Procedural Audio for Video Games:

Are we there yet ?

Nicolas Fournel – Principal Audio Programmer

Sony Computer Entertainment Europe

Overview

- What is procedural audio ?
- How can we implement it in games ?
 - Pre-production
 - Design
 - Implementation
 - Quality Assurance

What is Procedural Audio ?

First, a couple of definitions...

Procedural

refers to the process that computes a particular function

Procedural content generation

generating content by computing functions



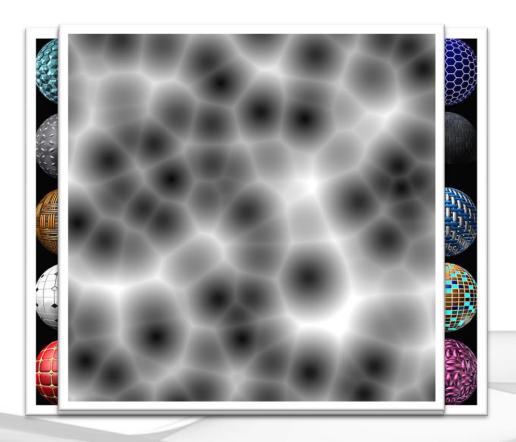
Landscape generation

- Fractals (terrain)
- L-systems (plants)
- Perlin noise (clouds)

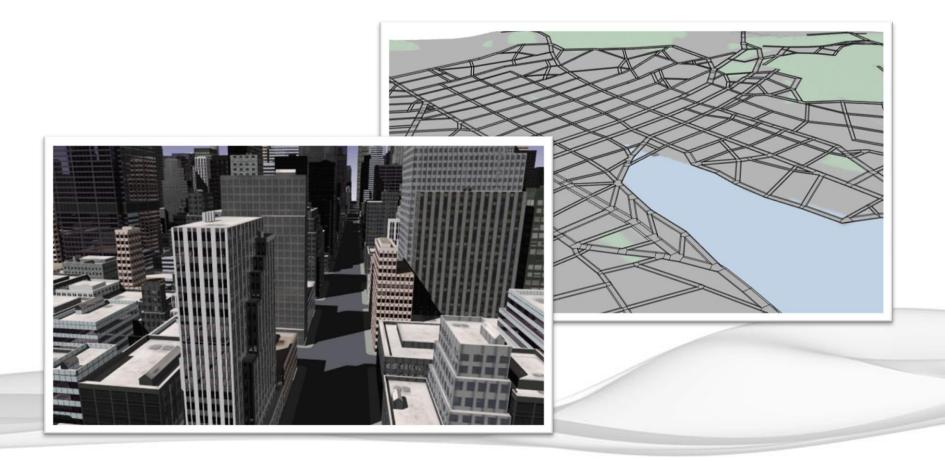


Texture generation

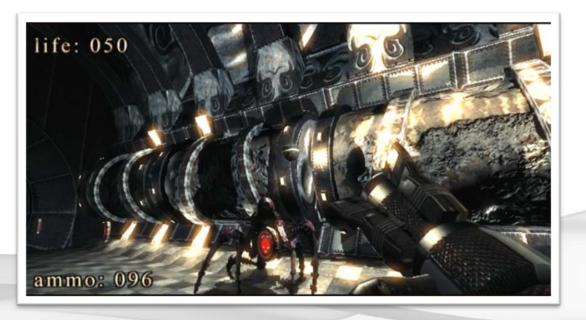
- Perlin noise
- Voronoi diagrams



City creation (e.g. CityEngine)



- Demo scene: 64 Kb / 4Kb / 1 Kb intros
- .kkrieger: 3D first person shooter in 96K from Farbrausch



Procedural content in games

A few examples:

- Sentinel
- Elite
- DEFCON
- Spore
- Love



Present in some form or another in a lot of games

What does that teach us ?

Procedural content generation is used:

- due to memory constraints or other technology limitations
- when there is too much content to create
- when we need variations of the same asset
- when the asset changes depending on the game context

What does that teach us ?

- Data is created at run-time
- Is based on a set of rules
- Is controllable by the game engine

Defining Procedural Audio

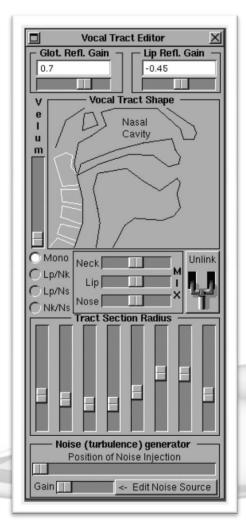
For sound effects:

- Real-time sound synthesis
- With exposed control parameters
- Examples of existing systems:
 - Staccato Systems: racing and footsteps
 - WWISE SoundSeed (Impact and Wind / Whoosh)
 - AudioGaming

Defining Procedural Audio

For dialogue:

- real-time speech synthesis
 e.g. Phonetic Arts, SPASM
- voice manipulation systems
 e.g. gender change, mood etc...



