3 Konzept des webbasierten Lernsystems „I-Light“

3.1 Lernmethode

Wie im Abschnitt Handlungsbedarf bereits festgestellt wurde, muß auf dem Gebiet der architektonischen Lichtplanung die Vermittlung von Anwendungswissen verbessert werden. Die Vermittlung von Faktenwissen zu Grundlagen und Produkten erscheint weniger verbesserungsbedürftig, da es auch theoretisch erlernt werden kann, wofür die vorhandenen Lernmittel, wie die klassischen Printmedien ausreichen.

Es geht beim Anwendungswissen im Prinzip um das Wissen, wie Licht als gestaltgebender Baustoff in der Architektur angewendet werden kann. D.h. vor allem um die Integration der visuellen Wirkung sowohl des Lichts (Software), als auch der Anordnung und Form von Leuchten oder Fenstern (Hardware) in der Architektur. Immer wieder auftretende Fragestellungen sind z.B. „wie erreicht man eine gewünschte Lichtwirkung“ oder „wie beeinflussen Oberflächen den Lichtverlauf“ oder „was kann man mit lichtlenkenden Elementen bewirken“. Diese gilt es zu beantworten. Dies ist umso schwieriger, weil eine Lichtwirkung nicht nur von der Lichtquelle allein, sondern von der komplexen Wechselwirkung mit Oberfläche und Form abhängt. Die Faktoren dürfen also nicht einzeln, sondern nur in Verbindung untereinander gesehen werden.

In der bisherigen Praxis der Hochschulangebote an architektonischer Lichtplanung wird solches Anwendungswissen in Demonstrationsräumen (Mockup-Räumen oder Licht- bzw. Raumlaboren) durch praktische Veranschaulichung vermittelt. Hier können z.B. verschiedene Beleuchtungssituationen bzw. Leuchten geschaltet oder die Decke verschoben werden. Möglichkeiten zum Selbststudium mit weiterführenden Versuchen auch außerhalb der organisierten Besuchszeiten sind allerdings aus den schon erwähnten betriebstechnischen und wirtschaftlichen Einschränkungen kaum möglich.

Dieses vertiefende Selbststudium soll mit diesem Lernsystem möglich werden. Die Lernmethode des entdeckenden Lernens soll in den *virtuellen Lichtlaboren* konsequent auf eine breitere Basis gestellt werden. Der Umzug ins „Virtuelle“ bringt für das entdeckende Lernen folgende Vorteile:

- *Virtuelle Lichtlabore* sind einfacher zugänglich. Es gibt keine an den Arbeitszeiten festgemachte Öffnungszeiten. Außerdem entfällt der Reiseaufwand. Der Zugriff auf *virtuelle Lichtlabore* ist also zeit- und ortsunabhängig.
- Die Freiheitsgrade der Änderungen sind größer. Das Verschieben einer Decke oder der Austausch von Leuchten ist im realen Lichtlabor aufwendig. Im virtuellen Lichtlabor ist dieser Aufwand kleiner - man braucht beispielsweise keine hydraulischen Maschinen - und damit sind die Änderungen auch schneller zu bewerkstelligen. Darüberhinaus können jetzt auch nicht nur Einbauelemente, sondern auch tragende Elemente wie beispielsweise Wände und deren Materialität verändert werden. Damit kann der Charakter des Raums komplett verändert werden: ein dunkeler, schmaler und langer Raum kann zu einem hellen, hohen und weiten Raum verändert werden. D.h., es können leicht unterschiedliche Raumproportionen z.B. eines Flurs oder eines Foyers erstellt werden.
- Weil diese Änderungen einfacher zu bewerkstelligen sind, können auch Versuchsreihen durchgeführt werden, in denen man eine Kenngröße (Licht, Material, Form) isoliert verändern und in ihrem Einfluss auf die Lichtwirkung studieren kann.
- Die Variabilität ermöglicht es neben Leuchten auch Fenster einzufügen und so die Lichtuntersuchungen auf das Tageslicht oder auf die Kombination elektrisches Licht/Tageslicht auszuweiten.
- Da die Dokumentationen der Versuche in den *virtuellen Lichtlaboren* digital vorhanden sind, lässt sich ihre Zusammenstellung automatisieren. Das verbessert die Vergleichbarkeit der einzelnen Ergebnisse und stellt die Erkenntnisse auf eine objektive Basis.
- Mehrere Benutzer sind gleichzeitig möglich, da jeder Benutzer automatisch mit einer Kopie des *virtuellen Lichtlabors* arbeitet.
- Die Darstellung muss nicht auf die visuelle Wirkung des Licht beschränkt sein. Im *virtuellen Lichtlabor* können vielmehr Darstellungen aus didaktischen Gründen ergänzt, bzw. überlagert oder ausgeschaltet werden, was im realen Lichtlabor nicht geht. Dazu gehört das an den Schemata der Lichtplanung angelehnte Darstellen der Lichtkegel. Aber auch die Ausblendung des indirekten Lichts, des Licht also, das von den Oberflächen zurückgeworfen wird.

Die der Lernmethode zugrundeliegende Lernsoftwaretypologie der Simulation [STE-99] ergibt sich aus den Ausführungen der vorherigen Kapitel. Hier soll nur noch mal

aufgeführt werden, dass die Simulation als Exploration das entdeckende Lernen durch aktive Einbeziehung des Benutzers fördert. *„Tell me and I will forget, teach me and I will remember, involve me and I will understand”*[1].

Die virtuellen Lichtlabore haben die Gestalt verschiedener architektonischer Räume. Sie sollen typische Architekturräume sein, die aber in einer möglichst reduzierten Gestalt allgemeingültige Rückschlüsse ermöglichen, die dann auf individuelle und detailliertere Projekte übertragen werden können. Daher sollen die virtuellen Lichtlabore in Form von architektonischen Planungsbeispielen angeboten werden.

