Perceptually Based Scheduling Algorithms for Real-time Synthesis of Complex Sonic Environments

Hesham Fouad
James K. Hahn
Dept. of EE & CS
The George Washington University
Washington, DC 20052 USA
hesham@seas.gwu.edu
hahn@seas.gwu.edu

James A. Ballas
Naval Research Laboratory
Code 5513
Washington, DC 20375-5337
ballas@aic.nrl.navy.mil

ABSTRACT
In this paper, we present a technique for managing overload conditions that occur when computational resources are not sufficient to evaluate all the active sound sources in a Virtual Environment. A real-time scheduling strategy is introduced which degrades less important sound sources so that resource constraints are met. Finally, scheduling algorithms are considered based on their effect on listeners' perception of the resultant sound quality.

Keywords
virtual environment, sound synthesis, graceful degradation, real-time scheduling

INTRODUCTION

Synthetic sound generation is a computationally expensive process and resources may often not be sufficient to evaluate all the active sound sources in a complex Sonic Environment (SE). A major focus of our work has been on devising graceful degradation schemes for managing overload conditions so that the perceptible effects of overload are minimized. The Virtual Audio Server (VAS) was used as a framework for studying real-time scheduling algorithms that manage the sound generation process. VAS was developed as a general framework for the study of problems associated with integrating sound into Virtual Environment (VE) interfaces. The system provides an extensible scheduler that facilitates the development of real-time scheduling algorithms.

THE VAS SYSTEM

VAS is partitioned into four functional areas as depicted in fig. 1. The system has a client/server architecture that facilitates load balancing by distributing the graphics and audio processing on different machines. VAS's distributed architecture is based on *Remote objects.* Each VAS auditory object with which the client interacts has a corresponding remote object on the client's machine. Remote objects maintain their state and communicate with their corresponding VAS objects through a Remote Procedure Call (RPC) interface when their state changes due to a client interaction. In this fashion, an object-oriented interface to the server can be maintained and communication between client and server is minimized.

Sonic Scene Elements model the elements comprising the Sonic Environment (SE). They consist of the *Auditory World*, which maintains the overall state of the server and provides access to the other objects in the world. *Auditory Actors* model sound producing entities in the world. Two derivations of the Auditory Actor: *Auditory Space*, and *Listener* model the enclosures within the SE and the listener respectively. Auditory Actors have a sound repertoire consisting of a set of *Sound* objects, each representing a sound source. They evaluate sound samples in real-time and write the resultant samples to VAS *Devices*. An instance of a Device object is attached to each Sound object and provides it with a device independent interface to any spatialization device used by the server. Finally, the *Scheduler* manages the real-time evaluation of active sounds in the server.

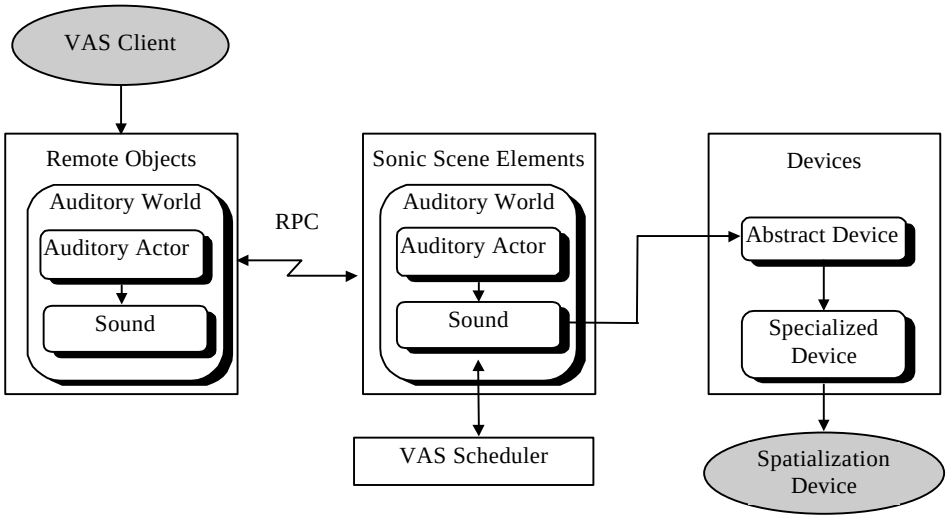

Figure 1 VAS system architecture

VAS Sounds as active objects such that each Sound object has a thread of execution associated with it. This thread is responsible for evaluating the sound signal and writing the resultant samples to the device object. Upon instantiation, Sound objects register themselves with the Scheduler providing it with an evaluation routine, which when executed, generates the samples. The Scheduler determines the execution time allotted to this routine.

