A Visual User Interface For Creation And Manipulation Of Auditory Scenes

Alireza Darvishi
University of Zurich
ETH Zentrum
Clausiusstrasse 59
CH - 8092 Zurich
Phone: +41 (01) 632 77 95
Fax: +41 (01) 363 00 35
E-mail: darvishi@inf.ethz.ch

ABSTRACT
This paper describes a software prototype which allows visual composition and manipulation of everyday sounds based on a suggested concept called "Auditory Scenes" [3]. The description of user interface components in the Java software prototype is the main topic of this paper. The concept of "Auditory Scenes" assumes various perceptual attributes for each individual sound in the scene as well as the temporal, spatial and other relationships between them. An introduction to the suggested concept "Auditory Scenes" is given first. Various components of the implemented user interface and a conclusion are discussed in the following section.

Keywords: Everyday sound composition and manipulation, User Interface, Visual programming, Java, Sound Synchronisation

INTRODUCTION
This section gives an overview of the suggested concept "Auditory Scenes", two grammar approaches based on this concept as well as a short description of the developed software prototype. Note that the concept of "Auditory Scenes" does not consider "Auditory Scene Analysis" [3], which describes the ability of listeners to separate the acoustic events arriving from different environmental sources into separate perceptual representations (streams).

The concept of Auditory Scenes relies on our everyday perception of sounds in daily life. It assumes various perceptual attributes such as duration, volume, position in space, and others, for each individual sound in the scene as well as the temporal, spatial and other relationships between them.

An Auditory Scene is not an empty subset of G x T x T x V x E^3 where G is the set of sounds, T is the set of points in time (start, finish), V is the set of values for volume and E^3 is the set of co-ordinates for each individual sound.

The two developed grammars based on the above concept represent (1.) the hierarchical and (2.) the autonomous concept. The first grammar approach is similar to music composition and is basically a temporal composition of everyday sounds. This approach allows less interaction and is suitable for applications such as radio play productions. By contrast, the second grammar approach defines the event driven non-hierarchical composition of everyday sounds. [5] This approach is more appropriate for interactive environments with external events such as user input (mouse, keyboard, head turn in a VR system, etc.). The semantic of both Grammars is specified by using the method of algebraic specification. [5]

The two grammar approaches and their algebraic specification allow high abstract and universal description of auditory scenes, which can easily be implemented on various platforms (portability). The grammar represents the syntax while the algebraic specification represents the semantic of the language. [2]

The developed software prototype implements a subset of the above hierarchical approach. Different other temporal relationships such as "AFTER", "DURING" and others are foreseen in the grammar [4] but not implemented in the prototype. In this software prototype an auditory scene is described by a hierarchy of four sound types:

* Sample Sound: Is a sampled sound which is either recorded or synthesised.
* Serial Sound: This composite sound type allows sounds to be played sequentially. The length of this composite sound type is the sum of the duration of all participating sounds in the composite.

* Start Parallel: All the element in the composite are played simultaneously. Each element will be played as long as its duration. The duration of the this composite sound type is the length of the longest duration of participating sounds.

* Master Parallel: All the participating sounds will be played simultaneously. The duration of this composite sound type is the duration of the first defined sound.

The last three sound types are so-called composite sound types because they contain other elements (sound types). The composite sound types can be defined recursively e. g. a Start Parallel composite sound type can define elements which are again Start Parallel. The recursion stops with sample sound types. Hence the auditory scene is viewed as a tree hierarchy. The leaves of this tree are the sample sounds.

A VISUAL USER INTERFACE ENVIRONMENT FOR CREATION AND MANIPULATION OF AUDITORY SCENES

Based on above defined sound types various user interface components for creation and manipulation of auditory scenes are discussed in this section. A sample scene derived from a radio play production is given in order to demonstrate how the software prototype represents the structure of the scene. The main focus is on various sound types within this scene.

Sample Scene:
A man opens the door. The sound of the wind is audible in the background the whole time. He walks into the room and than turns the radio on. At the same time he opens a bottle of wine and pours himself a glass. After a few seconds he turns the radio off and says "always boring music".

The above sample scene can be viewed as an auditory scene with a hierarchical structure and can be described as follows:

There are two parallel sound types in this scene. The background wind sound type is audible throughout. The second sound type consists of series of sounds. These two sound types are named "Wind" and "Walking Man". The "Walking Man" sound type is a composite sound type and has the following four elements:

* door squeaking sound
* Step sound
* a composite sound type which consists of two parallel sounds; the radio sound and the sound of drinking
* the spoken sounds when the man says "always boring music"

Visualising the Structure Of Auditory Scenes

The hierarchical structure of an auditory scene is shown by using a network structure. Each sound type is presented by a rectangular box and associated colour (note that the colour is not visible in this paper). The length of a box is proportional to the duration of the sound. The "Startparallel", "Masterparallel" and "Serial sound" boxes have their own colour and thus can be easily recognised. The "Startparallel" and "Masterparallel" are presented on a horizontal line and the "Serial sounds" on a vertical line. When the path is closed it means that connected sound types will be finished simultaneously (see the Illustration in next Section).

