Software modules for HRTF based dynamic spatialisation

Ignacio Sánchez and Jesús Bescós
Grupo de Tratamiento de imágenes, Universidad Politécnica de Madrid
jbc@gti.upm.es, http://www.gti.ssr.upm.es

Abstract

This paper describes the object oriented design and development of software modules intended to enhance multimedia presentations with sound sources spatialisation, and environmental effects (reverberation), allowing dynamic reconfiguration of the input sound parameters. Implementations have been carried out on a PC platform, on top of the Win32 API. The resulting modules (in fact C++ classes) have later been integrated into a working application for demonstration purposes.

1 Introduction

The application of digital sound into computers has recently exploded due to the high availability of reasonable quality low cost equipment, mainly promoted by the entertainment market.

In particular, the manipulation of sound in a virtual space, a research field traditionally *owned* by psychoacusticians and musicians, *hidden* beyond university labs, and showed via prototypes, has just recently arrived to end users.

In this direction, the deployment of VRML 97 [1] has brought sound to the VRML 1.0 silent virtual worlds, and has definitely standardised a model for 3D sound, via the inclusion of sound nodes into the virtual worlds. Once the model was set up, we only had to *wait* for implementations. By April 1997, Intel presented the "first all-software solution for interactive 3D sound and effects": their Realistic Sound Experience (3D RSX) development kit [3]. Among many other features, they provided 3D sound spatialisation via the use of the HRTF measures available at the MIT [3]

The work presented in this paper was motivated by the successive implementation steps of a model defined in the research Group to generalise real time multimedia data flow handling [4]. As developments were carried out previous to the arrival of the aforementioned 3D sound technologies, the coded libraries have been fully implemented, that is, they are not based on any third party commercial library.
The objective was to create software modules intended to enhance multimedia presentations with sound sources spatialisation, and environmental effects (reverberation). The remaining sections describe first the design principles, execution environment, and final implementation of such modules, and then, a sample application developed to test the achieved results.

2 Design and development

According to the kind of multimedia applications that we are typically involved in, to basic object oriented programming principles, and to the state of the art in this researching field, the design was based on certain well defined requirements.

In order to limit the complexity of this first approach, accepted sound sources should consist of omnidirectional ones, whose response is stored in monaural PCM coded sound files, which show typical values for the sampling rate and bits-per-sample parameters.

Spatialisation should be based on both the low quality and high quality generic HRTF measures available at the MIT (also available headphone compensation filters have been used for the testing equipment).

Depending on the source binary rate and on the selected filters complexity, signal processing should be performed either with a discrete temporal convolution or via an FFT, in order to achieve real time operation.

The design should allow for dynamic modification of the spatialisation parameters (azimuth, elevation), and hence real time replacement of the localisation filters via a multithreaded scheme (in order to avoid replacement delays and/or undesired sound artifacts).
Data outputs should consider either PCM WAV files, or direct playback via an MCI compliant sound card. Of course, this second requirement clearly imposes some limitations on the working environment.

In order to achieve greater realism, also basic reverberation effects should be introduced.

As the final objective of these sound processing modules is to assemble them into large specific applications (entertainment, presentations, sound aided physically impaired integration, etc.), the design of the required functionality should focus on modularity, and try to be accessible via friendly APIs. Finally, special attention should be paid to minimise execution times of every atomic operation, in order to allow real time performance even when multiplexing several independent sound sources.

2.1 Environment

The answer to a low cost and efficient processing environment which allowed for real multithreading, standard I/O devices (specially sound cards) management libraries, easy access to high resolution timing resources, and availability of efficient programming environments, was to work over Windows NT OS, its MCI (Media Control Interface), and its Multimedia Extensions.

2.2 Overall description

In order to directly code the module oriented approach adopted for the design phase, we have implemented all modules as C++ classes. *Figure 1* depicts a functional diagram of the involved modules or classes.

The basic module, the Filtering Module consists of a general purpose (configurable) unidimensional signals digital filter, implemented in a *CFilter* class.

Two of these filter modules (one for the right channel and another for the right one) are combined to generate the main module, the Spatialisation Module, represented by the *CHrtf* class. This module copes with the management of the HRTF library.

