

Smearing Discontinuity :: In-Game Sound

Jeremy Yuille

RMIT University

E-mail: jeremy.yuille@rmit.edu.au

ABSTRACT: On the whole, digital games treat sound with something akin to a mix of fear, contempt and jealousy. Sound is used to accentuate visual transitions, to help create atmosphere, or to inform us of an event that is not visible. Sounds are triggered at discrete points in game time and space, and are very rarely the object of fascination or final goal.

This paper explores the possibility of continuity between game sound events, through the introduction of real-time synthesis. It reports on experiments conducted by the author this year with game systems for use in installations, and attempts to chart the territory of sound driven digital games, particularly with respect to real-time performance..

KEYWORDS: sound, games, music, synthesis, MIDI, real time

MUSIC AND FX

Why is sound left until last; 'tacked on' at the end of production, fighting for processor cycles? Is it purely because game developers cannot spare the computation? Are their mechanics and graphics engines too important to sacrifice a frame hit? The answer, unfortunately, is yes, but not for the reasons you'd expect.

Marketing in the industry is biased toward the visual. Read any review of a major release and in-game sound will probably get a mention, but it will not be examined with the scrutiny afforded rendering engines and game mechanics, so perhaps a look at the less commercial aspects of game development will give a clearer indication of possible futures.

Commercial game engines are co-opted, hacked and even built from scratch for use in installation-based artworks[1]. If you change the emphasis and success criteria of a title from mass appeal to being more about phenomenological engagement, then sound has a more important role in the fulfillment of those criteria.

Over the past year, I was involved with design and specification of the sound engine on such a project, titled *evolglyph*. A description of the project is provided here, taken from the artist's statement:

Evolglyph is a digital system that generates dynamic, three-dimensional icons that evolve

within an abstract virtual world. This world is an eco-system of signs and symbols that explores the idea of emergent language. Elements of the language may be interacted with by sending signals through their geometric structure. These signals create movement and sound in the generated icons which illustrate the internal processes within the system. Subsequently, themes of language and meaning, synaesthesia, process, representation and reality, and the nature of digital media are explored by the work.[5]

Presented with this kind of scope, the search for a suitable sound engine was my primary concern. What we wanted was a system that could play at least 16 different sounds (voices) simultaneously with a control interface that would accommodate higher order musical structures and processes. If that engine could also process those sounds in some continuous way, so much the better. Since the visual world of *Evolglyph* was to be entirely procedural, there would not be a lot of processor overhead to play with. To better understand the decision we came to, let's quickly examine the way sound is deployed in games that situate the player in 3space with freedom of movement.

TERRITORY

These games will usually have a soundtrack, genre associated with the content. Layered on top of this will be the ambient sounds of spaces and objects in the world. Depending on the API, the space that these sounds inhabit will tend to be spherical within which the sound fades from inaudible to full volume. Sound areas may overlap, resulting in a mixture of two or more soundfiles, often with some kind of spatialisation applied to them. These soundfiles need to be incredibly well crafted for two reasons:

- 1) to allow them to loop indefinitely without becoming repetitive or boring, and
- 2) because they may be mixed at any level with any other adjacent sound during runtime.

Current development practice reduces perceived repetition in music and ambient sound by constantly shuffling through libraries of soundfiles that are variations on one another.

The next layer of sounds relates to the actions of entities in the game. These include picking an item up, footsteps associated with walking and running etc, manipulating objects such as re-loading, shooting, opening doors etc. These types of sounds are handled in much the same fashion as the ambient ones, the major differences being the importance of synchronization between the sound and the visual action it represents, and that most of these sounds will not loop.

I CAN'T DO THAT

Reading a review of the PS2 port of *Rez*, a shooter on rails with a musical twist, made me wonder if a major part

of the image/sound issue is the different ways we engage with each sense, particularly where creativity and intent are concerned.

