

Università di Milano




3D Game Audio (notes)

Marco Tarini



1

Course Plan



- lec. 1: **Introduction** ●
- lec. 2: **Mathematics** for 3D Games ●●●●●●
- lec. 3: **Scene Graph** ●
- lec. 4: **Game 3D Physics** ●●●●+●●
- lec. 5: **Game Particle Systems** ●
- lec. 6: **Game 3D Models** ●●
- lec. 7: **Game Textures** ●●
- lec. 9: **Game Materials** ●
- lec. 8: **Game 3D Animations** ●●●●
- lec. 10: **Networking** for 3D Games ●
- lec. 11: **Artificial Intelligence** for 3D Games ●
- lec. 12: **3D Audio** for 3D Games ● (highlighted with a red arrow)
- lec. 13: **Rendering Techniques** for 3D Games ●

For a much more in-depth discussion of many of the subjects of this lecture, see the courses «[Sound in interaction](#)» and «[Elaborazione dei Segnali](#)»

2

Audio in 3D games: what we will (briefly!) cover



- Assets for audio
 - Data structures
 - For sound effects, ambient sounds, music, voiceovers
 - Notes on authoring / obtaining them (who, how...)
- Sound engine
 - Integration
 - Relationship with other aspect (animations)
- Sound rendering
 - Basic ops (2D or 3D)
 - 3D spatialized sound (emitter/receivers in the scene graph)
 - Interaction with the rest of the scene

3

Audio in games: game-design point of view



- **Sound effects**
 - authored by: **Sound Designers / Foley**
 - *informative function*
- **Ambient sounds**
 - authored by: **Sound Designers / Foley**
 - *immersive function*
- **Voiceovers**
 - authored by: **Dialog writers + Voice actors**
 - *narrative (=story-telling) function*
- **Music / (Under-)Score**
 - authored by: **Composers**
 - *emotional function*

e.g.:

dialogs (linear / non-linear)
commentary (non-linear)
narration (linear)

"Sound makes it **real**
Music makes you **feel**"

6

Audio in games: game-design point of view



Sound effects are super **informative**

- effective way to clarify things to the player.
- examples:
 - out of ammo:
 - gun just doesn't shoot → wrong key? a bug?
 - gun goes "click" → player gets it
 - doors closes *behind* player in 1st person view
 - sound door-slam effect: let him know!
- can substitute / abstract animation. Examples:
 - character collects object
 - object just disappears from scene → cheesy
 - pick-up animation? → hard to do right, delay affects gameplay
 - add pick-up sound instead (abstract) → acceptable
 - character changes outfit (RPG)
 - just swap character models → cheesy
 - add cloth undressing/dressing sound (abstract) → acceptable

7

Audio in games: dev-team roles



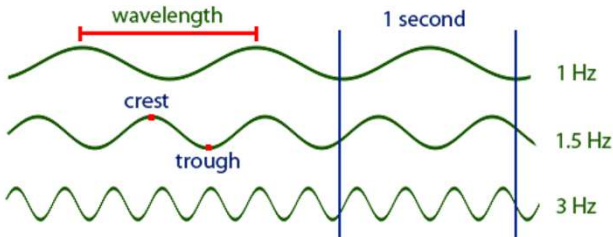
- Composer
- Sound Designer
- Foley
- Sound Integrator
- Audio Programmer
- Tool programmer
(for audio related tasks)



8

Sound wave

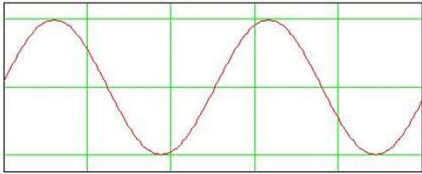
- Air pressure as a function of time
- **frequency** : (measured in 1/sec = Hz)



9

Sound wave

- Air pressure as a function of time
- Waves:
 - **frequency** (→ “pitch”, audible = from ~20 Hz to ~20 kHz)
 - **amplitude** (→ “volume”, level, loudness)



- Perception
 - as with most senses, sensorial response is roughly **logarithmic** with physical quantity (e.g.: **decibel** for **amplitudes**, **notes** for **frequencies**)

10

Sound wave & perception 101

What it is:
physical property of the sound wave

What it is perceived like:
by the human hearing system

<p>Amplitude ½ trough-to-crest distance on Y</p>	<p>Level or loudness (colloquially, Volume) how loud the sound is</p>
<p>Frequency = 1/wavelength crest-to-crest distance on X</p>	<p>Pitch how high-pitched or low-pitched the sound is [<i>Ita: acuto o grave</i>]</p>

← logarithmic →

← exponential →

<p>Spectrum (which frequencies are present)</p>	<p>Timber, tone</p>
--	----------------------------

11

Physical phenomenon: wave sums Perception of timber, tone

The graph illustrates the physical phenomenon of wave sums. It shows three vertically stacked plots sharing a common horizontal time axis t .

- The top plot shows a blue wave with amplitude A_1 and period $1/f_1$.
- The middle plot shows a green wave with amplitude A_2 and period $1/f_2$.
- The bottom plot shows the sum signal $x(t) = s_1 + s_2$ in red, which is the result of the two waves being added together.

12

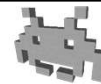
Sound as assets: compression



- **PCM** (pulse-code modulation)
 - uncompressed: just sampled and quantized
- **ADPCM** («Adaptive», «Differential» PCM)
 - one way to compress PCM
 - stores 4-bit *prediction errors* (in place of 16-bit values)
 - fixed-compression rate: 4:1
 - fast (on-the-fly, HW supported) decompression
 - not very good compression / quality rate
- **MP3**
 - works great
 - one example of perceptual encoding
 - HW supported, but needs de-compression *before* it is played

