reprint from: *Proc. ICMC93*
International Computer Music Conference, 10-15 September 1993, Tokyo, Japan

Using Physically-Based Models and Genetic Algorithms for Functional Composition of Sound Signals, Synchronized to Animated Motion

Tapio Takala

Department of Computer Science
Helsinki University of Technology
02150 Espoo, Finland
tta@cs.hut.fi

James Hahn, Larry Gritz,
Joe Geigel, Jong Won Lee

Dept. of Electrical Eng. and Computer Science
The George Washington University
Washington, DC 20052, USA
hahn|gritz|geigel|won@seas.gwu.edu

Abstract

We represent sound signals as general functional compositions, called "Timbre Trees". Externally these are LISP-like expressions, internally they are implemented as C++ data structures. Nodes of the tree can be arithmetic operations, analytic functions or noise generators. Vectorized operations are provided for compact expression of additive spectral synthesis, and convolution operators for modeling acoustical environment (reverberation) within the same structure. A similar script language is also used to define three-dimensional animated motion. Simulation determines collisions and other sound-causing interactions between objects, and generates timbre trees from which exactly synchronized soundtracks can be prepared. Heuristic physically-based vibration models are used to determine the timbre of simulated instruments. Because it is often difficult to find the right composition of functions and their parameters that make up a desirable sound, we use genetic algorithms to mutate timbre trees and allow the user to guide their evolution. Time-variable parameters allow continuous metamorphosis between geometric objects and their sounds. Using this methodology, we have produced a variety of convincing animated scenes.

Keywords: sound synthesis, physical models, animation, multimedia, virtual reality

1 Introduction

The generation of sounds and their synchronization to graphics are important problems not yet addressed adequately to date. With virtual environments, the work has concentrated mainly on head-related transfer functions used as real-time convolution filters to give localization cues [Wenzel 1992]. Soundtracks of animations – whether synthetic or recorded – are usually synchronized by skillful handwork. Lytle [1991] used a common MIDI score to drive both sound synthesizers and an animated orchestra. In our previous work [Takala and Hahn 1992] we let the motion to determine sound effects. Pointing out analogies between image and sound synthesis, we used parallel pipelines to produce both (figure 1).

In this paper, we describe a more flexible system for the generation, evaluation, and use of sound. Due to the analogy to shade trees in image synthesis [Cook 1984], we call *timbre trees* the LISP-like expressions written in our general Sound Script Language. The language is similar in nature to Pixar's RenderMan used in graphics [Upstill 1989] or to MOSAIC used in music [Morrison and Adrien 1991]. The sound scripts can be generated by hand or automatically by a motion control system. Mapping the motion control parameters (position, orientation, forces, etc.) to parameters of the timbre tree allows automatic generation of synchronized sound whose timbre can vary with time, based on the motions being generated. Acoustical effects of the environment are also calculated and appended to the timbre tree as convolution nodes. The final timbre tree is then evaluated to generate the soundtrack.

2 Timbre Trees

Sound synthesis through functional composition of simple modules has been successfully explored in several systems ranging from MUSIC V to Kyma [Scaletti 1991], among others. Although related to our approach, these music-oriented systems do not address the issues of sounds generated from and synchronized to computer animation or virtual environments.

We represent sound compositions in a general tree-structured format, which we call "Timbre Trees". Externally these are represented as expressions in a LISP-like textual language; within an interpreter they are

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
15 SEP 1993	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Using Physically-Based Models and Genetic Algorithms for Functional Composition of Sound Signals, Synchronized to Animated Motion	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Department of Electrical Engineering and Computer Science The George Washington University Washington, DC 20052**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**SAR**	**5**	

implemented with C++ data structures. Each node of the timbre tree may represent one of the following:

- a numerical constant, or a vector of them
- named variable, passed from motion control, like "*t* " (time), "*angular-momentum*", etc.
- function taking subtrees as arguments
- digitally sampled sound
- reference to named timbre trees from a library.