Ein weiterer Aspekt der Lernmethode ist die Unterscheidung in Exploration und Präsentation bei den *virtuellen Lichtlaboren*. Die Exploration soll gekennzeichnet sein durch die Möglichkeiten, das dreidimensionale Planungsbeispiel interaktiv verändern und durchwandern zu können. Hier soll das Licht und die Architektur abstrakt dargestellt werden, ganz in der Tradition der lichtplanerischen Beleuchtungskonzeptdarstellung, nur mit dem Unterschied, dass sie hier in eine räumliche Dimension erweitert wurde. Ergänzt wird die Exploration durch eine Präsentation der fotorealistischen Abbildung des Lichts und der Architektur, die vor allem die visuelle Wirkung des Lichts aufzeigen soll. Diese *duale Darstellung* von „Ursache“ und „Wirkung“ des Lichts soll beide in einen sich gegenseitig ergänzenden Zusammenhang stellen. Aus der abstrakten lichtplanerischen Darstellung kann nicht unmittelbar die Lichtwirkung abgeleitet werden und umgekehrt von der visuellen Wirkung nicht die Position und Anordnung der Lichtquellen und ihrer Lichtausbreitung. Die beiden Darstellungen sollen von verschiedenen selbstentwickelten Programmen ermöglicht werden. Die Exploration mit dem *Szeneneditor* und die Präsentation mit dem *Szenenviewer*.

Die fotorealistische Darstellung der interaktiv erstellten Planungsbeispiele kann nur über computerunterstützte Techniken zur Lichtsimulation erreicht werden. Ansonsten müsste jede denkbare Konstellation eines interaktiv erstellten virtuellen Planungsbeispiels schon mal im Vorfeld real gebaut und fotografiert worden sein, was kaum zu bewerkstelligen ist. Die Präsentation der fotorealistischen Simulation eines Planungsbeispiels soll hier nicht als das Ziel angesehen werden, sondern vielmehr als ein Instrument zur qualitativen Evaluierung eines Beleuchtungsentwurfs. Sie ist Teil des Entwurfs und nicht Mittel zur Präsentation eines abgeschlossenen Entwurfs. Die Rückschlüsse aus den Simulationen fließen in die Änderung des Beleuchtungsentwurfs ein und tragen zu einer Optimierung oder kreativen Erweiterung des Entwurfs bei. Die

[1] B. Franklin, us-amerikanischer Gelehrter und Politiker, *1706 Boston

Simulation muss dabei natürlich so einfach wie möglich zu erstellen sein, damit sie immer wieder zur Überprüfung, Optimierung und Generierung von Beleuchtungsentwürfen herangezogen werden kann. Insofern soll die Simulation nicht nach dem Abschluss eines Entwurf zu dessen verkaufsorientierter Präsentation eingesetzt, sondern bereits während des Entwurfs als Evaluationsinstrument integriert werden.

3.2 Funktionelle Anforderungen

Dieser Abschnitt erläutert die funktionellen Anforderungen an die Exploration und Präsentation der virtuellen Lichtlabore aus der Sicht des Benutzers. Eine Zusammenfassung unter den Gesichtspunkten der Softwareentwicklung befindet sich als Pflichtenheft im Anhang.

Der Benutzer soll über das WWW auf das Lernsystem mit den virtuellen Lichtlaboren zugreifen können. Sie sollen vorher vom Administrator des Systems eine Zugriffsberechtigung erhalten. Sie regeln, dass verschiedene Benutzer jeweils ihre individuellen Daten speichern und laden können. Bei der Einwahl in das Systems sollen die notwendigen Programme möglichst automatisch dem Benutzer zur Verfügung stehen. Auf eine lokale Installation von CAD, Lichtsimulations- und Dokumentationsprogrammen soll verzichtet werden. Sie soll sich auf ein Minimum an Software, z.B. Browser-Plugins, beschränken. Unmittelbar nach der Einwahl soll der Benutzer eine Übersicht von mehreren Planungsbeispielen präsentiert bekommen.

Nach der Auswahl des Planungsbeispiels soll in einem Browser-Fenster eine grafische Oberfläche als Benutzungsschnittstelle des Virtuellen Lichtlabors erscheinen, die mit der Maus und Tastatur bedient werden soll. Der Inhalt dieses Fensters soll sich an den gängigen Oberflächen von CAD-Programmen orientieren und in mehrere Bereiche unterteilt sein. Zum einen soll das Planungsbeispiel in einer dreidimensionalen perspektivischen Ansicht gezeigt werden. Die Bewegung durch den dreidimensionalen Raum soll uneingeschränkt möglich sein, damit von jeder Stelle aus die Szene betrachtet werden kann. Die Namen der einzelnen Objekte des Planungsbeispiels (Wände, Fenster, Leuchten u.a.) sollen in einer Objektübersicht aufgelistet werden, in der sie auch markiert, bzw. angewählt werden sollen. Auf die markierten Objekte sollen Befehle ausgeübt werden können, die im Befehlsmenü angewählt werden sollen. Der Übersichtlichkeit halber soll es nur die wichtigsten Funktionen enthalten. Nach der

Aktivierung erscheinen temporäre Befehls-Optionsfenster die nach dem Abschluss des Befehls wieder verschwinden.

Die Änderungs- bzw. Editiermöglichkeiten an den Planungsbeispielen sollen wirklichkeitsbezogen und aus didaktischen Gründen eingeschränkt sein. Sie sollen effizient sein und damit das Erstellen von mehreren Varianten betonen. Der Benutzer soll schnell mehrere Varianten ausprobieren bzw. erstellen, speichern und fotorealistisch überprüfen können. Dazu müssen intelligente Kopier-, Gruppierungs- und Ausrichtungsfunktionen für die einzelnen Objekte des Planungsbeispiels gegeben sein. Alle Objekte sollen nicht in allen denkbaren, sondern nur in ihren wichtigsten Eigenschaften verändert werden können. Kopieren und Löschen soll nur von bestimmten Objekten möglich sein, damit der Charakter des Planungsbeispiels nicht verloren geht. Die Eigenschaften und Änderbarkeit der Objekte soll wirklichkeitsnah sein, d.h. bei manchen Objekttyp (Wänden) macht eine Neigung keinen Sinn. Sie soll deshalb gar nicht erst angeboten werden.

