# Future Technologies for Game Sound Design

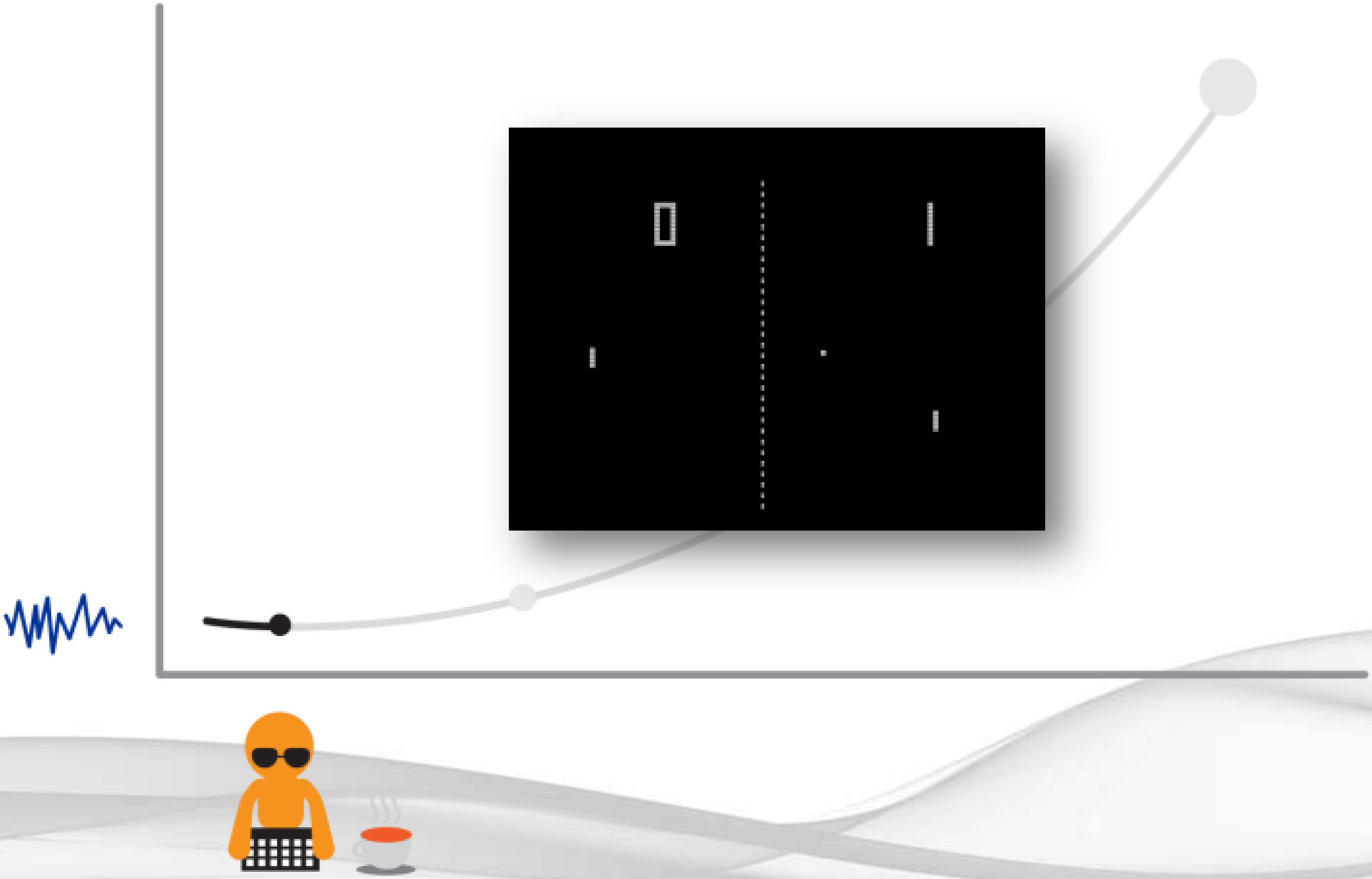
Nicolas Fournel – Principal Audio Programmer

Sony Computer Entertainment Europe

Worldwide Studios
Creative Services Group

SONY

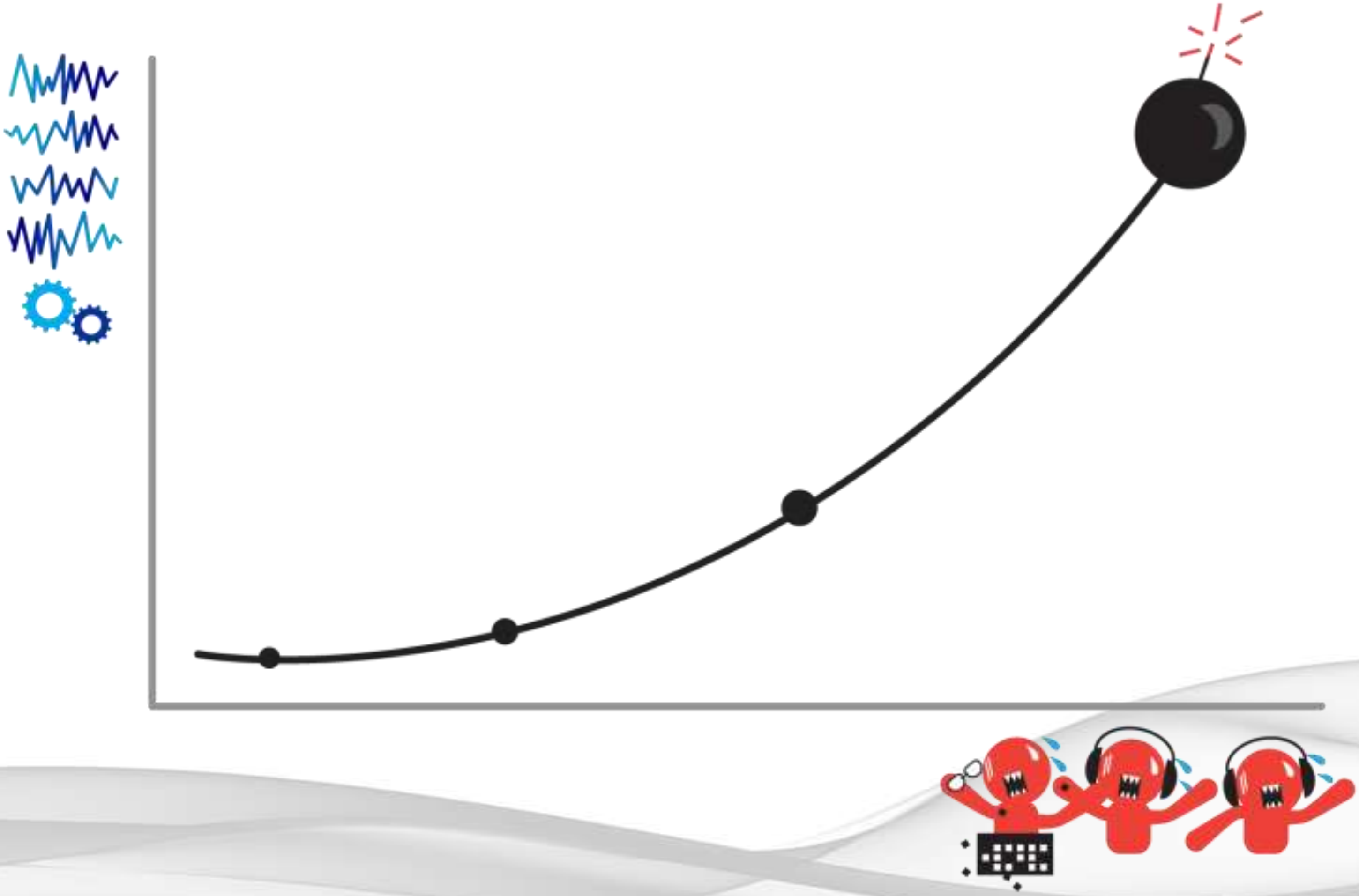
# Game Audio Evolution



# Game Audio Evolution

# Game Audio Evolution

# Game Audio Evolution



# Overview

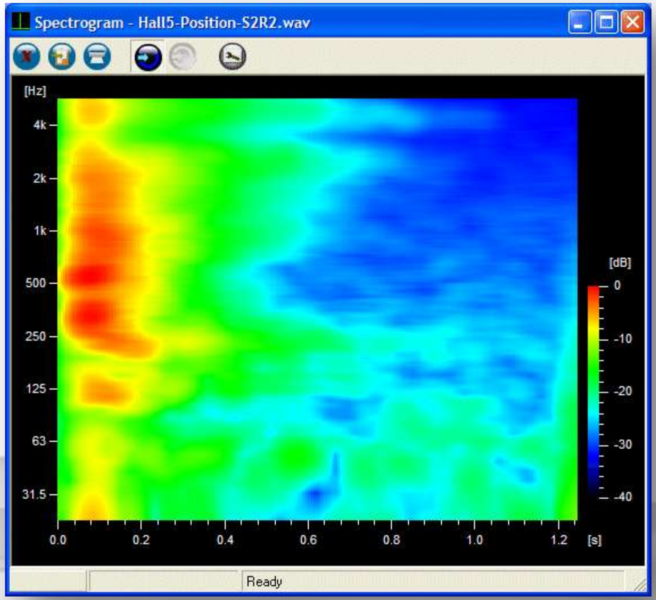
- Audio Analysis
- Procedural Audio
- Combining them

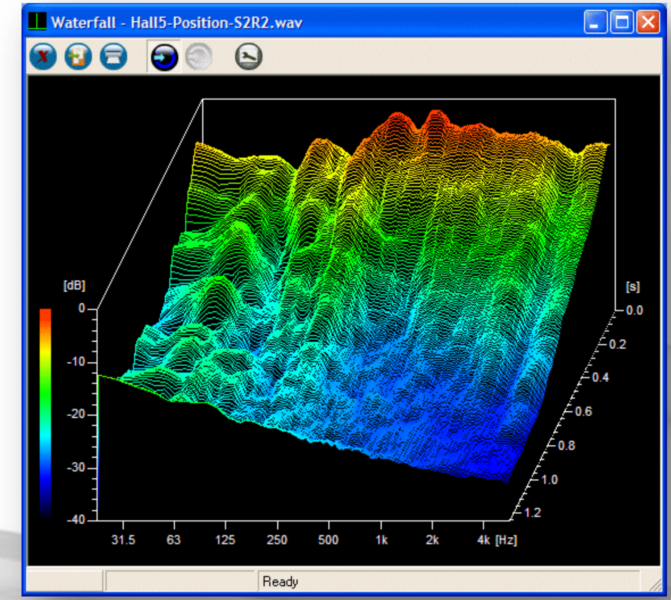
# Audio Analysis

# Audio analysis is more than…

RMS (Root Mean Square)

FFT (Fast Fourier Transform)




