Chapter 2

Sound Spatialization Management

In a virtual reality environment, users are immersed in a scene with objects which might produce sound. The responsibility of a VR environment is to present these objects, but a practical system has only limited resources, including spatialization channels (mixels), MIDI/audio channels, and processing power. A sound spatialization resource manager controls sound resources and optimizes fidelity (presence) under given conditions. For that, a priority scheme based on psychoacoustics is needed. Parameters for spatialization priorities include intensity calculated from volume and distance, orientation in the case of non-uniform radiation patterns, occluding objects, frequency spectra (low frequencies are harder to localize), expected activity, and others. Objects which are spatially close together (depending on distance and direction) can be mixed. Sources that can not be spatialized separately can be mixed as ambient sources. Important for resource management is the resource assignment, i.e., minimizing swap operations, which makes it desirable to look-ahead and predict upcoming events in a scene. Prediction is achieved by monitoring objects' position, speed, and past evaluation values (i.e., priorities, probabilities, ...). Fidelity is contrasted for different kind of resource restrictions and optimal resource assignment.

To give standard and comparable results, the VRML97 specification [Bell *et al.*, 1997] is used as an application programmer interface. Applicability is demonstrated with a helical keyboard [Herder and Cohen, 1996], a polyphonic MIDI stream driven animation including user interaction (a user may move around, playing together with programmed notes). The developed sound spatialization resource manager gives improved spatialization fidelity under runtime constraints. Application programmers and virtual reality scene designers are freed from the burden of assigning mixels and predicting the sound

sources locations.

Spatial sound used in virtual reality environments fulfills different purposes. As passive feedback [Burdea and Coiffet, 1994, pp. 234-236] it enhance the realism of the display and informs the user about scene changes. In case of active feedback, the sound is directly coupled to user interaction (e.g., use of a manipulator, movement). Information presented include the position, orientation, movement, and speed of objects and the user in the scene. Besides that, room acoustic helps to understand the scene and get a feeling for space, walls, and materials. In this chapter, we mostly neglect room acoustics and focus on direct sound, which is important for localization of objects in space.

