Mitschian, Haymo

Objektorientierte Programmierung: Perspektiven für computergestützte Lern- und Lehrprogramme DaF

Unterrichtswissenschaft 21 (1993) 3, S. 261-280

Unterrichtswissenschaft

Zeitschrift für Lernforschung
21. Jahrgang / 1993 / Heft 3

Thema:
Lernen in Unternehmen

Verantwortlicher Herausgeber:
Heinz Mandl

Allgemeiner Teil

Haymo Mitschian

Objektorientierte Programmierung: Perspektiven für computergestützte Lern- und Lehrprogramme DaF

Objectorientated programming: Perspectives for computer based educational software

Einige der Gründe, weshalb DaF-Lehr- und Lernprogramme nicht im erhofften Umfang entstehen, liegen in den Verständigungsproblemen zwischen Fremdsprachendidaktikern und Programmierern. Die auf graphischen Benutzeroberflächen mögliche objektorientierte Programmierung erlaubt eine relativ klare Aufgabenverteilung, wobei die Erstellung der Oberfläche den Didaktikern, die Entwicklung des Programmcodes den Programmierern zufällt. Der Bereich sich überschneidender Kenntnisse — Computerwissen auf Seiten der Didaktiker, fremdsprachendidaktisches Wissen der Programmierer — wird dadurch auf ein Minimum reduziert. Die objektorientierte Programmierung bietet also die Voraussetzungen, um sowohl fremdsprachendidaktisch als auch computertechnisch hochstehende Lernprogramme zu entwickeln. Außerdem wird durch die Basis der graphischen Benutzeroberfläche eine Standardisierung erreicht, die den Benutzern den Zugang zu den Programmen erleichtert.

The lack of software for teaching and learning German als a foreign language is partly due to difficulties in communication between language teachers and programmers. Objectorientated programming based on a graphical user interface offers a comparativly straight separation of tasks with teachers developing the interface and programmers writing code. Thus a need for overlapping knowledge — computer know-how by teachers and language teaching knowledge by programmers — is reduced to a minimum. Therefore, objectorientated programming has the presupposition for the production of high quality software as far as methods of language teaching and use of technical resources are concerned. Moreover graphical user interfaces are enhancing operation standards and by that facilitating user access to learning software.

„Nach mehr als zehn Jahren computergestütztem Fremdsprachenunterricht in Deutsch kann man sich über den schleppenden Gang des Fortschritts nur wundern: Auch mit dem eifersüchtigen Partikularismus einsamer Helden hat das etwas zu tun" (Räkel & Steinfeld 1992, 145). Als Folge der Entwicklung und einsetzenden Verbreitung der Mikroprozessoren zu Beginn der achtziger Jahre zeichnete sich zunächst ein langsamer Anstieg an Softwareprodukten für den DaF-Unterricht ab, der jedoch nach einem Höhepunkt im Jahr 1985 drastisch abbrach: für 1990 konnte lediglich eine Produktion auf diesem Sektor verbucht werden (Walti 1992, 147). Neben den — freiwillig oder gezwungenermaßen — „einsamen Helden" sieht Walti (ebd., 147f.) andere Ursachen für das versiegende Engagement:

— die gewachsenen Anforderungen, die an neue Programme gestellt werden, verzögern ihr Erscheinen;
— rückläufige Haushaltsmittel für Schulen und Universitäten beschränken die Aufnahmebereitschaft des Marktes und damit auch die Produktion;
— eine geringe Bereitschaft, einmal erarbeitetes — und beherrschtes — Material aufzugeben, um sich mit neuen Programmen auseinanderzusetzen;
— eine große Gruppe von Skeptikern, die befürchten, „der Computer könnte auf Grund seiner großen Leistungsfähigkeit den Lehrer in naher Zukunft ersetzen", wobei nach wie vor für Lernprogramme ein Mißverhältnis zwischen Aufwand und Leistung konstatiert wird; demgegenüber steht eine nur kleine Gruppe von „Technikbegeisterten";
— trotz einer generell befürwortenden Haltung in Publikationen hat eine elementare Auseinandersetzung mit dem Computer als Lern- und Lehrmittel bislang kaum stattgefunden;
— ein allgemeines Einvernehmen darin, „daß der Entwicklungsstand heute noch nicht so weit ist, daß sich große finanzielle Investitionen tatsächlich schon lohnen."

Diese Ursachenanalyse konzentriert sich, vom ersten Punkt abgesehen, auf die Anwenderseite: neben finanziellen Vorbehalten scheint der Mangel an Flexibilität sowohl in praxisbezogener als auch theoretischer Hinsicht einer ansonsten befürworteten Entwicklung entgegenzustehen. Vor allem zur Entstehung der m.E. völlig unbegründeten Befürchtung, Computer würden in absehbarer Zeit die Lehrer verdrängen, hat auch die Produzentenseite entscheidend mit beigetragen, indem sie durch die Ankündigung von High-Tech-Produkten, die selbst bei Fertigstellung nicht in den Unterrichts- und Lernalltag übertragbar waren, die Erwartungen und Hoffnungen an die Computer in unerfüllbare Höhen geschraubt hat.

Die finanziellen Vorbehalte dürften auch nur zum Teil auf die tatsächlichen Kosten zurückzuführen sein. Dem Entwicklungsschub bei den Mikroprozessoren Mitte der achtziger Jahre entsprach eine ähnliche Erscheinung auf dem Sektor der Hardwareausstattung. In diesem Zeitraum wurden mit hohem finanziellen Aufwand Maschinen angeschafft, die damals wegen fehlender Software und langen Einarbeitungszeiten kaum oder überhaupt nicht genutzt werden konnten, inzwischen jedoch nach technischen Gesichtspunkten in fast allen ihren Komponenten so überholt sind, daß nur mehr der Weg in die (hoffentlich umweltfreundliche) Entsorgung bleibt.

1. Softwaremangel und Systemvielfalt

Wesentlicher als alle bisher genannten Gründe dürfte die verwirrende Vielfalt der Systeme auf dem Computermarkt der Verbreitung von

Lernsoftware entgegenstehen. Dies nicht nur deshalb, weil fundierte Systementscheidungen nur mehr von Experten vorgenommen werden können, die alle in Frage kommenden Optionen auch kennen und beherrschen müssen, um einen echten Vergleich durchführen zu können. Der von vornherein nicht zu große Markt wird durch die weitgehend inkompatiblen Systemvarianten in kleinere Segmente aufgeteilt, die kaum ausreichende Absatzchancen für Softwareprodukte bieten. Leistungsfähige Produzenten beschränken sich deshalb entweder auf die breitesten — damit allerdings auch sehr allgemeinen — Bereiche des Sprachlernens wie Vokabel- oder Grammatiktraining, oder sie schrecken durch die von einer geringen Stückzahl hochgetriebenen Preise viele Interessenten ab.[1] Was bleibt, sind die „einsamen Helden", die mit nur mäßiger Aussicht auf Entlohnung und nicht unwesentlich getragen von einer „Technikbegeisterung" in geringem Umfang für neue, einsetzbare Produkte sorgen. Der Mangel an attraktiver Software erleichtert nun wiederum den Verzicht auf Hardwareanschaffungen und hält damit einen Kreislauf in Gang, der einerseits den Mangel verstärkt, andererseits auch die dringend notwendige Diskussion über Stellenwert und Möglichkeiten des Computers beim Fremdsprachenerwerb behindert.
Mit aller Vorsicht und sicher begleitet von heftigem Widerspruch lassen sich heute Tendenzen zu einer Vereinheitlichung auf dem kommerziellen — und damit finanzierbaren — Hard- und Softwaremarkt erkennen. Der Markt ist aufgeteilt zwischen Apple-Macintosh-Produkten auf der einen, und IBM-Hardware auf der anderen Seite, wobei hier die größten Anteile den lediglich IBM-kompatiblen Maschinen zufallen. Die Commodore-Atari-Familie scheint bei fremdsprachlichen Lernprogrammen nicht mehr vertreten zu sein. Diese Aufteilung spiegelt sich im vorhandenen Softwareangebot für DaF wider: von den 46 in die „Dokumentation zu Computerprogrammen für Deutsch als Fremdsprache" (1992, 168-301) aufgenommenen Programmen laufen 18 auf Apple-Macintosh, 17 auf IBM und Kompatiblen, 11 sind für beide Systeme lieferbar. Ob alle Standards auf Dauer nebeneinander existieren oder ob einer die anderen vollständig verdrängen kann, läßt sich derzeit nicht absehen. Apple-Macintosh besaß lange Zeit vor allem wegen der benutzerfreundlichen graphischen Oberfläche bzw. der multimedialen Grundausstattung einen deutlichen Entwicklungsvorsprung vor den hauptsächlich mit dem Betriebssystem DOS ausgestatteten IBM- und kompatiblen Geräten. Der seit einigen Jahren starke Preisverfall bei Produkten der letztgenannten Gruppe, ermöglicht durch hohe Verkaufszahlen, weist auf eine Tendenz in diese Richtung. Mit herbeigeführt haben dürfte diesen Trend die graphische Benutzeroberfläche Windows, die ähnlich der Macintosh-Oberfläche konzipiert ist. Es spricht einiges dafür, daß diese Art der Oberflächengestaltung die DOS-Ebene nach und nach ersetzt bzw. vollständig verdrängt,[2] wodurch sich die Hardwaresysteme äußerlich betrachtet kaum mehr voneinander unterscheiden werden.