Defining Procedural Audio

For music:

- Interactive music / adaptive music
- Algorithmic composition

SSEYO Koan, Direct Music

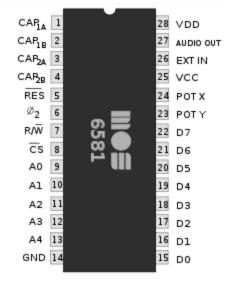
Image: Constraint of the second sec	Demo2 Hold The Chicken \$ chicken.bnp \$ chicken.dlp \$ chicken noise \$ Chicken noise \$ Chicken.sgp \$ Chicken.sgp \$ Chicken.sgp \$ Chicken.sgp \$ Chicken.sgp \$ Style1 \$ Style1 \$ Bands \$ Style1 \$ Bands \$ Style1	 A 1/4 1: Chords (Key=C -0#'s) 1: Tempo 1: 4 (4): Sequence 1: 1 (1): Sequence 1: 2 (2): Sequence 1: 3 (3): Sequence 1: 5 (5): Sequence 1: TimeSig 1: Bands 1: SysEx 	1 2 3 4 5 6 7 1 12 3 4 5 6 7 1CM7 130.00 1 1 1 4 /4 1 1 1 4 /4 1 1 4 /4 1 1 4 /4 1 1 130.00 1 1 1		
12 13 14 15 16 17	For Help, press F1		114 bytes of data.		
18					4 Þ Ø

Early forms of Procedural Audio

The very first games were already using PA !

- Texas Instrument SN76489
 3 square oscillators + white noise
 (BBC Micro, ColecoVision, Mega drive & Sega Genesis)
- General Instrument AY-3-8910 (Intellivision, Vectrex, MSX, Atari ST, Oric 1)
- MOS SID (Commodore 64)
 3 oscillators with 4 waveforms + filter + 3 ADSR + 3 ring modulators etc...

Yamaha OPL2 / OPL3 (Sound Blaster) : FM synthesis



Pre-Production

When to use PA?

Good candidates:

- Repetitive (e.g. footstep, impacts)
- Large memory footprint (e.g. wind, ocean waves)
- Require a lot of control (e.g. car engine, creature vocalizations)
- Highly dependent on the game physics (e.g. rolling ball, sounds driven by motion controller)
- Just too many of them to be designed (vast universe, user-defined content...)

Obstacles

- No model is available
 - don't know how to do it !
 - not realistic enough !
 - not enough time to develop one !
- Cost of model is too high and/or not linear
- Lack of skills / tools
 - no synthesis-savvy sound designer / coder
 - no adequate tool chain

Obstacles

- Fear factor / Industry inertia
 - It will replace me !
 - It won't sound good !
 - If it's not broken, don't fix it
- Citation effect required
- Legal issues
 - synthesis techniques patented
 - (e.g. waveguides / CCRMA and before that FM synthesis)





Two approaches to Procedural Audio

Bottom-Up:

- examine how the sounds are physically produced
- write a system recreating them

Top-Down

- analyse examples of the sound we want to create
- find the adequate synthesis system to emulate them

Or using fancy words...

- Teleological Modelling process of modelling something using physics laws (bottom – up approach)
- Ontogenetic Modelling

process of modelling something based on how it appears / sounds (top –down approach)

Which one to choose ?

Bottom-up approach requirements:

- Knowledge of synthesis
- Knowledge of sound production mechanisms (physics, mechanics, animal anatomy etc...)
- Extra support from programmers

Top-down approach usually more suitable for real-time:

- Less CPU resources
- Less specialized knowledge needed

Ultimately depends on your team skills

Which one to choose ?