Worldwide Studios
Creative Services Group

# More fun with spectra

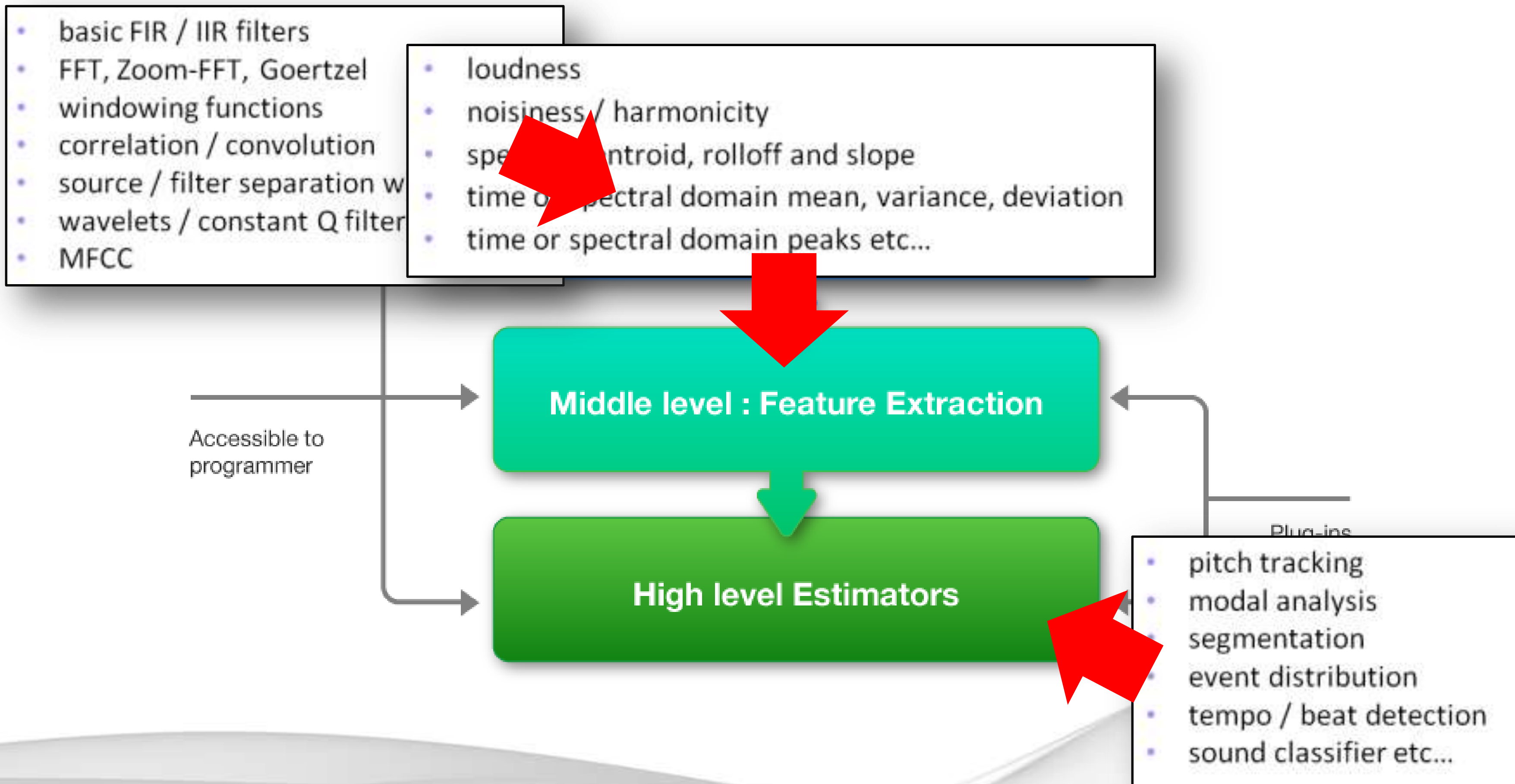
Other features based on the spectrum

- Spectral Flux, Rolloff, Centroid, Flatness, Kurtosis etc…
- Noisiness / inharmonicity / even and odd harmonics

Other ways to detect spectral content

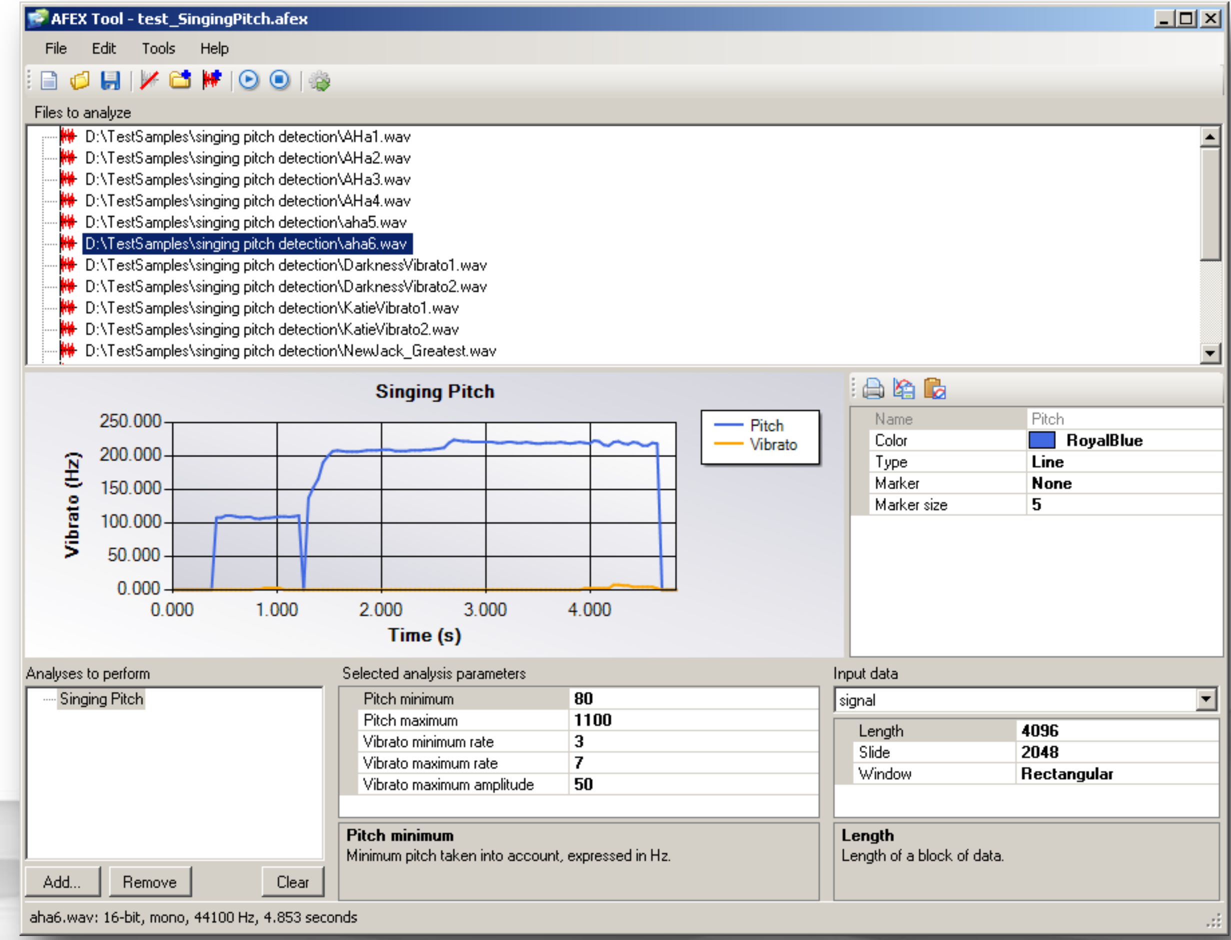
- Goertzel algorithm
- Constant Q filter banks
- Wavelets

# SCEE's AFEX



basic FIR / IIR filters
FFT, Zoom-FFT, Goertzel
windowing functions
correlation / convolution
source / filter separation w
wavelets / constant Q filter
MFCC
loudness
noisiness / harmonicity
spe ntroid, rolloff and slope
time o pectral domain mean, variance, deviation
time or spectral domain peaks etc...
Accessible to programmer
Middle level : Feature Extraction
High level Estimators
Plug-ins
pitch tracking
modal analysis
segmentation
event distribution
tempo / beat detection
sound classifier etc...

# AFEX Tool




Worldwide Studios
Creative Services Group

SONY

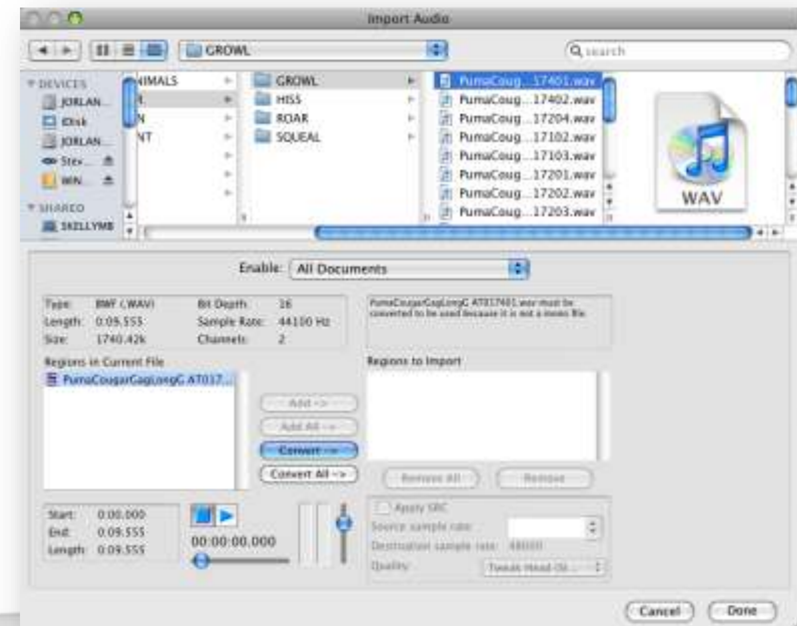
# Game Design Applications

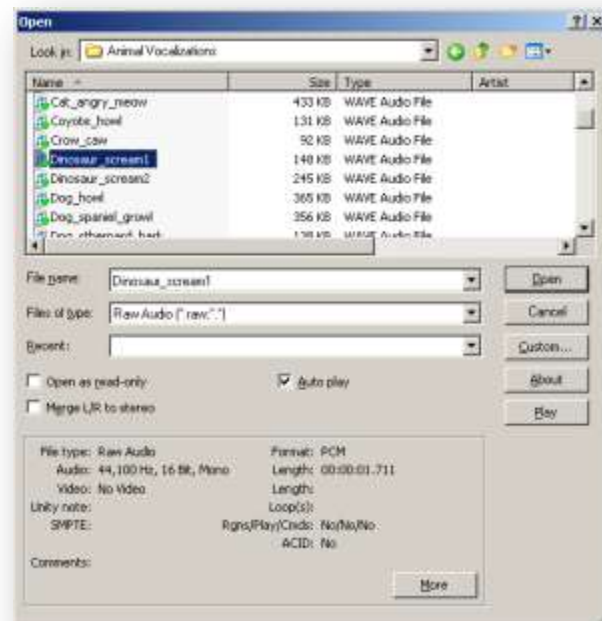
- Singing games
- Levels generated by user music
- Voice recognition
- Sound classification

Even more important with social gaming !