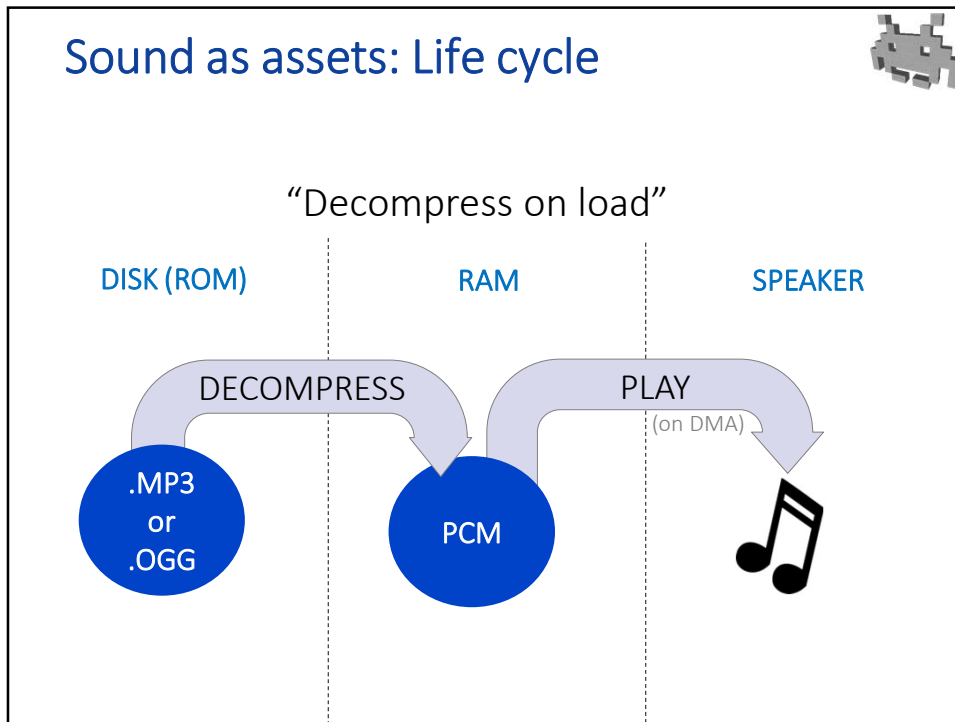
15

Assets for sounds: most common file formats

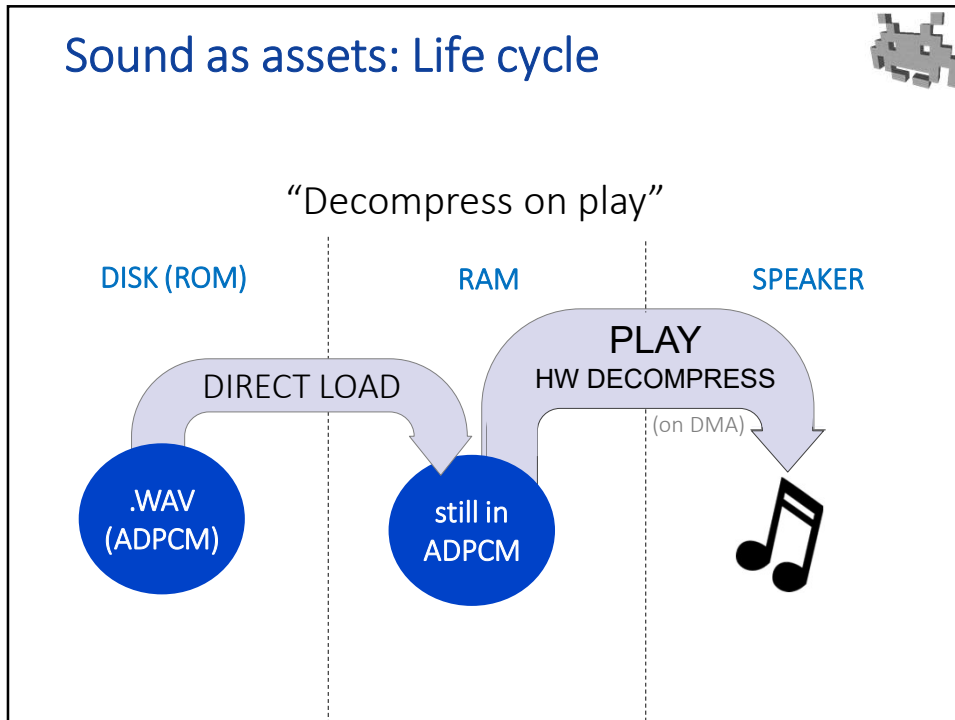


- **.mp3**
 - perceptual encoding
 - good balance between compression-ratio / quality
 - common for final releases / distributions
- **.ogg** (vorbis)
 - optimized for music
 - usually best quality for compressed
- **.wav**
 - uncompressed (PCM)
 - not much used as assets (e.g. unity will compress them)
 - or, compressed (ADPCM)
 - common in production

16



17



18

Sound as assets: Life cycle



- **Static Load** «load first, then play as needed»
 - the good: immediate play
 - the bad: costs RAM (good for small / few sound fxs)
- variant: **decompress on Load**
 - more processing load
- variant: **decompress on Play**
 - requires HW support
 - less RAM, more audio-latency
 - only ADPCM compression (poor ratios or poor quality)
- **Dynamic Load** «when you need: load, then play»
 - the good: saves RAM
 - the bad: audio-latency (audio-lag)
- variant: **streaming** «when you need, play as you load»
 - using audio buffer (small dedicated memory, FIFO)
 - good solution for long files (e.g., musical scores)

19

compare: ADPCM – audio compression, with: DXT (aka S3TC) – texture compression



- unlike more sophisticated compression schemes (e.g., MP3, JPEG respectively), they are designed for **fast, on-the-fly decompression**
 - so, data can be kept compressed in RAM
 - decompress on *USE*
 - hardware decompress → hardwired decompress algorithm
- the same price is paid:
 - poor compression rates
 - *fixed* compression rates – no adaptivity
 - compressed size does not depend on content
 - lossy – and very much so
 - poorer quality compared to alternatives
- similar considerations / choices apply, for example:
 - way 1: employ that compression on disk → fast/direct asset loading
 - way 2: employ a better compression scheme on disk → cheaper on storage / bandwidth, but requires decompression **and recompression** on loading

20

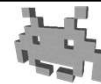
Latency in audio: perceptually crucial



- Latency is crucial in audio synchronization
 - Multimodal: audio VS not audio
e.g., VS video, tactile (keystroke) VS audio
 - Monomodal: audio VS audio
e.g., sound effect 1 VS sound effect 2
- max tolerated latency for video (e.g., "60ms is too much")
>>
max tolerated latency for audio (e.g., "5ms is too much")
- Known (empirically) to degrade experience *a lot*
 - True for games, VR, movies...

21

Specialized assets for music




- Store a digital score instead?



the digital equivalent of this ↑ :
an asset describing which notes
are to be sung during which interval,
with which instrument,
effect (*crescendo*, *staccato*) etc.

22

Specialized assets for music



- Store some sort of digital *score* instead?
- The *traditional* music asset in games
 - any classic game tune you can remember was originally stored in this way
 - Think tunes of Pacman, Super Mario Bros, Tetris,
 - the only way – until the '90
- Standard format: **MIDI**
- Pros:
 - much **cheaper** to store
 - perfect for **procedural** music
 - e.g., non linear soundtrack
- Cons:
 - requires instrument library (samples) at runtime
 - limits expressiveness
 - e.g., voice, choir, subtleties
 - limits authoring procedures
 - requires processing in real time


what used to make this a strict necessity

may make this still attractive today (a bit)

what made this almost abandoned today

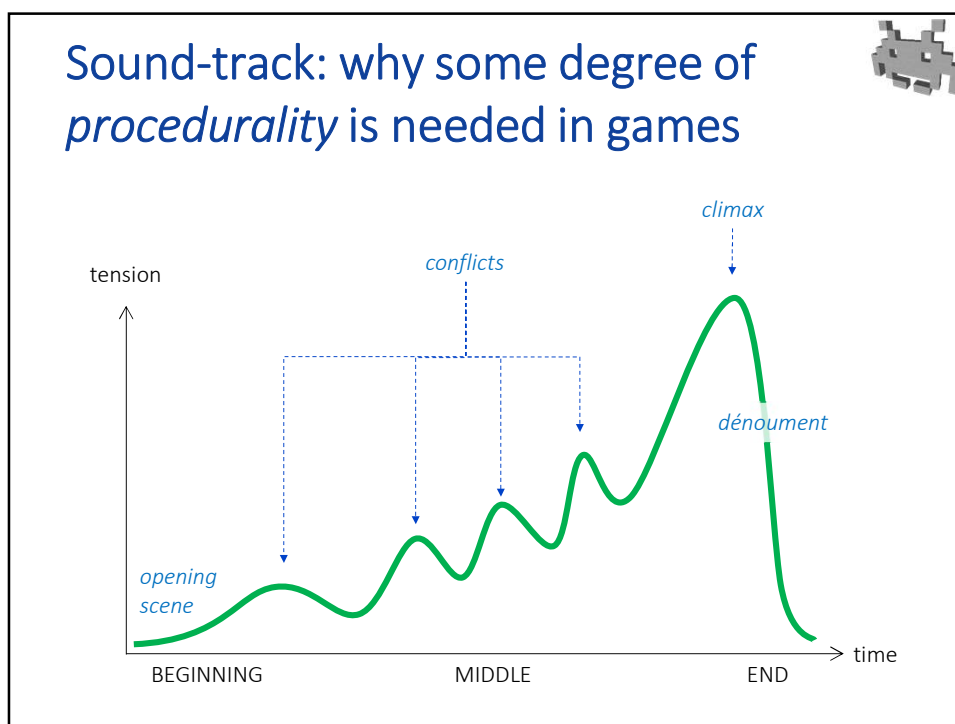
23

Assets for music today



- Music as just another **sampled sound wave**
(as any other audio)
 - maybe looped
- Typically made of «stem» (sub-tracks)
 - «bass» stem
 - «guitar» stem
 - «choir» stem ...
- Option 1: pre-mix all stems, bake the result
- Option 2: keep stems separated, mix in realtime
 - more resource consuming (computation/RAM)
 - useful for dynamic re-tuning and **non-linear** music
 - allows for some form of procedurality

