Zur Architektur interaktiver Systeme zur Erzeugung virtueller Umgebungen

Jörg Sahrhage* und Jens Blauert**
*jetzt: Blaupunkt-Werke GmbH, Hildesheim (joerg.sahrhage@de.bosch.com)
**Institut für Kommunikationsakustik, Ruhr-Universität Bochum, Bochum

1. Einleitung

Der Fortschritt der Computersimulationsverfahren hat dazu geführt, dass Schallfelder in Räumen quasi naturgetreu oder zumindest perzeptiv plausibel simuliert werden können. Signale aus solchen simulierten Schallfeldern können hörbar gemacht, d.h.„auralisiert“ werden. Die Zuhörer erleben dann Hörszenarien, die auf „virtuellen“, computersimulierten Schallfeldern beruhen. Insbesondere, wenn die Zuhörer interaktiv in die Hörszenarien einbezogen werden – z.B. indem sie sich in ihnen bewegen können - kann der Eindruck von „Immersion“ erzeugt werden. Die Zuhörer fühlen sich dann in die virtuellen Umgebung einbezogen – in ihr „präsent“. Dies gilt besonders, wenn die virtuelle Umgebung weitere Sinnesmodalitäten einbezieht – z.B. die taktile und/oder visuelle.

Interaktivität erfordert, dass die virtuelle Umgebung auf Aktionen des Zuhörers praktisch unmittelbar reagiert. Bei Bewegungen des Benutzers ist z.B. die auditive Komponente der virtuellen Umgebung mit Verzögerungen von unter 60 ms und mit einer Auffrischungsrate von mindesten 15 pro Sekunde zu erstellen, damit die perzeptive Präsenz nicht verloren geht. Hieraus ergibt sich die Notwendigkeit, die verwendeten Algorithmen und Softwarestrukturen im Hinblick auf die benötigte Verarbeitungsgeschwindigkeit zu optimieren. Hierzu werden im Folgenden eine Reihe von Hinweisen gegeben, die sich aus einem in Bochum entwickelten System zur Erzeugung einer interaktiven auditiv-taktilen virtuellen Umgebung [1, 2, 4, 5, 6] ergeben haben. Sie lassen sich unschwer auf ähnliche, insbesondere multimodale Systeme übertragen, wie sie heute immer mehr an praktischer Bedeutung gewinnen.

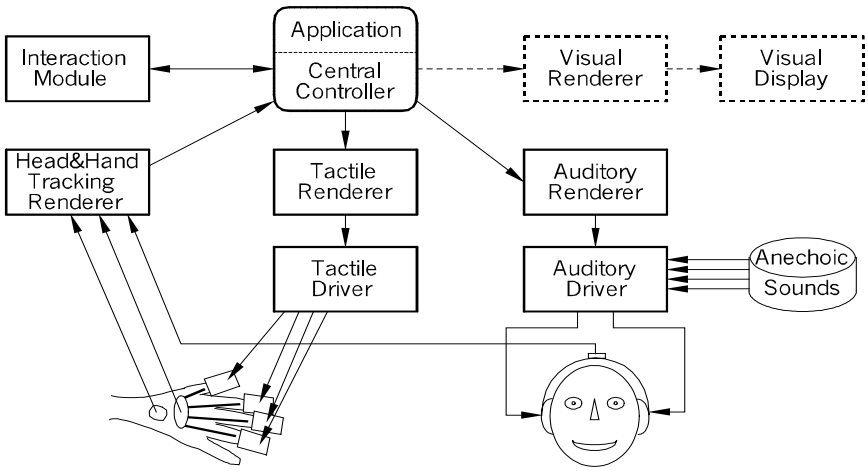

Bild 1: Architektur eines Systems zur Erzeugung bimodaler interaktiver virtueller Umgebungen

2. Zur Systemarchitektur

Die hierarchische und modulare Architektur des Systems ist in Bild 1 schematisch dargestellt. Alle wichtigen Komponenten sind als eigenständige Prozesse realisiert. Die Kommunikation zwischen diesen geschieht mittels einer Reihe vorgefertigter Informationstelegramme („Events“). Die Events durchlaufen grundsätzlich einen zentralen Steuerprozess („Central Controller“), der jedes Ereignis inhaltlich analysiert und gemäß der in der „Application“ festgelegten Regeln reagiert. Bei diesen Reaktionen handelt es sich im einfachsten Fall um die Weiterleitung an einen ereignisgesteuerten Darstellungsprozeß, z.B. die Schallfeldsynthese („Auditory Renderer“), es können jedoch z.B. auch virtuelle Objekte bewegt, eigenschaftlich verändert, oder zum Erscheinen oder Verschwinden gebracht werden. Zur Kontrolle über das Zeitverhalten und über die Synchronität der eingebundenen Prozesse empfiehlt sich eine Client/Server-Architektur, wobei der Central Controller den einzigen Klienten darstellt, während die Renderer als Server fungieren, indem sie entweder Daten auf Anforderung des Central Controllers bereitstellen, oder von diesem erhaltene Daten verarbeiten und an die Display-Einheiten (z.B. Auditory Driver) weitergeben. Dieses Verhalten kann durch das Kommunikationsprotokoll explizit erzwungen werden.