Importance of using audio analysis / visualisation software

Basic method:

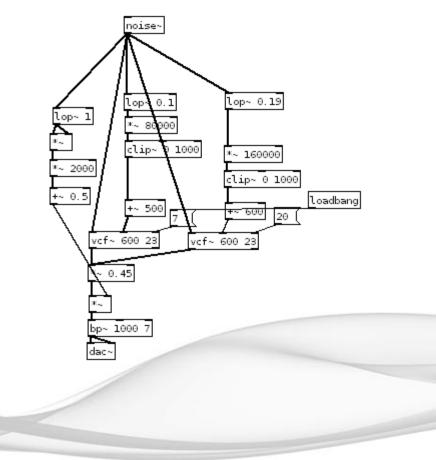
- Select a set of similar samples
- Analyse their defining audio characteristics
- Choose a synthesis model (or combination of models) allowing you to recreate these sounds

Good example of bottom-up versus top-down design

 Computational fluid dynamics to generate aerodynamic sound (Dobashi / Yamamoto / Nishita)



 Noise generator and bandpass filters (<u>Subtractive synthesis</u>)

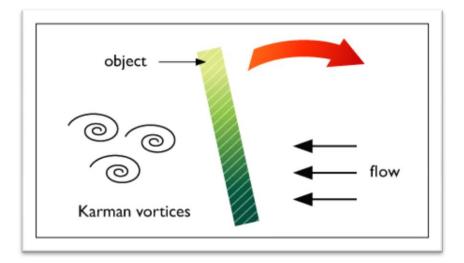


Wind Demo



Procedural Model Example : Whoosh

- Karman vortices are periodically generated behind the object (primary frequency of the aerodynamic sound)
- Using classic subtractive synthesis is cheaper



• Ideal candidate for motion controllers

Procedural Model Example : Whoosh

Heavenly Sword:

- about 30 Mb of whooshes on disk
- about 3 Mb in memory at all times

Recorded whooshes

0 📢

Subtractive synthesis (SoundSeed)

() (

Aerodynamics computations



Procedural Model Example Water / Bubbles

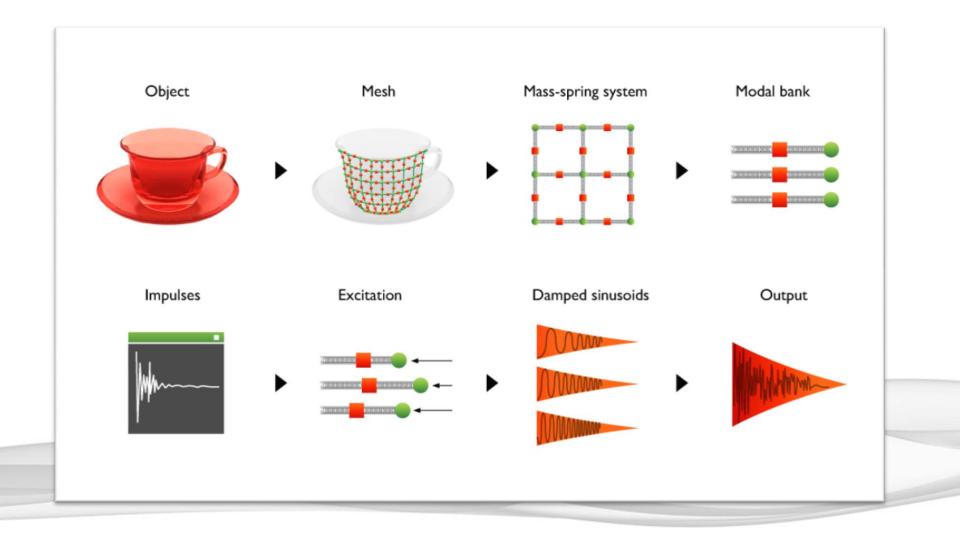
Physics of a bubble is well-known

- Impulse response = damped sinusoid
- resonance frequency based on radius
- Energy loss based on simple thermodynamic laws
- Statistical distributions used to generate streams / rain
- Impacts on various surfaces can be simulated

Bubbles generated with procedural audio

Bubbles Demo





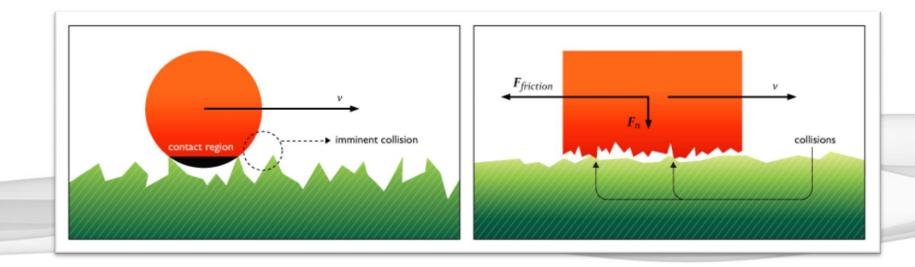
Other solutions for the analysis part:

- LPC analysis
 Source Filter separation
- Spectral Analysis
 Track modes, calculate their frequency, amplitude and damping

Different excitation signals for:

- Impacts (hitting)
- Friction (scraping / rolling / sliding)

Interface with game physics engine / collision manager



"Physics" bank for Little Big Planet on PSP:

- 85 waveforms
- 60 relatively "complex" Scream scripts
- Extra layer of control with more patches (using with SCEA's Xfade tool)

Impacts generated by procedural audio

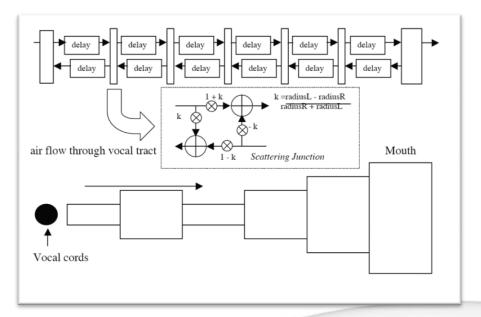


Impacts Demo



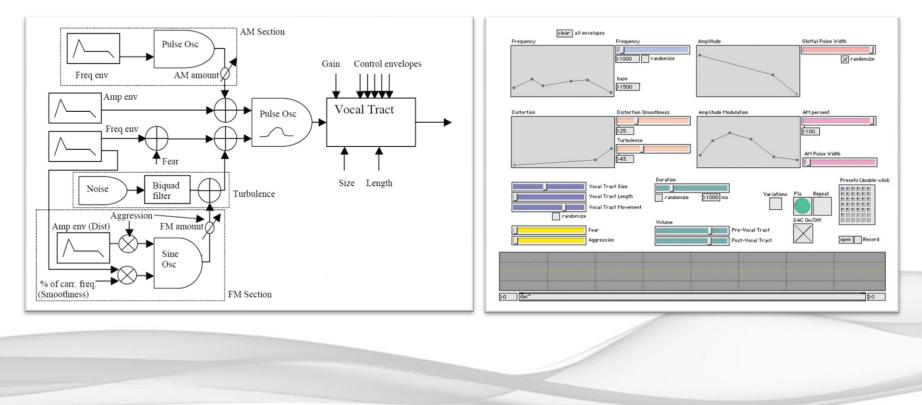
Procedural Model Example : Creature

- Physical modelling of the vocal tract (Kelly-Lochbaum model using waveguides)
- Glottal oscillator



Procedural Model Example : Creature

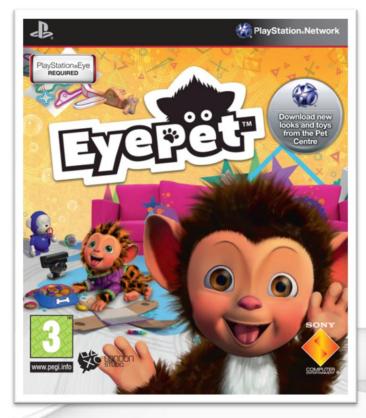
Synthasaurus: an animal vocalization synthesizer from the 90s.



Procedural Model Example : Creature

Eye Pet vocalizations:

- Over a thousand recordings of animals
- 634 waveforms used
- In 95 sound scripts



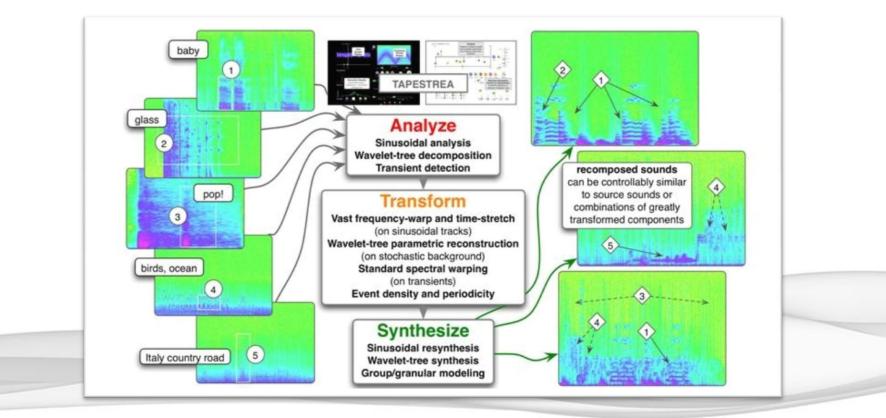
Sound texture synthesis / modelling

A sound texture is usually decomposed into:

- deterministic events
 - composed of highly sinusoidal components
 - often exhibit a pitch
- transient events
 - brief non-sinusoidal sounds
 - e.g. footsteps, glass breaking...
- stochastic background
 - everything else !
 - resynthesis using wavelet-tree learning algorithm

Sound texture synthesis / modelling

Example: Tapestrea from Perry R Cook and co.



