# Zeitvariante Beschreibung virtueller Szenen für die Echtzeit-Auralisierung instationärer Schallfelder

Frank Wefers, Michael Vorländer

*Institut für Technische Akustik, RWTH Aachen*
*Kopernikusstraße 5, 52074 Aachen*
*{fwe, mvo}@akustik.rwth-aachen.de*

## Zusammenfassung

In diesem Beitrag wird eine neuartige Baum-basierte Szenenbeschreibung für akustische virtuelle Szenen vorgestellt. Die Datenstruktur beschreibt die Objekte der Szene und ihre Parameter entlang einer zeitlichen Achse. Hierbei darf die Eingabe und Ausgabe der Daten die Echtzeitverarbeitung nicht negativ beeinflussen (z.B. Prioritätsinversion). Dies wird durch die Unterbindung von wechselseitigem Lese-/ Schreibzugriff sowie nicht-blockierenden Datenstrukturen sichergestellt. Durch Wiederverwendung wird die Speicherplatzeffizienz erhöht. Die Schnittstellen für darauf aufsetzende Algorithmen (Bewegungsmodelle, Simulationsverfahren) werden erläutert. Am Beispiel einer Freifeld- und Raumakustik-Simulation wird die Integration in die Echtzeit-Auralisierung vorgestellt.

## Einleitung

Im Sinne einer physikalisch-basierten Auralisierung muss zwischen den Zeitpunkten der Schallabstrahlung an der Quelle, des Schalleinfalls beim Hörer sowie der dazwischenliegenden Ausbreitungsdauer im Medium unterschieden werden. Bei den meisten gegenwärtigen Auralisierungskonzepten ist dies jedoch nicht der Fall. Bei raumakustischen Simulationen ist es meist hinreichend die virtuelle Szene vereinfacht nur im Momentanzustand zu beschreiben (die Ausbreitungsdauern sind gering). Bei Außenszenarien (z.B. Verkehrslärm, Fluglärm) kann diese Vereinfachung aber bereits dazu führen, dass die Simulation dem Benutzer teils nicht mehr plausibel erscheint. Anwendungen wie diese stellen die Auralisierung vor eine neue Herausforderung: die Parameter der Objekte in einer virtuellen Szene (Positionen, Orientierungen, Schalldruckpegel, Richtwirkungen der Objekte sowie Mediumszustand, usw.) entlang einer zeitlichen Historie zu speichern. Aus dieser Beschreibung müssen die effektiven Ausbreitungsdauern bestimmt und in der Schallfeldsimulation berücksichtigt werden [1].

## Anforderungen

Die akustische virtuelle Szene wird von verschiedenen Instanzen verändert (Schreibzugriff) und abgefragt (Lesezugriff). Abbildung 1 gibt einen Überblick hierüber. Diese Zugriffe erfolgen in aller Regel nebenläufig und daher muss der Zugriff auf die Szenendatenstruktur entsprechend programmiertechnisch synchronisiert werden.

Die Eingabeseite beinhaltet den Benutzer selbst als Akteur, sowie verschiedene Eingabegeräte. Benutzereingaben (Navigation, Selektion, Modifikation) sind üblicherweise niederfrequent und geschehen meist sporadisch. Motion Tracker und andere Eingabegeräte (z.B. 3D-Zeigegeräte, Gamepads, u.A.) kennzeichnen sich durch hochfrequente, kontinuierliche Veränderungen der virtuellen Szene (z.B. Kopfposition des Hörers abgetastet mit 120 Hz). Schreibzugriffe auf die Szene sollen mit geringem Aufwand und kurzfristig möglich sein. Geringfügige Verzögerungen oder Wartezeiten (einige 10 µs) sind allerdings tolerierbar.

Ausgabeseitig ist an die Datenstruktur direkt die Echtzeit Audioverarbeitung angekoppelt. Dies bedeutet, dass Szenenparameter innerhalb zeitkritischer, hochpriorisierter *callbacks* bzw. *interrupts* ausgelesen werden müssen (innerhalb weniger Millisekunden Rechenzeit). Ein Lesezugriff innerhalb dieses Ausführungspfades ist in der Regel nicht vermeidbar, da effektive Ausbreitungsdauern für eine hochqualitative Auralisierung quasi-kontinuierlich (d.h. pro Abtastwert) berechnet und eingestellt werden müssen. Hierzu werden die Trajektorien beweglicher Szeneobjekte (Quellen, Hörer) mittels Bewegungsmodellen (z.B. exponential smoothing [2], Kalman-Filter [3]) prädiziert bzw. interpoliert. Diese Modelle erfordern Zugriff auf die Positions- und Orientierungsdaten entlang der zeitlichen Historie (motion samples). Das Auslesen von Szenedaten hat daher eine übergeordnete Priorität. Es darf nicht durch das Schreiben auf der Datenstruktur blockiert werden (Prioritätsinversion). Die Kernanforderung ist ein ungehinderter, direkter Lesezugriff unter Vermeidung jeglicher blockierender Synchronisationsmechanismen (wie Threadbedingungen, Mutexe, Semaphoren, usw.).

