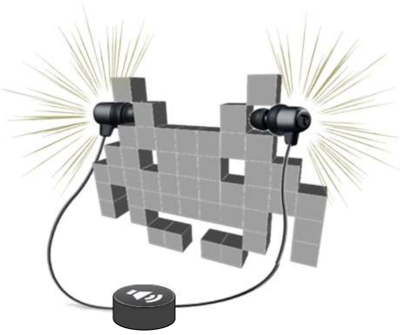
Università di Milano
3D VideoGames 2020/2021

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. 8: Game 3D Animations ●●●
lec. 9: Game 3D Audio ●
lec. 10: Networking for 3D Games ●
lec. 11: Artificial Intelligence for 3D Games ●
lec. 12: Game 3D Rendering Techniques ●●

For a much more in-depth discussion of many of the subjects of this lecture, see the course «Sound in interaction»

2

Game Audio: intro

- 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 unsubtle

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 / 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**"*

4

Audio in games: game-design point of view

- Sound effects are super informative
  - effective way to make things clear 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 1^{st} 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

5

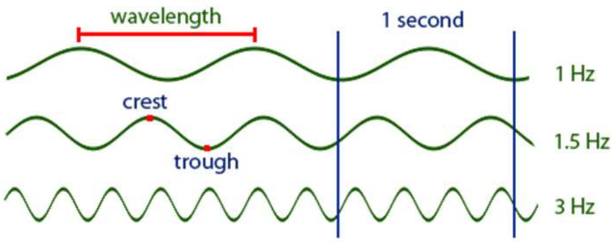
Audio in games: dev-team roles

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

6

Sound wave

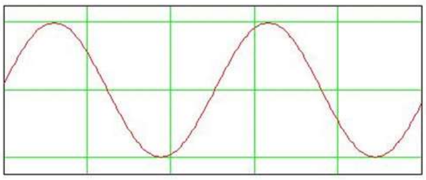
- Air pressure as a function of time
- frequency : (1/sec = 1 Hz)



7

Sound wave

- Air pressure as a function of time
- Waves:
  - frequency (Hz, audible = ~32 to ~16K),
  - amplitude (→ "volume", level, perceived loudness)



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

8

Sound wave & perception 101

What it is: *physical property of the sound wave*	*What it is perceived like:* *by the human hearing system*
Amplitude (crest-to-trough, or crest-to-crest)	Level or loudness (colloquially, Volume) how loud the sound is
Frequency (1/wavelength)	Pitch how high-pitched or low-pitched the sound is *[Ita: acuto o grave]*

logarithmic →

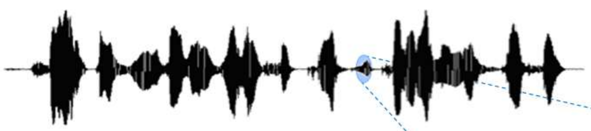
← *exponential*

Spectrum (which frequencies are present)	Timber, tone

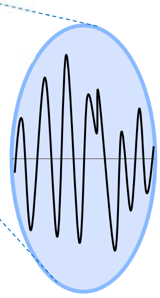
9

Sound wave as assets

- Air pressure as a function of time



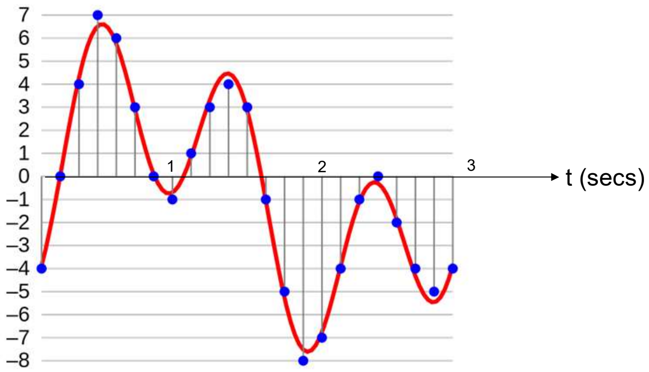
- To digitalize it ("PCM"):
  - sample it
    - at some fixed rate
    - typically, 24-48 KHz
  - quantize samples
    - at some fixed precision
    - typically, 14-24 bits per sample
  - then maybe compress it



10

PCM – Pulse Code Modulation

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



11

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
  - needs de-compression *before* it is played