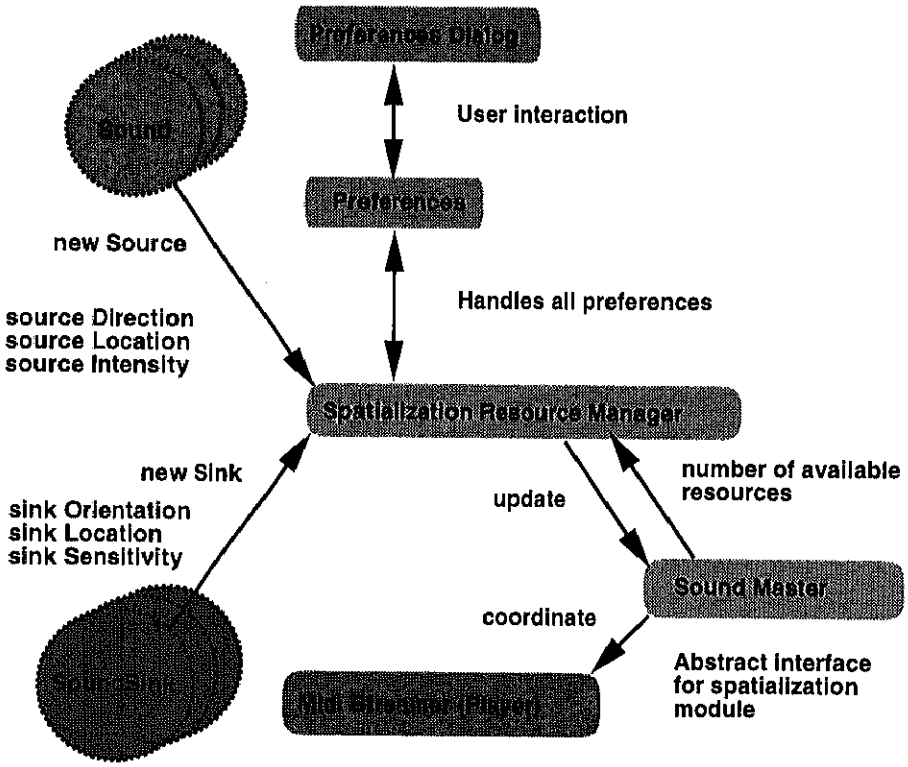

Figure 2.1: System schematic

Figure 2.1 shows a system schematic. We applied and tested the spatialization resource manager with two applications. In the Helical Keyboard project, shown in Figure 1.4 [Herder and Cohen, 1996], keys are objects in space which are activated by a MIDI stream. The number of requested simultaneous spatialized sources is defined by the polyphony of the song.

In the case of multiple sinks (see Section 2.2.3), the quantity of required resources (i.e., mixels) increases with the number of sinks. Each sink defines its own space, which is then is folded together with its siblings'. In a second application developed to show the capabilities of the resource management, objects follow a motion test pattern of distribution functions.

2.1 Requirements

Besides to the requirements given in the introduction (see page 13), the requirements for the algorithm introduced in this chapter are extended to:

- support for multiple sinks,
- using the API defined in Chapter 5, giving the application programmer easy control over resource assignment process,
- dynamic resource allocation/control, and
- low computation costs.

2.2 Resource management

Resource management can be static, in which the resource assignment is predefined by the VR scene designer, or dynamic, which means the mapping from source to channel is established at runtime. In systems in which the number of users and their sound spatialization requests are not predefined and the number of resources are limited, dynamic assignment of the resources will allow maximum system use.

What are sound spatialization resources? "Mixels,"— acronymic for '[sound] mixing elements,' in analogy to pixels, taxels (tactile elements), texels (texture elements), or voxels (a.k.a. boxels)— since they form the raster across which a soundscape is projected, define the granularity of control and degree of spatial polyphony.

Input (monaural) audio channels are associated with sources in the virtual space. Dynamic resource allocation assigns the source↔sink mappings to mixels, whose number is determined by the breadth of the directionalizing backend.

2.2.1 Strategies

A spatialization resource manager must decide how input channels are mapped to the resources. Figure 2.2 shows the principal task. Out of all relevant sources, sources for spatialization are selected. Our implementation is based mainly on the application programmer interface given by VRML 2.0 (see also Section 5).