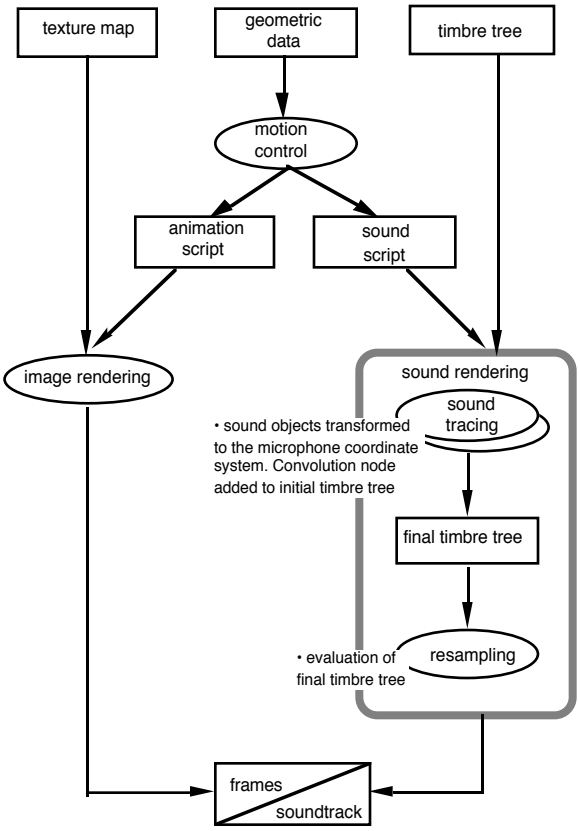


Fig 1: Parallel pipelines for image and sound rendering

Standard mathematical functions (+, -, *, /, exp, log, etc.) were implemented, as well as elementary waveforms (sawtooth, triangle, square, sine), various types of noise, and some signal processing functions (filtering, convolution). We provided a mechanism for extending the language to add new functions of arbitrary complexity by coding these in C++.

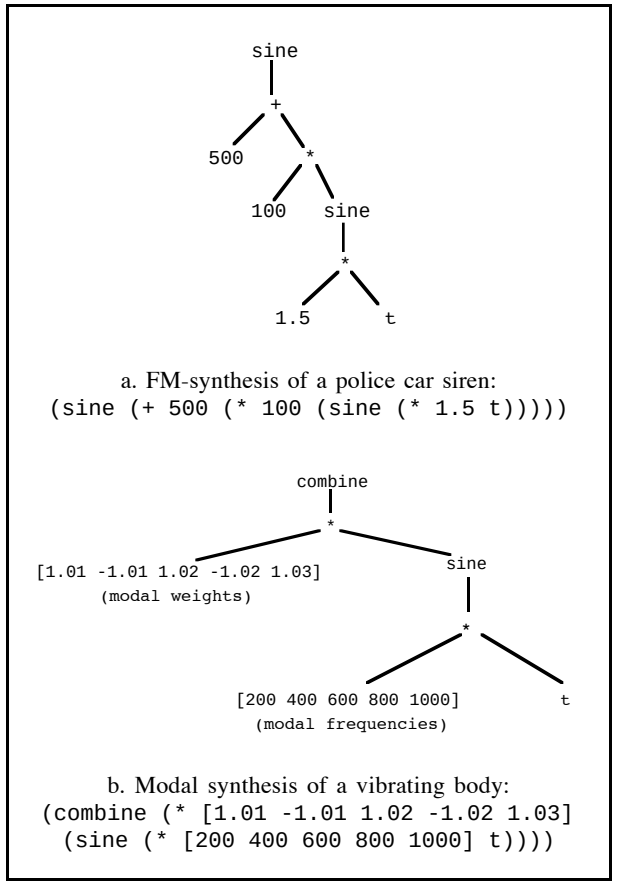
The language also supports vector operations where a single node operates on an array of parameter values. In figure 2b, the modal frequencies of a rigid body form a vector, and their relative weights another. A *combine* node sums up the vector components produced below it and returns a single scalar value. This results in a simpler, more readable, and better manageable tree.

3 Growing Timbre Trees

Timbre trees can be constructed by hand using a library of elementary nodes and trees, or derived from rigorous physical principles. Behavioral models, commonly used to animate complex systems of interacting individuals, apply to sound as well.. Once an initial trees is constructed, it can be altered by genetic algorithms to generate other trees.

3.1 Physically-Based Sounds

Physical models have recently gained growing interest in computer music, as indicated by the overview and taxonomy by Smith [1991]. However, exact vibrational analysis of a general mechanical body is extremely complicated, partly due to poor knowledge of material properties. Fortunately approximations often suffice.



a. FM-synthesis of a police car siren:

```
(sine (+ 500 (* 100 (sine (* 1.5 t)))))
```

b. Modal synthesis of a vibrating body:

```
(combine (* [1.01 -1.01 1.02 -1.02 1.03]
  (sine (* [200 400 600 800 1000] t))))
```

Figure 2: Timbre trees and corresponding expressions

Spectral synthesis is a well-known method to represent modal vibrations. Often these are harmonic, like in the example above. In some objects like bells or drums, the modes are so complex that they can be

represented with statistical distribution. A general form for such a timbre tree is:

```
(combine
   (*
     (sine (* t (rvector n_f base_f max_f)))
     (damp (* t (* damp_ratio
                   (rvector n_f base_f max_f))))
   )
 )
```

where *n_f* is the number of vector components randomly generated within the range (*base_f*, *max_f*). by the operator `rvector` The same vector is used in node `damp` to set the decay rate of each mode proportional to the frequency, as is natural in many physical systems.