12

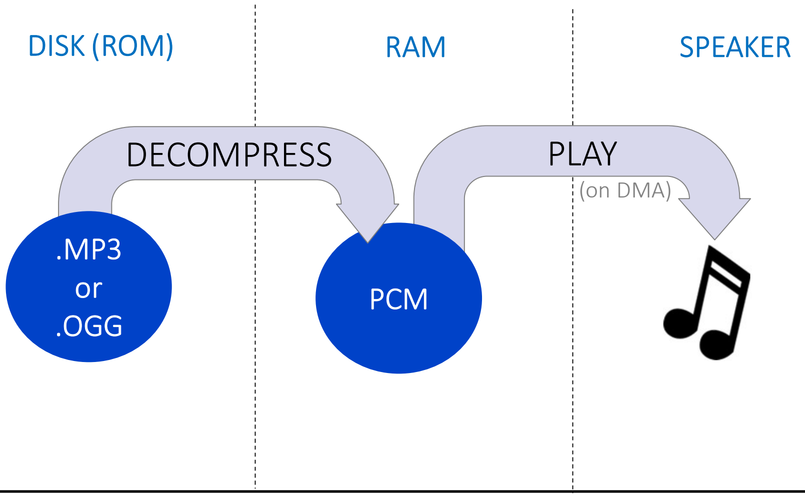
Assets for sounds: most common file formats

- .mp3
  - perceptual encoding
  - good balance between compression-ratio / quality
  - most common mass-storage format
- .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)

13

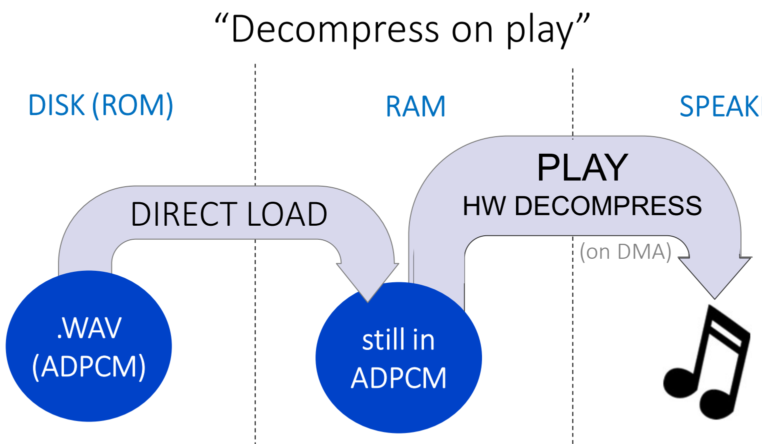
Sound as assets: Life cycle

“Decompress on load”



14

Sound as assets: Life cycle



15

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
    - costs *even more* RAM
  - variant: decompress on Play
    - requires HW support
    - 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)

16

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

17

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…

18

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.

19

Specialized assets for music

- Store a digital *score* instead?
- The *traditional* music asset in games
  - any classic game tune you can remember was originally stored in this way
    - (think Pacman, Super Mario Bros, Tetris, …)
  - the only way – until the ’90
- Example file format: MIDI
- Pros:
  - much cheaper to store — what used to make this a strict necessity
  - perfect for procedural music — makes this still attractive today (a bit)
    - (e.g. non linear soundtrack)
- Cons:
  - requires instrument library (samples) at runtime
  - limits expressiveness — made this almost abandoned today
    - (e.g. voice, choir, subtleties)
  - limits authoring procedures

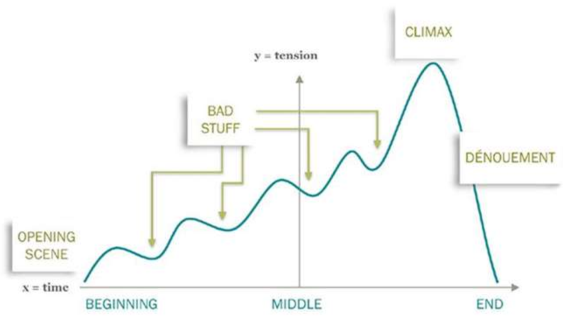
20

Assets for music today

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

21

Sound-track: why some degree of *procedurality* is needed in games



22

Specialized assets for Ambient Sounds

- Ambience track (“drone” – from *ita*: bordone)
  - 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 :-(

23

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

24

Middleware for sounds in games

oculus

fmod

Resonance Audio
by Google

STEAM AUDIO

Libs: OpenAL , Wwise …

25

Sound Rendering: *basic* playback tasks

Main Asset:

the sound buffer

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

in any game, even in a 2D setting

- 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 filters
  - convolutions of sound buffer
  - useful to add procedural effects as reverb, echos…
- 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

