6 CONTIGRA: Der Autorenprozeß und seine Werkzeuge

Im vierten Kapitel wurde die dokumentzentrierte CONTIGRA-Architektur vorgestellt, die sich aus den wichtigen Ebenen der Komponenten- und Anwendungsentwicklung zusammensetzt. Für die deklarative Erstellung von 3D-Komponenten und 3D-Applikationen auf Basis von Dokumenten wurden geeignete Grammatiken entwickelt, die in Form der XML-Schemata *CoApplication, CoComponent* und *CoComponentImplementation* im Kapitel 5 beschrieben sind. Instanzdokumente dieser Formate lassen sich prinzipiell mit einfachen Texteditoren erstellen, was jedoch ein sehr mühseliger Vorgang bereits für mäßig komplexe Anwendungen ist. Daher ist es nötig, den gesamten auf der CONTIGRA-Architektur basierenden Autorenprozeß mit Hilfe von geeigneten und komfortablen Werkzeugen zu unterstützen.

In diesem Kapitel wird deshalb zunächst eine Analyse existierender 3D-Autorenwerkzeuge durchgeführt. Diese macht nicht nur deutlich, daß für den deklarativen CONTIGRA-Ansatz eine spezielle Werkzeugentwicklung nötig ist, sondern gibt auch zahlreiche Anregungen für Editorbestandteile und Funktionalitäten. Im Anschluß an die Analyse wird der speziell für die CONTIGRA-Architektur konzipierte Autorenprozeß mit seinen Phasen und beteiligten Personen vorgestellt. Daraus ergeben sich konkrete Anforderungen an ein 3D-Autorenwerkzeug und seine Bestandteile. Es folgt die Darstellung des entwickelten Autorenwerkzeuges CONTIGRABUILDER im Überblick und aus Nutzersicht, wobei die bisher realisierten Editoren im Mittelpunkt stehen. Das folgende Unterkapitel ist einigen Detailaspekten der prototypischen Realisierung gewidmet. Neben Informationen zum CONTIGRABUILDER werden auch Transformationsmöglichkeiten von CONTIGRA-Dateien in spezifische 3D-Grafikformate und in Webpräsentationen von 3D-Komponenten vorgestellt. Damit liegen auch für die Ebene der Komponentendistribution erste Implementierungen vor. Schließlich wird das Kapitel mit der Darstellung von Beispielanwendungen abgerundet.

Bisher sind nur wenige Informationen über einen strukturierten Entwicklungsprozeß dreidimensionaler virtueller Umgebungen in systematisierter und anwendungsbereiter Form verfügbar. In den Dissertationen von Kaur [Kaur98] und Paelke [Paelk02] werden zwar typische Designphasen bei der Entwicklung von VE aufgeführt, sie eignen sich jedoch weniger für die Entwicklung von 3D-GUIs oder interaktiven 3D-Grafikanwendungen ohne räumliches Vorbild. Für den Entwurf derartiger Software lassen sich eher Erfahrungen aus der Entwicklung von Multimediaanwendungen heranziehen, etablierte Vorgehensmodelle gibt es jedoch noch nicht. Das liegt vor allem an den zugrundeliegenden 3D-Formaten bzw. Architekturen. Es ist ein Unterschied, ob 3D-Anwendungen ausschließlich programmiert werden, vorgefertigte Komponenten bzw. Schablonen zum Einsatz kommen oder aber nahezu ausschließlich deklarative XML-Dokumente erstellt werden (s.a. [Dachs03b] und [X3DCPWorkshop@]). 3D-Autorenwerkzeuge orientieren sich somit nicht nur am jeweiligen Vorgehensmodell oder den beteiligten Autorengruppen, sondern stehen auch in enger Beziehung zu den zugrundeliegenden Formaten und Architekturkonzepten.

So soll es in diesem Kapitel nicht um eine tiefergehende Analyse oder Generalisierung von Vorgehensmodellen, Entwicklungsprozessen oder Methodologien im Bereich 3D-Authoring gehen. Auch Bereiche wie die Spezifikation von 3D-Anwendungen, ihre Distribution oder das Content Management werden bewußt nicht näher betrachtet. Statt dessen liegt der Fokus auf der Vorstellung eines auf die CONTIGRA-Architektur zugeschnittenen Autorenprozesses mit seinen Werkzeugen.

6.1 Existierende 3D-Autorenwerkzeuge

Eine Vielzahl von 3D-Autorenwerkzeugen ist inzwischen für die verschiedensten 3D-Grafikformate erhältlich. Bei einem Großteil der Werkzeuge werden jedoch Kernaufgaben – wie die komfortable Erstellung komplexer 3D-Anwendungen aus wiederverwendbaren Einheiten oder die effektive Modellierung von Verhalten – nur ungenügend unterstützt. Dennoch wurden existierende Ansätze untersucht, weil sie Hinweise und Anregungen bezüglich der Konzeption des CONTIGRA-Autorenwerkzeuges bieten. In [Vitzt02] erfolgte eine umfassende Analyse relevanter Autorenwerkzeuge und Forschungsarbeiten. Neben bekannten *2D - User Interface Buildern* wurden insbesondere 3D-Autorenwerkzeuge detailliert analysiert und miteinander verglichen. Einen Vergleich von Autorenwerkzeugen speziell hinsichtlich ihrer Möglichkeiten zur Verhaltensmodellierung legt die Arbeit von Rukzio [Rukzi02] dar. Vorausgegangen war diesen Arbeiten die Analyse von spezialisierten Autorenwerkzeugen für Web3D-Grafik in [Rukzi01]. Für einzelne Bewertungskriterien und die ausführlichen Darstellungen sämtlicher Werkzeuge wird auf die genannten Quellen verwiesen.

Web3D-Autorenwerkzeuge lassen sich prinzipiell in die folgenden Gruppen einteilen, wobei das zugrundeliegende Format als Basis eine wichtige Rolle spielt [Dachs03b].

- Einfache *Texteditoren* ohne Syntaxkontrolle für offengelegte deklarative oder imperative ASCII-Formate, wie z.B. VRML97 oder Java3D.
- *Integrierte Programmierumgebungen* für rein imperative 3D-Grafikformate (z.B. Eclipse [Eclipse@] oder JBuilder [JBuilder@] für Java3D).
- *Syntaxgesteuerte Programmeditoren* für offengelegte ASCII-Formate, teilweise auch mit visuellen Tools (z.B. VrmlPad [VRMLPad@] für VRML97).
- Allgemeine oder auf eine Grammatik spezialisierte *XML-Editoren* für offengelegte XML-Formate, wie X3D oder CONTIGRA (z.B. XML-Spy [XMLSPY@] oder X3D-Edit [X3DEdit@]).

- Allgemeine *3D-Modellierungswerkzeuge*, die Exportfunktionalität für VRML97 oder proprietäre Formate anbieten (z.B. Alias Maya [Maya@] mit Export nach VRML97 und Shockwave 3D).
- *Multimedia-Autorenwerkzeuge* mit 3D-Grafikerweiterungen (z.B. Macromedia Director [Director@]).
- *Visuelle Werkzeuge für VRML97/X3D,* wobei von der einst großen Zahl nur wenige Erfolg hatten, darunter Cosmo Worlds [CosmoWorlds@], Internet Scene Assembler [ISA@] oder Vizx3d [Vizx3d@].
- *Visuelle Werkzeuge für proprietäre Web3D-Formate,* wobei zu dieser Gruppe die erfolgreichsten Autorenwerkzeuge für 3D-Technologien gehören, darunter Viewpoint [VET@], Cult3D [Cult3D@], Adobe Atmosphere [Atmosphere@], AXEL [AXEL@] oder Virtools [Virtools@].
- Teils visuelle *Autorenwerkzeuge aus Forschungsprojekten* (z.B. Alice [Alice@] oder CONTIGRABUILDER).

Aus diesen Gruppen wurden in Form der folgenden Unterkapitel wichtige und einen primär visuellen Autorenprozeß unterstützende ausgewählt, die jeweils mit typischen Vertretern dargestellt sind. In der Abbildung 25 sind dazu Bildschirmschnappschüsse von im Text erwähnten Autorenwerkzeugen zu sehen.

6.1.1 Allgemeine und spezialisierte XML-Editoren

Da sowohl X3D als auch CONTIGRA XML-basierte Dateiformate anbieten, liegt die Nutzung von XML-Editoren zur Erstellung dreidimensionaler Szenen nahe. Allgemein verwendbare und ausgereifte Autorenwerkzeuge wie z.B. XML-Spy [XMLSPY@] bieten dabei eine Vielzahl von Editiermöglichkeiten für XML-Schemata und Instanzdokumente. Dazu zählen hierarchische Ansichten, syntaxgesteuertes Texteditieren und automatische Vervollständigungsfunktionen. Der wesentliche Nachteil ist jedoch die fehlende Spezialisierung für ein bestimmtes XML-Format.

Das im Rahmen der X3D-Spezifikation vom Web3D-Consortium entwickelte X3D-Autorenwerkzeug *X3D-Edit* [X3DEdit@] basiert auf dem IBM-Tool *Xeena* und ist auf die X3D-DTD spezialisiert. Statt textueller Erstellung und nachträglicher Validierung gegenüber einer DTD oder einem Schema erfolgt hier das Editieren mit Hilfe einer grafischen Baumansicht (s. Abbildung 25, 1). Je nach aktuellem Kontext können nur die XML-Elemente eingefügt werden, die an einer bestimmten Stelle möglich und gültig sind. Die Auswahl der Elemente erfolgt aus Knotenpaletten, Attribute können typsicher für das gewählte Element in einem Editor festgelegt werden. Vorschaumöglichkeiten in VRML97- und X3D-Viewern sind mittels eingebauter XSLT-Transformationen und externem Browseraufruf möglich. Auch für die Erstellung von CONTIGRA-Dokumenten wäre ein angepaßtes Xeena-Werkzeug denkbar. Hauptnachteil ist jedoch neben den fehlenden visuellen Werkzeugen die enge Orientierung am zugrundeliegenden Format und daraus resultierend die fehlende Abstraktion bzw. Vereinfachung für den Benutzer.

Das Werkzeug *XML Wizard* [Still01] wurde zum Editieren von XML-basierten Kursdokumenten für Lehr-/Lernanwendungen entwickelt. Module bzw. Plug-Ins erlauben jedoch auch die Anpassung an andere XML-Grammatiken, so daß neben dem baumorientierten Editieren einer XML-Struktur auch spezialisierte Editoren für bestimmte XML-Elemente eingebunden werden können. Die Arbeit ist vor allem interessant durch die Untersuchungsergebnisse hinsichtlich der Plug-In-Architekturen. Stiller kommt zu dem Ergebnis, daß Plug-In-Architekturen stark an das Aufgabengebiet und Rahmenwerkzeug gebunden sind, so daß sich Eigenentwicklungen oft einfacher gestalten als Anpassungen vorhandener Architekturen.

