Bezirksregierung Köln
Dezernat 45.3

Unterregionalisierte Fortbildung
für
Lehrerinnen u. Lehrer

Materialien für den Unterricht
im Fach Mathematik

Einsatz des Computers im Mathematikunterricht

Tagungsband zum
Workshop
der Uni Essen und der Planungstagung Mathematik der BR Köln und Düsseldorf

Mathematische Lernbausteine in Software-Umgebungen - ihre Entwicklung, Integration und flexible Nutzung

Alfred Schreiber
Bildungswissenschaftliche Hochschule Flensburg, Universität

Vortrag Universität Essen, 24. September 1996

Meine Damen und meine Herren:

Man braucht nur die Kataloge von Lernsoftware-Anbietern aufzuschlagen, um eine inzwischen beachtliche Fülle von Programmen zu finden, in denen es um Mathematik und Mathematiklernen geht. Nicht allein die Quantität des Angebots ist gestiegen, auch die Qualität hat sich seit Beginn der 90er Jahre zum Teil erheblich verbessert.

Bevor ich Ihnen genauer erkläre, was ich in meinem Vortragsthema unter *Lernbausteinen* und *Software-Umgebungen* verstehe, möchte ich zur Einstimmung und Orientierung zunächst einen kurzen Rundgang mit Ihnen unternehmen. Dabei stelle ich einige ausgewählte Beispiele vor; an ihnen soll jeweils verdeutlicht werden:

- ein charakteristischer Typus von Lernsoftware,
- sein fachlicher und didaktischer Einsatzbereich,
- evtl. vorhandene immanente Probleme.

Den im Folgenden als [Beispiel] gekennzeichneten Stellen entspricht im Vortrag jeweils eine Sequenz von Abbildungen oder die Demonstration eines Programmausschnitts einer Software.

1. Computer-Algebra-Systeme. Den größten Fortschritt gibt es zweifelsohne auf dem Gebiet der Werkzeuge, allen voran den sog. Computer-Algebra-Systemen (CAS). Mit diesen Programmen läßt sich nicht nur numerisch rechnen, wie man das von einem Computer dem Begriff nach erwartet. Sie führen auch symbolische Operationen aus wie das Lösen von algebraischen Gleichungen, Differentiation und Integration reeller Funktionen, Summation von Reihen, Taylorreihenentwicklung, zeichnen Funktionsbilder, erzeugen geometrische Objekte u.v.a.m. In DERIVE (für den Schulunterricht) kennen Sie bereits einen Vertreter dieser Softwaregattung. Das Programm paßt in seiner DOS-Version noch auf eine Diskette, läuft in einer (etwas antiquierten) Umgebung und ist auch für Schüler gut und relativ leicht handhabbar.

Am anderen Ende haben wir mächtige (unter Unix, Windows und vielen anderen Betriebssystemen laufende) Werkzeuge wie MATHEMATICA oder MAPLE. Sie eignen sich für echte wissenschaftliche oder ingenieurtechnische Anwendungen.

Nehmen wir als Beispiel einmal MATHEMATICA. Für dieses Programm gibt es inzwischen auch eine Reihe pädagogisch ausgerichteter Beiträge: die Zeitschrift "Mathematica in Education" (seit 1991), unzählige Artikel zum Unterrichtseinsatz (i.a. auf Hochschulniveau), Notebook-Beispiele elementarmathematischen Inhalts und - wie könnte es anders sein - reichhaltiges und frei zugängliches Material auf dem Server von Wolfram Research (speziell die Quellensammlung "Mathsource").

Beim Arbeiten in MATHEMATICA schreibt man seine Texte und Formeln in ein sogenanntes Notebook. Es ist hierarchisch in Zellen untergliedert, die sich zwecks besserer Übersicht gruppieren und ausblenden lassen. [Beispiel]

Zellen können gewöhnlichen Text, Formelausdrücke oder Grafik (jeweils als Input oder Output) enthalten. Der jeweilige Zelleninhalt läßt sich dann passenden Aktionen unterziehen: Formeln werden umgeformt oder ausgewertet, Grafiken werden geometrisch und farblich aufbereitet oder durch Sequenzbildung animiert usw.