Derzeit steht man bei Anschaffungen im Fremdsprachenbereich noch vor der Entscheidung zwischen den genannten Systemen, bei IBM und Kompatiblen zusätzlichen vor derjenigen zwischen DOS und Windows. Der Verbreitung von Windows in Schulen und Universitäten stehen vermutlich noch die vergleichsweise hohen Hardwarevoraussetzungen entgegen (Empfehlungen Microsoft für Version 3.1: Betriebssystem MS-DOS 5.0 oder höher, 80386SX-Prozessor oder höher, 640 KB konventioneller Speicher sowie 2048 KB Erweiterungsspeicher, Festplatte mit mindestens 8MB freiem Speicher, VGA-Farbmonitor, Maus; für Multimedia-Einsatz zusätzlich Audio-Karte, CD-ROM-Laufwerk, ev. Video-Digitizer). Allerdings ist diese Hardware für den Multimedia-Betrieb unentbehrlich, wodurch die finanziellen Vorbehalte gegenüber Windows entfallen. An Multimedia aber, also der Verbindung herkömmlicher Computernutzung mit Ton und Bild bzw. Film, dürfte für die Fremdsprachendidaktik und -methodik kein Weg vorbeiführen. Vor allem die Möglichkeit, gesprochene Sprache funktional in Lernprogramme integrieren zu können, eröffnet einen völlig neuen Bereich von Lernzielen und damit auch neue Wege der Übungsplanung am Computer.

Voraussetzung ist, daß derartige Lernprogramme entstehen und Absatzmöglichkeiten finden. Für Windows sind mittlerweise einige Entwicklungsinstrumente auf dem Markt, die wesentlich zur Vereinfachung der Programmiertätigkeit beitragen. Die Möglichkeiten, die sich durch diese Instrumente, wie z.B. das unten beschriebene Visual Basic, für die Lernprogrammierung eröffnen, könnten der beschriebenen Computerkonfiguration auch in Lehrinstitutionen zum Durchbruch verhelfen und dadurch mittelbar für ein ausreichend breites Angebot an Softwareprodukten sorgen.

Die Auswirkungen der Hardwareausstattungen auf die Art der produzierten Software wurden bereits genannt. Auf der Suche nach Absatzmärkten bearbeiten kommerzielle Anbieter primär den Vokabel- und Grammatikbereich. Dieses Bestreben nach möglichst breiter Verwendbarkeit verursacht eine Reihe von Mängeln, die von Fremdsprachendidaktikern regelmäßig kritisiert werden. Denn die in den Programmen erkennbare Methodik fällt gelegentlich sehr weit hinter den heutigen Stand der Fachdiskussion zurück oder es werden methodische Varianten eingesetzt, die sich als nur unzulänglich auf die Unterweisung mittels Computer übertragbar erweisen.

Eine unmittelbare Auswirkung der angestrebten Einsatzbreite ist ein nahezu willkürlicher, kontextloser Programmaufbau ohne begründete Progression, da jede Verbindung zu einer Systematik Beschränkungen mit sich brächte. Die Übungsmuster basieren im Kern auf einfachen Vergleichen mit ebenso einfachen Richtig-Falsch-Rückmeldungen nach dem ersten Lösungsversuch. Methodische Besonderheiten wirken oft aufgesetzt, z.B. dann, wenn eine Interaktion zwischen Lernenden und Lernmedium durch Anrede mit Vornamen lediglich vorgetäuscht wird.

Technischen Gags, die eingesetzt werden, weil sie einfach zu programmieren sind, fehlt die funktionale Anbindung an Lernziele. Ähnliches gilt für weitgehend funktionslose Wahlmöglichkeiten, die zwar die Leistungsfähigkeit der Computer demonstrieren, Lernenden den Zugang aber unnötig erschweren. Computergerecht sind Cloze-Tests, bei denen jedes x. Wort getilgt wird. Obwohl diese Übungsform für bestimmte Feinlernziele durchaus sinnvoll sein kann, sollte immer auch eine gezielte Lückenwahl ermöglicht werden. Damit kommen aber bereits didaktisch-methodische Überlegungen mit ins Spiel, die nicht mehr von reinen Programmspezialisten bearbeitet werden können. Nicht selten sind dagegen Übungen und Lernspiele, die ohne Computer wesentlich einfacher durchzuführen wären.
Ein oft angesprochenes Manko sind fehlende Alternativen bei Benutzereingaben bzw. insgesamt eine nicht ausreichende Fehlertoleranz. Hierbei handelt es sich um ein Problem, dessen Lösung primär der didaktischen Seite der Programmierung zufällt. Ein auch weitverzweigter Vergleich zwischen Ein- und Vorgaben gekoppelt an eine breite Palette von Programmreaktionen ist eine der Standardaufgaben der Programmierung. Hard- und Softwarekomponenten der heutigen Computergeneration sind in der Lage, eine große Zahl solcher Vergleiche in kaum wahrnehmbarer Zeit durchzuführen. Der kritische Punkt liegt in der Vorphase der Programmierung, nämlich in der Antizipation aller oder zumindest einer Vielzahl denkbarer Reaktionen der Benutzer.
Ein weiterer zentraler Kritikpunkt ist die Benutzerfreundlichkeit, die im wesentlichen von der Oberflächengestaltung abhängt. Befehlsgesteuerte Programme stellen dabei die höchsten Anforderungen an die Benutzer, erfordern eine vergleichsweise lange Einarbeitungsphase bis hin zur routinemäßigen, automatisierten Beherrschung frequenter Befehlsfolgen. Menügesteuerte Software erleichtert den Zugang und sorgt für mehr Transparenz im Programm- und damit Übungsverlauf. Parallel zur Entwicklung der Betriebssysteme dürfte die Zukunft der Lernprogramme in graphischen Oberflächen liegen, die zumindest zum Teil einen intuitiven Zugang zu den Programmstrukturen eröffnen, wodurch die Einarbeitungsphasen noch einmal deutlich verkürzt werden, auf längere Sicht sogar entfallen könnten, wenn es gelänge, einen Bedienungsstandard zu etablieren.
Fehlende Standards der Oberflächengestaltung erschweren nicht nur den Lernenden die Arbeit am Computer. Jeder Programmierer bzw. jede Programmierergruppe entwickelt eigene Konventionen der Kommunikation mit dem Computer, die zu einer verwirrenden Vielfalt an Befehlsfolgen, Tastenbelegungen oder Bildschirmreaktionen führen. Besonders hemmend wirkt sich diese Uneinheitlichkeit bei den sogenannten Autorenprogrammen aus, die den Benutzern, meist den Lehrenden, durch die Eingabe von Texten oder Vokabular grundlegende Möglichkeiten zur Gestaltung eigener, damit auch kontextbezogener Übungen zur Verfügung stellen (vgl. Brücher 1991, 176 zu den

Standardisierungsproblemen, die selbst bei Produkten eines Herstellers auftreten können). Die zögerliche Haltung gegenüber den Autorenprogrammen (Walti 1992, 151f.) dürfte zum Großteil auf den Einarbeitungsaufwand zurückzuführen sein, der durch programmspezifische, singuläre Funktionsweisen verursacht wird.

