Verwaltung spatio-temporaler Audiodaten für die Wellenfeldsynthese

Thomas Heimrich, Kai-Uwe Sattler
Technische Universität Ilmenau
Datenbanken und Informationssysteme
{thomas.heimrich|kus}@tu-ilmenau.de

Katrin Reichelt, Gabriel Gatzsche
Fraunhofer Institut für
Digitale Medientechnologie
{mch|gze}@idmt.fraunhofer.de

Abstract: Neue Anwendungsgebiete für Datenbanksysteme resultieren aus neuen Entwicklungen in der Medientechnologie. Mit dem am Fraunhofer-Institut entwickelten IOSONO-System können Schallquellen an einer beliebigen Position in einem Hörraum (z.B. Kinosaal) erzeugt werden. So entstehen räumliche Klänge in einer bisher nicht gekannten Qualität. Die Verwaltung der benötigten spatio-temporalen Audiodaten erfordert spezielle Datenbanktechniken. Wir zeigen, wie die von uns entwickelten Ausgabebedingungen für die Datenmodellierung und die korrekte Datenausgabe genutzt werden können. Für die Datenausgabe erzeugen wir Ausgabe-Schedules, die die Grundlage für die Optimierung der Datenorganisation bilden.

1 Einführung

Spatio-temporale Daten – d.h. Daten mit Raum- und Zeitbezug – müssen heutzutage in verschiedenen Anwendungsbereichen verwaltet werden. Die Aufgabenstellungen aus Sicht des Datenmanagements sind dabei insbesondere

- die Modellierung und Repräsentation der Raum- und Zeitbeziehungen beispielsweise in Form von Constraints sowie
- das Retrieval unter Berücksichtigung dieser Beziehungen bzw. Constraints.

Ein neues Anwendungsgebiet für die Verwaltung spatio-temporaler Daten ist das IOSONO-Klangwiedergabesystem. IOSONO ermöglicht die Wiedergabe von hochqualitativem räumlichen Klang basierend auf dem Ansatz der Wellenfeldsynthese [BV94]. Klänge sind an einer stabilen Position sowohl innerhalb als auch außerhalb des Hörraums wahrnehmbar. Tonmeistern wird es ermöglicht, einer großen Anzahl in einem Raum befindlichen Zuhörern eine räumlich klar aufgelöste wahrnehmbare Klangszene zu präsentieren. Der „Sweet-Spot", d.h. der Punkt der optimalen Klangwahrnehmung beschränkt sich nicht wie bei 5.1-Surroundsoundsystemen auf einen bestimmten Punkt im Raum, sondern erstreckt sich über die gesamte Hörfläche, auf der sich der Hörer frei bewegen kann. IOSONO ist damit insbesondere für hochqualitative räumliche Wiedergabe wie z.B. für Filme und Konzerte geeignet.

Das weltweit erste Kino, das mit einem IOSONO-System ausgestattet ist, befindet sich in Ilmenau. Ein zweites Referenzsystem mit Anwendung im Kinobereich wurde im Sommer 2004 in Los Angeles in der Nähe der Hollywood-Filmstudios als Demosystem für Regisseure, Filmproduzenten usw. installiert. Desweiteren befindet sich ein drittes IOSONO-System an der Universität von Surrey (Großbritannien) in einem Virtual-Reality-Labor.

Zur Wiedergabe der Klangszene wird eine hohe Anzahl um die Wiedergabefläche angeordneter Lautsprecher verwendet, welche separat durch ein oder mehrere *Renderer* angesteuert werden. Das IOSONO-System wird deshalb nicht mit bereits vorberechneten Lautsprechersignalen gefüttert, wie es bei kanalorientierten Systemen (z.B. ein 5.1-Surroundsoundsystem) der Fall ist, sondern mit *objektorientierten Klangszenen*. Beim objektorientierten Ansatz wird jedes Klangereignis zusammen mit seinen Eigenschaften wie z.B. der Position innerhalb einer Klangszene an das IOSONO-System übertragen, welches mit Hilfe ein oder mehrere Renderer die Signale für jeden einzelnen Lautsprecher des Systems berechnet. Vorteile dieses Ansatzes sind u.a. die Möglichkeit der Anpassung der Klangwiedergabe an die persönliche Wiedergabesystemkonfiguration des Hörers, der Interaktion des Benutzers mit der Klangszene wie z.B. das nachträgliche Verändern der Position einer Klangquelle oder der Neugestaltung und Wiederverwendung von Szenenbestandteilen.

