Jogos e Simulação – Audio

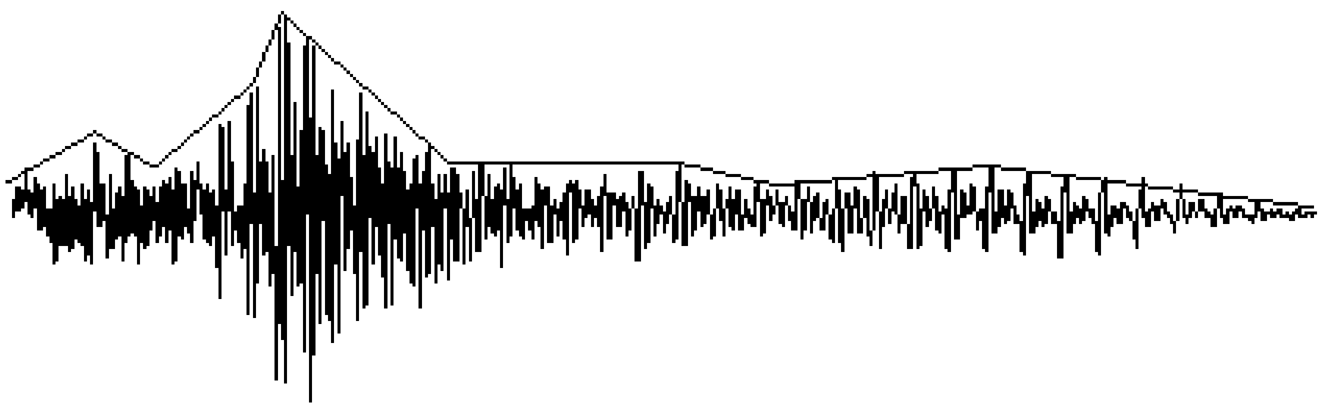
Sound Synthesis

and

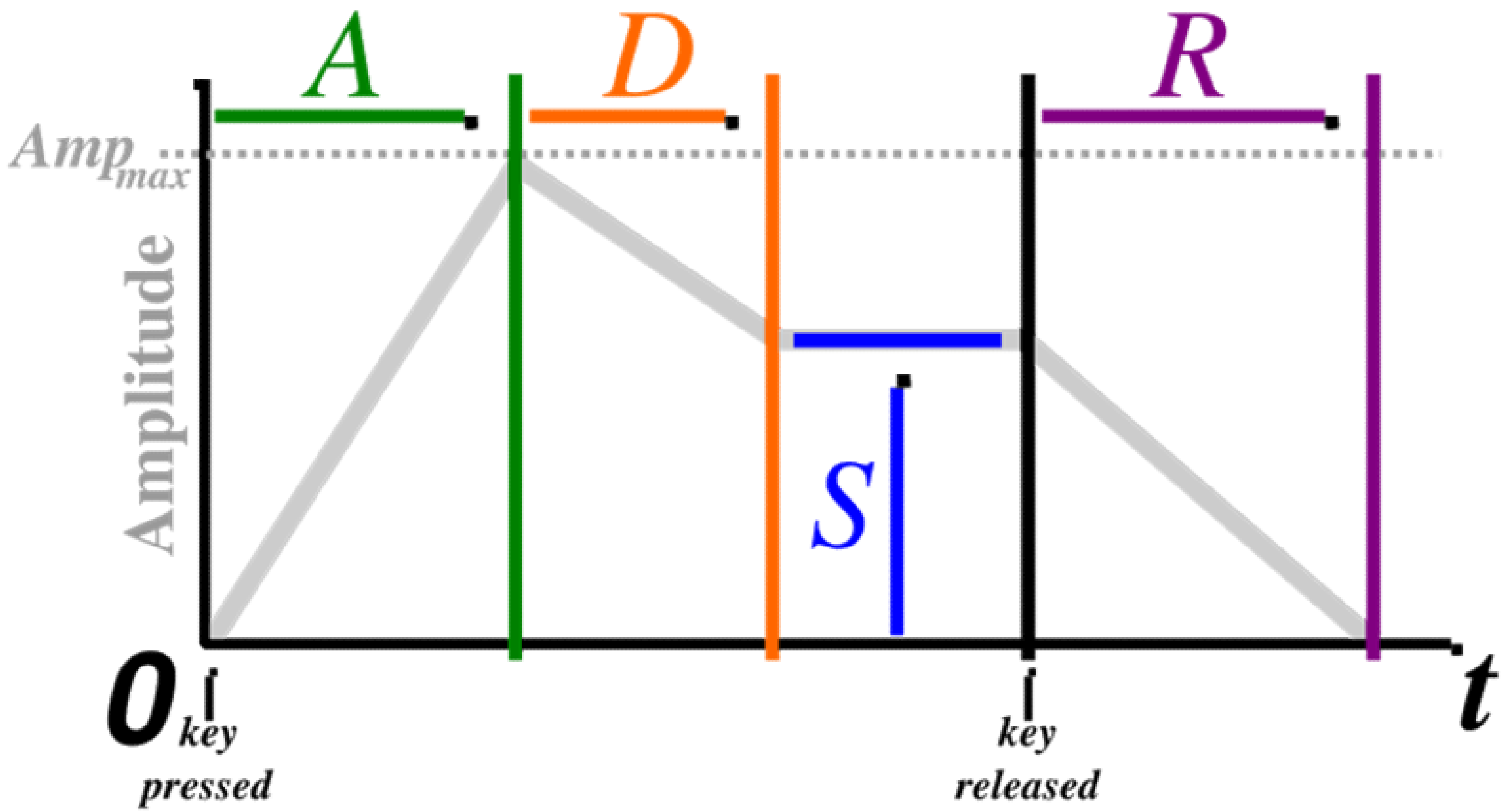
Modeling

Background

- (Amplitude) envelope – the function (of time) that describes how the maximum amplitude of the waveform changes over time.



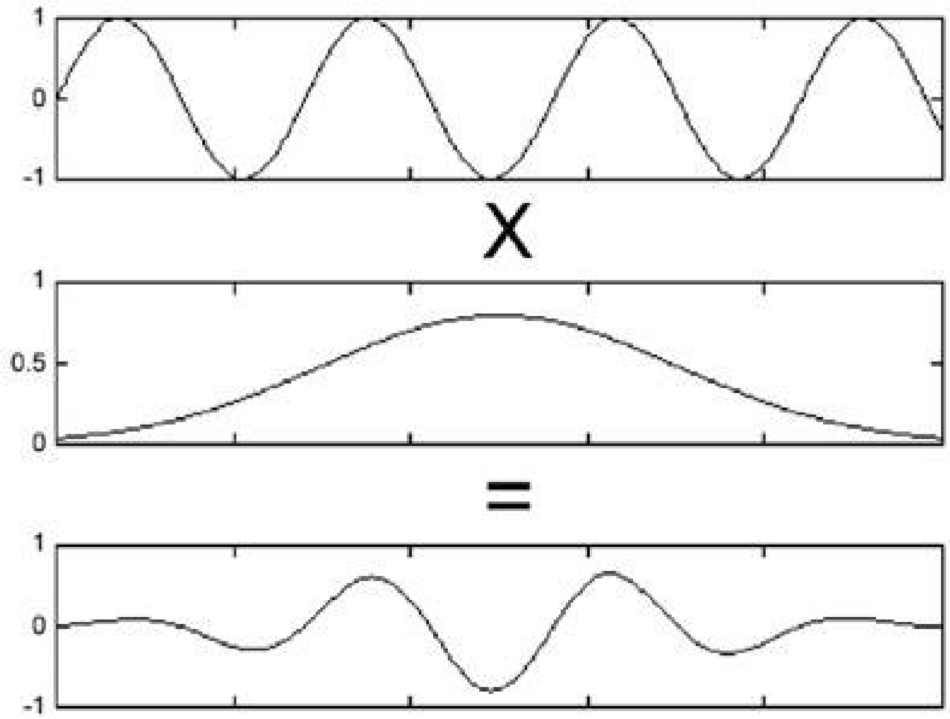
Background



- Legend:
  - A – attack
  - D – decay
  - S – sustain
  - R – release

Background

- (Amplitude) envelopes can be used to change the amplitude shape of the waveform (with point-to-point multiplication).



