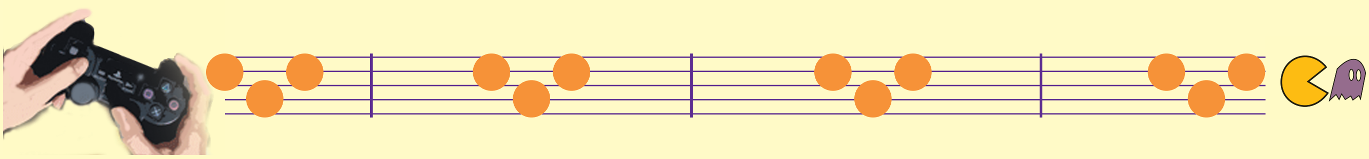

Sound and Music for Games: Prokofiev to Pac-Man to Guitar Hero

By Francis Rumsey
Staff Technical Writer

The recent AES 35th International Conference, *Audio for Games*, held in London in February, illustrated the way in which sounds and music for games are evolving in response to user expectations and changing game genres. The traditional film score, such as written by Prokoviev for *Alexander Nevsky*, was an essentially static creation designed to match the screen action. While the composer might have worked closely with the director to create a combined music/film entity that satisfied the viewer on a number of aesthetic levels, the sound track was fixed for all time in a single form. Games have many things in common with films but differ in the essential fact that they are unpredictable to some degree, and yet still have music and effects as an accompaniment. For this reason the sounds that accompany games cannot be entirely fixed and need to be allowed to change or evolve according to the current game state. How this has been dealt with to date, and what might be achieved in future, are tackled in a number of interesting papers from the conference.

REAL-TIME ADAPTIVE MUSIC IN PERSPECTIVE

McAlpine et al. provide a comprehensive review of the issues surrounding the creation of real-time adaptive music in their paper "Approaches to Creating Real-Time Adaptive Music in Interactive Entertainment: A Musical Perspective." They remind us that the marriage between music and film was successful because both had linear structures, and the film provided a narrative framework on which could be built a musical score. It becomes clear very quickly that the number of possible states, players, contexts, and cultures across which a game might be played makes it very difficult to score appropriate musical sequences in advance. The composer is to some extent blind to the possible combinations of circumstances that might arise in a complicated game.

The authors briefly review the history of game music, showing how it evolved from the early video game in which the most common music was synthesized "on the fly," consisting mainly of monophonic melodies and synthetic effects. This was the sound of *Space Invaders* and *Pac-Man*. This moved on rapidly through games that used the potential of prerecorded music tracks from CD-ROM, such as *Quake*, through dedicated audio engines such as FMOD that facilitate the rapid development of game audio for a number of platforms, to fully interactive music-making using game controllers in the shape of musical instruments, as one finds in modern games such as the popular *Guitar Hero*. An important distinction is made between passive musical components and those that can be considered interactive. They authors explain that what is meant by the latter is actually real-time adaptive music—music that is composed in response to inputs from the player and which provides feedback that may bring about behavioral change.

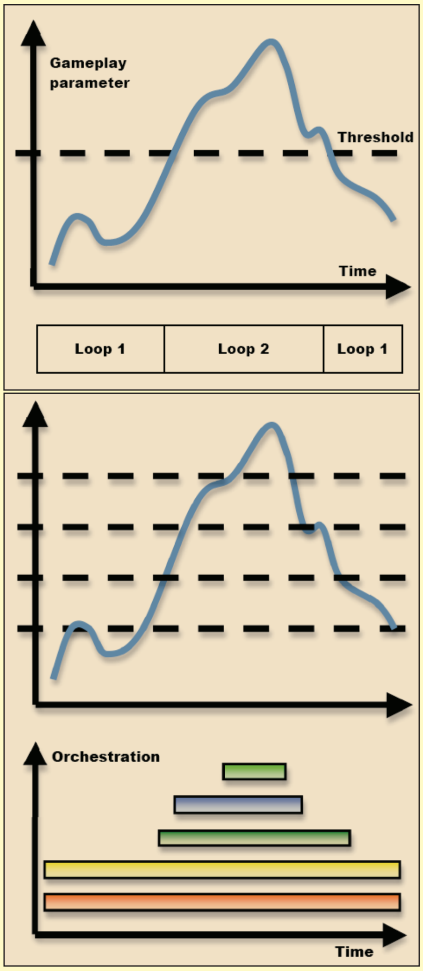

Fig. 1. Top, horizontal resequencing introduces different loops depending on gameplay parameters; bottom, vertical reorchestration introduces new score elements depending on gameplay parameters (courtesy McAlpine et al.).