User Interface Actions

The implemented software prototype facilitates following actions:

* Defining a new scene (new-function)
* Uploading an existing scene
* Storing a new or modified scene
* Importing an auditory scene
* Inserting a new element (sound type) into the auditory scene
* Deleting a sound type (cut-function)
* Copying a existing sound type (copy/paste-functions)
* Viewing and hiding the content of a composite sound type (browsing function)
* Undo function
* Viewing and changing the parameters of sound types
* Changing the view e. g. zoomin and/or zoom out
* An editor for textual description of auditory scenes based on developed hierarchical grammar
* Playing the whole or a part of an auditory scene
* Stop and suspend functions while playing an auditory scene

The following illustration is a snapshot of the implemented software prototype. The structure of the sample auditory scene is illustrated by using this software.

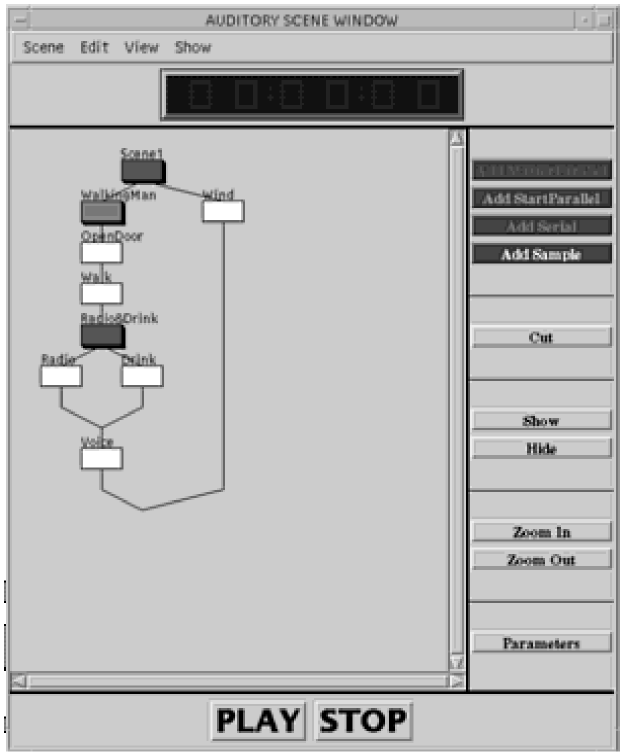

Figure 1: Snapshot of the sample scene

CONCLUSION

This paper has introduced an implemented software prototype for the creation and manipulation of auditory scenes based on a suggested concept called "Auditory Scenes". By using the software, auditory scenes can be generated and manipulated visually (direct manipulation). The prototype was developed on a Ultra- Sun workstation and using the JDK(Java Developer Kit). Sound synchronisation played an important role in this work. The Multithread Mechanism in Java has been used in order to implement the sound synchronisation. This model allows us, based on two primitive methods "Play" and "Stop", to implement higher functions such as repetition and changing the duration of a sound and the like as well as building composite sound types. Other composite sound types can be easily implemented by using this model. The JMF (Java Media Frame Work) [1] was not available at the time of developing this prototype. Therefore some functions had to be realised from scratch and others such as "changing the volume of an Auditory Scene", could not be programmed. The software prototype was developed in order to prove the validity of the concept of Auditory Scenes. Most of the other currently available tools such as Kyma [6] are based on physical models.

REFERENCES

[1] Michael Albers and David J. Rivas. Audios future in java language.
http://www.santafe.edu/~icad/ICAD96/proc96/rivas5.htm.
[2] J.A. Bergstra, J.Heering, and P.Klint. Algebraic specification. ACM Press, New York, 1989.
[3] Albert S. Bregman. Auditory Scenes Analysis. MIT Press, Cambridge, MA, 1990.
[4] A. Darvishi and H. Schauer. Formal description of auditory scenes. In IEEE Proceedings on Computer Animation, 1997.
[5] Alireza Darvishi. Formal Description of Auditory Scenes. PhD thesis, University of Zurich, spring 1997. To appear.
[6] Carla Scaletti, The Kyma Language for Sound Specification. Champaign: Symbolic Sound Corp., 1990.