Darüberhinaus sollen wirklichkeitsbezogene logische Zusammenhänge darstellbar sein, z.B. dass eine Leuchte an einer Stromschiene hängt und automatisch mit ihr verschoben und gedreht wird. Objekte gleichen Typs sollen beim Austausch ihre Eigenschaften vererben können, d.h. beim Austausch einer Leuchte sollen Position und Ausrichtung automatisch erhalten bleiben. Kopieren, Drehen und Neigen soll sowohl in Bezug auf ein absolutes Koordinatensystem als auch auf ein lokales Koordinatensystem eines Bezugselements möglich sein. In bestehenden Programmen sind die Änderungsmöglichkeiten oft zu kompliziert und orientieren sich nicht an möglicherweise hierarchisch verschachtelten Koordinatensystemen von Objektgruppen. Dort wirkt sich eine Neigungsänderung eines Spotlights auf die gesamte Leuchte aus, obwohl eigentlich nur der Reflektor innerhalb des feststehenden Bügels geneigt werden soll. Jedes Planungsbeispiel soll durch Objekte aus einer Objektbibliothek ergänzt werden können.

Die Darstellung der Planungsbeispiele soll in diesem Szeneneditor möglichst abstrakt sein. Sie soll damit den Modellcharakter betonen und sich nicht an einer möglichst fotorealistischen Qualität orientieren. Dabei soll eine Sprache entwickelt werden, in der zu den didaktischen zweidimensionalen Symbolen aus den herkömmlichen Darstellungen der Beleuchtungskonzepte äquivalente dreidimensionale Symbole gefunden werden. Das Licht, das von Leuchten oder sonstigen Lichtquellen austritt, soll hier als Lichtvolumen (z.B. als Kegel) dargestellt werden, um abschätzen zu können, welcher Bereich direktes Licht erhalten wird. Um die Ausrichtung von

Akzentlicht zu erleichtern, soll auch ein Lichtstrahl dargestellt werden können, der leicht auf ein Ziel einzujustieren ist. Zu den verschiedenen Darstellungsmodi von Leuchten soll auch die Darstellung nur der architektonischen Form der Leuchte gehören, d.h. ohne Lichtvolumen oder Lichtstrahl.

Der Benutzer muss seine Einstellungen bzw. entworfenen Planungsvarianten speichern, aufrufen und löschen können. Dazu muss er sie benennen und kann eine Projektbeschreibung, z.B. zum Grund der Untersuchung eintragen. Damit sie sich nicht mit denen anderer Benutzer in die Quere kommen, ist eine Benutzerverwaltung notwendig, die jedem Benutzer einen individuellen Bereich für seine Daten garantiert. Auf das virtuelle Lichtlabor soll zeit- und ortsunabhängig zugegriffen werden, d.h. ein Benutzer soll sowohl von zuhause, als auch von verschiedenen Arbeitsplätzen darauf zugreifen können. Seine Daten sollen also nicht auf den verschiedenen Rechnern (Redundanz), sondern zentral gespeichert und von dort überall hin verfügbar gemacht werden.

Um die Eingaben zusätzlich auf ihre visuelle Wirkung des Lichts zu überprüfen, soll der Benutzer auf einfache Weise fotorealistische Simulationen vom System anfordern können. Die abstrakten Objekte sollen hier zusätzlich automatisch durch ihre realen Gegenstücke ersetzt werden, d.h. die Gestalt einer Leuchte, die im Editiermodus durch einen einfachen Zylinder repräsentiert wird, sieht hier realistisch aus. Da die Simulationsergebnisse möglichst schnell dem Benutzer verfügbar gemacht werden sollen, sollten sie optimal vorbereitet sein und zudem auf leistungsfähigen Rechnern ablaufen. Da der Rechner des Benutzers dafür nicht in Anspruch genommen werden soll, soll ein System entwickelt werden, das die Simulationen automatisch auf entsprechende Ressourcen im Internet verteilt.

Die Simulationsergebnisse sollen als Bilder in einem separaten Szenenviewer dargestellt werden. Die Interaktion innerhalb des Szenenviewers konzentriert sich auf den Vergleich der fotorealistischen Bilder der Varianten eines Planungsbeispiels. Hier sollen zweidimensionale Darstellungen ausreichen. Zu jeder Variante sollen Bilder aus verschiedenen Standpunkten aufgerufen werden, in denen dann nochmals zwischen zwei Lichtmodi umgeschaltet werden kann. Der Indirektlicht-Modus zeigt ein Bild, bei dessen Simulation das indirekte, also auch vom Material abstrahlende Licht, berücksichtigt wird. Im Direktlicht-Modus wird nur die Verteilung des direkten Lichts simuliert. Durch das Umschalten zwischen diesen Modi soll abgegrenzt werden können, welchen Einfluss die Oberflächen neben dem direkten Licht auf das Erscheinungsbild des Raums haben. Das Umschalten zwischen den Varianten,

Standpunkten und Lichtmodi soll über eine intelligente Schalt-Matrix erfolgen. Es soll z.B. möglich sein, bei konstantem Standpunkt und Lichtmodus alle Varianten durchzublättern. Oder für jede Variante die Standpunkte oder im Direktlichtmodus alle Varianten etc.

Der Benutzer soll jederzeit zwischen der abstrakten und der fotorealistischen Darstellung umschalten können, d.h. zwischen Szenenditor und Szenenviewer. Diese duale Darstellung soll in einer innovativen Weise die Vorteile beider Darstellungen verbinden. Bei der fotorealistischen Darstellung wird die visuelle Wirkung des Lichts gezeigt. Es fehlt aber die Möglichkeit, nachvollziehen zu können, woher das Licht kommt. Die entsprechende Ursache des Lichts kann wiederum in der abstrakten Darstellung anhand des Lichtvolumens festgestellt werden.

Die Bedienung und Gestaltung des Szeneneditors und Szenenviewers soll allgemeine Ansprüche an die Softwareergonomie erfüllen, d.h. übersichtlich gestaltet und leicht zu bedienen sein. Die inhaltlich orientierte Anwendung soll vor der notwendigen Technik im Vordergrund stehen.

