COMP4431
Multimedia Computing

Basic Digital Audio Concepts

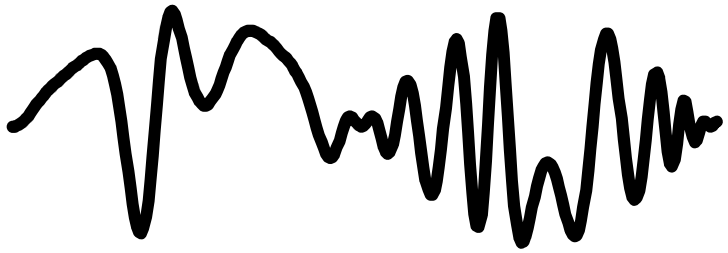
David Rossiter

This Presentation

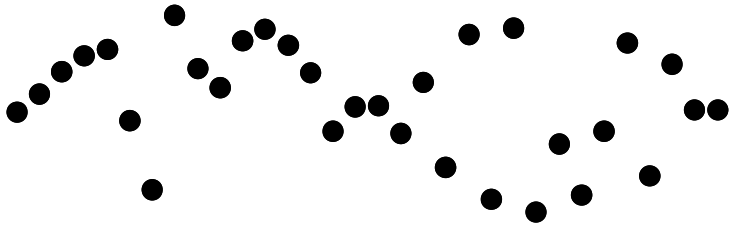
- Topics covered in this presentation:
  - Digital Audio
  - Human Hearing
  - Headphone Issues
  - Audio Sample Ranges
  - Clicks
  - Clipping
  - How Many Samples?
  - Amplitude Control

What is Digital Audio?

- We hear sounds because our ears detect the vibration of sound waves
- Digital audio means the digitalization of these sound waves, i.e. storing the sound waves as numbers



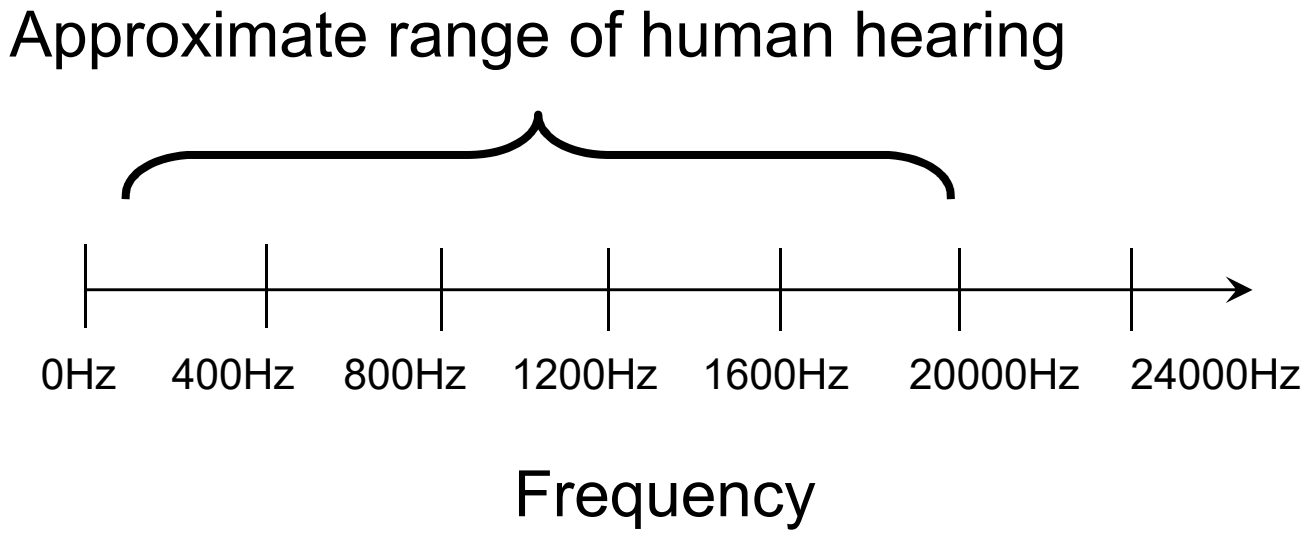
A Sound Wave



A Digitalized Sound Wave

What Can Humans Hear?

- Human can hear frequencies in the range 20Hz to 20,000Hz, roughly:



Be Careful of Your Headphones!