Das System enthält eine zentrale (Welt-)Datenbank, in der die Beschreibungen aller Objekte der virtuellen Welt abgelegt sind. So wird sichergestellt, dass alle Renderer und alle anderen Prozesse hinsichtlich der virtuellen Welt mit den gleichen Daten arbeiten. Zusätzlich ist es jedem Prozess erlaubt, eine Kopie dieser Datenbank seinen eigenen Bedürfnissen anzupassen. Es kann vorkommen, dass in einem speziellen Szenario eine große Anzahl virtueller Objekte benötigt wird, obwohl die Anzahl der Objekte, die der Zuhörer interaktiv beeinflusst, wesentlich kleiner ist. Als Beispiel sei der Rundgang durch ein Haus genannt: Solange die Zimmertüren verschlossen sind, interessieren aktuell nur die Objekte im gerade begangenen Raum. Alle anderen Objekte können dann in der Datenbank als latent (verborgen) definiert werden, die nach Bedarf wieder angeschaltet (sichtbar gemacht) werden können. Verborgene Objekte existieren für die Renderer quasi nicht. Zur weiteren Vereinfachung werden solche

Objekte, die sich in der virtuellen Umgebung nur gemeinsam bewegen, zu Objekt-Clustern zusammengefasst (z.B. Tischplatte und Beine eines Tisches). Alle Events, die sich auf ein solches Cluster beziehen, wirken auf alle seine Teile, so als handelte es sich um ein einziges Objekt.

3. Rendererstrukturen

Renderer haben die Aufgabe, auf definierte Events mit einem sinnvollen Darstellungsvorgang zu reagieren. Daher sind beim Rendererentwurf Routinen für die Handhabung aller vorkommenden Events vorzusehen. Um diesen Gestaltungsvorgang zu vereinfachen, wurde eine Renderer-Basisklasse geschaffen, die alle Grundfunktionen eines Renderers in Hinblick auf die Verwaltung der Welt-Datenbank und die Kommunikation mit dem Central Controller enthält. Die Vorteile eines solchen Minimalrenderers sind offensichtlich. Für jeden spezialisierten Renderer braucht nur eine Erweiterung des Basisrenderers derart vorgenommen werden, dass diejenigen Routinen erweitert, ersetzt oder zusätzlich implementiert werden, die die Behandlung derjenigen speziellen Events betreffen, die für den speziellen Renderer relevant sind. Änderungen in der Welt-Datenbank gehen durch das Prinzip der Vererbung bei jeder Neukompilierung automatisch auf alle von dem Basisrenderer abgeleiteten Renderer über.

Im Idealfall kann die Darstellung der virtuellen Umgebung durch die Renderer „synchron“, d.h. im gleichen Takt unmittelbar im Anschluß an den Empfang und die Verarbeitung eines jeden Events in der Datenbank, erfolgen. Das setzt allerdings voraus, dass jeder der beteiligten Renderer ausreichend schnell ist, um innerhalb der Zeitspanne einen kompletten Aktualisierungszyklus durchzuführen, bis das nächste Event beim Central Controller zum Versand ansteht. Falls einer der Renderer zu langsam ist, kommt es bei einem derartigen Kommunikationsprotokoll zu unerwünschten Konsequenzen, denn die nächste Auffrischung der virtuellen Umgebung wird blockiert, bis dieser langsame Renderer sein Bestätigungssignal gesendet hat. Um dies zu vermeiden, können die Renderer „teilsynchron“ betrieben werden. Dazu werden die beim Renderer ankommenden Events in einer Warteschlange der Kommunikationsschnittstelle zwischengespeichert. Alle Events, die im Verlaufe des letzten Aktualisierungszyklus des Renderers in der Warteschlange aufgelaufen sind, werden zunächst ausgelesen und vor einem erneuten Rendering in die zentrale Welt-Datenbank eingearbeitet. Erst wenn die Warteschlage abgearbeitet und die Datenbank somit aktualisiert ist, wird der nächste Darstellungszyklus gestartet und mit einer Bestätigung abgeschlossen. Auf diese Weise kann ein Blockieren des Systems wirksam verhindert und sichergestellt werden, dass immer die aktuellsten Daten zur Darstellung kommen, jedoch um den Preis, dass die Darstellungen der langsameren Renderer an Gleichmäßigkeit einbüßt.