Synthetic sounds are represented in VAS using Timbre Trees [5], a representation developed specifically for use in computer animation. A Timbre Tree consists of a set of parameterized signal generating and signal modifying nodes arranged in a tree structure. A tree is evaluated by instantiating its parameters and traversing it in a pre-order fashion.

REAL-TIME SCHEDULING

The VAS Scheduler uses a real-time scheduling strategy for managing the sound generation process. Such a strategy is necessary when the correctness of a computation depends not only on the results produced but also on the timeliness of those results. The scheduling strategy used in VAS is based on the imprecise computation model. In a seminal paper Chung et.al [1] describe a technique for evaluating monotone processes using a model for imprecise computations. A monotone process is one that is guaranteed to produce increasingly accurate results as it is allowed to execute longer. The imprecise model partitions a task into a mandatory part and an optional part. The mandatory part is that required to produce results at the minimum acceptable precision. The mandatory task set is scheduled as a hard real-time task set so that the scheduler guarantees the execution of all the tasks in the set. Any remaining time in the schedule is used to schedule the optional parts of the task set.

The use of the imprecise computation model in the VAS scheduler forms a basis for our use of graceful degradation in the sound generation process. The scheduler initially assigns all active sounds enough execution time to generate a sound at the minimum acceptable resolution. Following that, the scheduler assigns each sound additional execution time in order to minimize some error metric. The scheduling algorithm determines how additional time is allocated to the optional part of the active sounds based on the available time in the schedule and the priority of the active sounds.

In the following sections we describe our technique for prioritizing sounds and for iteratively evaluating sounds. We then consider various scheduling algorithms based on their effect on the perceived quality of the generated sounds.

Prioritizing Sounds

In order to prioritize sounds in an SE we must determine which sounds the listener is attending to. This, of course, is impossible to do specifically since attention is subjective; the listener cognitively decides what to pay attention to. The best we can do is to attempt to guess what the listener may be paying attention to, based on the state of the listener and the state of the sounds in the environment. We use three factors to rate sounds in the environment: the listener's gaze, the intensity of the sound, and the age of the sound.

The use of the listener's gaze is based on the orienting response [2]. This is a human response to aural stimuli where the listener will attempt to support the perception of aural stimuli through visual correspondence. In effect listeners will turn their head so that they can see what they're listening to.

The intensity of a sound is important due to masking phenomenon [4]. A higher intensity sound will tend to mask a lower intensity sound if the two sounds are within the same frequency band. Determining the frequency content of a sound, however, requires a Fourier Transform operation that we opted against given the real-time constrains placed upon the system. We approximate this factor by simply giving louder sounds a higher scale factor.

The final scaling factor is based on the adaptation response of the human aural system [4]. Our sensitivity to aural stimuli decreases as the presence of the stimuli persists. This process continues for approximately three minutes after which it levels off. In effect, we adapt to persistent sounds in our environment making a sound's age important.

The problem of predicting a listener's attention from environmental factors is necessarily speculative due to the complexity of the human hearing process, especially when cross-modal perception is considered. We consider this approach as a starting point. Further work is necessary to study the effectiveness of this scheme with experimental data.

Iterative Generation of Sounds

The imprecise computation model requires that we iteratively evaluate a Timbre Tree at successively higher resolutions. If the iteration is stopped before full resolution is reached, the evaluation routine will not have generated all the samples. The missing samples are calculated using interpolation. To facilitate this, a buffer structure was devised which we call the *Interpolating Buffer* or *IBuffer*. The IBuffer consists of a set of mini-buffers in a stacked configuration as depicted in fig. 2. The depth of this stack determines the number of iterations necessary before full resolution is reached. On each iteration, the evaluation routine calculates a block of samples equivalent to 100 ms at 1/d of the global sampling rate, where d is the depth of the buffer. The resultant sample block is written to the IBuffer that fills one of its mini-buffers with the samples.

The order in which the mini-buffers are filled is predetermined by the IBuffer to minimize the number of empty mini-buffers between any two full mini-buffers. The starting time of the evaluation at each iteration is offset by an amount $p*d$,

where p is the period of the target sampling rate and d is the depth of the current mini-buffer. Once the evaluation routine has completed its last iteration, any empty mini-buffers are filled by linearly interpolating between the two closest full mini-buffers. The content of the mini-buffers is then copied to an output buffer in an interleaved fashion as shown in fig. 2.

