MULTIBIN: A BINAURAL AUDITION TOOL

Brian Carty *Victor Lazzarini*

Sound and Digital Music Technology Group,
National University of Ireland, Maynooth
Co. Kildare, Ireland

brian.m.carty@nuim.ie victor.lazzarini@nuim.ie

ABSTRACT

MultiBin is a new tool for binaural audition of multiple sound sources in a user definable environment. Although designed to be flexible in its application, its primary function is to provide dynamic multi-channel binaural simulation. It is built upon two new Csound binaural reverberation opcodes. An early reflection opcode, based on an image source method and a Head-Related Transfer Function interpolation algorithm previously introduced by the authors provides dynamic source and listener location. This is complemented by a later reverberation opcode which provides a diffuse reverb based on a parametric Feedback Delay Network model which considers interaural coherence.

1. INTRODUCTION

Head-Related Transfer Functions (HRTFs) are essentially frequency domain functions which describe how a sound is altered from a particular source location to the ear [1]. Pairs of HRTFs inherently consider sound localisation cues such as interaural differences and spectral transformations. HRTFs are typically used in binaural processing tools, and can be used to artificially spatialise sound in a virtual listening environment [1, 2]. Typically, environmental processing is desirable in this scenario. Reverberation thus needs to be considered [2, 3].

This paper discusses issues involved with the development of a flexible binaural audition tool, MultiBin. Each key aspect of the application is dealt with in turn. The first challenge is providing smooth, artefact free source trajectories: dynamic HRTF processing. Binaural early reflections will then be discussed in the context of HRTF spatialisation. A later reverberant tail which considers interaural coherence completes the environmental processing. Finally, an intuitive Python GUI is outlined.

2. CSOUND OPCODES

MultiBin uses the Csound API to allow Python to send dynamic, user generated information to an instance of Csound from the host application [4]. Csound thus deals with the low level DSP required to dynamically represent sound sources and a listener in a user defined sonic environment. The relevant Csound opcodes and their background will now be discussed.

2.1. HRTF Processing

Two novel approaches to HRTF interpolation and dynamic source trajectory processing were recently introduced by the authors. The approaches, as well as background and validation of the algorithms are discussed in previous publications [5, 6]. Briefly, the algorithms aim to provide accurate, efficient HRTF processing while minimising data analysis, compression or transformation. The algorithms developed are realised in the Csound opcodes `hrtfmove`, `hrtfmove2` and `hrtfstat` [4].

`hrtfmove` employs magnitude interpolation and phase truncation to allow dynamic spatialisation of input audio using overlap add convolution processing. Alternatively, a more traditional minimum phase plus delay process is implemented by this opcode. An optional flag allows the user to switch processing modes.

`hrtfmove2` takes a more functional approach, augmenting a spherical head model for interaural phase with low frequency scaling factors based on empirical HRTF data. This psychoacoustically motivated approach models phase in accordance with the limitations and frequency dependant nature of the auditory localisation system [7, 8]. `hrtfstat` implements the same algorithm, but exploits potential optimisations for static source processing.

The algorithms (Minimum Phase-based processing, Phase Truncation, the Augmented Spherical Head and an anchor condition with no interpolation) were tested subjectively with regard to ability to provide smooth, artefact free source trajectories. The novel algorithms performed extremely well. Interestingly, the results also highlight both the need for interpolation (the anchor condition performed in the poor-fair range) and the good performance but potential problems with minimum phase processing [7].