24



25

Specialized assets for Ambient Sounds

- Ambience track (“soundscape”, “background”)
 - the old-school way: just a sound asset (not specialized)
 - looped and long (e.g., ~10 min)
 - typically, low-pitch
 - problems: heavy (long!), repetition artifacts
- Better way: procedural blend of individual FXs
 - according to customizable randomized rules
 - e.g., randomized repetitions, at randomized times
- Authoring: specialized game tools
 - e.g., see <http://rpg.ambient-mixer.com/>
- Still no standardized asset format for this :-)

26

Specialized assets for Ambient Sounds



Example:

- Instead of a Drone loop for:
 - a street traffic scene
 - a jungle
 - a computer room
- Use a random blend of:
 - car horns, engines
 - animal noises
 - individual beeps

27

Middleware for sounds in games



Libs / API for sounds in games



28

Authoring sound effects (task of the Sound Designer)



- Remember: as any asset, you can buy / get them from **Libraries / Repositories**
 - a common solution in practice
- **Capture**
 - Digital artist: “Foley”
 - Field capture (for ambient sounds)
- **Synthetize**
 - by sound editing



30

Voice Overs (brief notes)

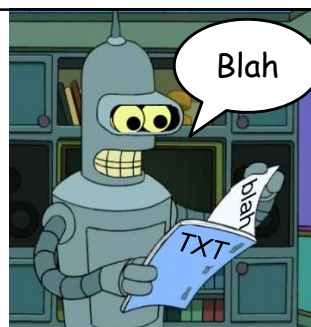


- Two kinds:
 - Linear
 - e.g., cutscenes dialogs, narrations
 - Non-linear (e.g., driven by a state machine – see AI lecture)
 - e.g., dialogs trees
 - e.g., running commentary (of a football match)
- Technically, it’s nothing special: just a sound fx.
- But, several practical challenges:
 - *Lots* of assets! (also implying file names, folders nightmare)
 - Localization often needed
 - Expensive production (\$\$\$), late in the development
 - During early stages: better to use placeholders

31

Speech Synthesis (or “text to speech”)

- A.I. frontier
- currently: still not good enough
 - not *believable* enough
 - human voice = we are all expert = difficult to trick us
 - we are not even in the audio “uncanny valley” yet?
 - not *expressive* enough (emotions, characterizations)
 - i.e., virtual voice actors are not ... good voice actors
- just a matter of time?
- when it will be here, it will
 - free games from most issues of **voice-over assets**
 - get us all the usual advantages of **procedurality**



33

A summary of ways to author sound assets

- Synthesized / simulated / procedural fxs :
 - baked
 - (not that common)
 - Captured fxs :
 - hardware: a good microphone (and ADC)!
 - by: “Foley artists”
 - very often: just bought / downloaded from repositories
 - Voice :
 - hardware: a good microphone (and ADC)!
 - by voice actors
 - sometimes, during motion capture sections
 - speech synthesis? (won't be used for some time yet)
 - Composed (for music) :
 - musicians: frequent 3rd members of 3-man dev teams
 - recent improvements of tools (both HW and SW)
 - e.g. chorus with arbitrary lyrics now attainable
 - a few game composer gained substantial fame!
- } then,
sound
editing



34

Blend-shapes for «lip-sync»

sound wave of a voice-over

Phoneme Recognition

weights for Facial-Morphs

Aa Oo, U, W Oh Ee C, D, G

M, B, P L V, F Ss Sh, Ch

*Game: Blue Toad Murder Files (Relentless Software, 2010)
Artist: Kelly Ford, <https://kelly4d.artstation.com/>*

35

Research topic: from voiceovers to NPC animations

- With Machine Learning (data driven)

sound wave of a voice-over

ML

skeletal animation for a virtual character believably gesticulating while speaking

“Style-Controllable Speech-Driven Gesture Synthesis Using Normalising Flows”
Simon Alexanderson et al, CGF (Eurographics 2020)

36

What triggers sound fxs in a typical game-engine?



- fxs explicitly started from **scripts**
 - e.g. at **collision response**
 - e.g. accompanying all sorts of **game logic**
 - anything from “doors opening” to “level completed”
- fxs associated to scene **Objects**
 - constantly looped fx from a source, e.g. a radio
- fxs associated to **interface elements**
- fxs as **Actions** of the **AI** (see AI lecture)
 - see: **AI** for **NPCs**
- fxs associated to **Animations** (see animation lecture)
 - e.g. *footsteps* fxs during walk
 - e.g. *detach from ground / Land* fxs during jumps
 - e.g. *air-swishes* during sword swings
 - convenient to ease action/sound synchronization

37

Sound Rendering: *basic playback tasks*

in any game,
even in a 2D setting

Main Asset:



the sound **buffer**

the digitalized sound wave,
ready to be sent
to the speaker



- **Mixing**
 - **Linear combinations** of waves
 - E.g.: cross-fade 2 sound, maybe with **transition functions** etc.
- **Tweak / Tune:** (useful to randomize sounds – e.g., footsteps!)
 - **Level** (~“loudness”) – **amplitude scaling**
 - both **pitch** and **speed** – **time scaling**
 - *only pitch*, or *only speed* (a bit less trivial)
- **Sound filtering**
 - **convolutions** of sound buffer with (small) kernels
 - useful to add **procedural** effects such as **low-pass / attenuation** ...
- **Prioritization**
 - why: limited «polyphony» -
the engine can mix only up to n sounds (e.g., $n = 64$)
 - solution: game-dev assigns a priority to each sound fx


38

Sound Rendering in 3D games

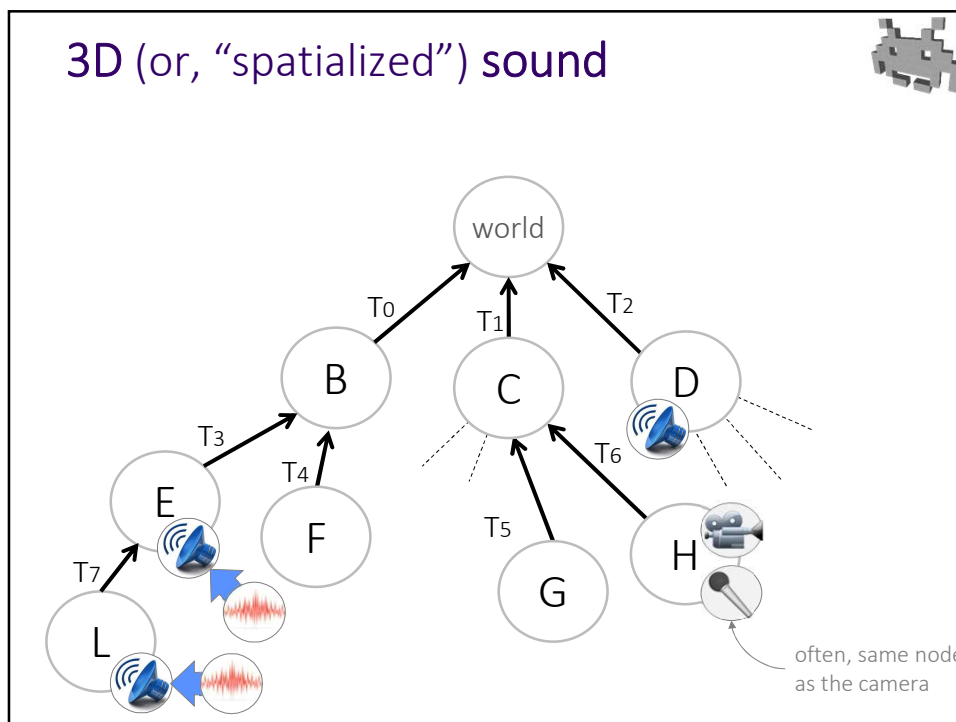
3D (or, "spatialized") sound

- sounds which are:
 - **emitted** from a virtual source (somewhere in 3D)
 - **received** from a virtual microphone (somewhere in 3D)
 - **propagated** across the 3D scene
- useful abstractions used in games:
 - the **listener**
 - the **source(s)** } sitting in nodes of the scene graph!