26

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

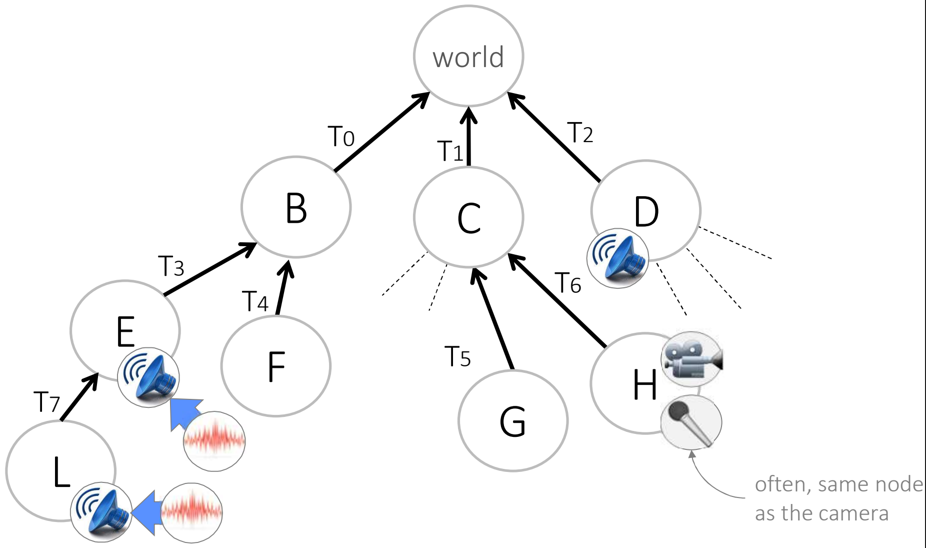
27

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» 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): 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. *~10 µs*

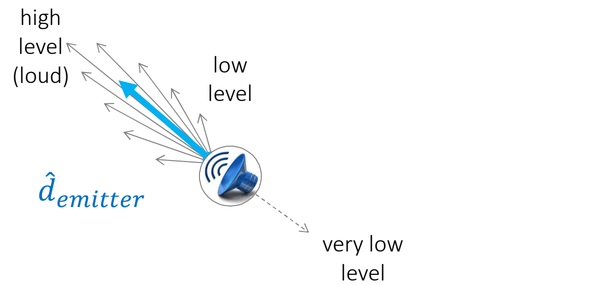
28

3D (or, "spatialized") sound



29

Directional (anisotropic) sound emitters



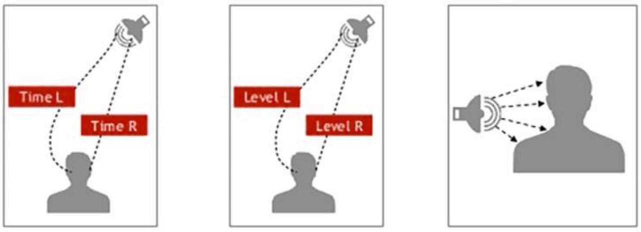
direction of emitter
(i.e., z-axis of emitter space)

versor
from emitter to listener

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

30

Listener *orientation* is also important

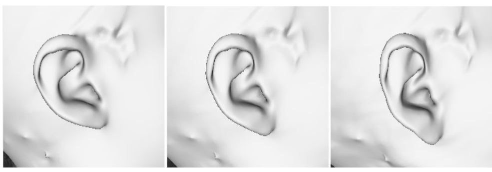


interaural time difference | interaural level difference | anisotropic spectral cues

31

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.