# Selecting sample files

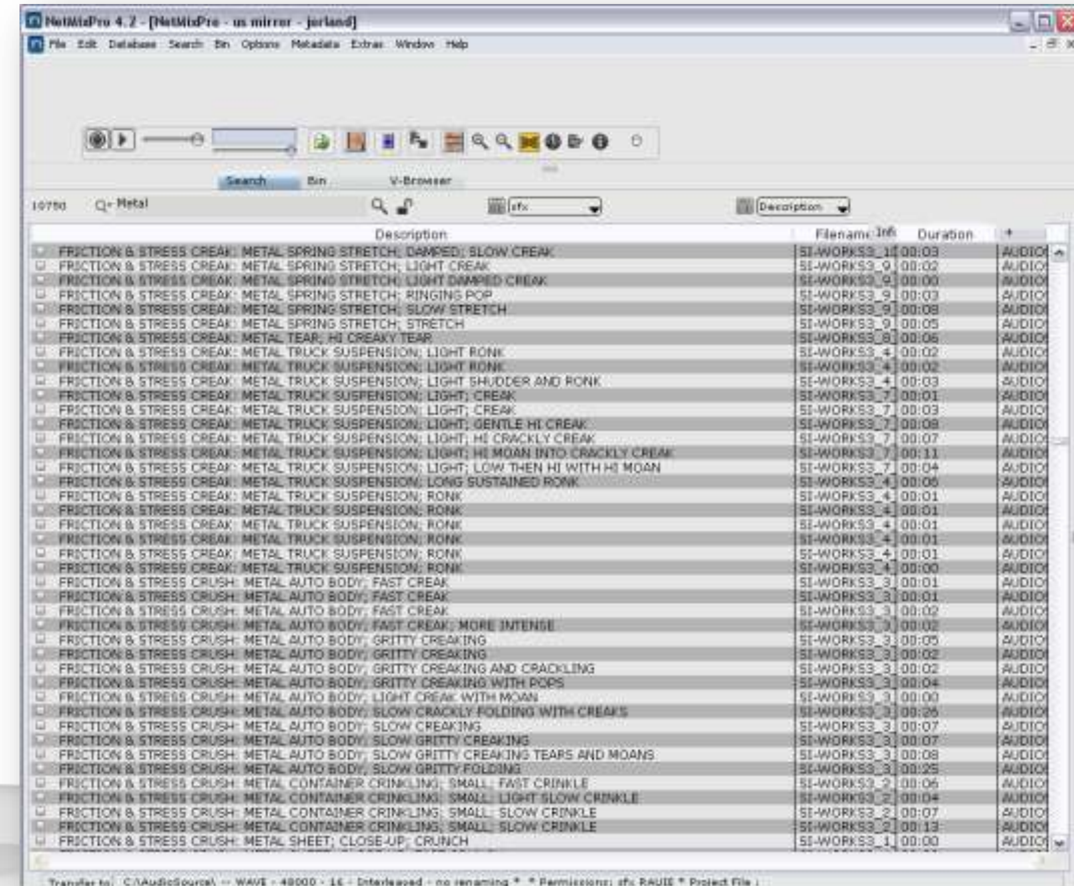
Typical sample files browsers:



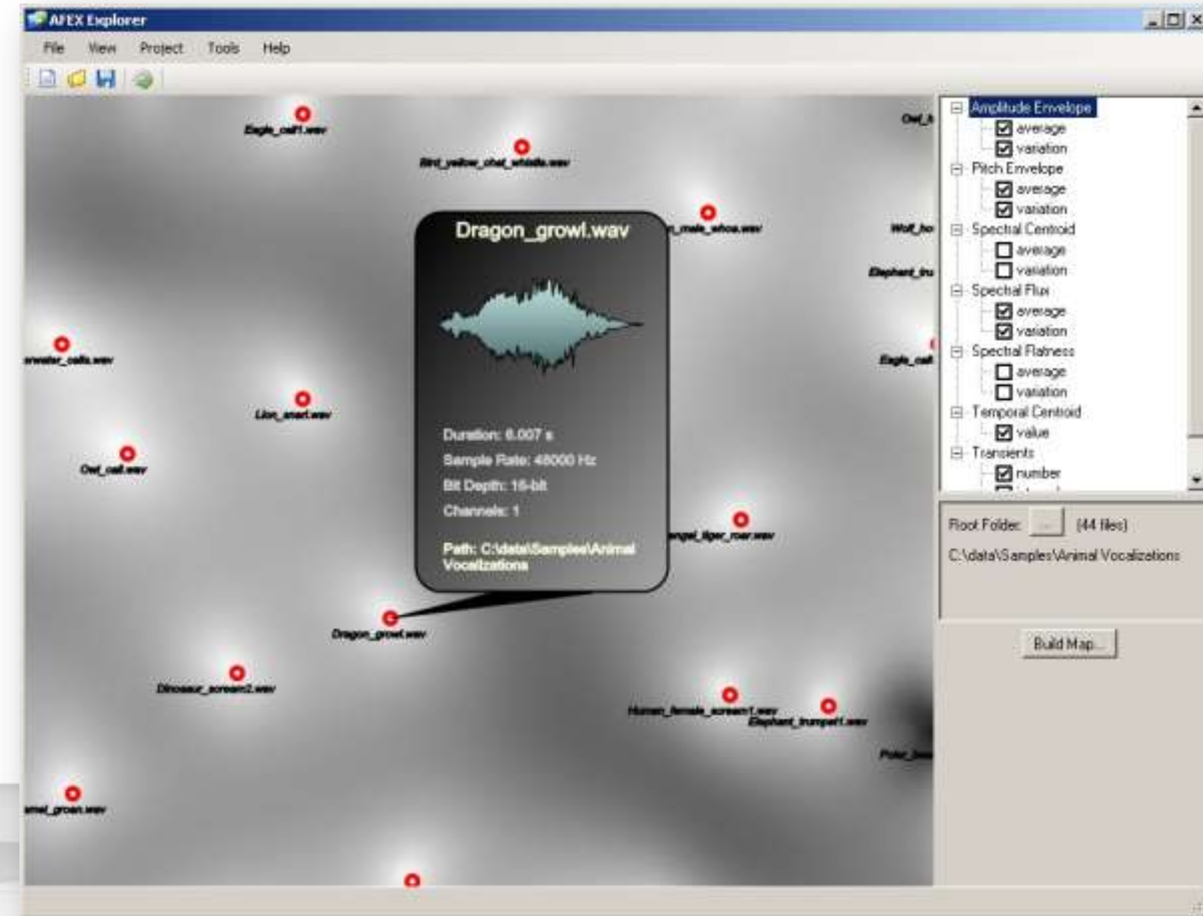

[AFEX Browser]