Hinter den meisten der aufgelisteten Softwareproblemen läßt sich ein grundsätzliches Dilemma der Lernprogrammierung erkennen, dessen Kernpunkt die Kooperation von Fremdsprachendidaktikern mit Programmierungsexperten ist. Da die beiden involvierten Arbeitsbereiche stark erfahrungs- und routineabhängig, damit aber auch zeitintensiv sind, werden die Fälle von Personalunionen immer seltener. Vor allem die nach wie vor rasante Entwicklung im Bereich der Programmierungssoftware gestatten es nicht-professionellen Programmieren kaum mehr, auf dem neuesten Stand der Dinge zu bleiben, ohne sich gleichzeitig zu weit von ihrem Ausgangsgebiet zu entfernen. Qualitativ hochstehende Lernprogramme lassen sich von daher fast nur noch in Zusammenarbeit zwischen Experten aus beiden Lagern erstellen, eine Vorgehensweise, die auch bislang oft gewählt worden ist. Zur Ausnutzung aller Optionen, die sowohl die Fremdsprachendidaktik und -methodik als auch die professionelle Programmierung bieten, ist eine weitgehende Wissensüberschneidung erforderlich. Vielen Programmen für den DaF-Unterricht bzw. allgemein im Umfeld des Fremdsprachenerwerbs ist die nicht vollständige Deckung beider Bereiche deutlich anzumerken.

Als Folge davon weisen die Produkte genau diejenigen Defizite in ihren didaktisch-methodischen Elementen auf, die von der Kritik angesprochen werden: technische Spielereien ohne Lernbezug, funktionslose Optionen, einfache Programmreaktionen. Vor allem die Oberflächengestaltung belegt eine deutliche Anlehnung an herkömmliche Programmierungsaufgaben wie Textverarbeitung oder Datenverwaltung, kaum jedoch spezifisch fremdsprachendidaktische Ansätze. Der Weg der Programmkonzeption verläuft denn auch meist vom Computer in Richtung Didaktik: computergerechte Verfahren werden in den Fremdsprachenbereich übertragen, während der umgekehrte Weg, nämlich die Entwicklung von Programmstrukturen für spezifisch fremdsprachendidaktische Anforderungen kaum eingeschlagen wird. An einigen Beispielen zeigt sich auch eine Fehleinschätzung der Programmierungsmöglichkeiten. Zum Teil werden die Ressourcen der Programmierung nicht ausgeschöpft, erkennbar an einfachen Programmabläufen, aufgesetzten Methodikansätzen und wiederum der Anlehnung an gängige Verwendungszwecke, oder aber überschätzt. Letzteres gilt für alle Versuche der Rubrik „vollständiger Lehrerersatz", aber auch für viele Simulationsprogramme, die zwar technisch aufwendig (und mit hohen Hardwareanforderungen) gestaltet sind und die Möglichkeiten des Computers ausschöpfen, die Lernenden jedoch auf kurze, bruchstückartige Reaktionen restringieren.

Wie erwähnt, kann der Ausweg aus diesem Dilemma, von Einzelfällen abgesehen, nicht in der Ausbildung von Doppelspezialisierungen liegen, also entweder Programmierern mit umfangreicher Unterrichtserfahrung oder Fremdsprachenlehrern mit professionellen Programmierungsfertigkeiten. Notwendig sind vielmehr Entwicklungsverfahren, die eine konsistentere Kooperation mit minimalisierten Reibungsverlusten zwischen Fremdsprachendidaktikern und Programmierern erlauben, und die den Bereich sich überschneidender Kenntnisse möglichst gering halten. Der Fremdsprachendidaktik fällt dabei sicherlich die Verantwortung zu, für das immer noch notwendige gemeinsame Wissen zu sorgen.
Durch eine unter diesem Aspekt ungünstige, zum Teil strukturell immanente Schwerpunktverlagerung scheint die universitäre Computer- und Softwareforschung in absehbaren Zeiträumen hier keine Lösungen anzubieten zu haben. Die Betonung des Forschungsaspekts führt dazu, daß man sich für Entwicklungsarbeiten nicht zuständig fühlt, es sei denn, es handelt sich um „zukunftsweisende", damit immer auch nur langfristig realisierbare Komponenten der Softwarekonzeption. Der Wert dieser Forschung ist unbestritten, auch dann, wenn die Produkte nie den Weg in die praktische Umsetzung finden, da sich im Laufe der Forschungsarbeiten das technische Umfeld (verfügbare Hardware, elementar modifizierte Betriebssysteme, unerwartete Speicherkapazitäten u.a.) grundlegend verändert hat. Sofort einsetzbare Produkte entstehen nur selten, weshalb von der universitären Forschung kaum unmittelbare Impulse zur Qualitätssteigerung von Lernprogrammen ausgehen (vgl. Fechner 1991, 89).
Ein auf diese Weise eng gefaßter Forschungsbegriff behindert die Entstehung einer Fachdiskussion, die von der Perspektive der Adressaten ausgeht. Die wissenschaftliche Behandlung der didaktisch-methodischen Seite des Softwareeinsatzes müßte zumindest teilweise anhand spezifisch dafür entwickelter Programme erfolgen. Um über die Ebene bloßer Programmkritik hinauszukommen, muß die Möglichkeit vorhanden sein, in Lernprogramme eingreifen, Programmabläufe und selbst Programmkonzeptionen verändern zu können. Dazu reichen jedoch die Laufversionen der Programme nicht aus, die bei vielen kommerziellen Lernprogrammen auch noch gegen Einsicht und Veränderungen geschützt sind. Selbst die Entwicklungsversionen mit Quellcode sind für nicht an der primären Programmierung beteiligte Personen wenig hilfreich: der Quellcode ist oft trotz eventuell vorhandener Normierungen oder eingehaltener Programmierungskonventionen für Außenstehende nicht mehr zu durchschauen. Was bleibt, ist die eigenständige Programmierung, die jedoch, wie erwähnt, einen schweren Stand an den Universitäten hat.

2. Objektorientierte Programmierung

Die Situation im Bereich der Lernprogrammierung läßt sich also als eine Art „Teufelskreis" beschreiben: solange qualitativ hochstehende Software

in ausreichendem Umfang fehlt, werden sich potentielle Benutzer bei der Anschaffung geeigneter Hardware zurückhalten; die fehlende Einsatzbreite behindert einerseits die Fachdiskussion, die folglich auch keine wissenschaftlich fundierten Konzepte zur Weiterentwicklung von Lernprogrammen hervorbringt, andererseits reduziert sie die Absatzchancen für Software, damit deren Produktion und so weiter.

Ein Ausbruch aus diesem Regelsystem wird jedoch immer wahrscheinlicher. Der Druck auf den Fremdsprachensektor, sich mit mehr Engagement den Optionen der computergestützten Sprachvermittlung zu stellen, wächst mit der fortschreitenden Entwicklung der Computertechnik. Etwa die Offerten multimedialer Sprachbehandlung — im Unterricht oder bei autonomen Lernformen — müssen in didaktisch-methodische Konzepte gefaßt und für spezifische Einsatzbereiche erschlossen werden. Gleichzeitig mit der Bereitstellung graphischer Benutzeroberflächen, die Computer allgemein besser zugänglich machen, entstehen auch objektorientierte Programmierungssysteme, die wesentlich zur Entschärfung der Verständigungsprobleme zwischen Programmierern und Fremdsprachdidaktikern beitragen können. Am Beispiel von Visual Basic für Windows sollen diese Möglichkeiten detailliert dargestellt werden.

