

# CS1315: Introduction to Media Computation

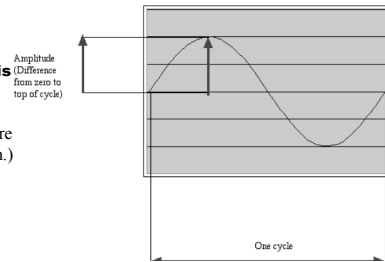
Sound Encoding and Manipulation

## Today's class: Sound encoding

- Theory: What sound is and how we digitize it
- Practice: Playing sounds. Querying a sound's properties

## How sound works: Acoustics, the physics of sound

- Sounds are waves of air pressure
  - **Sound comes in cycles**
  - **The frequency of a wave is the number of cycles per second (cps), or Hertz**
    - (Complex sounds have more than one frequency in them.)
  - **The amplitude is the maximum height of the wave**



The diagram shows a sine wave oscillating between two horizontal lines. A vertical double-headed arrow on the left indicates the amplitude, labeled 'Amplitude (Difference from zero to top of cycle)'. A horizontal double-headed arrow at the bottom indicates the duration of one full cycle, labeled 'One cycle'.

## Volume and pitch: Psychoacoustics, the psychology of sound

- Our perception of volume is related (logarithmically) to changes in amplitude
  - **If the amplitude doubles, it's about a 3 decibel (dB) change**
- Our perception of pitch is related (logarithmically) to changes in frequency
  - **Higher frequencies are perceived as higher pitches**
  - **We can hear between 5 Hz and 20,000 Hz (20 kHz)**
  - **A above middle C is 440 Hz**

## “Logarithmically?”

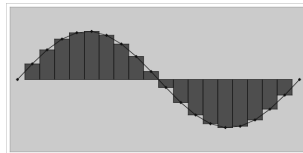
- It’s strange, but our hearing works on *ratios* not *differences*, e.g., for pitch.
  - We hear the difference between 200 Hz and 400 Hz, as the same as 500 Hz and 1000 Hz
  - Similarly, 200 Hz to 600 Hz, and 1000 Hz to 3000 Hz
- Intensity (volume) is measured as *watts per meter squared*
  - A change from 0.1W/m<sup>2</sup> to 0.01 W/m<sup>2</sup>, sounds the same to us as 0.001W/m<sup>2</sup> to 0.0001W/m<sup>2</sup>

## Decibel is a logarithmic measure

- A *decibel* is a ratio between two intensities:  $10 \cdot \log_{10}(I_1/I_2)$ 
  - As an absolute measure, it’s in comparison to threshold of audibility
  - 0 dB can’t be heard.
  - Normal speech is 60 dB.
  - A shout is about 80 dB

## Digitizing Sound: How do we get that into numbers?

- Remember in calculus, estimating the curve by creating rectangles?
- We can do the same to estimate the sound curve
  - **Analog-to-digital conversion (ADC) will give us the amplitude at an instant as a number: a *sample***
- How many samples do we need?



Use *openSoundTool* to view a sound graphically

## Nyquist Theorem

- We need twice as many samples as the maximum frequency in order to represent (and recreate, later) the original sound.
- The number of samples recorded per second is the *sampling rate*
  - If we capture 8000 samples per second, the highest frequency we can capture is 4000 Hz
    - That’s how phones work
  - If we capture more than 44,000 samples per second, we capture everything that we can hear (max 22,000 Hz)
    - CD quality is 44,100 samples per second

## Digitizing sound in the computer

- Each sample is stored as a number (two bytes)
- What's the range of available combinations?
  - **16 bits,  $2^{16} = 65,536$**
  - **But we want both positive and negative values**
    - To indicate compressions and rarefactions.
  - **What if we use one bit to indicate positive (0) or negative (1)?**
  - **That leaves us with 15 bits**
  - **15 bits,  $2^{15} = 32,768$**
  - **One of those combinations will stand for zero**
    - We'll use a "positive" one, so that's one less pattern for positives

## +/- 32K

- Each sample can be between -32,768 and 32,767

Why such a bizarre number?

Because  $32,768 + 32,767 + 1 = 2^{16}$

< 0

> 0

0

i.e. 16 bits, or 2 bytes

Compare this to 0..255 for light intensity

(i.e. 8 bits or 1 byte)

## Sounds as arrays

- Samples are just stored one right after the other in the computer's memory (Like pixels in a picture)
- That's called an *array*
  - **It's an especially efficient (quickly accessed) memory structure**



## Working with sounds

- We'll use `pickAFile` and `makeSound`.
  - **We want .wav files**
- We'll use `getSamples` to get all the *sample objects* out of a sound
- We can also get the value at any index with `getSampleValueAt`
- Sounds also know their length (`getLength`) and their sampling rate (`getSamplingRate`)
- Can save sounds with `writeSoundTo(sound,"file.wav")`

## Working with Samples

- We can get sample objects out of a sound with **getSamples(sound)** or **getSampleObjectAt(sound, index)**
- A sample object remembers its sound, so if you change the sample object, the sound gets changed.
- Sample objects understand **getSample(sample)** and **setSample(sample, value)**