UNIVERSITÄT HAMBURG
FACHBEREICH INFORMATIK
ARBEITSBEREICH TIS

# Studienarbeit:

## Modell einer computergestützten Lernumgebung

Christoph Haas
Hamburg, 7. Dezember 1999

Betreuer:
Dr. Werner Hansmann

**Zusammenfassung**

Diese Arbeit untersucht die Möglichkeiten von Hypertext als Werkzeug zur Erstellung digitaler Lehrbücher. Zur Verdeutlichung der Überlegungen wird ein Grundgerüst eines Hypertextdokuments zum Thema Computergrafik erstellt. Das Dokument wird nach Abschluß dieser Arbeit von anderen Autoren fortgesetzt.

Hinweis:

In diesem Dokument werden der Einfachheit und Lesbarkeit halber ausschließlich männliche Personenbezeichnungen verwendet. Falls Sie sich als Leserin dadurch benachteiligt fühlen, so ersetzen Sie diese bitte durch die entsprechenden weiblichen Personenbezeichnungen.

# Inhaltsverzeichnis

# 1 Einleitung

Wissen hat nur selten eine rein sequentielle Struktur. Damit es überhaupt vermittelt werden kann, wird es in Büchern oder im Rahmen von Vorlesungen künstlich sequentialisiert. In Wirklichkeit sind diese Informationen viel komplexer organisiert und Teile davon stehen mit anderen Teilen logisch in Verbindung. Diese Struktur läßt sich am geeignetsten mit der Hilfe von Hypertext darstellen. Mit der Darstellung dieser Strukturen befaßt sich diese Arbeit.

Folgende Ziele stehen dabei im Vordergrund:

- Unter Berücksichtigung lernpsychologischer Überlegungen soll ein Wunschmodell eines Hypertextdokuments (z.B. als Begleitmaterial zu Vorlesungen) entwickelt werden.

- Es soll ein grundlegendes Hypertextdokument zum Thema Computergrafik erstellt werden, das diese Überlegungen verdeutlichen wird. Dieses Dokument soll in der Praxis am Fachbereich Informatik eingesetzt, im Rahmen weiterer Studienarbeiten erweitert und von Studenten getestet werden.

Diese Studienarbeit baut auf der Arbeit von Birgit Heyer und Dirk Platt [**?**] auf, die in ihrer Arbeit verschiedene Hypertextsysteme für die UNIX-Plattform auf ihre Tauglichkeit für dieses Projekt untersucht haben. Ein quantitativer Vergleich der verschiedenen Systeme Guide, HTML, Hyper-G und KMS ergab, daß KMS bezüglich der angesetzten Kriterien am geeignetsten sei. Hauptargument der Autoren ([**?**] S. 29):

> Sowohl HTML als auch (insbesondere) Hyper-G kämen von der Funktionalität her trotz eines geringeren „Punkt-Ergebnisses“ durchaus für einen Einsatz als Übungssystem in Frage, da die Einschränkungen beider Systeme (anscheinend) in einem akzeptablen Rahmen liegen. Beide sind jedoch für den Einsatz mit (welt-)weit verteilten Hypertexten vorgesehen und bieten daher dem Leser auch die *Möglichkeit*, durch explizite Angabe einer Zieladresse, den von uns *bereitgestellten Hypertext zu verlassen*. Diese Möglichkeit wäre aber beim Einsatz als Lernsystem im Sinne einer effizienten Nutzung äußerst hinderlich. Insbesondere diese Eigenschaft hat uns von einer Entscheidung für diese Systeme Abstand nehmen lassen.

Dieses Ergebnis wurde allerdings nicht in die vorliegende Arbeit einbezogen. Es sollte nicht in der Verantwortung des Autors liegen, den Leser auf seinen eigenen Themenbereich einzuschränken. Gerade die Möglichkeit, ein Dokument mit anderen zu verknüpfen, ist eine wichtige Eigenschaft von Hypertext. Auf diese Art können einzelne Themenbereiche Verknüpfungen mit Themenbereichen anderer Hypertextdokumente enthalten.

Zum Zeitpunkt der Arbeit von Birgit Heyer und Dirk Platt gab es HTML allerdings lediglich in einer grundlegenden — vom W3-Consortium entwickelten – Version. Die Einbindung von interaktiven Anwendungen war damit noch nicht möglich. Dank der von der Firma Sun entwickelten plattformunabhängigen Programmierumgebung **Ja-**

**va** steht jetzt auch diese Möglichkeit offen. Durch die Wahl von HTML wird sogar die Möglichkeit offenstehen, das Dokument — eventuell schon während seiner Entwicklung — der Internet-Öffentlichkeit zur Verfügung zu stellen.

Das von Birgit Heyer und Dirk Platt schließlich ausgewählte System KMS schien auf den ersten Blick durch eine eigene WYSIWYG (what you see is what you get)-Autorenumgebung besonders interessant zu sein. Leider stellte sich bereits nach kurzer inhaltlicher Arbeit mit der KMS-Autorenumgebung heraus, daß KMS (zumindest in der vorliegenden Version) bei größeren Dokumenten unübersichtlich bis unbenutzbar war.

Korrekterweise muß allerdings gesagt werden, daß es sich bei HTML nicht um eine Autorenumgebung sondern eine reine Beschreibungssprache handelt. Durch die zunehmende Entwicklung und Verbreitung des Internets (und damit von HTML) finden zukünftige Autoren allerdings eine Vielzahl von Dokumentationen und Hilfsprogrammen zu HTML, die die Erstellung von HTML-Seiten und Java-Applikationen erleichern.

# 2 Konzepte computerunterstützten Lernens

In diesem Kapitel sollen verschiedene Konzepte computerunterstützten Lernens und die Position von Hypertext in diesen Konzepten dargestellt werden. Außerdem werden die verschiedenen Komponenten eines Hypertextdokuments erklärt.

## 2.1 Lernumgebungen

Das Hauptgerüst des Hypertextdokumentes soll aus einem Hypertextdokument bestehen. Dennoch können in den Verfeinerungen andere Systeme eingesetzt werden. Hier soll eine Übersicht über gebräuchliche Lernumgebungen gegeben werden [**?**], in die auch Hypertext einzuordnen ist. Neue Autoren können mit Hilfe dieser Umgebungen interaktive Lernumgebungen schaffen, um z. B. abstrakte Zusammenhänge zu veranschaulichen.

### Tutorielles Lernen

*Tutorielles Lernen* ist eine komplexe intelligente Lernumgebung. Das System hält einen ständigen Dialog mit dem Lerner, um dynamisch den bereits erreichten Wissensstand zu ermitteln. Je nach Erfolg des Lerners wird der Lernstoff noch einmal wiederholt, zum nächsten Kapitel verzweigt oder auf Fehler hingewiesen. Dieses System benötigt nicht nur die eigentlichen Lehrinhalte sondern auch eine Verzweigungslogik, die vom Autor programmiert werden muß. In unserem Zusammenhang ist diese

Umgebung nicht sinnvoll. Der Student sollte selbst beurteilen können, welche Teile ihn interessieren und lediglich in seiner Auswahl unterstützt werden.