Visual Basic ist ein Programm-Entwicklungssystem, das speziell auf die Erstellung von graphischen Anwendungsprogrammen ausgerichtet ist. Es funktioniert nicht in der linearen Art anderer Systeme, die am Programmanfang beginnen und Schritt für Schritt bis zum Ende ablaufen, sondern die Programme reagieren auf Ereignisse, die von Objekten erkannt werden. Objekt kann jeder beliebige Punkt auf der Bildschirmoberfläche sein oder abgegrenzte Bereiche wie Bilder, Textfelder, Schaltflächen oder Menüpunkte, aber auch die Zeit. Die Liste der Ereignisse reicht von allen Arten der Mausaktionen oder Tastenanschlägen bzw. -kombinationen bis hin zu generell jeder Veränderung auf oder hinter dem Bildschirm.

Das entscheidende Merkmal und gleichzeitig Dreh- und Angelpunkt des Systems ist die graphische Benutzeroberfläche (GUI = *Graphical User Interface).* Über sie laufen alle Ein- und Ausgaben sowie die Steuerung der Programme, die durch den Bezug auf graphische Elemente explizite Handlungsanweisungen für den Benutzer erübrigt. Bildlich — also unabhängig von natürlichen oder technischen Sprachen — können Programmfunktionen erkannt und aktiviert werden, wodurch die Verständigung auf spezifische Handlungskonventionen für weite Bereiche entfällt. Nicht graphisch darstellbare Programmoptionen oder -modifikationen können über Menüs abgerufen werden, also über Listen weitgehend unchiffrierter Handlungsanweisungen, die bei entsprechend klarer Formulierung die mit ihnen verbundenen Programmreaktionen ohne weitere Erklärung erkennen lassen. Möglich, wenngleich aus programmtechnischen Gründen sicher nicht mehr nötig, ist die

schriftliche Eingabe von Befehlen, was jedoch in Sprachlernprogrammen funktional eingesetzt werden kann.
Ist es statthaft, zuerst die Systemvielfalt in Sachen Computer zu beklagen und anschließend die Vorteile eines völlig neuartigen Programmiersystems anzupreisen, in der Absicht, es nicht nur in die Diskussion, sondern auch in die Praxis der Lernprogrammierung einzubringen? Der Entwicklungssprung in Richtung Multimedia, der sich derzeit im PC-Bereich vollzieht, eröffnet nicht nur neue Möglichkeiten, erfordert auch neue Instrumente und Mittel für den Umgang mit diesen Optionen. Die unbefriedigende Situation im Bereich der Lernsoftware, wie sie von vielen Seiten empfunden wird, zwingt ebenso zur Suche nach besseren Lösungen. Mit den graphischen Benutzeroberflächen scheint darüber hinaus ein Entwicklungsendpunkt erreicht worden zu sein, der beginnend bei binären Zahlencodes über diverse Compiler und Interpreter hin zu Betriebssystemen auf einer Ebene angelangt ist, auf der die vom Computerbenutzer geforderte Einarbeitung in die Technik auf ein unvermeidbares Minimum reduziert ist.
Den Rahmen für Visual Basic liefert Windows, eine Kombination aus Betriebssystem und graphischer Benutzeroberfläche. Seine annähernd monopolartige Verbreitung bei den IBM- und kompatiblen Computern (s. Fußnote 2) könnte zu einer Standardisierung der Benutzerstrukturen führen, die sich nur positiv für die naturgemäß international ausgerichtete Fremdsprachenvermittlung auswirken kann. Die Einarbeitung in die Konventionen eines mit Visual Basic entwickelten Lernprogramms bleibt von daher nicht nur von singulärem Nutzen eben für dieses Lernprogramm, sondern erschließt gleichzeitig den Zugang zu einer vermutlich in absehbarer Zeit durchgängig üblichen Herangehensweise an Computer. Der umgekehrte Weg, ausgehend von der Beherrschung der von Windows vorgegebenen Konventionen, erleichtert den Zugang zu Lernprogrammen, die nach denselben Prinzipien funktionieren.
Aus Sicht der Programmierung entlastet ein leistungsfähiges Programmumfeld von vielen, oft sehr zeitaufwendigen Arbeiten. Schon allein das „freihändige" Zeichnen einer Bildschirmoberfläche kann sehr mühevoll sein, vor allem dann, wenn diese Oberfläche auch auf eine Reihe von Ereignissen reagieren soll. Professionell entwickelte Routinen und Problemlösungen können in Visual-Basic-Programme komplikationslos integriert werden und verhindern so Programmierungsarbeiten, die immer wieder das Rad neu erfinden. Die Kommunikation mit anderen Anwendungsprogrammen aus Windows, vor allem den Multimedia-Komponenten, ist über einen flexiblen Datenaustausch mit diesen Programmen gegeben (DDE: *Dynamic Data Exchange)*, die „Außenwelt", also andere Sprachen oder unter anderen Systemvoraussetzungen geschriebene Programme, erschließen *Dynamic-Link-Libraries* (DLL). Diese Verbindungsoptionen erlauben durch den Rückgriff auf vorhandene Anwendungen wie Textverarbeitung, Rechner, Malpro-

gramm, Druckersteuerung u.a. die Entwicklung multifunktionaler, gleichzeitig relativ standardisierter Programme, vermeiden dadurch auch zeitintensive Detailarbeiten oder Erprobungsdurchgänge.
Gegenüber Basic als tragender Programmiersprache bestehen einige Vorbehalte: es sei zu langsam, zu unstrukturiert und zu umständlich. Inwiefern die Geschwindigkeit bei Sprachlernprogrammen relevant ist, kann dahingestellt bleiben, denn bei Visual Basic handelt es sich um eine der vielen Weiterentwicklungen des alten Basic zu einer höheren Programmiersprache, auf die die meisten Einwände von vornherein nicht mehr zutreffen. Die Programmgeschwindigkeit wird heute in erster Linie von den Hardwarekomponenten begrenzt. Besonders der Bildschirmaufbau beansprucht oft so viel Zeit, daß die Abarbeitung des Programmcodes nicht mehr ins Gewicht fällt. Die Steuerung durch Objekte und zugeordnete Ereignisse sorgt in Visual Basic — wie auch in anderen objektorientierten Programmsystemen — fast automatisch und gezwungenermaßen für eine sinnvolle Programmstruktur. Hier ist die Standardisierung des Programmcodes durch die Aufteilung in objekt- und ereignisabhängige Prozeduren weitaus fortgeschrittener als in linearen Systemen, wodurch eine Programmtransparenz entsteht, die kooperative Entwicklungsarbeiten begünstigt. Die Bindung an Windows hat die Entstehung einer Reihe von leistungsstarken Programmierwerkzeugen für frequente Programmroutinen ermöglicht, die ansonsten nur umständlich zu realisieren wären. Schließlich garantieren die Kommunikations- und Importmöglichkeiten über DDE- und DLL-Verbindungen ein nach allen Spezialisierungsseiten hin offenes System.
Wie zu zeigen sein wird, fallen diese Spezialisierungen nicht in das Aufgabengebiet der an der Programmerstellung beteiligten Fremdsprachendidaktiker. Um die ihnen zugedachten Funktionen wahrnehmen zu können, reichen Kenntnisse in bestimmten Teilbereichen aus (s. nächsten Abschnitt). Basic, ursprünglich entwickelt als nicht zweckgebundene Programmiersprache für Anfänger (BASIC: *Beginners Allpurposes Symbolic Instruction Code),* weist auch als Visual Basic noch eine Reihe von Elementen auf, die den Einstieg in die Programmierung erleichtern und somit die vielfach vorhandene Schwellenangst vermindern können. Betrachtet man auf der einen Seite die üblichen Hilfsmittel des Fremdsprachenunterrichts — Texte, Bilder, Ton- und Videoaufnahmen —, auf der anderen Seite die Möglichkeiten der heutigen Computergeneration — Text- und Graphikverarbeitung, Multimedia —, dann erscheint es absehbar zu sein, daß computergestützten Hilfsmitteln in naher Zukunft eine ähnliche Bedeutung zukommt wie derzeit etwa den Lehrbüchern bzw. Lehrwerken. Analog zu den Lehrwerksspezialisten unter den Fremdsprachendidaktikern, die nie ausgebildete Drucker und selten Graphiker sind, entsteht dadurch ein Bedarf an Lehrpersonal mit Computererfahrung, das sich in enger Anbindung an ihr Fach mit den rechnergestützten Unterrichtsmethoden beschäftigt. Computer- bzw. Programmierkenntnisse werden dadurch zumindest in Ansätzen Teil der