Implementation

Implementation Requirements

- Adapted tools
 - higher-level tools to develop procedural audio models
 - adapted pipeline
- Experienced sound designers
 - sound synthesis
 - sound production mechanisms
- Experienced programmers
 - sound synthesis
 - DSP knowledge

Implementation with Scripting

Current scripting solutions:

- randomization of assets
- volume / pan / pitch variations
- streaming for big assets

Remaining issues:

- no timbral modifications
- still uses a lot of resources (memory or disk)
- not really dynamic

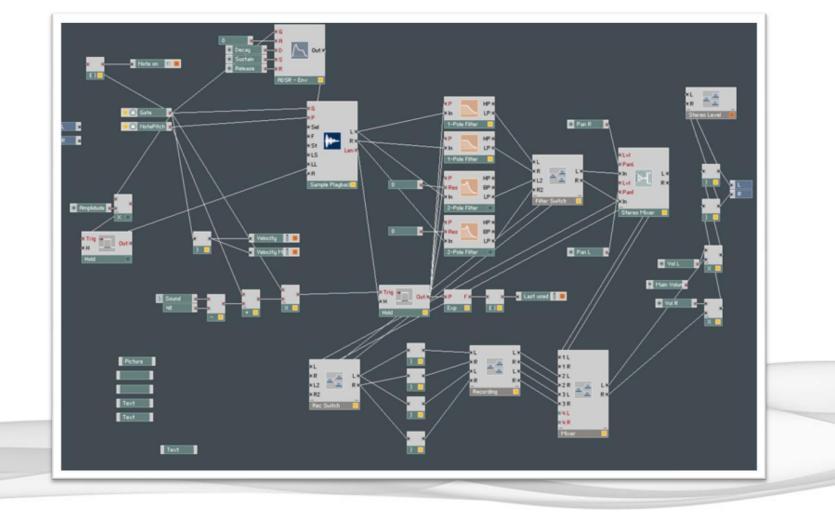
- A "simple" patch in Sony Scream Tool:
- 11 concurrent scripts
- each "grain" has its own set of parameters