Ist ein System wie MATHEMATICA für den Unterricht geeignet? Das hängt von der Situation ab. Der Reichtum an Funktionen und Möglichkeiten und die dadurch bedingte hohe Systemkomplexität machen MATHEMATICA natürlich zu einem sehr anspruchsvollen Werkzeug. Das ist aber an und für sich noch kein allein entscheidendes Kriterium. Für die pädagogische Eignung wesentlich ist vielmehr die Frage: Läßt sich die Komplexität verbergen? Läßt sich das System ausschnittweise und dabei auch entsprechend einfach nutzen?

Bis zu einem gewissen Grade kann man dies bejahen, wenn z.B. der Schüler (Student) zunächst die Rolle eines Lesers dessen übernimmt, was ein Lehrer (Dozent) vorbereitet hat.

Einige kritische Fragen drängen sich an dieser Stelle auf:

- Lohnt sich die Anschaffung und Installation eines solchen Werkzeugs, um dann damit nach dem Motto "Mit Kanonen auf Spatzen schießen" einfache Fragestellungen zu behandeln? (Meines Erachtens kann die Antwort unter Umständen durchaus "Ja" lauten. Es bleibt so eine Reserve für sich weiterentwickelnde Nutzung. Man vergleiche die Situation einmal mit der bei einer Textverarbeitung.)

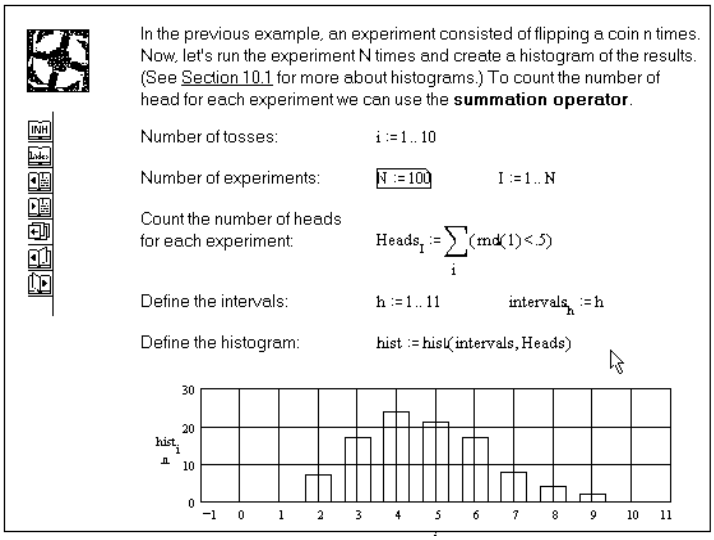
- Sind die Präsentationsmöglichkeiten des Materials didaktisch angemessen? Die Darstellung von Formeltext in der Eingabe ist erzwungenermaßen eindimensional, und die nicht allzu komfortable Ausgabe gelingt nur mit speziellen Befehlen. In diesem keineswegs nebensächlichen Aspekt unterscheiden sich die bekannten CAS zum Teil beträchtlich. (In der aktuellen Version 3.0 wurde MATHEMATICA diesbezüglich grundlegend überarbeitet.)

- Wie schwierig ist der Übergang von der mehr oder weniger rezeptiven Nutzung vorgefertigter Lehrtexte hin zu einem aktiven und produktiven Umgang mit mathematischen Begriffen und Problemen? Wenn wir uns einmal Schüler oder Studenten vorstellen, die mit dem mathematischen Stoff selbst ihre Probleme haben, so erscheint das didaktische Potential hier doch einigermaßen eingeschränkt. Die Notwendigkeit, sich vorher mit den jeweils nutzbaren CAS-Funktionen vertraut zu machen, erzeugt neuen Lernbedarf und baut neben den schon bestehenden inhaltlichen Schwierigkeiten neue Hürden auf.

Das im letzten Punkt angesprochene Problem gilt tendenziell für alle Computeralgebra-Programme vom Schlage MATHEMATICA. Aber auch für ein schulgerecht ausgelegtes Werkzeug muß erst einmal eine Einarbeitung und ein entsprechender Lernvorlauf in Kauf genommen werden. Daher: Bei aller Begeisterung, die Fachdidaktiker und Software-Entwickler für die (in der Tat nicht nur schöne, sondern auch produktive) neue Werkzeug-Welt empfinden, darf diese Aneignungsproblematik nicht bagatellisiert werden.

