Phya and VFoley, Physically Motivated Audio for Virtual Environments

Dylan Menzies[1]

[1]*De Montfort University, Leicester, UK, LE1 9BH*

Correspondence should be addressed to Dylan Menzies (rdmg at dmu.ac.uk)

ABSTRACT

Phya is an open source C++ library that facilitates physically motivated audio in virtual environments. A review is presented and recent developments in the context of game audio, including the launch of VFoley, a project using Phya as the basis for a fully fledged virtual sound design environment. This will enable sound designers to rapidly produce rich Foley content from within a virtual environment, and author enhanced objects for use by Phya enabled applications.

1. INTRODUCTION

Triggered or looped sample playback has been the dominant method of generating interactive sound in computer games. As game environments become more sophisticated, the gap between graphical and audio content has become more apparent. Natural sounds are produced sympathetically to the continuous dynamical interactions that drive them, and vary continuously. To some extent this can be emulated by using variations of samples, blending and filtering them. Physical modeling can capture the complexity of natural sound objects, at the cost of increased complexity. Processing time may or may not suffer, as memory access to read samples is itself costly. Physically modelled sound synthesis was developed initially to simulated musical instruments. In [1], a new surface contact model was linked with a modal resonator simulating object resonance. The inputs to this system are the forces and velocities of contacts between physically simulated objects. It becomes very clear that the careful marriage of macro object dynamics with audio rate interactions accounts impressively for the overall sound. Furthermore, graphical cues which accompany sonic interactions, become more meaningful, because they are in greater sympathy. This work has provided a new direction for developing convincing audio for interactive scenes containing physical objects that can be manipulated in the virtual world or game. Around the same time comprehensive general purpose physics engines were starting to appear, primarily for games but also simulations. These engines are now providing complex physical behavior in games, and have also paved the way for interactive sound that exploits them. Some glue code is required between Phya and the engine, but this is minimized through the structuring of Phya and additional utility functions. Phya is currently demonstrated using the excellent open source Bullet engine[1]. Similar glue code will be required for other engines.

Phya[2], [2], is a project with two main goals, the first being to provide a framework for facilitating interactive sound generation using a physics engine as the control source. The second is to develop sound models, to generate new kinds of sound as efficiently and practically as possible. These two facets appear at first well separated, however for best results they should be considered together, along with the limitations of the physics engine. Engines vary in their behavior, and in any case were designed for graphical effects not audio. For our practical purposes the purpose of physical modeling is to elegantly capture the most relevant parts of a system, rather than every detail. For Phya, the macro dynamics of the physics engine provides a common physical element. The original intention was that the sound models are physically focused as well. Relaxing this focus opens up a more creative approach, that is particularly suitable in a game context, while still being grounded in the overall macro dynamical behavior. This is the meaning of the phrase *Physically motivated audio*.

There follows an overview of structure, and operating system of Phya, and then a review of the sound models.

[1] www.bulletphysics.com

[2] Details at www.cse.dmu.ac.uk/ dylan

2. FRAMEWORK

2.1. Core objects

Phya is built in C++, and is based around a core set of classes, that can specialized and extended. Each sounding object is represented by a *Body* object, which points to an associated *Surface* and *Resonator* object, see Figure 1. Bodies can share the same Surface and Resonator if required in order to handle groups of objects more efficiently. Collisions states are represented using *Impact* and *Contact* objects that are dynamically created and released as collisions occur between physical objects. Impacts are intended to be created by a significant normal component of contact impulse that would normally be caused by an impact collision. Impacts delete themselves when complete.

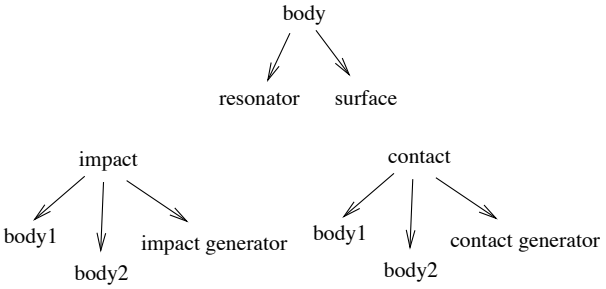

Fig. 1: Main objects in Phya, and pointers.