# Browsing sound effects libraries



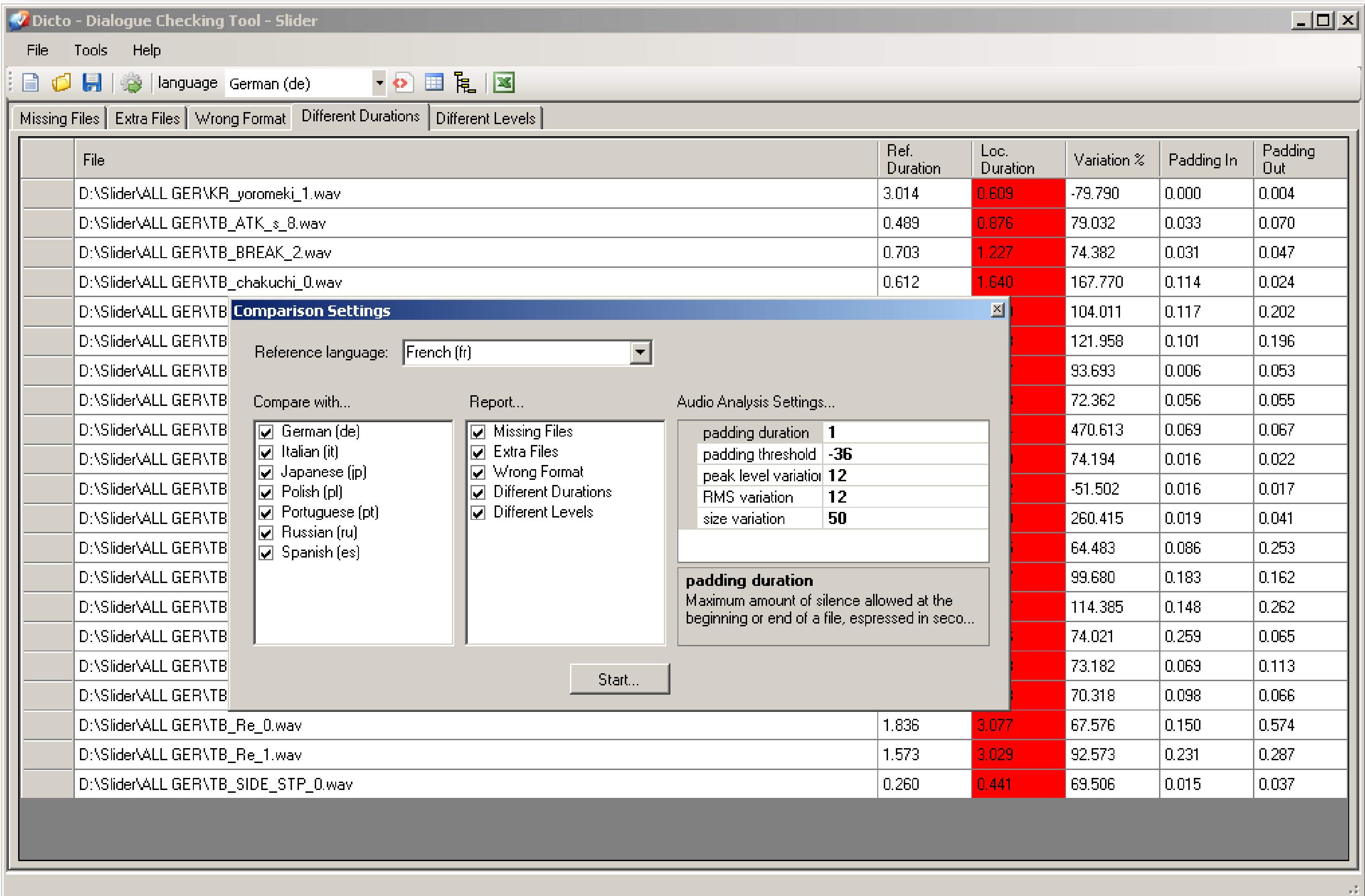

Worldwide Studios
Creative Services Group

# Example: AFEX Explorer




Worldwide Studios
Creative Services Group

# Dicto: checking dialogue assets



# Some of the games Dicto helped


Worldwide Studios
Creative Services Group

SONY

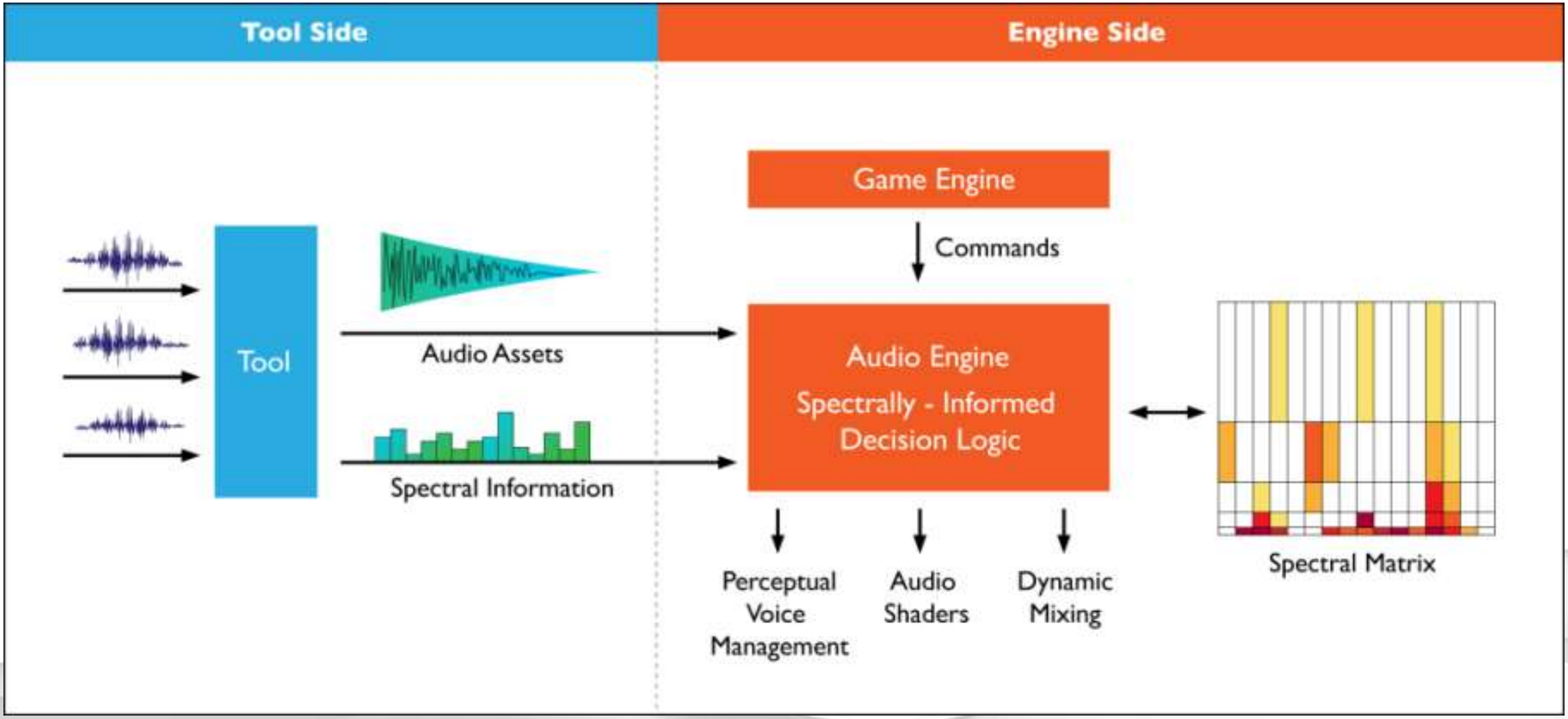
# Audio analysis at run-time

- Most audio engines are…

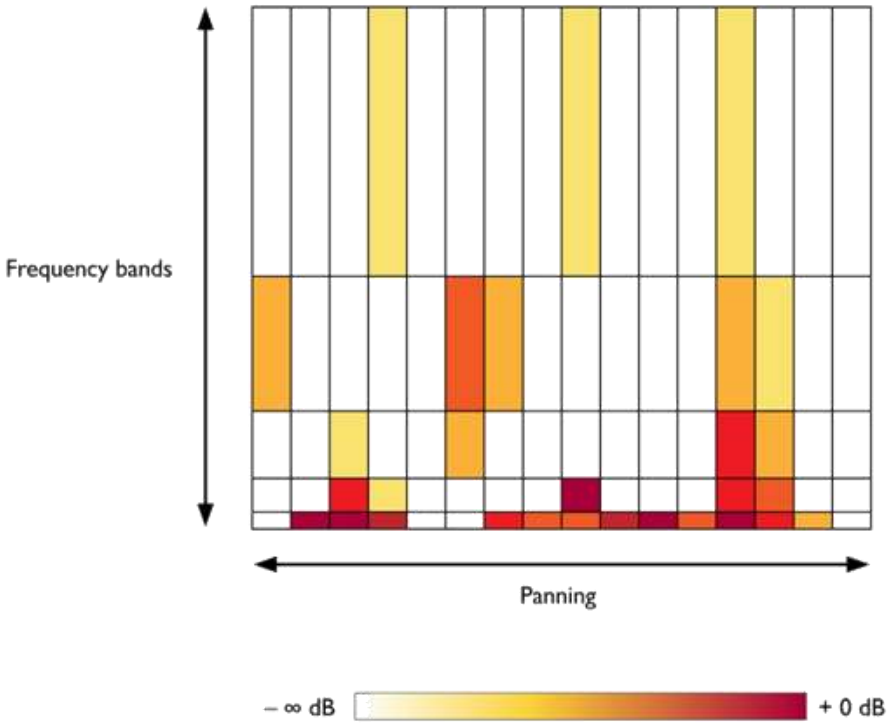
deaf !

They make decisions that impact the audio output of a game without knowing what they are playing.