	Delay Ital Pan Priority	Grain Online Vol. Ran Priority	Grain. Delay Tol. Priority	Grain. Delay Ital Pas. Priority
Graft. E: Start Chill Studie(Lative)	0	Oran Oslay No. Percent 9:05, amit_replt_0000 more 0 100 30 0 9:05, amit_replt_0000 more 0 100 300 0 9:05, amit_replt_0000 more 0 100 300 0 9:05, amit_replt_0000 more 0 300 0 0 9:05, amit_replt_0000 more 0 300 0 0 9:05, amit_replt_0000, amit_l 0 300 201 0	Random Philipped (2029a)	Parter (2) XX0
Start Onki Sound (2:000)		SOC_and_ruph_iOOF.mar.1 0 E30 200 0	Parlar (1) 2008	Random (Inday (In2000 State) 0
Start ONESound (Eduard)) Start ONESound (Eduard)		200_000_000_000_000_000_00000000000000	Corp Start 0 Random Salay (3-1179 tota) 0	sat Register (log 2 = 0) 0 Long Start 0
(Start Oild Sound (School)		and the second s	mail new a mail and a	
Start Only Sound Chordata)			mellange 0 55 250 0 mellange 17 40 47 0	bat 2 mar 0 80 185 0
Start Onld Sound (10-0700)			1	batleav 15 40 255 0
			Go to Randon Marker (1-4) B	
				Instruction Instruction Instruction Instruction Instruction Instruction Instruction Instruction
			Parlor (1) 1300	? Text Register (reg 0 = 2) 0
			Randon Selar (1-142 tota) 0	? fast Register (log 0 = 2) 0 ① Go to Marker (l) 0 ② Loop End 0
			50C_0mB5.max 0 90 67 9	Loop Cod 0
			10% 0.001 0.000 0	
			Watter Name 0	
			Go to Random Mariae (1-4)	
			Review (2) 2008	
			Start (see) Start (see)	
			► SOC_Dell2.esp 0 90 105 0	
			▶ 500_0mH2.max 25 60 255 0	
			wat for Haces 0	
			Go to Randon Martar (1-4)	
			Long End Polar Polar	
			Loop Start 0	
			Sandon (silor (b-140) tota) 0	
			▶ 500_0+83.com 0 00 00 0	
			▶ SCC_CHER.max 25 40 230 8	
			Start (Sec) Sec Sec Start (Sec) Sec Sec Sec Sec (Sec) Sec Sec Sec	
			Go to Marker (2) 0	
			E copted a	
Marker (1)	Delay Vul Pan Priority	Grain Delay Yul Pan Philarity B Honlow (0) 5890	Grain Defay Val. Fan Priority (b) Marker (1) 2000	Grain. Delay Nat. Pran. Priority
Harlar (I)	5000	Balan (I) 5000	B Highlan (1) 7000	Martar (1) 2500
Random Delay (D-2000 tols)		Randam Delay (0-2000 Indu) 0	The Assestant (Section 10)	Randam Cellay (2-3018 table) 0
Set Regular (reg 2 = 1) Loop Start		Sat Register (reg 0 = 1) 0 Long Start 0	Set Repairs (eq.0 = 1) 0	Set Register (reg.0 = 1) 0 Set Register Fundam.(reg.2 = 30.10
Randow Photosend (100%)		Random Philhered (1975.)	Landau Hickland (1875)	Log Sat 8
bah 5.mar	0 80 255 0 15 50 105 0	Random Play () ests of ()		Long (Lor) 0 Total Regulator (log 0 = 1) 0 Set Regulare Random (log 1 = 1 0
bah 5 bah 5 wat for Vacuus	15 50 105 0	Lands The () and () 0 Lands The () and () 0 40 251 Lands Lange 0 40 251 0 Lands Lange 0 40 251 0 Lands Lange 0 40 251 0 Lands Tange 0 40 251 0 Lands Lange 0 40 251 0 Lands Lange 0 40 251 0	Fanden Reg (5 strid 1) 0 Noti new 0 40 rand 0 Striggt Red (3) new 0 40 rand 0 Not (2) neg (40, 5) new 0 40 rand 0 Not (2) neg (40, 5) new 0 40 rand 0 Not (2) neg (40, 5) new 0 40 rand 0 Striggt Family (3) new 0 40 rand 0	A Set Register Random (reg 1 = 1 0
wat for vacuu		bahdisaar 15 20 125 0 bahdisaar 0 40 225 0	SOC_Dog_Mod_BLoad 0 45 and 0 howElever 0 45 and 0	1 test Register (leg.0 = 2) 0 1 test Register Random (leg.1 = 8 0
Increment Register (reg 0) Text Register (reg 0 = 2)		bach 7 may 0 60 255 0	Nord Lower 0 45 rand 0 SXL Drag Nord, SL over 0 45 rand 0	Set Register Random (reg 1 = 8 0
Go to Marker (1)		bable.com 0 40 255 0	Nord 2-mar 0 45 rand 0	Inst Register (log 0 = 3) O Inst Register (log 0 = 1) O
Loop End		Telephone Operation Operation <t< td=""><td>what for traces 0</td><td>ter Register Reaction (erg (= 4 0) test Register (erg (= 5 0) test Register (erg (= 4 0) test Register (erg (= 4 0) test Register Reaction (erg (= 4 0</td></t<>	what for traces 0	ter Register Reaction (erg (= 4 0) test Register (erg (= 5 0) test Register (erg (= 4 0) test Register (erg (= 4 0) test Register Reaction (erg (= 4 0
		Wat for losses 8		Set Register Random (reg 1 = 4 0
		Increased Register (reg.1) 0	? Text Register (reg (I = 1)	Random-PR/Sternd (100%) 0
		feat Register (ing 0 = 2) Gr to Revier (0) 0	Calt Tester (1) 0	Randon Ray (2 only of 1) 0
		Contemporation (1) 0	- 1999 B	505_0rd_0read7.expr 0 rt rt2 0 505_0rd_0read7.expr 0 rt3 rt2 0
				▶ 50C,8rd,0me0t.max 0 41 42 0
				505,0x4,0xx40,xxx 0 41 42 0 505,0x4,0xx40,xxx 0 41 42 0
				The first Country I at at a
				SOC_Brd_CrandCours O r1 r2 0 SOC_Brd_CrandCours O r1 r2 0
				SOC_Bel_CrowDraw 0 r1 r2 0 SOC_Bel_CrowDraw 0 r1 r4 0 SOC_Bel_CrowDraw 0 r1 r4 0 SOC_Bel_CrowDraw 0 r1 r4 0
				• Vial for Yokes
				Go to Harlar (0) Randan Delev (0-05 Mele) 10
				Go to Harlar (I) 0 Random Delay (In-15 tota) 12 Loop Drd 0
Marker (1)	Delay Ital Pan Priority	Gran Delay Yul Pan Provity	Grain Onlay Yeal Pan Presetty IP Start Old/Sound (InstrictionSI) 0 0 0 0	
Set Register (reg 0 = 1)	0	fait Explain (reg.) = () for log Start	Rart Orid Sound (Incidentical)	
International (englisher (englisher) Engelstein (Engelstein (Engelstein)) exclutertitik neur exclutertitik neur exclutertitik neur exclutertitik neur exclutertitik neur becomment familien (englisher)	0	Loop Start Ø		
Random/Play (4 sets of 1)	0			
ordation was	0 110 67 0	c quadra Reg (1 args of 1) 0		
coletti en	0 110 67 0 0 110 67 0 0 110 67 0	orthat13.max 0 10 200 0 orthat12.max 0 10 200 0		
crichard 21 may	0 110 67 0	Collectioner 8 90 290 8		
Wat for Hoces	0	wat for vacuum 0		
Increment Register (reg 0)	0	- Increment Register (eng.()		
Increment Register (reg () Text Register (reg (= 7)	0	Increment Register (reg () Prost Register (reg 2 = 21)		
Go to Marker (0) Loop End	0	Garto Rafler (1)		
and the		E COPTOR D		