Sound synthesis techniques

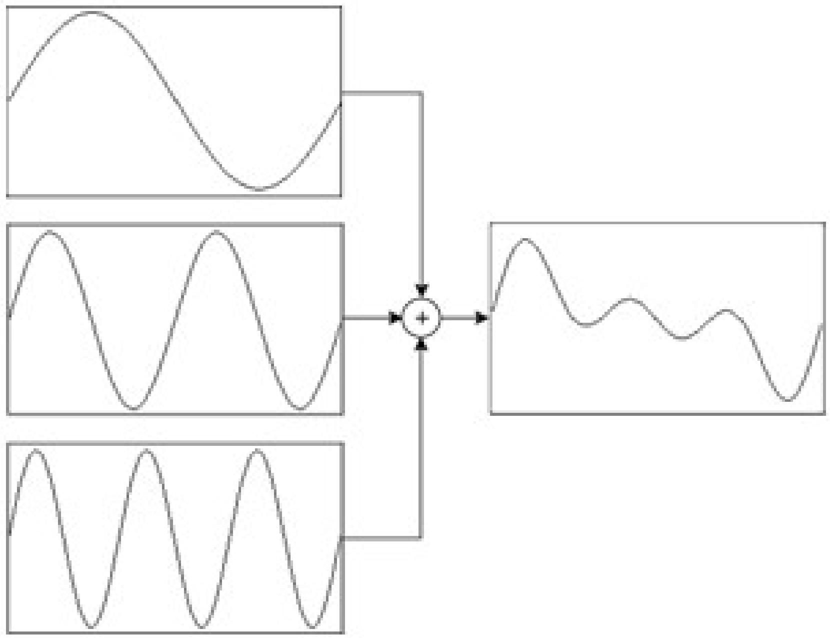
- Additive synthesis – sound generated by adding sine wave (it may use wavetable synthesis).
- Wavetable synthesis – adds samples from a wavetable.
- Concatenative synthesis – concatenation of short samples (ex. for speech and music). (Better methods for speech exist – with models of the vocal tract.)
- Granular synthesis – combines sound grains (1 to 100 ms) to make a sound (texture). Grains can overlap and are shaped by an envelope to avoid clicks and control amplitude. (Differs from concatenative synthesis in the way that the samples/grains are concatenated.)

Sound synthesis techniques

- Frequency modulation – given an initial simple waveform, it changes the spectrum by shaping its instantaneous frequency.
- Physical modeling synthesis – it uses mathematical models (physics).
- Data driven synthesis – sinusoidal modeling synthesis, spectral modeling synthesis, among other techniques (more details later). It starts from a real sound, and modifies the original sound.

There are many other techniques…

Additive synthesis

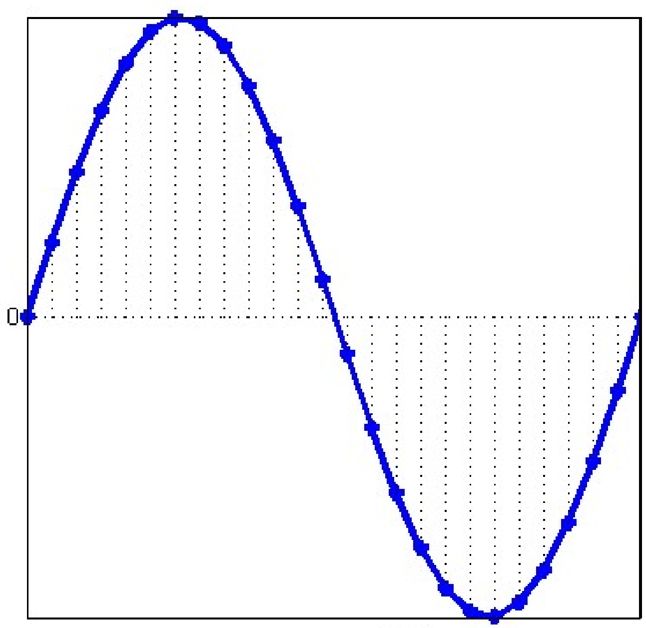


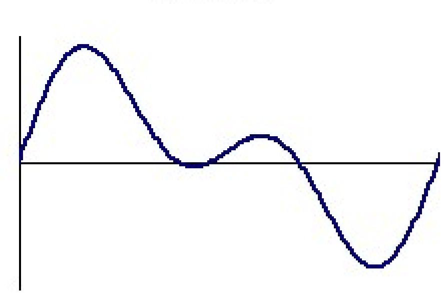
$$y(t) = \sum_{n}^{N} A_n \sin(2\pi f_n t)$$

But, there's a (computationally) cheaper way!

Wavetable and oscillators

$s(t) = A \sin(2\pi f t + \theta)$



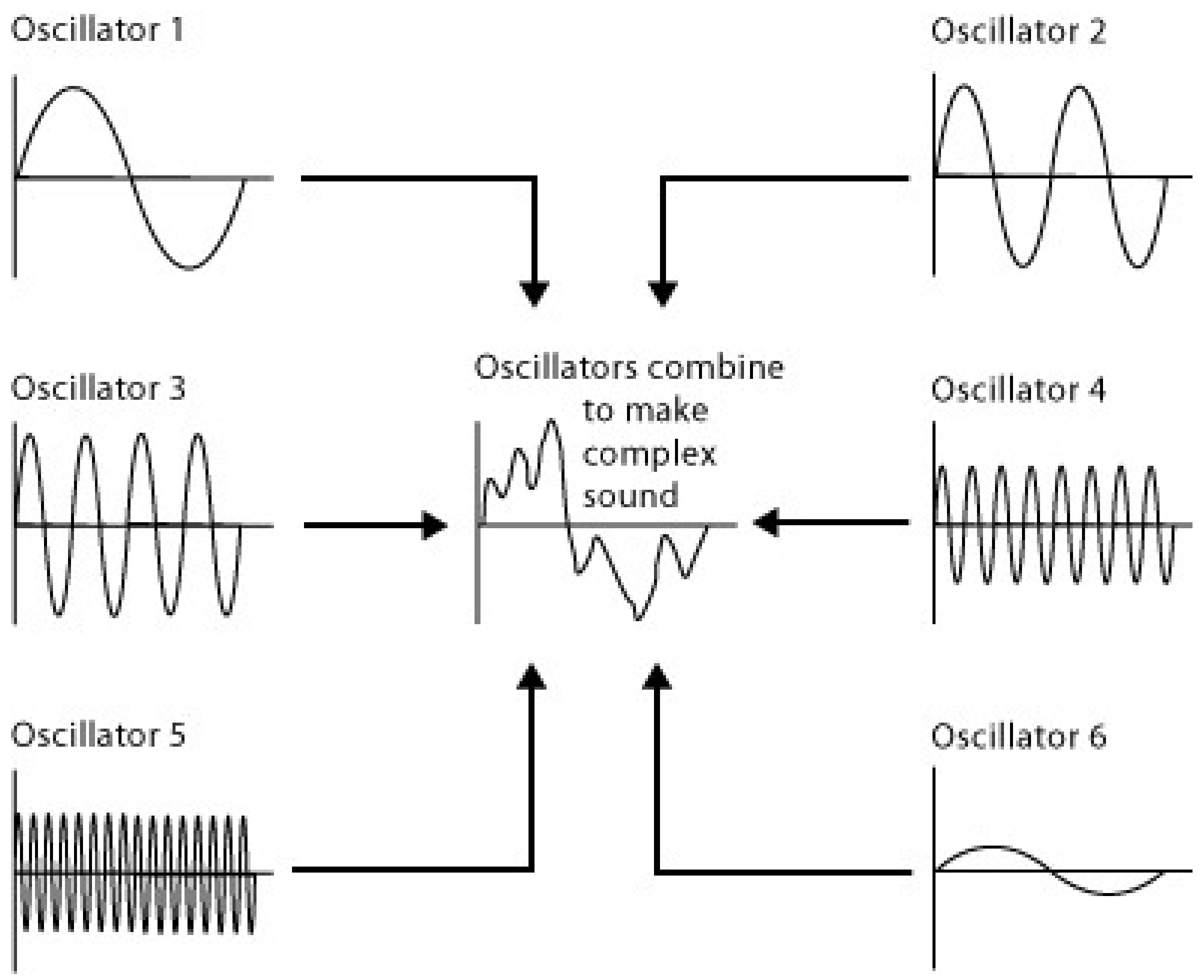


address	wavetable value
0	0.00
1	0.06
2	0.11
3	0.17
4	0.23
5	0.28
6	0.34
7	0.39
8	0.44
9	0.49
10	0.54
11	0.59
12	0.64
13	0.68
14	0.72

