Persona: Ein anthropomorpher Präsentationsagent für Internet-Anwendungen

Dissertation
zur Erlangung des Grades Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultät I der
Universität des Saarlandes

vorgelegt von

Jochen Müller

Saarbrücken
2000

Dekan:	Prof. Dr. R. Schulze-Pillot-Ziemen
Berichterstatter:	Prof. Dr. Dr.h.c. W.Wahlster
	Prof. Dr. H. Hagen
Tag des Kolloquiums:	25. September 2000

Danksagung

Die vorliegende Arbeit entstand im Rahmen der vom BMBF geförderten Projekte PPP und AIA.

Mein besonderer Dank gilt Herrn Prof. Dr. Dr.h.c. Wolfgang Wahlster, der die Arbeit betreute und der es mir ermöglichte, im Rahmen der Projektarbeit diese Arbeit zu verfassen.

Herrn Prof. Dr. Hans Hagen danke ich für seine Arbeit als Zweitgutachter.

Der Projektleiterin der Projekte PPP und AIA, Frau Dr. Elisabeth André, möchte ich für die umfangreichen Anregungen und die gute Unterstützung danken.

Herr Peter Rist und Herr Bernhard Kirsch haben die Bitmaps für den Präsentationsagent PPP-Persona erstellt.

Ein weiterer Dank gilt allen Mitarbeitern und Studenten der Projekte PPP und AIA, insbesondere Stephan Baldes, Frank Biringer, Dr. Andreas Butz, Patrick Gebhard, Gerd Herzog, Dr. Antonio Krüger, Dr. Susanne van Mulken, Stefan Neurohr, Renato Orsini, Dr. Thomas Rist und Valentin Tschernomas.

Meinen Eltern danke ich für die Unterstützung während meiner Studienzeit.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Die Elemente eines Computersystems, mit denen der Benutzer den engsten Kontakt hat, sind die Benutzeroberflächen. Sie stellen dem Benutzer die Funktionalität des gesamten Programms zur Verfügung. Benutzerschnittstellen entscheiden häufig über die Akzeptanz des gesamten Programmsystems. Der zunehmende Gebrauch von Computersystemen in allen Teilbereichen des täglichen Lebens führt einerseits dazu, Rechner zur Bewältigung höchst komplexer Aufgabenstellungen einzusetzen, andererseits müssen aber immer mehr computerunerfahrene Benutzer mit den Benutzerschnittstellen solcher Programme fertig werden. Gerade Anfänger schrecken oft vor den Schwierigkeiten der Computerhandhabung zurück, sie koppeln sich von der Entwicklung der Informationsgesellschaft ab.

Neben tatsächlichen Schwierigkeiten, wie z.B. der Auswahl eines korrekten Befehls zur Programmsteuerung, gibt es teilweise auch eine psychologische Hemmschwelle, die es für den Benutzer zu überwinden gilt.

Im Folgenden soll ein System realisiert werden, das die geschilderten Probleme bewältigen helfen soll. Um den Benutzer durch eine Hypermedia-Präsentation zu führen, wird eine animierte Figur, Persona, eingesetzt. Persona ist in ein komplexes Präsentationsystem integriert, das benutzeradaptive Präsentationen generiert. Der Präsentationsagent geleitet den Betrachter durch die Präsentation und repräsentiert dem Benutzer gegenüber das gesamte Präsentationssystem. Persona ist als Ansprechpartner für den Betrachter gedacht, der sich bei Problemen von diesem helfen lassen kann. Aufgaben, die der Benutzer an den Agenten stellt, übermittelt dieser unmittelbar an das Präsentationssystem. Neben dem Einsatz in WWW-Applikationen kann der Agent auch in normalen Desktop-Applikationen eingesetzt werden.

Das Persona-System wird momentan in verschiedenen Forschungs- und Industrieprojekten, die das Deutsche Forschungszentrum für Künstliche Intelligenz durchführt, eingesetzt:

PPP: Desktop-Präsentationssystem (Abschnitt 7.1), Forschungsprojekt

AIA: Präsentationssystem für das World Wide Web (Abschnitt 7.2), Forschungsprojekt

Presence: Präsentationssystem für das DFKI, Forschungsprojekt

VVA: Virtuelle Versicherungsagenten, Industrieprojekt (DaimlerChrysler AG)

ShopAss: Einsatz als „Shopping-Assistent “ , Industrieprojekt (Otto Versandhaus)

Das lateinische Wort „persona“ bezeichnet die Rolle, die jemand unabhängig von der Konstitution der eigenen Person im Beruf, im Alltag, im gesellschaftlichen Ganzen wahrnimmt. Eine andere Deutung ist die Rolle (das Gesicht, die Maske), die im Theater gespielt wird (Metapher Welttheater) [Flashar, 1998]. Persona dient als „Maske“ des Präsentationssystems, es verbirgt sich hinter ihr [Wahlster, 1993a].

Neben der Bezeichnung Persona wurde auch die Bezeichnung „Wichtel“ während der Entwicklungszeit verwendet. Das althochdeutsche Wort „Wicht“ bezeichnete die „Sache“, gleichzeitig aber auch den Zwerg. Zwerge sind im Märchen oft Helfer der Menschen [Krischke, 1997], so dass diese Bezeichnung auch für einen Helfer bei der Programmbedienung bzw. einen Agenten gerechtfertigt ist.

Wie in Abschnitt 1.2.1 dargestellt wird, entspricht der Einsatz von Präsentationsagenten in Benutzeroberflächen zu Präsentationssystemen der konsequenten Weiterentwicklung von Benutzerschnittstellen. Zunehmend wird die Manipulation von Objekten auf dem virtuellen Schreibtisch durch Delegation von Aufgaben an das Computersystem abgelöst.

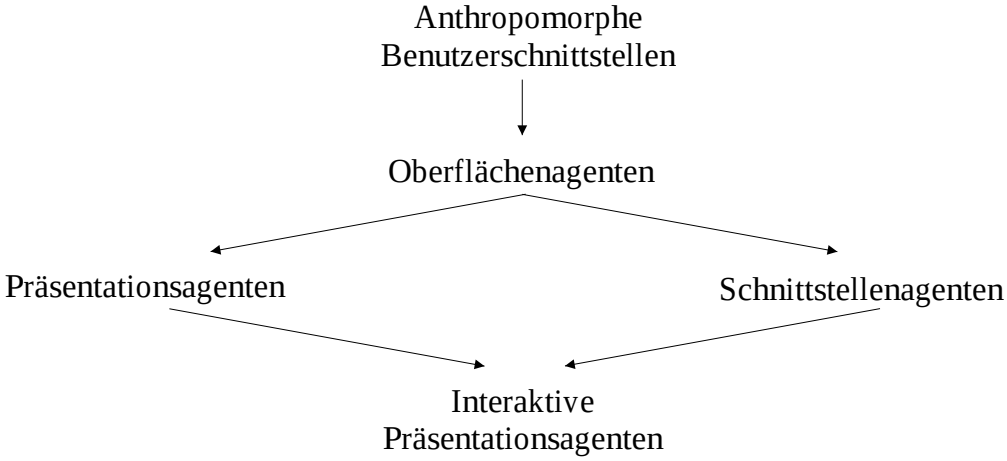


Abbildung 1.1: Einordnung von Präsentationsagenten

Eine anthropomorphe Benutzerschnittstelle ist eine Benutzerschnittstelle, die aus einer im weitesten Sinne menschenähnlichen Figur besteht.

Obwohl anthropomorphe Benutzerschnittstellen viele Ausprägungen besitzen (z.B. Auskunftsterminals, deren äußere Form einer menschlichen Figur nachempfunden ist), sollen hier nur Präsentationsagenten berücksichtigt werden. Da der in dieser Arbeit entwickelte Präsentationsagent auch als Benutzeroberfläche zu dem eingesetzten Präsentationssystem dient, werden auch teilweise Funktionen von Schnittstellenagenten benötigt. Trotzdem steht bei diesem nicht die Interaktivität im Vordergrund, sondern die umfangreichen Präsentationsmöglichkeiten.

1.1 Aufgaben personifizierter Präsentationsagenten

Ein Präsentationsagent, der in einer multimedialen Präsentation eingesetzt wird, muss die unterschiedlichsten Aufgaben erfüllen:

Präsentationsmoderator: Der Präsentationsagent führt durch die Präsentation und übernimmt dabei die Aufgaben, die ein menschlicher Präsentator bei Präsentationen hat, wie z.B. auf bestimmte Objekte hinweisen, Erklärungen abgeben, Ratschläge erteilen usw. Alle diese Aktionen müssen durch das Präsentationssystem aufeinander abgestimmt und koordiniert werden. Um diese Aufgabe erfüllen zu können, ist es vorteilhaft, wenn der Präsentationsagent ein möglichst menschliches Erscheinungsbild zeigt.

umfangreiche Präsentationsmöglichkeiten: Damit der Präsentationsagent in unterschiedlichen Applikationen eingesetzt werden kann, muss er eine Vielzahl von Präsentationsmöglichkeiten anbieten. Diese umfassen einerseits eigene Fähigkeiten des Präsentationsagenten, andererseits muss der Agent mit anderen Objekten, die in der Präsentation verwendet werden, interagieren können, so dass sich Daten und Objekte auf sinnvolle Art in die Präsentation einbeziehen lassen.

Anpassbarkeit an unterschiedliche Anwendungsapplikationen: Ein personifizierter Präsentationsagent ist in einer Vielzahl von Anwendungen einsetzbar. Damit er einfach in eine beliebige Anwendung integriert werden kann, muss er über eine Schnittstelle verfügen, die es der Anwendungsapplikation erlaubt, den Agenten leicht anzusteuern. Außerdem muss der Agent gut auf spezielle Bedürfnisse der Anwendung anpassbar sein, d.h. sein Befehlsvorrat muss sich leicht erweitern lassen.

reaktives Verhalten: Der Präsentationsagent muss sich an äußere Veränderungen seiner Umgebung anpassen können. Das umfasst einerseits Veränderungen, die durch Benutzerinteraktionen zustande kommen, wie z.B. das Verschieben eines Fensters, kann sich aber auch auf andere Programme beziehen (falls der Agent in der Benutzeroberfläche zu diesem Programm eingesetzt ist).

Benutzerinteraktion: Ein wichtiges Merkmal einer Benutzeroberfläche ist die Fähigkeit des Agenten, mit dem jeweiligen Benutzer zu kommunizieren, d.h. er kann nicht nur reine Präsentationsaufgaben gegenüber dem Benutzer wahrnehmen, sondern er muss auch verschiedene Methoden zur Verfügung stellen, mit denen ein Benutzer mit dem Präsentationsagent und damit mit dem System, welches er verkörpert, in Beziehung tritt.

1.2 Warum Präsentationsagenten ?

Seit einiger Zeit werden immer neue animierte Oberflächenagenten, also animierte Figuren, die bei der Benutzeroberfläche von Computersystemen eingesetzt werden, entwickelt. Es gibt auch für animierte Agenten geeignete und weniger

geeignete Einsatzgebiete. Im Folgenden sollen nun die Vorteile und Nachteile solcher Agenten untersucht und mögliche Einsatzgebiete aufgezeigt werden.

1.2.1 Entwicklung von Benutzeroberflächen

In der Anfangszeit bestanden die Benutzeroberflächen aus Textein- und Ausgaben. Der Benutzer musste sich die benötigten Befehle mit den jeweiligen Argumenten einprägen. Sind die Befehle dem Benutzer nicht bekannt, gibt es für ihn keine Möglichkeit, das System zu bedienen.

Begünstigt durch immer leistungsfähigere Prozessoren oder andere Systemkomponenten (Grafikkarten, Soundkarten, Speicher) wurden unterdessen starke Fortschritte gemacht, diese Benutzerschwierigkeiten zu vermindern. Erst die Verfügbarkeit leistungsfähiger Computersysteme ermöglichte es, Resourcen des Rechnersystems zur Gestaltung einer aufwendigeren Benutzerschnittstelle zu verwenden.

Durch das Aufkommen grafischer Benutzeroberflächen und die Einführung der Schreibtischmetapher wurde die Bedienung des Systems auch für unerfahrenere Benutzer wesentlich erleichtert. Beispiele für solche Systeme sind z.B. Apple Macintosh, Microsoft Window oder X11/Motif. Diese Systeme zeichnen sich durch folgende Eigenschaften aus [Preece et al., 1994, S. 270]:

- bedeutungstragende Objekte werden visuell sichtbar gemacht
- Aktionen werden schnell und inkrementell ausgeführt
- Aktionen können widerrufen werden
- Ersatz einer komplexen Kommandosprache durch die direkte Manipulation der beteiligten Objekte (dargestellt durch Ikonen)

Auf diese Art konzipierte Schnittstellen bewirken, dass sich Anfänger schneller mit einem System auskennen, da sie unmittelbar das Resultat ihrer Aktion nachvollziehen können. Es wird ihnen ermöglicht, ein System teilweise schon durch reines Ausprobieren der angebotenen Möglichkeiten zu erschließen.

Der Benutzer kann durch Direktmanipulation der dargestellten Objekte Funktionen des Programms auslösen. Dabei werden zur Dateneingabe neben Textfeldern auch Knöpfe, Schieberegler und sensitive Grafiken eingesetzt. Die Programmausgabe wird durch die Verwendung verschiedener Fenster gegliedert.

Wird die Benutzeroberfläche sehr umfangreich, weil das dahinterstehende Programm eine hohe Mächtigkeit aufweist, kann auch eine normale Benutzeroberfläche, die sich an die Schreibtischmetapher anlehnt, schnell unübersichtlich werden. Für unerfahrenere Programmbenutzer stellt die Bedienung einer solchen Oberfläche ein großes Problem dar.

Um auch in diesen Fällen die einfache Bedienbarkeit des Systems zu gewährleisten, versucht man durch „Style-Guides“, die die Hersteller von Betriebsystemen herausbringen, dem Benutzer eine ergonomische Arbeitsumgebung zu ermöglichen. Style-Guides beschreiben beispielsweise die Anordnung der grafischen Bedienelemente auf dem Bildschirm für Kommandos, die häufig in Programmen vorkommen, wie z.B. eine Datei öffnen oder ausdrucken. Dadurch

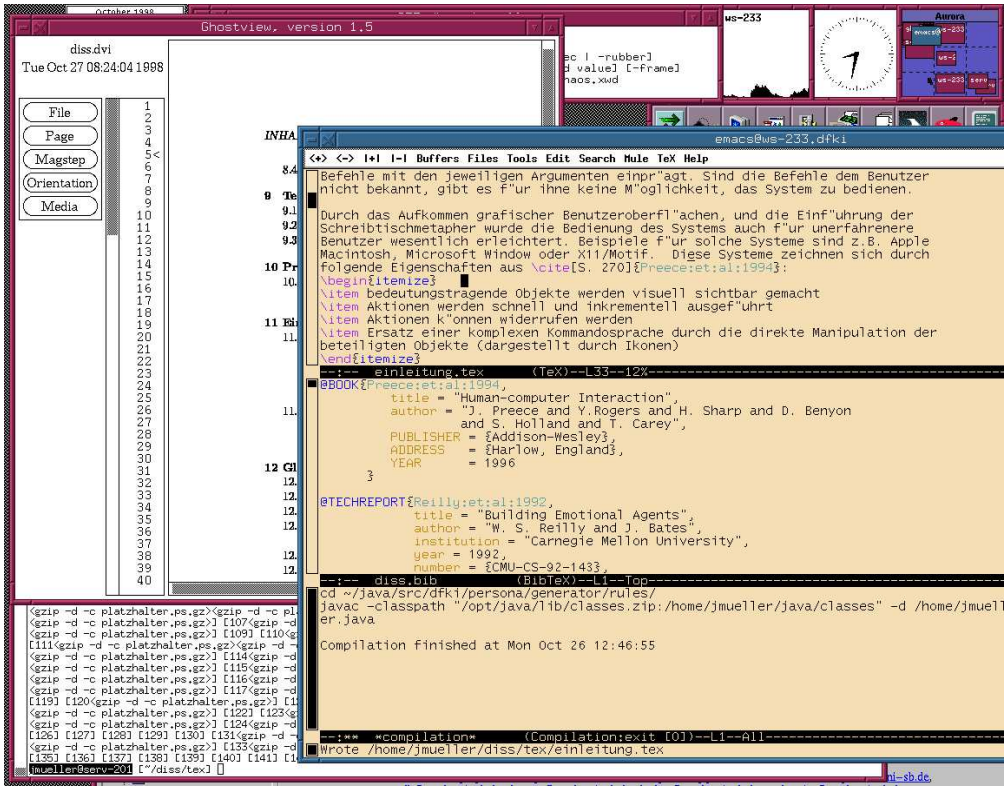


Abbildung 1.2: Bildschirm während eines Arbeitstages

kann sich der Benutzer schneller in unbekannte Programme einarbeiten und sich problemloser zurechtfinden. Darüber hinaus erlaubt man ihm, sich seinen eigenen Desktop nach seinen privaten Wünschen einzurichten (z.B. durch benutzerspezifische Konfigurationsdateien für den Fenster-Manager bei dem X-Window-System). Eine weitere Möglichkeit ist der Einsatz von intelligenten Fenster-Managern, die die Aufteilung der zur Verfügung stehenden Bildschirmfläche optimieren [Graf, 1997].

Nicht nur bei der Bedienung von Computersystemen kann es zu Orientierungsschwierigkeiten kommen. Auch bei der Vermittlung komplexer Sachverhalte, wie in Bedienungsanleitungen zu Geräten oder in Lernsystemen, kann der Betrachter durch die auf ihn hereinbrechende Informationsflut überfordert werden. In konventionellen (gedruckten) Bedienungsanleitungen wird der Betrachter mit einer Folge von Bildern konfrontiert, die er mit Hilfe eines begleitenden Textes interpretieren muss. Je nach Qualität der Beschreibung kann der Betrachter die Erklärung nicht immer einfach nachvollziehen. Es kommt oft vor, dass er sich in der Reihenfolge der Handlungsschritte täuscht oder dass er die Erklärungen nicht den richtigen Bildern zuordnen kann.

Bei technischen Geräten spielt es eine immer größere Rolle, dass die Funktionen einfach zu bedienen sind [Henke, 1998]. Oft werden komplizierte Funktionen eines Gerätes nicht ausgenutzt, nur weil sie nicht ohne Einarbeitungszeit benutzt werden können. Daher wird seit einiger Zeit mehr Wert auf die Entwicklung einfacherer Benutzeroberflächen bei technischen Geräten wie Wecker oder Radios

gelegt. Allerdings gibt es natürlich eine Gruppe von Benutzern, die unbedingt jede Fähigkeit ausnutzen wollen und dafür auch bereit sind, mehr Zeit zu investieren. Dann können sie allerdings das Gerät besser auf ihre Bedürfnisse abstimmen.

Es werden ständig neue Techniken eingeführt, um die Informationsflut, die auf einen Benutzer einströmt, zu beherrschen. Gleichzeitig werden neue Informationsquellen, wie z.B. das WWW, erschlossen, woraus sich neue Orientierungsschwierigkeiten ergeben.

Bei Hypermedia-Präsentationen ist oft zu beobachten, dass ein unerfahrener Benutzer sich durch die Vielzahl der angebotenen Informationen, die zusätzlich noch durch Hyperlinks verbunden sein können, überfordert fühlt. Nicht immer ist klar, in welcher Reihenfolge er die Links besuchen muss, um möglichst effizient an die gewünschte Information heranzukommen. Wird der Informationsraum auf das Internet ausgedehnt, so werden die Navigationsprobleme noch gravierender. Die angebotenen Suchmaschinen helfen zwar, Informationen über Stichwörter zu finden, allerdings sind dabei die gefundenen Informationen noch nicht genug auf den jeweiligen Benutzer abgestimmt. Der Betrachter muss nach wie vor alle gefundenen Seiten nach den gewünschten Informationen durchsuchen. Besser wäre es, eine neue Seite könnte generiert werden, die relevante Informationen für den Benutzer enthält und die so gestaltet ist, dass wichtige Informationen schnell erfassbar werden.

Wie schon angedeutet, waren die Benutzerschnittstellen im Laufe der Computerentwicklung einer starken Veränderung unterworfen. Durch die gesteigerte Leistungsfähigkeit der Rechnersysteme war es möglich, auch Benutzeroberflächen zu entwickeln, die mehr Wert auf Ergonomie legen. Gleichzeitig hat sich aber auch die Zielgruppe gewandelt. Während sich früher vorwiegend Spezialisten mit Computersystemen beschäftigten, dringen Computer immer weiter in alle Lebensbereiche vor, so dass die Anzahl der Anfänger und der unerfahrenen Computerbenutzer gestiegen ist. Bei erfahrenen Computerbenutzern stellt die Verwendung text-basierter Schnittstellen zur Kommandoeingabe in den meisten Fällen kein Problem dar, jedoch muss bei unerfahrenen Benutzern, die selten oder kaum mit einem Computer gearbeitet haben, die Benutzerschnittstelle einfacher zu bedienen sein. In vielen Fällen kann die Vewendung von Benutzerschnittstellen, die Direktmanipulation der dargestellten Objekte zulassen, für den Benutzer eine große Hilfe sein. Beispielsweise werden Informationsterminals in Museen oder in Banken eingesetzt, die es erlauben, über berührungssensitive Bildschirme die dargestellten Objekte direkt über einen Fingerdruck zu manipulieren. Die Technologie erleichtert dem ungeübten Benutzer den Zugang zu den Informationen. Durch die Direktmanipulation von Objekten (Menüs, Buttons, Schieberegler, sensitive Grafik) erspart sich der Benutzer das Lernen einer komplexen Kommandosprache. Er sieht unmittelbar die Folgen seiner Handlung, was den spielerischen Umgang mit Computersystemen fördert. Für einen geübten Benutzer ist es jedoch zu mühsam, sich durch mehrere Menüs zu klicken, wenn er mit der Eingabe eines textuellen Kommandos sein Ziel unmittelbar hätte erreichen können. Z.B. können Dateioperationen einerseits über grafische Dateisystem-Manager durch Manipulation der einzelnen Objekte mit der Maus durchgeführt werden, während man durch die Eingabe regulärer Ausdrücke gan-

ze Gruppen von Dateien auf einmal bearbeiten kann (Abbildung 1.3).

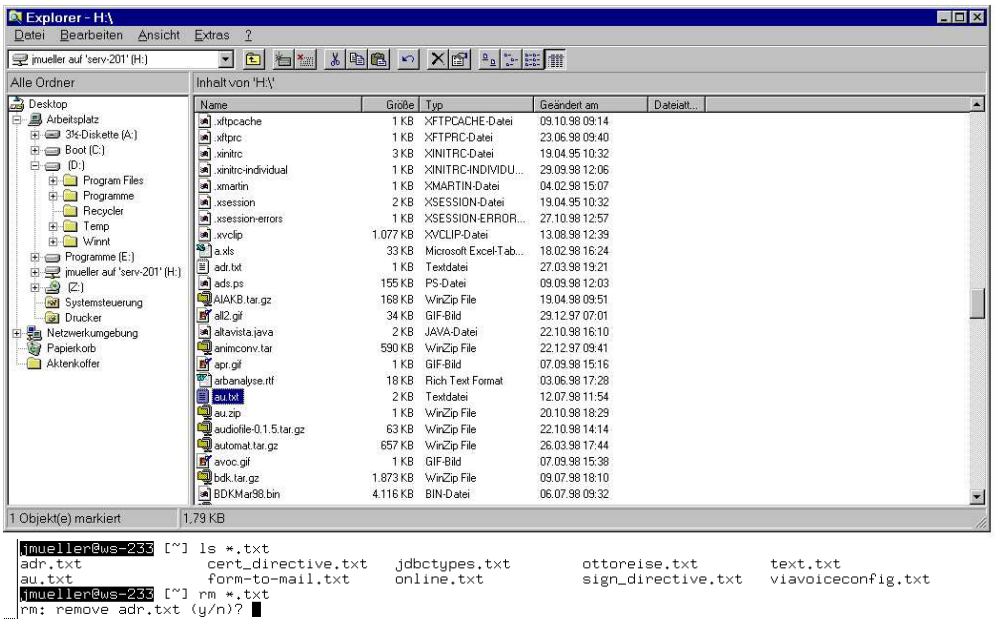


Abbildung 1.3: Manipulation von Dateien mittels (Windows NT-) Datei-Manager und über Unix-Kommandozeile (Lösche alle Dateien, die mit .txt enden)

Eine Möglichkeit, auch bei interaktiven Oberflächen eine Automatisierung von Aktionen zu erreichen, ist die programmgesteuerte Ansteuerung der Benutzerschnittstelle. [Bharat und Sukaviriya, 1993] beschreiben ein System, das es erlaubt, Benutzerinteraktionen mit einer grafischen Benutzeroberfläche zu simulieren. Dazu interpretiert das System ein Skript, das die Benutzerinteraktion beschreibt, und generiert hieraus die benötigten Fenster-Events, um das Anwendungssystem anzusteuern.

Um die Benutzerfreundlichkeit der Benutzerschnittstellen mit Direktmanipulation zu steigern, wird auf unterschiedlichen Wegen versucht, die Kommunikation zwischen Benutzer und Computersystem natürlicher zu gestalten.

Zur Bedienung der Systeme, also zur Dateneingabe, werden unterschiedliche Werkzeuge und Vorgehensweisen verwendet. Neben Eingabewerkzeugen wie Tastaturen, Maus, Track-Balls, berührungssensitiven Bildschirmen oder Datenhandschuhen wird auch die Steuerung der Systeme über gesprochene Sprache [IBM, 1998] oder auch über Eingabesysteme für Gesten [The Human-Computer Interaction Group (MIT), 1998] durchgeführt (Abbildung 1.4).

Gleichzeitig versucht man auch die Ausgabe der Programme natürlicher zu gestalten und sie an die Gewohnheiten ihrer Benutzer anzupassen. Zu diesem Zweck werden die Benutzeroberflächen grafisch aufwendig gestaltet (3D-Effekte, 3D-Oberflächen). Ein Beispiel ist ein Dateimanager, der in Irix von Silicon Graphics wahlweise verwendet werden kann. Mit dessen Hilfe ist es möglich, durch die dreidimensional visualisierte Dateisystemstruktur „hindurchzufliegen“. Außerdem erlaubt man dem Benutzer, sich eine Benutzeroberfläche nach seinen

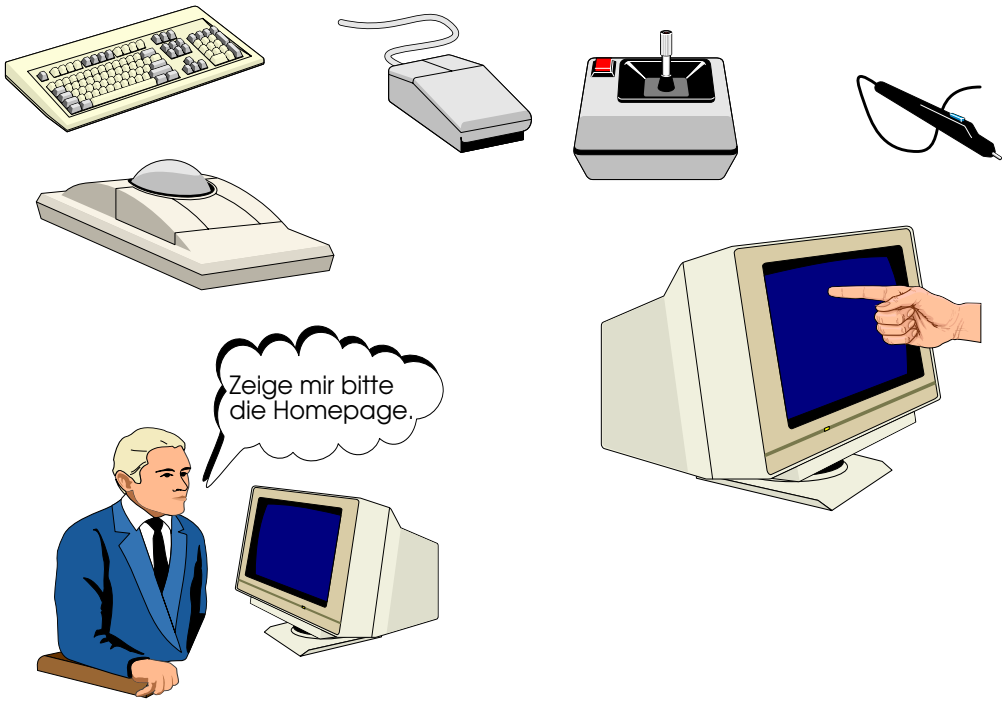


Abbildung 1.4: Verschiedene Möglichkeiten zur Ansteuerung von Computersystemen

Bedürfnissen zu gestalten und diese Einstellung zu speichen (z.B. als Konfigurationsdatei eines X11-Fenster-Managers). Überdies versucht man die Auswirkungen der Benutzereingaben auf das Programm durch Multimediaeffekte verständlicher zu machen. Beispielsweise kann der Aktion „Datei löschen“ eine bestimmte Audioausgabe zugeordnet werden, so dass der Benutzer auch eine hörbare Rückmeldung seiner Anweisung erfährt [Microsoft, 1998b]. Eine andere Möglichkeit ist, Programme mit der Fähigkeit zur Ausgabe synthetischer Sprache auszurüsten. Ein Textverarbeitungsprogramm ist so imstande, einem Benutzer einen Text zur Kontrolle vorzulesen [IBM, 1998]

[Chang und Ungar, 1993] verwenden Animationen, um dem Benutzer einer fensterbasierten Oberfläche die Vorgänge besser verständlich zu machen. Sind bestimmte Aktionen für Benutzer schwer zu verstehen, erleichtern Animationen, die im Zeichentrickstil entworfen sind, ein einfacheres Verständnis. Sie verwenden bestimmte Animationstechniken, wenn Fenster oder andere Oberflächenelemente durch den Benutzer manipuliert werden. Beispielsweise werden folgende Techniken gebraucht:

- Bewegungen werden durch Motion Blur , also das Markieren der Spur eines Objekts, verdeutlicht.
- Um den Benutzer auf eine Aktion besonders hinzuweisen und vorzubereiten wird beispielsweise beim Bewegen von Objekten eine kurze Bewegung in umgekehrter Richtung vor die eigentliche Aktion vorangestellt.
- Analog der realen Welt wird die Bewegungsgeschwindigkeit dynamisch

verändert. Eine Aktion fängt langsam an, wird beschleunigt und am Ende wieder abgebremst.

- Bewegungsaktionen laufen in der realen Welt selten auf geraden Linien. Bewegen sich Objekte auf Kurven, so wirken sie lebendiger und natürlicher.

- Normalerweise endet eine Bewegung in der realen Welt nicht mit dem plötzlichen Stillstand eines Objekts. Der unvermittelte Übergang zwischen Aktion und Ruhe wird in Zeichentrickfilmen oft dadurch symbolisiert, dass ein Objekt nach dem Bremsvorgang zittert oder vibriert.

Die Benutzung einer cartoonartigen Benutzerschnittstelle gibt dem Benutzer das Gefühl, unmittelbar in die Handlung eingebunden zu sein.

[Ender, 1997] stellt ein Programm (Roomancer) vor, das die Verzeichnisse und Dateien, die in einem Computer abgelegt sind, als virtuelles Haus visualisert. In diesem Haus wird der Benutzer selbst durch eine Figur, die er wählen kann und die ihn in der virtuellen Umgebung repräsentiert, visualisiert. Sind die Rechner vernetzt, ist auch ein Besuch auf anderen Rechnern möglich.

Da viele Techniken versuchen, den Umgang mit einem Computersystem natürlicher zu gestalten und ihn den täglichen menschlichen Verhaltensweisen anzupassen, liegt es nahe, das Programm selbst dem Betrachter gegenüber durch eine animierte Figur zu repräsentieren.

1.2.2 Vorteile animierter Präsentationsagenten

Die Repräsentation eines Softwaresystems durch einen animierten Agenten bereichert das System durch eine Vielzahl neuer Möglichkeiten:

Überwindung der Hemmschwelle. Durch den Einsatz von animierten Agenten kann ein Anfänger eine eventuell vorhandene Hemmschwelle bei der Benutzung von Computern leichter überwinden [Adelson, 1992].

Natürlichere Kommunikationsform. Die Repräsentation des Systems durch eine animierte Figur ermöglicht eine natürlicher wirkende Präsentation. Durch die Wahl einer menschlich wirkenden Figur kann z.B. ein Vortrag mit einem realen Präsentator nachempfunden werden. Ohne eine animierte Figur könnten die besonderen Präsentationshilfsmittel wie Gestik und Mimik nicht zum Zuge kommen.

Neue Präsentationsmöglichkeiten. In einer Präsentation mit einer animierten Figur können neue Präsentationsmittel eingesetzt werden, die ohne die vermenschlichte Darstellung des Präsentationsystems nicht benutzt werden könnten. Dazu zählen insbesondere die Zeigegesten, die auch von einer Sprachausgabe begleitet sein können.

Steigerung des Erinnerungsvermögens. Durch die vielfältigen Präsentationstechniken eines Präsentationsagenten kann oft das Erinnerungsvermögen des Betrachters gesteigert werden. Über die auffällige Präsentation

fällt es dem Betrachter auch leichter, sich an die präsentierten Sachverhalte zu erinnern.

Steigerung des Vertrauens. Durch die visuelle Repräsentation des Präsentationssystems vertrauen Benutzer öfter den Angaben, die das Computersystem macht.

Ausnutzung des Spieltriebes. Durch den visuellen und akustischen Effekt, den der Agent auslöst, wird die Bedienung eines Programms aufgelockert und der Spieltrieb des Benutzers angeregt. Wie in Abschnitt 8.1 beschrieben, macht die Bedienung des Programms dem Benutzer mehr Spaß, was natürlich den Arbeitseifer steigert.

1.2.3 Geeignete und weniger geeignete Anwendungsgebiete

Nicht alle Benutzeroberflächen lassen sich in allen Umgebungen gleich gut einsetzen. Beispielsweise führt die Steuerung des Computers durch Spracheingabe bzw. die Benutzung von synthetischer Sprache als Ausgabemedium innerhalb von Großraumbüros zu Schwierigkeiten, da sich viele gleichzeitige Anwender unweigerlich gegenseitig belästigen.

Ähnlich verhält es sich bei vielen Erweiterungen von Benutzerschnittstellen. Oft ist die Zweckmäßigkeit auch von der Erfahrung des Nutzers abhängig. Selbst bei Menüsteuerungen sind hier schon Unterschiede zwischen den einzelnen Benutzergruppen möglich. Während Anfänger bzw. Nutzer, die selten ein bestimmtes Programmm oder eine Funktion verwenden, gerne die Hilfe der Menüsteuerung annehmen, benutzen viele erfahrene Anwender lieber die entsprechenden Tastenkombinationen, um den gleichen Effekt zu erzielen, da sie so nicht die Hand von der Tastatur nehmen müssen und es außerdem schneller geht. Allerdings müssen sie die entsprechende Tastenkombination auswendig beherrschen.

Auch die Einsatzmöglichkeiten animierter Agenten sind stark von dem Benutzerprofil des jeweiligen Anwenders sowie von der jeweiligen Applikation abhängig. Konfiguriert man einen Präsentationsagenten als Hilfsassistenten innerhalb eines normalen Anwendungsprogramms, wie z.B. einer Textverarbeitung, so sind oftmals Anfänger für die gebotene Hilfe dankbar, während fortgeschrittene Anwender lieber auf die Hilfe verzichten, da sie durch andere Techniken (z.B. durch einfaches Ausprobieren der Funktionen) schneller zum Ziel kommen. Anfänger probieren in vielen Fällen nicht so gern neue unbekannte Funktionen aus, da sie fürchten, etwas falsch zu machen, was sie nicht mehr rückgängig machen können. In solchen Fällen kann der Einsatz eines Präsentationsagenten sinnvoll sein, denn er hilft dem unerfahrenen Benutzer, seine Hemmschwelle abzubauen.

Auch das persönliche Benutzerprofil ist bei der Konzeption eines Präsentationsagenten wichtig. Die Akzeptanz eines solchen Agenten hängt stark davon ab, wie sich der Agent auf die persönlichen Vorlieben des Benutzers einstellen kann. Viele Leute haben es gerne, wenn der Agent lebhaft ist und während der Präsentation witzige (eventuell auch überzogene) Effekte ausführt, andere Benutzer möchten auf gar keinen Fall, dass der Agent unnötige (nur der Unterhaltung bzw. der Auflockerung dienende) Effekte ausführt. Möglicherweise

spielt bei diesem Effekt auch das Alter des Benutzers eine Rolle. Zeigt man die gleiche Präsentation mit einem animierten Agenten zwei verschiedenen Betrachtern, so kann es sein, dass der eine Zuschauer sagt, der Agent könne ruhig noch mehr Effekte in die Präsentation einfließen lassen, um sie lustiger zu gestalten, während ein anderer Betrachter sich schon von den gezeigten Effekten gestört fühlt.

Ein Präsentationsagent kann auch dazu eingesetzt werden, Kommunikationsformen, die sonst nur natürlichen Personen vorbehalten sind, zu pflegen. Beispielsweise kann er die Gebärdensprache verwenden (entweder als Lernsystem zum Erlernen der Gebärdensprache oder zur realen Kommunikation mit gehörlosen Benutzern).

Der Einsatz eines animierten Agenten innerhalb eines reinen Präsentationssystems (z.B. Präsentation von Gebrauchsanweisungen, Tutorensystem, Präsentation von Daten aus dem WWW, Informationssystem) ist von den persönlichen Computerkenntnissen nicht so stark beeinflusst, da in diesem Anwendungsfeld Informationen vermittelt werden, die nicht unmittelbar mit den Fähigkeiten zur Programmbenutzung verknüpft sind. Außerdem werden bei Präsentationssystemen im Allgemeinen verschiedene ausschmückende Effekte benutzt, so dass in diesem Umfeld ein Präsentationsagent leichter akzeptiert wird. Auch spielt der Zeitfaktor bei der Bedienung eines Präsentationssystems keine so große Rolle wie in einem echten Anwendungsprogramm, da die meiste Zeit durch das Betrachten bzw. das Verstehen der dargebotenen Informationen verbraucht wird und nicht bei der Bedienung des Programms. Bei einem Informationssystem kommt es vielmehr auf eine verständliche, einprägsame, aber auch unterhaltende Darstellung der Informationen an. Bei einem Informations- bzw. Tutorialsystem dient der Präsentationsagent als Ansprechpartner für den Benutzer, an den er sich bei Problemen wenden kann. Da der Präsentationsagent als Repräsentant des gesamten Präsentationssystems eingesetzt ist, kann dieser die Präsentation den Benutzerwünschen entsprechend anpassen (z.B. ausführlicher, schneller). Um den Präsentationsagenten in solchen Umgebungen effektiv einsetzen zu können, muss er mit den passenden Fähigkeiten ausgestattet sein. So ist z.B. ein Zeigestock ein unentbehrliches Werkzeug in einer Präsentation. Auch muss er in der Lage sein, dargestellte Objekte zu manipulieren, d.h. er muss eine Repräsentation von ihnen besitzen. Der Einsatz eines Präsentators innerhalb eines Präsentationssystems entspricht den Gewohnheiten vieler Betrachter, da oftmals reale Präsentationen (z.B. Vorträge, Schulungen, Werbeveranstaltungen) durch einen oder mehrere Personen durchgeführt werden, die dem Publikum das Informationsmaterial erläutern und auf seine Fragen eingehen.

Trotz vieler positiver Stimmen zu animierten Agenten gibt es auch Gegenargumente [Shneiderman, 1997]:

- Die Interaktion mittels natürlicher Sprache mit einem Computersystem ist ineffektiv. Diese Interaktionsform wird oft von anthropomorphen Benutzeroberflächen eingesetzt. Die Steuerung über direktmanipulierbare Oberflächen ist der Sprachsteuerung vorzuziehen (Ausnahme: Benutzer mit Behinderungen).

Abbildung 1.5: Einsatz eines animierten Präsentationsagenten in einer Benutzerschnittstelle

- Durch einen Agenten besteht die Gefahr, die Kontrolle über das System zu verlieren und dass der Benutzer nicht über alle kausalen Zusammenhänge der einzelnen Aktionen informiert wird.

Die Nützlichkeit eines Präsentationsagenten hängt stark von der jeweiligen Anwendung ab.

Benutzeroberflächen, die nur von geschultem Personal bzw. erfahrenen Benutzern bedient werden, wie z.B. in Leitständen oder Überwachungsterminals, sind normalerweise nicht das Haupteinsatzgebiet animierter Schnittstellenagenten. Allerdings können diese bei der Aus- und Weiterbildung an solchen Einrichtungen eingesetzt werden und die Funktionen der Oberfläche erklären. Hier dient der Präsentationsagent dazu, den Benutzer effizienter in der Bedienung einer komplexen Benutzeroberfläche zu schulen. Er kann sich eine Präsentation beliebig oft und ohne Unterstützung von Dritten immer wieder ansehen und sein Wissen vertiefen, gleichzeitig besteht die Möglichkeit, da die Präsentation dynamisch generiert wird, dass immer die neuesten Änderungen in der Bedienung, aber auch in der Didaktik der Schulung in den Bedienungskurs einfließen können.

In Tabelle 1.1 werden einige Vor- und Nachteile zusammengefasst.

Vorteile	Nachteile
Überwindung einer Hemmschwelle bei der Benutzung von Computersystemen	Gefahr des Verlustes der Kontrolle über das System
Natürlichere Kommunikationsformen	Direktmanipulierbare Oberflächen effektiver als Sprachsteuerung (erfahrene Benutzer)
Neue Präsentationsmöglichkeiten	
Ausnutzung des Spieltriebes	
Mehr Vertrauen in Computerergebnisse	
Steigerung des Erinnerungsvermögens	

Tabelle 1.1: Mögliche Vor- und Nachteile animierter Agenten in Benutzeroberflächen

1.3 Ziel der Arbeit

Das Ziel der vorliegenden Arbeit ist die Erforschung einer wissenschaftlichen Methodik zum Entwurf und zur Realisierung von Präsentationsagenten. Um dieses Ziel zu erreichen, müssen zunächst die folgenden grundlegenden Fragen beantwortet werden.

1. Wie können Präsentationsagenten auf bestehenden Benutzeroberflächen oder Web-Seiten eingesetzt werden, ohne dass die zugrunde liegende Anwendungssoftware neu programmiert werden muss ?

2. Wie kann XML so erweitert werden, dass komplexes Präsentationsverhalten der Persona auf möglichst abstraktem Niveau spezifiziert werden kann ?

3. Wie kann die Lebendigkeit, die Vertrauenswürdigkeit und die Emotionalität eines Präsentationsagenten so parametrisiert werden, dass eine Benutzer- und Situationsadaption möglich wird?

4. Mit welchen Mechanismen lässt sich das Persona-Verhalten durch direkte Manipulation von grafischen Elementen einer zu präsentierenden WWW-Seite steuern ?

5. Wie können Präsentationsagenten für Internet-Anwendungen so optimiert werden, dass minimale Ladezeiten und eine geringe Belastung des Client-Rechners garantiert werden kann ?

6. Können die Methoden zur Erzeugung eines einzelnen Präsentationsagenten so verallgemeinert werden, dass mehrere Instanzen von Präsentationsagenten gleichzeitig bei einer Präsentationsaufgabe erscheinen und dabei unterschiedliche kommunikative Rollen wahrnehmen ?

Als praktisches Ziel soll ein Präsentationsagent entwickelt werden, der an unterschiedliche Anwendungsysteme einfach adaptierbar ist. Dabei sollen folgende für einen Präsentationsagenten wichtige Punkte eine besondere Berücksichtigung finden:

- Um die Ansteuerung des Präsentationsagenten zu vereinfachen, soll die **Ansteuerung auf hohem und abstrakten Niveau** erfolgen. Der animierte Präsentationsagent soll sich nahtlos in Präsentationssysteme einfügen lassen. Die genaue Realisierung der jeweiligen Aktion wird von dem Agenten in Abhängigkeit der jeweils aktuellen Situation selbständig ausgewählt.

- Der Präsentationsagent soll für einen Benutzer des Präsentationssystems, in das der Agent eingebettet ist, die erzeugte Präsentation **leichter verständlich und unterhaltsamer** gestalten. Aus diesem Grund sollte der Agent auch möglichst **lebendig** wirken.

- Der Agent soll sich sowohl in **WWW- als auch in Desktop-Anwendungen** einsetzen lassen. Für den Einsatz innerhalb des World-Wide-Webs müssen die speziellen Erfordernisse in Bezug auf übertragene Datenmenge und auf die Anbindung des Präsentationssystems berücksichtigt werden.

- Das **visuelle Erscheinungsbild** des Präsentationsagenten muss **einfach veränderbar** sein, da er sich so besser für unterschiedliche Anwendungsgebiete konfigurieren lässt.

- Der Präsentationsagent muss ein **Definitionswerkzeug zur Beschreibung neuer Gesten** bereitstellen, so dass sich neue Gesten in das System integrieren lassen.

- Der Präsentationsagent muss dem Präsentationsystem verschiedene **Möglichkeiten zur Visualisierung von unterschiedlichen Objekttypen** zur Verfügung stellen, so dass der Präsentationsagent die dargestellen Objekte manipulieren kann.

- Der Präsentationsagent dient als Schnittstelle zwischen dem Benutzer und dem Präsentationsystem. Daher muss er auf **Benutzereingaben reagieren** können.

- Der Präsentationsagent soll möglichst vielseitig einsetzbar sein. Daher muss er auf einfache Art **erweiterbar** und an spezielle Erfordernisse **anpassbar** sein. Dieses Merkmal umfasst sowohl die Schnittstelle zum Präsentationssystem als auch die Fähigkeiten des Präsentationsagenten selbst.

1.4 Aufbau der Arbeit

In der vorliegenden Arbeit soll ein Präsentationsagent implementiert werden, der den im vorherigen Abschnitt beschriebenen Anforderungen genügt.

Im folgenden Kapitel wird der aktuelle Stand der Forschung zu animierten Agenten und benachbarten Themengebieten untersucht.

Bei der Entwicklung von personifizierten animierten Präsentationsagenten spielt der Einsatz von Gestik und Mimik eine große Rolle. Abschnitt 3.1 gibt daher einen Überblick über die Verwendung von Gestik und Mimik bei der menschlichen Kommunikation und die daraus resultierenden Implementierungsentscheidungen für den Präsentationsagenten. Weiterhin werden Präsentationsmöglichkeiten, die sich durch den Einsatz von Präsentationsagenten ergeben, vorgestellt. Aus diesen Untersuchungen wird die Systemkonzeption des Persona-Systems hergeleitet.

Kapitel 4 und 5 beschreiben die Implementierung des Präsentationsagenten. Dabei werden die Generierungskomponente und die Abspielkomponente für die berechneten Animationen getrennt behandelt, um die stufenweise Zerlegung komplexer Präsentationshandlungen in primitive Handlungen besser darstellen zu können. Weiterhin wird auf die Generierung von Präsentationen mit mehreren Agenten eingegangen. Im Anschluss werden Besonderheiten der WWW- bzw. der Desktop-Implementierung beschrieben.

Das Kapitel 7 beschäftigt sich mit der Integration des implementierten Präsentationsagenten in verschiedene Anwendungssysteme. Dabei wird auch auf die speziellen Erfordernisse der Anwendungssysteme eingegangen. Zusätzlich werden weitere Anwendungsmöglichkeiten vorgestellt.

In Kapitel 8 werden die erzielten Ergebnisse zusammengefasst und Erweiterungsmöglichkeiten aufgezeigt. Um die Wirkung von animierten Präsentationsagenten zu untersuchen, wurden verschiedene Evaluierungen durchgeführt. Im Kapitel 8.1 werden Evaluierungen anderer Präsentationsagenten und die Evaluierung des in dieser Arbeit entwickelten Präsentationsagenten beschrieben.

Kapitel 2

Agenten und verwandte Themengebiete

In der letzten Zeit werden immer neue Oberflächenagenten entwickelt und auf verschiedenen Einsatzfeldern genutzt. Gleichzeitig werden Techniken wie Gesichtsanimation, Sprachsynthese, Spracherkennung, die bei animierten Agenten zum Einsatz kommen können, weiterentwickelt. Im Folgenden sollen hauptsächlich animierte Agenten betrachtet werden, von der softwaretechnischen Bedeutung des Wortes „Agent“ soll an dieser Stelle abstrahiert werden. Sie werden in verschiedenen Ausprägungen eingesetzt (z.B. Präsentationsagent, Schnittstellenagent).

Das wirtschaftliche Interesse an Agententechnologien wird dadurch deutlich, dass auch populäre Zeitschriften Artikel über Agenten veröffentlichen [Obmann, 1998; Henke und Seeger, 1997]. Dabei werden auch Agenten mit antropomorpher Oberfläche berücksichtigt.

Animierte Agenten können in Ausbildungssystemen eingesetzt werden. Sie unterstützen den Lernenden bei seinen individuellen Problemen, helfen durch räumliche Trennungen hervorgerufene Barrieren zu überbrücken und unterstützen die Zusammenarbeit mehrerer Personen [Boy, 1997].

Das folgende Kapitel soll einen Überblick über bestehende Agentensysteme (hier hauptsächlich im Sinn von Oberflächenagenten) geben und verwandte Arbeitsgebiete untersuchen.

2.1 Was sind Agenten ?

Ein alter Traum vieler Menschen ist, Maschinen zu konstruieren, die dem Menschen Arbeiten abnehmen und so als künstliche Butler alltägliche Aufgaben übernehmen. In Sciencefictionfilmen und -romanen spielen solche künstliche Wesen eine wichtige Rolle. Das oft für diese Wesen gebrauchte Wort Roboter hat seine Ursprünge in dem tschechischen Wort robota für Frondienst, Schinderei oder harte Arbeit. Die Wurzeln dieser Vorstellung sind in dem Werk R.U.R. von Karel Čapek zu suchen [Bradshaw, 1997].

In der Realität werden Roboter allerdings in den meisten Fällen weniger spektakulär eingesetzt. Mechanische Roboter werden z.B. in der Autoindustrie

benutzt, um Fahrzeugkarosserien zu schweißen. Neben den mechanischen Robotern setzt sich auch immer mehr der Begriff des Software-Roboters durch. Darunter sind Softwaresysteme zu verstehen, die eigenständig Aufgaben übernehmen und die den Menschen von langwierigen und stumpfsinnigen Aufgaben entlasten. Ein Beispiel für solche Systeme sind Suchroboter , die Informationen aus dem World Wide Web suchen und für den Benutzer passend aufbereiten. Der Einsatz solcher Systeme hilft dem Benutzer, die gewünschten Informationen schneller zu erhalten und ermöglicht ihm, sich einen Überblick über Informationen zu verschaffen. Um im World Wide Web effizient Informationen suchen zu können, ist man meistens auf die Unterstützung von Suchmaschinen angewiesen [Turau, 1998]. Der Index dieser Suchmaschinen wird fast immer automatisch mit Hilfe von Web-Robotern (Spider) aufgebaut, die WWW-Server analysieren und ihre Informationen in den Datenbanken der Suchmaschinen ablegen. Sie kommen an neue Seiten durch Verfolgen der URLs, die in den Dokumenten abgelegt sind. Darüber hinaus können Web-Roboter auch zu komplexeren Aufgaben eingesetzt werden. Um z.B. Homepages zu finden (Ahoy!, http://www.ahoy.com), werden die Suchstrategien entsprechend bestimmter Heuristiken abgeändert (Generierung von URL, Einschränkung auf bestimmte Server, Textanalyse).

Direktmanipulierbare Benutzerschnittstellen, die Operationen und Objekte als Ikonen repräsentieren, werden inpraktikabel, falls die Aufgabenstellung eine gewisse Größenordnung überschreitet. Beispielsweise ist es oft einfacher, statt viele Objekte einzeln per Maus zu manipulieren, ein Skript zu verwenden, das die ganze Aufgabe übernimmt. Falls viele Ikonen zur Bedienung eines Programmes benötigt werden, kann leicht der Überblick verloren gehen. Ein anderes Problem ist, dass der Benutzer einer direktmanipulierbaren Benutzerschnittstelle die Ausführung eines Befehls abwarten muss, obwohl viele Anwendungen ein Scheduling von Aktionen verlangen. Außerdem ist es wünschenswert, wenn Software in der Lage ist, System-Überwachungsaufgaben zu erfüllen. Eine weitere Schwierigkeit ist die Komposition komplexer Operationen auf Basis primitiver Operationen über eine direktmanipulierbare Oberfläche. Je komplexer eine Aufgabenstellung wird, umso schwieriger wird ihre Lösung mittels einer solchen Oberfläche. Normalerweise bieten solche Schnittstellen keine automatischen Lernmechanismen an, so dass das System nicht von der Wiederholung von Vorgängen profitieren kann. Auch hier bietet sich die Übertragung solcher Aufgaben auf Programme (Agenten) an.

Software wird immer schwieriger zu bedienen, so dass eine neue Interaktionsmöglichkeit mit Computersystemen notwendig wird [Koda und Maes, 1996]. Da die Informationsflut ständig weiter steigt, besteht ein zunehmender Bedarf an Filterung der Daten bzw. Aufbereitung von Daten für die persönlichen Erfordernisse eines Benutzers.

Ein Ziel der Software-Agenten ist, den Benutzer von einfachen und eintönigen Aufgaben zu entlasten. Dem Benutzer soll es ermöglicht werden, Aufgabenspezifikationen auf hohem Niveau zu tätigen, die der jeweilige Software-Agent dann ausführt.

Software-Agenten sollen sich genau dieser Problemen annehmen:

- Skalierbarkeit: Agenten können Informationen filtern und den Benutzer von unwichtigen Informationen entlasten.

- Agenten sollen Aufgaben im Hintergrund ohne ständige Überwachung von Benutzern ausführen. Falls die Aufgabe erledigt wurde, melden sich die Agenten bei dem Benutzer. Außerdem sollen sie selbständig die Überwachung und die Steuerung von Systemen übernehmen.

- Da Agenten auf abstraktem Niveau (z.B. auf der Ebene von Zielbeschreibungen) instruiert werden, ist es ihnen oft möglich, entsprechend den aktuellen Gegebenheiten aus unterschiedlichen Strategien eine passende auszuwählen. Daher kann auf unvorhergesehene Ereignisse angemessen reagiert werden.

- Mittels eines Benutzerprofils kann der Agent sich besser auf die Wünsche des Anwenders einstellen.

2.2 Eigenschaften von Agenten

Software-Agenten helfen dem Benutzer, Informationen seinen Wünschen gemäß zusammenzustellen (virtuelle Dokumente) [Bradshaw et al., 1997].

Nach der Auffassung von [Negroponte, 1997] stellt die ideale Benutzerschnittstelle die Metapher eines gut ausgebildeten Butlers dar, der seinen Besitzer von den täglichen Arbeiten entlastet und auch in eingeschränktem Maß eigene Entscheidungen trifft. Eine andere mögliche Aufgabe für einen persönlichen Agenten ist die Filterung von Daten in Bezug auf das jeweilige Benutzerprofil. Da der Agent die Vorlieben seines Besitzers kennt, kann er die für ihn wichtigen Informationen finden und benutzeradaptiv präsentieren.

Ein Agent sollte für seinen Besitzer die Rolle eines guten alten Freundes spielen, der die Vorlieben des Benutzers kennt. Der Agent wirkt noch sympathischer, wenn er durch eine Figur auf dem Bildschirm verkörpert wird.

Nach [Bradshaw, 1997; Etzioni und Weld, 1995; Franklin und Graesser, 1996] zeichnen sich Agenten durch folgende Eigenschaften aus: Agenten haben die Fähigkeiten, auf ihre Umwelt zu reagieren. Beispielsweise müssen sie in der Lage sein, Benutzereingaben verarbeiten zu können. Außerdem sollten Agenten wenigstens zu einem gewissen Grad in der Lage sein, selbständig Entscheidungen zu treffen und zu handeln. Agenten können die ihnen gestellten Aufgaben entweder allein oder auch zusammen mit anderen Agenten lösen. Die Kommunikation sollte mehr auf der Basis von Sprechakten geschehen als auf der Grundlage von niederen Protokollkonstrukten. Agenten sollten auf Grund einer Wissensbasis selbständig neue Pläne generieren und anwenden, um die Flexibilität zu steigern. Glaubhafte Agenten sollten ihren inneren Zustand durch das Zeigen von Emotionen dem Benutzer verdeutlichen können. Dabei sollte möglichst die Identität des Agenten über einen längeren Zeitraum gleich bleiben. Agenten, die sich selbständig an ihre Umgebung anpassen und aus ihren Erfahrungen lernen können, sind einfacher einsetzbar. Mobile Agenten haben zusätzlich die Mög-

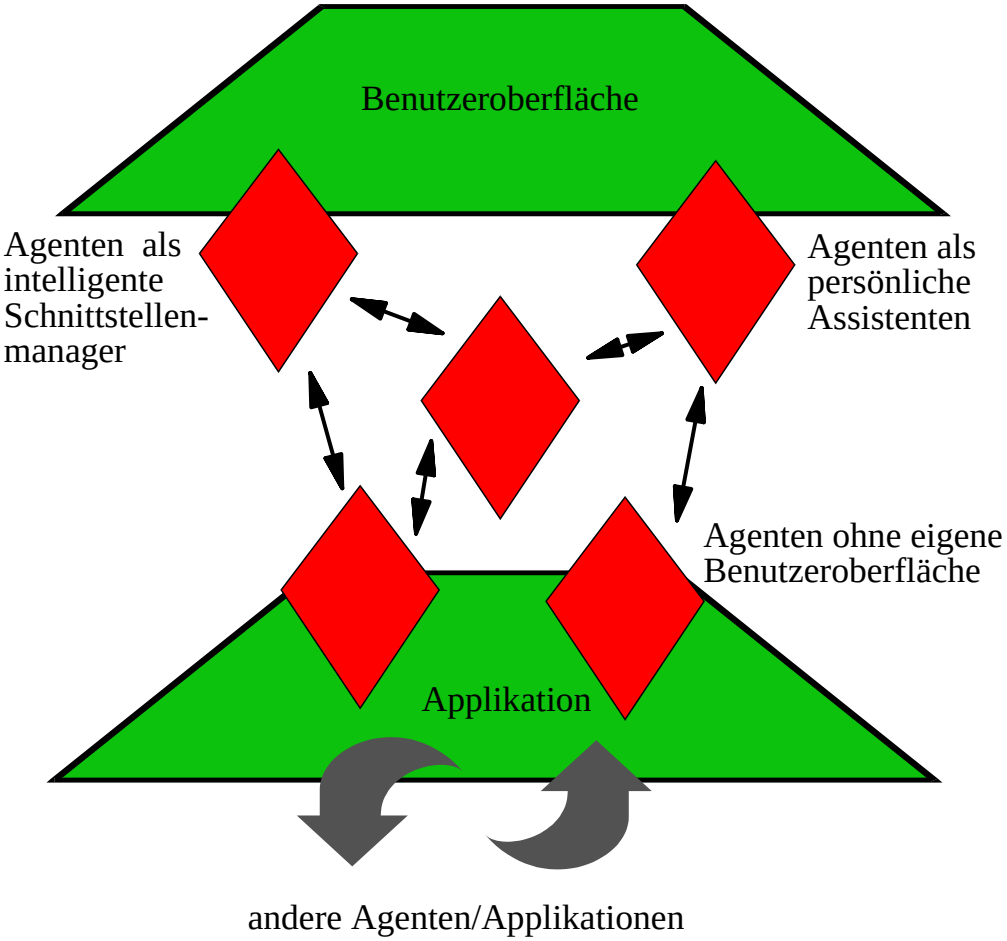


Abbildung 2.1: Agenten können sowohl an der Benutzeroberfläche als auch innerhalb der eigentlichen Applikation eingesetzt werden. Sie kommunizieren untereinander und auch mit dem Benutzer und anderen Agenten oder Applikationen (nach [Bradshaw, 1997, S.19], verändert).

lichkeit, von einem Rechner auf einen anderen Rechner zu wandern und dort weiterzuarbeiten.

Eigenschaften von Agenten sind [Bradshaw, 1997]:

Personalisierung: Der Agent hat Wissen über den Benutzer (z.B. Interessen, Wissen, aktuelles Arbeitsgebiet)

Proaktiv: Der Agent handelt für seinen Benutzer (auch ohne besondere Aufforderung)

Autonom: Der Agent handelt ohne direkte Ansteuerung/Kontrolle durch den Benutzer (zielgesteuert).

Adaptiv: Der Agent kann sich an neue Umstände anpassen.

Reaktivität: Der Agent reagiert auf Eingaben.

Kollaboratives Verhalten: Der Agent besitzt die Fähigkeit, mit anderen Agenten zu kommunizieren, um ein gemeinsames Ziel zu erreichen.

Zeitliche Kontinuität: Der Agent verhält sich auch über einen längeren Zeitraum entsprechend seiner Identität.

Persönlichkeit: Der Agent zeigt Eigenschaften, die ihn glaubhaft erscheinen lassen (z.B. Emotion)

Mobilität: Der Agent kann sich von einer Plattform zu einer anderen bewegen.

Soziales Verhalten: Der Agent kann sich mit anderen Agenten oder Menschen mittels einer speziellen Sprache verständigen.

Beispielsweise kann die E-Mail-Verteilung auf Basis von Benutzerprofilen (Information Lens) ein gutes Hilfsmittel darstellen [Malone et al., 1997]. Das System erlaubt die Verteilung und Klassifizierung von E-Mail entsprechend Benutzerpräferenzen auf bestimmte benutzerdefinierte Ordner. Bei diesem System, das sowohl feste Eingabefelder als auch Freitext handhaben muss, wird die Auswahl der Mails über benutzerdefinierbare Regeln geleistet.

Aus mehreren Agenten können Agentenhierarchien bzw. Agentennetze zusammengestellt werden, die auch komplexere Anfragen bearbeiten können [Knoblock und Ambite, 1997]. Komplexe Anfragen werden von den übergeordneten Agenten in primitivere Anfragen für die untergeordneten Agenten aufgespalten und diese weiterpropagiert.

Zur Kommunikation zwischen Agenten wurden verschiedene Sprachen entwickelt, wie z.B. CORBA oder KQML [Finin et al., 1997; Genesereth, 1997]. Die Programmierung von Agenten erfordert eine besondere agentenorientierte Programmierung, die die Fähigkeiten, Wünsche und Arbeitsvoraussetzungen für einen Agenten modelliert [Shoham, 1997].

Mobile Agenten werden häufig in plattformunabhängigen Sprachen wie Telescript oder Java implementiert [Cohen und Levesque, 1997; White, 1997]. Dadurch wird es ihnen möglich, auf unterschiedlichen Plattformen ihre ihnen übertragenen Aufgaben durchzuführen. Der Präsentationsagent Persona kann z.B. als Java-Applet auf unterschiedlichen Plattformen eingesetzt werden.

2.3 Schnittstellenagenten

Ein Agent kann charakterisiert werden als ein Softwaresystem, das selbständig Ziele verfolgt, Pläne erstellt und autonom auf Eingaben und Ereignisse reagiert [Erickson, 1997]. Neben dieser internen Charkterisierung des Systems wird oftmals der Agent durch eine animierte Figur für den Benutzer visuell sichtbar gemacht.

Dadurch ergeben sich für den Programmierer viele neue Probleme. Agenten müssen sich eigenständig an den Benutzer adaptieren können (z.B. an den Lernerfolg bei einem Tutorsystem), allerdings kann es dabei auch zu Schwierigkeiten kommen, wenn z.B. der Benutzer nicht weiß, warum das System auf eine bestimmte Art handelt. Wird Ursache und Wirkung zu stark verwischt,

kann die Gefahr bestehen, dass der Benutzer eine Reaktion des Agenten falsch interpretiert.

Vergleicht man die Desktop-Metapther mit der Agenten-Metapher, so kommt man zu folgenden grundlegenden Erwartungen, die ein Benutzer an ein sichtbares Objekt auf dem Bildschrim hat: Ein Objekt der Desktop-Metapher ist ein sichtbares Objekt, das an einer bestimmten Stelle auf der Benutzeroberfläche zu finden ist. Das Objekt ist passiv, es führt nicht selbständig Aktionen aus. Ein solches Objekt kann allerdings andere Objekte enthalten. Im Gegensatz zu den passiven Objekten sind Agenten der agentenbasierten Oberfläche aktiv, sie können selbständig Aktionen initiieren oder ihren Platz verändern. Sie reagieren auf Eingaben und können Wissen über verschiedene Sachgebiete haben.

[Mase, 1997] unterscheidet aufgrund der Fähigkeiten 3 Gruppen visueller Agenten:

Avatar: Ein Avatar wird ständig von einem Benutzer/Besitzer kontrolliert.

Agent: Der Agent macht selbständig die Bewegungsplanung, so dass der Benutzer von dieser Aufgabe entbunden wird.

Assistent: Ein Agent, der als Assistent eingesetzt wird, erhält Pläne und Ziele als Eingabe, die er weitgehend autonom ausführt. Er überwacht den Benutzer und seine virtuelle Umgebung.

Mit der Hilfe von Agenten ist es möglich, die Komplexität von Problemen besser vor den Benutzern zu verbergen [Norman, 1997]. Agenten führen Aktionen an Stelle des Benutzers aus, auch ohne ihre direkte Steuerung. Eine Gefahr bei der Vermenschlichung von Agenten, d.h. Softwaresystemen, durch die Visualisierung des Agenten durch einen Charakter ist, dass manchmal sehr große, eventuell zu große Erwartungen an das System gerichtet werden. Stattet man einen Agenten mit einer Spracherkennung aus, die dem Agenten gestattet einen begrenzten Sprachumfang zu verstehen, so weckt man unter Umständen bei dem Benutzer die Hoffnung, der Agent könne einen viel größeren Sprachumfang verstehen. Der Agent erscheint viel intelligenter, wenn er natürliche Sprache erkennen kann. Die direkte Programmierung eines Agenten, sei es in herkömmlicher Form oder in grafischer Form, ist nicht adäquat der Ansteuerung des Agenten. Die Bewältigung des Koordinierungsproblems bei zahlreichen Aktionen verlangt nach einer intelligenteren Lösung.

Ein Lösungsweg für dieses Problem beschreibt [Bruderlin et al., 1997]. Hierbei wird eine hierarchisch strukturierte Schnittstelle zur Ansteuerung des Agenten verwendet. Dabei werden auf den unterschiedlichen Schichten verschiedene Abstraktionsstufen eingesetzt, über die der Benutzer mit dem Agenten kommunizieren und ihm Befehle erteilen kann. Auch bei PPP-Persona wird eine Schnittstelle verwendet, die es erlaubt, über verschiedene Abstraktionsstufen den Agenten anzusteuern.

Ein Schnittstellenagent kann folgendermaßen definiert werden [Laurel, 1997]: Ein Schnittstellenagent ist eine computergesteuerte Figur, die an Stelle des Benutzers in der virtuellen Computerumgebung agiert. Da der Charakter und das Aussehen eines animierten Agenten unmittelbar Zu- bzw. Abneigung des Benutzers hervorrufen können, ist es oft nützlich, dem Benutzer Werkzeuge zu geben,

die ihm erlauben, die Eigenschaften des Agenten nach eigenem Geschmack zu modifizieren. Die Benutzung eines Schnittstellenagenten sollte für den Benutzer wahlfrei sein. Nur Benutzer, die solche Agenten einsetzen wollen, sollten sie auch sehen, den übrigen sollten andere Möglichkeiten angeboten werden.

Die Vermenschlichung von unbelebten Objekten ist im täglichen Leben häufig anzutreffen. Viele Gebrauchsgegenstände des alltäglichen Bedarfs erhalten oft Namen oder werden in Formulierungen vermenschlicht. Da liegt es nahe, ein Softwaresystem, also ein eigentlich unsichtbares Subjekt, durch eine Figur zu repräsentieren.

Wird im Alltag ein Objekt, sei es eine Maschine oder aber auch ein Tier, vermenschlicht, werden meist nur die für den Menschen nützlichen Züge vermenschlicht, nicht gleich das ganze Objekt.

Die Darstellung eines Agenten als Figur bringt verschiedene Vorteile [Laurel, 1997]. Durch die Verwendung von Gestik und Mimik kann der aktuelle Systemzustand unabhängig von Sprache und Kultur ausgedrückt werden. Die figürliche Darstellung des Agenten regt zur Konversation an. Die Metapher eines Charakters verdeutlicht schließlich genau die Eigenschaften, die man von einem Agenten erwartet: Ansprechbarkeit,Kompetenz und die Fähigkeit, Aktionen selbständig auszuführen. Der Benutzer möchte keine psychologischen Untersuchungen anstellen müssen, um den Zweck einer Geste des Schnittstellenagenten zu ergründen, oder nicht mühselig ausprobieren, wie der Agent auf eine bestimmte Eingabe reagiert; der Agent sollte ihm keine zusätzlichen Probleme schaffen, sondern ihn unterstützen. Deshalb ist es wünschenwert, wenn das Handeln des Agenten klar und unmittelbar durchschaubar ist.

Charakter können auch ihr Wesen verändern: beispielsweise durch Lernen oder die jeweilige Situation.

Nach [Wilson, 1997] lässt sich die Art der Beziehung zwischen einem intelligenten Programm und einem Benutzer einer der folgenden 4 Gruppen zuordnen:

Werkzeug. Das Programm macht das, was der Benutzer angeordnet hat. Beispiele hierfür sind herkömmliche Programme.

Guru. Das Programm stellt einige Überlegungenen an und liefert dem Benutzer die Ergebnisse. Beispiele hierfür sind medizinische Diagnoseprogramme, die nach Eingabe von Symptomen eine Diagnose stellen können.

Kollege. Der Dialog zwischen Anwender und dem Programm verläuft ähnlich wie unter natürlichen Kollegen, z.B. wenn es darum geht eine zweite Meinung zu einem Problem zu erfahren. Ein Beispiel hierfür ist ein Lernprogramm, das eine Lösung des Schülers kritisiert und Verbesserungsvorschläge macht.

Assistent. Das Programm übernimmt gewisse Teilaufgaben nach besonderer Aufforderung des Benutzers.

Intelligente Agenten mit Benutzerschnittstelle können jede der letzten 3 Rollen übernehmen, wobei allerdings der Schwerpunkt auf der Assistentenrolle liegt.