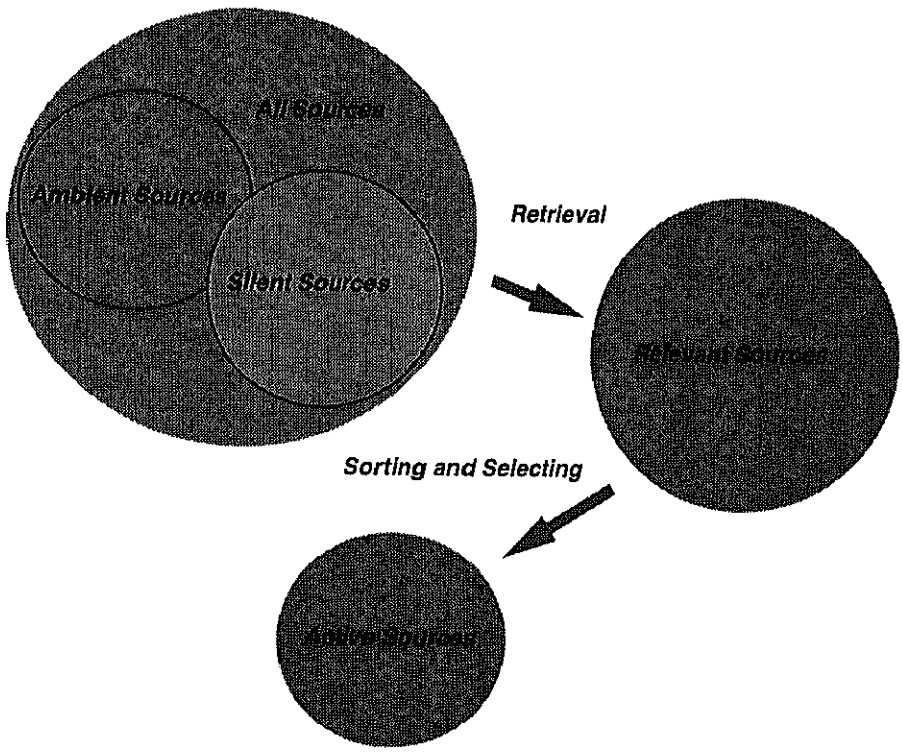

Figure 2.2: Source sets for spatialization

A simple algorithm would prioritize the relevant sources, assigning them to resources starting with the highest priority until no more resources were available. This operation might involve preemption of a source, which would be swapped out of the set of active sources. The following sections develop a more detailed and sophisticated approach.

Filtering relevant resources

Filtering Algorithm 1 is given in pseudo-code.

For simplicity the algorithm assumes an ellipsoidal radiation pattern of the sound sources and a spherical sensitivity pattern for sound sinks. The

Algorithm 1 Simple filtering algorithm

```
for each sink in sinks do
  if sink is enabled then
    for each source in sources do
      if source is ambient then
        if source is background then
          add source to background set of sink
        else
          add source to ambient set of sink
        end if
      else
        if source is low frequency then
          add source to ambient set of sink
        else
          distance ⇐ distance(source,sink)
          if ( not(sourceInSinkFarRange(source,sink) and
               sinkInSourceAudibleRange(source,sink)) ) then
            add source to inactive set of sink
          else
            volume ⇐ f(intensity, sensitivity, distance)
            if ( sourceInSinkNearRange(source,sink) and
                 sinkInSourceCoreRange(source,sink) ) then
              volume ⇐ 1
            end if
            if volume < minVolume then
              add source to inactive set of sink
            else
              add source to active set of sink
            end if
          end if
        end if
      end if
    end for
  end if
end for
```

domain	trigger	operation	cost
multiprogramming/multi-tasking CPU	preemption, interrupt (OS timeslice)	swap out jobs (linked list of processes)	thrashing
virtual memory, caching RAM	page fault	swap out pages to disk	time
audio resource management	"interrupt" or reprioritization	swap out mixel to ambient space & spatialize new mixel	"fidelity," soundscape stability

Table 2.1: Resource management

function `f` defines the attenuation of sound in the medium and is used in Algoritm 1 and 2 for setting the volume value:

$$\texttt{f}(intensity, sensitivity, distance) = \frac{intensity \times sensitivity}{distance^2} \tag{2.1}$$

Intensity and sensitivity are linearized and normalized gain values of source and sink, respectively.[1]

The boolean function `sourceInSinkFarRange` returns True if the source is within the far range sphere of the sink. Sources which are outside of this sphere are not audible to the specified sink. The field `farDistance` is field of sink controlled by an application.

$$\begin{aligned}&\texttt{sourceInSinkFarRange}(source, sink) = \\ &\quad distance(source, sink) < \texttt{farDistance}\end{aligned} \tag{2.2}$$

The boolean function `sourceInSinkNearRange` returns True if the source is within the near range sphere of the sink. Sources which are outside of this sphere are attenuated or not audible to the specified sink. The field `nearDistance` is field of sink controlled by an application.

$$\begin{aligned}&\texttt{sourceInSinkNearRange}(source, sink) = \\ &\quad distance(source, sink) < \texttt{nearDistance}\end{aligned} \tag{2.3}$$

Equation 2.4 defines when a sink is in the audible range of a source. The fields `maxBack`, `maxFront`, `location`, and `direction` specify an audible

[1] In this notation, "intensity" is not the physically defined term of acoustical power per unit area.

ellipsoid. If the sink is not inside of the ellipsoid, then the source is not audible by the sink. An ellipsoid is given by two focal points $f1$ and $f2$. Assume $f1$ is the location of the source node, then $f2$ can be calculated by $f1 + \texttt{direction}/|\texttt{direction}| * (\texttt{maxFront} - \texttt{maxBack})$. Let ls be the sink location, then the source is audible if $|ls - f1| + |ls - f2| \leq \texttt{maxBack} + \texttt{maxFront}$.

