

1

2

Game Audio: intro



- A fundamental aspect of game-design
 - Impact cannot be overestimated
 - for immersion
 - for emotion
 - for gameplay
 - for story-telling
 - remember that we don't focus on game-design aspects in this course
- The main technical aspects of game sound are, however, quite subtle

3

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



- Assets for audio
 - Data structures (storage, incl. compression)
 - Solution for: sound fxs, ambient sounds, music, voiceovers
 - Notes on authoring / how to obtaining them (who, how...)
- Sound engine
 - Integration with the rest of the engine
 - Relationship with other aspect (animations)
- Sound rendering
 - Basic play-back operations (not 3D)
 - 3D spatialized sound (emitters/receivers in the scene graph)
 - Interaction of sound with the rest of the scene (geom proxies)

4

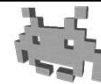
Audio in games: game-design point of view



- Sound effects
 - *informative function*
- Ambient sounds
 - *immersive function*
- Voiceovers
 - *narrative function*
- Music / (Under-)Score
 - *emotional function*

5

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



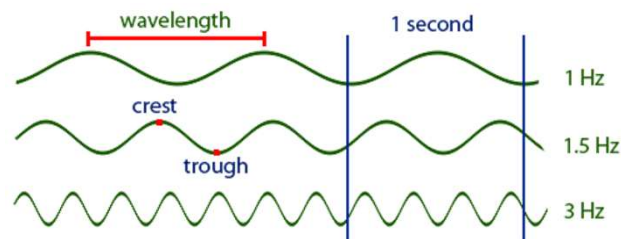
- Composer
- Sound Designer
- Foley
- Sound Integrator
- Audio Programmer
- Tool programmer
(for audio related tasks)



8

Sound wave

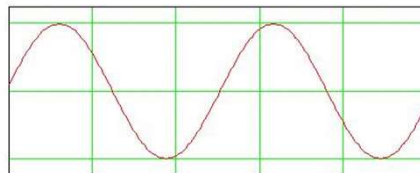
- Air pressure as a function of time
- **frequency** : (measured in 1/sec = Hz)



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**, **musical notes** for **frequencies**)

10

Sound wave & perception 101

What it is:
physical property of the sound wave

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

What it is perceived like:
by the human hearing system

← logarithmic
exponential →

Spectrum (which frequencies are present)	Timbre, tone
--	---------------------

11

Physical phenomenon: wave sums Perception of timber, tone

12

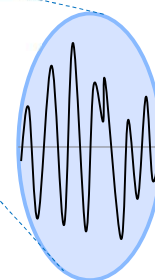
Sound wave as assets

- Air pressure as a function of time



- To digitalize it ("PCM"):

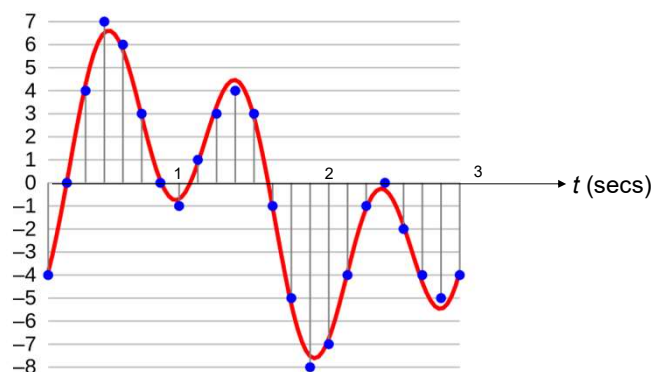
- **sample** it
 - at some fixed **rate**
 - typically, between 32-48 KHz (44.1 KHz standard for CD, 32 KHz low quality) (voice: 16 KHz is sufficient)
- **quantize** samples
 - at some fixed **precision**
 - typically, 14-24 bits per sample (in 8bit systems: guess)
- then maybe **compress** it



13

PCM – Pulse Code Modulation

- Toy example: 8 Hz sampling, 4 bit quantization:



14

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

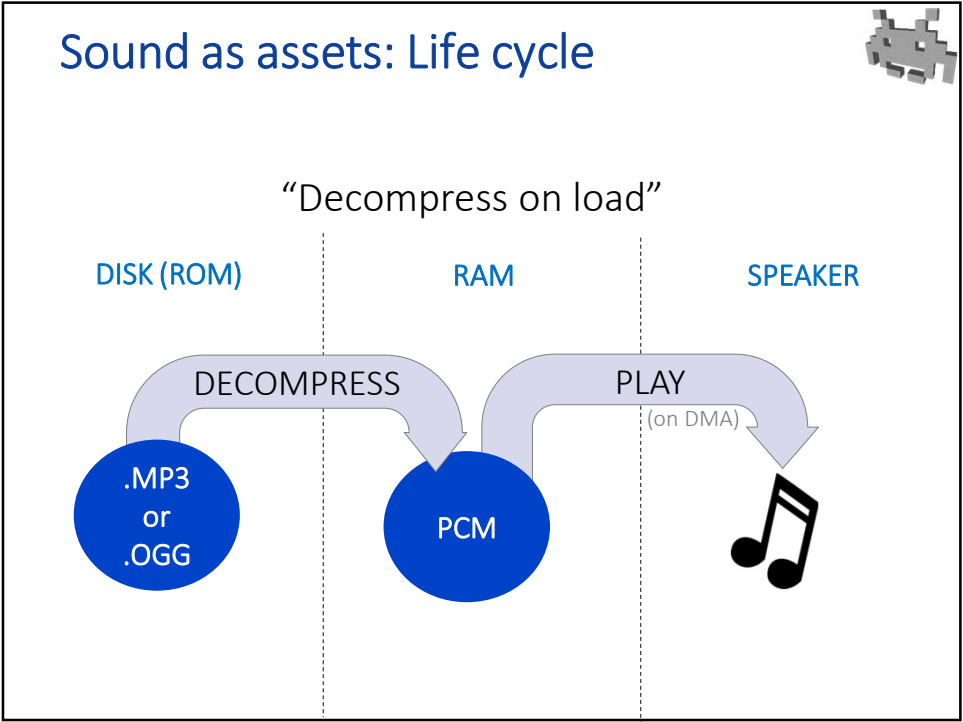
15

Assets for sounds: most common file formats

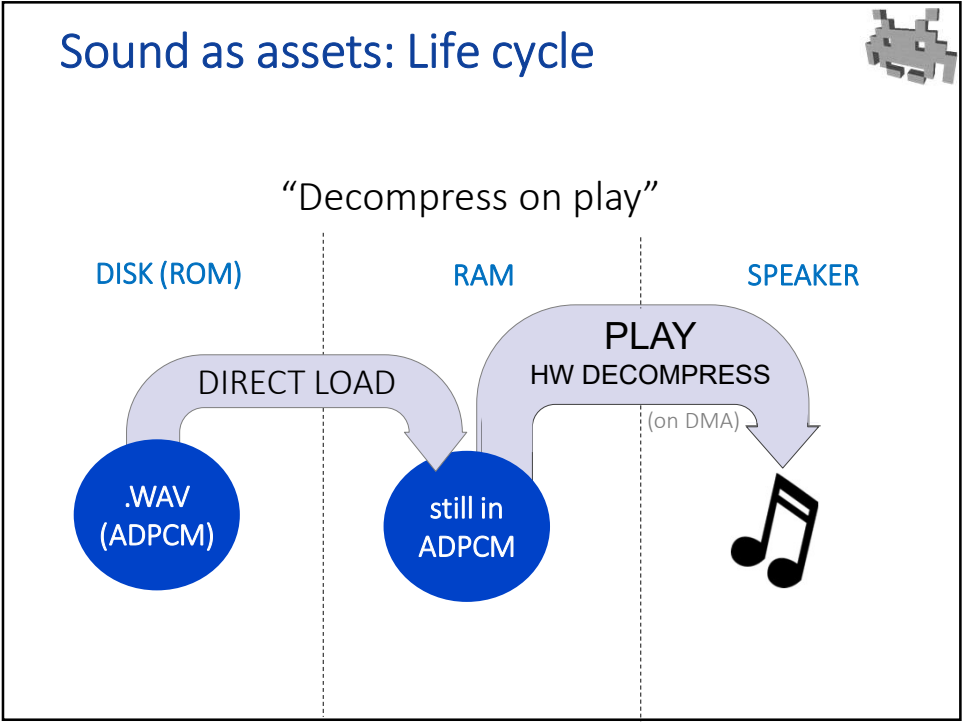


- **.mp3**
 - perceptual encoding
 - good balance between compression-ratio / quality
 - common for final releases / distributions
- **.ogg** (vorbis)
 - optimized for music
 - usually best quality for compressed
- **.wav**
 - uncompressed (PCM)
 - not much used as assets (e.g. unity will compress them)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