Weitere Anforderungen an einen animierten Benutzerschnittstellenagenten:

1. Der Agent (bzw. die Benutzeroberfläche des Agenten) muss die Möglichkeit besitzen, Aufgaben vom Benutzer entgegenzunehmen.
2. Um entscheiden zu können, ob der Agent weiterhin die ihm übergebene Aufgabe ausführen soll, ist es notwendig, jede Aktion zu überwachen.
3. Der Agent sollte in der Lage sein, seine Handlungen zu begründen.
4. Es ist wichtig, dass der Benutzer ein mentales Modell entwickelt, das mit den Fähigkeiten des Agenten konsistent ist.
5. Der Agent soll seinen inneren Zustand, Zustandsänderungen und Aktionen visuell repräsentieren.
6. Der Agent soll auf mögliche Benutzeraktionen hinweisen.
7. Der Agent soll die Mehrdeutigkeit anderer Symbole auflösen.

In [Kessler und Kilgore, 1997] finden sich weitere wichtige Punkte, die bei dem Design glaubhafter animierter Agenten beachtet werden sollten:

Skript: Eine Drehbuchvorlage hilft bei dem Design eines glaubhaften Auftretens

Zusammenarbeit: Wichtig ist, dass die künstlerische Ausgestaltung und das Design in enger Zusammenarbeit mit den Entwicklern erfolgt.

Aktionsliste: Zunächst muss eine Menge von Aktionen festgelegt werden, die der Agent ausführen kann.

Verbindung einzelner Aktionen: Die Übergangsstellen zwischen den einzelnen Aktionen müssen sich für die Kombination der Aktionen zu komplexen Aktionen eignen. Im einfachsten Fall endet jede Geste in einer einzigen definierten Körperhaltung. Dadurch kann es allerdings zu einer Verminderung der Glaubwürdigkeit kommen. Besser ist es (auch aus Gründen der Komplexität bei vielen Aktionen), unterschiedliche Übergänge zuzulassen.

Close-Loop Interactions (CLI): Interaktionen mit dem Benutzer werden oft durch eine Schleife realisiert , in deren Hauptteil der Agent Aktionen ausführt (wie Eingabe verarbeiten, Präsentationen ausführen).

komplexe Verzweigungshierachie: Je komplexer die Verzweigungshierarchie ist, umso abwechselungsreicher wird die Präsentation und umso lebendiger wirkt der Agent.

Idle-Time-Action: Hilft Lebendigkeit des Agenten auszustrahlen.

Sonstiges: Ein Agent sollte möglichst wenig Wiederholungen zeigen. Er sollte sich den Dialogverlauf merken können. Außerdem sollte er seine Aufgabenstellung durch seine Handlung deutlich machen.

Sprachausgaben: Der Agent sollte sprechen können: Er wirkt so intelligenter.

Benutzerschnittstellen, die auf direkter Manipulation der dargestellten Objekte beruhen, setzen voraus, dass der Benutzer die Ausführung jeder Aktion initiiert, abwartet und überwacht [Maes, 1997]. Diese Herangehensweise ist für ungeübte Benutzer nicht immer unmittelbar einzusehen (z.B. werden Buttons mehrmals hintereinander geklickt, obwohl die alte Aktion noch nicht abgeschlossen wurde.).

Agenten unterstützen den Benutzer, indem sie die Komplexität einzelner Aktionen vor ihm verbergen. Sie übernehmen an Stelle des Benutzers verschiedene Aufgaben. Sie können als Trainer oder Lehrer eingesetzt werden (Tutor-Systeme). Sie unterstützen den Benutzer bei der Zusammenarbeit (CSCW). Sie überwachen Prozesse und Aktionen (von Systemen und von Benutzern). Die Einsatzmöglichkeiten sind vielfältig: Auswahl von Mail, Musik, Filmen, Terminabsprache usw.. Am einfachsten für den Benutzer wäre es, wenn der Agent sich selbständig benutzeradaptiv situationsgerecht aktiviert. Dann muss ein Benutzer nicht die Notwendigkeit für den Einsatz eines Agenten erkennen und initiieren. In diesem Fall muss der Agent über eine große Wissensbasis verfügen. Der Benutzer hat keine genauen Vorstellungen über das tatsächliche Wissen des Agenten, z.B. wenn durch maschinelle Lernverfahren Agenten in die Lage versetzt werden, selbständig Informationen über den Benutzer zu erlernen. Durch diese Lernmethode ist der Agent auch befähigt, Erklärungen für bestimmte Reaktionen zu liefern. Der Einsatz solcher Agenten ist für Benutzer weniger aufwendig und der Agent kann sich besser auf neue Angelegenheiten bzw. Interessen des Benutzers einstellen.

Beobachtet ein Mail-Agent den Benutzer beim Lesen von Mail, so kann er durch Anwendung von Lernverfahren dessen Gewohnheiten erkennen und nach dessen Vorbild selbständig Mail bearbeiten (z.B. Sortieren, Löschen, in Ordner sortieren). Bei dem in [Maes, 1997] vorgestellten System wird der aktuelle interne Zustand des Agenten (z.B. arbeitend) durch ein gezeichnetes Gesicht dargestellt (Abbildung 2.2).

Analog zum Mail-Agenten wurde ein Kalenderagent implementiert. Ein Nachrichtenagent bietet die Möglichkeit, Themengebiete zu spezifizieren, nach denen er sucht. Das Spezialgebiet eines Agenten wird durch sein visuelles Erscheinungsbild dargestellt (z.B. Sportler: Sportnachrichten) (Abbildung 2.3).

Animierte Agenten können auch dazu eingesetzt werden, um Informationen, die auf Web-Seiten zu finden sind, menschlicher darzustellen. Mit Agenten, die zur Analyse natürlichsprachlicher Eingaben fähig sind, kann ein Gesprächsdialog geführt werden [Focus, 1997].

2.4 Agentenumgebungen

Die im Folgenden beschriebenen Systeme realisieren eine Umgebung, in der sich die Agenten in einer virtuellen Welt aufhalten, wobei der Benutzer die Möglichkeit hat, mit diesen zu interagieren. Bei diesen Systemen steht nicht die Präsentation von Daten oder die Unterstützung des Benutzers bei der Lösung von Aufgaben im Vordergrund, sondern eine möglichst natürlich wirkende Simulation der kommunikativen Prozesse zwischen den Agenten und dem Benutzer.

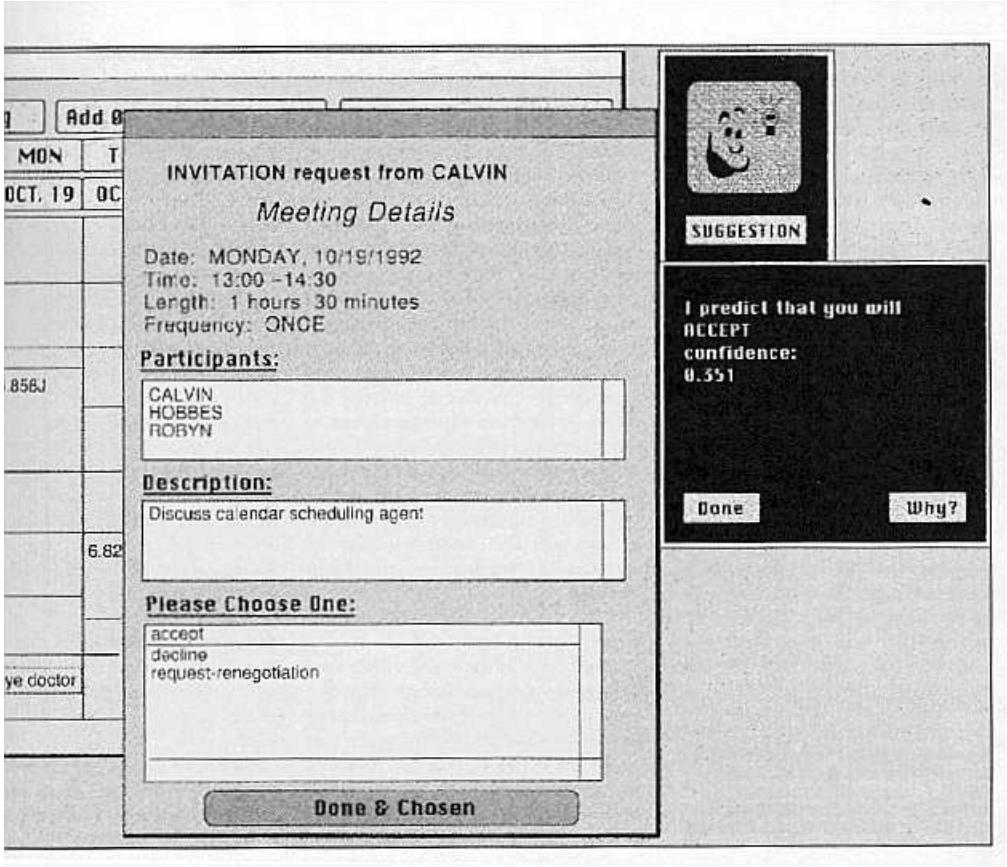


Abbildung 2.2: Der interne Zustand des Kalenderagenten wird durch eine Karikatur visualisiert (aus [Maes, 1997]).

Virtuelle Agenten, die in virtuellen Welten vorkommen, lassen sich in zwei Gruppen untergliedern [Thalmann, 1997]: Zum einen werden diese Agenten von einer realen Person gesteuert, die sich durch den Agenten in der virtuellen Welt repräsentieren lässt. Andererseits kommen auch Agenten vor, die selbständig agieren bzw. ein komplexes System in der virtuellen Welt repreäsentieren. Ein virtueller Agent nimmt seine Umgebung durch Sensoren war, die ihm ein passendes Abbild seiner Umgebung liefern, damit er seine Entscheidungen treffen kann (z.B. Simulation von Sehen, Hören, Greifen). Gesten lassen sich zur nichtverbalen Kommunikation zwischen den einzelnen Agenten einsetzen.

Im Folgenden werden einige Beispiele von Oberflächenagenten beschrieben. Bei der Implementierung dieser Systeme wurden unterschiedliche Ziele verfolgt. Einige Agenten, wie z.B. Steve, dienen zu Ausbildungszwecken in einem Tutorialsystem, andere Systeme, wie z.B. das Oz-System, dienen zur Untersuchung und Erforschung von Agentensystemen.

2.4.1 Das Oz-Projekt

Das Ziel des Oz-Projekts ist die Entwicklung einer Umgebung, die die Konstruktion virtueller Welten erlaubt [Bates et al., 1992b; Reilly, 1996]. In diesen Welten

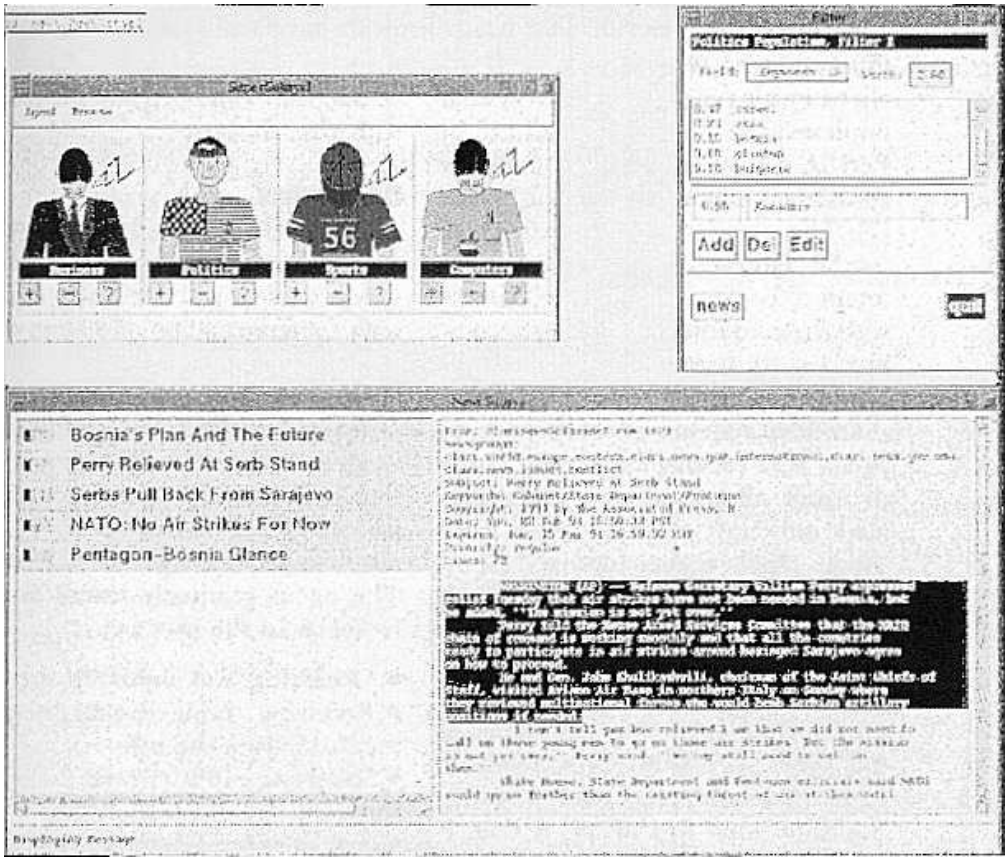


Abbildung 2.3: Passend zu dem Themengebiet der Nachrichten wird eine äußere Erscheinungsform des Agenten gewählt (aus [Maes, 1997]).

kann der Benutzer mit den in der Welt befindlichen Agenten kommunizieren und so an dem Geschehen in der Welt teilnehmen. Die Agenten kommunizieren auch mit anderen Agenten. Unter den implementierten Agenten befinden sich Lyotard [Reilly und Bates, 1992], ein simulierter Kater, und die Woggles [Bates, 1994]. Der Autor einer virtuellen Welt beschreibt die Eigenschaften und Fähigkeiten der simulierten Bewohner mittels spezieller Definitionssprachen. Die zugrunde liegende Agentenarchitektur Tok erlaubt Reaktivität, Zielverfolgung, Emotion und soziales Verhalten der Agenten. Tok besteht aus mehreren Teilkomponenten [Bates et al., 1992a]:

Hap: Modul zur Reaktionsgenerierung und Zielverfolgung. Hap wählt passend zur aktuellen Situation (Weltzustand, Ziel, emotionaler Zustand, interner Zustand des Agenten) ein Ziel aus, das entsprechend einem fest abgespeicherten Planspeicher aufgelöst wird. Dabei findet kein Planen im herkömmlichen Sinne statt, da alle Aktionen abgespeichert vorliegen. Hap speichert alle aktiven Ziele und Pläne in dem sogenannten Active-Plan-Tree (APT). Der Baum wird entsprechend der erfüllten Teilziele bzw. der neu hinzugekommenen Ziele expandiert oder kontrahiert [Bates et al., 1992b]. Hap unterstützt reaktives Verhalten (Zielverfolgung kann zeitwei-

se unterbrochen werden, neue Ziele können generiert werden, Ziele können bei der Verfolgung eines anderen Zieles nebenbei zufällig erreicht worden sein, Zielverfolgung kann ganz abgebrochen werden, es können gleichzeitig mehrere Ziele verfolgt werden).

Em: Modul zur Behandlung von Emotionen und Behandlung von Beziehungen zwischen Agenten. Em modelliert emotionale und soziale Aspekte des Agenten [Reilly, 1996]. Während Hap arbeitet, kommt es vor, dass Ziele neu erzeugt werden, erfüllt werden oder fehlschlagen. Diese Informationen benutzt Em, um den inneren Gefühlszustand des Agenten zu modellieren. Beispielsweise führt das Erreichen eines Zieles zur Steigerung der inneren Zufriedenheit des Agenten, während das Verfehlen eines Zieles die Freude dämpft. Je nach Wichtigkeit des Zieles ändert sich auch der innere Gefühlszustand. Dabei können sich einzelne Emotionen gegenseitig beeinflussen. Der Agent kann ebenfalls zu bestimmten Objekten positive oder negative Einstellungen zeigen. Em erlaubt die Übersetzung des aktuellen Gefühlszustandes in passende Gesten in Abhängigkeit von dem jeweiligen Agenten. Auf diese Art sind unterschiedliche Persönlichkeiten modellierbar [Reilly und Bates, 1992].

Gump: Gump erlaubt es den Agenten textuelle Eingaben zu verarbeiten.

Glinda: Glinda ist das regelbasierte Sprachgenerierungsmodul von Oz [Kantrowitz und Bates, 1992]. Ausgehend von einer formalen Beschreibung des Satzinhaltes erzeugt Glinda eine natürlichsprachliche englische Aussage.

Es gibt zwei Oberflächen zu Oz. Während die eine Version eine reine textuelle Kommunikation zulässt, gibt es für die Woggles auch eine grafische Benutzeroberfläche (Abbildung 2.4).

Das Oz-Projekt legt den Schwerpunkt auf die Modellierung und die Artikulation von Emotionen und Eigenschaften der beteiligten Agenten [Reilly und Bates, 1992; Bates et al., 1992a]. Sie leben und interagieren miteinander in einer virtuellen Welt, die von einem Autor geschaffen wird. Die Handlung in Oz verläuft analog zu dem vom Autor vorgegebenen Schema. Um den Übergang zwischen den Szenen zu erreichen setzt das Oz-System verschiedene Techniken ein, die ihren Ursprung beim Film haben. Diese Techniken, die eigentlich für visuelle Effekte entwickelt wurden, sind beim Oz-System an die textuelle Ausgabe adaptiert worden [Smith und Bates, 1989].

Im Gegensatz zu den bei der Entwicklung von PPP-Persona verfolgten Grundzielen ist nicht die Präsentation von Daten oder Wissen das Hauptziel, sondern die Entwicklung von glaubwürdigen Agenten, die zur Unterhaltung des Benutzers die virtuellen Welten bevölkern.

2.4.2 Steve

Der pädagogische Agent Steve (Soar Training Expert for Virtual Environments) [Rickel und Johnson, 1997] unterstützt das Erlernen von Arbeitsvorgängen. Steve ist Teil einer Virtuell Reality Applikation. Steve wird hauptsächlich in einem Lernsystem zum Umgang und Reperatur verschiedener Maschinen und Geräten

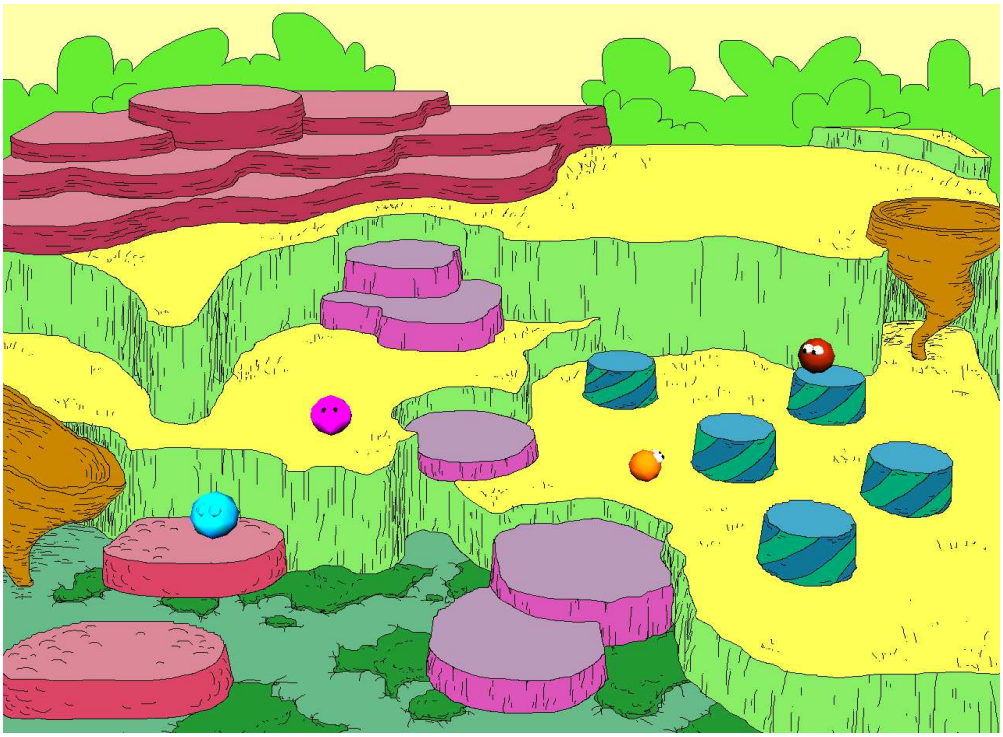


Abbildung 2.4: Oz: Woggles in ihrer virtuellen 3D-Welt (aus [Bates, 1994])

eingesetzt. Im Gegensatz zu gewöhnlichen Tutorsystemen kann er, da er einen „Körper“ besitzt, Aktionen demonstrieren. Der Schüler hat es so leichter, sich eine Handlung oder Aktion vorzustellen. Steve beobachtet ständig Veränderungen in seiner virtuellen Welt und reagiert auf diese. Außerdem beobachtet Steve auch die Reaktionen seines Schülers. Daher kann das System bei erkennbaren Lernschwierigkeiten eingreifen und die Präsentation entsprechend abändern und frühzeitig Fehler des Lernenden korrigieren. Steve wird durch einen Kopf und eine Hand, die dreidimensional modelliert sind, visuell repräsentiert (Abbildung 2.5).

Steve überprüft jeweils, welche Teilziele schon erreicht wurden. Er zeigt und erklärt falls gewünscht die eventuell noch restlichen nötigen Ziele. Durch das ständige Überprüfen der erreichten Teilziele und die gegebenenfalls notwendige Reaktion auf noch nicht erreichte bzw. wieder rückgängig gemachten Teilziele kann sich Steve an den Zustand der virtuellen Welt anpassen, was mit abgespeicherten Präsentationen nicht möglich wäre. Da Steve sich die Zustände merkt, die zur Ausführung einer Handlung geführt haben, bzw. die kausalen Zusammenhänge in den Plänen, kann er Begründungen für seine Aktionen liefern. Durch die körperliche Darstellung wird es überhaupt erst möglich, physikalische Aktionen realistisch zu demonstrieren. Reichhaltigere Präsentationen ergeben sich aus dem Einsatz von Gestik und Mimik.

In neueren Versionen des Steve-Systems können mehrere Agenten eingesetzt werden, so dass auch die Demonstration von Aktionen möglich ist, die die Zusammenarbeit mehrerer Personen erfordert.

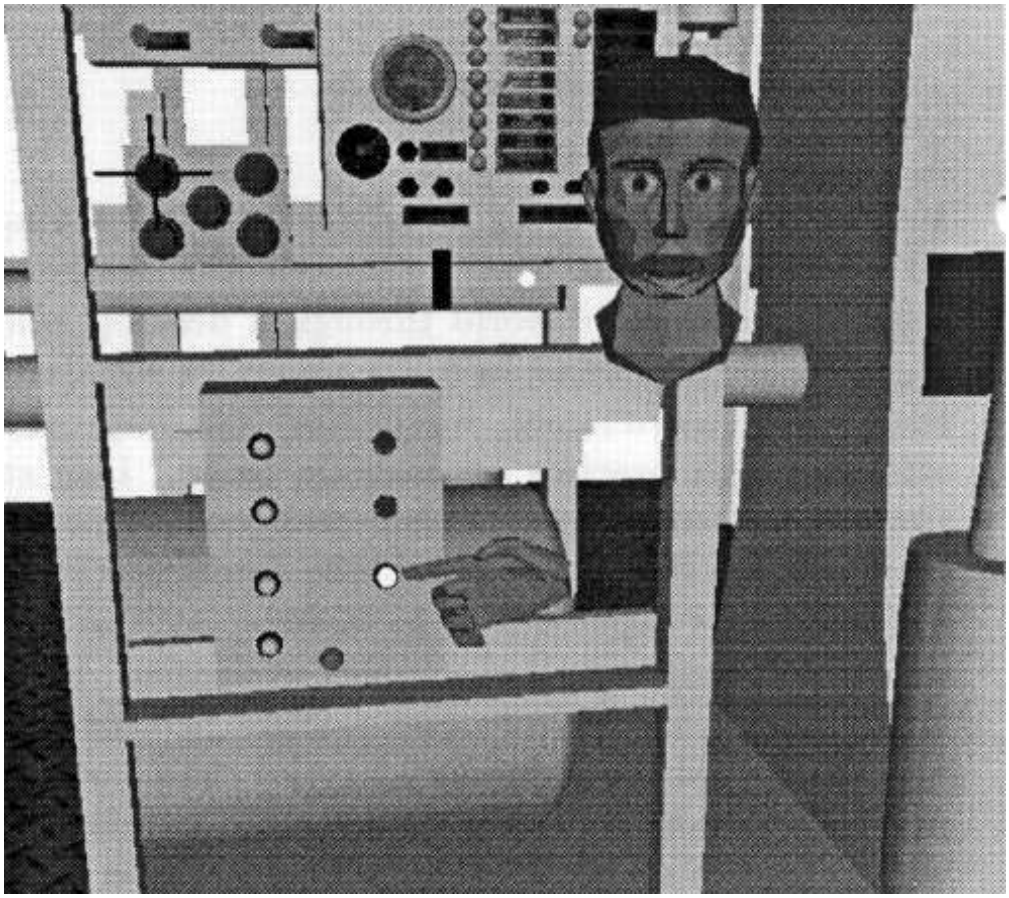


Abbildung 2.5: Der pädagogische Agent Steve (aus [Rickel und Johnson, 1997])

2.4.3 Merlyn

Merlyn ist ein autonomer intelligenter Agent, der dazu dient, Kinder durch virtuelle Welten zu geleiten und es ihnen so erlaubt, auf einfache Weise diese virtuellen Welten zu erkunden und Erfahrungen zu sammeln [Doyle und Hayes-Roth, 1997]. Manchmal können die virtuellen Welten reine Phantasiewelten sein, in denen der Benutzer z.B. Abenteuer bestehen muss (text-basierte Adventure-Spiele, Multi-User Dungeon (MUD) [Bahl, 1996]). Eine weitere Anwendungsmöglichkeit sind „virtuelle Klassenzimmer“, in denen der Schüler sich mit Hilfe von Merlyn ein Thema erarbeitet. Durch seine integrierte Persönlichkeit (er ist als alter und weiser Mann modelliert) stellt er mehr einen Begleiter und Führer dar als ein einfaches Informationswerkzeug. Merlyn hat die Fähigkeit, Informationen aus der virtuellen Welt des MUD's in Erfahrung zu bringen. Über bestimmte Standardfragen, die die meisten MUD's beantworten können, versucht Merlyn seine Umgebung automatisch kennenzulernen, um die nötigen Schlüsse für eine Führung ziehen zu können. Daneben besteht aber auch die Möglichkeit, die virtuelle Welt zu annotieren, so dass Merlyn auch weitergehende Hilfestellungen leisten kann. Merlyn beherrscht verschiedene einfache Gemütszustände (z.B. er lacht, falls man ihn anlacht, oder er wird durch viele Akionen müde).

Merlyn ist ein Agent, der in textbasierten MUD-ähnlichen Umgebungen als Führer eingesetzt werden kann. Da er keine grafische Oberfläche hat, kann er sich nicht selbst visualisieren noch die zu präsentierenden Daten.

2.4.4 Das Olga-Projekt

Das Olga-Projekt [Beskow und McGlashan, 1997] verwendet einen animierten Agenten, um Verbraucherinformationen zu Mikrowellenherden zu präsentieren. Der Agent wird durch eine 3D-modellierte Figur visualisiert (Abbildung 2.6), die in der Lage ist, Gesten auszudrücken und Sprache zu verwenden. Dabei wird die Sprache lippensynchron generiert. Die Verhaltensteuerung des Olga-Agenten erfolgt über ein Regelsystem und parametrisierte Templates. Nach den Autoren ist ein wichtiger Grund für den Einsatz eines animierten Agenten die bessere Verständlichkeit der gesprochenen Erläuterungen, da die Gestik und Mimik des Agenten das Verstehen positiv beeinflussen. In einem von den Autoren durchgeführten Experiment führte der Einsatz des animierten Agenten zu einer Steigerung der Verständlichkeit einer Präsentation von 30% (reine synthetisierte Sprache) auf 47% (synthetisierte Sprache in Verbindung mit Mimik) [Beskow et al., 1997]. Die Ansteuerung des Agenten erfolgt über die Skriptsprache TCL/TK.

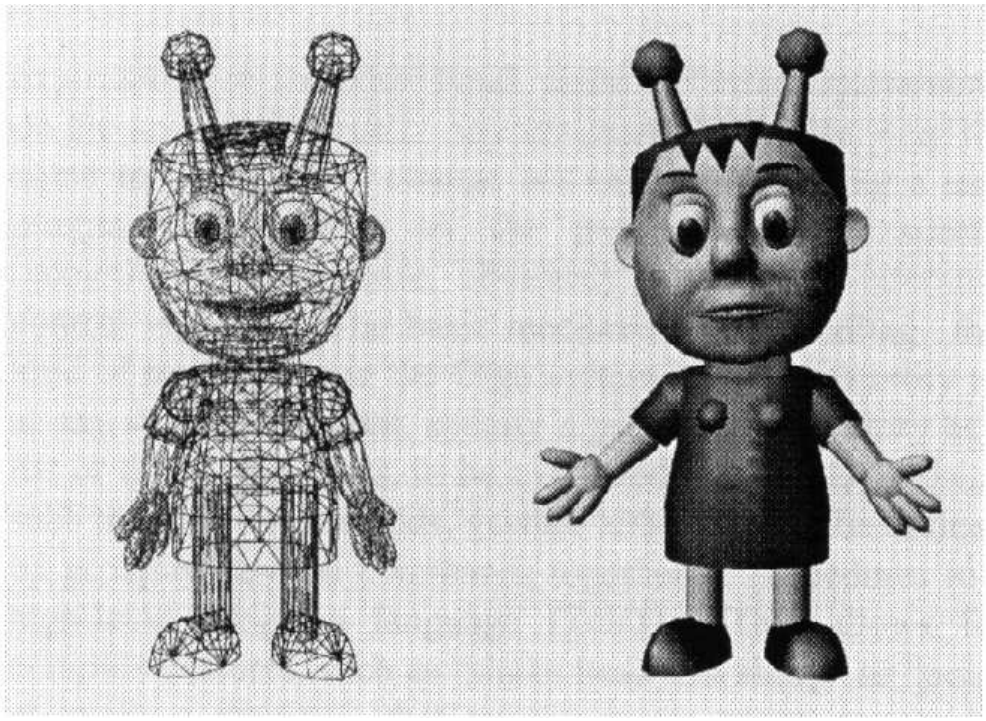


Abbildung 2.6: Der Agent aus dem Olga-Projekt (aus: [Beskow und McGlashan, 1997])

2.4.5 Jack

Jack [Badler et al., 1997] ist ein System, das ursprünglich dazu entwickelt wurde, Maschinen wie z.B. Autos oder Baumaschinen schon während ihrer Konstruktion auf Ergonomie zu testen. Dazu realisiert Jack eine 3D-modellierte menschliche Figur, die sich analog den natürlichen Bewegungsmöglichkeiten eines realen Menschen bewegt. Durch Jacks interne Konzeption sind nur solche Bewegungen erlaubt, die auch ein Mensch ausführen kann. Dadurch ist es z.B. Autokonstrukteuren schon während der Konstruktion möglich, die Sitzposition des Fahrers daraufhin zu testen, ob Bedienungselemente gut erreichbar sind.

Die Ansteuerung des Agenten erfolgt in zwei Stufen. Die Steuerungskomponente, die die Ansteuerung auf primtiver Ebene erledigt, ist SCA (Sense, Control, Action). Sie besteht aus einem Netz von Prozessen, die Sensorknoten (Erkennung von Weltzuständen), Kontrollknoten (Ansteuerung der Aktionsknoten auf Basis von Sensordaten) und Aktionsknoten (Ansteuerung des Menschmodells) definieren. Das Haupteinsatzgebiet von SCA ist die Bewegungsplanung. Die Ansteuerung des Systems auf abstrakterem Niveau wird über die PaT (Parallel Transition Networks) -Netze durchgeführt. Sie werden durch einen endlichen Zustandsautomaten implementiert, wobei die einzelnen Knoten aus Prozessen, anderen PaT-Netzen oder aus spezialisierten Planern/Inferenzsystemen bestehen können. Es werden die Knoten in der Reihenfolge ihrer Priorität ausgeführt, die vom aktuellen Knoten erreichbar sind und deren Constraints erfüllt sind.

Viele täglichen Aktionen des Menschen laufen analog bestimmter Schematas ab. Solche Schemata ergeben sich aus dem kulturellen Umfeld, sind persönliche Eigenheiten oder werden durch äußere Vorschriften vorgegeben (z.B. Spielregeln). Manche Schemata werden ohne nachzudenken und ohne Variationen immer wieder neu ausgeführt, während andere Schematas an die jeweiligen Gegebenheiten angepasst werden müssen. Es ist möglich, dass Menschen mehrere Schematas gleichzeitig benutzen. Solche Schematas aus dem realen Leben werden in dem System als Skript umgesetzt (s.a. Abschnitt 4.2).

Das Verhalten des Agenten wird von vielen Faktoren bestimmt. Wenn der Agent in dem Zustand „müde“ ist, dann wird beispielsweise die Laufgeschwindigkeit des Agenten beeinflusst. Die internen Zustände (wie z.B. „Müdigkeit“) werden vom Animator eingestellt.

Der Animator gibt dem System Befehle auf hohem Abstraktionsniveau. Das System versucht dann das Ziel entsprechend dem aktuellen Weltzustand selbständig zu erfüllen.

Auf Basis des animierten Agenten Jack wurde ein Präsentationsagent entwickelt [Noma und Badler, 1997]. Die Eingabe für den virtuellen Präsentator besteht aus einem mit speziellen Kommandos annotierten Text. Der Text wird durch den Präsentator vorgelesen und die Kommandos werden von ihm ausgeführt. Innerhalb des Präsentationssytems gibt es eine Tafel, auf die benutzerdefinierbare Objekte geschrieben werden können (Abbildung 2.7). Der Präsentationsagent kann auf diese Objekte zeigen und Erläuterungen geben. Die Körperbewegungen des Agenten werden durch ein Netz gesteuert, dessen Knoten jeweils für bestimmte Teilaufgaben einer Geste zuständig sind (z.B.

Sprechen, Gehen). Die Ansteuerung erfolgt entweder über eine Datei- oder über eine Socket-Schnittstelle.

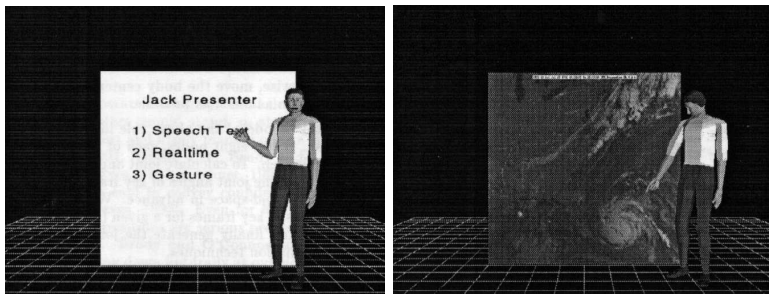


Abbildung 2.7: Jack als Präsentationsagent (aus: [Noma und Badler, 1997])

Der Autor beschreibt drei wichtige Anforderungen an einen virtuellen anthropomorphen Präsentator:

- Der Agent sollte sich so natürlich wie möglich bewegen.
- Die Bewegungsgenerierung sollte in Echtzeit erfolgen. Dabei sollten Bewegungen von synchronisierter Sprache begleitet werden.
- Die Benutzerschnittstelle zu dem Präsentationsagenten sollte es Anwendern ermöglichen, neue Präsentationen zu erstellen, ohne sie detailiert vorgeben zu müssen. Der Anwender soll mit dem Agenten auf abstrakterem Niveau kommunizieren können.

Die Darstellung des Präsentators als menschliche Figur erlaubt es, verschiedene, insbesonders der Kommunikation dienende Methoden einzusetzen. Durch die Körperhaltung und Gesten können die Präsentationen einprägsamer dargestellt werden. Eine Zeigegeste kann beispielsweise durch eine Augenbewegung in Richtung des bezeichneten Objekts eingeleitet werden, so dass der Betrachter auf die Geste eingestellt ist und diese dann schneller erfasst. Überhaupt spielen Augenbewegungen bei der menschlichen Kommunikation eine wichtige Rolle.

In [Kessler und Kilgore, 1997] werden weitere wichtige Eigenschaften eines animierten Schnittstellenagenten beschrieben:

- Der Agent ist getrennt vom Computer, aber er kennt ihn.
- Der Agent weiß alles über die Applikation.
- Der Agent kennt die Aktionen des Benutzers.
- Der Agent kennt sich selbst.
- Der Agent hat „Gedanken“ und „Gefühle“.
- Der Agent verhält sich wie ein Experte auf seinem Gebiet.
- Der Agent gibt dem Benutzer nicht das Gefühl, dumm zu sein.

Bei der Entwicklung eines Agenten für eine spezielle Applikation ist es wichtig, dass die Applikationsentwickler und die Agentenentwickler eng zusammenarbeiten, um eine gute Integration des Agenten in die Applikation zu erreichen. Um einen Agenten möglichst lebendig erscheinen zu lassen, ist es wichtig, die Handlungen des Agenten abwechslungsreich zu gestalten (z.B. durch Definition einer Auswahl von äquivalenten Aktionen). Zusätzlich werden Gag-Sequenzen eingestreut. Ruheaktionen sollten mehrstufig angelegt sein, d.h. je nach Länge der Ruhephase werden passende Aktionen ausgewählt. Werden vom Agenten Gemütszustände intern gespeichert, lassen sich komplexe Verhaltensweisen generieren, die nicht fest einprogrammiert wurden. Nach Meinung der Autoren ist gesprochene Sprache für einen Agenten, der lebendig und natürlich wirken soll, wichtig. Nach Untersuchungen der Verfasser kamen Agenten mit Sprachausgabe bei den Benutzern besser an als Agenten ohne Sprachausgabe. Allerdings sollte man dem Benutzer die Wahl lassen, ob er Audioausgabe wünscht oder nicht.

2.4.6 Cosmo

Cosmo [Lester et al., 1997b] ist ein Agent, der eine Lernumgebung für das Erlernen des Routings im Internet bereitstellt. Grund für die Entwicklung dieses animierten Agenten in einer solchen Lernumgebung war: Steigerung des Lerneifers. Die Glaubwürdigkeit des Agenten fördert die Motivation entscheidend. Agenten, die virtuelle Welten „bewohnen“, müssen Objekte in ihrer Welt so behandeln, wie wirkliche Menschen mit Objekten umgehen. Das bedeutet, dass sie fähig sein müssen, Objekte über Sprache und Zeigegesten zu annotieren. In einem Tutorsystem, wie es Cosmo als Agent des Internet Advisors darstellt, müssen Zeigeoperationen und sprachliche Objektreferenzen eindeutig sein. Das System muss also in der Lage sein, Mehrdeutigkeiten zu erkennen und diese zu beseitigen (z.B. durch explizite Zeigegesten). Das Planungssystem des Internet-Advisors generiert zunächst Erklärungen (auf Anfrage, bei Zögern oder Fehlern des Lernenden). Diese Erklärungen werden dann einer Komponente übergeben, die eventuelle Mehrdeutigkeiten beseitigt. Die so erzeugten Präsentationsaufgaben werden danach der Steuerungskomponente des Assistenten übergeben. Das Bildmaterial für den Internet Advisor wurde mittels 3D-Software erstellt und gerendert. Im laufenden System werden nur noch die Bitmaps verwendet (Abbildung 2.8).

Cosmo ist ein pädagogischer Agent. Sein Aufgabengebiet ist es, den Betrachter über ein Themengebiet zu informieren [Johnson et al., 2000]. Weitere pädagogische Agenten sind Herman the Bug [Lester et al., 1999a], ein System, das Kindern helfen soll, den Aufbau von Pflanzen zu verstehen, und WhizLow [Lester et al., 1999b]. WhizLow dient zur Erklärung und zum Verständnis des Aufbaus von Computersystemen.

2.4.7 IMPROV

Das IMPROV-Projekt [Goldberg, 1997] beschäftigt sich mit der Entwicklung virtueller 3D-Welten, in denen sowohl vom Computer als auch von Menschen gesteuerte Agenten miteinander interagieren können.

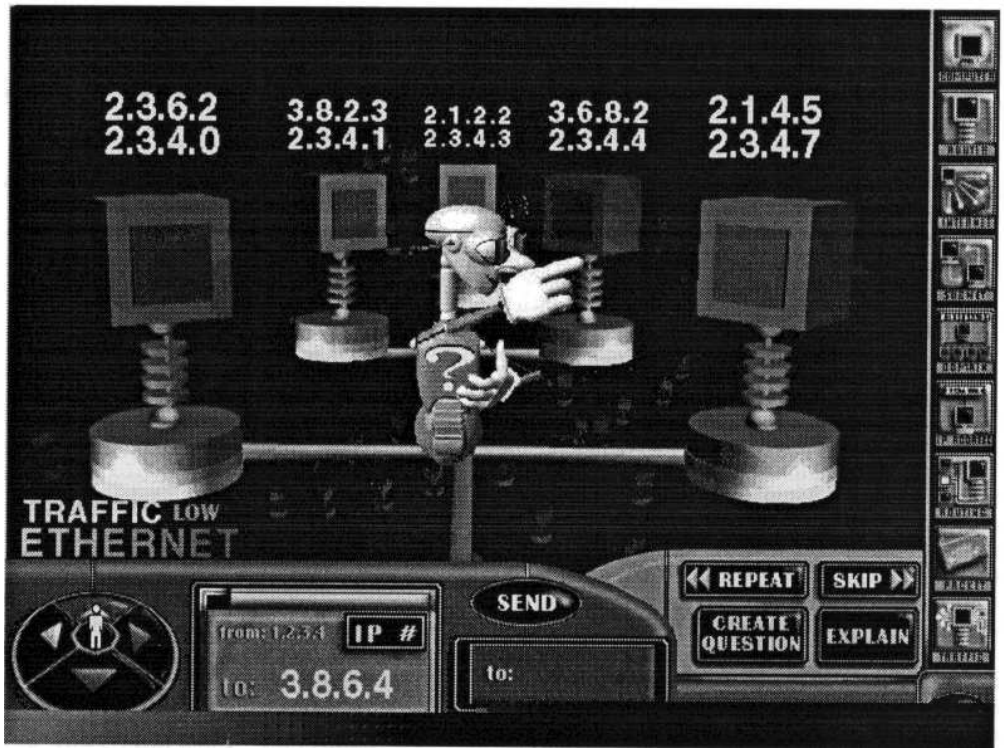


Abbildung 2.8: Cosmo (aus: [Lester et al., 1997b])

Es wird eine Architektur zur Ansteuerung von Agenten vorgeschlagen, die aus mehreren Ebenen besteht. Das Behaviour-System verwaltet ein Netzwerk von Behaviours. Entsprechend dem aktuellen Zustand wird eine passende Aktion ausgewählt und an das Bewegungssystem (Motor Controller) weitergegeben.

In IMPROV ist eine Aktion entweder als atomare Aktion oder als eine Folge anderer Aktionen definiert. Einem Agenten ist es möglich, gleichzeitig mehrere Aktionen durchzuführen. Da nicht alle Aktionen parallel ablaufen können, werden die Aktionen in Klassen unterteilt, wobei nicht gleichzeitig mehrere Aktionen einer Klasse bearbeitet werden können. Werden zwei Aktionen aus einer Klasse gleichzeitig gestartet, so wird die erste abgebrochen. Mehrere Aktionen können in einem Skript zusammengefasst werden. Dabei können einzelne Aktionen aus einer Liste gewichteter Aktionen bestehen, von denen jeweils eine passende Aktion ausgewählt wird.

Jeder Agent kann eine Liste mit Attributen verwalten, die bei der Auswahl von passenden Aktionen evaluiert werden. Über die Attribute wird das Verhalten des Agenten charakterisiert. Eine Besonderheit des IMPROV-Projekts ist es, dass auch Objekte in der virtuellen Welt und sogar die Kamera als Agent betrachtet werden können. Daher können auch neben den Agenten weitere Objekte spezielle Verhaltensweisen zeigen.

2.4.8 Agenten von Microsoft

Heute werden Computer überwiegend als Werkzeuge eingesetzt. Zukünftig werden sie zunehmend als Assistenten genutzt werden [Ball et al., 1997]. Immer mehr Details werden vor dem Benutzer verborgen. Softwaresysteme werden im stärkeren Maße größere Aufgaben selbständig und autonom erledigen.

Eine Benutzerschnittstelle für einen Computer-Assistenten benötigt weitreichendere Möglichkeiten als die herkömmlichen Oberflächen bieten können, die auf der Direktmanipluation von Objekten beruhen.

Assistenten klären im Dialog Probleme, präsentieren ihre Lösungsvorschläge und Ergebnisse auch ohne unmittelbare Aufforderung.

Assistenten treffen Entscheidungen auch ohne Zutun ihres Besitzers. Sie brauchen nicht die Zustimmung zu jeder Entscheidung einzuholen, da sie über das notwendige Domänenwissen und Benutzerprofil verfügen.

Muss das System dennoch den Benutzer bei seiner Arbeit unterbrechen, so sollte es solche Momente auswählen, in denen der Benutzer nicht unnötig gestört wird (d.h. das System muss den aktuellen Auslastungsgrad des Benutzers ermitteln können und entsprechend intelligent reagieren).

Agenten sollten die sozialen und emotionalen Umstände, unter denen die Benutzerinteraktion abläuft, berücksichtigen. Je besser sich der Agent auf den Benutzer (z.B. Tageszeit, Stimmung) einstellt, um so schneller akzeptiert ihn der Anwender.

Die natürlichste Art einen Dialog mit einem Agenten zu führen ist nach [Ball et al., 1997] der Dialog in natürlicher gesprochener Sprache. Der Agent muss in der Lage sein gesprochene Sprache zu erkennen und zu verarbeiten. Er sollte auch in natürlicher gesprochener Sprache antworten.

Ein Agent des Microsoft Persona Projects, der Papagei Peedy, dient als Assistent zur Auswahl von Musik CDs.

Das System kann Spracheingabe verarbeiten. Die Dialog-Komponente entscheidet, wie Peedy auf Eingaben reagiert. Die Figur selbst benutzt Sprachausgabe, Animation und Soundeffekte, um lebendiger zu wirken.

Der Benutzer kann mit Peedy sprechen, ohne eine spezielle Kommandosprache zu verwenden. Dabei besteht das Repertoir aus ungefähr 150 typischen (für eine Applikation in CD-Zusammenhang) Ausdrücken. Aus diesen möglichen Eingaben werden 17 Befehlsformen extrahiert, auf die Peedy reagiert.

Die Dialog-Komponente startet entsprechend einem erkannten Befehl eine passende, vordefinierte Befehlsfolge, die Peedy ausführt. Dabei ist die gewählte Aktion von vorherigen Eingaben abhängig. Zur Spracherkennung wird das Whisper-System von Microsoft Research eingesetzt. Alle möglichen Benutzeräußerungen werden für das System als kontextfreie Grammatik definiert. Entsprechend der Eingabe wird das passende Template ausgewählt.

Der Microsoft Agent [Microsoft, 1997] ist eine Active-X-Komponente, eine animierte Figur, die sich frei über den Bildschirm bewegen kann. Obwohl sich der Agent auch aus einem WWW-Browser (dem Internet Explorer) heraus ansteuern lässt, setzt der Einsatz dieser Technologie doch die Installation der entsprechenden Active-X Komponente auf dem Client-Rechner voraus. Der Agent hat folgende Eigenschaften:

- Die Figur befindet sich immer oberhalb der anderen Fenster.
- Der Benutzer kann die Figur mit der Maus verschieben.
- Sie verfügt über ein bestimmtes Repertoire an Idle-Time-Gesten
- Klickt der Benutzer auf die Figur, so zeigt sie ein Pop-Up-Fenster, das mögliche Kommandos (für Agent und für Applikation) anzeigt.
- Der Agent ist fähig, Sprechblasen und Ausgaben mittels Sprachsynthesizer zu machen.
- Der Agent kann Spracheingabe verwenden.
- Der Agent hat keinen Zeigestock oder ein ähnliches Werkzeug (Lupe o.ä.) für Zeigegesten zur Verfügung.
- Da der Agent absolute Koordinaten relativ zum Hauptfenster für Positionsangaben verwendet, sind Zeigegesten schwierig zu koordinieren. Insbesondere wenn der Benutzer den Agenten mit der Maus verschiebt. Darüberhinaus wird der Einsatz in WWW-Präsentationen dadaurch erschwert, dass der Agent keine Repräsentation des Präsentationsmaterials hat. Zeigt er z.B. auf ein Element einer WWW-Seite und der Benutzer verschiebt oder schließt das Browser-Fenster, so reagiert Genie nicht auf diesen Event, er zeigt immer noch auf diese, nun falsche Position.

Abbildung 2.9: 2 Microsoft-Agenten: Peedy als CD-Ratgeber (aus: [Ball et al., 1997]) und Genie als Active-X-Komponente (aus: Microsoft [1997])

2.5 Andere Systeme

In [Huard und Hayes-Roth, 1997] wird ein System beschrieben, mit dessen Unterstützung Kinder interaktiv Geschichten erfinden und durch Charaktere darstellen lassen können. Durch Ansteuerung des Charakters über eine interaktive Benutzeroberfläche kann der Fortgang der Geschichte beeinflusst werden (z.B.

Emotionen, Befehle für Figur). Die Schnittstelle stellt jeweils die aktuell möglichen Aktionen dar, so dass nur eine begrenzte Eingabemenge möglich ist. Der Charakter der Figur lässt sich über Schieberegler steuern.

[Lachman, 1997] beschreibt den Einsatz des als Active-X Komponente implementierten Microsoft Agenten [Microsoft, 1997] in verschieden Systemen als Teil der jeweiligen Benutzeroberfläche. Animierte Agenten können dazu eingesetzt werden, Systemzustände zu visualisieren. Mit der Hilfe dieser Agenten kann der Benutzer auf Besonderheiten wie z.B. eingehende Mail aufmerksam gemacht werden, ohne ihn von seiner Arbeit zu stark abzulenken. Die vom Agenten gezeigte Geste kann der Wichtigkeit des Ereignisses angepasst werden. Außerdem kann der Agent auch dazu eingesetzt werden, einen Benutzer auf wichtige Elemente von WWW-Seiten hinzuweisen (Oberflächenagent in Maitre-D). Nach Meinung des Autors müssen Idle-Gesten eines Schnittstellenagenten begründet sein.

In [Mase, 1997] wird der Einsatz eines Agenten als Museumsführer vorgestellt. Eine anderes System dient als Assistent, der an einem Gespräch von Personen teilnehmen und hilfreiche Informationen geben kann.

[Piesk und Trogemann, 1997] beschreibt ein System, das durch einen 3D-Agenten auf unterhaltsame Art Lehrinhalte vermitteln will (Abbildung 2.10). Ein Autor legt die Struktur der Lehreinheiten in einer Hypertextstruktur ab. Die Struktur dient als Ausgangsbasis für die Generierung. In der Hypertextstruktur werden mögliche Reaktionen des Benutzers antizipiert und der Autor gibt an, wie das System darauf reagieren soll.

Abbildung 2.10: Der Agent aus [Piesk und Trogemann, 1997]

Gestik und Mimik unterstützen und ergänzen die verbale Sprache. Die Gesamtdarstellung wird einprägsamer und die anthropomorphe Gestalt ermöglicht eine emotionale Beziehung zum Agenten und damit zum Lehrmedium. Die Mensch-Maschine-Kommunikation wird durch die sich wechselseitig ergänzende verbale und nonverbale Sprache verbessert. Durch die Dialogform wird aktives Lernen gefördert.

Emotionen und Gestik des Agenten sollen in diesem System den Lernenden

motivieren und das Feedback liefern (freundlicher Gesichtsausdruck bei wahren oder trauriger Gesichtsausdruck bei falschen Antworten). Durch den Unterhaltungseffekt kann der Lehrinhalt einprägsamer vermittelt werden. Das emotionale Verhalten der Figur wird durch den Autor im Dialogskript definiert. Das System wählt entsprechend den aktuellen Emotionen die Gesten aus und spielt die passenden Animationen aus einer zuvor generierten Animationsbibliothek ab.

Das System Letizia [Lieberman, 1995] beobachtet das Verhalten des Benutzers beim Surfen durch das WWW. Das System analysiert die vom Benutzer betrachteten Seiten und erstellt ein Benutzer-Profil auf dieser Basis. Letizia sucht parallel zum Benutzerprofil passende Seiten und zeigt diese dem Benutzer an. Sobald der Benutzer auf eine andere Seite wechselt, verwendet Letizia diese Seite als neue Startseite für die Suche. Letizia ist ein Beispiel für einen Agenten ohne grafische Oberfläche (in Form einer anthropomorphen Gestalt).

[Dohi und Ishizuka, 1997] beschreibt das System VSA, das dem Benutzer die sprachgesteuerte Ansteuerung des Netscape WWW-Browsers erlaubt. Dabei wird das System dem Benutzer mittels eines 3D-animierten Gesichts präsentiert. Auf diese Weise kann der Benutzer auf natürliche Art mit dem System interagieren. Das 3D-Gesicht wird durch Texture-Mapping einer Fotografie auf ein 3D-Modell eines Kopfes berechnet. Das Spracherkennungssystem reagiert auf Befehle wie „Zeige mir URL x“ oder „Zurück“ usw. Der Suchagent des VSA-Systems kann nur solche Seiten bearbeiten, auf denen der Benutzer die für ihn relevante Informationen markiert und klassifiziert hat. Diese Technik ist insbesondere dann von Nutzen, wenn die Information auf der Seite häufig geändert wird (z.B. Wetterinformationen, Datenbankergebnisse), aber das Layout der Seite konstant bleibt. Es wird die Vermutung geäußert, dass eine anthropomorphisierte Oberfläche mit Spracheingabe hauptsächlich für einen Benutzerkreis interessant ist, der keine oder wenig Erfahrung im Umgang mit Computern hat, bzw. für Benutzer mit Behinderungen. Erfahrene Benutzer können ein solches System schneller mit Maus und Tastatur bedienen.

Das System Tigrito [Maldonado et al., 1997] erlaubt es Kindern, über Schieberegler die Stimmung einer auf dem Bildschirm dargestellten Figur zu beeinflussen. Entsprechend der Einstellung werden passende Aktionen ausgewählt und dem Benutzer zur Wahl gestellt. Die gewählte Aktion wird ausgeführt. Eine zweite Figur auf dem Bildschirm reagiert auf diese Aktionen und verändert ihre interne Stimmung, die ebenfalls visualisiert wird, entsprechend. Passend zur aktuellen Stimmungslage wird eine passende Quicktime-Videosequenz abgespielt, die die Agenten gemäß ihrer Stimmungslage und Aktion visualisieren.

[Binsted, 1998] beschreibt ein System, das ein Spiel, gespielt mit dem Robo-Cup Simulator, kommentiert. Der Kommentator wird durch einen Kopf visualisiert, der mittels eines Sprachsynthesizers automatisch generierte Kommentare zu annotierten Spielszenen generiert. Dabei ergreift das System Partei für eine Mannschaft und kann auch emotional auf Ereignisse reagieren. Als Eingabe erhält das System eine Beschreibung der Spielsituation, die in bestimmten Zeitabständen auf den neuesten Stand gebracht wird. Die einzelnen Fakten sind mit einem Prioritätsmaß versehen, dementsprechend werden die Kommentare erzeugt. Das Emotionsmodul verwaltet Regeln, die der aktuellen Situation auf

dem Spielfeld und dem bisherigen emotionalen Zustand gemäß eine der sechs grundlegenden Emotionen (Angst,Traurigkeit,Ärger,Fröhlichkeit, Ekel, Überraschung) [Ekman und Rosenberg, 1997] mit jeweils einer bestimmten Intensität versehen, dem Generator für Sprache und Mimik zuleiten, der die passenden Äußerungen generiert.

Viele animierte Figuren werden als Avatare in Chat-Umgebungen eingesetzt, also als Stellvertreter ihres Besitzers. Sie werden von ihrem Besitzer gesteuert. Von diesen Systemen gibt es einfache zweidimensionale Versionen, in denen Comic-Figuren mit Sprechblasen Verwendung finden [Microsoft, 1998a] und komplexe 3D-Systeme auf VRML-Basis [Fellner und Hopp, 1997], [Interactive, 1997b],[Interactive, 1997a],[Sony, 1997],[Honda et al., 1997],[Mitsubishi Electric, 1999].

Ein Präsentationsagent kann auch eine Vertreterfunktion in CSCW-Systemen einnehmen. Ein Beispiel für ein CSCW-System ist das System des CoNus-Projekt von Siemens ZFE [Hohl, 1996], Abbildung 2.11. Dieses System ermöglicht die Zusammenarbeit von mehreren Personen in virtuellen Räumen, wobei Personen durch Piktogramme symbolisiert werden. In diesem System haben die Piktogramme allerdings nur die Aufgabe, die momentan anwesenden Personen statisch zu visualisieren.

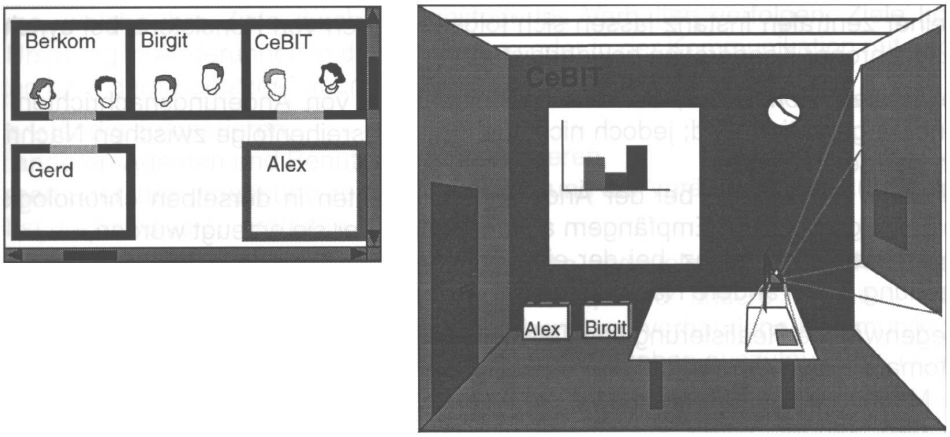


Abbildung 2.11: CSCW-System des CoNus-Projekts von Siemens ZFE [Hohl, 1996]

Das in [Spierling et al., 1997] vorgestellte System erweitert die bislang vorherrschende Schreibtischmetapher um den Begriff des Virtuellen Sekretariats. Das System unterstützt die Bereitstellung von Agenten für die unterschiedlichsten Büroaufgaben (z.B. Terminkoordination, Email-Sortierung, Telefondienst usw.). Dabei können die Agenten durch einen animierten Agenten mit dem Benutzer in Kontakt treten und mit ihm kommunizieren (z.B. durch Mimik). Der

Oberflächenagent visualisiert die internen Systemzustände (Abbildung 2.12).

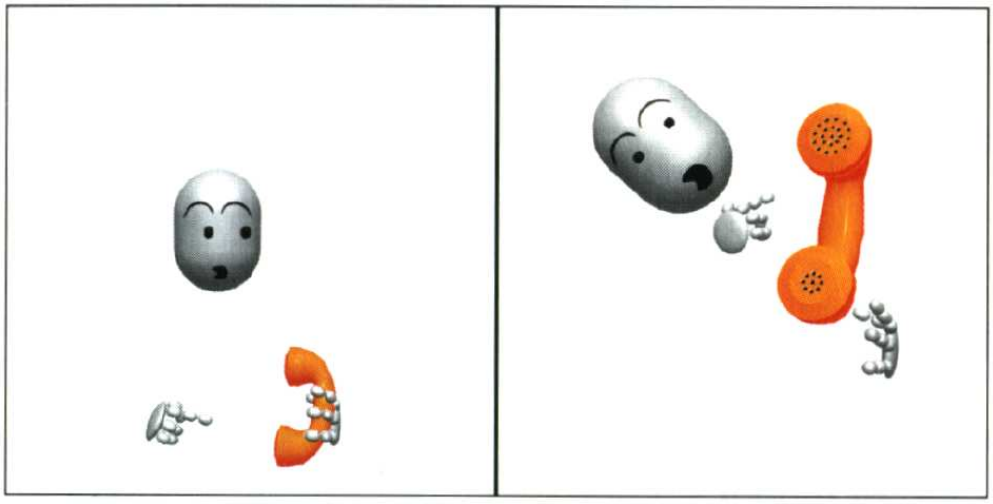


Abbildung 2.12: Virtuelles Sektretariat mit Visualisierung des Systemzustandes durch einen Oberflächenagenten (aus: [Spierling et al., 1997])

Agent 7 ist eine animierte Figur, die in Powerpoint-Präsentationen und in WWW-Seiten (Installation eines besonderen Players ist notwendig) eingesetzt werden kann [7th Level, 1998]. Der Autor einer Präsentation hat die Wahl zwischen verschiedenen Charakteren. Die Sprachausgabe wird durch Digitialisierung der natürlicher Sprache über ein Mikrofon geleistet. Die Figur bewegt bei dem Abspielen der Audio-Datei die Lippen.

Das ZDF setzt eine 3D-Figur, die auf Basis von Active-X (analog zu Microsoft Genie) implementiert ist, ein, um den Zuschauer durch die Programmvorschau zu führen [Römer, 1998]. Dazu muss sich der Benutzer zunächst die entsprechende Komponente auf dem Rechner installieren und kann dann die Figur benutzen.

[Kaatz, 1997] beschreibt die Möglichkeit einer Internet-Moderation durch eine animierte Zeichentrickfigur, die vorgefertigte Animationen ausführt.

Das in [Manske und Mühlhäuser, 1997] beschriebene System setzt aus gezeichneten Comic-Strips Animationen für die Animation einer Figur zusammen. Die Figur kann sich frei über den Bildschirm bewegen. Die Übergänge zwischen den einzelnen Aktionen werden von Hand mittels eines speziellen Editors festgelegt. Das System verwendet vorgefertigte Audioclips zur Untermalung der einzelnen Aktionen.

Eine weitere Forschungsrichtung beschäftigt sich mit der Entwicklung des interaktiven Kinos [Post, 1998]. Zuschauer können in die Rolle einzelner Darsteller schlüpfen und in die laufende Handlung eingreifen . Darüber hinaus wird an der Erkennung menschlicher Gefühle bei der Interaktion gearbeitet (z.B. über Sprache, Körperhaltung, Mimik).

2.6 Visualisierungs- und Animationstechniken

Bei der Entwicklung von computeranimierten Figuren sind zwei Hauptrichtungen auszumachen. Eine Forschungsrichtung hat hauptsächlich Schnittstellenagenten zum Ziel, die in normalen Benutzerschnittstellen eingesetzt werden. Dabei kommt es auf die schnelle Generierung der Aktionen an, wobei allerdings unter Umständen Einbußen bei der Darstellungsqualität hingenommen werden müssen.

Im Extremfall wird auf eine Generierung einer animierten Figur aus vordefinierten Videosequenzen auf Quicktime-Basis zurückgegriffen [Puppettime, 1998].

Die andere Entwicklungsrichtung hat das Ziel, menschliche Modelle möglichst realitätsnah zu visualisieren. Durch den großen Berechnungsaufwand sind diese Techniken oft noch nicht in Benutzerschnittstellen einsetzbar, was bei steigender Computerleistung allerdings machbar erscheint.

Zur Visualisierung von Figuren oder auch Teilen von Figuren wie Kopf oder Händen werden die unterschiedlichsten Techniken angewandt. Die Filmindustrie ist bestrebt, in Filmen virtuelle Charaktere einzusetzen [Freyermuth, 1997]. Es werden viele qualitativ hochwertige Animationen erzeugt. Durch den großen Rechnenaufwand sind diese meistens allerdings nicht für Realzeitanwendungen benutzbar. In den Filmen treten die Figuren dabei entweder in Kombination mit realen Schauspielern wie in Caspar [Industrial Light & Magic, 1995] oder Jurassic Parc [Industrial Light & Magic, 1993] auf oder die Filme sind vollständig computergeneriert wie Toy Story [Pixar, 1995]. In diesem Anwendungsgebiet kommt es auf eine möglichst fotorealistische Darstellung (z.B. mit Hilfe von Ray-Tracing) und realitätsnahe Animation der Figuren an. Die Animation der Drahtrahmenmodelle ist aufwendig und arbeitsintensiv. Einerseits werden Daten durch Digitalisierung von Menschen oder Tieren gewonnen [Cyberware, 1998], die dann animiert werden. Momentan werden andererseits die Modelle in den meisten Fällen noch per Hand bzw. durch Motion Capture animiert. Bei Motion Capture steuert ein Animator durch seine eigenen Bewegungen die Bewegungen des Modells. Der nächste Schritt ist das autonome Handeln der virtuellen Schauspieler. Es werden verschiedene Verfahren zur Verbesserung der Figurenansteuerung entwickelt. Wichtig ist in diesem Zusammenhang die Kollisionserkennung zur Wegplanung, aber auch zur Modellierung von Kleidung für die Figuren [Magnenat-Thalmann und Volino, 1997]. Hierzu wird die Kleidung wie ein eigenständiges Objekt behandelt. So ist es zum Beispiel möglich, dass die Figur ein Kleidungsstück auszieht und auf den Boden wirft. Gleichzeitig werden Akteure mit der Fähigkeit ausgestattet, mit ihrer virtuellen Welt in Kontakt zu treten. Sie verfügen über Seh-, Hör- und Berührungssensoren, so dass eine komplexe auf der Umgebung beruhende Aktionsplanung ermöglicht wird [Thalmann et al., 1997] (z.B. Laufen, Springen, Greifen). Viele dieser Techniken sind allerdings noch nicht für Animationen in Realzeit geeignet. Das Problem ist Gegenstand weiterer Forschung.

Teilweise geht die Entwicklung von animierten 3D-Modellen von Menschen oder auch Tieren in Richtung von exakten Simulationen auf Grund physikalischer oder medizinischer Gesetzmäßigkeiten. Das System,das in [Koch et al.,

1997] beschrieben wird, generiert beispielsweise mit Hilfe eines mathematischen Modells (3D-Federsystem) Gesichtsausdrücke eines 3D-Kopfes. Dabei werden Muskeloperationen und Interaktionen zwischen den einzelnen Komponenten simuliert.

Der Gesichtsausdruck ist ein wesentlicher Bestandteil menschlicher Kommunikation. Daher wird in vielen Projekten an einer Simulation und Visualisierung eines 3D-modellierten Gesichtes gearbeitet. Gesucht wird ein Modell, das komplexe Vorgänge beim Mienenspiel einfach simulieren kann.

In [Morishima et al., 1997] wird ein auf der Simulation der Muskelkontraktion beruhendes Modell des menschlichen Kopfes verwendet, um realistische Mundbewegungen lippensynchron mit der Sprachausgabe zu generieren. Die zum Aufbau der Datenbasis benötigten Daten wurden durch Auswertung von Videosequenzen mit Mundbwegungen natürlicher Personen gewonnen. Mit der momentan zur Verfügung stehenden Hardware ist es noch nicht möglich, diese rechenintensive Animation in Realzeit durchzuführen.

[Hasegawa und Sakaue, 1997] beschreibt ein System, das dreidimensional modellierte Gesichter visualisiert und das Steuerungsbefehle zur Animation über eine Socket-Schnittstelle empfangen kann. Das System kann 4 Gesichtsausdrücke darstellen: Lächeln, Überraschung, Ärger, Ratlosigkeit. Dabei kann der Gesichtsausdruck auch nur für einen Teil des Gesichtes definiert werden. Das System stellt außerdem einen Rumpf mit 2 Armen zur Verfügung, die ebenfalls angesteuert werden können.

Auch andere Körperteile oder Organe werden zu Lehrzwecken simuliert, aber auch für Tests, z.B. in der Automobilindustrie. Beispielsweise beschreibt [DeCarlo et al., 1998] die Modellierung und Simulation einer Lunge.

In vielen Fällen ist aber für die Aufgabe eines Präsentationsagenten eine so genaue und rechenaufwendige Simulation der einzelnen Körperfunktionen nicht notwendig.

2.7 Zusammenfassung

Anthropomorphe Agenten werden als nützliche Erweiterung von Benutzeroberflächen angesehen. Besonders ihr Einsatz als Präsentationsagent kann den Wert einer Präsentation erheblich steigern. Durch ihn wirkt eine Präsentation einprägsamer und unterhaltsamer. Der Betrachter kann sich besser an die einzelnen Elemente der Präsentation erinnern.

An einen Präsentationsagent werden verschiedene Anforderungen gestellt. Der Agent sollte ein umfangreiches Gestenmaterial zur Verfügung haben, um möglichst viele unterschiedliche Gesten darstellen zu können, so dass die Präsentation nicht langweilig wirkt. Wichtig in diesem Zusammenhang sind Vitalaktionen, also Aktionen, die der Agent ausführt, wenn er keine besonderen Aktionen ausführen soll. Diese Vitalaktionen bewirken eine größere Lebendigkeit des Agenten. Der Einsatz als Präsentationsagent erfordert eine Repräsentation der dargestellten Präsentationselemente. Nur so kann der virtuelle Präsentator auf Elemente zeigen und sie beschreiben. Damit der Betrachter die Präsentation mit der Figur als eine zusammenhängende Applikation wahrnimmt

ist es wünschenswert, den Agenten in die Präsentation zu integrieren. Steht er getrennt von ihr, so kann es geschehen, daß die Figur als nicht zu der Präsentation gehörend erscheint und deshalb auch ihren Zweck nicht vollständig erfüllen kann.

Die Darstellung des Agenten als menschenähnliche Figur gestattet es einerseits Präsentationsmethoden anzuwenden, die der Betrachter von Vorträgen realer Menschen gewohnt ist (z.B. Zeigegesten, Sprache, Lupe), und andererseits auch Techniken anzuwenden, die nur vom Computer her bekannt sind (Knöpfe, Schieberegler, Textfenster). Durch die Kombination beider Welten ist es so möglich, komplexe interaktive Präsentationen zu erstellen.

Um den Präsentationsagenten in Präsentationssysteme integrieren zu können, muss er über entsprechende Schnittstellen verfügen. Die Ansteuerung muss möglichst einfach und auf hohem Niveau erfolgen. Die Fähigkeiten des Agenten müssen sich leicht an neue Domänen anpassen lassen, und er muss auch mit automatisch generierten Präsentationen arbeiten können. Durch Aufbau von Gesten- bzw. Aktionsbibliotheken für einen Agenten, die entsprechend der Anwendung instantiiert werden müssen, kann der Aufwand beim Anpassen an neue Domänen verringert werden.