$$\begin{aligned}
&\texttt{sinkInSourceAudibleRange}(source, sink) = \\
&\quad distance(sink,source) + \\
&\quad distance(sink,location(source) + \\
&\qquad\qquad \texttt{direction}/length(\texttt{direction}) * \\
&\qquad\qquad (\texttt{maxFront} - \texttt{maxBack})) \\
&\quad < \texttt{maxBack} + \texttt{maxFront}
\end{aligned} \tag{2.4}$$

A sink is in the core range of a source is defined in Equation 2.5 in the same way.

$$\begin{aligned}
&\texttt{sinkInSourceCoreRange}(source, sink) = \\
&\quad distance(sink,source) + \\
&\quad distance(sink,location(source) + \\
&\qquad\qquad \texttt{direction}/length(\texttt{direction}) * \\
&\qquad\qquad (\texttt{minFront} - \texttt{minBack})) \\
&\quad < \texttt{minBack} + \texttt{minFront}
\end{aligned} \tag{2.5}$$

The values `minFront`, `maxFront`, `minBack`, and `maxBack` are attributes of the source, as defined by the API in Chapter 5, conforming to the Sound node definition [Carey and Bell, 1997, p. 277–284]; values `farDistance` and `nearDistance` are associated with the sink. Figure 2.3 shows the different ranges for sound source and sink as defined in Section 5.2 and Section 5.3. Source frequency is calculated by source channel, which in the case of MIDI is the normalized note number.

Sorting

For all active sources, as determined by techniques in this section, priority is calculated with Algorithm 2 based on volume, source and sink priority.

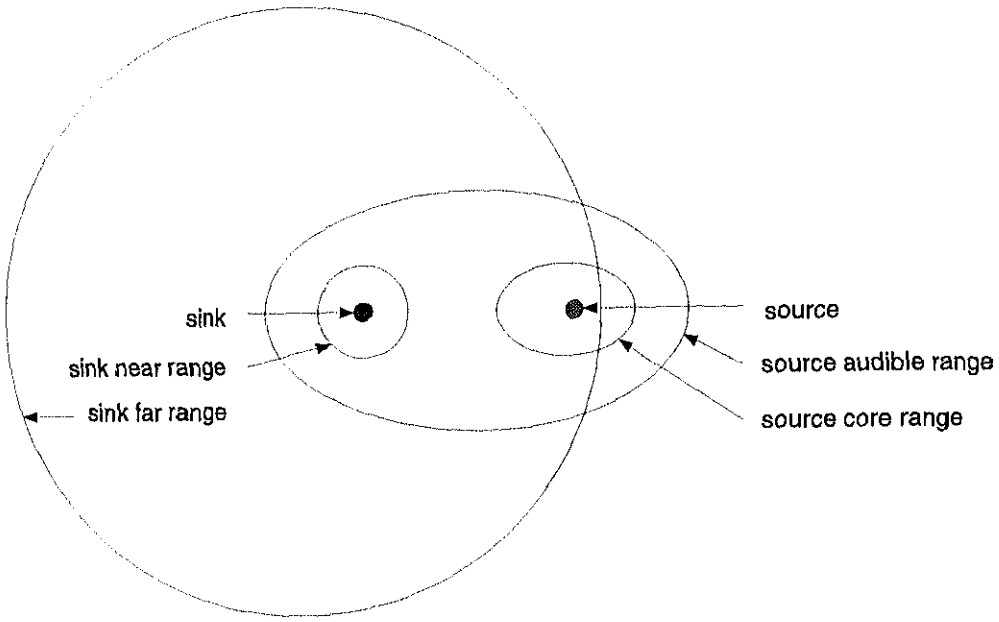

Figure 2.3: Audible and intensity ranges for sound source and sound sink

Algorithm 2 Simple algorithm to calculate source processing priority

```
for each sink in sinks do
  if sink is enabled then
    for each source in active sources attended by sink do
      distance ⇐ distance(source,sink)
      if ( sourceInNearRange(source,sink) and
           sinkInSourceCoreRange(source,sink) ) then
        volume ⇐ 1
      else
        volume ⇐ f(intensity, sensitivity, distance)
      end if
      source processing priority ⇐ volume * (sink priority + 1) * (source
      priority + 1)
    end for
  end if
end for
```

For efficiency, this calculation can be done together with filtering of relevant source. Sorting the set for priority and using the best for spatialization would already give a good strategy for resource allocation, but in the next section we optimize it further.