Generell gilt, dass das System möglichst ohne großen Programmieraufwand erweiterbar sein soll. Es soll daher ein Baukastensystem sein, in dem der Administrator neue Benutzer genauso einfach ergänzen kann, wie Planungsbeispiele, Objekte oder Bibliotheken. Solche Eigenschaften des Systems sollen sich vorzugsweise aus Textdateien generieren, die ohne Programmierkenntnisse verändert oder ergänzt werden können.

Die wichtigsten funktionellen Anforderungen hier noch mal zusammengefasst:

- Der Zugriff auf das Lernsystem soll über das WWW erfolgen.
- Das System soll mehreren registrierten Benutzern Zugriff auf die virtuellen Lichtlabore der architektonischen Planungsbeispiele ermöglichen
- Benutzer sollen ihre Daten (benutzerspezifisch) speichern und laden können.
- Keine schwer zu bedienenden kommerziellen CAD und Lichtsimulationsprogramme sollen auf den PCs der Benutzer installiert sein, d.h. dort keine Voraussetzungen an Software.
- Die Bedieneroberfläche des Lernsystems soll grafisch sein und die Eingaben mit Maus und Tastatur möglich sein.

- Im Szeneneditor soll eine dreidimensionale perspektivische Ansicht auf das Planungsbeispiel, ein Befehlsmenü und eine Objektliste (Szeneneditor) integriert sein.
- Die Darstellung des Lichts und der Architektur im Szeneneditor soll abstrakt sein und sich von einer aus der Lichtplanung gewohnten zweidimensionalen Darstellungsform in eine räumliche Dimension erweitern.
- Die Funktionen zum Ändern bzw. Entwerfen eines Beleuchtungsentwurfs sollen wirklichkeitsbezogen sein, d.h. logische Zusammenhänge zwischen den Elementen sollen genauso möglich sein wie das Kopieren, Gruppieren und Ausrichten der Elemente.
- Im Szenenviewer soll eine Schaltmatrix das schnelle Wechseln der Varianten, Standpunkte und Direktlichtmodi ermöglichen, die den direkten Vergleich unterstützt.
- Die fotorealistische Simulation soll auf Knopfdruck erfolgen und dabei nicht auf dem PC des Benutzers, sondern via Internet auf einem entfernten Hochleistungsrechner ablaufen.
- Die Planungsbeispiele und deren Elemente sollen als erweiterbares Baukastensystem entwickelt werden.

3.3 Technologischer Aufbau

Für die Realisierung dieses Lernsystems stehen theoretisch drei Wege offen:

- Verbesserung/Erweiterung existierender CAD- und Lichtsimulations-programme
- Neuentwicklung von CAD- und Lichtsimulationsprogrammen
- Entwicklung eines Systems, das die existierenden CAD- und Lichtsimulationsprogramme integriert und sie über eine innovative Schnittstelle leicht zugänglich macht.

Eine Verbesserung oder Erweiterung von existierenden CAD- und Lichtsimulationsprogrammen ändert nichts an Tatsache, dass sich der Anwender in einer langen Einarbeitungszeit mit den komplexen Programmen vertraut machen muss. Eine Neuentwicklung von CAD- und Lichtsimulationsprogrammen scheidet ebenfalls aus, weil dazu zu viel entwickelt werden muss, was prinzipiell schon vorhanden ist. Die Entwicklung eines Systems, das die existierenden CAD und Lichtsimulations-Programme integriert und sie über eine neu zu entwickelnde Schnittstelle leicht zugänglich macht, scheint am sinnvollsten und innovativsten.

Die Schnittstelle wird als Frontend entwickelt, das die existierenden CAD und Lichtsimulations-Programme als Backend integriert und nutzt. Ähnlich einem Client/Server Aufbau hat dieses System den Vorteil, das die Backends nicht auf dem Rechner des Benutzers installiert sein müssen. Dort läuft nur das Frontend, das über das Internet die Backends „fernsteuert“, die im Internet verteilt werden. Das Frontend nutzt dabei die Möglichkeit vieler Programme, auch über Kommandozeilen gesteuert werden zu können, und sorgt dafür, das entsprechende Kommandos bei den Backends eingehen. Das Frontend wird mit entsprechenden Webtechnologien realisiert. Bei der Entwicklung wird darauf geachtet, dass nur die wirklich notwendigen Funktionen und Befehle integriert werden, damit die Bedienung des Systems möglichst einfach und intuitiv erfolgen kann. Außerdem können neue Funktionen angeboten werden, die automatisch z.B. eine webbasierte Dokumentation der Ergebnisse liefern, die bisher in aufwendiger Kleinarbeit manuell mit Bildverarbeitungs- und Dokumentationsprogrammen erstellt werden musste.

Der Zugang zum Lernsystem erfolgt über einen Standard Webbrowser, d.h. die Benutzeroberfläche wird vollständig über HTML definiert. Alle Seiten werden in eine

übersichtliche Navigationsstruktur, z.B. in ein Menübereich eingebunden, aus dem die verschiedenen Bereiche leicht zu erreichen sind.

Für die Darstellung der Benutzerschnittstelle wird auf die vielfältigen multimedialen Formate zurückgegriffen, die ein Webrowser darstellen kann. Für die Definition von dreidimensionalen Daten wird das VRML Format benutzt, die der Webbrowser dreidimensional darstellen kann, wenn er mit einem entsprechenden Plugin, z.B. Cosmo Player erweitert wird. Das bedeutet, das die Planungsbeispiele im VRML Format erstellt werden müssen.

