CENTER FOR COMPUTER RESEARCH IN MUSIC AND ACOUSTICS MAY1988

Department of Music
Report No. STAN-M-49

DESIGNING DIGITAL INSTRUMENTS with HIERARCHICAL WAVEGUIDE NETWORKS

G. E. Garnett

Research sponsored in part by
The System Development Foundation
and
Dynacord, Germany

CCRMA
DEPARTMENT OF MUSIC
Stanford University
Stanford, California 94305

I. Introduction

Waveguide filters provide an efficient approximate means to model basic physical acoustics (see [Smith85b]). Using these techniques makes it relatively easy for an instrument designer or user to understand what he or she is working with; there is a clear correspondence between the waveguide model parameters and the corresponding physical gestures and/or mechanisms that are being modeled.

Furthermore, since hierarchical networks (see [Garnett88a]) of waveguide filters can be plugged together in arbitrary shapes, so long as the basic connection principles are adhered to, it is easy to design large complex networks as a collection of connected, simple sub-networks, which I call modules. These modules can be stored in a library and recalled for later use. They share all the pluggability of simple waveguide filters and can be joined together in any way desired.

I will not go into any technical details of physics, waveguide filters or specific modeling techniques, these topics are discussed in the references. What I will present here is a high-level view of the kinds of instruments that can be modeled, a classification system that allows key model elements to be factored into generic types and a brief look at a computer environment that facilitates working with this kind of modeling.

II. Classification of Physical Instrument Types

In this section I merely categorize the many acoustic elements of musical instruments into useful conceptual classes in order to better understand what can be gained from hierarchization and modularization.

A. Generators

What I am calling generators are those elements of a physical system that provide energy to the system. These can be further grouped into generators that last only a short while,

sporadic generators (or non-interactive generators), and generators that last longer, usually for the duration of the note, called *continuous generators* (or interactive generators).

Some examples of sporadic generators are:

1) hammers, such as piano hammers or drum sticks
2) plectra, harpsichord quills, finger plucking and guitar picks.

Some examples of continuous generators are:

3) reeds, double and single
4) vibrating lips and vocal cords
5) frictional generators such as the violin bow
6) air jets, as in the flute and organ pipes

B. Resonators

The family of resonators can be grouped into one, two and three dimensional sub-types.

The one-dimensional resonators are exemplified by a stretched string.

The two-dimensional resonators can represent vibrating plates or boards, such as a soundboard or a drum membrane.

Three-dimensional resonators are usually volumes of air trapped within another medium. So, woodwind bores and concert halls are examples.

C. Radiators

Radiators are very similar to resonators in physical terms, and in fact, it is sometimes very difficult to decide into which category a particular object should be placed, however, conceptually there is a big difference. The chief effect of a resonator is to, as its name foretells, resonate or bring out a particular set of frequencies. The chief effect of a radiator, on the other hand, is to amplify the entire sound. I would have called this class of objects amplifiers, except for the fact that, in almost every case, they end up doing far more than is implied by simple amplification, they tend to substantially "color" the sound. Most important among the radiators are the "bells" of

wind instruments and the soundboards of keyboard instruments.

D. Modifiers

Modifiers alter one or more of the basic characteristics of the sounding object. For example, the vibrato induced on a violin by the wiggling of the stopping finger actually is altering the size of the string resonator in a quasi-periodic fashion. The damper on a piano string is also considered a modifier; it serves to alter the decay characteristic of the string resonator in such a way as to curtail the resonance.

III. An Environment for Instrument Design

A basic software environment has been built up in Smalltalk-80™ [Goldberg] using the inherently hierarchical tools provided by this object-oriented programming language and user interface. A graphical network drawing program is incorporated. A user can design a network out of basic components and *visually* plug them together into larger units. This graphical network is then compiled into executable code and run.

A. Building a simple hierarchical waveguide network

Each of the above categories of generic acoustic system types has been coded as a Smalltalk class (see [Goldberg] for a description of classes and objects). The resonators and radiators are subclasses of the **HWNetworkModule** class. The generators are subclasses of a more generic type, the **SoundObject** class.

The **HWNetworkModule** class includes all the basic functionality needed to build, combine and execute hierarchical networks of waveguides.

