

## The Importance of Implementation

\*\* This was originally a 2 part article published in Industry trade magazine 'Develop' in their March and April 2002 issues. We received a fair few emails after its publication and therefore chose, on that basis, to make it available online in its original uncut version. \*\*

In this essay we'll be discussing what we believe to be the most underrated area of video game audio – *implementation*. We'll be covering the process from design of the API through to the inclusion of audio content in the game itself. (For the most part, our efforts here will be concentrated on sound effects).

The implementation of sound into games is equally as important and creative a process as the composition and creation of the audio content itself. Both are crucial in the quest to create a believable and immersive sound environment. The Sound Designer's role is widely accepted today in the field of games, but the implementation process is generally poorly catered for. In order to improve the situation, we believe the developer needs two things:

1. **an Audio Programmer.** A programmer with knowledge and skills in the audio domain dedicated to either providing tools to allow the sound designer to implement sound, or handling the implementation process personally. (We have always favoured the latter as the programmer can bring an extra set of creative eyes and ears to the process as well as a technical sense of what's achievable.)
2. **an Audio API (Application Programming Interface.)** This is essentially a piece of code that communicates the game environment in real-time to the platform's sound hardware, allowing the sound designer/programmer to manipulate content in a far more creative fashion than simply assigning 'death sound 3' to 'fat enemy 4' (!)

### **The Need to Specialise**

The phrase, "Jack of all trades, master of none" is probably relevant here. In the past people have lacked expertise in specific areas as they have been expected to cover too many other disciplines that game development requires. As technology has moved forward, rather than sticking with generic job roles of the past ("Musician" - anyone ☺) people are now looking to **specialise** in order to push the limits of each individual discipline. An artist is no longer simply an artist; he or she may specialise in concept art, animation, modelling or texturing. We should be moving towards a situation where an "audio programmer" is not just a content programmer who's got nothing to do for a week or two! These jobs should be filled with people who have training in the areas of digital audio, signal processing, physics and maths etc.

### **API Functionality – what should it do?**

Here are a few suggestions of the tools that we feel should be included in any such API. There are probably hundreds more we haven't – a lot of which would be specific to your particular needs. Programmers with a background in audio will help to add more functionality to the list and push the boundaries still further.

### Front End

It's easier to manipulate in-game sounds by creating a basic front end with the ability to change values in real-time and instantly hear the results whilst the game is running.

### Pitch / Volume

1. Random Pitch / Volume Offset - Reality Emulation. Sound is organic and never repeats itself exactly. This is particularly evident in constantly repeating sounds such as footsteps or a ball being kicked. Each time the sound appears to repeat, in reality it is subtly different. We all acknowledge this subconsciously so repetition of samples in a game environment is unnatural to us. This can be rectified by use of a random pitch / volume offset (e.g. + / - 3 values).
2. The use of rapid pitch bend and volume changes to create special effects (e.g. Vibrato) – the added bonus of making completely new sounds from existing samples.
3. Exponential pitch & volume drop offs over a user-defined radius. You can hear an ambulance siren from a long way off, but it only becomes deafening when it's almost on top of you.
4. Volume Ducking to expose one audio element over all others – normally dialogue.

### Dynamic Loading

As RAM is still such a precious commodity, it can prove hard to keep resident all samples required for every entity without making sacrifices. Unless it's implemented early on in the design stage, it becomes almost impossible to persuade the dev team to implement incremental loading on audio if it's not already implemented for graphics and level code. If you're in the position where you can load new sounds two or three times a level, it'll increase the amount of RAM you have by 50-100%. Of course, you'll still need your generic player sounds, but maybe you could give them a bit of a lift by introducing a couple of new variations on each load?

### Scripting

There are situations that come up – for which you'd love to provide audio for - but are limited either by RAM, or the inability for the code to trigger a sound since there's nothing for the sound code to actually *latch* on to. A few examples spring to mind:-

1. sounds that are heard but not seen – a wolf howl or church bell chime in a graveyard.
2. triggering multiple sounds delayed over time for a single entity
3. cross fading between long looping samples to give the impression of sound evolving over time.
4. layering football chants.
5. pitch control over time i.e. make a wind sample sound more blustery the higher up you go.