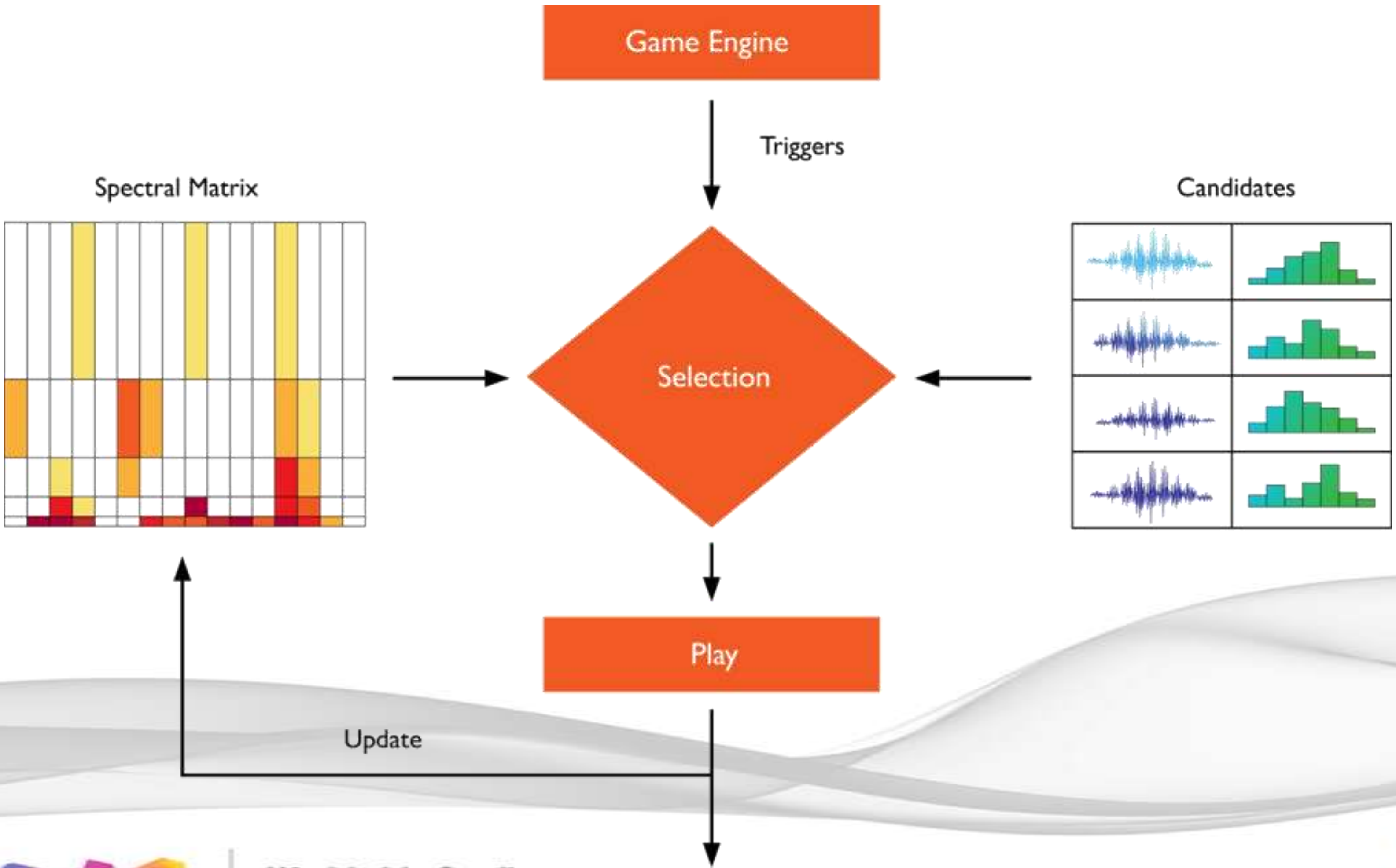
# Audio Engines



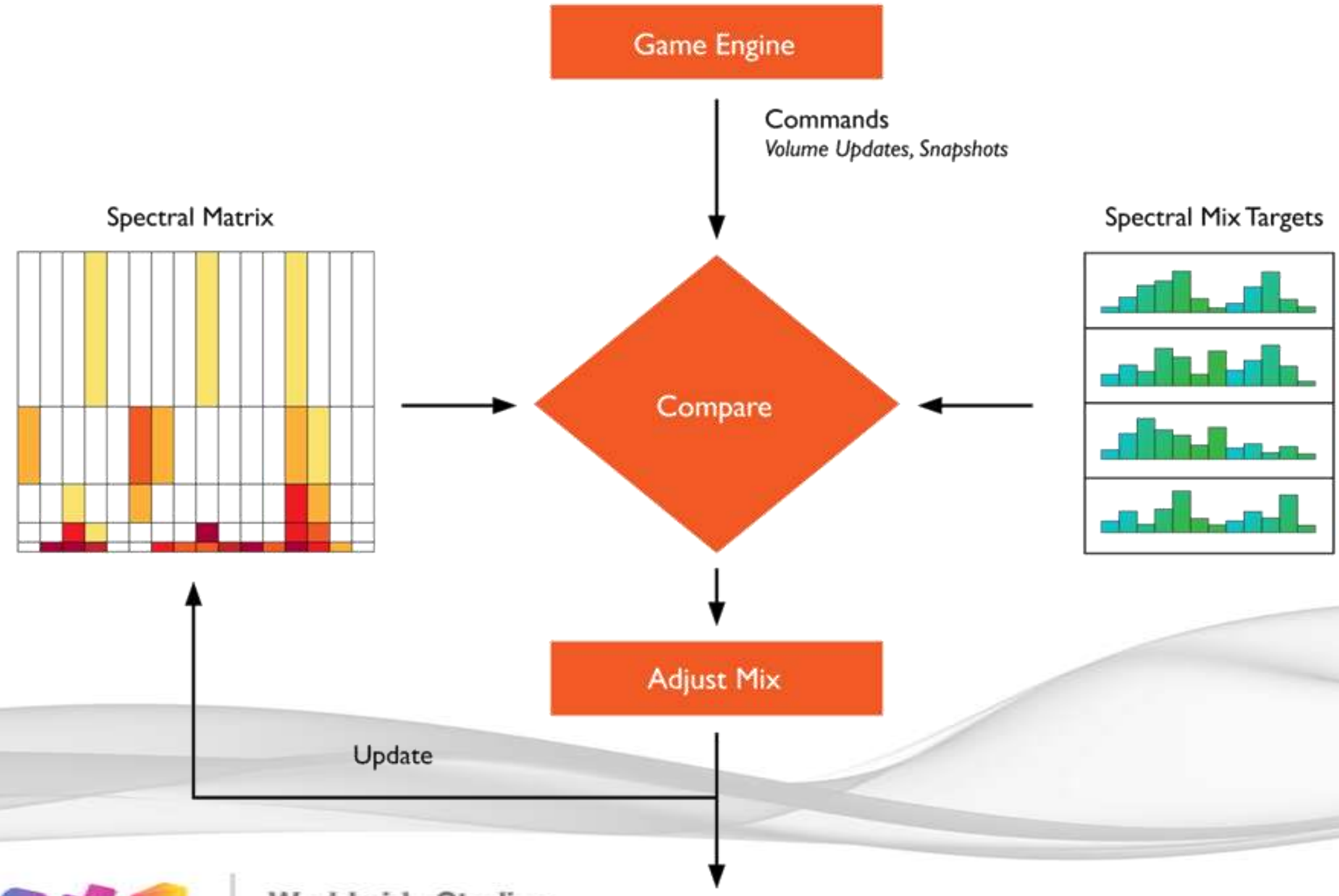
# Spectral Matrix



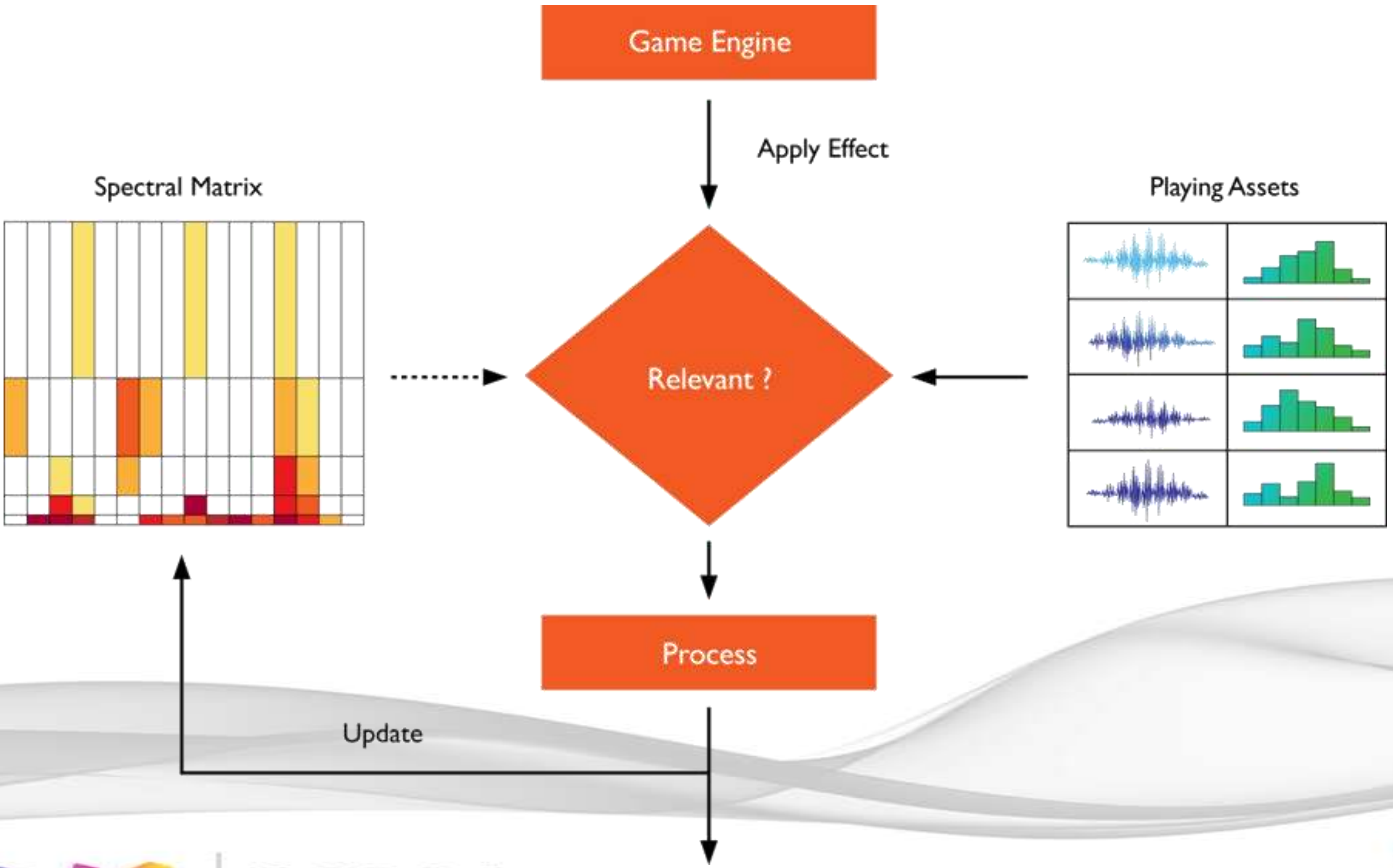
# Perceptual Voice Management



# Dynamic Mixing



# Audio Shaders



# Procedural Audio

# What is Procedural Audio ?

For sound effects:

- Real-time sound synthesis
- With control parameters coming from other sub-systems

- Examples of existing systems:
  - WWISE SoundSeed (Impact and Wind / Whoosh)
  - AudioGaming
  - SCEE's Spark

Applies also to music (algorithmic composition) and dialogue (Phonetic Arts)

# 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...)

# Two approaches to Procedural Audio

Bottom-Up:

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

Top-Down

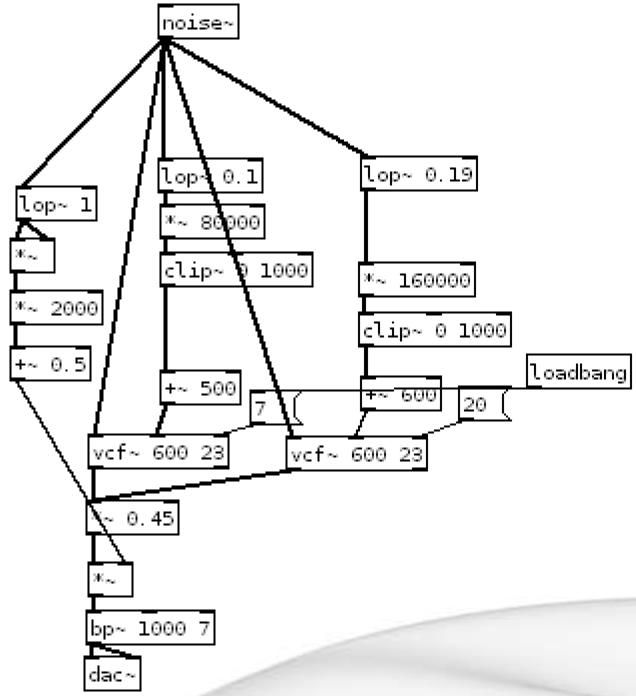
- analyse examples of the sound we want to create
- find the adequate synthesis system to emulate them

# Procedural Model Example : Wind

Good example of bottom-up versus top-down design

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

- Noise generator and bandpass filters (Subtractive synthesis)

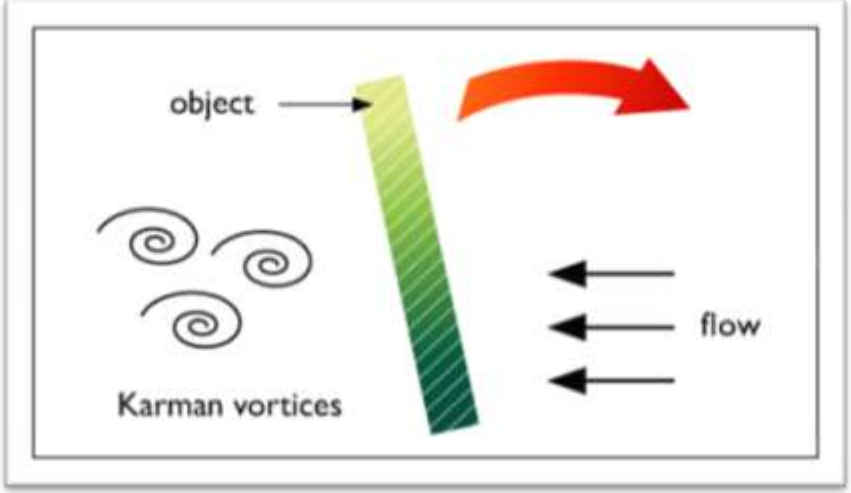


# Procedural Model Example : Whoosh

- 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

Subtractive synthesis (SoundSeed)

Aerodynamics computations

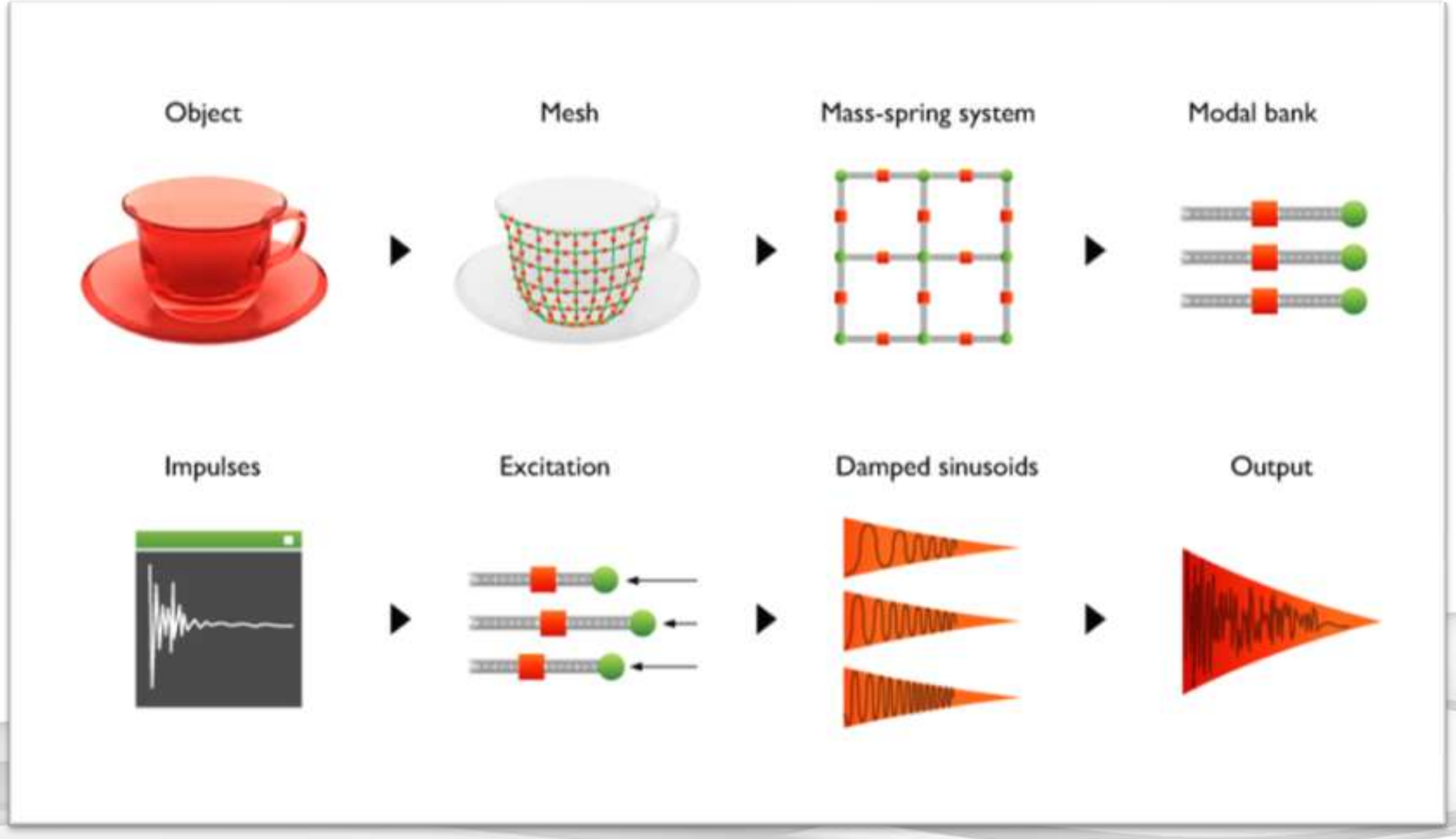
# Procedural Model Example
# Water / Bubbles