Contacts are used to represent a continuously varying interaction caused by sustained contact, and should be created when a physical contact has occurred which is not an impact. The contact should be tracked and the corresponding Contact updated with the velocity of the contact relative to each of the colliding surfaces, and the surface force at the contact. The velocity of the objects at the contact point, relative to the world, is usually straightforward to find using velocity and angular velocity obtained from the engine. Because they are integrated quantities they are smooth and well behaved. In some cases the contact velocity can be also be derived directly from the dynamics of the objects by knowing in addition the surface curvatures at the contact, which usually fall into one of a few cases determined by primitives. However it is often simpler to find the contact velocity by differencing contact position. This generates a more noisy signal that is useful in many practical cases, but can also be smoothed if required. Geometric utility functions are provided in Phya to help with these calculations.

Smoothing may also be desirable if an object has a polygonal collision mesh, but the intention is a smooth object. Visually the object appears smooth, but the audio will not be smooth unless the contact data, especially the force, is smoothed.

Each Surface class has corresponding *ContactGenerator* and *ImpactGenerator* classes for generating surface sound. Pools of Contact, Impact and SurfaceGenerator objects can be pre-initialized for speed.

2.2. Signal routing and sympathetic resonance

The default signal routing allows sound generated at each surface to feed the resonator of both colliding objects, as well as adding surface sound directly to the final output. It is also possible to add some routing between resonators, to simulate sympathetic resonance, as you might find in a compound object of different materials. Care is required to prevent non-decaying feedback! Output can be provided as either a simple mix, or a simple stereo spatialization with distancing, or separately from each body by callback, which enables connection to external sophisticated 3D sound systems.

2.3. Contact damping

Damping through surface contact is a very common phenomenon that can add a lot of realism, and is implemented globally by accumulating damping factors on each resonator from each surface it is contact with, prior to updating the output of the resonator.

2.4. Contact hardness

Another general attribute of surfaces is hardness, meaning the springiness of the surface. Generators have the option to use the colliding surface hardnesses to modify the excitation. Harder surfaces generally have less contact damping.

2.5. Process view

Phya processes and an external engine is shown in Figure 2. The collision system in the environment simulator must provide updates to Phya's collision state. This is generally achieved in the glue code using a combination of engine callbacks, iterations and user pointers within engine bodies and Phya bodies. Changes to the collision state propagate within Phya to the audio thread, which generates audio samples as required by the audio output buffer. Parameter update is normally at the physics engine update rate. Phya internally upsamples some parameters to audio rate to achieve better audio continuity.

Biggest potential pitfall is finding a way for Phya to keep track of continuous contacts.

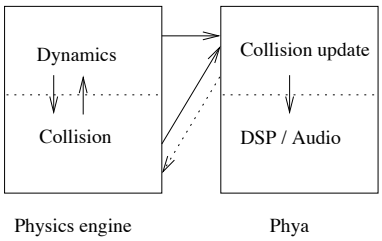

Fig. 2: Phya process view.

2.6. Tracking contacts

Some collision engines do not use *persistent contacts*, meaning they forget information about contacts from one collision frame to another. Even if they are persistent, the contacts generally come in groups centred around a contact patch area. Within the group contacts can move rapidly in a way which may not reflect the contact dynamics well. On the other hand Phya would ideally like to have one well defined nd persistent contact per patch, because it must support audio processes that generate excitations continuously during a contact. The problem can be handled either by modifying the collision engine, which is hard or not possible, or searching contact lists. In the simplest case, the physics engine provides a list of non-persistent physical contacts at each collision step, and no other information. For each physical contact, the associated Phya bodies can be found and compared with a list of current Phya contact pairs. If no pair matches a new Phya contact is formed. If a pair is found, it is associated with the current physical contact. For any pairs left unmatched, the associated Phya contact is released. See Figure 3. This works on the, mostly true, assumption that if a physical contact exists between two bodies in two successive frames then that is a continuous contact evolving. It is common that flat surfaces are in contact. In this case a contact patch may spread over a wide area, and each contact may be valuable in capturing part of the dynamics of the object. Engines that keep persistent contacts are easier to handle. The Bullet engine performs well in this regard.