Ein Rückblick auf die vergangenen Jahre macht aber deutlich: Im allgemeinen wird auch komplexe Software zunehmend leichter zugänglich, intuitiv erschließbar und handhabbar. Das liegt zum einen an den verbesserten und stärker standardisierten Schnittstellen multimedialer Fenstersysteme. Es liegt aber auch an den Ideen, die einige Hersteller auf fachspezifischer Ebene realisieren.

Ein Beispiel hierfür liefert (das vor allem im Ingenieurwesen beliebte) MATHCAD. Sein Arbeitsblatt nimmt Text, Grafik und Formeln an jeder Position auf (nicht hierarchisch strukturiert wie ein MATHEMATICA-Notebook). Beim Eingeben eines mathematischen Ausdrucks erscheint dieser sofort satztechnisch richtig und wird auf Wunsch auch sogleich vom Formelprozessor ausgewertet. Autoren und Lehrer können auf diese Weise ihre Lehrtexte frei eingeben und so gestalten, daß Schülern noch Spielraum für eigene Experimente bleibt. Ein interessantes Beispiel dafür sind die in didaktischer Absicht (allerdings mit einem speziellen "Authoring Kit") erstellten elektronischen Handbücher zu MATHCAD, z.B. das von Heidi Russell geschriebene Skript "Algebra II". [Beispiel]



Aus: H. Russell, Algebra II (Mathcad Education Library, Cambridge 1993)

Die Navigationsmöglichkeiten in einem solchen Handbuch sind gut. Setzt man daneben ein eigenes Arbeitsblatt, so lassen sich die Elemente vom Handbuch leicht auf das Blatt übertragen und die Anregungen und Aufgaben des Lehrers sofort parallel zum Text aufgreifen und bearbeiten.

2. Dynamische Geometrie. Wechseln wir zu einem anderen Werkzeugtyp, der seit einigen Jahren in dediziert pädagogischer Absicht entwickelt wurde: Programme zur dynamischen Geometrie. Auf dem Markt gut verfügbare Vertreter dieser Gattung sind CABRI GÉOMÈTRE, GEOMETER'S SKETCHPAD, EUKLID, THALES. Worum es hierbei geht, versteht man am besten durch eine kleine praktische Demonstration, die ich in dem von Roland Mechling entwickelten Shareware-Programm EUKLID durchführe (und die sinngemäß ebenso möglich wäre in anderen Programmen). [Beispiel]

Einige Besonderheiten möchte ich herausstellen:

- Die Unmittelbarkeit der beweglichen geometrischen Objekte ist hier naturgemäß größer als bei den in CAS gebotenen Elementen. Die Handhabung ist leichter, der Experimentierfreude wird Raum gegeben, und entdeckendes Lernen kann in Gang kommen.

- Ein besonderer Vorteil gegenüber traditionellem Zeichenwerkzeug ist die Herstellbarkeit von Ortslinien.

- Die umfassenden Editier- und Recording-Funktionen der Geometrie-Programme ermöglichen es dem Lehrer, interaktive (sogar animierte) Arbeitsblätter für seine Klasse vorzubereiten. Für schwächere Schüler lassen sich Blätter mit mehr Vorgaben bzw. Hilfen entwickeln.

Eine Auswertung dessen, was ein Schüler macht, findet (in Programmen der Kategorie Werkzeuge) typischerweise nicht statt. In der Praxis bedeutet dies, daß diese Art Lernsoftware-Einsatz tutoriell begleitet und in den normalen Klassenunterricht eingebettet werden muß - eine zugegeben nicht immer ganz leichte Aufgabe.