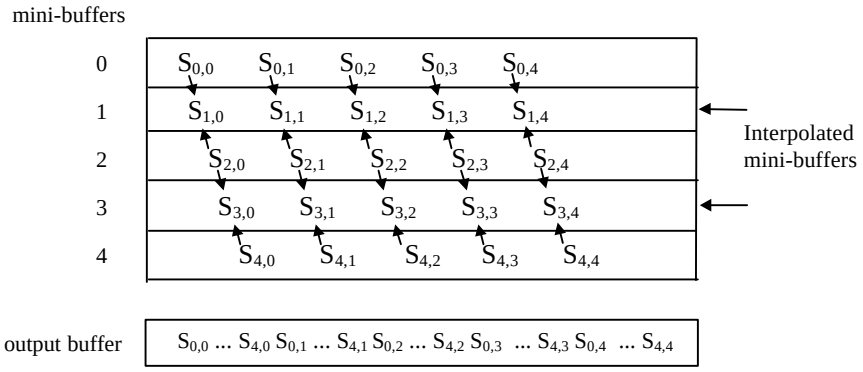

Figure 2 The interpolation step and output buffer. The IBuffer is depicted after three iterations. The samples in mini-buffers 0,2 and 4 are evaluated. Those in 1 and 3 are interpolated.

Scheduling Algorithms for Imprecise Computations

The final requirement necessary for a graceful degradation scheme is a scheduling algorithm for allocating execution time to the optional tasks in the schedule. While a number of scheduling algorithms have been devised for the imprecise computation model, the Least Utilization (LU) algorithm minimizes the average error given the characteristics of the sound evaluation problem [1]. The LU algorithm orders tasks based on their weighted utilization factor. This is the ratio of a task's priority and the percentage of the available time that is required by the task in order to be fully executed. Execution

time is then allocated to the ordered tasks in a greedy fashion where optional task O_1 receives as much of the available time as it requires. Any remaining time is allocated to task O_2 and so on.

The behavior of the LU algorithm poses a number of problems when the algorithm considered for use in scheduling sound evaluation routines. During overload conditions, large disparities in quality can occur between sounds that are close in priority. Furthermore, because the optional tasks are not ordered by priority but by the weighted utilization factor, the most important sounds may receive their minimum acceptable precision while less important sounds are fully evaluated. Finally a problem occurs when priority crossover occurs between two such sounds, where abrupt changes in the quality of the sounds occurs.

We can remedy one of these problems by simply ordering the optional tasks based only on their priority. This guarantees that the most important sounds will receive as much of their required time as possible. Such an algorithm was implemented and incorporated into VAS. The Priority Order (PO) orders the optional tasks based on their priority and assigns them execution time in a greedy fashion. As expected the PO algorithm remedies one of the problems mentioned above. The effects of the remaining problems are still pervasive in the resulting sound.

The problem with both the LU and PO algorithms lies in the large disparities that are possible in the execution time allotted to the set of active sounds. These disparities occur because of the greedy assignment of time to optional tasks. During overload a small number of sounds receive the majority of the execution time disregarding the priority of the remaining sounds. We quantify this behavior using a measure of the fairness of an algorithm. We define the average fairness index of an algorithm as a measure of its equity in assigning execution time to tasks based on their priority. A fair algorithm will assign each task a percentage of the available time based on its relative importance.

The Priority Allocation algorithm (PA) assigns each optional task an execution time that is approximately proportional to its priority. The actual time assigned to an optional task may be less than that indicated by its weight because the proportion of assigned time based on a task's priority may exceed the total execution time of the optional task. Variations also occur due to rounding of the assigned time to the nearest multiple of the iteration time of the sound. Any resulting free time in the schedule is assigned based on the LU algorithm. Having satisfied the fairness constraint, the LU algorithm assigns the remaining time in the schedule so that the average error is reduced.

SUBJECTIVE EVALUATION OF SCHEDULING ALGORITHMS

A listener study was conducted in order to test the perceptible effects of degradation when the LU, PO and PA algorithms were used. We chose a test developed by the Swedish Broadcasting Corporation and used by the ISO MPEG/Audio in the establishment of the international MPEG standard for storage and retieval of moving picutures on digital media [3]. The method employed is a Triple Stimulus, Hidden Reference, Double Blind test. Subjects are presented with three items A-B-C. Each item consists of an audio segment. Item A is always the reference. Items B and C contain the object and a hidden reference. Because the test utilizes a hidden reference design, the subjects do not know which of B and C are the hidden reference. Subjects rate the amount of imparement detected between items A-B and A-C using a five point, continuous scale with one decimal.