2.7. Limiting

The unpredictable nature of physical environmental sound requires automated level control, both to ensure it is sufficiently audible and also not so loud to dominate

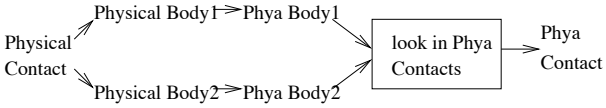

Fig. 3: Find a Phya contact from a Physical contact.

other audio sources or to clip the audio range. It can have a similar effect to compression and limiting in music production, by increasing interest by bringing out detail, but if overdone can become fatiguing. The first line of defense is limiting dynamic parameters, force and velocity, that feed the generators. Then each output stream can be limited using a short look-ahead brick wall limiter, that can guarantee a limit, while also reducing annoying artifacts that would be caused without any look-ahead. Too much look-ahead would compromise interactivity, however the duration of a single audio system processing vector, which is typically 128 samples, is suitable.

3. SOUND MODELS

3.1. Impacts

If an impact occurs over a short period, as is usual, the audible effect on a resonators output mainly just depends on the spectral profile of the excitation, which can be approximated further in terms of the spectral centroid. Considering a simple model for contact dynamics provides a useful way to generate a range of impacts. If the surfaces have spring constants k_1, k_2, then from elementary dynamics the combined spring constant is $k = (k_1^{-1} + k_2^{-1})^{-1}$. A model which just takes k to be the lesser value is also useful. In this way surfaces of differing hardness can interact and produce reasonable, and interesting results. The duration of the impact is $\pi\sqrt{m/k}$ where m is the effective mass $(m_1^{-1} + m_2^{-1})^{-1}$, which can also be approximated by the lesser mass. If one object is fixed like a wall, the effective mass is the free object's mass. Note the duration is independent of amplitude in this linear, non-stiff case. The trajectory of the impulse is describe by a cosine, a shown in Figure 4, however it is sufficient to use a simple triangle pulse with the same duration and amplitude. The impact displacement amplitude in this model is, $A = v\sqrt{m/k}$ where v is the relative normal contact speed. To give the sound designer more freedom over the relation between collision parameters and the impact amplitude, a piecewise linear scheme is used with an upper limit also providing a primary stage of audio level limiting. Note that the masses used for im-

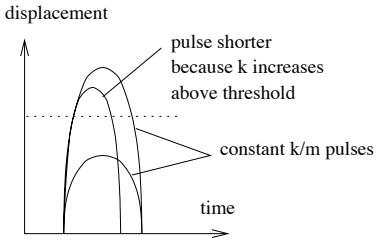

Fig. 4: Displacements from three impacts, one of which is stiff.

pact generation do not have to be in exact proportion to the dynamics engine masses.

3.2. Stiffness

Realistic surfaces are often stiff, they can be modelled more accurately by a spring constant that increases with displacement, causing reduced duration and a brighter signal. This variation provides a naturally useful cue to the dynamics of a collision, where amplitude alone is ambiguous because of distance attenuation. It is also an important aspect of expression in music, and therefor of aesthetic interest in virtual environments. The effect on the displacement trajectories is shown in Figure 4. Again, the variation of duration can be controlled under a piecewise linear scheme for flexibility.

3.3. Complex impacts

Collisions are rarely very simple. An initial impact may be followed shortly afterwards by a sequence of smaller impacts and periods of sustained contact before either maintaining contact or separating. With a good physics engine it is possible to track this process and produce a good overall effect by combining simple impacts and contacts.

