Dynamically Configurable Feedback/Delay Networks: A Virtual Instrument Composition Model

Michael Hamman
School of Music
University of Illinois Urbana-Champaign
m-hamman@uiuc.edu

Abstract

A virtual instrument is developed which consists of a system of feedback/delay networks that can modify themselves while the system executes. The system consists of two layers: signal propagation and signal control. Signal propagation is accomplished as an initial signal is propagated through a specified network of delay and multiplier elements. Signal control is accomplished by attaching 'control nodes' to any or all signal propagation components. Control nodes act either as independent agents or as part of a global control structure. Output signals can be fed back into both propagation and control layers to further modulate the behavior of the system. The system allows the composer/engineer to link pattern-generating algorithms to those of sound synthesis and to design systems in which both its components, and their interactions, are formulated as dynamically configurable agents.

1. General Description of the System

resNET is a sound synthesis/composition software system in which structure is conceived as a network of interacting agents. Its current prototype version is written in C++ and has been developed on an Intel 80486-based machine and currently produces sounds by writing output samples to disk.

Functionally, the system delineates two layers: signal propagation and signal control. At the signal propagation layer, a network of interconnected modules is defined. These modules are grouped into two types: an exciter and a resonator. As its name implies, an exciter provides initial input signals to a system. Similarly, a resonator acts as a resonating 'body' into which a signal is dispatched. A third module type provides spatial placement information to the output signal. These module groupings can be interconnected in either feed-forward or feed-back configurations, allowing for complex re-propagation of the generated signal throughout the system, as in the following example:

```
In this example, output from an exciter is fed into a resonator circuit; the output signal from the resonator circuit is simultaneously forwarded to a signal out module (which spatializes the output) and fed back to the exciter. The output of the signal out module is simultaneously sent out to a D/A converter, or stored medium) and fed back as input into the exciter and resonator modules.

At the signal control layer, each component of the signal-propagation network is attached to a control node. A control node acts either as an independent agent, or as component of a larger integrated sub-system, and it dynamically constrains the behavior of its attached module. But, while a control node may determine the constraints defining the behavior of a signal-propagation module, it is the module itself which defines exactly how those constraints are to be applied. Moreover, components at the signal propagation level can themselves modulate the behavior of its attached control node or even of control nodes attached to other propagation components. In the following diagram, control and propagation layers are shown as distinct subsystems with bidirectional pipes connecting them.
```

Sound Synthesis Techniques 394
2. Specification of Excitor, Resonator, and Output Module Groupings

An excitor generates a signal in one of three manners: read from disk (a sound file stored to disk); generate using a supplied formula specification (an algorithm); and input from a live signal. An excitor is encapsulated so that its output is always of a specific type regardless of its method of generation.

A resonator is comprised of a network of delay and multiplication components interconnected according to specifications made by the designer using an ASCII script file. The following shows a script file fragment along with its diagrammatic representation:

```
 Delay
 delay2 = 1
 Mult
 control = .99
 Network
 pluse + in
 delay2 =-pluse
 delay0 = pluse
 mult0 = delay0
 out = delay0
```

A minimal script file consists of three sections, each marked with a .Command processor flag. Delay lengths are specified within the .Delay section. Similarly, multiplier values are specified within the .Mult section. Finally, the network patch is specified in the .Network section. As shown in the script file, delay0 is set to a length of 1 and the value for mult0 is set to .99. The .Network specification models a simple low-pass filter. It should be noted that since precise ordering of network components is essential to the correct implementation of a design and because such an ordering may not always be intuitive to the designer, all input script files are subjected to re-ordering prior to execution, thereby allowing the designer to represent a design without having to worry about correct ordering. In future versions of the software, the designer may order a design either as a script file or as a graphical circuit design like the one shown above.

The output module handles spatial placement in the current prototype version. Control data for spatial placement is obtained from particular components within a resonator network, or from control nodes operating at the control layer. The output module also watches for word overflow, calling a designer-supplied interrupt should overflow occur. One such interrupt -- called BounceOverflow() -- causes the overflowing signal to 'bounce' off the 16-bit wall. This allows for interesting timbre designs whereby the output of the system is purposely driven to be very slightly unstable: the output module's 'bounce' interrupt acts as a kind of waveshaper. Another such interrupt throws away the current sample, reduces all of the multiplier values by some specified value (usually very small), causes the entire system to jump backward in its iterations by one, and then allows the resonating subsystem to generate another sample.

In future implementations, the output module could be patched directly to remote modules which buffer its output signals and reorders them in time according to input algorithms, audio signals from live performers, or other control signals.