Specific add-ons, intended to carry out the reverberation processes, enlarge the previous module to compose the Acoustic Environment Simulation Module, represented by the *CEnhancedHrtf* class.
Finally, another layer, the I/O one is added to the latter, hence conforming an independent self-contained object, represented by the *CSound_3D* class, the final interface with the application developer.

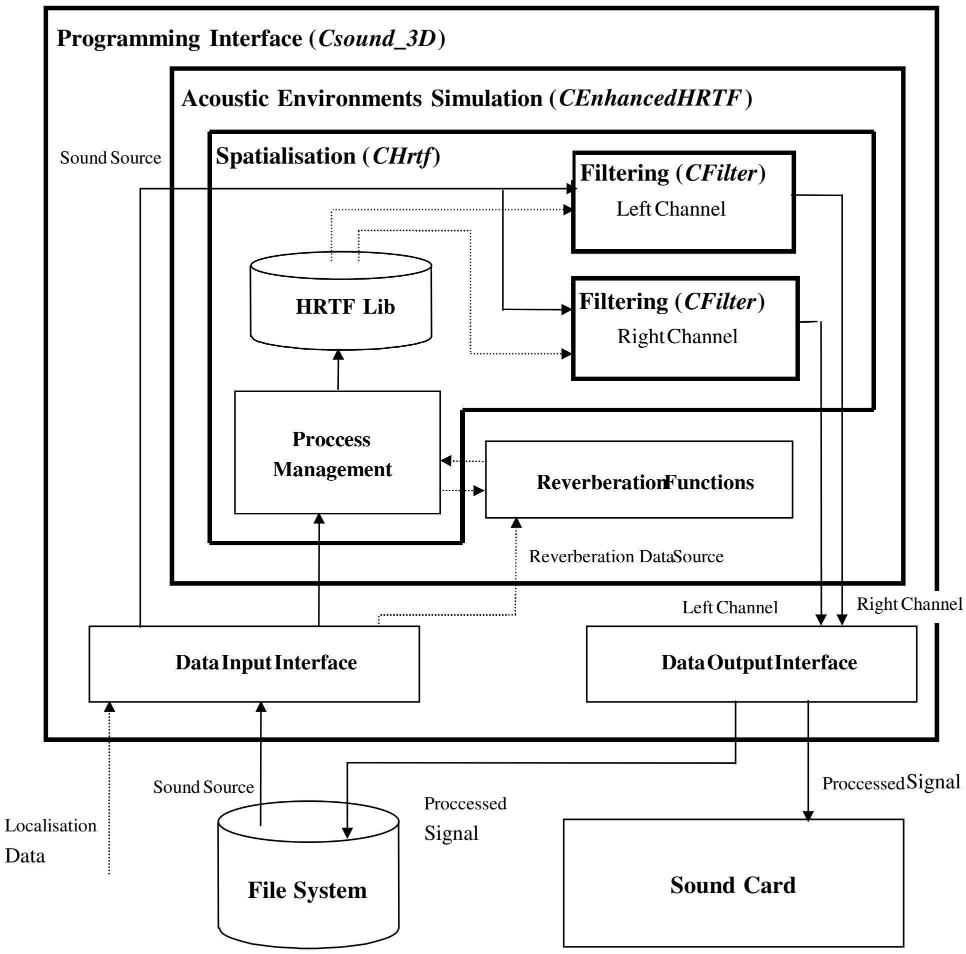

Figure 1. Functional diagram showing the relationship among the implemented modules/classes

2.3 Implementation details

As specified, the filtering module configuration parameters are the impulse response, its sample rate, the convolution domain (either linear convolution in the time domain via buffers and delays, or block convolution in the frequency domain via an FFT), and an optional scaling factor included here for efficiency reasons). The latter could be automatically selected (currently, this is done by the application developer) to maximise efficiency and minimise delay, depending on the length of the finite impulse response. Once the filter is configured input/output sample/block functions are available.