To compensate for the simplified geometry used to represent surfaces, an option is provided to simulate a grazing layer, see Figure 5. At the time of impact the collision velocity is used to estimate a dwell time in the layer, which is used by the impact to run a contact for this period, after which the impact exits. The component of velocity parallel to the surface is used for the surface contact speed.

A further facility is provided to trigger samples on impact, in a conventional way, using random selection from a pool, with filtering. If the collision sound is matched to the sound produced by the resonance, an effective hybrid can be produced. The matching can be achieved through the analysis process described later in the resonator section.

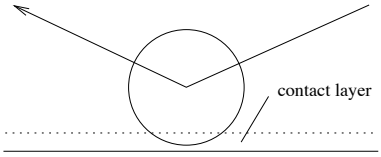

Fig. 5: A grazing impact.

4. CONTINUOUS CONTACTS

4.1. Generic contact model

Contact generation is a continuous process that requires regular update over the contact life. In [1], a generator is introduced consisting of an interpolating loop player, controlled by speed of contact relative to the corresponding surface, fed to a lowpass filter, controlled by the relative speed of the surfaces at the contact. The idea is that the loop player reproduces the surface profile, like a needle on a record, and is then filtered to convey the reduced excitation energy caused by the degree of rolling action rather than scraping at the contact. This is quantified by the relative slip speed between surfaces at the contact. Zero slip speed occurs for pure rolling, while pure slip occurs when the contact point is fixed relative to one of the surfaces, and moves over the other with a speed which is therefore equal to the slip speed.

Phya has a class of surface generators that adopt this filtering action, while the profile generator is specified as a further specialization, see Figure 6.

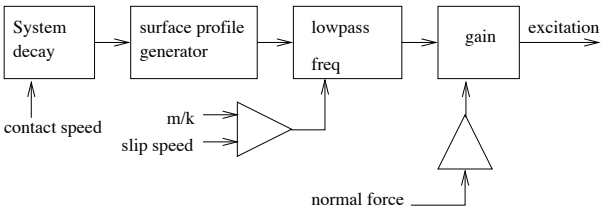

Fig. 6: Surface excitation from rolling and sliding.

The contact excitation is amplified by the normal force, in the same way impacts are modified by collision energy. More subtle modifications can be made to the lowpass filter according to the surface hardness, and also the normal force to simulate stiffness. The control functions within the model are regulated by piecewise linear functions, again for flexibility.

A useful additional option is a large lowpass filter acting on the contact relative to body speeds, with time constant

around 1 second. This *system decay* filter can be used to simulate surfaces of a particle or fluid nature, that take a while to settle once disturbed. It can be used with any of the profile generators described below. The system decay filter introduces a third layer of dynamics, in addition to the physics engine macro dynamics, and the audio rate micro dynamics.

4.2. Profile generators

[1] provides examples with looped white noise to generate profiles. This is effective, but expresses a fairly narrow range of surface types. It is suggested to take recordings of surfaces instead, however the results are disappointing when applied to more granular surfaces. As the contact speed falls, the spectrum shifts mostly to low frequencies, leaving a little of the high frequency events that would be heard in the real surface. The reason is that the recording process does not properly capture the discontinuous nature of much of the surface interaction. Figure 7 illustrates how discontinuous events can arise from interaction between continuous surfaces.

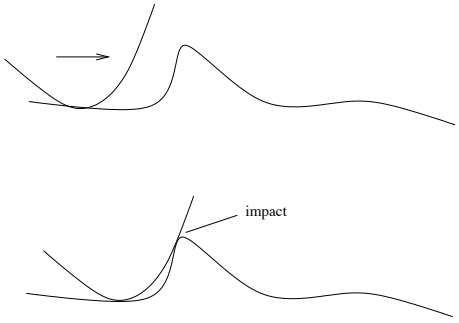

Fig. 7: Micro-impact occuring due to contact geometry

These could be represented using a much higher sample rate, but this is very inefficient and in practice not workable. A more practical approach is to preserve discontinuities by not resampling them. The crudest method to achieve this is by simply not interpolating through the discontinuity, generating some alias noise that is generally not noticeable.