In diesem Beitrag stellen wir Datenbanktechniken vor, die zur Modellierung und Datenbereitstellung von spatio-temporalen Audiodaten im IOSONO-System dienen. Die Entwicklung dieser Datenbanktechniken ist Gegenstand eines laufenden Projektes.

2 IOSONO-Systemarchitektur und Anforderungen

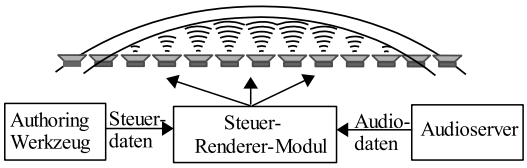

Abbildung 1: Aufbau und Datenfluss des IOSONO-Soundsystems

Abbildung 1 zeigt schematisch den Aufbau und den Datenfluss des IOSONO-Systems im gegenwärtigen Stand. Das entwickelte Authoring-Werkzeug erlaubt es dem Nutzer, Szenen zu erstellen, zu editieren und das IOSONO-System damit zu steuern [MRB+03]. Eine Szene besteht sowohl aus Informationen zu den einzelnen virtuellen Audioquellen als auch aus den Audiodaten. Die Eigenschaften der Audioquellen und die Referenzen auf die Audiodaten werden in einer XML-Szenendatei gespeichert. Die Audiodaten selbst werden auf dem Audioserver abgelegt und von dort aus an das Renderer-Modul übertragen. Um ein Wellenfeld zu berechnen, benötigt das Renderer-Modul Informationen über die einzelnen Audioquellen wie z.B. die Positionen der Audioquellen. Aus diesem Grund werden die Szenendaten (als Steuerdaten) ebenfalls an das Renderer-Modul übertragen. Anhand

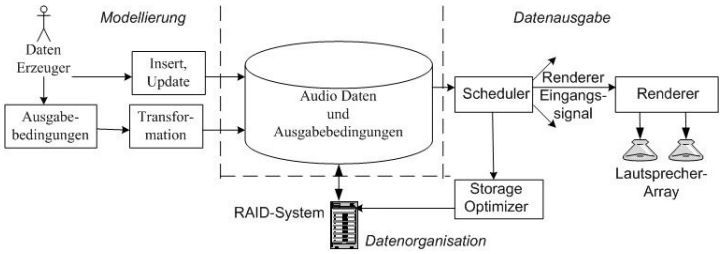

Abbildung 2: Datenverwaltung für das IOSONO-System

der Steuerdaten und der dazugehörigen Audiodaten ist das Renderer-Modul in der Lage, für jeden einzelnen Lautsprecher das entsprechende Signal zu berechnen. Durch die Überlagerung der einzelnen Lautsprechersignale entsteht das synthetisierte Wellenfeld. Aus den Erfahrungen mit diesem System werden folgende Anforderungen abgeleitet:

Authoring: Bei der Produktion von Audioszenen werden Allen-Relationen [All83] zur Modellierung der räumlichen und zeitlichen Beziehungen verwendet, wobei Audioobjekte und Beziehungen unabhängig voneinander verwaltet werden. Dadurch kann die Konsistenz nicht gewährleistet werden und bei der Modifikation von Audiodaten können z.B. zeitliche Bedingungen verletzt werden. Somit ergibt sich ein Bedarf nach einer flexiblen Modellierung und Konsistenzprüfung. Ferner werden Such- und Kompositionsoperationen für Objekte und Szenenteile benötigt, die eine einfache Wiederverwendung ermöglichen.

Rendering: Die Audiodaten werden entsprechend der zeitlichen Bedingungen als Datenstrom im WAV-Format an alle Renderer geschickt. Jeder Renderer nimmt zunächst eine Geometrieberechnung vor, um die für ihn relevanten Audiodaten zu ermitteln. Darüber hinaus müssen die Daten zeitsynchron zum Film auf allen Renderern zur Verfügung stehen. Das aktuelle IOSONO-System ist in der Lage, 32 Audioquellen gleichzeitig zu berechnen. In komplexen Anwendungen wie z.B. Kinofilmen oder Szenen mit immersiven Atmosphären (z.B. Regen oder Applaus) ist es aber erforderlich, dass eine höhere Anzahl an Audioquellen gleichzeitig berechnet wird. Damit ergeben sich noch höhere Anforderungen an die Echtzeit-Datenbereitstellung.