### Intelligente tutorielle Systeme

Eine Erweiterung der vorigen Umgebung sind *intelligente tutorielle Systeme*. Das Programm verfährt bei der Ermittlung des Wissensstandes des Lerners nicht nach einer vorgegebenen Verzweigungslogik sondern nutzt die Möglichkeiten der künstlichen Intelligenz, um den Lernfortschritt zu ermitteln.

> If I had to reduce all of educational psychology to just one principle I would say this: The most important single factor influencing learning is what the learner already knows. Ascertain this and teach him accordingly.
>
> *David P. Ausubel*

Diese Individualisierung wird durch intelligente tutorielle Systeme erreicht. Das Lernen ist am effektivsten, wenn dem System genau bekannt ist, was der Lerner bereits weiß und kompetent entscheiden kann, welches Thema am geeignetsten in Folge gelernt werden sollte.

### Drill

Den *Drill* könnte man als computergenerierte Prüfungssituation betrachten. Das Programm stellt wie ein Prüfer Fragen im Multiple-Choice-Verfahren an den Prüfling, um

dessen Wissensstand zu beurteilen. Durch Auswertung der Antworten entscheidet das Programm, ob weitere Fragen gestellt oder dem Lerner seine Lücken aufgezeigt werden. Dieses System ist zur Prüfungsvorbereitung interessant und könnte eine interessante Erweiterung des Hypertextdokumentes darstellen.

**Simulation**

*Simulationsprogramme* sollen Phänomene und komplexe Gesetzmäßigkeiten anschaulich verdeutlichen. Der Lerner kann bestimmte Parameter einer Simulation steuern und deren Einfluß beobachten. Diese Umgebung wird später durch interaktive Teile (z. B. in Java) realisiert. Der Leser wird z.B. besser in der Lage sein, ein Gefühl für die Parameter einer Splinefunktion zu bekommen, wenn er die Parametern verändern und die Auswirkungen direkt erkennen kann, als wenn er lediglich die Formel auswendig lernt. Diese Subumgebung ist im Kontext der Computergrafik sehr hilfreich, denn gerade hier gilt: „Ein Bild sagt mehr als tausend Wort."

**Lernspiele**

*Lernspiele* sind eigentlich kein eigenes Konzept sondern verwirklichen eine der anderen Umgebungen in einem spielerischen Zusammenhang, um den Lerner in seiner Arbeit zu motivieren. Das bekannte Spiel „Trivial Pursuit" ist beispielsweise ein typischer spielerischer Drill. Es bleibt den Autoren überlassen, den Leser stilistisch zu motivieren.

### Problemlösungsumgebungen

Eine Weiterentwicklung des Drills ist die *Problemlösungsumgebung*. Sie stellt dem Lerner eine komplexe Aufgabe, zu der der Lerner einen eigenen Lösungsweg ermitteln muß. Diese Methode dient dazu sicherzustellen, daß der Lerner nicht lediglich den Lernstoff auswendig gelernt hat, sondern diesen auch anwenden kann. Ebenso wie der Drill ist diese Methode in diesem Kontext nur schwer einsetzbar.

### Hypertext und Hypermedia

Ein *Hypertextdokument* hat an sich keine eigene Logik oder Intelligenz. Es ordnet Textfragmente entlang eines Graphen an. Das Hypertextsystem bietet Navigationsmittel an, die es dem Lerner erlauben, sich durch diesen Graphen zu bewegen und sich einzelne Textfragmente anzeigen zu lassen. Das Dokument, das im Rahmen dieser Studienarbeit entsteht, wird kein reines Hypertextdokument sein, denn die einzelnen Fragmente auf den Knoten des Graphen werden nicht nur Text sondern auch Grafik enthalten. Im Rahmen späterer (Studien-)Arbeiten wird das Hypertextdokument zu einem Dokument wachsen, in dem auch weitere Medien und Umgebungen (wie Simulationen) vorhanden sein werden.

## 2.2 Steuerung der Umgebung

Die oben genannten Lernumgebungen lassen sich nach ihrer Haupteigenschaft charakterisieren, nämlich ob der Lerner das System steuert oder umgekehrt.

**Systemsteuerung**

Zu den systemgesteuerten Lernumgebungen zählen Drill und tutorielles Lernen. Das System muß eine ausreichende Kompetenz besitzen, um die geeignete Reihenfolge der Lerninhalte zu ermitteln. Eine Systemsteuerung ist wünschenswert, wenn der Lerner noch nicht in der Lage ist, selbständig Inhalte auszuwählen oder durch den neuen Stoff geführt werden möchte. Der große Nachteil dieses Systems ist, daß der Lerner in keinem Fall Einfluß auf die Steuerung nehmen kann. Geht das System von einer falschen Voraussetzung aus, läßt sich dies häufig nicht manuell korrigieren.

**Lernersteuerung**

Simulationen, Lernspiele, Problemlösungsumgebungen und Hypertext zählen zu den lernergesteuerten Lernumgebungen. Selbst wenn das System eine gewisse Eigenintelligenz besitzt (wie z.B. intelligente tutorielle Systeme), so kann und muß der Lerner Einfluß auf die Steuerung nehmen. Der Extremfall dieser Steuerungsart ist Hypertext, da dieses System alleine auf die Auswahlentscheidungen des Lerners angewiesen ist. Deshalb muß dem Lerner aber klar die Struktur eines Hypertextdokuments und seine aktuelle Position im Dokument aufgezeigt werden.

## 2.3 Hypertext

Auch wenn der Einsatz von Hypertext den meisten Anwendern erst seit dem World-Wide-Web bewußt ist, so reichen die Wurzeln dieser Idee bis in die vierziger Jahre

zurück. Der Begriff *Hypertext* an sich wurde 1965 durch Ted Nelson geprägt:

> Let me introduce the word "hypertext" to mean a body of written or pictorial material interconnected in such a complex way that it could not conveniently be presented or represented on paper. It may contain summaries or maps of its contents and their intercorrelations; it may contain annotations, additions and footnotes from scholars who have examined it.

Nach seiner Idee könnte man auf diese Art das gesamte Wissen der Welt in einem gewaltigen Hypertextdokument anordnen. In gewisser Weise entspricht das World-Wide-Web im Internet dieser Idee.