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

# Procedural Model Example : Solids



# Procedural Model Example : Solids

Other solutions for the analysis part:

- LPC analysis

  Source – Filter separation

- Spectral Analysis

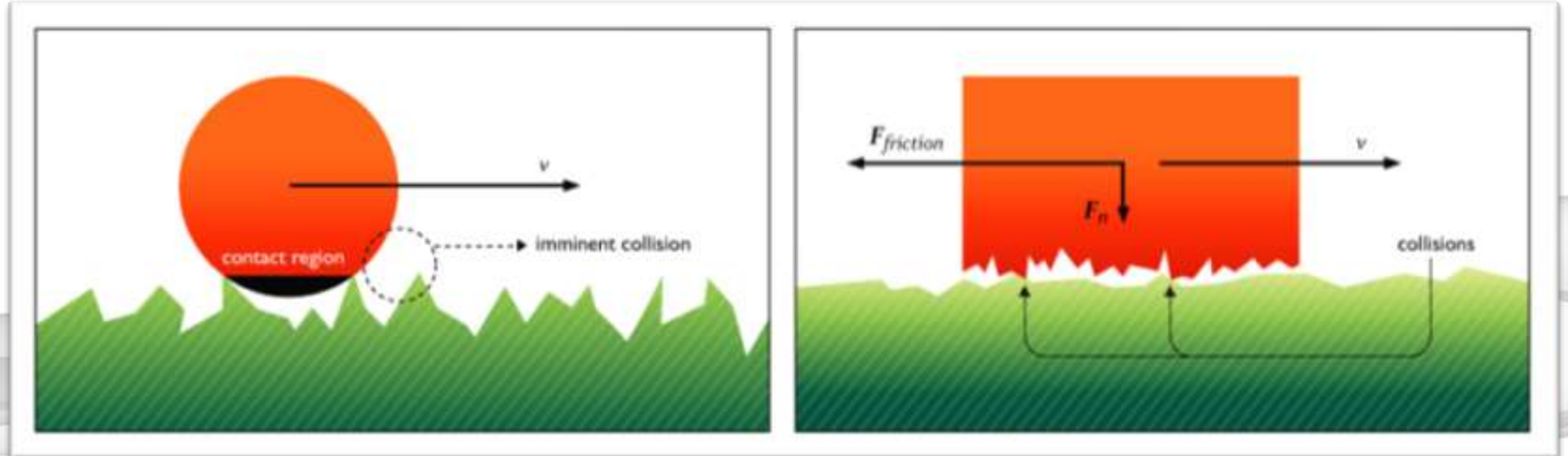
  Track modes, calculate their frequency, amplitude and damping

# Procedural Model Example : Solids

Different excitation signals for:

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

Interface with game physics engine / collision manager



# Procedural Model Example : Solids

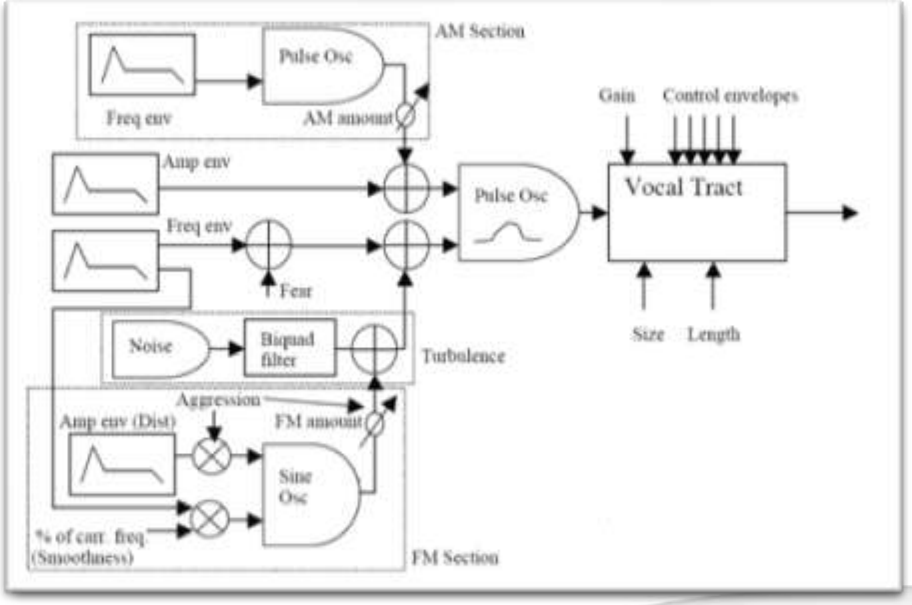
"Physics" bank for Little Big Planet on PSP:

- 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]

# Procedural Model Example : Creature

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



# Procedural Model Example : Creature

Eye Pet vocalizations:

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

Eye Pet waveforms

Synthasaurus

# Pros

- Offers a lot of advantages compared to static sounds (non repetitive, dynamic, lots of control parameters)
- One model = many sounds !
- Some models can be implemented very easily
  - Impacts / contacts
  - Footsteps
  - Air / Water
- Procedural audio is not necessarily more CPU expensive

# Cons

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

# Combining
# Procedural Audio & Audio Analysis

Worldwide Studios
Creative Services Group

SONY

# One of the main tasks of sound designers:

Transforming static sounds into dynamic ones

# 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:

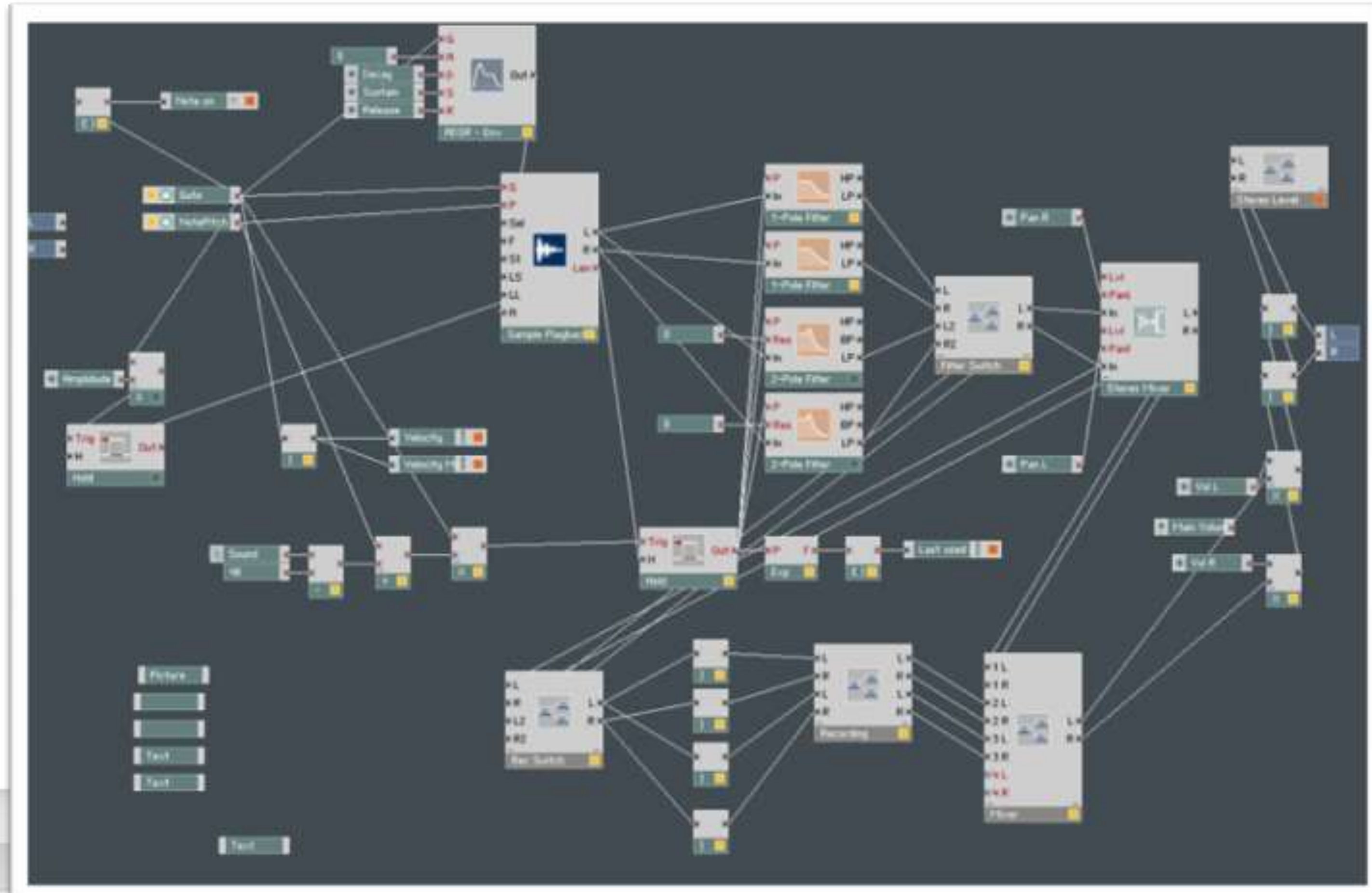
- 11 concurrent scripts
- each “grain” has its own set of parameters

Worldwide Studios
Creative Services Group

# Implementation with Patching

- Tools such as Pure Data / MAX MSP / Reaktor
- Better visualisation of flow and parallel processes and where the control parameters arrive in the model
- Sometimes hard to understand due to the granularity of operators
- Requires a PhD in mechanics, animal anatomy, physics etc..