Ziel unserer aktuellen Arbeiten ist es daher, unter Verwendung von Datenmodellierungs- und Datenmanagementechniken *(1)* die gespeicherten Audiodaten und deren Ausgabe konsistent zu den definierten zeitlichen und räumlichen Bedingungen zu halten und *(2)* die Einhaltung der Zeitanforderungen bei der Bereitstellung der Daten für die Renderer zu garantieren. Abbildung 2 zeigt die Architektur eines solchen datenbankgestützten Systems. Das Datenbanksystem speichert neben den Audiodaten auch *Ausgabebedingungen*. Sie definieren, wie die gespeicherten Audiodaten ausgegeben werden sollen, wobei zeitliche und räumliche Beziehungen modelliert werden können. Die Berücksichtigung derartiger Ausgabebedingungen betrifft die folgenden Ebenen:

Die *Datenmodellierung* muss die Definition von zeitlich/räumlichen Ausgabebedingungen für Audiodaten erlauben. Für die Modellierung können Audiodaten als temporale Intervalle betrachtet werden. Die Position der Audioquelle wird durch einen Punkt im

3-dimensionalen Koordinatensystem modelliert.

Die *Datenausgabe* oder das *Scheduling* muss die definierten Ausgabebedingungen beachten, indem ein Ausgabe-Schedule mit einer zeitlichen und räumlichen Ordnung der Audioobjekte erzeugt wird. Die räumliche Ordnung entspricht der Zuordnung zu Renderern, die zeitliche Ordnung der Abspielreihenfolge.

Die *Datenorganisation* ist entscheidend für die effiziente Datenausgabe. Die notwendige Transferrate kann von Medien von CD oder DVD nicht gewährleistet werden. Daher ist eine festplattenbasierte Verwaltung notwendig. Aber auch hierbei sind für komplexe Szenen Optimierungen des Speicherlayouts erforderlich, um die Einhaltung der Zeitforderungen zu garantieren.

3 Existierende Datenbanktechniken für spatio-temporale Daten

Die existierenden Datenbanktechniken haben ihren Ursprung in der Verwaltung von Multimedia-Dokumenten. Die Arbeiten [IM99, MPS+00] führen temporale Bedingungen in interaktive Multimedia-Dokumente ein. Diese Bedingungen beschränken nur die Nutzerinteraktionen. Für Multimedia-Dokumente existieren viele temporale Synchronisationsmodelle [BF98]. Das von uns verwendete Allen-Modell (Allen-Relationen [All83]) ist ein grundlegendes Synchronisationsmodell. Synchronisationsmodelle wurden unabhängig von Multimedia-Datenbanksystemen entwickelt. Die Konsistenz zwischen den Multimedia-Daten und den Synchronisationsbedingungen wurde daher nicht berücksichtigt. Die von uns verwendeten *Difference Constraints* wurden in [ASS00] für die temporale Synchronisation benutzt. Wir haben *Ausgabebedingungen* bereits in [Hei04a] eingeführt. Dort findet sich eine formale Beschreibung und Klassifikation von Ausgabebedingungen. In [Hei05] beschreiben wir die Transformation von Ausgabebedingungen in *Difference Constraints*. [Hei04b] zeigt, wie Ausgabebedingungen in ein Typsystem integriert werden können.

4 Datenmodell

Ausgabebedingungen definieren gültige Zustände für Ausgabeobjekte. Wir haben eine *Ausgabefunktion* f_{out} eingeführt. Diese bildet ein gespeichertes Audioobjekt *(ao)* aus der Menge aller gespeicherten Audioobjekte *(AO)* auf ein Ausgabeobjekt *o* ab. Wir betrachten hier nur Ausgabefunktionen, die raum-/zeitabhängige (spatio/time-depentent) Ausgabeobjekte erzeugen. Als Notation für diese Funktionen verwenden wir $f_{s/td}(ao)$. Gleichung 1 zeigt die formale Definition einer Domäne einer Ausgabefunktion. Alle Ausgabeobjekte o, die von der Funktion $f_{out}(ao)$ erzeugt werden, sind Elemente dieser Domäne.

$$OUTOBJ^{AO}_{f_{out}} = \{o \mid o = f_{out}(ao) \wedge ao \in AO\} \quad (1)$$

Die Ausgabebedingungen beschränken die Menge $OUTOBJ^{AO}_{f_{out}}$. Die allgemeinste Form einer Ausgabebedingung ist $\{\forall, \exists\} o \in OUTOBJ^{AO}_{f_{out}} : F$. Hier ist F eine räumliche

oder zeitliche Bedingung. Für das IOSONO-System besteht die Ausgabefunktion aus einer Geometrieberechnung. Es werden Audios als Ausgabeobjekte erzeugt, die eine räumliche Position im Kinosaal haben. Die Ausgabeobjekte sind also raum- und zeitabhängig. Eine detaillierte Klassifikation und formale Spezifikation von Ausgabebedingungen haben wir in [Hei04a] gegeben.