## Zustände

Grundlegend für die Datenstruktur ist die Zusammensetzung aus Zuständen (*states*). Ein Zustand beschreibt ein einzelnes Datum oder mehrere zusammengesetzte Objekte in einem Zeitpunkt. Die Basis bilden hierbei elementare Zustände, wie Skalare, Vektoren, Matrizen, aber auch Wahrheitswerte oder Bitvektoren. Diese Grundbausteine werden zu höherwertigen Datenstrukturen zusammengefügt. Alle Zustände, egal ob elementar oder zusammengesetzt, können ihrerseits beliebig von anderen Zuständen referenziert werden (*pointers*). Hierdurch werden keine Daten kopiert und die Darstellung der Szene bleibt in ihrem Speicherbedarf kompakt. Ein speziellen Datentyp stellen Container dar (Arrays, verkettete Listen, Hashes, usw.). Diese enthalten eine Anzahl von Referenzen auf andere Zustände.

Die Konstruktion wird nun an einem Beispiel erläutert. Der in der Hierarchie ranghöchste Zustandstyp ist die Szene selbst. Sie ist zusammengesetzt aus Containern für die virtuellen Schallquellen und Hörer. Zusätzlich können Felder für Mediumszustände oder die Szenengeometrie angefügt werden. Jede Quelle und jeder Hörer wird durch einen eigenen Zustand beschrieben. Diese Zustände sind beispielsweise zusammengesetzt aus Feldern, wie der Position (3-Element Vektor), der Orientierung (Quarterion), dem Schalldruckpegel bzw. der Schallleistung (Skalar), einer Richtcharakteristik (Referenz), sowie weiteren Attributen. Abbildung 2 illustriert diese Zusammenhänge am Beispiel der Schallquellen in einer virtuellen Szene.

## Modifikation

Alle Zustände unterliegen einem festen Lebenszyklus: Sie werden durch Ableitung eines dedizierten Vorgänger-zustandes erzeugt (Rückwärtsreferenz jeweils bekannt). Danach dürfen sie beliebig modifiziert werden. Ist die Bearbeitung abgeschlossen werden sie fixiert und sind ab diesem Zeitpunkt dann nur noch lesbar (*read-only*). Ein Ursprungszustand darf beliebig oft abgeleitet werden (Baum-Struktur). Die genauen Arbeitsschritte werden erneut an einem Beispiel der Positionsänderung einer Schallquelle erläutert:

- Zunächst wird der Momentanzustand der Szene (*head state*) **H** ermittelt.
- Im enthaltenen Containerzustand **C** für Schallquellen wird der betreffende Schallquellenzustand **S** gesucht. Diese Suche geschieht durch *hash maps* in $\mathcal{O}(1)$.
- Aus dem ermittelten Zustand **S** wird ein neuer Schallquellenzustand **S'** abgeleitet, wobei alle Werte und Referenzen von **S** kopiert werden.
- Der vorherige Positionszustand **P** in **S** wird abgeleitet in einen neuen Zustand **P'**.
- **P'** bekommt die neue Position gesetzt und wird anschließend fixiert.
- **S'** bekommt **P'** zugeordnet und wird fixiert.
- **C'** wird aus **C** abgeleitet und bekommt den aktualisierten Zustand **S'** für die Schallquelle gesetzt.
- Letztlich wird ein neuer Szenezustand **H'** aus **H** abgeleitet, welcher **C'** als Schallquellenliste enthält.

Üblicherweise geschehen mehrere Änderungen an einer Szene parallel. Auf einfache Weise können somit mehrere Operationen zeitlich exakt synchronisiert werden.

## Aktualisierung

Die Auralisierung selbst bekommt jeweils den aktuellen Szenenzustand mitgeteilt. Ihre Aufgabe ist es, diesen Zustand dann möglichst schnell zu reproduzieren (geringe Latenz). Hierzu muss der vorherige Zustand mit dem neuen Zustand verglichen werden, um die Änderung zu detektieren und daraus die notwendigen Aktualisierungen herzuleiten. Diese Vergleiche müssen ebenfalls möglichst kostengünstig durchzuführen sein und sollen keine unnötigen Rechenschritte erfordern. Um dem gerecht zu werden, wird im vorgestellten Ansatz jeder Zustand (unabhängig seines Typs) mittels eines 64-Bit Ganzzahlwertes (Schlüssels, *ID*) identifiziert. Prinzipiell werden nur fixierte Szenezustände der Auralisierung übergeben. Dies impliziert, dass auch alle untergeordneten Subzustände fixiert sind.
Diese Konvention erlaubt dann den kostengünstigen Vergleich zweier Zustände durch einfachen Vergleich ihrer Schlüssel (*IDs*). Sind diese identisch, müssen folglich auch die Daten identisch sein (inklusive aller untergeordneten Felder). Umgekehrt können aber zwei Zustände unterschiedlicher Schlüssel durchaus gleiche Werte enthalten. Dies kann gegebenenfalls zusätzlich berücksichtigt werden. Der effektive Vergleich der Szenenzustände resultiert letztlich in einer Reihe einfacher Vergleichsoperationen.