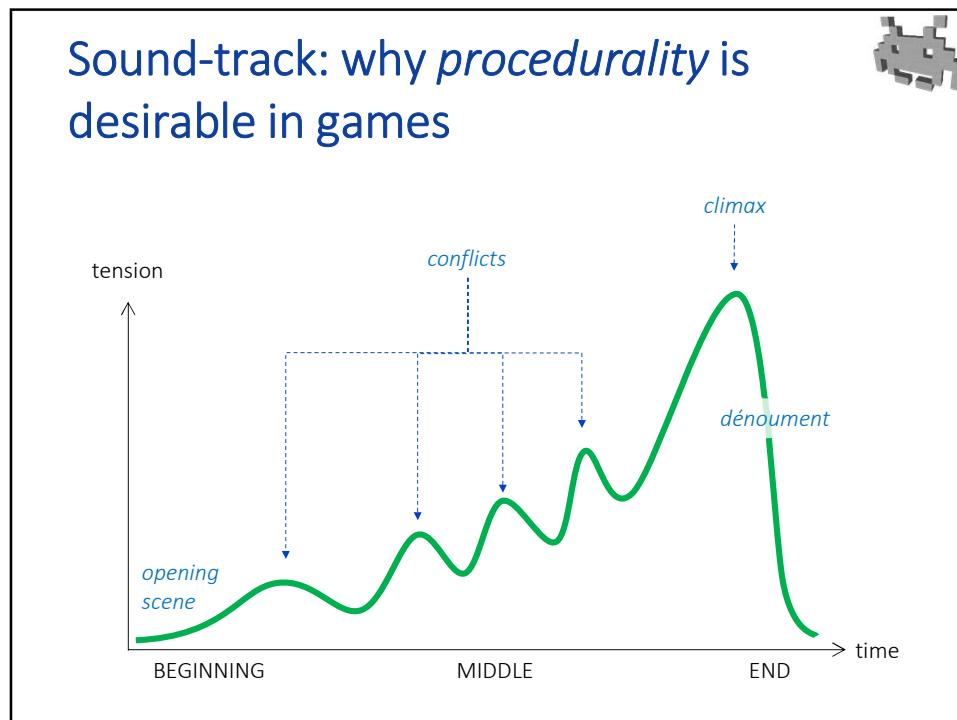
23

Assets for music today



- Music as just another **sampled sound wave**
 - maybe looped (as any other audio)
- Typically made of «stem» (sub-tracks)
 - «bass» stem
 - «guitar» stem
 - «choir» stem ...
- Option 1: single track, with all stems pre-mixed
 - E.g., a plain recording of the performance
 - Or digitally composed musing, pre-mixing and baking stem
- Option 2: keep stems separated, mix them in real-time
 - more resource consuming (computation & RAM)
 - note: today, with ML, it's easy to obtain this from a single-track audio
 - useful for dynamic re-tuning and **non-linear** music
 - that is, allows for some mild form of procedurality (adaptivity)

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/> (have fun)
- Still no standardized interchange format for this asset :-)

26

Specialized assets for Ambient Sounds

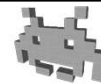


Example:

- 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](#)



29

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

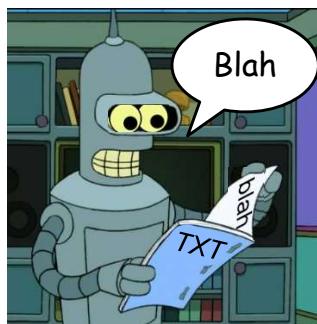
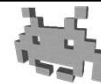
Voice Overs



- 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

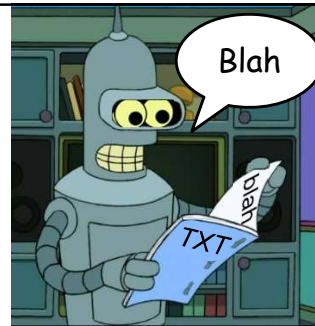
Speech Synthesis (or “text to speech”)



32

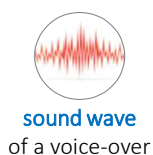
Speech Synthesis (or “text to speech”)

- 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**



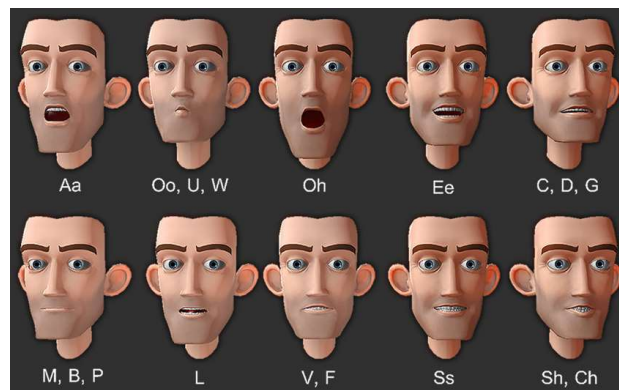
33

Voiceovers and animations 1/2: Blend-shapes for «lip-sync»



Phoneme Recognition

weights for
Facial-Morphs

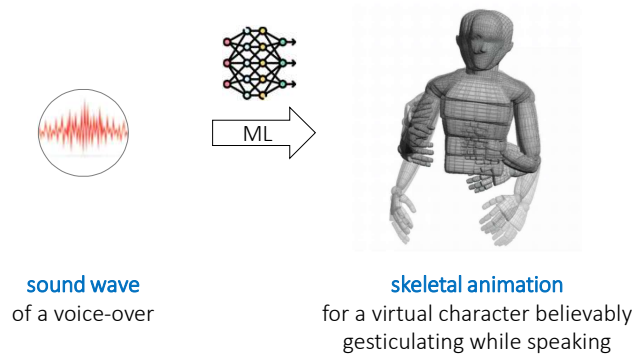


Game: Blue Toad Murder Files (Relentless Software, 2010)
Artist: Kelly Ford, <https://kelly4d.artstation.com/>

34

Voiceovers and animations 2/2: [research topic] from voiceovers to NPC animations

- With Machine Learning (data driven)



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

35

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

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** (see animation lecture)
- fxs synched to certain times of an **Animations**
 - 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 2D ones