The following Smalltalk method, to be sent to the class **HierarchicalWN,** will serve as a simple example:

```
toStream: aStream samples: numSamples
```

```
    "This is a simple test to generate a hierarchical
network and run it for numSamples writing the output
of one waveguide to the stream aStream. Evaluate the
following to try it out:"
    "HierarchicalWN toStream: (SoundFile
newFileNamed:'HWGN.1') samples: 500"

    | network out aModule |

"make network an instance of a HierarchicalWN"
    network <- self new: 'a hierarchical network'.
"make an initial port"
    network addNode: (Port new) at: 1.
"make a second port"
    network addNode: (Port new) at: 2.
"make an initial edge"
    network addEdge: ((Edge new)
                        admittance: 10.0;
                        initialSize: 8
                        node1: (network nodes at: 1)
                        node2: (network nodes at: 2)) at: 1.
"create an independent module"
    aModule <- (HierarchicalWN new: 'a module').
    aModule addNode: (Port new) at: 3;
             addNode: (Node new) at: 4.
    aModule addEdge: ((Edge new)
                        admittance: 1.0;
                        initialSize: 5
                        node1: (aModule nodes at: 3)
                        node2: (aModule nodes at: 4)) at: 1.
"add the new module to the network at node 1"
    network addModule: aModule
          linking: (aModule nodes at: 3)
          to: (network nodes at: 1).
"initialize the coefficients of the whole network"
    network resetGamma.
    network setGamma.
    network setAlphas.
"load it with an impulse just for testing purpose"
    (((network edges) at: 1) waveguide) output1: 2000.0.
"run the simulation for numSamples"
    1 to: numSamples do:[ :i |
"generate the next state of the network"
          network next.
"grab an arbitrary sample of the network as output"
          out <- ((network nodes) at: 2) sum.
          aStream nextPutFloat: out].
    aStream flush.
"make a picture of the resulting data"
    SoundView openOn: aStream length: 500.
    ^network
```

The quoted text in boldface is comments. The remaining text is Smalltalk code. This particular method simply sets up a generic network and adds to it a sub-network (called aModule), then executes the resultant network writing the samples to a disk file. The second to last line (SoundView openOn: aStream length: 500) opens a Smalltalk window to display the generated samples in an interactive time vs. amplitude plot. The organization of the soundfile storage, analysis and viewing system I describe in a forthcoming paper [Garnett88b].

As I mentioned above, the current environment includes a simple graphical waveguide network editor. It allows a designer to use the mouse to draw waveguide network junctions, represented as circles, and to connect them with waveguide branches, represented as lines. Future additions will allow the user to add arbitrary generator modules, each with its own user-defined icon, at any point in the network. This leads to a very flexible and versatile kind of graphical patch system. In fact, the class of generators has already been expanded to include such objects as oscillators and simple non-waveguide filters along the lines of standard Music V type "unit generators."

Currently, the graphic interface itself is crude, but, since Smalltalk provides a large number of tools for manipulation of bitmaps and visual icons, it is expected that a little work in this direction will bring about large rewards.

IV. References

[Garnett88a] G. E. Garnett and B. Mont-Reynaud, "Hierarchical Waveguide Networks," submitted for publication

[Garnett88b] G. E. Garnett, "Signals, Music and DSP," in preparation.

[Goldberg] A. Goldberg and D. Robson, *Smalltalk-80: The Language and its Implementation,* Addison-Wesley Publishing Company, Reading, Massachusetts, 1983.

[Smith85b] J. O. Smith, "Waveguide Digital Filters," Center for Computer Research in Music and Acoustics (CCRMA), Dept. of Music, Stanford University, March 1985. (In preparation: Draft copies available upon request.)

V. A Short Bibliography

[Adrien] J. M. Adrien and X. Rodet, "Physical Models of Instruments: A Modular Approach, Application to Strings," Proc. 1985 Int. Conf. Computer Music, Vancouver Canada, Computer Music Assoc., 1985.

[Benade] A. H. Benade, *Fundamentals of Musical Acoustics,* Oxford University Press, New York, 1976.

[Garnett87] G. E. Garnett, "Modeling Piano Tone Using Waveguide Digital Filters," Proceedings ICMC, Urbana-Champaign, 1987

[McIntyre83] M. E. McIntyre, R. T. Schumacher, and J. Woodhouse, "On the Oscillations of Musical Instruments," Journal of the Acoustical Society of America, 74, 5, pp.1325--1345 , Nov. 1983.

[Morse] P. M. Morse, *Vibration and Sound,* published by the American Institute of Physics for the Acoustical Society of America, 1976 (1st ed. 1936, 2nd ed. 1948).

[Smith86a] J. O. Smith, "Elimination of Limit Cycles and Overflow Oscillations in Time-Varying Lattice and Ladder Digital Filters," Music Dept. Tech. Rep. STAN--M--35, Stanford University, May 1986. Shorter version, Elimination of Limit Cycles in Time-Varying Lattice/Ladder Filters in *Proc. IEEE Conf. Circuits and Systems*, San Jose, May 1986.

[Smith86b] J. O. Smith, "Efficient Simulation of the Reed-Bore and Bow-String Mechanisms," Proc. 1986 Int. Conf. Computer Music, The Hague, Netherlands.