Lehreraus- oder -fortbildung. Ein auf Basic fundiertes Programmiersystem könnte in diesem Kontext von Vorteil sein. Neben der vergleichsweise leichten Einstiegsmöglichkeit wurde und wird in Schulen, wenn Programmierung überhaupt auf dem Stundenplan steht, überwiegend in die Arbeit mit Basic eingeführt. Bei vielen Studenten sind heute von daher bereits Kenntnisse vorhanden, auf die in der Lehrerbildung zurückgegriffen bzw. aufgebaut werden könnte.
Selbst ohne Ambitionen zur Entwicklung von Lehrmaterialien eröffnen bereits geringe bis mittlere Programmierungsfertigkeiten einen breiten Anwendungsbereich im Fremdsprachenunterricht, wenn die Isolierung von Einzelprogrammen durchbrochen wird. Unter Windows erstellte Autorenprogramme, z.B. zur Anfertigung von Cloze-Tests, erlauben den Import von Texten aus allen gängigen Textverarbeitungen, ersparen damit nicht nur doppelte Tipparbeit, sondern auch wieder die Einarbeitung in unterschiedliche Systeme. Auf ähnliche Weise können mit einem geringen Aufwand an Zeit und Know-how selbstkorrigierende Übungen zu speziellen Lernthemen oder individuellen Lernproblemen erstellt werden.
Von diesen lediglich prognostizierbaren Verwendungstendenzen abgesehen, liegt der entscheidende Vorteil der objektorientierten Programmierung in der Trennung von Oberflächengestaltung und Programmcode. Die methodisch-didaktischen Elemente von Lernprogrammen treten fast ausschließlich an der Benutzeroberfläche auf, also in den Programmteilen, die Lernende auch zu Gesicht bekommen. Das bedeutet, daß mit der Festlegung der Oberflächen eines Programms diejenigen Arbeiten bereits weitgehend erledigt sind, die in den Aufgabenbereich der Fremdsprachenexperten fallen.
Die Entwicklung eines Lernprogramms mit einem objektorientierten Programmierungssystem läuft üblicherweise in folgenden Schritten ab:

1. Programmidee (Lerngegenstand, Lernziele)
2. Programmkonzeption (Übungsverlauf, Hilfestellungen)
3. Oberflächengestaltung (Übertragung der Programmkonzeption auf den Bildschirm)
4. Entwicklung des Programmcodes
5. Technische und didaktisch-methodische Erprobung
6. Nachbearbeitung und Erstellung der Endversion.

Je nach Aufgabenstellung sind für die Codeentwicklung mehr oder weniger spezialisierte Programmierungsfertigkeiten notwendig, ebenso für die Phasen der technischen Erprobung sowie der Fertigstellung der Endversion, beides Vorgänge zur Optimierung des Programmcodes. Für die Schritte 1 bis 3 lassen sich die Programmierungskenntnisse in etwa so beschreiben (vgl. auch das Umsetzungsbeispiel im nächsten Abschnitt): Schritte 1 und 2 — Programmidee und -konzeption: erforderlich ist eine allgemeine, nicht in Details gehende Kenntnis des jeweils aktuellen Stands der Programmiertechnik, um sowohl eine Unter- wie auch Überforderung der technischen Seite zu vermeiden. Notwendig ist vor

allem, von Seiten der Didaktik alle für die Lösung der spezifischen Programmidee vorhandenen technischen Möglichkeiten anzufordern und die Reibungsverluste in der Kommunikation mit den Programmierungsexperten zu minimieren. Nicht benötigt werden aktive Programmierungsfertigkeiten.
Schritt 3 — Oberflächengestaltung: nötig sind aktive Programmierungskenntnisse bezogen auf die graphischen Elemente der Benutzeroberfläche, über die die Steuerung der Lern- und Übungsvorgänge erfolgt. Neben der leicht erlernbaren Handhabung der Instrumente zur Erzeugung statischer Bildschirmelemente ist der kreative Umgang mit Farbgestaltung und Elementpositionierung notwendig, bei bewegbaren bzw. sich selbst bewegenden Elementen sind zusätzlich grundlegende Mathematikkenntnisse notwendig.

3. Umsetzungsbeispiel zur Oberflächengestaltung

Bei einer Kooperation zwischen Fremdsprachendidaktikern und Programmierungsexperten reduziert sich folglich der Bereich gemeinsamer Kenntnisse im wesentlichen auf die Oberflächengestaltung, verbunden mit der Beschreibung der Reaktionen, die auf bestimmte Ereignisse hin ausgelöst werden sollen. In welchem Umfang dabei programmtechnische Kenntnisse auf Seiten der Fremdsprachendidaktiker unentbehrlich sind, kann an einem einfachen Umsetzungsbeispiel veranschaulicht werden.
Ausgangspunkt der Programmidee in diesem Beispiel sind die Schwierigkeiten von DaF-Lernern mit der im Deutschen üblichen Zahlenartikulation, speziell der Reihenfolge von Zehner- und Einerstellen sowie der Aufbau nach Tausendergruppen. Lerngegenstand ist also die Zahlenaussprache, Lernziel die Beherrschung der besonderen Wortstellung zur Verbesserung des Hörverstehens und der Sprechfertigkeiten. Da es sich bei den Schwierigkeiten mit diesem Sprachphänomen nicht um ein kognitiv-intellektuelles Problem handelt — die Regeln sind leicht verständlich —, entsteht bei Fehlern kein besonderer Erklärungsbedarf, stattdessen sollte eine hohe Übungsfrequenz erreicht werden.
Von daher eignet sich eine spielerische Übungsform mit hochfrequenten Übungsschritten, hier in Anlehnung an die Regeln von sog. Memory-Spielen. Die Übungszahlen werden sowohl in Worten als auch in Ziffern vorgegeben, auf paarweise gekennzeichnete Felder in zwei Blöcken verteilt.
Zu Spielbeginn sind alle Felder abgedeckt, aufgedeckt und gelesen werden kann jeweils nur ein Feld pro Block. Wird in beiden Blöcken dieselbe Zahl aufgedeckt, bleiben die Felder geöffnet, ansonsten wird das vorher aktivierte Feld wieder verdeckt. Der Zahlenbereich soll vom Benutzer selbst gewählt werden, um den Schwierigkeitsgrad der Übung individuell steuern oder besonders problematische Zahlen üben zu