Das Einsatzgebiet eines Präsentationsagenten ist einerseits in normalen Desktop-Appliaktionen, andererseits aber wegen der fortschreitenden Verbreitung des Internets auch immer mehr das WWW. Um keine zu großen Anforderungen an die Hardware zu stellen, ist es momentan notwendig, die Gesten des Präsentationsagenten aus einzelnen vorberechneten (oder gezeichneten) Bitmaps zusammenzusetzen.

In Tabelle 2.1 werden verschieden Systeme verglichen, die Oberflächenagenten nutzen. Vergleichskriterien sind:

anthropomorphe Visualisierung: Visualisierung des Agenten in anthropomorpher Darstellung

Einsatzgebiet: Haupteinsatzgebiet des Agenten

WWW-Einsatz: Einsatzfähigkeit des Agenten im World Wide Web

Desktop-Einsatz: Einsatzfähigkeit in Desktop-Applikationen

Aussehen änderbar: Einfache Änderung des visuellen Erscheinungsbildes

Erweiterbarkeit: Anpassbarkeit und einfache Erweiterungsfähigkeit

Interaktion: Benutzerinteraktion möglich (Sprache, Bedienelemente)

Sprachsynthese: Einsatz von Sprachsynthese

Audio-Import: Integration von Audioclips (Sprache, Effekte)

Zeigegesten: Möglichkeit von Zeigegesten

Datenvisualisierung: Bereitstellung von Datenvisualisierungsverfahren

Datenrepräsentation: Interne Repräsentation präsentierter Daten

Plattformunabhänigkeit: Lauffähigkeit auf verschiedenen Hardware-Plattformen

Präsentationssystem: Integration in Präsentationssystem

Kriterium	Oz	Steve	Merlyn	Olga	Jack	Cosmo	Peedy	Genie	PPP-Persona
anthr. Visualisierung	ja	ja	nein	ja	ja	ja	ja	ja	ja
Einsatzgebiet	Unterhaltung	Training	Unterhaltung	Training	Präsentation	Ausbildung	Unterhaltung	Oberflächenagent	Präsentation
abstrakte Kommandos	nein	nein	nein	nein	nein	nein	nein	nein	ja
WWW-Einsatz	nein	nein	nein	nein	nein	nein	nein	ja	ja
Desktop-Einsatz	ja	ja	ja	ja	ja	ja	ja	ja	ja
Aussehen änderbar	nein	nein	nein	nein	nein	nein	nein	ja	ja
Erweiterbarkeit	nein	nein	nein	nein	ja	nein	nein	nein	ja
Interaktion	ja	ja	nein	nein	nein	nein	ja	ja	ja
Sprachsynthese	nein	ja	nein	ja	ja	ja	ja	ja	ja
Audio-Import	nein	nein	nein	nein	nein	nein	nein	nein	ja
Zeigegesten	nein	Hand	nein	Hand	Hand	Hand	nein	Hand	Hand/ Zeigestock/ Lupe/ Laserpointer
Datenvisualisierung	nein	nein	nein	nein	nein	nein	nein	nein	ja
Datenrepräsentation	nein	nein	nein	nein	nein	nein	nein	nein	ja
Plattformunabhängig	nein	nein	nein	nein	nein	nein	nein	nein	ja
Präsentationssystem	nein	nein	nein	nein	nein	nein	nein	nein	ja

Tabelle 2.1: Vergleich einiger Systeme

Kapitel 3

Konzeption eines Präsentationsagenten

Ein Präsentationsagent kann für eine Präsentation im Vergleich zu herkömmlichen Präsentationen neue Präsentationstechniken einsetzen. Diese umfassen einerseits gestische und mimische Äußerungen, die der Agent aufgrund seines anthropomorphen Erscheinungsbildes nutzen kann, andererseits ergeben sich durch die geleitete Führung durch die Präsentation neue Darstellungsformen.

3.1 Gesten und Mimik

Gesten und Mimik haben einen wichtigen Anteil an der zwischenmenschlichen Kommunikation [Krause, 1995]. 40% des sensorischen Eingangs erfolgen über den Sehnerv. Unter Gestik im engeren Sinne wird die Information übermittelnde Bewegung der Hände und Arme verstanden. Im weiteren Sinne sind es die Signale aussendenden Änderungen der Körperhaltung als sprachbegleitendes Element; Gesten können jedoch auch unabhängig von der Sprache eingesetzt werden, Beispiele sind Kopfnicken und Kopfschütteln. In nahezu allen Kulturen (Ausnahmen z.B. Griechenland, Bulgarien, Teile Indiens) bedeutet ein Kopfschütteln „nein“ (Entwicklung aus einer Verstärkung des Wegbewegens des Kopfes) , ein Nicken des Kopfes „ja“ (Entwicklung aus der Verstärkung einer Art einer Demutsgeste).

Die menschliche Mimik wird seit langer Zeit untersucht. Der schwedische Anatom C.H. Hjortsjö entwickelte Schemazeichnungen des Gesichts mit genau definierten Veränderungen. Er schuf das „Normalgesicht“, bei dem die mimische Muskulatur ganz entspannt ist (Abbildung 3.1).

Durch Anwendung der Hjortsjö-Aktionseinheiten ergeben sich die unterschiedlichen mimischen Äußerungen. In [Ekman und Friesen, 1978] wird ein System, das Facial Action Coding System (FACS), vorgestellt, das es ermöglicht, alle mimischen Veränderungen des Gesichts zu beschreiben. Es basiert auf der Aufzählung aller 46 „Action Units“, die die Gesichtsveränderungen bewirken. Durch ihre Kombination ist es möglich, eine große Anzahl von mimischen Äußerungen des Gesichts zu beschreiben. Es wird im Allgemeinen von den sechs Grundemotionen Freude,Trauer, Wut, Ekel, Überraschung und Angst

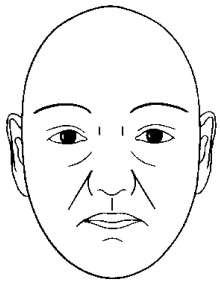


Abbildung 3.1: Von C.H. Hjortsjö entwickeltes Normalgesicht (aus [Krause, 1995]).

ausgegangen.

Das Facial Action Coding System läßt sich zur Ansteuerung eines simulierten Gesichts nutzen, das als zusätzlicher Kommunikationskanal in einem Dialogsystem dienen kann [Knop, 1997].

Im Allgemeinen erkennen die Menschen die jeweiligen Gefühle gut aufgrund der entsprechenden Mimik, es werden durchschnittlich 70% Genauigkeit erreicht [Elliott, 1997]. Die nationale Zugehörigkeit spielt dabei keine große Rolle. Tabelle 3.1 zeigt die Erkennungsrate der 6 Grundemotionen bei Menschen unterschiedlicher nationaler Zugehörigkeit.

Nation	Freude	Überraschung	Trauer	Angst	Ekel	Wut
Estland	90	94	86	91	71	67
Deutschland	93	87	83	86	61	71
Griechenland	93	91	80	74	77	77
Hong Kong	92	91	91	84	65	73
Italien	97	92	81	82	89	72
Japan	90	94	87	65	60	67
Schottland	98	88	86	86	79	84
Sumatra	69	78	91	70	70	70
Türkei	87	90	76	76	74	79
USA	95	92	92	84	86	81

Tabelle 3.1: Erkennungsrate von Emotionen bei Menschen verschiedener Kulturkreise (nach [Krause, 1998])

3.1.1 Visualisierung von Mimik

Da Gestik und Mimik in der Kommunikation zwischen Menschen eine große Rolle spielen, liegt es nahe, die gute Erkennungsfähigkeiten des Menschen für Emotionen auch in Benutzeroberflächen von Computersystemen auszunutzen.

[B.Reeves und C.Nass, 1996] zeigt, dass Benutzer zu ihrem Computer Beziehungen aufbauen, die echten sozialen Beziehungen zu Menschen ähneln.

Stattet man die Benutzerschnittstelle eines Computers mit einem anthropomorphen Interface aus, wird der Rechner vom Benutzer mehr als Partner als ein

reines Werkzeug betrachtet [Isbister, 1995].

Für die Glaubhaftigkeit von Schnittstellenagenten ist es wichtig, dass sie ihren internen Zustand für den Benutzer sichtbar machen und sie ihn so dem Benutzer vermitteln können (z.B. über Gesichtsausdruck, Emotionen) [Sheth, 1994; Lashkari et al., 1994; Kozierok und Maes, 1993].

Je realistischer die Darstellung eines Agenten ist, um so höher sind die Erwartungen des Benutzers an ihn [Laurel, 1990], obwohl auch einfache Zeichnungen ausreichen, um den gewünschten psychologischen Effekt zu erzielen [B.Reeves und C.Nass, 1996].

Nach [Porter, 1997] ist es bei einem Lifelike-Character wichtiger, dass die Figur glaubhaft wirkt bzw. man seine Gedankengänge spürt. Eine realistische (physikalisch korrekte) Darstellung ist nicht so wichtig. Manchmal kann es notwendig sein, bestimmte Aktionen zu übertreiben, um so besser die Ziele des Agenten zu verdeutlichen.

In der Psychologie ist bekannt, dass unterschiedliche Gesichtstypen verschiedene Reaktionen bei Menschen auslösen können. Sympathie oder Ablehnung wird zum Teil auch durch das Unterbewusstsein beim Betrachten des Gesichts erzeugt. Die Wirkung von Gesichtstypen wird beispielsweise am Fachbereich Psycholgie der Universität des Saarlandes untersucht [Marguier, 1998].

Agenten müssen sich auf den Benutzer und dessen Vorstellung von Attraktivität einstellen. Dazu müssen die Agenten ein entsprechendes Modell aufbauen, so dass sie ihre Strategien auf den Benutzer abstellen können [Mark und Voss, 1997].

Umgekehrt gibt es Bemühungen, Computersysteme mit der Fähigkeit auszustatten, die aktuellen Gefühle des Benutzers zu erkennen und entsprechend darauf zu reagieren [Foerst, 1997]. Dabei ermöglicht die Verwendung unterschiedlicher Sensoren, Rückschlüsse auf die aktuellen Emotionen des Benutzers zu ziehen. Durch die Auswertung der Daten kann sich der Agent besser auf den Benutzer einstellen. Gerade in Lernsystemen ist es sinnvoll, die aktuellen Gefühle bzw. Emotionen in die Präsentationsplanung einzubeziehen, da eventuelle Unsicherheiten besser erkannt werden und das System rechtzeitig ohne aktive Mitwirkung des Benutzers die Art der Präsentation des Lehrstoffes beeinflussen und passend aufbereiten kann.

Ein wichtiger Aspekt bei dem Einsatz von animierten Agenten in Lernumgebungen ist die Motivation des Lernenden. Dabei spielen Emotionen eine große Rolle [Elliott et al., 1997]. Ist der Agent in der Lage, Emotionen auszudrücken, wird das Lernen auf folgende Arten vereinfacht:

- Hat der Lernende den Eindruck, der Agent sorge sich um seinen Lernerfolg und nehme an seinen Bemühungen Anteil, kann dies einen starken Motivationsschub bewirken.

- Erkennt der Agent die Emotionen des Lernenden, kann er Schwierigkeiten im Lernvorgang schneller feststellen und frühzeitig berichtigend in den Ablauf eingreifen.

- Agenten, die Emotionen ausdrücken können, haben es leichter, Begeisterung für ein Lerngebiet zu wecken.

- Agenten können durch ihr Erscheinungsbild und durch ihr gesamtes Verhalten für mehr Spaß beim Lernen sorgen.

Dieser „Persona-Effekt“ hat eine positive Wirkung auf den Lernerfolg, der durch ein Lernsystem mit personifiziertem Interface ausgenutzt werden kann.

Im täglichen Leben wird oft die Mimik von gezeichneten Gesichtern verwendet, um Gemütszustände in einer bestimmten Situation vereinfacht und schablonenhaft zu katalogisieren. In Abbildung 3.2 ist ein Bewertungsbogen für den Service an einer Bundesautobahnraststätte dargestellt (Bundesautobahnraststätte Jura-Ost). Insbesondere an einer Autobahnraststätte mit internationalem Publikum ist es wichtig, allgemein verständliche Zeichen zu verwenden. Der Besucher soll seine Erfahrung mit dieser Raststätte ausdrücken, indem er statt eines Benotungssystems auf dem Formular Gesichter mit unterschiedlicher Mimik ankreuzt.

Abbildung 3.2: Bewertungsformular der Bundesautobahnraststätte Jura Ost

Das System Bruzard [Klein, 1998] visualisiert Emotionen durch Animation eines 3D-Gesichts. Bruzard ist fähig, Emotionen und die damit verbundenen Emotionsübergänge zu visualisieren. Dabei werden Emotionsintensitäten und der zeitliche Verlauf einer Emotion berücksichtigt.

Neben echter 3D-Animation werden auch weniger rechenzeitaufwendige Techniken eingesetzt. Ein Woggel [Bates et al., 1992b] kann seinen inneren emotionalen Zustand über seine Farbe signalisieren.

Da Persona als Bitmap-orientiertes Programm konzipiert ist (Abschnitt 4.1), werden die einzelnen emotionalen Zustände als gezeichnete Bitmaps realisiert.

3.1.2 Gesten

Eng verbunden in Bezug auf nonverbale Kommunikation mit Mimik sind die Gesten. Darunter sind die gesprächsbegleitenden Bewegungen von Armen und Händen zu verstehen. Außerdem sind hier auch im erweiterten Sinne alle Bewegungen des Körpers während der Kommunikation gemeint.

Im täglichen Leben spielen Gesten eine wichtige Rolle. Sie sind sogar nach dem Psychologen Lars Feldmann oft noch wichtiger als sprachliche Äußerungen [Junge, 1998].

Die Körpersprache ist ein wesentlicher Bestandteil menschlicher Kommunikation. Bei Vorträgen oder auch bei Verkaufsgesprächen spielt Gestik und Mimik eine große Rolle. Nach [Pease, 1993] können bei Verkaufsgesprächen schon kaum wahrnehmbare Gesten den Verkaufserfolg beeinflussen. Für diese Berufsgruppen werden Kurse und Seminare angeboten, um die Teilnehmer in die Lage zu versetzen, entsprechende Gesten bewusst einzusetzen.

In der Stummfilmzeit mussten Gestik und Mimik besonders stark eingesetzt werden, da sie ja die einzige Kommunikationsform des Schauspielers waren.

Viele wichtige Kommunikationsgesten werden mit den Händen und den Armen ausgeführt (Tabelle 3.2).

Gesten sind teilweise angeboren, können aber auch erlernt sein. Gesten sind oft auch von Kulturkreis zu Kulturkreis unterschiedlich.

Gesten dürfen nie getrennt voneinander betrachtet werden. Sie müssen immer im Kontext aller Gesten interpretiert werden. Einzelne Gesten können unterschiedliche Sachverhalte ausdrücken, je nachdem in welchem Zusammenhang sie stehen. Der Gebrauch der Gesten ist auch abhängig vom Bildungsstand und vom Status einer Person. Gesten verändern sich mit dem Alter. Oft ist zu beobachten, dass Gesten für den gleichen Anlass mit zunehmendem Alter weniger ausgeprägt dargestellt werden.

Die Basis einer Kommunikation ist der Blickkontakt zwischen beiden Gesprächspartnern [Pease, 1993]. Für eine gelungene Kommunikation ist es notwendig, den Gesprächspartner anzusehen: eine wichtige Tatsache für einen Präsentationsagenten, der durch Blickkontakt die Aufmerksamkeit eines Betrachters steuern kann. Ein langer Blickkontakt kann auch dazu beitragen, bei den Gespächspartnern ein gegenseitiges positives Empfinden auszulösen.

Gesten verändern sich mit dem Alter. Oft sind Gesten von Kindern einfacher zu durchschauen, da sie ihre Gefühle häufig nicht so stark unterdrücken wie Erwachsene. In Abbildung 3.3 ist die Entwicklung einer Geste, die oft eine Lüge begleitet, vom Kindesalter bis zum Erwachsenen dargestellt. Mit zunehmendem Alter werden die Gesten abgeschliffen und weniger offensichtlich.

In Büchern zur Rhetorikschulung werden fünf Gestik-Grundpositionen beschrieben, die bei Vorträgen wichtig sind, Abbildung 3.4. Ihre Bedeutung ist (von links nach rechts):

1. Zusammenfasssung - Ganzheit - Offenheit

2. Abweisung - Abwehr - Distanz

3. Betonung - Aufmerksamkeitserzeugung

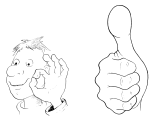
Geste	Deutung
	Eine Geste, die die Handflächen sichtbar macht, erweckt den Eindruck von Offenheit und Wahrheit.
	Hochziehen der Schultern zusammen mit einer Armbewegung, die zu sichtbaren Handflächen führt, wird zum gestenhaften Ausdruck des Nichtverstehens eingesetzt.
	Versuch, Zeit zu gewinnen (um nachzudenken)
	Kragen dehnen: Unsicherheit (Lüge ?, Ärger ?)
	Kratzen am Nacken: Geste der Unsicherheit
	Nase reiben:Ausdruck von Zweifel
	Person möchte nicht weiter zuhören, möchte selbst etwas sagen.
	Ausdruck von Erfolg (Regionsabhängig) Ausdruck einer positiven Erwartungshaltung: Das Reiben der Handinnenflächen aufeinander symbolisiert die Erwartung eines positiven Ereignisses. Die Geschwindigkeit, mit der eine solche Geste ausgeführt wird, gibt Aufschluss darüber, wer in den Augen des Ausführenden der Nutznießer des Erfolgs ist. Bei einer schnellen Geste ist mehr sein Gegenüber gemeint, bei einer langsamen Geste meint er sich selbst. Werden nur die Finger gegeneinander gerieben, so ist dies oft der Ausdruck, der bei Erwartung von Geld gebraucht wird.
	
	Ausdruck der Unaufrichtigkeit
	Nachdenken, Überprüfen einer Angelegenheit
	Ausdruck von Frustation, Ärger (evtl. auch über sich selbst)

Tabelle 3.2: Gesten und mögliche Deutungen (nach [Pease, 1993], verändert)

Abbildung 3.3: Entwicklung einer Geste mit zunehmenden Alter (nach [Pease, 1993])

4. Besondere Sache - etwas Delikates

5. Appell - Nachdruck - geballte Kraft

Abbildung 3.4: Gestik-Grundpositionen (nach [Neumann, 1985])

3.2 Präsentationstechniken

Durch die Benutzung eines personifizierten Präsentationsagenten erschließen sich eine Vielzahl neuer Präsentationstechniken. Obwohl es auch möglich ist, einige der im Folgenden beschriebenen Techniken ohne Präsentator einzusetzen, wirken sie im Zusammenhang mit einer Präsentationsfigur natürlicher und für den Betrachter eingängiger, da er diese Techniken aus dem realen Leben gewohnt ist und sich nicht an neue visuelle Effekte gewöhnen muß.

3.2.1 Zeigegesten

In nichtpersonalisierten Präsentationen werden zur Annotation und zum Hinweis auf wichtige Objekte Pfeile, Farben, Umrandungen oder Ähnliches verwendet (Abbildung 3.5). Bei statischen Annotationen kann das Präsentationssystem nicht sicher sein, dass der Betrachter genau zu dem gewünschten Zeitpunkt die annotierten Objekte beachtet.

In Präsentationen, die unter der Kontrolle eines Präsentationsagenten ablaufen, können diese Techniken durch Zeigegesten des Präsentators ersetzt werden.

Um die Wirkung der Zeigegeste zu unterstützen und zu verstärken, kann die Zeigegeste durch eine vorangehende Kopfdrehung mit Blick des Präsentators auf das hervorzuhebende Objekt eingeleitet werden.

Da die ganze Präsentation zeitlich koordiniert abläuft, besteht für das Präsentationssystem eine höhere Evidenz dafür, dass die Annotation eines Objekts

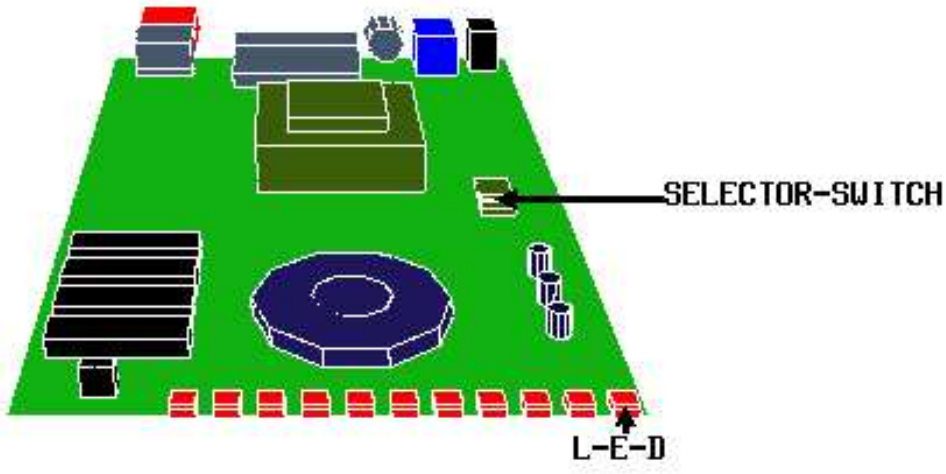


Abbildung 3.5: Annotation von Grafiken mittels Pfeilen

durch eine Zeigegeste vom Betrachter zum richtigen Zeitpunkt wahrgenommen wird.

Dabei sind verschiedene Zeigegesten möglich. Falls der Präsentationsagent nahe genug an das zu bezeichnende Objekt heranreicht bzw. wenn sichergestellt ist, dass er mit seinem Körper keine wichtigen Informationen verdeckt, kann er mit einem Finger auf das Objekt zeigen (Abbildung 3.6, links).

Falls die Gefahr besteht, dass der Präsentationsagent im Laufe der Zeigegeste ein wichtiges Objekt mit seinem Körper verdeckt, so kann es angebracht sein, einen Zeigestock zu verwenden (Abbildung 3.6, rechts).

Abbildung 3.6: Annotation von Grafiken mit Zeigegesten

Die Farbe des Zeigestocks kann angepasst werden, falls eine Grafik besonders hervorgehoben oder der Kontrast zum Hintergrund verstärkt werden soll.

Ändert man dynamisch in zeitlichen Intervallen die Farbe des Zeigestocks, so erhält man einen Blinkeffekt, der dazu verwendet werden kann, eine besondere Aufmerksamkeit des Betrachters zu erzielen.

Werden in einer Präsentation gleichzeitig mehrere Objekte eingesetzt, zwischen denen Beziehungen bestehen, so bietet sich der Gebrauch einer beidhändigen Zeigegeste an. Mit Hilfe einer solchen Geste kann die Beziehung zwischen zwei Objekten besonders unterstrichen werden (Abbildung 3.7).

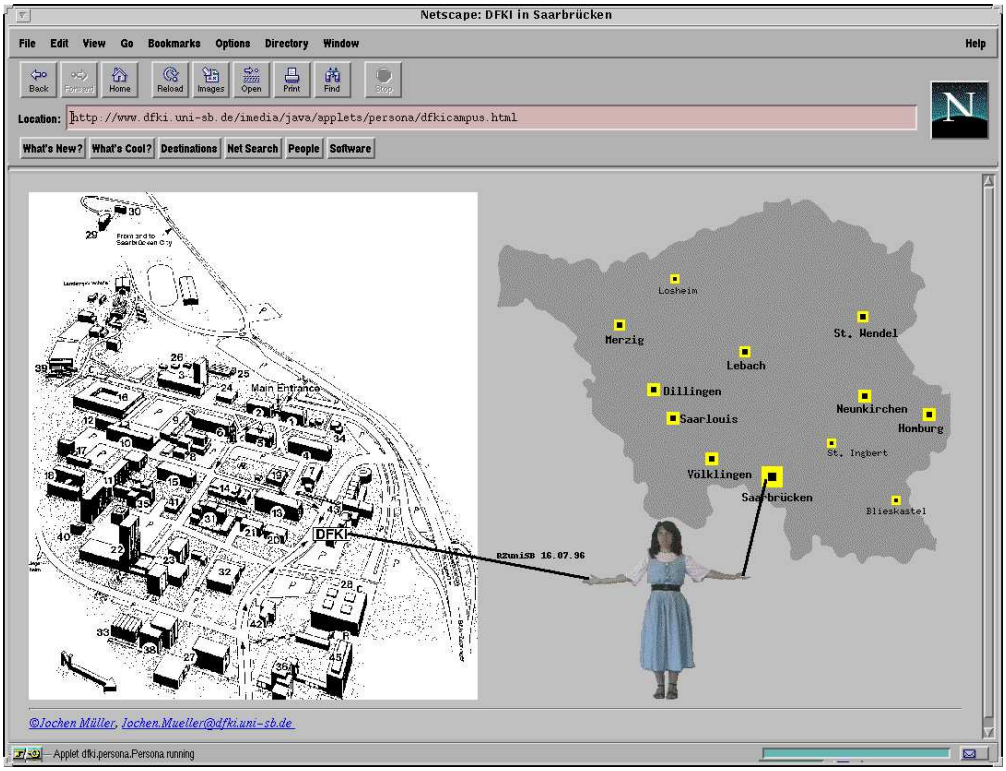


Abbildung 3.7: Verdeutlichung einer Beziehung zwischen zwei Objekten mittels einer beidhändigen Zeigegeste

Während der eine Zeigestock dazu benutzt wird, die Lage des Campus Saarbrücken innerhalb des Saarlandes zu beschreiben, deutet der andere Zeigestock auf die Lage des DFKI-Gebäudes innerhalb der Übersichtskarte des Campus. In diesem Fall stellt die zweihändige Zeigegeste eine Beziehung zwischen zwei Karten mit unterschiedlichem Maßstab her.

Eine andere Anwendung ist die visuelle Herstellung einer Beziehung zwischen einem Objekt und der näheren Beschreibung dieses Objekts. Während ein Zeigewerkzeug auf das zu beschreibende Objekt weist, zeigt das andere auf die genauere textuelle Beschreibung. Gleichzeitig kann der Agent durch synthetische Sprache noch nähere Erläuterungen abgeben.

Diese Art der Beschreibung erlaubt es dem Betrachter sofort ohne näheres Hinsehen festzustellen, dass zwischen den beiden bezeichneten Objekten eine Beziehung besteht, da er ähnliche Zeigegesten aus der realen Welt kennt und einzuordnen weiß.

Neben einem Zeigestock können auch andere Zeigegeräte eingesetzt werden. Dazu zählen beispielsweise Laserpointer (Abbildung 3.8) oder auch die Nutzung einer Taschenlampe, um wichtige Objekte zu beleuchten und somit hervorzuheben. Um die Funktionalität einer Taschenlampe zu integrieren, muss neben der eigentlichen Taschenlampe auch der Lichtkegel implementiert werden. Zu diesem Zweck ist der beleuchtete kreisförmige Bereich heller darzustellen, was durch einen Filter ermöglicht wird, der die Farben der Pixel dort entsprechend aufhellt.

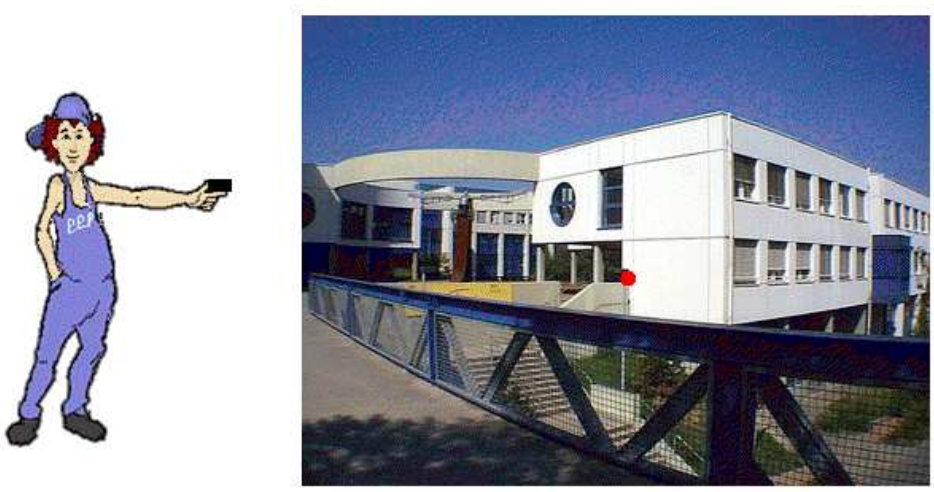


Abbildung 3.8: Ein Laserpointer als Zeigewerkzeug.

3.2.2 Datenquellen für Zeigegesten

Eine wichtige Eigenschaft eines Präsentationsagenten ist die Fähigkeit, Zeigegesten auszuführen. Für diese Zeigegesten benötigt das Präsentationssystem Informationen über die Lage des zu beschreibenden Objekts sowie eine möglichst umfassende Erläuterung, aus der der aktuelle Text, den der Präsentationsagent während der Zeigegeste sprechen soll, generiert werden kann.

Neben den im Folgenden vorgestellten Techniken ist es bei Präsentationen sinnvoll, wenn der Agent auf einzelne Elemente der Benutzeroberfläche, wie z.B. Textfelder, Menüs oder Knöpfe, zeigen kann, um dem Benutzer eine Hilfestellung bei der Bedienung des Programmes anzubieten. Zu diesem Zweck wird für den Agenten eine Repräsentation der Benutzeroberfläche benötigt, so dass dieser jederzeit die entsprechenden Koordinaten ermitteln kann. Um diese Daten zu erhalten, ist eine umfangreiche Kommunikation mit dem Window-System notwendig.

3.2.2.1 Zeigegesten unter Verwendung von WWW-Material

Beim Einsatz des Präsentationsagenten als Präsentator von Informationen, die aus dem WWW gewonnen wurden, d.h. die Domaine des AIA-Systems, Abschnitt 7.2, kommt es oft vor, dass Bilder in bestehenden WWW-Seiten gefunden werden und diese auch in die neu generierte Präsentation integriert werden sollen. Im allgemeinen Fall ist es schwierig, Informationen über das Bild nur aus der Bitmap zu gewinnen. Eine vollständige Bildanalyse erfordert großen Rechenaufwand, das Ergebnis reicht für eine Interpretation zur Generierung von Zeigegesten nicht immer aus. Für eine grobe Einordnung des Bildes in bestimmte Kategorien genügen oft auch einfachere Verfahren [Endres, 1999]. Leider lassen sich aufgrund dieser Daten meist keine echten Zeigegesten generieren, möglicherweise lässt sich aber automatisch ein beschreibender Satz erzeugen. In den meisten Fällen wird man bei Bitmap-Material nicht um eine Annotierung per Hand herumkommen, um Zeigegesten effektiv einsetzen zu können.

Allerdings kann in dem besonderen Fall, wenn das Bild von einer WWW-Seite stammt und mit der Maus anklickbar war, also eine Image-Map hatte, eine einfachere Vorgehensweise gewählt werden. Durch Analyse der Annotation des Images innerhalb des HTML-Codes (Abbildung 3.9) kann man in vielen Fällen auf den Bildinhalt zurückschließen.

```
<IMG ISMAP ALIGN="Top" SRC="nsgcmap3.gif" ALT="" USEMAP="#nsgcmap3" WIDTH="751" HEIGHT="599">
<MAP NAME="nsgcmap3">
  <AREA SHAPE="RECT" HREF="richj.htm" COORDS="580,237,694,265">
  <AREA SHAPE="RECT" HREF="cbrej.htm" COORDS="616,178,752,208">
  <AREA SHAPE="RECT" HREF="victmenu.htm" COORDS="625,83,714,109">
  <AREA SHAPE="RECT" HREF="invmenu.htm" COORDS="486,154,595,182">
  <AREA SHAPE="RECT" HREF="antj.htm" COORDS="452,228,565,251">
  <AREA SHAPE="RECT" HREF="guysmenu.htm" COORDS="448,309,597,337">
  <AREA SHAPE="RECT" HREF="pictj.htm" COORDS="392,252,463,276">
  <AREA SHAPE="RECT" HREF="hfxmenu.htm" COORDS="332,354,413,378">
  <AREA SHAPE="RECT" HREF="lunej.htm" COORDS="189,431,312,464">
  <AREA SHAPE="RECT" HREF="queej.htm" COORDS="156,480,245,511">
  <AREA SHAPE="RECT" HREF="shelj.htm" COORDS="96,534,210,566">
  <AREA SHAPE="RECT" HREF="yarmj.htm" COORDS="2,492,109,519">
  <AREA SHAPE="RECT" HREF="digbj.htm" COORDS="36,436,105,467">
  <AREA SHAPE="RECT" HREF="kngj.htm" COORDS="171,329,234,363">
  <AREA SHAPE="RECT" HREF="annj.htm" COORDS="75,381,187,408">
  <AREA SHAPE="RECT" HREF="kngj.htm" COORDS="169,333,233,361">
  <AREA SHAPE="RECT" HREF="hantj.htm" COORDS="267,322,330,349">
  <AREA SHAPE="RECT" HREF="colcj.htm" COORDS="271,270,391,295">
  <AREA SHAPE="RECT" HREF="cumbj.htm" COORDS="175,229,313,257">
</MAP>
```

Abbildung 3.9: Beispiel für Image-Map: Nova Scotia Counties [Government of Nova Scotia, 1999]

Eine Image-Map liefert eine Abbildung von Bildbereichen auf URL's. Verfolgt man nun die zu jedem Bildbereich abgespeicherte URL und analysiert die entsprechende Zielseite, so ist es oft möglich, eine passende Bildbeschreibung zu diesem Bildbereich zu generieren. Eventuell liefert schon der Titel der Seite, also der Text zwischen <title> und </title> eine Beschreibung des Bildes. Im allgemeinen Fall muss man allerdings die entsprechende Seite genauer untersuchen.

Abbildung 3.10 zeigt eine Präsentation, in der die besonderen Eigenschaften eines Bildes mit einer untergelegten Image-Map ausgenutzt wurden. Das Bild zeigt einen Stadtplan von Portsmouth [University of Portsmouth, 1998]. Auf der Originalseite bewirkt ein Mausklick auf eine Region die Verzweigung auf eine neue HTML-Seite, deren Titel dem Namen des Stadtteils entspricht. Für diese Präsentation wurden die Originalseite mit der anklickbaren Karte und ihre referenzierten Seiten analysiert und die Daten in die Wissensbasis des Präsentationssystems übernommen. Aus dieser Wissensbasis wurde unter Verwendung des Präsentationsplaners und der entsprechenden Generatoren für HTML und Persona eine neue Präsentation generiert. Im Laufe dieser Präsentation annotiert Persona einzelne Stadtteile und nennt ihren Namen mittels Sprachsynthese.

Auch andere Besonderheiten von Bildern lassen sich für die Generierung von Zeigegesten ausnutzen. Der Server Mapquest [Mapquest, 1998] stellt Karten und Stadtpläne zu vielen Regionen der Welt zur Verfügung. Zu einigen deutschen Städten kann er in der aktuellen Version sogar bei Angabe einer Adresse einen Stadtplan berechnen, bei dem die gesuchte Position durch einen roten Punkt markiert ist. Dabei zentriert Mapquest die Karte so, dass der rote Punkt genau in der Mitte der Bitmap der Karte liegt.

Verwendet man diese Bitmaps in einer Präsentation mit Persona, so kann in diesem besonderen Fall die Zeigegeste leicht berechnet werden: Der Zeigestock muss nur genau in die Mitte der Bitmap zeigen. Für den Betrachter sieht es dann so aus, als zeige der Präsentator auf den roten Punkt, der die gesuchte Adresse symbolisiert (Abbildung 3.11).

3.2.2.2 Generierung von Zeigegesten aus 3D-Daten

Werden in einer Präsentation mit Persona visualisierte 3D-Modelle eingesetzt, so hat man die Wahl zwischen 2 Vorgehensweisen: Einmal kann man zunächst eine Bitmap generieren und dann bei der eigentlichen Präsentation nur die erzeugte Bitmap benutzen. Bei WWW-Präsentationen müssen in diesem Fall auf Client-Seite keine aufwendigen Berechnungen mehr durchgeführt werden, so dass die Präsentation auch auf weniger leistungsfähigen Systemen ablaufen kann. Zur Berechnung der Bilder können je nach Anforderungen unterschiedliche Systeme zum Einsatz kommen, wie z.B. GeoDisplay [Müller, 1994], Geomview [Phillips, 1994] oder RayShade [Kolb, 1999]. Erlaubt das Generierungssystem, wie GeoDisplay, den direkten Zugriff auf die internen Datenstrukturen, so kann man die für die Generierung der Zeigegesten benötigten Koordinaten unmittelbar ermitteln (Abbildung 3.12). Bei anderen Systemen müssen die Koordinaten aus den Kameraparametern zunächst berechnet werden.

Die zweite Möglichkeit besteht darin, auf Client-Seite eine 3D-Visualisierung

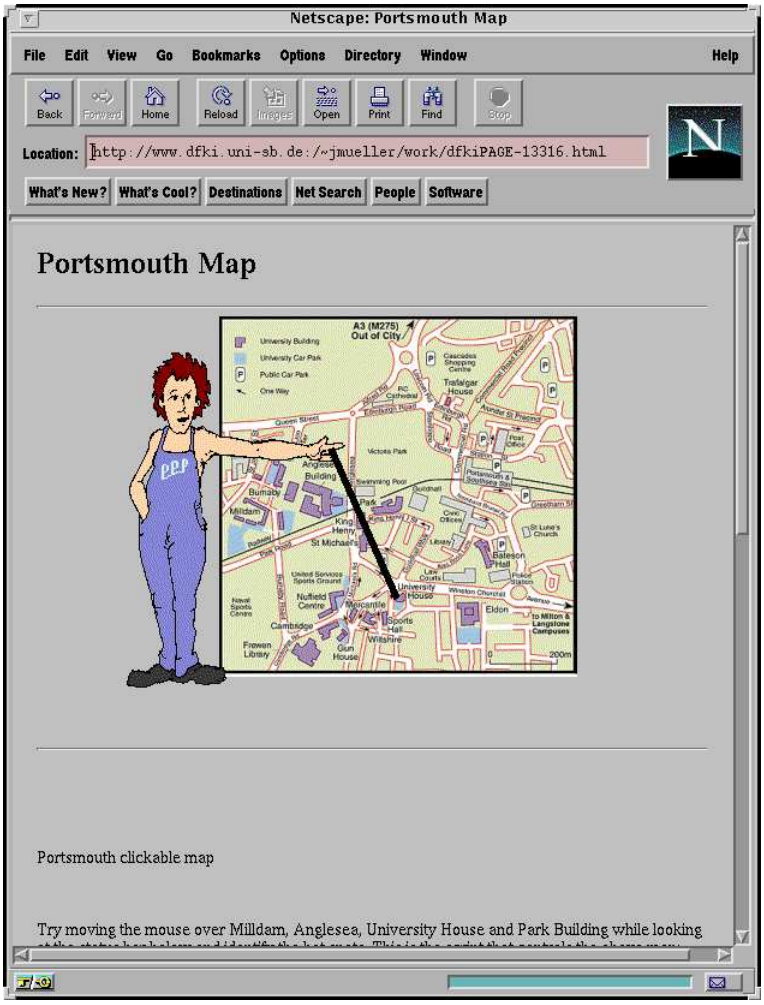


Abbildung 3.10: Nutzung einer Image-Map zur Generierung einer Beschreibung des Stadtplanes von Portsmouth

durchzuführen, was eine leistungsfähige Hardware voraussetzt. Wird zur Darstellung der Bilder auf Client-Seite das 3D-Gadget (Abschnitt 5.2) eingesetzt, so können die benötigten Daten von dem 3D-Gadget zur Verfügung gestellt werden.

Bei der Konfiguration des Präsentationsagenten als Desktop-Assistenten muss neben den Koordinaten relativ zu dem Window auch die Position des Windows auf dem Bildschirm berücksichtigt werden. Dabei muss gewährleistet sein, dass der Benutzer während des Präsentationsablaufs ein Fenster mit der Maus verschieben kann (Abbildung 3.13).

3.2.2.3 Zeigegesten auf Textelemente

Texte sind in Präsentationen wichtige Elemente. Um besonders wichtige Stellen innerhalb eines Textes für den Betrachter schnell und dynamisch hervorzuheben, können Zeigegesten des Präsentationsagenten eingesetzt werden. Beispielswei-

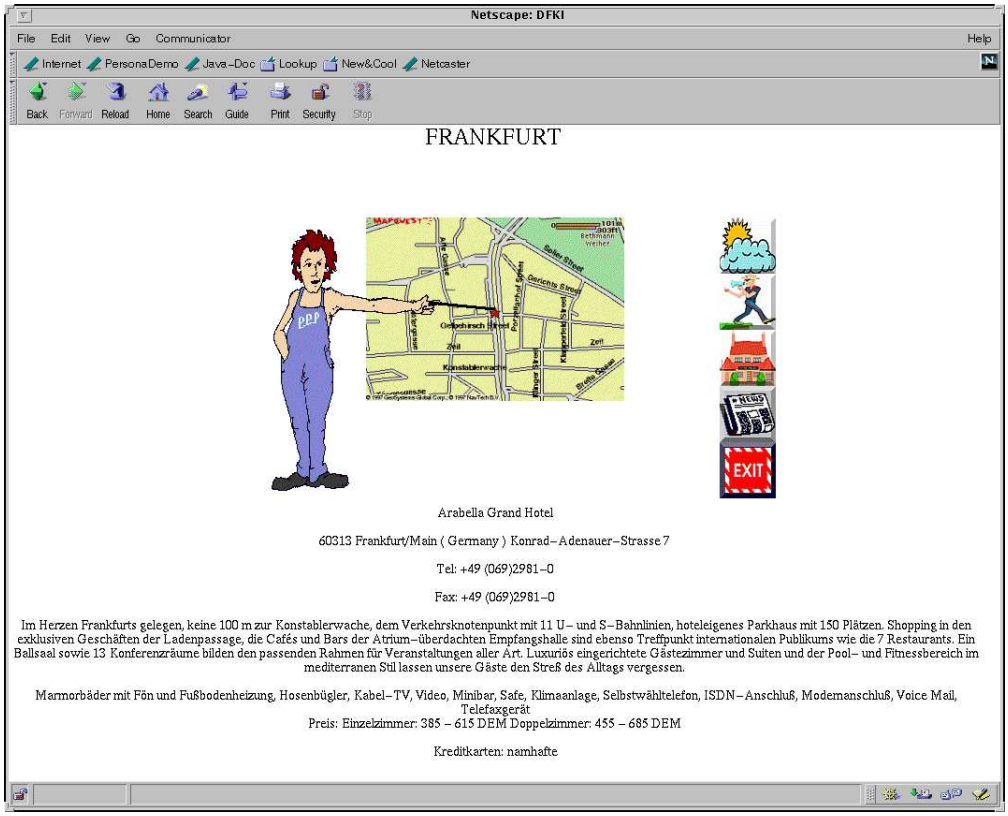


Abbildung 3.11: Präsentation einer von Mapquest generierten Karte

se kann der Präsentationsagent einen kurzen Überblick über einen bestimmten Sachverhalt geben und für genauere Informationen auf bestimmte Textstellen hinweisen. Diese Vorgehensweise ist sinnvoller, als den ganzen Text durch den Präsentationsagenten vorlesen zu lassen, da es so dem Betrachter ermöglicht wird, den gezeigten Text schnell zu finden, zu überfliegen und nur bei besonderem Interesse genauer nachzulesen. Würde man den Text vorlesen lassen, so müsste der Betrachter auf das Ende dieser Aktion warten, obwohl er schon nach den ersten Textzeilen festgestellt hat, dass ihn der Text nicht weiter interessiert.

Wie bei Bildern ist auch bei Zeigegesten zu unterscheiden, ob Positionsinformationen über entsprechende Textstelle zur Verfügung stehen oder nicht. Liegt der Text als Bitmap (z.B. eingescannte Zeitungsartikel) vor, so ist man (wenn man von der Verwendung von OCR- und Textanalyseverfahren absieht) darauf angewiesen, per Hand die entsprechenden Koordinaten zu ermitteln. Generiert man die Textdarstellung allerdings mit dem in Abschnitt 5.2.3 vorgestellten Modul, so können die benötigten Daten aus den von dem Modul bereitgestellten Datenstrukturen berechnet werden.

3.2.3 Komplexe Zeigeoperationen

In realen Vorträgen, die mit Bildmaterial (z.B. dargestellt mittels Overhead-Projektoren, Diaprojektoren oder auch auf Tafeln) untermalt werden, verwendet

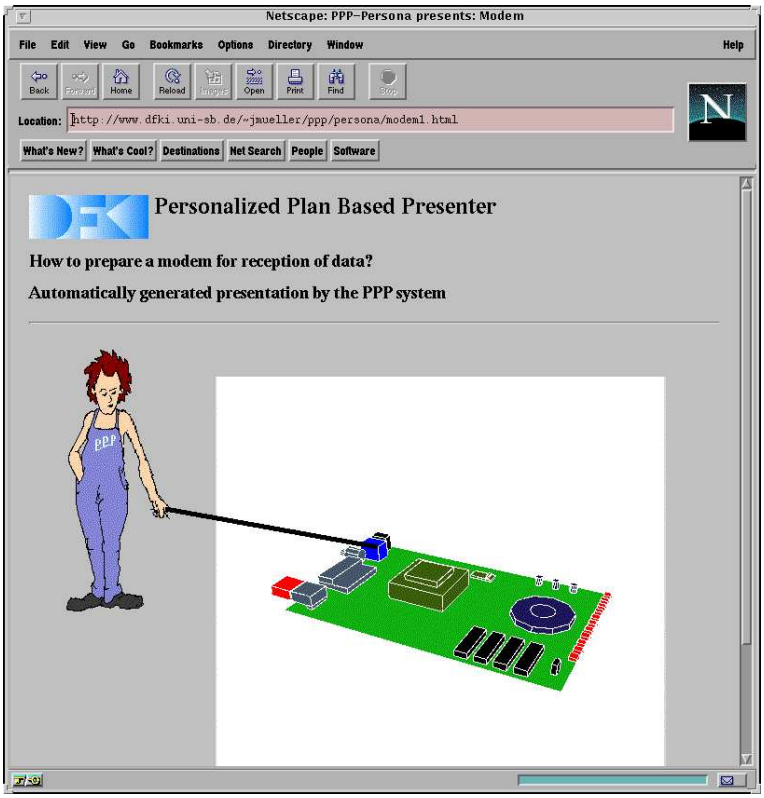


Abbildung 3.12: Präsentation der Funktionsweise eines Modems auf Basis eines 3D-Modells

der Vortragende nicht nur einfache punktuelle Zeigeoperationen, sondern auch komplexere Zeigegesten. Einige Beispiele sind:

Unterstreichungsgesten: Bei diesen Gesten fährt der Präsentator entlang einer imaginären Linie. Diese Geste wird oft eingesetzt, wenn Texte oder Formeln erklärt werden sollen. Der reale Präsentator liefert dann meistens weitere Erklärungen zu dem bezeichneten Satz.

Nachfahren von Linien: Bei der Beschreibung von Diagrammen verfolgt oftmals der Vortragende die logisch aufeinanderfolgenden Elemente:

- Darstellung eines Algorithmus als Flussdiagramm. Der Vortragende verdeutlicht den Programmfluss, indem er bei seiner Beschreibung die einzelnen Wege nachfährt.
- Darstellung eines Rohrleitungssystems. Der technische Vorgang wird dadurch verdeutlicht, dass die Flussrichtung der einzelnen Flüssigkeiten durch Zeigegesten angedeutet werden (analog bei Stromkreis).
- Bei der Beschreibung von Wegen mit Hilfe einer Karte werden die passenden Straßen mit dem Zeigewerkzeug entlanggefahren, so dass der spätere Weg im Voraus symbolisch „abgefahren“ wird.

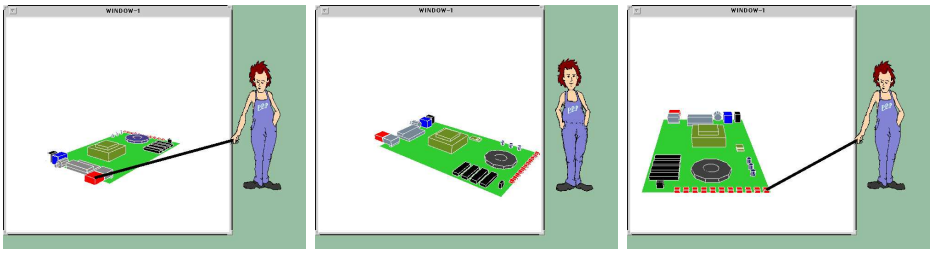


Abbildung 3.13: Der Desktop-Präsentationagent zeigt das Modem in einem Fenster

Umfahrungsgesten: Sollen größere Elemente während einer Präsentation hervorgehoben werden, so bieten sich Umfahrungsgesten an. Zur Realisierung einer solchen Geste müssen die entsprechenden Stützstellen für die Umfahrungskurve bzw. das -polygon ermittelt werden. Diese Daten können beispielsweise den Imagemap-Daten, den Instanzen der VisualizationGadgets oder entsprechenden Generierungssystemen (z.B. für Grafik) entnommen werden.

Diese Gesten lassen sich mit dem Präsentationsagenten Persona realisieren, indem die primitiven Zeigeoperationen so angesteuert werden, dass der Eindruck entsteht, als würde das Zeigewerkzeug das berechnete Polygon nachfahren und so die entsprechende Geste ausführen.

In dem System ZORA [Jung et al., 1989] werden Zeigegesten passend zu dem zu zeigenden Objekt ausgewählt (z.B. entsprechend der geometrischen Ausdehnung) und visualisiert. Gemäß der Klassifikation des zu zeigenden Objekts werden punktuelle Zeigegesten, umfahrende Gesten oder unterstreichende Gesten ausgewählt.

Der Persona-Generator berechnet zu einen Objekt passende Zeigepunkte, indem er die Ausmaße des Objekts berücksichtigt.

3.2.4 Der Lupeneffekt

Im Laufe einer Präsentation kommt es oft vor, dass man in ein bestehendes Bild ein anderes Bild einblenden will. In herkömmlichen (d.h. ohne Präsentationsagenten) Präsentationen verwendet man Insets, Abbildung 3.14.

In Präsentationen mit Persona bietet es sich an, für diesen Zweck einen speziellen Zeigestock, nämlich die Lupe, einzusetzen. Da der Betrachter den Bewegungen des Präsentationsagenten folgt, wird er auch zum Betrachten der Lupe bzw. des Lupeninhaltes animiert.

Einerseits kann man die Lupe dazu einsetzen, einen bestimmten Bildausschnitt unter der Lupe vergrößert darzustellen und ihn so dem Betrachter genauer zu präsentieren. Zu diesem Zweck kann man im einfachsten Fall die darunterliegende Bitmap vergrößern und abbilden, was über die VisualizationGadgets dem Client unmittelbar möglich ist. In manchen Fällen, wenn nur einfache und nicht zu starke Vergrößerungen gefordert sind, kann das ausreichen. Einen besseren Effekt erhält man aber allerdings, wenn man spezielle Bitmaps, die das dargestellte Objekt wirklich in einem anderen Maßstab zeigen, einsetzt.

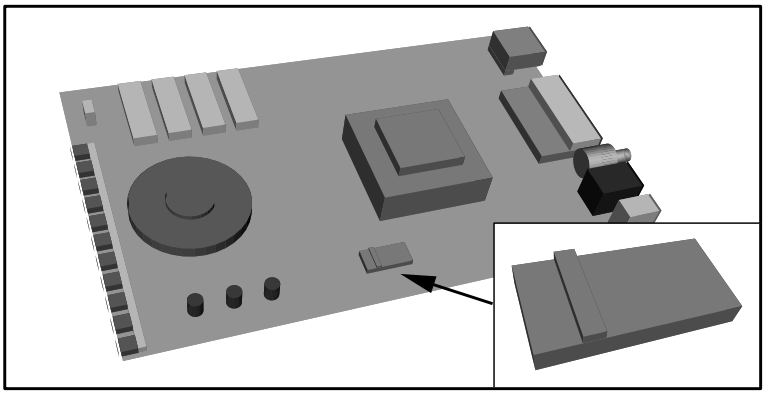


Abbildung 3.14: Einblendung von Vergrößerungsausschnitten (aus [Rist, 1995])

Man kann diese beiden Techniken auch mischen, wenn man bei dem Umschalten zwischen den einzelnen Bildern versucht, durch pixelweise Vergrößerung einen fließenenden Übergang zwischen den Bildern zu erhalten.

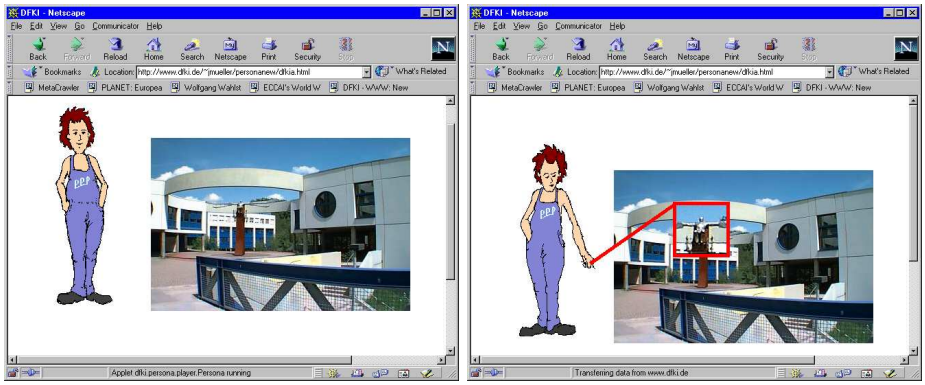


Abbildung 3.15: Lupe vergrößert Objekte

Eine Erweiterung dieser Technik besteht darin, dass man mit der Lupe nicht nur Vergrößerungen eines Objekts sichtbar macht, sondern Objekte darstellt, die mit dem Objekt unter der Lupe in direkter Verbindung stehen. In Abbildung 3.16 wurden die realen Objekte (nämlich die Kirche bzw. das DFKI-Gebäude) in die Karte eingezeichnet. Durch die Lupe konnten so zwei Medien miteinander kombiniert werden.

Noch einen Schritt weiter geht die Präsentation von Abbildung 3.17. Hier werden nicht nur beliebige Vergrößerungen bzw. andere Sichtweisen eines Objekts in eine Abbildung eingefügt, sondern es werden andere Sachverhalte, die nur indirekt mit dem Objekt in Verbindung stehen, visualisiert.

Durch diese Geste wird das Sprichwort „etwas unter die Lupe nehmen“ unmittelbar in eine Präsentation umgesetzt. Bei solchen Präsentationen werden somit sogar unterschiedliche Datentypen miteinander verknüpft (hier: Kartenmaterial, Fotos und statistische Informationen).

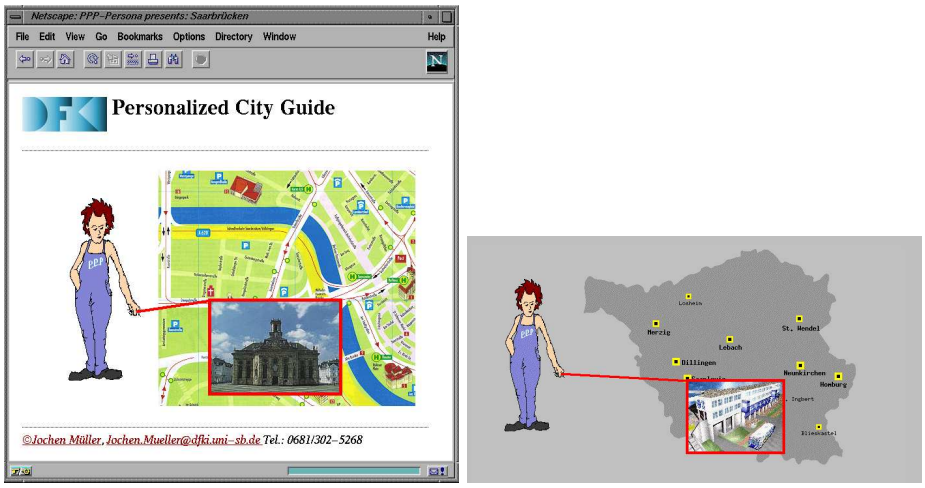


Abbildung 3.16: Lupe visualisiert zusätzliche Objekte in Karten

3.2.5 Humor

Bei so starker Personifizierung eines Präsentationssystems wie durch animierte Präsentationsagenten bietet sich ein weiteres Kommunikationsmittel an: Humor. Humor wird oft in realen Präsentationen eingesetzt, um den Vortrag aufzulockern und eine entspanntere Atmosphäre zu schaffen. Auch in einer generierten Präsentation kann Humor nützlich sein. Durch den Einsatz des Kommunikationsmittels Humor können viele Benutzerprobleme abgeschwächt werden [Nack, 1997]. Auf diese Weise kann z.B. Langeweile überbrückt werden (z.B. beim Warten auf ein Ergebnis, wobei die eigentliche Berechnung längere Zeit in Anspruch nimmt) oder eine humorvolle Entschuldigung kann dazu verwendet werden, die Enttäuschung des Benutzers zu mildern, falls seine gestellte Aufgabe nicht vollständig gelöst werden konnte (Selbstkritik). Ein humorvoller Kommentar kann ebenfalls zur Auflockerung einer Präsentation eingesetzt werden. Da es schwierig ist, einen vollständigen Wissensbereich mit einem Computersystem abzudecken, können humorvolle Einlagen helfen, Wissenslücken zu überdecken. Eine humoristische Einlage beinhaltet einerseits eine sprachliche Äußerung und zum anderen eine visuelle Geste, die die Sprachausgabe begleitet. Nicht in jeder Situation ist es passend, den Benutzer mit Witzen zu konfrontieren. Beispielsweise ist es für ein Präsentationssystem unpassend, das Stilmittel Humor einzusetzen, falls der Benutzer in Eile ist und er deshalb kein Interesse an irgendwelchen Verzögerungen hat. Daher ist es notwendig, dass die Generierungsmöglichkeit von Humor vom Benutzer abgeschaltet oder modifiziert werden kann.

In dem System PPP-Persona wird die Integration von Humor in eine Präsentation durch Verwendung spezieller Planungsstrategien gewährleistet, die entsprechend der aktuellen Sprache, der Situation und den Benutzerpräferenzen humoristische Einlagen für die Präsentation generieren. Dazu greift das System auf eine Datenbank mit vorbereiteten Witzen und Slapstickeinlagen zurück. Aus einer geeigneten Menge von möglichen Ausgaben wird zufallsgesteuert eine pas-

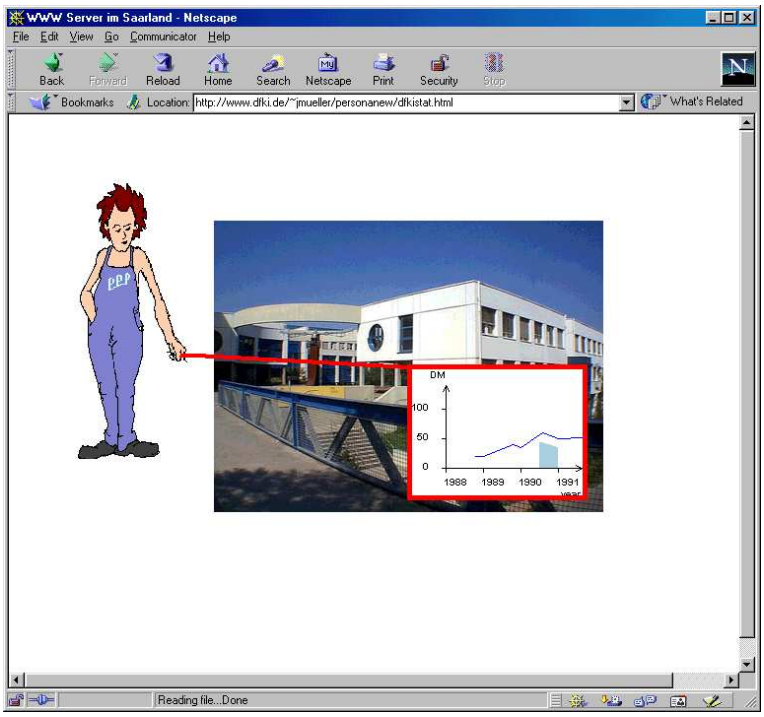


Abbildung 3.17: Visualisierung der Geschäftsentwicklung, wobei die Lupe über dem Firmengebäude der betrachteten Firma steht.

sende ausgewählt und in die Präsentation integriert.

Ein besonderes Anwendungsfeld für einen Präsentationsagenten in diesem Zusammenhang ist der Einsatz innerhalb von WWW-basierten Marketingaktionen. Beispielsweise könnte der Präsentationsagent ein Blickfang auf der Startseite des werbenden Unternehmes sein und in regelmäßigen Abständen Witze erzählen und so den Betrachter dazu animieren, die WWW-Seite immer wieder zu besuchen.

3.3 Einsatz mehrerer Präsentationsagenten

Während in den meisten Präsentationen ein Präsentator ausreicht, können in manchen Sonderfällen mehrere Präsentatoren von Vorteil sein. Der Einsatz mehrerer Präsentatoren bewirkt eine weitere Auflockerung der Präsentation und erhöht ihre Lebendigkeit. Über die reine Unterhaltungswirkung hinaus steigern mehrere Präsentatoren den Lerneffekt und verbessern den Gesamteindruck der Präsentation.

3.3.1 Anwendungsszenarios

Die Verwendung von mehreren Präsentationsagenten in einer Präsentation erlaubt den Aufbau von Dialogen, an denen sowohl virtuelle als auch reale Personen teilnehmen.

Neben den im Folgenden beschriebenen Anwendungsmöglichkeiten für den Einsatz mehrerer Agenten gibt es noch die Einsatzmöglichkeit für Persona innerhalb eines Chat-Systems, bei dem die einzelnen Benutzer durch einen Avatar repräsentiert werden. Ein solches System wird in Abschnitt 7.3 beschrieben.

3.3.1.1 Lernumgebungen

Animierte Präsentationsagenten können viele Funktionen in interaktiven Lernumgebungen übernehmen. In einer Anwendung ist der Einsatz mehrerer virtueller Lehrer, die durch die Präsentationsagenten visualisiert werden, denkbar. Dabei können die einzelnen Lehrer verschiedene Spezialgebiete übernehmen, um dem Schüler, also dem Betrachter, eine komplexe Situation zu erläutern. Der Gebrauch von Präsentationsagenten hilft dem Schüler, eine eventuell vorhandene Hemmschwelle dem Lernsystem gegenüber zu überwinden. Eine andere Möglichkeit ist, neben einem virtuellen Lehrer auch einen virtuellen Schüler in einer Präsentation zu verwenden. In einem solchen Szenario bearbeiten Lehrer und Schüler gemeinsam eine Aufgabenstellung. Der virtuelle Schüler richtet an den virtuellen Lehrer die Fragen, die dieser zu beantworten hat. Der reale Schüler, auf den die Präsentation abzielt, soll so ermuntert werden, über das Problem genauer nachzudenken und eigene Fragen an den virtuellen Lehrer zu stellen (Abbildung 3.18).

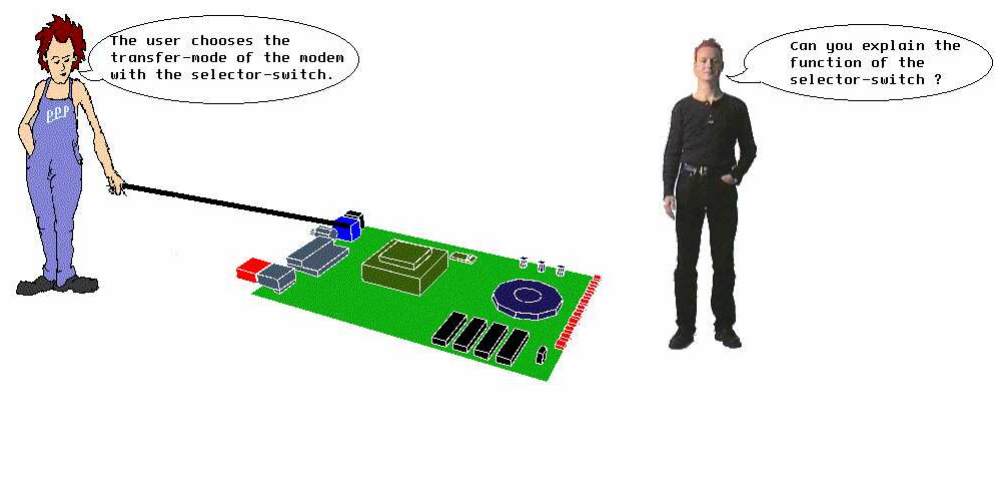


Abbildung 3.18: Virtueller Lehrer und virtueller Schüler beim gemeinsamen Bearbeiten eines Problems

Dadurch, dass während einer solchen Präsentation kein reiner Monolog entsteht, ist die gesamte Präsentation lebendiger und unterhaltsamer. Der reale Schüler fühlt sich in die Präsentation eingebunden und wird ihr leichter folgen.

Durch die Fragen des virtuellen Schülers können schon viele Fragen des realen Schülers vorweggenommen und vom System beantwortet werden. Auch können die Fragen des virtuellen Schülers eine psychologische Unterstützung für den realen Schüler bedeuten. Er sieht, dass auch andere Leute Probleme mit dem behandelten Stoff haben. Eine weitere mögliche Anwendung ist die Simulation

von Prüfungssituationen, die einem Schüler helfen, sich besser auf eine Prüfung vorzubereiten.

In manchen Anwendungsgebieten (hauptsächlich bei der Behandlung von praktischen Ausbildungsthemen) kann es auch angebracht sein, dass der Lehrer bestimmte Tätigkeiten demonstriert und der Auszubildende sie wiederholt. Bei der Simulation kann der Schüler eventuell zunächst die typischen Fehler durchspielen, die vom Lehrer korrigiert werden. Auf diese Weise wird der reale Schüler frühzeitig auf mögliche Fehler hingewiesen.

3.3.1.2 Einsatz von Haupt- und Nebenpräsentatoren

In Nachrichtensendungen oder in Fernsehmagazinen werden häufig Moderatoren eingesetzt, die durch die gesamte Sendung führen. Für spezielle Themengebiete oder für aktuelle Vor-Ort-Berichterstattungen werden andere Reporter eingesetzt (z.B. Wettervorhersage, Börsenentwicklungen, Live-Schaltungen an andere Orte). Dem Fernsehzuschauer soll der Eindruck vermittelt werden, die von diesen spezialisierten Reportern vorgetragenen Nachrichten seien aktueller und fundierter recherchiert.

Präsentationssysteme können sich ebenfalls dieses Effektes bedienen und so ihre Glaubwürdigkeit und den Eindruck größerer Aktualität steigern. Der Eindruck der Authentizität und Kompetenz kann auch durch das visuelle Erscheinungsbild des Präsentators unterstrichen werden. Ein Reiseagent, der z.B. Reisetipps zu Schottland gibt, könnte einen Kilt tragen, während ein Präsentator des aktuellen Wetters auch durch einen animierten Frosch (Wetterfrosch) visualisiert werden könnte. Der Agent, der als Repräsentant des Reisebüros auftritt, kann, wenn er einen Reisevorschlag vorstellt, die spezialisierten Agenten befragen. Ein anderes Beispiel ist die Präsentation des aktuellen Wetterberichts (Abbildung 3.19). Ein Hauptpräsentator befragt den spezialisierten Präsentator nach der aktuellen Wetterlage. Beim Betrachter ergibt sich der Eindruck von höherer Authentizität und Aktualität. Verschiedene Agenten können in dieser Anwendung die unterschiedlichen meteorologischen Beobachtungsstationen repräsentieren und das jeweilige aktuelle Wetter übermitteln.

3.3.1.3 Ein Verkaufssystem

Ein System, das zum Verkauf von Waren über das Internet eingesetzt wird, soll möglichst einen Kunden zum Kauf der angebotenen Waren veranlassen. Auch in diesem Szenario können mehrere Agenten sinnvoll sein. Zur Unterstützung des Verkaufssystems kann ein freundlich aussehender virtueller Verkäufer eingesetzt werden, der dem Kunden die Vorzüge der Waren darstellt. Um die Verkaufspräsentation glaubwürdiger zu gestalten, kann dem Verkäufer eine weitere Person gegenübergestellt werden, der ebenso wie der Verkäufer vom Verkaufssystem gesteuert wird. Die Aufgabe der zweiten Persona ist, mit schwachen Argumenten gegen das zu verkaufende Produkt zu argumentieren. Da die Argumente vom Verkaufssystem ausgewählt werden, ist es für den Verkäufer einfach, diese Argumente zu entkräften. Auf diese Weise können einige der möglichen Gegenargumente des Käufers vorweggenommen werden und in der Verkaufsstrategie

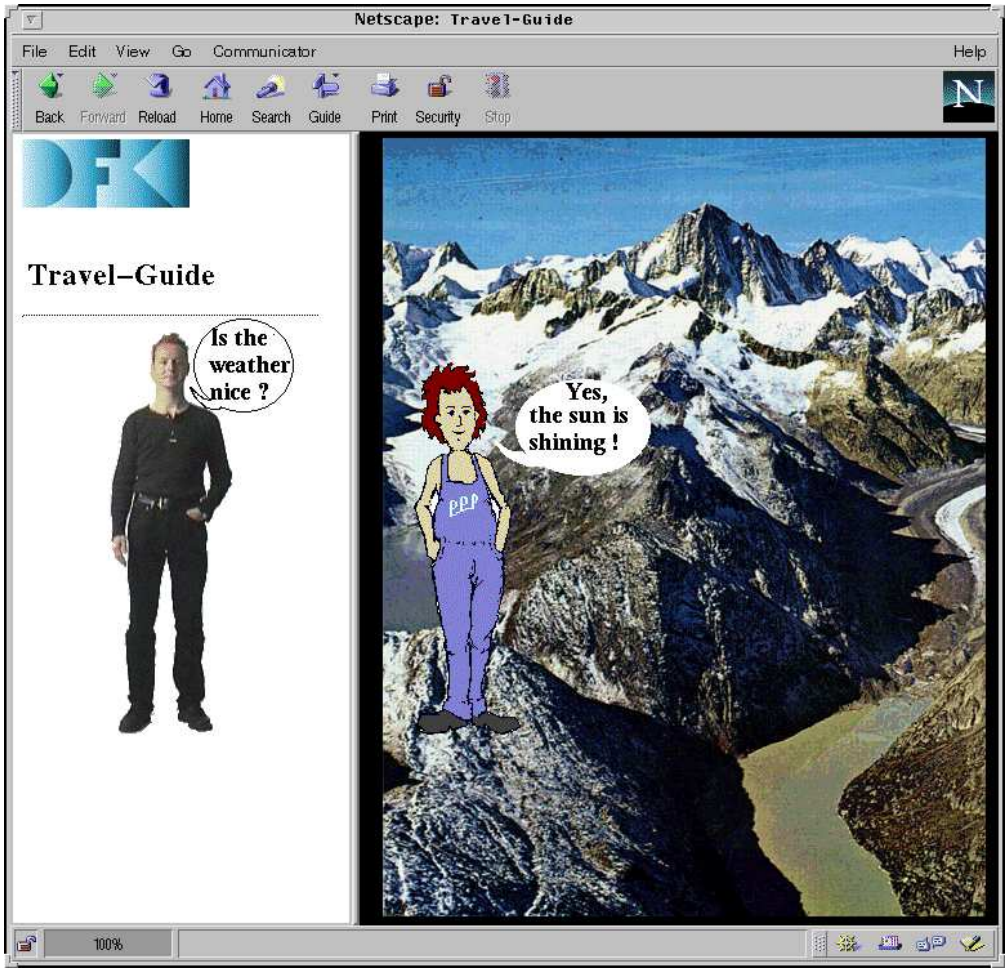


Abbildung 3.19: Einsatz von spezialisierten Moderatoren und allgemeinen Moderatoren in einer Präsentation

berücksichtigt werden. Dennoch könnte sich bei einer solchen Verkaufstrategie der Kunde unter Umständen objektiver beraten fühlen.

