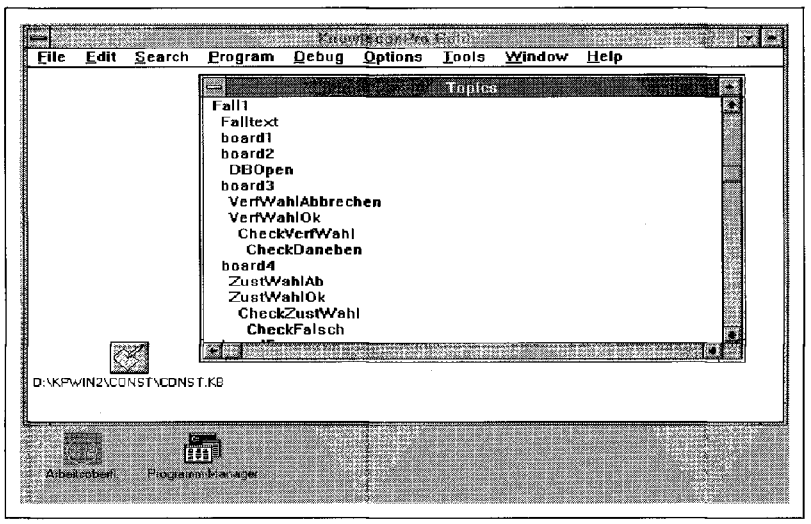


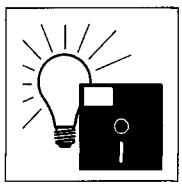
Abb. 4: Überblick über die Topic-Struktur

Ein Topic wird nur aktiviert, wenn es aufgerufen wird: dies kann durch eine Nachricht (Event) erfolgen, die der Anwender ausgelöst hat (zB. Klicken auf einen Hypertextbegriff) oder auch durch ein anderes Topic. In der Regel beginnt eine Applikation mit dem Aufruf der Topics, die die Anwendung initialisieren (zB. `topic Setup`) und das Hauptfenster öffnen (zB. `topic Hauptfenster`), und wartet dann auf die Benutzereingaben (Events), die weitere Topics ansteuern.

Jedes Topic kann auch untergeordnete Topics enthalten:

```
topic Haupttopic.
...
      topic Subtopic.
      ...
      end. (*Subtopic*)
end. (*Haupttopic*)
```

Obiges `Haupttopic` enthält hier ein untergeordnetes `Subtopic`.




Diese Hierarchiestufung, die sich so ergibt, orientiert sich am `Topic Main`[25], welches das höchstrangigste Topic darstellt und vom System gestellt wird, also immer und nur einmal vorhanden ist.
Topics können zur Verwaltung und Steuerung der Windows-Steuerungsobjekte verwendet werden: jedes Fenster kann so seine Elemente (Buttons, Check-Boxes, Combo-Boxes, ...) als Subtopics selber verwalten. Die Struktur und Funktionalität der Topics kann aber beliebig gestaltet werden; ein Topic kann ua. als Variable, Liste, Regeleinheit und Objekt dienen.

topic als "Gesprächsthema"

Die Bezeichnung Topic lehnt sich an ein "Gesprächsthema" an: Entsprechend der Struktur einer Unterhaltung können beliebige Informations-, Interaktions- oder Regeleinheiten gebildet werden, die das System anschließend je nach Anforderung aktiviert. Das Topic wurde entwickelt, um der Gefahr mangelnder Strukturierung von Hypertext-[26] und Expertensystemapplikationen ("Informations-Spaghetti") entgegenzuwirken, ohne auf Flexibilität verzichten zu müssen[27].