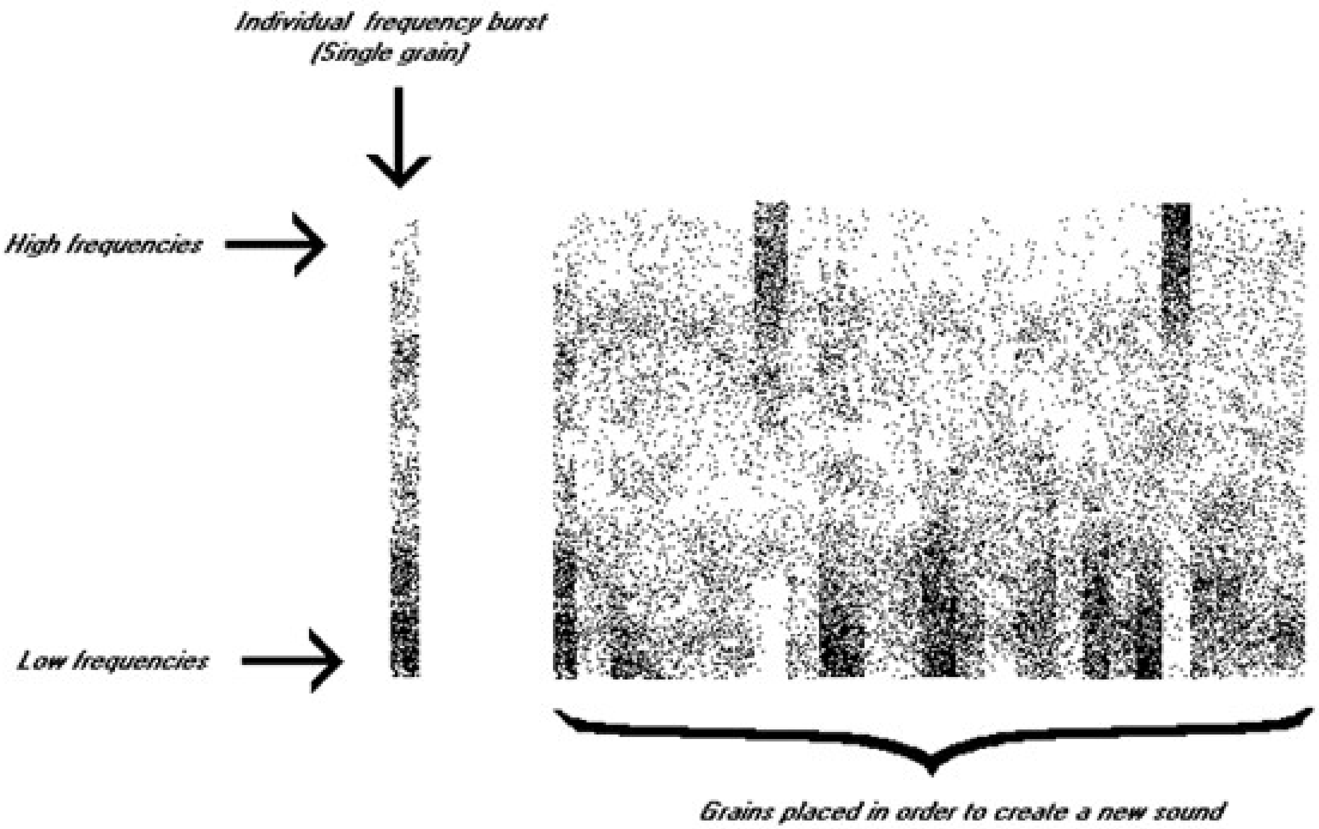
.
.
.

- Wavetable – a table that stores one cycle of the waveform.
- Oscillator – produces repetitive signal.

Additive synthesis



Granular synthesis



- Example:
  - Paul Lansky

Sound modeling

Physical modeling

physical properties

Data-driven

waveforms

model of sounds

Most common: physical modeling synthesis (with mathematical model)

Physical modeling and synthesis

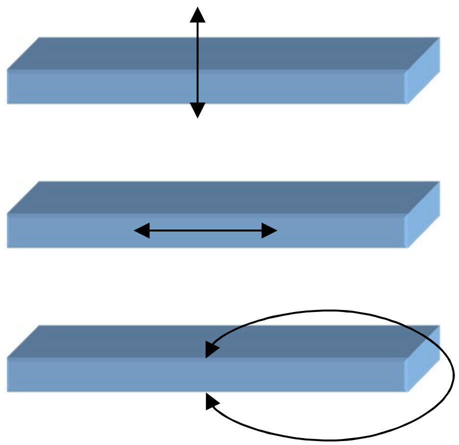
- https://www.youtube.com/watch?v=7xzKyIq9h3s
- https://www.youtube.com/watch?v=BjZ7CV6giII

Physical modeling and synthesis

The sound generated by an object when it is struck depends on the way the object vibrates.

Ex.: modes of vibration of bars and other highly symmetric objects (like rectangular blocks):

- Transversely
- Longitudinally
- Torsionally



This vibration is determined by:

- properties of the object (structural invariants):
  - size,
  - geometry,
  - material (elasticity, internal friction)
- type and properties of the event (transformational invariants)
  - location of impact
  - way (strength,...) the object is struck.

Physical modeling and synthesis

- Gaver (94) proposed a physically motivated synthesis model for solid objects:

$$\mathcal{M} = \{\mathbf{f}, \mathbf{A}, \tau\}$$

where:
f – vector of frequency partials
A – matrix of gains (initial amplitudes) for each partial at different locations (the gains of different modes depend on the location of contact)
τ – vector of decay factors

- The response for an impulse at location *k* is:

$$y(t) = \sum_n a_{nk} e^{-t/\tau_n} \sin(2\pi f_n t)$$

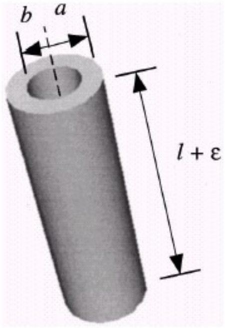
Simple geometries

$$y(t) = \sum_n A_n e^{-t/\tau_n} \sin(2\pi f_n t)$$

$$f_n = \frac{\pi}{2l^2}\sqrt{\frac{Q\kappa^2}{\rho}}\beta_n^2$$

$$A_n = (-1)^{n-1}\frac{lU}{\pi^2\beta_n^2}\sqrt{\frac{8\rho}{Q\kappa^2}}$$

$$\tau_n = 16\times 10^8(\pi\rho / 4\kappa^2 f_n^3)$$



f – frequency

A – amplitude

τ – decay rate

l – length

κ– radius of gyration $\kappa = \frac{1}{2}\sqrt{a^2 + b^2}$

a – outer radius

b – inner radius

Q – Young's modulus of elasticity

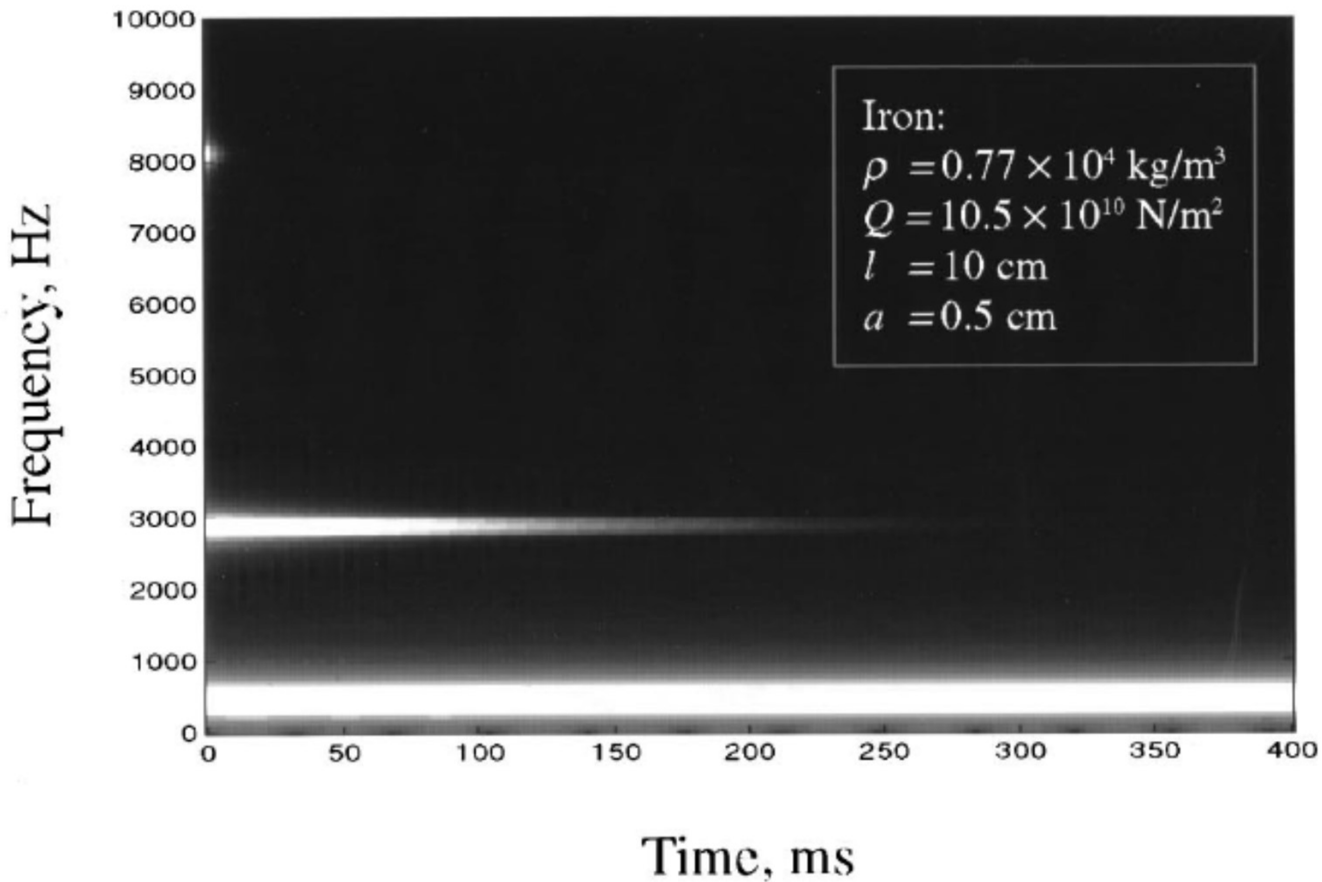
ρ – density

U – gain $U = P/\left[\rho\pi(a^2 - b^2)\right]$

P – force

Simple geometries

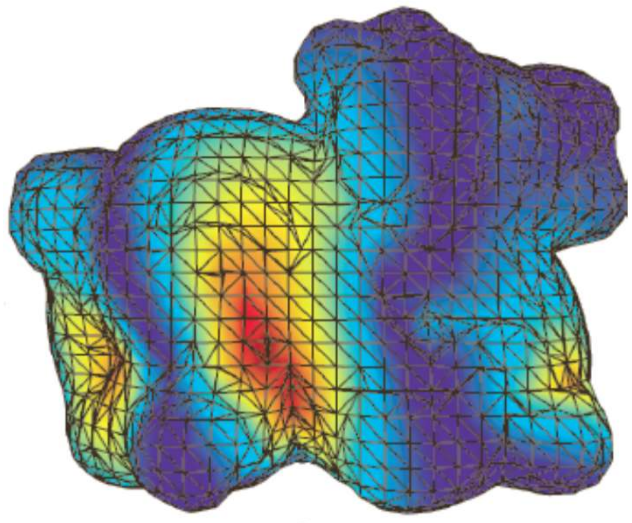
(Lutfi, 2001)



synthesized

Complex geometries

- James, Barbič and Pai
  - rigid body simulators for – modes of vibration and parameters
  - very realistic
  - computationally expensive



Complex geometries

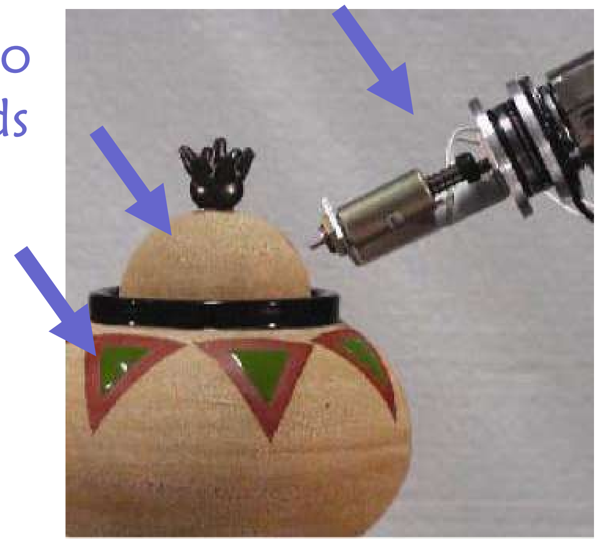
- van den Doel, Pai, Richmond and colleagues developed algorithms to synthesize sounds of contact interaction for:
  - Impact
  - Scraping, Sliding and
  - Rolling.

- Novelty: they model the contact interaction. Older models model only the resonance of the objects involved.

  Resonance model + contact model

- The sounds obtained provide important perceptual cues that combined with graphics and dynamics simulation give a better illusion of reality.

Complex geometries



- They segment the surface of the (real) object into areas of different textures and get samples of sounds in those areas with a contact microphone.
- They fit the parameters of the model to the recorded sounds.

$$y(t) = \sum_n a_{nk} e^{-t/\tau_n} \sin(2\pi f_n t)$$

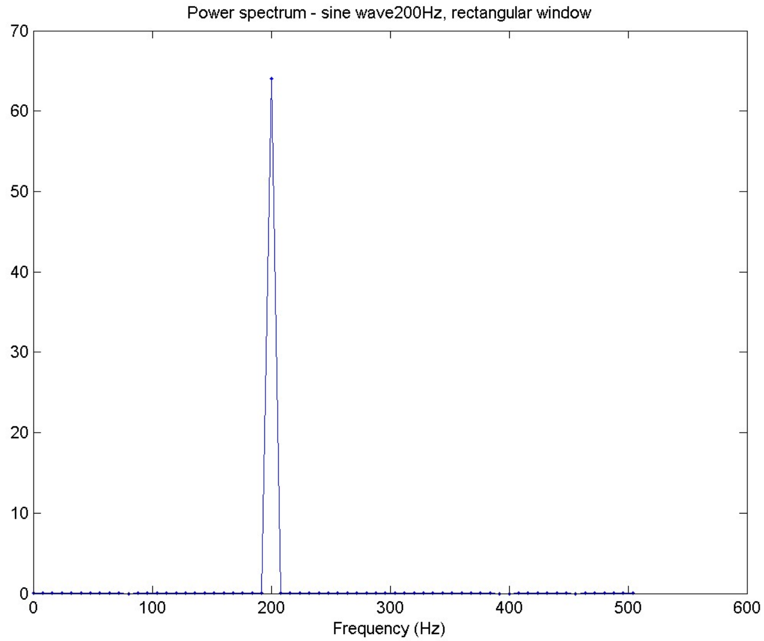
- They create a "sound-space" of the object: mapping points from the sound-space to points on the real object where samples have been recorded.
- acmesiggraph2001.mpg

