First Things First, Second, and Third: It's About Repetition

Bits of this chapter, give or take a few changes, first appeared in *Game Developer Magazine*.

The most important thing I can tell you is this:

Repetition is the problem.
Repetition is the problem.
Repetition is the problem.
Repetition is the problem.
Repetition is the problem.

Certainly repeating sound effects are the first thing to clue a gamer in to the fact that he's been ripped off. If you've ever sauntered past a pike with a sound designer's head on it, you can be certain that it was placed there by a gamer who didn't want to hear that one certain sound another time. Regardless of variations in filtering, speed, pitch, reverb, panning, and phase, no matter how huge the game's world is, the third time you hear that bomb blast with the splat at the end, or hear the same guy say "Stand up so I can kill you again," your subconscious is acutely aware that this is not, in fact, Reality. The acceptable film school or design committee way to state that the game's designers deserve to be tortured for all eternity is to say that the repetition of sound effects "compromises the willing suspension of disbelief." In other words, it makes the game an ineffectual joke rather than an immersive alternate environment.

Repetition of music does the same thing as repetition of sound effects, but it sports some special subtleties that are worth emphasizing here.

A normal, intelligent game developer will budget for about an hour of music for his game, more or less. For the sake of this chapter, let's call it an hour. That hour is destined to be stretched over, let's say, a 40-hour entertainment experience. That's the normal, intelligent way to do things, and it's pretty stinking ridiculous.

Think about it. Say there was a law in one of those places outside Texas—like Zanzibar or, say, Canada—that decreed that in order for any citizen to be allowed to hear a piece of music, he would have to commit to listening to it at least 40 times. Tell you what, the population of Canada would consist solely of the beaten, smoking corpses of Canadian composers.

Compare this one-to-forty ratio to your being forced to work a full week as a taxi driver with only one audio cassette in your cab. But remember, it's a cassette that you didn't choose. Perhaps this is the reason that so many game players like to run about shooting and killing things. Sometimes I consider the tragic loss of NPCs and reflect on the possibility that if a few concerned citizens like us could solve the problem of repeated audio in games, perhaps we could save the lives of literally billions of innocent animated characters.

The first step toward solving this problem, saving those virtual lives, keeping our own heads off of pikes, and preventing the eternal torture of our designers is to be aware—and make other members of the development team aware—that the "normal, intelligent" approach to game audio is horribly flawed. Solving the problem might be beyond the scope of this book, but planting the seeds to allow you, dear Reader, to be among those who solve it is certainly not beyond its scope.

To help you on your way to Virtual Saviourhood, here are some of the techniques that have been tried with various degrees of success.

Method 1: Ignore the Problem

Here's what you do: **Nothing**.

Positives: Sadly, this is the best way to keep your job. It's probably closest to the way that any experienced game developers on your team handled audio on the last game. It's easy on the entire development

team. It's inexpensive, and it does not challenge anybody's ideas, authority, or sense of confidence. Furthermore, you can rest assured that the game will sound the way a game is expected to sound.

Negatives: The game will sound the way a game is expected to sound.

Method 2: Tiptoe Around the Problem

Here's what you do: Don't use one repeating tune for an entire level of a game. It seems normal and intelligent to have a "first level" tune and a "cave level" tune and an "underwater" tune, but it's just plain old school. There's no excuse, and it will badly injure the ears of anybody playing the game and kill anybody listening who isn't playing the game. So don't do it. Okay? Just don't. If any one tune in your game repeats for more than five minutes, you should do one of the following:

1. Change to another tune after five minutes.

2. Stick a hot fork into your own eye, you evil moron.

Reuse your resources in different circumstances. I know you want special "cinematic" pieces, and "payoffs," and a unique piece for the puzzle with the cute duckies, and such. But the math is simple. If the game's budget is for 20 minutes of music, and the game is constructed so that music plays for an hour in a given session, the music is going to repeat somewhat. And remember that three repetitions of the music would happen only in the best possible circumstances, meaning all music has the same odds of repeating. But suppose you get greedy about special-case music. The more of your music that goes to special one-time cases, the more the other tunes have to repeat to cover for it. Reuse that "Binky meets the cougar" tune as a "tense puzzle-building" or "will we win the pony race?" background piece. The players won't mind; the situation will be different enough that they'll experience it as two different pieces of entertainment. The people listening and not playing will be grateful for one less repetition of that incessant **"riding the pony"** music.