"Falltopics"

Der Inhalt und die Struktur eines Topics wird sich deshalb idealerweise aus dem zu lösenden Problem ergeben[28]. Ein Lernprogramm, das verschiedene zu lösende Fälle beinhaltet[29], kann zB. für jeden Fall ein Topic enthalten; diese "Falltopics" wiederum entsprechende untergeordnete Lern- oder Fragentopics.
Das Topic bildet auch die Grundlage für die objektorientierten Eigenschaften (siehe 3.2.6) von KnowledgePro, sowie für die Erstellung von Expertensystemen (3.4); ein Topic kann auch Multimedia-Anwendungen steuern[30].

Variablen als Topics

Auch Variablen sind in KnowledgePro Topics. Sie müssen nicht deklariert werden (die Deklaration würde sich mit der Möglichkeit des Backward Chaining spießen; siehe 3.4) und können nach Wunsch auch lokal definiert werden; dadurch wird das Innere des Topics nach außen abgeschirmt, bleibt aber trotzdem noch direkt ansprechbar[31].
Einem Topic können sowohl Zahlen als auch Zeichenketten oder (in eckigen Klammern) Listen zugewiesen werden; es gibt also keine Datentypen im herkömmlichen Sinn.
Beispiele:

```
Zahl is 5.
Wort is Zeitschrift.
```

Der Wert eines Topics wird mit `?Topicname` oder `value_of (Topicname)` ermittelt. Dabei wird nach dem Topic in einer bestimmten hierarchischen Reihenfolge gesucht, die für den Aufbau eines Expertensystems (3.4) wichtig ist.
`?Zahl` gibt demgemäß 5 zurück, `?Wort` "Zeitschrift".
Text- und Listenzuweisungen erfolgen entsprechend:

```
Satz is 'Dies ist ein Satz'.
(*Die Anführungszeichen sind wegen der enthaltenen Leerzeichen nötig*)
```

Satzliste is ['Dies ist der erste Satz.', 'Dies ist der zweite Satz.'].
Listeninhalte werden – falls nicht anders gewünscht – mit jedem Element in einer Zeile ausgegeben; auf diese Weise können auch Texte als Listen (jede Zeile ein Listenelement) gehandhabt werden.
`?Satzliste` stellt sich dementsprechend auf dem Bildschirm so dar:

```
Dies ist der erste Satz.
Dies ist der zweite Satz.
```

"Library Browser"

Oft verwendete Topics bzw. Klassen (3.2.6) von Topics oder mit dem Designer erstellte Oberflächen-Topics können in einer Datei zusammengefaßt und als Bibliothek gespeichert werden. Der integrierte "Library-Browser" (der ebenfalls mit KPWin geschrieben wurde und auch im Quellcode beiliegt) ermöglicht die Auswahl des gewünschten Topics, welches dann über die Zwischenablage direkt in die Applikation übernommen werden kann.

---

25 an "Main" in C/C++ angelehnt

26 zu dieser Gefahr vgl. Steins, a. a. O., S. 2248

27 KPWin++ Handbuch, Kapitel 1–4

28 Aus rechtstheoretischer Sicht besonders interessant ist die Verbindung zur Topik: "Es kommt stets auf die Gebrauchsfunktion an, die einem Topos seine Evidenz verleiht." (Lothar Bomscheuer, Topik. Zur Struktur der gesellschaftlichen Einbildungskraft., Suhrkamp 1976, S. 208.) Genau dasselbe gilt auch für die hier beschriebenen Topics!

29 Dieses und die folgenden Beispiele lehnen sich großteils an "Constitutor" an, dem Prototyp einer interaktiven Lernumgebung für das österreichische Verfassungsrecht.

30 zur Einbindung von Video: Karl Sarnov, KnowledgePro enthält Videoanschluß, c't 10/92, S. 122

31 Ein lokales Topic kann durch seine direkte "Adresse" angesprochen werden: diese geht vom Topic Main aus und enthält alle in der Hierarchie zwischengeordneten Topics; zB.: "Main:Fall1:Frage5" spricht das lokale Topic Frage 5 direkt an.