2.2.2 Reservation scheme

Resource reservation is available on three different levels. An application might reserve resources in advance via the API. In a spatialization server environment, the server might reserve resources. The spatialization resource manager reserves resources for sources which are currently not active but have shown in the past large values in probability in moving and high priorities. The past priorities (maximum and average) are set in contrast to the current active sources.

2.2.3 Multiple sinks

The spatialization resource manager must also consider multiple sinks [Cohen, 1995] [Herder and Cohen, 1996] which idiom was developed partly to address the issue of soundscape control in an shared context, allowing users to redirectionalize multiple sources without moving them (which might disturb other users) by installing and adjusting colocated sinks. An arbitrary number of users might experience a relaxed common view of a conference room or concert hall, each designating (possibly shared) sinks and using a personal gain adjustment profile, individually tuned for hearing acuity and control/display characteristics.

Roles of sources and sinks A classification of sound sources and sinks is given in Table 1.1. Multiple sinks allow forked presence in auditory space. This is being like being in more than one place at once, a concept familiar from teleconferencing and studio recording. Two methods have been suggested for disambiguating the paradoxes of multiple presence. One is to partition the sources across the sinks, in which case the required mixels number is the same as a virtual space with a single sink. A second method crosses all sources and sinks, in which case required mixels number is the product.

2.3 Optimal sound spatialization manager

A spatialization module has a limited number of channels. An optimal spatialization resource manager would assign the channels to minimize the dif-

ference between a configuration with limited number of resources and an ideal one with unlimited resources.

2.4 Implementation

Our prototype was developed on an SGI Indigo 2 Extreme, connected with an Acoustetron II from Aureal/Crystal River Engineering and Roland Sound Module. The Open Inventor graphics toolkit was expanded for classes (nodes) to support the spatial sound extensions, which were used for our virtual reality applications. Open Inventor is a superset of the VRML 1.0 standard [Bell *et al.*, 1995], which does not support sound or dynamic behavior of objects.

Bibliography

[Bell *et al.*, 1995] Gavin Bell, Anthony Parisi, and Mark Pesce. The Virtual Reality Modeling Language, Version 1.0 Specification, May 1995. `http://www.vrml.org/Specifications/VRML1.0/`.

[Bell *et al.*, 1997] Gavin Bell, Rikk Carey, and Chris Marrin. ISO/IEC 14772-1:1997: The Virtual Reality Modeling Language (VRML97), 1997. `http://www.vrml.org/Specifications/VRML97/`.

[Burdea and Coiffet, 1994] Grigore Burdea and Philippe Coiffet. *Virtual reality technology.* Hermes, 1994. ISBN 0-471-08632-0.

[Carey and Bell, 1997] Rick Carey and Gavin Bell. *The Annotated* VRML *2.0 Reference Manual.* Addison-Wesley Developers Press, 1997. ISBN 0-201-41974-2.

[Cohen, 1995] Michael Cohen. Besides immersion: Overlaid points of view and frames of reference; using audio windows to analyze audio scenes. In ICAT/VRST: *Int. Conf. Artificial Reality and Tele-Existence/Conf. on Virtual Reality Software and Technology*, pages 29–38, Makuhari, Chiba; Japan, November 1995.

[Herder and Cohen, 1996] Jens Herder and Michael Cohen. Design of a Helical Keyboard. In Steven P. Frysinger and Gregory Kramer, editors, ICAD *'96 — Int. Conf. on Auditory Display*, Palo Alto, CA; USA, November 1996.