WaRM: A Framework for Modelling Resonant Acoustic Environments

Joel Laird, Tom Huns, Paul Masri, Nishan Canagarajah
Digital Music Research Group, University of Bristol
5.01 Merchant Venturers Building, Woodland Road, Bristol BS8 1UB, United Kingdom
Tel: +44 (0)117 954-5192; Fax: +44 (0)117 954-5206
email: Joel.Laird@bristol.ac.uk; URL: http://www.fen.bris.ac.uk/elec/dmr/

Abstract

WaRM (Waveguide Resonance Modelling) was created as a software tool for constructing three-dimensional resonant spaces, from guitar boxes to concert halls, using digital waveguides. It has been implemented in MATLAB as a script-based description language and is in fact capable of specifying any acoustic structure of any dimension that can be realised with a combination of delay-lines and filters. The scripting language has been devised such that the description of the resonant structure alone is sufficient for complete simulation, freeing the user from consideration of task scheduling or signal flow between model elements. Using just seven keywords, the statements specify creation and interconnection of reflectors, resonators and diffuse reverberators. The simplicity of the syntax and structure is demonstrated with two worked examples, which illustrate its effectiveness for specifying simple reverb effects and complex virtual room geometries alike.

1. Introduction

The digital waveguide has become popular in musical instrument synthesis because it provides a concise, efficient and accurate way of modelling wave motion that enables real-time physical modelling. The WaRM (Waveguide Resonance Modelling) framework has been constructed on this basis as a tool for synthesis of resonant and reverberant spaces. It has been implemented as a script-based description language in MATLAB, within a user-interface that includes an editor and parser, compilation and execution functions.

The language is composed of seven keywords that describe unambiguously the structure of the resonator, using a combination of delay-lines, filters, diffusion matrices and interconnections. Section 2 introduces the acoustic phenomena of resonant environments and explains how they can be modelled using these building blocks.

Section 3 presents the syntax of the WaRM description language, which is used to specify the construction of a resonator. This structural information is sufficient for complete simulation, so the user does not need to consider the timing of events between the model elements.

Two example room models are implemented in section 4 to demonstrate that WaRM is capable of simulating simple ‘reverb’ algorithms to sophisticated contemporary virtual environments. Finally, in the conclusions, web-resources are provided and planned improvements are described.

2. What Makes a Room Sound like a Room (and how can it be modelled)?

Of the paths travelled by sound waves between a source and a listener, the direct path and the early reflections are the most distinct and provide aural cues to the environment size and shape [2]; see figure 1a-b. The direct path is the shortest distance between source and listener; it is usually sufficient to model this with a delay-line and possibly some attenuation. Early reflections are the sounds that arrive via only a few boundary reflections and they provide the first aural cues of the distance and materials of the boundary surfaces. They can also be modelled with a tapped delay-line and attenuation (which may be frequency-dependent to account for the materials of the boundary surfaces). Additionally, some dispersion may occur (primarily at the boundaries) resulting in frequency-dependent wave speeds, which can be modelled with an all-pass filter.

The later echoes may include dominant frequencies, as standing waves become established. In an enclosed space, there are usually several resonant modes, which give rise to this effect of ringing. Each mode represents the cumulative distance that sound must travel in order to return to the same location travelling in the same direction; see figure 1c. Each of these closed-loop paths may be modelled by a looped delay-line (the length of which is determined by the path’s length) and frequency-dependent attenuation.

Later echoes are also more attenuated and diffuse than early reflections and they tend to arrive at the listener in close succession, so that their effect is
blurred giving the overall impression of reverberation. Diffusion can result in energy being partially transferred between resonant modes as scattering (at rough boundaries) alters the path of some of the 'sound rays'. This creates interaction between the modes, which can be modelled with an interconnection matrix that links all the looped delay-lines and leaks (the higher frequency) energy from each mode into the others; see figure 1d.