In der Einleitung wurde der Vorteil von Hypertext gegenüber konventionellen Lehrtexten genannt: der nicht-sequentielle Aufbau, der die Informationen geeigneter darstellen kann. Diese Aussage ist natürlich nur relativ, denn kein Leser ist gezwungen, einen Lehrtext Seite für Seite zu lesen, sondern kann mit Hilfe des Index, des Inhaltsverzeichnis und mit Verweisen beliebig das Dokument durchstöbern. Immerhin ist die Information eindimensional angeordnet, was nicht der eigentlichen Informationsstruktur entspricht.

Hier setzen die Möglichkeiten von Hypertext an. Zunächst eine formale Definition des Begriffs Hypertext:

**Definition Hypertext:** Hypertext bezeichnet das Konzept, Wissen in Informationseinheiten zu zerlegen und mit Hilfe eines Graphen zu organisieren. Die Informa-

tionseinheiten werden als Textfragmente realisiert und auf Knoten des Graphen abgebildet.

Entsprechend lautet die Definition für ein Hypertextdokument:

**Definition Hypertextdokument:** Ein Hypertextdokument ist eine konkrete Menge von Textfragmenten, die in einem Graphen angeordnet sind. Die einzelnen Fragmente können auch andere Elemente als Text enthalten, solange die Gesamtstruktur gewahrt bleibt.

**Hypertextknoten**

Innerhalb des Graphen, der ein Hypertextdokument beinhaltet, unterscheidet man zwischen *inhaltlichen* und *organisierenden* Knoten. Ein inhaltlicher Knoten trägt dabei ein Textfragment bzw. repräsentiert einen Inhalt. Alle anderen Knoten sind organisierend. Im allgemeinen Fall liegen die einzelnen Seiten auf inhaltlichen Knoten, während Inhaltsverzeichnisse und Indizes auf organisierenden Knoten liegen. Die Trennung zwischen beiden Knotenarten ist nicht immer klar, da durch die Verknüpfung von Seiten nur selten eine eindeutige Trennung von Inhalt und Verzeichnissen zu sehen ist. In diesem Fall wird es nicht einmal rein inhaltliche Knoten geben, da von jedem Knoten aus eine einfache Navigation ermöglicht werden soll.

**Hypertextkanten**

Kanten (*Links*) in einem Hypertextdokument verbinden Knoten untereinander. Hier seien die über- und untergeordneten Kantentypen unterschieden:

- *Intensionale Kanten* sind nicht im Dokument selbst gespeichert sondern werden zur Laufzeit des Browsers ermittelt. Diese unterscheidet man wiederum in:
  - *Dynamische Kanten (Retrieval-Kanten)* sind die Ergebnisse komplexer Suchanfragen. Ein Beispiel für dynamische Kantengeneration wäre ein automatischer Schlagwortindex.
  - *Vokative Kanten* ergeben sich über namentliche Referenzen. Wären verschiedene Wörter einer Hypertextseite mit entsprechenden Lexikonseiten verbunden, so wären dies vokative Kanten.
- *Extensionale Kanten* sind explizit im Hypertextdokument vom Autor vorgegeben. Diese werden unterschieden in:
  - *Relationale Kanten (referierende Kanten)* sind eins-zu-eins-Kanten. Dabei verbinden assoziative Kanten beliebige Fragmente, während annotierende Kanten einzelne Elemente einer Seite mit einer Annotation verbinden.
  - *Inklusive Kanten (organisierende Kanten)* sind eins-zu-$n$-Kanten. DeRose unterscheidet hier noch weitere Kantenarten, die weit über das eigentliche Hypertextkonzept hinausreichen, um einzelne Fragmente, die in einer logischen Reihenfolge stehen, zu sequenzialieren.

**Organisierende Elemente**

*Organisierende Elemente* sind in diesem Fall organisierende Knoten und die damit verbundenen Kanten.

- *Hierarchisch* organisierende Elemente ordnen Textfragmente in einer Hierarchie an. Diese Elemente findet man meist in Inhaltsverzeichnissen. Dem Benutzer eines Browsers stellen sich in einem entsprechenden Verzeichnis alle Unterpunkte auf einen Blick dar.

- *Sequentiell* organisierende Elemente bilden eine künstliche Sequenz verschiedener Textfragmente. Die Sequenz ist eine vom Autor für sinnvoll gehaltene Reihenfolge. Dem Leser ist aber nicht auf einen Blick die Struktur dieser Sequenz deutlich, da nur der jeweils folgende Knoten direkt anwählbar ist.

# 3 Design von Lernumgebungen

Nachdem gängige Lernumgebungen im letzten Kapitel dargestellt wurden, sollen jetzt die Anforderungen von Autor und Leser eines Hypertextdokuments betrachtet werden.

## 3.1 Anforderungen des Autors

Hypertext ist ein sehr universelles Medium. Da dem Autor aber keine Strukturen vorgegeben sind, stehen ihm alle Möglichkeiten zur Gestaltung seines Dokumentes offen. Der Entwurfsvorgang geschieht top-down und gliedert sich in vier Phasen:

1. Analyse: Zuerst werden die vorhandenen Lerninhalte kategorisiert. Der gesamte Themenkomplex wird in Fragmente gespalten und die Zusammenhänge dieser Fragmente untereinander festgestellt. In diese erste Phase gehen noch keine didaktischen Überlegungen ein. Auch der Hypertext-Inhaltsgraph wird hier noch nicht definiert.

2. Synthese: Dann werden die Lernziele definiert. Die einzelnen Inhaltsfragmente werden durch Präsentationen (Lernumgebungen anderer Art wie z.B. Simulationen, Grafiken) ergänzt. Schließlich wird der Inhaltsgraph erstellt und die Fragmente miteinander verbunden.

3. Evaluierung: In der letzten Phase werden im Rahmen der Möglichkeiten des Hypertextsystems die Inhalte implementiert und vom Autor getestet.

4. Dokumentation: Um nachfolgenden Autoren die Metaarbeit am Hypertext zu ersparen, damit sich diese auf das Wesentliche ihrer speziellen Arbeit konzentrieren können, ist eine ausführliche Dokumentation (Styleguide) der erprobten Strukturen nötig.

## 3.2 Anforderungen des Lesers

Leider ist es hier nicht möglich, den typischen Benutzer des Hypertextsystems zu kategorisieren. Für einige grundlegende Betrachtungen unterscheiden wir vorerst zwischen Anfängern und Fortgeschrittenen.

Für den Anfänger ist es besonders wichtig, ihn nicht zu frustrieren. Er sollte auf einen Blick erkennen können, wie er eine Hierarchieebene zurücknavigiert bzw. ob er sich gerade in einem organisierenden oder inhaltlichen Knoten befindet. Ebenso sollte sofort deutlich sein, ob noch weitere Seiten folgen. Da das hierarchische Lesen eines Dokumentes ungewohnt ist, soll auf eine Systemsteuerung gewechselt werden können z.B. in Form einer Sequenzialisierung. Diese — auch „Guided Tour“ genannte — Steuerung zerlegt das hierarchisch aufgebaute Dokument in eine Sequenz von Seiten analog zur Struktur eines Buches. Möchte der Leser ein Buch einfach von vorne nach hinten lesen, so vermeidet er eine aufwendige rekursive Navigation.