In preparing the test, twelve audio sequences were generated each consisting of the three A-B-C items. The items where generated by recording the result of evaluating two SEs which we refer to as City and Waves. The SEs were designed to exhibit dynamic behaivior and impose varying degrees of load on the system. In general, the Waves SE imposes much heavier load conditions. In order to generate the reference items, the SEs were evaluated using two processors so that degradation did not occur. In generating the object items, the sounds were evaluated using only one processor resulting in degradation.

Twenty subjects were presented with the generated sequences and asked to rate each item B and C in comparison to A. The sounds were presented to the subjects using loud speakers in a group session with 10 subjects participating in each session. The subjects were given a practice session so that they were familiar with the procedure.

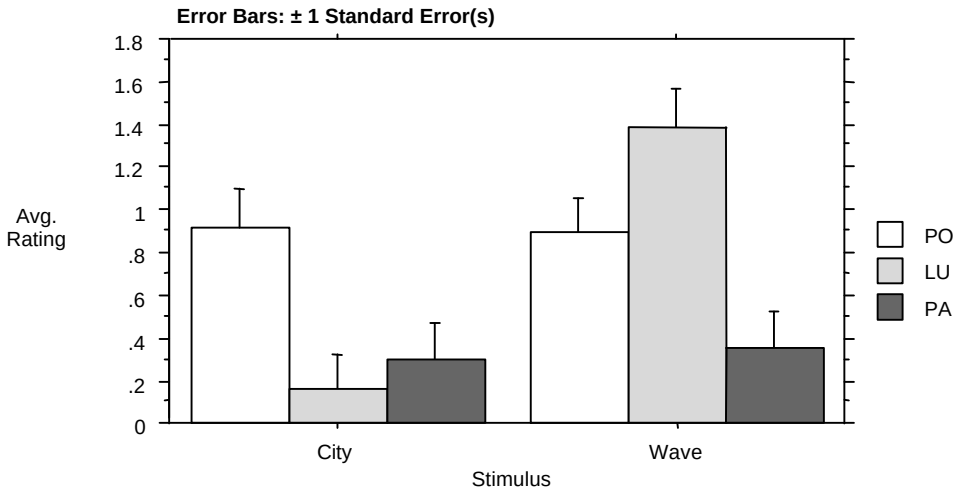

Figure 3 Interaction bar plot for Algorithm effect by stimulus. The difference between the ratings for the hidden reference and the object are used. Lower numbers indicate a better rating.

The effect of the algorithm by stimulus is shown in fig. 3. The results showed that when the system was slightly overloaded, the average error dominated and there was no significant difference in the rating of the LU and PA algorithms. Under heavy overload conditions however, the fairness of the algorithms came into play and the PA algorithm received significantly higher ratings.

CONCLUSION

The use of real-time scheduling techniques enables us to manage the sound generation process. When this process is expressed in terms of the imprecise computation model, graceful degradation can be enacted when computational resources are exceeded. The Priority Allocation algorithm described in this paper assigns execution time to sounds so that the perceptible effects of overload are minimized.

REFERENCES

1. J. Y. Chung, J.W.S Liu, and K. J. Lin, *Scheduling Real-Time, Periodic Jobs Using Imprecise Results*, Proc IEEE RTS, 1987.
2. S. Coren, C. Porac, and L. M. Ward, *Sensation and Perception 2nd ed*., Academic Press, Orlando, Florida, 1984.
3. C. Grewin, *Methods for Quality Assessment of Low Bit-Rate Audio Codecs*. In Proc. of the 12th International AES Conference, pp. 97-108, June 1993.
4. S. Handel, *Listening*, The MIT Press, Cambridge Massachusetts, 1989.
5. T. Takala, J. Hahn, L. Gritz, J. Geigel, J.W. Lee, *Using Physically-based Models and Genetic Algorithms for Functional Composition of Sound Signals, Synchronized to Animated Motion*, International Computer Music Conference, September 1993.
6. J. V. Tobias, *Foundation of Modern Auditory Thoery*, vol. 1, Academic Press, NY and London, 1970.