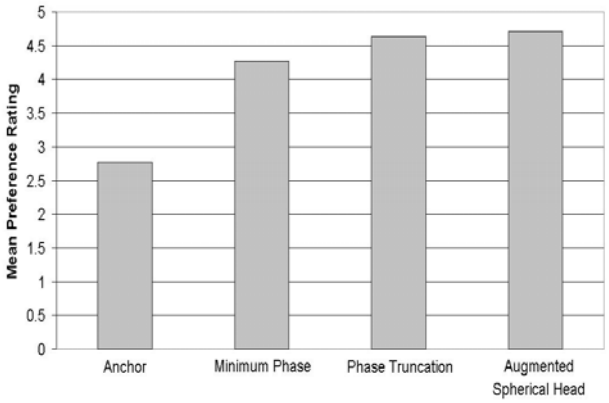

Figure 1: *Preference Test Results.*

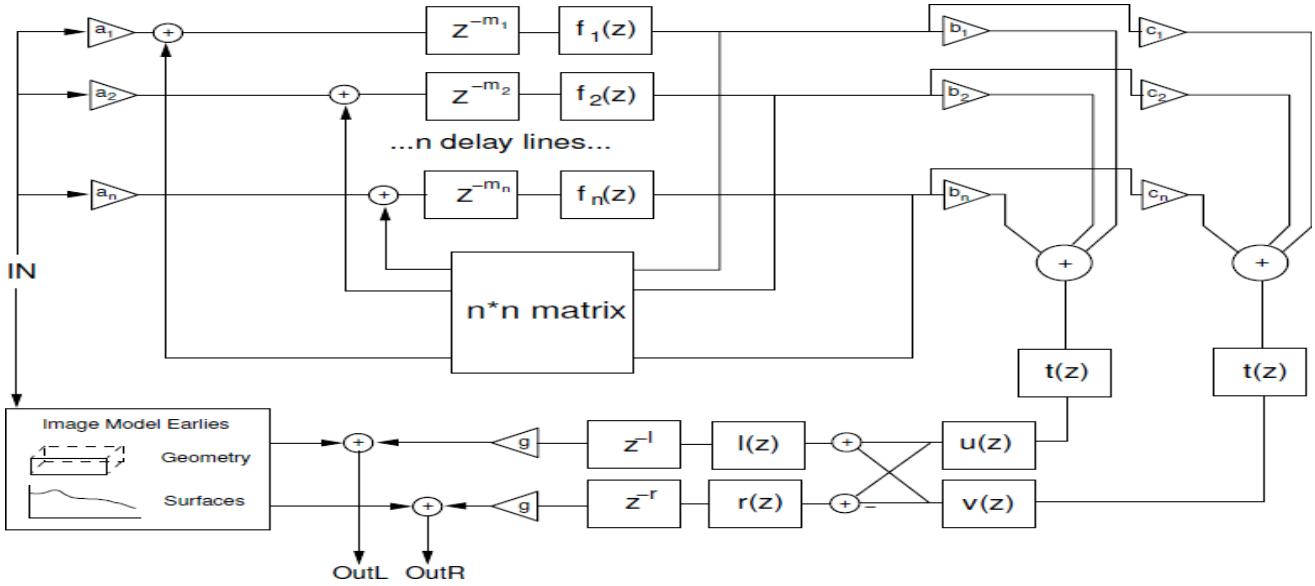

Figure 2: *Schematic of the Reverb Model.*

2.2. Early Reflections

MultiBin does not use the HRTF opcodes directly, but re-implements the Phase Truncation algorithm for early reflection processing. The success of the algorithm in subjective tests and its efficiency make it a suitable candidate. A brief code example is perhaps an appropriate introduction to the opcode.

```
aearlyl, aearlyr, ilow, ihigh, imfp
              hrtfearly ain, srcx, srcy, srcz,
                        lx, ly, lz, "datal.raw",
                        "datar.raw", 1
```

The opcode, `hrtfearly`, outputs the left and right processed audio, a low and high frequency reverb time, calculated using the Norris-Eyring reverb formula, and the mean free path for the room in question. The latter 3 values are intended to be used as inputs to the later reverberant field opcode, discussed below. Dynamic, control rate source and listener x, y and z geometric location values are the main inputs, following the mono audio input. Left and right HRTF data files are the next arguments. Finally, a default room can be chosen: small, medium or large. An image source model [9, 10] is used to process the early reflections dynamically.

Optional parameters for more advanced use are also available. These include the Phase Truncation fade length [5, 6], the sampling rate, the order of reflection processing, whether floor and ceiling reflections are considered, a dynamic head rotation value, and room parameters. The size of the room, as well as the high and low frequency absorption coefficients and parameters of 3 band pass filters for each surface can be set.

Processing a number of sources, each with a (user definable) number of reflections can quickly become computationally costly, so optimisation is crucial. Interpolation is only performed if source orientation with respect to the listener changes. . Memory allocation, Fourier Transform Processing and dynamic source trajectory processing are also optimised.

2.3. Later Reverb

The opcode `hrtfreverb` is a later, reverberant field unit generator, which employs a dynamic Feedback Delay Network (FDN) and various filters to process the output binaurally [10]. The opcode builds on the interaural coherence [11] addition to the Jot FDN [12] by considering the parametric scenario, as opposed to measured impulses. It also considers flexible early reflection processing.

