Obiwannabe

Use the source...

Sponsored by the number 2.219531668871 (Gervers optimal sofa constant)

Introduction for games developers

Whats so cool about native digital synthesis?

Imagine that 3D rendering engines had never been invented and all games still worked like the old game Myst, by presenting a series of flat photographs. That is the state of sound in current games technology in 2006 (it is changing as I update this introduction in 2008). Sound designers create a set of static files, one for each event or scene in a game and these play over and over. Every time you pickup an object or perform an action a preset file is played to the sound system. A little realtime parameterisation occurs on some of the better engines, perhaps a bit of reverb linked to the local geometry, some panning or surround processing, but that's it.

A typical game can contain 100MB of samples which must be edited, looped, packaged and named properly to provide hooks for the level designer or runtime engine. All these sounds have to be collected and preprocessed for use. Although a good sound designer can work miracles with transformations and editing existing sounds the problem remains... the sounds are static. At best the code can currently produce a graded selection or a random selection from a set of static files.

Native digital synthesis is the future for multimedia and games. Instead of recording a bunch of sounds like photographs, the synthesist creates code to be executed at run time. These are the sounds, expressed as procedures or formulas which will be run by the client hardware. Synthesis is a direct analogy to the 3D rendering engine used for most modern games, only difference is that it's a sound rendering process.

Advantages

There are a number of distinct advantages to the native synthetic approach. The size of code required to generate many hours of sound can be compact, occupying only a few kilobytes. While the space efficiency is a staggering improvement the real power of synthesis is far more subtle. The most exciting thing is the way that objects, actors and events coincide with the 3D object aspects of game development. In other words, code can be built

right into the objects in a game to handle their sounds at the same level as physics and other attributes. This means that sonic behavior for almost infinite permutations of object-object interactions may be implicit in a design, something no sample library however large could handle. Furthermore these features may be modified at runtime depending on context. It is possible to easily reuse algorithms by deriving subtlety new instances, scaling or combination. During this exploration we will see many analogies to the 3D development process with direct counterparts to the modeling, texturing, animation and lighting processes familiar to most games developers. Later we examine the reasoning at the root of these analogies and consider the idea that there is a schism in games development which ignores massive optimisations possible by realising that the visual properties of objects and their sonic properties are deeply linked.

Synthesis (or "procedural audio") in games development

When I first wrote this guide in 2005 the adoption of advanced sound design in games was in its infancy. Since then there is much exciting progress to report. The technology has become essential in many games based upon vehicles such as car racing and aircraft based titles. Adoption has been slow because rather few people who understand how to work with these methods. In these online tutorials and in the "Designing Sound" textbook I attempt to remedy that to some degree. Skills taught on traditional multimedia development tend to come from a music basis, employing standard techniques from the recording industry. These roots are essential to everyday practice but are unhelpful in advanced real-time sound design since synthesis is as different to traditional sound design as 3D design is to photography.

Synthesis is a technical art, based partly in mathematical theories, psychology, physics and engineering. It requires a good all round scientific and artistic discipline, focus, imagination and patience for development to proceed. In my teaching of advanced audio methods at several universities and colleges in the UK we approach the subject of sonic behaviour by dealing with the physical behaviour of an object, within an object oriented framework. The effects of the environment and player actions combine to activate methods that, instead of triggering sample replay, run pieces of code to generate the sounds. It is often said by sound designers that the status quo in the established process of games development favours visual aspects above all other considerations. Sound is often an afterthought. Synthetic/procedural audio challenges that, putting sound right back into the frame on an equal footing to visual components.

It requires something of a revision in how sound is done. Most games engines treat sound as an event based faculty instead of building sound

right into the core as a realtime multi-parameter process along with physics and lighting where it belongs. Procedural audio makes more use of continuous control data and requires more thought about the behavioural properties of game objects. Once this change is embraced the rewards are astonishing. Default sonic behaviours make asset management much easier. Unlimited variations and combinations of object interactions are possible (for an overview of procedural audio technology see here).

As the technology grows in strength we are seeing more uses in games. EA used Pure Data in the Spore title to generate sound algorithmically. New applications for the iPhone and other mobile devices are built on the principles of real-time sound generation. While synthesis has acquired some bad cultural connotations through music the outlook for synthetic sound based on physical/behavioural principles in games is very bright. Hopefully these tutorials will engage your enthusiasm for making sounds from first principles and open up the possibilities of procedural sound to a wider group.

What's in these tutorials?

This "Practical guide to synthetic sound design" is quite unique. Most students of synthesis come from a good maths background. Courses in music technology and electroacoustic composition tend to focus on the theory first and leave the applications to the creativity of the designer during their practical career. Most of this gets forgotten in the same way that mathematics, so badly taught in our schools, is forgotten the moment students leave education, because of a lack of meaningful context. It is the ambitious aim of this programme to teach principles of synthesis in a refreshing way with as little recourse to maths or computer science as possible. This is made possible by the remarkable software written by Miller Puckette and others called PureData, a 21st century visual programming language for DSP. The author of this guide Andy Farnell is a computer scientist with a thorough background in sciences and digital media. As far as possible written descriptions are given and equations and graphs are not used unless absolutely necessary. The practical approach relies on asking important questions about the underlying physical nature of sound and psychological questions about the use and context of sound in multimedia.

During the exercises we will touch on physics, perceptual psychology, music and many other disciplines, but you don't have to be an expert in any of these things to start understanding and creating sound through synthesis. Hopefully you will learn a little bit about these subjects along the journey. Some enthusiasm for the material, patience to experiment and a little creative insight are all you need to go a long way. A basic understanding of sound and some familiarity with the software application are assumed.

Ideally this course should be studied as a complement to reading a few essential textbooks given in the reference section, or as a practical counterpart to Miller Puckette's deeper theoretical treatment of Pure Data synthesis in the book "The Theory and Techniques of Electronic Music".

Sound is becoming more important in all kinds of multi-media and interactive applications. It is 20 years since the basic foundations of digital audio were laid down and in much of that time things have not changed other than to provide more of the same. Miller Puckette said "We need to undo the MIDI revolution". In games development I think we need to undo the sampling revolution too.

Summary of points for game developers

- Space efficiency - massively smaller, orders of magnitude smaller in fact.
- No libraries - no searching for that lost glass smash noise, so faster development once you have a few algorithms in your toolbox.
- No editing, looping, normalising and tagging of thousands of files
- No royalties or problems clearing some obscure recording, everything generated from first principles.
- Uniqueness - your games or media application sounds will be unlike anybody else's.
- Realtime parameterisation - sounds are defined as they are created on the client.
- Coherent object modeling - factoring code from physics and 3D object properties with sound for a unified approach.
- Object re-usability - write code that can be scaled or twisted easily into new uses.
- Dynamic runtime LOD - a different approach to mixing where the actual nature of sounds change not just their level as they move into the background.
- Architecture/bit rate independent - future proof, your algorithms will run in ever higher quality as the hardware improves.

Internal Links

next

tutorials list

Some useful external links

Real sound synthesis for interactive applications - Perry Cook

Procedural audio research and development node -Nicolas Fournel (maintainer)

The sounding object - research initiative

The theory and techniques of electronic music (Pure Data) - Miller Puckette

Designing Sound - Andy Farnell (My text on advanced game and interactive audio)

Computer Sound Design - Eduardo Miranda