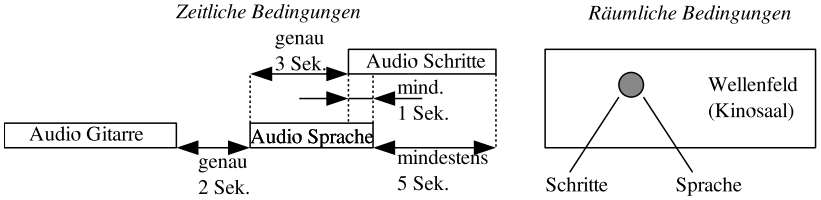

Abbildung 3: zeitliche und räumliche Bedingungen zwischen Audiodaten

Zur Modellierung zeitlicher Beziehungen nutzen wir Allen-Relationen. Die ursprünglichen Allen-Relationen erlauben nur eine qualitative Modellierung von temporalen Beziehungen. Wir haben daher die Allen-Relationen *before, overlap* und *during* so erweitert, dass quantitative Angaben möglich sind. Eine genaue Beschreibung unserer Erweiterungen der Allen-Relationen findet sich in [Hei05].

Die zeitlichen Beziehungen in Abbildung 3 lassen sich durch die erweiterten Allen-Relationen *GITARRE* $before_V(2,0)$ *SPRACHE* und *SPRACHE* $overlaps_{V_s}(3,0,5)$ *SCHRITTE* modellieren. Erstere Relation definiert ein variables Zeitintervall $[2,(2+0)]$ (d.h. exakt 2 Zeiteinheiten) zwischen dem Ende von *GITARRE* und dem Start von *SPRACHE*. Die zweite Relation definiert ein variables Zeitintervall $[3,(3+0)]$ (d.h. exakt 3 Zeiteinheiten) zwischen dem Start von *SPRACHE* und dem Start von *SCHRITTE*. Die Zeit zwischen dem Ende von *SPRACHE* und dem Ende von *SCHRITTE* ist mindestens 5.

Um räumliche Beziehungen zu modellieren, setzen wir die Audioquellen relativ zueinander in Beziehungen (z.B. *quelleA left(c,l) quelleB* – Audioquelle A ist mindestens c und höchstens $c+l$ Längeneinheiten links von Audioquelle B). Eine detaillierte Beschreibung dieser Modellierung haben wir in [Hei05] gegeben. Als Beispiel wollen wir die in Abbildung 3 dargestellten Bedingungen formal definieren. Wir gehen davon aus, dass die Tabelle *AudioModell(Szene,Quelle,AudioObjekt1,AudioObjekt2,AudioObjekt3)* existiert. Das Attribut *Szene* steht für die Szenen-Nummer innerhalb das Films. Die *Quelle* ist ein Objekt in der Szene, das der Ursprung der Geräusche ist. Das kann z.B. eine bestimmte Sprechperson innerhalb der Szene sein. Die Attribute *AudioObjekt1,AudioObjekt2,AudioObjekt3* sind Audio-Objekte, die der Quelle in der entsprechenden Szene zugeordnet sind. Um das Beispiel einfach zu halten, haben wir uns auf drei Audioobjekte und ein einfaches Design beschränkt. In der Realität werden erheblich mehr Audioobjekte benötigt. Wir gehen davon aus, dass das Tupel $\langle$*1,Person1,Gitarre,Sprache,Schritte*$\rangle$ in der Relation *AudioModell* existiert. Die in Abbildung 3 dargestellte räumliche Beziehung lässt sich durch die Relation *Sprache spatial-equal Schritte* modellieren. Die Ausgabebedingungen aus Abbildung 3 lassen sich formal wie folgt definieren:

$$\forall o_1 \in \{o | o = f_{s/td}(ao) \wedge ao \in \{u.AudioObjekt1 | AudioModell(u) \wedge u.Szene = 1$$

$$\wedge u.Quelle = Person1\}\}:$$
$$\exists o_2 \in \{o | o = f_{s/td}(ao) \wedge ao \in \{u.AudioObjekt2 | AudioModell(u) \wedge u.Szene = 1$$
$$\wedge u.Quelle = Person1\}\}:$$
$$\exists o_3 \in \{o | o = f_{s/td}(ao) \wedge ao \in \{u.AudioObjekt3 | AudioModell(u) \wedge u.Szene = 1$$
$$\wedge u.Quelle = Person1\}\}:$$
$$(o_1\ before_V(2,0)\ o_2) \wedge (o_2\ overlaps_{V_s}(3,0,5)\ o_3) \wedge (o_2\ \textit{spatial-equal}\ o_3)$$