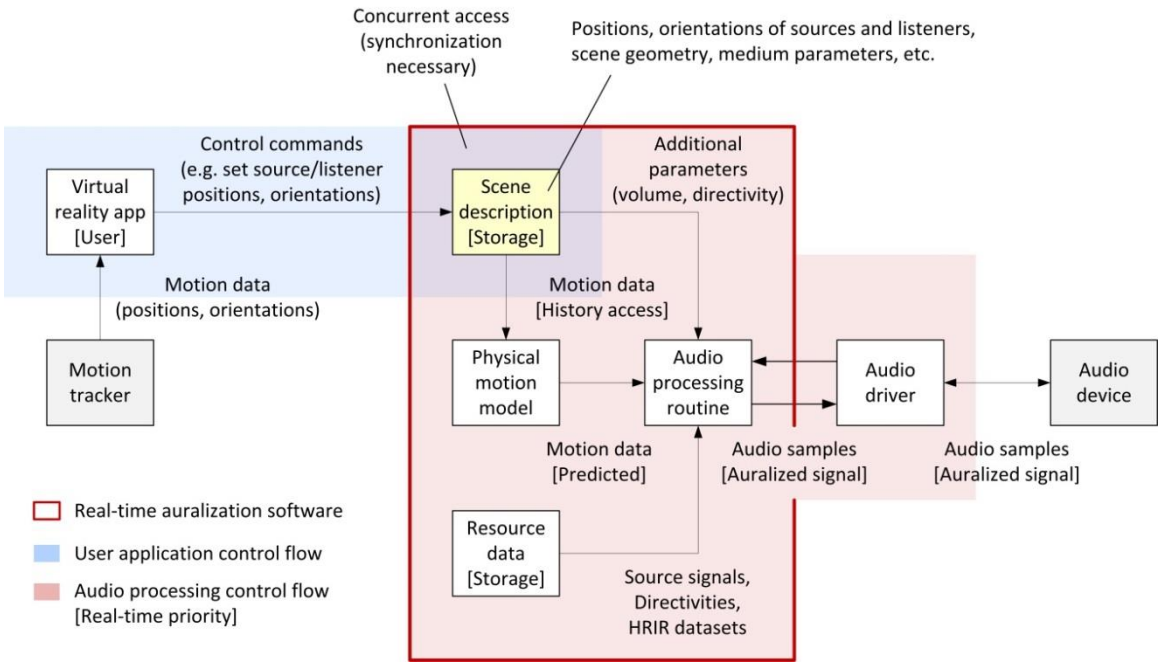


**Abbildung 1:** Verschiedene Akteure und Komponenten eines Computersystems zur Echtzeit-Auralisierung

## Synchronisation

Das Konzept, einmalig fixierte Daten nicht mehr zu verändern, erlaubt eine elegante Synchronisierung zwischen den Schreibern und Lesern auf der Datenstruktur. Das Übergabedatum (neuer Szenenzustand) besteht letztlich aus einem einzigen Schlüssel. Dieser kann mittels eines atomaren Ganzzahlwertes realisiert werden (*atomic read/write*). Mehrere Leser müssen nicht untereinander synchronisiert werden. Gleichzeitiges Modifizieren der Szene bedarf nicht zwangsweise einer Synchronisation. Nach einer Änderung muss lediglich die Referenz auf den Momentanzustand aktualisiert werden. Dies geschieht ebenso durch atomare Schreibzugriffe. In der Regel empfiehlt es sich, zumindest die nutzerseitigen Änderungen zu gruppieren (exklusives Modifikationsrecht).

## Allozierung/Deallozierung

Eine besondere Herausforderung in Echtzeit-Systemen ist die Realisierung von asynchronen Allozierungs- und Deallozierungsoperationen. Bei der konkreten Problem-stellung bedeutet dies, dass im Kontext des Anwenders neue Szenenzustände abgeleitet werden müssen (*Allokation*), wohingegen die Entscheidung über die Freigabe (*Deallokation*) von Zuständen durchaus im Kontext der Audio-Verarbeitung (*callback*) gefällt wird. Dies betrifft beispielweise alte Positionsdaten, welche nicht mehr seitens der Bewegungsmodelle benötigt werden. Deallokationen sind hier zu vermeiden, um den Ablauf der Audioverarbeitung nicht unnötig zu stören.