The FDN is illustrated in Figure 2, as well as frequency dependent reverb filters ($f_n(z)$), compensating tone correction filters (t(z)), vectors to ensure 2 uncorrelated output (**b** and **c**), coherence matching filters (u(z), v(z)), average HRTF filters (l(z), r(z)) appropriate delay and gain factors for the later reverberant tail and finally early reflection processing (discussed in more detail in [10]).

Once again, the opcode can be initialized and used in a flexible manner:

```
alatel, alater, idel hrtfreverb ain, ilow, ihigh,
                                "datal.raw","datar.raw"
```

Outputs are the left and right channels of the binaural reverberant tail and its appropriate delay time. The low and high frequency reverb time [12], HRTF data files and audio input are the only required inputs. Sampling rate, mean free path and order of early reflection processing are optional inputs. The latter 2 arguments are used to derive the appropriate delay time for the late reverberant field.

3. MULTIBIN

The MultiBin application is built on the above binaural reverberation opcodes. Essentially, the goal of the application is to allow flexible virtual spatial environments, with a particular focus on virtual multi-channel (sources constituting loudspeakers). Figure 3 illustrates a typical instance of MultiBin. Upon running the application, the canvas [13] shows a centred listener with no sources in their sonic environment. A default medium

room is setup. Users can add sources to the canvas in a simple and flexible way. Each source is numbered, which links it to a specific Csound channel. Sources can be added and removed using simple menu options. Therefore, the user may setup a flexible virtual environment.

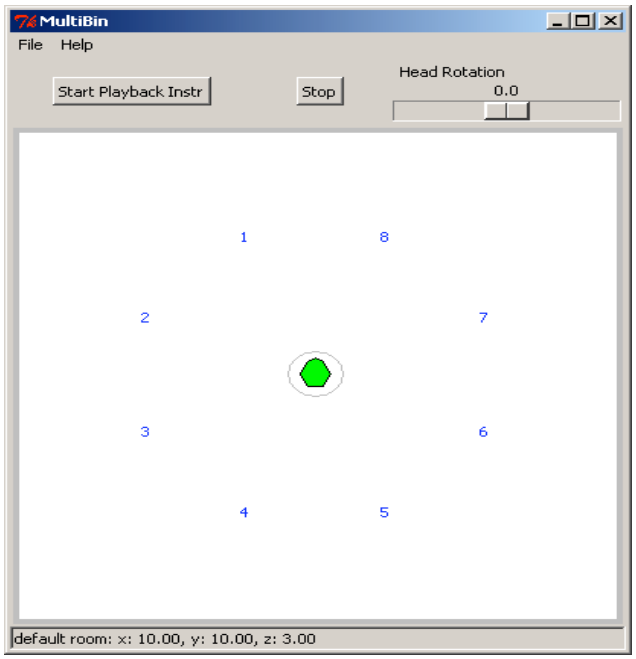

Figure 3: *A Typical Instance of MultiBin.*

Sources can be moved around the environment by selecting and dragging. The HRTF interpolation algorithms discussed above allow for real time realisation of these movements. The listener may also move using similar intuitive user control. Head rotation is also implemented. Playback of the Csound controlled source material is also a simple and intuitive process (play and stop buttons). The File menu allows a user to setup an 8 channel ambisonic room, as well as a similar VBAP setup. The parameters of these loudspeaker setups are based on the existing Csound opcodes for ambisonic and VBAP processing. All sources can also be easily cleared.

A user may therefore audition a multi-channel file in headphones. The approach taken here differs from previous approaches that use sweet spot virtual multi-channel binaural processing [14]. The user may move out of the sweet spot to audition off centre listening. Perhaps a loudspeaker in a specific room cannot be physically placed at the optimum location. The loudspeaker can be dynamically moved in real time to audition any potential problems. Any multi-channel algorithm can thus be auditioned in a dynamic fashion. The ambisonic and VBAP opcodes in Csound allow for convenient preparation of appropriate source material, motivating their inclusion as defaults. Equally, however, other setups, such as simple Wave Field Synthesis arrays could be auditioned.

From a design point of view, the system allows optimal dynamic multi-channel audition, but also caters for other virtual spatial applications. The ease with which sources can be added, removed and dragged around the canvas creates a flexible and creative workspace for spatial audio. Implementation of the application will now be discussed in more detail.