Don't get me wrong, I do like a bit of audience participation, but I've paid my money to be entertained by the professionals, so you do your job and let us tone-deaf drunks do ours. [4]

When I read something like this, I wonder what it is that makes us think that traversing a set of sonic possibilities is in any way different to navigating a visual construct? Perhaps it has something to do with the way we see and hear reality. We choose more of what we see than we choose of what we hear. Does this make seeing an act that we unconsciously connect with creative will and intent (we look *at* something) whereas hearing is a sense identified with absence of choice (we try to shut annoying sounds out of our personal space, often with difficulty)?

SIMULACRA

In a recent article on gamasutra Andrew Boyd of Stormfront Studios (audio for LOTR, The Two Towers) was commenting on a particularly difficult problem that arose in a recent project.

In *Blood Wake*, ..., the "chain gun" sound presented a difficult problem. The player boat almost always has at least one chain gun (and, depending on upgrades, as many as four chain guns) and the enemy boats often have them too. We needed to make a rapid-firing weapon sound that was powerful and impactful(sic), could be listened to for extended periods without becoming irritating, and was able to support multiple playback instances simultaneously.[2]

He then goes on to describe what is essentially an engine for creating 'clouds' of randomly pitched gunshots, very similar to the technique of granular synthesis [6]. It does surprise me that developers have not used a synthesis engine to create an infinite number of possible versions of some of the most widely used sounds in digital games; explosions and footsteps. In fact, Perry Cook describes a complete system of Walking Synthesis[3] using *Physically Inspired Stochastic Event Modeling* which uses meta data from analysis of walking and the agitation of different surface media, to resynthesise the sounds of walking, under parametric control. What you get is a system that can sound like a man, woman or child running, jumping and walking on any surface you care to analyse. The only limit is your analysis, the files of which are significantly smaller than a decent set of individual footstep recordings.

Another interesting correlation between music and game production is described later in the article;

If we had been working on a film, none of that

would have been necessary. We would have purpose-built each combat sequence with its own gun sounds, and modulated them to get exactly the character we needed at the moment.[2]

This is of particular interest to me because it mirrors that duality of linear and non-linear production found in musical recording and performance: Recording is about rendering, multi-tracking, and studio trickery, seldom in real time, whereas performance, at least within computer music, has increasingly become more about systems for real time synthesis and improvisation.

What is possible in one arena often becomes extremely difficult, or even irrelevant, in the other. What you run up against is (again) the constraints of your hardware. A parametric engine for walking might be able to simulate any action between crawling to sprinting on any surface from marble to clover, it may even be able to make realistic gunshots and explosions, but it will be too specific to synthesize sounds like speech, birds tweeting or cars driving past. Time and again developers in any digital medium run up against this general/specific issue: it's folly to attempt a system that is completely procedural, while it's extremely limiting to have to define every occurrence of everything in a system. During the planning of Evolglyph we faced this issue, joking that it was the difference between a "World" and a "world".

We sought the ideal method of synthesis that would provide the widest set of sonic possibilities with the lowest processor overhead. However, we soon realised that this bottom-up process - where the basic rules of the system were defined and combined to define an aesthetic outcome, was too general. Plus, there was no way we would be able to define such a system with our hardware limitations. What we needed to do was define aesthetic goals at the same time, and work from both ends into the middle. This may seem obvious in retrospect, but one of the major artistic aims of the project was:

"Representation of process: Signals are mapped through networks of individual atoms. The EVOLGLYPH system uses the abstract, numerical representation of information as the starting point for this process, reconnecting the values to take another form. Image to sound, sound to image. Data = anything." [5]

In order to not try and design a generalist "World" sound engine capable of any sound type, careful thought had to be given to the finished aesthetic. What did we want the game to sound like? And how would categories of sounds relate to one another? We began by sitting down and going through examples of music and sound recordings that were thought to be interesting and appropriate to the existing game design. From this, a list of sound types was drafted, and it quickly became obvious because there were so many different *types* of sounds, we were going to be

MelbourneDAC 2003

building an 'instrument' based engine.