Implementation with Patching

- Tools such as Pure Data / MAX MSP / Reaktor
- Better visualisation of flow and parallel processes
- Better visualisation of where the control parameters arrive in the model
- Sometimes hard to understand due to the granularity of operators

A "simple" patch in Reaktor...



Another solution

Vendors of ready-to-use Procedural Audio models:

- easy to use but...
- limited to available models
- limited to what parameters they allow
- limited to the idea the vendor has of the sound

Examples:

- Staccato Systems already in 2000...
- WWISE SoundSeed series
- AudioGaming

Going further...

Need for higher-level tools that let the designer:

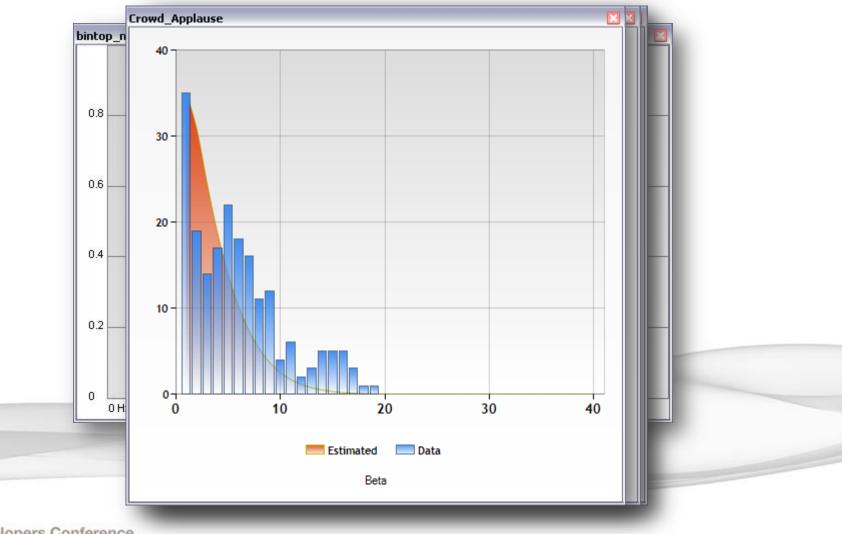
- create its own model
- specify its own control parameters
- without having an extensive knowledge of synthesis / sound production mechanisms

without having to rely on third party models

Importance of audio features extraction

- To create models by detecting common features in sounds
- To provide automatic event modelling based on sound analysis
- To put the sound designer back in control

Think asset models, not assets



Implementation: Typical modules

Lots of different ways to organize modules, different levels of granularity

3 main types of modules:

- Event generation: probability distributions
- Audio synthesis: subtractive, modal, granular, F.M, waveguides...
- Parameter Control : envelope generators, Perlin noise, excitation modelling (friction, sliding etc...)

