

Università di Milano
3D VideoGames 2018/2019




3D Game Audio *(notes)*

Marco Tarini



1


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 subtle

2

Sound in games: game-design point of view




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

e.g.:

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

3

Sound in games: game-design point of view

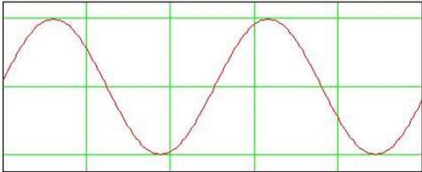


- **Sound effects** are mighty **informative**
 - effective way to make things clear to the player.
 - examples:
 - out of ammo:
 - gun just doesn't shoot → wrong key? a bug?
 - gun goes "click" → player gets it
 - doors closes *behind* player in 1st person view
 - sound door-slam effect: let him know!
 - 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

4

Sound wave

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

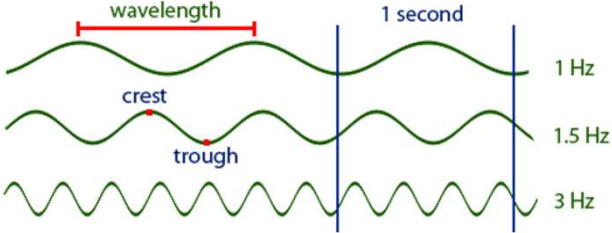


- Perception
 - as with most senses, response is roughly logarithmic with physical quantity (see: decibel)

5

Sound wave

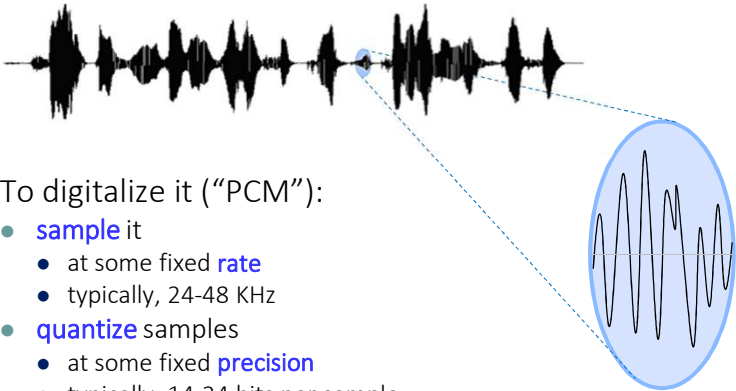
- Air pressure as a function of time
- frequency : (1/sec = 1 Hz)



6

Sound wave

- Air pressure as a function of time

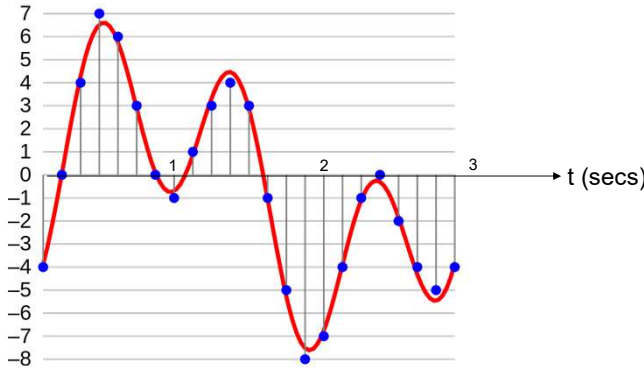


- To digitalize it ("PCM"):
 - **sample** it
 - at some fixed **rate**
 - typically, 24-48 KHz
 - **quantize** samples
 - at some fixed **precision**
 - typically, 14-24 bits per sample
 - then maybe **compress** it