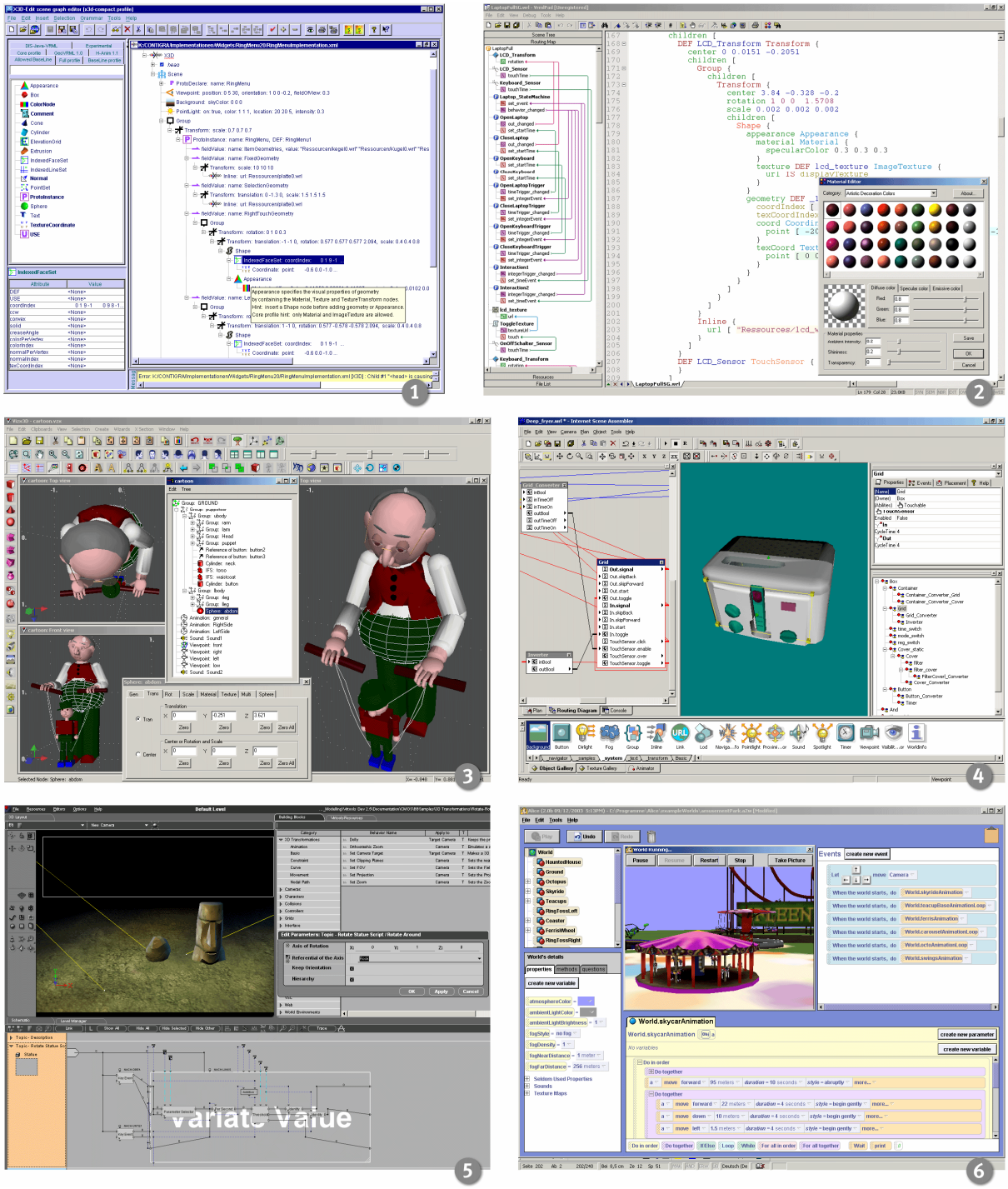

Abbildung 25: 3D-Autorenwerkzeuge: X3D-Edit (1), VrmlPad (2), Vizx3D (3), ISA (4), Virtools Dev (5), Alice (6)

6.1.2 Visuelle Werkzeuge für VRML97 / X3D

Für das Web3D-Standardformat VRML97 gibt es verschiedene Erstellungsmöglichkeiten und Autorenwerkzeuge, wie bereits im Unterkapitel 2.3.1 erwähnt. Es ist festzustellen, daß trotz der mehrjährigen Spezifikationsphase von X3D als Nachfolgeformat erst wenige Werkzeuge zum Editieren von X3D existieren. Dies liegt unter anderem an den Inkonsistenzen und der bisher noch nicht erfolgten Standardisierung des Formates. Das in der vorangegangenen Werkzeuggruppe bereits vorgestellte *X3D-Edit* baut jedoch jeweils auf dem aktuellen Stand von X3D auf. Auch für die Erstellung von VRML97-Dateien wurde eine lohnenswerte Alternative zu simplen Texteditoren entwickelt, das Programm *VrmlPad* [VRMLPad@] der Firma

Parallel Graphics (s. Abbildung 25, 2). Neben syntaxgesteuerter automatischer Code-Vervollständigung, Fehlererkennung und einem Ressourcen-Manager bietet es auch Vorschau-Möglichkeiten und visuelle Werkzeuge, z.B. eine Baumansicht der Szene oder eine sogenannte *Routing Map* zur Visualisierung von Knotenverknüpfungen. Deshalb wird es auch hier mit aufgeführt, obwohl es eigentlich zur Gruppe *Syntaxgesteuerte Programmierwerkzeuge* gehört. Preview-Möglichkeiten und Hilfsmittel zum Optimieren und Publizieren einer VRML-Szene runden dieses primär für Programmierer entwickelte Werkzeug ab.

Das 3D-Autorenwerkzeug *Vizx3D* [Vizx3d@] der Firma Virtock Technologies ist das Nachfolgeprogramm zum VRML97-basierten *Spazz3D*. Wenn auch ein eigenes Dateiformat verwendet wird, so steht der Editor doch in der Tradition ausgereifter VRML-Editoren mit 3D-Modellierungsfunktionalität, wie z.B. dem ursprünglich für SGI-Plattformen entwickelten Programm *Cosmo Worlds*. Wie der neue Name *Vizx3D* bereits suggeriert, lassen sich neben VRML97-Szenen nun auch X3D-Dateien exportieren. Damit dürfte dieses Werkzeug als der erste verfügbare visuelle X3D-Editor gelten. Eine Vielzahl grafischer Werkzeuge und Wizards sowie verschiedene Szenenansichten mit Positionierungshilfen erlauben die schnelle Erstellung dreidimensionaler Szenen (s. Abbildung 25, 3). Neben dem Dateiimport sind auch diverse einfache Modellierungsfunktionen integriert. NURBS werden genauso wie polygonale Modelle unterstützt, die sich mit einem *Indexed Face Set Editor* im Detail manipulieren lassen. Die Unterstützung von multiplen Texturen, *Universal Media* [UniversalMedia@] und *Humanoid Animation* – Avataren [H-anim@] runden das Werkzeug ab. Für Verhaltensmodellierung erweist sich das Werkzeug als ungenügend [Rukzi02], auch Keyframe-Animationen sind nur mühselig zu erstellen. Zwar werden Codedetails für den Nutzer völlig verborgen, dafür ist der Ansatz jedoch immer noch zu sehr Szenengraph-orientiert.

Die Firma Parallel Graphics hat sich auf Werkzeuge und Dienste rund um VRML97 spezialisiert und bietet neben dem oben erwähnten VrmlPad mit dem *Internet Scene Assembler (ISA)* [ISA@] ein High-Level-Modellierungswerkzeug zur Erstellung von interaktiven VRML97-Szenen an. Von Szenengraphen wird teilweise durch einen komponentenartigen Ansatz abstrahiert. Dabei kommen VRML97-Prototypen zum Einsatz, die im ISA als Objekte bzw. Objektbausteine bezeichnet werden und in Form visueller Galerien verfügbar sind. Die vorgefertigten Objektbausteine lassen sich mit Manipulatoren in der 3D-Ansicht ausrichten, einfach animieren und über Ereigniseditoren und ein *Routing Diagram* verknüpfen bzw. mit Interaktivität versehen. Das Werkzeug (s. Abbildung 25, 4) ist flexibel konfigurierbar und bietet viele wichtige Funktionen zum komfortablen Editieren einer VRML97-Szene.

6.1.3 Visuelle Werkzeuge für proprietäre 3D-Formate

Bei dieser Werkzeuggruppe handelt es sich in der Regel um kommerziell vertriebene, visuelle und anspruchsvolle Autorenwerkzeuge. Zugrunde liegt ein eigenes, nicht veröffentlichtes Dateiformat, das mit Hilfe des Autorensystems erstellt wird und dann in einem Player im Web dargestellt wird. Da die Formate nicht standardisiert sind und häufig in binärer Form vorliegen, spielen leistungsfähige Werkzeuge eine entscheidende Rolle zur Erstellung interaktiver 3D-Web-Anwendungen. Zu bemerken ist für einige der Werkzeuge ihre Fokussierung auf ein bestimmtes Anwendungsgebiet. So dient z.B. der *Cult3D Designer* [Cult3D@] der Firma Cycore primär der Erstellung von E-Commerce-Anwendungen und Produktpräsentationen, während andere Werkzeuge für die Entwicklung von Avataren und Charakteranimationen konzipiert wurden.

Das 3D-Autorenwerkzeug *AXEL* [AXEL@] der Firma MindAvenue stellt eine Neuentwicklung einer Web3D-Technologie auf Basis des AXEL-Dateiformates dar. Das ausgereifte Werkzeug ähnelt in seiner Erscheinung zunächst Vizx3D oder ISA, bietet jedoch eine klarere Benutzungsschnittstelle und interessante Editoren. Es hat seine Stärke unter anderem in der

Integration verschiedenster Medien, darunter Bilder, 3D-Modelle, Videos, Sounds und Flash-Animationen. An Exportfunktionen sind neben ausführbaren Programmen auch QuickTime, VRML97, Flash, MPEG-4 und andere Vektor- und Bildformate möglich. Interaktives Verhalten und komplexere Animationen sowie Objekt-Constraints lassen sich mit dem visuellen *Interaction Editor* und *Sequencer* sehr gut realisieren. Dieses Werkzeug läßt folglich nur wenige Wünsche offen und bietet wichtige Anregungen für eigene Lösungen.

Das Autorenwerkzeug *Virtools Dev* [Virtools@] der französischen Firma Virtools ist ein besonders leistungsfähiges Tool im Bezug auf visuelle und komplexe Verhaltensmodellierung für interaktive 3D-Grafik. Mehr als 400 vordefinierte, wiederverwendbare Verhaltensbausteine stehen zur Verfügung und können per grafischem Skripting miteinander verbunden werden. Ein hierarchisches Keyframe-Animationssystem, Kollisionsmanagement, physikalische Simulationen, die integrierte Virtools-Skriptsprache und eigene, mit dem C++-SDK entwikkelte Verhaltensbausteine erlauben die Realisierung nahezu beliebiger 3D-Funktionalität. Objekte werden in gängigen 3D-Modeling-Formaten importiert und können in der Szene angeordnet und beleuchtet werden. Die visuelle Bearbeitung von Verhalten und Interaktivität erfolgt dann im Werkzeug *Schematic* mit Hilfe einer ausgereiften grafischen Darstellung von Verhaltensbausteinen und ihren Verknüpfungen. Auch Virtools Dev (s. Abbildung 25, 5) basiert auf einem eigenen Web3D-Format und bietet einen speziellen Player dafür an.

6.1.4 Autorenwerkzeuge aus Forschungsprojekten

Innerhalb dieser Gruppe lassen sich Werkzeuge finden, die im Rahmen von Forschungsprojekten zumeist als Prototypen entstanden sind. Deshalb bieten sie nicht den Leistungsumfang der kommerziell vertriebenen Tools, weisen jedoch teilweise interessante Aspekte auf oder demonstrieren einen bestimmten Lösungsansatz. Auch der bereits unter 6.1.1 erwähnte *XML Wizard* und der in diesem Kapitel später vorgestellte CONTIGRABUILDER lassen sich dieser Gruppe zuordnen.

Das 3D-Autorenwerkzeug *Alice* der Carnegie Mellon University ([Pierc97b], [Alice@]) ist primär auf das Sammeln erster Programmiererfahrungen mit interaktiver 3D-Grafik für Schüler und Studenten ausgerichtet. So bietet es eine übersichtliche und intuitive Benutzungsoberfläche mit einer eher spielerischen Herangehensweise (s. Abbildung 25, 6). In der inzwischen vorliegenden und vollständig neu in Java implementierten Version 2 lassen sich vorgefertigte Verhaltensbausteine und Welt-Objekte einfach parametrisieren. Dabei fällt der zwar visuelle, aber deutlich an Programmierkonstrukten orientierte Ansatz auf. Anspruchsvolle interaktive Szenen sind mit den mitgelieferten Animations- und Interaktionsbausteinen trotz der möglichen Verwendung von Python-Skripten nur schwer zu erstellen. 3D-Szenen werden im eigenen Alice-Dateiformat abgelegt.