Auch in einem Verkaufssystem kann es sinnvoll sein, unterschiedliche, auf bestimmte Themengebiete spezialisierte Verkaufsagenten einzusetzen. Beispielsweise kann bei einem Auto-Shop ein normaler Autoverkäufer die wirtschaftlichen Vorzüge des Autos beschreiben, während ein (auch äußerlich als solcher erkennbarer) virtueller Automechaniker technische Besonderheiten des Autos herausstellen kann [André und Rist, 2000].

Der Dialog kann noch interessanter gestaltet werden, wenn der Dialog nicht nur einen Käufer und den Verkäufer umfasst, sondern wenn gleichzeitig mehrere Käufer und der Verkäufer auftreten. Beispielsweise können sich die Käufer für unterschiedliche Aspekte interessieren und dem Verkäufer die entsprechenden Fragen stellen. Abbildung 3.20 veranschaulicht einen solchen Dialog, der mit dem Präsentationsplaner PrePlan [André, 1995] generiert wurde. Als Eingabe erhält PrePlan die Rollen (Käufer, Verkäufer), die Interessensgebiete (z.B. Preis, Sportlichkeit, Komfort) und die Verhaltensmerkmale (z.B. schüchtern,

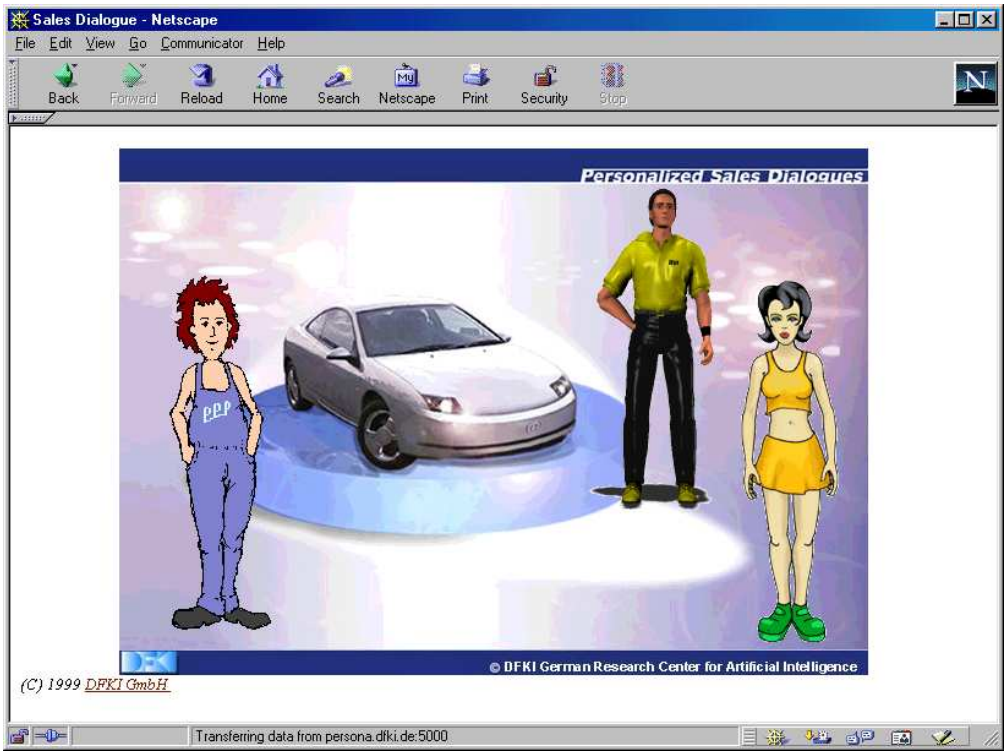


Abbildung 3.20: Animierte Charaktere in einem Verkaufssystem

dominant). PrePlan erzeugt aus diesen Eingaben einen Dialog zwischen den Käufern und dem Verkäufer, der über den Persona-Generator in ein für den Persona-Player verständliches Format überführt wird.

Durch derart aufgebaute Dialoge kann das Verkäufsgespräch unterhaltsamer und lebendiger gestaltet werden, was bei einem Verkaufssystem essentiell ist, da die Benutzer sich möglichst oft und lange mit dem System beschäftigen sollen.

3.4 Definition eines Präsentationsagenten

Ein Präsentationsagent P lässt sich als Tupel $P = (S, B, N, U, R, V, K, I, O, C, \delta)$ seiner Eigenschaften definieren. Dabei ist

S: Menge der zur Verfügung stehenden Sprachsynthesizer (z.B. für Deutsch oder Englisch). Falls $S = \emptyset$, dann kann der Agent keine Sprachausgabe synthetisieren. Er ist in diesem Fall auf andere Methoden angewiesen (z.B. Sprechblasen, vorgenerierte Audio-Dateien).

B: Menge der möglichen Visualisierungen des Agenten (z.B. Bitmaps, 3D-Modelle)

N: eindeutiger Name des Agenten (Kommunikation zwischen einzelnen Agenten)

U: Umgebung, in der sich der Agent befindet (z.B. Applet, Desktop)

$R = G \cup A$: Regelmenge, die zur Ansteuerung des Agenten dient. Sie besteht aus Regeln zur Ansteuerung der Gesten G und Steuerungsregeln A.

$V \subset G$: Regelmenge zur Ansteuerung der Vitalgesten, die dafür sorgen, dass der Agent lebendig wirkt (z.B. Atmen)

K: interner Zustand des Agenten; Elemente aus K repräsentieren die aktuelle Geste/Aktion, die Position des Agenten sowie die Parameter für die aktuelle Aktion (z.B. Texte, Punktangaben, Grafiken)

I: Eingabekanäle (z.B. Schnittstelle zum Präsentationssystem, Benutzerinteraktion)

O: Ausgabekanäle (z.B. Bildschirmausgabe)

C: geordnete Liste der auszuführenden Präsentationsaktionen

δ: Evaluierungsrelation $\delta : I \times C \times K \times R \times O$

Der Präsentationsagent mit dem eindeutigen Namen N zeigt eine Präsentation gemäß der vom Präsentationssystem vorgegebenen Präsentationsaufgaben C. Er kommuniziert mit seiner Umgebung über mehrere Ein-/Ausgabekanäle I und O. Das audiovisuelle Erscheinungsbild wird durch die Sprachsynthesizer S und seine grafische Gestaltung B bestimmt. Je nachdem in welcher Umgebung U er eingesetzt wird, können sich seine Eigenschaften ändern. Die aktuelle Handlung des Agenten wird neben dem aktuellen Auftrag durch den aktuellen Zustand K bestimmt. Entsprechend den definierten Präsentationsregeln R evaluiert der Agent mittels der Relation δ die aktuellen Eingaben, Aktionen und den Zustand und generiert daraus die entsprechende Reaktion. Um den Agenten lebendig wirken zu lassen, werden einzelne Vitalgesten in die Präsentation eingestreut.

3.5 Systemarchitektur

Der Präsentationsagent besteht aus zwei Hauptkomponenten: der Generierungskomponente und der Abspielkomponente. Die Generierungskomponente erzeugt aus einer abstrakten Beschreibung der Präsentation und den Präsentationsdaten ein Skript, das von der Abspielkomponente unmittelbar ausgeführt werden kann. Gleichzeitig generiert sie alle mit der Präsentation verbundenen Daten und stellt sie der Abspielkomponente zur Verfügung. Die Eingabe für die Generierungskomponente kann entweder von Hand erstellt oder durch ein anderes Präsentationssystem erzeugt werden.

Durch die Teilung in Generierungskomponente und Abspielkomponente ist es möglich, die Komponenten auf unterschiedlichen Rechnern ausführen zu lassen, so dass WWW-Applikationen ermöglicht werden.

Die Generierungskomponente ist als Schnittstelle zwischen einem Präsentationssystem und dem Präsentationsagenten anzusehen. Einerseits dient sie zur Ansteuerung des Persona-Players, andererseits verarbeitet sie auch Benutzereingaben und fordert in diesem Fall entsprechende Aktionen des Präsentationssys-

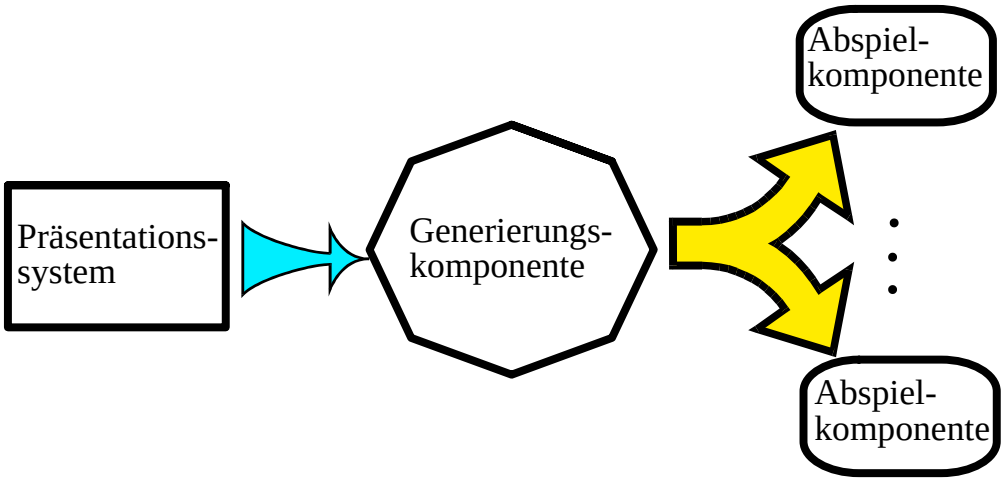


Abbildung 3.21: Überblick über die einzelnen Module des Persona-Systems

tems an. Die Generierungskomponente erzeugt die für eine Präsentation von dem Präsentationsagenten benötigten Daten.

Obwohl die Skripte für den Persona-Player auch von Hand geschrieben werden können, ist es aus Gründen der Bequemlichkeit und der Datenkonsistenz sinnvoll, das Skript und die damit verbundenen Daten mittels eines Generierungsverfahrens erzeugen zu lassen. Damit ist es möglich, die Aktionen auf hohem abstrakten Niveau zu spezifizieren. Um diese Skripte von dem Persona-Player ausführen zu lassen, müssen zunächst die komplexen Befehle in primitivere Konstrukte umgeformt werden.

Das im Folgenden vorgestellte Regelsystem erlaubt den Aufbau komplexer Handlungsschritte aus einfacheren Persona-Aktionen. Bei der Konstruktion fließen neben zeitlichen Randbedingungen auch Informationen über die aktuelle Persona-Konfiguration (z.B. Gemütszustand, Temperament usw.) ein.

Eine Expansion durch den Persona-Generator eines Persona-Skripts, das auf hohem Abstraktionsniveau erstellt wurde, führt zu einem Skript aus primitiven Aktionen, die von dem Persona-Player unmittelbar ausgeführt werden können. Gleichzeitig erzeugt er die für eine Präsentation benötigten Daten wie Audiodateien für die Sprachausgabe oder Bildmaterial in der für die Präsentation passenden Skalierung. Der Generator erhält seine Eingabe entweder unmittelbar vom Anwender, indem dieser das Generator-Skript per Hand erstellt oder wie im AIA bzw. PPP-System von einem Präsentationsplaner oder einem anderen Präsentationssystem. Eine weitere Aufgabe des Generators ist die korrekte Ansteuerung der einzelnen primitven Gesten, d.h. der einzelnen Bitmapsequenzen. Er muss die kontextabhängige Zerlegung komplexer Gesten durchführen und dafür sorgen, dass die einzelnen Gesten ohne Sprünge logisch aufeinander folgen. Um dies zu erreichen, muss er unter Umständen die für den Übergang notwendigen Zwischengesten bestimmen und in die Präsentation einfügen. Bei der Zerlegung müssen der aktuelle Zustand des Präsentationsagenten und die Benutzerpräferenzen berücksichtigt werden. Die folgende auszuführende primitive Geste ist in den meisten Fällen von der Vorgängergeste abhängig. In der

Animation sollen keine Sprünge oder unzusammenhängende Bildfolgen entstehen, deshalb sind eventuell verbindende Gestenelemente einzufügen. Gleichzeitig müssen die vom Präsentationsplaner vorgeschriebenen zeitlichen Vorgaben eingehalten werden. Je nach Benutzerpräferenzen werden lebhaftere oder eher ruhige Präsentationen erzeugt. Eine weitere Aufgabe des Persona-Generators ist die Verteilung der Aufgaben an die einzelnen Präsentationsagenten, falls in einer Präsentation mehrere Instanzen von Persona eingesetzt werden sollen. Zusätzlich löst der Persona-Generator symbolische Sprungadressen innerhalb der Persona-Skripts auf und ermöglicht so die Definition von Prozeduren innerhalb des Skriptes.

Der Persona-Generator kommuniziert über die Planerschnittstelle mit dem Präsentationsplaner. Einerseits schreibt der Präsentationsplaner die Präsentationsaufgaben für den Präsentationsagenten in den Eingabepuffer des Generators, andererseits kann der Generator bei dem Präsentationsplaner ein neues Präsentationsziel in Auftrag geben (Abbildung 3.22).

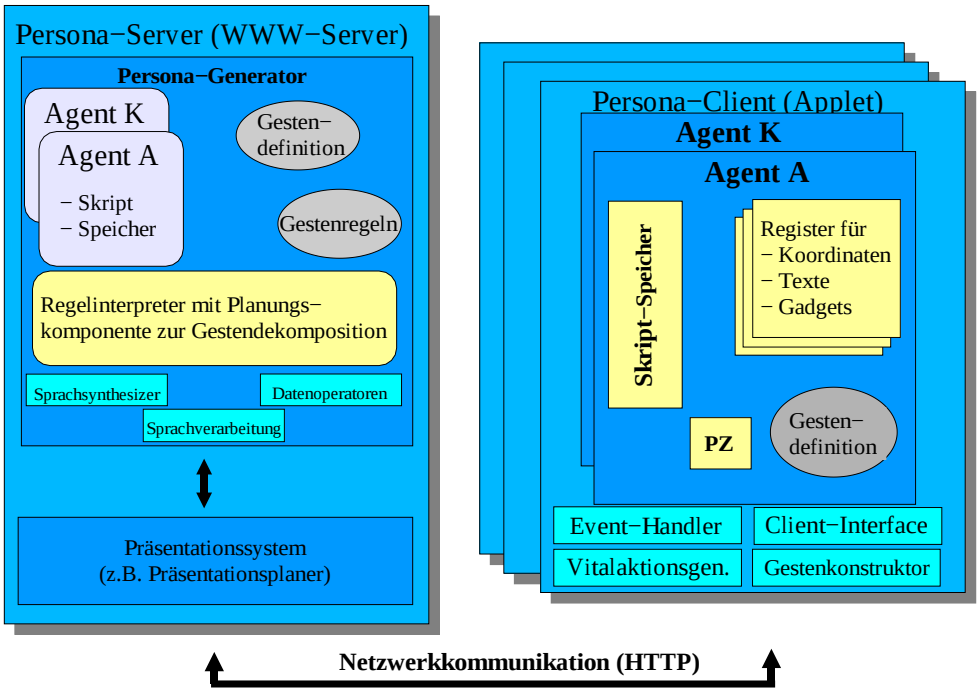


Abbildung 3.22: Architektur des Persona-Systems

Außerdem kann die Eingabeschnittstelle des Persona-Generators neben der direkten Kommunikation mit Programmen ihre Eingabe auch über das Dateisystem beziehen, so dass ein Benutzer auch von Hand entwickelte Persona-Skripte mit dem Generator bearbeiten kann.

Der Persona-Generator verwaltet für jeden Präsentationsagenten eine eigene Datenstruktur, die den aktuellen Zustand des Agenten repräsentiert. Entsprechend den Anweisungen des Eingabeskripts werden die Befehle an verschiedene Agenten verteilt, die die Befehle gemäß ihres jeweiligen internen Zustandes weiter expandieren.

Jede Repräsentation eines Präsentationsagenten innerhalb des Persona-Generators verwaltet eine dynamische Menge von System- und Benutzerregistern, die zur Steuerung des Agenten eingesetzt werden (Gedächtnis). Dabei sind die Register durch den Benutzer bzw. durch Aktionen jederzeit veränderbar oder es können neue Register hinzugefügt werden.

Jeder Befehl innerhalb des Persona-Skriptes wird gemäß seiner Definition in primitive Befehle zerlegt, die unmittelbar von der Abspielkomponente verarbeitet werden können.

In einem letzten Schritt werden alle symbolischen Sprungadressen (z.B. in unbedingten Sprunganweisungen oder in Menübefehlen) aufgelöst und durch ihre numerischen Äquivalente ersetzt. Die generierten Skripte und die dazugehörigen Daten (z.B. Audiodateien für die Sprachausgabe) werden schließlich an die entsprechenden Instanzen der Abspielkomponente weitergeleitet.

Nach der Zerlegung wird das Skript in den Skriptspeicher des Zielagenten kopiert. Die virtuelle Maschine des Agenten evaluiert das Skript an der durch den Programmzähler (PZ) definierten Stelle und führt die einzelnen Aktionen aus, wobei neben den audiovisuellen Ausgaben auch Daten für bestimmte Aufgaben in Registern manipuliert werden können. Neben der Eingabe vom Persona-Generator bearbeitet die Maschine auch Benutzereingaben und kann mit externen Programmen wie auch anderen Agenten kommunizieren. Der Gestenkonstruktor führt entsprechend einer Gestendefinition und dem aktuellen Zustand eine Geste aus. Falls keine Aktionen durch die Maschine auszuführen sind, fügt der Vitalaktionsgenerator Gesten ein, um den Agenten lebendiger erscheinen zu lassen.

In Abbildung 3.22 ist die Gesamtarchitektur des Persona-Systems in der WWW-Konfiguration dargestellt, für die Desktop-Konfiguration werden geringfügige Änderungen notwendig (Abschnitt 6.1). Das System ist als Client-Sever-Architektur konzipiert. Die Server-Komponente ist in einen WWW-Server integriert, die Client-Komponente ist als Java-Applet realisiert. Client und Server kommunizieren über das HTTP-Protokoll, so dass keine Beeinträchtigungen durch Firewall-Systeme zu erwarten sind. Durch den Einsatz im World Wide Web müssen gleichzeitig mehrere Client-Instanzen vom Server bearbeitet werden können.

Kapitel 4

Realisation der Generierungskomponente

Die Generierungskomponente hat die Aufgabe, komplexe Persona-Befehle in primitive Konstrukte, die die Abspielkomponente ausführen kann, zu zerlegen. Zusätzlich werden weitere benötigte Daten, wie z.B. Audiodateien für die Sprachausgabe, erzeugt.

4.1 Gestenkonzeption für Persona

Eine Animation von Persona besteht aus einer Bildsequenz, die mit einer Audioausgabe verknüpft ist. Dabei entspricht die Bildsequenz einer oder mehreren Gesten, die Persona ausführt. Jedes Bild wird aus zweidimensionalen Zeichenoperationen erzeugt. Um eine einfachere Ansteuerung zu gewährleisten, unterscheidet man zwischen primitiven und komplexen Gesten. Die komplexen Gesten setzen sich aus mehreren primitiven Gesten zusammen. Eine primitive Geste wird dagegen aus einer eventuell mit einer Audioausgabe verknüpften Bildsequenz gebildet.

Daneben lassen sich die Gesten entsprechend ihrer Einsatzgebiete weiter klassifizieren:

Komplexe Präsentationsakte: Der Präsentationsagent erhält vom jeweiligen Anwendungsprogramm Präsentationsaufgaben, die er selbständig ausführen soll. Diese Aufgaben, wie komplexe Zeigeoperationen oder Emotionsäußerungen, werden in einfachere Gesten zerlegt.

Navigationsaktionen: Unter diesem Oberbegriff lassen sich alle Gesten (komplex und primitiv) zusammenfassen, die eine Positionsänderung von Persona bewirken. Persona muss sich zu neuen Positionen bewegen lassen, z.B. um sich dadurch in eine bessere Zeigeposition zu bringen. Die Fortbewegungsart kann dabei von dem eingesetzten Charakter abhängen (Beispiele: Gehen, Springen, Fliegen).

Reaktive Aktionen: Um die Benutzerinteraktion zu erleichtern, muss Persona auf Benutzereingaben reagieren können. Dazu werden die reaktiven

Aktionen zur Verfügung gestellt: Die Anzeige eines Menüs oder das Verschieben von Persona mit der Maus gehören zu diesen.

Vitalaktionen: Damit Persona möglichst lebendig auf den Betrachter wirkt, muss sie sich auch in Phasen, in denen keine Präsentationsakte ausgeführt werden, bewegen. In diesen Fällen werden Vitalaktionen gezeigt, die im Gegensatz zu den vom Anwendungsprogramm initiierten Aktionen selbständig von Persona gestartet werden.

Elementaraktionen: Alle Aktionen, die sich nicht weiter in primitivere Aktionen aufteilen lassen, sind Elementaraktionen. Elementaraktionen bestehen aus einer nicht unterbrechbaren Bildsequenz, die eventuell von einer Audioausgabe begleitet wird.

Abbildung 4.1: Klassifikation der Persona-Aktionen

4.1.1 Gewinnung von Bildmaterial

Alle Gesten des Präsentationsagenten werden als Bitmapfolgen definiert. Die einzelnen Bitmaps können aus einer Vielzahl von Quellen gewonnen werden. Eine Möglichkeit ist, die Bitmaps von einem Zeichner entwickeln zu lassen. Diese cartoonartigen Darstellungen können einfach an den Geschmack der jeweiligen Betrachter angepaßt werden. Außerdem haben sie den Vorteil, dass sie vollkommen frei gestaltet werden können. Es können beliebige Phantasiefiguren als Präsentationsagenten verwendet werden. Oft kommen die vom Zeichner gestalteten Figuren bei dem Benutzer besser an als die auf anderem Weg gewonnenen

Darstellungen. Gezeichnete Figuren sind meistens detailreicher als durch andere Verfahren erzeugte Bilder. Es können alle Techniken, die aus Zeichentrickfilmen oder Comic-Heften bekannt sind, eingesetzt werden. Auch die Ladezeit, die in WWW-Anwendungen wichtig ist, kann durch Verwendung von Cartoongesten niedriger ausfallen, da in der Regel die Farbpalette relativ klein ist. Eine andere Möglichkeit ist die Verwendung eines Framegrabbers zur Gewinnung von Videoframes. Diese Videoframes können analog zu den gezeichneten Gesten eingesetzt werden. Allerdings müssen sie ebenfalls meistens manuell nachbearbeitet werden, da z.B. der mit der Videokamera erfasste Hintergrund entfernt werden muss. Nur in seltenen Fällen kann dies automatisch über Filter erledigt werden, weil Beleuchtungseffekte erschwerend wirken. Falls allerdings einfarbige Hintergrundflächen Verwendung finden und keine Schlagschatten auftreten, hält sich der manuelle Aufwand in Grenzen, da dann nur noch in Problembereichen die mittels Filter vorverarbeiteten Bilder nachbearbeitet werden müssen. Hat man allerdings die Möglichkeit Blue-Screen-Technologie einzusetzen, kann dieser Vorgang stärker automatisiert werden. Der Vorteil der Video-Frames ist darin zu sehen, dass reale Personen in Präsentationen als Präsentatoren auftreten können. Diese Agenten können etwa als virtueller Stellvertreter der realen Person eingesetzt werden. Außerdem müssen keine neuen Charaktere entworfen werden, was eine Einsparung von Arbeitszeit bedeutet.

Eine dritte Möglichkeit ist die Verwendung von dreidimensionalen Modellen. Im Gegensatz zu dem in Abschnitt 8.3 beschriebenen Verfahren soll aber hier vorab in einem Vorverarbeitungsschritt aus einem dreidimensionalen Modell unter Einbeziehung von Lichteffekten durch bildgebende Verfahren wie Ray-Tracing oder Radiosity [Foley et al., 1992] die Bitmap berechnet werden. Diese Bitmap kann dann analog zu den vorher beschriebenen Methoden eingesetzt werden. Auf dem Markt ist eine Vielzahl von Programmen erhältlich, die sich der Modellierung und der Animation dreidimensionaler Modelle von Charakteren annehmen. Zur Generierung der Bitmaps muss das Modell so manipulierbar sein, dass die für eine Bewegungssequenz erforderlichen Bilder berechnet werden können. Im Gegensatz zu der direkten Darstellung von 3D-Modellen kommt diese Vorgehensweise auch ohne aufwendige Hardware, wie z.B. Z-Buffer, aus. Außerdem lassen sich bei allen hier verwendeten Verfahren die Bitmap-Daten mit anderen zweidimensionalen grafischen Objekten, wie z.B. Linien (Abschnitt 4.1.2), kombinieren.

4.1.2 Der Persona-Editor

Der Persona-Editor dient zur interaktiven Definition der Persona-Gesten. Dabei müssen die benötigten Bitmaps definiert werden, die danach zu Bitmapsequenzen zusammengesetzt werden. Außerdem erlaubt der Persona-Editor das Unterlegen der Bitmapfolgen mit Audioausgaben, das Hinzufügen von Zeitinformationen und die Definition von Ruheaktionen.

Als Eingabe können entweder GIF-Animationen oder einzelne Bilder verarbeitet werden. Um die Übertragungszeit bei dem Einsatz in WWW-Applikationen zu verkürzen, ist es sinnvoll, mehrere Einzelbilder in einer großen Bilddatei zusammenzufassen, da so weniger Einzelanfragen an den WWW-Server gestellt

Abbildung 4.2: Übersicht über verschiedene von Persona eingesetzte Bitmaptypen: 3D modellierte Figur, Video-Frames und handgezeichnete Bilder

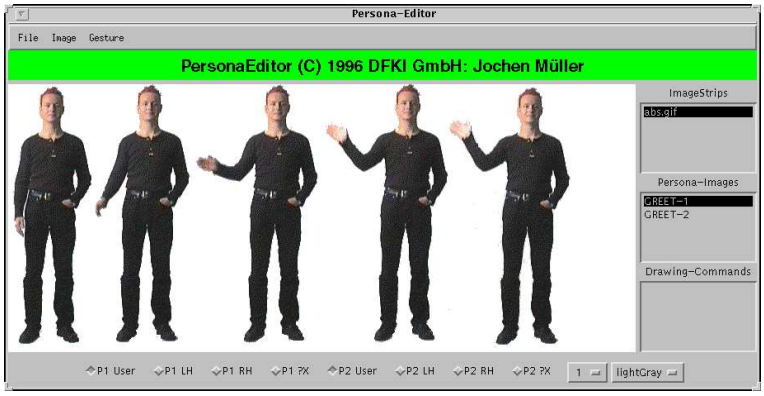


Abbildung 4.3: Der Persona-Editor

werden müssen, die relativ zeitaufwendig sind.

Der Persona-Editor hat die Möglichkeit, mehrere Gesten, die als GIF-Animation vorliegen, auf einmal zu importieren. Dabei werden mehrfach vorliegende Bitmaps automatisch erkannt und aussortiert. Beispielsweise wird die Bitmap mit der Grundhaltung von Persona in vielen Gesten vorkommen. Durch Vergleich der einzelnen Bitmaps untereinander können solche Bilder erkannt und durch Referenzen ersetzt werden. Der Importfilter fasst auch die Bilder einer Animationssequenz nach Entfernung der doppelt vorkommenden Bilder in eine große Bitmap, in ein Image-Strip, zusammen. Bei dem Import von GIF-Animationen werden nicht nur die Bilder eingelesen, sondern es wird auch eine neue Geste, die der Animation entspricht, angelegt.

Ist die Eingabe für den Editor ein Image-Strip, so erlaubt der Editor die interaktive Auswahl der einzelnen Bilder.

Hat der Gestenentwickler die einzelnen Bilder definiert, so kann er noch ein-

zelne grafische Objekte hinzufügen, wie Linien, gefüllte und nichtgefüllte Rechtecke und Ovale, Texte mit verschiedenen Zeichensätzen und andere Bitmaps. Dabei können Farbe und Strichstärken frei gewählt werden.

Dadurch, dass sich neue Bilder auch aus Teilen anderer Bitmaps zusammensetzen lassen, kann eine weitere Reduktion der zu übertragenden bzw. zu speichernden Datenmenge erfolgen (Abb 4.4). Ein Einsatzgebiet des Präsentationsagenten ist das WWW, deshalb muss die übertragene Datenmenge möglichst gering gehalten werden, so dass sich der Mehraufwand bei der Erstellung der Gestenbasis lohnt. Je nach Gesten lassen sich nun ungefähr 70% der Daten für die Bitmaps einsparen.

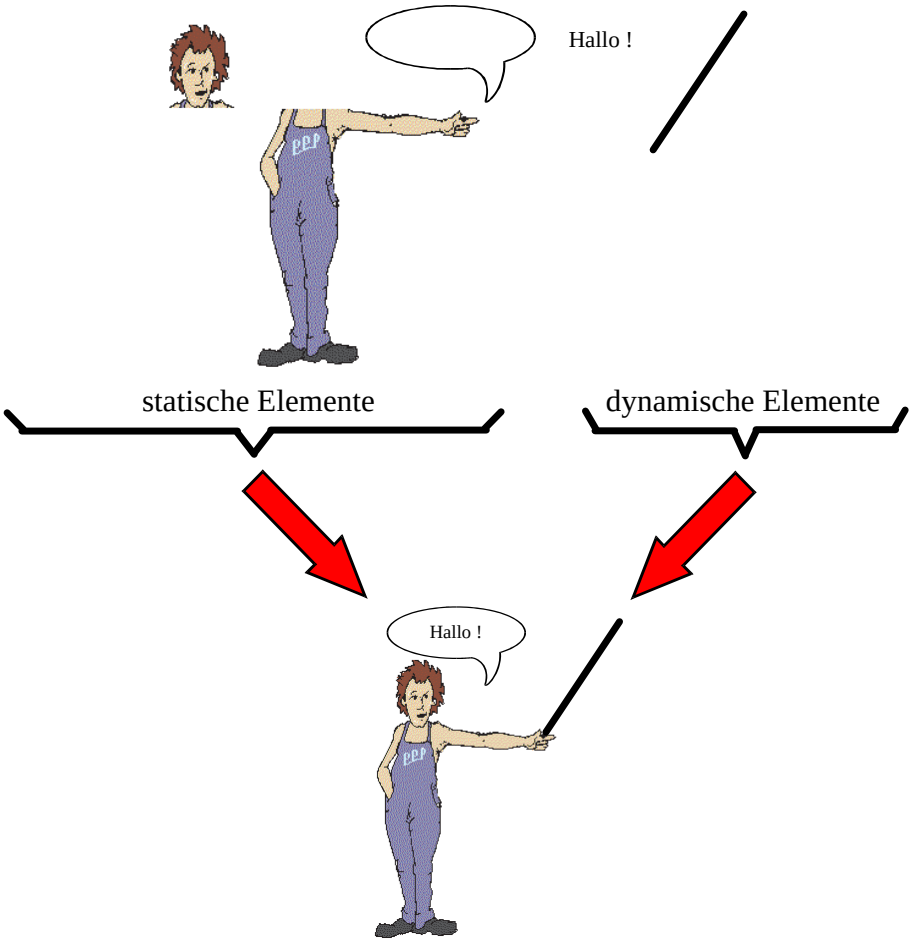


Abbildung 4.4: Konstruktion eines Gesten-Frames aus grafischen Einzelteilen. Ein Frame besteht aus einem statischen und einem dynamischen Anteil. Beide werden zur Laufzeit kombiniert.

Die grafischen Elemente können interaktiv definiert werden. Dabei kann die Definition noch Variablen enthalten, die erst während des Abspielvorgangs der Bildsequenz ersetzt werden. Beispielsweise wird bei der Definition des Zeigestocks ein Punkt, z.B. der Handgriff des Zeigestocks, mit der Maus festgelegt und die Position des anderen Endes des Stockes wird als Variable festgelegt,

da diese Information erst zur Präsentationszeit bekannt ist. Bei der Ausgabe von Sprechblasen kann der Text erst zur Ablaufszeit eingefügt werden und muss deshalb bei der Gestendefinition als Variable definiert werden.

In einem zweiten Schritt werden nun die einzelnen Gesten-Frames zu den primitiven Gesten zusammengefügt. Dabei können einzelne Frames auch in mehreren primitiven Gesten verwendet werden.

Der Persona-Editor erlaubt die interaktive Definition der Offsets der einzelnen Bilder zueinander. Dabei muss man zwischen zwei Offsettypen unterscheiden. Zum einen muß ein Offset festgelegt werden, der die bei der Definition der einzelnen Frames entstehenden Positionierungsfehler ausgleicht, d.h. dieser Offset sorgt dafür, dass zwei aufeinander folgende Bilder genau übereinander zu liegen kommen. Hätte man diesen Offsettyp nicht zur Verfügung, müsste man alle Bilder gleich groß zeichnen. Da man in diesem Falle als Bildgröße die maximal mögliche Ausdehnung von Persona annehmen müsste (d.h. z.B. beide Arme ausgestreckt, Körper maximal gedehnt), würde man viel Speicherplatz und Übertragungszeit bei Einsatz in WWW-Applikationen sinnlos verbrauchen. Stattdessen kann für jedes Frame ein Offset definiert werden, der, bevor dieses Frame dargestellt wird, zu der aktuellen Position addiert wird, danach wird diese Addition wieder rückgängig gemacht.

Der andere Offsettyp wird bei Gesten benötigt, bei denen die Position von Persona dauerhaft verändert wird. Dies ist insbesondere bei Bewegungsgesten wie Laufen und Springen der Fall. Zu jedem Frame kann innerhalb einer Geste ein Offset definiert werden, der zu der aktuellen Position addiert wird.

Passend zur Konzeption einer Personageste kann jede Geste von einer Audioausgabe begleitet werden. Entsprechend bietet der Editor die Möglichkeit an, eine Audiodatei begleitend zu einer Geste zu definieren. Audiodateien können ebenfalls mehrfach in verschiedenen Gesten verwendet werden.

Der Persona-Editor erlaubt die Definition der Dauer der einzelnen Frames innerhalb einer Geste. Die Gesamtdauer einer primitiven Geste ergibt sich dann aus der Summe der einzelnen Framezeiten.

Zusätzlich können über den Editor einzelne Gesten als Ruheaktionen klassifiziert werden. Dabei muss der Designer die Zeitdauer festlegen, nach deren Ablauf diese Ruheaktion frühestens gestartet werden kann (s. Abschnitt 5.3).

Oft können primitive Gesten durch Kopieren bzw. Kopieren und Umkehren der Bildreihenfolge definiert werden. Diese Funktionalität des Editors erleichtert dem Gestenentwickler die Arbeit.

Eine weitere Fähigkeit des Editors ist die automatische Generierung von WWW-Präsentationen aller definierten Gesten. Dadurch kann der Entwickler die definierten Gesten ohne Aufwand betrachten und eventuelle Fehler ermitteln.

Durch die besondere Konzeption der Gesten (dynamisches Zusammensetzen der einzelnen Frames, wobei in der Definition noch Variablen enthalten sein dürfen, die erst zur Laufzeit ersetzt werden) wird dem Einsatzbereich von Persona als WWW-Präsentationsagent besonders Rechnung getragen. Erst durch einen speziellen Editor wird es möglich, diese komplexen Gestendefinitionen zu erstellen.

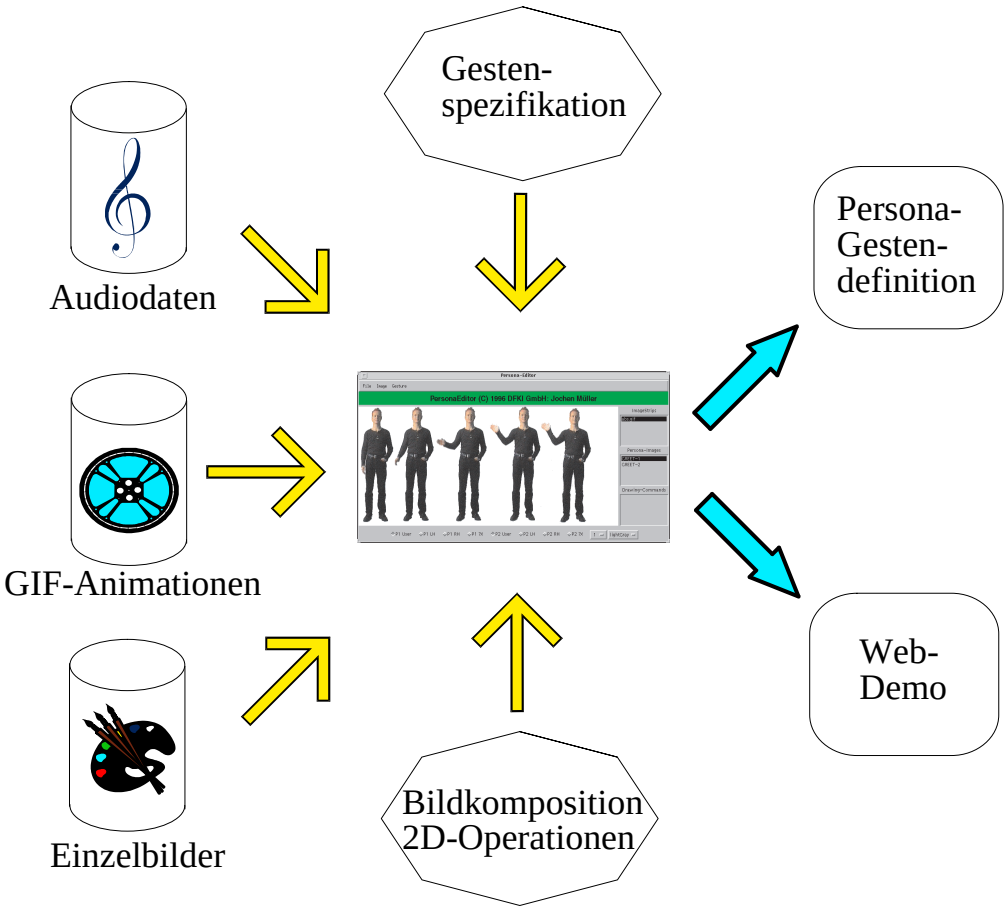


Abbildung 4.5: Funktionaler Überblick über den Persona-Editor

4.2 Regelsystem zur Steuerung der Persona-Engine

Die Eingabe des Generators erfolgt in Form eines Skriptes, dessen Syntax dem XML-Standard [Bray et al., 1999] angepasst ist. Das Eingabeskript enthält Befehle zur Definition von primitiven und komplexen Gesten, zur Ansteuerung des Persona-Players sowie zur Interaktion zwischen Präsentationsplaner und Persona-Player.

Das im Folgenden vorgestellte Regelsystem erlaubt den Aufbau komplexer Handlungsschritte aus einfacheren Persona-Aktionen. Bei der Konstruktion fließen neben zeitlichen Randbedingungen auch Informationen über die aktuelle Persona-Konfiguration (z.B. Gemütszustand, Temperament usw.) ein.

Die für eine Präsentation zur Verfügung stehenden Befehle setzen sich aus domänenspezifischen und -unspezifischen Befehlen zusammen. Zur einfachen Realisierung einer konkreten Applikation muss es für den Anwender möglich sein, sich auf einfache Art neue Befehle zu definieren. Dies wird umso notwendiger, wenn der Persona-Player um neue Befehle erweitert wird (s. Abschnitt 5.1). Die Persona-Regeln (domänenabhangig und -unabhangig) werden wie das Skript selbst in XML-Syntax definiert.

Jede Persona-Regel erfasst die von ihr benötigte Zeit, so dass jederzeit ein Abgleich mit dem mit Zeitinformationen annotierten Präsentationsplan und dem Persona-Skript erfolgen kann.

Der Generator bietet außerdem die Möglichkeit, den Generierungsvorgang auf vielfältige Art zu dokumentieren. Zum einen unterstützt er die Möglichkeit, zu jeder Regeldefinition eine Beschreibung abzulegen, aus der eine Regeldokumentation über XSL-Skripte [Deach, 1999] in einer vom XML/XSL-Prozessor unterstützten Sprache wie HTML oder LaTeX erzeugt werden kann. Zu diesem Zweck kann die Dokumentation der einzelnen Regeln in den Regeltext eingebunden werden.

Der Generierungsvorgang kann durch den Generator als Zerlegungsbaum visualisiert werden. Dabei stehen die Blätter des Baumes für die primitiven Aktionen, die in das vom Persona-Player direkt ausführbare Skript eingefügt wurden.

Der Befehlsfluss innerhalb des generierten Skriptes lässt sich ebenfalls als Graph visualisieren, so dass Sprünge nachvollziehbar werden.

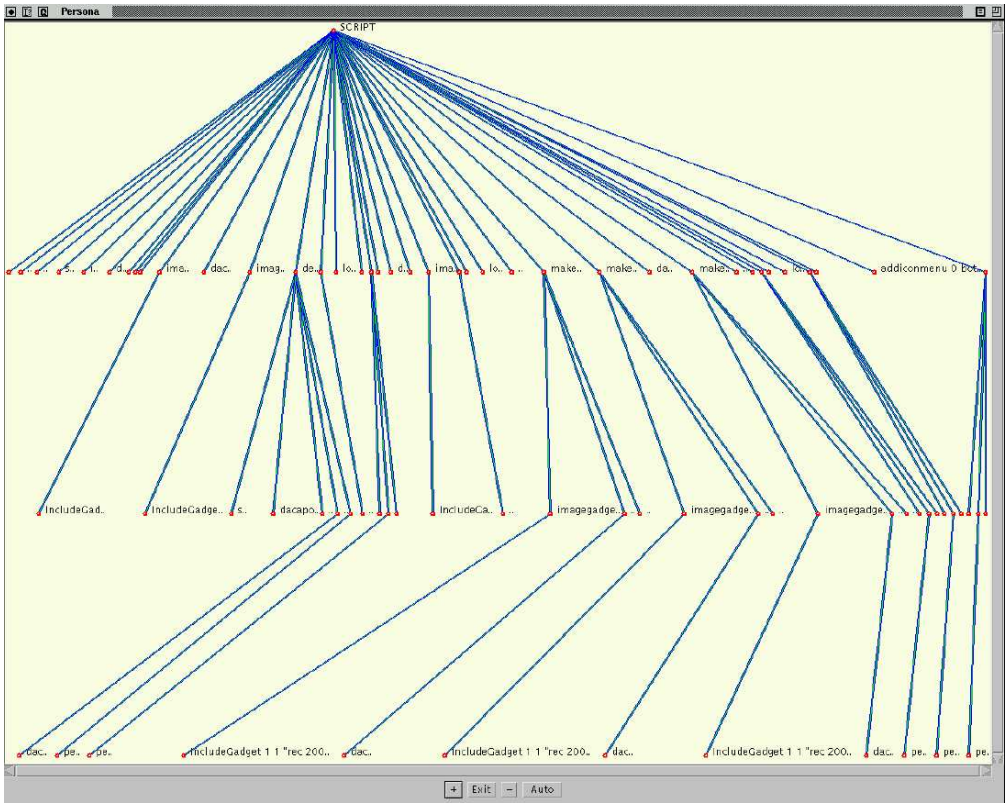


Abbildung 4.6: Visualisierung der Dekomposition von Gesten durch den Generator

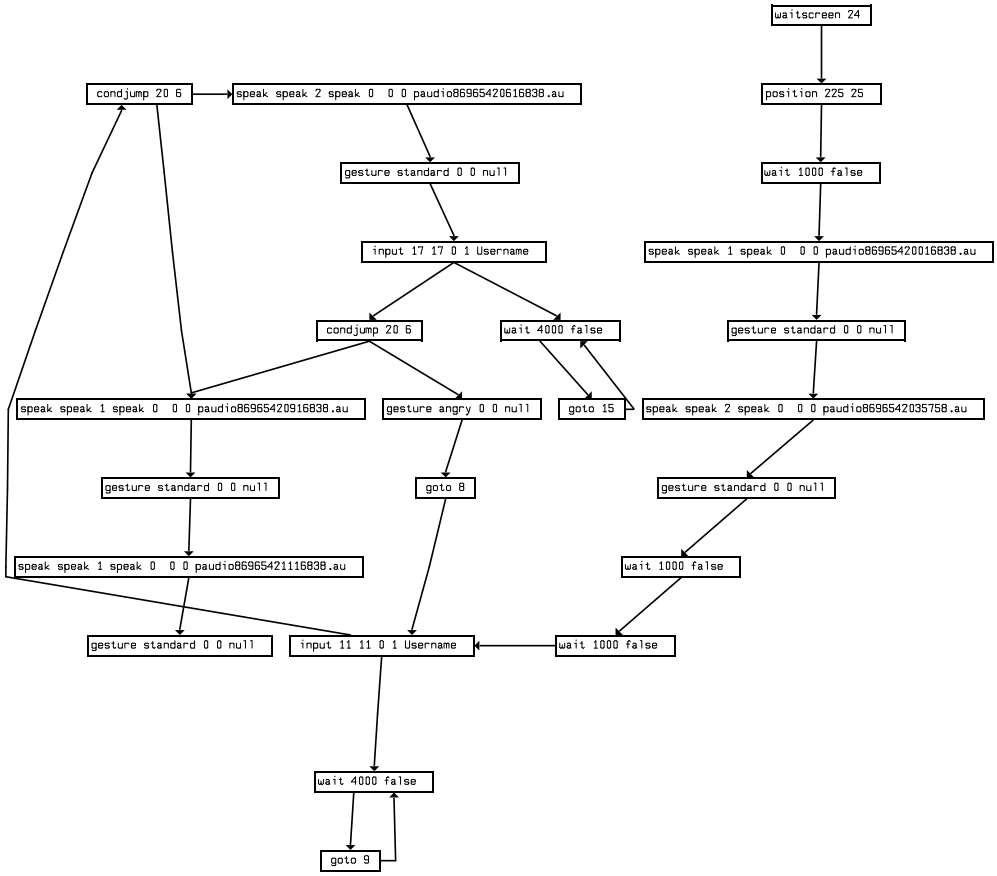


Abbildung 4.7: Visualisierung des Programmflusses innerhalb eines Persona-Skriptes

4.3 Sprachdefinition

Die auf XML basierte Steuerungssprache des Persona-Generators wird von einem XML-Parser [IBM, 1999] gelesen und in das Document Object Model DOM [Wood und Hors, 1999] überführt. Durch die Dokumentenverarbeitung auf Basis von DOM kann von dem verwendeten Parser abstrahiert werden, er kann beispielsweise auch gegen andere Parser ausgetauscht werden, ohne dass Module des Persona-Generators geändert werden müssten.

Die Definition einer Persona-Umgebung erfolgt durch den PERSONA-Tag:

```
<PERSONA width="600" height="300" color="white">
 ...
</PERSONA>
```

Die Attribute definieren die Größe der Fläche, die für die Präsentation mit Präsentationsagenten zur Verfügung steht, und die zu verwendende Hintergrundfarbe.

Innerhalb des PERSONA-Tags werden die einzelnen Präsentationsagenten, die innerhalb dieser Fläche auftreten sollen, definiert:

```
<PERSONASCRIPT name="wichtel" bitmap="cartoon" speech="m-e">
 ...
</PERSONASCRIPT>
 ...
<PERSONASCRIPT name="wizard" bitmap="video" speech="m-g">
 ...
</PERSONASCRIPT>
```

Jeder Agent hat einen eindeutigen Namen. Das äußere Erscheinungsbild wird über das Attribut bitmap definiert. Der Sprachsynthesizer wird mittels Attribut speech gewählt. Dabei wird der Type des Sprachsynthesizers bzw. seine Einstellungen nach dem Schema $s - l[-t]^*, s \in \{m, w\}, l \in \{g, e, f, \ldots\}$ gesetzt.

s: Geschlecht des Sprechers (männlich $= m$, weiblich $= w$)

l: Sprache (z.B. Deutsch $= g$, Englisch $= e$, Französisch $= f$)

Falls speziellere Versionen von Sprachsynthesizern verfügbar (z.B. für bestimmte Tonfälle) sind, kann durch Anfügen weiterer Schlüsselwörter eine genauere Spezifikation erfolgen.

Innerhalb jedes PERSONASCRIPT-Tags können Gesten definiert und auch aufgerufen werden. Mit Hilfe des INCLUDE-Befehls können externe Dateien eingebunden werden, als Argument wird die URL der einzufügenden Datei übergeben. Auf diese Weise lassen sich Bibliotheken mit Verhaltensmustern für Präsentationsagenten anlegen:

```
<INCLUDE url="http://persona.dfki.de/generator/complex.xml"/>
```

Innerhalb des Skriptes wird unterschieden:

- Definition primitiver Gesten/Befehle
- Definition komplexer Gesten/Befehle
- Anwendung der Gesten/Befehle

4.3.1 Primitive Gesten

Unter primitiven Gesten werden im Folgenden Befehle verstanden, die unmittelbar einem Befehl der Persona-Player-Eingabesprache entsprechen oder die unmittelbar durch eine Subklasse der Java-Klasse PersonaPrimitiveRule implementiert sind. Analog zur Persona-Abspielkomponente (Abschnitt 5.1) ist es möglich, Anweisungen in Form einer Java-Klasse zu implementieren, die vom Regelinterpreter dynamisch geladen und ausgeführt wird. Dadurch wird die Erweiterung des Generators auch um komplexe Funktionen vereinfacht. Spezielle Anwendungsprobleme werden so lösbar. Um eine über diese Schnittstelle implementierte Regel zu definieren, muss eine Unterklasse zu der in Abbildung 4.8 angegeben Java-Klasse implementiert werden. Der Klassenname entspricht

```
public class PersonaRule {
  String name;
  public PersonaRule (String name) {
     this.name = name;
     }

  public String getName() {
     return name;
     }

  public int generateScript(Hashtable actual,
                            PersonaAgent agent) {
     return 0;
     }
}
```

Abbildung 4.8: Die Oberklasse PersonaRule

dem Regelnamen. Die Funktion generateScript definiert die Aktionen, die diese Regel ausführen soll.

Die Methode generateScript erhält als Eingabe eine Hashtabelle mit den aktuellen Parametern und das Agentenobjekt, das diesen Befehl ausführen soll. Als Ergebnis liefert sie eine Abschätzung über die von der Aktion benötigte Zeit.

Trifft der Regelinterpreter auf eine so definierte Regel, so lädt er dynamisch die passende Klasse und evaluiert diese.

Einen Spezialfall stellen die Befehle dar, die unmittelbar durch einen Befehl der Eingabesprache für den Persona-Player ausgedrückt werden können. In diesem Fall enthält die Definition der Regel Variablen (durch $ gekennzeichnet), die durch die aktuellen Variablen ersetzt werden.

Im Eingabeskript für den Generator werden primitive Regeln wie in Abbildung 4.9 angegeben definiert. Die primitiven Gesten werden von

```
<PRIMITIVEGESTURES> ...  </PRIMITIVEGESTURES>
```

umschlossen.

Eine primitive Geste hat einen eindeutigen Namen. Falls es einen Befehl des Persona-Players gibt, der dieser Gestendefinition des Generators entspricht, so kann der Player-Befehl über das Attribut „script“ übergeben werden. Dabei können durch $ markierte Variablen benutzt werden, die aus der Menge der aktuellen Variablen belegt werden. Falls der Befehl durch eine Java-Klasse definiert ist, so enthält das Attribut „script“ das Schlüsselwort „extern“.

Mittels des Attributs „alternatives“ besteht die Möglichkeit, eine Liste mit alternativen Gesten anzugeben. Ist die Hauptgeste (d.h. die Geste, die nach gesture in der Skript-Zeile steht) in den momentan benutzten Bitmaps nicht vorhanden, so werden diese Gesten der Reihe nach getestet. Auf diese Weise hat man die Möglichkeit, Namensunterschiede zwischen den einzelnen Bitmapbasen auszugleichen. Dies ist insbesondere dann sinnvoll, wenn man verschiedene

```
<PRIMITIVEGESTURES>

<DEFPRIMITIVE
        name="stickrightpoint3"
        alternatives="rightspeakstickpoint3"
        script="gesture stickrightpoint3 $x $y null"/>


<DEFPRIMITIVE name="IncludeGadget" script="extern"/>
...
</PRIMITIVEGESTURES>
```

Abbildung 4.9: Definition primitiver Regeln für den Generator. Ist die Regel durch eine Java-Klasse implementiert, so wird dies durch das Schlüsselwort „extern“ definiert. Falls die Regel einen entspechenden analogen Befehl in der Sprache des Players besitzt, wird die Kommandozeile angegeben, wobei die durch $ markierten Variablen durch die aktuellen Parameter ersetzt werden.

Bitmaptypen für unterschiedliche Personainstanzen verwenden und mit der gleichen Regelmenge ansteuern will, wobei nicht in jeder Bitmapmenge alle Gesten definiert sind.

Heißt beispielsweise eine Zeigegeste für einen Bitmaptyp „POINTTO “ und für eine andere Bitmapmenge „ZEIGEAUF “ , so ist es mit dem Attribut „alternatives “ möglich, eine primitive Geste für beide Bitmapsequenzen zu definieren.

4.3.2 Komplexe Gesten und Aktionssequenzen

Eine Folge von Definitionen komplexer Gesten wird von

```
<COMPLEXGESTURES> ... </COMPLEXGESTURES>
```

umschlossen. Eine komplexe Geste wird mittels

```
<DEFINE-GESTURE name="gestenname"> ... </DEFINE-GESTURE>
```

definiert. Eine komplexe Geste besteht aus einer Aktionssequenz. Dabei können die einzelnen Aktionen entweder primitive Gesten, komplexe Gesten oder spezielle Schlüsselwörter sein. Die speziellen Schlüsselwörter dienen dazu, die Expansion der Regeln zu steuern. Die Parameter der komplexen Geste werden entsprechend weiterpropagiert.

Das aktuelle Personaskript für einen Agenten enthält nicht nur die Definition von Befehlen und Gesten, sondern auch deren Aufruf. Diese Aktionen werden im Skript durch die Tags

```
<ACTION> ... </ACTION>
```

umschlossen. Wird der Agent gestartet, werden alle in diesem Abschnitt angegebenen Befehle in der Reihenfolge ihres Auftretens ausgeführt. Die Parameter der einzelnen Befehle müssen spezifiziert sein. Es können alle definierten Gesten

```
<PERSONA width="600" height="300" color="white">
  <PERSONASCRIPT name="wichtel" bitmap="cartoon" speech="m-g">
    <INCLUDE url="http://persona.dfki.de/primitives.xml"/>
    <COMPLEXGESTURES>
       <DEFINE-GESTURE name="sayhello">
         <greet/>
         <wait duration="1000"/>
         <speak text="Hallo. Ich bin Persona."/>
       </DEFINE-GESTURE>
    </COMPLEXGESTURES>
    <ACTION>
      <sayhello/>
    </ACTION>
  </PERSONASCRIPT>
</PERSONA>
```

Abbildung 4.10: Beispiel einer Eingabe für den Generator

(primitive und komplexe) verwendet werden. Daneben gibt es spezielle Befehle, die für die Ablaufsteuerung genutzt werden können. Ein Besipiel ist in Abbildung 4.10 angegeben.

Innerhalb einer Aktionssequenz können symbolische Sprungadressen definiert werden. Solche Sprungadressen müssen einen eindeutigen Namen besitzen. Symbolische Sprungadressen können dazu benutzt werden, um „Unterprogramme“ innerhalb des Skriptes zu definieren, die spezielle Aufgaben übernehmen können. Beispiele für solche Unterprogramme sind z.B. Handlungen, die wiederholt vorkommen können, z.B. die Beschreibung eines Buttons, die immer dann wieder gezeigt wird, wenn der Benutzer mit der Maus über den Button fährt. Insbesondere bei der Generierung von WWW-Präsentationen ist die Verwendung von symbolischen Sprungadressen wichtig, da während der Generierungsphase das ganze Skript erzeugt werden kann, welches zum Benutzer übertragen wird. Zur Präsentationszeit ist dann keine Interaktion mit dem Server mehr notwendig, weil die Präsentation schon kodiert in den Unterprogrammen für die einzelnen Teilpräsentationen vorliegt. Es kann so zu keinen Verzögerungen im Präsentationsablauf kommen.

4.3.3 Definition des Persona-Temperaments

Bei einer Präsentation gibt es äquivalente Zerlegungen für ein Präsentationsziel, welche sich allerdings in der Ausdrucksstärke bzw. in der Stärke der Emotionsäußerung oder auch in der Lebendigkeit unterscheiden. Die einzelnen Betrachter weichen bei der Wahrnehmung solcher Verhaltensmuster stark voneinder ab. Es bleibt dem Betrachter überlassen, je nach Lust und Laune zu wählen, ob eine Präsentation mehr oder weniger lebendig wirken oder ob Persona durch stärkere oder schwächere Emotionen ihren internen Zustand zum Ausdruck bringen soll. Falls der Benutzer keine Wahl trifft, wird ein mittlerer Wert als Standardeinstel-

lung angenommen. Manchmal kann aus einem Benutzermodell schon auf eine gute Wahl dieses Parameters geschlossen werden. Ein Indiz ist z.B. das Alter des Betrachters. Ziehen jüngere Betrachter lebendige Präsentationen vor, so wünschen ältere Betracher eher ruhige und möglichst sachliche Präsentationen. Zeigt man die gleiche Präsentation zwei unterschiedlichen Betrachtern, ist dem einen Betrachter die Präsentation nicht lebendig genug, während der andere sich schon von zuviel Aktivität auf dem Bildschirm erschlagen fühlt. Häufig ruft die gleiche Präsentation bei Betrachtern unterschiedliche Reaktionen hervor. Während für den einen Betrachter Persona zu lebendig und aktiv ist, verlangt ein anderer noch mehr Temperament von Persona.

Das Temperament, das der Präsentationsagent während der Präsentation zeigt, hat einen großen Einfluss auf die gesamte Wirkung einer Präsentation. Den Temperamentgrad von Persona bestimmt die Variable DECORATION-LEVEL, die jederzeit vom Präsentationsplaner bzw. durch den Benutzer geändert werden kann. Hat die Variable den Wert 0, so entspricht dies einer extrem sachlichen Präsentation. Es werden nur die absolut notwendigen Aktionen durchgeführt. Der Wert 1 bedeutet maximales Temperament. Zur Auflockerung werden in die Präsentation an vielen Stellen Aktionen eingefügt, die besondere Lebendigkeit ausstrahlen.

Die Funktionalität wird durch die Spezialform

```
<INSERT-DECORATION>
 <gestenname ... decoration-level="val1"/>
 ...
 <gestenname ... decoration-level="valn"/>
</INSERT-DECORATION>
```

realisiert. Erreicht der Generator einen solchen Befehl, so vergleicht er die Werte val1,..., valn mit dem aktuellen Wert der Variable DECORATION-LEVEL und führt die entsprechende Aktion aus. Zuerst wird geprüft, ob die zur Verfügung stehende Zeit noch ausreichend ist (d.h. falls keine Zeit mehr zur Verfügung steht, werden alle Befehle innerhalb des Tags INSERT-DECORATION ignoriert). Ein Beispiel für eine Anwendung ist in Abbildung 4.11 angegeben.

Neben dieser einfachen Auswahl von Gesten können auch komplexe Befehlsmakros innerhalb von INSERT-DECORATION eingesetzt werden, um die Präsentation abwechslungsreicher zu gestalten.

4.3.4 Zufallsauswahl von Befehlen

Um die erzeugten Präsentationen abwechslungsreicher zu gestalten, ist es wichtig, dass nicht immer die gleichen Befehlssequenzen benutzt werden. Zu diesem Zweck kann man versuchen, zu einigen Aufgaben äquivalente, aber trotzdem unterschiedliche Sequenzen zu definieren und dann aus der Menge der Möglichkeiten zufällig eine aktuelle Präsentationssequenz auszuwählen.

Zu diesem Zweck beherrscht der Persona-Generator den Befehl,

```
<NONDETERMINISTIC-CHOICE>
 <AKTION1/>
```

```
<INSERT-DECORATION>
 <readbook decoration-level="0.7"/>
 <respire decoration-level="0.3"/>
 <standard decoration-level="0"/>
</INSERT-DECORATION>
```

DECORATION-LEVEL=0.2 DECORATION-LEVEL=0.4 DECORATION-LEVEL=0.8

Abbildung 4.11: Anwendung von INSERT-DECORATION

```
...
<AKTIONn/>
</NONDETERMINISTIC-CHOICE>
```

der es erlaubt, zufällig eine Aktion auszuwählen und seine Dekomposition an dieser Stelle in das primitive Persona-Skript einzufügen.

Durch die Definition möglichst vieler solcher Aktionen kann eine besonders abwechslungsreiche Präsentation erzeugt werden, wobei der Effekt noch durch Schachtelung mehrerer dieser Aktionen verstärkt werden kann.

Im Gegensatz zu der im Persona-Player implementierten Funktion, die es erlaubt aus einer Menge von Aktionen, die in vollständig expandierter Form innerhalb des primitiven Skriptes für den Persona-Player vorliegen, zufällig eine Aktion auszuwählen, können mit diesem Befehl nicht nur bestehende Teilskripte ausgewählt werden, sondern es können auch neue Daten (z.B. Sprachausgabe) generiert werden.

```
<DEFINE-GESTURE name="visualize-searching">
  <NONDETERMINISTIC-CHOICE>
    <read-book/>
    <thinking/>
    <speak "Ich suche. Bitte haben Sie Geduld."/>
   </NONDETERMINISTIC-CHOICE>
</DEFINE-GESTURE>
```

Abbildung 4.12: Visualisierung einer Suchanfrage: Bei jeder neuen Suchanfrage wird aus der Menge der definierten Aktionen zufällig eine Geste ausgewählt.

4.3.5 Visualisierung des Gefühlszustandes

Der Präsentationsagent stellt eine Möglichkeit bereit, Gesten in Abhängigkeit vom aktuellen „Gefühlszustand“ auszuwählen. In dieser einfachen Realisierung wird dieser Gefühlszustand des Agenten durch eine Variable FEELING realisiert, die einen Wert zwischen 0 und 1 annehmen kann. In diesem einfachen Modell würde 0 einen tieftraurigen Agenten und 1 einen überaus glücklichen Agenten repräsentieren. Im Verlaufe einer Präsentation kann durch Manipulation von FEELING der Gemütszustand des Agenten verändert werden. Beispielsweise kann durch das Fehlschlagen eines Planungszieles die Stimmung des Agenten sinken, d.h. die Variable FEELING wird verringert. Diese Vorgehensweise ist auch im Oz-System anzutreffen [Bates et al., 1992c]. Durch Verwendung des Konstrukts

```
<SHOWFEELING>
 <AKTION1 actual-feeling="f1"/>
 ...
 <AKTIONn actual-feeling="fn"/>
</SHOWFEELING>
```

wird es innerhalb eines Skriptes möglich, eine passende Geste auszuwählen.

4.3.6 Bedingte Anweisungen

Die Eingabesprache für die Präsentation enthält auch ein Konstrukt, das analog zu den if-Anweisungen gewöhnlicher Programmiersprachen funktioniert.

```
<IF condition="Bedingung">
 <IFAKTION/>
 <ELSEAKTION/>
</IF>
```

Dabei kann die Bedingung in der aktuellen Version $<, >$ (für numerische Daten) und $=$ für beliebige Datentypen als Operator enthalten. Eine Bedingung kann Konstanten und Variablen enthalten. Die Bedingung muss in Präfix-Notation vorliegen. An Stelle von IFAKTION bzw. ELSEAKTION können beliebige Gesten aufgerufen werden.

4.3.7 Zeitverteilung

Will man in einer Präsentation die für eine Präsentation zur Verfügung stehende Zeit berücksichtigen, so muss einerseits schon vom Präsentationsplaner die entsprechende Information für den Präsentationsagenten bereit gestellt werden (Abschnitt 7.1.1), andererseits muss der Präsentationsagent die Fähigkeit besitzen, die Zeit-Constraints bei der Umsetzung der Präsentationsaufgabe zu berücksichtigen.

In Realanwendungen ist es notwendig, auf zeitliche Benutzerpräferenzen einzugehen. Beispielsweise ist es nicht sinnvoll, einen Benutzer, der unter Zeitdruck

steht, mit unnötigen (d.h. nur der Ausschmückung dienenden) Animationssequenzen zu stören, sondern man sollte wenn irgend möglich die Präsentationsaufgabe in möglichst kurzer Zeit durchführen. Obwohl die Hauptaufgabe bei der Konzipierung einer effizienten Präsentation unter Berücksichtigung von zeitlichen Constraints bei dem Präsentationssystem liegt [Rist et al., 1997], kann doch auch der Präsentationsagent innerhalb seiner Möglichkeiten die Präsentation anpasssen.

Um die Zeitverteilung innerhalb einer Präsentation zu organisieren, stellt der Persona-Generator das Konstrukt

```
<TIME-DISTRIBUTION>
  <ACTION1 timedistribution="t1"/>
            ...
  <ACTIONN timedistribution="tN"/>
</TIME-DISTRIBUTION>
```

zur Verfügung. Mit dessen Hilfe ist es möglich, die für eine Präsentationsaufgabe eingeräumte Zeit auf die einzelnen Präsentationsschritte zu verteilen. An Stelle von ACTION1 ...ACTIONN können beliebige Persona-Regeln benutzt werden.

Das Konstrukt wertet die Umgebungsvariable MAXDURATION aus, die die maximale Zeitdauer des aktuellen Präsentationsschrittes definiert. Durch das Attribut „timedistribution “ jedes Subkonstrukts wird diese Gesamtzeit auf die einzelnen Aktionen verteilt. Die Werte dieser Attribute sind Zahlen zwischen 0 und 1. Sie repräsentieren den Zeitanteil, den diese Aktion an der Gesamtzeit der ganzen Operation hat. Durch die Möglichkeit, Konstrukte zu schachteln, ergibt sich der Vorteil, Zeitconstraints feiner zu spezifizieren. Wenn eine Zeitüberschreitung bei der Abarbeitung der Regel eintritt, so werden die restlichen Konstrukte ignoriert.

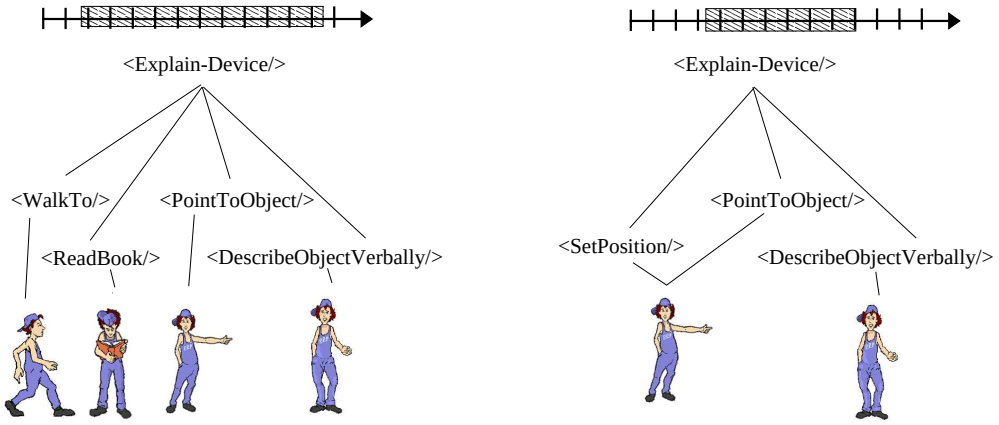


Abbildung 4.13: Anpassung der Animation an zeitliche Bedingungen

In Abbildung 4.13 ist dargestellt, wie sich die Expansion des gleichen Makros unter unterschiedlichen zeitlichen Bedingungen ändern kann. Hat man viel Zeit zur Verfügung, so kann auch die Präsentation mit mehr ausschmückenden

Elementen ausgestattet werden. Muss die Präsentation in kürzerer Zeit ablaufen, so werden Gesten durch kürzere Gesten ersetzt (z.B. WalkTo wird zu SetPosition) oder ganz weggelassen (z.B. ReadBook).

Durch Einfügung von WAIT-Anweisungen mit variabler Länge (in Abhängigkeit der von TIME-DISTRIBUTION propagierten Zeitspanne) lässt sich die Zeitdauer einer Präsentation individuell in kleinen Intervallen an die Präferenzen des Präsentationssystems anpassen. Eine Präsentation kann so relativ einfach zeitlich gestreckt oder gestaucht werden.