3. Strategies for Timbre Design

While reex/syn could be used to design timbres in which traditional signal processing constructs are employed, its real utility as a design system is in employing experimental and incremental extensions to those traditional constructs. Moreover, even sequences of extended operations can be realized through the application of control layer modules.

As a first step in designing such a network, we begin with a plucked string circuit attached to a simple filter module.

ICMC Proceedings 1994 395 Sound Synthesis Techniques
With all components fixed, a variety of plucked string-like timbres result. With the introduction of variability to some of the components, a far greater variety of timbres can be designed. As an example, a particular configuration is considered.

In this example, delay 1 has a length of 31 samples (as is the length of the noise burst excitor input); multipliers g1 and g2 each have a value of 5; delay 3 has a length of 2. All of the other components (including the passting module) are variable and, as such, can be piped to a control node.

Each signal propagation component which is attached to a signal control node, tells that node its maximum range of variability, its minimum and maximum values, and passes a pointer to an iterative function which is used to control its behavior. This function is either an independent function – and therefore does not exchange information with functions calling other signal propagation components – or it passes control to a single 'global' function within which control nodes can exchange information according to the specification of that function.

In this example, if multiplier components g4 and g5 are piped to mutually symmetrical functions, such as sines functions with opposing phase, the resulting waveform consists of varying band-pass and band-reject filtered sounds. g1 is set to 1.0, and is attached to a sine function with an amplitude at around 1000 times the value of the multiplier. With delay 2 set at around 900, and its range between 200 and 1800, the resulting behavior ranges from discernable melodic structures to continuously transforming timbres, depending on the control functions defined.

By experimenting with incremental variations of each a configuration, one generates timbres which can be characterized as fixed, continuously transforming, or intermittent. Intermittent timbres are those which seem to ride a fine line between being perceived as a single continuously transforming timbre or as a sequence (perhaps overlapping) of single timbres, either fixed or transforming.

4. Linking Composition and Timbre Design

An important goal in the design of resNET has been to explore ways in which micro-level structures (for instance, those which generate a 'single' auditory event) and macro-level structures (for instance, those which generate an entire composition or a sequence of auditory events) can be joined systematically — i.e. as a matter of system-level design. Toward this end, resNET is designed such that the signal control layer modulates the signal propagation components, and not the signals themselves. This design allows the composer to 'algorithmically' specify procedural inputs to the control nodes modulating signal-propagation components. The term 'procedural inputs' refers to inputs which alter the behavior of a procedure or function, rather than modifying the parameters associated with that procedure or function.

As an example of such a design, consider a network in which each component is patched to all other components through a multiplier:

Sound Synthesis Techniques 396
ICMC Proceedings 1994
In this simple example, each multiplier is attached to a control node. The control layer is constructed as a single 'global' function into which the output signal is fed. Within the control layer, the output signal is analyzed, and the results of that analysis is used in 'tuning' the individual multipliers in order to heuristically favor particular output signal types. Since some multipliers will eventually be zeroed out, a specific configuration emerges. By changing the procedure inputs to the analysis module, different configurations can be specified. The activity involved in iterative generation of procedural inputs to such a system is of interest to the designer who wishes to systematically incorporate large-scale morphological structures into the generation of individual timbral events, and vice versa.

5. Conclusions and Future Developments

resNET represents an effort at specifying a computer (virtually) instrument which can extend the notion of what an instrument is. An attempt is made to extend the traditional notion of instrument from one which allows for the incremental realization of specific musical goals to one which allows for the incremental specification of musical states. Consistent with this effort is a desire to incorporate composition specification directly into the design of an instrument.

Future development plans include the addition of a DSP coprocessor (based on the TMS320C50 processor) to the hardware configuration (currently a 68030-based computer), and to port the software in Windows NT (or Chicago) for development of multithreaded processing and a graphical user interface.

resNET has been used in the composition of topologies/surface/oblique angles/initial parameters, a work for two-channel tape. In this work, a small collection of networks representing known models, such as filters and reverberators, are broken down into their subcomponents and then reassembled according to compositional logic which only obliquely reference the timbres normally associated with the original network configurations. Future compositional plans include a new work for bass and computer in which a computer 'bowed string' instrument is modeled, its signal-propagation configurations forming the basis for both computer-live-performer interaction and for transformations of timbral and formal construction.

6. Acknowledgment

The author gratefully acknowledges Adam Cain for his input into the initial design of the system, for his assistance in analyzing the correct performance of the system, and for providing valuable theoretical insights into its behavior.

7. References


ICMC Proceedings 1994 397

Sound Synthesis Techniques