# A “simple” patch in Reaktor…



# Another solution

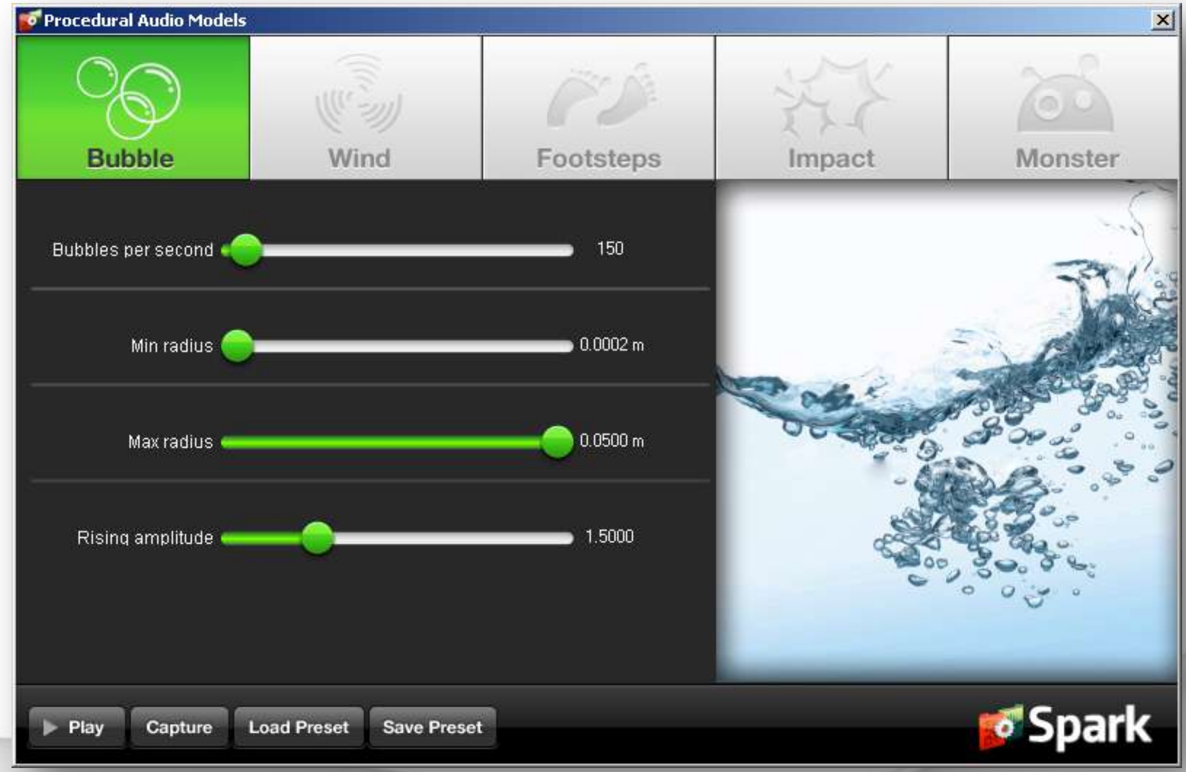
Vendors of ready-to-use Procedural Audio models:

- 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
- SCEE's Spark

# Spark

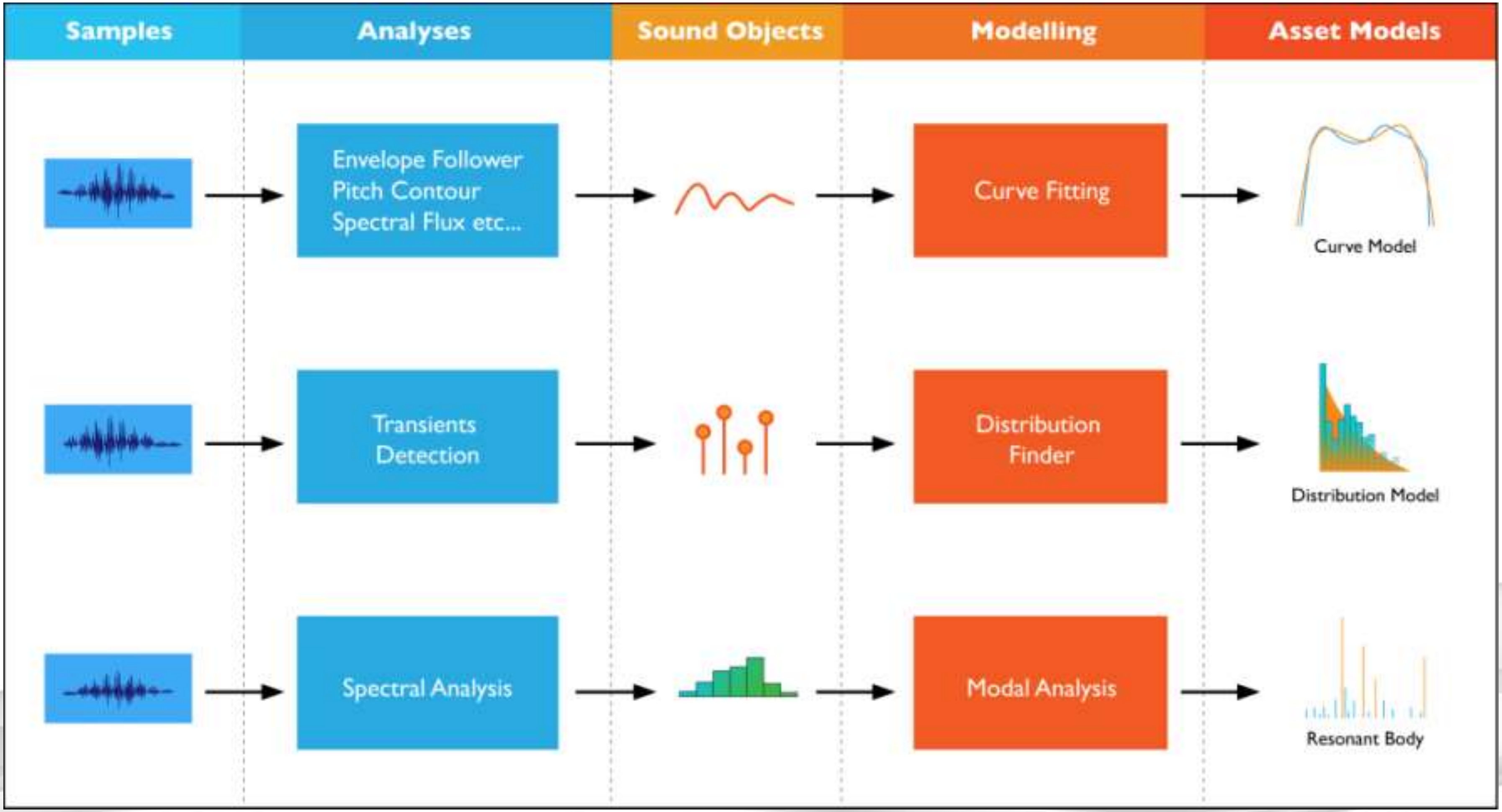

Procedural Audio Models
Bubble
Wind
Footsteps
Impact
Monster
Bubbles per second
150
Min radius
0.0002 m
Max radius
0.0500 m
Rising amplitude
1.5000
Play
Capture
Load Preset
Save Preset
Spark

# Going further…

Need for higher-level tools that let the designer:

- create its own dynamic model of a sound
- specify its own control parameters
- without having an extensive knowledge of synthesis / sound production mechanisms
- without having to rely on third party models

# Think asset models, not assets



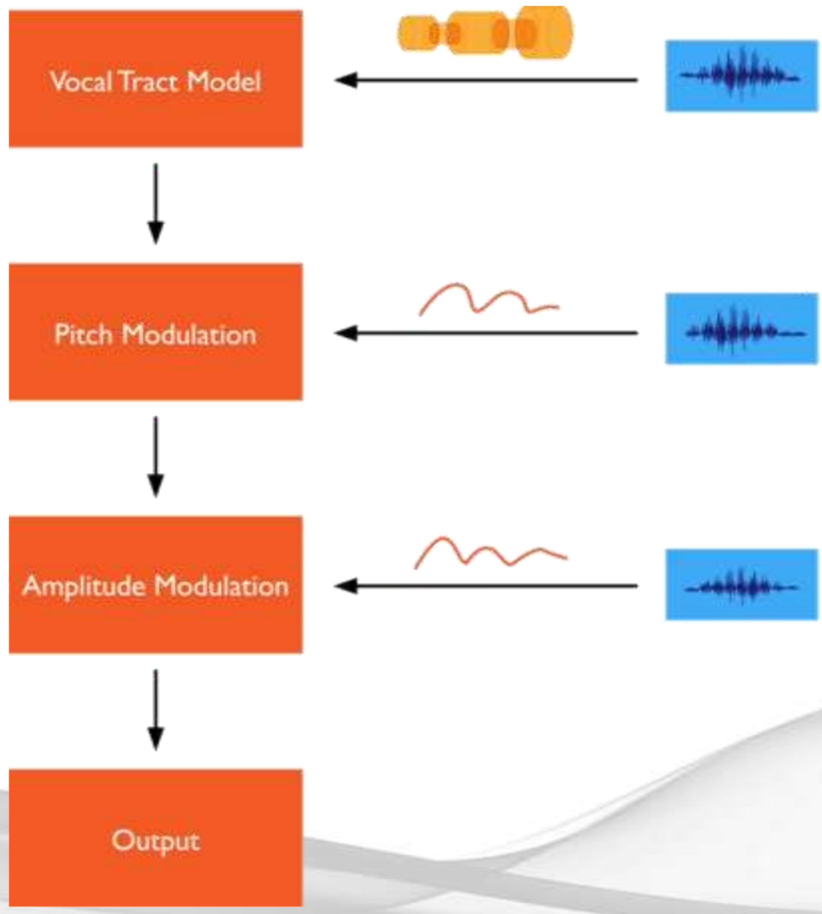

Worldwide Studios
Creative Services Group

SONY

# Because we are using analysis…

- We can use our own sounds as basis for a model
- The modules can implement more complex behaviours
- We can have a smaller number of modules