4.4 Berechnung von Gestenübergängen

Damit eine fließende Animation des Präsentationsagenten möglich wird, müssen die von ihm verwendeten Gesten logisch korrekt aufeinander folgen. Dabei sind Sprünge zwischen den einzelnen Gesten zu vermeiden. Ist beispielsweise die aktuelle Geste eine Zeigegeste mit der linken Hand und ist die nächste Geste eine Zeigegeste mit der rechten Hand, so müssen, damit die Gesten logisch aufeinander folgen, zwischen den beiden Gesten Aktionen eingefügt werden, die nacheinander die eine Zeigegeste beenden und danach in die korrekte zweite Zeigeposition übergehen. Um die Generierung einer Präsentation für den Anwender zu vereinfachen, muss der Generator in der Lage sein, diese Gestenübergänge automatisch zu berechnen. Unter Anwender wird hier entweder der Benutzer, der die Eingabespezifikation von Hand erstellt, oder ein Präsentationssystem, das die Spezifikation generiert, verstanden. Um dies zu ermöglichen, können die primitiven Gesten mit Vor- und Nachbedingungen annotiert werden. Durch diese Bedingungen wird der Inhalt der Gesten für das Generierungsmodul logisch beschrieben. Aus der logischen Gestenbeschreibung kann der Generator die Übergänge bestimmen und an den entsprechenden Stellen in das Skript einfügen. Da komplexe Gesten in primitive Gesten zerlegt werden, genügt es, die primitiven Gesten mit Bedingungen zu annotieren.

Die aktuellen Zustände des Präsentationsagenten in Bezug auf die eingesetzten Gesten werden durch Tupel dargestellt. Beispielsweise bedeutet das Tupel (leftarm standard), dass sich der linke Arm in der Standardposition befindet. Der aktuelle Zustand eines Agenten ist eine Menge solcher Tupel.

Um die Vor- und Nachbedingungen für die primitiven Gesten formulieren zu können, wird die Syntax zur Definition dieser Gesten um die Attribute „precondition “ und „postcondition “ erweitert:

```
<DEFPRIMITIVE
        name="rightpoint3"
        precondition=
         "(leftarm standard) (rightarm standard) (icon noicon)"
        postcondition="(rightarm point3)"
        script="gesture rightpoint3 $x $y null"/>
```

Bevor die Geste ausgeführt werden kann, muss zunächst der Ausgangszustand, der über die Vorbedingungen definiert ist, hergestellt werden. Nach dem

Ende der Geste befindet sich der Agent in dem Zustand, der mittels der Nachbedingung definiert ist.

4.4.1 Generierung der Zwischengesten mittels Planung

Im Persona-Generator werden die Zwischengesten durch ein Planungsverfahren bestimmt. Es wurden zwei verschiedene Ansätze implementiert: Planung der Gesten mittels eines Planungsystems wie im Folgenden beschrieben und als zweiter Ansatz die Generierung eines Zustandsautomats (Abschnitt 4.4.2).

Der in den Systemen AIA [André et al., 1996] und PPP [André et al., 1997] eingesetzte Präsentationsplaner PrePlan [André, 1995] stellt ein Planungsverfahren zur Verfügung, das nicht nur zur Planung von Präsentationen eingesetzt werden kann. Da er auch in einer Java-Version vorliegt [Tschernomas, 1999b], kann er unmittelbar von dem Persona-Generator genutzt werden.

Beim Einlesen der primitiven Gestendefinitionen durch den Persona-Generator erzeugt dieser dynamisch aus den darin definierten Vor- und Nachbedingungen Planungsstrategien, die während der Laufzeit von PrePlan, der hier als Gestenplaner eingesetzt wird, evaluiert werden. Die aktuellen Zustände werden dabei in der Wissensbasis des Planers abgelegt.

Die Hauptstrategie für eine Geste mit dem Namen gesture muss zunächst Strategien anwenden, die überprufen, ob alle Vorbedingungen gelten. Falls diese nicht gelten, so muss zunächst der korrekte Startzustand hergestellt werden. Zur Überprüfung des Startzustandes sind für jede der in der Liste der Vorbedingungen vorkommende Zustandsvariable zwei Planungsstrategien notwendig: Eine Strategie, die zutrifft, falls der Zustand erfüllt ist:

```
(define-plan-operator :header (A0 (state val))
                      :constraints (BELP (state val))
                      :inferiors ((A1 (noop))))
```

In diesem Fall ist keine Aktion notwendig, die primitive Aktion noop hat keine Funktion. Falls allerdings der Zustand nicht gilt, so müssen zunächst Aktionen ausgeführt werden. Für diese Aktionen kommen diejenigen, die in ihrer Nachbedingung den fehlenden Zustand haben, in Frage. Für Gesten mit Nachbedingung wird daher für jede Nachbedingung eine Strategie der Form

```
(define-plan-operator
   :header (A0 (state val))
   :constraints (*AND*
     ((*OR* ( (BELP (pstate pval)) (BELP (pstate pval not)))))
        ...
      (*OR* ((BELP (qstate qval)) (BELP (qstate qval not)))))
   :inferiors (
               (A1 (othergesture))
               (A2 (update-state-val))
              )
)
```

erzeugt. Diese Strategie besagt, dass man die Geste „othergesture “ anwenden muss, um den Zustand (state val) zu erfüllen. Das ist aber nur dann möglich, wenn man sich in einem der möglichen Ausgangszustände für die Geste befindet. Zusätzlich darf die Geste nicht schon einmal rückgängig gemacht worden sein.

Die für die Bewegung benötigte eigentliche Strategie für die Geste gesture sieht dann wie folgt aus:

```
(define-plan-operator
          :header (A0 (gesture))
          :inferiors (
                      (A1 (state1 val1))
                            ...
                      (An (staten valn))
                      (At (insertprimitive gesture))
                      (Ak (update-statek-valk))
                            ...
                      (Am (update-statem-valm))
                     ))
```

In den Unterzielen A1 bis An wird zunächst überprüft, ob alle Vorbedingungen für die Geste erfüllt sind. Gegebenenfalls werden sie hergestellt. In dem Unterziel At wird schließlich die primitive Aktion oder Geste ausgeführt. Die Unterziele Ak bis Am aktualisieren den Zustand. Sie ergeben sich aus der Menge der Nachbedingungen. Eine Strategie update-state-val kann wie folgt formuliert werden:

```
(define-plan-operator :header (A0 (update-state-val))
                      :constraints (BELP (state val))
                      :inferiors (A1 (noop)))

(define-plan-operator :header (A0 (update-state-val))
                      :constraints (BELP (state v))
                      :effect ((BELP (state val))
                               (BELP (state v not)))
                      :deeffect (belp (state v))
                      :inferiors (A1 (noop)))
```

Falls der korrekte Zustand gesetzt ist, muss keine Änderung durchgeführt werden. Für alle anderen Zustandsbelegungen wird sich die Änderung gemerkt. Zusätzlich muss festgehalten werden, welche Aktion durchgeführt wurde, da es sonst zu einer Endlosschleife kommt. Die Operatoren effect und deeffect manipulieren die Wissensbasis des Planers.

Wird dieses Verfahren eingesetzt, so muss zur Berechnung der Gestenübergänge das Planungsverfahren gestartet werden. Der Vorteil dieser dynamischen Herangehensweise ist darin zu sehen, dass neue Gesten schnell in das System integriert werden können. Es sind nur die primitiven Gesten mit ihren Vor-

und Nachbedingungen zu definieren, wobei eventuell die Menge der möglichen Zustände erweitert werden muss. Nun kann das System ohne einen zusätzlichen Compilierungsschritt von den neuen Gestendefintionen Gebrauch machen. Das System lässt sich daher besser auf besondere Präsentationsaufgaben anpassen. Es ist so flexibler.

4.4.2 Kompilierung des Planungsverfahrens

Zusätzlich zu dem dynamischen Planen aus dem vorigen Abschnitt lässt sich aus einer deklarativen Beschreibung ein Automat generieren, der die möglichen Systemzustände verwaltet und die Gesten kontextabhängig zerlegt. Durch die Generierung eines Automaten, der zur Laufzeit nur noch evaluiert wird, kann die Laufzeit des Systems verbessert werden.

Das System DaCaPo [Biringer, 1997] war ursprünglich dazu konzipiert, aus einer deklarativen Gestenspezifikation einen Automaten berechnen zu lassen, der die kontextabhängige Zerlegung komplexer Gesten in primitive Gesten zur Laufzeit übernimmt. Der Automat liegt nach der Generierung als C-Programm vor, welches sich mit einem Standard-C-Compiler in Maschinencode übersetzen lässt. Das so erzeugte C-Programm wurde zusammen mit der X-Version des Präsentationsagenten (Abschnitt 6.1) als Desktop-Präsentationsagent eingesetzt. Der Automat ist in diesem Fall unmittelbar mit dem Player verbunden, was eine sofortige Reaktion des Agenten auf Benutzereingaben (wie z.B. Verschieben des Agenten mit der Maus) vereinfacht.

Ziel war es, DaCaPo auch in der Java-Version des Präsentationsagenten einsetzbar zu machen. Deshalb wurde der DaCaPo-Generator um die Fähigkeit, auch Java-Code zu erzeugen, erweitert. Jetzt kann dieser auch innerhalb des Persona-Generators auf Server-Seite eingesetzt werden kann.

4.4.2.1 Code-Erzeugung in DaCaPo

Aus der deklarativen Definition der Gesten wird Programmcode generiert, der einen deterministischen endlichen Automaten (DEA) repräsentiert [Ball et al., 1997].

Zu jeder primitiven Aktion wird zunächst eine Funktion erzeugt, die zuerst das Abspielen der Geste (d.h. Bitmapfolge und optional Ausgabe einer Audiodatei) initiiert. Im Anschluss wird der aktuelle interne Zustand des Automaten dem durch die Nachbedingungen der Geste definierten Zustand angepasst. Diese Gesten entsprechen innerhalb des aufgespannten Automaten einem Pfad der Länge 1. Gleichzeitig testet diese Funktion, ob in der aktuellen Gesten-Datenbasis die passende Geste definiert ist. Da der Automat für verschiedene Gestenmengen verwendet werden soll, müssen bei der Definition einer primitiven Geste für DaCaPo äquivalente Bitmapfolgen angegeben werden. Die generierte Funktion testet der Reihe nach die äquivalenten Gesten auf Existenz. Falls keine Geste vorhanden ist, wird die Standardgeste eingefügt. Der Rückgabewert der generierten Funktion ist die für die Ausführung der Geste benötigte Zeit.

Bevor allerdings die eigentliche Aktion ausgeführt werden kann, müssen zunächst die Vorbedingungen der Geste erfüllt werden. Zu diesem Zweck wird für

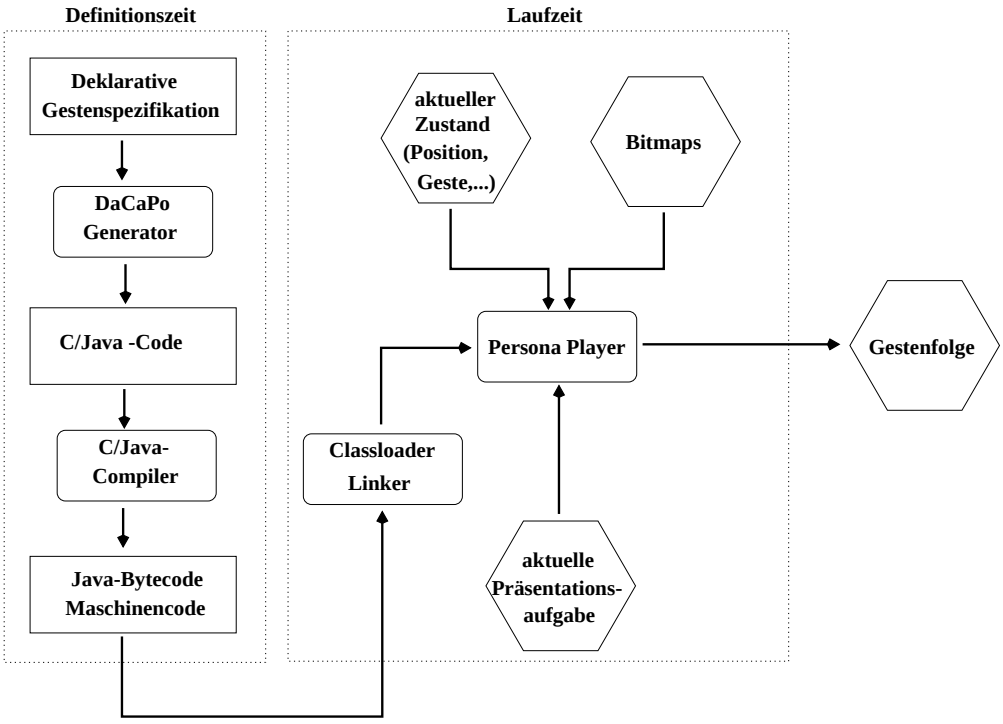


Abbildung 4.14: Gestenkompilation bei PPP Persona

jede Aktion mit Vorbedingungen eine Funktion generiert, die die Aufgabe hat, den geforderten Zustand herzustellen. Diese Funktion wird vor der eigentlichen Gestenausgabe aufgerufen.

Da der Automat zunächst unvollständig ist, d.h. es fehlen Übergänge zwischen Gesten, wird dieser durch einen Planungsschritt vervollständigt. Durch den Planungsschritt kann erreicht werden, dass noch fehlende Übergänge im Automaten, die für eine fließende Animation benötigt werden, ergänzt werden. Dabei wird ausgehend vom Zielzustand ein Operator gesucht, dessen Nachbedingung mindestens ein Literal enthält, das mit einem Literal des Zielzustandes unifiziert werden kann. Schrittweise kann auf diesem Weg ein Plan aufgebaut werden, wie der Zielzustand zu erreichen ist. Um herauszufinden, welche Ausgangszustände für eine Aktion möglich sind, wendet DaCaPo auf die Menge der Vorbedingungen alle möglichen primitiven Aktionen invers an, wodurch sich die Menge aller Zustände ergibt, aus denen die Aktion heraus gestartet werden kann. Diese Vorgehensweise erfolgt analog zur Methode des rückschreitenden Planens [Fikes und Nilsson, 1971]. Der Algorithmus terminiert, weil die Anzahl der möglichen Zustände beschränkt ist und weil angenommen wird, dass die Spezifikation widerspruchsfrei ist.

Neben der einfachen kontextsensitiven Zerlegung von Aktionen in primitive Aktionen erlaubt DaCaPo auch die Verwendung von Schleifenkonstrukten (solange eine Bedingung erfüllt ist, führe eine primtive Aktion aus) und bedingte Aktionen. Werden in der DaCaPo-Gestendefinition solche Aktionen eingesetzt, so werden diese in if bzw. while-Konstrukte der Zielsprache übersetzt.

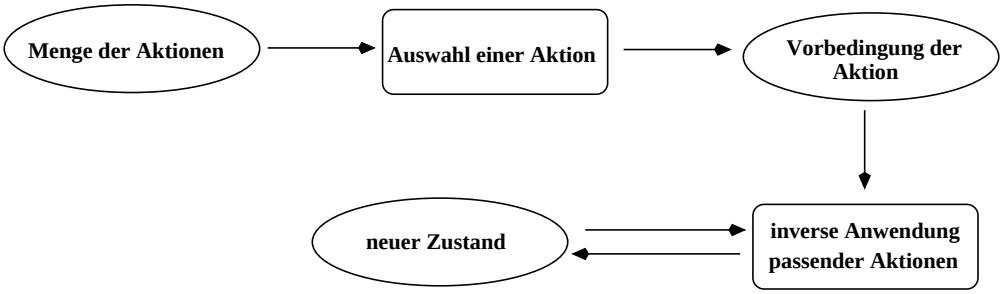


Abbildung 4.15: Uberblick über das Planungsverfahren von DaCaPo (nach [Biringer, 1997], verändert)

Durch diesen Planungsansatz können die fehlenden Übergänge im Verhaltensautomat berechnet werden. Aus den durch das Planungsmodul generierten Übergängen wird Programmcode generiert, der an die entsprechenden Stellen des bereits generierten Automats eingefügt wird.

DaCaPo (DaCaPo-C-Version) wurde in die X11-Version des Persona-Systems integriert . DaCaPo ist auch wahlweise im Generator der Persona-WWW-Version einsetzbar. In diesem Fall wird die Generierung von Java-Quellcode genutzt. Um die Ladezeit des Applets zu verringern, wird der Automat auf Serverseite benutzt, wo er im normalen Generator wahlweise integriert werden kann. DaCaPo wird in diesem Fall dazu gebraucht, um Aktionen zu koordinieren, die eine Bewegung von Persona bewirken, wie z.B. Zeigegesten oder Bewegungsaktionen. Befehle, die keine unmittelbare Bewegung von Persona nach sich ziehen, die z.B. eine eine Grafik anzeigen sollen, werden nicht von DaCaPo bearbeitet. Diese Gesten werden unmittelbar durch den Regelmechanismus des Generators in primitive Aktionen zerlegt.

Ein Problem bei dem Compilierungsansatz von DaCaPo ist sicherlich, dass die Erstellung von Animationen aufwendiger ist, da Compilierungsschritte (Deklarative Beschreibung $\rightarrow$ Java-Quellcode $\rightarrow$ Java-Bytecode) notwendig werden. Daher ist eine nicht so hohe Flexibilität wie bei dem dynamischen Planungsansatz gegeben. Kommt es auf die schnelle und flexible Definition von neuen Aktionen bzw. Gesten an, so ist der dynamische Planungsansatz vorzuziehen. Steht allerdings die schnelle Reaktion (z.B. bei Einsatz als Desktop-Präsentationsagent) im Vordergrund, so können durch den Compilierungsansatz Performanzvorteile erreicht werden. Je größer der Suchraum ist, desto größer ist der Vorteil des Compilierungsansatzes.

Ein weiteres Problem bei der Berechnung von Gestenübergängen ist, dass innerhalb eines Persona-Skripts bedingte und unbedingte Sprünge vorkommen dürfen. Als Lösung bietet sich das Einfügen von Gestenübergängen in das Präsentationsskript durch das System innerhalb der angesprungenen Programmstücke (Unterprogramme) an. Beim Aufruf wird allerdings ein konsistenter Zustand vorausgesetzt (z.B. dadurch dass sich der Agent an diesen Stellen in der Grundhaltung befindet).

4.5 Generierung der Sprechgeste

Die Verwendung von natürlicher gesprochener Sprache ist für ein lebendiges Erscheinungsbild des Präsentationsagenten unverzichtbar. Nur in bestimmten Fällen sollte auf die Verwendung von Sprachausgabe verzichtet und die Information durch andere Mittel übermittelt werden. Beispiele für solche Anwendungen, bei denen die Sprachinformationen z.B. über Sprechblasen (Abbildung 4.16) wiedergegeben werden sollten, sind beispielsweise:

- Umgebungen mit hohem Publikumsverkehr (z.B. Großraumbüros)
- schwerhörige Benutzer
- keine Audioausgabemöglichkeit

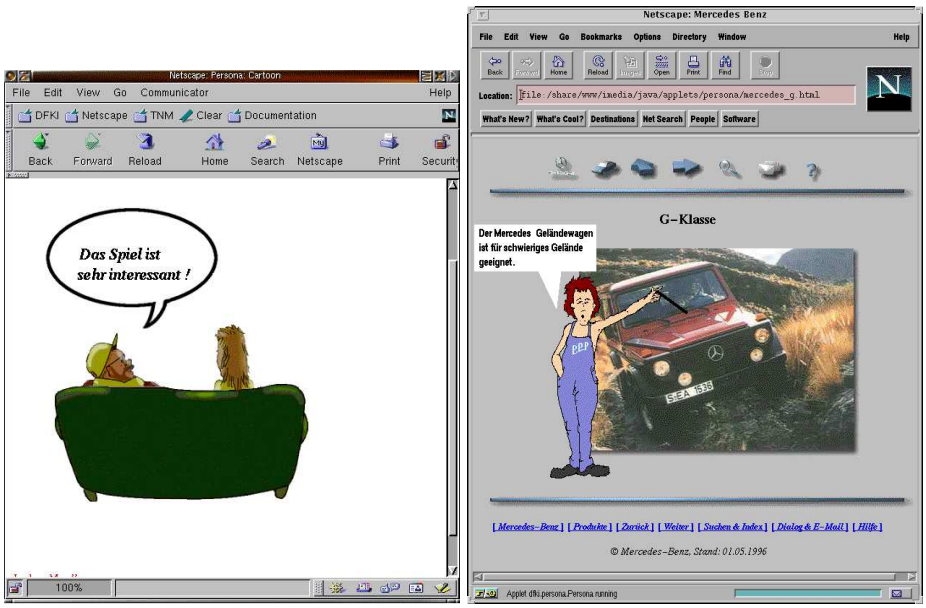


Abbildung 4.16: Sprechblasen als Ersatz für Sprachausgabe

Eine weitere Einsatzmöglichkeit für Sprechblasen ist die nochmalige Verdeutlichung des gesprochenen Textes. Um eine Information besonders deutlich wiederzugeben, kann es sinnvoll sein, zusätzlich zur Sprachausgabe den Text in einer Sprechblase zu visualisieren.

Ein Problem bei der Generierung von Sprechgesten ist die Abstimmung zwischen der Audioausgabe und der gezeigten Mimik des Präsentationsagenten. Wird das ganze Gesicht mittels eines 3D-Modells visualisiert, so ist es möglich, die Gesichtszüge durch Manipulation des Polygonnetzes der synthetisierten Sprache anzupassen [Morishima et al., 1997].

Da bei PPP-Persona aus Geschwindigkeitsgründen, wegen Hardwarebegrenzungen und des Einsatzes innerhalb von WWW-Browsern auf die Verwendung

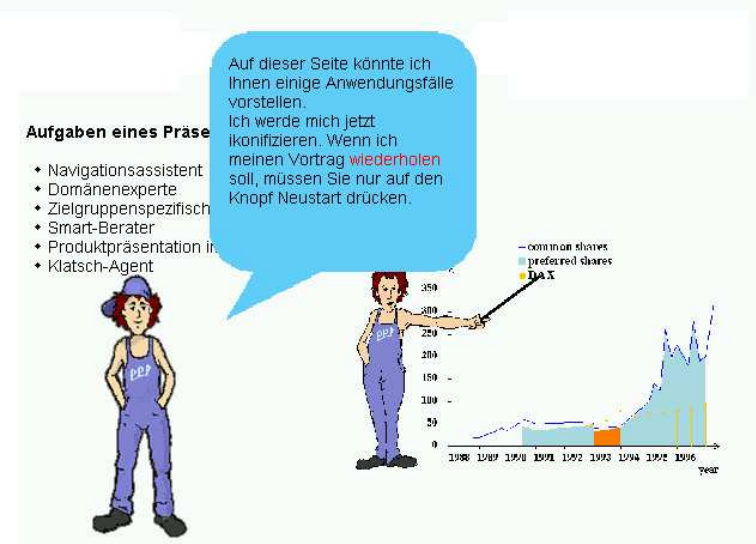


Abbildung 4.17: Persona zeigt eine Sprechblase, die formatierten Text und Links enthält.

eines 3D-modellierten (z.B. in VRML bzw. Java-3D) Charakters verzichtet wurde, müssen zur Visualisierung von Sprechgesten Bitmapfolgen genutzt werden.

Bei der Visualisierung der einzelnen Gesten durch Bitmap-Folgen muss man die zu einer Audiodatei passenden Mundbewegungen und Mimiken aus einem begrenzten Vorrat von möglichen Bitmaps zusammenstellen. Die unterschiedlichen Längen lassen sich nur durch Wiederholungen der einzelnen Bitmapsequenzen erzielen.

Bei der Verwendung von Sprachsynthesizern gibt es zwei grundsätzliche Vorgehensweisen. Entweder der Sprachsynthesizer gibt unmittelbar die synthetisierte Sprache aus oder der Sprachsynthesizer generiert zunächst eine Audiodatei, die später ausgegeben wird. Versucht man die direkte Ausgabe bei der Generierung, steht man vor dem Problem, dass die Laufzeit des verwendeten Sprachsynthesizers nicht vorhersagbar ist und dadurch die Gefahr besteht, dass die Synchronisation zwischen Audioausgabe und gezeigter Mimik verloren geht.

Beim Einsatz in WWW-Applikationen ist es besser, nur Audiodateien auf Client-Seite zu verarbeiten, da man so den Sprachsynthesizer auf Server-Seite belassen kann. Daher wird im Folgenden davon ausgegangen, dass der Sprachsynthesizer Audiodateien erzeugt, die erst in einem zweiten Schritt ausgegeben werden.

Kann der Agent Audiodateien für die Sprachausgabe einsetzen, so ist es neben der Verwendung eines Sprachsynthesizers auch möglich, natürlich gesprochene Sprache in den Präsentationen zu verwenden. Innerhalb des Persona-Generators ist das dadurch realisiert, dass an den Stellen, an denen Textstrings für die Sprachsynthese angegeben werden, auch URLs von Audiodateien stehen können. Falls der Generator in der Eingabe eine solche URL vorfindet, wird der Syntheseschritt bei der Generierung übersprungen, die Mimikgenerierung erhält die vorgegebene Audiodatei als Eingabe.

Damit der Präsentationsagent dynamisch neue Daten verarbeiten kann, ist es notwendig, dass er Sprache synthetisieren kann. Es ist in den meisten Fällen nicht möglich, abgespeicherte Audiodateien zu verwenden. In bestimmten Fällen kann zwar aus einer Menge vorgefertigter Sätze bzw. Teilsätze die aktuelle Ausgabe generiert werden, allerdings ist im allgemeinen Fall eine echte Sprachsynthese unverzichtbar.

Die Sprachgenerierung verläuft in mehreren Schritten:

- Generierung des zu sprechenden Textes
- Annotierung des Textes mit Betonungsinformationen für den Sprachsynthesizer
- Generierung einer Audiodatei aus dem annotierten Text mittels Sprachsynthesizers
- Analyse der generierten Audiodatei zu Synchronisation mit den Mundbewegungen
- Generierung der Animationssequenz
- Abspielen der Sequenz in Player

Zur Textgenerierung wird ein Template-basierter Ansatz verfolgt. Während des Planungsvorganges werden die für einen Satz notwendigen Bestandteile zu einem vollständigen Satz bzw. mehreren Sätzen zusammengefügt. Neben den eigentlichen Wörtern, Satzzeichen und vorgefertigten Teilsätzen werden auch Anweisungen für den Sprachsynthesizer berücksichtigt. Je nach Fähigkeit des verwendeten Sprachsynthesizers können so Emotionen und Betonung wichtiger Elemente in den Satz eingefügt werden, was zum Erreichen einer natürlicheren Sprache wichtig ist. Das Ausfüllen des Templates kann entweder innerhalb des Persona-Generators oder auch innerhalb des Präsentationssystems (z.B. im Präsentationsplaner von AIA bzw. PPP) erfolgen. Innerhalb des Persona-Generators ist es möglich, in Abhängigkeit der eingestellten Sprache zu einer Geste den passenden Text zu erzeugen, z.B. kann die Geste „Greet“ bei deutschem Sprachsynthesizer von dem Text „Guten Tag !“ , bei englischem Sprachsynthesizer von „Hello !“ und bei französischem Sprachsynthesizer von „Bonjour !“ begleitet werden.

Bevor der Eingabestring dem Sprachsynthesizer übergeben wird, erfolgt eine einfache Analyse der Satzart über die Interpunktionszeichen. Entsprechend der Satzart kann die Mimik gewählt werden. Beispielsweise wird bei einem Komma eine Pause in den Satz eingefügt, bei einem Fragezeichen ein fragendes Gesicht verwendet und ein Ausrufezeichen von einem aufforderndem Gesicht quittiert (Abbildung 4.18).

Der erzeugte Textstring wird nun dem Sprachsynthesizer übergeben. Dabei können unterschiedliche Sprachsynthesizer über ein einheitliches Interface angesprochen werden, so dass von jeweiligen Unterschieden abstrahiert werden kann. Entsprechend der aktuellen Sprache (momentan wird Deutsch, Englisch

Abbildung 4.18: Abbildung von Satzzeichen auf mimische Äußerungen

und Französisch unterstützt) und entsprechend dem Geschlecht des Präsentationsagenten wird ein passender Sprachsynthesizer ausgewählt, der dann eine Audiodatei aus der Texteingabe erzeugt. Das System verwendet für die englische Sprachausgabe den TrueTalk-Sprachsynthesizer [Entropic Research Laboratory, 1996] und für Deutsch und Französisch den Sprachsynthesizer des MBROLA-Projekts [The MBROLA Project, 1998].

Oft kommen in einer Präsentation gleiche Sätze vor, z.B. Begrüßung, Danke, Bitte usw. Daher ist es besser, synthetisierte Sätze zu speichern und, bevor ein Satz neu synthetistiert wird, zunächst zu überprüfen, ob es nicht schon eine passende Audiodatei für diesen Satz gibt, so dass man sich den relativ aufwendigen Aufruf des Sprachsynthesizers sparen kann. Zu diesem Zweck unterhält der Persona-Generator einen Cache, in dem er erzeugte Sätze ablegt, um auf die Audiodateien wieder zugreifen zu können.

Der nächste Generierungsschritt beschäftigt sich mit der Synchronisation der Audiodatei mit den Mundbewegungen des animierten Präsentators. Dabei bieten sich verschiedene Vorgehensweisen an. Im einfachsten Fall kann man versuchen, auf Grund der Textlänge über verschiedene Heuristiken auf die Länge der erzeugten Audiodatei zu schließen, um dann die Länge der Mundbewegungssequenz dieser Länge anzupassen. Da es im Allgemeinen aber schwierig ist, solche Heuristiken relativ sicher anzugeben, weil dies stark von dem jeweiligen Sprachsynthesizer abhängt (unterschiedlich lange Pausen bei Satzzeichen, unterschiedliche Sprechgeschwindigkeit), muss man doch andere Methoden verwenden. Eine Möglichkeit ist die Analyse der erzeugten Audiodatei. Will man die für die jeweiligen Laute passenden Mundewegungen finden und genau passende Animationssequenzen generieren, so muss man in den Audiosequenzen die entsprechenden Muster erkennen und aus diesen Informationen die Animationen auswählen. Allerdings ist dieser Vorgang zeitaufwendig und oft auch bei einem Präsentator, der durch eine ganze Figur dargestellt ist und nicht nur durch den Kopf (so dass der Betrachter nicht nur die Lippenbewegungen anfi-

xiert), nicht notwendig. Auch Spielfilme im Fernsehen sind nicht immer perfekt synchronisiert, so dass die Betrachter oft schon daran gewöhnt sind.

Für PPP-Persona wurde eine andere Möglichkeit gewählt. Die erzeugte Audiodatei wird nach Stellen durchsucht, an denen Stille herrscht. In Abbildung 4.19 wird eine Darstellung des Audiosignals gezeigt, die dem Satz „Hello. My name is PPP Persona." entspricht. Durch eine Schwellwertanalyse wurden einzelne Intervalle gefunden, in denen keine Audioausgabe vorkommt. Dabei wurden schon Rauscheffekte ausgefiltert, d.h. die Intervalle, die als Stille empfunden werden, müssen eine bestimmte minimale Länge haben.

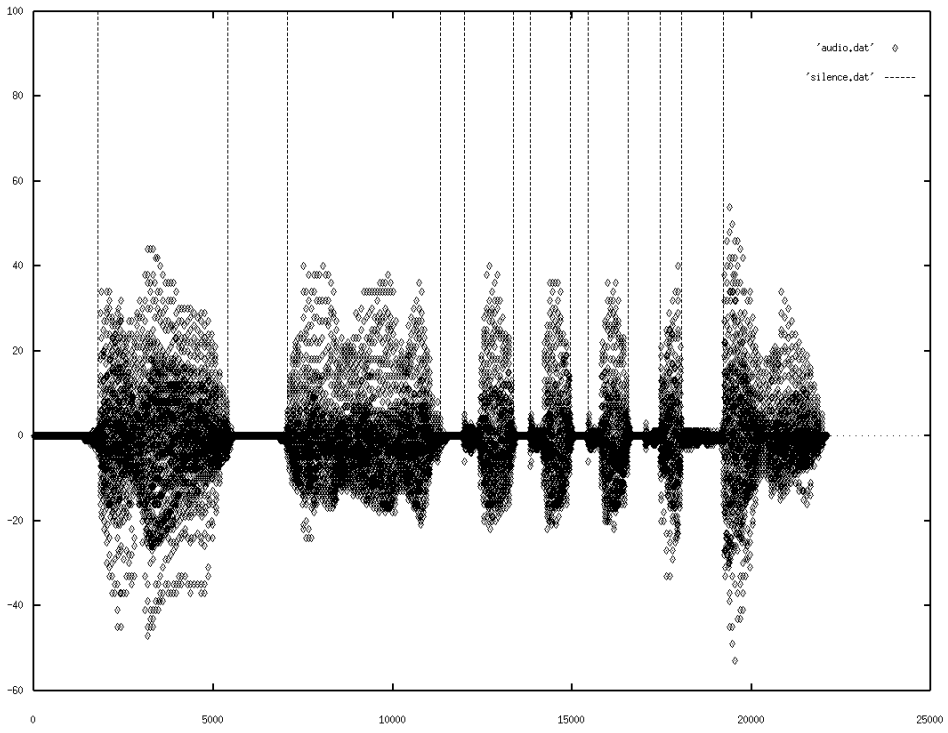


Abbildung 4.19: Intervalle ohne Audioausgabe in einer synthetisierten Audiodatei.

In einem nächsten Schritt fasst man Intervalle zusammen, die einer Audioausgabe entsprechen, aber durch kurze Pausen ohne Ausgabe unterbrochen werden, und die alleine zu kurz sind, um durch eine einzelne Sprechgeste visualisiert zu werden.

Zur Darstellung einer Sprechgeste von vorgeschriebener Länge hat das Persona-System in einer bestimmten Situation (z.B. während Persona mit dem Gesicht zum Betrachter steht) zwei verschiedene primitive Sprechgesten zur Verfügung, aus denen die Sprechsequenz zusammengestetzt wird. Dabei handelt es sich um eine kurze und eine längere Sprechgeste. Durch die Verwendung zweier unterschiedlich langer Sprechgesten kann die resultierende Sprechgeste besser an die Audiodaten angepasst werden. Zwischen den einzelnen primitiven Sprechgesten sind Sprechpausen möglich.

Aus der Intervallsequenz wird nun eine gültige Anweisung für den Persona-Player generiert. Das entsprechende Kommando „talk “ des Persona-Players (Abschnitt 5) startet die Ausgabe des Audioclips und zeigt die dazu passende Sequenz aus Mundbewegungen und Pausen. Die allgemeine Form des „talk “ Kommandos ist:

$$talk\ g_1\ g_2\ clip\ x\ y\ nr\ b_1\ a_1\ b_2\ a_2\ \ldots\ b_{nr-1}\ a_{nr}$$

Dabei ist:

g_1, g_2: Name der langen bzw. der kurzen Sprechgeste

clip: Name des Audio-Clips

x, y: Koordinaten für Zeigegeste

nr, b_i, a_i: Die Liste der Länge nr gibt die Verteilung der einzelnen Sprechgesten und Pausen an. Der Parameter b_i beschreibt die Art der Geste:

- $b_i = 0$: Das aktuelle Intervall ist ein Pausenintervall der Länge $b_{i+1} ms$.
- $b_i = 1$: Das aktuelle Intervall wird durch b_{i+1}-malige Wiederholung der Geste g_1 ausgefüllt.
- $b_i = 2$: Das aktuelle Intervall wird durch b_{i+1}-malige Wiederholung der Geste g_2 ausgefüllt.

Der Persona-Generator erzeugt aus synthetisierten Audiodateien und aus Audiodateien, die natürlich gesprochene Sprache enthalten, die gleichen Befehle für den Persona-Player. Auf Client-Seite, d.h. innerhalb des Web-Browsers, gibt es daher keine Unterschiede mehr in der Handhabung.

4.6 Das Teilmodul DocManager

Da der Persona-Generator eingesetzt werden soll zur Erweiterung von XML bzw. HTML um virtuelle Präsentatoren, wird eine Komponente benötigt, die in der Lage ist, XML- oder HTML-Dokumente einzulesen und die in diese Dokumente eingefügten Persona-Tags zu evaluieren (Abschnitt 6.2). Zu diesem Zweck wurde das Modul DocManager entwickelt. Dieses Modul hat die Fähigkeit, ein in XML-/HTML-verfasstes Dokument einzulesen und auf Teilelemente des Dokuments Prozessoren anzuwenden, die auf diesen Elementen die entsprechenden Transformationen vornehmen können.

Der DocManager liest ein XML-/HTML-Dokument ein. In der aktuellen Version wird der in Java implementierte XML-Parser „XML for Java“ von IBM eingesetzt [IBM, 1999]. Damit der XML-Parser auch HTML-Dokumente einlesen kann, wird für HTML-Dokumente eine Vorverarbeitung durchgeführt (über einen HTML-Parser [Bedersdorfer, in Vorbereitung]). Der XML-Parser liefert eine Repräsentation des Dokuments im Document Object Model (DOM, [Wood und Hors, 1999]) zurück. Da der DocManager nur auf der vom W3-Konsortium definierten Schnittstelle zu DOM arbeitet, ist es möglich, auch den XML-Parser zu wechseln, ohne dass Elemente des DocManagers beeinträchtigt sind.

Der DocManager wendet nun alle definierten Prozessoren auf das Dokument an, bis keine Änderungen mehr auftreten. Die jeweiligen Prozessoren manipulieren einzelne Knoten in der Baumstruktur des Dokuments. Die Prozessoren werden selbst innerhalb des Dokuments mit Hilfe eines speziellen Tags definiert, der unmittelbar von DocManager ausgewertet wird. Der DocManager wird so vollständig über das Startdokument spezifiziert.

Durch die Prozessoren können neue Knoten eingefügt, Knoten gelöscht oder auch nur verändert werden (Abbildung 4.20).

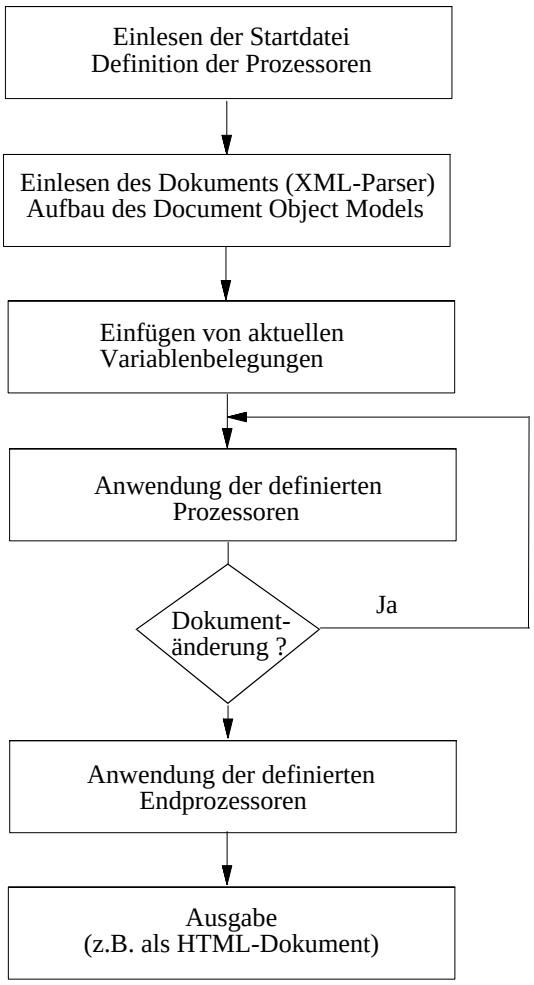


Abbildung 4.20: Verarbeitung eines Dokuments innerhalb des Modules DocManager

Der DocManager stellt Methoden zur Verfügung, die Elemente des Dokumentenbaums auslesen und modifizieren können. Solche Methoden umfassen das Auslesen von Attributen ebenso wie die Suche nach allen Knoten mit einem bestimmten Namen.

Nicht bei allen Problemstellungen ist die wiederholte Anwendung der Prozessoren notwendig. Beispielsweise ist bei normaler Anwendung des Persona-Generators keine Wiederholung nötig, so dass hierbei auf diesen Test aus Effi-

zienzgründen auch verzichtet werden kann.

Wurden alle Prozessoren auf das Dokument angewendet, so können vor der Ausgabe noch eventuell definierte Endprozessoren auf das Dokument für Aufräumarbeiten angewendet werden.

Das Verarbeitungsmodell des DocManagers ist universell auch bei anderen Problemen einsetzbar. Der Vorteil bei dieser Vorgehensweise ist die starke Kapselung der einzelnen Teilmodule (Prozessoren). Die Kommunikation verläuft nur über das DOM. Ein weiterer Pluspunkt ist die gute Repräsentation jedes Verarbeitungsschrittes. Zu jedem Zeitpunkt kann der aktuelle, innerhalb des DOM abgelegte Systemzustand ausgegeben werden, indem ein spezieller Prozessor, der das Dokument in Klarschrift ausgibt, definiert wird.

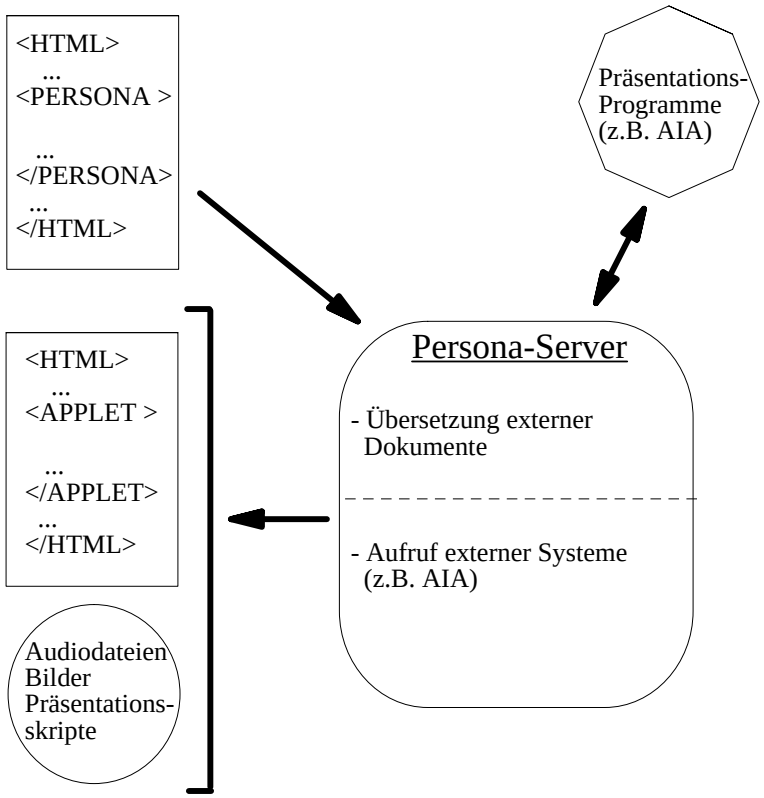


Abbildung 4.21: Der DocManager innerhalb des Persona-Generators

In Systemen wie AIA (Abschnitt 7.2) kann der DocManager dazu dienen, die einzelen Module des Systems zu kapseln. Jedes Teilmodul ist unabhängig von den anderen Verarbeitungsmodulen und kann in der Startdatei, in der die Prozessoren definiert werden, ein- und ausgeschaltet werden, ohne dass am Programmcode Änderungen durchgeführt werden müssen.

Der DocManager mit PersonaGenerator-Prozessoren ist in zwei Versionen einsetzbar. Neben dem Aufruf über die Kommandozeile - dabei ermöglicht diese Konfiguration die Offline-Verarbeitung von mit Persona-Tags versehenen HTML/XML-Seiten - steht auch eine Servlet-Version zur Verfügung, die es online erlaubt, solche Dokumente zu bearbeiten. Entweder werden die Dokumente

innerhalb des DocManager-Verarbeitungszyklus generiert (wie in AIA) oder der DocManager bzw. der Persona-Generator erhält ein Dokument von außen als Eingabe. Damit gelingt der Aufbau eines Persona-Servers, der die Möglichkeiten des Persona-Generators im Internet bzw. Intranet verfügbar machen kann.

4.7 Dialogverwaltung

Um den bisherigen Präsentationsverlauf in die weitere Präsentationsplanung einzubeziehen, ist es notwendig, eine Repräsentation des Dialogs aufzubauen. Diese Struktur ist notwendig, da der Präsentationsagent jederzeit an das Präsentationssystem eine Präsentationsanfrage richten kann. Das Präsentationssystem bzw. der Persona-Generator muss dann neue Daten oder auch ein neues Persona-Skript entsprechend dem Dialogzustand generieren. Zu diesem Zweck wird ein gerichteter Graph $G = (V, E)$ verwendet, der im Laufe eines Benutzerdialogs dynamisch aufgebaut wird. Dabei werden durch die Menge der Knoten V die Dialogzustände repräsentiert. Die Menge E repräsentiert den Übergang zwischen Dialogzuständen. Ein Dialogzustand $Z \in V$ ist ein Tupel $(N, U, S, \delta, \theta)$. Dabei ist

N: ein eindeutiger Name,

U: eine Benutzerkennung (Benutzername und Zeitmarke),

S: eine Menge mit Zustandsvariablen,

δ: eine Übergangsfunktion, die entsprechend der aktuellen Benutzereingabe den nächsten Dialogknoten berechnet, und

θ: eine Funktion, die gemäß der aktuellen Benutzereingabe eine passende Ausgabe generiert.

Jeder Knoten im Dialoggraph kann über seinen eindeutigen Namen referenziert werden. Sei $Z_i = (N_i, U_i, S_i, \delta_i, \theta_j) \in V$ der aktuelle Knoten, in dem sich der Dialog mit dem Benutzer befindet. Um eine neue Ausgabe für den Benutzer (z.B. eine neue HTML-Seite für WWW-Präsentationen oder ein neues Skript für die Persona-Engine) zu bestimmen, muss zunächst der Nachfolgeknoten im Dialoggraph durch Anwendung der Funktion δ_i auf die aktuelle Benutzereingabe ermittelt werden. Falls notwendig kann durch einen Seiteneffekt der Funktion δ_i der Dialoggraph vergrößert werden. Die Funktion θ_j des ermittelten Nachfolgeknotens $Z_j = (N_j, U_j, S_j, \delta_j, \theta_j) \in V$ berechnet, angewendet auf die aktuelle Benutzereingabe, die Ausgabe.

Die Menge der Zustandsvariablen S wird benötigt, Informationen über den Benutzer (z.B. die verwendete Sprache, die Bildschirmgröße usw.) zu speichern. Die Funktionen δ und θ können auch konstante Funktionen sein. Ist δ eine konstante Funktion, dann gibt es zu diesem Knoten nur einen Nachfolgerknoten. Falls θ eine konstante Funktion ist, so ist die Ausgabe für alle Benutzereingaben gleich (wichtig z.B. zum Puffern von schon einmal berechneten Daten). Die Benutzerkennung U braucht man, um die Knoten der einzelnen Benutzer

voneinander abzugrenzen, da die aktuelle Implementation die gleichzeitige Verwaltung mehrerer Benutzer erlaubt. Die Zeitmarke wird zur Speicherverwaltung benutzt. Ein Knoten, der ein gewisses Alter überschritten hat (d.h. man geht davon aus, dieser Knoten wird nicht mehr benötigt), wird von einer Systemroutine gelöscht.

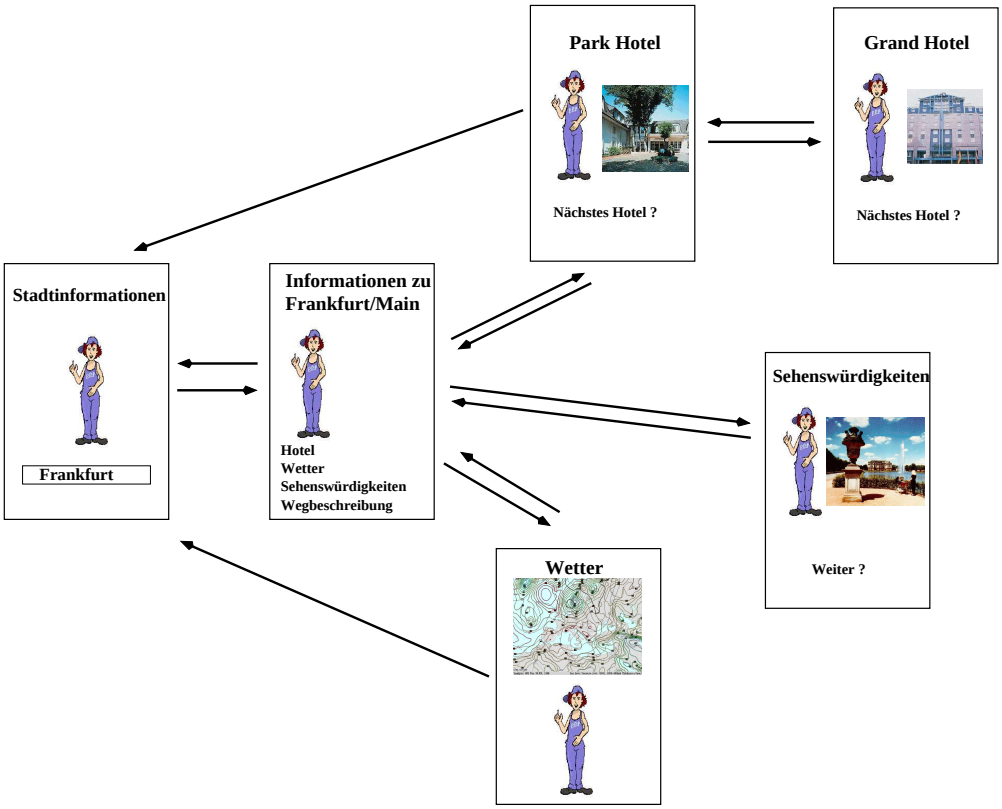


Abbildung 4.22: Beispiel für Dialoggraph und die entsprechende Ausgabe

Ein Beispiel ist in Abbildung 4.22 zu sehen. Es ist der Tourismusdomäne entnommen. Das Präsentationssystem soll touristische Informationen zu einer Stadt im WWW finden und dem Benutzer passend aufbereitet darbieten. Im ersten Dialogknoten ist das Eingabefenster zu sehen, bei dem der Benutzer die gesuchte Stadt angeben kann. Bei der Verarbeitung dieser Information sucht das System verschiedene Informationen (hier: Hotels, Wegbeschreibung zum Hotel, Sehenswürdigkeiten, Wetter) und bietet dem Benutzer die gefundenen Informationen in einer Übersicht an. Wählt der Benutzer die Hotelpräsentation aus, so wird ihm ein Hotel gezeigt (wird entsprechend einer Heuristik ausgewählt). Da es nicht sinnvoll ist, Informationen zu allen Hotels der Stadt zu suchen, wird schrittweise vorgegangen. Erst wenn der Benutzer ein anderes Hotel vom System anfordert, werden die Informationen im WWW gesucht und schließlich das Hotel präsentiert. In analoger Form werden die Sehenswürdigkeiten zu dieser Stadt gesucht. An mehreren Stellen des Dialogs sind Sprünge sowohl nach vorne als auch rückwärts möglich. Zu jedem Zeitpunkt muss das System wissen, welche Informationen schon vorhanden sind und welche Informationen noch gesucht

werden müssen.

4.8 Verwaltung mehrerer Agenten

Um Präsentationen mit mehreren Präsentatoren umsetzen zu können, müssen auf verschiedenen Ebenen des Präsentationssystems Anpassungen vorgenommen werden. Während die hierfür notwendigen Anpassungen der Abspielkomponente in Abschnitt 5.5 dargestellt wurden, sollen nun die eigentlichen Generierungskomponenten beschrieben werden.

Der Persona-Generator unterhält eine Tabelle aller momentan aktiven Präsentationsagenten. Zu jedem Agenten, der innerhalb der Abspielkomponente verwendet wird, gibt es eine Repräsentation innerhalb des Persona-Generators. Innerhalb dieser Repräsentation werden Informationen zum jeweiligen Zustand des Agenten abgelegt (z.B. visuelles Erscheinungsbild, aktuelle Bewegungsphase).

Der Eingabepuffer des Generators, der vom Präsentationssystem (z.B. der Präsentationsplaner PrePlan) gefüllt wird, wird vom Generator ausgelesen und die Befehle werden in entsprechende Aktionen umgesetzt. Dabei wird jeder Befehl der entspechenden Instanz des Präsentationsagenten zugeordnet.

Wie in Abschnitt 4.3 beschrieben wurde, wird innerhalb des PERSONA-Tags der Persona-Eingabesprache ein Präsentationsagent über den PERSONASCRIPT-Tag definiert. Ein PERSONASCRIPT-Tag führt zu einer weiteren Instanz eines Präsentationsagenten. Die innerhalb dieses Tags aufgeführten Befehle steuern den entsprechenden Agenten.

Um einen Dialog zwischen mehreren Präsentatoren zu generieren und Interaktion zwischen den einzelnen Gesprächspartnern zu simulieren, ist es sinnvoll, einzelne Dialogbeiträge der Agenten zeitlich zu koordinieren. Um dies zu erreichen, wird die Eingabesyntax des Persona-Generators erweitert. Die Grundstruktur bleibt erhalten. Mittels des PERSONA-Tags wird die Persona-Umgebung definiert und über den PERSONASCRIPT-Tag werden die einzelnen Agenten festgelegt:

```
<PERSONA width="640" height="350" color="white">
      <PERSONASCRIPT name="Wichtel" bitmap="cartoon">
        <INCLUDE url="file:/home/jmueller/persona/primitives.xml"/>
        <INCLUDE url="file:/home/jmueller/persona/complex.xml"/>
      </PERSONASCRIPT>
      <PERSONASCRIPT name="Cyberella" bitmap="cyberella">
        <INCLUDE url="file:/home/jmueller/persona/primitives.xml"/>
        <INCLUDE url="file:/home/jmueller/persona/complex.xml"/>
      </PERSONASCRIPT>
</PERSONA>
```

In diesem Beispiel soll ein Dialog zwischen den beiden Agenten „Wichtel“ und „Cyberella“ modelliert werden. Beide werden zur Laufzeit in einem Applet der Größe 640×350 Pixel vor weißem Hintergrund dargestellt. Bei der Definition werden die Darstellungsparameter (hier die verwendete Bitmap) angegeben. Im Anschluss werden die Gestendefinitionen für die beiden Präsentatoren geladen. In diesem Beispiel wird davon ausgegangen, dass beide über das gleiche Gestenrepertoire verfügen. Um den eigentlichen Dialog zu modellieren, stellt der

Persona-Generator ein spezielles Konstrukt zur Verfügung. Die Dialogbeiträge der einzelnen Dialogpartner werden mit dem Tag DIALOGPART geklammert:

```
<DIALOGPART agent="Wichtel">
 <greet/>
 <speak text="Guten Tag."/>
</DIALOGPART>
```

Werden diese Dialogelemente aneinandergefügt, so ergibt sich ein koordiniertes Gespräch zwischen den Beteiligten. Der Persona-Generator sorgt durch automatisches Einfügen von Synchronisationspunkten (Abschnitt 5.5) für die Entstehung eines Wechselgesprächs. In der Standardeinstellung für den Tag DIALOGPART werden die Aktionen so angeordnet, dass alle Agenten jeweils auf die Beendigung des Dialogbeitrages warten, bevor die nächste spezifizierte Aktion ausgeführt wird. Ist das nicht gewünscht, kann über das Attribut „synchronizewith“ eine Liste mit Agenten angegeben werden, die koordiniert werden sollen. Innerhalb des DIALOGPART-Tags können beliebig viele Aktionen stehen, die von der entsprechenden Figur ausgeführt werden.

Die Dialogverwaltung wurde so implementiert, dass durch einen Vorverarbeitungsschritt alle DIALOGPART-Tags bearbeitet und aus dem XML-Dokument entfernt werden. Die Aktionen werden zusammen mit den Synchronisationsanweisungen in die PERSONASCRIPT-Tags eingefügt. Die weitere Verarbeitung verläuft wie in Abschnitt 4.2 beschrieben.

Durch den Einsatz der DIALOGPART-Konstrukte wird es für den Präsentationsplaner vereinfacht, eine Präsentation mit mehreren Präsentatoren zu erstellen. Bei der Generierung müssen nicht mehrere Ausgabepuffer für die einzelnen Agenten verwendet werden, sondern der Planer kann auf einem einzigen Ausgabepuffer arbeiten. Die Verteilung der einzelnen Aktionen auf die jeweiligen Präsentatoren wird durch den Persona-Generator vorgenommen.

Auf die Planung von Präsentationen mit mehreren Beteiligten wird in Abschnitt 7.4.2 genauer eingegangen.

Die durch den Generierungsvorgang erzeugten Einzelskripte werden an die jeweiligen Persona-Instanzen auf Client-Seite übermittelt und von diesen durch die Abspielkomponente ausgeführt.

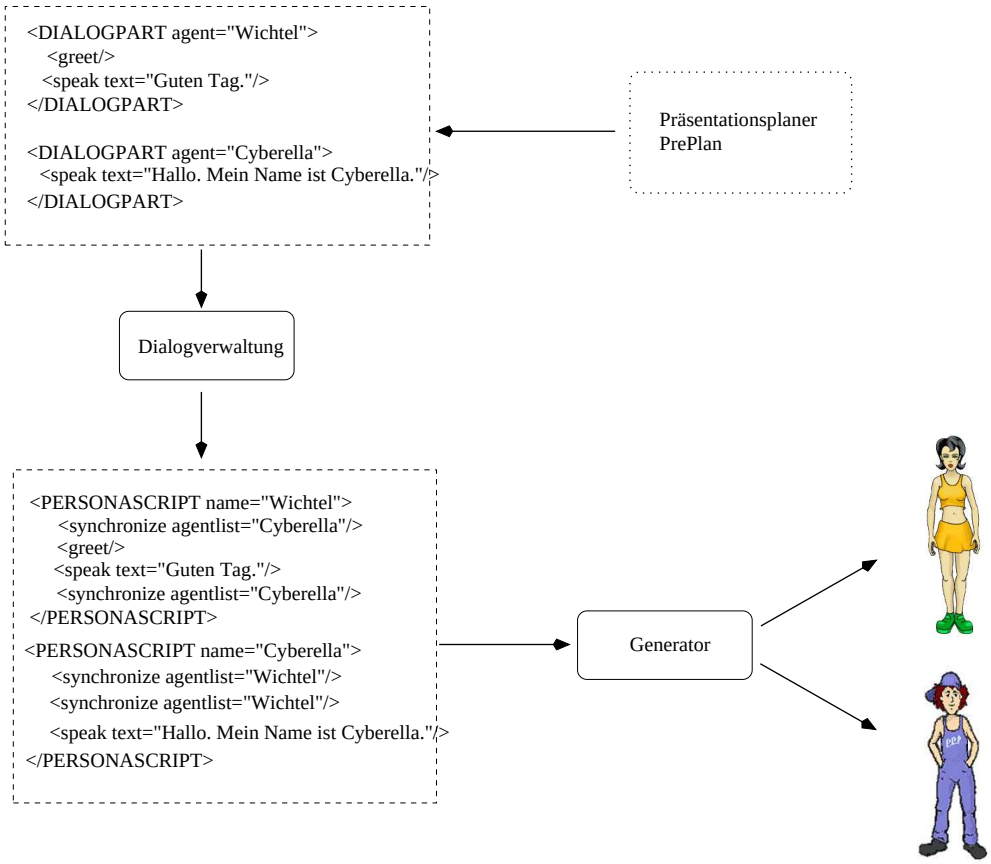


Abbildung 4.23: Generierung von Persona-Skripten für mehrere Agenten

Kapitel 5

Die Persona-Abspielkomponente

Das Abspielmodul des Persona-Systems dient dazu, eine Kommandosequenz, die vom Generierungssystem ausgegeben wurde, abzuspielen. Gleichzeitig realisiert das Modul auch Benutzerinteraktionen und weitere zur Präsentationszeit notwendige Operationen.

5.1 Die Persona-Engine

Die Persona-Engine ist die zentrale Komponente des Abspielmoduls. Sie steuert andere Teilkomponenten an und verarbeitet Eingaben.

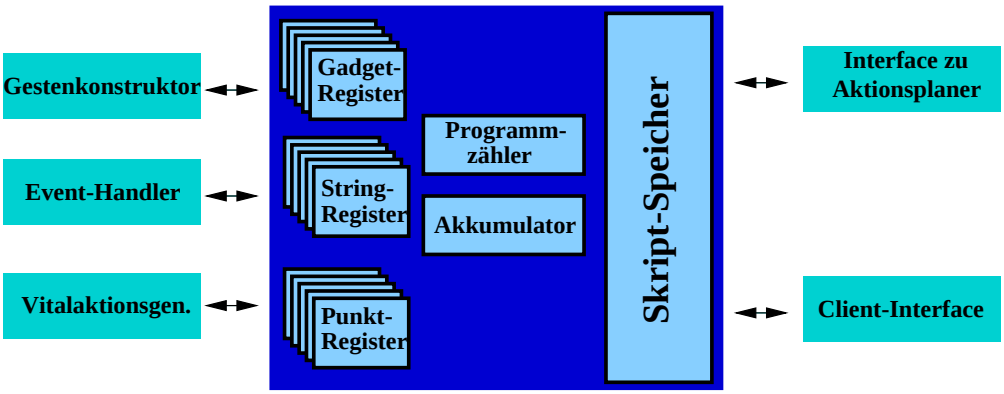


Abbildung 5.1: Architektur der Persona-Engine

Ein Persona-Skript, das vom Persona-Generator erstellt wurde, wird über ein Interface in den Skript-Speicher übertragen. Das alte Skript kann ersetzt werden oder das neue Skript wird an einer Adresse eingefügt. Der Einlesevorgang erfolgt wahlweise als separater Thread, so dass Persona dabei ein altes Skript abarbeiten kann. Auf diese Art kann die Zeit, die der Benutzer bis zum Start einer Präsentation warten muss, verkürzt werden. Beim Einlesen wird jeder Befehl mit seinen Parametern in ein Befehlsobjekt übersetzt, in dem alle Referenzen zu anderen Objekten (z.B. auf die Gestenobjekte) aufgelöst sind. Alle Daten, die zur Ausführung des Skripts benötigt werden, werden geladen.

Nach dem Ladevorgang wird der Programmzähler auf den Startwert gesetzt (0 oder beim Einfügen der Einfügeindex).

Dann beginnt die Persona-Engine alle Befehlsobjekte im Skript-Speicher abzuarbeiten. Die Adresse des aktuellen Befehls wird dem Programmzähler entnommen. Eine kleine Auswahl von Persona-Befehlen ist aus Tabelle 5.1 ersichtlich, die genaue Definition der momentan definierten Befehle kann Anhang A entnommen werden.

Befehl	Beschreibung
gesture laugh 0 0 null	Persona lacht
gesture speakstickleftpoint3 300 200 a3939x33.au	Persona zeigt mit dem Zeigestock auf Position (300,200) und benutzt für die Sprechgeste die Audio-Datei a3939x33.au
goto 20	Persona-Engine führt einen unbedingten Sprung zur Zeile 20 aus.
wait 3000 true	Persona-Engine wartet 3000 ms. Dabei werden Vitalaktionen ausgeführt.
loadurl http://www.dfki.de htmlframe	In dem Frame htmlframe wird die angegebene HTML-Seite angezeigt.

Tabelle 5.1: Auswahl von Befehlen, die die Persona-Abspielkomponente unmittelbar ausführen kann.

Jeder Befehl ist mit Hilfe einer eigenen Klasse (Abbildung 5.2) implementiert. Auf diese Weise können bei Bedarf neue Befehle integriert werden. Für einen neuen Befehl ist einfach eine neue Klasse, die Unterklasse der Befehlsoberklasse ist, zu definieren und die entsprechenden Methoden der neuen Klasse zu spezifizieren.

```
public PersonaCommand {
  public PersonaCommand() {}

  public void initialize(StreamTokenizer st,
                         PersonaState state) {}

  public void eval(PersonaState state) {}
}
```

Abbildung 5.2: Die Oberklasse PersonaCommand (Auszug)

Muss die Persona-Engine für eine spezielle Anwendung um Befehle erweitert werden, so genügt es, in der entsprechenden Klasse die Methoden „initialize“ und „eval“ zu definieren. Die Methode „initialize“ wird aufgerufen, wenn der Player ein Skript evaluiert. Das Kommando kann über den übergebenen Eingabestrom seine Parameter bestimmen und den Befehl zusammen mit den benötigten Daten initialisieren. Die eigentliche Aktion wird zur Laufzeit über den Aufruf von „eval“ ausgeführt. Über den übergebenen Parameter hat sie die Möglichkeit, auf Strukturen der Persona-Eingine zuzugreifen.

Nach Abarbeiten des Befehls wird der Programmzähler entsprechend der Befehlsdefinition modifiziert. Erreicht die Persona-Engine eine undefinierte Adresse (z.B. das Programmende ist erreicht), hält die Maschine an.

Der Akkumulator enthält jeweils das aktuelle Befehlsobjekt. Die Ausführung eines Befehls bewirkt in den meisten Fällen eine Modifikation der Gadget-, String- oder Punktregister.

In den Gadget-Registern können komplexe zu visualisierende Datentypen abgelegt werden (s. Kapitel 5.2).

In den String- bzw. Punktregistern werden aktuelle Werte für die Gestengenerierung gespeichert. Beispielsweise werden die aktuellen Koordinaten für Zeigegesten in das Punktregister abgelegt. Ein Anwendungsbeispiel für das Stringregister ist der aktuelle Text für Gesten, die Sprechblasen verwenden.

Der Gestengenerator entnimmt die aktuellen Werte aus den Registern und verwendet sie zur Instantiierung einer konkreten Geste. Er zeigt die für eine Geste definierten Bilder entsprechend den für diese Geste spezifizierten Zeitdefinitionen. Um die Geschwindigkeit unabhängig von der eingesetzten Hardware zu halten, wird die für das Zeichnen eines Gestenframes benötigte Zeit gestoppt und mit der für dieses Frame definierten Zeit verrechnet. Nimmt die Ausgabe mehr Zeit als vorgesehen in Anspruch, so können zur Einhaltung eines gesetzten Zeitlimits Frames wegfallen.

Der Gestengenerator fügt die aus Einzelteilen bestehenden Gestenframes (Abschnitt 4.1) zusammen und gibt diese aus, wobei die Geste eventuell noch von einer Audioausgabe begleitet sein kann (z.B. Audioclips, Sprache).

Ist die Persona-Engine in einem Modus, der die Generierung von Vitalaktionen zulässt, und führt die Persona-Engine einen Wait-Befehl aus bzw. ist das Skript abgearbeitet, so erzeugt der Vitalaktionsgenerator (Kapitel 5.3) eine Vitalaktion, die als aktuelle Geste abgespielt wird.

Das Auslösen von Events, die beim Event-Handler (Kapitel 5.4) registriert sind, (z.B. Tastatureingaben, Mausbewegungen) bewirkt ebenfalls eine Änderung im Programmfluss. Die Persona-Engine muss die mit einem Event assoziierten Aktionen ausführen.

Andere Programme können über das Client-Interface mit der Persona-Engine kommunizieren. Diese Schnittstelle wird beispielsweise zum Aufbau von Systemen mit mehreren Persona-Instanzen (Kapitel 3.3) oder zur Kommunikation mit anderen Anwendungsprogrammen genutzt.

5.2 Gadget-Register

Im Laufe einer Präsentation werden unterschiedliche Datentypen (z.B. Grafiken, Texte) dem Betrachter gezeigt. Diese Objekte werden zur Visualisierung der eigentlichen Präsentationsdaten eingesetzt.

Im Gegensatz zu dem in [Noma und Badler, 1997] beschriebenen System, das auf dem Jack-System aufbaut, wird in dem folgenden Verfahren ein erweiterbares System vorgestellt, das verschiedene Datentypen visualisiert und für die Präsentation verfügbar macht.

Das System stellt zur Visualisierung Bildschirmflächen innerhalb des Präsentationssystems zur Verfügung. Die innerhalb dieser Flächen visualisierten Datenobjekte sind vom Präsentationsagenten aus manipulierbar und über das Persona-Skript steuerbar. Dabei werden die visualisierten Objekte dynamisch sichtbar oder unsichtbar gemacht. Außerdem müssen diese Objekte zur Laufzeit modifizierbar (z.B. bezüglich der Bildschirmposition) sein. Daneben kann es sinnvoll sein, dynamisch Informationen über die dargestellten Informationen zu berechnen und für die weitere Präsentation zu verwenden. Ein Beispiel dafür ist die Ermittlung von Koordinaten für Zeigegesten.

Zur Datenvisualisierung stellt der Persona-Player ein spezielles Gadget-Register zur Verfügung, in das mehrere dieser Präsentationsobjekte abgelegt werden können. Dieses Gadgetregister bzw. die einzelnen Präsentationsobjekte können zur Laufzeit über das Persona-Skript modifiziert werden. Bei der Darstellung werden die Grafikobjekte in aufsteigender Ordnung der Registeradressen gezeichnet, so dass eine eindeutige Zeichenreihenfolge besteht. Dadurch können auch sich gegenseitig überlappende Grafikobjekte verwendet werden. Im Folgenden werden einige der implementierten Objekte und die Erweiterungsmöglichkeiten vorgestellt.

5.2.1 Bild-Gadget

Ein Bild-Gadget stellt eine Bitmap-Grafik an einer bestimmten Stelle des Bildschirms dar. Das Bild-Gadget kann jedes von Java standardmäßig unterstützte Bildformat darstellen (z.Zt. GIF und JPEG). Obwohl es den einfachsten Gadgettyp repräsentiert, lässt es sich doch sehr universell einsetzen. Eine große Anzahl von Präsentationsmaterial liegt in Bildform vor oder lässt sich leicht in diese konvertieren. Beispielsweise lassen sich so Fotos von Sehenswürdigkeiten bei dem Aufbau virtueller Stadtführer nutzen. Stadtpläne liegen oft schon in digitalisierter Form vor und können ebenfalls integriert werden. Ein großer Nachteil bei der Repräsentation von Daten als Bild ist, dass normalerweise keine Bildrepräsentation vorliegt, über die man Informationen über den Bildinhalt beziehen kann, was z.B. zur Generierung von Zeigegesten notwendig wäre.

Wenn zur Skript-Ladezeit ein Befehl zum Anzeigen eines Bild-Gadgets gefunden wird, wird die Bitmap geladen, so dass zur Laufzeit keine Verzögerungen als Folge von Netzzugriffen auftreten können. Diese Vorgehensweise ist auch bei den anderen Gadget-Typen realisiert. Die Position des Bildes ist über das Persona-Skript jederzeit veränderbar. Je nach Wahl der Bildgröße sind unterschiedliche Effekte zu erzielen. Im Normalfall ist das angezeigte Bild kleiner als der für die Persona-Präsentation reservierte Platz (Abb. 5.3, links).