Fortgeschrittene Benutzer werden mit dem Aufbau des Dokumentes bereits vertraut sein und möchten eher ohne großen Aufwand eine gewünschte Seite erreichen können. Zum direkten Nachschlagen oder zur Prüfungsvorbereitung empfiehlt sich

außerdem ein suchbarer Index.

# 4 Das Wunschmodell

In diesem Kapitel sollen Vor- und Nachteile von drei bestehenden Hypertextsystemen untersucht und ein Wunschmodell als Grundlage des späteren Dokumentes erstellt werden.

## 4.1 Beispiele bestehender Systeme

**HTML und World-Wide-Web**

Die HyperText Markup Language ist eine Beschreibungssprache für Hypertext-Dokumente aus der SGML-Gruppe (Standard Generalized Markup Language). Sie wurde im Rahmen des WWW-Projektes (World Wide Web) am CERN von Tim Berners-Lee entworfen und 1992 von Dan Connolly als Dokumenttyp (DTD = Document Type Definition) standardisiert. Das World Wide Web ist der Hypertext-Informationsdienst im Internet. Einzelne Hypertextseiten werden mittels HTML beschrieben und auf verschiedenen Servern im Internet abgelegt. Ein WWW-Browser interpretiert diese HTML-Daten und zeigt sie dem Benutzer an. WWW-Browser gibt es mittlerweile für viele Computersysteme.

Das World-Wide-Web ist ein Dienst des Internet. Die technische Struktur des Internets wird zwar zentral koordiniert, die Inhalte unterliegen allerdings keiner Vorschrift. Hypertextinhalte sehen also von Seite zu Seite verschieden aus, so daß man sich als Leser ständig der jeweiligen Struktur anpassen muß.

Damit in diesem gewaltigen Informationspool überhaupt noch Informationen gefun-

den werden können, stellen durch Sponsoren finanzierte Organisationen und Firmen sogenannte „Suchmaschinen“ zur Verfügung. Diese hochperformanten Datenbankrechner fragen ständig verschiedene Inhalte aus verschiedenen Servern ab, um einen Stichwortindex aufzubauen. Trotz der Milliarden Dokumente ist ein Finden gesuchter Inhalte innerhalb weniger Sekunden möglich.

Die Inhalte sind so verschieden wie ihre Autoren. Sie unterliegen keiner Formvorschrift (Styleguide). Auch herrscht allgemeine Uneinheit über die Features von HTML. Am CERN werden zwar die Standards für HTML gesetzt, aber die Browser verschiedener Firmen führen mit jeder Version neue de-facto-Tags (Elemente der HTML) ein, so daß auch hier — wie in anderen Bereichen der Informatik — Standard und Praxis häufig weit auseinander liegen. Derzeit kämpfen die Firmen Netscape und Microsoft um die Vormachtstellung ihrer HTML-Browser.

Würden die derzeitigen Möglichkeiten von HTML (aktuelle Version 3.2) von allen verfügbaren Browsern unterstützt, so könnte der Leser außer Text auch Animationen, Tonwiedergabe und telefon-ähnliche Kommunikation nutzen. Durch die Interaktion von Browsern und Servern können theoretisch beliebig komplexe Hypertextumgebungen geschaffen werden.

Vorteile von HTML:

- Hypertext-Browser sind für viele Computersysteme verfügbar. Durch die Verwendung von Java sind plattformunabhängige Anwendungen möglich.
- Die meisten Browser erlauben das Anlegen von Lesezeichen (Bookmarks), um

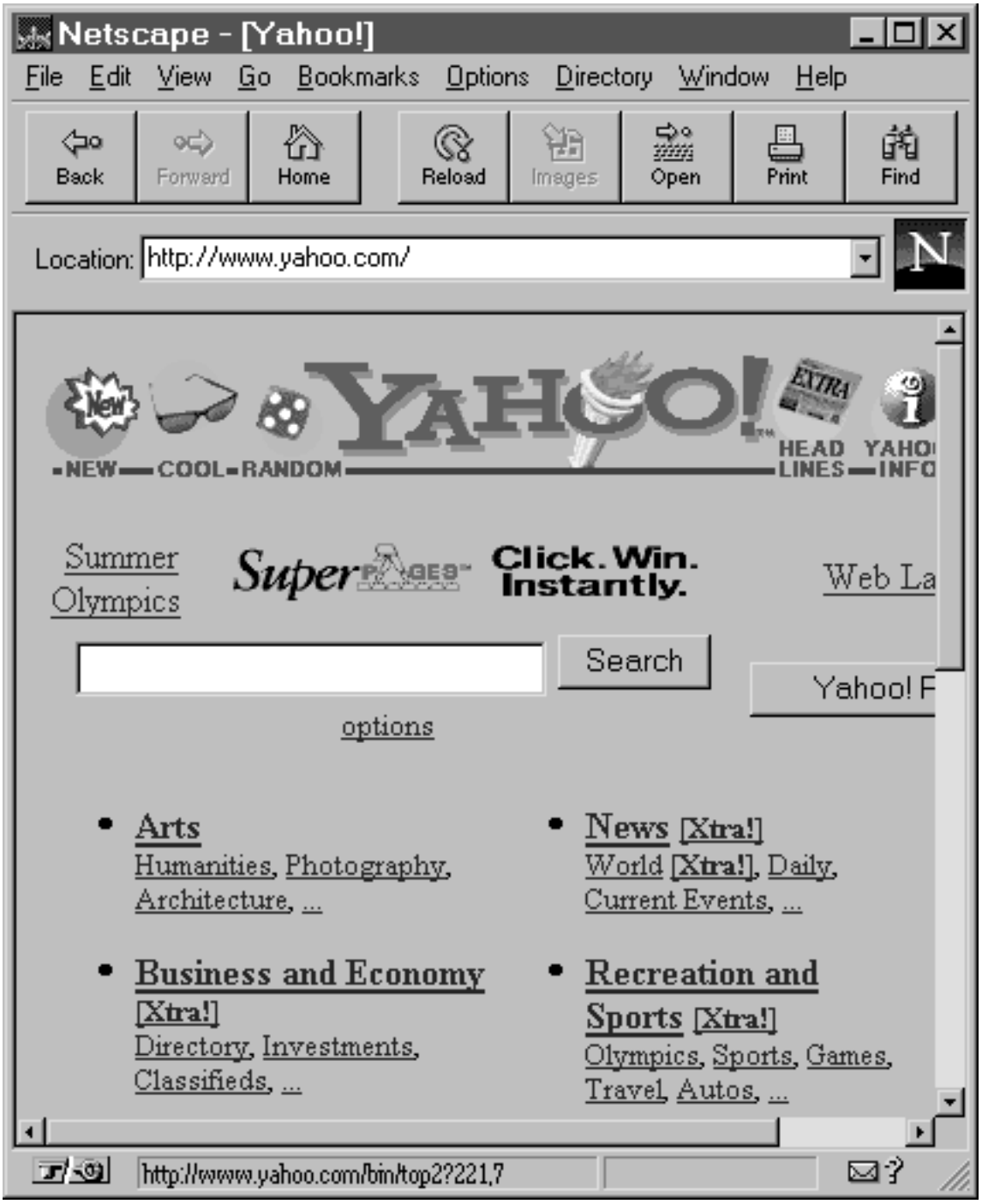


Abbildung 1: WWW-Browser der Firma Netscape

4. *Das Wunschmodell*

   auf verschiedene Dokumente und Inhalte schnell zugreifen zu können.