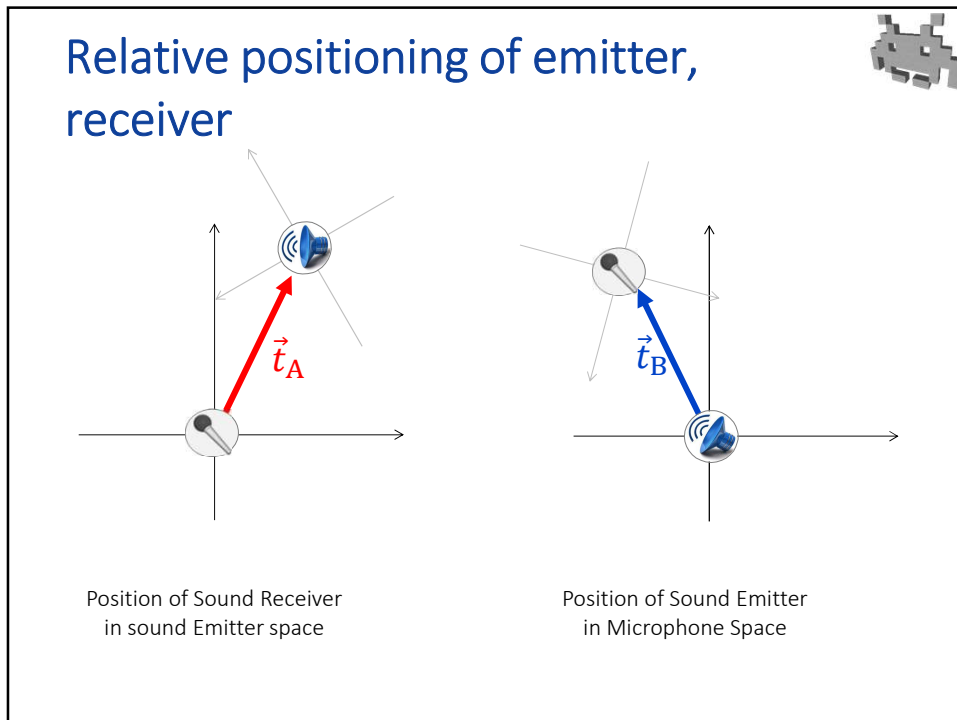
note: position and orientation



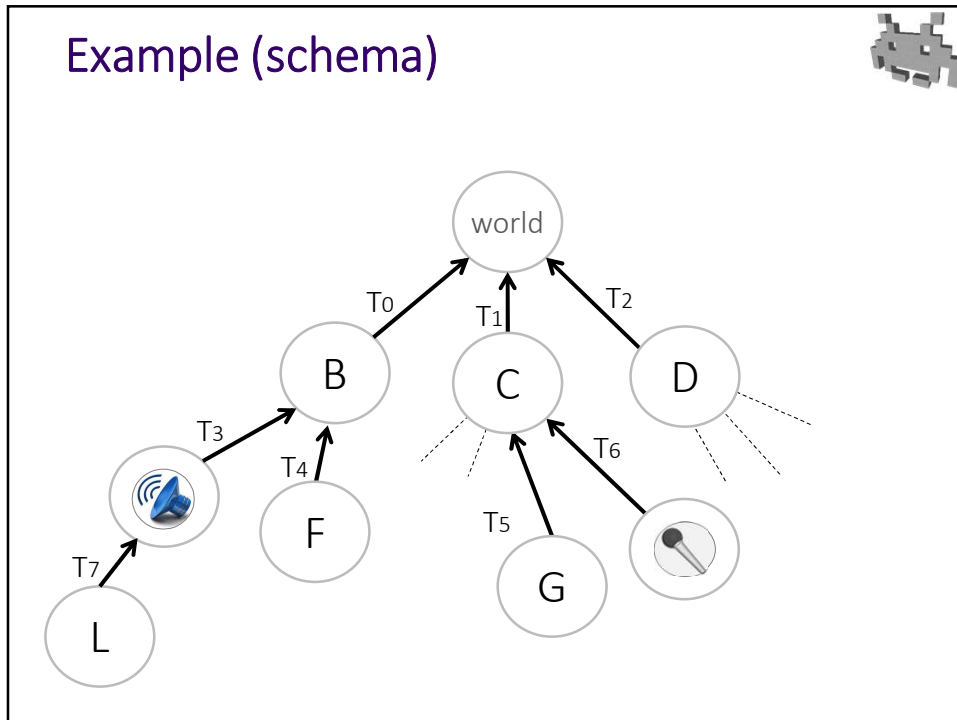
39



40





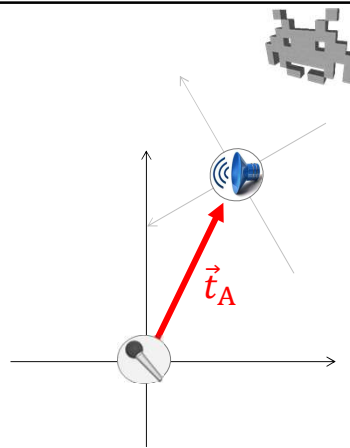
41



42



Example 1/2

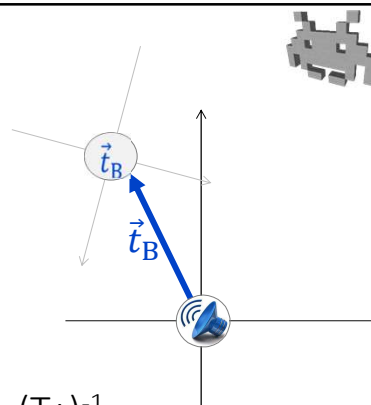
- What is the position of  in the space of  ?
- Answer:
the translation component \vec{t}_A of $T_A = (T_6)^{-1} \cdot (T_1)^{-1} \cdot T_0 \cdot T_3$
- Needed for determining:
 - Distance emitter-receiver $\|\vec{t}_A\|$ for the level falloff
 - Direction toward sound source $\hat{t}_A = \vec{t}_A / \|\vec{t}_A\|$ determining ILD, ITD and anisotropic spectral cues



43

Example 2/2

- What is the position of  in the space of  ?
- Answer:
the translation component \vec{t}_B of $T_B = (T_3)^{-1} \cdot (T_0)^{-1} \cdot T_1 \cdot T_6 = (T_A)^{-1}$
 - Note: it is not $-\vec{t}_A$
- Needed for determining:
 - Angular fall-off functions for anisotropic sound-emitters



44

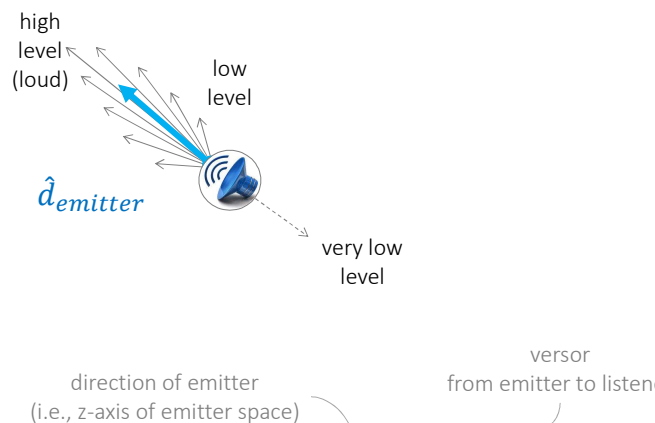
3D (or, “spatialized”) sound: for direct sound propagation