Die Darstellung der Objekte in einer listenförmigen Objektübersicht und die Möglichkeit, über Befehle aus einem Befehlsmenü auf sie einwirken zu können, z.B. sie verschieben zu können, ist allerdings keine Standardanwendung mehr. Die Standardanwendung beschränkt sich auf die bloße Durchwanderung einer geometrisch statischen Szene, die im Vorfeld aus 3D-CAD Programmen im VRML Format exportiert wird. Die Spezifikation von VRML erlaubt aber auch, dass Eigenschaften von Objekten als Variablen definiert werden, deren Werte von „außen“ über eine Schnittstelle dynamisch gesetzt werden können. Realisiert werden kann dies durch eine Kombination aus Cosmo-Player und einem Java-Applet, die über die EAI-Schnittstelle (External Authoring Interface) kommunizieren. Das Java-Programm wird als Applet in die HTML Seite eingebettet und kann auch als grafisches Befehlsmenü gestaltet werden, über das der Benutzer Befehle aktiviert. Es bietet weiterhin die Möglichkeit, dass beim Mausklick auf eine Befehlstaste ein temporäres Fenster - ein sogenanntes Flyout - aufgeht, in dem der Benutzer die jeweiligen Optionen des Befehls eingeben kann.

Die geforderten Änderungs- und Editiermöglichkeiten können nur realisiert werden, wenn die Objekte der VRML Szene entsprechend intelligent d.h. objektorientiert definiert werden. Dazu ist die Entwicklung eines objektorientieren Szenenmodells notwendig, in dem die Objekte und ihre wesentlichen Eigenschaften Form, Farbe, Darstellungsmodi, Dreh- und Rotationsachsen u.a. definiert werden. Die Konzeption und die Realisierung über das Java-Applet „Szeneneditor“ stellt eine Innovation dar. Das Java-Programm muss die Benutzereingaben entgegennehmen, sie auf das intern verwaltete objektorientierte Szenenmodell übertragen und für eine entsprechende Synchronisation mit der Darstellung im CosmoPlayer sorgen.

Der Benutzer erhält diese HTML Seite, die das Java-Applet und die VRML Szene im CosmoPlayer enthält, nur über das Internet. D.h. er muss eine entsprechende

Webadresse (URL) anwählen, die auf einen Webserver verweist, der diese HTML-Dokumente enthält und sie ihm dann über das Netzwerk sendet. Für die Errichtung dieses speziellen Webservers wird auf den Internet Information Server (IIS) von Microsoft zurückgegriffen, der im Hochschulbereich sehr günstig zu bekommen ist. Der entsprechende Rechner muss vorher als NT-Server konfiguriert sein. Die geforderte Benutzerveraltung kann über die Microsoft Standardauthentifizierung erreichet werden.

Alle Webserver haben typischerweise die Eigenschaft, dass sie vornehmlich zum „Ausliefern“ von HTML-Dokumenten an den Client gedacht sind. Die geforderte Speicherung der Benutzereingaben muss mit Eigenentwicklungen realisiert werden. Der Standard dazu sieht Benutzereingaben in sogenannte „Formulare“ vor, die auf den Webserver übertragen und dort der CGI-Schnittstelle (Common Gateway Interface) weitervermittelt werden. Diese Schnittstelle kann dann ihrerseits Programmroutinen (meistens in der Programmiersprache Perl geschrieben) anstoßen, die Benutzereingaben in Dateien auf dem Webserver speichern. Auf diesen Standard wird hier nicht zurückgegriffen, weil zum einen die zu speichernden Daten zu komplex sind und umfangreiche Formulare zur Folge hätten, die unter softwareergonomischen Gesichtspunkten nicht mehr zu vertreten wären. Außerdem sieht Java auch dazu Lösungen vor. Die CGI Schnittstellen können durch Java-Servletts ersetzt werden, die mit den Applets im Browser über die RMI-Schnittstelle (Remote Method Invocation) sehr gut kommunizieren. Die Anforderungen an das System die Benutzereingaben auf dem Server zentral zu verwalten bzw. zu speichern, zu löschen und zu lesen, können hiermit prinzipiell erfüllt werden.

Die Anforderungen der Benutzerverwaltung, dass jeder Benutzer seine Daten auf einem für ihn reservierten Bereich ablegen kann, erfordern eine zusätzliche Entwicklung. Der Szeneneditor muss also nicht nur wissen, welches Planungsbeispiel er darstellt, bzw. verwaltet, sondern auch von welchem Benutzer es verändert wird. Dies kann dem Applet über sogenannte Applet-Parameter vermittelt werden, die in dem HTML Dokument definiert sind, und beim Aufrufen der HTML-Seite bzw. beim Initialisieren des Applets gelesen werden. Beim Speichern und Löschen sorgen diese Parameter dafür, dass die Dateioperationen nur in den für die jeweiligen Benutzer reservierten Bereiche auf dem zentralen Server erfolgen.

Die Dateioperationen werden nicht direkt vom Servlet ausgelöst, sondern von Programmroutinen, die das Servlet anstößt. Je nach Benutzereingabe (Speichern, Löschen oder Rendern der Szene) werden dabei unterschiedliche Programmroutinen

gestartet. Beim Speichern ist wichtig, dass der Benutzer auf die gespeicherte Szene wieder zurückgreifen kann. D.h. es müssen neben dem eigentlichen Abspeichern auch entsprechende HTML Seiten generiert werden, über die der Benutzer wieder per Mausklick auf die gespeicherte Planungsvariante zurückgreifen kann. Beim Löschen müssen diese Seiten, bzw. die Link aus dem Inhaltsverzeichnis, auch wieder entfernt werden. Das Rendern soll schließlich das Speichern einschließen und eine Simulation des Planungsbeispiels anstoßen. Für diese Realisierung der umfangreichen Programmroutinen eignet sich die Programmiersprache Perl. Sie können durch das Servlet angestoßen und mit Parametern versorgt werden. Darüber hinaus können sie auch Systemkommandos auf Betriebssystemebene ausführen, die z.B. für das Löschen von Dateien notwendig sind. Außerdem können sie den Aufruf weiterer Programme einbetten und so die geforderte fotorealistische Simulation des Planungsbeispiels anstoßen.