Parameter estimation & synthesis

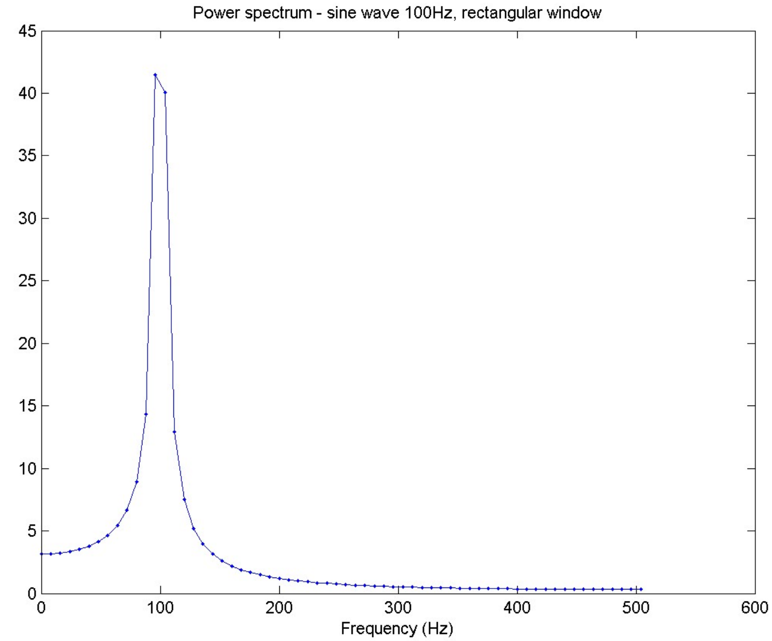
Problems with FT:

- leaking

Fourier component

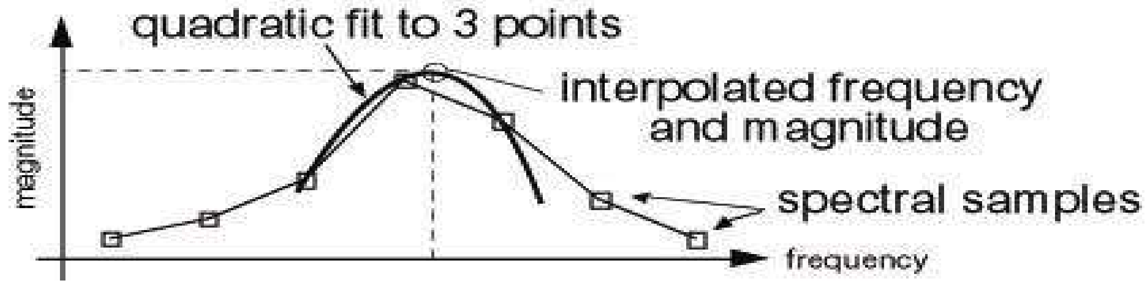


Not Fourier component



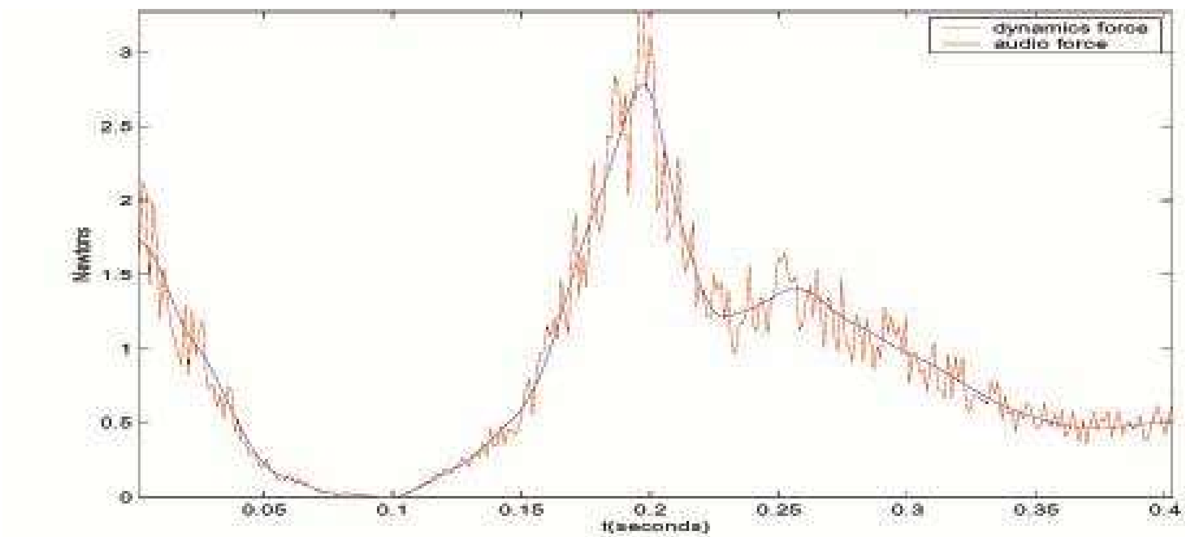
Parameter estimation & synthesis

- Parameter estimation: The modal frequencies are first estimated from the average power spectrum of the recorded sounds
  - Corrected for background noise
  - Using quadratic interpolation to identify the peaks given by the DFT.



- Synthesis:
  - The estimated parameters are used to simulate contact on points where measurements were taken.
  - At intermediate points (i.e., locations between mesh vertices) they interpolate the gains from the vertex gains.
    There is no need for frequency interpolation because the gains a_{nk} at different vertices have the same modal frequencies

Simulation of contact interactions



- Modal resonance models are not sufficient. We also need a model of the contact interactions.
  The detailed micro-collision dynamics of contact interactions are so complex that deriving it from physical laws seems infeasible.
- Stochastic models may be appropriate because contact interactions have some randomness.
- The audio-force is used to excite resonance models

Sinusoidal modeling and synthesis

- Phase vocoder – signal modeling technique developed for analysis and synthesis of speech.
- Phase vocoder limitations – good for harmonic signals with static pitch BUT natural sounds can be inharmonic and can have slowly time-varying frequencies (not purely periodic).
- Alternatives to phase vocoder that deal with inharmonic and pitch-changing sounds:
  - Sinusoidal modeling and synthesis
  - Spectral modeling synthesis
- Other alternatives:
  - George and Smith, 92, Ding and Qian, 97…

Sinusoidal modeling and synthesis

- Sinusoidal modeling and synthesis – deals with inharmonic and pitch-changing sounds.
  - MQ modeling (McAulay and Quateri)
  - PARSHL (Smith and Serra)

waveforms

↓

Analysis: representation/model of sounds

↓

Modifications to the representation (optional)

↓

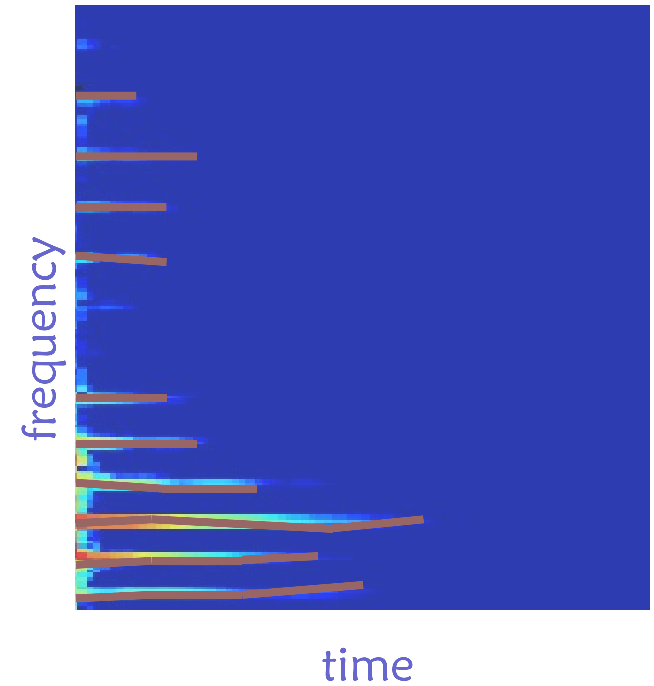
Synthesis: sounds (waveforms)

