Game Audio

Coding vs. Aesthetics

Leonard Paul of Lotus Audio

Vancouver, Canada

Game Audio : Coding vs. Aesthetics

Code
Coder
Technology
Left-brain
Science
etc..

vs?

Content
Composer
Creativity
Right-brain
Art
etc..

Game Audio : Coding vs. Aesthetics

Good game audio is:

Code, content, technology, creativity, science, art, left brain, right brain and the composer and coder all brought together as one.

Game Audio Process Levels

Level 1: Short Term
Unaware of most existing practices
Misapplication of practices

Level 2: Medium Term
Aware of existing practices
Application of existing techniques

Level 3: Long Term
Proactive practice of techniques
Apply & create effective techniques

Game Audio Topics

1) Prototyping

2) Peer Review

3) Audio Control Parameters

4) Voice and Memory Usage

5) Tools

6) Automated Mixing

History

- Good old days : Coder/Composer
- 1980's : FM + MIDI Musicians
- Streaming : Pro-tools Musicians
- 2000 : Film Composers

But...

Games are not movies!

Software schedules are not deterministic.

More money = more people = more gaps

Today

In 2001:
US video game sales at $9.3 billion in revenues vs. Hollywood's $8.1 billion

Video games adopting big budgets and management style of film studios

Solutions?

- Narrow and bridge the gaps
- View the problem of audio as a whole
- Open process & free flow of ideas
- Don't force one side onto the other
- Don't pigeonhole employees' talent

"Renaissance" of game audio.

Prototyping

Level 1: A first prototype is quickly built, but due to timeline constraints, it awkwardly evolves into final project

Level 2: A prototype is made and later thrown out, but much of the code remains the same. Some view the prototype as a waste of time.

Level 3: Multiple iterative prototypes are made rapidly. Final is built from best elements. Entire process is archived for future reference.

Prototyping : Environments

Graphic object-oriented audio environments:

- Native Instrument's Reaktor
- Pure Data & Max/MSP by Miller Puckette
- AudioMulch by Ross Bencina

Prototyping : Editing

- Make many sketches
- Edit out non-essential elements
- Strengthen & underline key elements
- Have a friend review

Prototyping : Traps

- Attachment to the prototype to final project
- Focusing on the easy problems
- Adding too much bells & whistles

Peer Review

Level 1: People give periodic feedback on audio. Coder and composer primarily work separately.

Level 2: Peers regularly evaluate audio describing good and bad points. Composer and coder distribute workload.

Level 3: Composer and coder receive and participate in open peer reviews and objectively self-evaluate working in a synergistic manner.

Peer Review : Overview

- Reviews are tossed when schedule looms
  - Participants drag feet into reviews
- Review should provide help and learning for trouble spots and acknowledge good work

Peer Review : Advantages

- Catch early design flaws
- Identify pipeline bottlenecks
- Inspire confidence by identifying good work

Peer Review : Learning

- Sharing of good ideas & processes
- Avoid hiding & covering up mistakes
- Bridge gaps between peers & bonding
- Open avenue to getting help

Audio Control Parameters

Level 1: Sound tags are placed by tagging animation frames in a text file.

Level 2: Sound tags are placed directly in animations by artists. Audio derives control parameters from game state.

Level 3: Additional AI layer is added between game state and audio to make parameters possible for composer to use.

Audio Control Parameters : Methods

1) Game state (Implicit)

Good : Flexible, reactive

Bad : At mercy of any game changes

2) Sound tags (Explicit)

Good : Reliable, clear

Bad : Maintenance overhead, simplistic

Audio Control : Authenticity

- Don't be a "victim" of the game audio state
- Support composer's vision
- 3D audio may be "accurate" but not "interesting"

Voice and Memory Usage

Level 1: Composer has no way of accurately knowing the audio memory map and voice utilization, so he uses a spreadsheet.

Level 2: Composer is provided a run-time memory map and voice utilization output.

Level 3: Composer's memory map and voice utilization output includes statistics on frequency of usage and relative percentages.

Voice and Memory Usage : Overview

Sore point between composer and coder:

- Composer doesn't have enough info
- Coder sometime has to fix resource problems

Voice and Memory Usage : Solutions

1) Volume Culling

2) Sound Sphere Reduction

3) Voice Stealing

4) Instance Capping

5) Sub-Mixing

6) Usable run-time statistics

Tools

Level 1: Composer given text file to tweak volumes, pitch bends and other parameters.

Level 2: Composer given a GUI to modify parameters at run-time as well and compiled scripting.

Level 3: Composer is provided a graphical object-oriented environment which they can tweak at runtime as well as interpreted scripts.

Tools : Problems

- Tools are often at alpha state (ie. barely work)
- No schedule for tools development
- No QA (composer must constantly "complain")
- Maintaining tools not fun for coder

Tools : Reuse

Coders always think they can do it better

- Use existing formats (ie. MIDI)
- Use 3rd party tools (ie. Cubase) to generate data

Tools : Learning Curve

- Tools often proprietary so composers must learn during the project schedule
  - No dedicated training time
- Often things are obvious for coder, not so obvious for composer (usability)
- Test with real-world data from last project

Tools : Object-Oriented & Scripting

- Pure Data / Max+MSP
- Reaktor
- Python / Lua
- More control for composer, but balance with requirements of control
- Divide work between coder & composer

Automated Mixing

Level 1: Audio content integrated by coder with no knowledge of audio mixing.

Level 2: Coder creates real-time faders for composer for run-time tweaking.

Level 3: Composer is provided a system where they can define the behaviour of the audio mix.

Automated Mixing : Overview

- Most non-audio types do not understand it
- Often not acknowledged as a major issue
- Setting volumes for samples non-realtime is often a nightmare for composer

Automated Mixing : Licensed to Mix

- Licensed content is often (ie. always) late
- Licensed music complicates mix
- Need freedom to master licensed tracks

Automated Mixing : Max Headroom

- Out of headroom? L1 it! (yikes!)
- Relative loudness, drop other levels
- No video game mastering guidelines

Automated Mixing : Auto-mix

- Code/Script decides mix levels
- Difficult AI related topic of: "How would a mixer mix the game?"

Automated Mixing : Overload

- Once mix is automated, how to control it?
- Detail required, but must be clear & flexible
- Must actually work
- Difficult challenge

Future

- Real-time synthesis
- Custom real-time DSP effects
- Samples less static with DSP modulation
- New burden for composer?
- Tools & coder should look to music gear biz

Conclusion

- Don't be stuck in job titles!
- We are all creators in a creative process
- We are all engineers in a software engineering process

Jump in, try something new & have fun!

Contact

Info [at] lotusaudio.com