Ist die Bitmap des Bildes größer als der für die Persona-Präsentation benötigte Platz, so lassen sich leicht Panorama-Effekte erzielen. Dazu wird das Panoramabild als Hintergrund (z.B. in Gadgetregister 0) abgelegt. Da das Bild größer als der zur Verfügung stehende Platz ist, ist nur ein Bildausschnitt sichtbar. Während der Präsentation verschiebt man nun den sichtbaren Bildausschnitt durch Modifikation der Bildschirmposition, so dass vorher sichtbare Bildteile geclippt werden, während neue Bildteile auf der anderen Seite sichtbar werden. Für den Betrachter der Präsentation ergibt sich der Eindruck eines

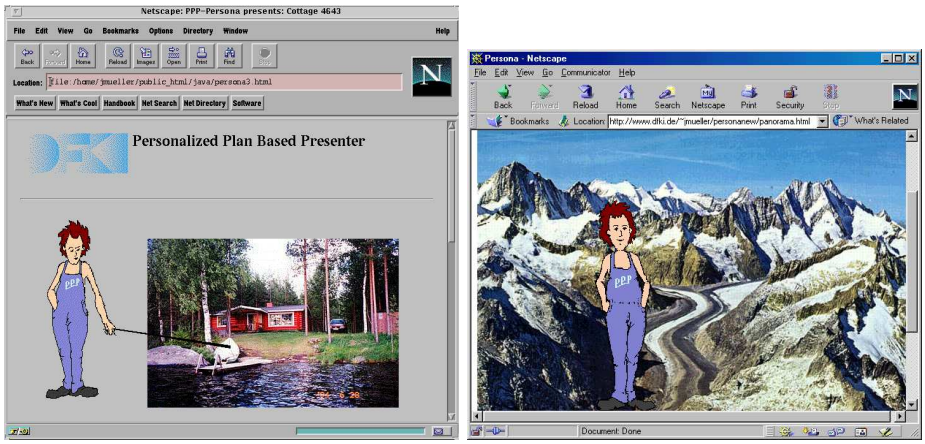


Abbildung 5.3: Einsatz des Bild-Gadgets: Neben kleinen Bildern, die eingeblendet werden, können mit Hilfe von den bildschirmfüllenden Bildern Panoramaffekte erzielt werden.

Kameraschwenks oder einer von Filmen bekannten Panorama-Darstellung (Abb 5.3, rechts).

5.2.2 3D-Gadget

Neben der Möglichkeit, aus dreidimensionalen Objektmodellen durch Verfahren wie z.B. Ray-Tracing Bitmap-Bilder zu erzeugen und sie mittels Bild-Gadget in die Präsentation einzubeziehen, kann durch Verwendung des 3D-Gadgets direkt eine dreidimensionale Abildung eines Objekts, das im OFF-Format [Phillips, 1994] vorliegt, erstellt werden (Abb. 5.4). Dabei sind Kamerastandpunkt und Blickpunkt frei wählbar und über das Persona-Skript ansteuerbar. Im Gegensatz zur Darstellung verschiedener Perspektiven eines 3D-Objekts über eine Folge von Bitmaps muss das Drahtrahmenmodell nur einmal übertragen werden und die einzelnen Ansichten werden durch das 3D-Gadget berechnet. Dadurch ergibt sich eine starke Verringerung der zu übertragenden Datenmenge, was bei WWW-Applikationen vorteilhaft ist.

Die Darstellung dreidimensionaler Objekte wird häufig bei der Generierung von technischen Dokumentationen benötigt [Rist, 1995]. Oft gibt es von den in der Dokumentation beschriebenen Geräten Drahtrahmenmodelle, die zur Veranschaulichung eingesetzt werden können.

Durch Modifikation der Beobachtungsperspektive über das Persona-Skript kann das gezeigte Modell aus beliebigen Perspektiven betrachtet werden. Gleichzeitig können Koordinaten zur Generierung von Zeigegesten durch das 3D-Gadget berechnet werden.

Zur Implementierung wurde auf ein schon bestehendes Java-Applet zur Visualisierung von 3D-Objekten zurückgegriffen [Meyer, 1997]. Um dieses Applet als 3D-Gadget verwenden zu können, war nur die Implementierung der entsprechenden Schnittstelle notwendig (Abschnitt 5.2.6).

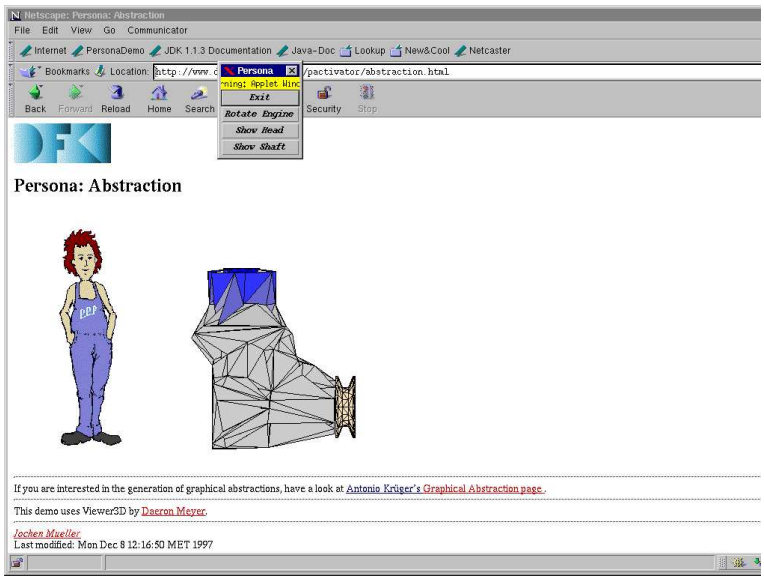


Abbildung 5.4: Visualisierung von Drahtrahmenmodellen durch das 3D-Gadget [Krüger, 1999]

5.2.3 Text-Gadget

In vielen Präsentationen (z.B. Lernprogramme, WWW-Führer, Bilanzdarstellungen) wird die Ausgabe von Texten und Tabellen benötigt. Damit der Präsentationsagent den dargestellten Text manipulieren bzw. auf Textelemente zeigen kann, benötigt er eine Repräsentation des dargestellten Textes. Zur Generierung von Zeigegesten braucht er die genaue Position des dargestellten Textelements. Auch zur Integration von sensitiven Textelementen ist diese Information (Abschnitt 5.4.4) erforderlich.

Das Text-Gadget liest einen Text ein, der analog zu HTML mit verschiedenen Tags ausgezeichnet ist [Samid, 1998]. Diese in den Text eingebetteten Tags bewirken

- die Definition von Schrift- und Hintergrundfarbe,
- die Wahl von Schriftarten,
- die Formatierung (Zeilenumbruch, Tabellendefinition, Ausrichtung),
- die Definition von Referenzen auf einzelne Textblöcke, so dass auf diese zugegriffen werden kann.

Falls der darzustellende Text nicht in dem für das Text-Gadget zugewiesenen Raum visualisiert werden kann, wird er in einzelne Seiten aufgeteilt, die über das Persona-Skript aufrufbar sind. Neben der vollautomatischen Steuerung über das Persona-Skript kann diese Funktion auch durch den Benutzer interaktiv aufgerufen werden (s. Kapitel 5.4.4), indem die Scrollfunktion an Mausevents gekoppelt wird.

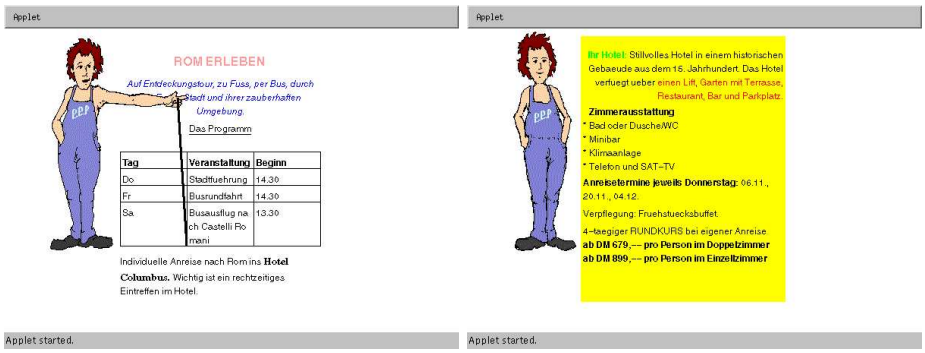


Abbildung 5.5: Einsatz des Text-Gadgets

Zur Koordinierung der Zeigegesten wird die Position eines Textblockes durch Zugriff über seine definierte Referenz ermittelt und dem Gestengenerator übergeben.

Außer durch Zeigegesten kann ein Textelement auch durch visuelle Effekte wie Fettdruck, auffällige Farbwahl oder auch durch Blinken hervorgehoben werden. Diese Techniken können dynamisch über das Persona-Skript zur Laufzeit angewählt werden.

5.2.4 Balken-, Kuchen- und Liniengrafik

In vielen Anwendungsbereichen müssen numerische Daten visualisiert werden. Ein breites Anwendungsfeld sind Geschäftsgrafiken. Unternehmensentwicklungen oder Börsenkurse werden oft als Liniengrafik dargestellt. Neben der reinen Darstellung der Daten sind noch weitergehende Informationen zu den Daten notwendig, um Zeigegesten ausführen zu können. Aus diesem Grund (und wegen einer Verringerung der übertragenen Datenmenge) ist es sinnvoll, den Präsentationsagenten mit der Fähigkeit zur Visualisierung solcher Daten auszustatten.

Weitere Beispiele für den Einsatz von Diagrammen:

- Kurvendiskussion (Ausbildung)
- technische/naturwissenschaftliche Anwendungen (Visualisierung von Experiment-Ergebnissen)
- Aktienkursentwicklung
- statistische Daten

Da der Präsentationsagent die Visualisierung der Daten übernimmt, kann man auch dynamisch, d.h. während der Präsentation, das Erscheinungsbild der Grafik verändern. Beispielsweise können bestimmte Elemente hervorgehoben dargestellt werden.

Ein anderer Grafiktyp, der häufig im Bereich betriebswirtschaftlicher oder statistischer Anwendungen benötigt wird, sind die Kuchendiagramme. Durch Kuchenteilstücke können die Größe einzelner Anteile übersichtlich visualisiert

werden. Ein Anwendungsbeispiel ist die Sitzverteilung im Bundestag. Bei Kuchengrafiken ist es oft üblich, hervorgehobene Daten durch „Herausziehen“ von Segmenten darzustellen.

Die Gadgets zur Erzeugung der Balken-, Torten und Liniendiagramme lesen eine Datei mit den numerischen Daten ein und berechnen aus ihnen die entsprechenden Diagrammtypen [Schaumann, 1998]. Dabei sind die Diagramme umfassend an die Präsentation anpassbar. Neben Zeichensatz, Farbwahl und Orientierung ist auch der Umfang der Legende wählbar. Auch bei diesen Gadget-Typen gibt es eine interne Repräsentation der dargestellten Daten, so dass Zeigegesten und sensitive Grafik möglich werden.

Die Darstellung einzelner Daten (d.h. Tortenstücke, Balken, Punkte) können zur Laufzeit dynamisch über das Persona-Skript so modifiziert werden, dass sie hervorgehoben erscheinen (z.B. Tortenstück herausziehen, Balken schraffieren, Punkte mit anderer Farbe markieren).

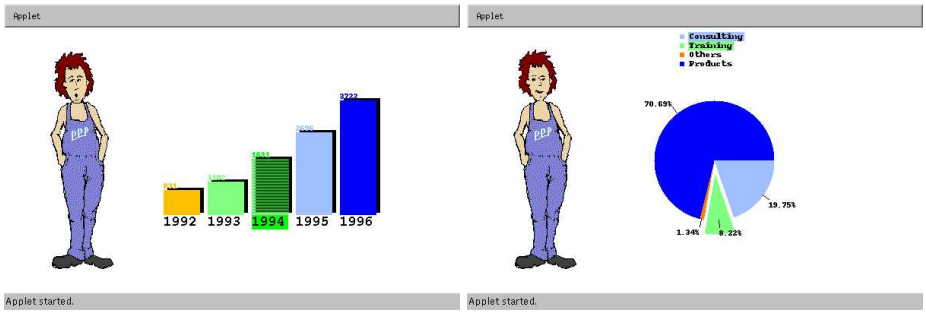


Abbildung 5.6: Datenvisualisierung mit Balken- und Tortendiagrammen

Das Gadget für Liniengraphen kann mehrere Graphen gleichzeitig darstellen. Dabei ist es möglich, für die einzelnen Kurven unterschiedliche Darstellungsformen zu wählen:

- Linienzüge
- Punktdarstellungen
- Flächengraphen.

Dadurch lassen sich die einzelnen Datenmengen visuell einfacher voneinander unterscheiden. Der jeweils sichtbare Aussschnitt des Koordinatensystems ist über das Persona-Skript wählbar.

In allen Fällen kann der Graph durch eine Legende, die als extra Gadget abgelegt wird, näher erläutert werden. Auch von der Legende besteht eine interne Repräsentation, so dass auch hier Zeigegesten und sensitive Elemente verwirklicht werden können.

5.2.5 Diashow-Gadget

Das Zeigen einer Bilderfolge ist eine häufige Taktik bei Präsentationen, wobei zu jedem Bild jeweils ein Kommentar abgegeben wird. Es bietet sich also

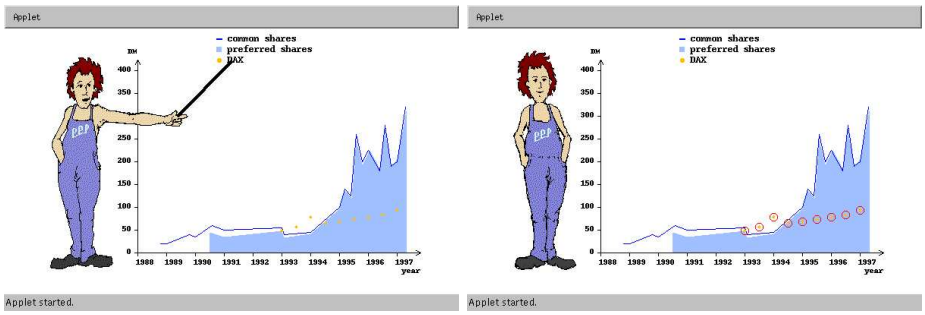


Abbildung 5.7: Datenvisualisierung mit Graph-Gadget

an, dieses Verfahren durch ein eigenes Gadget zu unterstützen. Das Diashow-Gadget liest eine Liste von Dateinamen ein und lädt die dazu gehörenden Bilder. Während der Präsentation kann nun durch die Bildsequenz geblättert werden (vorwärts, rückwärts, Einzelauswahl). Das jeweils sichtbare Bild wird über das Persona-Skript angewählt. Um die Übergänge zwischen den Bildern fließender zu gestalten, wurden spezielle Überblendungsfunktionen implementiert [Weiss, 1998]:

- Ein-/Ausblenden von Bildern
- Zoom-Effekte
- Bild-Verdrängung (Ein Bild wird aus dem sichtbaren Bereich herausgeschoben, während ein anderes nachgeschoben wird.)
- Bitoperationen zwischen Bildern

5.2.6 Implementierung weiterer Gadgettypen

Sich leicht an neue Anforderungen anzupassen ist eine wichtige Eigenschaft des flexiblen Präsentationsagenten. Im Fall der Gadgets bedeutet dies: Es soll ohne große Programmänderungen möglich sein, neue Gadgettypen einzuführen und in Präsentationen einzusetzen. Beispiele für weitere Gadgets sind z.B.:

- Visualisierung mathematischer Funktionen (Kurvendiskussion)
- Visuelle Simulation von Geräten/Vorgängen in Lernprogrammen (z.B. elektronische Schaltungen, chemische Prozesse)
- Visualisierung von Musik unter Verwendung der Notenschreibweise

Die Benutzung neuer Gadgettypen wird durch die objektorientierte Strukturierung und durch die Verwendung der eingesetzten Programmiersprache Java [Sun Microsystems, 1998b] stark erleichtert.

Alle Gadgets haben eine gemeinsame Oberklasse, nämlich das Visualization-Gadget (Abb. 5.9).

Um ein neues Gadget in die bestehende Klasssenhierarchie (Abb. 5.10) einzuordnen und damit die Menge der von dem Präsentationsagenten benutzbaren

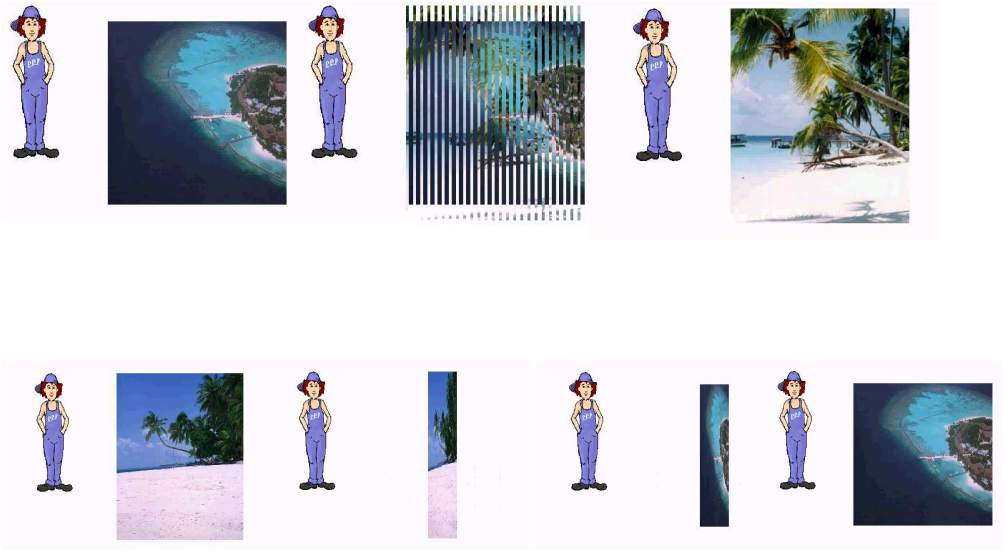


Abbildung 5.8: Diashow-Gadget

Gadgets zu erweitern, muss die neue Klasse von der Oberklasse Visualization-Gadget (oder einer ihrer Unterklassen) erben.

Die neue Klasse wird über ihren Klassennamen im Persona-Skript angesprochen und vom Laufzeitsystem dynamisch geladen und ausgeführt.

In einer Unterklasse zu VisualizationGadget müssen einzelne Methoden überdefiniert werden. Standardmäßig haben sie keine Funktion. Im Minimalfall müssen die Methoden „initialize“ und „paint“ überdefiniert werden. Eine Instanz wird mittels „initialize“ initialisiert. Die Parameter geben die Größe der zugewiesenen Zeigefläche an. Außerdem wird vom System der aktuelle Grafikcontext und das Persona-Applet übergeben. Die übergebene URL verweist auf eine Datei mit Daten für das Objekt. Das Format dieser Datei ist dem Implementierer der Klasse überlassen. Die paint-Methode wird beim Neuzeichnen der Oberfläche von der Persona-Engine aufgerufen. Über die Methode „modify“ ist es möglich, Befehle aus dem Persona-Skript an das VisualizationGadget zu schicken. Dadurch können Parameter (z.B. Zeichenmodi) bei dem Objekt über das Persona-Skript zur Laufzeit verändert werden. Die Funktion „findPointPosition“ liefert zu einem Objekt mit einem bestimmten Namen einen Referenzpunkt zurück. Bei Zeigeoperationen, die dieses Objekt referenzieren, wird dieser Punkt genutzt. Soll Persona ein Objekt mit dem Zeigestock umfahren, so werden mehrere Referenzpunkte benötigt, die über die Methode „findPointPath“ ermittelt werden.

```
public class VisualizationGadget {
   public VisualizationGadget() {
    // Konstruktor
  }
  public void initialize(Graphics g,int x,int y,int width,
                         int height,Applet applet,URL url) {
    // Einlesen der Daten, Vorbereitung der Grafikausgabe
  }
  public void paint() {
    // Zeichne Ausgabe
  }
  public void modify(String command) {
    // Modifikation des Gadget-Zustandes
    // durch das Persona-Skript
  }
  public Point findPointPosition(String name) {
    // liefert die Position zu dem angegebenen Objekt
  }
  public Point[] findPointPath(String name[]) {
    // liefert einen Polygonzug, der die
    // angegebenen Objekte beinhaltet
  }
}
```

Abbildung 5.9: Die Oberklasse VisualizationGadget

5.3 Generierung von Vitalaktionen

Oft kommt es während Präsentationen vor, dass längere Wartepausen eingelegt werden sollen, sei es aus dem Zwang heraus, im Hintergrund Berechnungen ausführen zu müssen, oder weil man dem Betrachter die Möglichkeit bieten will, sich die angebotene Information näher zu betrachten. In beiden Fällen sollte der Präsentationsagent nicht unbeweglich stehen bleiben, besser ist es, wenn er Vitalaktionen ausführt. Solche Aktionen suggerieren dem Benutzer Aktivität des Präsentationssystems. Er hat die beruhigende Gewissheit: Das System arbeitet und ist nicht abgestürzt. Gleichzeitig wirkt der Präsentationsagent lebendiger und natürlicher. Allerdings dürfen solche Vitalaktionen den Betrachter nicht von der eigentlichen Präsentation ablenken.

Der Vitalaktionsgenerator sorgt als Teil des Abspielmoduls für die Auswahl der passenden Vitalaktionen und initiiert ihre Ausgabe durch die Persona-Engine.

Beispiele für Vitalaktionen sind Atmen, Fußwippen oder Kopfdrehen. Einige Vitalaktionen sind in Abbildung 5.11 dargestellt.

Die Wahl der passenden Geste wird beeinflusst durch die schon verstrichene, die noch verbleibende Ruhezeit und die Tatsache, ob überhaupt Vitalaktionen ausgeführt werden dürfen. Beispielsweise darf während einer Zeigegeste keine

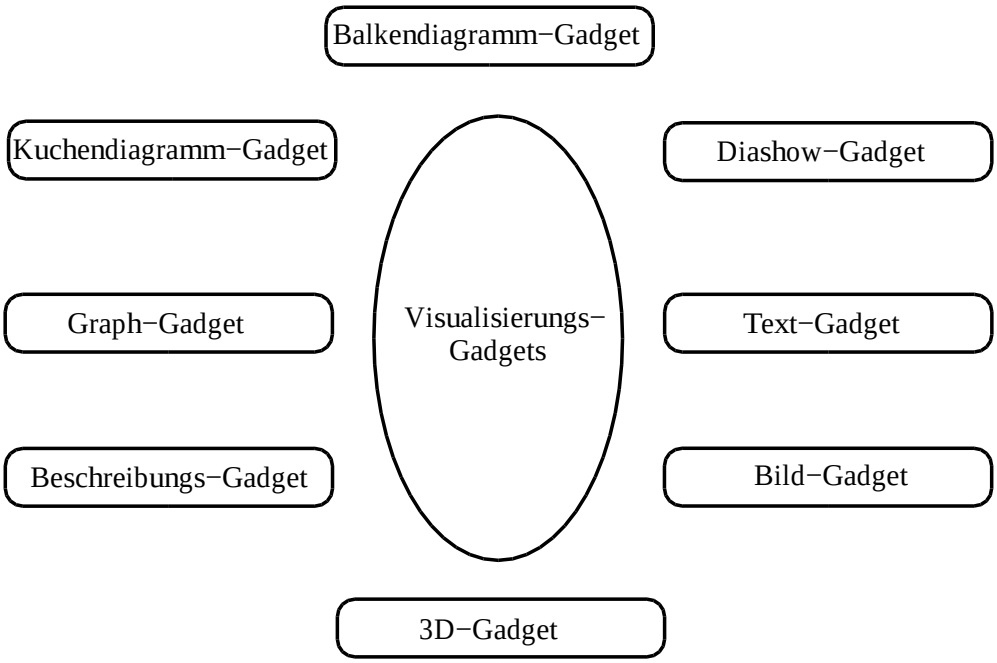


Abbildung 5.10: Übersicht über die Klassenhierarchie der Gadgets

Vitalaktion ausgeführt werden, weil sie die Zeigegeste selbst stören würde. Wenn möglich sollte nicht mehrmals hintereinander die gleiche Vitalaktion ausgeführt werden (Abbildung 5.12).

Mit Hilfe des Gesteneditors (Kapitel 4.1.2) kann definiert werden, dass eine Geste als Vitalaktion verwendet werden kann. Gleichzeitig wird festgelegt, nach wieviel Ruhezeit die Geste ausgeführt werden darf.

Die Daten für eine Vitalaktion werden nur geladen, wenn Vitalaktionen ausgeführt werden dürfen (oder die Geste wird auch für andere Zwecke benötigt).

Wird während des Ladevorganges des aktuellen Skripts festgestellt, dass Vitalaktionen benötigt werden, so werden die entsprechenden Daten geladen. Kommt es während der Abarbeitung des Skriptes zu einer Phase, in der Vitalaktionen abgespielt werden sollen, so werden aus der Menge der definierten Vitalaktionen diejenigen ausgwählt, die entsprechend ihrer Definition nach der aktuell verstrichenen Zeit eingesetzt werden dürfen. Aus dieser Menge wird zufällig eine Geste ausgewählt und gestartet.

5.4 Benutzerinteraktion - der Event-Handler

Die Player-Komponente des Präsentationsagenten muss auch mit Benutzereingaben umgehen können. Dabei ist in manchen Fällen eine weitere Verarbeitung von Benutzereingaben durch das Persona-Kernsystem bzw. das Präsentationssystem notwendig, allerdings erfolgt auch in diesen Fällen eine Vorverarbeitung innerhalb des Persona-Players.

Der Persona-Player bewältigt die Low-Level-Verarbeitung von Events, die

Abbildung 5.11: Beispiele von Vitalaktionen

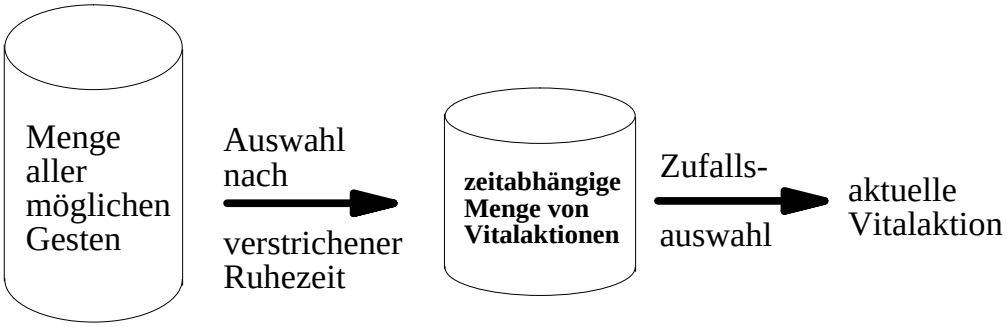


Abbildung 5.12: Auswahl einer Vitalaktion durch die Persona-Engine

von der Benutzerinteraktion herrühren, wie z.B. Maus- und Tastaturevents. Diese Events triggern entweder direkt Aktionen, die als Reaktion auf solche Eingaben bei dem Persona-Player registriert sind, sie bewirken einen Sprung zu einer anderen Adresse im Persona-Skript oder sie werden zur weiteren Verarbeitung an die Server-Komponente weitergereicht. Neben Events, die durch Benutzerhandlungen ausgeführt werden, gibt es auch Events, die aus technischen Gründen vom Event-Handler ausgeführt werden müssen. Ein Beispiel für eine solche Aktion ist das Erneuern des Window-Inhalts bei Refresh-Events, die der Window-Manager des Fenstersystems an die jeweiligen Applikationen schickt, wenn Fenster infolge von Sichtbarkeitsänderungen (Deikonifizierung, Fenster in der Hierarchie nach vorne bringen) erneuert werden sollen.

5.4.1 Dragging

In bestimmten Situationen kann es sinnvoll sein, den Benutzer die Position des Präsentationsagenten auf dem Bildschirm durch direkte Manipulation frei wählen zu lassen. Bei sehr komplexen Präsentationen ist es möglich, dass Persona vor einem Bildschirmelement steht, welches sich der Benutzer gerade ansehen will. Auch falls es nicht notwendig sein sollte, Persona mit der Maus zu verschieben, kann der Unterhaltungseffekt, der durch diese Aktion erreicht wird, trotzdem die Präsentation auflockern. Möchte das Präsentationssystem das Draggen von Persona mit der Maus zulassen, so muss es drei primitive Gesten bei

dem Persona-Player registrieren: START_DRAG, DRAGGING, END_DRAG (Abb. 5.13).

Abbildung 5.13: Elemente der Dragging-Geste

Außerdem muss festgelegt werden, welche Maustaste das Dragging steuert. Die Geste START_DRAG wird von Persona-Player abgespielt, wenn der Benutzer die festgelegte Maustaste über Persona drückt, die Geste DRAGGING wird abgespielt, wenn der Benutzer bei gedrückter Maustaste die Maus verschiebt, und END_DRAG wird gespielt, wenn der Benutzer nach einem Dragging-Vorgang die Maustaste wieder loslässt. Neben dem Abspielen der primitiven Geste DRAGGING muss zusätzlich noch die Position von Persona der Bewegung des Mauszeigers angepasst werden. Da die Definition der Dragging-Gesten über das Persona-Skript erfolgt, kann auch während einer laufenden Präsentation zeitweise das Dragging ein- und ausgeschaltet werden.

Nicht nur in Desktop-Applikationen kann es vorkommen, dass der Präsentationsagent bei komplexen Präsentationen vor Bildschirmelementen steht, sondern auch in WWW-Applikationen. Da immer mehr Applikationen eine WWW-Schnittstelle erhalten, werden auch komplexe Benutzeroberflächen im World Wide Web eingesetzt, so dass auch hier diese Problematik auftritt.

5.4.2 Pop-up-Menü

Pop-up-Menüs erlauben es dem Benutzer, während der Ausführung des Persona-Skriptes in den Programmfluss einzugreifen. Über Befehle des Skriptes lassen sich Pop-up-Menüs aktivieren (Abb. 5.14). Mit jedem Menüpunkt sind Sprungziele im Persona-Skript verknüpft. So lassen sich über das Menü Unterprogramme anspringen, die Teilpräsentationen einer großen Präsentation entsprechen. Außerdem kann so der Benutzer auswählen, welche Präsentationsteile er noch einmal zu sehen wünscht. Neben den lokalen Sprüngen innerhalb eines geladenen

Persona-Skriptes kann natürlich auch ein neues Skript vom Präsentationsserver über das Pop-up-Menü angefordert werden.

Abbildung 5.14: Das Pop-up-Menü von Persona

Die Ausführung des Persona-Skriptes wird durch ein sichtbares Pop-up-Menü nicht blockiert. Der Präsentationsagent macht das Menü sichtbar und fährt dann mit der Ausführung des Skriptes fort. Dadurch kann auch bei sichtbarem Menü eine laufende Präsentation weitergeführt werden. Soll der Präsentationsagent tatsächlich auf eine Benutzereingabe warten, so fügt das Generierungssystem eine Endlosschleife in das Skript ein, die nur über die Wahl eines Menüpunktes verlassen werden kann. Gleichzeitig ist aber auch die folgende Strategie möglich: Der Präsentationsagent zeigt das Pop-up-Menü, so dass der Benutzer einen Menüpunkt anwählen kann. Falls der Benutzer aber nach einer gewissen Zeitspanne keine Eingabe getätigt hat, wird das Menü wieder unsichtbar gemacht und die Präsentation läuft entsprechend einer bestimmten Voreinstellung weiter. Auf diese Art ist der Benutzer nicht zu einer Eingabe gezwungen, sie wird ihm freigestellt.

5.4.3 Pop-up-Eingabefenster

Neben einer Auswahl von vordefinierten Möglichkeiten (siehe Pop-up-Menü) muss der Präsentationsagent auch beliebige textuelle Eingaben zulassen. Zu diesem Zweck bietet Persona ein Pop-up-Eingabefenster an. Analog zum Pop-up-Menü wird es über das Persona-Skript aktiviert. Auch hier läuft die Präsentation weiter, falls das Eingabefenster sichtbar ist. Der Präsentationsagent kann so weitere Erläuterungen zu dem Eingabefenster machen. Z.B. kann er den

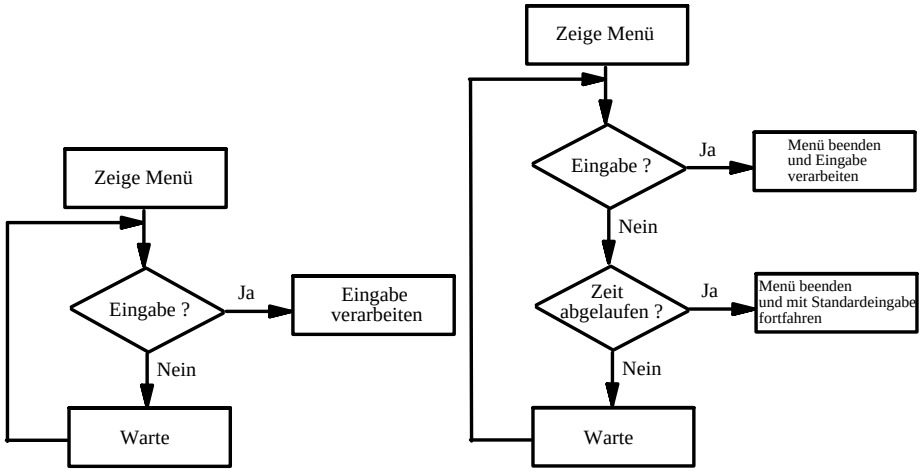


Abbildung 5.15: Programmfluss bei Pop-up-Menü mit und ohne Zwang zu einer Benutzereingabe

Benutzer über den Zweck der Eingaben informieren oder ihm zusätzlich ein passendes Beispiel geben. Drückt der Benutzer im Eingabefenster die Return-Taste, so wird das Fenster entfernt. Die weitere Verarbeitung hängt von dem definierten Modus ab. Im einfachsten Fall wird der Eingabestring in ein definiertes Stringregister des Players kopiert, von wo aus er eventuell weiter zum Präsentationsserver geschickt werden kann. Zusätzlich kann das Pop-up-Eingabefenster aber auch noch die Eingabe testen. Die Testprozedur lässt sich einfach erweitern, so dass auch applikationsspezifische Tests auf Client-Seite durchgeführt werden können. Bei der Definition des Eingabefensters müssen zwei Sprungadressen durch das Generierungssystem festgelegt werden: eine Sprungadresse für einen erfüllten Test und eine Sprungadresse für einen fehlgeschlagenen Test. Dadurch kann die Skriptausführung je nach Erfolg des Testes an einer anderen Adresse weitergeführt werden.

In Abb. 5.16 ist ein Beispiel für eine Benutzerinteraktion abgebildet, wobei ein Test der Benutzereingabe ausgeführt wird. Am Anfang wird das Eingabefenster gezeigt und der Präsentationsagent spricht: „Bitte nennen Sie mir ihren Namen." Falls der Benutzer wirklich den Namen eintippt, so spricht der Präsentationsagent: „Vielen Dank",und ist fertig. Falls der Benutzer keine Eingabe macht, sagt der Präsentationsagent zunächst: „Ich benötige ihren Namen", und zeigt wieder das Eingabefenster. Gibt der Benutzer jetzt immer noch keine Eingabe ein, so zeigt der Präsentationsagent ein ärgerliches Gesicht und fängt wieder von vorne an. Diese Schleife kann der Benutzer nur durch eine Eingabe verlassen.

5.4.4 Sensitive Grafik

Ein wichtiges Gestaltungselement einer Präsentation ist sensitive Grafik. Durch sie wird es dem Benutzer ermöglicht, auf dargestellte Objekte mit der Maus zu klicken und weitere Informationen über sie zu erhalten. Da nahezu beliebige Bildschirmbereiche so anklickbar sind, wird der Interaktivitätsgrad einer sol-

Abbildung 5.16: Persona fragt nach dem Benutzernamen

chen Präsentation stark gesteigert. Die Abspielkomponente kann einerseits den Präsentationsagenten selbst sensitiv machen. Dazu muss lediglich eine Sprungadresse definiert werden, die im Falle eines Klick-Events innerhalb der Persona-Bitmap angesprungen wird. Andererseits können aber auch beliebige andere Bildschirmbereiche sensitiv gemacht werden. Auch hier muss eine Sprungadresse durch das Generierungssystem definiert werden. Die Koordinaten der sensitiven Bildschirmbereiche können direkt übergeben werden, es kann aber auch ein Zugriff auf die interne Repräsentation der Daten der VisualizationGadgets erfolgen.

Im linken Teil der Abbildung 5.17 ist eine Modemplatine dargestellt. Klickt der Benutzer auf ein Bauteil, so gibt der Agent eine Erläuterung zu diesem Bauteil ab. Sein Skript hat zu jedem Bauteil ein Unterprogramm, das mit dem jeweiligen Bauteil verknüpft ist.

Im rechten Teil der Abbildung ist ein Textausschnitt zu sehen, in dem einzelne Wörter anklickbar sind. Im Gegensatz zu dem Modembeispiel, in dem die Bereiche durch direkte Angabe der Koordinaten definiert wurden, wird hier die interne Repräsentation des Textes innerhalb des Text-Gadgets ausgenutzt um den von einem Textblock überdeckten Bildschirmbereich zu bestimmen. Neben der direkten Ansteuerung von Gesten des Präsentationsagenten lassen sich so auch die aus dem WWW bekannten Hyperlinks realisieren: das Anklicken eines Wortes bewirkt die Anzeige einer anderen Textpassage.

Oft werden in Präsentationen Menüs verwendet, die immer verfügbar sind, um dem Benutzer eine ständige Eingriffsmöglichkeit zu bieten. Darüber hinaus kommen häufig Menüs zum Einsatz, deren Elemente durch Ikonen realisiert sind, was einen Vorteil bei der Entwicklung mehrsprachiger Präsentationen bedeutet, da in diesem Fall kein Text übersetzt werden muss und das Menü so für jede Sprache geeignet ist. In Abbildung 5.18 ist ein solches Menü abgebildet. Das Generierungssystem hat aus einer Datenbank, die Motive für Ikonen vorhält, die für diese Präsentation passenden ausgewählt und sie zu einer Menüleiste

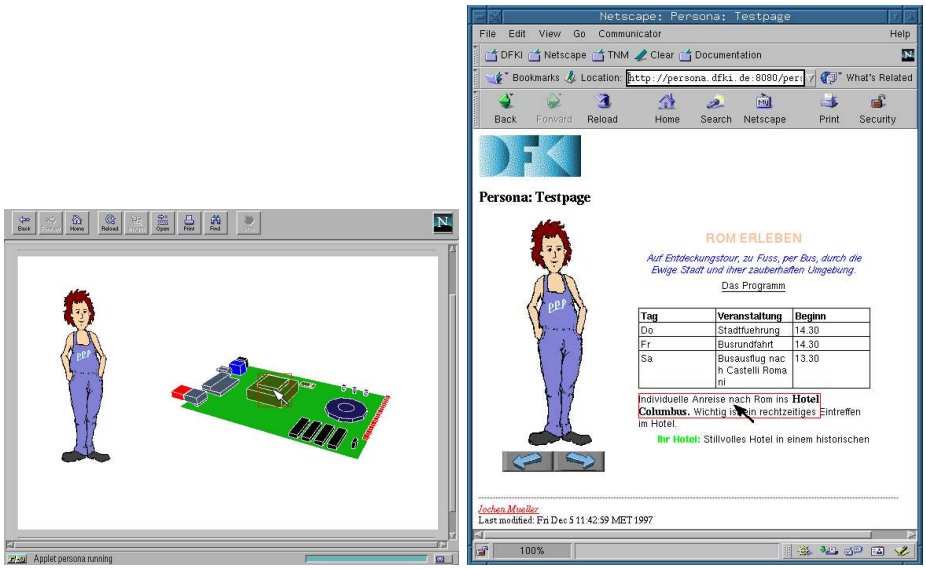


Abbildung 5.17: Sensitive Grafik

kombiniert. Gleichzeitig wurden die einzelnen Bildelemente sensitiv gemacht und an die entsprechenden Unterprogramme gekoppelt.

5.4.5 Andere Eingabeelemente

Bestimmte Applikationen erfordern die Definition weiterer Eingabeelemente über die oben beschriebenen vordefinierten hinaus. Zu diesem Zweck wird von der Abspielkomponente eine Schnittstelle zur Verfügung gestellt, die es erlaubt, die internen Variablen (z.B. Register, Programmzähler) zu modifizieren. Da die Java-Version der Abspielkomponente eine Unterklasse der Java-Klasse Component implementiert, kann sie ohne größeren Aufwand in eine Benutzeroberfläche integriert werden.

In Abbildung 5.19 wurde der Präsentationsagent um eine Text-Eingabezeile erweitert, die im Gegensatz zu dem Pop-up-Eingabefeld ständig verfügbar ist. In dem abgebildeten Szenario wird dieses Eingabefeld dazu verwendet, mit dem Präsentationssystem in natürlicher geschriebener Sprache zu kommunizieren. Es dient also als Ersatz für eine echte Spracheingabe, die nicht in jedem Umfeld sinnvoll ist (z.B. in Großraumbüros).

Ein anderes Eingabeelement sind Schieberegler (Abbildung 5.20). In diesem Szenario, in dem der Präsentationsagent in der X-Window-Version zu sehen ist, kann der Benutzer über den Schieberegler die Geschwindigkeit der Präsentation beeinflussen.

Durch die Einführung der Java-Swing-Elemente [Gutz, 1998] wird die verfügbare Palette der Eingabeelemente stark erweitert. Auch diese Elemente können über die beschriebene Schnittstelle in die Abspielkomponente integriert werden.

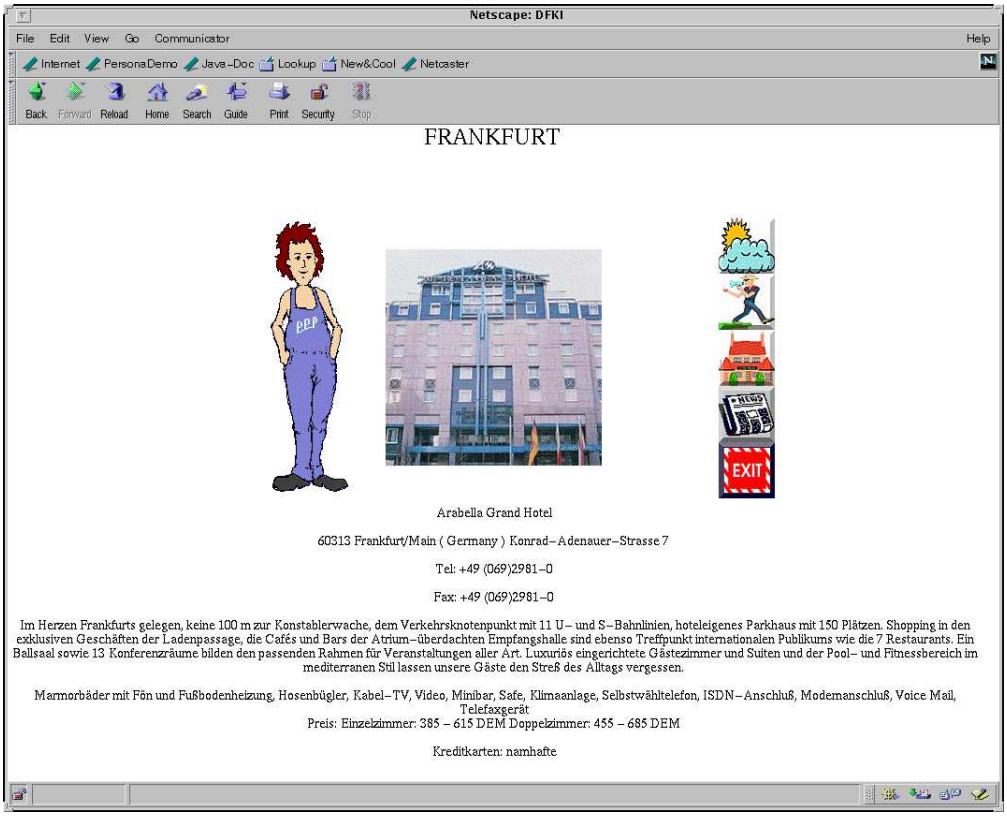


Abbildung 5.18: Ein Menü mittels sensitiven grafischen Elementen

5.4.6 Spracheingabe

Die Verwendung einer Spracheingabe steigert das natürliche Erscheinungsbild des Präsentationsagenten beträchtlich. Innerhalb des Persona-Systems dient die Spracheingabe zur Steuerung des Präsentationsagenten. Da dieser das Präsentationssystem repräsentiert, kann somit auch das durch ihn repräsentierte System angesteuert werden. Dabei wird die Übersetzung der Audio-Information in eine Textinformation auf Client-Seite, d.h. innerhalb des Applets, erledigt, die Weiterverarbeitung der Sprachinformationen wird auf dem Server durchgeführt:

Benutzer spricht in Mikrofon $\longrightarrow$ Umwandlung der Audioinformation in Textinformation $\longrightarrow$ Übertragung zum Server $\longrightarrow$ Informationsextraktion $\longrightarrow$ Ansteuerung des Anwendungssystems

Da das Persona-Applet innerhalb eines WWW-Browsers läuft, ist die Integration eines Spracherkenners auf Client-Seite relativ kompliziert. Es gibt mehrere Möglichkeiten einen Spracherkenner zu integrieren:

- Da die Spracherkenner meistens in C/C++ implementiert sind, müssen sie als Plugin in den Browser integriert werden. Dazu ist es notwendig, eine passende Schnittstelle für sie zu entwickeln. Durch browserspezifische Funktionen ist es möglich, auf das Plugin zuzugreifen und die erkannten Texte zu ermitteln. Um ein solches System benutzen zu können, muss sich der Benutzer zunächst das Plugin installieren.

Abbildung 5.19: Persona und Eingabezeile

- JavaSpeech ist eine Standardschnittstelle von Java zu Spracherkennungs- und Sprachsynthesesystemen [Sun Microsystems, 1998a]. Durch sie wird es möglich, auf alle unterstützen Spracherkenner bzw. Sprachsynthesizer über eine uniforme Schnittstelle zuzugreifen. Beispielsweise existieren Treiber für das System IBM ViaVoice [IBM, 1998], so dass dieser Spracherkenner integriert werden kann. Für diese Methode muss der Anwender ein unterstütztes Spracherkennungsprogramm auf seinem Rechner installieren, und das Persona-Applet muss signiert sein, da es mit einem auf dem Client-Rechner installierten Programm kommuniziert.

- Eine weitere Möglichkeit besteht darin, die Fähigkeit vieler Spracherkennungssysteme unter Microsoft Windows auszunutzen, ihre Ausgabe in ein jeweils aktives Textfenster einer beliebigen Applikation zu kopieren. Implementiert man für Persona ein spezielles Texteingabefenster (eventuell sogar ohne sichtbaren Text), das dafür sorgt, dass es während des Spracherkennungsvorganges immer den aktuellen Fokus erhält, so kann eine große Zahl von Texterkennern innerhalb des Persona-Systems eingesetzt werden, wobei noch nicht einmal eine echte Schnittstelle zwischen dem Persona-Applet und dem Spracherkenner benötigt wird. Aus diesem Grund muss auch das Applet nicht signiert sein, denn es kommuniziert nicht direkt mit lokalen Komponenten. Da das Persona-Applet so einen schon installierten Sprachsynthesizer ausnutzen kann, ohne weitere systemspezifische Software zu benötigen, ist diese Lösung wahrscheinlich im Internet am einfachsten einzusetzen.

Der von einem Spracherkennungssystem gelieferte Text muss nun zur Bestimmung seines Inhalts in einem nächsten Schritt weiter analysiert werden. Dieser weitere Verarbeitungsschritt ist im aktuellen System als Server-Lösung realisiert, um die Größe des Applets klein halten zu können. Die Schnittstelle des Persona-Applets ist so konzipiert, dass es für die weitere Verarbeitung un-

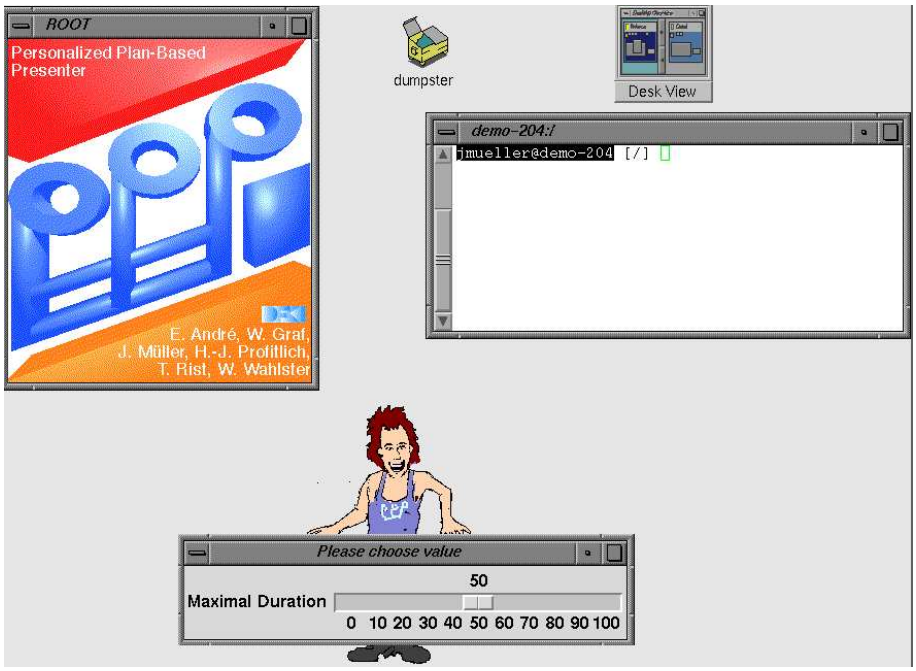


Abbildung 5.20: X-Window-Version von Persona mit Schieberegler

erheblich ist, ob die Eingabe über den Texterkenner erfolgt oder ob der Text direkt über ein Eingabefeld eingegeben wird. In beiden Fällen wird der Textstring zum Server gesendet und dort weiter analysiert. Als Resultat erhält das Applet ein neues Skript, das ausgeführt wird, um die Reaktion des Systems auf die Eingabe darzustellen.

Im aktuellen System, in dem Persona als Präsentationsagent innerhalb des AIA-Systems (Abschnitt 7.2) eingesetzt wird, erfolgt nach der Analyse des Eingabetexts eine Anfrage an das Präsentationssystem, das eine neue Präsentation generiert. Die Analyse des Textes erfolgt durch eine spezielle Dialoganalysekomponente [Tschernomas, 1999a], die auf nichtdeterministischen Automaten beruht. Die Automaten, die mit Hilfe eines Editors interaktiv erstellt werden, werden dazu eingesetzt, aus dem Text, der mittels Morphix [Finkler und Neumann, 1988] morphologisch annotiert wurde, durch grammatische Analyse die Satzstruktur zu bestimmen. Aufbauend auf dieser Information extrahiert der Parser weitergehende Informationen, die zum Auffüllen der Wissensbasis des Planungssystems genutzt werden.

5.4.7 Externe Programme

Die Abspielkomponente stellt auch für externe Programme eine Schnittstelle bereit, über die diese Einfluss auf den weiteren Ablauf des Persona-Skriptes nehmen können. Zum einen können über diese Schnittstelle mehrere Persona-Player, die in getrennten Playern eingesetzt werden, miteinander kommunizieren (Abschitt 5.5), zum anderen können aber auch andere Programme benutzt werden.

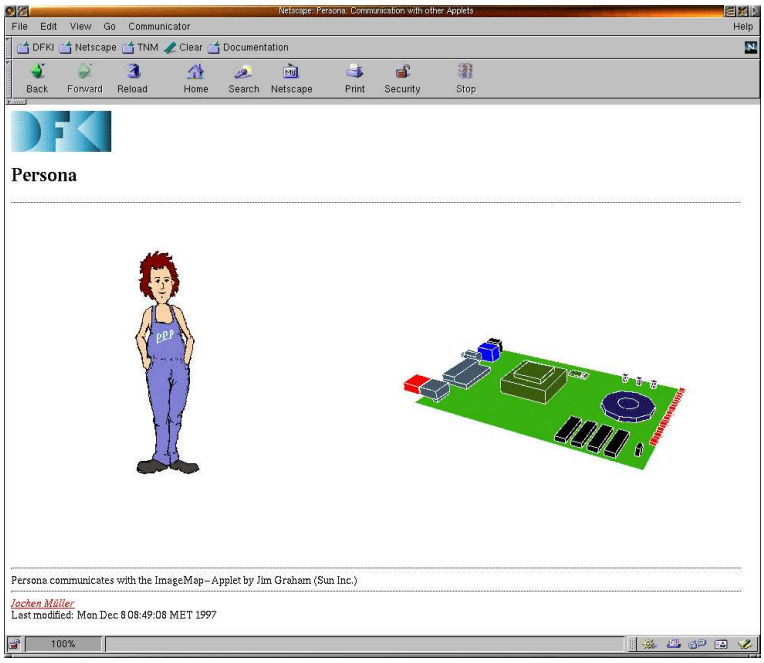


Abbildung 5.21: Persona und Imagemap-Applet

Abbildung 5.21 zeigt zwei Applets auf einer HTML-Seite: den Präsentationsagenten und ein Applet, das auf Benutzereingaben mit der Maus reagiert. Dieses Applet ist als Demonstration im Java Development Kit von Sun Microsystems enthalten [Sun Microsystems, 1998b]. Um eine Zusammenarbeit mit dem Präsentationsagenten zu ermöglichen, wurde es um eine Schnittstelle erweitert, so dass das Imagemap-Applet mit dem Persona-Applet kommunizieren kann.

Eine Anwendung für diese Technik könnte beispielsweise eine komplexe Simulation eines technischen Vorganges sein, die von dem Präsentationsagenten präsentiert und gesteuert wird. Das Simulationsprogramm müsste in diesem Fall mit einer passenden Schnittstelle ausgerüstet werden, die von Persona aus angesteuert werden kann. Auf diese Art wäre es möglich, Simulationen, die unter Kontrolle von Persona (angesteuert über das Persona-Skript) ablaufen, mit Hilfe von Persona kommentieren zu lassen. Eine solche Anwendung ist beispielsweise für Ausbildungszwecke denkbar.

5.5 Mehrere Präsentatoren in einem Applet

Sollen in einer Präsentation mehrere Präsentationsagenten eingesetzt werden, so muss deren Handlung koordiniert und synchronisiert werden. Die Steuerungskomponente, Abschnitt 3.3, liefert eine Dekomposition eines komplexen Präsentationsskriptes in Präsentationsskripte, die nur primitive Aktionen enthalten. Dabei wird für jeden Präsentator ein eigenes Skript generiert.

Zu jedem Präsentationsagenten gibt es eine eigene Instanz der Persona-Engine, die jeweils als eigenständiger Thread realisiert ist.

Zu Synchronisationszwecken fügt das Generierungssystem Synchronisationspunkte in die Persona-Skripte ein. Jede Definition eines Synchronisationspunktes beinhaltet eine Liste mit den Namen der Agenten, mit denen die Synchronisation durchgeführt werden soll. Erreicht eine Persona-Engine einen solchen Synchronisationspunkt, so prüft sie, ob die anderen Engines schon auf sie warten. Falls dies der Fall ist, so werden diese informiert und die Skripte werden an der Stelle nach den Synchronisationspunkten fortgeführt. Falls noch nicht alle anderen zu synchronisierenden Agenten warten, so registriert sich die Persona-Engine als wartende Maschine und geht in eine Warteschleife über, bis auch die anderen Agenten die entsprechende Stelle im Skript erreicht haben.

Die Synchronisation der einzelnen Threads ist deshalb besonders wichtig, weil es nicht möglich ist, die exakte Ausführungsdauer der Aktionen im Voraus zu berechnen, z.B. ist die ausführende CPU stark belastet oder es kommt zu Verzögerungen infolge von hoher Netzlast.

Abbildung 5.22 zeigt ein Beispiel mit drei Agenten. Das erste Kommando in jedem Skript ist ein Synchronisationskommando. Dadurch wird sichergestellt, dass alle Agenten zur gleichen Zeit mit der Präsentation beginnen und es nicht durch hohe Netzlast zu Verzögerungen kommt. Nach der Synchronisation der drei Threads werden die jeweils folgenden Kommandos der Skripte ausgeführt. Der nächste Synchronisationspunkt betrifft nur die Agenten 1 und 2. Agent 3 ist davon nicht betroffen. In diesem Beispiel ist die letzte ausgeführte Aktion die letzte Aktion von Agent 3, da diese Aktion nach der Synchronisation der Agenten 2 und 3 ausgeführt wird und dieser Synchronisationspunkt nach der Synchronisation der Agenten 1 und 2 erreicht wird.

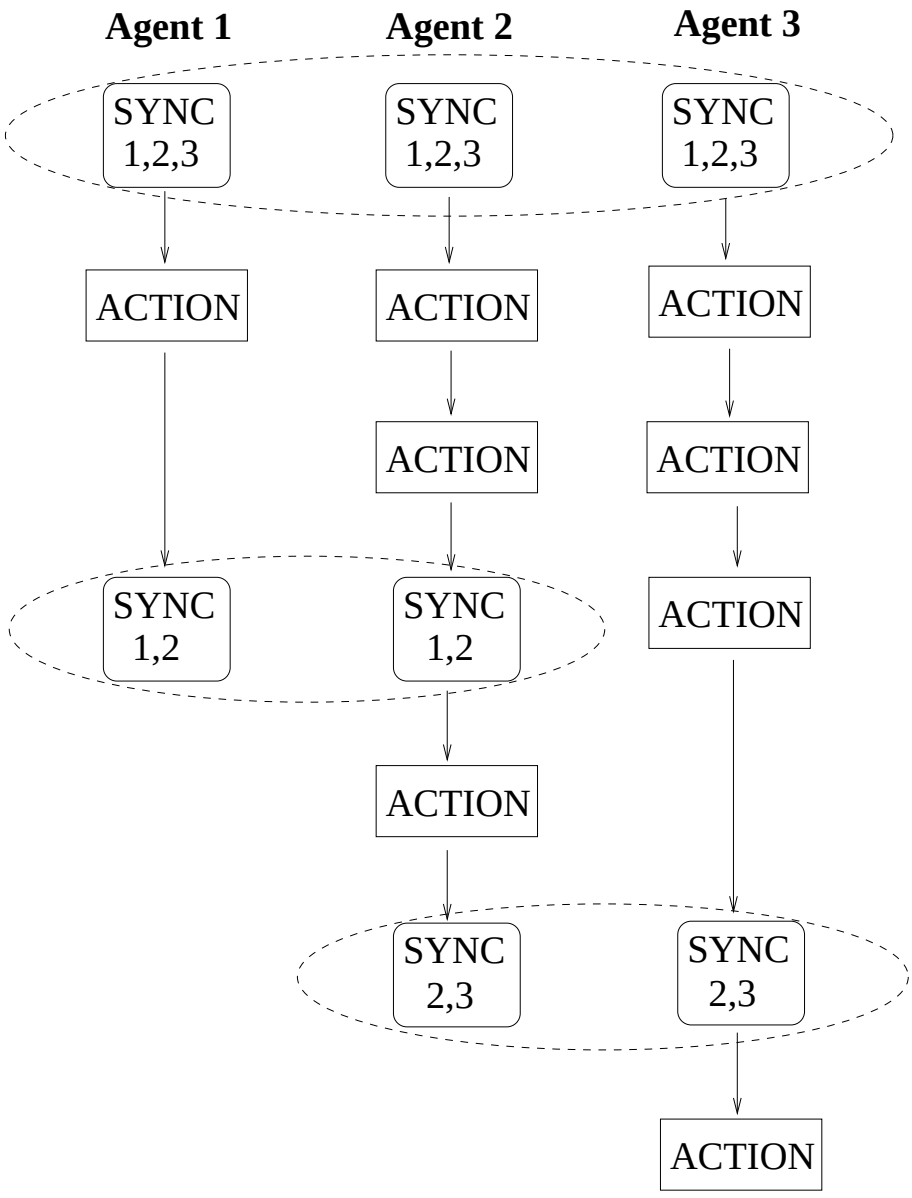


Abbildung 5.22: Synchronisation mehrerer Präsentationsagenten

Kapitel 6

Desktop- und WWW-Präsentationen

Der Präsentationsagent PPP-Persona lässt sich als animierter Agent in Desktop-Applikationen und in WWW-Präsentationen einsetzen. Für beide Anwendungsszenarios gibt es Versionen des Präsentationsagenten, die auf die jeweiligen speziellen Anforderungen abgestimmt sind.

6.1 Persona als Desktop-Präsentator

Neben der Implementation des Präsentationsagenten als Java-Komponente zum Einbinden in Java-Applikationen oder zum Einsatz in WWW-Präsentationen gibt es auch eine Variante zur Benutzung als Desktop-Präsentationsagent. Im Gegensatz zur ersteren Version kann sich dieser Präsentationsagent frei über den Bildschirm bewegen. Er ist dabei nicht an Fenster gebunden und kann sich ohne Rücksicht auf andere Bildschirmelemente an beliebige Positionen bewegen.

6.1.1 Grafische Realisierung

Die Implementation basiert auf X11 unter Verwendung der X11-Shape Extensions [Packard, 1989]. Mit Hilfe der X11-Shape-Extensions lassen sich Fenster mit einer unregelmäßigen Begrenzung definieren. Dies ist notwendig, da das Persona-Window nicht größer als die Figur sein soll. Zum einen darf von dem Persona-Window nur die tatsächlich von der Figur überdeckte Fläche benutzt werden, zum anderen stellt diese Vorgehensweise sicher, dass nur Persona betreffende Events (Tastatur-, Maus-Events) auch an Persona gesendet werden. Falls der Mauszeiger während eines Events nicht genau über dem Präsentationsagent steht, werden die Events an andere Applikationen, deren Fenster eventuell unter Persona liegen, weitergeleitet.

Will man die X11-Shape-Extension verwenden, muss neben der Bildinformation der Figur auch eine Bitmaske vorliegen, die bestimmt, welche Bildpunkte zum Fenster gehören und welche nicht. Diese Bitmaske wird bei der Persona-Implementation automatisch beim Laden der Bilder berechnet. Dazu wird die

Transparenzinformation, die in der GIF-Datei der Gestenbilder abgelegt ist, ausgelesen.

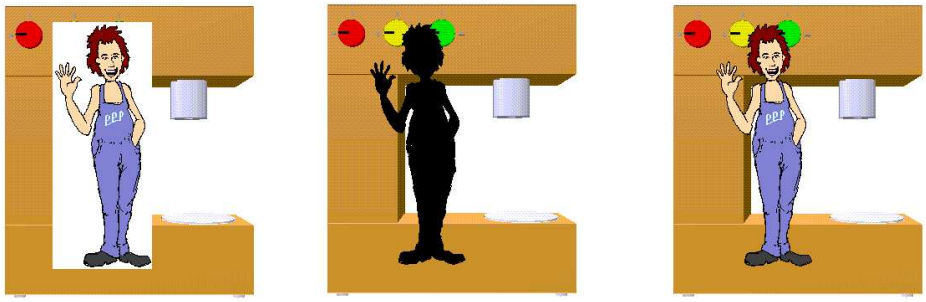


Abbildung 6.1: Wirkung der X11-Shape-Extension

Mit Hilfe dieser Bitmaske ist es möglich, die Ausdehnung von Persona pixelgenau zu definieren. Wie aus Abbildung 6.1 ersichtlich, wäre das Fenster von Persona ohne Verwendung dieser Technik rechteckig. Kombiniert man aber die Bitmap von Persona, die die Farbinformation enthält, mit der Bitmaske, die die Information über die Ausdehnung enthält, so ergibt sich die gewünschte Wirkung.

6.1.2 Anbindung an das Anwendungssystem

Der Präsentationsagent ist in diesem Anwendungsszenario als Server-Komponente implementiert. Das Anwendungsprogramm sendet als Client Präsentationsaufgaben an den Server, die dieser selbständig abarbeitet. Die Kommunikation zwischen dem Anwendungsprogramm und dem Persona-Server erfolgt über RPC (remote procedure call) [Sun Microsystems, 1990]. Neben den Anfragen, die über das RPC-Applikationsinterface beim Server eintreffen, müssen auch Events (z.B. Maus-Events, Tastatur-Events, Window-Events) bearbeitet werden. Dabei kann der Persona-Server einen Teil der Events selbst bearbeiten (z.B. Refresh-Events) während andere Events wie Menüeingaben an das Anwendungsprogramm übermittelt werden.

6.1.2.1 Interaktion zwischen Persona-Server und Window-Manager

Um dem Präsentationsagenten zu ermöglichen, auf Ereignisse innerhalb der Benutzeroberfläche zu reagieren, ist eine Kommunikation mit dem Window-Manager notwendig. Über den Window-Manager kann der Präsentationsagent die Position und Lage der einzelnen Fenster bestimmen. Zur Berechnung einer korrekten Zeigegeste muss er z.B. die Position des zu zeigenden Objekts relativ im entsprechenden Fenster bestimmen und zusätzlich die Position des Fensters ermitteln (Abbildung 6.3). Durch diese Technik ist es möglich, dass ein Benutzer das Fenster mit der Maus verschiebt und dennoch eine Zeigegeste korrekt ausgeführt werden kann.

Auch bei der Berechnung der jeweiligen Position des Präsentationsagenten muss die aktuelle Position der Fenster berücksichtigt werden, will man nicht,

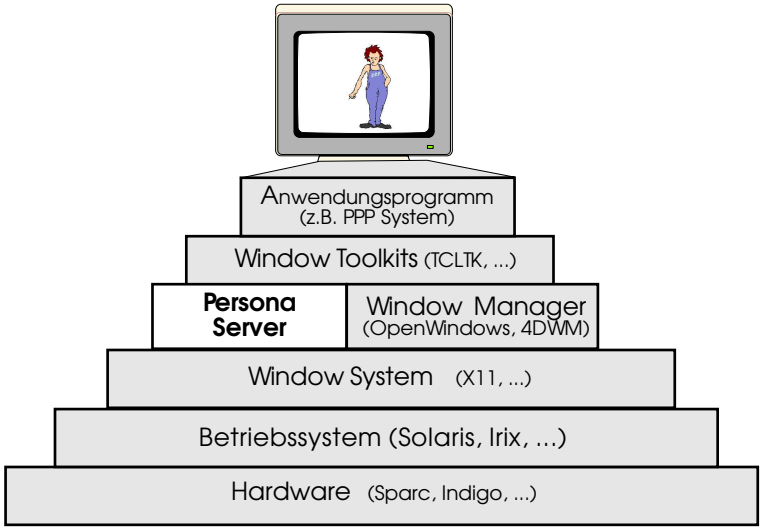


Abbildung 6.2: Integration des Persona-Servers in das Geamtsystem (aus [André et al., 1996a])

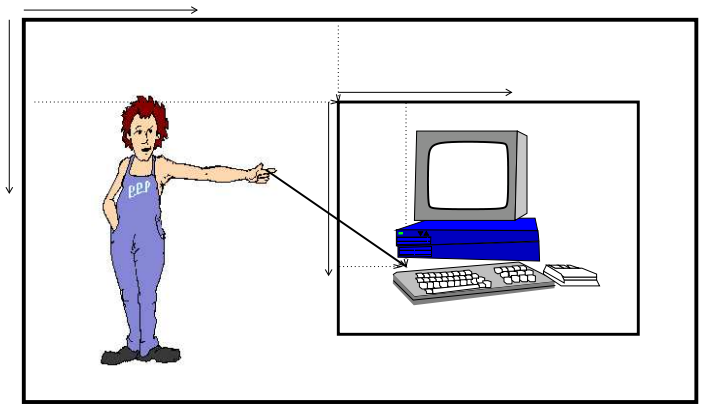


Abbildung 6.3: Berechnung von Zeigegesten

dass der Präsentationsagent genau vor dem Fenster steht, über dessen Inhalt er referiert. Also muss bei der Wahl des Standplatzes die Position der momentan wichtigen Fenster in Betracht gezogen werden. Soll bei Positionsänderungen zusätzlich darauf geachtet werden, dass der Präsentationsagent wenn möglich nicht vor den Fenstern vorbeiläuft, so müssen die Fensterpositionen in die aktuelle Bewegungsplanung einfließen und es muss jeweils ein optimaler Weg berechnet werden.

Der Persona-Server stellt auch Funktionen zur Überwachung von Windows zur Verfügung. Mit Hilfe dieser Funktionen kann der Präsentationsagent auf Manipulationen der registrierten Windows seitens des Programms oder des Benutzers reagieren. Beispielsweise kann der Agent dafür sorgen, dass bestimmte Fenster immer im Vordergrund oder an bestimmten Bildschirmpositionen stehen.

Damit sich der Präsentationsagent vor und auch zwischen übereinander liegenden Fenstern aufhalten kann, muss er die Reihenfolge, in der die Fenster gezeichnet werden, manipulieren können. Im Normalfall legt der Windowmanager diese Reihenfolge fest (bzw. der Benutzer, wenn er mit der Maus Änderungen in der Fensterhierarchie vornimmt. Soll sich der Agent zwischen bestimmten Fenstern aufhalten, so muss er ständig seine Position innerhalb der Fensterhierarchie kontrollieren und gegebenenfalls die korrekte Reihenfolge wieder herstellen.

Aufgrund der technischen Realisierung des Agenten als RPC-Server ist es notwendig, die erwähnten Überwachungsfunktionen von einem externen Prozess durchführen zu lassen, da der RPC-Server durch das Warten auf die RPC-Kommandos blockiert ist. Der externe Prozess ist ein weiterer Client für den Server, der die Aufgabe hat, Window-Events abzufangen, aufzubereiten und in geeignete Kommandos für den Server umzusetzen. Außerdem hat er die Aufgabe des Generators für Ruhegesten übernommen, d.h. er generiert Ruhegesten, falls der Server keine anderen Aktionen ausführen muss. Beim Eintreten der entsprechenden Ereignisse fordert der Überwachungsprozess vom Persona-Server-Prozess die notwendigen Aktionen an.