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

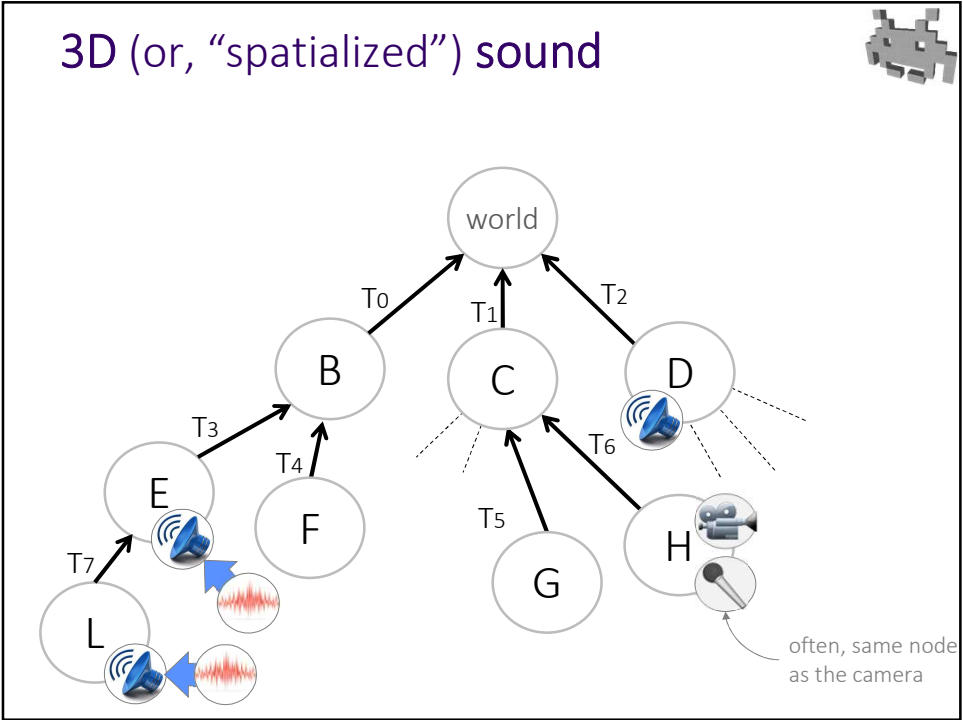
Sound Rendering in 3D games

3D (or, “spatialized”) sound

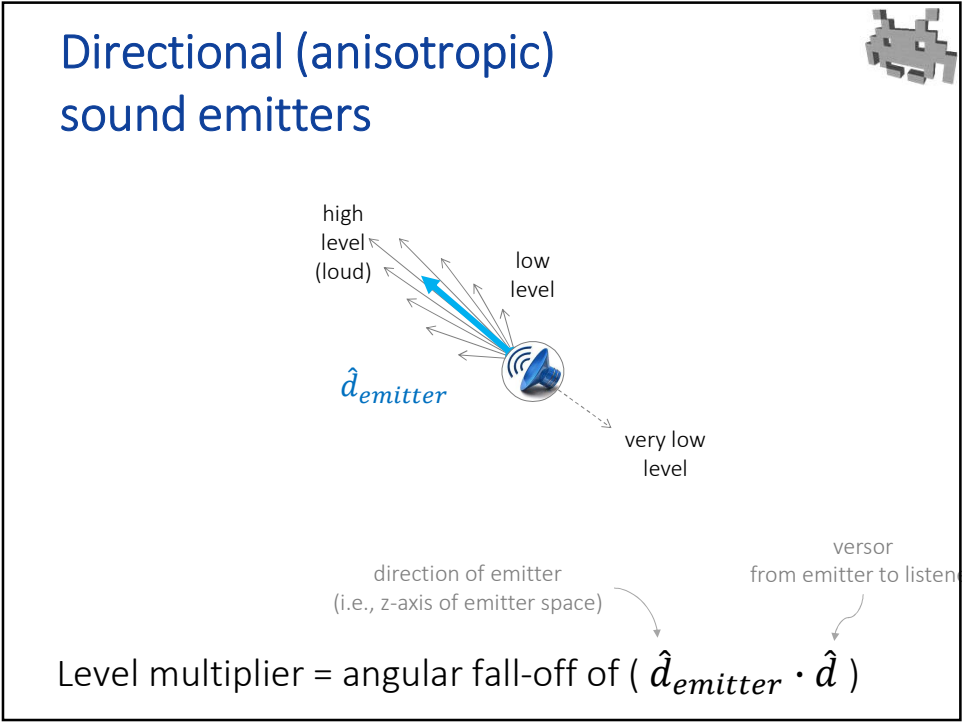
- sounds which are:
 - emitted from a virtual source (somewhere in 3D)
 - received from a virtual microphone (somewhere in 3D)
 - propagated across the 3D scene
- useful abstractions used in games:
 -  the listener
 -  the source(s) } sitting in nodes of the scene graph!

note:
position
and
orientation

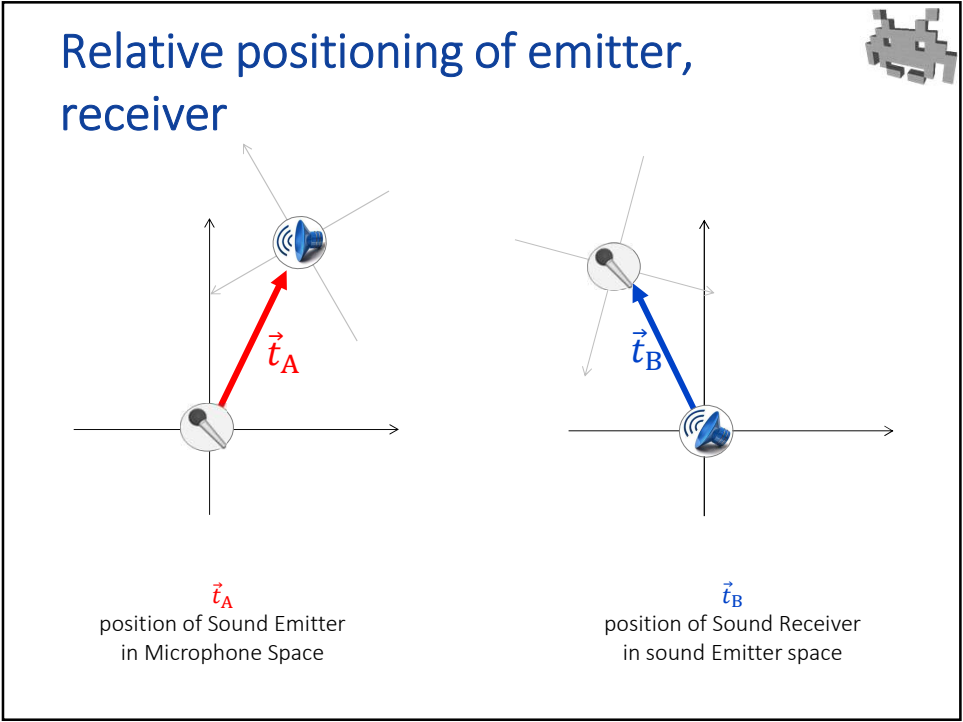
39



40

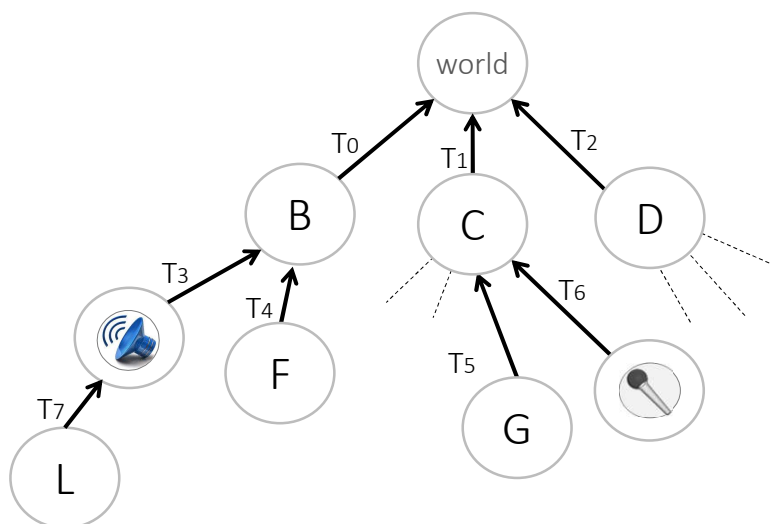


41





42

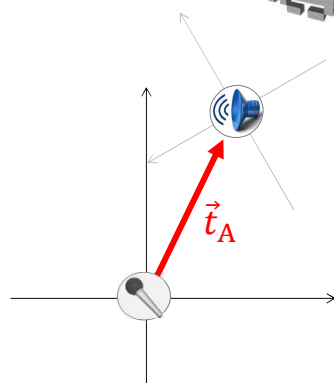
Example (schema)