![Diagram of elements of room modelling](image)

Figure 1 - Elements of Room Modelling

3. Construction and Simulation

3.1 WaRM Syntax for Construction

The seven keywords used in WaRM are:

- **source**, **listen** - the input/output ports, which may be audio streams for real-time use or files for off-line calculation;
- **filter** - for frequency dependent attenuation (low-pass), fractional delays (all-pass) or general IIR filtering (eg. high-pass);
- **delayline** - a shift register for implementing delays and closed-loop paths;
- **node** - a junction for interconnecting delaylines; it may be a single-input, single-output connection (with optional filtering) or an n-input, n-output junction with a complex combination of filters, gains and mixing for implementing a diffusion matrix;
- **associate** - incorporates a scaling factor and filter(s) into the output stage of a delayline or node; for nodes with multiple outputs, this signal processing is applied to each identically;
- **join** - connects delaylines (at any tap position) to nodes and source or listen objects to any combination of delaylines and nodes.

See the examples in section 4 for an idea of code structure.

3.2 WaRM Simulation

The WaRM script is a structure description and must be externally compiled and executed. No scheduling or prioritisation of components is needed in the language, because the framework makes operation unambiguous. There are only two primary object types: delayline and node, which may only be connected to the other type. During simulation the computation focus alternates between these types without preference on order. There is special case for the source and listen object types, to which delaylines and nodes may both be connected, and these have additional protocols.

The following sequence encapsulates the complete task scheduling for one iteration:

1. Read data into source(s).
2. Perform all node processing, where order of nodes processed is unimportant:
   a) fetch data from delaylines and sources;
   b) compute the filtering 'equation' (separately within each node cell if it is a matrix);
   c) combine additive terms within matrix processing;
   d) perform filter and gain processing associated to the node as a whole;
   e) place results on node outputs.
3. Advance clock: advance shift registers in all delaylines by one sample.
4. Perform all delayline processing, where order of delaylines processed is unimportant:
   a) fetch data from nodes and sources;
   b) perform in-place filter and gain (on output tap only).
5. Write data out of listen(s):
   a) read from delayline taps, node outputs and sources;
   b) scale and sum;
   c) output.

3.3 Additional Options and Constraints

There is insufficient space in this paper to describe more than this overview, but the full set of syntax rules and options is available from the web URL at the top of the paper.

However, it is useful to note that waveguides can be constructed using a pair of delayline objects and nodes can function as scattering junctions. Therefore, it is even possible to construct waveguide meshes in WaRM for two or higher dimensional finite difference models.
4. Example Room Environments

4.1 Basic Non Room-Specific Reverb

The simplest kind of 'reverb' can be constructed from comb filters. The effect is not specific to any particular room or geometry, but provides the sense of space. In WaRM, these are simulated using delay-lines with feedback. A reasonably complex example is shown in figure 2 and in the WaRM script following.

```
soundin = source ('C:\AUDI\DRYVOCAL.WAV');
soundout = listen ('C:\AUDI\FXVOCAL.WAV');

echo = delayline(362);  
multicomb = delayline(2594);
lpf1 = filter (lowpass, 1.127000);
lpf2 = filter (lowpass, 1.143000);
lpf3 = filter (lowpass, 1.940000);
lpf4 = filter (lowpass, 1.580000);

feedfwd = node (lpf1);
feedbackA = node (0.45);
feedbackB = node (0.3*lpf2);
feedbackC = node (0.15*lpf3);
feedbackD = node (0.1*lpf4);

join (soundin, echo);
join (soundout, multicomb, 0.1);
join (feedfwd, echo.n, multicomb);
join (feedbackA, multicomb, 784, multicomb);
join (feedbackB, multicomb, 1647, multicomb);
join (feedbackC, multicomb, 1832, multicomb);
join (feedbackD, multicomb, 5, multicomb);
```

Additional syntax notes for this example:

- gain and filter combinations can be specified with a pseudo-equation (where * means serial combination and +, - mean summation) so that complex processing is possible including parallel and serial paths;
- delayline taps are specified with a . suffix, where .0 is the first tap and .n means the final tap (after its associated post-processing).