Die Ausgabefunktionen $f_{s/td}$ erzeugt räumlich/zeitliche Ausgabeobjekte. Die Geometrieberechnung der Wellenfeldsynthese ist Teil der Ausgabefunktion. Ergebnis der Ausgabefunktion und damit Ausgabe des Datenbanksystems sind Audiodaten und Steuerdaten (Positionsdaten der Klangquelle) für die Renderer. Die Renderer erzeugen mit diesen Daten das Wellenfeld und damit erst die eigentliche Klangquelle im Raum.

5 Scheduling und Rendering

Die räumlich/zeitlichen Ausgabeobjekte jeder Szene wurden relativ zueinander modelliert. Für die Ausgabe aller Audioobjekte einer Szene oder eines Filmes müssen diese in eine absolute räumliche und zeitliche Ordnung gebracht werden. Diese Ordnung wird als Ausgabe-Schedule bezeichnet. Ein Schedule ist ein Ausgabeplan, der die Audiodaten entsprechend der Ausgabebedingungen anordnet.

Eine Voraussetzung für die Erzeugung eines Ausgabe-Schedules ist die Konsistenz der definierten Ausgabebedingungen. Die definierten Ausgabebedingungen müssen daher auf ihre Konsistenz überprüft werden. Dabei wird auch geprüft, ob die Ausgabebedingungen mit den gespeicherten Audiodaten erfüllt werden können. Diese Überprüfung kann bereits beim Insert/Update der Audiodaten durchgeführt werden.

Wir haben erweiterte Allen-Relationen verwendet, um zeitliche Beziehungen zwischen Ausgabeobjekten zu modellieren. Eine Menge erweiterter Allen-Relationen lässt sich in polynomialer Zeit auf Widersprüche überprüfen. Bei der Verwendung der ursprünglichen Allen-Relationen muss ein *Interval Algebra Network* genutzt werden, um die Widerspruchsfreiheit einer Menge von Allen-Relationen zu testen. In [VK86] wurde gezeigt, dass dies ein NP-hartes Problem ist.

Die definierten räumlichen Beziehungen zwischen Ausgabeobjekten müssen auch auf Widerspruchsfreiheit getestet werden. Wir haben eine Möglichkeit gefunden, um beide Gruppen von Ausgabebedingungen einheitlich darzustellen und auf die selbe Weise auf Widerspruchsfreiheit zu testen.

Wir verwenden *Difference Constraints* als datenbankinterne Repräsentation sowohl für zeitliche als auch für räumliche Beziehungen zwischen Ausgabeobjekten. Difference Constraints sind Ungleichungen der Form $a - b \leq c$. a und b sind Variablen. c ist eine Konstante. Sie gibt einen Mindestabstand an, der zwischen a und b liegt. Eine Menge von Difference Constraints kann man in polynomialer Zeit auf Widersprüche untersuchen [RSJM99]. Sowohl zeitliche als auch räumliche Bedingungen lassen sich mit Difference Constraints ausdrücken. Die Details der Transformation von erweiterten Allen-Relationen und räumlichen Relationen in Difference Constraints haben wir in [Hei05] beschrieben.