- consequent auto-tuning of
 - **level**: (linked to perceived “loudness”) according to source-listener **distance**
 - with a given (dev-controlled) «roll-off» or «fall-off» function
 - E.g. $1/d$ or $1/d^2$
 - **pitch**: (Doppler effect) according to **relative speed** or source w.r.t. listener
 - **interaural time difference (ITD)** and **interaural level difference (ILD)**: difference of sound arrival time between the two ears. Used by brain for **sound localization**. Gives illusion of sound **relative location** w.r.t. head using stereo speakers. It’s SMALL! e.g. $\sim 10 \mu s$

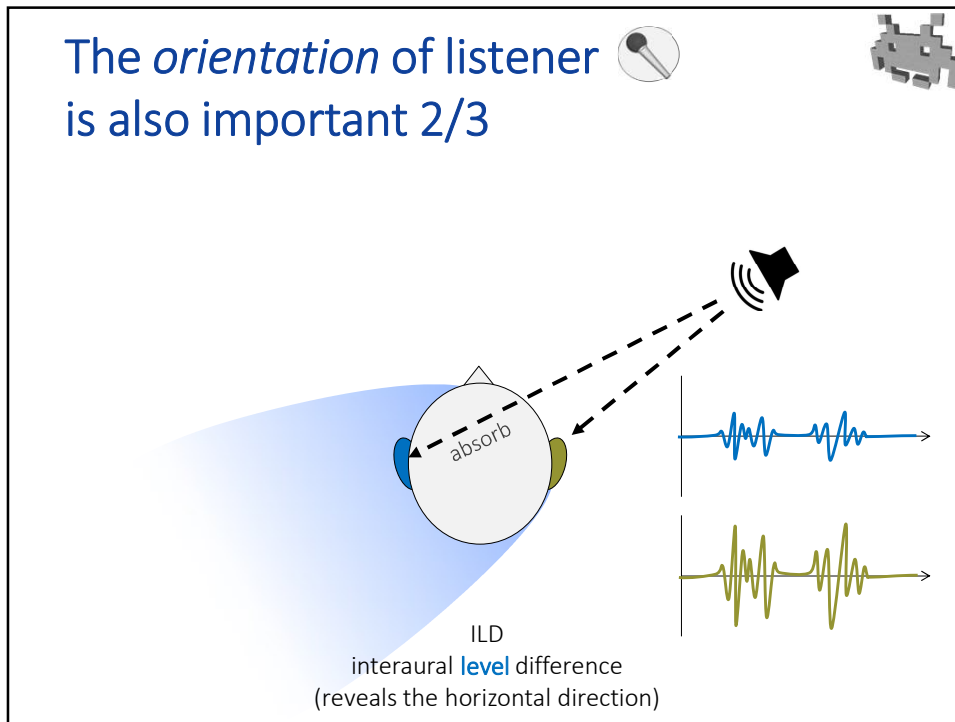
45

Directional (anisotropic) sound emitters

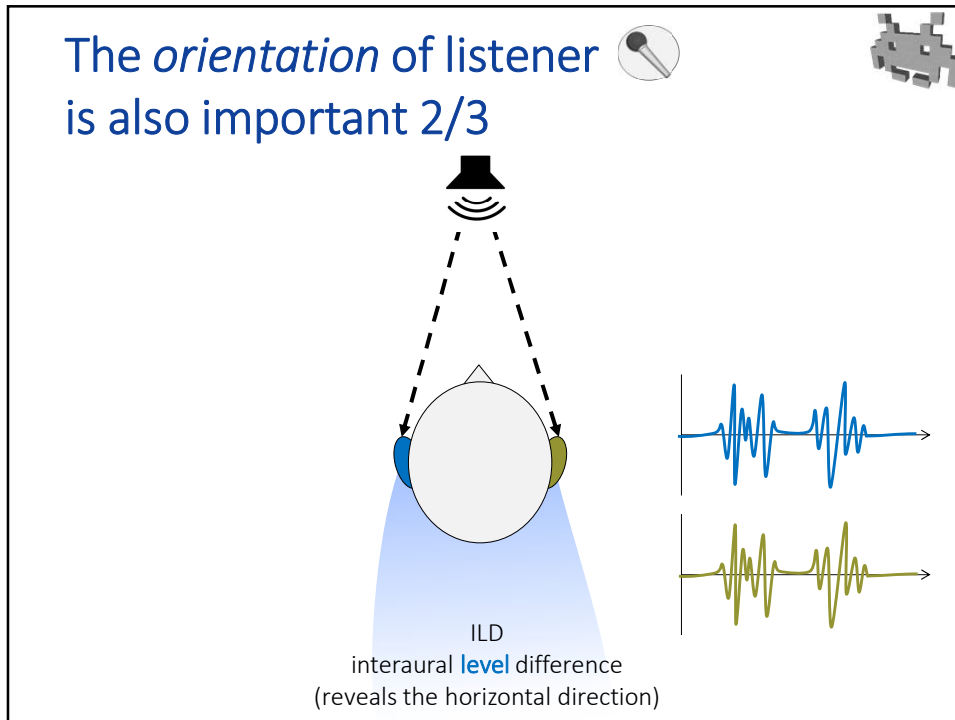


Level multiplier = angular fall-off of $(\hat{d}_{emitter} \cdot \hat{d})$

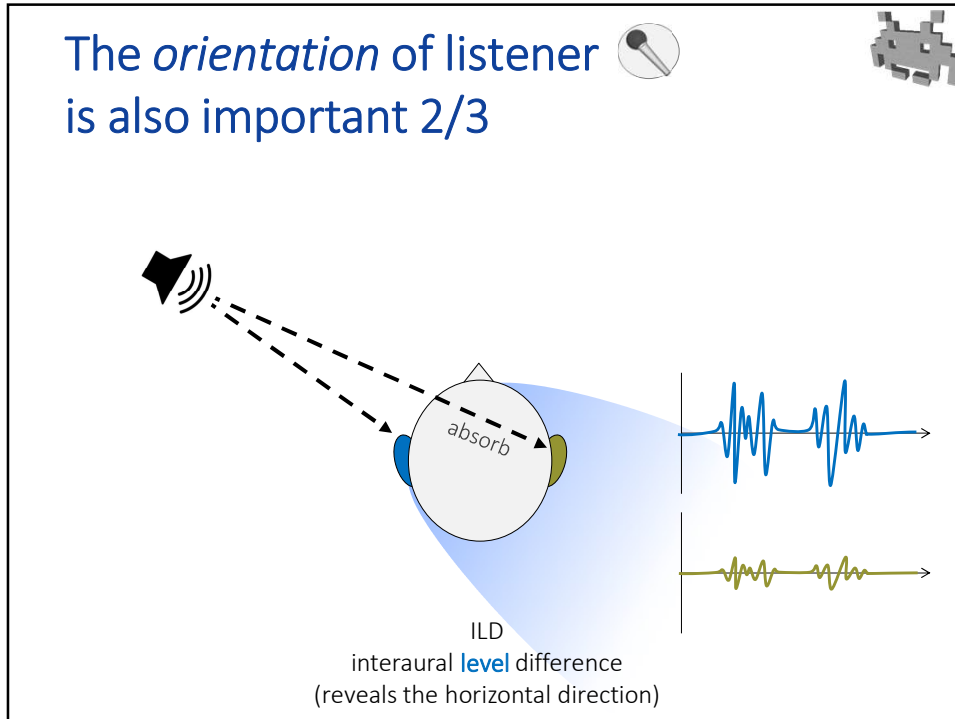
46



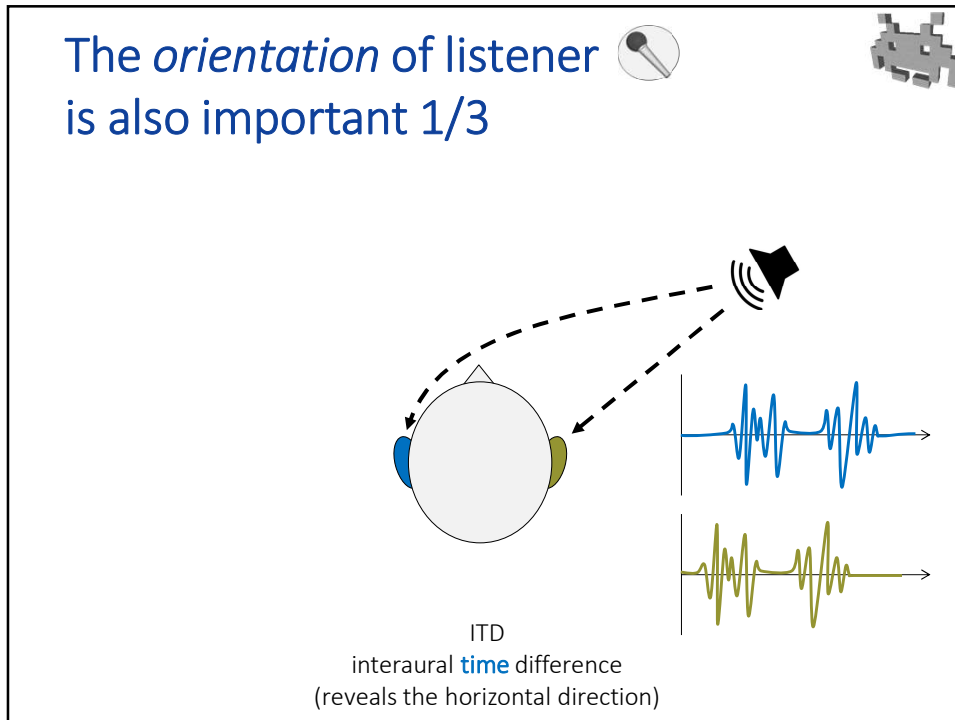
48



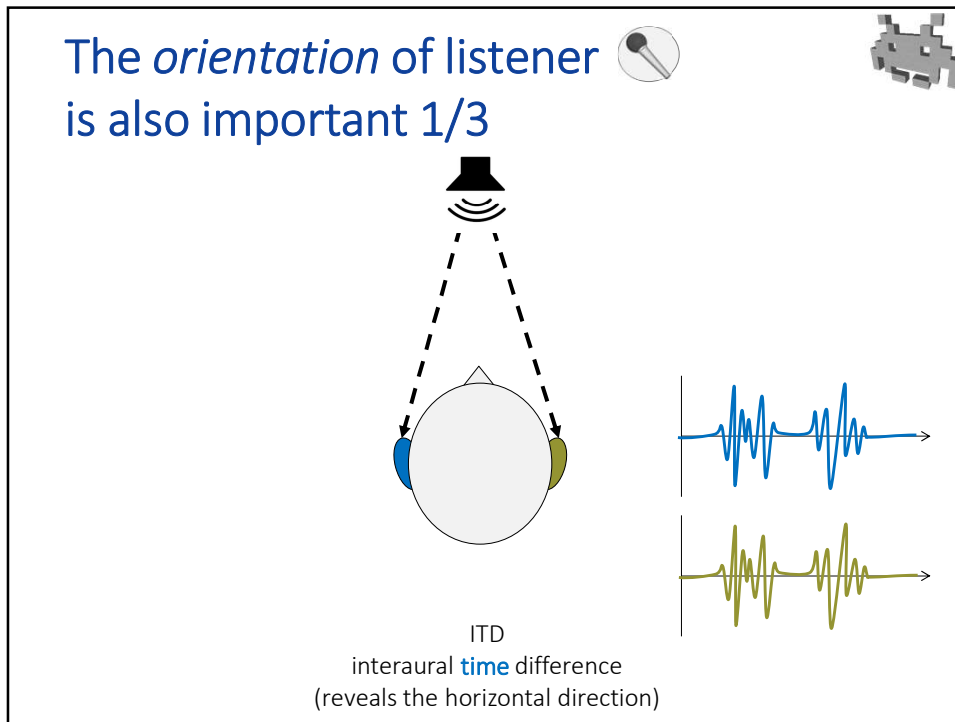
49



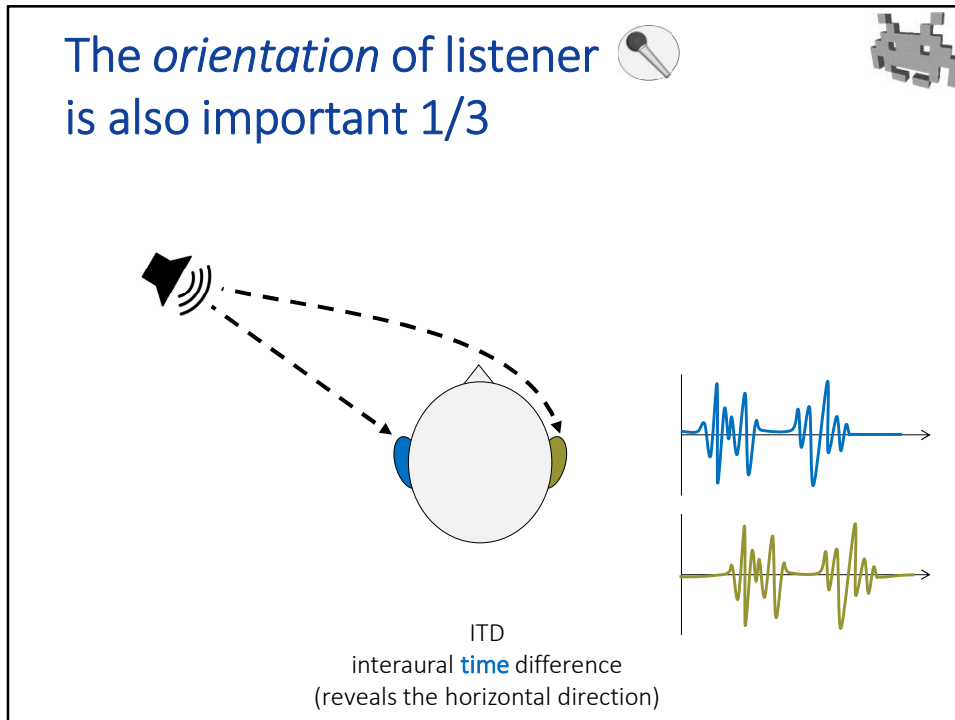
50



51

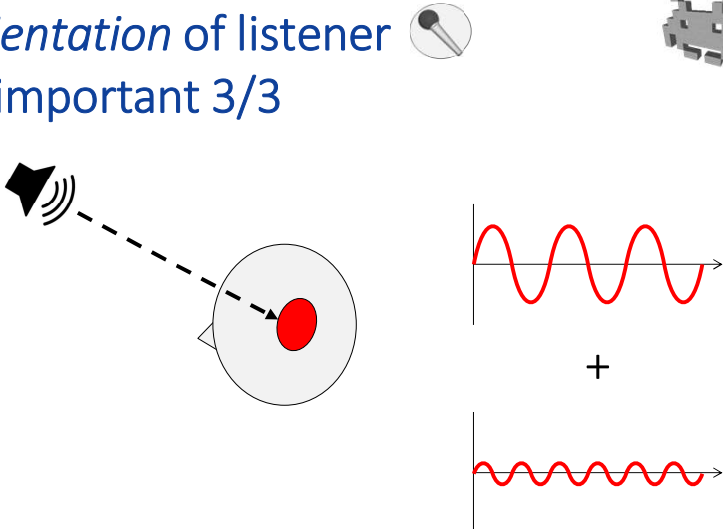


52



53

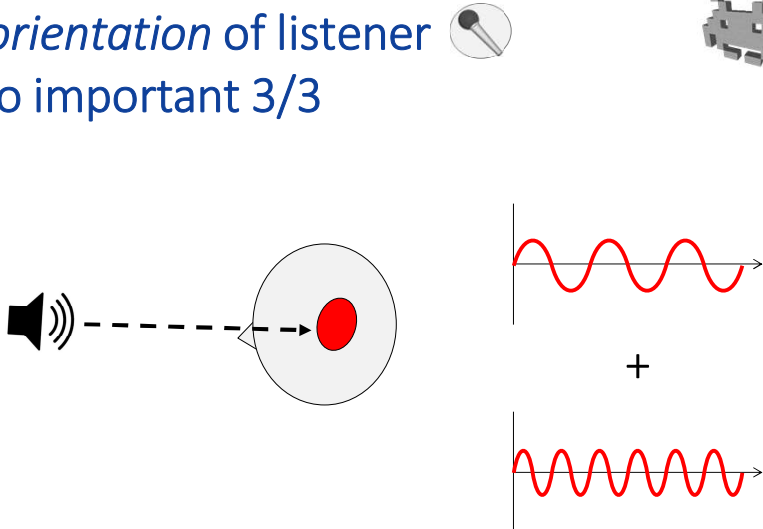