Innerhalb ihrer Forschungsarbeiten zur Verknüpfung von Java3D und JavaBeans zu *3D Beans* [Dörne00] entwickelten Dörner und Grimm auch das visuelle Werkzeug *3D Beanbox*. Damit ist die Konfiguration und Assemblierung von 3D-Komponenten möglich, die Verhalten und Geometrie kapseln. Da sie den JavaBeans-Konventionen genügen, existieren auch 2D-Ansichten der Komponenten für herkömmliche UI-Builder. Umgekehrt lassen sich in der 3D Beanbox auch 2D-Beans laden. Die Funktionalität des Werkzeuges beschränkt sich auf Dateiarbeit, einfache geometrische Manipulationen innerhalb der 3D-Ansicht sowie Konfiguration und Verknüpfung von Komponenten in nicht-visuellen Editoren. Hieraus leitet sich die Beobachtung ab, Autorenwerkzeuge möglichst nicht als optische 1:1 – Repräsentationen des zugrundeliegenden programmiertechnischen Modells zu realisieren.

6.1.5 Fazit

Da die komponentenbasierte CONTIGRA-Architektur auf Basis von XML-Auszeichnungssprachen einen neuartigen Ansatz für die Erstellung interaktiver 3D-Grafik darstellt, lassen sich existierende Autorenwerkzeuge entweder gar nicht oder nur mit deutlichen Anpassungen dafür einsetzen. So lag die Entscheidung für die Eigenentwicklung eines auf die Architektur zugeschnittenen Werkzeuges nahe. Bei der Analyse der zahlreichen visuellen 3D-Autorenwerkzeuge konnte jedoch eine Reihe von Gemeinsamkeiten herausgearbeitet werden, die Grundlage für die Konzeption des CONTIGRA-Autorenwerkzeuges waren. Die an [Vitzt02] (S. 46) angelehnte Tabelle 21 listet noch einmal wichtige 3D-Autorenwerkzeuge mit ihren jeweiligen Bestandteilen bzw. Editoren zum Vergleich auf. Kursiv gekennzeichnete Worte sind dabei die Originalnamen der Werkzeugbestandteile im entsprechenden 3D-Autorenwerkzeug.

Werkzeug / Bestandteil	**Vizx3D**	**Internet Scene Assembler**	**Cult3D Designer**	**AXEL**	**Virtools Dev**	**Alice 2**
Interaktive Baumansicht der Szenenbestandteile	*Scene Tree Window*	*Scene Tree Window*	*Scene Graph Editor*	*World Explorer*	*Level Manager*	*World*
Property-Editoren	*Node Property Pages*	*Object Properties*	*Object Properties*	*Parameter Editor*	*Attribute & Variable Manager*	*Properties*
Auswahl-Paletten	Objekte über Werkzeug-leiste, sonst diverse Listen	*Object Galleries* für Knoten und Objekte	*Actions*, *Events*, *Java Actions*, *Sounds*, …	-	*Building Blocks, Virtools Resources*	diverse Listen und textuelle Kollektionen
Zuordnung Ereignis-Objekt-Aktion	über *Animation Property Page*	*Routing Diagram,* Event-Editoren	*Event Map*	*Interaction-Editor*	*Schematic*	*Events*
2D-Editoren bzw. 2D-Ansichten	mehrere 2D-Ansichten einstellbar	*Plan Window*	-	mehrere *Viewports* einstellbar	-	-
3D-Editoren bzw. 3D-Preview	*3D-Views* + X3D&VRML-Preview + Browser-Simulation	Editor im *Perspective View Window*	*3D-View* mit Manipulato-ren, Preview	Editor im *WebCam View*	3D Layout und *Play*-Modus	3D-Editor und Abspiel-möglichkeit
Animationseditoren	über *Property Pages* und *Animation Wizard*	*Key Frame Animator*	*Events* & *Actions* - Fenster, *Time Line View & Event Map*	*Sequencer*, Animations- und Con-straints-Tools	*Schematic*	Animations-editor
Tools für Statistik, Test, Optimierung, Performance	Browser Debug mit Event Graph	*Debugging Window*	Internetopti-mierung beim Publizieren	*Browse*-Modus und *Web Integra-tion* Tool	-	World Statistics

Tabelle 21: Gegenüberstellung visueller 3D-Autorenwerkzeuge mit ihren Bestandteilen

Dabei deutet die Auswahl der wichtigen Hauptbestandteile in dieser Tabelle bereits auf jene Editoren hin, die ein eigenes Autorenwerkzeug ebenso besitzen sollte. Auch in der Abbildung 25 lassen sich diese Editoren gut erkennen, z.B. interaktive Baumansichten in (1), (3) und (4), Property-Editoren in (2) und (3), eine Auswahl-Palette in (4) unten, 3D-Ansichten in (3-6), Interaktions- und Verhaltenseditoren in (2) und (4) links sowie (5) und (6) unten.

6.2 CONTIGRA-Prozeßphasen und beteiligte Autoren

6.2.1 Phasen des Autorenprozesses

Die typischen Phasen der komponentenorientierten Softwareentwicklung müssen für den deklarativen CONTIGRA-3D-Komponentenansatz angepaßt und näher untersetzt werden. Das liegt einerseits an den Besonderheiten der Erstellung von 3D-Anwendungen auf Basis von deklarativ beschriebenen Dokumenten und andererseits an dem Umstand, daß Programmierer hier nur eine Fachrolle parallel zu Mediendesignern, Soundexperten, Konzeptionisten und anderen Autoren einnehmen. Diese Rollen werden unter 6.2.2 näher beschrieben und in Zusammenhang mit den hier beschriebenen Phasen gebracht. Auch in anderen Arbeiten wurden ähnliche Phasen identifiziert, so z.B. bei [Geige98] die Phasen *Strukturelles Design, Geometrie-Design, Interaction Behaviour Design* und *Conversion & Packaging*.

Die Abbildung 26 zeigt den CONTIGRA-Autorenprozeß mit seinen fünf Phasen im Überblick. Dabei werden gemäß der unter 4.4.2 vorgenommenen Einteilung die beiden Ebenen bzw. Hauptaufgaben Komponentenentwicklung (links) und Anwendungsentwicklung (rechts) getrennt dargestellt. Es gibt jedoch viele Gemeinsamkeiten, was darin begründet ist, daß eine entwickelte Applikation auch als komplexe Komponente gekapselt werden kann. Umgekehrt wird eine komplexe Komponente durch Hinzufügung von Beleuchtung, Kamera etc. zu einer 3D-Anwendung. In der Abbildung bedeuten Teilphasen mit gestricheltem Rand, daß hierfür externe Werkzeuge zum Einsatz kommen und diese nicht im CONTIGRA-Autorenwerkzeug bearbeitet werden. Dies ist sinnvoll, da es für viele Aufgaben – wie z.B. Spezifikation, Mediendesign oder Content Management – bereits ausgereifte Werkzeuge gibt. Gestrichelte Pfeile bedeuten alternative Abläufe, in der Regel unter Auslassung bestimmter Teilaufgaben. Die meisten Phasen erlauben auch Rücksprünge in vorangegangene Phasen, da es sich um einen iterativen Autorenprozeß handelt. Diese wurden jedoch aus Gründen der Übersichtlichkeit im Diagramm weggelassen.

Die *Konzeptionsphase* besteht aus der Anforderungsanalyse und der Spezifikation, wobei hier die Definition von Inhalt, Struktur und Funktionalität der Komponente bzw. Anwendung erfolgt. Für diese Aufgaben sollten keine neuen Werkzeuge entwickelt werden. Lediglich die Spezifikation einer Komponente könnte anteilig auch bereits in Form eines CONTIGRA-Schnittstellendokuments *(CoComponent)* vorgenommen werden, das aber ebenso auch erst in der Test- und Auslieferungsphase erstellt werden kann. Die Phase *3D-Modellierung und Medien-Design* dient der Gestaltung audiovisueller Medienobjekte, wofür auf ausgereifte Tools zur Bearbeitung der entsprechenden Medien zurückgegriffen wird und Dateien dann später nur noch importiert bzw. verknüpft werden.

Die Phase der *Integration und Anwendungsentwicklung* stellt die Hauptphase bei der Entwicklung von CONTIGRA-Komponenten und -Applikationen dar. Erkennbar sind dabei die beiden Aufgabengruppen auf Szenengraphniveau und auf der Abstraktionsebene von Komponenten. Die Überlappungen der einzelnen Teilaufgaben in beiden Entwicklungssträngen bedeuten, daß Komponenten primär neu aus Szenengraphbestandteilen aufgebaut werden, während Applikationen vor allem aus Subkomponenten zusammengesetzt sind. Natürlich können Komponenten auch Unterkomponenten enthalten und 3D-Anwendungen auch zusätzliche Szenengraphbestandteile. Die Phase der *Funktionsprogrammierung* beinhaltet als Hauptaufgabe die Verknüpfung der einzelnen Subkomponenten und Szenengraphbestandteile. Manchmal – und insbesondere bei der Entwicklung neuer Komponenten – wird die Funktionalität, welche sich durch deklarative Verknüpfung existierender Komponenten, Szenengraph- und Verhaltensknoten erzielen läßt, nicht den Anforderungen genügen. Daher können komponenten- oder anwendungsspezifische Funktionen in Form von eigenen Verhaltensknoten mit entsprechendem Code hinzugefügt werden (s. 5.4.4).

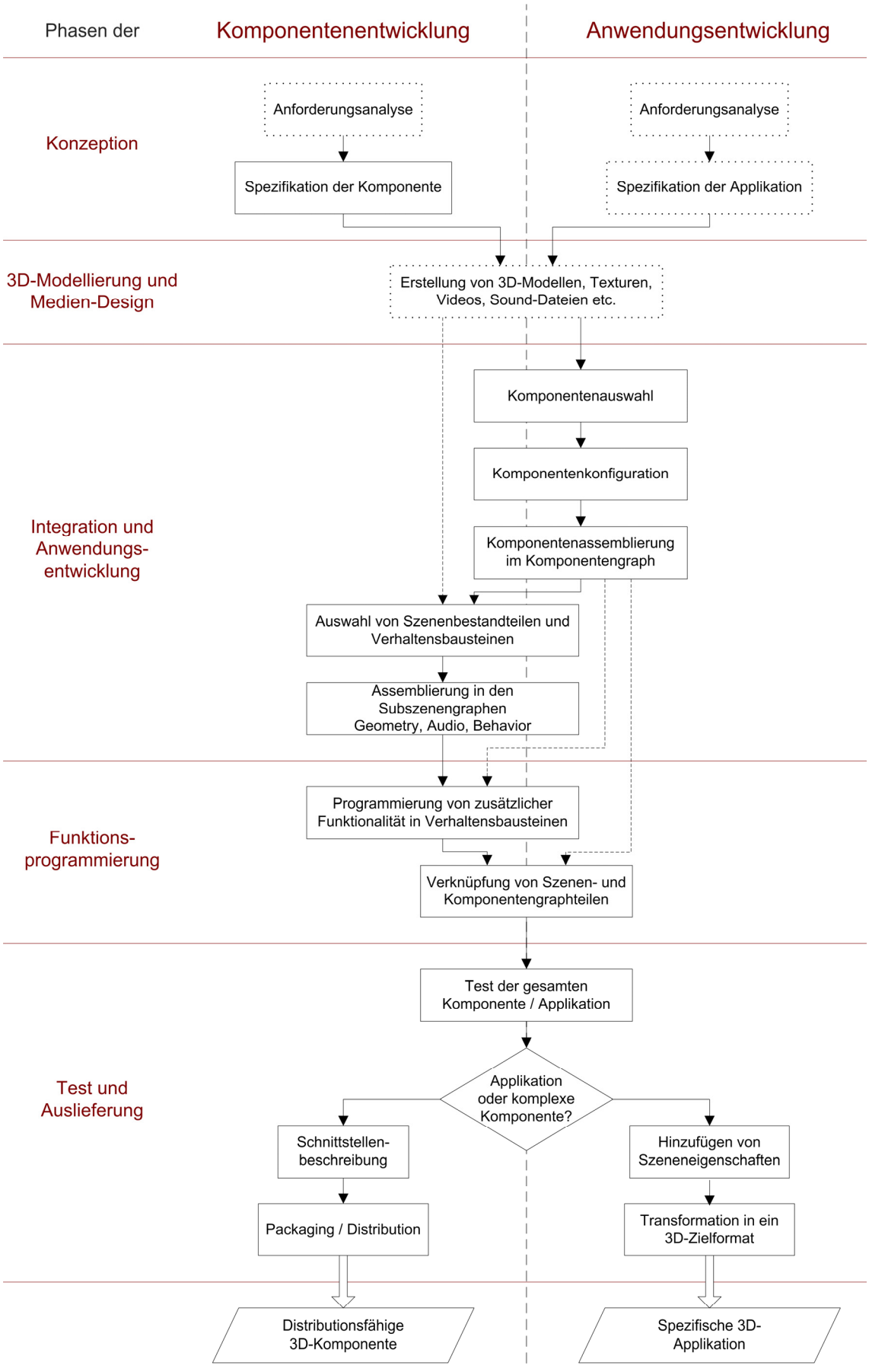

Abbildung 26: Die Phasen des CONTIGRA-Autorenprozesses

Schließlich erfolgt in der Phase *Test und Auslieferung* die Überprüfung, inwieweit Realisierungen von Komponenten und Applikationen den Anforderungen gerecht wurden. An dieser Stelle kann noch entschieden werden, ob aus der erstellten interaktiven Szene eine distributionsfähige Komponente oder eine in konkrete 3D-Zielformate übersetzte Anwendung erzeugt wird. In dieser Phase ist – wie in vorangegangenen auch – ein Rücksprung zu anderen Teilaufgaben möglich, da es sich bei CONTIGRA um einen iterativen Rapid Prototyping Autorenprozeß handelt, der schrittweise Konfigurationen, Verbesserungen und funktionale Erweiterungen erlaubt. Dabei erfolgt die sukzessive Verfeinerung der 3D-Anwendungen durch alle beteiligten Autoren, die im folgenden näher beschrieben werden.