This involves creating many small sound files that are triggered with pitch and loudness parameters by the game. As much as I would have loved to use a procedural synthesis engine, the desired musical "feel" of the game required sounds from multiple synthesis types, and more importantly, while the programmers I was working with were fanatical about the networking, mechanics and procedural meshes they were building into the system, they had little experience in designing and optimizing Digital Signal Processing (DSP) code.

The SoundFont format was decided upon after testing a number of other API's (these included DirectSound, fMod, PortAudio and an external Mac running Max/MSP). SoundFonts controlled via MIDI were chosen because they relocate the processing of audio to the sound card itself, as opposed to the CPU, which is kept busy trying to render as many frames per second as possible.

The major factors that contributed to the final choice were a combination of generality and efficiency. The system would run from a generic sound card, saving CPU cycles, and its architecture favored sets of 'instruments' that could be designed in parallel with the rest of the game.

MIDI is used to communicate between the game engine and the sound card, making asynchronous production possible. The sound designer can place a dummy set of sounds into a soundfont and just replace those sounds later. The programmers can test audio behavior before the final soundset is created, and the sound designer can test a soundset by playing some MIDI files, without needing the game to be compiled or having to mess around with filename structures. The standardised specification and implementation of MIDI makes tweaking the system relatively easy.

SoundFonts also provide rudimentary filtering and modulation synthesis in addition to pitch, level, stereo positioning and basic effects to each voice.

THE ENGINE

So the playback engine was decided, but that is only half the solution, it is the algorithm for *what* sound will be played *when* that creates relationships between visual events and the sounds associated with those events.

Fortunately there existed a system of hierarchies and behaviors associated with the game elements and their interaction. These visual, spatial and temporal structures were translated and mapped into the musical sphere, using a programming layer that sent MIDI commands to the sound card. These commands would consist of *which instrument to play, what note to play* and the *intensity of that note*.

During this process it became increasingly evident that

our choice of MIDI events triggering standardised instruments was a good one. The inherent western musical relationship of MIDI and SoundFonts (128 notes with a 12 tone/octave default mapping) made it possible to work in a more abstract manner, using elements of western music theory like harmony and common scales. These traditional musical devices were coupled with more recent stochastic and probabilistic methods to give a "weighted randomness", within certain ranges, to the MIDI data.

Between these discrete events, a stream of continuous controller data manipulates the envelopes and filters of each instrument, thereby multiplying the number of possible voices without any extra production.

At time of writing, Evolglyph is still in production, and while initial tests have indicated possible tweaks to the system, things are looking (and sounding) good. Evolglyph is due to be installed in the Screen Gallery of the ACMI sometime later this year.

REFERENCES

1. "game art archive" [selectparks](http://www.selectparks.net/sp5.htm), accessed: 2 Feb 2003
<<http://www.selectparks.net/sp5.htm>>
2. Boyd, Andrew "When Worlds Collide: Sound and Music in Film and Games" [gamasutra](http://www.gamasutra.com/features/20030204/boyd_01.shtml) accessed: 2 Feb 2003
<http://www.gamasutra.com/features/20030204/boyd_01.shtml>
3. Cook, Perry R. Real Sound Synthesis for Interactive Applications. A K Peters Ltd, Natick MA, 2002. pp 191 - 200
4. Hyde-Smith, Alex "Rez review for PS2" [Music 4 Games](http://www.music4games.net/r_rez_ps2.html), accessed: 2 Feb 2003
http://www.music4games.net/r_rez_ps2.html
5. Innocent, Troy "Artist's Statement" [Evolglyph](http://www.evolglyph.com) 2002.
6. Roads, Curtis. "Asynchronous Granular Synthesis." Representations of Musical Signals. Cambridge, Massachusetts: The MIT Press, 1991.