Für die fotorealistische Simulation wird das Programm Lightscape gewählt. Es kann komplett im sogenannten Batchbetrieb gesteuert werden. Dazu müssen die entsprechenden Perl-Programme entwickelt werden, die Lightscape im Batchbetrieb ansprechen. Die Simulation erfordert eine vollständige Beschreibung der Szene, die im DXF-Format vorliegen muss. Die DXF-Datei muss vom Szeneneditor geliefert und über das Servlet direkt gespeichert werden. D.h. das Applet hat die zusätzliche Aufgabe, die Szene nicht nur im VRML Format für den CosmoPlayer zu generieren, sie objektorientiert zu verwalten und zu speichern, sondern sie beim Speichern auch noch in ein DXF konformes Format umzuwandeln. Dies stellt eine Innovation dar und ist insofern aufwendig, weil zum einen die Koordinatensysteme beider Systeme unterschiedlich sind (die Y-Achse beim DXF entspricht der negativen Z-Achse bei VRML) und weil das DXF-Format keine objektorientierte Beschreibung der Elemente zulässt. Die VRML Szene zerfällt im DXF in ihre einzelnen Bestandteile, d.h. logische Bezüge der Objekte untereinander (Gruppenzugehörigkeiten u.a.) und der Elemente gehen verloren. Diese DXF-Datei soll aber die Grundlage sein, aus der der Szeneneditor die VRML Szene wieder darstellen kann, also müssen diese Bezüge im DXF File untergebracht werden. Dazu muss das DXF-Format in einer innovativen Weise ergänzt werden, sodass zum einen der Szeneneditor daraus das objektorientierte VRML Modell und Lightscape das äquivalente Modell fotorealistisch generieren kann.

Bei der fotorealistischen Darstellung sollen ja die Objekte nicht mehr so abstrakt wie im VRML Modell dargestellt werden, sondern eine realistische Gestalt annehmen. Dies läst sich insofern realisieren, indem in der DXF-Datei nicht die Gestalt der Objekte, sondern nur der Name als Platzhalter definiert wird. Der Szeneneditor soll dann für die

VRML-Darstellung anhand des Objektnamens die Gestaltdefinition aus einer externen Datei holen, und diese auf die im DXF definierten Positionen und Ausrichtungen übertragen. Diese Datei soll beim Starten des Applet gelesen werden und dann die Objektbibliothek initialisieren, aus der Objekte zum Entwerfen und Verändern der Szene entnommen werden können. Die Konfigurationsdatei soll Textdatei sein, die vom Administrator des Systems einfach erstellt werden kann. D.h. der Szeneneditor benötigt zur VRML-Darstellung der Szene die Objektbibliothek-Datei und die DXF-Datei. Welche genau er laden soll, wird ihm auch hier wieder über entsprechende Applet-Parameter in der HTML Seite mitgeteilt. Diese Konfiguration von außen ermöglicht die individuelle Anpassung des Szeneneditors an die geforderten verschiedenen Planungsbeispiele und Benutzer ohne überflüssige Redundanz.

Wird die DXF-Datei zur fotorealistischen Simulation herangezogen, kann Lightscape eine ähnliche Ersetzung der Platzhalter mit Objekten aus vorbereiteten Lightscape Bibliotheken vornehmen. Diese Objekte haben den Vorteil, dass sie schon für eine Simulation optimiert sind und damit gute Simulationsergebnisse garantieren. Durch diese zwei Bibliotheken eines Objekts (Konfigurationsdatei für die abstrakte VRML Darstellung und Lightscape Bibliothek für die fotorealistische Darstellung) kann die geforderte duale Darstellung der Szene realisiert werden.

Der Szenenviewer ist gegenüber dem Szenenditor leichter zu realisieren. Dafür genügt eine HTML Seite, in der eine Javascript Funktion die Aufgaben der Schaltmatrix übernimmt. Die HTML Seite wird von einem Perl-Programm erzeugt und bei jeder neu abgespeicherten Variante des Planungsbeispiels entsprechend aktualisiert.

Die Präsentation der Lichtwirkung der verfügbaren Leuchten ist im Gegensatz zum virtuellen Lichtlabor, indem ständig neue Daten generiert werden, eine geschlossene Sammlung von Daten. Die Interaktion ist hier auf das intelligente „Blättern“ zwischen diesen vorbereiten Bildern und Texten beschränkt. Die notwendige Interaktion ist für eine HTML basierte Realisierung zu hoch. Dafür eignet sich das Programm Director besser. Außerdem kann hier die Darstellung nicht durch individuelle Browsereinstellungen verändert, bzw. verfälscht werden. Die Director Animation kann im Shockwave Format gespeichert werden und dann von jedem Browser, der mit dem kostenlosen Shockwave Plugin erweitert wurde, problemlos dargestellt werden.

Zusammenfassung

Jedes der drei Frontends (Szeneneditor, Szenenviewer und Shockwave) ist eine HTML Datei, die mit spezifischen Dateien (A-F) vom Webserver via http an den

Webbrowser übertragen wird und die entsprechenden Funktionalitäten bereitstellt (Abb. 58).