4.2.1. Loop profile generator

Phya contains a loop profile generator, which has a non-interpolated discontinuity option for granular textures. However, it is probably most useful for simulating moving surfaces such as fluid or leaves, using steady state loops of these surfaces. In this case changing the speed of loops generally does not work well, as the spectral make up of the sound, and temporal character of micro events, stays the same except for an overall lowpass filtering and change of event rate. So an option is provided to keep the loop speed constant. A single loop can produce surprisingly good results. The opening of the filter creates a subjective impression of an increased event rate. Additional loops can be added either of the same sample, or progressively more energetic steady states. Introducing these progressively creates a more robust impression of increased event rate. Another approach is to mix sound grains, typically of 0.2 seconds length, taken from a single sample, in other words a form of regranulation. However, this does not sound so convincing due to the numerous fade points, and is not currently implemented in Phya.

4.2.2. Bump profile generator

As well as using stored profiles, Phya also has some generative profile models. The simplest simulates a series of bumps of varying width, height and separation. The width and separation are governed by approximate poisson processes, which allows the average event rate to be scaled easily to match the contact speed. The height uses a programmable distribution. Varying these parameters and the filter settings generates a wide range of fixed granular surface excitations, suitable for driving the main body resonance. Using the system energy filter can give an impression of loose grains.

4.2.3. PHISM profile generator

Phya contains a more complex model that is useful for surfaces where there are many collisions between loose particles. It is based around the PHISM model, [3], with the main addition of the dynamically controlled lowpass filtering stage that is integral in Figure 6. A poisson event stream is filtered to generate a sum of exponential decays, which are then used to modulate a noise source, forming a summed stream of noisy hits. A biquad filter can be used to shape the overall spectrum and provide simple resonance where needed, see Figure 8.

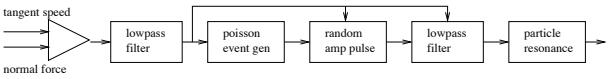

Fig. 8: Modeling loose surface particle sound.

The lowpass section has a significant effect, allowing very convincing interactive surfaces to be synthesized for a range gravel types, sand, paper, foil, leaves. The main limitation with a single contact is that at any time

the population of particles all have the same energy and spectral characteristics, whereas a real population would have a spread. This can be achieved by running concurrent contacts with varying parameters. The previous bump based profile generator actually has this spectral spread built into its design, provided by the spread of bump widths, and this certainly adds to its effectiveness compared with fixed bump widths. The PHISM type generator does not require later body resonance, however this can be used to simulate a situation such as a thin layer of gravel on a metal plate.

4.2.4. Friction profile generator

Smooth frictional surfaces can cause characteristic stick and slip oscillation. This is implemented using a simple lateral elastic model, in which the surfaces stick until the lateral spring forces connecting the surface and main body exceeds a threshold depending on the normal force. The wave form generated is the lateral movement of the surface. The variation of the normal force as part of the collision adds complexity and realism to the result.

4.3. Buzzing

Buzzing and rattling are very common contact processes, caused by vibrating objects in light contact. Like stiff collisions, the result depends in a non-linear way with the strength of interaction, and so provides a distance-independent cue for that strength. Objects that are at first very quiet can become loud when they begin to buzz, due to the nonlinear transfer of low frequency energy up to higher frequencies that are radiated better. Precise modeling of this with a dynamics-collision engine would be expensive. However, a useful approximation can be made by clipping the signal from the main vibrating object, as shown in Figure 9, and feeding it to the resonant objects that are buzzing against each other. This process can be applied in Phya as part of the mix in the output of a resonator, or in the bridge between two resonators interacting.