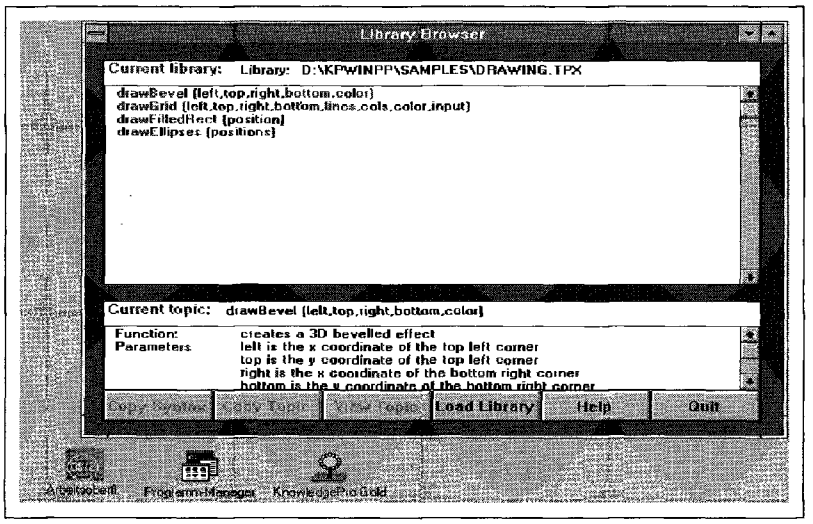


Abb. 5: Der Library Browser zeigt den Inhalt einer Topic-Bibliothek.

Die Einbindung von ganzen Bibliotheken als externer Datei ist auch mit der schon erwähnten Direktive `include` möglich. Diese Anweisung ermöglicht es, funktional zusammengehörige Gruppen von Topics in einer Datei zu belassen und hilft so mit, den Überblick über ein größeres Projekt zu bewahren.

3.2.3 Listen

Besonders an LISP erinnert die Fähigkeit von KnowledgePro, Listen zu verarbeiten[22] (der Name LISP kürzt ja "List Processing" ab[32]).
Die zahlreichen Befehle zur Listenverarbeitung (ua.: `combine, element, first, gets, last, list_length, one_of, sort, where, ...`) lassen alle – wie in LISP – die zu bearbeitende Liste unverändert. Dadurch werden Seiteneffekte vermieden und können rekursive Schleifen gestaltet werden[33] [34].
Listen können auch beliebig in Strings bzw. wieder zurück gewandelt werden (`list_to_string, string_to_list`).
Besonders praktisch erweisen sich parallel geführte Listen: In Constitutor[29] werden zB. die Fragen mit einer Liste verwaltet, die (hoffentlich) richtigen Antworten befinden sich auf einer zweiten parallelen und die vorgesehenen Reaktionen auf falsche Antworten warten auf einer dritten parallelen Liste: Wird eine bestimmte Frage gestellt (dh. ein bestimmtes Fragetopic aktiviert), so kann das System aus der Positionierung in der ersten Fragenliste die richtige Antwort sowie die angesagte Reaktion/Sanktion (ebenfalls Topics!) bei einer falschen Eingabe ermitteln.

Besonders praktisch: Parallel geführte Listen

Für solche parallelen Listen stehen weitere spezifische Befehle zur Verfügung (`gets_c` ordnet die Elemente einer Liste den entsprechenden Elementen einer anderen zu[35]. Da jedes Listenelement zudem selber wieder eine Liste sein kann, ist für weitere Flexibilität vorgesorgt.

3.2.4 Text

Die Listenfunktionen ermöglichen zusammen mit Dateifunktionen (zB.: `read, write`) und Hypertextbefehlen einen sehr einfachen und effizienten Umgang mit Texten. Unterstützt werden unformatierte ASCII-Texte als Zeichenketten (Strings) oder Listen. Die Formatierung jeder Zeichenkette wird innerhalb der Anführungszeichen direkt übernommen; die Zuweisung:

```
Satz is 'Überschrift.

   Kapitel 1
    Dies ist ein Satz'.
```

übernimmt also auch die Abstände links sowie die Leerzeilen.