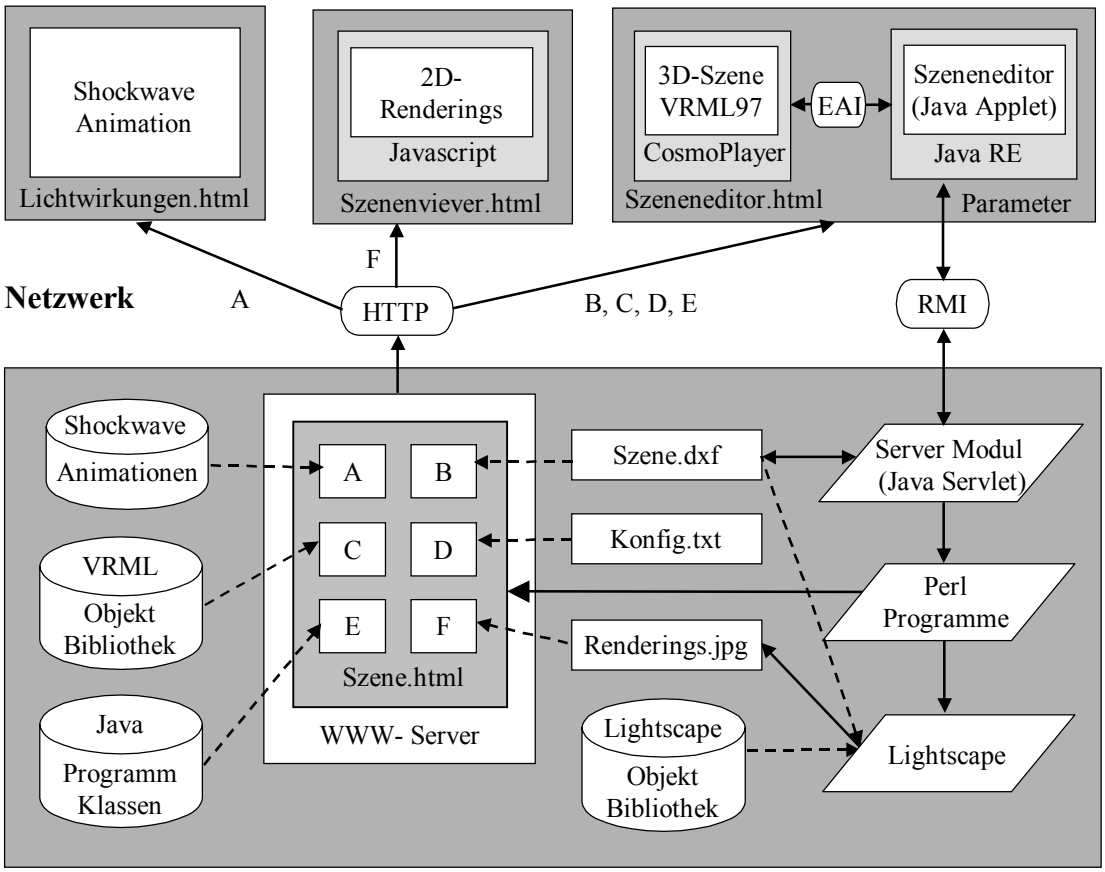
Die HTML-Datei der Enzyklopädie bindet die Shockwave Animationen (A) und die des Szenenviewers die Lightscape Renderings (F) ein, die der Benutzer über den Szeneneditor erstellt hat. Mit dem Aufruf des Szeneneditors wird das Java-Applet „Szeneneditor“ geladen, das aus mehreren Java-Klassen besteht (E). Parameter in der HTML Seite sorgen beim Initialisieren des Applets dafür, das für das jeweilige Planungsbeispiel die entsprechende DXF Datei (B) aufgerufen wird.

Anhand der ebenfalls beim Initialisieren referenzierten Konfigurationsdatei (D) kann das Applet die VRML Gestalt anhand der VRML-Objektbibliothek (C) aufbauen. Die erfolgreiche Initialisierung endet mit der objektorientierten Verwaltung der Szene und der Darstellung im Cosmo Player. Jetzt kann der Szeneneditor Benutzereingaben entgegennehmen.

Beim Speichen der Szene berechnet der Szeneneditor vom aktuellen Stand der VRML-Szene eine dxf-konforme Abbildung. Er sendet sie via RMI an das Server Modul (Java Servlet), das es als DXF-Datei (B) direkt auf dem Webserver speichern kann. Der Name der Datei ergibt sich aus den Benutzereingaben, den entsprechenden Pfad hat der Szeneneditor durch die Parameter aus der HTML Seite erhalten. Zusätzlich startet das Server Modul ein Perl Programm, das die HTML Seite generiert, über die der Benutzer wieder auf sein Planungsbeispiel zugreifen kann. Beim Löschen wird diese HTML Seite entsprechend gelöscht.

Beim Rendern wird zusätzlich die Simulation mit Lightscape ausgelöst. D.h. ein entsprechendes Perl Programm ruft Lightscape mit zusätzlichen Optionen auf. Sie verweisen auf eine DXF Datei und die Lightscape-Bibliothek. Aus beiden generiert Lightscape fotorealistische Renderings (F), deren Standpunkte und sonstige Optionen vom Benutzer im Szeneneditor definiert wurden. Zusätzlich erstellt oder aktualisiert das Perl-Programm das HTML-Dokument `Szene.html`, in das die Renderings der Simulation automatisch eingebettet werden. Sie bilden den Szenenviewer, mit dem der Benutzer seinen Entwurf fotorealistisch kontrollieren kann.

Frontend im Webbrowser



Backend mit Webserver, Server Modul und Lightscape

Abb. 58 Technologischer Aufbau des Frontend/Backend Systems

Zusammenfassend eine Kurzbeschreibung der wichtigsten Komponenten:

- Szeneneditor
- Szenendatei
- Konfigurationsdatei
- Webserver
- RMI
- Perl

- *Szeneneditor*
  Hier handelt es sich um ein neu entwickeltes Programm auf der Basis der objektorientierten Programmiersprache Java. Es besteht aus einem Kernprogramm und weiteren Klassen, die alle im Kapitel 4.2.3.1 „Programme im Verzeichnis Class“ näher beschrieben werden. Der Szeneneditor ermöglicht die interaktive Veränderung der Szene. Er synchronisiert über die vorhandene EAI-Schnittstelle (External Authoring Interface) seine interne objektorientierte Repräsentation der Objekte einer Szene mit deren 3D-Präsentation (nach VRML 97) im CosmoPlayer. Der Szeneneditor erhält beim Aufruf Parameter, die ihm mitteilen, welche Szene er darzustellen (Szenendatei) hat, wo er ggfs. eine neue speichern soll und wie die darzustellenden Objekte aussehen sollen (Konfigurationsdatei). Diese Parameter ermöglichen die Verwendung des Szeneneditors für alle Benutzer, alle Planungsbeispiele sowie deren Elemente.