Different game music elements are distinguished. These include the following: title music, which may set the overall mood of a game at the outset; menu or hi-score music, which usually consists of short loops that fill the silence while the player goes through menus; cut scene music, which typically accompanies action sequences that may be from films, or prerendered gameplay, usually used between levels of a game; event-driven cues; and finally in-game music, which is the primary accompaniment for the player's activities. It is in the latter case that the main challenges of interactive music arise. The three main methods currently used to generate interactive music have been simple event-driven music cues, horizontal resequencing, and vertical reorchestration. The latter two are illustrated graphically in Fig. 1. Horizontal resequencing involves the insertion of ➥

a new musical loop when a particular gameplay parameter exceeds a certain level, whereas vertical reorchestration builds up the complexity of the score by introducing new components as the gameplay parameters change.

They authors show how the traditional musical problems of "engineering" tension and release patterns in a score can be hard to control in games because of the difficulty of predicting the rate at which action is going to take place. This introduces the problem of the correct pacing of musical material. They suggest that solutions to such problems are likely to be complex, involving sophisticated musical intelligence engines that can resequence and reorchestrate content appropriately. Two approaches are suggested. The first treats gameplay as a dynamic control system for music. However, this is hard to use in practice because there is no apparent reason why emergent gameplay parameters should provide a suitable structure for the creation of appropriate music. The second option, using music as a dynamic control system for the game, seems more promising and might suit some types of games that could be led by the music. One of the further options they suggest involves building on the musical concept of improvisation. An improvisation engine might be able to embellish or adapt basic musical lines in terms of melody, rhythm, or harmony to complement the gameplay. The skilled composer is still important because good basic themes and style rules will be crucial to the success of such an approach.

GAME AUDIO LAB

Huiberts et al. describe a system that enables rapid prototyping of nonlinear sound for games in "Game Audio Lab—An Architectural Framework for Nonlinear Audio in Games." They explain that innovation in this field requires a flexible platform free of commercial secrecy and constraints that allows access to the individual game parameters so they can be converted into more meaningful and useful music and sound controllers. They give the example of "level of threat" as a composite game variable that is not easily accessible in most games. It has to be constructed by bringing together other variables that are available, such as the number of enemies within a certain distance, the distance to the enemies, the health of the avatar, and so forth. Existing audio integration tools such as GameCODA, ISACT, and FMOD do not allow the individual game variables to be intercepted and adapted in this way when they arrive from the game engine.

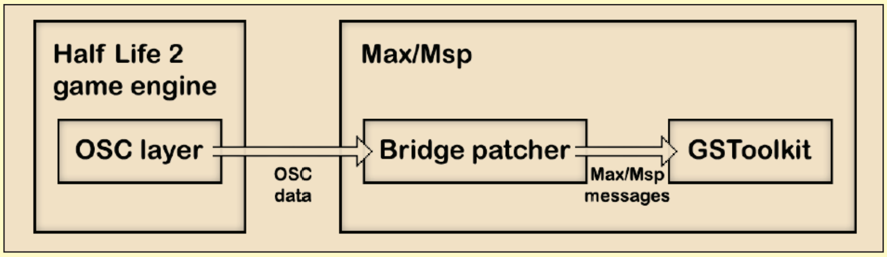

Fig. 2. Software framework for Game Audio Lab (courtesy Huiberts et al)