The *orientation* of listener is also important 3/3



anisotropic spectral cues
(reveals mostly the vertical direction, and disambiguates forward-backward)

54

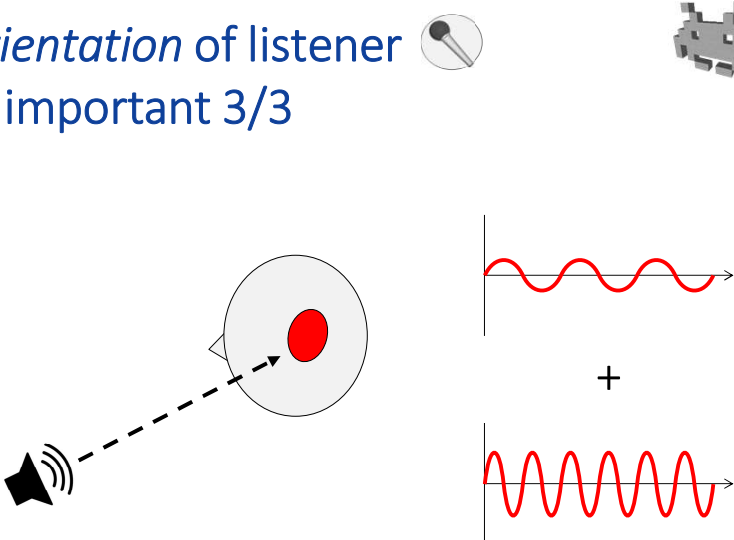
The *orientation* of listener is also important 3/3



anisotropic spectral cues
(reveals mostly the vertical direction, and disambiguates forward-backward)

55

The *orientation* of listener is also important 3/3



anisotropic spectral cues
(reveals mostly the vertical direction, and disambiguates forward-backward)

56

Anisotropic spectral cues for personalized ear shapes (advanced task!)

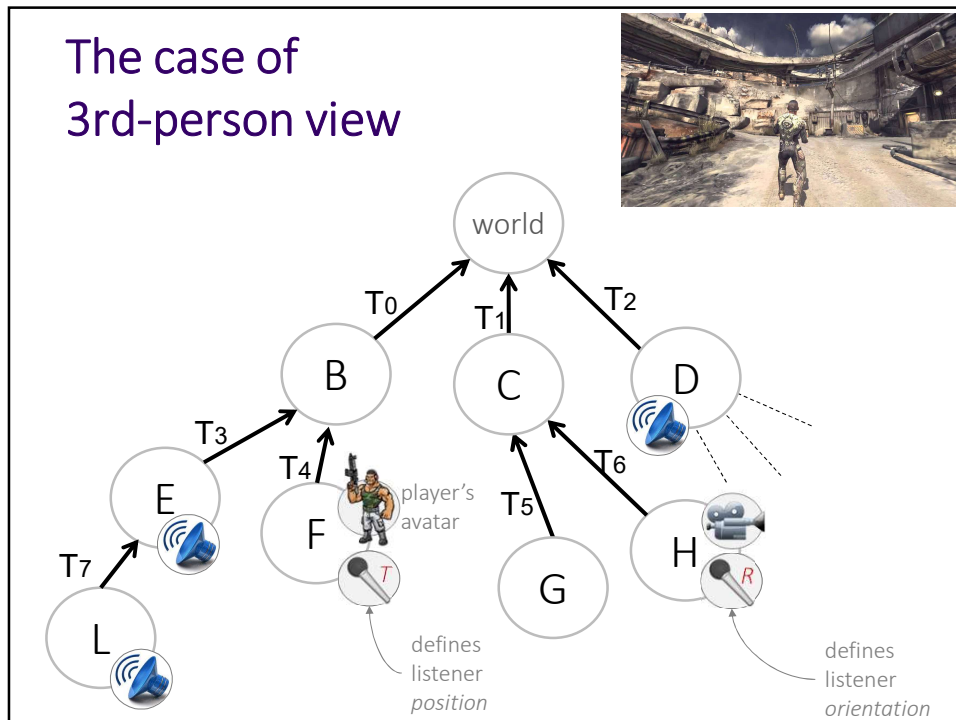
- Spectral clues: an “anisotropic” stereo sound filter which depends on sound incoming **direction**
 - in listener reference frame (listener orientation counts!)
- Requires a 3D model of the hear of the listener.



- More commonly, approximations are used

“Reconstructing head models from photographs for individualized 3D-audio processing”
M Dellepiane et al, CGF 27 (7) - (Pacific Graphics)

57



59