This timbre tree is a parametric representation for a wide class of sounds. Convincing bell sounds were produced by combining approximately 20-100 frequency components. Wood-like sounds were created by increasing the number of frequency components and the damping rate, whereas higher frequencies and less damping produced metallic sounds (like striking a cymbal). Up to thousands of components result in a noisy sound (like a bass drum), but make the tree impractical to evaluate in analytic form.

3.2 Behavioral Modeling

We have used timbre trees to represent the buzzing sound of behaviorally-controlled bees in an animated short [Gritz et al. 1992]. Our first guess to simulate the bee's wing motion was a sawtooth wave of a suitable frequency. To avoid an instrument-like pure tone, the frequency peak was widened by adding a high frequency noise term. Even a noisy tune is dull, if the mean frequency is constant. A living thing's continuously varying activity was simulated by deviating the nominal frequency by a low frequency noise computed by a second-order Markov process (a process similar to one generating "random fly" motion). The resulting LISP-expression is:

```
(sawtooth (+
   (* t
      (+ nominal-freq
         (* 12 (noise (* t LF-noise)))))
   (* 0.25 (noise (* t HF-noise)))))
```

For a swarm of bees, we used the single tree in vector mode, evaluating one instance for each bee.

3.3 Genetic Algorithms

A well-known problem with functional sound synthesis is how to find desirable instances within the large domain of possible sounds. We have approached the problem by using genetic algorithms (GA). In music, GAs have so far been used for composition only [Horner and Goldberg 1991]. Details of GA techniques can be found in [Koza 1992].

Sims [1991] used GA to explore graphical images, defined by LISP-expressions. The system generated a number of images for the user to judge. The most promising were cross-fertilized and mutated to make new generations over and over, until a desirable texture was found.

Timbre trees can evolve exactly the same way. We start with a guess (often a heuristic or physical model). Several mutated versions of the tree are then computed to generate a series of other "related" timbre trees. At each node of the tree, a mutation may occur. It can be one of the following:

- node is replaced by a new random expression
- numerical constant replaced by another one
- parameter node replaced by a different parameter, a numerical constant, or a function
- function replaced by another (for example, `*` may mutate into `+`, or `sin` mutate to `exp`)
- the order of a function's arguments may be switched, the function replaced by one of its arguments, etc.

Crossing-over, analogous to sexual reproduction, is a variation where the above mentioned replacements are not random but nodes exchanged among parent trees.

The new generation trees are evaluated and the resulting sounds are played one at a time for the user, who rates each tree's "fitness". Those with highest scores are used for further mutation, and the cycle continues until satisfactory sounds are found. Since the user will kill off undesirable mutations, the search is directed towards a particular goal.

Using these tools, we have been able to produce an entire class of bee-like sounds (ranging from mosquitos to chain saws). Similar processes lead us to a class of police sirens some of which are clearly derivatives of the original sound but which defy description and would have been difficult to conceive without GAs.

4 Integrated Rendering System

Many sound timbres can be generated by striking an object in different ways. We have approached the problem by constructing and attaching a timbre tree for a class of physical objects and excitation modes (colliding, sliding, rolling, etc.). The parameters for these are extracted from physically-based simulations [Hahn 1988].

4.1 Mapping of Parameters

Parameters that define an instance of a sound can be mapped to parameters generated from motion control systems or physical processes (such as collision location, collision impulse, shape of object, etc.). This way, both timing and the timbre of the sound can be synchronized to the sound-causing events.

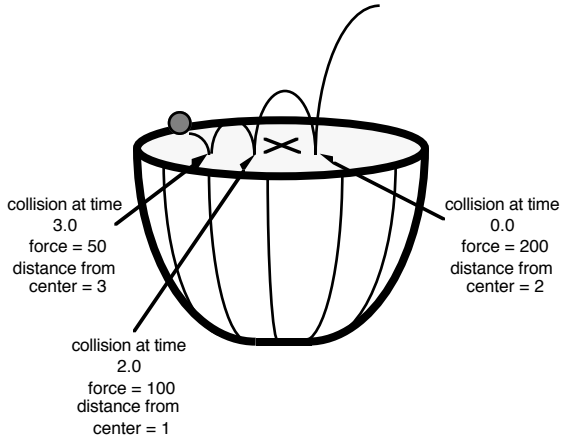


Figure 3: Object bouncing on a drum

For example, the timbre tree for a drum struck by an object could be the same modal synthesis tree as in section 3.1, with amplitude and damping rate as parameters. A physically-based motion control system may calculate for each strike the timing, position relative to the center, and the impact force (figure 3). These values, respectively, are fed to the damping rate and loudness parameters of the sound. As a result, a collision sound is produced every time the object strikes the drum.