43

Example 1/2



- What is the position of  in the space of  ?
- Answer:
the translation component \vec{t}_A of $T_A = (T_6)^{-1} \cdot (T_1)^{-1} \cdot T_0 \cdot T_3$

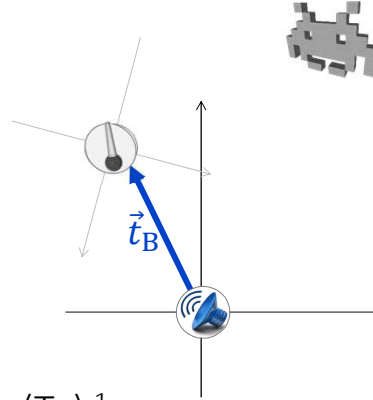


- Needed for determining:
 - Distance emitter-receiver $\|\vec{t}_A\|$ for the level falloff
 - Direction toward sound source $\hat{t}_A = \vec{t}_A / \|\vec{t}_A\|$ determining ILD, ITD and anisotropic spectral cues

44

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



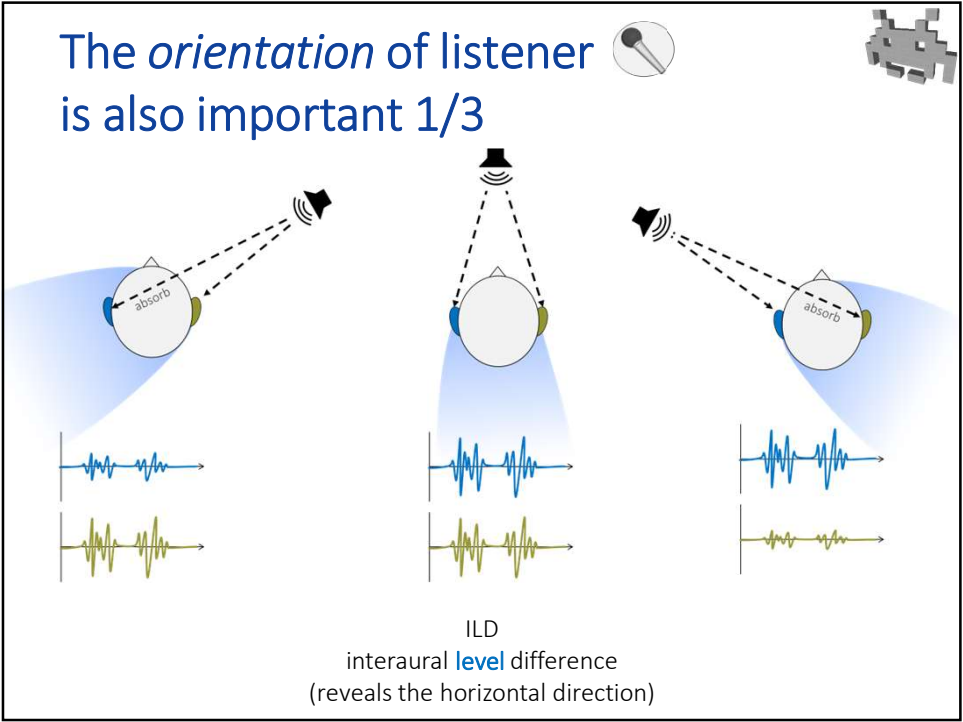
45

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

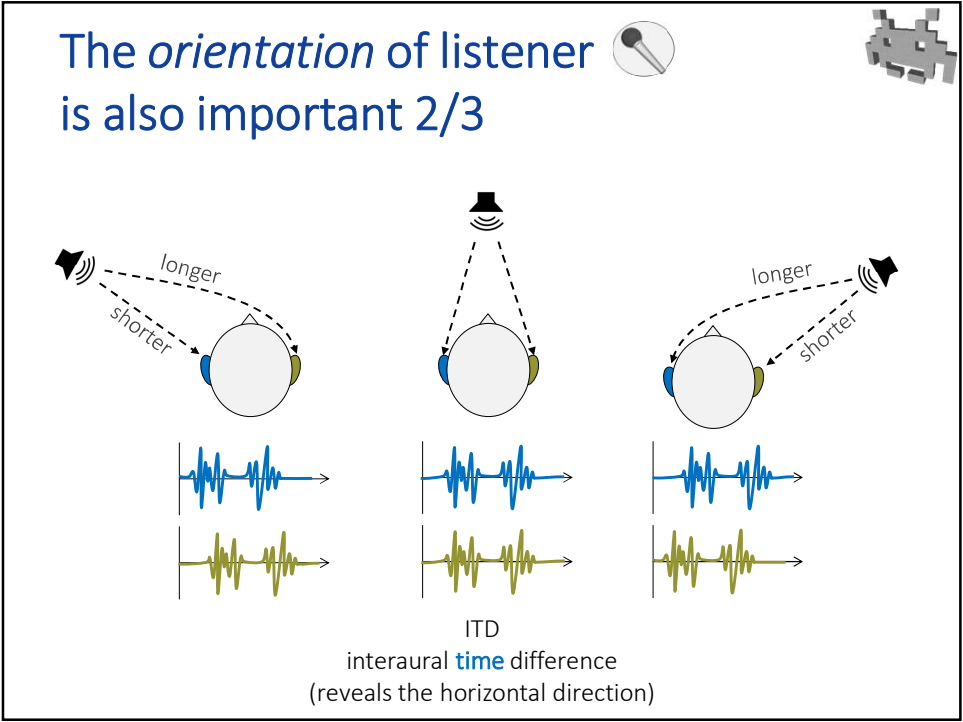
- consequent auto-tuning of
 - level**: (linked to perceived “loudness”) according to source-listener **distance**
 - with a given (dev-controlled) «roll-off» or «fall-off» function
 - E.g. $1/d$ or $1/d^2$
 - pitch**: (Doppler effect) according to **relative speed** or source w.r.t. listener
 - anisotropic spectral clues** and **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. $\sim 10 \mu s$