Definiert man zeitliche Bedingungen mit Difference Constraints, dann werden den Variablen *a* und *b* jeweils Start- oder Endzeitpunkt von temporalen Intervallen zugewiesen. Die Konstante *c* gibt den maximalen Abstand zwischen diesen Zeitpunkten an. Betrachten wir als Beispiel die temporalen Bedingungen aus Abbildung 3. Dort wurden die erweiterten Allen-Relationen *GITARRE* $before_V(2,0)$ *SPRACHE* und *SPRACHE* $overlaps_{V_s}(3,0,5)$ *SCHRITTE* benutzt. Die Präsentationsdauern der beteiligten Audios legen wir wie folgt fest: Gitarre=30, Sprache=15 und Schritte=25 Sekunden. Aus Abbildung 3 ist leicht zu sehen, dass $GITARRE\ before_V(2,0)\ SPRACHE$ den Difference Constraints $(st(SPRACHE) - end(GITARRE) \leq 2) \wedge (end(GITARRE) - st(SPRACHE) \leq -2)$ entspricht. Die Relation $SPRACHE\ overlaps_{V_s}(3.0.5)\ SCHRITTE$ entspricht den Difference Constraints $(st(SPRACHE) - st(SCHRITTE) \leq -3) \wedge (end(SPRACHE) - end(SCHRITTE) \leq -5)$. Die Präsentationsdauern werden auch durch Difference Constraints ausgedrückt. Für das Audioobjekt *GITARRE* wären das folgende Difference Constraints: $(end(GITARRE) - st(GITARRE) \leq 30) \wedge (st(GITARRE) - end(GITARRE) \leq -30)$. Für *SPRACHE* und *SCHRITTE* werden die Difference Constraints in gleicher Weise gebildet. Nach der Transformation der zeitlichen und räumlichen Bedingungen entsteht jeweils eine Menge von Difference Constraints. Diese Menge bezeichnet man auch als *System von Difference Constraints*. Formal ist dieses System wie folgt definiert:

Definition 1 ($\langle V, C\rangle$). ist ein *System von Difference Constraints*. V ist eine Menge von Variablen und C ist eine Menge von linearen Ungleichungen der Form:
$v_i - v_j \leq c_k$ mit $v_i, v_j \in V, c_k = constant, 1 \leq i, j \leq n, 1 \leq k \leq m$
Das System hat m lineare Ungleichungen und n Variablen.

Um eine Datenausgabe entsprechend der Ausgabebedingungen zu ermöglichen, müssen die Ausgabebedingungen mit den entsprechenden Ausgabeobjekten erfüllbar sein. Ein System von Difference Constraints kann als Graph dargestellt und mit dem bekannten *Bellmann-Ford-Algorithmus* gelöst werden [CLR00, RSJM99]. Lösung bedeutet, dass die Start- und Endzeitpunkte für temporale Ausgabeobjekte und die Koordinaten für räumliche Ausgabeobjekte so gewählt werden, dass alle Ausgabebedingungen erfüllt sind. Der *Bellmann-Ford-Algorithmus* ermittelt die minimale Lösung. Diese Lösung hat die kleinstmöglichen zeitlichen bzw. räumlichen Abstände zwischen den beteiligten Ausgabeobjekten. Für die Datenorganisation ist diese Lösung wichtig, da sie das Optimierungsziel für den Datenzugriff ist. Der Algorithmus von *Bellmann und Ford* arbeitet auf einem *Constraint Graphen*. Der Constraint Graph wird entsprechend folgender Definition gebildet:

Definition 2 ($G = \langle V, E\rangle$). ist ein gerichteter und gewichteter *Constraint Graph*. V ist eine Menge von Knoten und E ist eine Menge von gewichteten Kanten:
$V = \{v_0, v_1, \ldots v_n\}$
$E = \{(v_j, v_i) : v_i - v_j \leq c_k, 1 \leq i, j \leq n, 1 \leq k \leq m\} \cup \{(v_0, v_1), \ldots, (v_0, v_n)\}$
Kantengewichte: $w(v_j, v_i) = c_k, j > 0; w(v_0, v_n) = 0, n > 0$

Die Variablen werden zu Knoten in den Graphen und die Konstanten zu Kantengewichten. Wir haben einen zusätzlichen Knoten v_0 eingeführt. Er dient als Startknoten für den

Bellmann-Ford-Algorithmus. Ausgehend von diesen Knoten ermittelt der Algorithmus die kürzesten Wege zu allen anderen Knoten. Stößt der Algorithmus dabei auf einen negativen Zyklus, so ist die Menge der definierten Difference Constraints nicht widerspruchsfrei [CLR00]. Der Algorithmus bricht ab und liefert *FALSE* zurück. Wird kein negativer Zyklus gefunden, so ermittelt der Algorithmus die Werte für die kürzesten Wege und gibt *TRUE* zurück. Der Algorithmus hat eine Komplexität von $O(n \cdot m)$. Die Komplexität hängt also polynomial von der Anzahl der Variablen und der Anzahl der Difference Constraints ab. Abbildung 4 zeigt einen Constraint Graph für die temporalen Ausgabebedingungen, die