Sinusoidal modeling and synthesis

- Modeling sinusoids:

Sum of sinusoids with slowly varying amplitude and frequency

$$x(t) = \sum_{r=1}^{R} A_r(t) \cos \theta_r(t)$$



- A – instantaneous amplitude
- θ – instantaneous phase

- Analysis:

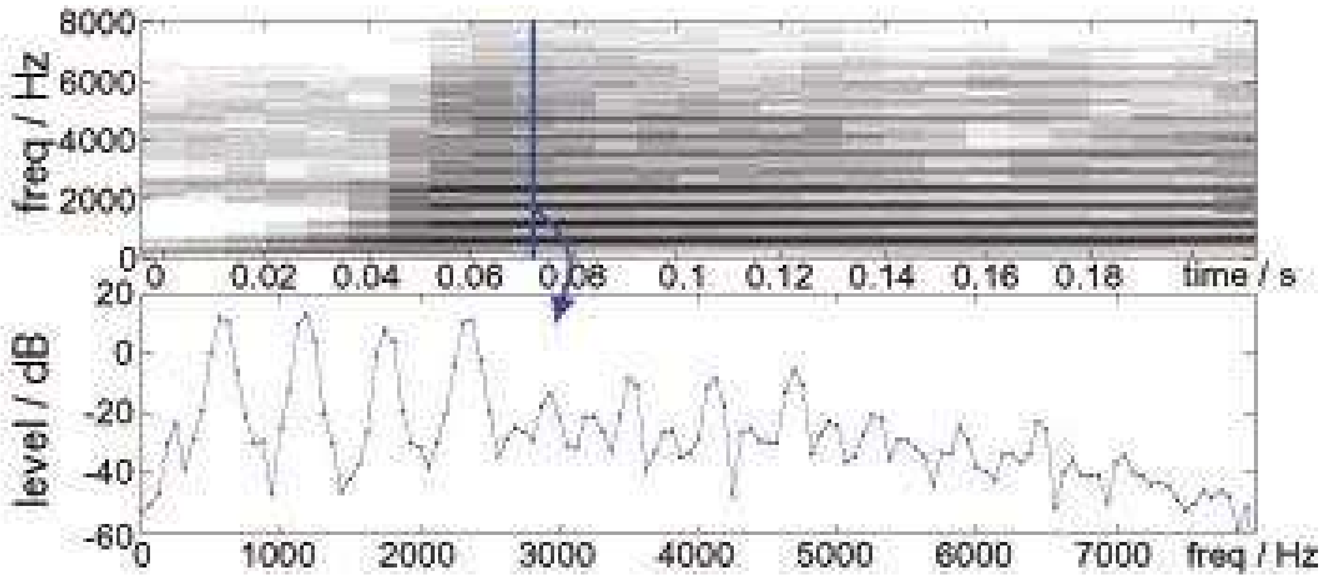
  Peak tracking algorithm
- Synthesis:

  bank of oscillators + additive synthesis

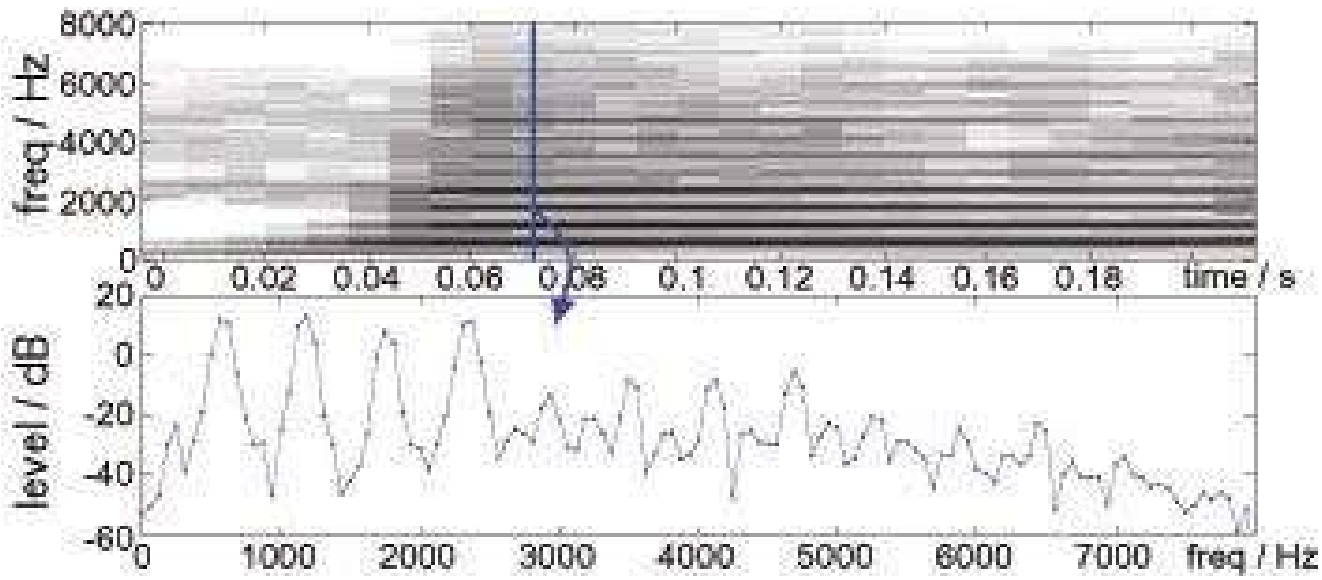
Peak tracking algorithm

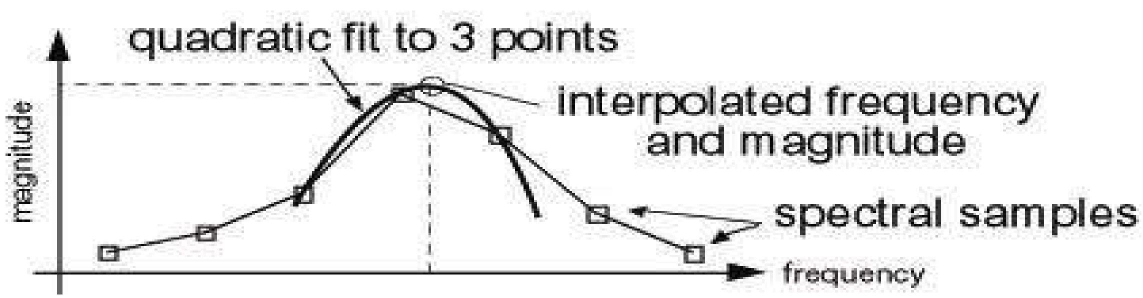
Look for horizontal energy ridges in the STFT

- 1st step:
  - Find the magnitudes and frequencies of the STFT peaks at each time step