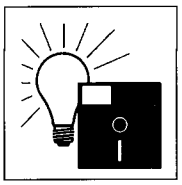
---

32 zu LISP als "Listenverarbeiter" vgl. Doris Appleby, LISP, BYTE, 11/1991, S. 165–169,

33 Appleby, a. a. O., zu Rekursion S. 168, Seiteneffekte S. 169

34 ein Beispiel zur rekursiven Fakultätsberechnung: Karl Sarnov, Mehrkämpfer. Programmiersystem KnowledgePro (Windows), c't 10/1991, S. 141

35 So bewirkt [Zeitschrift, Verlag] gets_c [jurPC, MediConsultGmbH], daß dem Topic Zeitschrift "jurPC" und dem Topic Verlag "MediConsultGmbH" zugewiesen wird.




Zur weitergehenden Formatierung sowie zur Font- und Farbgestaltung von Texten stehen zahlreiche Steuerzeichen zur Verfügung: `#n` bewirkt zum Beispiel einen Umbruch, `#fred` eine rote Textdarstellung (zu den besonders wichtigen Hypertextmarken siehe 3.2.5.1). Der Befehl `create_font`, ermöglicht die Einbindung sämtlicher Windows-Fonts.
Diese Steuerzeichen und Fonts-Befehle können im selben Fenster beliebig gemischt werden[36]: so lassen sich leicht Überschriften hervorheben bzw. verschiedenartige Hypertextverknüpfungen (zB. zu Gesetzen, zu Gerichtsurteilen) grafisch ansprechend darstellen. Darüberhinaus können die Fonts auch beliebigen Screen-Objekten zugeordnet werden, die Text enthalten, also auch einer List-Box oder einem Button. Auch automatischer Textumbruch oder eine rechtsbündige Darstellung kann leicht realisiert werden[37].

3.2.5 Hypertext/Hyperregionen/Grafik

Hypertext-Management

Hypertext ist eine Grundkomponente von KnowledgePro: Jeder Textteil, der von den Marken `#m` umgeben ist, wird als Hypertext interpretiert; befindet sich der Cursor über diesem Textteil, so wechselt er automatisch zum "Hypercursor": falls nicht abweichend definiert (`hyper_cursor`) ist dies ein Handsymbol.
Bei Aktivierung des Hypertexts durch Anklicken des Hypertextes wird die ganze Anwendung nach einem Topic mit demselben Namen wie des angeklickten Textteiles abgesucht und dieses gegebenenfalls aufgerufen. Wird ein solches Topic nicht gefunden, so wird ein bestimmtes Topic (mit dem Namen `topic mark`) aktiviert, welches diesfalls das Hypertexthandling übernimmt. Das Topic mit dem übereinstimmenden richtigen Namen oder das `topic mark` können nun den gefundenen Hypertext anzeigen oder in einer Datei nach dem Begriff suchen; sie können aber auch eine Datenbank abfragen oder eine Grafik (mit weiteren Hyperregionen) laden oder eine beliebige andere Aktion setzen. In Constitutor[29] analysiert `mark` den Hypertextbegriff weiter und bestimmt anhand der Endung (zB. "BGB") aus welcher Gesetzesdatenbank der gesuchte Paragraph oder Artikel hervorzuholen ist. Soll verhindert werden, daß sofort auf namensgleiche Topics zugegriffen wird (zB. um zuerst eine Synonymliste zu prüfen), so kann ein `topic super_mark` definiert werden, welches im Fall der Hypertextaktivierung jedenfalls Priorität genießt. Durch diese dreistufige Verarbeitung der Hypertextaktivierung kann flexibel jede beliebige "Hyper-Funktionalität" gestaltet werden.

"Erstellung eines Hypertexts nur mehr durch die persönliche Schreibgeschwindigkeit begrenzt ..."