A particular feature of the Spatialisation Module is

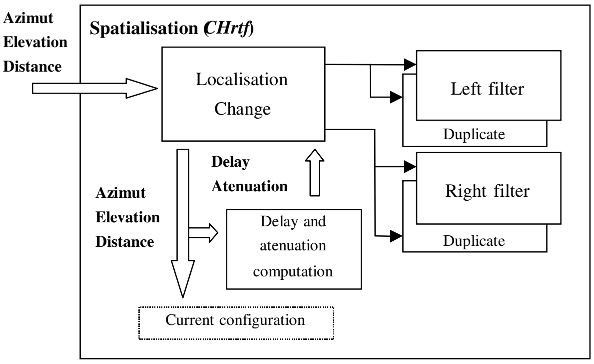

Figure 2. CHrtf functional diagram

that the localisation parameters (azimuth, elevation, and distance) can be changed on-line. A parallel thread takes the responsibility to recalculate values for all the internal structures and filters, and then *quickly* replace them, so that the normal input/output operation is minimally disturbed (i. e., non heard), and that the operation is not much delayed. The aim was to allow dynamic real time spatialisation (monaural sound source plus a defined trajectory) as opposed to a statically processed sequence (binaural sound source).

Data used for spatialisation comes from the work carried out by Bill Gardner and Keith Martin at the MIT [3]. Their measured HRTFs with an original length of 16383 samples have been pre-processed (reduced to 512 and 128 long impulse responses --which correspond to two different available quality levels--, and ordered for efficient dynamic operation) and kept in an HRTF Lib (a data file).

Distance is simulated just via independent intensity scaling (via the use of the inverse square rule relative to the *egocentric centre*) of the signal applied to the right and left channels.

Reverberation is achieved via convolution with an impulse response, previously calculated as the addition of a first order ray tracing process (for early reflections) applied to thethraedric rooms, plus a random exponentially decreasing sample sequence uncorrelated between left and right channels (for late reverberation). This algorithmic solution for geometrically simple environments can be replaced, for more complex scenarios, by the direct introduction of a measured impulse response, via manual configuration of the reverberation filters.

3 Future work

Regarding the filtering process, optimisation of the FFT implementation (which it is currently fairly optimised) could be further achieved (20-30% speed increase) via any of the reported *divide and conquer* algorithms, which take profit of symmetries with impulse responses whose length is a power of two.

However, a more important problem for real-time applications is the time delay (greater as the quality, that is, the impulse response length, increases) inherent to the FFT algorithm. To solve it, some authors [5] have proposed hybrid (time domain - frequency domain) convolution.

Regarding the spatialisation process, in order to achieve greater realism of the simulated sound trajectories, a finer resolution for available HRTF measures would be desired. This could be performed via lineal interpolation of the available data.

Open space distance simulation should consider the effect of the medium, characterised as a function of the sound frequency. Also the doppler shift should be taken into account if we deal with large distance variations.

Finally, reverberation in non controlled synthetic environments (that is, when a measured impulse response is not available) should consider the frequency dependent effect of the medium and the reflecting surfaces.

4 Test application

In order to demonstrate in a practical way the functionality and performance of the developed modules, a MS Windows test application has been developed using the provided capabilities.

Figure 3 shows the user interface of the application. It has been designed so that all the basic aspects of 3D sound and reverberation are covered.