for mono or stereo speakers
for stereo speakers

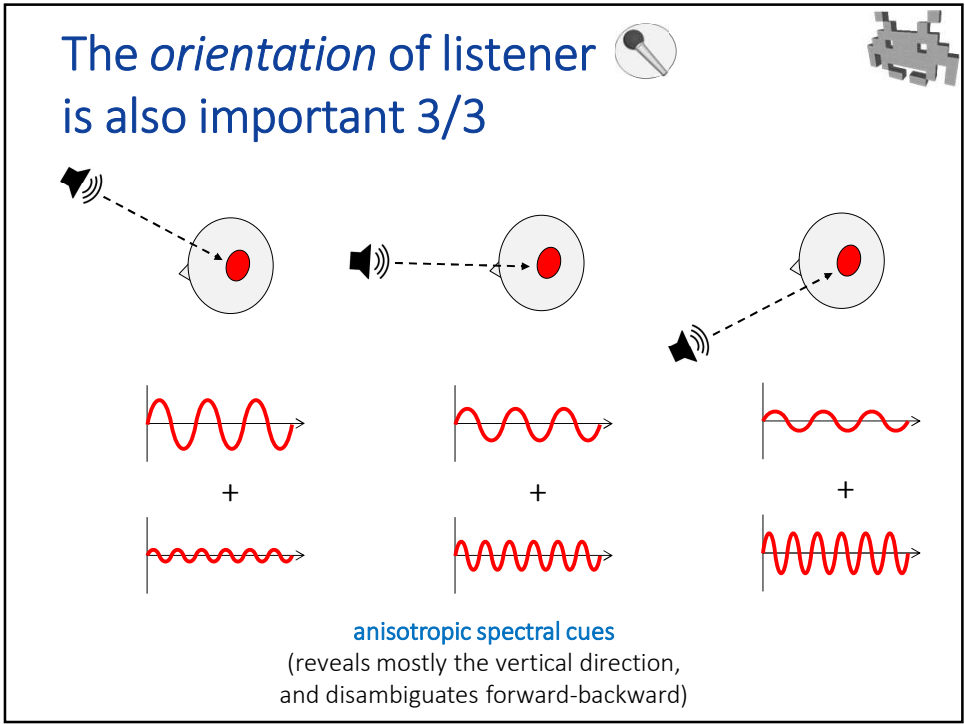
46



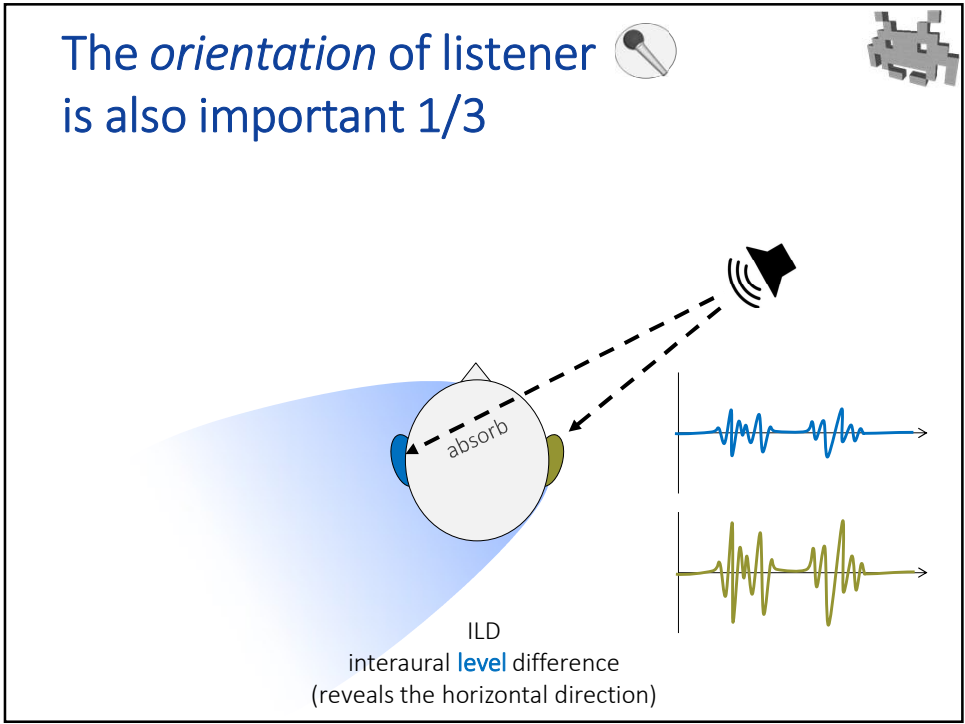
47



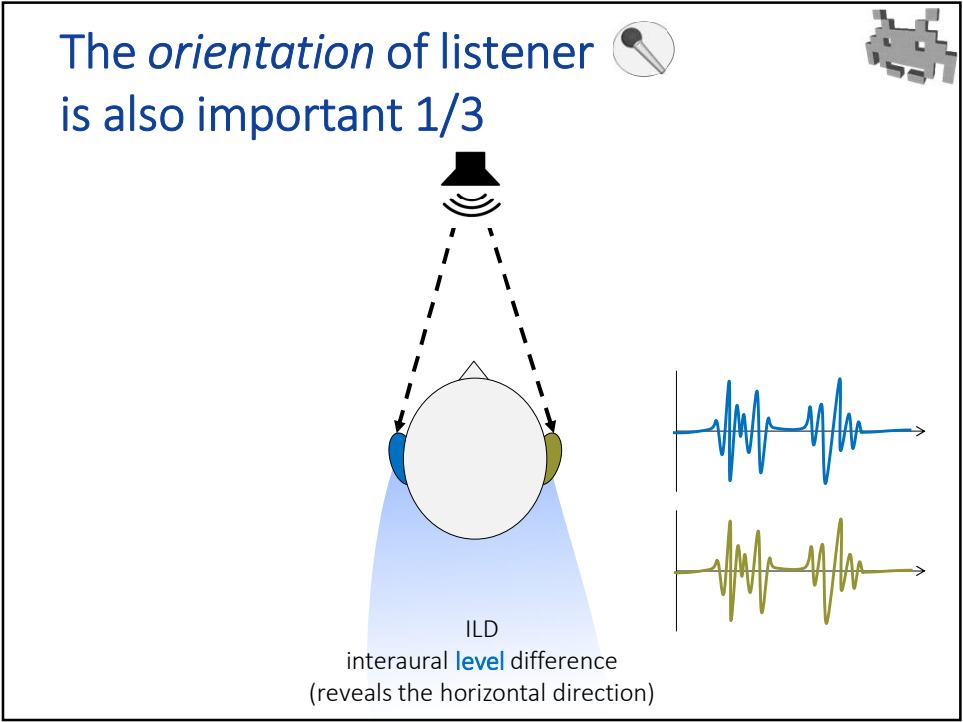
48



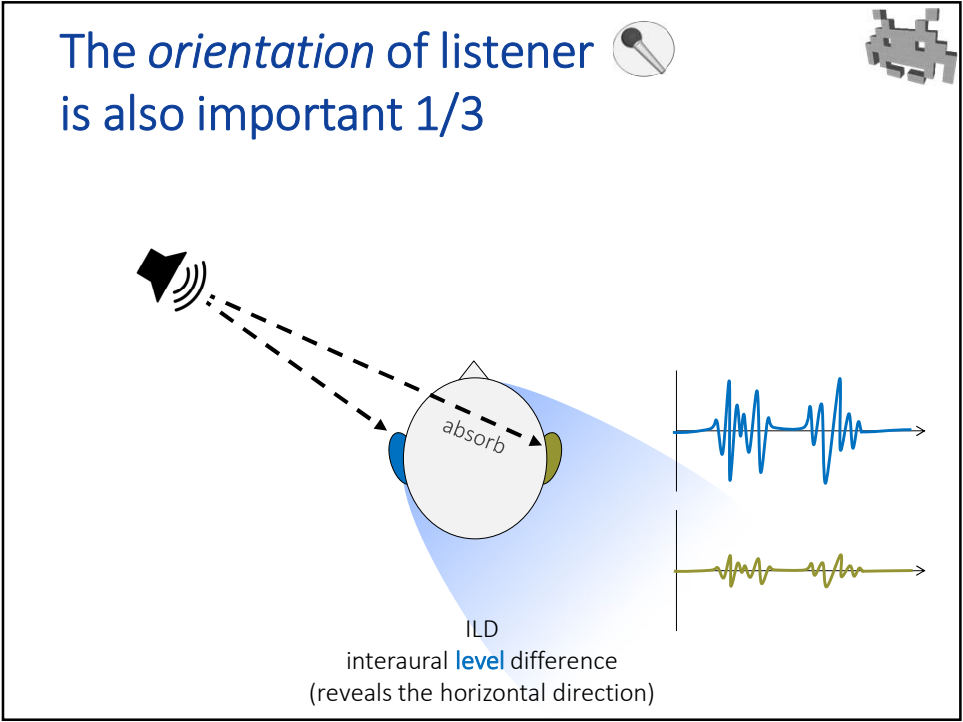
49



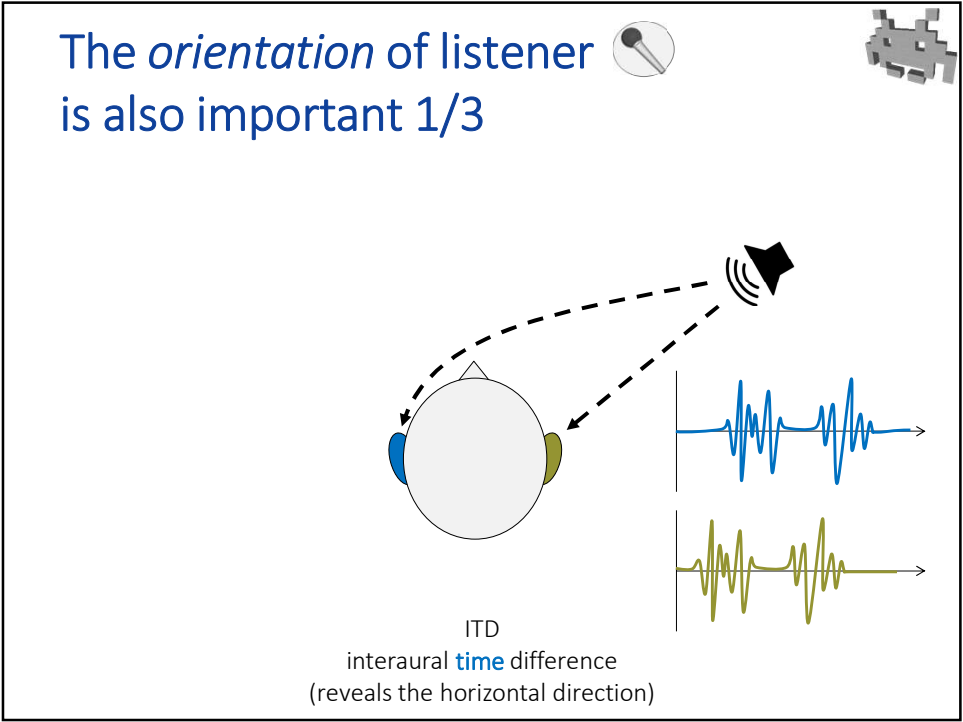
50



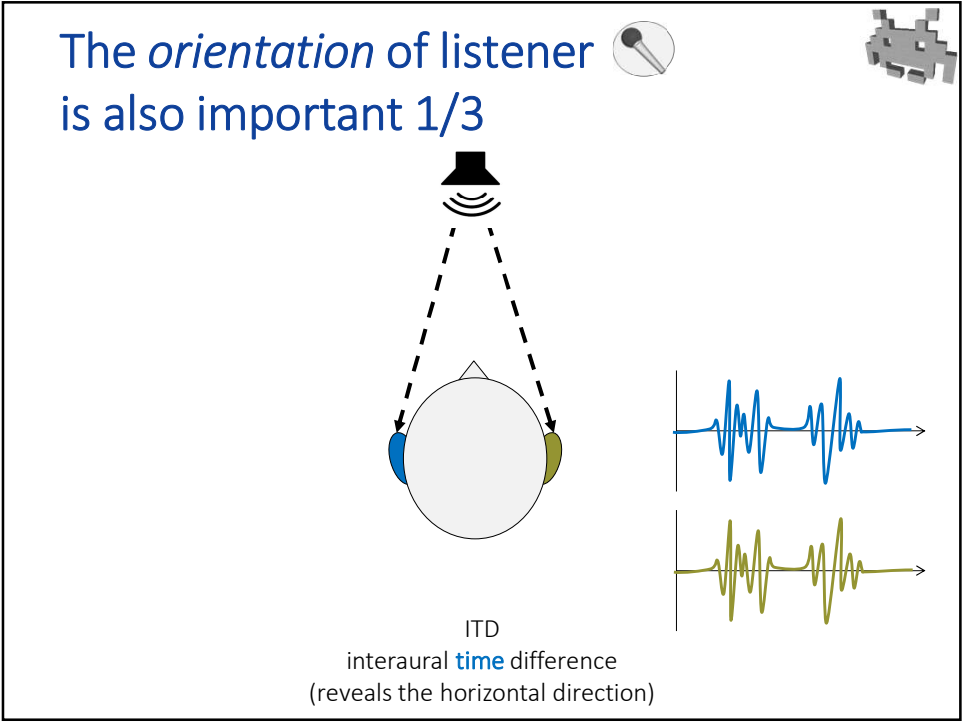
51



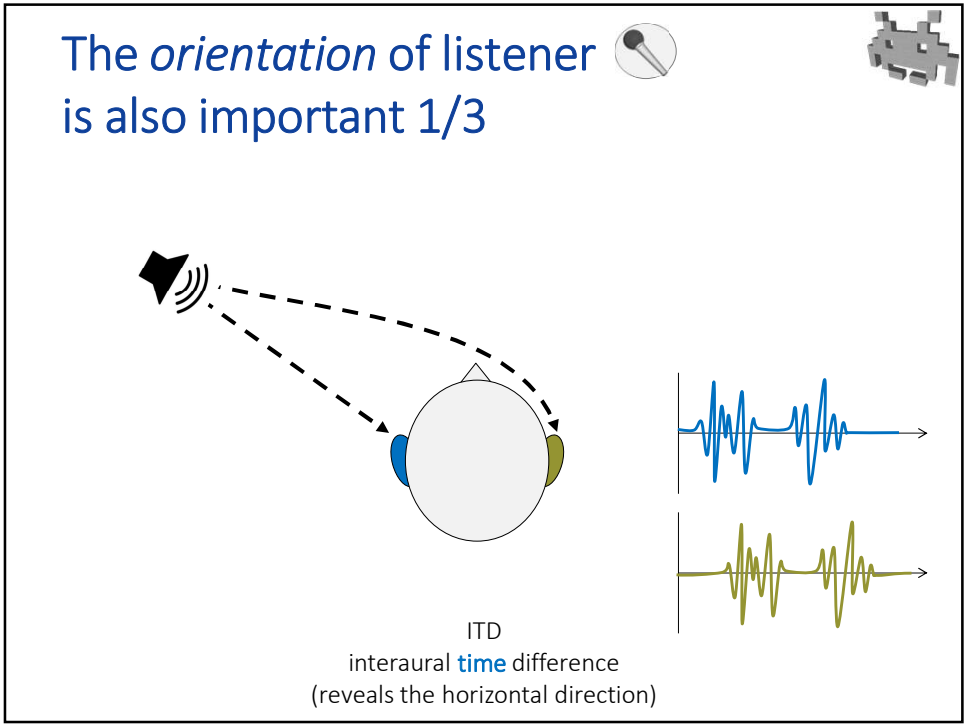
52



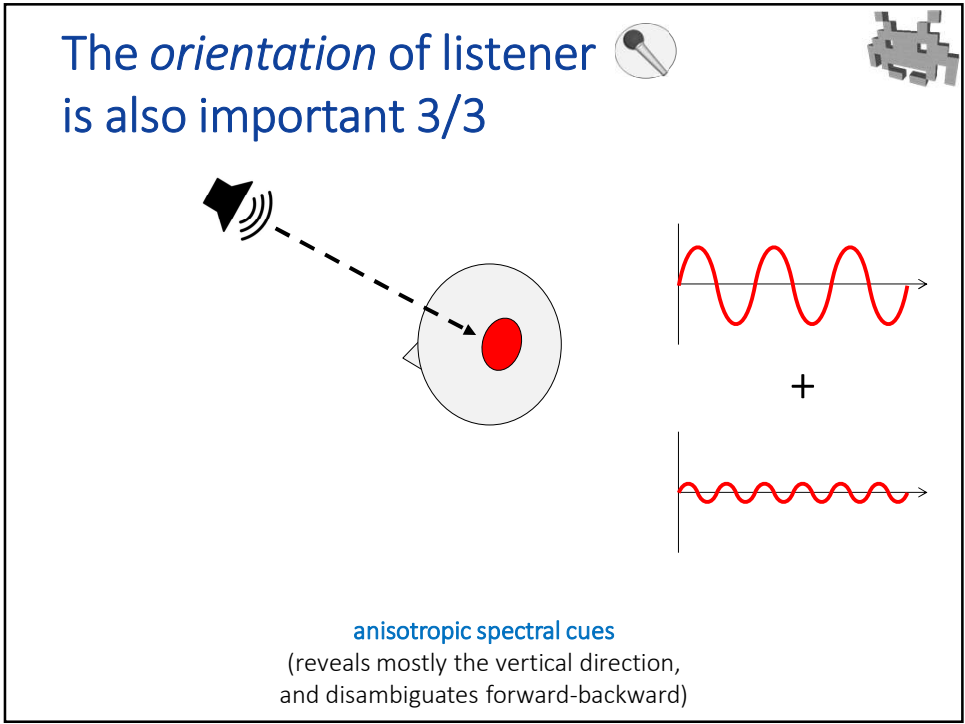
53



54



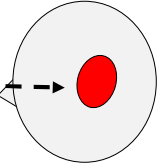



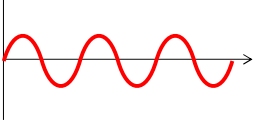
55



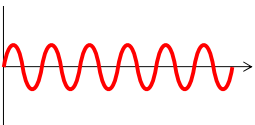
56

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





+



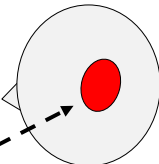



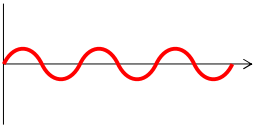
anisotropic spectral cues

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