In order to implement a version of this system, the authors adapted a game called *Half Life 2*, which has an open-source software development kit that offers a number of tools to customize the game. By modifying the game so as to enable it to use the Open Sound Control (OSC) protocol, they were able to interface it with the music/sound software application Max/MSP, via a so-called bridge patcher (see Fig. 2). OSC is a protocol that enables networked control of music and sound devices, rather like a sophisticated version of MIDI, and the bridge software developed by the authors enabled them to combine the individual game variables so as to create composite variables such as that mentioned above. The GSToolkit shown in the diagram is a Max/MSP application that acts as a dynamic sample player with real-time DSP, enabling the designer to alter the mapping of composite control information to sound-processing algorithms. Certain sound instances in the game are routed via the external workstation containing the Max/MSP software, to handle those components that will be separately processed, while the remaining sounds are handled normally by the game platform's audio engine. In the case of the authors' example, only the weapon sounds and ambience layers of the game were externally handled. Two composite game variables, namely level of threat and level of success were used to control selected sounds. For example, level of success changed the gun sounds to make them more or less satisfying. Various parameters of the ambient sounds such as volume, filtering, and reverberation were also modified using these two game variables, in order to change the intensity of the ambience effect. The authors suggest that the principles of this example also lend themselves to music-compositional applications involving generative and procedural approaches.

BRIDGING THE GAP BETWEEN RECORDINGS AND SYNTHESIS

The problem of how to render the sound of complex contact interactions between animated objects was addressed by Picard et al. in "Retargeting Example Sound to Interactive Physics-Driven Animations." Such sounds include sliding and scraping, as well as individual impacts or breaking sounds. They refer in particular to animation using physics engines, extracting parameters from the engine to control the audio processs. (A physics engine is a computer program that simulates physical objects and their motion according to the laws of physics.) The reason for considering a new method is that conventional approaches tend to use prerecorded samples synchronized with the contact events arising from the animation engine, but this leads to repetitive-sounding audio, and good matching between animation and sound is hard to achieve. It is also necessary to arrange for specific sounds to be associated with each contact event, which is time-consuming for the game author.

In the approach adopted by the authors, original audio recordings are analysed and classified as impulsive or continuous sounds, the latter being subdivided into sinusoidal and tran-

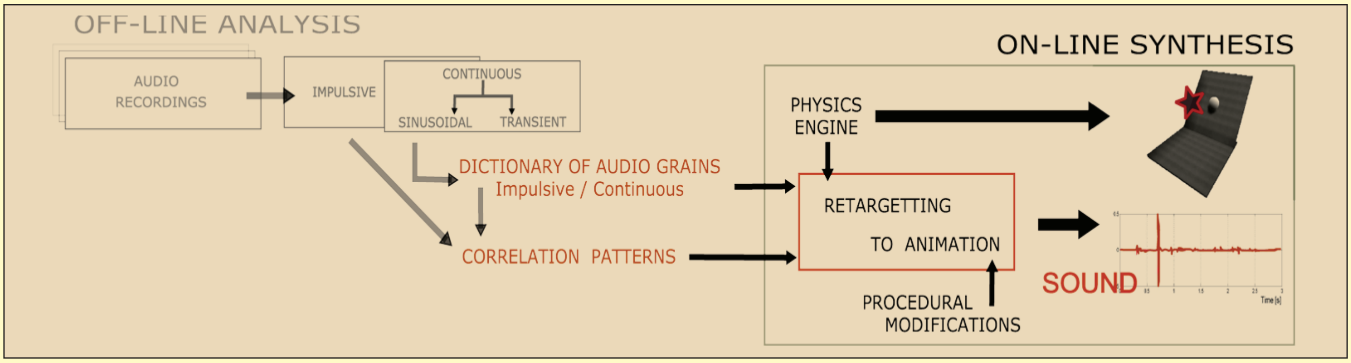

Fig. 3. Overview of the approach used by Picard et al. for retargeting the grains of recorded sounds according to animation object parameters (Figs. 3 and 4 courtesy Picard et al.)

sient components using spectral modelling synthesis (SMS). From this is compiled a dictionary of short segments known as audio grains, which are subsequently used in a resynthesis method targeted to the animation parameters of the object derived from the physics engine. This process is shown in Fig. 3. Audio grains are typically between 0.01 and 0.1 seconds long, corresponding to the sort of time period that might represent the individual impacts in a more complex sound. A measure of spectral flux is used to estimate the grain extraction, which is a measure of the rate at which the power spectrum of the signal is changing. Grains with energy below a certain threshold are discarded in order to limit their number. Following this the original audio recordings are correlated with the collection of grains to identify moments of maximum correlation between them, in order to select an appropriate number of grains that best represent each audio recording over a period of time. This leads to a compact database with a smaller memory footprint than the original recordings. During resynthesis the most relevant grains are concatenated using overlap-add blending.