4.2 Virtual Room Modelling

A powerful technique that directly computes the effect of each reflection is the image method [3], in which a point source is reflected in each wall to identify where sound appears to come from after reflection. These virtual sources are then reflected themselves in an iterative process that repeats for each reflection, until the signals have sufficiently decayed. The path from a virtual source to the listener represents an actual path that the sound travels from the real source to the listener. Therefore, once this method has been applied, the room geometry can be discarded and the real and virtual sources can be used for synthesis. The only disadvantage of the approach is that if the geometry is complex or the attenuation at each boundary is small, many sources (and hence *delay-lines and filters also) are required.

Several room models use the image method to model the early reflections only and model later reflections as described in section 2 (figure 1c-d). Huopaniemi et al. implement diffusion by all-pass filtering the sum of the outputs of a set of resonant mode delay-lines and distributing this signal back into all the delay-lines, completing the loop [4]. In an alternative scheme, a kind of the ball-within-the-box, Rocchesso uses Gerzon's Feedback Delay Network [5] to model diffusive reverberation [6]. Both of these designs can be implemented in WaRM script. The example in the following code and in figure 3 overleaf implements the ball-within-the-box method to simulate the three principal modes of a medium-sized room.

```
soundin = source ('C:\AUDI\ORCHESTRA.WAV');
soundout = listen ('C:\AUDI\CONCERT.WAV');

echo = delayline(1126);
modeA = delayline(2648);
modeB = delayline(1589);
modeC = delayline(796);

lpf1 = filter (lowpass, 1.160000);
lpf2 = filter (lowpass, 1.140000);
lpf3 = filter (lowpass, 1.830000);
lpf4 = filter (lowpass, 1.960000);
lpf5 = filter (lowpass, 1.780000);
lpf6 = filter (lowpass, 1.480000);
directpath = node (0.999);
earlyrefA = node (0.98*lpf1);
earlyrefB = node (0.92*lpf2);
earlyrefC = node (0.91*lpf3);
diffusionmat = node (0.8727, -0.2757, 0.4030, 0.8727, -0.2757, 0.4030, 0.8727, -0.2757, 0.4030, 0.8727, -0.2757, 0.4030, 0.8727)

associate (modeA, 0.92, lpf4);
associate (modeB, 0.97, lpf5);
associate (modeC, 0.99, lpf6);

join (soundin, echo);
join (soundout, directpath, 1, modeA, 0.1, modeB, 0.1, modeC, 0.1);
join (directpath, echo 265, *);
join (earlyrefA, echo 473, modeA);
join (earlyrefB, echo 812, modeB);
join (earlyrefC, echo, modeC);
join (diffusionmat, modeA, modeB, modeC, modeA, modeB, modeC)
```

ICMC Proceedings 1999 – 547 –


Figure 3 - Virtual Room example in WaRM

Additional syntax notes for this example:

- the symbol > is used in the join statement to specify a node that is not connected to a delayline at its input/output;
- connections to listen objects may be from delaylines and/or nodes; each one is followed by its scaling factor in the script for a weighted summation;
- join statements take the form 
  join(node, from_delay, to_delay);
  if a suffix is not provided then 
  from_delay.n and to_delay.0 are the defaults.

5. Conclusions

WaRM has been devised as a framework for rapid prototyping of resonant acoustic environments, using the techniques of digital waveguides. The examples demonstrate that simple and complex structures may be created with only a handful of self-explanatory (and hence easy to debug) statements. As a tool, WaRM provides a level of abstraction from the details and scheduling of signals, so that a designer can focus on the structure.

WaRM is a platform-independent framework and it has been presented here as a description language. It has also been implemented in MATLAB within a GUI environment that includes editing, parsing, construction (compiling) and execution operations. There are plans to convert the code to C++ (which is much more efficient at implementing iterative functions) and to create a graphical construction palette of the pick-and-place variety. As mentioned earlier, full details of the syntax rules can be found at the Digital Music Research Group web site: www.bris.ac.uk/elec/dmr/.

References