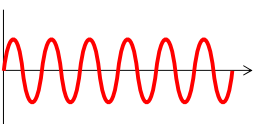
57

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





+



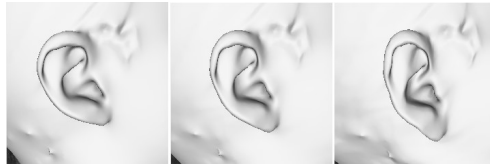
anisotropic spectral cues

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

58

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.

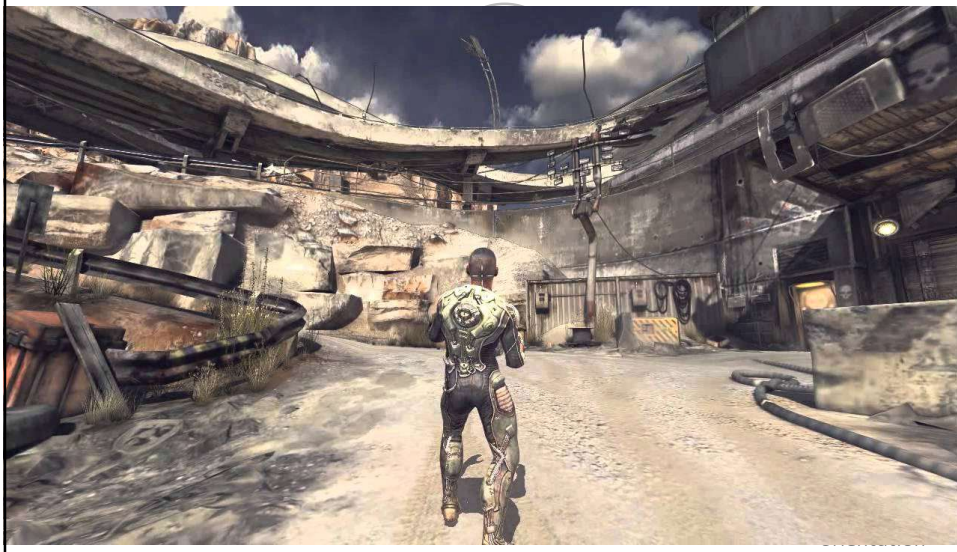


- More commonly, approximations are used

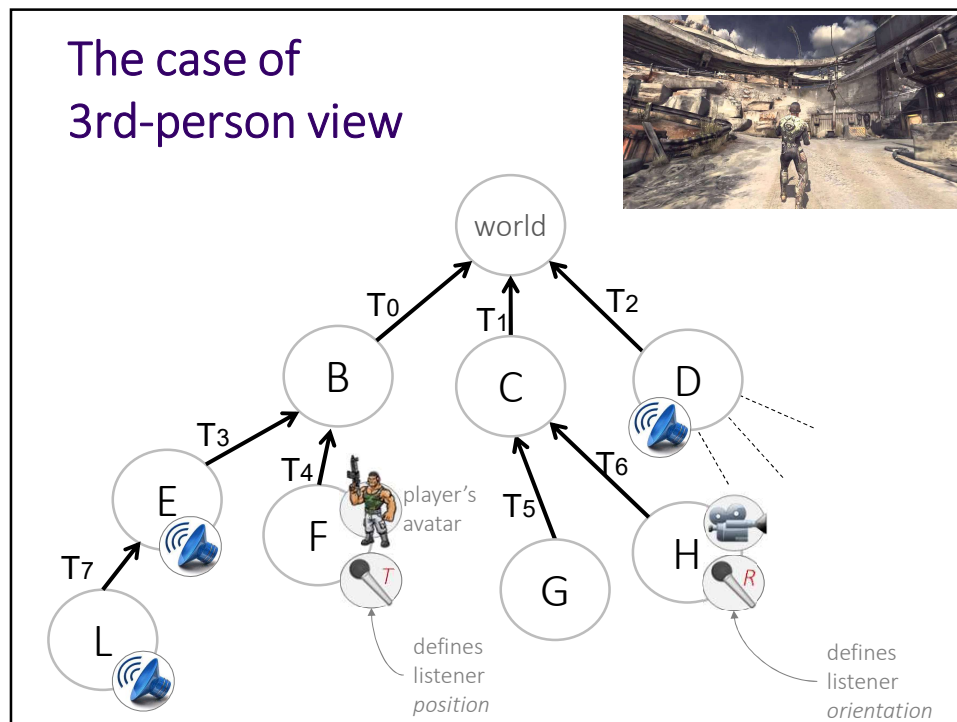
“Reconstructing head models from photographs for individualized 3D-audio processing”
M Dellepiane et al, CGF 27 (7) - (Pacific Graphics)

59

The case of 3rd-person view



60



61

Sound Rendering: sound propagation in the 3D scene

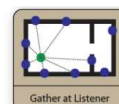
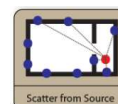
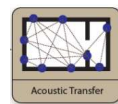
- So far, we considered the 3D effects of sound-waves propagated directly from emitter to microphone
- In reality, sound-waves interact with solids in the 3D scene
- Three basic phenomena:
 - Absorption:**
some* energy of the sound-wave is lost (dissipated into heat)
 - Reflection:**
some* part of the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

62

Sound Rendering: sound propagation in the 3D scene

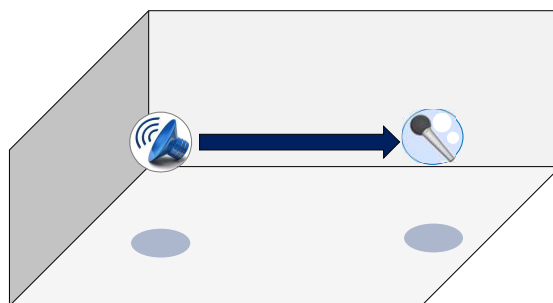
- Using **collision proxies**!
- Targets simulation of effects by:
 - Absorption (occlusion, obstruction)
 - Transmission (muffling)
 - Reflections (reverb, echoes)
- Active research topic
 - Currently: no standard solution adopted by 3D games
 - Often, tricks coded *ad-hoc* by the **sound programmer**