In order to match the resynthesized sounds to animated pictures involving object contacts, physical parameters from the animation engine such as penetration forces and relative velocities against time are used to determine whether the contact is impulsive or continuous. Impulsive grains are then aligned with penetration force peaks. Continuous contacts are detected where the penetration force and relative velocity are constant. For example, rolling is found by looking for situations in which the relative velocity is equal or close to zero. Using SMS sounds can be modified spectrally according, say, to the object velocity. The authors looked at the effect of surface roughness and were able to identify and control the spectral changes resulting from rubbing against different surfaces. Time-stretching is also facilitated with reasonable ease using this method of resynthesis. Some examples of the results of this work, with audio files, can be found at Cécile Picard's website, http://evasion.inrialpes.fr/Membres/Cecile.Picard/SupplementalAES/

INTERACTIVE SOUND SYNTHESIS

In "Design and Evaluation of Physically Inspired Models of Sound Effects in Computer Games," Böttcher and Serafin consider various synthesis models for use with interactive sound effects. They concentrated on simulating swordlike sounds, using a Nintendo Wii remote as a controller. They point out that most game manufacturers have tended to concentrate on multilayer prerecorded soundscapes, whereas there is great potential for physically-inspired sound models that could lead to a greater variety of sounds during game interaction. Physically-inspired models, as opposed to physical models, they say, have a basis in physical phenomena but are concerned more closely with perceived sound quality than with accurate physical simulation. Purely physical models do not currently seem to deliver as high a sound quality as samples at the present time, which limits their applicability in game engines.

In order to investigate some different approaches to this problem, they interfaced a Wii controller to the Max/MSP synthesis software application, using intermediary software known as OSCulator to convert the control data from the Wii into Open Sound Control format. Four different sound-generation approaches were compared, consisting of sampled sound, subtractive synthesis, granular synthesis, and a physically-inspired model. For the sampled sound they experimented with a number of potentially suitable swords and sticks, settling on a recording derived from a long thin bamboo stick at two different levels of impact (surprisingly, this appeared to create the most convincing sound). One or the other sample was replayed depending on the acceleration of the Wii remote, in similar fashion to current game engines. The subtractive synthesis method was based on bandpass-filtered noise, with the cutoff and bandwidth of the filter being mapped to the acceleration of the Wii in a perceptually appropriate way. Granular synthesis was based on grains of the high-impact sample mentioned above, using the Wii acceleration to govern the speed, amplitude, and duration of the grain replay. For the physically-inspired model, the authors used an approach based on modal synthesis with an exciter and resonator, using a filtered version of the above sample as the exciter. The resonator simulated five resonant peaks that had been previously removed from the sample, and the Wii remote acceleration was used to control the amplitude and frequency of the peaks. The resulting sounds were panned according to the horizontal orientation of the Wii