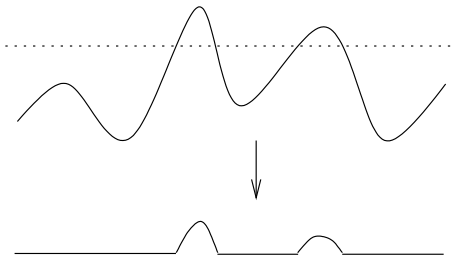

Fig. 9: Clipping of resonator output to provide buzz excitation.

5. RESONATORS

5.1. Modal resonators, calibration, location dependence

There are many types of resonator structure that have been used to simulate sounding objects. For virtual environments we require a minimal set of resonators that can be easily adapted to a wide variety of sounds, and can be efficiently run in numbers. The earliest forms of resonator used for this purpose were modal resonators [4, 1] which consist of parallel banks of second order resonant filters, each with individual coupling constants and damping. These are particularly suited to objects with mainly sharp resonances such as solid objects made from glass, stone and metal. It is possible to identify spectral peaks in the recording of a such an object, and also the damping by tracking how quickly each peak decays, [5]. A command line tool is included with Phya for automating this process. The resultant data is many times smaller than even a single collision sample.

Refinements to this process included sampling over a range of impact points, and using spatial sound reconstruction. To implement this fully would be complex and costly, without great gain. A partial solution is to break a collision object into several different collision objects, each mapped to a different Phya body.

5.2. Diffuse resonance

For a large enough object of a given material the modes become very numerous and merge into a diffuse continuum. This coincides with the emergence of time domain structure at scales of interest to us, so that for instance a large plate of metal can be used to create echos and reverberation. For less dense, more damped material such as wood, noticeable diffuse resonance occurs at modest sizes, for instance in chairs and doors. Such objects are very common in virtual environments and yet a modal resonator is not efficiently able to model diffuse resonance, or be matched to a recording. Waveguide methods have been employed to model diffuse resonance either using abstract networks, including banded waveguides [6], feedback delay networks [7] or more explicit structures such as waveguide meshes [8, 9]. An alternative approach introduced in [10], is to mimic a diffuse resonator by dividing the excitation into frequency bands, and feeding the power in each into a multi-band noise generator, via a filter that generates the time decay for

each band, see figure 10. This *perceptual resonator* provides a diffuse response that responds to the input spectrum. When combined with modal modeling for lower frequencies it can efficiently simulate wood resonance, and can be easily manipulated by the sound designer. A similar approach had been used in [11] to simulate the diffuse resonance of sound boards to hammer strikes, but without the spectral input tracking.

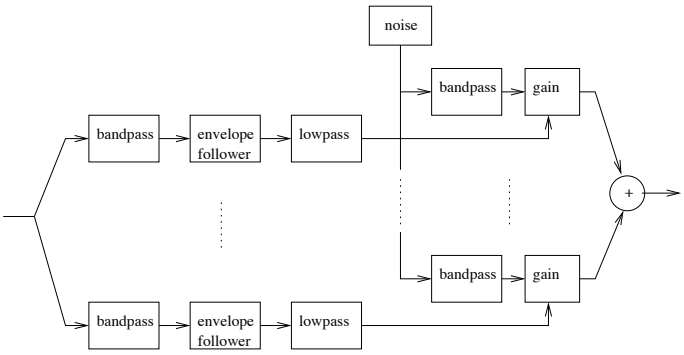

Fig. 10: Outline of a perceptual resonator.

5.3. Deformable objects

There is a common class of objects that are not completely rigid, but still resonate clearly, for example a thin sheet of metal. Such objects have variable resonance characteristics depending on their shape. While explicit modeling of the resonance parameters according to shape is expensive, an excellent qualitative effect that correlates well with visual dynamics, is to vary the resonator parameters about a calibrated set, according variations of shape from the nominal. This can be quantified in a physical model of a deformable model by using stress parameters or expansion factors.