- *Szenendatei (Szene.dxf)*
  Hier handelt es sich um ein neues Datenformat. Es soll ein Planungsbeispiel und die vom Benutzer erzeugten Varianten eindeutig beschreiben. In ihr werden die Namen der vorkommenden Elemente sowie deren Position und Ausrichtung definiert. Es basiert auf dem Format DXF, wurde aber um Einträge ergänzt, damit sowohl der Szeneneditor als auch das Lichtsimulationsprogramm anhand dieser Datei die Szene darstellen können. Es handelt sich aber um eine Metabeschreibung, da die Gestalt der Elemente hier nicht erläutert wird. Sie wird für die abstrakte Darstellung aus einer weiteren Datei, der Konfigurationsdatei, und für die photorealistische Darstellung aus einer Lightscape Bibliothek entnommen. Diese Metadatei ermöglicht die unterschiedlichen Darstellungen, die für die duale Darstellung notwenig sind. Beim Speichern einer Szene schreibt der Szeneneditor die neue Konstellation benutzerspezifisch in eine DXF-Datei, die – falls die Option Rendern angegeben wurde – direkt vom gleichzeitig gestarteten Lichtsimulationsprogramm Lightscape verarbeitet wird. Dabei werden die vorhandenen Objekte in der Szenendatei durch alternative Objektdarstellungen aus einer Lightscape-Bibliothek ersetzt. Zu jeder Szene existiert genau eine Szenedatei, ihr Format wird im Kapitel 4.2.8.3 „DXF-Szenendatei“ näher erläutert.

- *Konfigurationsdatei (Konfig.txt)*
  In dieser Textdatei werden vor allem die VRML-konformen Eigenschaften der Elemente beschrieben. Hier sind auch mögliche Elementierungen definiert, d.h.

logische Bezüge zwischen den einzelnen Bestandteilen. Des weiteren sind hier die veränderbaren Eigenschaften und ihre Ausgangswerte festgelegt, und unter welchem Namen sie ggfs. gruppiert in der Objektliste erscheinen. Für die eigentliche Gestalt der Elemente wird auf zusätzliche Dateien verwiesen, die im VRML Format für jedes Element vorliegen (siehe die Kapitel VRML Dateien, 4.2.4.2, 4.2.5.2 und 4.2.7.2). Alle Dateien werden vom Szeneneditor ebenfalls beim Start gelesen, der daraufhin diese Objekte a) im CosmoPlayer darstellt (sofern sie in der Szenendatei aufgeführt sind) oder b) diese Objekte zum Einfügen in die Szene bereitstellt. Die Konfigurationsdatei ist nach einer neu entwickelten Struktur aufgebaut, die leicht ergänzt werden kann. Sie wird benötigt, damit man jedem Planungsbeispiel einen individuellen Baukasten an Elementen zur Verfügung stellen kann. Die Elemente unterscheiden sich in ihren Eigenschaften. Für sie wurde ein objektorientiertes Szenenmodell entwickelt, das im gleichnamigen Kapitel näher erläutert wird.

- *Webserver*
  Damit überhaupt die Programme und Daten zum Rechner des Benutzer kommen, wird auf das Hyper Text Transfer Protokoll (http) von Standard-Webservern und auf die Möglichkeit in den HTML Seiten weitere Dateien und Programme zu referenzieren zurückgegriffen. Wenn der Benutzer über seinen Webbrowser die Webseite Szeneneditor.html anfordert, werden die eingebetteten Bestandteile der Seite übertragen (Abb. 58). Dies ist zum einen der Szeneneditor („E“ aus dem „Topf“ der Javaprogramm-Klassen), der die Szenen- und Konfigurationsdatei („B“ und „D“) referenziert, die sich ihrerseits der VRML Objektbibliothek („C“) bedient. Im Fall des Szenenviewers sind nur die Bilder der Lichtsimulationen („F“) eingebettet. Das Programm, das die Schaltmatrix ermöglicht, ist als Javascript Bestandteil der Webseite „Szenenviewer.html“. Im Falle der Lichtwirkungen ist nur die Shockwave Animation („A“) eingebettet.

- *RMI*
  Die klassische Funktion eines Webservers ist das Ausliefern von Informationen bzw. Senden von Webseiten, wenn sie von einem Webbrowser über das http-Protokoll angefordert werden. Sind die Daten angekommen, besteht keine Verbindung mehr. Der umgekehrte Weg, d.h. das Ablegen von Dateien auf dem Webserver, ist nicht vorgesehen. Diese Funktion, die beim Speichern der Szenendatei benötigt wird, muss ergänzt werden. Dazu wird wieder auf die Möglichkeiten der Programmiersprache Java zurückgegriffen, die mittels RMI

(Remote Method Invocation) das Speichern ermöglicht. Dazu wurde ein neues Programm - das Server Modul – entwickelt, das quasi als Gegenstelle zum Szeneneditor, der auf dem PC des Benutzers aktiviert ist, nun auf dem Webserver die Speicherung (der Szenendatei) umsetzten und darüber hinaus noch weitere Perl-Programme anstoßen kann.

- *Perl-Programme*
  Damit ein Benutzer wieder auf seine abgespeicherte Szene zurückgreifen kann, muss sie in Form einer HTML Seite aufrufbar sein. Für dessen Erzeugung ist ein Perl-Programm verantwortlich, das die Szenedatei und alle anderen Komponenten einbettet. D.h. beim Speichern wird nicht nur die Szenendatei abgelegt, sondern automatisch eine entsprechende HTML-Seite generiert. Ihr Aufbau entspricht der eingangs beschriebenen Datei Szeneneditor.html. Wird beim Speichern die Option „Rendern“ aktiviert, wird zusätzlich eine automatische Lichtsimulation mit Lightscape ausgeführt. Dabei wird die Szenendatei eingelesen und die Objekte durch Darstellungen aus den Lightscape Bibliotheken ersetzt. Für jede Szene wird von jedem angegeben Standpunkt ein Bild berechnet, das sowohl das direkte als auch das indirekte Licht veranschaulicht. Zusätzlich wird die Datei Szenenviewer erzeugt bzw. aktualisiert, mit der die verschiedenen Varianten vergleichend betrachtet werden können. Alle neu entwickelten Perl-Programme werden im gleichnamigen Kapitel näher beschrieben.