➥

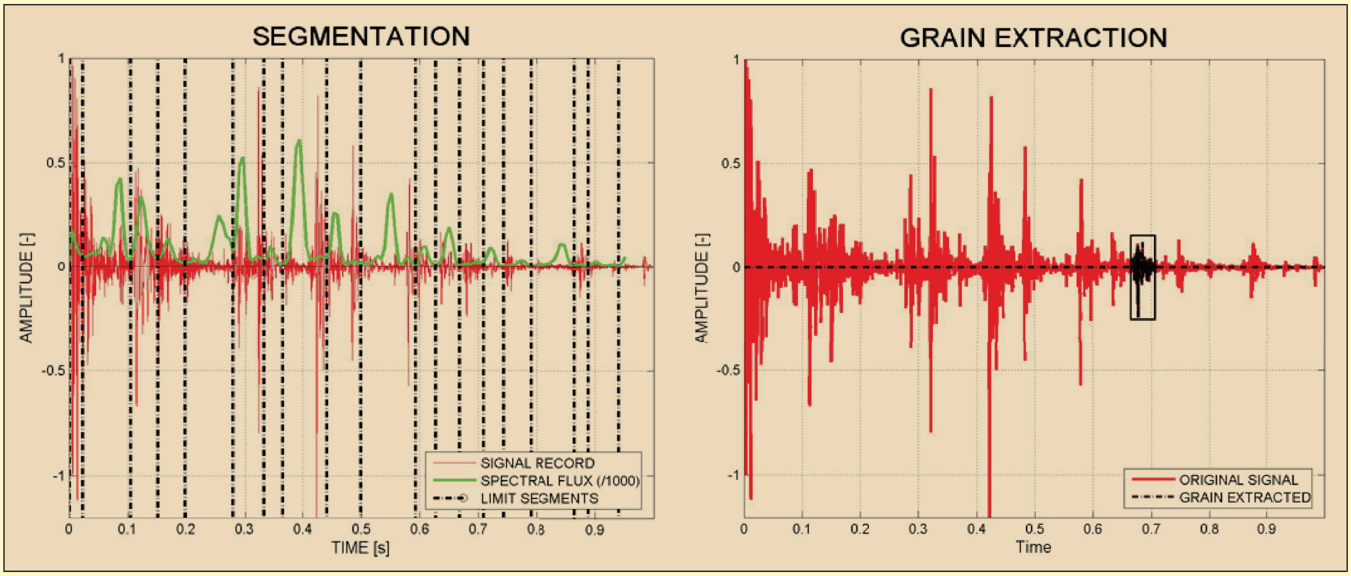

Fig. 4. An example of the segmentation of impulse-like sound events according to a spectral flux estimation (left) and the resulting audio grain (right)

remote. An audio-only game was devised in which the player was supposed to strike an elusive opponent with the sword. This occurred within a background soundscape including the occasional voice of the opponent, as well as mood sounds, footsteps, and crackling branches.

A user test was set up involving a number of staff and students who had familiarity with the Wii controller, asking them to rate a number of aspects including the sound quality, realism, interaction, and gesture control. They included additional questions related to preference, difference in nuance of interaction, and entertainment value. The method of synthesis was randomly chosen before each test. The sound quality of the granular synthesis approach was judged to be the highest, followed closely by the subtractive method. The modal synthesis implementation scored below average. As far as realism was concerned, the granular method also came out on top, which it also did in relation to the connection between gesture control and the sound produced. The authors commented that they felt the sound quality influenced the way people perceived the other test factors, and in some cases this seemed to cause the subjects to get irritated or stop playing more quickly. As shown in Fig. 5, the granular synthesis approach also won the day in terms of overall preference, with users saying that they found it more realistic and varied, as well as being more fun to play. It seemed that there was little difference in perceived nuance of interaction between the synthesis approaches. However, they noticed that body gestures differed substantially depending on which type of sound was involved. The subtractive synthesis implementation, for example, caused users to move in more detailed and diverse ways, which the authors regarded as one of the most interesting outcomes of their research. This type of synthesis was also regarded as the most entertaining of the four tested.

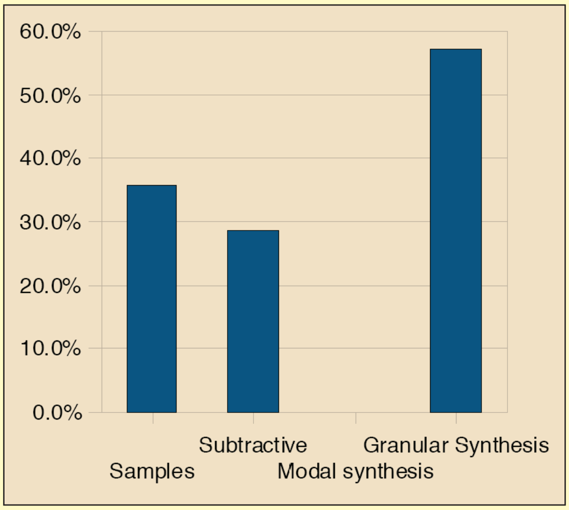

Fig. 5. Proportion of test subjects preferring different methods of synthesis for sword sounds (courtesy Böttcher and Serafin)

They authors concluded that in order to get a better idea about the possibilities of the physically-inspired modal synthesis method, the sound quality would have to be improved in future tests. One possibility would use a continuous input such as noise for the excitation, instead of a static sample, in order to enable more subtle response to body gestures. An interesting comment at the end of the paper concerns the degree to which users actually want to hear physically correct sounds. In fact the subtractive synthesis approach was more closely based on creating something that sounded appropriate, and the authors wondered if users might prefer "cartoonish" sounds to be used when playing games. Visual feedback might also change the user response.