Implementation : Interface

Requires an even greater interaction between sound designer, game designer and programmer

Control parameters can come from a lot of subsystems:

- Animation
- Physics
- Al
- Gameplay

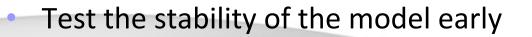
Requires a uniform interface with all game subsystems

Implementation : Parameters

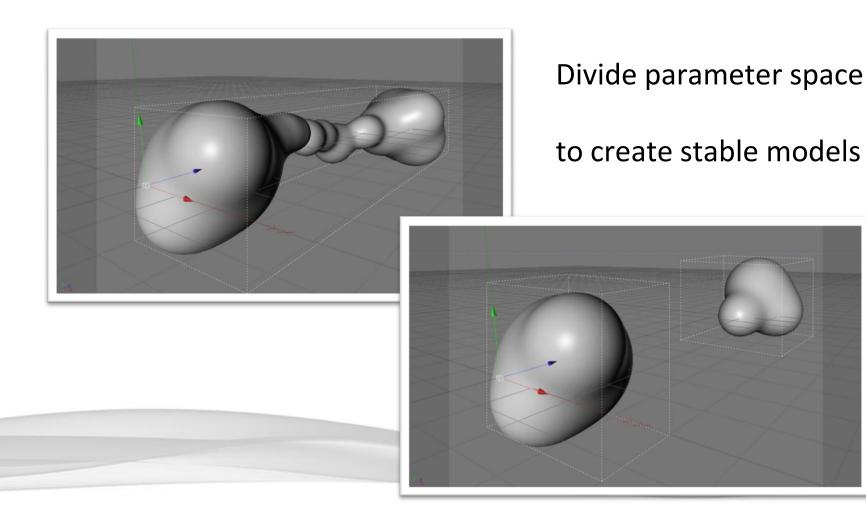
You can add all the parameters you want

It's a trap !

- Limit the number of parameters
- Limit their range



Implementation : Parameter space



Implementation: CPU Usage

The bad news

- Highly dependent on model
- Even dependent on parameters ! (e.g. number of grains, main pitch)
- Non linear models (FOF)

It's not so bad...

- Typical sample playback uses resources also (resampling, filter...)
- Some algorithms are not more CPU hungry than a simple EQ

Implementation: CPU Usage

Mitigating factors:

- Depends if modular / fixed architecture for a few chosen models ("interpreted" a la PD, or "compiled")
- LOD: for different sounds and inside the same sound
- Dependent on update rate (control signal)
- Important to have tools display some metrics about CPU usage in the tools
- Granularity of modules

Quality Assurance

QA: typical sound bugs

- The sound effect is not playing
 - is it loaded ?
 - is it triggered ?
 - is it a voice management issue? Not enough free voices?
 - priority is too low?
- The sound effect is not looping
 - wrong looping points
 - bad settings (must be flagged as looping ?)
 - voice cut off by voice manager

QA: more typical sound bugs

- Wrong volume / panning:
 - wrong 3D settings
 - errors in 3D positioning code ?
- The sound is stuck in looping mode:
 - sfx not stopped
 - hardware voice not released
- Garbage data is played
 - sample data not correctly loaded / encoded / decoded
 - something is writing over our data etc...
 - stuttering \rightarrow streaming issue

What kind of bugs are they ?

- Easily detectable
- Mostly quantitative bugs
- Do not require specific audio knowledge
- Any tester can be assigned
- There is a known list of possible causes



QA: PA sound bugs

- Synthesis vs. playback: qualitative aspect (sounds like this or that)
- P.A. model more complex and controlled by more subsystems than sample playback
 - harder to describe the exact conditions under which a bug occurs
 - harder to reproduce it
- CPU cost not linear: harder to deal with something not playing...

QA: PA sound bugs

- Fixing the issue is harder
- Modifying the model may be required
- Different structure will not have the same CPU cost or control parameters
- Might bring up new audio glitches

QA: solutions

- Education of testers (ideally a specific audio tester)
- Testers should know about the audio models or be able to refer to them
- The stability of the model must be tested in the tools as much as possible

Are we there yet ?

The good news

- Some models can be implemented very easily
 - Impacts / contacts
 - Footsteps
 - Air / Water
- They offer a lot of advantages compared to static sounds
 - Procedural audio is not necessarily CPU expensive

...

The bad news

- Not a solution for everything
- It is still harder to implement
- Mostly due to lack of:
 - trained sound designers / programmers / testers
 - adapted tools / run-time
 - ready-to-use models

Solutions

- Get better tools (higher-level, importance of audio features extraction)
- Educate teams across disciplines
- This will help the creation of procedural models database
- Share models across the industry



Thank you !

Any questions ?