- Example situation - you develop a program to generate high frequency signals
- But when you listen to the output you can't hear it
- You spend hours checking your code…
- However, the only reason you couldn't hear any output was because the headphones/ speakers were physically unable to make those frequencies

Audio Value Ranges

- Audio has negative as well as positive values and the average value is zero
- So, if 1 byte is used to store one sample:
  - there are 256 different levels that can be used
  - the range is -128 to 127
- Similarly, if 2 bytes are used to store one sample:
  - there are 65,536 different levels that can be used
  - the range is -32,768 to 32,767
- The above use the **two's complement** representation, both negative and positive are OK

Web Audio API

- We will use the Web Audio API to do audio work
- This should be part of all modern browsers
- To be safe, we will stick with Chrome

Browser compatibility

Desktop | Mobile

Feature	Chrome	Edge	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)
Basic support	14 webkit	(Yes)	23	Not supported	15 webkit 22 (unprefixed)	6 webkit

Browser compatibility

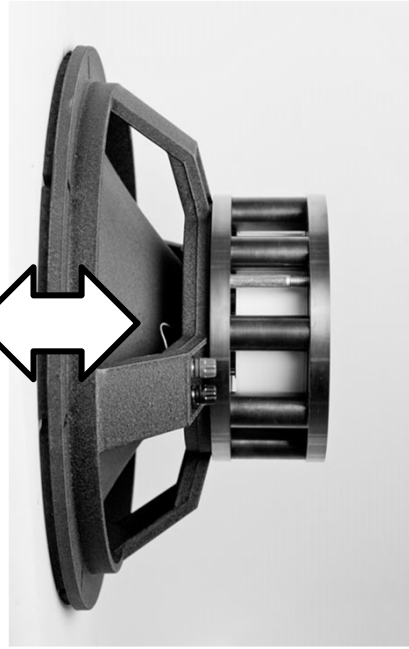
Desktop | Mobile

Feature	Android	Chrome	Firefox Mobile (Gecko)	Firefox OS	IE Phone	Opera Mobile	Safari Mobile
Basic support	Not supported	28 webkit	25	1.2	Not supported	Not supported	6 webkit

Audio Value Ranges

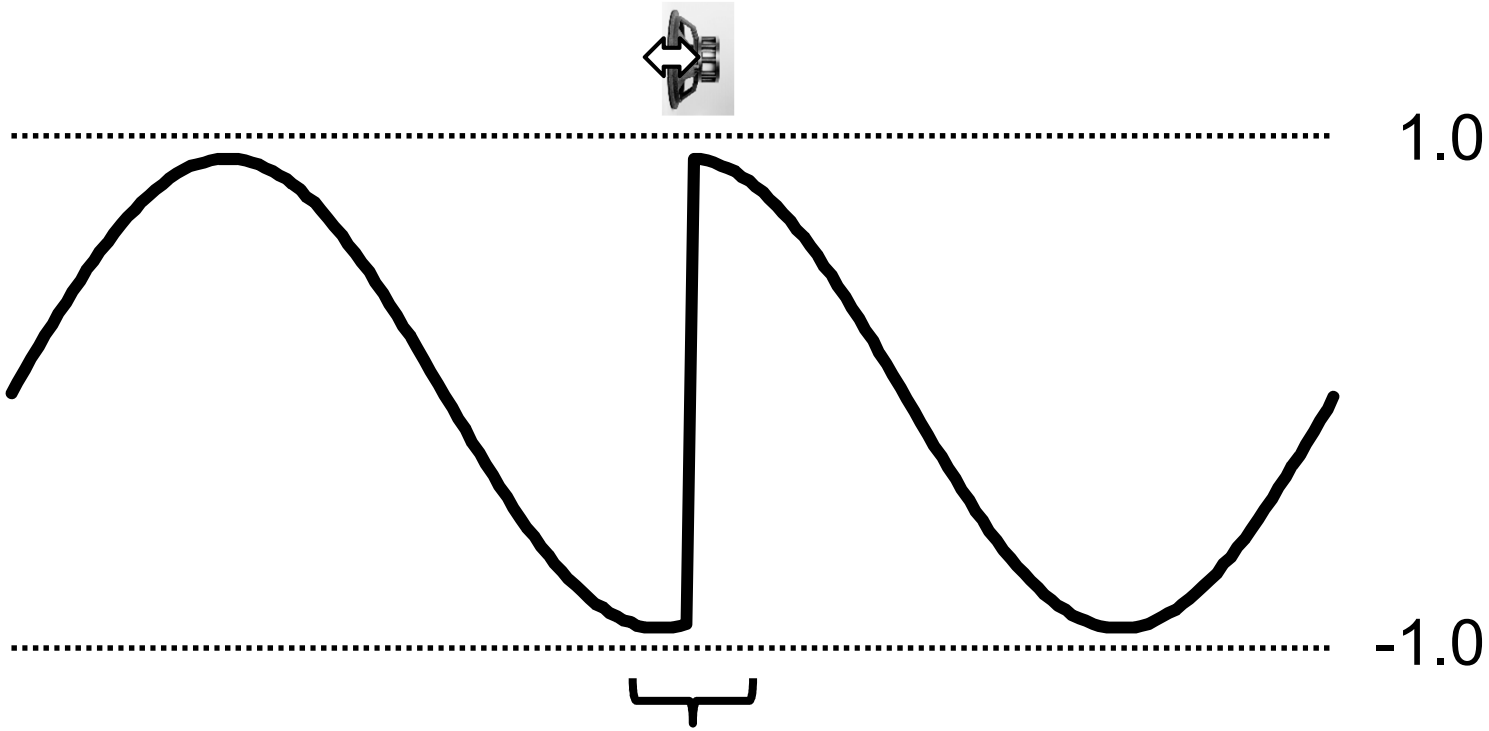
- Using the Web Audio API means that the nasty values like -128, 127, are ‘hidden’ from the programmer
- Instead, audio samples are stored as floats in the range of +1 to -1, which is easier to program
- However, when the program outputs sound, the float values are usually ‘secretly’ converted to 2 byte representation and then sent