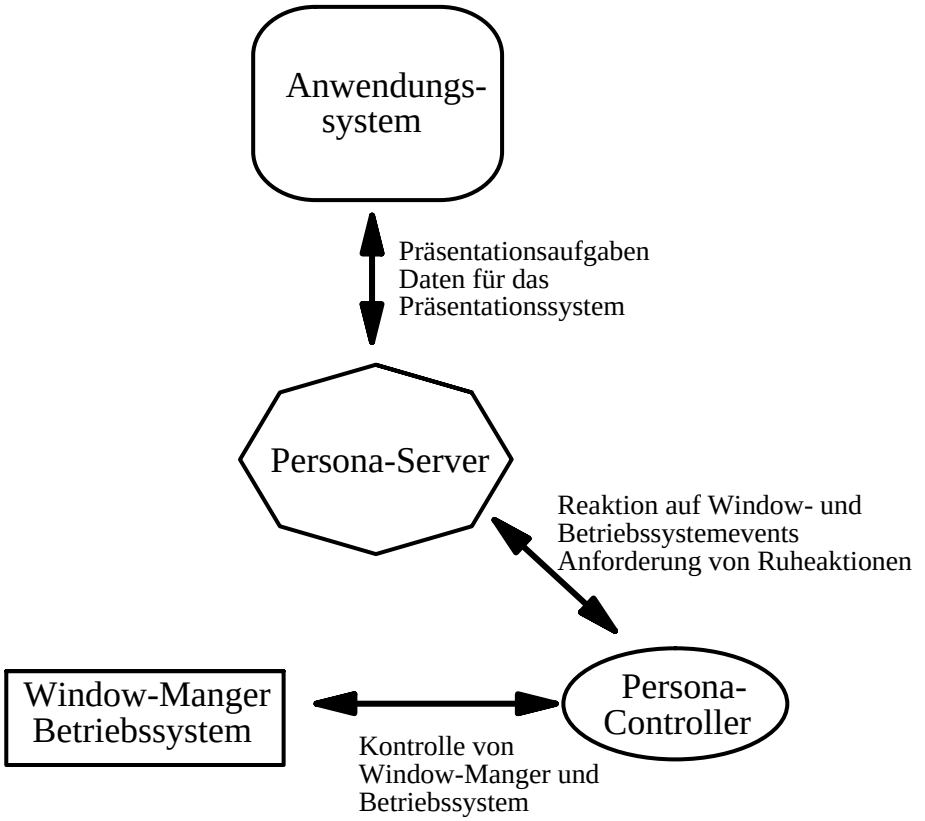


Abbildung 6.4: Die einzelnen Komponenten der Desktop-Version des Präsentationsagenten

In der aktuellen Version sind der Persona-Server und auch der Persona-Controller als C-Module implementiert. Um die Entwicklung der WWW-Version und der Desktop zu vereinheitlichen, könnte man die Persona-Engine der WWW-

Version mit den Ausgabekomponenten der Desktop-Version koppeln. Zum gegenwärtigen Zeitpunkt ist es noch nicht möglich, von Java aus die X11-Shape-Extension direkt über eine standardisierte Java-Schnittstelle anzusteuern. Deshalb sind für die Desktop-Version systemspezifische Module (in C) notwendig. Diese systemabhängigen Module müssten dann über JNI (Java Native Interface) in das Java-Programm eingebunden werden. Durch Nutzung dieser Technik ist es dann möglich, von Java aus Persona frei über den Bildschirm (ohne Fensterrahmen) zu bewegen.

6.2 Techniken zur WWW-Integration

Bei der Generierung von WWW-basierten Präsentationen muss auf die besonderen Anforderungen eingegangen werden. Bei WWW-Anwendungen gibt es üblicherweise eine Trennung zwischen Client und Server-Seite. Beide Komponenten laufen auf unterschiedlichen Rechnern. Die Kommunikation zwischen ihnen ist normalerweise nur beschränkt möglich (z.B. durch Firewall-Systeme).

6.2.1 Konzeption

Die WWW-Konfiguration des Präsentationsagenten Persona besteht aus einem Client, der in Form eines Applets in WWW-Seiten eingebettet ist, welcher zur Präsentationszeit die Präsentation abspielt und die Benutzeroberfläche für das System zur Verfügung stellt. Die zweite Komponente des Systems ist der Server, der aufgrund der vorhandenen Daten und mit Hilfe eines Präsentationssystems die für die Präsentation benötigten Daten und Informationen bereit hält.

Zur Kommunikation zwischen Client und Server wird das HTTP-Protokoll eingesetzt [Gettys et al., 1999]. Durch die Beschränkung auf das HTTP-Protokoll ist die Anwendung gegenüber Firewalls nicht empfindlich. Ein weiterer Vorteil ist: Es müssen keine speziellen Server verwendet werden, sondern es können WWW-Server mit Java-Servlet-Unterstützung eingesetzt werden. Im Gegensatz zu einer Konfiguration, die aus WWW-Server und spezialisierten Präsentationsservern besteht, ist dieser Aufbau einfacher wartbar.

Im Zusammenhang mit dem Präsentationsagenten Persona sind drei unterschiedliche Request-Klassen an den Server zu unterscheiden:

- Abruf der Startseite bzw. der Startpräsentation durch den Client vom Server. Der Server stellt eine HTML-Seite mit einem eingebetteten Persona-Applet zur Verfügung. Außerdem generiert der Server ein Persona-Skript inklusive der zugehörigen Daten für die Startpräsentation.

- Nachladen eines neuen Skripts für die Persona-Engine. Dabei erhalten der Persona-Generator und auch eventuell das Präsentationssystem den Auftrag eine neue Präsentationsaufgabe zu lösen. Das resultierende Skript wird mittels HTTP an die Persona-Engine übermittelt und entweder an das laufende Skript angefügt oder das aktuelle Skript wird von dem neuen Skript ersetzt. Das Nachladen des Skriptes kann auf der Seite der Persona-Engine als eigenständiger Thread erfolgen, so dass eine laufende Präsentation durch das Nachladen nicht unterbrochen wird.

- Datenrequests. Anforderung von Präsentationsdaten durch den Client (Persona-Bitmaps, Bilder, Audiodaten, Textdaten usw.)

Der im Persona-System eingesetzte WWW-Server ist in der Lage, diese Requests auszuführen. Da bei dieser Konzeption das reine Ausliefern der Daten und die Verarbeitung von Daten im WWW-Server zusammengefasst sind, werden keine weiteren Server-Prozesse neben dem WWW-Server gebraucht.

6.2.2 Verteilung der Last auf mehrere Präsentationsserver

In WWW-Applikationen ist es üblich, dass viele Clients gleichzeitig auf einen WWW-Server zugreifen und neben statischen Dokumenten auch dynamische Dokumente anfordern. Dadurch wird auf dem Server eine hohe Rechen- und I/O-Last erzeugt. Um die Skalierbarkeit für diese Anwendungen zu erhalten bietet sich ein mehrstufiges Konzept an:

- Einsatz mehrerer WWW-Server, dynamische Zuweisung vom Hauptserver
- Trennung statischer Seiten und dynamisch generierter Seiten, Speicherung auf getrennten Servern
- Verteilung der Last auf mehrere Threads innerhalb des Servers
- Dynamische Verteilung der Generierungsjobs auf spezialisierte Präsentationsserver

Dabei entsprechen die ersten zwei Möglichkeiten der gängigen Vorgehensweise bei WWW-Server-Betreibern (z.B. normale Netscape-Seiten kommen von www.netscape.com, dynamische Seiten kommen von cgi.netscape.com).

Neben diesen gängigen Möglichkeiten der Lastenverteilung kann man im Fall der dynamischen Generierung von HTML-Dokumenten bzw. Persona-Scripten weiter optimieren. Viele WWW-Server bieten die Möglichkeit, neben der ineffizienten Verwendung von CGI-Skripten (sie starten jeweils neue Prozesse) sogenannte Java-Servlets [Sun Microsystems, 1999] einzusetzen. Ein Beispiel für einen solchen Server ist der Apache-Server [Apache Software Foundation, 1999]. Ein Servlet kann wie ein normales CGI-Skript angesprochen werden, wobei die eventuell vorhandenen Parameter wie bei CGI-Skripten übergeben werden. Allerdings entsprechen innerhalb des Servers die Servlets normalen Java-Klassen. Entgegen dem CGI-Mechanismus wird für Servlets kein neuer Prozess gestartet, sondern das Servlet wird innerhalb des Speicherbereichs des WWW-Servers als neuer Java-Thread ausgeführt. Das Starten eines Threads verläuft ungleich schneller als der Start eines neues Prozesses, dadurch kann eine höhere Effizienz erzielt werden. Läuft der WWW-Server auf einer Multiprozessor-Maschine, so werden die Threads vom Java-Laufzeitsystem auf die einzelnen Prozessoren verteilt, ohne dass der Programmierer dafür besondere Vorkehrungen treffen muss. Durch Verwendung von Threads können einerseits gleichzeitig mehrere Präsentationen generiert werden. Das ist bei dem Einsatz in WWW-Servern wichtig, denn es kann vorkommen, dass viele Benutzer gleichzeitig auf den Server

zugreifen (Abbildung 6.5). Andererseits lässt sich auch der eigentliche Generierungsvorgang weiter parallelisieren, was allerdings in der aktuellen Version noch nicht implementiert ist.

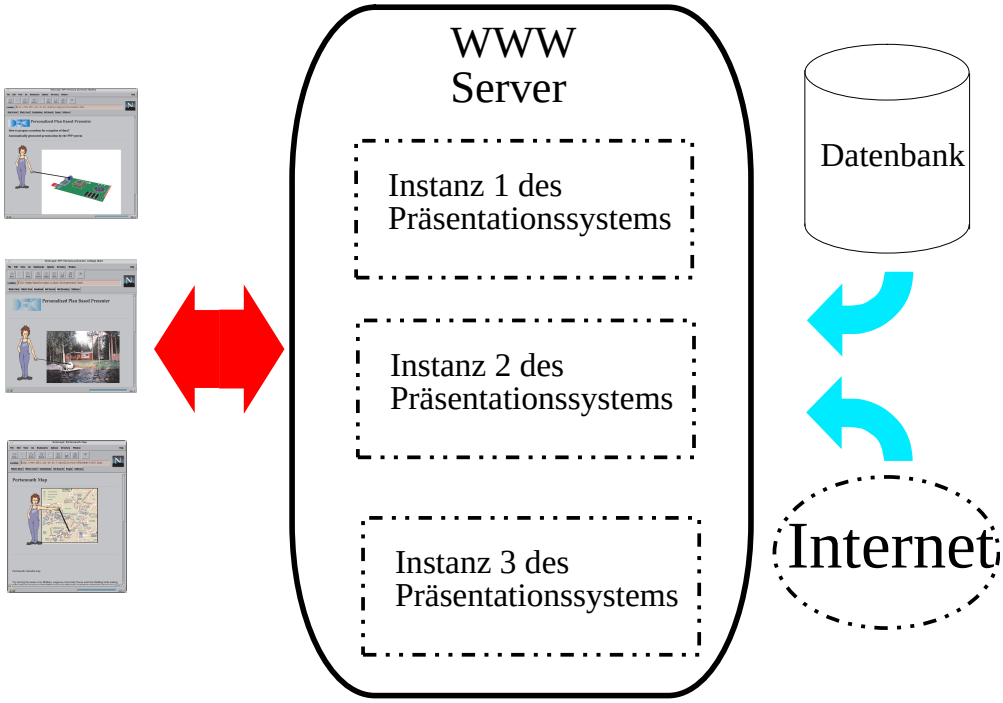


Abbildung 6.5: Verteilung der Rechenlast über mehrere Threads innerhalb des WWW-Servers

Zusätzlich können die einzelnen Präsentationsaufgaben auf mehrere Rechner, die Präsentationsserver, verteilt werden. Hierbei bestimmt der Hauptserver dynamisch einen Präsentationsserver, der zu diesem Zeitpunkt am wenigsten belastet ist. Dieser Präsentationsserver erhält vom Hauptserver alle Generierungsparamter und liefert die fertige Präsentation an den Hauptserver zurück (Abbildung 6.6).

Zur Kommunikation zwischen Hauptserver und den einzelnen Generierungsservern kann entweder das HTTP-Protokoll oder normale Socket- bzw. RPC-Kommunikation verwendet werden. Bei diesem Aufbau ist auch die Spezialisierung einzelner Server auf bestimmte Teilaufgaben möglich.

6.2.3 Integration des Persona-Applets in HTML-Dokumente

Eine WWW-Präsentation unter Verwendung des Präsentationasagenten Persona besteht aus HTML-Dokumenten, in die neben anderen Objekten auch Persona-Applets eingefügt sind. Soll im Laufe der Präsentation der in HTML verfasste Anteil der Seite ausgetauscht werden, so bietet sich eine Frame-basierte Lösung an. Während das Persona-Applet einen Frame belegt, können die anderen Frames für die Ausgabe von HTML-Dokumenten genutzt werden. Durch diese Vorgehensweise lassen sich die HTML-Anteile dynamisch im Laufe der Prä-

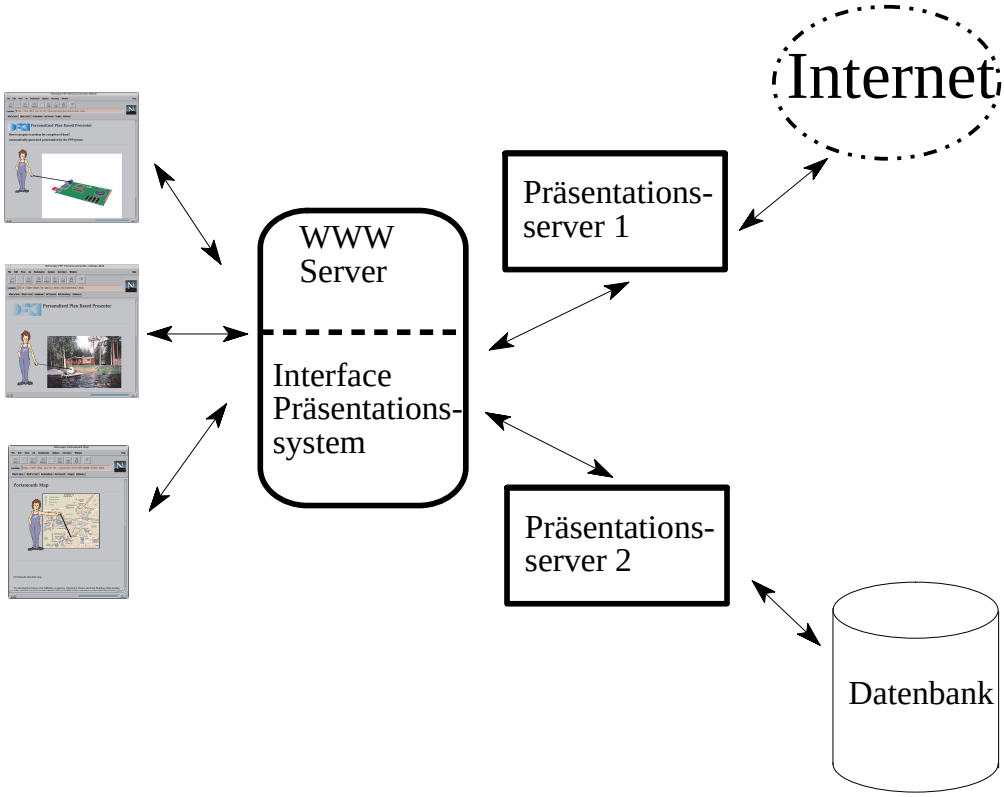


Abbildung 6.6: Verteilung der Rechenlast auf mehrere Präsentationsserver

sentation austauschen und das Persona-Applet bleibt unverändert auf der Seite stehen. Dabei kann das Austauschen eines Frameinhaltes über das Persona-Skript veranlasst werden. Würde man eine komplett neue Seite mit dem Persona-Applet nachladen, so würde sich für den Betrachter keine fließende Präsentation ergeben, da die Ladezeit des Applets den Fluss unterbrechen würde (Abbildung 6.7).

Trotzdem kann es unter grafischen Aspekten sinnvoll sein, den Rahmen um die einzelnen Frames unsichtbar zu machen, um somit die Illusion einer einheitlichen Seite zu erwecken. Auf diese Weise wird der Eindruck verstärkt, dass Persona kein Fremdkörper ist, sondern in die Seite voll integriert ist. Abbildung 6.8 zeigt als Beispiel eine Ausgabe des AIA-Systems (s. Abschnitt 7.2), bei der diese Technik eingesetzt wurde.

Bestimmte Einschränkungen (z.B. Persona kann nicht auf Elemente der HTML-Seite zeigen) lassen sich nur durch Integration dieser Elemente in das Persona-Applet umgehen (analog zu den VisualisationGadgets (Abschnitt 5.2)). Dadurch gewinnt Persona die vollständige Kontrolle über die dargestellte Information. Folgende Implementation bietet sich an, wenn auf Elemente im HTML-Code gezeigt werden soll: Das Applet wird unmittelbar neben das Element platziert und Persona führt eine entsprechende Geste aus (Abbildung 6.9).

Über Ansteuerung von JavaScript-Funktionen aus dem Persona-Skript heraus lassen sich in manchen WWW-Browsern auch dynamisch HTML-Elemente

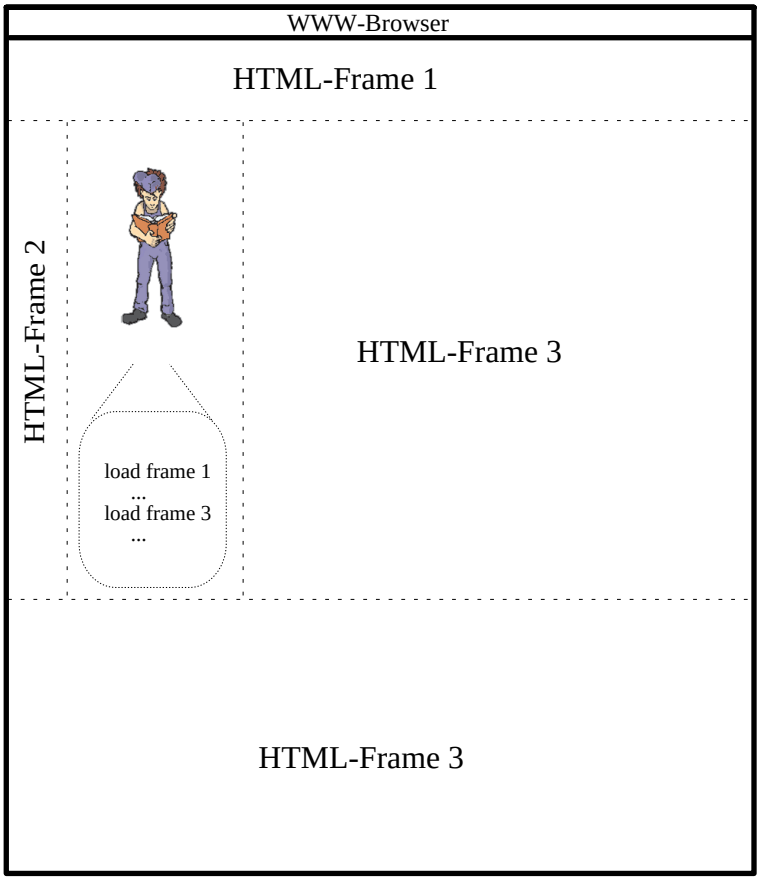


Abbildung 6.7: Austauschen des Frameinhalts über das Persona-Skript

modifizieren (z.B. Bilder auf der Seite austauschen).

Abbildung 6.8: Beispielpräsentation aus dem AIA-System

Abbildung 6.9: Persona zeigt auf HTML-Text

Kapitel 7

Anwendungen

Das folgende Kapitel beschreibt einige Projekte, die bereits unter Einbeziehung von Persona realisiert wurden. Außerdem werden weitere Anwendungssysteme vorgeschlagen.

7.1 Das System PPP

Das System PPP (Personalized Plan Based Presenter) [André et al., 1997] ist ein multimediales Desktop-Präsentationssystem, das einem Benutzer vorwiegend technisch orientierte Beschreibungen liefern kann. Die Präsentationselemente, wie Texte oder Grafiken, werden von dem Präsentationsplaner PrePlan [André, 1995] zu einer Gesamtpräsentation kombiniert. Die präsentierten Daten werden von speziellen Generatoren aus Wissensbasen generiert. Beispiele für solche Generatoren sind

- ein Grafikgenerator, der aus 3D-Objektbeschreibungen (z.B. annotierte Drahtrahmenmodelle) geeignete Abbildungen für die jeweilige Präsentation erstellt. Dabei umfasst der Generierungsvorgang z.B. die Wahl des Beobachterstandpunkts, die Generierung von Explosionszeichnungen usw.
- Layoutkomponente: Layout von Dokumenten, d.h. Anordnung der Komponenten eines Dokumentes auf den Seiten (Window-Manager)
- Text-Generierungskomponente: Generierung natürlichsprachlichen Textes aus der Datenbasis
- Annotierungskomponente: Annotation von Objekten mittels Pfeilen und Beschriftungen
- Persona-Generator: Generator für die Ansteuerung des Präsentationsagenten Persona

Als Ausgabemedium werden von PPP einzelne Windows des X11-Systems genutzt. Der Präsentationsagent PPP-Persona, der die Aufgabe hat, den Benutzer durch die Präsentation zu führen, Erläuterungen abzugeben und als Ansprechpartner zur Verfügung zu stehen, kann sich frei auf dem Bildschirm

bewegen, wobei er auch auf den Window-Manager zugreifen kann, um die Eigenschaften (wie Positon oder Größe) der gezeigten Fenster zu manipulieren.

PPP plant und generiert Texte und Grafiken, ausgehend von einer Wissensbasis, die z.B. die Drahtrahmenmodelle der zu präsentierenden Objekte enthält, unter Berücksichtigung verschiedener Präsentationsparameter (Sprache, Benutzervorwissen, Dokumenttyp usw.). Das System PPP muss, da es im Gegensatz zu dem Vorgängersystem WIP [André et al., 1993] Online-Präsentationen erzeugt, zusätzlich die zeitliche Koordination der einzelnen Präsentationsschritte beachten.

Eine Beispieldomäne des Systems PPP ist die Generierung und Präsentation von technischen Bedienungsanleitungen. Abbildung 3.13 aus Abschnitt 3.2 zeigt PPP-Persona bei der Präsentation einer Platine eines Modems. Zuerst hat das System ein Window mit der Grafik generiert. Nach dem Aufbau der Grafik muss sich PPP-Persona einen geeigneten Platz in der Nähe des Windows suchen und kann nun mit der Beschreibung der einzelnen Komponenten beginnen. Während bei gedruckten Dokumentationen die einzelnen Komponenten oft durch Pfeile und Text-Strings annotiert werden (wobei das PPP-System diese Variante ebenfalls anbietet), werden im PPP-System die einzelnen Komponenten darüber hinaus durch Zeigegesten und durch Sprachausgabe (mittels eines Sprach-Synthesizers) über PPP-Persona beschrieben.

Bei dieser dynamischen Annotationsart kann das System die Geschwindigkeit, mit der die einzelnen Objekte eingeführt werden, bestimmen.

PPP-Persona dient in der Rolle des personifizierten Präsentationssystems auch als Ansprechpartner für den Benutzer. In PPP wird diese Möglichkeit dadurch ausgenutzt, dass beim Anklicken der Persona das PPP-System-Menü erscheint, so dass der Benutzer dem System Anweisungen geben kann (Abbildung 7.1).

Für PPP wird die X11-basierte Version des Persona-Servers verwendet, Abschnitt 6.1. Der Persona-Server kommuniziert über RPC (remote procedure call) mit dem Präsentationssystem bzw. mit dem Präsentationsplaner PrePlan von PPP. Neben der Kommunikation mit dem Präsentationsplaner muss Persona mit dem Constraint-basierten Window-Management-System [Graf und Neurohr, 1995], das die Positionierung der Ausgabefenster in PPP übernimmt, kommunizieren, um Informationen über das aktuelle Bildschirmlayout zu erhalten.

Der Persona-Server kommuniziert über sein Plattform-Interface mit dem Betriebssystem, dem Window-System und externen Programmen wie Sprach-Synthesizer und Audioplayer. Umgekehrt werden Events (z.B. Mausklicks, Refresh-Events) dem Persona-Server über das Plattforminterface übermittelt. Die internen Komponenten des Persona-Servers umfassen einen Verhaltensmonitor, einen Event-Handler und einen Gesten-Konstruktor.

Der Event-Handler empfängt über das Plattform-Interface Events und prüft, in welcher Weise Persona auf diesen Event reagieren muss. Falls ein Event empfangen wird, für den eine besondere Reaktion von Persona definiert ist (z.B. Refresh-Event, Mausklick des Benutzers auf Persona), wird der entsprechende Auftrag an den Verhaltensmonitor erteilt.

Der Persona-Server muss sicherstellen, dass die Zeitrestriktionen, die ihm vom Präsentationsplaner gesetzt werden, so gut wie möglich eingehalten werden

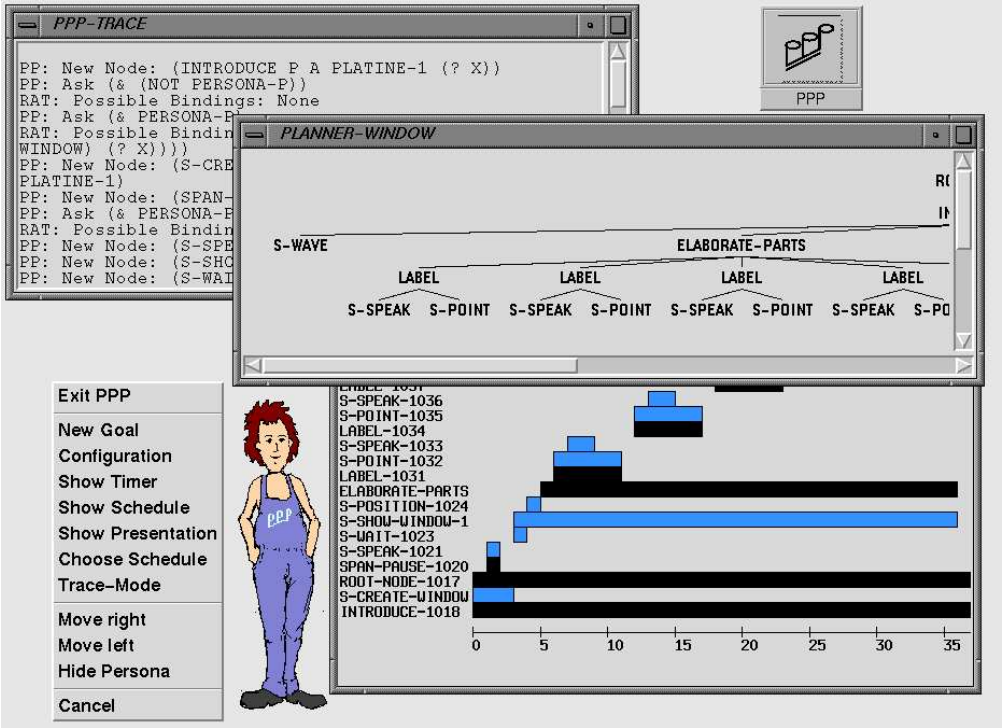


Abbildung 7.1: PPP-Persona zeigt das PPP-Systemmenü

(Bei der Ausgabe eines Audiofiles wird ein externer Audioplayer eingesetzt, der über den Systemaufruf *exec* aktiviert wird. Falls der Audioprozess mehr Zeit benötigt, als ihm zusteht, wird er mittels *kill* abgebrochen.).

Da die Programmierung des Persona-Servers auf Basis des X11-Systems unter Verwendung der Programmiersprache C erfolgen musste, ist der Persona-Server von der Betriebssystemplattform abhängig. Es existieren momentan Implementationen des Persona-Servers für Sun Sparc (Solaris) und für Silicon Graphics (Irix) Workstations.

7.1.1 Planung von Persona-Aktionen in PPP

Die Generierung von Multimedia-Präsentationen wird in dem System PPP als Planungsproblem aufgefasst. Ausgehend von einem kommunikativen Ziel erzeugt ein Präsentationsplaner einen Präsentationsplan, dessen primitive Aktionen an die einzelnen Generatoren, wie Grafikgenerator, Textgenerator usw., weitergereicht werden [Rist et al., 1997].

Die Ansteuerung der Abspielkomponente des Präsentationsagenten Persona erfolgt über eine Kaskade von Planungs- und Generierungskomponenten. Bei jedem Schritt wird ein komplexes Präsentationsziel weiter verfeinert und immer weiter in primitive Einzelaktionen zerlegt, die schließlich von der Abspielkomponente direkt ausgeführt werden können.

Die Planung der Gesamtpräsentation erfolgt mit Hilfe des Präsentationsplaners Preplan [André, 1995]. PrePlan zerlegt ein vorgegebenes Präsentations-

ziel in für ihn primitive Präsentationsschritte. Das Planungsverfahren verwendet Präsentationsstrategien, die entsprechend einer Wissensbasis ausgewählt und instantiiert werden. Dabei lässt sich die Menge der Präsentationsstrategien in domänenunabhängige und in domänenabhängige Präsentationsstrategien unterteilen. Die domänenunabhängigen Strategien beinhalten allgemeines Präsentationswissen, das bei jeder Präsentation eingesetzt werden kann. Die für die Planung erforderliche Wissensbasis kann von Hand gefüllt werden, sie kann aber auch teilweise mit der Hilfe von Informationsagenten [Bauer und Dengler, 1999; Bedersdorfer, in Vorbereitung] aufgefüllt werden.

Der Präsentationsplaner generiert ausgehend von der deklarativen Spezifikation der Präsentation, über die Präsentationsstrategien sowie die Wissensbasis einen Präsentationsplan, der alle Elemente der Präsentation umfasst. Innerhalb des gesamten Präsentationsplanes gibt es Subpläne für die Ansteuerung des Präsentationsagenten Persona. Diese Subpläne entsprechen Teilbäume des gesamten durch den Planer generierten Planungsbaumes. Dabei entsprechen die Blätter Aktionen, die aus Sicht des Präsentationsplaners nicht weiter zerlegbar sind.

Ein Beispiel für ein kommunikatives Ziel ist z.B. die Aufgabe, dem Benutzer eine Modemplatine zu erläutern: (Introduce System User circuit-board-1 ?window). Wird der Präsentationsplaner auf ein solches Ziel angesetzt, sucht er nach einer geeigneten Präsentationsstrategie. Im Beispiel wäre folgende Strategie anwendbar:

(S1) **Header:** (Introduce System User ?object ?window)
Inferiors:
((A1 (Create-Graphics P A ?object ?window))
(A2 (S-Show-Window System User ?window ?object))
(A3 (S-Wait System User))
(A4 (S-Position Persona User))
(A5 (Elaborate-Parts Persona User ?object ?window))
Qualitative:
((A1 (before) A2) (A3 (starts) A2) (A3 (meets) A4) (A2 (equals) A5)
Metric:
((10 $\leq$ Duration A2 $\leq$ 40) (1 $\leq$ Duration A4 $\leq$ 2))
Start: A1

Gemäß dieser Strategie wird zunächst eine Grafik für die Modemplatine generiert. Dann wird ein Window geöffnet und nach einer kurzen Pause begibt sich Persona in eine geeignete Zeige-Position. Anschließend werden die Platinenkomponenten dem Betrachter erläutert. Während S-Show-Window, S-Wait, S-Position aus der Sicht des Planers primitive Aktionen sind, die unmittelbar an die Generatoren (Window-Management-System bzw. Persona-Server) weitergereicht werden, sind Create-Graphics und Elaborate-Parts komplexe Operationen, die vom Planer weiter zerlegt werden. Die weitere Zerlegung von Elaborate-Parts führt zu Zeigegesten und Sprechakten. Die Zeigegesten werden vom Persona-Server ausgeführt. Da das System über eine explizite Repräsentation der dargestellten Objekte verfügt, kann es die Position der abgebildeten Objekte einfach bestimmen und zur Generierung von Zeigegesten an den Persona-Server senden. Der vom Sprach-Synthesizer auszugebende Text wird

mittels eines Textgenerators [Harbusch et al., 1991] automatisch erzeugt. Analog führt die Zerlegung des Teilzieles Create-Graphics zu einem Aufruf an den Grafikgenerator [Rist, 1995; Müller, 1994], der eine adäquate Abbildung der Modemplatine erstellt.

Das Ergebnis des Planungsprozesses ist ein Präsentationsplan, den man als Baum visualisieren kann, Abbildung 7.2. Dabei entsprechen die Blätter des Planungsbaumes den für den Planer primitiven Aktionen, die an weiter spezialisierte Generatoren weitergeleitet werden.

Neben der eigentlichen Dekomposition des Präsentationszieles in seine primtiven Komponenten erfolgt die zeitliche Koordination der einzelnen Aktionen. Die zeitlichen Constraints werden in den Präsentationsstrategien spezifiziert. Die Aktionen werden gemäß ihrer zeitlichen Spezifikation in einem Schedule angeordnet. Die zeitliche Spezifikation erfolgt mit Hilfe von qualitativen und metrischen Constraints [Kautz und Ladkin, 1991]. Neben der relativen Spezifikation von zeitlichen Constraints ist es möglich, die Zeitintervalle durch die Angabe ihrer minimalen bzw. maximalen Dauer weiter einzuschränken.

Durch die zeitliche Koordination der einzelnen Akte durch den Scheduler erhält man eine Folge von Aktionen, deren minimale bzw. maximal mögliche Zeitdauer bekannt ist.

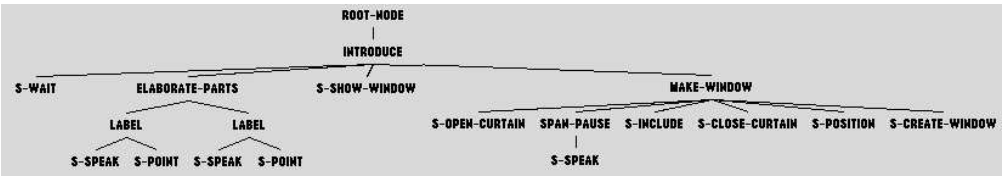


Abbildung 7.2: Beispiel eines Präsentationsplanes

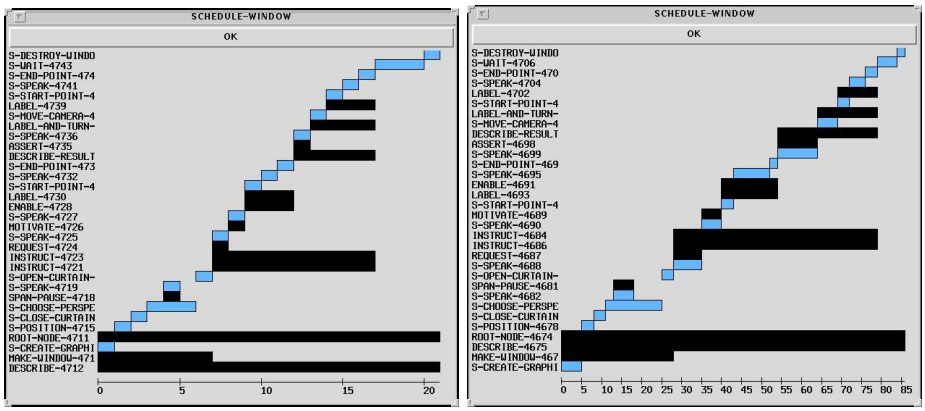


Abbildung 7.3: Vorausberechneter Schedule und angepasster Schedule

In der Generierungsphase wird die Folge der primitiven Aktionen abgearbeitet und die einzelnen Generatoren für die primitiven Aktionen unter Berücksichtigung der jeweils gültigen Zeitspezifikation angestoßen.

Dabei kann eine Überschreitung des festgelegten Zeitrahmens vorkommen, so dass die Ausführung der Präsentation fehlschlägt. In diesem Fall müsste eine

Neuplanung unter geänderten Voraussetzungen stattfinden.

Wie aus Abbildung 7.3 ersichtlich ist, ist das System in der Lage, die Präsentationsgeschwindigkeit an den Benutzer und an seine Ablaufumgebung anzupassen. Falls notwendig, wird der zunächst berechnete Schedule entsprechend den tatsächlichen Gegebenheiten angepasst. Es ist nicht in allen Fällen möglich, die Zeitrelationen für alle Aktionen im Voraus zu berechnen. Konfiguriert man Persona als Desktop-Präsentationsassistenten, so kann es vorkommen, dass in Folge hoher Systemlast die einzelnen Aktionen nicht innerhalb der vorgegebenen Zeit abgearbeitet werden können. In solchen Fällen wird die tatsächlich benötigte Zeit gestoppt und der Schedule an die neue Situation angepasst.

Dies kann notwendig werden, falls der Benutzer explizit eine Veränderung der Präsentationsgeschwindigkeit wünscht, bei Benutzerinteraktion oder falls die Ausführung einer Aktion mehr Zeit verbraucht. Ein Beispiel für eine solche Aktion ist das Ziel, das Persona veranlasst, sich in eine gute Zeigeposition neben dem Fenster zu bewegen. Falls sich das Fenster in unmittelbarer Nähe befindet, ist die Ausführung dieser Aktion weniger zeitaufwendig, steht Persona allerdings in größerer Entfernung zu diesem Fenster, so beansprucht die Bewegungsgeste mehr Zeit. Die Ablaufgeschwindigkeit einer Präsentation wird auch von der zur Verfügung stehenden Hardware bestimmt. Dieser Effekt muss ebenfalls bei der Berechung des Präsentationsschedules berücksichtigt werden. Stellt sich während der Präsentation beispielsweise heraus, dass die vorhandene Hardware für die geplante Präsentation zu langsam ist, kann durch Änderung des Präsentationsplanes (Ersetzung einer Teilaufgabe durch eine Resourcen sparendere Aufgabe) eine Präsentation trotzdem benutzerfreundlich gestaltet werden.

Die Persona-spezifischen primitiven Aktionen schreiben die Aktionsspezifikationen für den Persona-Generator in den Eingabepuffer des Generators. Bei der Auswertung des Eingabepuffers liest der Generator die Aktionsspezifikationen und zerlegt diese gemäß den definierten Persona-Regeln in primtive Persona-Anweisungen, die direkt von der Abspielkomponente verarbeitet werden können. Dabei werden die zeitlichen Vorgaben des Präsentationsplaners bzw. des Schedulers bei der Generierung berücksichtigt. Um den Schedule anzupassen, liefert der Generator bzw. der Player die für die jeweilige Aktion benötigte Zeit zurück.

7.2 Web-Persona und das AIA-System

Das AIA-System (Adaptive Communication Assistant for Effective Infobahn Access) [André et al., 1996] hat das Ziel, dem Benutzer langwieriges Suchen im Internet nach Informationen zu ersparen. Das AIA-System baut auf den Arbeiten zu dem PPP-System auf, allerdings werden neben lokalen Datenbasen wie in PPP auch Daten genutzt, die im Internet verfügbar sind. Das System sucht selbständig die benötigten Informationen im Internet oder aus lokalen Datenbanken und kombiniert die Resultate zu einer neuen Präsentation. Da die Präsentation für jeden Benutzer dynamisch neu generiert wird, kann sie auch einfacher auf die individuellen Bedürfnisse des Benutzers abgestimmt werden.

Der Zugang zu dem System wird über ein WWW-basiertes Interface er-

möglicht. Das Präsentationssystem erzeugt dynamische HTML-Dokumente für den jeweiligen Benutzer. Die Informationen auf der WWW-Seite werden dem Benutzer von der Java-Version des Präsentationsagenten präsentiert.

Ausgehend von den Informationen, die der Benutzer anforderte, ermittelt das System zunächst mit Hilfe verschiedener Suchverfahren die für den Aufbau der Präsentation benötigten Daten [Bauer und Dengler, 1999; Bedersdorfer, in Vorbereitung]. Diese können sowohl aus lokalen Datenbanken als auch von anderen WWW-Servern gewonnen werden. Je nach Grad der Struktur des Ursprungsdokuments fallen umfangreiche Analysen an, will man an die geforderten Informationen gelangen (z.B. Verfolgung von Links, Textanalyse, Grafikanalyse). Die von den Systemen gelieferten Ergebnisse werden in die Wissensbasis des Präsentationsplaners abgelegt. Der Präsentationsplaner erzeugt aus diesen Daten (zusammen mit allgemeinen Wissensbasen sowie einem Präsentationsziel) ein Dokument, welches zunächst in einem XML-Dialekt repräsentiert ist. Die Formatierung dieses Dokuments in die für aktuelle WWW-Browser verständliche Sprache HTML erfolgt durch Anwendung eines XSL-Stylesheets auf das von PrePlan erzeugte XML-Dokument. Durch diese Vorgehensweise kann von den Formatierungs- und Layoutproblemen während des Planungsvorganges abstrahiert werden. Man erhält auf diese Weise eine Trennung zwischen dem Dokumenteninhalt und der Formatierung des Dokuments. Preplan fügt in das von ihm generierte XML-Dokument spezielle Persona-Befehle (s. Abschnitt 4.2) ein, die das Verhalten des Präsentationsagenten steuern. Bevor das Dokument zu dem Benutzer ausgeliefert werden kann, werden diese speziellen Tags des Dokuments vom Persona-Generator evaluiert und gegen korrekten HTML-Code ersetzt. Der Persona-Generator erzeugt gleichzeitig alle Daten, die für den Präsentationsagenten während einer laufenden Präsentation benötigt werden (z.B. skalierte Bilder, Steuerungsskript, Audio-Daten mit synthetischer Sprache usw.).

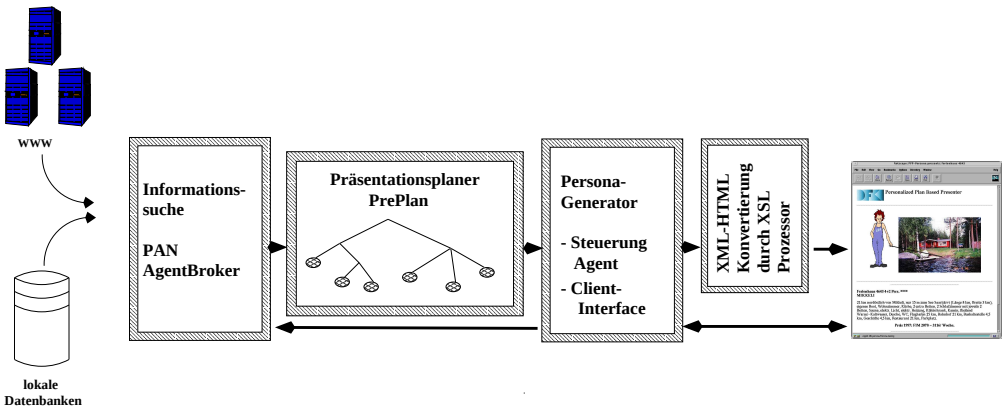


Abbildung 7.4: Architektur des Systems AIA

Der Präsentationsagent repräsentiert grafisch das vollständige System. Er ist der Ansprechpartner des Benutzers. Der Benutzer kommuniziert über den Agenten mit dem System, sei es über natürlichsprachliche Texteingabe oder per Menü oder Maus. Über den Agenten werden neue Präsentationsziele an das

System gestellt, wobei das System die Suchkomponenten zur Ermittlung der geforderten Daten einsetzt. Wird eine Suchanfrage vom Benutzer an den Präsentationsagenten gerichtet, so werden zunächst die benötigten Informationen im WWW bzw. in Datenbanken gesucht und in der Wissensbasis von PrePlan angelegt. Aus diesen Daten wird nun eine neue Präsentation erzeugt.

In AIA gibt es zwei mögliche Kommunikationsformen zwischen dem Präsentationsagenten und dem Präsentationssystem. Zum einen kann eine neue HTML-Seite angefordert werden, wobei die erzeugte Seite entweder in einem anderen Frame erscheint oder die Seite mit Persona ersetzt. Andererseits kann aber auch nur ein neues Skript für Persona angefordert werden. In diesem Fall werden von Persona ausschließlich neue Aktionen ausgeführt, die HTML-Seite bleibt unverändert.

7.3 Persona Chat System

Anwendungen wie ein Chat-System oder andere CSCW-Anwendungen befähigen Benutzer, die räumlich getrennt voneinander sind, dazu, gemeinsam an einem Projekt zu arbeiten. Das System muss die Kommunikation und den Datenaustausch zwischen den Arbeitsgruppen sicherstellen. Durch diese Anwendungen werden virtuelle Räume geschaffen, in denen die Benutzer Daten austauschen können.

Bei dem hier vorgestellten Einsatzgebiet dient Persona als Avatar, also zur Repräsentation einer realen Person durch eine computeranimierte Figur. Im WWW ist Chat eine beliebte Einrichtung. Ein Chat-System (z.B. IRC, Internet Relay Chat) erlaubt es, mit anderen Computerbenutzern Online-Diskussionen zu führen. Eventuell können auch Dateien (z.B. Bilder, Texte, Audio) ausgetauscht werden. Eine erweiterte Form bilden Anwendungen des CSCW (Computer Supported Cooperative Work). Die Benutzer dieser Anwendungen sind in der Lage, gemeinsam ein Programm zu bedienen, wobei die Kommunikation zwischen den Benutzern über ein Netzwerk stattfindet. CSCW Programme bieten die Möglichkeit, dass mehrere Benutzer gleichzeitig eine Benutzeroberfläche bedienen und alle Benutzer auch die Eingaben und Manipulationen der anderen Benuzter sehen. Oft werden die Systembenutzer auch visuell per Live-Video-Übertragung sichtbar gemacht. Auf diese Weise können die Benutzer gleichzeitig sowohl die beteiligten Partner als auch die Interaktionen, die die anderen Benutzer mit der Applikation tätigen, beobachten.

In solchen Umgebungen können animierte Agenten die Rollen der einzelnen Gesprächsteilnehmer übernehmen und die Teilnehmer in dem virtuellen Konferenzraum vertreten. Dabei werden die Agenten von ihren jeweiligen Besitzern gesteuert. Gesprächsbeiträge der einzelnen Teilnehmer werden zu den anderen Teilnehmern übertragen. Neben Gesprächstexten können das Gesten (z.B. Zeigegesten, Begrüßungsgesten), aber auch Emotionsäußerungen sein.

Zur Visualisierung der Agenten bietet sich der Einsatz von digitalisiertem Videomaterial an, so dass die Authentität des Agenten verstärkt wird. Im Gegensatz zu einer echten Videokonferenz sind die übertragenen Datenmengen bedeutend geringer, da die einzelnen Videoframes immer wieder verwendet wer-

den können. Sobald ein bestimmter Anteil an Gestendaten übertragen wurde, werden in vielen Fällen nur Steuerkommandos übermittelt.

Neben einer reinen Übermittlungsfunktion könnte das Dialogsystem auch weitere Aufgaben übernehmen. Durch den Einsatz eines Präsentationssystems wie AIA [André et al., 1997] oder auch WIP [André et al., 1993] können den Präsentationsagenten auch komplexere Präsentationsaufgaben gestellt werden, die diese dann selbständig ausführen.

Darüber hinaus kann das System berücksichtigen, dass an der virtuellen Gesprächsrunde Leute beteiligt sind, die unterschiedliche Sprachen sprechen. In diesem Fall können durch den Einsatz von Übersetzungssystemen wie z.B. Verbmobil [Wahlster, 1993b] die Gesprächsbeiträge von jedem Gesprächsteilnehmer in seiner jeweiligen Sprache generiert werden.

In Abbildung 7.5 ist die Konzeption des Chat-Systems dargestellt. Als zentrale Komponente dient ein spezieller WWW-Server, der die Präsentationsaufgaben der Client-Rechner entgegennimmt und über integrierte Module (z.B. Präsentationssystem, Steuerungskomponente für animierte Agenten, Übersetzungskomponente) die geforderten Informationen liefert. Die generierten Daten werden von ihm an die einzelnen Präsentationsagenten verteilt.

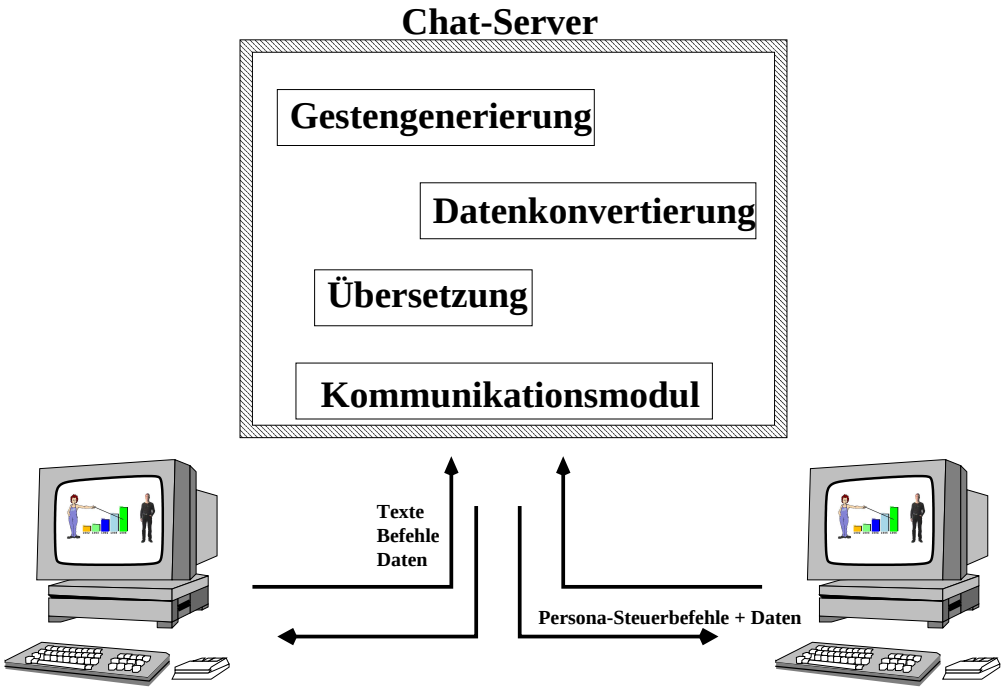


Abbildung 7.5: Konzeption des Chat/CSCW-Systems

Im Folgenden wird die Architektur eines Systems skizziert, bei dem ein Benutzer mit anderen Benutzern über das Internet kommunizieren kann. Ein Benutzer wird mit Hilfe des Präsentationsagenten Persona visualisiert. Er kann sich aus einer vorgegebenen Gruppe von Personas diejenige aussuchen, die ihn in der virtuellen Welt vertreten soll. Der Benutzer steuert die Bewegungen seines Avatars. Die Bewegungsanweisungen werden einerseits an die lokale Instanz

des Agenten übermittelt und andererseits an die Repräsentanten des Agenten bei den angemeldeten Benutzern. Dadurch dass stets nur Steuerungssequenzen und keine Bildsequenzen übrtragen werden wie bei Videoapplikationen, sind geringere Übertragungsraten notwendig.

Will ein Benutzer anderen Benutzern etwas mitteilen, so gibt er die Nachricht textuell ein (Spracherkennung ist ebenfalls denkbar). Die Information wird an seinen Repräsentanten bei den anderen Gesprächsteilnehmern übermittelt, der die Nachricht entweder über eine Sprechblase oder auch über synthetisierte Sprachausgabe wiedergeben kann.

Daten können den Gesprächsteilnehmen übermittelt werden, indem sie per File Upload zum Server geschickt werden, wo sie aufbereitet werden können (z.B. Konvertierung, Skalierung, Übersetzung). Im Anschluss werden die Daten an die Clients verteilt und mit Hilfe der grafischen Elemente von Persona, Abschnitt 5.2, visualisiert.

Ein Szenario für den Einsatz eines solchen Systems ist beispielsweise die Diskussion über die aktuelle Unternehmensbilanz in einem internationalen Unternehmen. Der Ablauf könnte in diesem Fall so aussehen:

1. Die Teilnehmer melden sich bei dem System an. Sie definieren ihren Avatar und bestimmen die von ihnen gewünschte Sprache.

2. Das System erstellt für jeden Diskussionsteilnehmer eine Instanz des Präsentationsagenten und verteilt die Repräsentanten an die Benutzer.

3. Die Teilnehmer geben ihren Gesprächsbeitrag ein, das System transferiert den optional übersetzten Text an die anderen Gesprächsteilnehmer.

4. Die Aktionen der jeweiligen Agenten werden von ihrem Besitzer gesteuert.

5. Die Bilanzdaten werden von einem Gesprächsteilnehmer in das System eingespeist, das es bearbeitet und an die anderen Teilnehmer weiterleitet. Der Avatar des Benutzers stellt die Bilanz mittels seines Moduls zur Visualisierung statistischer Daten 5.2.4 dar. Die Benutzer (bzw. ihre Avatare) können auf einzelne Elemente der Grafik zeigen und Kommentare abgeben.

6. Die Teilnehmer beenden den Chat.

7.4 Modellierung von Gesprächsformen

Im Folgenden soll erläutert werden, wie mit Hilfe des Präsentaionsplaners PrePlan und des Persona-Genrators verschiedene Gesprächsformen, die aus dem Alltagsleben bekannt sind, simuliert werden. Als 1. Fallbeispiel dient die Generierung eines Vortrages (also im Wesentlichen eines Monologes). Als zweites Beispiel wird eine Diskussion modelliert. In diesem Fall sind mehrere Agenten in den Ablauf eingebunden und müssen vom Planer berücksichtigt werden.

7.4.1 Der Vortrag

Der Vortrag ist im Gegensatz zur Diskussion weitgehend ein Monolog. Der Vortragende erläutert einen Sachverhalt vor einem Publikum, das in den meisten Fällen zunächst nur zuhört. Während des größten Teils der Vortragszeit geht die Initiative vom Sprechenden aus. An viele Vorträge schließt sich allerdings eine Diskussion an, in der Fragen an den Redner gestellt werden. Erst jetzt werden die Zuhörer aktiv.

Normalerweise ist jeder Vortrag nach einem analogen Schema aufgebaut. Zunächst begrüßt der Redner die Zuhörer. Im Anschluss gibt er eine kurze Einführung in das Thema und stellt den Aufbau des Vortrages vor. Danach wird im Hauptteil der Vortrag entsprechend der vorgestellten Gliederung gehalten. Der Vortragende schließt den Vortrag mit einer knappen Zusammenfassung und eröffnet eventuell die Diskussion, in der die Zuhörer weitere Fragen an den Votragenden stellen bzw. unterschiedliche Standpunkte vertreten können. Nach Ende der letzten Wortmeldung schließt der Vortragende den Vortrag, indem er sich für die Aufmerksamkeit der Zuhörer bedankt und sich verabschiedet.

Will man nun einen Votrag mit einem Präsentationsagenten gestalten, so bietet es sich an, genau diesen Ablauf in Präsentationsstrategien umzusetzen.

In realen Vorträgen dienen oft Hilfsmittel wie Overhead-Projektoren, Dias oder Videos zur Unterstützung des Redners. Der Redner benutzt verschiedene Zeigewerkzeuge wie Zeigestock, Laserpointer oder einfach die Hand, um den Zuhörern des Vortrages seine Anschauungsmaterialien näher zu bringen und zu erläutern.

Der Präsentationsagent Persona kann ebenfalls durch den Einsatz von Visualization-Gadgets Anschauungsmaterial in den simulierten Vortrag einbauen. Er kann mit verschiedenen Zeigegesten die dargestellten Objekte annotieren.

Im Folgenden werden die Präsentationsstrategien für einen Vortrag entwickelt, den der Präsentationsagent in Deutsch halten soll. Als Ausgabemedium für den generierten Vortrag soll das World-Wide-Web dienen, daher müssen die nun beschriebenen Strategien HTML-Seiten generieren, die den Präsentationsagenten PPP-Persona als Java-Applet enthalten.

Die Anfangsstrategie

```
(Define-Plan-Operator
   :HEADER (A0 (Give-Talk ?theme))
   :INFERIORS (
               (A1 (Greet-Audience ?theme))
               (A2 (Give-Introduction ?theme))
               (A3 (Give-Overview-Of-Talk ?theme))
               (A4 (Main-Talk ?theme))
               (A5 (Make-Summary ?theme))
               (A6 (Start-Discussion ?theme))
               (A7 (End-Talk ?theme))
              )
)
```

spiegelt genau den beschriebenen Aufbau eines Standardvortrages wieder. Diese Hauptgliederungspunkte des Vortrages werden nun in primitivere Aktio-

nen zerlegt. Die Strategien Greet-Audience und End-Talk enthalten zusätzlich Aktionen für den Aufbau der HTML-Seite und die Konfiguration des Präsentationsagenten:

```
(Define-Plan-Operator
   :HEADER (A0 (Greet-Audience  ?theme))
   :CONSTRAINTS (*AND* ((BELP (SHORTTITLE ?theme
                                          ?shorttitle))
                        (BELP (LONGTITLE  ?theme
                                          ?longtitle))))
   :INFERIORS (
               (A1 (Generate-WWW-Lecture-Environment
                                          ?shorttitle))
               (A2 (Show-Text-Slide ("<center> <big>"
                    ?longtitle "</big> </center>")))
               (A3 (SPersona (<speak
                    text=\"Herzlich willkommen
                    zu dem Vortrag"
                    ?longtitle ".\"/>")))
              )
)

(Define-Plan-Operator
   :HEADER (A0 (End-Talk ?theme))
   :CONSTRAINTS (BELP (ENDSLIDETEXT ?theme ?text))
   :INFERIORS (
               (A1 (Show-Text-Slide ("<center> <big>"
                    ?text "</big> </center>")))
               (A2 (SPersona (<speak text=\"Vielen Dank
                    f"ur ihre Aufmerksamkeit.\"/>")))
               (A3 (Flush-WWW-Lecture-Environment))
              )
)
```

Dabei dienen die Ziele Generate-WWW-Lecture-Environment und Flush-WWW-Lecture-Environment zur Generierung einer Arbeitsumgebung für den Präsentationsagenten (Aufbau einer passenden HTML-Seite mit integriertem Persona-Applet). Innerhalb dieser beider Strategien muss also der PERSONA-Tag für den Persona-Generator erzeugt werden. Bei beiden Strategien wird analog zur Titel- und Endfolie eines realen Vortrags ein Textblock mit einem passenden Text gezeigt. Parallel wird mittels Sprachsynthesizer ein passender Text ausgegeben. Zusätzlich könnten an diesen Stellen noch ausschmückende Gesten (z.B. eine Verbeugungsgeste) eingefügt werden. Mit einer zufälligen Auswahl treffender Sätze (Abschitt 4.3.4) könnte der Text außerdem abwechslungsreicher gestaltet werden.

Auch das Planungsziel Give-Introduction kann in ähnlicher Art gelöst werden. Dazu legt man verschiedene Einführungstexte bzw. Bilder in der Planer-Datenbasis ab und lässt sie über den Präsentationsagenten darstellen.

Mit Hilfe des Planungsziels Give-Overview soll dem Betrachter ein Überblick über den Aufbau des Vortrages gegeben werden. Innerhalb des Überblicks werden die wichtigsten Themengebiete des Vortrages genannt und in einer Gliederung geordnet.

```
(Define-Plan-Operator
   :HEADER (A0 (Give-Overview  ?theme))
   :INFERIORS (
               (A1 (Start-Slide "txt"))
               (A2 (Collect-Main-Items-for-Slide ?theme
                                                 item0))
               (A3 (Flush-Slide))
              )
)

(Define-Plan-Operator
     :HEADER (A0 (Collect-Main-Items-for-Slide ?theme
                                               ?item))
     :CONSTRAINTS (*AND* (
                          (BELP (MAINITEM ?theme ?item))
                          (BELP (ITEMTITLE ?item ?title))
                          (BELP (DESCRIPTION ?item ?desc))
                          (BELP (NEXTITEM ?theme ?item
                                          ?nextitem))
                         )
                   )
     :INFERIORS (
                 (A1 (Add-Item-To-Slide ?item ?title))
                 (A2 (SPersona ("<DescribeTextItem id=\""
                       ?item "\" text=\"" ?desc "\"/>")))
                 (A3 (Collect-Main-Items-for-Slide ?theme
                       ?nextitem))
                )
)
```

Die beiden Aktionen Start-Slide und Flush-Slide bewirken, dass ein Textfenster im Persona-Applet geöffnet wird, in das die einzelnen Textstücke geschrieben werden. Collect-Main-Items-For-Slide filtert alle Hauptpunkte aus dem Vortragsmaterial heraus und erstellt eine Inhaltsliste innerhalb des erzeugten Textfensters. Der Präsentationsagent zeigt auf die einzelnen Listenelemente und liest den entsprechenden Text vor. Zu jeder Überschrift eines Abschnitts ist in der Wissensbasis ein erläuternder Text gespeichert, der von Persona während der Zeigegeste vorgelesen wird.

In analoger Vorgehensweise zu Give-Overview wird auch Make-Summery aufgebaut. Zu einzelnen Vortragspunkten kann man wichtige Ergebnisse ablegen, die durch den Planoperator Make-Summary in einer Übersicht zusammengefasst werden. Zusätzlich kann man ein Resumee des gesamten Vortrages in

der Datenbasis speichern und durch den Präsentationsagenten als Zusammenfassung des gesamten Vortrages wiedergeben lassen.

Der Hauptteil eines realen Vortrages besteht im Wesentlichen aus einer Folge von Fakten, die durch verbale und bildliche Erläuterungen verdeutlicht werden. Genau dieses Vorgehen wird durch den Planoperator Main-Talk simuliert. Mit dessen Hilfe werden alle Vortragspunkte sequentiell abgearbeitet und gemäß ihrer internen Struktur weiter über spezialisiertere Planoperatoren zerlegt. Dabei kann ein Vortragsabschnitt aus einem oder mehreren „Folien" bestehen, die durch VisualizationGadgets realisiert sind. Der Aufbau der einzelnen Folien wird durch die Präsentationsstrategien bzw. durch den Inhalt der Wissensbasis bestimmt. Der grundsätzliche Aufbau einer Präsentationsstrategie für eine „Folie" besteht darin, die Grafik darzustellen, einen einleitenden Satz auszugeben und danach die Grafik zu beschreiben bzw. Anmerkungen zur Grafik entsprechend der Wissensbasis zu verbalisieren. Dabei müssen die unterschiedlichen Grafiktypen (z.B. Text, 3D-Grafik oder Diagramme) unterschiedlich behandelt werden. Jeder Grafiktyp besitzt bestimmte Eigenschaften, die in den Strategien berücksichtigt werden müssen. Z.B. kann ein Objekt innerhalb eines 3D-Gadgets gedreht werden, so dass sich der Betrachter das Objekt von allen Seiten ansehen kann.

Der Planoperator Start-Discussion initialisiert die interaktive Komponenten. Er erzeugt in Zusammenarbeit mit dem Persona-Generator ein Script, das es dem Benutzer erlaubt, einzelne gezeigte Folien noch einmal zu präsentieren. In diesem Verlauf können auch sensitive Grafiken eingesetzt werden, so dass der Benutzer einzelne Elemente mit der Maus anklicken kann, wobei durch den Maus-Event neue Planungsziele, die zur weiteren Erläuterung des Objektes führen, getriggert werden. Je nach verwendeten Strategien kann der Präsentationsagent die Folien über ein Menü oder über eine Button-Leiste zur Auswahl anbieten. Durch Einsatz eines Spracherkenners (Abschnitt 5.4.6) können die entsprechenden Plaungsziele auch über natürliche gesprochene Sprache angesteuert werden. Die Auswertung des Planoperators Start-Discussion umfasst auch eine Erklärung seiner interaktiven Fähigkeiten, die der Benutzer ausschöpfen kann.

Eine andere Realisierung für den Planoperator Start-Discussion könnte in der Verwendung weiterer Präsentationsagenten bestehen, die zu den geäußerten Argumenten mögliche Fragen stellen bzw. Gegenargumente vorbringen könnten. In diesem Fall werden Strategien analog zu Abschnitt 7.4.2 eingesetzt und mögliche Anmerkungen des realen Betrachters durch das Präsentationssystem vorweggenommen und entsprechend den eigenen Zielen eingesetzt. Der reale Zuschauer wird auf diese Weise ebenfalls durch das System simuliert und stärker in das Geschehen einbezogen. Der Vortrag wird so lebendiger und objektiver.

7.4.2 Die Diskussion

Innerhalb einer Diskussion soll der Diskussionsgegenstand aus unterschiedlichen Sichtweisen heraus betrachtet werden. Diese sind einander gegenüberzustellen. Es gibt Diskussionen mit und ohne Diskussionsleiter. Im Folgenden soll zunächst die Modellierung einer Diskussion mit zwei Diskussionspartnern ohne Diskussionsleiter beschrieben werden. Auf analoge Weise kann die Erweiterung

auf mehrere Diskussionsteilnehmer oder auf Diskussionen mit einem Diskussionsleiter erfolgen. Natürlich muss klar sein, dass ein solcher Diskussionsverlauf nicht auf dem tatsächlichen Abwägen von Vor- und Nachteilen beruht, sondern der Diskussionsausgang, d.h. der Sieger der Diskussion, durch die Planungsstrategien vorab bestimmt wird. Das Planungssystem steuert den Verlauf der Diskussion, wobei es auch die Reaktionen bzw. die Diskussionsargumente beider Seiten generiert. Die einzelnen Standpunkte der Teilnehmer bzw. die Art ihres Auftretens können über Generierungsparameter, d.h. über Belegung einzelner Planungsvariablen definiert werden.

Wie in einer realen Diskussion tragen nicht nur die rein sachlichen Argumente zum Sieg innerhalb einer Diskussionsrunde bei, sondern auch Gesten und Mimik, welche von animierten Präsentationsagenten dargestellt werden können.

Ein Ziel des Diskutanten ist die Durchsetzung des eigenen Standpunktes, ein anderes die Erforschung der gegnerischen Meinung. Wenn möglich soll ein Konsens herbeigeführt werden.

Grundsätzlich gibt es bei der Modellierung einer Diskussionsrunde zwei Möglichkeiten, wie der Präsentationsplaner zur Generierung des Dialogs eingestzt werden kann. Entweder ein Präsentationsplaner steuert alle an der Diskussion beteiligten Agenten oder für jeden Agent existiert eine eigene Instanz des Präsentationsplaners. Im zweiten Fall bestünde die Realisiation der Diskussionsrunde in der Implementierung eines Multiagentensystems, bei dem jeder Teilnehmer und die Kommunikation zwischen den Teilnehmern explizit repräsentiert ist. Im Folgenden wird beschrieben, wie eine Diskussion mittels eines einzigen Präsentationsplaners realisiert werden kann. Diese Lösung wurde deshalb gewählt, weil das Gesamtsystem einen bestimmten Sachverhalt aus nur einer Sicht beschreiben soll. Der Einsatz mehrerer Agenten dient nur zur lebendigeren Präsentation und dazu, dass die Präsentation auf den Betrachter objektiver wirken soll. Das Resultat der Diskussion ist durch die Planungsstrategien und durch die Wissensbasis des Planers vorgegeben. Die Verdeutlichung eines Präsentationszieles lässt sich unter Verwendung eines einzigen Präsentationsplaners einfacher erreichen als bei der Nutzung mehrerer Präsentationsplaner, zumal der Kommunikationsaufwand (Initialisierung, Koordiniation der Gesprächspartner, Protokoll) geringer ist. Die gesamte Präsentation für alle Agenten kann vollständig und durchgehend geplant und realisiert werden.

Um eine erfolgreiche Diskussion zu erreichen, müssen die Gesprächspartner bestimmte Einstellungen bzw. Verhaltenweisen zeigen [Lucas, 1979]. Wichtig ist es z.B., eine Diskussion Schritt für Schritt zu führen. Zudem müssen die Gesprächsteilnehmer wechselseitig auf die Argumente der anderen eingehen, damit sich eine Diskussion entwickeln kann. Die einzelnen Redebeiträge sollten knapp und treffend abgefasst werden. Zu langes Reden einer Person kann leicht langweilig für den Zuhörer werden.

Bei einer größeren Diskussion kann ein Diskussionsleiter erforderlich werden, der dafür sorgt, dass die Diskussion geordnet abläuft.

Durch Definition der Plaungsstrategien legt man den Typ der einzelnen Diskussionsteilnehmer fest. Bei der Modellierung müssen unterschiedliche Charaktere berücksichtigt werden, wie z.B. (aus [Lucas, 1979]):

- der Streiter
- der Positive
- der Alleswisser
- der Redselige
- der Schüchterne
- der Ablehnende
- der Uninterssierte
- „der Höhergestellte “
- der Ausfrager

Die Diskussion lebt von der Argumentation und der Gegenargumentation. Bei einem Verkaufsgespräch treffen zwei sehr unterschiedliche Interessenlagen aufeinander. Der Verkäufer will auf jeden Fall seine Ware möglichst teuer verkaufen. Der Kunde möchte seinerseits die Ware möglichst günstig erwerben. Beide Gesprächspartner wenden zur Durchsetzung ihrer Ziele verschiedene rhetorische Techniken an. Verkäufer werden häufig in solchen Techniken besonders ausgebildet [Lucas, 1979]. Ein guter Verkäufer wird nicht versuchen, den Kunden totzureden, sondern er versucht, ihn durch geschickte Fragen auszuhorchen und seine Wünsche zu erfahren. Um bei dem Kunden das Gefühl zu wecken, man schenke ihm eine hohe Aufmerksamkeit, reagiert der Verkäufer auf eine Kundenäußerung mit sogenannten „Verstärkern “ (z.B. mit „ja, genau “ , „richtig “ , „da stimme ich voll zu “). Auch soll der Verkäufer auf Eigenheiten des Käufers eingehen, die dessen Selbstbild und Selbstdarstellung stützen (z.B. „Sie gehören zu den wenigen Kunden, die ... “).

Bei Diskussionen gibt es drei Typen von Argumentationsarten [Neumann, 1985]:

Inhaltliche Argumentation: Thesen und Argumente, die auf die jeweilige Sache hin ausgerichtet sind.

Taktische Argumentation: Aufbau der Diskussion, um den Gesprächspartner/Zuhörer zu beeinflussen.

Rollenbezogene Argumentation: Argumente werden entsprechend der jeweiligen Rolle der Gesprächspartner ausgewählt.

Auch die Art, wie Argumente im Verlauf einer Diskussion vorgetragen werden, kann ihren Ausgang beeinflussen.

Latente Argumentation: Unpersöhnliche, (scheinbar) wertneutrale Information über Sachverhalte

Deliberative Argumentation: Der Zuhörer erhält vom Redner einen Ratschlag.

Imperative Argumentation: Der Redner gibt dem Zuhörer eine Anweisung.