6.2.2 Gruppen von Autoren

Der dargestellte Autorenprozeß impliziert zunächst eine Unterteilung in Autoren, die Komponenten entwickeln und solche, die Komponenten zur Anwendungsentwicklung verwenden, also konfigurieren und assemblieren. So erfordert z.B. die Konzeption einer Bibliothek von universell einsetzbaren und gut parametrisierten 3D-Komponenten ein anderes Expertenwissen als die domänenspezifische Erstellung einer interaktiven 3D-Anwendung. Da jedoch beide Entwicklungsstränge wichtige Aufgaben und Arbeitsschritte teilen, sollen die Autoren der verschiedenen Fachdisziplinen hier nur einmal und unabhängig vom konkreten Entwicklungsziel genannt werden. Weiterhin bedeutet die Einteilung in unterschiedliche Autorengruppen nicht, daß es in jedem konkreten Fall eine 1:1 – Abbildung auf projektbeteiligte Personen gibt. Es ist beim CONTIGRA-Autorenprozeß – ebenso wie bei Multimediaproduktionen generell – nicht unwahrscheinlich, daß bestimmte Aufgaben in Personalunion oder bei größeren Projekten durch mehrere Fachleute parallel bearbeitet werden. Deutlich wird in den folgenden Gruppenbeschreibungen, daß alle beteiligten Entwickler zudem verschiedene Werkzeuge benötigen, aber auch unterschiedliche Zugänge, Editoren und Metaphern zu einem integrierenden 3D-Autorenwerkzeug, das interdisziplinäres Arbeiten unterstützt. Außerdem sollten Verantwortlichkeiten im Entwicklungsteam klar festgelegt werden, wozu auch vorhandene Projektsteuerungs-Software verwendet werden kann.

Konzeptionisten / Regisseure

- konzipieren dreidimensionale Gesamtanwendungen und legen deren Bestandteile und Funktionalität in Zusammenarbeit mit Designern, Programmierern, Audio- und Fachgebietsexperten fest, agieren also primär in der *Konzeptionsphase.*
- wählen Bestandteile / Komponenten der geplanten virtuellen Umgebung aus und erstellen möglicherweise auch erste visuelle Prototypen der Applikation; begleiten die gesamte Entwicklung – und somit alle anderen Phasen – aus einer High-Level-Sicht.
- benötigen zunächst traditionelle Werkzeuge, wie Papier und Stift, Textverarbeitungs- und Grafikprogramme, Storyboard-Tools, Projektplanungssoftware, aber auch ein Rapid-Prototyping-fähiges 3D-Autorenwerkzeug.

Komponenten- und Anwendungsprogrammierer

- sind mit der Entwicklung distributionsfähiger 3D-Komponenten bzw. der Erstellung des funktionalen Teils eines Gesamtsystems beauftragt.
- können für die Assemblierung aus Szenengraph- und Komponententeilen sowie die Anpassung der Parameter verantwortlich sein, also unter Mitwirkung anderer Experten die Phase der *Integration und Anwendungsentwicklung* abdecken.
- müssen große Detailkenntnisse der Architektur und der Formate besitzen, vor allem im Bezug auf Szenengraphen, Verknüpfungskonstrukte, Verhaltensknoten und Programmiersprachen zur eigentlichen Implementierung *(Funktionsprogrammierungsphase).*

- verwenden gängige Softwareentwicklungswerkzeuge und XML-Editoren, aber auch ein 3D-Autorenwerkzeug zur Integration und Verknüpfung.

Mediendesigner und 3D-Modellierer

- erstellen einerseits in der Phase *3D-Modellierung und Mediendesign* 3D-Modelle, Test-Animationen, Texturen und anderes Bildmaterial, Videos etc.
- sind andererseits für den visuellen Entwurf der Applikation verantwortlich und passen Komponenten optisch an, sorgen für deren Integration in den 3D-Raum und werden so in der Phase *Integration und Anwendungsentwicklung* gestalterisch aktiv.
- benötigen Werkzeuge für 3D-Modeling & Animation, zur Bild- und Videobearbeitung; für die Anwendungsentwicklung ein 3D-Autorenwerkzeug mit Editoren zur visuellen Manipulation der 3D-Szene.

Audio-Experten

- erstellen analog zu Mediendesignern in der Phase *3D-Modellierung und Mediendesign* Feedback-Sounds, Hintergrundgeräusche, Spezial- und Raumklangeffekte.
- sind für den akustischen Entwurf einer Anwendung in der Phase *Integration und Anwendungsentwicklung* verantwortlich, legen Raumklangparameter fest und ordnen z.B. Geometrien Halleigenschaften zu.
- benötigen einerseits verschiedene Audiobearbeitungswerkzeuge, andererseits auch ein visuelles 3D-Autorenwerkzeug mit speziellen Raumklangeditoren.

Fachgebietsexperten

- werden mit dem Konzeptionisten / Regisseur primär in der *Konzeptionsphase* aktiv und begleiten dann mit domänenspezifischem Rat aus dem Anwendungsgebiet die anderen Autoren in allen weiteren Phasen.
- werden auch in der Phase *Test und Auslieferung* überprüfen, inwieweit die Applikation / Komponente den Anforderungen gerecht wurde, können also neben Ratgebern zugleich auch Auftraggeber sein.
- benötigen wie Konzeptionisten / Regisseure Standardwerkzeuge und Rapid-Prototyping-fähige 3D-Autorenwerkzeuge.

Usability-Experten / Tester

- werden primär in der letzten Phase *Test und Auslieferung* aktiv, indem sie die erstellten Komponenten und 3D-Anwendungen testen und mit den Anforderungen vergleichen.
- sollten auch bereits in der *Konzeptionsphase* und den Hauptentwicklungsphasen agieren, um Fehlentwicklungen zu vermeiden und Machbarkeitsuntersuchungen durchzuführen.
- benötigen Vorschau-Funktionalität für in Entwicklung befindliche Anwendungen, also Zugang zum 3D-Autorenwerkzeug, aber auch typische Usability-Tools und evtl. Neuentwicklungen speziell für den Bereich 3D-Benutzungsoberflächen.

6.3 Anforderungen an ein 3D-Autorenwerkzeug

Es ist in den vorangegangen Abschnitten deutlich geworden, daß sich eine Zusammenarbeit der verschiedenen Autoren auf ein 3D-Autorenwerkzeug gründen kann, das von allen Entwicklern neben ihren Spezialwerkzeugen gemeinsam verwendet wird. Dies kann in der vollen Ausbaustufe ein Groupware-Werkzeug sein, das jedoch nicht Gegenstand dieser Arbeit ist. Mit einem Autorenwerkzeug kann auch als Einzelnutzerlösung eine verständliche Abstraktionsebene für alle Beteiligten geschaffen werden. So sollte es den beteiligten Disziplinen unterschiedliche Zugänge bzw. Sichtbarkeiten (z.B. der Komponentenschnittstellen) für ein interdisziplinäres Arbeiten anbieten. Diese und andere grundlegende Anforderungen an den Autorenprozeß und seine Werkzeuge wurden bereits im Kapitel 4.1 aufgestellt. Nach der Darstellung der Phasen des Autorenprozesses und der Autorengruppen werden nun spezielle Anforderungen an ein 3D-Autorenwerkzeug auf Basis der CONTIGRA-Architektur vorgestellt. Diese bildeten auch die Basis für die Entwicklung des Werkzeuges CONTIGRABUILDER. Für eine detailliertere Auflistung von Anforderungen wird auf das unter [KP2002@] abrufbare Pflichtenheft „Editormodule für den CONTIGRABUILDER“ verwiesen.

Grundlegende Werkzeugbestandteile und Funktionen

- Leistungsfähige Importmodule für Geometrien, Sounds, Texturen, Videos in verschiedenen Formaten und mit Vorschau-Funktionalität.
- Exportfunktionen in verschiedene Zielformate mit (halbautomatischen) Optimierungen.
- Visuelle Ansicht der Szenengraphhierarchien und umfassende Editiermöglichkeiten der Baumansicht(en) (Selektieren, Löschen, Einfügen, Verschieben, Benennen etc.).
- Importmöglichkeiten von 3D-Komponenten und Anordnung ausgewählter Komponenten in visuellen Paletten.
- Komponentengraphdarstellung zur Szenenkomposition (z.B. als Hierarchiedarstellung) mit zahlreichen Editiermöglichkeiten.
- Intuitive Komponentenkonfiguration mit geeigneten Parametereditoren.
- Einfache Benutzbarkeit durch visuelle Editoren mit schneller Prototyping-Funktionalität und gleichzeitig Möglichkeiten zur genauen Kontrolle von Details (z.B. Einstellung exakter Parameterwerte).
- Einstellen von (möglichst nicht-restriktiven) Autorenrollen bzw. Nutzungsmodi und entsprechende Anpassung der zur Verfügung stehenden Editoren und Sichtweisen für interdisziplinäres Arbeiten.
- Ikonische Repräsentation von Optionen und Objekten, Kontextmenüs bzw. kontextorientiertes Angebot von Funktionen / Optionen, *drag and drop* – Unterstützung.
- Plug-In-Architektur für Erweiterungen mit speziellen oder alternativen Editoren.
- Portierbarkeit des Werkzeuges bzw. Plattformunabhängigkeit.

Interaktiver 3D-Editor

- 3D-Vorschauansicht mit möglichst permanenter Aktualisierung; als 3D-Editor mit integrierten 3D-Widgets für Transformationen und zur Manipulation anderer Parameter.
- Gitter- und Achsendarstellung sowie Snap-Tools für den 3D-Editor, um Objekte bzw. Komponenten gut plazieren und in Relation zueinander positionieren zu können.
- Abspiel-/Testmodus für Animationen und die Interaktionen der Endanwendung, da die 3D-Ansicht sich sonst im Editormodus befindet und z.B. Transformationswidgets bei Objektauswahl eingeblendet würden.

- Unterstützung der Verhaltens- und Raumklangmodellierung durch geeignete Tools und Visualisierungen, evtl. in verschiedenen Arbeitsmodi.
- Alternative 2D-Ansichten aus bestimmten Perspektiven mit ähnlicher Funktionalität wie beim 3D-Editor.

Verhaltens-, Animations- und Verknüpfungseditoren

- Textuelle und grafische Verknüpfungseditoren (Interaktionseditoren) mit diversen Editiermöglichkeiten und geeigneter Visualisierung der unterschiedlichen Beziehungen.
- Visuelle Erstellung von Verknüpfungen durch Auswahl von Bestandteilen der Szene, z.B. auch im 3D-Editor.
- Auswahlmöglichkeiten für alle existierenden Verhaltensbausteine und Verknüpfungstypen, möglichst geeignete Gruppierung in Paletten.
- Zeitorientierter Editor vor allem für Animationen.