3.1. Implementation Detail

The Tkinter Python module [13], which uses the Tk GUI toolkit is the cross platform tool used to create an appropriate GUI for the MultiBin application. A simple Csound file is controlled by sending user triggered messages from the GUI. This Csound file consists of a playback instrument, which essentially plays back the source audio. Csound offers several possible options to do this, including playing back sound samples stored in tables, or directly reading multi-channel files. Each stream of audio is played back on a numerically labelled channel. A global parameter reading instrument reads head position in the GUI.

Reverberation processing is then dealt with. An early reflection instrument is assigned to each source, reading the dynamic source trajectory, and taking parameters from user inputs. When the user adds a new source, it appears on the canvas of the GUI as a number which relates to the channel it is reading from. Therefore, the user has direct control over the audio linked to each source. Finally, a late reverb instrument processes all sources using the `hrtfreverb` opcode. Reverb time and room outputs of `hrtfearly` can be used as inputs to `hrtfreverb`.

The Python code defines the GUI, which will now be discussed. The main interactive element of the application is the canvas widget, which allows user input and control. The application is designed as a class, with methods for movement of objects on the canvas and other control functions. Control of items on the canvas is maintained using the Tkinter item methods. The application class constructor initiates an instance of Csound, whose parameters are updated by the user. For example, if a source is moved, updated x and y values are sent to Csound, which updates the location of the source relative to the listener at the control rate. The constructor also sets up the menus and GUI, initialises variables and binds class methods/callbacks to user operations/events, such as mouse button clicks.

A number of methods are used to allow for adding sources in a well defined manner. A generic function adds an item to the canvas, increments the control variable which keeps track of the number of active sources and turns on an instance of the `hrtfearly` instrument. A second method calls this generic method, and deals with the user defined location of the source, using data from the dialog window illustrated in figure 4. This method also ensures the source location is legal (inside the defined room). Location data is stored in an instance of another class, which defines the location dialog window and data. As can be seen in figure 4, direct pixel location can be entered, or a polar approach can be taken, motivated by typical multi-channel setups. An angle and distance from room centre can be entered to define source location. The tkSimpleDialog module is used to create the source location class/dialog window.

Figure 5 shows the ‘new scene’ dialog window, similarly realised as a class developed using the tkSimpleDialog module. Users can either simply define the room size, or choose to enable complex parameters, and enter surface parameters for high and low frequency reverb time and wall response band pass filter gain factors. The canvas will always be 400 pixels across, with room geometry ratios dictating the height. The size of the room dictates the size of the grey circle centred at the listener position. Inside this circle, HRTF processing is not performed, as near field HRTF modelling is not considered [15].