Sound Rendering: sound propagation in the 3D scene

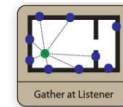
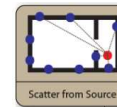
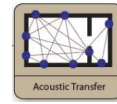
- So far, we considered the 3D effects of sound-waves propagated directly from emitter to microphone
- In reality, sound-waves interact with solids in the 3D scene
- Three basic phenomena:
 - **Absorption:** some* energy of the sound-wave is lost (dissipated into heat)
 - **Reflection:** some* part of the sound-wave bounces off (e.g.) walls
 - **Transmission:** some* part of the sound-wave passes through solid objects

* how much of it?
It depends on the materials, and the wave-length

60

Sound Rendering: sound propagation in the 3D scene

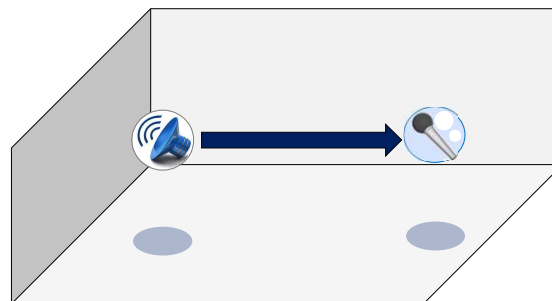
- Reuse **collision proxies**!
- Targets simulation of effects by:
 - Absorption (occlusion, obstruction)
 - Transmission (muffling)
 - Reflections (reverb, echoes)
- Active research topic
 - Currently: no standard solution adopted by 3D games
 - Often, tricks coded *ad-hoc* by the **sound programmer**