Als Lösung wird das Konzept der *Pools* aufgegriffen. Ein Pool verwaltet mehrere Objekte, welche unbenutzt oder vergeben sein können. Alle Objekte sind mit atomaren Referenzzählern versehen. Die elementare Operation Anforderung (*request*) liefert ein freies Objekt zurück. Verwendungen des Objektes werden durch Inkrementierung des Zählers beschrieben. Objekte verfügen über eine Rückwärtsreferenz zu ihrem übergeordneten Pool. Innerhalb der zeitkritischen Audioverarbeitungen müssen daher nur Referenzen entfernt werden (atomares Dekrement). Der Pool sucht bei der nächsten Anfrage in den von ihm verwalteten Objekten nach einem freien Element. Da diese Anfragen auf dem Kontext der Anwendung gestellt werden (siehe oben) wird eine Prioritätsinversion vermieden und die Deallozierung in den unkritischen Kontext überführt. Ferner ermöglichen Pools eine direkte Wiederverwendung der Datenbereiche.

## Fazit

Die Echtzeit-Auralisierung zeitveränderlicher virtueller Szenen mit bewegten Quellen und Hörern erfordert weiterführende Datenstrukturen, die über bisherige Ansätze hinausgehen. Eine einfache Beschreibung der Szene im Momentanzustand ist hier nicht mehr hinreichend. Die unterschiedlichen Zeitpunkte der Schallabstrahlung seitens der Quellen sowie Schallwahrnehmung durch Hörer, müssen getrennt berücksichtigt werden. Die Parameter einer Szene müssen entlang einer zeitlichen Historie beschrieben werden. Die Echtzeit-Anforderungen erfordern den Einsatz spezieller Datenstrukturen, um eine reibungslose Audioverarbeitung zu gewährleisten.

Es wurde eine neuartige Datenstruktur zur Beschreibung virtueller Szenen vorgestellt, welche diesen Anforderungen Rechnung trägt. Das Grundkonzept sind Zustände, welche kombiniert und abgeleitet werden können. Synchronisation zwischen mehreren Schreibern und Lesers wird vermieden durch das Konzept Zustände zu fixieren (*read only*). Zustände können auf einfache Weise anhand ihrer Schlüssel verglichen werden. Die schnelle Änderung einer Szene wird durch die Verwendung von Pools unterstützt. Dies erlaubt ein hohes Maß der Wiederverwendung von Speicherbereichen und vermeidet ungünstige Operationen innerhalb der zeitkritischen Audioverarbeitung.

## Danksagung

Die Autoren danken der Deutschen Forschungsgemeinschaft für die Förderung des Projekts „Echtzeit-Auralisierung instationärer Schallfelder mit bewegten Quellen und Hörern“.

## Referenzen

[1] H. Strauss, “*Implementing Doppler Shifts for Virtual Auditory Environments*”, 104th Convention of the Audio Engineering Society (AES), Amsterdam, 1998.

[2] J. J. LaViola, “Double exponential smoothing: *An alternative to Kalman filter-based predictive tracking*”, Workshop on Virtual environments (EGVE '03), Zürich, 2003.

[3] A. Kiruluta, M. Eizenman und S. Pasupathy. „*Predictive head movement tracking using a Kalman filter”,* IEEE Transactions on Systems, Man and Cybernetics,Vol. 27, No. 2, 1997.

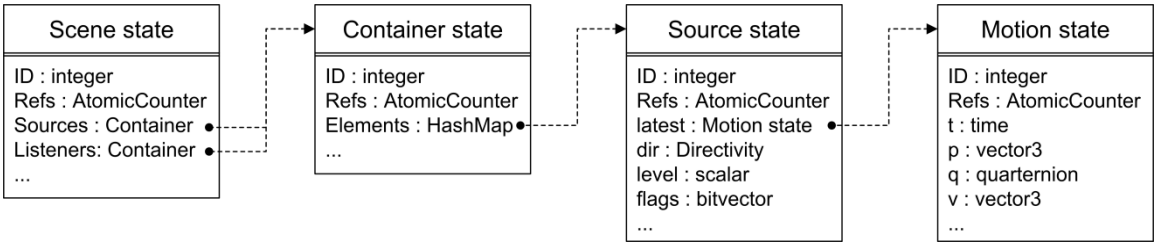


**Abbildung 2:** Zusammensetzung von Zuständen am Beispiel der Schallquellen in einer virtuellen Szene