4.2 Environmental Effects

A sound's transmission through the environment, including reflections and refractions, can be seen as a transformation of the sound from object space to microphone space. The effect of this transformation can be expressed as a convolution filter.

Convolutions are applied to timbre trees by attaching "environmental nodes", which depend on the objects' positions, orientations, and attributes as defined in a sound script. The result of applying an environmental node is itself a timbre tree, which will be evaluated in the creation of the final soundtrack.

For modeling these environmental effects we have used simple heuristics instead of sophisticated acoustical theory. Main components of our sound propagation model are attenuation and delay due to distance, occlusion factors based on bounding volumes, and reflection models similar to those used in graphics.

4.3 Sound Morphing

Time varying parameters have been used successfully in animation and modeling languages where the nature of motion and the structure of a model change in time as the parameters change. In our case, it is the timbre that will be modified as its parameters change in time.

One interesting application is in "sound morphing." By varying the parameter values of a timbre tree that defines a class of sounds, one can smoothly convert one instance to another of the same class. With this technique, we have made a continuous metamorphosis between bee-like and chainsaw-like sounds.

5 Conclusion

In the past, graphics and sound have been produced almost independently. It has become obvious that both senses and the added impact of synchronized sound and images will play a bigger role in providing a more complete experience. In this context, it is important that the design and use of sound occurs in conjunction with the generation of motions and images.

We have emphasized similarities between sound and animated images, both in their generation process and in their representation as tree-like structures. Timbre trees are a combination of shade trees from graphics and functional compositions from sound synthesis. Several methods earlier used in graphics are here applied to sound generation. Examples of physically-based and behavioral models are demonstrated with timbre trees. For interactive selection and development, of sounds, genetic algorithms are found to be useful.

Future extensions include more coupling between parameters of the motion and the parameters of the timbre tree. For example, the motions of deformable objects can be used to generate sounds (e.g. a flag snapping in the wind). We are also in the process of developing real-time sound systems to be used in virtual reality applications.

Acknowledgements

This work has been supported by the Naval Research Laboratory (NAVY N00014-91-K-203) and NASA Goddard (NCC 5-43). We also like to thank the students in the Computer Graphics and User Interface Group at the George Washington University, especially David Florek, Jean Favre, and Daria Bergen for their help in implementing parts of this system.

References

[Cook 1984] Robert Cook. "Shade Trees". Proc. SIGGRAPH'84, ACM Computer Graphics, Vol.18, No.3, pp.195-206.

[Gritz et al. 1992] Larry Gritz, Daria Bergen and Rudy Darken. *Graphic Violence* [video]. In SIGGRAPH'92 screening room show.

[Hahn 1988] James Hahn. "Realistic Animation of Rigid Bodies". Proc. SIGGRAPH'88, ACM Computer Graphics, Vol.22, No.3, pp.299-308

[Horner and Goldberg 1991] Andrew Horner and David E Goldberg. "Genetic Algorithms and Computer-Assisted Music Composition". Proc. ICMC-1991, p.479-482

[Koza 1992] J. Koza. *Genetic Programming*. MIT Press, Cambridge, MA, 1992.

[Lytle 1991] Wayne Lytle. *More Bells and Whistles* [video]. In SIGGRAPH'90 film show. Also described in Computer, Vol.24, No.7, July 1991, p.4 and cover.

[Morrison and Adrien 1991] Joseph D Morrison and Jean-Marie Adrien. "Control Mechanisms in the MOSAIC Synthesis Program". Proc. ICMC-1991, p.19-22.

[Scaletti 1991] Carla Scaletti. "The Kyma/Platypus Computer Music Workstation". In S.Pope (ed.) *The Well-Tempered Object: Musical Applications of Object Oriented Software Technology*, MIT Press, 1991.

[Sims 1991] Karl Sims. "Artificial Evolution for Computer Graphics". Proc. SIGGRAPH'91, ACM Computer Graphics, Vol.25, No.3, pp.319-328, 1991.

[Smith 1991] Julius Orion Smith. "Viewpoints on the History of Digital Synthesis". Proc. ICMC-1991, p.1-10.

[Takala and Hahn 1992] Tapio Takala and James Hahn. "Sound Rendering". Proc. SIGGRAPH'92, ACM Computer Graphics, Vol.26, No.2, pp.211-220.

[Upstill 1989] Steve Upstill. *The RenderMan Companion*. Pixar/Addison-Wesley, 1989.

[Wenzel 1992] E.M. Wenzel. "Localization in Virtual Acoustic Displays". Presence: Teleoperators and Virtual Environments, 1, 80-107, 1992.