5.4. Cartoon style effects

Pushed far enough, modulating frequency based on deformation becomes increasing cartoon like. The structure of Phya is open enough to accommodate other such effects. For instance, frequency/amplitude/damping can be linked to object acceleration or height. The physics engine takes care of creating cartoon-like parameter trajectories. Going one step further special cartoon generators and resonators could be created and run within the system. The sound will still be physically motivated, as described in the introduction, even if the sound models are very unphysical.

6. VFOLEY

There is an obvious need to provide an interactive authoring environment for Phya, in much the same way as graphical and physical systems can be authored interactively in 3D. This allows changes to sound objects to be made and tested immediately by manipulating objects in the virtual physical environment, using the mouse or other controllers. The system should be able to import standard files containing geometric and physical information about objects, and then export the same with additional information about the Phya objects. Existing formats such as COLLADA[3] are suitable for this.

Such a project, called VFoley, has started[4] , and is aimed initially at a slightly less general goal, to create a virtual Foley environment, where foley sounds can be generated quickly outside of a studio environment, and with the additional inherent creative possibilities. To aid the production process, a replay and edit system will be incorporated, such as that used in machinima production. The system will benefit from more advanced controllers, such as 3D mice that can be used to orient and position objects with better control.

It is expected that VFoley will catalyze the use of Phya, by providing sound designers the ability to experiment without programming. Depending on the results, this can then justify programming effort to incorporate Phya in a project.

7. CONCLUSION

Driving sound models using dynamical information from comprehensive physics engines, creates a rich multimodal system with many intuitive parameters that can be tuned by the sound designer. The result is to make a physically oriented game more sonically interesting and the overall game play more engaging. Sound models and the overall structure have been carefully designed to be efficient but expressive. Interaction effects such as contact damping and hardness, are very simple, but harnessed correctly within the system, add a lot of interesting behavior.

8. REFERENCES

[1] K. van den Doel, P. G. Kry, and D. K. Pai. Foleyautomatic: Physically-based sound effects for interac-

[3] www.collada.org

[4] VFoley is supported by a Creative Industries Technology Strategy Board grant, in conjunction with the EPSRC/AHRC.

tive simulation and animation. In *Computer Graphics (ACM SIGGRAPH 01 Conference Proceedings)*, 2001.

[2] D. Menzies. Scene management for modelled audio objects in interactive worlds. In *International Conference on Auditory Display*, 2002.

[3] P. Cook. Physically informed sonic modeling (phism): Synthesis of percussive sounds. *Computer Music Journal*, 21:3, 1997.

[4] J. K. Hahn, H. Fouad, L. Gritz, and J. W. Lee. Integrating sounds and motions in virtual environments. In *Sound for Animation and Virtual Reality, SIGGRAPH 95*, 1995.

[5] K. van den Doel. *Sound Synthesis for Virtual Reality and Computer Games*. PhD thesis, University of British Columbia, 1998.

[6] G. Essl, S. Serafin, P. Cook, and J. Smith. Theory of banded waveguides. *Computer Music Journal*, spring 2004.

[7] D. Rochesso and J. O. Smith. Circulant and elliptic feedback delay networks for artificial reverberation. *IEEE trans. Speech and Audio*, 5(1):1997, 1997.

[8] S. A. Van Duyne and J. O. Smith. Physical modeling with the 2-d digital waveguide mesh. In *Proc. Int. Computer Music Conf., Tokyo*, 1993.

[9] S. A. Van Duyne and J. O. Smith. The 3d tetrahedral digital waveguide mesh with musical applications. In *Proceedings International Computer Music Conference*, 2001.

[10] D. Menzies. Perceptual resonators for interactive worlds. In *Proceedings AES 22nd International Conference on Virtual, Synthetic and Entertainment Audio*, 2002.

[11] J. O. Smith and S. A. Van Duyne. Developments for the commuted piano. In *Proceedings of the International Computer Music Conference, Banff, Canada*, 1995.