3D-Audio-Editoren

- Visueller Editor für den Audiograph (Soundquellen, Räume, LOD etc.) mit zahlreichen Editiermöglichkeiten, auch Einfügen von vereinfachten Raumgeometrien.
- Editor für die Einstellung globaler Audioparameter und der Standard-Hall- und Mediumseigenschaften.
- Intuitives Editieren der Verknüpfung zwischen komplexen Szenen- und einfachen Raumklanggeometrien, evtl. halbautomatische Anpassungsmöglichkeiten.
- Audio-Editier-Modus im 3D-Editor, dabei geeignete Visualisierung nicht sichtbarer Objekte und vereinfachter Raumklanggeometrien, wie z.B. Soundquellen, Raumbereiche und Hindernisobjekte; Einsatz interaktiver 3D Widgets zur Manipulation.
- Preset-Editor zum Import und Export von vorgefertigten Einstellungen für Audiogruppen bzw. Audio-Objekte.

6.4 Der CONTIGRABUILDER im Überblick

6.4.1 Charakteristik

Der CONTIGRABUILDER ist ein 3D-Autorenwerkzeug für die Erstellung dreidimensionaler Grafikkomponenten und interaktiver 3D-Anwendungen. Zugrunde liegt die CONTIGRA-Architektur mit ihrem deklarativen, XML-basierten Dokumentenmodell (s. Kap. 5). Somit kann man die entwickelte Software auch als spezialisierten Editor für das Erstellen von CONTIGRA-XML-Dokumenten betrachten, die auf den Grammatiken *CoApplication, CoComponent* und *CoComponentImplementation* basieren. Da es sich aber um ein visuelles Autorenwerkzeug handelt, stellt es keine 1:1 – Abbildung der intern verwendeten Formate dar, sondern bietet verschiedenartige grafische Editoren, wie sie auch in anderen, im letzten Unterkapitel beschriebenen 3D-Autorenwerkzeugen zur Verfügung stehen.

Es handelt sich um einen Prototypen ohne die Reife kommerzieller Produkte, der jedoch einen Ansatz für eine intuitivere und interdisziplinäre High-Level-Erstellung von 3D-Grafikanwendungen darstellt. Nach der Klassifikation von *User Interface Software* in [Myers00] zählt der CONTIGRABUILDER zur Kategorie der *Interface Builders* bzw. *Interactive Graphical Tools* zur Benutzungsschnittstellenentwicklung. Zugleich enthält er aber auch Anteile eines *Component Systems* und *Prototyping Tools*. Abbildung 27 zeigt das Werkzeug im Überblick, welches im folgenden mit seinen Bestandteilen aus Nutzersicht vorgestellt werden soll.

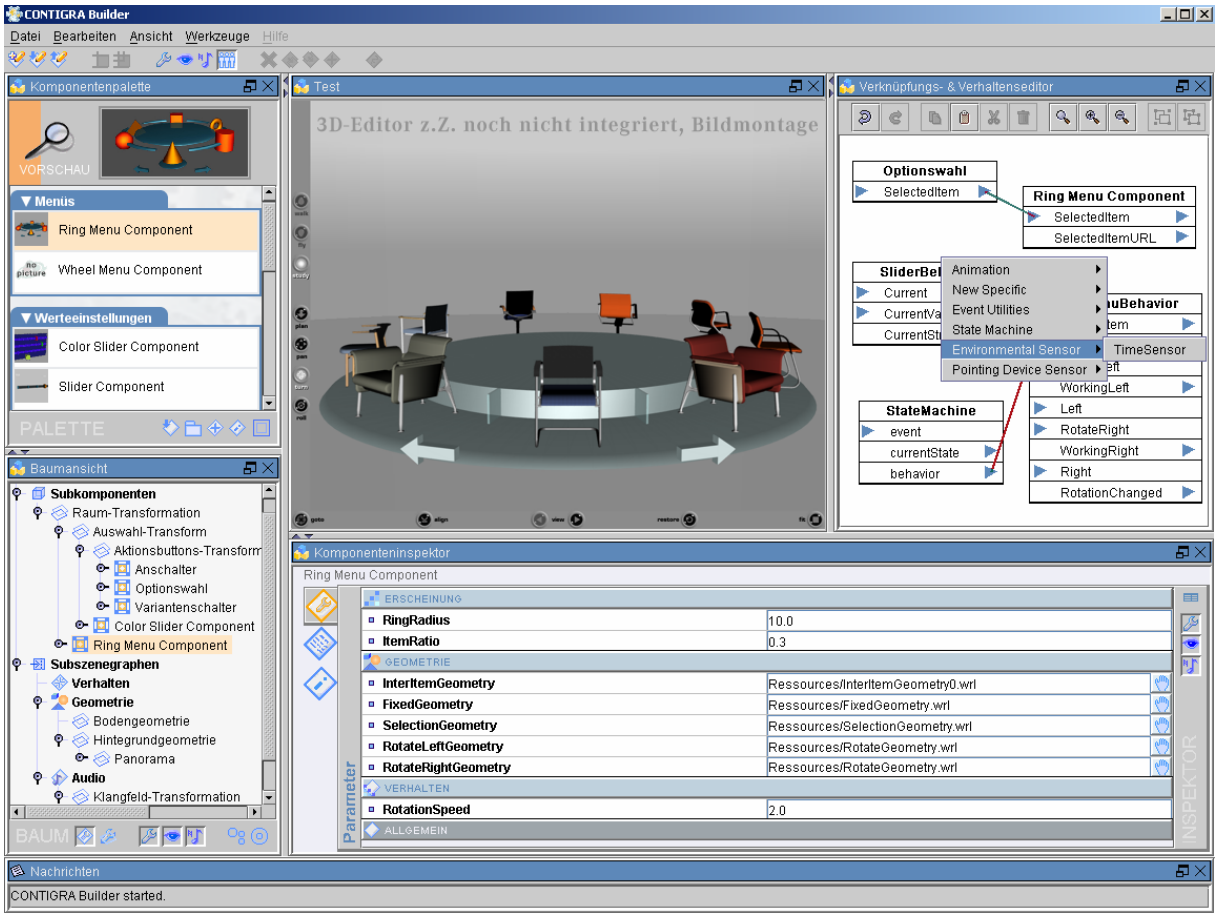

Abbildung 27: Das 3D-Autorenwerkzeug CONTIGRABUILDER im Überblick

Die Konzeption und der technische Werkzeugrahmen des CONTIGRABUILDER entstanden als Ergebnis der Arbeit von Vitzthum [Vitzt02] und Editorkomponenten im Rahmen eines Komplexpraktikums [KP2002@]. Dazu zählen die in der Abbildung 27 links oben sichtbare *Kom-*

ponentenpalette zur Auswahl und Anordnung von CONTIGRA-3D-Komponenten, die darunter befindliche *Baumansicht* mit Komponentengraph und den Subszenengraphen Verhalten, Geometrie und Audio sowie der rechts unten erkennbare *Komponenteninspektor* zum Editieren von Parametern. Der zentral angeordnete *3D-Editor* befindet sich noch in Entwicklung. Rechts davon ist der Prototyp eines *Verknüpfungs- und Verhaltenseditors* [Rukzi02] zu sehen.

6.4.2 Arbeitsmodi und prinzipielle Werkzeugnutzung

Je nach Entwicklungsaufgabe werden im CONTIGRABUILDER die beiden Benutzungsmodi *ComponentBuilder* und *ApplicationBuilder* unterschieden. Mehrere Teilaufgaben können für beide Entwicklungsstränge (s. Unterkapitel 6.2.1) durchgeführt werden, womit auch gleiche oder ähnliche Editoren zur Verfügung stehen sollten.

Der Modus *ComponentBuilder* dient der Erstellung von 3D-Komponenten, wobei primär Komponentenprogrammierer mit Szenengraphkenntnissen, für die visuelle Ausprägung einer Komponente aber auch Mediendesigner und 3D-Modellierer beteiligt sind. Aus vorhandenen Szenengraphbestandteilen werden benötigte über Dateidialoge bzw. Textansichten selektiert und in den entsprechenden Transformationshierarchien der *Baumansicht* angeordnet. Ebenso können benötigte Subkomponenten aus der *Komponentenpalette* gewählt, in der *Baumansicht* angeordnet und im *Komponenteninspektor* konfiguriert werden. Schließlich erfolgt die Verknüpfung von Szenengraphknoten und Komponenten bzw. ihrer Felder und Parameter im *Verknüpfungs- und Verhaltenseditor* oder in textuellen Editoren, wobei auch zusätzliches, applikationsspezifisches Verhalten in Form neuer Behavior3D-Knoten hinzugefügt werden kann. Der Inspektor erlaubt im *ComponentBuilder*-Modus das Editieren aller Eigenschaften der Komponentenschnittstelle. Das betrifft sowohl die Festlegung, welche Parameter von welchem Typ eine Komponente nach außen sichtbar macht, als auch die Zuweisung von Default-Werten für Parameter und das Setzen von Entwickler- und Metainformationen. Alternativ dazu kann die Komponentenschnittstelle auch mit Hilfe eines *Wizards* erstellt werden.

Da im Modus *ApplicationBuilder* durch die Arbeit mit vorgefertigten 3D-Komponenten vom Szenengraphniveau abstrahiert wird, können mehrere Autoren verschiedener Fachgebiete auf einfache Art und Weise 3D-Anwendungen erstellen. Die visuelle *Komponentenpalette* stellt eine Bibliothek frei verfügbarer oder auch firmenspezifischer Komponenten dar, aus der Autoren die geeigneten Komponenten wählen und per *drag and drop* in *3D-Editor* oder *Baumansicht* ziehen und so der gesamten Szene hinzufügen können. Dabei werden nötige geometrische Transformationen entweder visuell mit 3D-Tranformationswidgets oder textuell innerhalb der Baumansicht editiert. Die Konfiguration der Komponenten im Sinne ihrer optischen Erscheinung oder auch ihres Verhaltens erfolgt dann im *Komponenteninspektor,* der das Einstellen von Parametern erlaubt. Zusätzliche Szenengraphbestandteile können, wie oben beschrieben, in der Baumansicht hinzugefügt werden. Schließlich erfolgt auch in diesem Modus die Verknüpfung der einzelnen Bestandteile auf grafische Art und Weise im *Verknüpfungs- und Verhaltenseditor* bzw. alternativ im *3D-Editor*. Um aus der komplexen interaktiven Szene eine 3D-Applikation zu machen, können in einem Szeneneditor bzw. im 3D-Editor noch Kameraeinstellungen geändert und Lichtquellen, Blickpunkte etc. definiert werden. Dies kann auch mit Hilfe eines *Wizards* erfolgen, der zusätzlich auch die Transformation in ein ausführbares 3D-Zielformat vornimmt.

Innerhalb des CONTIGRABUILDERS wird unabhängig vom Nutzungsmodus intern eine CONTIGRA-Applikation erzeugt, die dann auf eine komplexe Wurzelkomponente verweist. Diese besitzt Subszenengraphen und einen Komponentengraph und kann als eigentliche 3D-Szene in beiden Modi editiert werden. Erst beim Speichern muß entschieden werden, ob die gesamte Applikation oder nur die Wurzelkomponente abgelegt werden soll.

6.4.3 Realisierte Editoren

Die bereits realisierten Editoren werden hier in kompakter Form vorgestellt. Einige Bestandteile des CONTIGRABUILDERS, darunter das Nachrichtenfenster, das Menüsystem, ein Programmkonfigurationseditor und die Komponentenpalette, sind als Kernwerkzeuge fest in das Rahmenwerkzeug integriert. Die meisten Editoren sind jedoch als Editor-Plug-Ins realisiert, da es für sie auch alternative Realisierungen bzw. zusätzliche Werkzeuge geben kann.