Wie es nach Steins[38] für das Hypertexthandling von ToolBook kennzeichnend ist, daß die ToolBook-Hilfe selber *nicht* mit ToolBook erstellt wurde, so ist für KPWin bezeichnend, daß das KPWin-Hilfssystem selber mit KnowledgePro geschrieben ist und daß der dazugehörige Quellcode sowie eine weitere, vereinfachte Variante als Hypertext-Engine beiliegt: ENGINE.KB erlaubt es, beliebigen mit den Hypertextmarken versehenen Text als Hypertext einzulesen und darzustellen. Wie Sarnov[39] anmerkt, ist dadurch die Erstellung eines Hypertexts nur mehr durch die persönliche Schreibgeschwindigkeit begrenzt.

Ratsam: Hypertext-Index

Bei umfangreicheren Hypertexten ist die Erstellung eines Indexes ratsam; dazu liegt (auch im Quellcode) das Programm INDEX.KB bei. Selbst bei Textdateien mit einem Megabyte und mehr Umfang lassen sich dadurch akzeptable Zugriffszeiten realisieren (zu größeren Text- und Datenmengen vgl. auch das zusätzlich erhältliche Tool Wrap unten 5.3).

Einfaches Beispiel für Hypertext:

```
window(). (*Öffnet ein Standardfenster*)
text ('Die ist eine Hypertext Demo für #mjurPC#m!').
(*jurPC ist durch die Marken als Hypertext definiert*)
topic jurPC.
window().
text (' jurPC: Die Zeitschrift mit Diskettenbeilage!').
button (Ok, continue).
wait().
close_window().
end.
```

---

36 Bei Visual Basic muß dazu anstelle einer "Text-Box" eine "Picture-Box" verwendet werden.

37 Auf dem KnowledgeGarden Forum in CompuServe (Go Knowledge) finden sich dazu Beispiele.

38 Steins, a. a. O., S. 2249

39 Sarnov, Mehrkämpfer, c't 10/1991, S. 142

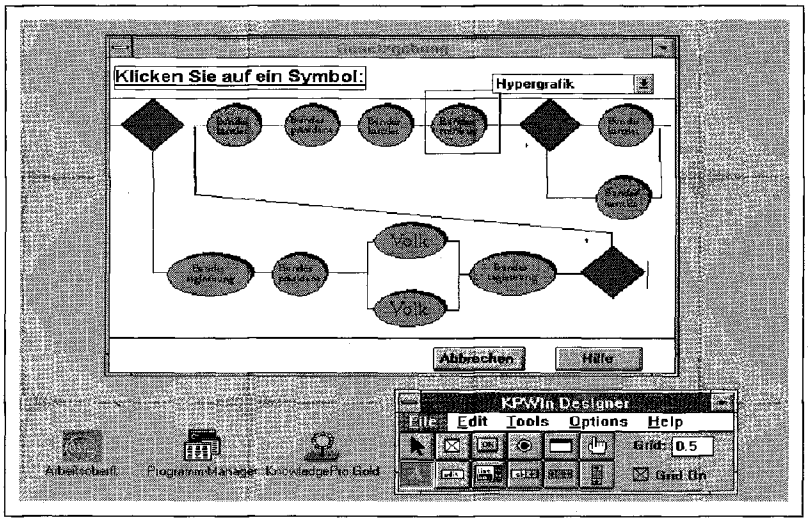


Abb. 6: Mit dem KPWin-Designer können auch Hyperregionen festgelegt werden.

Bei Aktivierung des Hypertextes "jurPC" durch einen Mausklick wird das gleichnamige Topic aufgerufen; dieses öffnet ein Fenster mit Text und wartet auf das Anklicken des OK-Buttons, der das Fenster wieder schließt.

Hyperregionen

Nach dem selben Muster lassen sich über beliebigen Icons oder Windows-Bitmaps "hyperregions" setzen, die die jeweilige Region als "Hyperfläche" definieren und nach dem selben Muster ein zu definierendes Topic bzw. `super_mark` oder `mark` rufen.