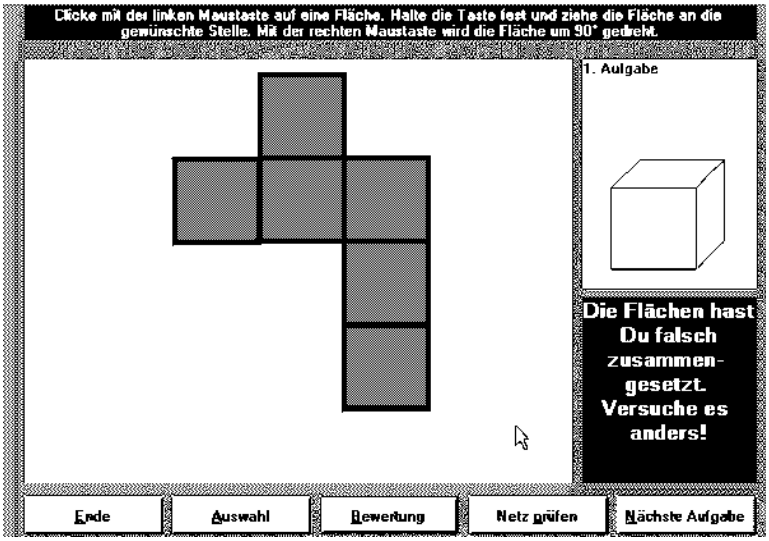
3. Übungsprogramme, Tutorien. Das letzte Gebiet, das ich bei diesem kurzen Rundgang betreten möchte, ist das der Übungsprogramme und Tutorien. Die Tradition dieses Lernsoftwaretyps geht weit zurück bis in die 60er Jahre und wird heute durch einen bunten Reigen von Programmen fortgesetzt, die ein Vertreter eines bekannten Verlages einmal treffend "Nachmittagssoftware" genannt hat. Er spielte damit auf den Zeitpunkt der Nutzung an, nämlich nach der Schule, was im allgemeinen auch heißt, daß der legitime Zweck dieser Programme die häusliche Nachhilfe oder Nachbereitung des Schulstoffs ist. Dazu werden dem Schüler Aufgaben zur Bearbeitung

vorgelegt, deren Lösung anschließend das Programm ausgewertet und zum Anstoß einer weiteren Interaktion benutzt (wobei uns der zwischen Drill- und Lehrprogrammen bestehende Unterschied hier nicht näher zu beschäftigen braucht).

Werfen wir an dieser Stelle einmal einen unvoreingenommenen Blick in drei Programme dieses Typs:

Die angeblich einmal meistverkaufte Lernsoftware zur Mathematik ("MatheBlaster", Davidson/Klett) ist im Kern ein Übungsprogramm zur Arithmetik mit Schwierigkeitsgraden für 6- bis 12-jährige Kinder; seine (laut Handbuch über 50000 möglichen) Aufgaben sind in ein multimedial unterlegtes galaktisches Abenteuerspiel mit umwelterzieherischem Anstrich verpackt. Wer will, kann mit einem Editor eigene Aufgabenpakete definieren, die das Programm dann in seiner Rahmenhandlung verwendet.

In etwa an dieselbe Altersgruppe richtet sich das Programm "Elly" (von J. Ingold, Cornelsen Software), das geometrische Aufgaben vorlegt und die Schüler-Antworten auch untersucht. Hier einige der besten Frames aus diesem Programm zum Streckenmessen, Flächenzeichnen, detaillierten Beschreiben von Körpern und Zusammensetzen von Würfelnetzen: [Beispiel]



Aus: J. Ingold, Elly für Windows (Cornelsen Software, 1994)

Schließlich noch einige Kostproben aus der jüngst in der TR-Verlagsunion erschienenen Reihe "Telekolleg II". [Beispiel]

4. Lernbausteine, Software-Umgebungen. Das bisher gesichtete Beispielmaterial sollte ein ausreichender Hintergrund für eine Vorstellung der Idee sein, mathematische Lernbausteine in Software-Umgebungen zu integrieren und zu nutzen. Ich erkläre zunächst, was ich in diesem Zusammenhang unter "Lernbaustein" und "Software-Umgebung" verstehen möchte. Es handelt sich naturgemäß nicht um exakte Definitionen, sondern um eine intuitive Eingrenzung des Gemeinten.

Lernbaustein

Hierunter möchte ich eine relativ selbständige Einheit aus einem informationellen Gegenstand (Lerninhalt) und einer zugehörigen (auf den Lerninhalt bezogenen) Interaktion verstehen.

Das simpelste Beispiel wäre etwa ein Lehrtext (Gegenstand), der gelesen bzw. durchgearbeitet (Interaktion) werden muß. Typische Lernbausteine in der Mathematik sind Aufgaben bzw. Probleme, die gelöst werden und deren Lösungen ausgewertet werden sollen.