6.4.3.1 Komponentenpalette

Die Komponentenpalette dient der Anzeige und Gruppierung zur Verfügung stehender Komponenten für ein bestimmtes Projekt oder eine Firma. Sie stellt damit eine einheitliche visuelle Komponentenbibliothek dar, die einen komfortablen Zugriff auf Komponenten unabhängig von ihrer Herkunft erlaubt. Sowohl der Import als auch die Zuordnung zu Gruppen ist möglich. Dabei können visuelle Hierarchien durch Ordner und Unterordner innerhalb einer Hierarchiestufe erzeugt werden. So ließen sich z.B. Teile der im Unterkapitel 3.4.2 vorgestellten Widget-Hierarchie abbilden. Vorschaubilder der selektierten Komponente in unterschiedlichen Größen und die ausführliche Darstellung von Metainformationen einer Komponente in einem separaten Fenster erleichtern die Auswahl passender Bestandteile der zu erstellenden Szene. Eine alternative Miniaturansicht stellt eine platzsparende Lösung für schnelles Arbeiten dar. Die Abbildung 28 zeigt das Werkzeug in beiden Varianten.

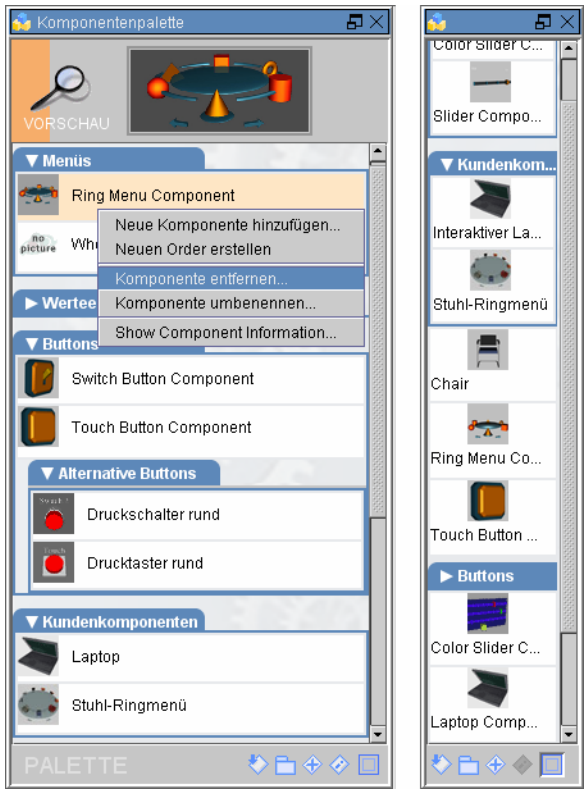

Abbildung 28: Komponentenpalette im CONTIGRABUILDER

6.4.3.2 Baumansicht

Der Baumansichts-Editor bildet neben dem 3D-Editor das zentrale Tool innerhalb des Autorenwerkzeuges CONTIGRABUILDER. Er bietet eine textuelle und ikonische Sicht auf die Realisierung einer Anwendung oder Komponente im Sinne der hierarchischen Anordnung ihrer Szenengraphbestandteile und Subkomponenten. Somit spiegelt er wesentliche Teile von *CoComponentImplementation*-Instanzdokumenten abzüglich sämtlicher Verknüpfungen wider. In der Abbildung 29 sind verschiedene Ansichten dieses Hierarchieeditors zu sehen.

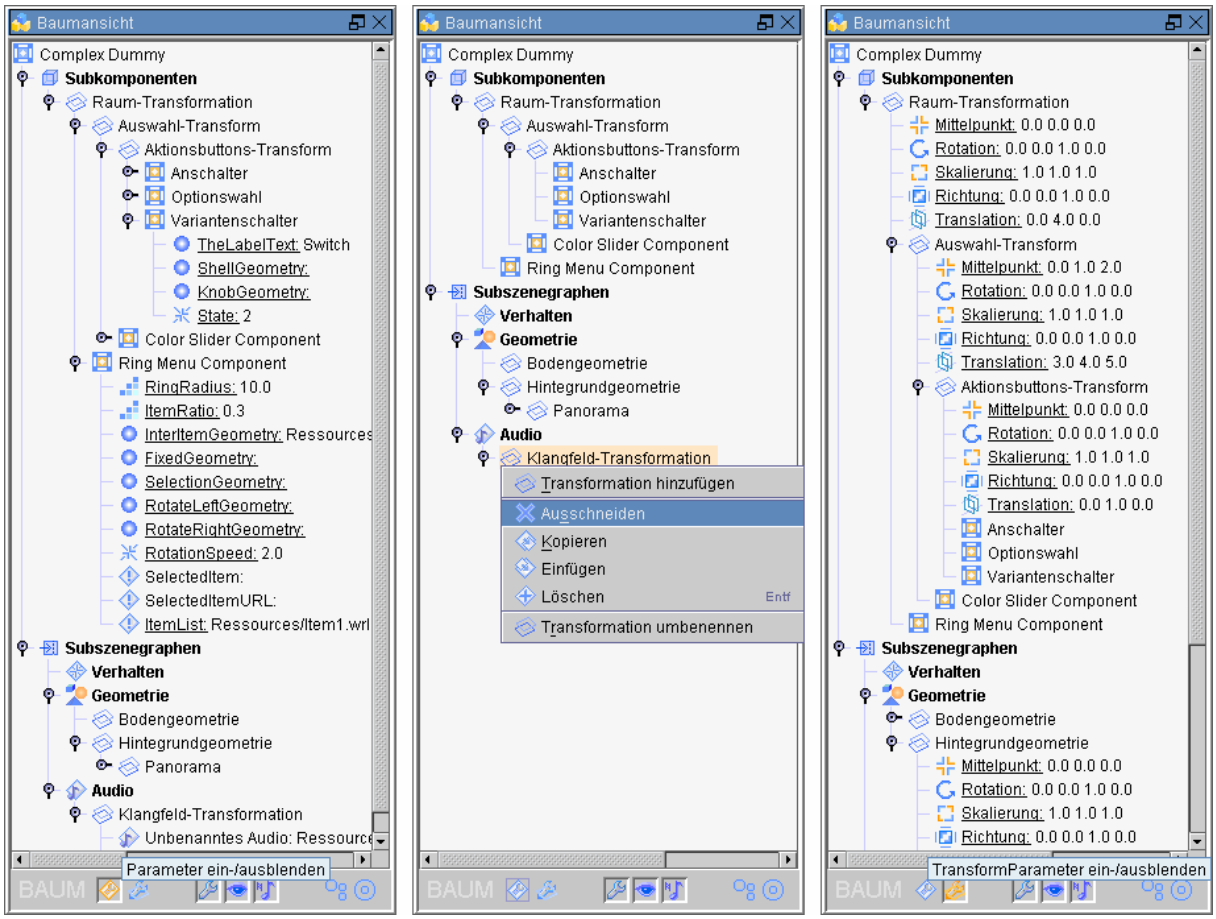

Abbildung 29: Baumansicht im CONTIGRABUILDER

Das Baumansicht-Werkzeug ist in die Bestandteile *Subkomponenten* und *Subszenengraphen* mit den Unterbäumen *Verhalten, Geometrie* und *Audio* unterteilt. Sämtliche Bestandteile lassen sich beliebig aus- und einklappen, wobei diverse Optionen je nach Interesse des Autors zur Verfügung stehen. Transformationshierarchien werden durch Hinzufügen von Transformationen erzeugt, die als Gruppenknoten für Subkomponenten oder Subszenengraphen dienen. Sowohl Transformationsfelder (Rotation, Skalierung etc.) als auch Komponentenparameter lassen sich auch textuell editieren, wodurch dieser Editor eine vor allem für Programmierer geeignete Alternative zum komfortableren Komponenteninspektor darstellt. Diverse Baumeditieroperationen sowie Umbenennungsmöglichkeiten stehen ebenfalls zur Verfügung. Eine dreidimensionale Alternativrealisierung für dieses Werkzeug stellt der *3D-Editor* dar, innerhalb dessen die hier textuell in Transformationshierarchien angeordneten Bestandteile tatsächlich an ihrer aktuellen Raumposition betrachtet und manipuliert werden können.

6.4.3.3 Komponenteninspektor

Der Komponenteninspektor dient der Anzeige und dem Editieren sämtlicher Schnittstelleneigenschaften einer gewählten Komponente. Somit stellt er ein visuelles Werkzeug für alle Teile eines *CoComponent*-Instanzdokuments dar. Ausgehend von der Einteilung in die drei Hauptbestandteile Parameter, Dokumentation und Meta-Informationen bot sich eine ebenenartige Darstellung in Form von *Tabbed Panes* an. In der Abbildung 30 ist der Komponenteninspektor mit den drei Bereichen *Parameter* (oben), *Dokumentation* (links unten) und *Information* (rechts unten) zu sehen.

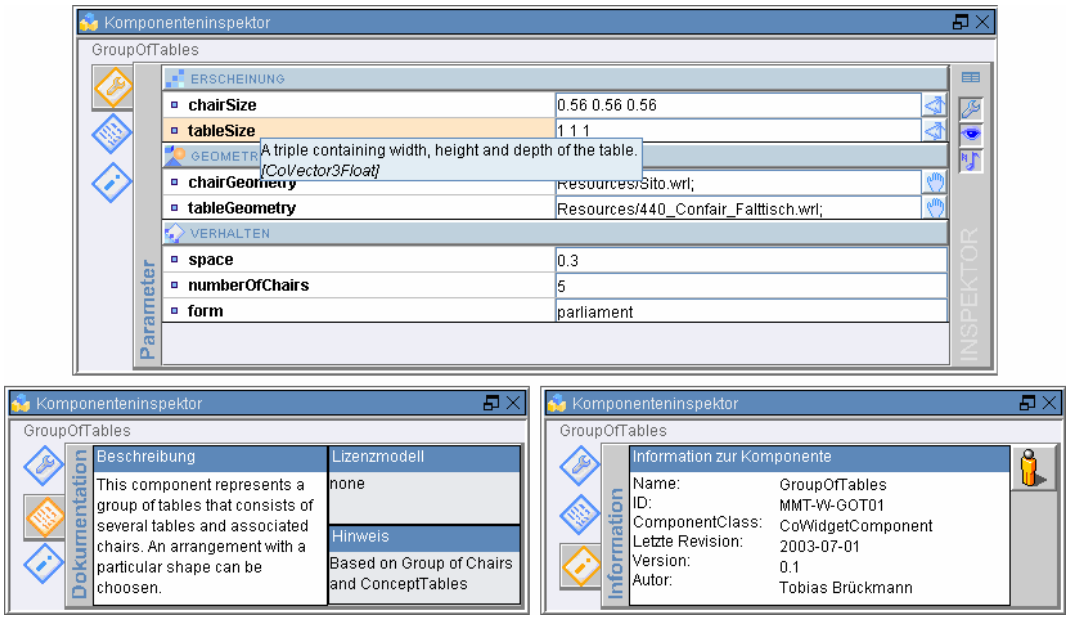

Abbildung 30: Komponenteninspektor im CONTIGRABUILDER

Während im *ApplicationBuilder*-Modus die Dokumentation der Komponente und weitere Metainformationen nur angezeigt, nicht jedoch editiert werden können, stellt der Parametereditor das wesentliche Werkzeug zur Konfiguration einer Komponente dar. Basis dafür ist das Parameterkonzept von Komponentenschnittstellen (s. Unterkapitel 5.3.3). Parameter sind nach den Kategorien *Erscheinung, Geometrie, Verhalten, Audio* und *Allgemein* unterteilt, wobei sich jede Gruppe kollabieren oder expandieren läßt. Je nach Parametertyp existieren verschiedene Editoren, dabei reicht die Spannweite von einfachen Textfeldern über Vektoreditoren, Dateiauswahldialoge und Farbwähler bis hin zu komplexen Parametereditoren.

6.4.3.4 Verknüpfungs- und Verhaltenseditor

Mit dem *Verknüpfungs- und Verhaltenseditor* werden Szenengraphbestandteile und Komponenten über ihre Felder respektive Parameter miteinander verknüpft. Basis ist das im Unterkapitel 5.4.5 dargestellte CONTIGRA-Verknüpfungskonzept. Somit repräsentiert dieser Editor sämtliche Verknüpfungsteile eines *CoComponentImplementation*-Instanzdokuments. Abbildung 31 zeigt eine Ansicht des Editors.