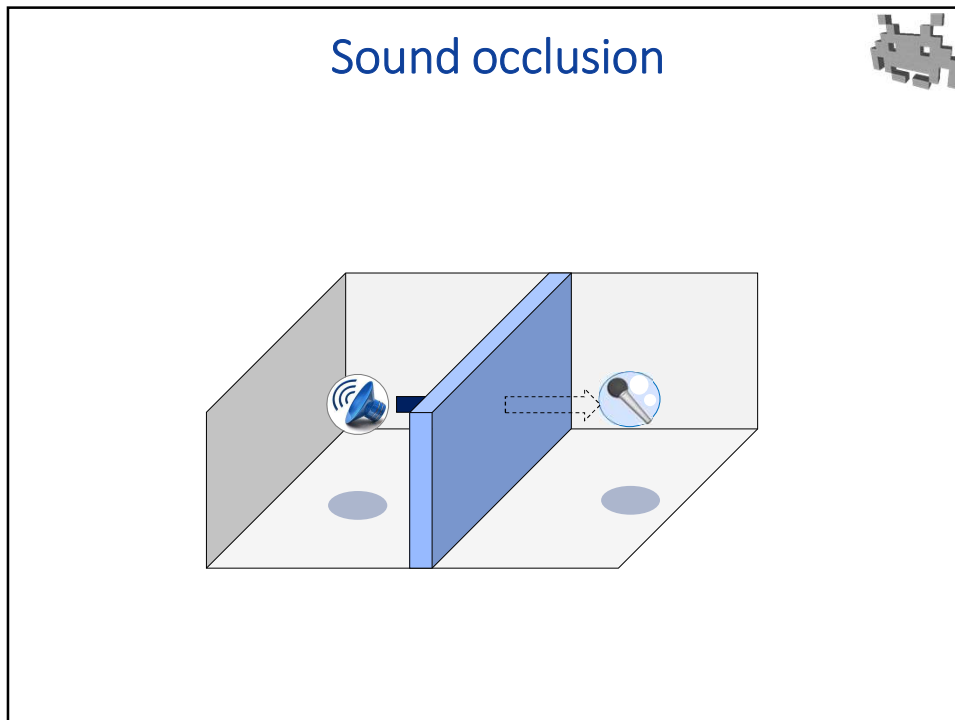
E.g. see: "Interactive Sound Propagation using Compact Acoustic Transfer Operators"
Lakulish Antani, Anish Chandak, Lauri Savioja, Dinesh Manocha
SIGGRAPH 2012

61

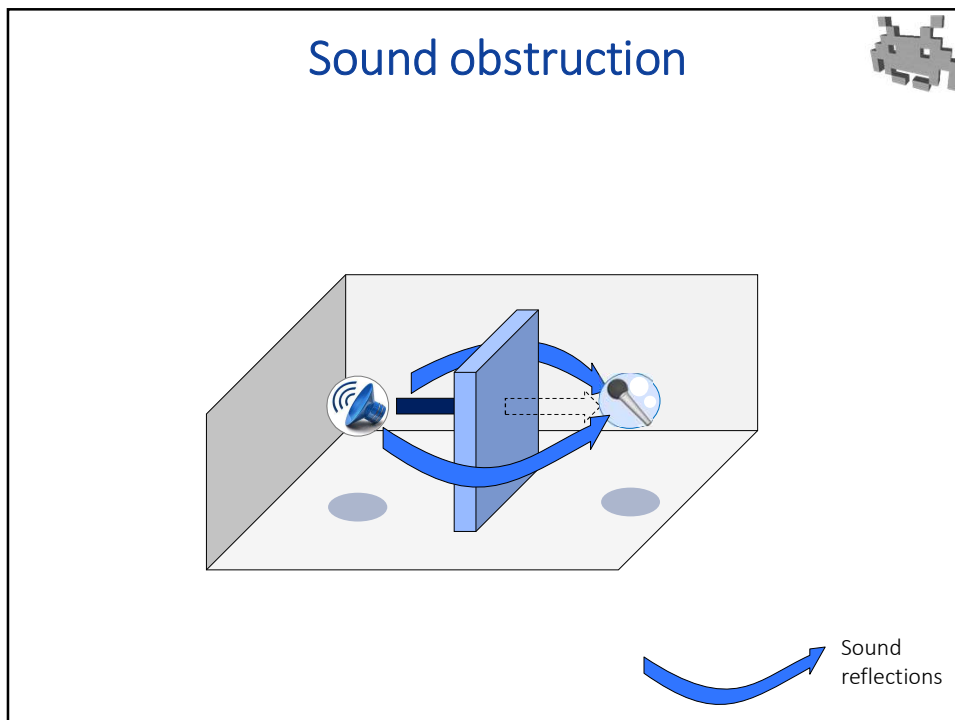
Direct sound propagation



62



63

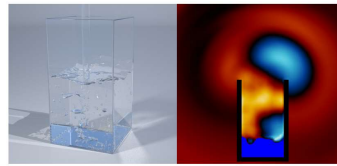
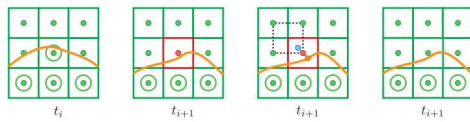


64

Sound Rendering: full computation of sound propagation in scene



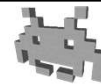
- e.g., for collisions
- using physical material specification
- not (yet?) used in games
 - but active research topic



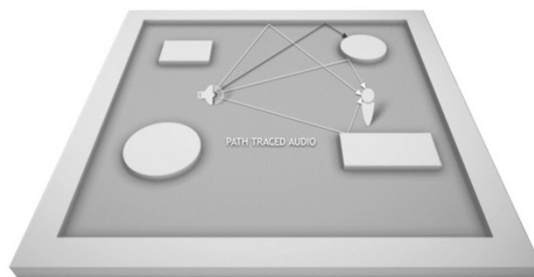
E.g. see: "Toward Wave-based Sound Synthesis for Computer Animation"
Jui-Hsien Wang, Ante Qu, Timothy R. Langlois, Doug L. James
SIGGRAPH 2018

65

Sound Reverb



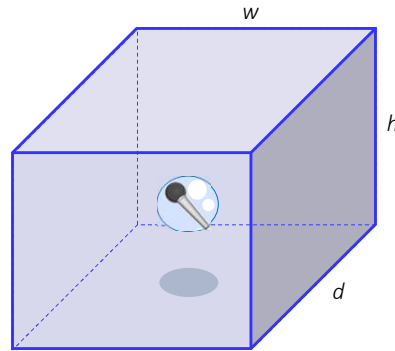
- Solution 1: path tracing (expensive!)



66

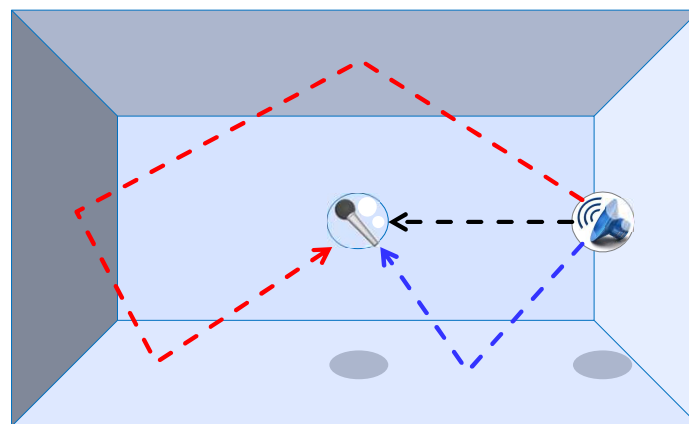
Sound Rerverb

- Solution 2: «shoe-box model»
 - An approximation that uses closed-form formulas



67

Shoe-box model for sound reverb



68