Ein grobes Einteilungsraster könnte vielleicht so aussehen:

- Texte (eventuell bebildert)
- Fachglossare (oder ähnliches)
- interaktive Objekte (z.B. bewegliche geometrische Figuren)
- diverse Typen (mathematischer) Aufgaben (einschließlich Antwortanalyse)

Wir haben gesehen, daß sich solche Lernbausteine auch in Computerprogrammen realisieren lassen. Wir konnten aber auch beobachten, daß die Präsentation des Inhalts und dann die Ausführung und Auswertung der zugehörigen Interaktion an eine spezielle Programm-Umgebung gebunden war:

- Eine in MATHEMATICA eingegebene Formel benötigt zu ihrer Evaluation den Systemkern von MATHEMATICA.
- Die Konstruktionen auf einem dynamischen Zeichenblatt werden nur innerhalb der jeweiligen Werkzeug-Umgebung verstanden und realisiert.
- Eine in einem Übungsprogramm gestellte Aufgabe kann nur bearbeitet und beurteilt werden innerhalb der jeweiligen Programmumgebung.

Das ist nicht weiter verwunderlich, schließlich haben wir uns daran gewöhnt (auch wenn es manchmal schwer fällt), uns auf die Besonderheiten der einzelnen Programme und ihrer unterschiedlichen Benutzungsoberflächen einzustellen. Und das im "Windows-Zeitalter"! Es geht aber nicht allein um die Frage der Bedienung; die fraglichen Lernbausteine sind eben von einem anderen Programmkontext aus nicht nutzbar. Sie lassen sich nicht verpflanzen und gestatten mangels entsprechender Schnittstelle keinerlei Zugriff von außen.

Ich möchte aber einmal umgekehrt fragen: Wäre es nicht möglich und sinnvoll, Lernbausteine mit weniger Laufzeit-Kontext und das heißt: mit weniger Ballast zu konzipieren?

Betrachten wir die damit verbundenen Vorteile:

- Die lokale Formalstruktur unterschiedlicher Lernbausteintypen läßt sich abstrahieren und zu einer höheren Reife (und damit auch didaktischen Qualität) entwickeln als dies bei der bisher üblichen Technik der Individual-Konzeption der Fall ist.

- Ein generisches Framekonzept (z.B. eine Weiterentwicklung des von Karl Eckel in seinem Buch "Didaktiksprache" beschriebenen "Kleinstunterrichts" in Gestalt eines Mikrotutors) könnte jedem Einzelbaustein die benötigte Schnittstelle vererben.

- Ein Autor arbeitet ökonomischer aufgrund der durch dieses Lernbausteinkonzept bedingten Schablonentechnik.

- Die Resultate lassen sich leichter pflegen und wiederverwenden. Auch die Portierbarkeit wird erleichtert, günstigstenfalls wird sogar Plattformunabhängigkeit erreicht.

- Bildungseinrichtungen könnten sich Sammlungen anlegen, Bausteine austauschen.

Natürlich gibt es auch die Kehrseite der Medaille:

- Die Schablonentechnik engt den Autor ein, vor allem wenn das Repertoire verfügbarer Typen nicht reichhaltig genug ist.

- Im allgemeinen wird die nahtlose Integration so erstellter Lernbausteine in geschlossene Multimedia-Lernsoftware-Produkte (wie heutzutage üblich) erschwert oder sogar unmöglich. (Ob dies am Ende wirklich ein Nachteil ist, ist ja gerade hier die Frage.)

Der zuletzt genannte Punkt liefert mir eine willkommene Überleitung zum nächsten Stichwort meiner Vortrags.

Software-Umgebungen

Dieser Begriff ist etwas verschwommen. Er zielt auf die bekannte Tatsache, da ßein Computer Software benötigt, um bestimmte Aufgaben auszuführen. Die unterste Schicht bildet dabei das Betriebssystem und die mit ihm verbundenen Dienste (also etwa: DOS/Windows). In einer solchen Umgebung können dann sog. Anwendungen laufen, beispielsweise Lernprogramme. Nun könnte man weiter meinen, ein Lernprogramm bilde eine Software-Umgebung für die in ihm enthaltenen Informations- und Aufgaben-Bildschirme. Daran ist soviel wahr, daß die besagten Frames eben ohne das Lernprogramm nicht existenzfähig sind. Wohlgemerkt: nicht ohne das Lernprogramm als ganzes! Dies ist für mich Anlaß genug, um einer Software das Prädikat "Umgebung" in einem engeren und strengeren Sinne zu verweigern.