In [Lucas, 1979] wird beschrieben, wie die sogenannte 5-Satz-Methode umgesetzt werden kann. Diese Methode baut in fünf Schritten einen Redebeitrag auf. Im Folgenden sei

1. Grund für das Sprechen
2. Darstellung des Sachverhaltes
3. Formulierung des Zieles
4. Suchen von Wegen zum Ziel
5. Aufforderung zur Handlung

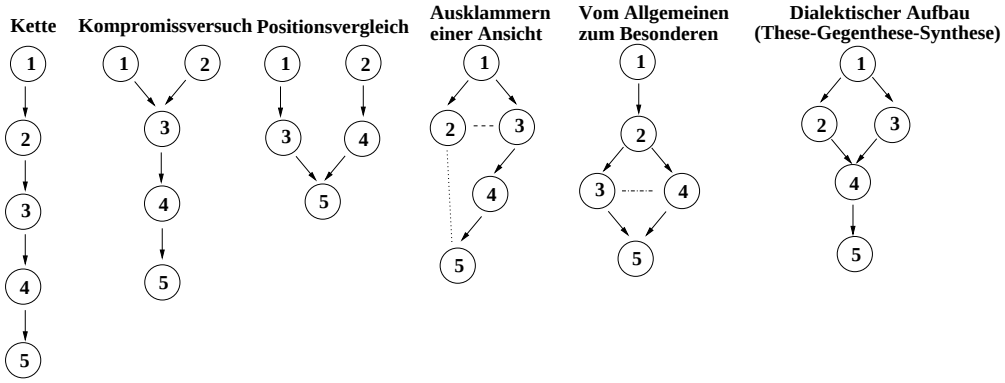


Abbildung 7.6: Verschiedene Möglichkeiten, einen Redebeitrag nach der 5-Satz-Methode zu strukturieren

Diese 5 Gesprächsschritte können auf unterschiedliche Weise kombiniert werden. Folgende Verknüpfungsmöglichkeiten sind gebräuchlich (Abbildung 7.6):

Kette: Die einzelnen Elemente sind zeitlich und logisch streng voneinander abhängig.

Kompromissversuch: Es werden zwei Standpunkte beschrieben. Im Anschluss wird ein möglicher Kompromissversuch dargestellt.

Positionsvergleich: Die Vorgehensweise beim Positionsvergleich ist, zunächst beide Standpunkte darzustellen. Im lezten Schritt findet dann der eigentliche Vergleich statt.

Ausklammern einer Ansicht: Bei dieser Vorgehensweise lenkt der Redner die Aufmerksamkeit auf einen anderen Aspekt. Der bisherige aktuelle Aspekt wird von ihm ausgeklammert: (z.B. 1: Wir reden schon eine Weile über ..., 2: Bislang dreht sich alles um ..., 3: Dabei wurde übersehen, dass ..., 4: Gerade das scheint mir besonders wichtig, weil ..., 5: Ich stelle den Antrag ...)

Vom Allgemeinem zum Besonderen: Zunächst wird der allgemeine Fall abgehandelt, dann wird vom allgemeinen Fall auf den vorliegenden besonderen Fall geschlossen.

Dialektischer Aufbau (These - Gegenthese - Synthese): Zunächst wird eine These vorgestellt, wobei zwei mögliche Ansichten dargelegt werden. Beide Ansichten werden verglichen und eine mögliche Synthese der beiden Aspekte hergestellt.

Neben dem logischen Aufbau eines Redebeitrages spielen auch andere rhetorische Techniken eine wichtige Rolle bei der Modellierung. Bei Diskussionen sind unterschiedliche Gegenargumentationstechniken gebräuchlich [Lucas, 1979]:

Abstreiten: Die Aussage des Diskussionsgegners wird abgestritten, er ist dann gezwungen, Beweise vorzulegen.

Bezweifeln: Durch Bezweifeln der Aussage des Gegeners kann dieser veranlasst werden, neue Argumente zu suchen, die ihn unter Umständen angreifbar machen.

Rückfrage: Durch Stellung einer Rückfrage kann Zeit gewonnen werden, die Antwort kann zur Schwächung der gegnerischen Verhandlungsposition genutzt werden.

Ja, aber ...: Diese Technik kann immer dann eingesetzt werden, wenn der Gegner nur eine Seite des Diskkussionsgegenstandes betrachtet (die für seine Verhandlungsposition dienlich ist).

Verzögerungstaktik: Unter Umständen ist es sinnvoll, nicht gleich auf ein Diskussionsargument zu antworten. Man kann sich den Punkt notieren und in Ruhe eine Gegenargumentation überlegen.

Schlagfertigkeit: Durch Schlagfertigkeit ist es manchmal möglch, einen Gegner beim kurzfristigen Verlassen seines Standpunktes zu treffen.

Rhetorische Frage: Durch eine Frage, die gleich von dem Fragenden beantwortet wird, soll der Gesprächsverlauf positiv für den Fragenden verändert werden.

Vorwegname von Argumenten: Durch Vorwegname von Gegenargumenten kann dem Gegner die Kontermöglichkeit genommen werden.

Ablenkung: Will ein Diskussionsteilnehmer zu einem Diskussionspunkt keine Stellungsnahme abgeben, so kann er versuchen, vom Thema abzulenken.

Für eine Diskussionsmodellierung kann man diese rhetorischen Techniken für den Aufbau von Diskussionsbeiträgen unmittelbar in Präsentationsstrategien umsetzen. Der Persona-Generator stellt Techniken zur Verfügung, um Dialoge zwischen Agenten zu synchronisieren (Abschnitt 4.8). Der Präsentationsplaner kann also unmittelbar die geplanten Redebeiträge an die generierte Ausgabe anfügen. Für die Implementation müssen zunächst die einzelnen Argumente zu

jedem Sachverhalt gesammelt werden. Die einzelnen Argumente und Gegenargumente werden wie bei der Beschreibung der 5-Satz-Methode in Präsentationsstrategien angeordnet. Im Verlauf der einzelnen Argumentationen werden die beschriebenen rhetorischen Techniken eingesetzt, die die Diskussion für den Betrachter interessanter machen sollen. Um die einzelnen Charaktertypen während der Diskussion zu modellieren, bietet es sich an, für einzelne Diskussionsbeiträge mehrere Alternativen zu formulieren, die den unterschiedlichen Charakteren angepasst sind, und den Betrachter die einzelnen Diskussionsteilnehmer entsprechend auswählen zu lassen. Die Auswahl der simulierten Diskussionsteilnehmer wird als Generierungsparameter in den Planungsprozess übernommen. Um die Präsentation abwechslungsreicher zu gestalten, ist es darüber hinaus sinnvoll, mehrere äquivalente Varianten zu definieren, aus denen der Planer oder auch der Persona-Generator zufällig eine Auswahl treffen kann.

Kapitel 8

Evaluation, Zusammenfassung und Ausblick

Das folgende Kapitel beschäftigt sich zunächst mit der Evaluierung des entwickelten Präsentationsagenten. Danach werden die erzielten Ergebnisse zusammengefasst. Zum Schluss werden mögliche Erweiterungen des Systems aufgezeigt.

8.1 Evaluation

In verschiedenen Arbeiten werden personifizierte Benutzerschnittstellen evaluiert. [Wexelblat, 1998] beschreibt ein Experiment, das versucht auf folgende Fragen eine Antwort zu finden:

- Wird ein anthropomorphisiertes Interface von Benutzern benutzerfreundlicher bewertet als ein Benutzerinterface mit Direktmanipulation ?
- Leidet die Effizienz bei der Ausübung der Aufgaben durch die Anthropomorphisierung ?

Zu diesem Zweck wurde ein WWW-basiserter Zugang zu einer Datenbank über Musiker eingerichtet. Eine Versuchsgruppe griff mit einem Standard-Interface auf die Datenbank zu, das Interface der anderen Versuchsgruppe enthielt eine Figur, die als anthropomorphisierte Repräsentation verschiedenener Agenten des darunterliegenden Informationssystems dient. Das Ergebnis ergab: Die Benutzer stuften das anthropomorphisierte Interface leicht benutzerfreundlicher ein als das Standardinterface. Auch gab es keine Einbußen in der Effizienz, wie oft befürchtet wird. Zusätzlich fanden die Benutzer das anthropomorphisierte Interface viel unterhaltsamer als das Standardinterface.

Die Hoffnung, dass die Benutzung eines animierten Schnittstellenagenten die Mensch-Maschine-Kommunikation verbessern möge, kann genauer gefasst werden, wenn man die möglichen Effekte auf den Benutzer klassifiziert [Dehn und van Mulken, in Druck]:

Positiver Systemeindruck: Ein mit einem animierten Agenten ausgestattetes System soll interessanter, unterhaltsamer oder informativer wirken als ein System ohne Agent.

Intelligenz:

- Menschenähnliche Figuren werden als intelligenter angesehen als andere geometrische Formen [King und Ohya, 1996]
- Menschenähnliche Agenten gelten als intelligenter als z.B. Agenten in Hundeform. Nach einer Interaktion mit den Agenten gibt es keinen Unterschied mehr in der Beurteilung (auch falls der Agent nicht visualisiert wird) [Koda und Maes, 1996]

Glaubhaftigkeit: Ein ausdrucksstarker Agent wird für glaubhafter gehalten als ein stummer Agent [Lester et al., 1997a].

Sympathie: Ein durch einen Agenten visualisiertes Programm erscheint dem Benutzer sympathischer als ohne Visualisierung [Koda und Maes, 1996]

Unterhaltungswert: Untersuchungen zeigen, dass eine Präsentation mit animierten Agenten als unterhaltsamer eingeschätzt wird als eine Präsentation ohne Agenten [Takeuchi und Naito, 1995; Koda und Maes, 1996; Lester et al., 1997a; van Mulken et al., 1998].

Nützlichkeit: Präsentationen mit einem animierten Agenten, der sich über Sprache, Gesten und Mimik verständlich machen kann, werden als hilfreicher empfunden als ohne einen solchen Agenten [Lester et al., 1997a; van Mulken et al., 1998; Cassell und Thórisson, in Druck].

Aufmerksamkeit: [Koda und Maes, 1996] und [van Mulken et al., 1998] stellten in Experimenten fest: Benutzer lassen sich nicht von einem animierten Agenten ablenken. Ein animierter Agent kann zu einer besseren Konzentrationsfähigkeit des Benutzers beitragen.

Erzielung eines wünschenswerteren Benutzerverhaltens: Möglicherweise beschäftigt sich ein Benutzer länger mit einem System, das einen Agenten verwendet, als mit einem System ohne Agenten.

Aufmerksamkeit: Benutzer verfolgen eine Präsentation mit animierten Agenten aufmerksamer als eine Präsentation ohne Agenten [Takeuchi und Naito, 1995].

Besseres Ergebnis: Der Benutzer erreicht ein insgesamt besseres Arbeitsergebnis durch ein System mit einem animierten Agenten statt ohne diesen. Beispielsweise erhält er ein tieferes Verständnis von der präsentierten Situation.

Problemlösung: Viele Untersuchungen zeigen keine signifikanten Unterschiede in der Problemlösungsfähigkeit von Benutzern von Systemen mit bzw. ohne animierte Agenten [Lester et al., 1997a; van Mulken et al., 1998]. Bei sehr komplexen Problemen kann ein expressiver Agent besser sein als ein wenig expressiver Agent.

Lernen: Bei reinen Gedächtnisaufgaben (Erinnerungsfähigkeit) sind keine Unterschiede feststellbar [Lester et al., 1997a; van Mulken et al., 1998]. Subjektiv wird aber oft der Lernstoff als einfacher empfunden.

Zur Analyse dieser Wirkungen wurden Experimente durchgeführt, die sich bezüglich des Versuchsaufbaus und der untersuchten Aspekte unterschieden:

Art der Animation des Agenten: z.B. 2D-cartoonartige Animationen, 3D-Darstellungen

Art des Vergleichs: z.B. Agent - System ohne Grafik, Agent - Symboldarstellung

Untersuchte Auswirkung: z.B. Bewertung durch Benutzer, Leistung des Benutzers

Aufgabenstellung: z.B. Pokerspiel, Betrachtung einer technischen Präsentation

8.1.1 Aufbau des Experiments

Das in [van Mulken et al., 1998] beschriebene Experiment wurde unter Verwendung des Präsentationsagenten PPP-Persona realisiert. Das Experiment bestand aus zwei Teilen. Im ersten Teil wurde den Versuchspersonen der Aufbau und die Funktionsweise eines Flaschenzuges erläutert (Abbildung 8.1). Dann wurden ihnen Verständnisfragen gestellt.

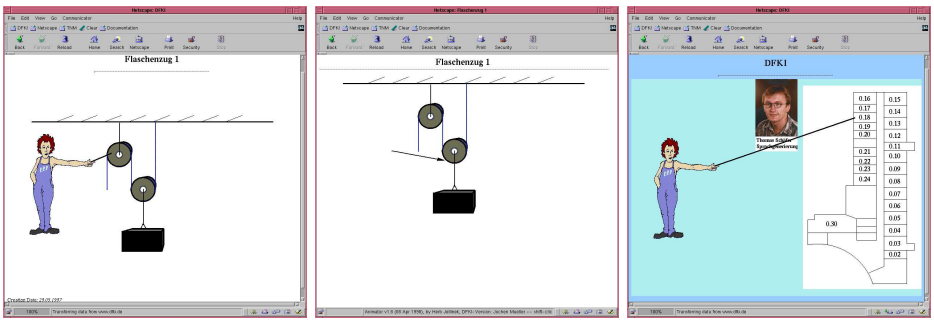


Abbildung 8.1: Experiment: Persona erklärt die Funktionsweise eines Flaschenzuges und stellt Mitarbeiter vor.

Im zweiten Teil des Experiments wurde die Gedächtnisleistung der Versuchspersonen geprüft. Ihnen wurden Mitarbeiter vorgestellt. Von jedem Mitarbeiter sollten sich die Versuchspersonen den Namen, das Arbeitszimmer und den Aufgabenbereich einprägen. Dieses Wissen wurde im Anschluss abgefragt. Schließlich wurden im Experiment noch allgemeine Fragen zum Experimentablauf und zu Persona gestellt. Zum Vergleich wurde das ganze Experiment auch in einer Konfiguration durchgeführt, bei der der Präsentationsagent bzw. seine Gesten durch eine Kombination aus Sprachausgabe und Pfeilen ersetzt wurde. Bei dem Experiment zeigte sich, Benutzer empfinden eine Aufgabenstellung subjektiv als leichter, wenn der Präsentationsagent aktiviert ist, erachten ihn als hilfreich und halten die Präsentation für unterhaltsamer. Die Wertung scheint von der Domäne abhängig zu sein. Bei der Flaschenzug-Domäne (also bei der Beschreibung eines technischen Sachverhaltes) wurde der Einsatz des Agenten positver bewertet als bei der Personaldomäne.

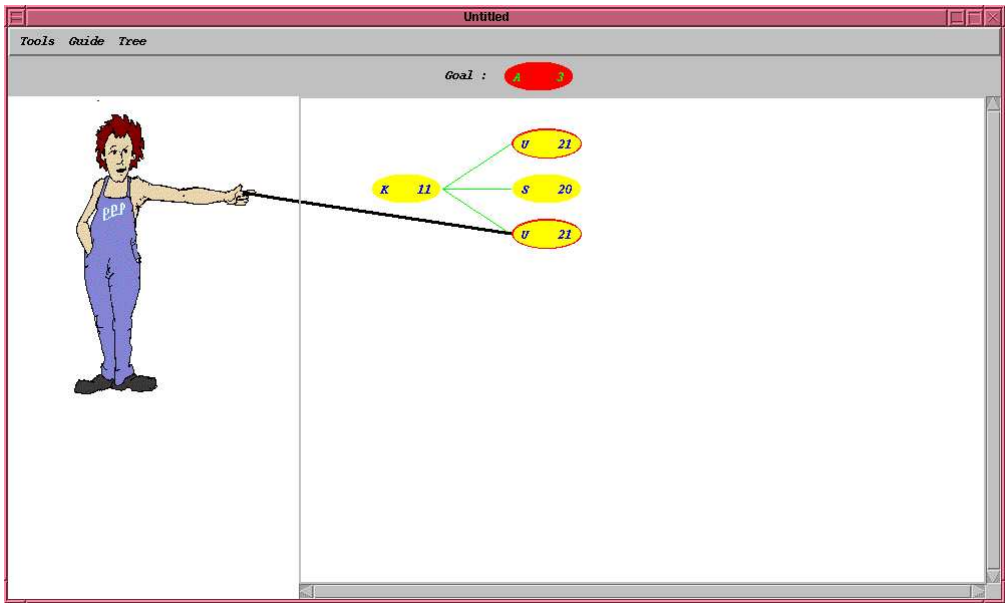


Abbildung 8.2: Experiment: Persona führt einen Benutzer durch einen dynamischen Graph

Ein weiteres Experiment wurde unter Verwendung von Persona durchgeführt. Ein Ziel des Experimentes war herauszufinden, ob ein Benutzer einem animierten Agenten mehr vetraut als einer rein textuellen Ausgabe bzw. einer Ausgabe mittels Sprachsynthesizer. Die Aufgabe des Benutzers bei diesem Experiment war die Navigation durch einen Graph, der dynamisch expandiert wurde [van Mulken et al., 1999]. Dem Benutzer wurden Hinweise gegeben, wie er am besten sein Ziel erreicht; manche Hinweise waren falsch, manche korrekt. Die Hilfstexte wurden entweder mittels Persona (Sprachausgabe, Zeigestock), textuell (mit Blinken des bezeichneten Knotens) oder per Sprachsynthese (ebenfalls mit Blinken des Knotens) übermittelt (Abbildung 8.2). Es konnte kein signifikanter Unterschied bei der Auswertung des Experiments zwischen den Konfigurationen festgestellt werden. Eventuell ist dies darauf zurückzuführen, dass es für die Benutzer klar war, dass Persona und auch die anderen Hilfen von dem gleichen Programm bzw. von einem Computer gesteuert wurden.

8.2 Erzielte Ergebnisse

Das System PPP-Persona stellt einen individuell auf die jeweilige Anwendung konfigurierbaren Präsentationsagenten zur Verfügung, der universell einsetzbar ist. Durch seine Konzeption lässt er sich einfach auf neue Anwendungen adaptieren. Er kann in ein komplexes planbasiertes Präsentationssystem eingebettet und von diesem angesteuert werden. Der Präsentationsagent ist in einer Version für das X-Window-System sowie in einer Java-Version verfügbar. Die Java-Version kann sowohl in Applets innerhalb eines WWW-Browsers als auch in Applikationen eingesetzt werden.

Durch die Verwendung der VisualizationGadgets (Diagramme, Bilder, Text) kann der Präsentationsagent mit den zu präsentierenden Daten interagieren. Die Menge der zur Verfügung stehenden Gadgettypen ist leicht erweiterbar. Alle Operationen auf den Gadgets lassen sich über das Persona-Skript steuern. Indem die ViusalizationGadgets eine Repräsentation ihrer Daten für den Präsentationsagenten zur Verfügung stellen, kann der Präsentationsagent diese Daten zur Generierung von Zeigegesten, sensitiven Grafiken oder auch für grafische Effekte (wie Blinken von Objekten) nutzen.

Der Persona-Generator erlaubt es auf hohem Abstraktionsniveau Gesten und Aktionen zu definieren. Diese Aktionen werden kontextabhängig in die jeweiligen primitiven Gesten übersetzt, die vom Persona-Player unmittelbar abgespielt werden können.

Das System Persona-Player, Persona-Generator und Präsentationsplaner kann verteilt eingesetzt werden. Während der Persona-Player innerhalb des WWW-Browsers läuft, wird der Persona-Generator bzw. das Präsentationssystem (Präsentationsplaner) auf dem WWW-Server installiert.

Für den Einsatz innerhalb von WWW-Applikationen wurden zur Steigerung der Datenübertragungsgeschwindigkeit verschiedene Techniken eingesetzt:

- Caching von Bild und Audiodaten auf Client-Seite: Bilder (Gestenbilder und Applikationsbilder) sowie Audiodateien (z.B. Soundclips, Sprachdaten) müssen nur einmal zum Client übertragen werden und können auch in neuen Kombinationen wiederverwendet werden.

- Der Persona-Player wird über ein Skript angesteuert, das ihm erlaubt, auch ohne Serverunterstützung Aktionen durchzuführen. Neue Skripte können im Hintergrund nachgeladen werden ohne die laufende Präsentation zu stören.

- Jedes Einzelbild einer Geste kann aus mehreren Bildelementen zusammengesetzt werden (statische und dynamische Bildelemente). Durch diese Technik ist es möglich, Bildbereiche, die in vielen Einzelbildern vorliegen, wiederzuverwenden und sie müssen daher nicht neu übertragen werden.

- Optimierte Bildübertragung: Es werden jeweils nur die Gestenbilder übertragen, die für das Skript benötigt werden.

Die primitiven Gesten werden über einen Editor definiert. Komplexe Gesten lassen sich über eine Sprache definieren und anwenden, die in XML-Dokumente einbettbar ist. Der Persona-Generator ist mit Hilfe des DocumentManagers in der Lage, XML bzw. HTML-Dokumente zu verarbeiten, so dass das Ergebnis auch in normalen Standard-WWW-Browsern lauffähig ist.

Das System lässt sich durch die Verwendung einer erweiterbaren Steuersprache innerhalb des Persona-Players einfach auf neue Anwendungsgebiete anpassen. Durch diese ist ebenfalls die dynamische und zeitliche Steuerung der Präsentationen möglich. Gleichzeitig bietet eine Skriptsprache auch die Möglichkeit, komplexere Präsentationen zu erstellen, da hier beispielsweise bedingte oder unbedingte Sprünge verwendet werden können (z.B. für Ruheaktionen).

Sprungbefehle ermöglichen den mehrmaligen Gebrauch von Skriptteilen im Laufe einer Präsentation (Unterprogramme), wodurch man die übertragene Datenmenge reduzieren kann.

Ein Präsentationsagent bietet zusätzliche Präsentationsmöglichkeiten an, die in vielen Anwendungsbereichen genutzt werden können.

Ein Ergebnis der Evaluierung war, dass das Verständnis einer Präsentation mit einem Präsentationsagenten subjektiv einfach leichter fällt als ohne einen Präsentationsagenten und deshalb die meisten Benutzer eine Präsentation mit Agenten einer Präsentation ohne Agenten vorziehen. Das Ergebnis scheint abhängig von der jeweiligen Domäne zu sein. Bei komplexen technischen Problemen war offenbar das Bedürfnis nach einem Präsentationsagenten größer als bei einer Gedächtnisaufgabe. In Systemen, wo es auf eine Motivation der Benutzer ankommt, wie bei Lernsystemen spielen Agenten offenbar eine wichtige Rolle.

In Tabelle 8.1 werden die in Abschnitt 1.3 formulierten Anforderungen aufgegriffen und ihre Lösung zusammengefasst. Dabei werden die Ziele,

- einfache Integration in bestehende Benutzeroberflächen bzw. Web-Seiten,
- abstrakte Definition des Persona-Verhaltens auf XML-Basis,
- Parametrisierung von Lebendigkeit, Vertrauenswürdigkeit und Emotionalität,
- Einsatz in Internet-Anwendungen (Web-Optimierung)
- Verallgemeinerung auf mehrere gleichzeitig aktive Präsentationsagenten,

besonders berücksichtigt.

8.3 Ausblick

Der Präsentationsagent ist momentan auf 2D-Umgebungen beschränkt. Modelliert man eine 3D-Figur beispielsweise in VRML, so können die Translationen/Rotationen der Objekte in der 3D-Welt für die Animation durch eine Java-VRML-Schnittstelle von dem Persona-Player aus angesteuert werden. Eine weitere Möglichkeit ist die Verwendung von Java 3D, der Java-Bibliothek zur Darstellung und Manipulation von 3D-Welten. Neben dem Präsentationsagenten muss in diesem Fall auch ein Äquivalent zu den VisualizationGadgets eingeführt werden. Zwar ist es möglich die 2D-Daten als Textur in der 3D-Welt einzusetzen, allerdings könnten bei 3D-Objekte auch diese Objekte in der 3D-Welt manipuliert werden. Versuchsweise wurde der Persona-Player zur Ansteuerung einer in VRML modellierten Funktion getestet.

Um eine bessere Integration des Präsentationsagenten in die HTML-Seite zu erreichen, kann man durch Ansteuerung von Persona mittels JavaScript auf Elemente der HTML-Seite zugreifen. Beispielsweise kann das Drücken eines Knopfes auf einer Seite eine bestmmte Aktion auslösen. Zu diesem Zweck ist es sinnvoll, das normale HTML um spezielle Zusatzbefehle zu erweitern. Eine so definierte Seite wird analog zur Verarbeitung im Persona-Generator eingelesen und die speziellen Tags werden durch eine Kombination von HTML und

Aspekt	Lösung	Abschnitt
Einfache Integration eines Präsentationsagenten in eine bestehende Anwendung	Schnittstelle auf XML-Basis, flexible API	4.2
XML-Integration	Durch Definition von Persona-Befehlen in einem XML-Dialekt kann die Steuerung in eine bestehende WWW-Seite integriert werden.	4.3
Benutzeradaption durch Parametrisierung	Während der Zerlegung komplexer Aktionen werden Benutzerpräferenzen berücksichtigt.	4.3.2
Beeinflussung der Persona-Aktionen durch Manipulation von Oberflächenelementen durch Benutzer	Interaktion mit Window-Manager/Anwendung, Repräsentation von Objekten (VisualizationGadgets), HTML-Integration	6.1, 6.2, 5.2
Web-Optimierung	Client/Server-Architektur, Implementierungssprache Java, Gestenkonstruktion	6.2, 5, 4.1
Verallgemeinerung auf mehrere gleichzeitig aktive Präsentationsagenten	spezielle Sprachkonstrukte, Synchronisationstechniken	4.8, 5.5

Tabelle 8.1: Zusammenfassung einiger erreichter Ziele

JavaScript Code ersetzt. Eine erste Implementierung dieser Technologie (PET, Persona Enabling Toolkit) wird momentan realisiert.

Wichtig für eine Präsentation mit Persona ist die möglichst gute Integration des Präsentationsagenten in die Anwendung. Dieser Ansatz wurde vielfach verfolgt. Beispielsweise hat die X-Window-Version des Präsentationsagenten Zugriff auf den Window-Manager und damit die Möglichkeit, die Position und Größe der Fenster zu ermitteln und auch zu manipulieren. Gleichzeitig erhält sie vom Präsentationssystem Informationen über den Inhalt eines Fensters, so dass auf Benutzereingaben besser zu reagieren ist (z.B. Verschieben oder Ikonifizieren eines Fensters, Anklicken eines Grafikelements). Durch die Integration der VisualizationGadgets in den von der Persona erreichbaren Bereich der WWW-Seite erhält des System auch bei WWW-Präsentationen die Fähigkeit, die dargestellten Daten zu manipulieren (z.B. Zeigegesten, sensitive Grafik). Hilfreich für eine bessere Benutzerführung wäre es, wenn der Präsentationsagent direkten Zugriff auch auf Bedienungselemente wie Knöpfe, Menüs oder Textfelder hätte. Damit auch auf solche Elemente Zeigegesten ausgeführt werden können, müssten diese in das Java-Applet einbezogen werden. Durch Verwendung der JavaBean-Technologie bzw. durch Verwendung von hierauf aufbauenden Konzepten ist eine solche Integration des Präsentationsagenten in hochinterkative WWW-Seiten gewährleistet, wodurch sich auch weitere Anwendungsgebiete ergeben, wie z.B. die Unterstützung des Benutzers beim Ausfüllen komplexer WWW-Formulare.

Eine weitere Erweiterungsmöglichkeit ist die Integration einer umfangreiche-

ren Dialogkomponente. Diese umfasst einerseits eine erweiterte Sprachverarbeitung (textuelle oder echte Spracheingabe) und andererseits eine umfassendere Analyse der Benutzerinteraktionen (Eingaben des Benutzers, angewählte Elemente) sowie die Integration dieser Elemente.

Um einem Anwender eine einfachere Spezifikation eines Dialogs zu ermöglichen, ist darüber hinaus die Entwicklung eines Editors wünschenswert, der es erlaubt, domänenabhängige Dialoge effizient und interaktiv zu erstellen. Zu diesem Zweck kann ein Template-basierter Ansatz genutzt werden, wobei die Grobstruktur des Dialogs durch die Domäne festgelegt ist und der Anwender innerhalb bestimmter Grenzen das Verhalten des Agenten modifizieren kann (z.B. Festlegung der Textelemente oder Gestenwahl). Durch die Entwicklung eines solchen Werkzeugs ist es auch für unerfahrene Computerbenutzer einfacher, Präsentationen für den Agenten zu spezifizieren und zu warten.

Anhang A

Befehlsliste des Persona-Players

Die im Folgenden definierten Kommandos können unmittelbar von dem Persona-Player ausgewertet werden. Die Liste enthält die standardmäßig definierten Kommandos. Es können jederzeit neue Kommandos definiert werden (s. Abschnitt. 5.1).

action *url query*

Beschreibung: Es wird ein neues Präsentationsskript vom Server (Generator) angefordert. Dabei werden Register dem Server zur weiteren Verarbeitung übergeben. Zur Spezifikation wird der String *query* genutzt. Er besteht aus Tokens 'sreg *regnr*' (Stringregister), 'pos' (aktuelle Position), 'pinfo *gadgetnr objnr* $name_1 \ldots name_{objnr-1}$'.

actionsequence *nr* $c_1 \ldots c_{nr}$

Beschreibung: Definition einer Folge von Sprungadressen, die bei einem Mausklick jeweils ausgeführt werden. Nach jeder Aktion wird die nächste Adresse als folgende Sprungadresse benutzt.

Beispiel: actionsequence 3 10 20 30

callagent *nr agentname*

Beschreibung: Der Agent mit dem angegebenen Namen führt den Befehl aus Zeile *nr* aus.

Beispiel: callagent 5 wichtel

event *startdrag dragging enddrag lm rm*

Beschreibung: Definition von Mausevents, auf die der Agent reagieren soll. Dabei definieren *startdrag*, *dragging*, *enddrag* Dragging-Gesten, die gezeigt werden, wenn der Benutzer den Agenten mit der Maus veschiebt. Sind die Gesten null, so wird das Dragging abgeschaltet. *lm*,*rm* sind Skriptadressen, an die die Engine springt, falls der Benutzer auf den Agenten mit der Maus (linke, rechte Taste) klickt. Bei dem Wert -1 wird diese Funktionalität abgeschaltet.

Beispiel: event startdrag dragging enddrag -1 -1

exit

Beschreibung: Programmabbruch

Beispiel: exit

gadget *type x y w h register data*

Beschreibung: Darstellung eines Gadgets vom Typ *type* (0: Bild, 1:3D, 2:Balkendiagramm, 3:Kuchendiagramm, 4:Legende, 5:Graf, 6:Text, 7:Slideshow) an der Stelle (x,y,w,h). Das Gadget wird in Register *register* abgelegt. Die Daten für das Gadget wird der Datei *data* entnommen.

gesture *name x y text*

Beschreibung: Persona führt die Geste name aus. Dabei wird x,y,text in bestimmte Register propagiert. Während die Geste ausgeführt wird, stehen diese Informationen z.B. für die Koordinaten des Zeigestockes bzw. für den Text der Sprechblase zur Verfügung.

Beispiel: gesture leftpoint1 300 100 null

goto *adr*

Beschreibung: Die Persona-Engine setzt den Programmzähler auf *adr*, d.h. sie führt einen Sprung zur angegebenen Adresse aus. Dabei kann *adr* eine Zeilennummer oder ein mit symbolicjump definierte Adresse sein.

Beispiel: goto 10

idles *onoff*

Beschreibung: Falls onoff true ist, sind Ruheaktionen erlaubt.

Beispiel: idles true

initclickforobject *reg adr name gadgetreg*

Beschreibung: Analog zu setsensitivearea, allerdings Berechnung der Fläche über VisualizationGadget-Funktionen

loaddoc *docurl fname*

Beschreibung: Das Dokument mit der URL *docurl* wird in dem Frame/Window mit dem Namen *fname* angezeigt.

Beispiel: loaddoc http://www.dfki.de mainframe

loadpresentation *url wait nr*

Beschreibung: Lädt ein neues Skript von der URL *url*. Falls *wait* gleich true ist, wird auf das Ende des Ladevorganges gewartet, sonst erfolgt der Ladevorgang als getrennter Thread. Das Skript wird an der Stelle *nr* in das alte Skript eingefügt. Falls nr kleiner als 0, wird das alte Skript durch das neue Skript ersetzt.

Beispiel: loadpresentation http://persona.dfki.de:8080/tmp/ newskript.txt false -1

menuhide

Beschreibung: Ein eventuell sichtbares Pop-Up-Menü wird unsichtbar gemacht.

Beispiel: menuhide

modifygadget *gnr com*

Beschreibung: Wendet den Befehl *com* auf das Gadgetregister *gnr* an. Der Befehl ist von dem Typ des jeweilgen Gadgetregisters abhängig.

Beispiel: modifygadget 2 rotate_0_10_0

noop

Beschreibung: Keine Auswirkung

Beispiel: noop

rndjump *nr l1 ... lnr*

Beschreibung: Es wird ein Sprung zu einer aus der Liste der Sprungadressen $l1 \ldots lnr$ zufällig ausgewählten Adresse ausgeführt. Die Adressen können entweder Zeilennummern oder mit symbolicjump definierte Adressen sein.

Beispiel: rndjump 3 a b c

point *name t register data*

Beschreibung: Der Präsentationsagent berechnet lokal durch Kommunikation mit dem angegebenen VisualizationRegister eine Zeigegeste. Als Basis dient die Zeigegeste mit dem Namen *name*. Zwischen den einzelnen Punkten wird t ms gewartet. Falls $register < 0$ ist, ist *data* eine Folge von X-Y-Werten (Anzahl = $|register|$) und die Zeigegeste erfolgt entlang dieses Polygonzuges. Sonst liefert das Grafikregister mit dieser Nummer die Werte. Falls auf *register* true folgt, so handelt es sich um eine Zeigegeste auf ein einzelnes Objekt, dessen Bezeichner unmittelbar folgt. Im anderen Fall soll von dem Grafikregister ein Polygonzug geliefert werden. Es folgen *pnr bez1 ... bezpnr* in der Befehlszeile.

Beispiel: point leftpoint1 300 2 false 4 a b c d

position *x y*

Beschreibung: Der Präsentationsagent wird am Punkt (x,y) positioniert.

Beispiel: position 40 40

sendform *name url*

Beschreibung: Die Daten der HTML-Form mit Namen *name* werden ausgelesen und an die angegeben URL geschickt.

Beispiel: sendform eingabe http://persona.dfki.de:8080/tmp/aiaconnect?

setreg *regnr val*

Beschreibung: Das Register *regnr* wird auf den Wert *val* gesetzt.

Beispiel: setreg 3 100

setsensitivearea *reg x y w h adr*

Beschreibung: Definition eines sensitiven Bereichs, der durch das Rechteck (x, y, w, h) definiert wird. Wird auf diesen Bereich mit der Maus geklickt, so wird das Programm ab Adresse *adr* fortgesetzt. Wird die Adresse -1 angegeben, so wird die Sensitivität abgeschaltet.

Beispiel: setsensitivearea 1 30 30 20 40 100

speak $geste_1$ g_1 $geste_2$ g_2 *x y audio*

Beschreibung: Einfache Form der Sprachausgabe (s.a. talk). Zur Audiodatei *audio* werden die Gesten $geste_1$, $geste_2$ g_1 bzw. g_2-mal abgespielt. Der Punkt x, y wird in die entsprechenden Register propagiert. Eine der Gesten kann auch null sein (wird ignoriert).

Besipiel: speak speaklong 2 speakshort 1 20 30 audio.au

symbolicjump *nr* $label_1$ $adr_1 \ldots label_{nr}$ adr_{nr}

Beschreibung: Definiert eine Liste mit symbolischen Sprungadressen.

Beispiel: symbolicjump 3 start 10 stop 20 end 50

syncagent *nr* $agent_1 \ldots agent_{nr}$

Beschreibung: Der aktuelle Agent wartet auf den Agenten mit dem angegebenen Namen (Synchronisation).

Beispiel: syncagent 1 wichtel

talk $geste_1$ $geste_2$ *audio x y nr* $e_1 \ldots e_{nr}$

Beschreibung: Sprechgeste, die eine bessere Abstimmung zwischen Audioausgabe und Bildsequenz erlaubt. Die Bedeutung von $geste_1$, $geste_2$, *x*, *y*, *audio* ist bei dem Befehl speak angegeben. Mit Hilfe des Arrays *e* wird die Folge der Bildsequenzen definiert. Ist $e_i = 0$, so gibt $ei + 1$ die Länge einer Pause in ms an. Ist $e_i = 1$, so wird die Geste $geste_1$ $ei + 1$-mal wiederholt. Bei $e_i = 2$ wird die Geste $geste_2$ benutzt.

Beispiel: talk speaklong speakshort audio.au 10 20 6 0 100 1 2 2 3

wait *time doidle*

Beschreibung: Die Ausführung des Skripts wird für *time* Millisekunden gestoppt. Falls *doidle* true ist, werden Vitalaktionen ausgeführt.

Beispiel: wait 3000 true

menu *nr text*1 *label*1 ... *textn labeln*

Beschreibung: Der Präsentationsagent zeigt ein Pop-Up-Menü mit den Einträgen *text*1 ... *textn*. Wenn ein Menüpunkt angewählt wird, erfolgt eine Verzweigung zur entsprechenden Adresse.

Beispiel: menu 2 Weiter 20 Start 10

input *ok error type register text*

Beschreibung: Der Präsentationsagent zeigt ein Eingabefenster mit der Beschriftung *text*. Das Ergebnis wird in dem Register abgelegt. Falls der mittels *type* angegebene Test positiv war, wird an die Stelle *ok* im Skript verzweigt, sonst an *error*.

Beispiel: menu 2 Weiter 20 Start 10

Index

Literaturverzeichnis

7th Level. Agent 7. http://www.7thlevel.com, 1998.

B. Adelson. Evocative Agents and Multi-Media Interface Design. In: *Proc. of the UIST'92 (ACM SIGGRAPH Symp. on User Interface Software and Technology)*, S. 351–356, Monterey, CA, U.S.A., 1992.

E. André. *Ein planbasierter Ansatz zur Generierung multimedialer Präsentationen*. Doktorarbeit, Universität des Saarlandes, 1995.

E. André, W. Finkler, W. Graf, T. Rist, A. Schauder, W. Wahlster. WIP: The Automatic Synthesis of Multimodal Presentations. In: *Intelligent Multimedia Interfaces*, S. 75–93, AAAI Press, 1993.

E. André, W. Graf, J. Müller, H.-J. Profitlich, T. Rist, W. Wahlster. AiA: Adaptive Communication Assistant for the Effective Infobahn Access. Technischer Bericht, Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, 1996.

E. André, W. Graf, J. Müller, H.-J. Profitlich, T. Rist, W. Wahlster. PPP: Personalized Plan-based Presenter. Technischer Bericht, Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, 1997.

E. André, J. Müller, T. Rist. Objektorientierte Multimedia-Präsentationsagenten für Benutzerschnittstellen der Multimedia-Kommunikation. In: *Fortschritte der objektorientierten Softwaretechnologien. ONLINE '96. Congressband VI.*, S. C626.01–C626.15, 1996a.

E. André, J. Müller, T. Rist. The PPP Persona: A Multipurpose Animated Presentation Agent. In: T. Catarci, M. Costabile, S. Levialdi, G. Santucci (Hrsg.), *Advanced Visual Interfaces*, S. 245–247, ACM Press, 1996b.

E. André, J. Müller, T. Rist. WIP/PPP: Automatic Generation of Personalized Multimedia Presentations. In: *ACM Multimedia 96*, S. 407–408, ACM Press, 1996c, Demo.

E. André, J. Müller, T. Rist. Computer-Generated Presentation Scripts for Life-Like Interface Agents. In: *Human-Computer Interaction: Software and Hardware Interfaces (Proc. of HCI-97)*, 1997a.

E. André, J. Müller, T. Rist. Ein Life-Like-Character als Präsentationsagent für WWW-Präsentationen. In: *Verteilte, integrierte Anwendungsarchitekturen: Die Software-Welt im Umbruch. ONLINE '97. Congressband VI.*, 1997b.

E. André, J. Müller, T. Rist. WebPersona: A Life-Like Presentation Agent for the World-Wide Web. In: *Proc. of the IJCAI-97 Workshop on Animated Interface Agents: Making them Intelligent*, S. 53–60, Nagoya, 1997c.

E. André, T. Rist. Presenting through Performing: On the Use of Multiple Lifelike Characters in Knowledge-Based Presentation Systems. In: *Proc. of the IUI-2000 (2000 ACM Intelligent User Interfaces Conference)*, S. 351–356, New Orleans, Louisiana, USA, 2000.

E. André, T. Rist, J. Müller. Guiding the User through Dynamically Generated Hypermedia Presentations with a Life-Like Presentation Agent. In: *Proceedings of the 1998 International Conference on Intelligent User Interfaces*, S. 21–28, San Francisco, CA, 1998a.

E. André, T. Rist, J. Müller. Integrating Reactive and Scripted Behaviors in a Life-Like Presentation Agent. In: *Proceedings of the Second International Conference on Autonomous Agents (Agents '98)*, S. 261–268, Minneapolis/St. Paul, 1998b.

E. André, T. Rist, J. Müller. WebPersona: A Life-Like Presentation Agent for the World-Wide Web. *Knowledge-Based Systems*, **11** (1), 25–36, 1998c.

Apache Software Foundation. Apache Software Foundation. http://www.apache.org, 1999.

N. Badler, B. Reich, B. Webber. Towards Personalities for Animated Agents with Reactive an Planning Behaviors. In: R. Trappl, P. Petta (Hrsg.), *Creating Personalities for Synthetic Actors: Towards Autonomous Personality Agents*, Lecture notes in computer science, Lecture notes in artificial intelligence, Springer, Berlin, Heidelberg, 1997.

A. Bahl. Spielraum für Rollentäuscher. *CT*, (8), 1996.

G. Ball, D. Ling, D. Kurlander, J. Miller, D. Pugh. Lifelike Computer Characters: The Persona Project at Microsoft. In: J. M. Bradshaw (Hrsg.), *Software Agents*, AAAI Press/The MIT Press, Menlo Park, CA, 1997.

J. Bates. The Role of Emotion in Believable Agents. *Communications of the ACM*, **Special Issue on Agents**, 1994.

J. Bates, A. Loyall, W. S. Reilly. Integrating Reactivity, Goals and Emotion in a Broad Agent. In: *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, Bloomington, Indiana, 1992a.

J. Bates, A. B. Loyall, W. S. Reilly. An Architecture for Action, Emotion and Social Behavior. Technischer Bericht CMU-CS-92-144, Carnegie Mellon University, Pittsburgh, PA, 1992b.

J. Bates, A. B. Loyall, W. S. Reilly. An architecture for action, emotion and social behaviour. In: *Proceedings of the Fourth European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, S. Martino al Cimino, Italy, 1992c.

M. Bauer, D. Dengler. TrIAs: Trainable Information Assistants for Cooperative Problem Solving. In: *Proc. Agents 99*, S. 260–267, 1999.

J. Bedersdorfer. *Ein agentenbasiertes Framework zum Information Retrieval aus dem Internet und anderen Datenquellen.* Diplomarbeit, Universität des Saarlandes, Saarbrücken, in Vorbereitung.

J. Beskow, M. Dahlquist, B. Granström, M. Lundeberg, K.-E. Spens, T. Öhman. The Teleface project - Multimodal speech communication for the hearing impaired. In: *Proceedings of Eurospeech '97*, Rhodes, Greece, 1997.

J. Beskow, S. McGlashan. Olga - a conversational agent with gestures. In: *Proceedings of Workshop Animated Interface Agents: Making them intelligent (IJCAI-97)*, Nagoya, Japan, 1997.

K. Bharat, P. N. Sukaviriya. Animating User Interfaces Using Animation Servers. In: *Proc. of the 6th Annual Symposium on User Interface Software and Technology (UIST'93)*, S. 69–79, Atlanta, GA, 1993.

K. Binsted. Character Design for Soccer Commentary. In: M. Asada (Hrsg.), *Proceedings of the second RoboCup Workshop*, S. 23–35, Paris, 1998.

F. Biringer. *DaCaPo: Steuerungskomponente für einen animierten Interfaceagenten.* Diplomarbeit, Universität des Saarlandes, Saarbrücken, 1997.

G. A. Boy. Software Agents for Cooperative Learning. In: J. M. Bradshaw (Hrsg.), *Software Agents*, AAAI Press/The MIT Press, Menlo Park, CA, 1997.

J. M. Bradshaw. An Introduction to Software Agents. In: J. M. Bradshaw (Hrsg.), *Software Agents*, AAAI Press/The MIT Press, Menlo Park, CA, 1997.

J. M. Bradshaw, S. Dutfield, P. Benoit, J. D. Woolley. KAoS: Toward An Industrial-Strength Open Agent Architecture. In: J. M. Bradshaw (Hrsg.), *Software Agents*, AAAI Press/The MIT Press, Menlo Park, CA, 1997.

T. Bray, J. Paoli, C. Sperberg-McQueen. Extensible Markup Language (XML) 1.0. Technischer Bericht, World Wide Web Consortium, 1999, http://www.w3.org/TR/1998/REC-xml-19980210.

B.Reeves, C.Nass. *The Media Equation.* Cambridge University Press, 1996.

A. Bruderlin, S. Fels, S. Esser, K. Mase. Hierarchical Agent Interface for Animation. In: *Proceedings of Workshop Animated Interface Agents: Making them intelligent (IJCAI-97)*, Nagoya, Japan, 1997.

J. Cassell, K. R. Thórisson. The power of a nod and a glance: Envelope vs. emotional feedback in animated conversational agents. In: *Applied Artificial Intelligence*, in Druck.

B.-W. Chang, D. Ungar. Animation: From Cartoons to the User Interface. In: *Proc. of the 6th Annual Symposium on User Interface Software and Technology (UIST'93)*, S. 45–55, Atlanta, GA, 1993.

P. R. Cohen, H. J. Levesque. Communicative Actions for Artificial Agents. In: J. M. Bradshaw (Hrsg.), *Software Agents*, AAAI Press/The MIT Press, Menlo Park, CA, 1997.

Cyberware. Cyberware Homepage. http://www.cyberware.com, 1998.

S. Deach. Extensible Stylesheet Language (XSL) Specification. Technischer Bericht, World Wide Web Consortium, 1999, http://www.w3.org/TR/WD-xsl.

D. DeCarlo, J. Kaye, D. Metaxas, J. R. Clarke, B. Webber, N. Badler. Integrating Anatomy and Physiology for Behaviour Modeling. Technischer Bericht, Center for Human Modeling and Simulation, University of Pennsylvania, 1998.

D. Dehn, S. van Mulken. The Impact of Animated Interface Agents: A Review of Empirical Research. *International Journal of Human-Computer Studies*, in Druck.

H. Dohi, M. Ishizuka. Visual Software Agent: A Realistic Face-to-Face Style Interface connected with WWW/Netscape. In: *IJCAI-97 Workshop on Intelligent Multimodal Systems*, Nagoya, Japan, 1997.

P. Doyle, B. Hayes-Roth. An Intelligent Guide for Virtual Environments. In: *Proceedings of Workshop Animated Interface Agents: Making them intelligent (IJCAI-97)*, Nagoya, Japan, 1997.

P. Ekman, W. Friesen. *Facial action coding system. Manual.* Consulting Psychologists Press, Palo Alto, California, 1978.

P. Ekman, E. L. Rosenberg (Hrsg.). *What the face reveals: Basic and applied studies of spontaneous expression using the facial action coding system.* Oxford University Press, 1997.

C. Elliott. I picked up catapia and other stories: A multimodal approach to expressivity for emotionally intelligent agents. In: *Proceedings of the First International Conference on Automomous Agents*, S. 451–457, 1997.

C. Elliott, J. Rickel, J. C. Lester. Integrating affective computing into animated tutoring agents. In: *Animated Interface Agents: Making them intelligent (IJCAI 1997)*, 1997.

G. Ender. Mein Haus, der PC. *C'T*, (16), 1997.

C. Endres. *Personal Picture Finder: A Softbot for the World Wide Web.* Diplomarbeit, Universität des Saarlandes, Saarbrücken, 1999.

Entropic Research Laboratory. TrueTalk Overview. http://www.entropic.com/truetalk.html, 1996.

T. Erickson. Designing Agents as if People Mattered. In: J. M. Bradshaw (Hrsg.), *Software Agents*, AAAI Press/The MIT Press, Menlo Park, CA, 1997.

O. Etzioni, D. Weld. Intelligent agents on the Internet: Fact, fiction and forecast. In: *IEEE Expert*, 1995.

D. Fellner, A. Hopp. MRT-VR - Verteilte dynamische Multi-User-3D-Umgebung. In: *Agenten, Assistenten, Avatars*, Darmstadt, Fraunhofer Institut für Graphische Datenverarbeitung, 1997.

R. Fikes, N. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, (2), 189–208, 1971.

T. Finin, Y. Labrou, J. Mayfield. KQML as an Agent Communication Language. In: J. M. Bradshaw (Hrsg.), *Software Agents*, AAAI Press/The MIT Press, Menlo Park, CA, 1997.

W. Finkler, G. Neumann. MORPHIX: A Fast Realization of a Classification-Based Approach to Morphology. In: H. Trost (Hrsg.), *4. Österreichische Artificial-Intelligence-Tagung: Wiener Workshop - Wissensbasierte Sprachverarbeitung*, S. 11–19, Springer, Berlin, Heidelberg, 1988.

M. Flashar. Schöne erwartbare Welt. *Frankfurter Allgemeine Zeitung*, 1998.

Focus. Gespräche mit dem Kunst-Charakter. *Focus*, (17), 1997.

A. Foerst. Feel the Difference. *C'T*, 1997.

J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes. *Computer Graphics. Principles and Practice*. Addison-Wesley, Reading, Massachusetts, 1992.

S. Franklin, A. Graesser. Is It an Agent or Just a Program ? A Taxonomy for Autonomous Agents. In: *Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages*, New York, Springer, 1996.

G. S. Freyermuth. Stars von der Stange. *C'T*, (7), 1997.

M. R. Genesereth. An Agent-Based Framework for Interoperability. In: J. M. Bradshaw (Hrsg.), *Software Agents*, AAAI Press/The MIT Press, Menlo Park, CA, 1997.

J. Gettys, J. Mogul, H. F. L. Masinter, P. Leach, T. Berners-Lee. Hypertext Transfer Protocol (RFC 2616). Technischer Bericht, World Wide Web Consortium, 1999.

A. Goldberg. IMPROV: A System for Real-Time Animation of Behavior-Based Interactive Synthetic Actors. In: R. Trappl, P. Petta (Hrsg.), *Creating Personalities for Synthetic Actors: Towards Autonomous Personality Agents*, Lecture notes in computer science, Lecture notes in artificial intelligence, Springer, Berlin, Heidelberg, 1997.

Government of Nova Scotia. Nova Scotia Counties. http://www.gov.ns.ca/homa/muns/info/counties/counties.htm, 1999.

W. Graf, S. Neurohr. Constraint-based layout in visual program design. In: *Proceedings of the 1995 IEEE Symposium on Visual Languages (VL '95)*, S. 116–117, Darmstadt, 1995.

W. H. Graf. *Intentionsgesteuertes Layout-Design multimedialer Präsentationen mit Constraints*, Band 148 von *DISKI*. infix, St. Augustin, 1997.

S. Gutz. *Up To Speed With Swing. User Interfaces with Java Foundation Classes*. Manning, Greenwich, 1998.

K. Harbusch, W. Finkler, A. Kilger. Incremental Syntax Generation with Tree Adjoining Grammars. In: W. Brauer, D. Hernandez (Hrsg.), *4th International GI Congress on Knowlegde-Based Systems*, S. 363–374, Munich, Springer, 1991.

O. Hasegawa, K. Sakaue. A CG Tool for Constructing Anthropomorphic Interface Agents. In: *Animated Interface Agents: Making them intelligent (IJCAI 1997)*, 1997.

R. Henke. Technik ohne Tücken. *Focus*, (35), 1998.

R. Henke, H. Seeger. Die Geister, die wir riefen .... *Focus*, 1997.

H. Hohl. Ein objektorientierter Ansatz zur Entwicklung skalierbarer Telekooperationssysteme. In: W. Wahlster (Hrsg.), *Online '96. Congress VI. Fortschritte der objektorientierten Softwaretechnologien*, Velbert, ONLINE, 1996.

Y. Honda, M. Rockwell, B. Röhl. Living Worlds, Making VRML 2.0 Applications Interpersonal and Interoperable. http://www.livingworlds.com/draft2/index.htm, 1997.

R. D. Huard, B. Hayes-Roth. Character Mastery with Improvisational Puppets. In: *Proceedings of Workshop Animated Interface Agents: Making them intelligent (IJCAI-97)*, Nagoya, Japan, 1997.

IBM. *IBM ViaVoice*. 1998.

IBM. XML Parser for Java. Technischer Bericht, 1999.

Industrial Light & Magic. Jurassic Park. http://www.imdb.com, 1993.

Industrial Light & Magic. Casper. http://www.imdb.com, 1995.

B. S. Interactive. Cyberhub. http://www.blacksun.com, 1997a.

O. Interactive. OZ-Virtual. http://www.oz-inc.com, 1997b.

K. Isbister. How do we decide when a computer character is intelligent ? Intelligence assessment and perceived intelligence in human-human interaction and how it relates to human-computer interaction. In: *Lifelike Computer Characters Conference*, Snowbird, Utah, 1995.

W. L. Johnson, J. W. Rickel, J. C. Lester. Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments. In: *International Journal of Artificial Intelligence in Education*, 2000, in Vorbereitung.

J. Jung, A. Kresse, N. Reithinger, R. Schäfer. Das System ZORA. Wissensbasierte Generierung von Zeigegesten. Technischer Bericht 54, SFB 314 (XTRA), Universität des Saarlandes, 1989.

J. Junge. Siegen auf die sanfte Tour. *Fensehwoche*, (35), 1998.

S. Kaatz. Mutter war ein Mac. *Focus*, (10), 220, 1997.

M. Kantrowitz, J. Bates. Integrated Natural Language Generation Systems. Technischer Bericht CMU-CS-92-107, Carnegie Mellon University, Pittsburgh, PA, 1992.

H. A. Kautz, P. B. Ladkin. Integrating Metric and Qualitative Temporal Reasoning. In: *Proc. of AAAI-91*, S. 241–246, Anaheim, CA, 1991.

M. Kessler, R. Kilgore. Bringing an Agent to Life. In: *Proceedings of Workshop Animated Interface Agents: Making them intelligent (IJCAI-97)*, Nagoya, Japan, 1997.

W. J. King, J. Ohya. The representation of agents: Anthropomorphism, agency and intelligence. In: R. Bilger, S. Guest, M. J. Tauber (Hrsg.), *Human Factors in Computing Systems: CHI 96 Electronic Conference Proceedings*, ACM, 1996.

J. Klein. Bruzard:A platform for computer expression of emotion. http://affect.www.media.mit.edu/projects/affect/AC_research/projects/bruzard.html, 1998.

C. A. Knoblock, J. L. Ambite. Agents for Information Gathering. In: J. M. Bradshaw (Hrsg.), *Software Agents*, AAAI Press/The MIT Press, Menlo Park, CA, 1997.

R. Knop. *Simulierte Gesichtsausdrücke als zusätzlicher Kommunikationskanal in einem natürlichsprachlichen Dialogsystem*. Diplomarbeit, Universität des Saarlandes, Saarbrücken, 1997.

R. M. Koch, M. H. Gross, A. A. Bosshard. Ein FEM-basierter Mimikgenerator für animierte anthropomorphe Avatare. In: *Agenten, Assistenten, Avatars*, Darmstadt, Fraunhofer Institut für Graphische Datenverarbeitung, 1997.

T. Koda, P. Maes. Agents with faces: The effect of personification. *Proceedings of the Fifth IEEE International Workshop on Robot and Human Communication*, S. 189–194, 1996.

C. Kolb. Rayshade-Homepage. http://www-graphics.stanford.edu/ cek/rayshade, 1999.

R. Kozierok, P. Maes. A Learning Interface Agent for Scheduling Meetings. In: *ACM SIGCHI International Workshop on Intelligent User Interfaces*, Orlando, Florida, 1993.

R. Krause. http://www.uni-sb.de/philfak/fb6/krause/nonverb.htm, 1995.

R. Krause. Emotion und Motivation. http://www.uni-saarland.de/philfak/fb6/krause/emo/, 1998.

W. Krischke. Dieseits des Nebellands der Diskurse. *Frankfurter Allgemeine Zeitung*, 1997.

A. Krüger. *Automatische Abstraktion in 3D-Graphiken.* Doktorarbeit, Universität des Saarlandes, 1999.

R. Lachman. Experiments in Mapping Character Animation to Computational State. In: *Proceedings of Workshop Animated Interface Agents: Making them intelligent (IJCAI-97)*, Nagoya, Japan, 1997.

Y. Lashkari, M. Metral, P. Maes. Learning Interface Agents. In: *Proceedings of the Twelfth National Conference on Artificial Intelligence*, S. 444–450, Menlo Park, Calif., American Association for Artifical Intelligence, 1994.

B. Laurel. *The Art of Human-Computer Interface Design.* Addison-Wesley, New York, 1990.

B. Laurel. Interface Agents: Metaphors with Character. In: J. M. Bradshaw (Hrsg.), *Software Agents*, AAAI Press/The MIT Press, Menlo Park, CA, 1997.

J. Lester, B. Stone, G. Stelling. Lifelike pedagogical agents for mixed-initiaitive problem solving in constructivist learning environments. *User Modeling and User-Adapted Interaction*, **9**, 1–44, 1999a.

J. Lester, L. Zettlemoyer, J. Gregoire, W. Bares. Explanatory lifelike avatars: Performing user-designed tasks in 3d learning environments. In: *Proceedings of the Third International Conference on Autonomous Agents*, 1999b.

J. C. Lester, S. A. Converse, S. E. Kahler, S. T. Barlow, B. A. Stone, R. S. Bhogal. The persona effect: Affective impact of animated pedagogical agents. In: S. Pemberton (Hrsg.), *Human Factors in Computing Systems: CHI 97 Conference Proceedings*, S. 359–366, New York, ACM Press, 1997a.

J. C. Lester, J. L. Voerman, S. G. Towns, C. B. Callaway. Cosmo: A Life-like Animated Pedagogical Agent with Deictic Believability. In: *Animated Interface Agents: Making them intelligent (IJCAI 1997)*, 1997b.

H. Lieberman. Letizia: An Agent That Assists Web Browsing. In: *International Joint Conference on Artificial Intelligence*, Montreal, 1995.

M. Lucas. *Reden lernen für Beruf und Freizeit.* Buch und Zeit, Köln, 1979.

P. Maes. Agents that Reduce Work and Information Overload. In: J. M. Bradshaw (Hrsg.), *Software Agents*, AAAI Press/The MIT Press, Menlo Park, CA, 1997.

N. Magnenat-Thalmann, P. Volino. Dressing Virtual Humans. In: R. Trappl, P. Petta (Hrsg.), *Creating Personalities for Synthetic Actors: Towards Autonomous Personality Agents*, Lecture notes in computer science, Lecture notes in artificial intelligence, Springer, Berlin, Heidelberg, 1997.

H. Maldonado, A. Picard, P. Doyle, B. Hayes-Roth. Tigrito: A Multi-Mode Interactive Improvisational Agent. Technischer Bericht, Knowledge Systems Laboratory. Stanford University, 1997.

T. W. Malone, K.-Y. Lai, K. R. Grant. Agents for Information Sharing and Coordination: A History and Some Reflections. In: J. M. Bradshaw (Hrsg.), *Software Agents*, AAAI Press/The MIT Press, Menlo Park, CA, 1997.

K. Manske, M. Mühlhäuser. An Open Architecture for Comic Actor Animation. In: *ACM Multimedia 97. Electronic Proceedings*, 1997.

Mapquest. Mapquest Homepage. http://www.mapquest.com, 1998.

A. Marguier. Adonis bevorzugt. *Saarbrücker Zeitung*, (1.9.1998), 1998.

G. Mark, A. Voss. What Constitutes Attractivity in Cyberspace ? In: *Agenten, Assistenten, Avatars*, Darmstadt, Fraunhofer Institut für Graphische Datenverarbeitung, 1997.

K. Mase. Aspects of Interface Agents: Avatar, Assitant and Actor. In: *Animated Interface Agents: Making them intelligent (IJCAI 1997)*, 1997.

M. Maybury, W. Wahlster (Hrsg.). *Readings in Intelligent User Interfaces*. Morgan Kaufmann, San Francisco, 1998.

D. Meyer. OFF-Viewer Applet. The Geometry Center, 1997.

Microsoft. Microsoft Agent Homepage. 1997.

Microsoft. Microsoft Chat. 1998a.

Microsoft. *Windows 98*. 1998b.

Mitsubishi Electric. Open Community. http://www.opencommunity.com, 1999.

S. Morishima, H. Sera, D. Terzopoulos. A Physics-based Talking Head for Interface Agent. In: *Animated Interface Agents: Making them intelligent (IJCAI 1997)*, 1997.

J. Müller. *GeoDisplay: Ein flexibles Visualisierungswerkzeug für Lisp*. Diplomarbeit, Universität des Saarlandes, Saarbrücken, 1994.

J. Müller. Ein Präsentationsagent für WWW- und Desktop Applikationen. In: *Festschrift anläßlich der Neubaueinweihung und des 10-jährigen Bestehends des Fraunhofer IGD. Agenten, Assistenten, Avatars*, Darmstadt, Fraunhofer Institut für Graphische Datenverarbeitung, 1997.

J. Müller, E. André, T. Rist. Ein Präsentationsagent für WWW- und Desktop-Applikationen. In: *Berichtsband des Workshops KI-Techniken für intelligente Mensch-Maschine-Schnittstellen auf der 21. Deutschen Jahrestagung für Künstliche Intelligenz*, 1997.

J. Müller, T. Rist, E. André. Presenting Graphical Objects with a Life-Like Character. In: *Proc. of the ECAI'98 Workshop on Combining AI and Graphics for the Interface of the Future*, S. 110–112, Brighton, UK, 1998.

F. Nack. Verstehen Sie Spaß ? - Betrachtungen zu Humor in der Mensch-Maschine-Kommunikation. In: *21. Jahrestagung für Künstliche Intelligenz. Workshop Techniken für intelligente Mensch-Maschine-Schnittstellen*, Freiburg, 1997.

N. Negroponte. Agents: From Direct Manipulation to Delegation. In: J. M. Bradshaw (Hrsg.), *Software Agents*, AAAI Press/The MIT Press, Menlo Park, CA, 1997.

R. Neumann. *Zielwirksam reden. Informieren, argumentieren, präsentieren.* Expert Verlag, 1985.

T. Noma, N. I. Badler. A Virtual Human Presenter. In: *Proceedings of Workshop Animated Interface Agents: Making them intelligent (IJCAI-97)*, Nagoya, Japan, 1997.

D. A. Norman. How Might People Interact with Agents. In: J. M. Bradshaw (Hrsg.), *Software Agents*, AAAI Press/The MIT Press, Menlo Park, CA, 1997.

C. Obmann. Im Auftrag des Herrn. *DM*, S. 69–73, 1998.

K. Packard. X11 Nonrectangular Window Shape Extension Version 1.0, X11 R5. Technischer Bericht, MIT X Consortium, MIT, Cambridge, Massachusetts, 1989.

A. Pease. *Body Language*. Sheldon Press, London, 1993.

M. Phillips. *Geomview Manual*. The Geometry Center, Minneapolis, 1994.

J. Piesk, G. Trogemann. Dialogfähige 3D-Charaktere in emotionsbasierten Lernumgebungen. In: *Agenten, Assistenten, Avatars*, Darmstadt, Fraunhofer Institut für Graphische Datenverarbeitung, 1997.

Pixar. Toy Story. http://www.imdb.com, 1995.

T. Porter. Creating Lifelike Characters in Toy Story. *SIGART Bulletin*, **8** (1-4), 10–14, 1997.

H.-J. Post. Kino gefühlsecht. *C'T*, (19), 1998.

J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, T. Carey. *Human-Computer Interaction*. Addison-Wesley, Harlow, England, 1994.

Puppettime. Puppettime. 1998.

W. S. Reilly, J. Bates. Building Emotional Agents. Technischer Bericht CMU-CS-92-143, Carnegie Mellon University, Pittsburgh, PA, 1992.

W. S. N. Reilly. *Believable Social and Emotional Agents*. Doktorarbeit, Carnegie Mellon University, Pittsburgh, PA, 1996.

J. Rickel, W. L. Johnson. Steve: An Animated Pedagogical Agent for Procedural Training in Virtual Environments. In: *Proceedings of Workshop Animated Interface Agents: Making them intelligent (IJCAI-97)*, Nagoya, Japan, 1997.

T. Rist. *Wissensbasierte Verfahren für den automatischen Entwurf von Gebrauchsgraphik in der technischen Dokumentation*. Doktorarbeit, Universität des Saarlandes, Saarbrücken, 1995.

T. Rist, E. André, J. Müller. Adding Animated Presentation Agents to the Interface. In: *Proceedings of the 1997 International Conference on Intelligent User Interfaces*, S. 79–86, Orlando, Florida, 1997.

D. Römer. Fast wie im Leben. *Focus*, (25), 1998.

V. Samid. *Ein Textvisualisierer für den PPP-Persona*. Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, Saarbrücken, 1998.

S. Schaumann. *FoPra: Intelligente Netzagenten. Geschäftsgrafik-Modul*. Saarbrücken, 1998.

B. Sheth. *A Learning Approach to Personalized Information Filtering*. Diplomarbeit, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1994.

B. Shneiderman. Direct Manipulation Versus Agents: Paths to Predicatble, Controllable and Comprehensible Interfaces. In: J. M. Bradshaw (Hrsg.), *Software Agents*, AAAI Press/The MIT Press, Menlo Park, CA, 1997.

Y. Shoham. An Overview of Agent-Oriented Programming. In: J. M. Bradshaw (Hrsg.), *Software Agents*, AAAI Press/The MIT Press, Menlo Park, CA, 1997.

S. Smith, J. Bates. Towards a Theory of Narrative for Interactive Fiction. Technischer Bericht CMU-CS-89-121, Carnegie Mellon University, Pittsburgh, PA, 1989.

Sony. Community Place. http://www.sonypic.com/vs, 1997.

U. Spierling, K. Pipke, W. Müller. Konzeption und Viusalisierung einer agentenbasierten Benutzungsoberfläche für Tätigkeiten im Büro der Zukunft. In: *Agenten, Assistenten, Avatars*, Darmstadt, Fraunhofer Institut für Graphische Datenverarbeitung, 1997.

Sun Microsystems. *Network Programming Guide*. Palo Alto, Calif., 1990.

Sun Microsystems. *Java Speech API Home Page*. Palo Alto, Calif., 1998a, http://java.sun.com/products/java-media/speech/.

Sun Microsystems. *JDK 1.1 Documentation*. Palo Alto, Calif., 1998b.

Sun Microsystems. Java Servlet API. Technischer Bericht, 1999.

A. Takeuchi, T. Naito. Situated facial displays: Towards social interaction. In: I. Katz, R. Mack, L. Marks, M. B. Rosson, J. Nielsen (Hrsg.), *Human Factors in Computing Systems: CHI 95 Conference Proceedings*, S. 450–455, New York, ACM Press, 1995.

D. Thalmann. In the Next Century, Virtual Worlds will be Inhabited With Autonomous and Intelligent Virtual Humans. In: *Festschrift anläßlich der Neubaueinweihung und des 10-jährigen Bestehends des Fraunhofer IGD*, Darmstadt, Fraunhofer Institut für Graphische Datenverarbeitung, 1997.

D. Thalmann, H. Noser, Z. Huang. Autonomous Virtual Actors Based on Virtual Sensors. In: R. Trappl, P. Petta (Hrsg.), *Creating Personalities for Synthetic Actors: Towards Autonomous Personality Agents*, Lecture notes in computer science, Lecture notes in artificial intelligence, Springer, Berlin, Heidelberg, 1997.

The Human-Computer Interaction Group (MIT). Hal: The Next Generation Intelligent Room. http://www.ai.mit.edu/projects/hal/, 1998.

The MBROLA Project. http://tcts.fpms.ac.be/synthesis/mbrola.html, 1998.

V. Tschernomas. *Entwicklung einer Java-basierten Dialoganalysekomponente für den Präsentationsagenten Persona*. Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, 1999a.

V. Tschernomas. *PrePlan Dokumentation (Java-Version)*. Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, 1999b.

V. Turau. Web-Roboter. *Informatik Spektrum*, (21), 159–160, 1998.

University of Portsmouth. Portsmouth Map. http://www.port.ac.uk/adc/maptest.htm, 1998.

S. van Mulken, E. André, J. Müller. The persona effect: How substantial is it ? In: H. Johnson, L. Nigay, C. Roast (Hrsg.), *People and computers XIII: Proceedings of HCI 98*, S. 53–66, Berlin, Springer, 1998.

S. van Mulken, E. André, J. Müller. An Empirical Study on the Trustworthiness of Life-Like Interface Agents. In: *Proceedings of HCI International 1999*, München, 1999.

W. Wahlster. Planning Multimodal Discourse. S. 95–96, 1993a.

W. Wahlster. Verbmobil: Translation of Face-To-Face Dialogs. In: O. Herzog, T. Christaller, D. Schütt (Hrsg.), *Grundlagen und Anwendungen der künstlichen Intelligenz(17. Fachtagung)*, S. 393–402, Springer, Berlin, Heidelberg, 1993b.

A. Weiss. *Implementierung von Tools zur Realisierung von graphischen Überblendtechniken*. Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, 1998.

A. Wexelblat. Don't Make That Face: a report on anthropomophizing an interface. Technischer Bericht SS-98-02, AAAI, AAAI Press, 1998.

J. E. White. Mobile Agents. In: J. M. Bradshaw (Hrsg.), *Software Agents*, AAAI Press/The MIT Press, Menlo Park, CA, 1997.

M. Wilson. Metaphor to Personality: the role of animation in intelligent interface agents. In: *Proceedings of Workshop Animated Interface Agents: Making them intelligent (IJCAI-97)*, Nagoya, Japan, 1997.

L. Wood, A. L. Hors. Document Object Model (DOM). http://www.w3.org/DOM/, 1999.