Berechnete Grafik

Seit der Version 2.3 wurden neben dem Verwenden von Bitmaps auch berechnete Grafik implementiert, deren Abwesenheit bisher einen Kritikpunkt bildete[40] [41].

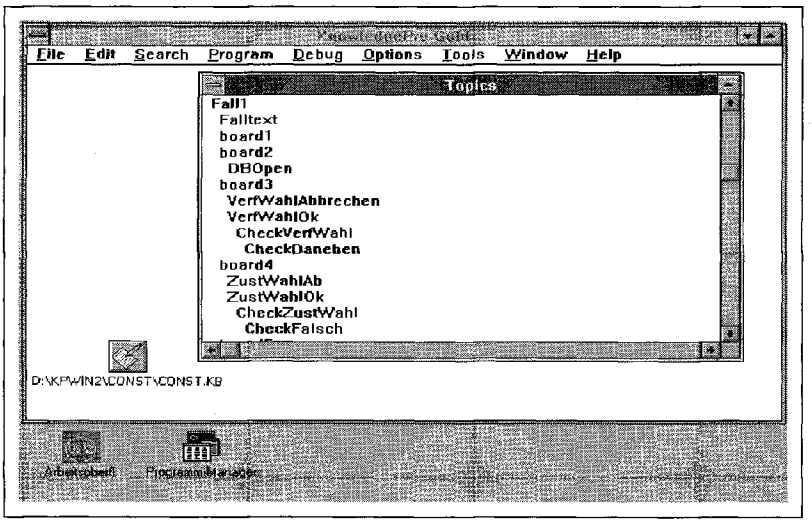


Abb. 7: Fallauswahl mit Hyperregionen in "Constitutor", einer interaktiven Lernumgebung für das österreichische Verfassungsrecht.

Das Beispiel zeigt ein schwarz umrandetes und grün gefülltes Rechteck:

```
Bleistift is create_pen (solid, 1, black). (*1 ist die Breite*)
Pinsel is create_brush (solid, green).
draw_rectangle ([ 1, 1, 20, 10], ?Bleistift, ?Pinsel).
(*Die eckige Klammer enthält die Maße für [ LINKS, OBEN, RECHTS, UNTEN]*)
```

3.2.6 Objekte

Objektorientierte Charakteristika

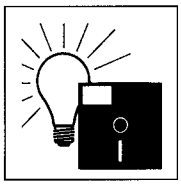
Die objektorientierten Features von KnowledgePro ermöglichen die Bildung von Klassen und Instanzen sowie die Mehrfachvererbung zwischen diesen.

Ausgangspunkt ist auch hier das Topic: ein Topic kann in beliebiger Form Befehle, Daten und Regeln beherbergen und stellt so ein frei zu definierendes Objekt im Sinne der objektorientierten Programmierung dar.