Do not use musical structures that utilize repetition to build familiarity. This is hard to get away from. Sure, conventional musical theory suggests that we play familiarity against variation to achieve tension. That's why conventional music uses forms such as AABA. But in a game, you're going to get 30 repetitions of the tune at least. Think about that. How many times have you listened to the CDs in your house? Even your favorite CD? In a game, you can concentrate on the variation and relax on the repetition. An hour into the game, the familiarity will be there, I guarantee it.

Fade to silence after two minutes of inactivity. Some games don't, and I have one thing to say about that: It's a **strong** indication that everybody on the development team lives **alone**.

Positives: These are brilliant ideas, and I've seen more short-run success come from them than from any other ideas. The costs are low, and the results are good. Do the above, and if your audio content is listenable and your game is playable enough to ship, you will put your game into the top five percent of great-sounding games.

Negatives: You will only be in the top five percent of great sounding games, and with all respect for my chosen profession, that's not yet something that everybody would want to subject his worst enemy to. Because why? Because this: Because even after you've executed all of this earnest cleverness, you're still trying to make an hour of music palatable for 40 hours. Frankly, a miracle is wanted.

Miracle, you say? Surely, Fat Man, that is too strong a word, and you are exaggerating the extent of this problem.

Perhaps you're right. Miracle is a strong word. And all we have to do is make an hour of audio suffice for 40 hours, which isn't that tough. It'd just be like making a loaf of bread suffice for 40. No miracle needed. No need to use such strong, potentially offensive language like "miracle." Let's try another word.

So instead, I'll put it this way. Making one hour of music suffice for forty hours of gameplay **is like polishing a turd, and you can't polish a turd.**

Well, you can, but it's difficult, it gets your rags all dirty and smelly, and in the end, what do wind up with? A shiny turd. And you can't feed the masses with *that*. They prefer bread.

Method 3: Engineer the Problem

Occasionally a producer, designer, programmer, or publisher will look at the one-to-forty problem and recognize it as an issue worthy of his Brilliant Problem-Solving Ability. He will then proceed to decide to solve the problem cleverly. Or, as some might say, "cleverly," in quotes.

I've heard a lot of ideas that qualify as Method 3. From time to time, a producer will insist that the audio for a game be produced using Microsoft Direct Music Composer, a program that will be discussed elsewhere in this book (unless my technical editor forgets to remind me to cover it, which I kind of hope happens because I don't want to get another call from Microsoft like the one I got the *last* time I misinterpreted Direct Music Composer in print). Sometimes a producer has an idea in mind for layering audio, so that more exciting instruments get added as action becomes more interesting. Sometimes there is a more ambitious artist who envisions bringing to fruition his own personal vision of the connection between graphics, gameplay, and audio that will somehow create an ever-varying reactive audio experience. Sometimes these ideas are stinking brilliant, and sometimes they're just plain stinking.

The proposed benefits that tie all the Method 3s together are that somehow the effort and resources that it takes to compose X minutes of music, plus some amount of additional engineering can result in 40X minutes of fresh entertainment. The mood of the developer is always hopeful. The lowest expectation is that a particular chosen Method 3 will affect the results of a composer's efforts the same way that the 39 mysterious "helper" parts of Hamburger Helper affects a single part of hamburger. The highest expectation is that the chosen Method 3 would be like shipping 39 robotic clones of the composer in the game box, each

clone brilliantly programmed by the composer to do his precise bidding, responding deftly and with precise real-time artistic insight to every subtle twitch of the gamer's joystick.

Positives: It certainly can look good on paper when a nice, scientific-looking designer struts into the room full of suits and proposes a never-before-heard audio machine that will use science, math, and good old Cutting Edge-ness[1] to allow music to sound different every time it's heard. Money people and marketing people will often sit up and listen, because they respond well to the idea of technological innovation. And well they should. Technological innovation has a history of tending to sell product. Moreover, I like when Method 3 comes up because it's a magnificent change from the ordinary when actual valuable programming resources get moved to the audio realm.

The strongest positive is, however, very, very much more significant than those in the last paragraph. The Big Time Beautiful thing about Method 3 is that *sometimes* the sounds that come from gameplay on these innovative sound engines can actually create audio entertainment experiences that could only have come from a game—experiences that will never be found in movies, television, or CDs. Method 3 thinking is precisely what's needed to break us of our awkward adolescent habit of considering anything "movie-like" to be good.