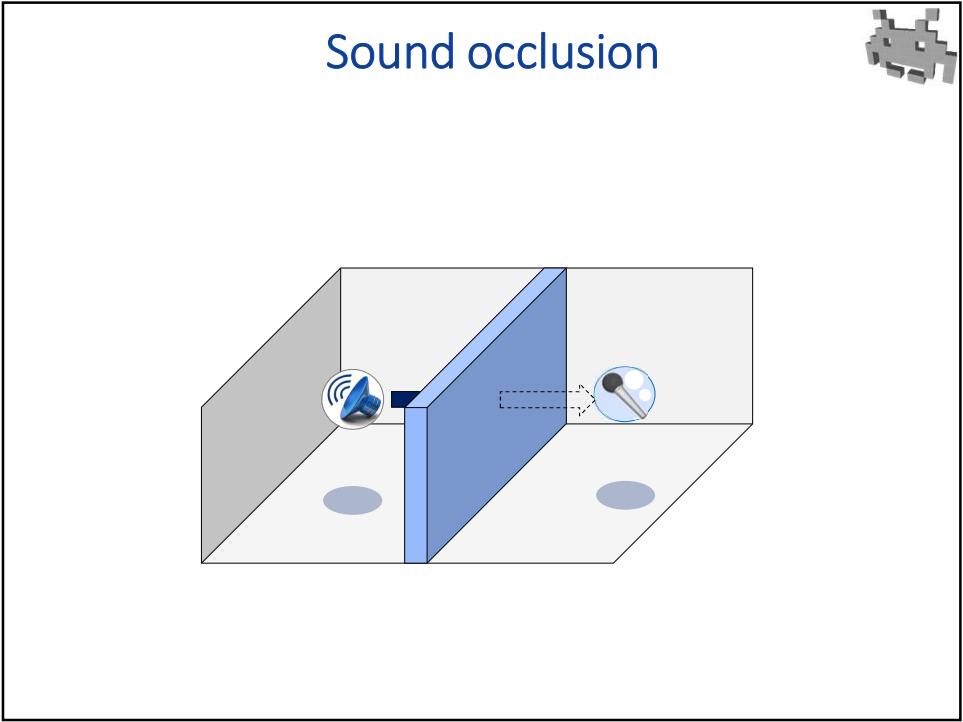
E.g. see: "Interactive Sound Propagation using Compact Acoustic Transfer Operators"
Lakulish Antani, Anish Chandak, Lauri Savioja, Dinesh Manocha
SIGGRAPH 2012

63

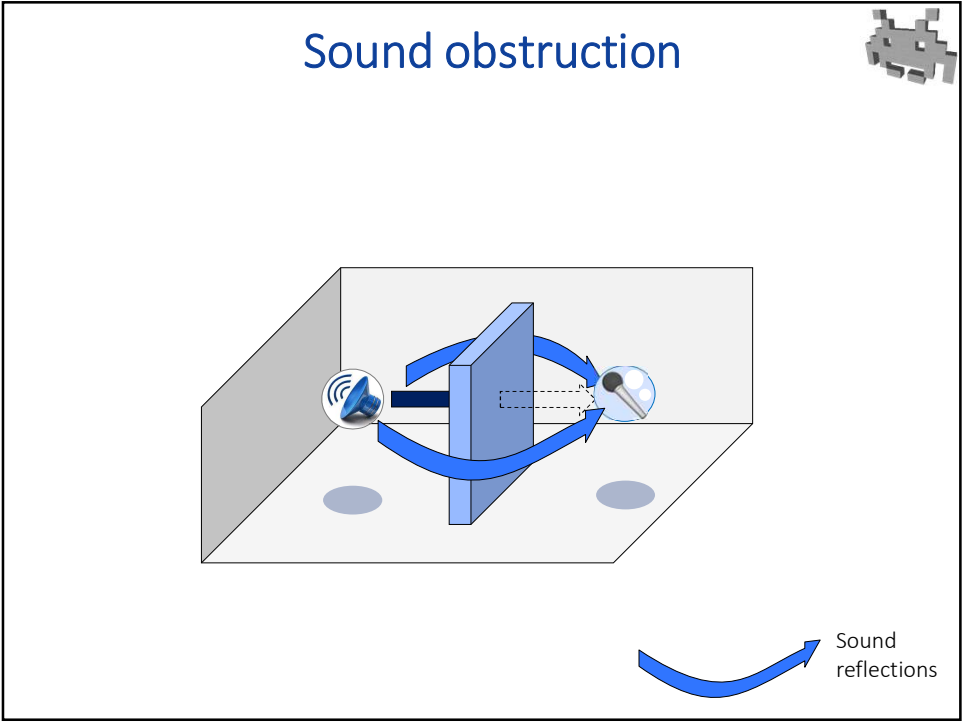
Direct sound propagation (the one we assumed so far)



64



65

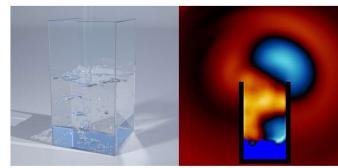
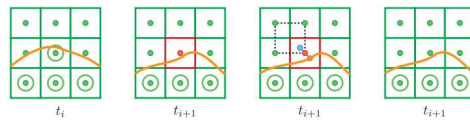


66

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



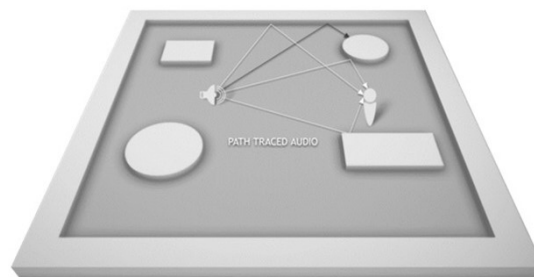
E.g. see: "Toward Wave-based Sound Synthesis for Computer Animation"
Jui-Hsien Wang, Ante Qu, Timothy R. Langlois, Doug L. James
SIGGRAPH 2018

67

Sound Reverb



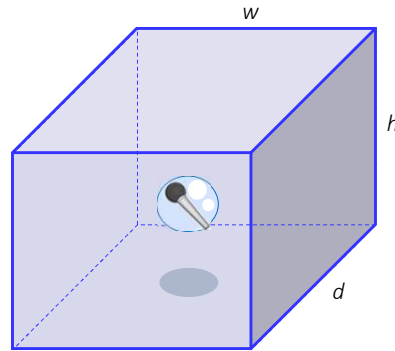
- Solution 1: path tracing (expensive!)



68

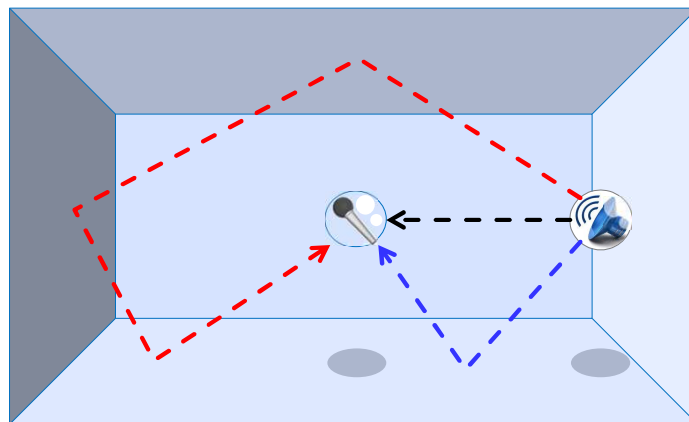
Sound Rerverb

- Solution 2: «shoe-box model»
 - a widely used approximation
 - comes with closed-form formulas!



69

Shoe-box model for sound reverb



70