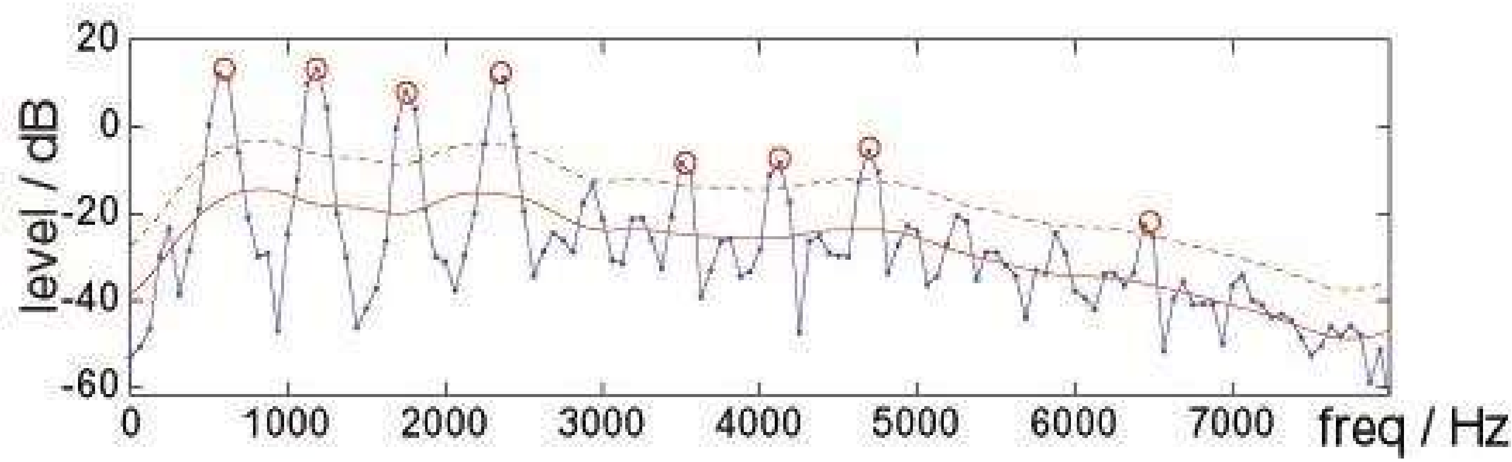
Peak tracking algorithm





Peak tracking algorithm

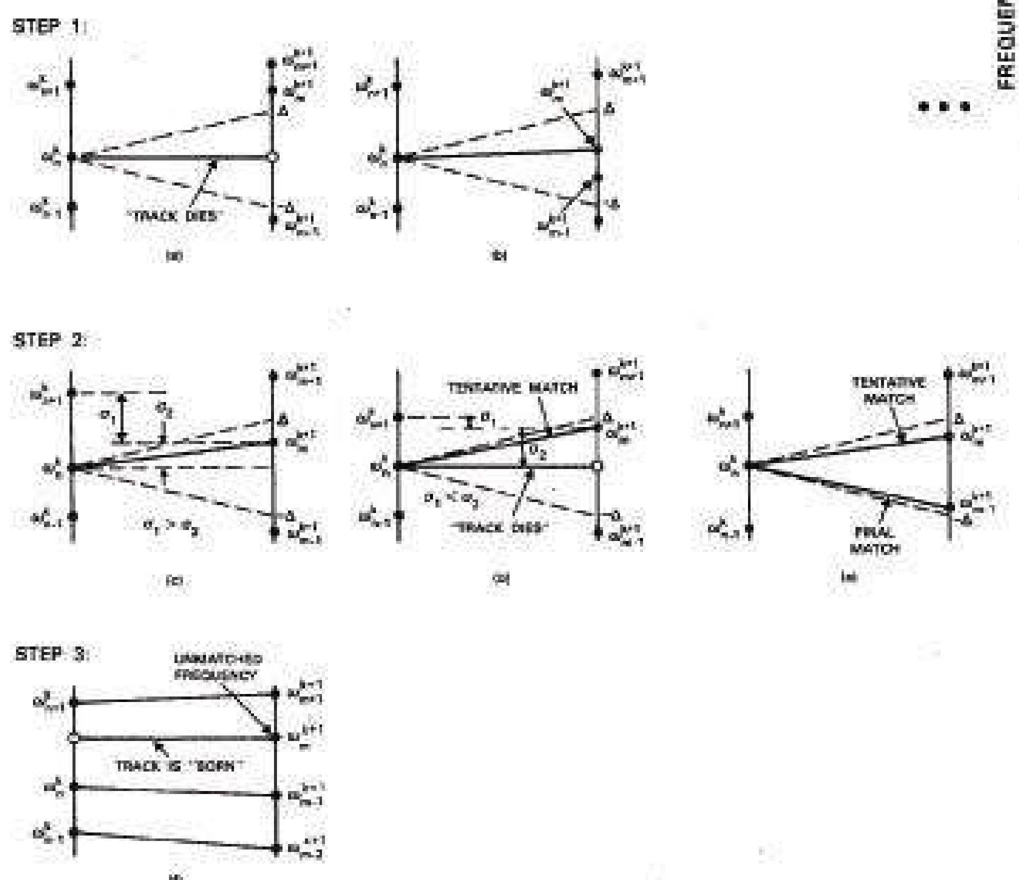
- 2nd step:
  - Ignore noise fluctuations. Threshold – above smoothed STFT.

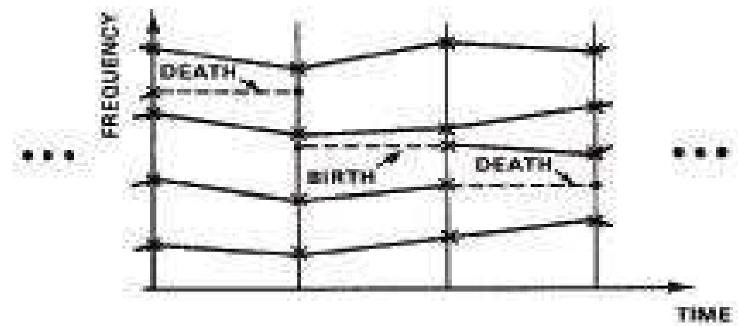


Peak tracking algorithm

3rd step:

- Build tracks: appending newly-found peaks to existing tracks, or creating new tracks.



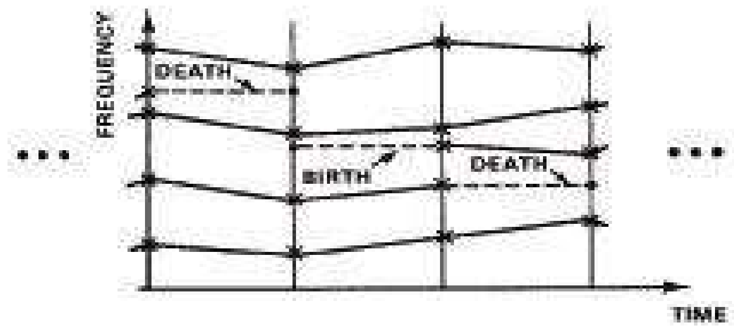


Peak tracking algorithm

If for each time frame we just build sinusoids with the frequencies and amplitudes of the peaks and then add all sinusoids together we will have discontinuities at frame boundaries.

So we have to smoothly interpolate the parameters from adjacent frames.

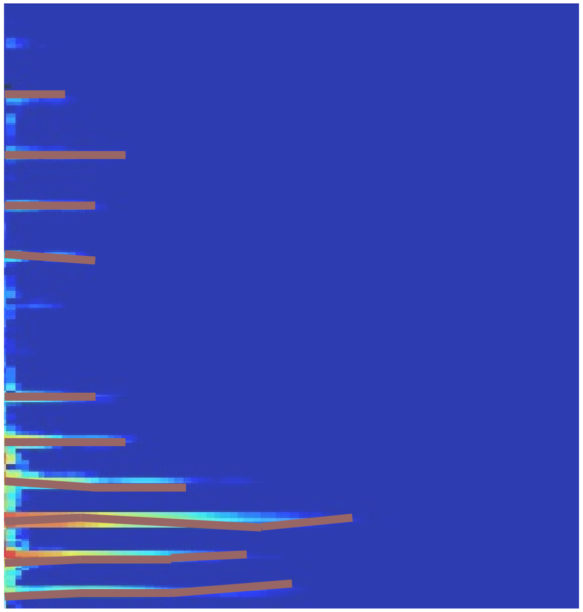
- 4th step:
  - Interpolation is done by track.



Sinusoidal modeling and synthesis

bank of oscillators + additive synthesis

- Idea: for each frequency component
  add a sinusoid of that frequency
- How? for each time frame:
  - build sinusoids with
    the interpolated parameter values and
  - add them together.



