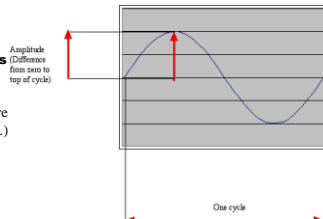


Chap. 2. How sound works:

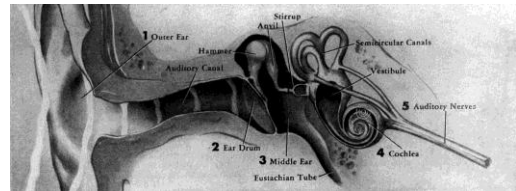
- Sounds are **waves of air pressure**

- 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**



HOW DOES HEARING WORK?

- × The outer ear “catches” sounds
- × The eardrum vibrates
- × The inner ear translates the vibrations to nerve impulses for the brain to interpret



Volume and pitch:

- 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?”

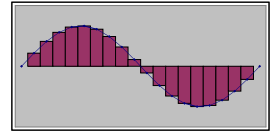
- It's strange, but our hearing works on *ratios* not *differences*, e.g., for pitch.
 - We hear the difference between 200 Hz and 400 Hz, as the same as 500 Hz and 1000 Hz
 - Similarly, 200 Hz to 600 Hz, and 1000 Hz to 3000 Hz
- Intensity (volume) is measured as *watts per meter squared*
 - A change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

Decibel is a logarithmic measure

- A *decibel* is a ratio between two intensities: $10 * \log_{10}(I_1/I_2)$
 - 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?



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

Sounds as arrays

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



Working with sounds

- We'll use `pickAFile` and `makeSound`.
 - `mySound = makeSound(pickAFile())`
 - Need to select `.wav` files
- We can also get the value at any index with `getSampleValueAt(mySound, index)`
- Sounds also know their length (`getLength`) and their sampling rate (`getSamplingRate`)
- Can save sounds with `writeSoundTo(mySound, "rC:\100\mediasources\xxx.wav")`

Demonstrating Working with Sound in JES

```
>>> myFile=pickAFile()
>>> print myFile
C:/100/mediasources/preamble.wav
>>> sound=makeSound(myFile)
>>> print sound
Sound of length 421109
>>> print getSampleValueAt(sound,1)
36
>>> print getSampleValueAt(sound,2)
29
```

Working with samples

```
>>> print getLength(sound)
220568
>>> print getSampleValueAt(sound,220568)
68
>>> print getSampleValueAt(sound,220570)
I wasn't able to do what you wanted.
The error java.lang.ArrayIndexOutOfBoundsException has occurred
Please check line 0 of
>>> print getSampleValueAt(sound,1)
36
>>> setSampleValueAt(sound,1,12)
>>> print getSampleValueAt(sound,1)
12
```

Another Example: Amplify by 2

```
>>> x = getSampleValueAt(sound,1)
>>> setSampleValueAt(sound,1,2*x)
>>> x = getSampleValueAt(sound,2)
>>> setSampleValueAt(sound,2,2*x)
.....
```

- But, there are thousands of these samples !
- Would be nice for a computer to *iterate* in a certain order
- => *loop*

Amplify sound by a twofold

```
>>> x = getSampleValueAt(sound, □)
>>> setSampleValueAt(sound, □, 2*x)
>>> x = getSampleValueAt(sound, □)
>>> setSampleValueAt(sound, □, 2*x)
.....
```

- Can we have □ step through 1, 2, 3, ..., 220568 ?
- => *for* □ *in* range(1:220570):

Try in Command Area

```
>>> for i in range(1,110000):
...   x = getSampleValueAt(sound, i)
...   setSampleValueAt(sound,i,2*x)
...
.....
```

Colon at the end of for loop

After "...", enter three blank spaces

- Remember....
 - A colon at the end of 'for' loop
 - Indent 2nd and 3rd lines

Write a Recipe (program)

- Frustrating to type in a set of commands
- Can we save a program, and re-use it ?
 - => **YES**
 - Write a program in Program Area
 - Click on 'Load' button
 - *Save it in C:\100 with xxx.py*
 - Enter a command to invoke the function

Write a Program

In Prog Area: `def incVol():`

```

sound = makeSound(pickAFile())
for i in range(1,110000):
    x = getSampleValueAt(sound, i)
    setSampleValueAt(sound,i,2*x)
return sound

```

Annotations: "block" labels with arrows pointing to the function definition and the for loop body.

Using it:

```

>>> s = incVol()
>>> play(s)
>>> writeSoundTo(s,"C:/mediasources/louder-g10.wav")

```

How did that work?

- When we call `incVol()`, the function `incVol` is executed (invoked)

```
>>> s=incVol()
```

```

def incVol():
    sound = makeSound(pickAFile())
    for i in range(1,110000):
        x = getSampleValueAt(sound, i)
        setSampleValueAt(sound,i,2*x)
    return sound

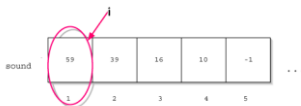
```

Starting the loop

- Select a wav file and name it `sound`
 - The `for` loop makes `i` be 1
- ```

def incVol():
 sound = makeSound(pickAFile())
 for i in range(1,110000):
 x = getSampleValueAt(sound, i)
 setSampleValueAt(sound,i,2*x)
 return sound

```



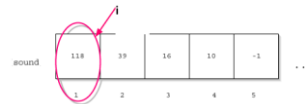
## Executing the block

- We get the value of the 1<sup>st</sup> sample and double its volume.

```

def incVol():
 sound = makeSound(pickAFile())
 for i in range(1,110000):
 x = getSampleValueAt(sound, i)
 setSampleValueAt(sound,i,2*x)
 return sound

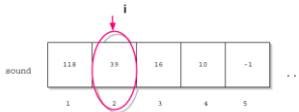
```



## Next sample

- Back to the top of the loop, and `i` will now be 2.

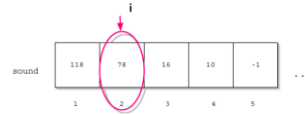
```
def incVol():
 sound = makeSound(pickAFile())
 for i in range(1,110000):
 x = getSampleValueAt(sound, i)
 setSampleValueAt(sound,i,2*x)
 return sound
```



## And increase that next sample

- Update the 2<sup>nd</sup> sample

```
def incVol():
 sound = makeSound(pickAFile())
 for i in range(1,110000):
 x = getSampleValueAt(sound, i)
 setSampleValueAt(sound,i,2*x)
 return sound
```



## And on through the sequence

- The loop keeps repeating until all the samples are doubled

## What are potential problems ?

```
def incVol():
 sound = makeSound(pickAFile())
 for i in range(1,110000):
 x = getSampleValueAt(sound, i)
 setSampleValueAt(sound,i,2*x)
 return sound
```

- 1.
- 2.

**Lab:**

- Write a program to decrease audio volumes by a half