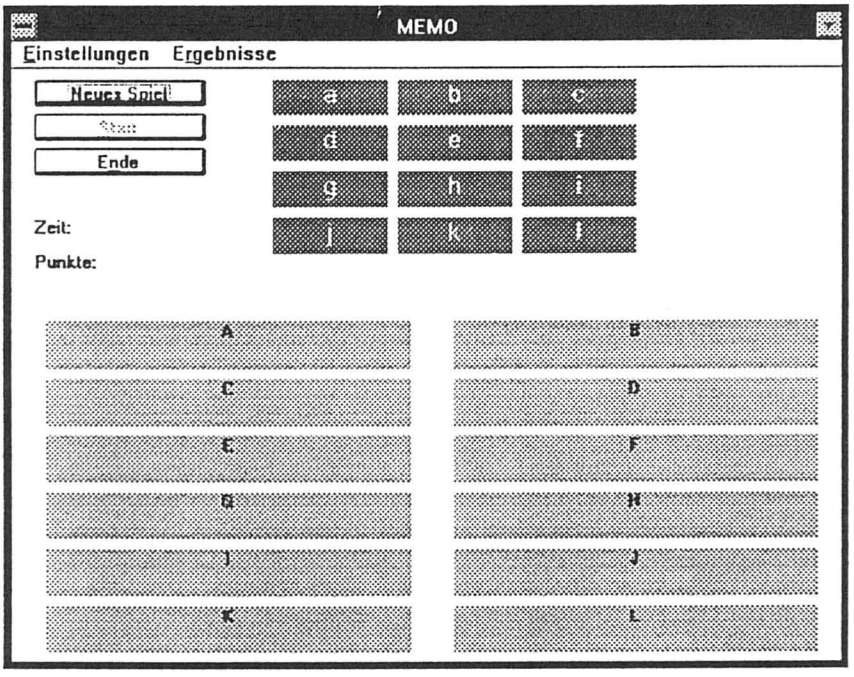

Abbildung 1:
Oberfläche des Memo-Spiels nach Programmstart

können; innerhalb dieses Bereichs werden die Zahlen zufällig vom Programm bestimmt. Die Spiele sind sowohl mit als auch ohne Zeitfaktor möglich, allerdings erhöht sich die Anzahl der zu erreichenden Punkte, wenn die Zeitmessung aktiviert wird. Diese wie auch die Ergebnismeldungen sollen primär motivierend wirken. Das Spielergebnis kann im Vergleich zu früheren in einer Ergebnisstatistik abgerufen werden, oder aber in Relation zu anderen Spielern, falls diese ihre Werte dort abgespeichert haben. Eine automatisch ablaufende Pausengraphik soll über die dadurch am Spielende entstehende Unterbrechung die Übungsphasen etwas strukturieren.

Obwohl Zahlen in Worten in realen Verwendungszusammenhängen nur selten ausgeschrieben auftreten, ist es mit dem Übungscharakter vereinbar, sie im Programm in schriftlicher Form und in Kontrast zur Schreibung mit Ziffern zu präsentieren. Zum einen können die Lernenden durch leises oder lautes Nachsprechen selbst die Verbindung zu unmittelbaren Hör- und Sprechleistungen herstellen, zum anderen werden durch die höhere Genauigkeit beim Lesen stützende, verfestigende Übungseffekte erzielt.

Aus dieser Programmkonzeption ergeben sich folgende Oberflächenelemente: Zahlenfelder, im Beispiel je zwölf in zwei Blöcken, Auswahllisten für die Optionen und Ergebnisspeicherung sowie Start- und Ende-

Befehle. Die Elemente sollen so gestaltet und positioniert werden, daß der Spiel- und damit der Übungsverlauf intuitiv erfaßt werden kann. Als Ausgangsschablone bietet sich in diesem Fall eine typische Windows-oberfläche mit Titel- und Menüleiste über der Arbeitsfläche an (vgl. Abb. 1). Die Titelleiste enthält neben dem Namen des Programmteils eine Kontrollbox in der linken oberen Ecke, die einen zusätzlichen Endebefehl enthält. Am rechten Rand der Titelseite erlaubt ein Minimierungssymbol die Verkleinerung der Oberfläche zu einem Icon, also einer kleinen Graphik, über die sich das damit gekennzeichnete Programm wieder aktivieren läßt. Über dieses Symbol kann zwischenzeitlich zu anderen Anwendungen gewechselt werden, ohne daß die Einstellungen des laufenden Programms verändert werden.

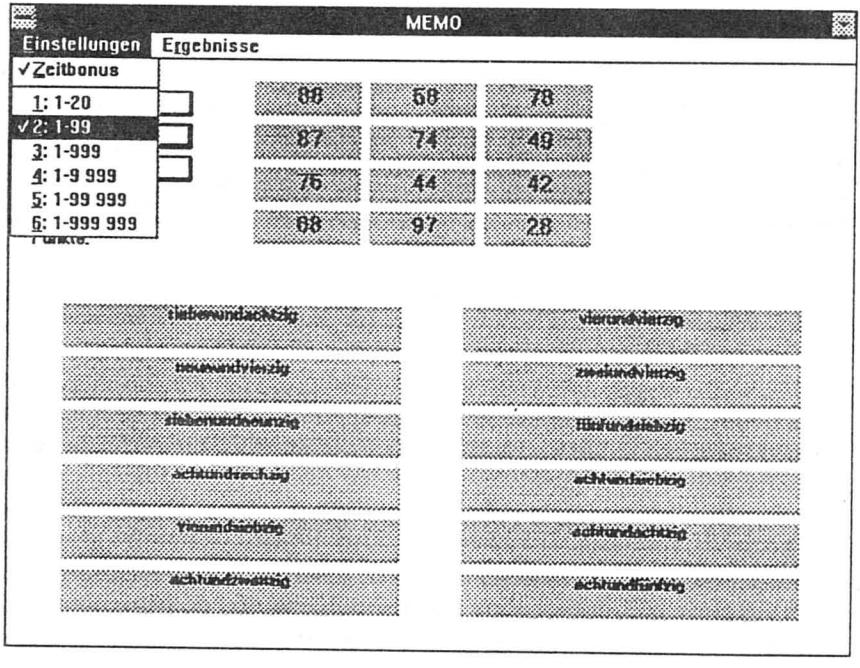

Abbildung 2:
Spielanfang mit geöffnetem Menü „Einstellungen"

In der Menüzeile darunter stehen die beiden Menüs „Einstellungen" und „Ergebnisse" (Abb. 1, geöffnet in Abb. 2 und 3). Ausgehend von der Überlegung, daß die Optionen, die hier zur Verfügung gestellt werden, seltener aktiviert werden als die Startbefehle, hat die Zusammenfassung dieser Punkte in normalerweise nicht sichtbaren Menügruppen den Vorteil, die Bildschirmoberfläche zu entlasten und damit übersichtlich zu halten. Die bei jedem Spiel benötigten Befehle für neue Zahlen bzw. den Spielbeginn hingegen sind permanent auf der Oberfläche sichtbar, um

den Zugriff auf sie zu vereinfachen. Da der Ausstieg aus Programmen über verdeckte Befehle — wie etwa derjenige in der Kontrollbox — oder über bestimmte Tasten nicht ohne Zusatzinformation erkennbar ist, erleichtert eine zwar selten benutzte, aber dennoch immer sichtbare Befehlsschaltfläche mit der Aufschrift „Ende" diesen Vorgang (zu Problemen beim Programmausstieg vgl. Brücher 1991, 177).
Mittels dieser graphischen Obeflächenelemente ist der Übungsverlauf bereits weitgehend festgelegt. Nach Programmstart soll der Benutzer zunächst die Fläche mit der Aufschrift „Neues Spiel" aktivieren. Soweit dies nicht schon allein durch die Benennung deutlich ist, liefert der Fokus noch einen Hinweis, der sich zu Beginn auf dieser Fläche befindet. Anschließend verschwinden die Buchstaben auf den Feldern und sichtbar werden die Zahlen, oben in Ziffern, unten in Worten (Abb. 2). Der gleichzeitig stattfindende Farbtonwechsel von dunkel nach hell soll die Aktivierung der Felder hervorheben. Voreingestellt ist der Zahlenbereich von 1 bis 99, die Zeitoption ist nicht aktiviert. Erst nach dem Betätigen des Startbefehls werden die Zahlen verdeckt und wieder durch die Kennbuchstaben ersetzt. Die Spieler können also selbst bestimmen, wieviel Zeit sie sich für das Einprägen der Position der Zahlenpaare nehmen. Aufgedeckt werden Felder entweder durch einfaches Anklicken mit der Maus oder durch das Drücken der betreffenden Buchstabentaste.