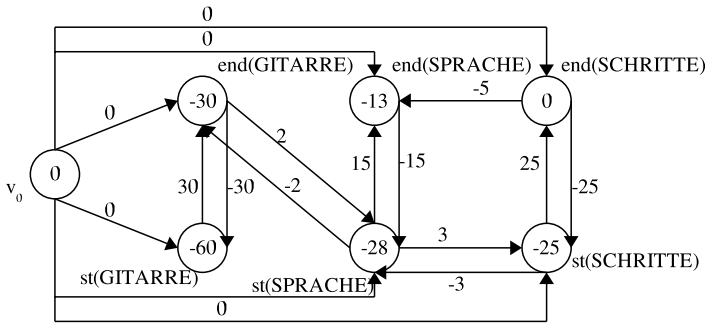

Abbildung 4: Constraint Graph

in Abbildung 3 dargestellt sind. Da der Graph keinen negativen Zyklus enthält, gibt der Bellmann-Ford-Algorithmus *TRUE* zurück und ermittelt die kürzesten Wege von v_0 zu jeden anderen Knoten. Die Werte in den Knoten stehen für die Gewichte dieser kürzesten Wege. Ordnet man diese Gewichte, so erhält man einen Ausgabe-Schedule für die zugehörigen Ausgabeobjekte. Dieser Schedule erfülle alle Ausgabebedingungen.

Der Bellmann-Ford-Algorithmus liefert negative Ergebniswerte. Wenn wir absolute Zeitwerte für den Start und das Ende von Audioobjekten haben wollen, müssen die Resultate von Bellmann-Ford in positive Werte umgewandelt werden. Dazu müssen sie um den Betrag des kleinsten Ergebniswertes verschoben werden. Im Constraint Graph aus Abbildung 4 ist -60 der kleinste Ergebniswert. Um positive Ergebnisse zu erhalten, muss jeder Ergebniswert um 60 verschoben werden. Der Ausgabe-Schedule, der sich aus Abbildung 3 ergibt wäre: st(GITARRE)=0, end(GITARRE)=30, st(SPRACHE)=32, end(SPRACHE)=47, st(SCHRITTE)=35, end(SCHRITTE)=60. Der Ausgabe-Schedule dient als Optimierungsziel der Datenorganisation. Hie könnten z.B. die Audiodaten für *SPRACHE* und *SCHRITTE* auf verschiedenen Platten gespeichert werden, um eine parallele Datenausgabe zu erleichtern.

6 Ausblick: Datenorganisation

Bei der Bereitstellung der Audiodaten für die Renderer kann für komplexe Szenen der Externspeicherzugriff schnell zum Engpass werden. So werden im aktuellen IOSONO-System Audiodaten im WAV-Format mit 24Bit@48kHz verarbeitet. Dies erfordert eine Datenrate von 48.000 Samples$/s \cdot 24$ Bit $= 1125$ KBit$/s \approx 1$ MBit$/s$ pro Audioobjekt. Für eine Szene mit gleichzeitig bis zu 100 Objekten werden demnach schon 100 MBit/s benötigt, womit schon die Grenzen handelsüblicher Festplatten erreicht sind.

Ziel ist es daher, die Renderer mit den benötigten Daten entsprechend der Szenenbeschreibung unter Echtzeitgarantien (d.h. synchron mit dem Video) zu versorgen. Auf technischer Ebene kann dies durch verschiedene Datenbanktechniken erreicht werden:

- *Partitionierung:* Durch Verteilung der Daten auf verschiedene Festplatten, die parallel gelesen werden können, lässt sich die Gesamtransferrate erhöhen.
- *Datentransformation:* Werden die Audioobjekte in komprimierter Form (z.B. im MP3-Format) abgelegt, kann erstens Speicherplatz gespart und zweitens die Anzahl der Festplattenzugriffe reduziert werden. Dies erfordert jedoch eine für die Renderer transparente Transformation (Decodierung).
- *Indexierung:* Gerade bei komprimierten Audiodaten unterstützt eine Indexierung den wahlfreien Zugriff, etwa wenn Sound ab einer bestimmten (Zeit-)Position abgespielt werden soll.
- *Caching:* Bei Schleifen oder der wiederholten Verwendung ein und desselben Objektes kann ein Cache die Anzahl der Plattenzugriffe und den Transformationsaufwand reduzieren.

Allerdings ist dabei zu berücksichtigen, dass die Einsatzumgebung in der Regel Kinos sind. Dies bedeutet, dass sich der Einsatz voll ausgestatteter DBMS verbietet, da zum einen Kosten und Administrationsaufwand viel zu hoch wären und zum anderen viele Features dieser Systeme nicht benötigt werden.