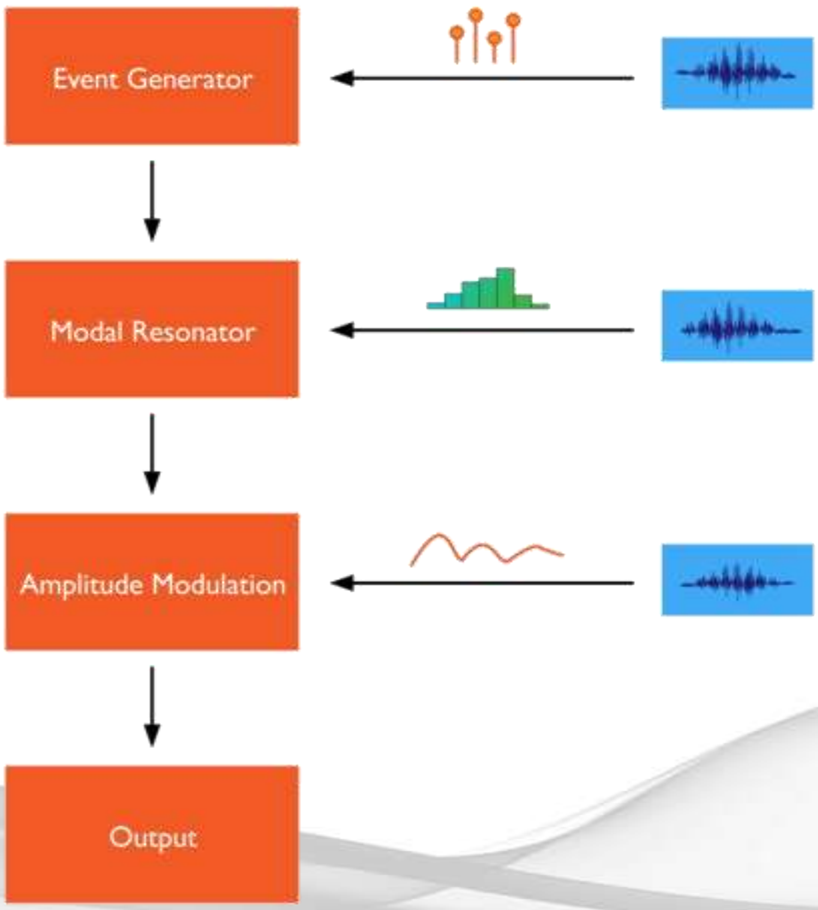
# Creature Vocalisations



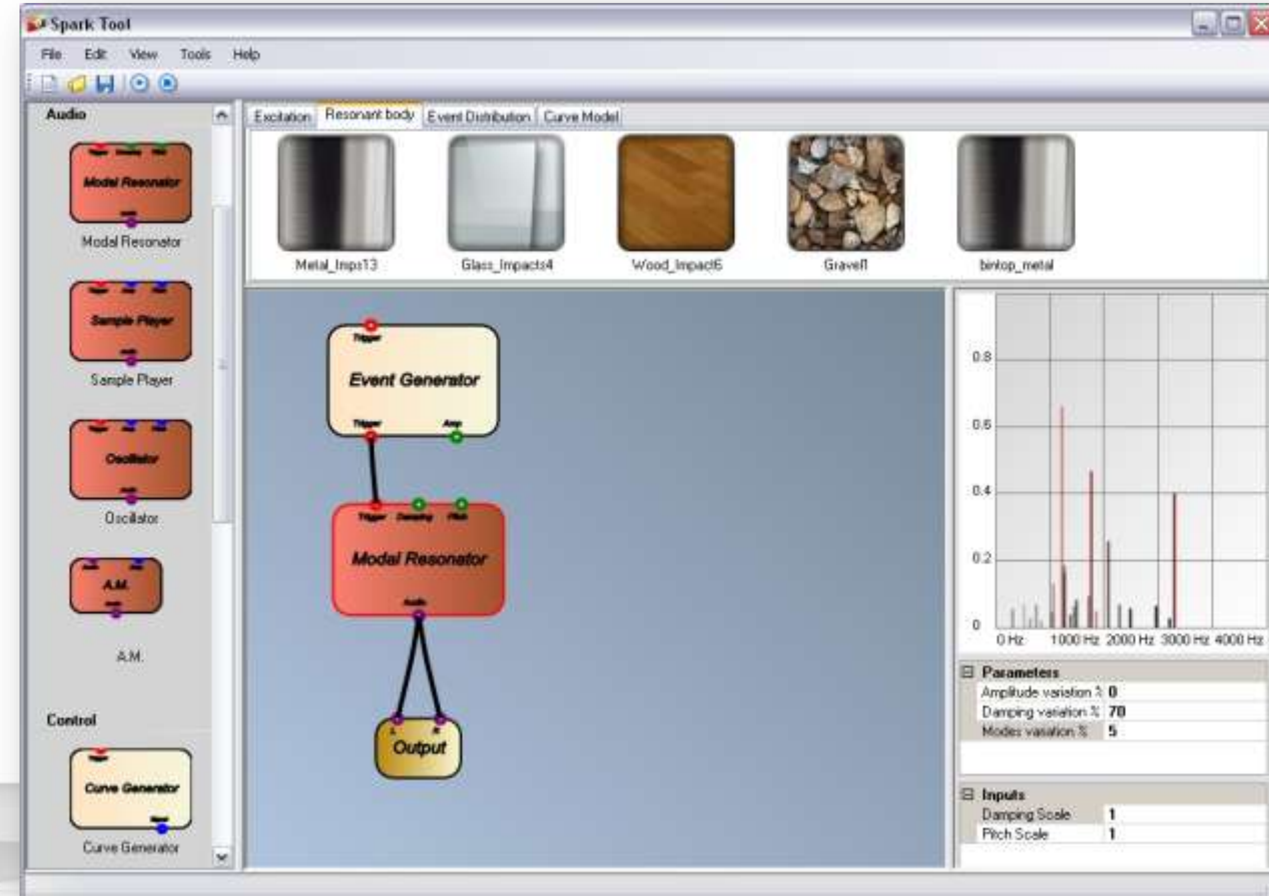

Worldwide Studios
Creative Services Group

SONY

# Debris / impacts



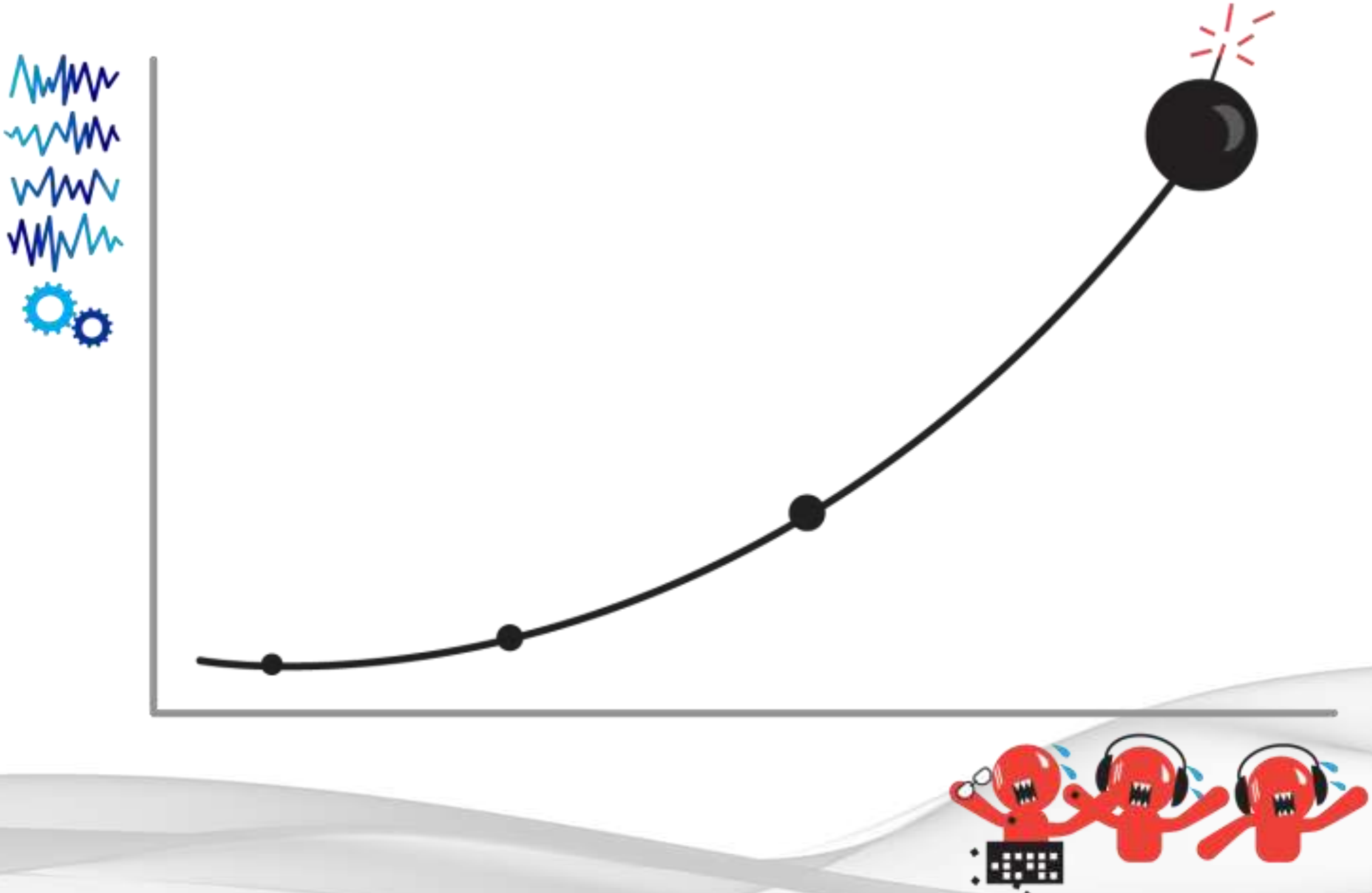
# Spark Tool = Spark + AFEX



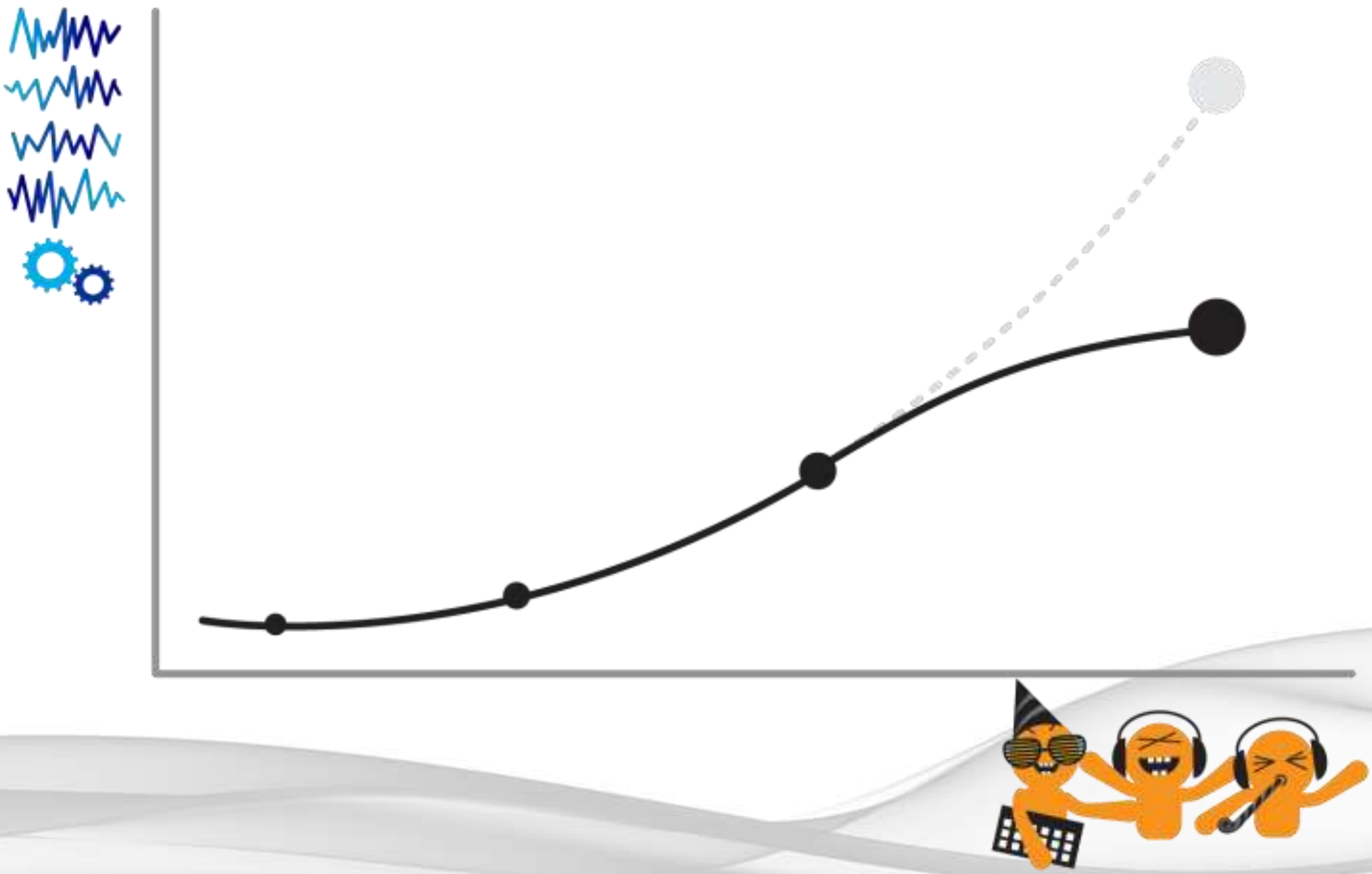

Worldwide Studios
Creative Services Group

# Conclusions

# How do we diffuse that bomb ?



# Smarter Audio Pipelines & Engines !



# Future Technologies for Game Sound Design

- Data-driven as much as possible
- Better integration with other game sub-systems
- Knowing your data is the key to smarter tools and engines (audio analysis)
- Generate audio at run-time when it makes sense (procedural audio using models created with audio analysis)

# Thank you!

# Any questions?

Contact: nicolasfournel@yahoo.com
nicolas_fournel@scee.net