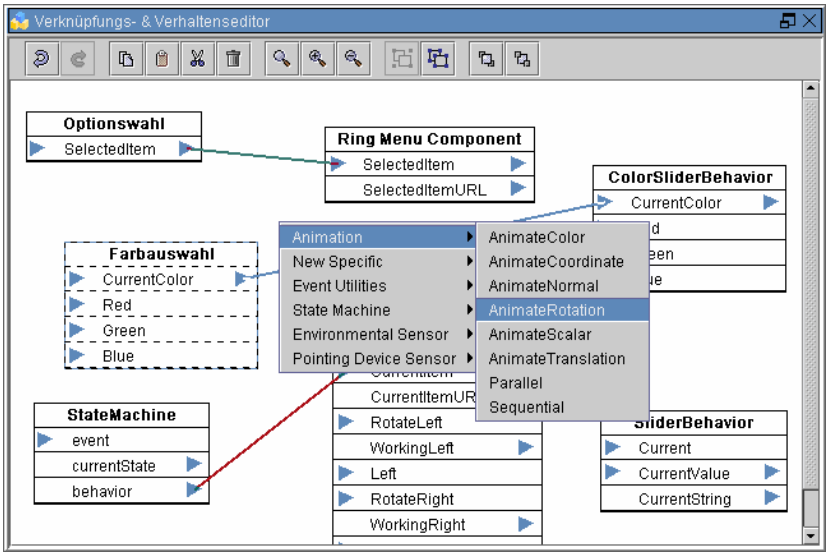

Abbildung 31: Verknüpfungs- und Verhaltenseditor im CONTIGRABUILDER

Knoten und Komponenten werden jeweils mit ihren Ports dargestellt, wobei visualisiert wird, ob sie Ereignisse empfangen oder senden können. Ports gleichen Datentyps lassen sich miteinander verknüpfen. Der durch unterschiedliche Farben visualisierte Verknüpfungstyp kann aus einem Kontextmenü gewählt werden. Auch das Hinzufügen neuer Verhaltensknoten ist über ein Menü möglich, diese werden automatisch dem Verhaltensgraph hinzugefügt.

6.4.4 Unterstützung verschiedener Autorenrollen

Wie bereits erwähnt, unterstützt der CONTIGRABUILDER mehrere Autorenrollen. Zur Zeit lassen sich die Modi *Programmierer, Designer, Soundexperte* und *Alle Autorenrollen* einstellen. Durch die globale Einstellung über einen Konfigurationsmenüpunkt bzw. Icons der Werkzeugleiste (in Abbildung 32 als Überlagerung gezeigt) werden alle Werkzeuge auf diese Autorenrolle synchronisiert bzw. andere Werkzeuge eingeblendet. So könnte beispielsweise für Designer die eher technische Baumansicht zugunsten des visuellen 3D-Editors oder alternativer grafischer Editoren ausgeblendet werden. Es handelt sich jedoch nicht um eine restriktive Einstellung, die bestimmte Funktionalität verbietet. So hat man in den Editoren *Baumansicht* und *Komponenteninspektor* ebenfalls Icons zur Einstellung von Autorenrollen, wobei diese hierbei nicht exklusiv, sondern überlagernd eingestellt werden können. Basis dafür ist die im Unterkapitel 5.3.3.3 erläuterte Definition der Sichtbarkeit von Komponentenparametern für Autorenrollen. In der Abbildung 32 läßt sich im linken Teil erkennen, wie sich die Zahl der Parameter für die Komponenten „Ring Menu Component“ und „Variantenschalter“ reduziert, wenn die Sichtbarkeitsrolle „Designer“ ausgeschaltet wird. Rechts im Bild ist der gleiche Effekt für ein Ringmenü im Komponenteninspektor zu sehen. Während Designer primär für die visuelle Konfiguration der einzelnen Geometrieparameter verantwortlich sind, ist für Programmierer beim Ringmenü die Zahl der Parameter auf die Einstellung von Drehgeschwindigkeit und konkreten Auswahlobjekten reduziert.

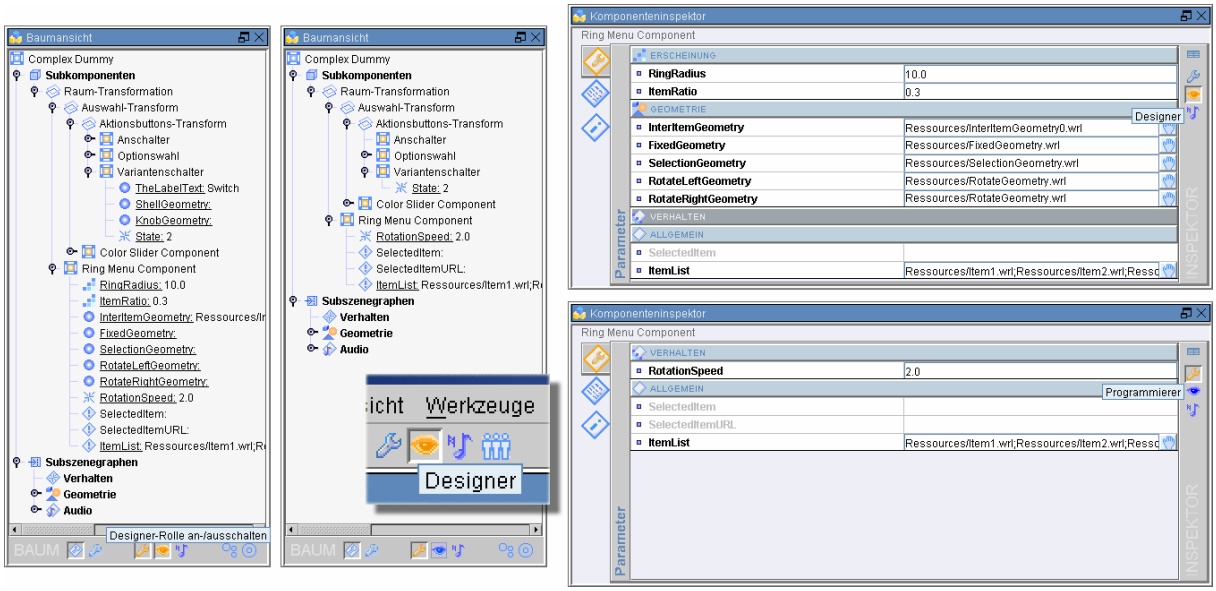

Abbildung 32: Einstellung der CONTIGRA-Autorenrollen und Auswirkung auf die Editoren *Baumansicht* und *Komponenteninspektor*

6.5 Die prototypische Realisierung

Der CONTIGRABUILDER stellt mit seinem Plug-In-Konzept ein erweiterbares Autorenwerkzeug zur Erstellung interaktiver 3D-Komponenten und -Applikationen dar. Eine detaillierte Beschreibung und UML-Diagramme der entstandenen Klassen sind in den Arbeiten von Vitzthum [Vitzt02] für den Werkzeugrahmen und die Plug-In-Architektur, von Ebert [Ebert02] für das interne Objektmodell und unter [KP2002@] für die wesentlichen Editorkomponenten zu finden. Auf eine Darstellung der Klassenhierarchie und weiterer Realisierungsdetails soll deshalb hier verzichtet werden. Es werden jedoch einige Realisierungsaspekte stichpunktartig zusammengefaßt und um Informationen zur Transformation von CONTIGRA-Anwendungen und -Komponenten in 3D-Grafikformate bzw. Webpräsentationen ergänzt.

6.5.1 Verschiedene Realisierungsaspekte

- Programmiersprache *Java*, Benutzungsoberfläche aus *Swing-Komponenten,* dadurch plattformübergreifende Lösung.
- Insgesamt *395 Java-Klassen* für Objektmodell, Rahmenwerkzeug und Editoren.
- *Model-View-Controller-Realisierung* sämtlicher Editoren.
- *Internationalisierung*: gemäß Java-Internationalisierungskonventionen; Spracheinstellung im Konfigurationsmenü möglich; z.Z. Englisch und Deutsch mittels sprachabhängiger Ressourcen-Dateien für alle Editoren und das Gesamtwerkzeug.
- *Speicherung von aktuellen GUI-Einstellungen (Properties/Presets)* in Form von XML-Dateien in separaten Verzeichnissen für jeden Nutzer (z.Z. Speicherung von Sprache und Autorenrolle sowie Detailkonfigurationen aller Editoren; eigener *ConfigManager*).
- *Eigener User Interface Manager (ContigraUIManager)* zur flexiblen Anpassung der gesamten Benutzungsoberfläche (z.B. Farben und Schriftarten für alle Swing-Komponenten) und separate UI-Manager für einzelne Editoren (*InspectorUIManager, PaletteUIManager* etc.).
- *Einheitliches UI-Design:* Konsistente Optik im gesamten Autorenwerkzeug; 287 speziell entworfene Icons; eigener Styleguide mit verbindlichen Regeln für Farben, Schriften und Icon-Design.
- *Flexibles Fenster-Management*: Ein- und Ausdocken, Ein- und Ausblenden sowie nutzerbestimmtes Dimensionieren von Editorfenstern (über *Java Split Panes*).
- *drag and drop* auch zwischen den einzelnen Editoren über spezielle Datenaustauschobjekte (*Transferables* mit zahlreichen *DataFlavors*) möglich.
- *Internes Objektmodell:* Spezialisierte Knoten- und Managerklassen mit *Convenience*-Funktionen für alle wichtigen XML-Elemente der Schemata *(Design-Time Data-Binding)*.
- *Plug-In-Architektur:* Erweiterungen um neue Editoren durch Java-Interfaces für verschiedene Plug-In-Typen möglich.

In der Abbildung 33 ist die Gesamtstruktur des CONTIGRABUILDERS mit wichtigen Klassen für Objektmodellknoten, Manager, Plug-Ins und Editoren im Überblick dargestellt.

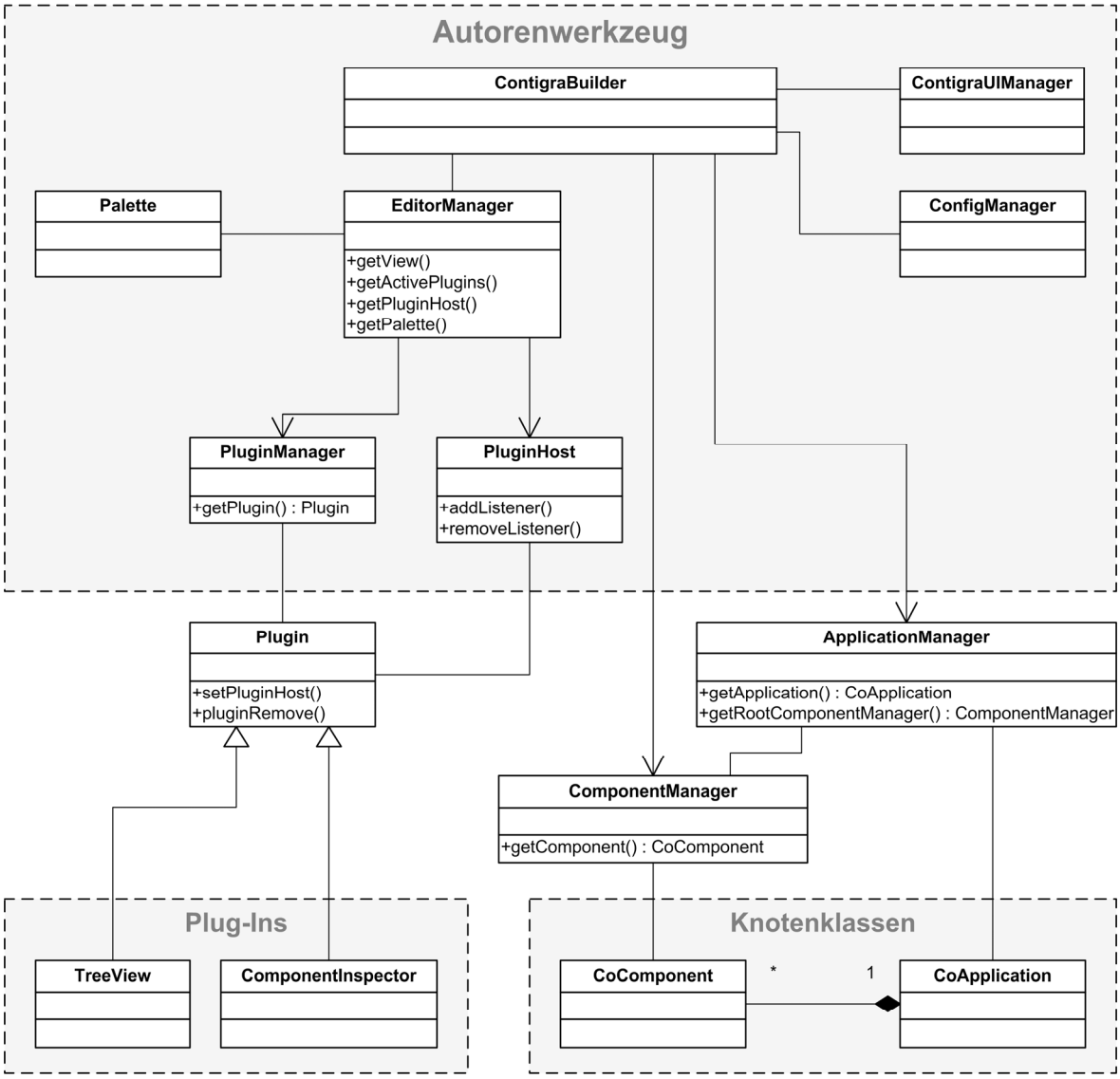

Abbildung 33: Die Gesamtstruktur des Autorenwerkzeuges CONTIGRABUILDER mit wichtigen Managern und Editoren (vereinfachte Darstellung)