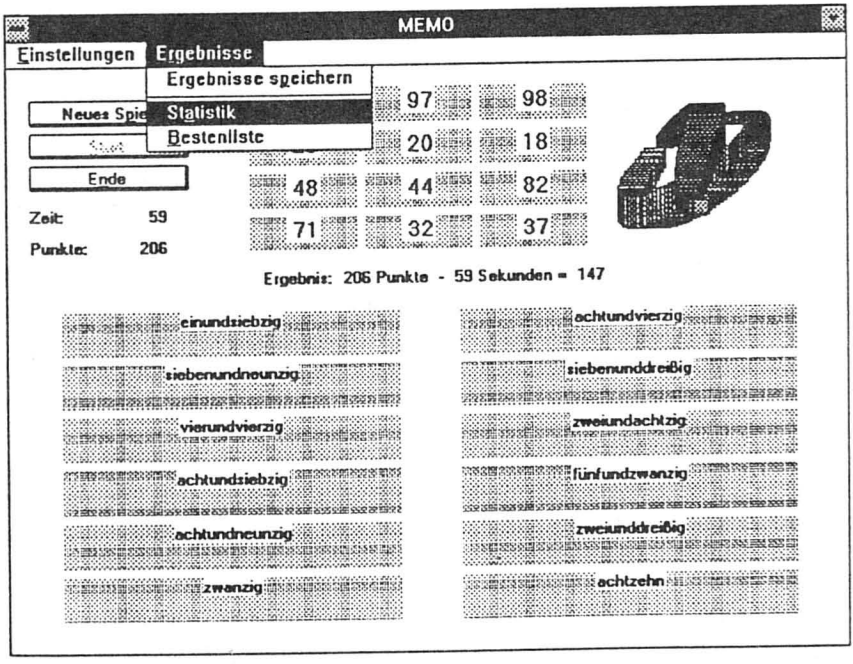

Abbildung 3:
Spielende mit geöffnetem Menü „Ergebnisse"

Die Spielzeit wird unabhängig von der gewählten Zeitoption angezeigt. Ständig aktualisiert wird der Punktestand. Je nach gewähltem Bereich gibt es für jede korrekte Aufdeckung eine festgelegte Anzahl von Punkten, für jede Nichtübereinstimmung wird ein Punkt abgezogen. Sind alle Felder geöffnet, dann erscheint die Pausengraphik (aus Kofler 1991) — sich kontinuierlich in Position, Größe und Farbe verändernde Rechtecke —, der Fokus wird zurück auf die Befehlsschaltfläche „Neues Spiel" gesetzt und der Startbefehl deaktiviert, da erst wieder mit neuen Zahlen gespielt werden kann (Abb. 3).
Veränderungen über die Menüs können jederzeit vorgenommen werden, wirken sich jedoch meist erst nach einem erneuten Spielstart aus. Ist die Zeitmessung aktiviert, erkennbar an einem Haken vor dem Menüeintrag, werden zwar die für den Spieldurchgang benötigten Sekunden vom Gesamtpunktestand abgezogen, gleichzeitig verdoppelt sich aber die Anzahl der möglichen Punkte, so daß dieser Spielmodus im Regelfall zu einem höheren Ergebnis führt.
Sollen die individuellen Spielergebnisse gespeichert werden, muß zunächst der Menüpunkt „Ergebnisse speichern" aktiviert und ein Kennwort eingegeben werden. Wurden unter demselben Kennwort bereits Ergebnisse gespeichert, dann erscheinen bei Aktivierung des Menüpunkts „Statistik" diese zusammen mit dem aktuellen Resultat als einfaches Balkendiagramm. Hinter dem Menüeintrag „Bestenliste" steckt eine nach Höchstpunktzahlen geordnete Auflistung, in der Spieler ihre Ergebnisse unter ihrem Namen oder einer Chiffre eintragen können. Funktion beider Ergebnislisten ist weniger die exakte Rückmeldung zu den Übungsergebnissen, die je nach gewähltem Bereich und Modus sehr unterschiedlich ausfallen, sondern mehr ein vielleicht motivierender Spielanreiz zur Steigerung der Übungstätigkeit.
Sind Programmidee und -konzeption entwickelt, die Programmoberflächen erstellt, dann sind die Aufgaben der an der Programmentwicklung beteiligten Fremdsprachendidaktiker erfüllt. Eine Beschreibung des Programmverlaufs, wie sie hier vorgenommen wurde, kann nun zusammen mit den Oberflächenentwürfen an den oder die Programmierer weitergegeben werden. Sie fügen den Programmcode an, der die gewünschten Prozeduren mit den Oberflächenereignissen verbindet. Anschließend kann das Programm in die technische und didaktisch-methodische Erprobung.
In realen Entwicklungsvorgängen werden die Kooperationspartner nicht nur einmal miteinander Kontakt aufnehmen, sondern sich von Anfang an absprechen und über Möglichkeiten und Schwierigkeiten informieren. Auch verfeinern sich Programmkonzeptionen oftmals noch bei der Übertragung auf den Computer: neue Optionen können hinzukommen, ursprünglich geplante entfallen oder entscheidend modifiziert werden. Von Vorteil für die didaktisch-methodische Entwicklung wären auch grundlegende Kenntnisse der Fremdsprachendidaktiker in Bezug auf die Programmierung einfacher Oberflächenreaktionen. Auf diese Weise

kann bereits in den ersten Phasen der Entwicklung die Benutzerfreundlichkeit, die Erschließbarkeit des Programmablaufs oder generell die optische Erscheinung des Programms kontrolliert werden. Im beschriebenen Beispiel wäre dies etwa das Aufdecken der Zahlenfelder, das durch folgende Programmzeilen gesteuert wird:

```
If Worte(X).BackColor = DUNKELBLAU Then
        Worte(X).BackColor = WEISS
        Worte(X).Caption = InWorten(X)
End If
If Ziffer(X).BackColor = DUNKELGELB Then
        Ziffer(X).BackColor = WEISS
        Ziffer(X).Caption = InZiffern(X)
End If
```

Erläuterung: Die Zahlenfelder, auf denen die Zahlen in Worten erscheinen, haben den Kontrollnamen „Worte" plus einer Indexzahl von 1 bis 12, ebenso die Ziffernfelder mit dem Kontrollnamen „Ziffer". Wird ein nicht geöffnetes Feld, also mit dunkler Hintergrundfarbe, angesprochen — durch Mausklick oder Drücken der entsprechenden Buchstabentaste —, dann liefert diese Aktion die Indexzahl unter X an den Programmteil. Daraufhin wechselt die Hintergrundfarbe des Feldes von Dunkelblau bzw. Dunkelgelb nach Weiß und als Aufschrift (= Caption) erscheint die Zahl, die unter derselben Indexnummer in der Variablen „InWorten" für die Wortfelder, in der Variablen „InZiffern" für die Ziffernfelder gespeichert ist.
Das dazu nötige Programmierungsknowhow kann schnell erlernt werden. Trotz dieser partiellen Überschneidung bleibt die klare Grenzziehung zwischen Oberflächengestaltung und Programmcode erhalten. Die Programmierer liefern die Prozeduren, die den Programmablauf gemäß den Vorgaben steuern. Die Einzelprozeduren sind in weitgehend selbständigen, abgeschlossenen Modulen enthalten, die in verwandten Programmkonzepten beliebig oft einsetzbar sind. So etwa kann das Modul, das Ziffern in Worte umwandelt, an verschiedene Stellen und in verschiedene Programme übertragen werden. An diesem Beispiel wird auch deutlich, daß die Programmiererseite bei der Entwicklung eines sprachbezogenen Lernprogramms zwar gelegentlich mit der linguistischen Seite der Sprachbehandlung in Berührung kommt, nicht dagegen mit den pädagogischen Aspekten. Beim Erstellen dieses Programmteils müssen die Regeln der Zahlenartikulation beachtet werden, die Einsatzstelle im Programm bzw. im Übungsverlauf spielt bei diesem Vorgang keine Rolle.