Hooray for Method 3.

Negatives: Ahem.

Well, my pet theory, which is almost certainly wrong, states that Method 3 is like trying to build a baby-sitting robot instead of being with your kids. My theory states that it is always better to direct all your audio energy toward making lots and lots and lots of warm, exciting, varying, heartfelt audio (see Method 4 below), and that you can do this better with a kazoo and a cassette recorder than with physically modeled 3D interactive vaporware.

[1] My brother Rick was accused of being cutting-edge. He responded negatively. He said he prefers to try to grasp the handle end.

But, theory aside, the problem is that, in practice, it's not music. Wait, that's too strong, of course I don't mean that, I don't even know why I said it. I take it back. I just mean to say, well—it's not *music*. Yet. That I know of.

I am *so* dying to be proven wrong on this point, but so far the *practical fact* is that I have not heard *any* music that has been produced by any "Method 3" engine that I would willingly play on my computer for any reason other than business research. *Please* send me an angry email that proves me wrong!!!

I have great optimism that this is not a permanent situation, nor is it due to a problem inherent in nonconventional, nonlinear composing methods. I merely think that it's a tools problem. The tools are hard to get a hold of, hard to understand, hard to operate, based on a single person's artistic vision of how music should react to games, or all four, and you just can't build a good robot-babysitter from them. The brilliant young composers who would potentially crack this nut are either not exposed to good tools—or any tools—or they're spending so much energy learning the tool they have that by the time they are finished writing a tune, there is no life-force left in them to put the levity and surprise and joy into music that makes it, well, *music*.

Really, I don't mean to make it sound so terrible. I heard one of the great proponents of Direct Music Composer say that the learning curve for that program is more a learning *cliff.* But the view, he said to its credit, is great from up there. Those are strong and wonderful words, and they deserve to be taken seriously. And as I've said earlier, if any game audio guys have jumped from said cliff from sheer frustration, the news hasn't gotten to me. We're all still alive.

Hey, you ever play *Lemmings*?
What a great game.
Oh, sorry. Back to the topic…

Method 4: Throw Resources at It

Here's what you do: In this method, more time, money, attention, and energy go to audio, with all of it going specifically into composition, rather than relying on new technology and the brilliance of our engineers to make things sound better. Instead of composing an hour of music and then doubling the expense by telling the programmer to cleverly make the thing bearable, this method suggests that we just double the amount of careful, intentional, heartfelt composition that happens.

Positives: This method comes with my personal guarantee that the game will be considerably more than twice as good from an audio stand-point. The rationale here is that first you bring twice the amount of entertaining music to the listener, which of course doubles the amount of Beautiful Human Loveliness (BHL) to which he is exposed. Double value, right there. Then, on top of that, you take the worst thing in the game player's life, which is certainly the over familiarity he has with the music he's stuck hearing for the next 40 hours, and you cut that pain in half. Not bad!

Negatives: You've doubled your music costs, and it's far from enough. You've only come a second 40th of the way to your requirements. Even though you've fed twice the music-hungry people that you had before, your crowd is still starving, and you still have to do the work of 40 loaves with two instead of just the one. And there just isn't enough dough to make 38 more loaves. How many more copies of the game will this doubly good music sell, if the music is still, even after doubling, a 20th of what is needed? And now, the budget guy has seen the music department double its expenses. Uh-oh. And then he heard the music guy tell another guy that there is no solving this problem until the budget is multiplied by 40!!! So, you know that Mister Budget Guy is now playing a little game of his own. Like, it's WWII, he's flying over London, heading a squad of German dive-bombers loaded with layoffs, and now he's looking at your music department through a bombsight. "Target spotted, Sir."

NYOOOWWWWWWWWWwww Dow Dow Dow Dow Dow…

ka-BOOM!

I can hear the soundtrack even now.

Method 5: Transcend It

Here's what you do: Redefine the industry. Shift the paradigms. Come up with something that hasn't been thought of before.

Positives: You solve the problem completely. You become a saint. Your game is loved by many and makes the world a better place.

Negatives: It's hard, and you will have to pay taxes on a lot of expensive things that you'll suddenly be able to afford. Also, it will be annoying to have to keep telling people who ask you what in the world gave you the courage and the insight to achieve such a thing, "I was inspired by that chapter in The Fat Man's book." What a pain *that* will be.

Repetition is the problem.