7

PCM – Pulse Code Modulation

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



8

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* play

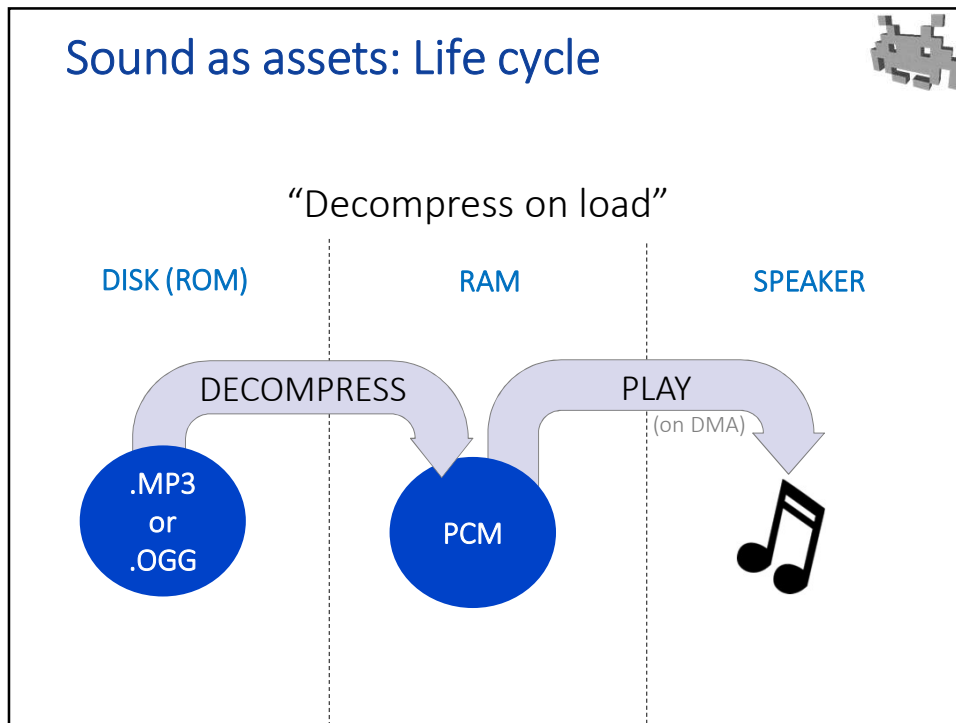
9

Assets for sounds: most common file formats

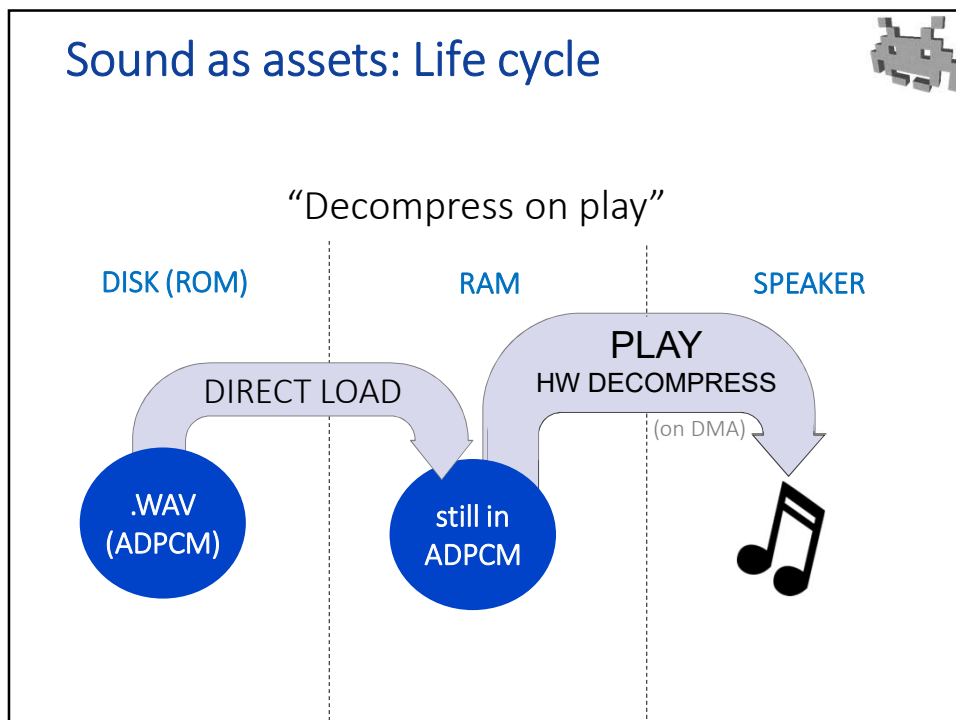


- **.mp3**
 - perceptual encoding
 - good balance between compression-ratio / quality
 - most common mass-storage format
- **.ogg** (vorbis)
 - optimized for music
 - usually best quality for compressed
- **.wav**
 - uncompressed (PCM)
 - not much used as assets (e.g. unity will compress them)
 - or,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

10



11



12

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)

13

Compare: ADPCM – audio compression, and 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

14