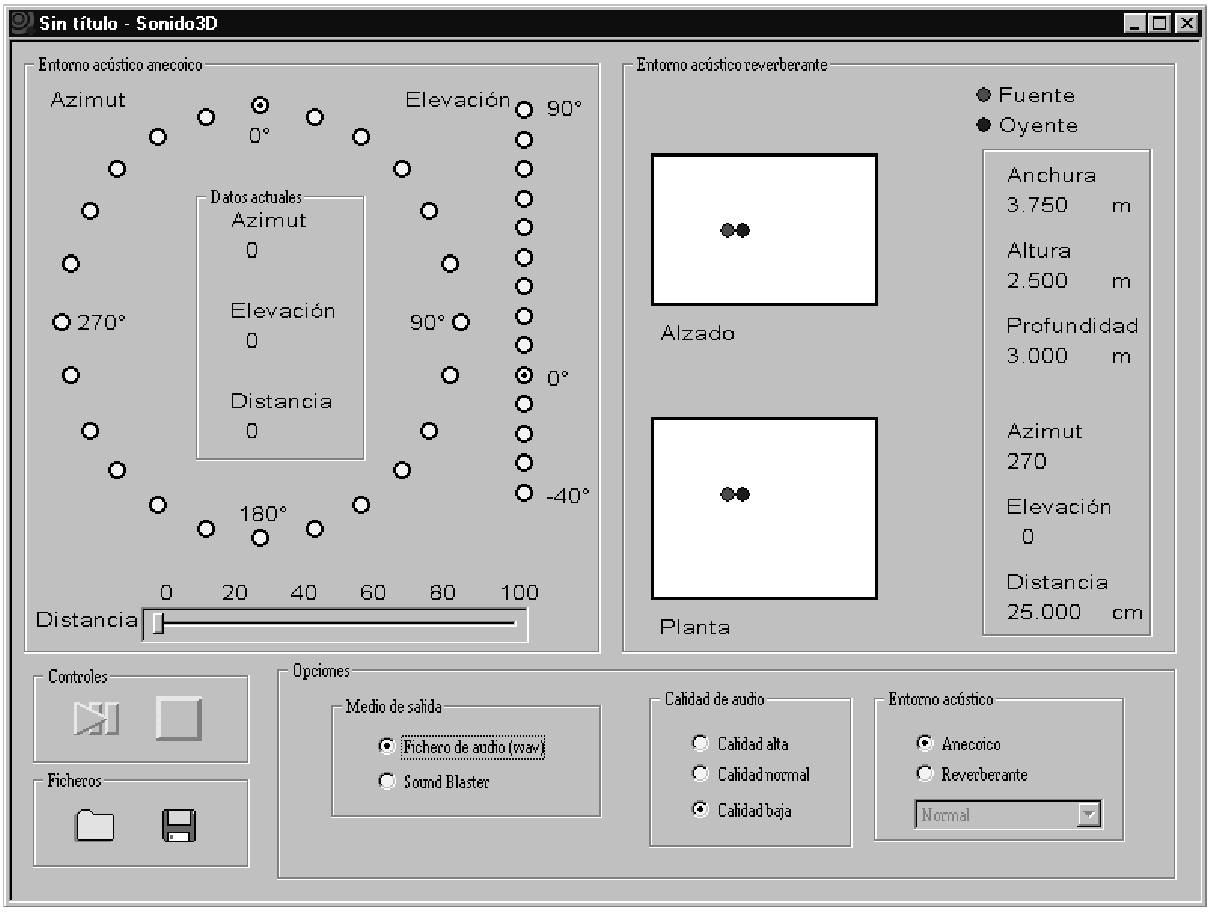

Figure 3. A screen capture of the test application

The sound source must be obtained from a PCM *.wav* file, while the spatialised output can be either saved to disk or directly redirected to an MCI compatible sound card.

The possibility to dynamically change the location of the sound source respect to the listener, can be applied in two different reverberant environments: an artificial anecoic acoustic environment (on the left half of the screen) or a reverberant one (on the right). Always limited by the input quality, it is possible to select the quality of the processed sound output (high, middle and low). Also the reflection index of the surfaces in the acoustic reverberating environment can be controlled.

5 Conclusions

The described work provides a set of ready-to-use and easy-to-use classes, intended to add sound spatialisation features to multimedia applications.

More than a contribution to 3D sound specifications and algorithms, this shows a working implementation of a dynamically configurable system on a low cost platform.

References

[1] VRML97, ISO/IEC 14772-1:1997. *The Virtual Reality Modelling Language*

[2] INTEL RSX (Realistic Sound Experience), at http://developer.intel.com/ial/rsx/

[3] W.G. Gardner, and K. Martin, "HRTF Measurements of a KEMAR", *J. Acoust. Soc. Am*, 97(6), pp 3907-3908, 1995.

[4] J. Bescós, et al., "From Multimedia Stream models to GUI generation", *Proceedings of the SPIE Conference on Voice, Video and Data Communications*, Vol. 2495, pp 136-146, Boston, November 1996.

[5] W.G. Gardner, "Efficient convolution without input-output Delay", *Presented at the 97th convention of the Audio Engineering Society,* San Francisco. Pre-print 3897, 1994