---

40 Sarnov, Mehrkämpfer, c't 10/1991, S. 146

41 Ulrich Klesper, Anwendungen selbstgemacht, Chip 5/1992, S. 172




Jedes Topic kann nun selber als Klasse zur Bildung eines neuen Topics als Instanz verwendet werden: mit dem Befehl `new` wird die Struktur des Klassentopics (also Variablenzuweisungen und Subtopics) in das neue Instanzen-Topic "kopiert"[42], dh. vererbt.
Wird dagegen ein neues Topic gebildet, welches eine abweichende Subtopic-Struktur aufweist, so kopiert der Befehl `im_a` nur diejenigen Subtopics der Klasse, die nicht schon im neuen Topic enthalten sind: es werden also die noch nicht vorhandenen Eigenschaften geerbt, wodurch eine neue Klasse entstanden ist.
Auf dieselbe Weise ist auch Mehrfachvererbung möglich: der Befehl `im_a` wird einfach mehrmals bezüglich verschiedener Klassentopics verwendet: das neue Objekt erbt die ihm jeweils noch fehlenden Eigenschaften.
Das Handbuch beschreibt weiterhin die Möglichkeit, ein Klassentopic derart zu kapseln, daß eine Instanz wie ein KnowledgePro Befehl gebildet werden kann[43].
Die objektorientierte Programmierung erlaubt es, oft benötigte Topics einfach – ohne Kopieren/Einfügen – wieder- und weiterzuverwenden: Constitutor[29] beinhaltet einige Klassen von Screen-Objects; wird zB. ein neues Frage-Window oder eine Dialogbox benötigt, so wird eine Instanz gebildet (`new`), soll das vorhandene Objekt modifziert werden, steht `im_a` zur Bildung einer neuen Instanz bereit.[44]
Die objektorientierten Eigenschaften können aber auch zur Speicherung von Wissen und Informationen verwendet werden.

3.2.7 Debugging

Fortsetzung nach Fehlermeldung möglich ...

Eine Interpretersprache bietet den Komfort, ohne Kompilierungsumweg direkt bei der Ausführung auf einen Fehler aufmerksam gemacht zu werden. Wenngleich die Programmausführung bei KPWin durch einfache Bestätigung der Fehlermeldung auch fortgesetzt werden kann (!), will der Fehler früher oder später doch einmal gefunden werden.

... aber manchmal etwas unscharfe Fehlermeldungen

Wird zu diesem Zweck nach der Fehlermeldung (in der Regel fehlt entweder ein Punkt oder eine Klammer) die weitere Ausführung abgebrochen, so findet sich der Cursor zwar in der mißlichen Zeile, doch kann es leider auch eine benachbarte sein: die KnowledgePro Fehlermeldungen sind etwas unscharf. Als Anfänger fühlt man sich angesichts der vielen Klammern leicht überfordert ...
Eine wenig elegante aber pragmatisch-wirksame Abhilfe schafft hier – neben der zumindest eingangs gebotenen Zurückhaltung bei der Verschachtelung – das temporäre Kopieren des verdächtigen Topics in ein neu geöffnetes Editierfenster mit neuerlichem Probelauf: Der Fehler ist so schnell eingekreist.
Daneben werden alle weiteren Vorteile eines Interpreters aufgeboten: Mit "Evaluate" kann zur gerade aktiven Anwendung frisch geschriebener Code in den Speicher dazugeladen und dieser so direkt am aktiven Programm erprobt werden.
Weiteres erlaubt eine Trace-Funktion die Schritt-für-Schritt Abarbeitung der Knowledge-Base, entweder zur direkten Beobachtung (single-step) oder mit gleichzeitiger Abspeicherung in eine Datei.
Und schließlich verschafft "Debug-Topics" dem Anwender einen grafischen Überblick über die Topic-Struktur sowie über den Status und Inhalt der Topics der gerade aktiven Anwendung; diese Funktion ist für Neueinsteiger hilfreich, bei größeren Applikationen und/oder vielen Regeln wird sie nahezu unentbehrlich..

3.2.8 Fortgeschrittenes

Der Tip im Handbuch, nicht lange herumzuüberlegen, sondern neue Ideen einfach auszuprobieren, ist wörtlich zu nehmen:

Klammerungen

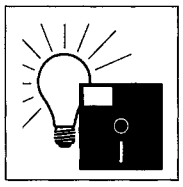
Befehle können – korrekte Klammerung vorausgesetzt – beliebig kumuliert werden;
Beispiel:
Da `?Var` den Inhalt einer Variable evaluiert, ermittelt `??Var` den Wert der ermittelten Variable (usf.!). Da `?Var` auch als `value_of (Var)` geschrieben werden kann, stellt sich `??Var` in Klammerform als `value_of (value_of (Var))` dar.