Eine echte Software-Umgebung ist der zu echten Lernbausteinen (oder allgemeiner: Komponenten) komplementäre Part. Z.B. sollte sie eine Aufgabe (oder Aufgabensequenz) an geeigneter Stelle aufrufen können (und diese dabei nur einen möglichst kleinen Kontext für ihre Abwicklung mit sich ziehen). Die Aufgabe sollte nicht auf eine ganz bestimmte Umgebung angewiesen bleiben. Umgekehrt sollte ihre Verbindung zu einer Umgebung in einer Referenz bestehen, die jederzeit modifiziert oder entfernt werden kann. Lernbausteine und die sie umgebende Software wahren also nach dieser von mir hier propagierten Auffassung eine relative Eigenständigkeit.

5. Stand der Dinge. Wie weit entspricht nun die Realität diesen eben entwickelten Postulaten?

Wir haben gesehen: Multimedia-Lernsoftware bietet (aus durchaus nachvollziehbaren Gründen) im allgemeinen nur Zugriff auf Inhalte innerhalb eines geschlossenen individuellen Produkts. Eine erste zaghafte Öffnung stellt die bei Vokabeltrainern (und z.B. auch bei "MatheBlaster") geübte Methode dar, einen Editor zur Erstellung eigener Aufgaben-Items mitzuliefern.

Gewöhnlich brauchen aber auch die neu hinzugefügten Items die ursprüngliche Programmumgebung in vollem Umfang.

Bei Werkzeugen (z.B. zur dynamischen Geometrie) könnte man sich sehr gut vorstellen, daß sich ein Zeichenblatt nicht nur als statische Grafik (via Zwischenablage), sondern samt seiner dynamischen Eigenschaften in eine andere Umgebung einfügen (einbetten) läßt. Die Realisierung dieser naheliegenden Option ist m.E. eine Frage der Zeit.

Computeralgebra-Systeme, zumindest die größeren, kommen der geschilderten Idee einer Software-Umgebung schon wesentlich näher. In der Regel verfügen sie über Schnittstellen zur wechselseitigen Kommunikation mit externen Modulen. Von besonderem Interesse ist dabei die Möglichkeit, anderen Programmen die Funktionen ihrer Systemkerne verfügbar zu machen. Für die Entwicklung interaktiver mathematischer Lernbausteine ergibt sich daraus ein beachtliches (bislang meines Wissens auf diesem Gebiet noch ungenutztes) Potential.

Ein schönes Beispiel für die Ausnutzung dieser Technik, allerdings auf dem Gebiet der Textverarbeitung, ist das Programm SCIENTIFIC WORKPLACE, mit dem perfekte LaTeX-Dateien erstellt und die schon beim Schreibvorgang satztechnisch korrekt entstehenden Formeln über eine eingebundene MAPLE-Engine oder den MATHEMATICA-Kern evaluiert werden. [Beispiel]

Ein CAS läßt sich aber schon auf einfachere Weise mit Lernbausteinen koppeln. Angenommen, in einem MATHEMATICA-Notebook werde zur Darstellung eines Sachverhalts Text, Grafik und MATHEMATICA-Input (etwa für eine Beispiel-Berechnung) vorbereitet. Dann ließe sich darüberhinaus auch eine Unterzelle vom Typ

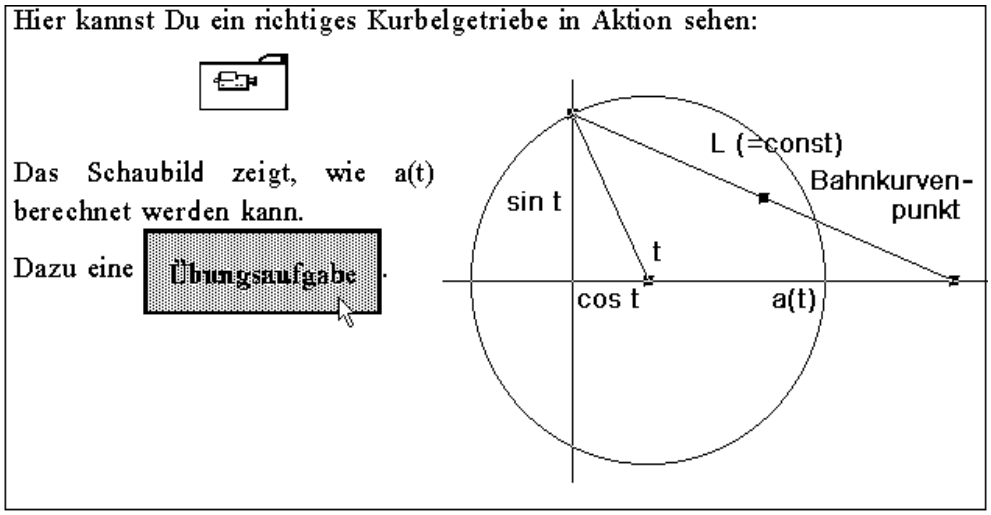
```
Run[ "Zugehöriger Lernbaustein" ]
```