With the ability to script sounds to be triggered over time, all of the above become feasible.

### Filtering

Consoles haven't been powerful enough to cater for this until now, but the ability to filter in real-time opens up unexplored avenues:

1. underwater environments with a varying low pass filter
2. telephone effects with a band pass filter
3. radio effects on Music with a band pass filter

### Reverb

Reverb is key in giving the impression of size and space. In an ideal world, the API would have the ability to change reverbs from room to room. Not only will it help increase the realism, it'll give each room a slightly different ambience without using any additional sound effects.

### Occlusion / Obstruction

1. When a sound is playing in a room, it will sound very different from outside when the door is open than when it is closed.
2. Moving from the chaos of a battlefield to shelter behind some rocks

### Middleware

For those developers who cannot afford to write proprietary API software, there are middleware providers that cater for your needs. What you won't get is a system tailored to your needs, but all of the major functionality of the hardware will be supported.

## **Effective Audio Implementation**

### *Communication*

The Audio Programmer & the rest of the Sound Department should have a good relationship. They will need to be in constant communication with each other. The sound designer should be considering technical issues of the audio API and it's functionality whilst creating the sound content for a game. Likewise, the sound programmer should be creative about the process of implementing sounds. Rather than view this process as a chore, he / she will experiment with the tools in the API and spend some time tweaking the sound to generate a more realistic result. This will obviously not be necessary with all sounds, but life can be given to some of the more ambient sounds. Take the example of a wind sample. A small looping wind sample will irritate the player if some sort of real-time manipulation is not employed. The sample should be pitch-bending over time, within fairly narrow boundaries if a relatively calm effect is required, and within much wider boundaries if "gale force" is required.

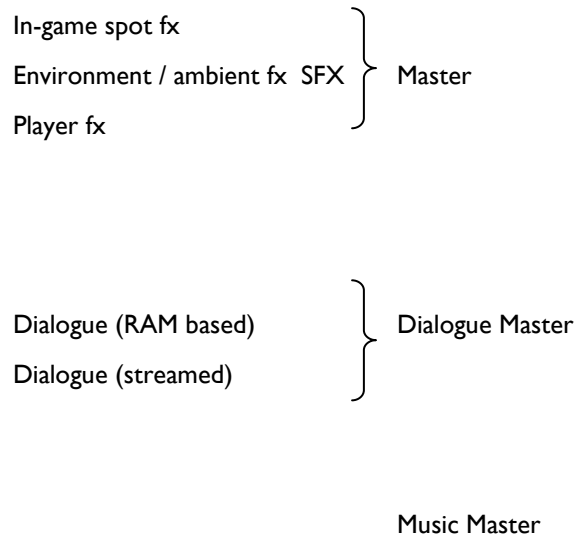
### Reviewing and testing

Once the game is alpha'd the next stage is to obtain an Alpha build to thoroughly test it. Hopefully you'll have a few weeks until Beta – a perfect time to tweak and perfect the sound design – lower a volume here, add a rumble there etc.

### To Dub or Not to Dub

What is dubbing? In a nutshell, it's where all of the audio elements come together – sound effects, music and dialogue. One good way of ensuring that the levels between these three disparaging elements remain consistent and evoke the correct emotions when required is to mix them in subgroups.

It could be organised like this:



So, you have individual levels for each element, and three overall master levels – which could be represented by three faders allowing the user to easily adjust the overall level.

Dialogue is generally regarded as the most important of the three, followed by sound effects and then music. Therefore it's essential to hear the dialogue regardless of what else is happening sonically. In a 'busy' environment, it could easily become an audio cacophony; therefore a 'ducking' feature would help. Ducking is simply fading down the other sound elements for the duration of the dialogue. The best form of ducking is exclusion – dialogue intensive scenes should not be accompanied by multiple explosions and thumping drum'n'bass!

A more structured approach to the implementation of sound will yield dividends in the long run. Film sound has evolved into the various disciplines today for very good reason. All we've done here is take a leaf from that book in order to leapfrog that evolutionary process.