Auf der Basis eines objektorientierten Programmierungssystems sind also Fremdsprachendidaktiker und Programmierungsexperten in der Lage, ihr Fachwissen in der Kooperation voll zur Geltung zu bringen, ohne gleichzeitig weit in das Fachgebiet des Partners eindringen zu müssen. Dadurch läßt sich sowohl die didaktisch-methodische als auch die technische Qualität von Lernprogrammen auf ein hohes, aktuell realisierbares Niveau bringen. Vielen Kritikpunkten, die derzeit existierenden Sprachlernprogrammen vorgehalten werden und die im wesentlichen aus Kommunikationsproblemen zwischen den beiden involvierten Disziplinen resultieren, kann damit von vornherein begegnet werden.

4. Perspektiven

Die Beschreibung des Zahlenspiels Memo exemplifiziert nur einen winzigen Ausschnitt der Möglichkeiten, die sich für die computergestützte Umsetzung didaktisch-methodischer Konzepte mit Hilfe der Instrumente ergeben, wie sie durch die Kombination einer graphischen Benutzeroberfläche mit einer objektorientierten Programmierung zur Verfügung stehen. Nur in Ansätzen dokumentierbar ist die Schnelligkeit, mit der sich auf diese Weise Vorhaben realisieren lassen, da hier eine determinierende Abhängigkeit von individuellen bzw. kooperativen Voraussetzungen gegeben ist. Festzustellen bleibt, daß sich ein gangbarer Weg eröffnet, auf dem dem Softwaremangel im DaF-Bereich begegnet werden kann. Dessen Beseitigung kann eine hinreichende Vorausleistung sein, um die theoretische sowie praktische Auseinandersetzung mit dem Lernmedium Computer zu fördern bzw. erst richtig in Gang zu bringen. Neben der vereinfachten Zugänglichkeit sowohl zu den Lern- als auch Autorenprogrammen über eine standardisierte Benutzeroberfläche steckt ein erhebliches Potential im modularen Charakter der objektorientierten Programmierung. Er gestattet eine sinnvolle Aufgabenverteilung zwischen Fremdsprachendidaktik und professioneller Programmierung, die zu lern- und computertechnisch anspruchsvollen Produkten führt. Die konstitutionell kooperative Arbeitsanlage erlaubt nahtlose Erweiterungen um weitere Spezifikationen. Fachlich orientiertem Fremdsprachenunterricht etwa, der in bestimmten Punkten und Entwicklungsphasen auf die Mitarbeit von Experten des betroffenen Faches angewiesen ist, kommt diese Konstellation entgegen. Fachspezialisten können punktuell und zeitlich begrenzt in die Kooperation einbezogen werden (vgl. Fach-Overhoff 1990). Ihr Know-how läßt sich in entsprechend ausgerichtete Programme einbinden, die wiederum von DaF-Experten ohne spezifische Fachkenntnisse an passender Stelle in Lernprozesse integriert werden. Schließlich ermöglicht der Modulcharakter nicht nur eine selbständige Beschäftigung mit Didaktik und Methodik losgelöst von primär programmtechnischen Fragestellungen.

Die linguistischen Aspekte der Sprachverarbeitung auf dem Computer werden ebenfalls von den didaktisch-methodischen getrennt, so daß beide Bereiche zunächst unabhängig voneinander weitergeführt werden können. Zeitintensive Problemlösungen auf der einen Seite führen dadurch nicht mehr zwangsläufig zu Stockungen auf der anderen.
Für einige der von Walti (1992, 147f.) angeführten Punkte, die eingangs zitiert wurden, finden sich Lösungsansätze. Den gewachsenen Anforderungen an Programme stehen leistungsstarke Instrumente gegenüber, die den Entwicklungsaufwand reduzieren und dadurch die Relation zur Leistung der Programme verbessern. Die Arbeits- und Zugangserleichterungen einer graphischen Oberfläche müßten genügend Anreiz für das Umsteigen von anderen Systemen bieten und die Perspektiven, die sich aus der allgemeinen Verbreitung und in Folge davon der Preisgestaltung multimedialer Komponenten ergeben, lassen kaum noch Zweifel an der Berechtigung von Investitionen zu.
Ein wichtiges Motiv zur Integration von Computern und Computerwissen in den DaF-Unterricht blieb bislang unerwähnt. Die Stellung des Deutschen als Fremdsprache wird wesentlich bestimmt von seinem Gebrauchswert als Mittel des Technologietransfers (vgl. z.B. zur VR China Hess 1992, 242-246). Computergestützter Fremdsprachenerwerb kommt dieser Nachfrage durch Übereinstimmung in Form und Gehalt entgegen. Durch die Einführung des Arbeitsmediums Computer in den DaF-Unterricht, oder zu seiner Vor- und Nachbereitung, wird der Technologietransfer bereits zu einem Teil des Unterrichts selbst.

Anmerkungen

[1] z.B. Sprachkurse Englisch der Telemedia (Bertelsmann AG) mit Preisen zwischen 8800 und 11800 DM.

[2] z.B. wurden von der Windows Version 3.0 weltweit 12 Mio. Stück verkauft; derzeit arbeitet die Herstellerfirma Microsoft am System Windows NT, das nicht mehr wie die bisherigen Windows-Versionen auf DOS angewiesen ist.

Literatur

ARBEITSKREIS COMPUTEREINSATZ IN DEN MODERNEN FREMDSPRACHEN (1990): Software für moderne Fremdsprachen. Augsburg.

ARBEITSKREIS DER SPRACHZENTREN, SPRACHLEHRINSTITUTE UND FREMDSPRACHENINSTITUTE (1989): Workshop „Computereinsatz im Fremdsprachenunterricht". *Fremdsprachen und Hochschule* 26.

BRÜCHER, K.H. (1991): Autorenprogramme im computergestützten Fremdsprachenunterricht. *Deutsch als Fremdsprache* 3, 175-180.

FACH-OVERHOFF, M. (1990): Computerunterstützte interaktive Lernprogramme für den naturwissenschaftlichen Unterricht: zur Konzeption und Wirkung der Einführung eines phänomennahen Begriffs „Atombindung". Frankfurt/M. u.a.

FECHNER, J. (1991): Eingaben des Lernenden und ihre Verarbeitung in Computerprogrammen für den Fremdsprachenunterricht. *Zielsprache Deutsch* 2, 88-92.

GÖTTMANN, H. (1991): Neue Wege mit neuen Medien: II. Internationales wissenschaftliches Kolloquium „Computer im Fremdsprachenunterricht" in Leipzig, 1991. *Info DaF* 3, 461-463.

HESS, H.-W. (1992): „Die Kunst des Drachentötens". Zur Situation von Deutsch als Fremdsprache in der Volksrepublik China. München.

KOFLER, M. (1991): Windows Programmierung mit Visual Basic. Bonn u.a.

MICROSOFT CORPORATION (Hg.) (1991): Microsoft Visual Basic Programmiererhandbuch. München.

RÄKEL, H.-H.S. & STEINFELD, T. (Hg.) (1992): Dokumentation zum computergestützten Unterricht in Deutsch als Fremdsprache. Bearbeitet v. L. Desjardins, B. Martin und K. Walti. *Info DaF* 2.

SEEL, N.M. (1992): Computer im Unterricht — Auf dem Weg zu multimedialen Lernumgebungen. *Unterrichtswissenschaft* 20, 73-82.

KÄMPER van den- BOOGAART, M. (1992): Deutschlernen mit dem Personalcomputer. *Lernen in Deutschland* 1, 62-74.

WALTI, K. (1992): Computereinsatz im Unterricht Deutsch als Fremdsprache. *Info DaF* 2, 146-167.

Anschrift des Autors:
Dr. Haymo Mitschian, Mittelbruchzeile 89, 13409 Berlin.