anbieten, über deren Ausführung der Notebook-Leser (=Lerner) nach Bedarf entscheiden kann. Natürlich ist diese Technik nur dann sinnvoll, wenn der Lernbaustein selektiv aufrufbar ist. Der Vorteil, den eine CAS-Umgebung dabei bietet, besteht in ihrer interaktiven Eingabe und den damit unmittelbar zum Zweck der Problemlösung durchführbaren Berechnungen.

Dieselbe Methode kann grundsätzlich in allen Umgebungen benutzt werden, welche die Definition lokal wirksamer Verknüpfungen zu einer Betriebssystem-Shell gestatten. Dazu gehören typischerweise Hypertextsysteme (von denen Ihnen sicherlich die Windows-Hilfe und das WWW bekannt sind). [Beispiel]

Internet-Seiten mit Interaktivität auszustatten, ist momentan ein hochaktueller Trend, gekennzeichnet durch Schlagworte wie "Java-Applets" oder "Universal Application Access" (dem unter dem Codenamen "Broadway" angekündigten Fernzugriff auf Anwendungen im WWW).

Soweit brauchen wir gar nicht zu gehen. Schon ganz gewöhnliche Textverarbeitungsprogramme wie WORD erlauben eine gewisse Interaktion. Stellen Sie sich dazu am besten ein Textdokument über ein mathematisches Thema vor, in dem nicht nur die zugehörigen Grafiken, sondern an den passenden Stellen auch Aufgaben eingefügt sind, die den Lesern des Dokuments mittels einer kleinen Schaltfläche (oder wie oft üblich: durch hervorgehobene Wörter) angeboten werden. [Beispiel]



Beispiel einer Textseite mit verknüpfter Videosequenz und Übungsaufgabe

6. Interaktive Aufgaben, Projekt DUAL. Das zuletzt gezeigte Beispiel soll zugleich daran erinnern, daß sich bereits viele Objekttypen (Grafik, Video, Audio, Tabellen etc.) über die sog. OLE-Schnittstelle in geeignete Umgebungen einbetten lassen. Für interaktive Aufgaben, für die man schon mit weitaus weniger als OLE auskommt, ist das aus verschiedenen Gründen noch nicht der Fall:

- Der Begriff "interaktive Aufgabe" ist relativ unbestimmt und läßt sich geradezu beliebig ausdifferenzieren; dies erschwert die Schaffung allgemeiner (und erst recht allgemein verbindlicher) Formate.

- Das Anwendungsfeld "Pädagogik" fällt im Vergleich zu den anderen Einsatzgebieten eher klein aus und zieht auch von der Sache her geringeres Interesse auf sich.

- Die bisherige und die gegenwärtige Entwicklerpraxis ist beherrscht von proprietären Formaten und Systemen. Ein Autor von Lernsoftware, der z.B. mit einem Autorensystem wie ToolBook oder Authorware arbeitet, bleibt stark an sein Werkzeug gebunden. Es unterstützt ihn bei der Erstellung geschlossener Lernprogramme, nicht jedoch bei der Entwicklung flexibel nutzbarer interaktiver Lernbausteine.