Generating Clicks



- At some point you will meet the ‘click’ sounds
- Click sounds occur when the difference between one sample and the next one is large
- The speaker moves from one position to the second position in a very short amount of time
- When doing this it pushes or ‘pulls’ the air in front of the speaker extremely quickly, causing a rapid pulse of air
- This is then perceived as a ‘click’ sound

An Example Click Sound



When this sample sequence is played,
a click may be heard at this point

PCM Values and Clicks

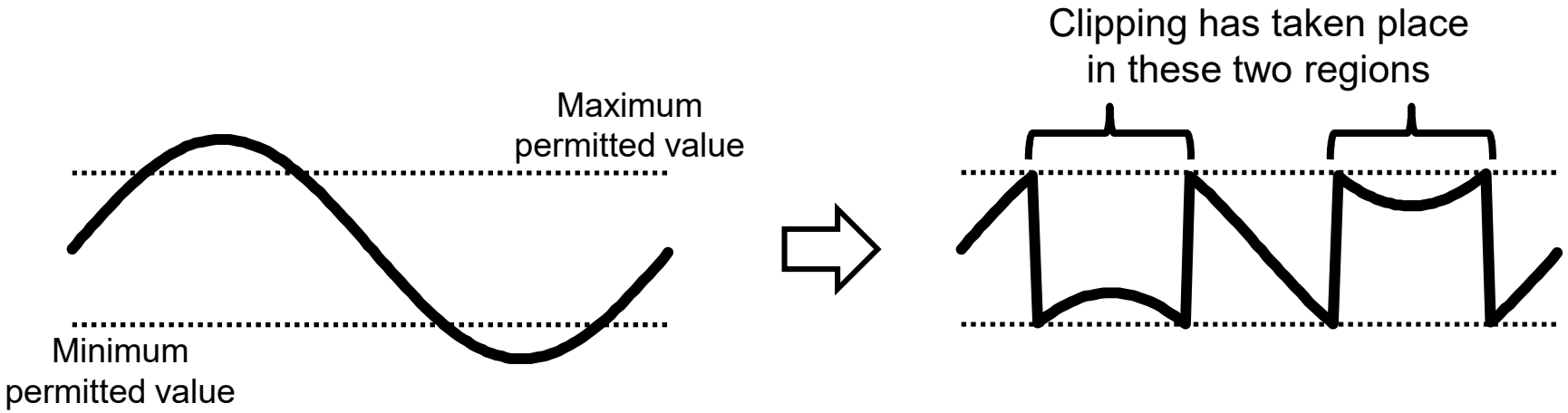
- PCM = **P**ulse **C**ode **M**odulation
- PCM is the way in which the samples are stored
- PCM method = ‘stored without any clever tricks’
- Examples of PCM sample sequences where you would probably hear a click are:

  . . . , -1.0, 1.0, . . .