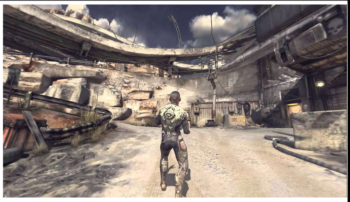


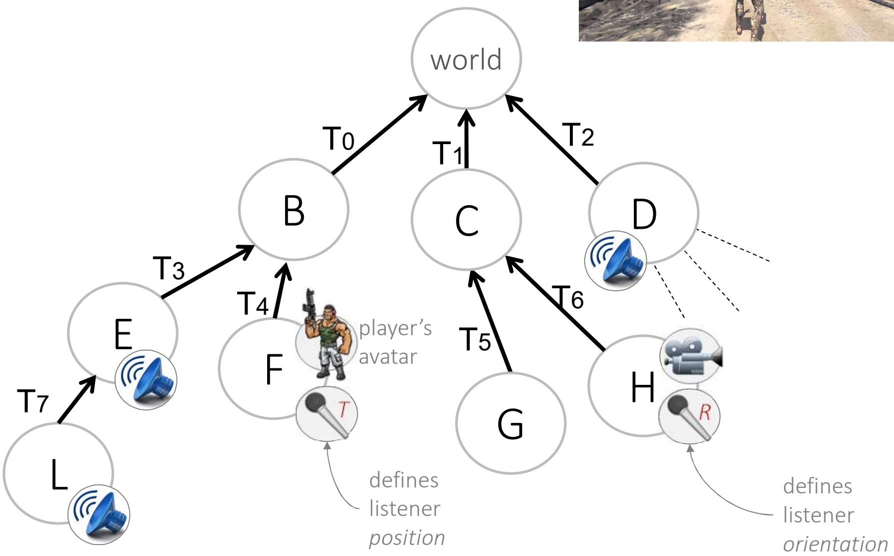
- More commonly, approximations are used

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

32

The case of 3rd-person view





34

Sound Rendering: sound propagation in the 3D scene

- So far, we only considered the 3D effects of sound direct propagated 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

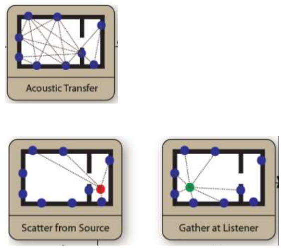
* *how much of it?* It depends on the materials, and the wave-length

35

Marco Tarini
Università degli studi di Milano

Sound Rendering:
sound propagation in the 3D scene

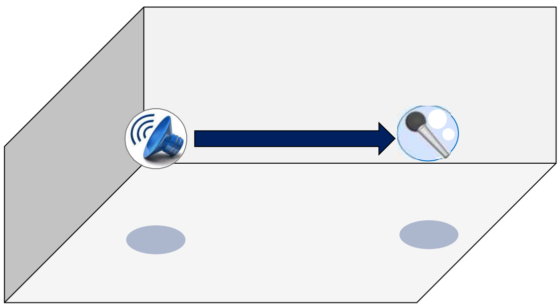
- Reuse collision proxies!
- Targets simulation of effects by:
  - Absorption (occlusion, obstruction)
  - Transmission (muffling)
  - Reflections (reverb, echoes)
- Active reseach topic
  - Currently: no standard solution adopted by 3D games
  - Often, tricks coded *ad-hoc* by the sound programmer