GEOMETRY-BASED REVERBERATION

The simulation of spatial acoustics, including early reflections and reverberation, is another important feature of game audio. Tsingos considers how geometrical-acoustical modelling can be used in games to improve the quality and flexibility of rendering. In his paper "Precomputing Geometry-Based Reverberation Effects for Games," he explains that the complexity of these approaches has typically been too high to be usable in games platforms so far.

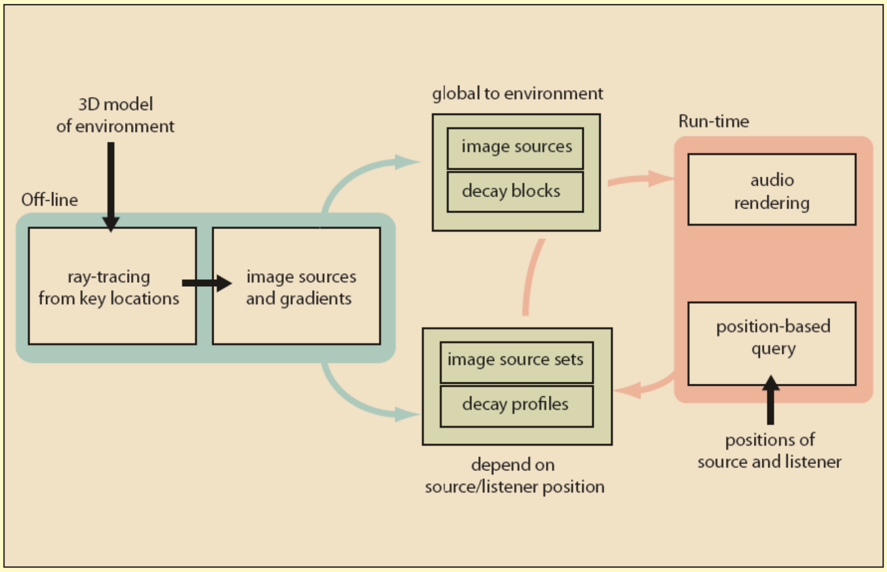

Fig. 6. Overview of the approach adopted by Tsingos for preprocessing and rendering reverberation (courtesy Tsingos)

Traditional artificial reverberators used in games tend to impose a single-room model, so they do not deal very well with outdoor spaces or detailed surface-proximity effects. In order to limit the complexity of implementation with geometrical simulation, Tsingos proposes to precompute the reflection patterns at key locations in a 3-D model of the space. Image-source gradients and hybrid directional-diffuse decay profiles are stored in compact forms so that they can be accessed in runtime without directly accessing the geometrical data of the modeled environment. The overall principle is depicted in Fig. 6.

In order to speed up rendering of multiple sources and make the runtime engine more efficient, Tsingos shows how tens of thousands of concurrent blocks of audio, representing direct sound, image sources, and reverberation decays, can be mixed using frequency-domain processing. This enables a masking model to be employed, so that only audible blocks are sent down the processing pipeline for further action. A two-stage sorting process is used, whereby the level of broadband energy is used to obtain a conservative estimate of masking, followed by a finer-grained decision for those blocks deemed audible by the first stage.

Giving an example of the computation requirements, he shows that a 14-room environment takes about 114 minutes to compute with one key location per room. After compression the data structures occupy only 300 Kbytes because late-decay blocks can be shared between decay profiles. Combining decay profiles at runtime using convolution can be used to simulate coupling between spaces, such as opening or closing doors.

POSTSCRIPT

Game audio is clearly getting smarter, thanks to developments in real-time music and sound generation. Music and sound effects, as well as environments, may be able to be synthesized in real-time based on the parameters of game states and objects in the near future, leading to more varied, compelling, and interactive user experiences.

Editor's note: The papers reviewed in this article, and all AES papers, can be purchased online at <www.aes.org/publications/preprints/search.cfm> and <www.aes.org/journal/search.cfm>. AES members also have free access to past technical review articles such as this one and other tutorials from AES conventions and conferences at <www.aes.org/tutorials/>.