Nachteile von HTML:

- Durch das weltweit verteilte System und teilweise niedrige Übertragungsbandbreiten erschweren lange Lade- und Suchzeiten das zügige Lesen.
- Es gibt kein standardisiertes Design. Mit Hilfe von HTML lassen sich beinahe beliebige Seiten realisieren. Da sich der Leser bei fast jeder neuen Seite neu auf die Strukturierung des Autors einstellen muß, gestaltet sich das Lesen und Navigieren teilweise schwierig.
- Hypertextseiten können praktisch beliegig lang sein und zwingen den Autor nicht zur Gliederung seiner Inhalte.

**Microsoft-Hilfesystem**

Um auf umfangreiche Handbücher zu verzichten, bietet die grafische Bedienoberfläche „Windows“ der Firma Microsoft ein integriertes Hypertext-Hilfesystem an. So umstritten Windows auch sein mag, das Hilfesystem ist professionell gestaltet.

Hier können Hilfstexte auf drei Ebenen gesucht werden: Inhalt, Index und Suchen. Auf der Ebene *Inhalt* sind vom Autor bereits übergeordnete Themen eingetragen worden, die sich bei Anwahl in weitere Unterpunkte zu diesem Thema untergliedern lassen. Die Ebene *Index* sammelt alle wichtigen Begriffe und faßt sie zu einer thematischen Suchliste zusammen. Vom Benutzer anpassen läßt sich die Ebene *Suchen*, da jeweils

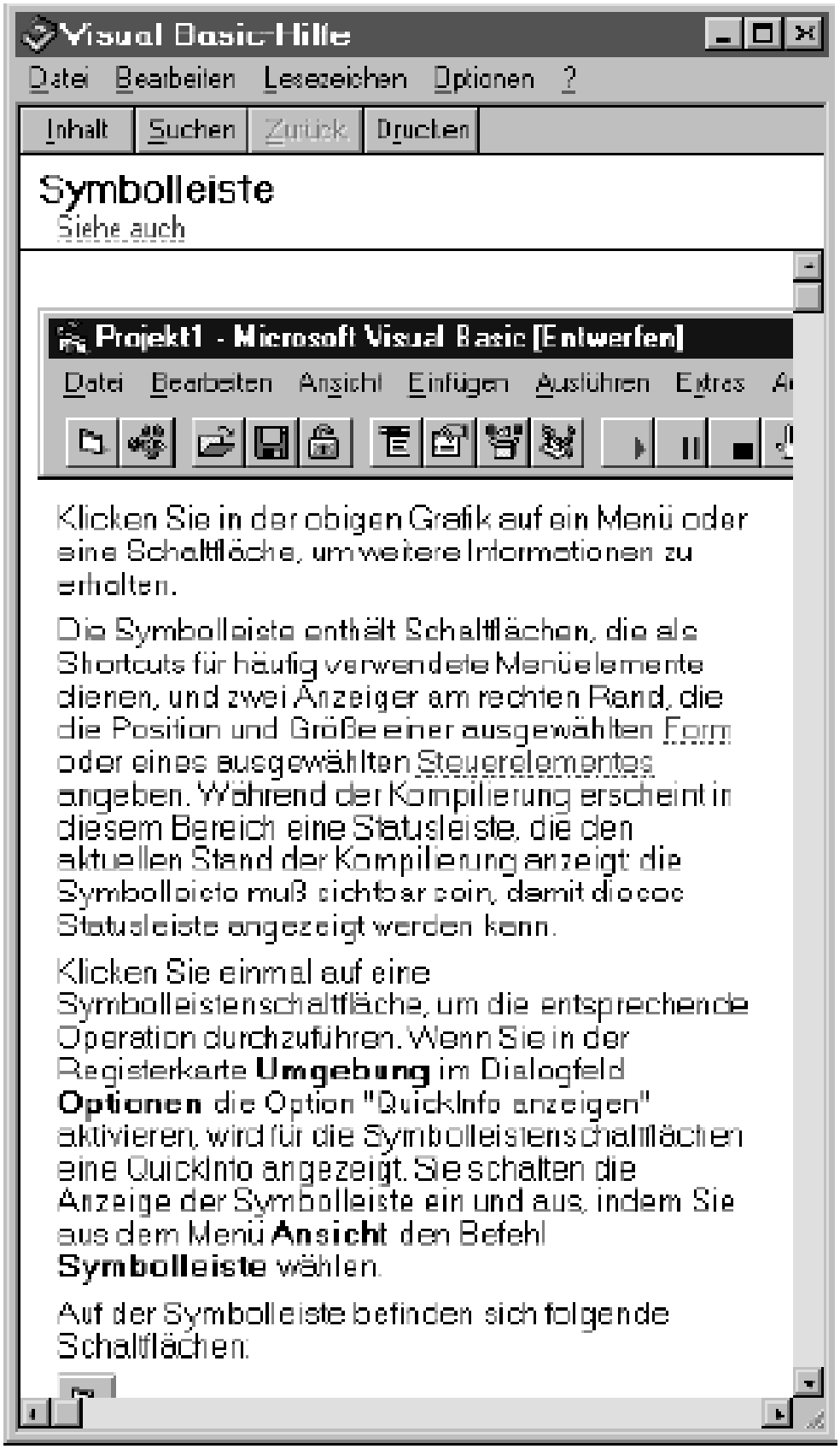


Abbildung 2: Hypertext-Browser der Oberfläche Microsoft Windows

anzugeben ist, ob eine kurze, mittlere oder lange Suchdatenbank erstellt werden soll. Bei der langen Version wird jedes einzelne Wort als Index eingetragen. Der Benutzer kann in der Suchebene eine umgangssprachlich formulierte Frage an das System stellen und erhält eine Liste mehr oder weniger zutreffender Hilfeseiten.