Wir verfolgen den Ansatz eines eingebetteten Datenbanksystems auf der Basis eines Storage Managers wie Berkeley DB. Dieses System soll dabei sowohl als Repository für Szenenbeschreibungen sowie Audioobjekte und deren Metadaten während des Authorings als auch als Storage Manager für das Rendering zum Einsatz kommen. Die „Auslieferung“ einer Audioszene kann dabei über einfache Datenbank-Dumps erfolgen.

Für das Authoring-System wird eine Schnittstelle bereitgestellt, die die Erzeugung und Bearbeitung von Objekten (Audioobjekten, Szenegraphen) sowie einfache Retrievaloperationen über die Metadaten ermöglicht und dabei auch Mehrbenutzerbetrieb zulässt. Für das Rendering besteht die Schnittstelle aus einem Strom mit mehreren Kanälen, entsprechend dem Schedule (Abschnitt 5), der zeitgetaktet ausgelesen werden kann. Die Kanäle repräsentieren hierbei die Audioströme in einem für den Renderer verarbeitbaren Format bzw. Parameterwerte für das Rendering. Jedem Renderer ist dabei ein eigener Strom zugeordnet – Synchronisation bzw. Scheduling werden vom Storage Manager übernommen.

Die entsprechenden Komponenten für einen solchen Storage Manager befinden sich zur

Zeit in der Entwicklung. In der ersten Phase werden dazu nur die Ausgabebedingungen berücksichtigt. Für die zweite Phase sollen auch die oben erwähnten Optimierungen der Datenorganisation einbezogen werden, indem bei der „Installation“ eines Films (d.h. beim Kopieren der Audioszenen von DVD auf die Fesplatte(n)) anhand der Szenenbeschreibungen und der verfügbaren Hardwarekonfiguration eine optimale Verteilung und Indexierung der Daten bestimmt wird.

Literatur

[All83] J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[ASS00] S. Adali, M.L. Sapino und V.S. Subrahmanian. An algebra for creating and querying multimedia presentations. *Multimedia Systems*, 8(3):212–230, 2000.

[BF98] E. Bertino und E. Ferrari. Temporal Synchronization Models for Multimedia Data. *TKDE*, 10(4):612–631, 1998.

[BV94] M. Boone und E. Verheijen. The Wave Field Synthesis Concept Applied to Sound Reproduction. *AES Convention Paper presented at the 96th AES Convention Februar 1994, Amsterdam*, 1994.

[CLR00] T. H. Cormen, C. E. Leiserson und R. L. Rivest. *Introduction to Algorithms*. Cambridge, Massachusetts: The MIT Press, 2000.

[Hei04a] T. Heimrich. Output Constraints in Multimedia Database Systems. In *4th Int. Workshop on Multimedia Data and Document Engeniering (MDDE icw CVPR)*. IEEE, 2004. Washington D.C., USA.

[Hei04b] T. Heimrich. An Output Schema for Multimedia Data in Multimedia Database Systems. In *6th. Int. Baltic Conference on Databases and Information Systems*, Seiten 125–134, 2004. Riga, Latvia.

[Hei05] T. Heimrich. Modeling Output Constraints in Multimedia Database Systems. In *11th. Int. Multi-Media Modelling Conference*. IEEE, 2005. Melbourne, to appear.

[IM99] M. Vazirgiannis I. Mirbel, B. Pernici. Integrity constraints for interactive multimedia scenarios. In *IEEE Multimedia*, 1999.

[MPS+00] I. Mirbel, B. Pernici, T.K. Sellis, S. Tserkezoglou und M. Vazirgiannis. Checking the Temporal Integrity of Interactive Multimedia Documents. *VLDB Journal: Very Large Data Bases*, 9(2):111–130, 2000.

[MRB+03] F. Melchior, T. Röder, S. Brix, S. Wabnik und C. Riegel. Authoring System for Wave Field Synthesis. *AES Convention Paper presented at the 115th Convention 2003 October 10, New York*, 2003.

[RSJM99] G. Ramalingam, J. Song, L. Joskowicz und R.E. Miller. Solving Systems of Difference Constraints Incrementally. *Algorithmica*, 23(3):261–275, 1999.

[VK86] M. B. Vilain und H. A. Kautz. Constraint propagation algorithms for temporal reasoning. In *Fifth National Conference on Artificial Intelligence*, Seiten 377–382, 1986.