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

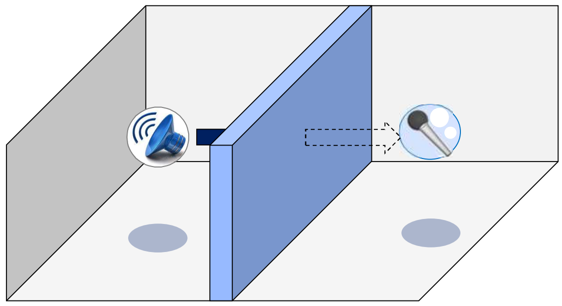
36

Direct sound propagation



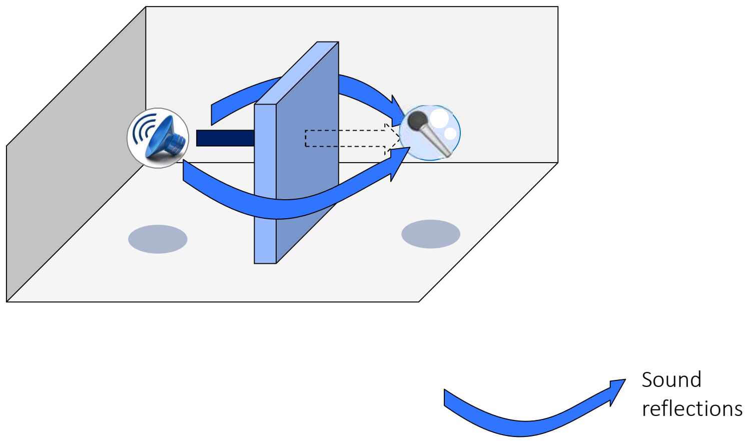
37

Sound occlusion



38

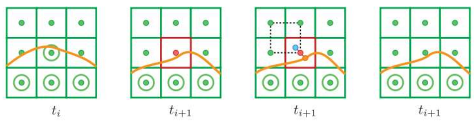
Sound obstruction

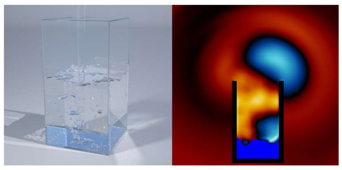


39

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



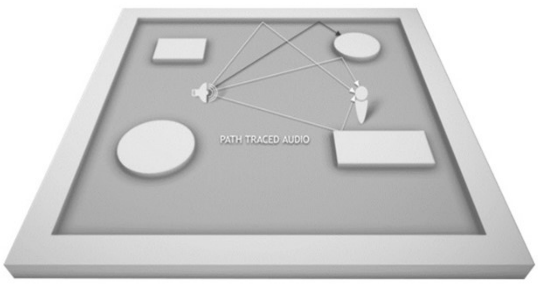


E.g. see: "Toward Wave-based Sound Synthesis for Computer Animation"
Jui-Hsien Wang, Ante Qu, Timothy R. Langlois, Doug L. James
SIGGRAPH 2018

40

Sound reverb

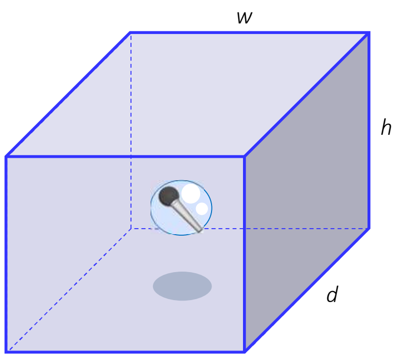
- Solution 1: path tracing (expensive!)



41

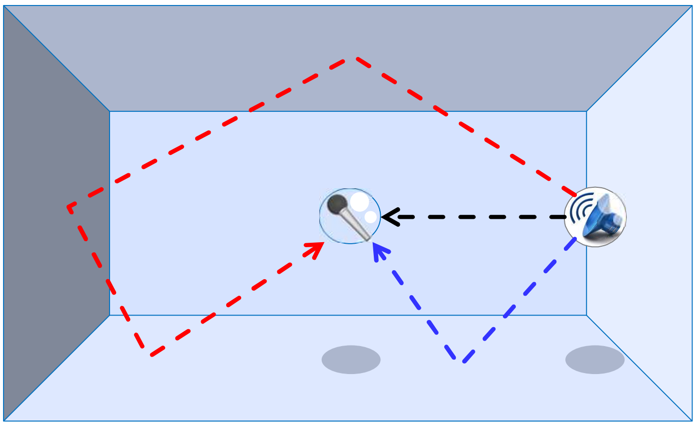
Sound Rerverb

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



42

Shoe-box model for sound reverb



43

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

44

Authoring sound effects (task of the Sound Designer)

- Remember: as any asset, you can buy / get them from Libraries / Repositories
  - Common (so many needed fxs, so little time)
- Capture
  - Digital artist: “Foley”
  - Field capture (for ambient sounds → drones)
- Synthetize
  - by sound editing
  - (rarer)

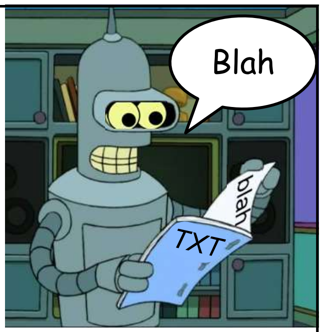
45

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

46

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
    - audio “uncanny valley” ?
  - 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

48

A summary of authoring sound assets

- Synthesized / simulated / procedural *fxs* :
  - baked
  - (rare)
- Captured *fxs* :
  - hardware: a good microphone!
  - by: "Foley artists"
  - very often: just bought / downloaded from repositories
- Voice :
  - hardware: a good microphone!
  - by voice actors
    - (sometimes, during motion capture sections)
  - speech synthesis? won't be used (for some time yet)

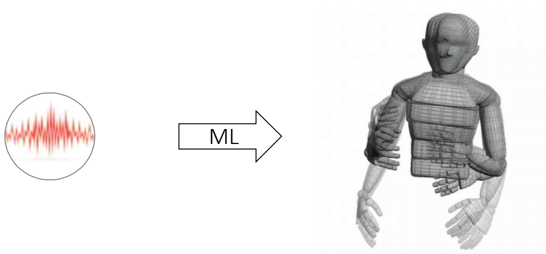
then, sound editing

- 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!

49

Research topic: from voiceovers to NPC animations

- With Machine Learning (data driven)



sound wave
of a voice-over

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)

50