Interessant sind auch die „Hottexts“ — eine Unterebene der Links. Ein punktiert unterstriches Wort läßt sich mit der Maus anklicken und öffnet ein Fenster mit einer kurzen Erklärung zu einem Begriff. Wird die Maustaste losgelassen, verschwindet dieses Fenster wieder. So lassen sich kurze Erklärungen einstreuen, ohne daß Leser aufwendig navigieren muß.

Auch hier lassen sich Lesezeichen (Bookmarks) anlegen. Jeder Benutzer kann sich sogar Anmerkungen in den Text schreiben, die farblich hervorgehoben und natürlich auch gespeichert werden.

Außer Text können hier beliebige Arten von Multimedia eingesetzt werden. Es lassen sich sogar fremde Applikationen starten, so daß sich auch hier viele andere Arten von Lernumgebungen realisieren ließen.

**KMS**

KMS wurde von der Firma Knowledge Systems Incorporated entwickelt (die vorliegende Version stammt von 1992).

Dieses System hat gegenüber den meisten anderen Hypertextsystemen einen großen Vorteil: das Design der einzelnen Seiten erfolgt direkt am Bildschirm, wo an-

dere Systeme eine Art Programmiersprache verwenden, die von den entsprechenden Browsern interpretiert wird. Hier fällt auch ein anderes Merkmal angenehm auf: eine Bildschirmseite ist eine festgelegte Darstellungseinheit. Somit werden zu lange Seiten vermieden und das Erscheinungsbild bleibt für den Leser gleichbleibend. So positiv dieses Konzept auch ist, so stellte sich schon nach kurzer Arbeitszeit mit dem eingebauten Editor heraus, daß umfangreiche Bearbeitungen zu einem Geduldspiel werden. Die Programmierer haben anscheinend mehr Zeit an das Konzept verwendet, als das Programm vom Standpunkt der Autoren zu testen.

Für größere Projekte kann man KMS als ungeeignet bezeichnen. Einerseits werden einmal vergebene Seitennamen nach Löschen einer Seite als belegt markiert, so daß die Numerierung der Seiten nichts mehr mit der eigentlichen Struktur des Dokumentes gemein hat, sobald erst einmal einige Änderungen gemacht werden. Am schwersten nachvollziehbar ist aber die Tatsache, daß nur ganze Textbausteine (Absätze) als ein Verweis auf eine andere Seite betrachtet werden können. Es macht im Kontext keinen Sinn, einen ganzen Absatz anzuwählen. Die Textverarbeitung des Systems ist rudimentär gestaltet wie die Textfunktion eines Malprogramms — ein automatischer Zeilenumbruch fehlt ebenso wie das direkte Verändern und Anzeigen der Eigenschaften eines Textes. Hier gilt: entweder notieren oder löschen und neu eingeben.

Vorteile von KMS:

- KMS ist keine Beschreibungssprache, sondern besteht aus einer WYSIWYG (what you see is what you get)-Autorenumgebung. Der Autor hat bereits in der

Designphase einen Überblick über das spätere Aussehen des Dokumentes.

- Der Autor wird zur Gliederung seines Inhalts gezwungen, da die Darstellungseinheit auf jeweils eine Bildschirmseite beschränkt ist.

- Das Design der Umgebung erlaubt das gleichzeitige Blättern auf zwei Seiten.

- In der UNIX-Umgebung ist eine Einbindung externer Programme möglich.

Nachteile von KMS:

- KMS erlaubt keine automatischer Indizierung von Schlagwörtern.

- Die WYSIWYG-Umgebung ist unergonomisch und unzureichend implementiert. Sowohl die Fähigkeiten zur Texteingabe als auch zur Eingabe von Zeichnungen und Skizzen sind extrem rudimentär.

- Es sind keine Hypertext-Links einsetzbar, da die kleinste Textmenge für einen Link bereits ein Absatz ist, was dem Hypertextkonzept widerspricht.

## 4.2 Ansprüche an das Hypertextdokument

Aus der Vorstellung dieser drei Systeme sollen jetzt die Vorteile extrahiert und die Nachteile möglichst vermieden werden. Außerdem sollen diese Punkte vor dem Hintergrund analysiert werden, daß das Hypertextsystem HTML bestimmten Beschränkungen unterliegt.

**Browser für alle Computersysteme verfügbar machen**

Da - wie bereits erwähnt - für HTML auf nahezu jedem System ein Browser verfügbar ist (vorzugsweise Netscape), kann das Dokument nicht nur am Rechenzentrum lokal sondern auch global über das Internet abgerufen werden. Spezielle Systembedingungen werden nicht gestellt. Lediglich eine aktuelle Version des Netscape Browsers (3.0 oder höher) ist nötig, damit die Java-Applikationen reibungslos funktionieren können. Leider ist es auch nicht möglich, mit vertretbarem Aufwand eine Version für andere Browser zu erstellen wie z.B. den textbasierten Lynx.

**Automatischen Index aufbauen**

Zu diesem Zweck wird das Programm „GlimpseHTTP“ eingesetzt. Es wurde an der Universität von Arizona entwickelt und erlaubt eine einfache Indizierung des gesamten Dokumentes. Somit läßt sich zwar kein Listenindex erstellen, dafür kann der Leser aber im gesamten Dokument komfortabel nach Schlagwörtern suchen.

**Komplexe Umgebungen**

Teil dieser Studienarbeit ist es nicht, umfangreiche Umgebungen zu erstellen. Es soll lediglich sichergestellt werden, daß das gewählte Medium in der Lage sein kann, komplexere Umgebungen als Hypertext zu implementieren. Dank der Programmiersprache Java werden hier beliebige Strukturen möglich sein, so daß auf diesem Wege beliebige andere Umgebungen wie Drills oder Simulationen programmiert werden

können.

### Lesezeichen

Jeder WWW-Browser erlaubt das Speichern von Lesezeichen (Bookmarks). Diese lassen sich (am Beispiel von Netscape) über ein Menü zu jedem Zeitpunkt anwählen. Da diese Daten lokal im Benutzerbereich gespeichert werden, kann jeder Leser seine eigenen Lesezeichen erstellen. Lediglich bei der Verwendung von Frames (Aufspaltung einer Hypertextseite) kommt es zu Schwierigkeiten, da das Lesezeichen nicht auf die zuletzt veränderte Seite zeigen würde.

### Lange Suchzeiten vermeiden

Das Hypertextdokument soll möglichst wenige Hierarchieebenen beinhalten, um eine zu starke Verschachtelung zu vermeiden.

### Styleguide

Im folgenden Kapitel wird ein Styleguide vorgestellt, der sich direkt aus den hier behandelten Punkten ergibt. Er soll garantieren, daß mehrere Autoren das einheitliche Design des Hypertextdokumentes erhalten.