Spectral modeling synthesis

- Sinusoidal modeling limitations – inefficient for signals with broad spectrum, like noise and transients.

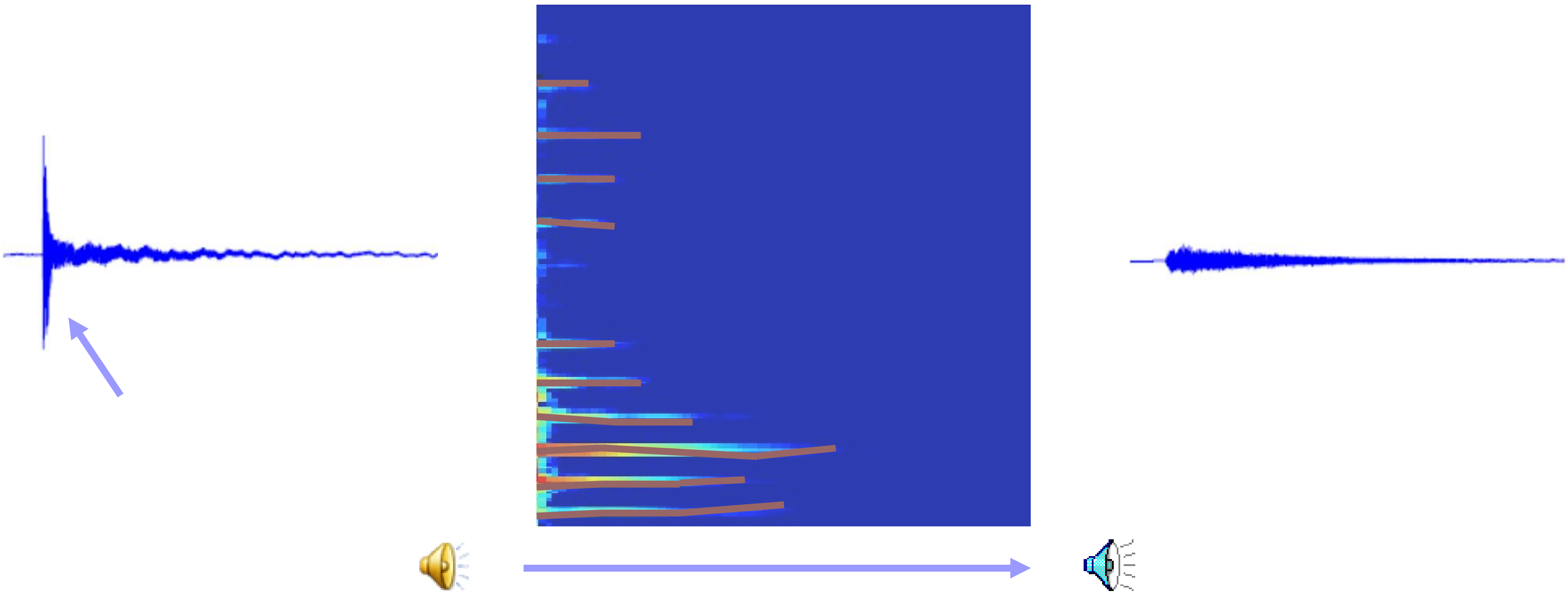
- Spectral modeling synthesis (SMS) (Serra):
  - Sinusoidal modeling + noise modeling

$$x(t) = \sum_{i=1}^{R} A_r(t)\cos\theta_r(t) + e(t)$$

- Analysis: Noise – time-varying frequency-shaping filter

- Synthesis: additive synthesis of bank of oscillators + time-varying filtered white noise

Importance of phase

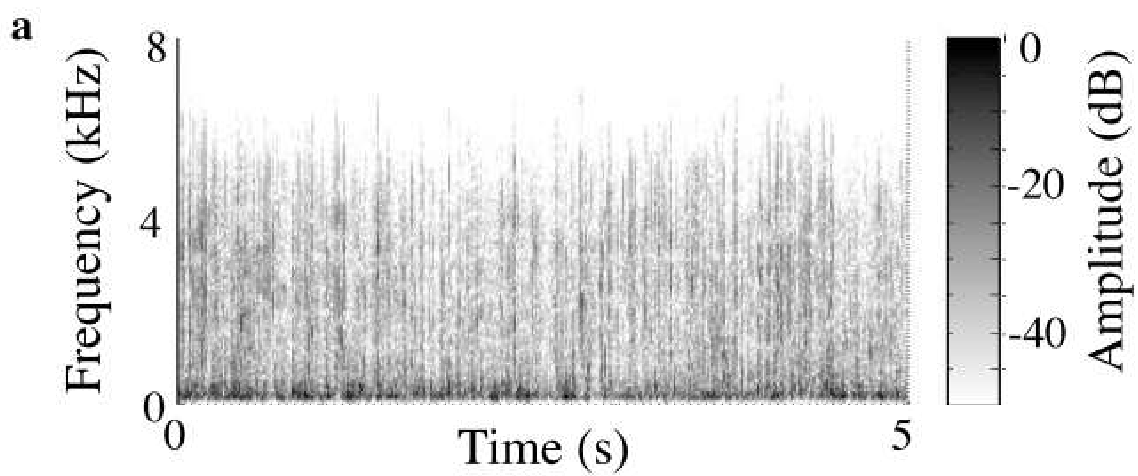


- Transients – perceptually important (for instance, in the recognition of musical instruments)
- Synthesis of transients requires correct phase alignment (SMS does not work in this case!)

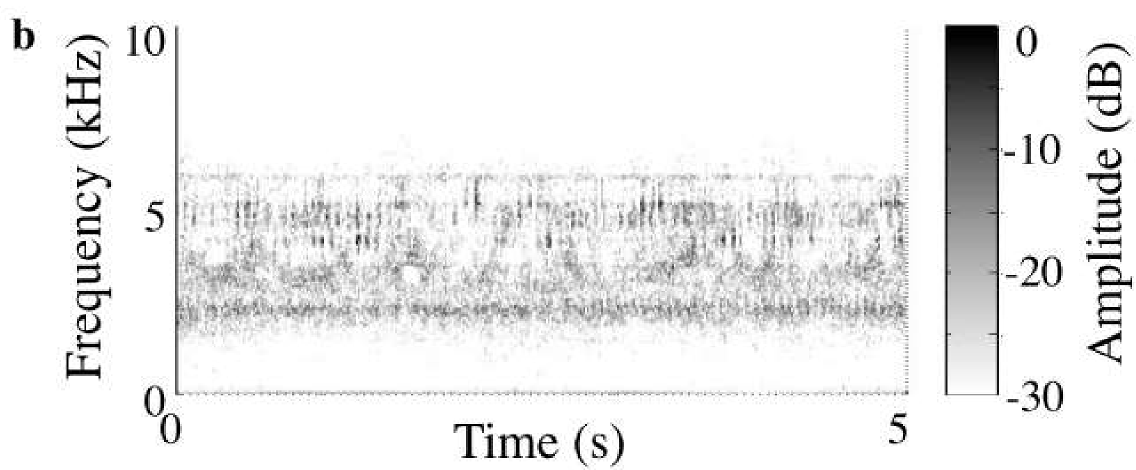
Sound texture synthesis with filter statistics

- Sound textures – rain, fire, insects, …
- Very high quality sounds can be obtained by using filters that reflect the statistics of real samples.
- Examples in: http://mcdermottlab.mit.edu/texture_examples/index.html

- Fire



- Swamp insects



3rd assignment

Assignment:

- Add sound to your game.
- Next week: demonstrations during class.
- Report: 1 or 2 pages (max 2 pág.) describing the sounds used and the sounds' effects/treatment. Until 12/june. In paper or by email (scavaco@fct – subject "[JS] ...")

Useful:

- Hello Audio Tutorial – https://jmonkeyengine.github.io/wiki/jme3/beginner/hello_audio.html
- Freesound.org
- OpenAL – open audio library. Includes áudio 3D with HRTFs
- JOAL – wrapper library to use OpenAl in Java.