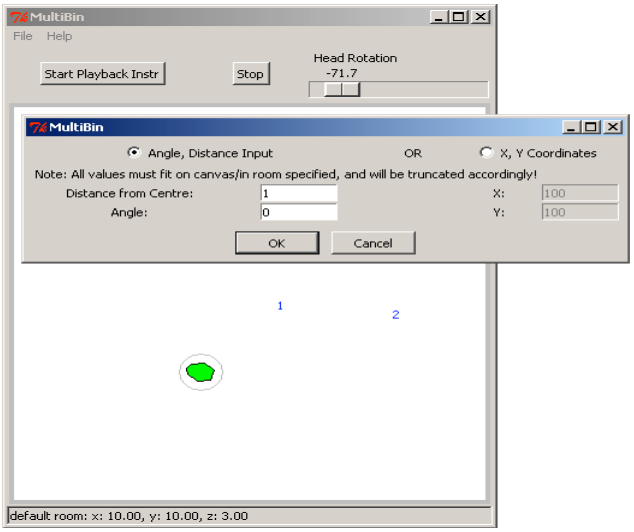

Figure 4: *New Source Dialog.*

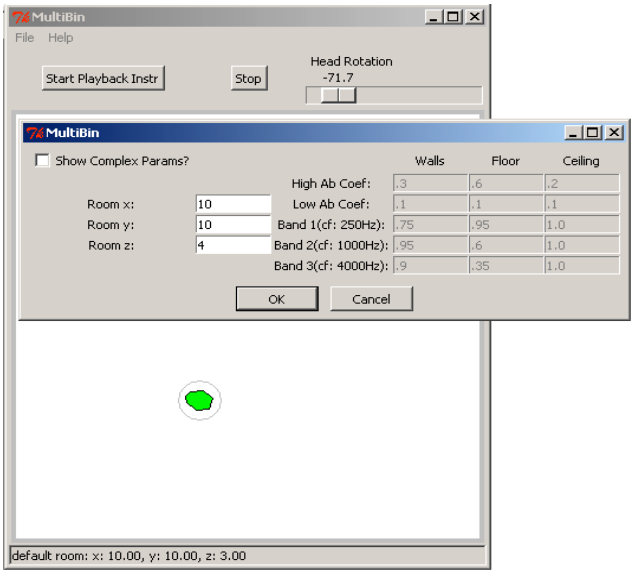

Figure 5: *New Scene Dialog.*

4. DISCUSSION & APPLICATIONS

Multibin is designed to be a flexible binaural tool, allowing dynamic source and listener behaviour in user definable virtual environments. Its primary intention as a binaural multi-channel audition application allows flexible listener scenarios. Unlike sweet spot multi-channel binaural approaches [14], off centre listening is purposefully allowed and indeed intended, as the listener may move around the sonic environment.

Flexible loudspeaker setups are also possible, allowing real world audition. It is also important to highlight that MultiBin does not aim to optimise multi-channel binaural processing [16]; its main design goal is to allow complete user control. Standard binaural limitations apply, including HRTF individualisation. Real time performance is prioritised over additional processing such as source directivity.

5. CONCLUSIONS

MultiBin is a flexible tool developed using Python and the Csound API. Csound binaural reverberation opcodes are utilized, which are based on previous HRTF interpolation research by the authors. MultiBin allows dynamic control over user definable virtual environments, with specific application to virtual multi-channel.

6. ACKNOWLEDGMENTS

This research is supported by the Irish Research Council for Science, Engineering and Technology: funded by the National Development Plan and NUI Maynooth.

7. REFERENCES

[1] Begault, D., *3-D Sound for Virtual Reality and Multimedia*, Maryland: NASA, 2000.

[2] Jot, J., Larcher, V. and Warusfel, O. “Digital Signal Processing Issues in the Context of Binaural and Transaural Stereophony,” *98th AES Convention*, Paris, France, 1995.

[3] Savioja, L., Huopaniemi, J., Lokki, T. and Väänänen, R. “Creating Interactive Acoustic Environments,” *JAES*, 47(9), pp. 675-705, 1999.

[4] http://www.csounds.com.

[5] Carty, B. and Lazzarini, V. “Binaural HRTF Based Spatialisation: New Approaches and Implementation,” *Proc. DAFx*, Como, Italy, pp. 49-54, September 2009.

[6] Carty, B. and Lazzarini, V. “Frequency-domain Interpolation of Empirical HRTF Data,” *126th AES Convention*, Munich, Germany, May 2009.

[7] Kulkarni, A., Isabelle, S. and Colburn, H., “Sensitivity of Human Subjects to Head-Related Transfer-Function Phase Spectra,” *JASA*, 105(5), pp. 2821-2840, May 1999.

[8] G. Kuhn, “Model for the interaural time difference in the azimuthal plane,” *JASA*, 62(1), pp.157-167, July 1977.

[9] Allen, J. and Berkley, D. “Image method for efficiently simulating small-room acoustics,” *JASA*, 65(4), pp. 943-950, 1979.

[10] Carty, B. and Lazzarini, V. “hrtfearly & hrtfreverb: Flexible Binaural Reverberation Processing,” *ICMC*, New York, USA, June 2010.

[11] Menzer, F. and Faller, C. “Binaural reverberation using a modified Jot reverberator with frequency-dependent interaural coherence matching,” *126th AES Convention*, Munich, Germany, May 2009.

[12] Jot, J. “Digital delay networks for designing artificial reverberators,” *90th AES Convention*, Paris, France, 1991.

[13] http://www.pythonware.com/library/tkinter/introduction/index.htm

[14] Noisternig, Musil, Sontacchi and Holdrich. 3D Binaural Sound Reproduction using a Virtual Ambisonic Approach. *IEEE Symposium on Virtual Environments*, pp. 174-178, 2003.

[15] Duda, R. and Martens, W., “Range dependence of the response of a spherical head model,” *JASA*, 104(5), pp. 3048-3058, 1998.

[16] Goodwin, M. and Jot, J., “Binaural 3-D audio rendering based on spatial audio scene coding,” *123rd AES Convention*, New York, USA, October 2007.