---

42 Da intern nur ein Pointer gesetzt wird, wirkt sich dies nicht negativ auf den Speicherhaushalt an.

43 KnowledgePro Windows, User Manual, S. 162 ff.

44 Für ein ausführliches Beispiel siehe Sarnov, c't 10/1991, S. 144



Auf dieselbe Weise können auch Werte zugewiesen werden:
`Var is Zahl.`
Da `?Var` jetzt "Zahl" ist, bewirkt `?Var is 5`, daß *Zahl* (als dem Wert von Var) 5 zugewiesen wird.
Oder, um dies mit obigem zu kombinieren, gleichbedeutend:
`Var is Zahl. Aktuell is Var. ??Aktuell is 5.`
Wie sich schon bei diesen Beispielen zeigt, ist die zu beachtende Grenze vor allem die Lesbarkeit bzw. die sorgfältige Kommentierung solcher Verschachtelungen!

Schreibfaulheit

Da die meisten Befehle auch Listen als Parameter akzeptieren, läßt sich Schreibarbeit sparen: Sollen zB. mehrere Fenster geschlossen werden, so muß nicht derselbe Befehl mehrmals hingeschrieben werden, sondern es genügen die Handies der Fenster als Liste:

```
close_window ([?Fensterl, ?Fenster2, ?Fenster3]).
(*Schließt die drei Fenster*)
```

Die Schreibfaulheit unterstützen auch weitere Befehle: Das simple `edit_file (DATEI)` lädt Text aus einer Datei in ein Editierfenster, komplett mit Menüzeile; `apply` bewirkt, daß alle Listenelemente von einem Topic abgearbeitet werden, ohne eine Schleife mit Zähler und Endbedingung formulieren zu müssen.

Eigenbau

Sämtliche KnowledgePro Befehle können durch eigene ersetzt werden.
Stößt KPWin bei der Programmausführung auf einen Befehl, so sucht es nämlich zunächst die in der Applikation vorhandenen Topics nach dem Befehlsnamen ab und wendet erst dann den KnowledgePro Befehl an.
Enthält ein Programm zB. ein `topic text` so wird dieses anstelle des KnowledgePro Befehls `text` verwendet.
Dadurch läßt sich jeder KnowledgePro Befehl beliebig umdefinieren. Da dieses Absuchen Zeit kostet und nicht immer benötigt wird, läßt sich dieses Feature ausschalten, sodaß jeweils sofort der KnowledgePro Befehl ausgeführt wird.

Dynamik

Topics können während der Laufzeit eines Programmes auch vom Programm selber erzeugt und ihnen Verhaltensweisen zugeordnet werden.

3.3 Kommunikation mit anderen Anwendungen

KPWin unterstützt den dynamischen Datenaustausch mit anderen Anwendungen (DDE) sowie den Aufruf von dynamischen Linkbibliotheken (DLL's).

DDE

Der dynamische Datenaustausch wird im Handbuch erklärt und kann mit Hilfe eines beigelegten Programmes anhand einer Abfrage einer Excel-Zelle genauer studiert werden; weiters liegt KPWin auch ein Installations-Programm im Source-Code bei, das per DDE dem Programm-Manager das neue KPWin-Programm inklusive Programm-Ikon hinzufügt, und für eigene Anwendungen verwendet werden kann.

DLL

Fortgeschrittene Anwender können mit Pascal oder C/C++ DLL's erstellen, und so rechenintensive Programmteile auslagern oder externe Programme einbinden (zB. neuronale Netze). Da Pascal bzw. C++ typengebunden sind, müssen DLL-Aufrufe natürlich auf den verwendeten Datentyp abgestimmt werden[45]. Einige der erhältlichen Zusatztools für KPWin (siehe unten 5.) sind selber DLL's, auf die mit dazugehörigen Topics zugegriffen werden kann.

(wird fortgesetzt)

[45] Die dazu notwendige Vorgangsweise findet sich im Handbuch: KnowledgePro Windows, User Manual, S. 241