Hinzu kommt, daß mathematikspezifische Lernbausteine ein ganz eigenes und schwieriges Problem darstellen. Ich möchte Ihnen das anhand meiner Bemühungen darstellen, mich dem Thema vor dem Hintergrund der hier vorgetragenen Ideen auch in der Praxis der Entwicklung zu nähern. Diese Annäherung gliedert sich in drei Etappen:

A. Konzeption und Entwicklung einer tutoriellen Umgebung, in der einzelne interaktive Lernbausteine präsentiert werden können. Die globalen (bausteinübergreifenden) Funktionen (z.B. Verwaltung des Lernerprofils, Navigation) übernimmt ein *Makrotutor*; die lokalen (auf den Lernbaustein bezogenen) Funktionen (z.B. Aufgabenstellung, Antwortanalyse, Kommentierung) realisiert ein *Mikrotutor*. Zusätzlich zu dieser Arbeitsteilung wurde eine weitgehende *Trennung von Form und Inhalt* verwirklicht. Dies macht es möglich, einzelne Lernbausteine selektiv und relativ unabhängig vom Makrotutor zu aktivieren.

B. Das in Etappe A entwickelte System DUAL (Akronym für "Didaktik-Umgebung für adaptive Lernprogramme") kann nur auf der Ebene des Mikrotutors gebietsspezifisch werden; dem Makrotutor erscheinen alle Lernbausteine ("Frames") lediglich als abstrakte undurchsichtige Items. Dies ist sozusagen der Preis für die Trennung der Komponenten. Im ersten Anlauf wurden zudem hauptsächlich die allgemeinen und klassischen CBT-Aufgabenformate zur Reife gebracht: diverse Auswahltypen, Zuordnungen, Lückentexte, freie Eingaben und, als mathematisch ausgelegter Bausteintyp, generative numerische Frames für die Erstellung von Rechenaufgaben. Wenn man sich ausschließlich auf dieses Repertoire beschränken muß, ist die Entwicklung überzeugender Aufgaben nicht ganz leicht. Dies zeigt eine erste Probe mit Lernbausteinen für das Grundstudium. [Beispiel]

Aus technischer Sicht war das Hauptergebnis dieser zweiten Etappe die Realisierung selektiver Aufrufe von DUAL-Lernbausteinen aus geeigneten

Windows-Umgebungen. Z.B. kann man, wie das folgende Beispiel zeigt, Aufgaben aus ganz unterschiedlichen Projekten (über ein Skript oder durch entsprechende Parameterübergaben) aktivieren. [Beispiel]

C. In der dritten und aktuellen Etappe sind vor allen Dingen zwei Probleme zu lösen:

1. die Umsetzung des Systems auf Windows,
2. die Entwicklung mathematikspezifischer Frametypen nach dem oben beschriebenen Bausteinkonzept.

Wie könnten mathematische Frames konkret aussehen? Beispiele hierfür liefern etwa die folgenden Bildschirme aus dem (unter der Leitung von R. Schulmeister entwickelten) Programm "LernSTATS", das Studierenden der Psychologie und Sozialwissenschaften entdeckendes Lernen auf dem Gebiet der deskriptiven Statistik ermöglich soll. [Beispiel]

Dasselbe gilt für ein noch im Rohbau befindliches Programm aus meiner Werkstatt, das sich als Minimalumgebung für mathematische (und andere) Aufgaben versteht. Eines der Ziele, die ich mir hierbei setze, ist die Realisierung interaktiver Figuren der Geometrie in Verbindung mit Aufgaben, die vom eingebauten Mikrotutor auch tatsächlich evaluiert werden. [Beispiel]

Meine Damen und Herren: ich bedanke mich für Ihre Aufmerksamkeit.

Literatur

Eckel, K.: *Didaktiksprache.* Grundlagen einer strengen Unterrichtswissenschaft. Köln, Wien: Böhlau Verlag, 1989.

Hamann, K.: Broadway - besser als Java oder nur anders? In: OBJEKTspektrum 4/1996, 13-14.

Schreiber, A.: Bausteine für Lernprogramme. Beschreibung und Implementierung. In: H.-J. Friemel u.a. (Hrsg.), Forum '90 - Wissenschaft und Technik, Neue Anwendungen mit Hilfe aktueller Computer-Technologien, Trier, 8./9. Oktober 1990. Proceedings, 333-348. Berlin; Heidelberg [u.a.]: Springer-Verlag, 1990.

Schreiber, A.: Eine Didaktik-Umgebung für adaptives Lernen (DUAL). In: Grundlagenstudien aus Kybernetik und Geisteswissenschaft, 33/1 (1992), 25-32.

Schulmeister, R.: *Grundlagen hypermedialer Lernsysteme. Theorie - Didaktik - Design.* Bonn; Paris [u.a.]: Addison-Wesley, 1996.