  . . . , 1.0, -1.0, . . .
- It is the *relative* difference that causes the click

Clipping

- If the samples are processed so that their new value is out of the permitted level it is called *clipping*
- Here is an example of what might happen:

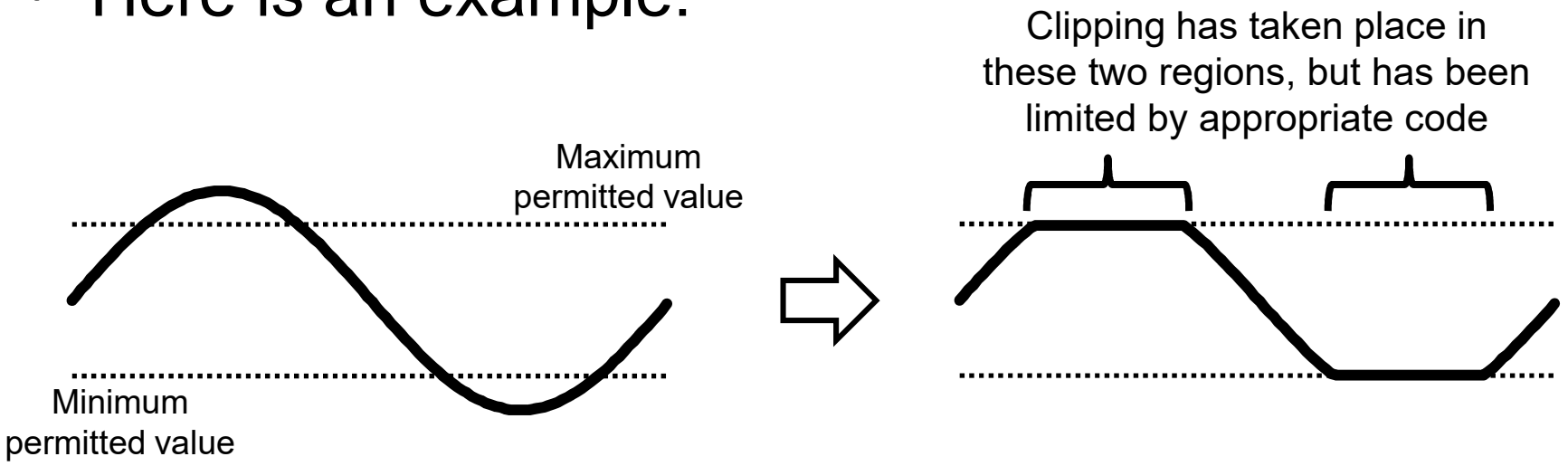


Audio waveform after some audio processing

Audio waveform after clipping

Handling Clipping

- You could add some code to control what happens when clipping takes place, to try to make the audio sequence more appropriate
- Here is an example:

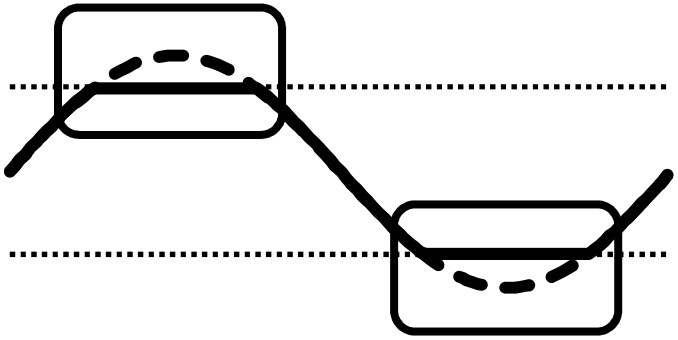


Audio waveform after some audio processing

Audio waveform after handling clipping

Avoiding Clipping

- Even if you handle clipping so that it is ‘more appropriate’, the audio is still ‘damaged’