4. Kommunikations- und Synchronisationsstrategien

Die Wahl der im Einzelfall anzuwendenden Kommunikationsstrategie – sie wird im wesentlichen durch den Steuerprozess (Central Controller) implementiert – hängt davon ab, inwieweit Synchronität zwischen unterschiedlichen Renderern erforderlich ist - z.B. wenn die Repräsentationen mehrerer Sinnesmodalitäten dargestellt werden sollen. Um Synchronität zu erlangen, würde der Central Controller allen beteiligten Renderern die Nachricht über eine Veränderung der virtuellen Umgebung gleichzeitig zusenden und mit der Bearbeitung des nächsten Ereignisses erst fortfahren, wenn er von allen Renderern eine Bestätigung der Ausführung erhalten hat. Nachteilig wäre wiederum, dass allen beteiligten Renderern dieselbe Auffrischungsrate aufgezwungen würde und somit der langsamste Renderer die System-Performanz begrenzte. Andererseits muss Synchronität im erforderlichen Maße gewährleistet sein, damit die Repräsentationen unterschiedlicher Sinnesmodalitäten perzeptiv verschmelzen.

Empfehlenswert hinsichtlich ihrer Rechenzeiteffizienz bzw. ihrer Eignung für zeitkritische Zusatzaufgaben sind wiederum teilsynchrone, ereignisgesteuerte Kommunikationsstrategien. Sie zeichnen sich dadurch aus, dass der Steuerprozess dann und nur dann aktiv wird, wenn er, vergleichbar einem Interrupt, von irgendeinem der Renderer ein neues Event zugespielt bekommen hat. Auf diese Weise läßt sich sein Rechenzeitbedarf auf ein Minimum reduzieren. Hat der Central Controller jedoch, zeitkritische Aufgaben, wie z.B. präzise Zeitmessungen oder das zeitlich präzise Auslösen bestimmter Veränderungen in der virtuellen Umgebung im Rahmen eines Experimentes, so empfiehlt sich die Variante, bei der der Central Controller seine Kommunikationsschnittstelle ebenso ungezielt, jedoch mit einstellbar hoher Abtastrate in nichtblockierender Weise nach eingegangen Events abfragt, um bei Bedarf darauf zu reagieren. Prinzipiell können alle Kommunikationsschnittstellen asynchron betrieben werden. In unserem System wurde allerdings die Kommunikationsschnittstelle des Head-and-Hand-Tracking-Moduls in der Regel blockierend, d.h. synchron betrieben. Dieses Vorgehen ist damit gerechtfertigt, dass es meistens nicht sinnvoll ist, die virtuelle Umgebung häufiger aufzufrischen als Positionsdaten von dem interaktiv agierenden Subjekt hereinkommen.

5. Schlussbemerkung

Die o.g. Architekturempfehlungen werden aufgrund der Ergebnisse umfangreicher Testläufe unseres Systems mit unterschiedlichen Architektur- und Strategievarianten gewonnen. Ein ausführlicher Bericht dazu ist in Vorbereitung [3]. Die Arbeiten wurden von der EU im Rahmen des Verbundprojektes SCATIS gefördert. Die Verfasser danken dem ehemaligen IKA-Mitarbeiter H. Lehnert, der viele Ideen zu dieser Arbeit beigesteuert hat.

Literaturhinweise

[1] Blauert, J., Lehnert, H., Sahrhage, J. & Strauss, H. (2000). An Interactive Virtual-Environment Generator for Psychoacoustic Research, I: Architecture and Implementation, ACUSTICA/acta acustica 86, 94-102.

[2] Djelani, Th., Pörschmann, Chr., Sahrhage J. & Blauert, J. (2000), An Interactive Virtual-Environment Generator for Psychoacoustic Research, II: Collection of Head-Related Impulse Responses and Evaluation of Auditory Localization, ACUSTICA/acta acustica (eingereicht)

[3] Sahrhage, J. (2000), Auditive virtuelle Umgebungen: Theorie, Realisierung und Anwendung (in Vorbereitung)

[4] Sahrhage, J., Blauert, J. & Lehnert, H. (1996), Implementation of an Auditory/Tactile Virtual Environment, Proc. 2nd FIVE Int. Conf. on Advances, Applications and Impact of Immersive Virtual Environments, 18-26, PERCRO, Pisa

[5] Shinn-Cunningham, B. G., Lehnert, H., Kramer, G., Wenzel, E. M., & Durlach, N. I. (1997). Auditory displays. In: Binaural and Spatial Hearing. R. H. Gilkey & T. B. Anderson (eds.), Lawrence Erlbaum, Mahwah, NJ.

[6] Strauss, H. & Blauert, J. (1995). Virtual Auditory Environments. Proc. 1st FIVE Int. Conf., 123-131, Queen Mary & Westfield Coll., London.