6.5.2 Transformation in 3D-Grafikformate bzw. Webpräsentationen

Mit dem CONTIGRA-Dokumentenmodell sollte eine Unabhängigkeit von proprietären 3D-Grafikformaten geschaffen werden. Da es sich um kein eigenes 3D-Format handelt, sondern – vereinfacht gesprochen – lediglich verschiedene Szenengraphbestandteile gekapselt und miteinander verknüpft werden, ist nach der Erstellung von CONTIGRA-Komponenten bzw. 3D-Applikationen eine Transformation in ein Zielformat notwendig. Das CONTIGRA-interne Objektmodell – bzw. seine externe Repräsentation in Form von XML-Dateien – kann demnach in spezifische 3D-Grafikformate oder aber in andere Präsentationsformate, z.B. HTML für Webportale, umgewandelt werden. Damit können auch die Ebenen *Distribution* und *Laufzeit* bei der Entwicklung komponentenbasierter 3D-Anwendungen (s.a. 4.4.2) werkzeugseitig unterstützt werden.

Für Transformationen gibt es zwei grundsätzliche Möglichkeiten. Die erste basiert auf dem Objektmodell, dessen Knotenklassen um Exporter-/Konvertierungsfunktionalität erweitert werden können, oder für das ein eigener Manager entwickelt werden kann, der verschiedene Transformationsklassen verwaltet. Die alternative Möglichkeit besteht in der direkten Arbeit mit der XML-Repräsentation, also den CONTIGRA-Dateien, und ihrer Konvertierung in verschiedene Zielformate unter Nutzung von XSLT-Stylesheets (XSL *Transformations* [XSLT@]).

6.5.2.1 Transformation in 3D-Grafikformate

In [Rukzi02] wurde eine XSLT-Stylesheet-Konvertierung von CONTIGRA-Szenen nach VRML97 / X3D realisiert und damit die Durchführbarkeit einer Formattransformation demonstriert. Sowohl die Grammatiken *CoApplication, CoComponent, CoComponentImplementation* als auch X3D sowie die *Audio3D*- und *Behavior3D*-Schemata liegen im XML-Format vor. Da CONTIGRA-Applikationen und -Komponenten aus verschiedenen Instanzdokumenten dieser Grammatiken bestehen, können für die Formattransformation nach X3D jeweils eigens entwickelte XSLT-Stylesheets genutzt werden. Die dabei entstehenden X3D-Dokumente lassen sich in einem nachfolgenden Schritt mit Hilfe eines vom Web3D-Consortiums [Web3D@] zur Verfügung gestellten XSLT-Stylesheets dann nach VRML97 konvertieren.

Die oben erwähnte alternative Transformation auf Codebasis wird in aktuellen Arbeiten untersucht und prototypisch umgesetzt. Dabei wird das CONTIGRA-Objektmodell als Realisierungsbasis für die Konvertierung von 3D-Applikationen in die imperativen Zielformate Java3D und OpenSG verwendet.

6.5.2.2 Erzeugung von Webpräsentationen

Die XML-Repräsentation von CONTIGRA-Komponenten kann ebenfalls genutzt werden, um Teile eines Webportals automatisch zu generieren, über das sich Komponenten suchen, betrachten, auswählen und herunterladen lassen. In [Spalt02] wird darauf hingewiesen, daß eine Einteilung eines Komponentenrepositoriums nach Kategorien plus Suchmöglichkeiten auf den Metadaten bereits eine gute Arbeitsbasis bieten. So wurden auch zwei XSLT-Stylesheets im Projektkontext entwickelt, mit denen sich eine über Kategorien navigierbare HTML-Präsentation der existierenden CONTIGRA-Komponenten erzeugen läßt (s. [CONTIGRA@]). In der Abbildung 34 ist ein erster Ansatz für ein künftiges Komponentenportal zu sehen, das natürlich dann eine erweiterte Funktionalität zur Suche und für den Download von Komponenten bieten sollte.

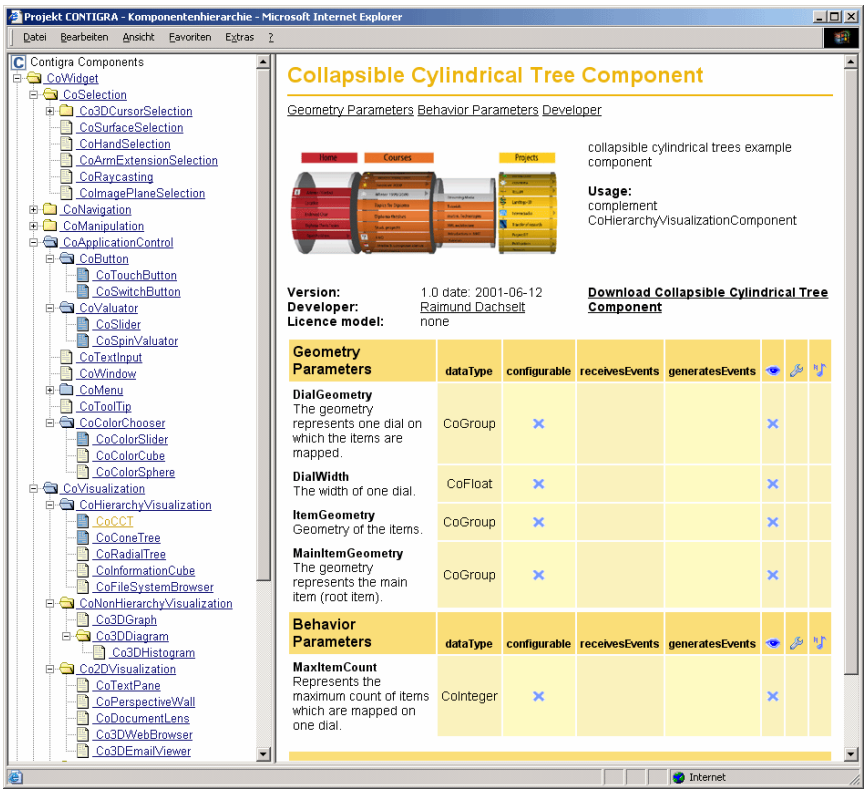

Abbildung 34: Prototypisches CONTIGRA-Komponentenportal

Aus der im Unterkapitel 3.4.2 vorgestellten und in einer XML-Datei beschriebenen Hierarchie von 3D-Widgets wird per Stylesheet-Transformation automatisch der links dargestellte Baum erzeugt. Für alle vorliegenden *CoComponent*-Instanzdokumente wird mit einem weiteren XSLT-Stylesheet jeweils ein Komponenten-Datenblatt (rechts in Abbildung 34) mit Metainformationen und Informationen zu anpaßbaren Parametern generiert. Alle benötigten Dateien und Verzeichnisse für dieses prototypische Komponentenportal werden vollautomatisch erzeugt, jedoch statisch auf einem Webserver abgelegt, da der Transformationsprozeß manuell gestartet werden muß.

In [Ebert02] wird für die gleiche Problematik ein alternativer Ansatz vorgestellt, der jedoch nicht auf den XML-Dateien und ihrer Konvertierung per XSLT beruht, sondern wiederum das CONTIGRA-Objektmodell als Realisierungsbasis verwendet. Dabei handelt es sich um eine vollautomatische und dynamische Generierung von Komponenteninformationen, die bei jedem Aufruf einer Webseite neu erzeugt wird. Für die Komponentenübersicht im Web wurde ein statischer HTML-Rahmen entwickelt, der mit Hilfe von JavaServer Pages mit dynamisch generierten Teilen ergänzt wird. Ebenfalls mit Hilfe von JavaServer Pages werden die einzelnen Datenblätter für Komponenten generiert. Die dynamische Generierung von Vorschaudateien im VRML97- und X3D-Format erfolgt mit Servlets. Ein prototypischer *WebManager* sorgt auch im Sinne eines Cachings für die Pufferung von Komponenteninformationen und bereits erzeugten 3D-Zieldateien. Während auf dem Server die Stylesheet-gestützte Transformation von CONTIGRA-Dateien nach X3D/VRML97 durchgeführt wird, erfolgt auf dem Client dann die Anzeige dieser Dateien in einem geeigneten Browser.

6.6 Beispielanwendungen

Zur Überprüfung der Praktikabilität des CONTIGRA-Ansatzes wurden verschiedene Beispielanwendungen und Komponenten entwickelt. Der aktuelle Stand der implementierten Widget-Komponenten läßt sich auf den Projektwebseiten [CONTIGRA@] abrufen. Neben diesen Standard-Interaktionselementen wurden jedoch auch mehrere projektspezifische 3D-Komponenten entworfen, um beide typischen Bereiche der Komponentenentwicklung abzudecken. Dazu zählen Komponenten für Konferenzmöbel, Stuhlreihungen oder Stuhl-/Tischkombinationen, wie sie in unterschiedlichen Ausprägungen in Abbildung 35 unten zu sehen sind. In der abgebildeten Anwendung lassen sich mit Hilfe eines Ringmenüs Stühle für die Stuhl-/Tischgruppen auswählen, deren räumliche Anordnung – im Bild die Rotation der Gruppen – über Schieberegler beeinflußt werden kann. Diese und andere 3D-Interaktionsstudien sind Teil einer in Entwicklung befindlichen, komplexeren Anwendung zur Einrichtung von Tagungsräumen und Veranstaltungssälen. Die Performance der Anwendungen hängt primär vom 3D-Zielformat ab (in den Beispielen VRML97), wobei im Transformationsprozeß CONTIGRA → Zielformat noch verschiedene Optimierungen denkbar sind.

In Abbildung 35 oben links ist die interaktive Produktpräsentation eines Laptops zu sehen, bei dem sich diverse Teile entfernen, ausklappen oder verändern lassen. An diesem Beispiel konnten zahlreiche Behavior3D-Knoten, vor allem aus dem Animationsbereich, überprüft werden. Während bei direkter Kodierung in VRML97 394 Quelltextzeilen nötig wären, sind es mit CONTIGRA nur 158. Rechts oben im Bild ist die Beispielanwendung *House of Sound* zu sehen, mit der zahlreiche Möglichkeiten des Audio3D-Formates demonstriert werden. Raumklangeigenschaften, Soundquellenpositionen oder Wände bzw. Öffnungen sind dabei einfach deklarativ beschrieben und lassen sich teilweise auch interaktiv zur Laufzeit manipulieren.

Abbildung 35: CONTIGRA-Beispielanwendungen