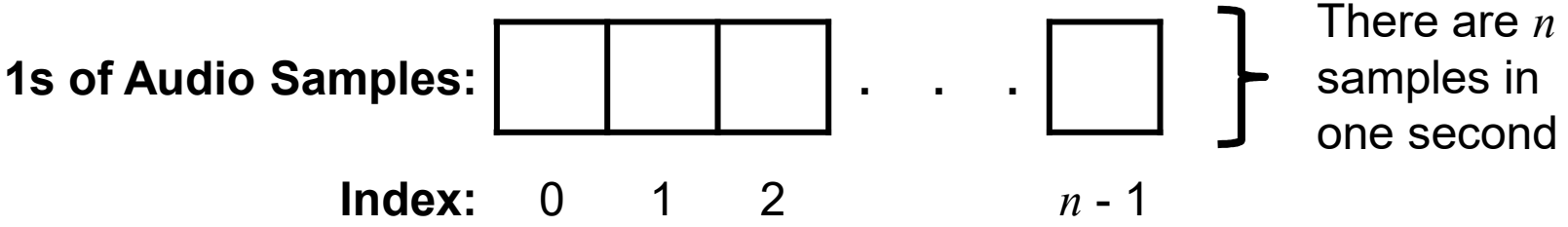


The audio will not be the same as the original

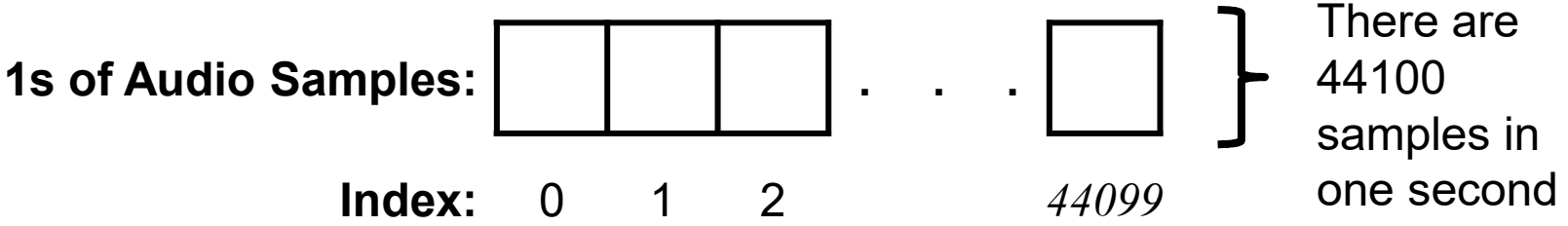
- So this approach is also not good
- It is much better to make sure you don’t get into this situation

How Many Samples?

- *Sample rate* is the number of samples for each second of audio
  - For example, if the sample rate is n:



  - For high quality audio we could use the same sample rate used by audio CDs, which is 44100 samples per second, like this:

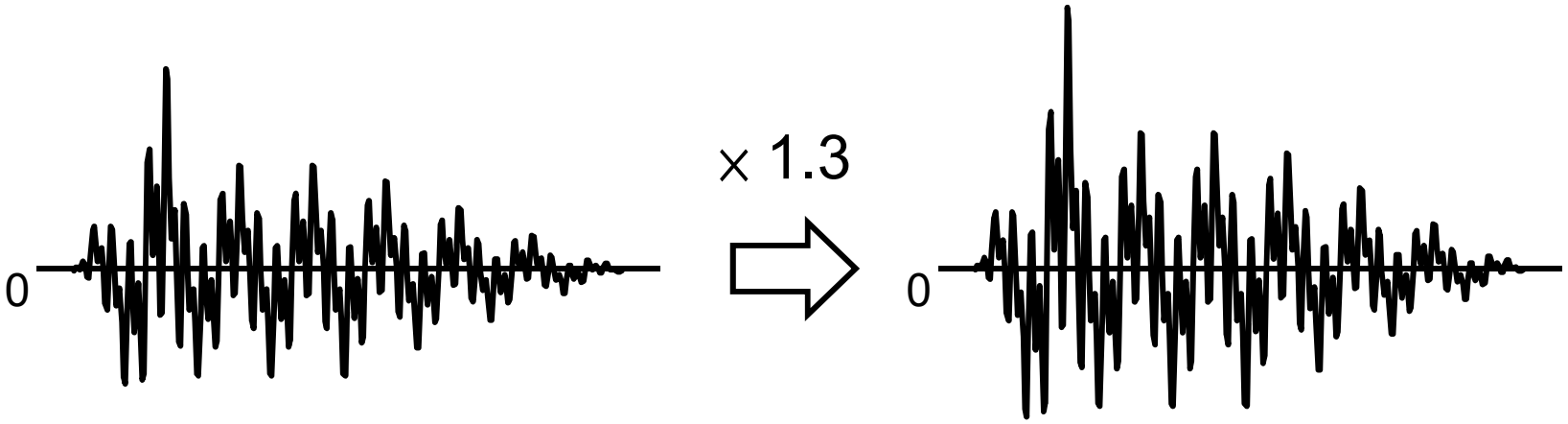


Amplitude Control

- *Amplitude* is the maximum of the absolute sample values, which affects the volume of the sound
- Amplitude control is one of the most basic and common audio processing operations
- For simple control of amplitude, samples are multiplied by a constant
  - To increase the volume, all the samples would be multiplied by a value >1
  - To decrease the volume, all the samples would be multiplied by a value <1

Amplitude Control Example

- For example, an increase in volume would be achieved by multiplying every sample by 1.3, as shown below:



This illustration assumes
no clipping has taken place

After Multiplying the Amplitude

- Both negative and positive numbers are treated equally, i.e. positive values stay positive, negative values stay negative
- In this way the general ‘shape’ of the digital waveform is the same as before, but it is now ‘stretched’ in a positive and negative direction
- The mean value will stay zero