**Hypertextseiten übersichtlich halten**

Die Hypertextseiten sollen ein möglichst gleichmäßiges Erscheinungsbild zeigen. Deshalb wird mit Hilfe der Frametechnik - mit der sich Seiten in unabhängige Unterbereiche teilen lassen - ein Inhaltsbereich und eine Kopfzeile erstellt, die zur Navigation dienen soll. Genaue Maße für die Menge der Informationen auf der Mittelseite lassen sich nicht festlegen. Schwerpunktmäßig soll aber eine Seite eher in mehrere Seiten unterteilt und mit Links verbunden werden, als daß eine einzige Seite zu lang wird und der Leser zuviel scrollen muß. Das Design der Navigationsleisten wird im Styleguide näher erläutert.

**Pfad immer übersichtlich halten**

Interessant wäre es, den Pfad auf die aktuelle Hypertextseite ständig im oberen Frame sichtbar zu machen. Diese sogenannten „Fisheye Views“ erlauben eine bessere Übersicht für den Leser. Leider ist es aber mit vertretbarem Aufwand nicht möglich, zwei Frames gleichzeitig zu verändern. Deshalb ist es wichtig, dem Benutzer möglichst deutlich seine Position im Dokument aufzuzeigen.

**Anmerkungen im Dokument**

Da dem Leser keine Rechte zur Veränderung des Dokumentes eingeräumt werden, können auch keine Anmerkungen vorgenommen werden. Es lassen sich lediglich Kommentare zu den Lesezeichen (Bookmarks) im Browser eingeben.

# 5 Umsetzung in HTML

Aus den Vorüberlegungen der letzten Kapitel wird in diesem Kapitel auf die Umsetzung der Ideen und Inhalte in HTML eingegangen. Hintergrund ist die Erstellung eines Hypertextdokumentes als Modell einer computergestützten Lernumgebung.

## 5.1 Styleguide

### Seitenaufbau

Der obere Frame einer Seite (siehe Seite **??**) ändert sich nicht und soll nur zur Navigation dienen. Dort befinden sich die Wörter *Inhalt*, *Info*, *Suchen* und *Hilfe*. Der Verweis *Inhalt* soll auf das Inhaltsverzeichnis des Hypertextdokuments zeigen. *Info* führt den Leser zu einer Seite mit Informationen zu diesem Projekt und deren Autoren. Mittels *Suchen* kann der automatisch erstellte Suchindex durchsucht werden. Zu diesem Zweck wird das Hilfsprogramm GlimpseHTTP der Universität Arizona eingesetzt. Die *Hilfe*-Seite erklärt noch einmal die Benutzung der Seiten und die Bedeutung der Symbole.

Im unteren Frame werden die eigentlichen Inhalte des Dokuments angezeigt.

### Dateistruktur

In der ersten Phase ist das Projekt auf dem Rechner

`ant6.informatik.uni-hamburg.de`

eingerichtet. Alle Beteiligten an diesem Projekt benötigen einen Account der Projektgruppe `ant_4`. Im Verzeichnis `~anttutor/WWW/cg/` befindet sich folgende Verzeichnisstruktur:

| | |
|---|---|
| `cg/` | Eigentliche Dokumentinhalte |
| `grafiken/` | Icons und wiederkehrende Grafiken |
| `hilfe/` | Inhalt der Hilfe-Seite |
| `index.html` | Startdokument (wird in den Browser geladen) |
| `info/` | Inhalt der Info-Seite |
| `suchen/` | Inhalt der Suchen-Seite |

**Dateiendungen**

Alle HTML-Dateien erhalten den Suffix `.html`.

**Schriftgrößen**

Hauptüberschriften werden mit dem Tag `<h1 align=center>` eingeleitet und mit `</h1>` abgeschlossen. Hauptüberschriften kommen meist nur einmal pro Seite vor.

Unterüberschriften werden mit dem Tag <h3> eingeleitet und mit </h3> abgeschlossen.

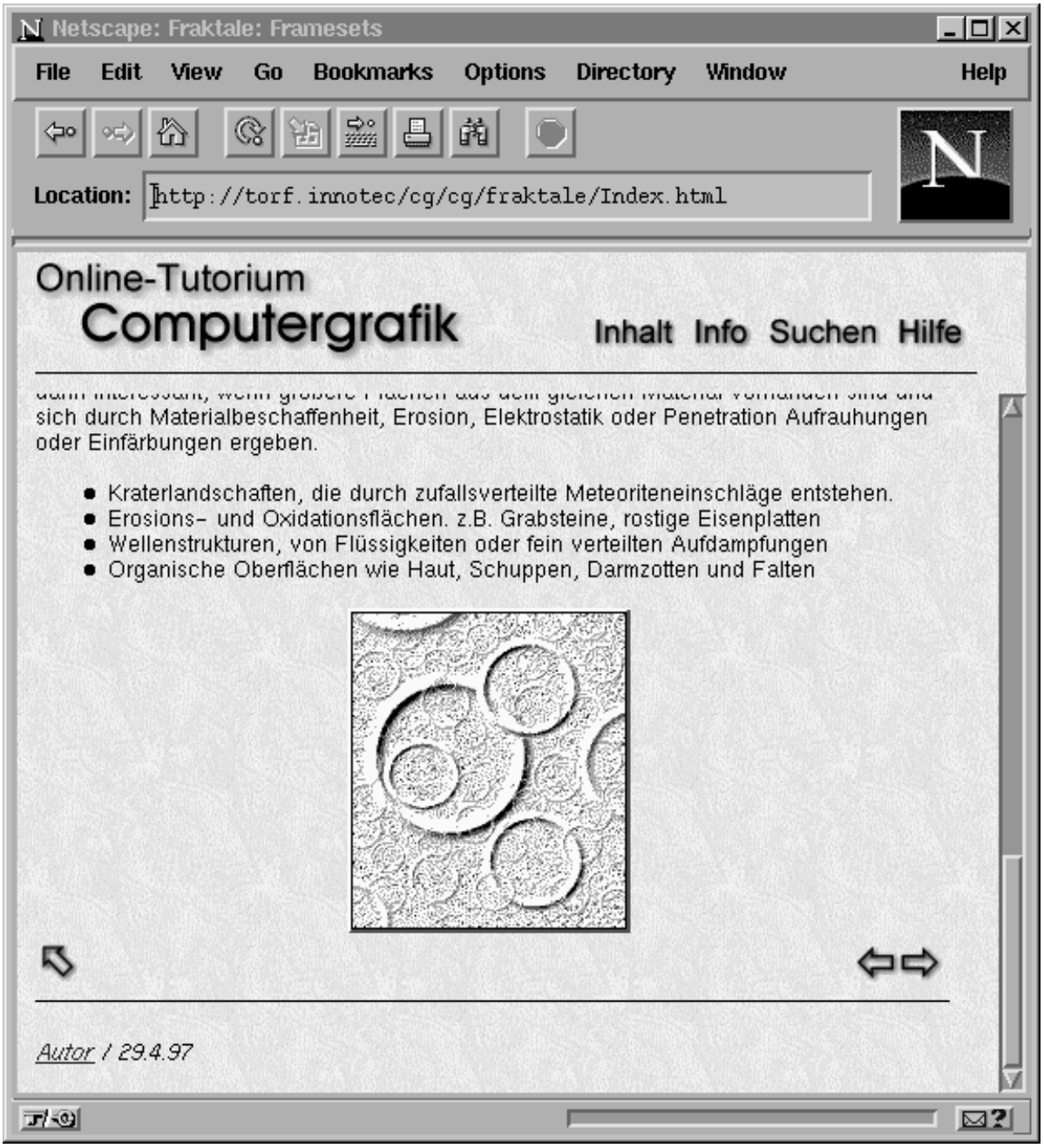


Abbildung 3: Beispielseite des Hypertextdokumentes

**Schriftarten**

Es wird keine Schriftart vorgegeben (z.B. Mittels des Tags `<font face=arial,helvetica>`. Der Leser soll selbst entscheiden können, wie ihm das Dokument angezeigt wird. Es empfiehlt sich allerdings erfahrungsgemäß, im Sinne der Lesbarkeit die Standardschriftart des Browsers auf Arial oder Helvetica umzustellen. Die häufig voreinstellte Schriftart Times ist bei normalen Schriftgrößen sehr schwer lesbar.

**Jeder Autor erhält ein eigenes Verzeichnis**

Damit das Gesamtprojekt auch ohne Organisation auf Dauer übersichtlich bleibt, legt jeder neue Autor nur ein Verzeichnis im cg-Verzeichnis für sein Projekt an. Er erstellt lediglich einen neuen Eintrag im Haupt-Inhaltsverzeichnis, der auf das eigene Inhalts-verzeichnis verweist, berührt sonst aber keine anderen Verzeichnisse. Die Startseite eines Verzeichnisses sollte immer `haupt.html` heißen.

**Frame-Ziele**

Da keine Notwendigkeit besteht, andere Frames als den Inhalts-Frame zu verändern, sollen nach Möglichkeit keine Frame-Ziele wie `_top` oder `_TOP` verwendet werden. Es steht dem Autor allerdings frei, den Inhalts-Frame aufzuteilen.

### Hintergrund

Als Hintergrund sollte die Grafik `/cg/grafiken/hintergrund.gif` verwendet werden, um ein gleichmäßiges Erscheinungsbild zu gewährleisten.

### Anrede des Lesers

Muß der Leser selbst angesprochen werden, so ist dieser mittels „Sie“ und „Ihnen“ zu titulieren.

### Pfeile anpassen

Die Pfeile, mit deren Hilfe der Leser eine Ebene zurück navigieren und die *Guided Tour* verwenden kann, müssen auf jeder Seite angepaßt werden. Führt in eine Richtung kein Pfeil (z.B. der Linkspfeil auf der ersten Seite bzw. der Rechtspfeil auf der letzten Seite eines Kapitels) so wird dieser Pfeil als inaktiv dargestellt. Zu diesem Zweck verwendet man statt der Grafiken (`parent.gif`,) `vor.gif` und `zurueck.gif` die Grafiken `vor0.gif` und `zurueck0.gif`.

### Relative Pfade

Innerhalb eines Kapitels (also der Arbeit eines Autoren) sind lediglich relative Pfade zu benutzen, damit ein gesamtes Kapitel später an einen anderen Ort verlagert werden kann. Lediglich Verweise auf allgemeingültige Grafiken oder die Hauptseiten sind absolut anzugeben.

**Sauberes HTML**

Auch wenn gängige Browser wie der Netscape Navigator keinere Fehler im HTML-Code verzeihen, so sollte immer darauf geachtet werden, daß der Code syntaktisch einwandfrei ist. Zu öffnenden Container-Tags gehören z.B. immer die schließenden Tags.

**Trennstriche**

Entgegen den Standardeinstellungen sollen Trennstriche (horizontal rules) nicht schattiert sein und die Größe eins haben (`<hr size=1 noshade>`).

## 5.2 Einbinden eigener Seiten

Autoren, die Seiten in das Hypertextdokument einfügen möchten, finden im Stammverzeichnis die Datei `vorlage.html`, in die eigene Inhalte übernommen werden können.

Folgende Schritte sind dafür auszuführen:

1. Bitte im Verzeichnisbaum im Verzeichnis `cg` einen geeigneten Pfad auswählen und ein neues Verzeichnis mit einem allgemeinverständlichen Namen erstellen.
2. Jetzt kann der eigentliche Inhalt der Seite in der Datei `text.html` eingefügt werden. Bitte darauf achten, daß der Inhalt nicht zu lang wird, um zu intensives Scrollen zu vermeiden.

3. Diese Seite muß jetzt noch mit dem Haupt-Inhaltsverzeichnis (im Falle der Hauptseite) bzw. mit den anderen eigenen Seiten mittels Verweisen (Links) verbunden sein.

## 5.3 Hinweise für Leser

- Als Browser wird Netscape 3.0 (oder neuer) empfohlen. Zu bekommen ist dieses Programm kostenlos (für Studenten) über FTP: `ftp.netscape.com`. Am Fachbereich ist es bereits installiert und kann direkt von einer Shell aus durch Eingabe von `netscape` gestartet werden.

- Da die Grafiken im Dokument teilweise mehr als 256 Farben verwenden, aber die jeweilige grafische Bedienoberfläche (Windows, XWindow...) eventuell für weniger Farben konfiguriert ist, empfiehlt es sich in der Konfiguration entweder eine eigene Farbtabelle (colormap) zu aktivieren oder das Dithering von Grafiken einzuschalten. Netscape muß zu diesem Zweck mit der Option „–install“ starten bzw. im Menü Options/General Preferences/Images der Punkt „Dithering“ anwählt werden.

- Bis das Projekt auf den öffentlichen Webserver des Fachbereichs übertragen wird, ist das Dokument über folgenden URL erreichbar:
  `http://ant6.informatik.uni-hamburg.de:4242`.

- Das Browser-Fenster sollte so weit wie möglich verbreitert werden, um eine bessere Übersicht zu gewährleisten.

# Literatur

[fellner92] W. D. Fellner: Computergrafik (BI-Wissenschaftsverlag Mannheim; Leipzig; Wien – 1992)

[scott83] J. E. Scott: Interaktive Computergrafik (Verlag Rudolf Müller – 1983)

[meyerh83] D. B. Meyerhoff: Hypertext und tutorielle Lernumgebungen: Ein Ansatz zur Integration (R. Oldenbourg Verlag – GMD-Bericht Nr. 223 – 1993)

[heypla95] B. Heyer, D. Platt: Hypertextsysteme auf UNIX-Rechnern (Universität Hamburg – Oktober 1995)