

- Previous Lecture:
 - Review matrix, cell array, structure array
- Today's Lecture:
 - Working with sound files
 - Review vector, graphics, struct array, cell array
- Announcement:
 - P5 due 4/11 (tomorrow) at 3pm
 - Review session Sunday 1-2:30pm, location TBA
 - Prelim 3 Tuesday 7:30-9pm

Sampling Rate

Given human perception, 20000 samples/second is pretty good (20000Hz or 20kHz)

8,000 Hz	required for speech over the telephone
44,100 Hz	required for audio CD
192,400 Hz	required for HD-DVD audio tracks

April 10, 2008 Lecture 22 5

Resolution also affects the quality

Typically, each sampled value is encoded as an 8-bit integer in the .wav file.

Possible values: -128, -127, ..., -1, 0, 1, ..., 127

Loud: -120, 90, 122, etc.

Quiet: 3, 10, -5

Magnitude

determines loudness

16-bit used when very high quality is required.

April 10, 2008 Lecture 22 6

`wavread` converts the 8-bit values to floating point values between -1 and 1

```
[y,rate,nBits]= wavread('austin.wav')
```

```

0.4609
0.3516
0.2734
0.2891
0.2500
0.1484
0.1094
0.1641
0.1484
0.0000
-0.1641
-0.2734
-0.3281
    
```

← y(50000:50012)

wavread

```
[y,rate,nBits]= wavread('austin.wav');
n = length(y);
```

```
n =
    54453
rate =
    11025
nBits =
     8
```

austin.wav encoded the sound with 54,453 8-bit numbers that were gathered over a span of about 54453/11025 secs

April 10, 2008 Lecture 22 8

wavread

```
[data,rate,nBits]= wavread('noCry.wav')
```

Name of the source file

The vector of sampled sound values is assigned to this variable

The sampling rate is assigned to this variable

The resolution is assigned to this variable

April 10, 2008 Lecture 22 9

Hearing and "seeing" the sound

```
[y,rate]= wavread('austin');
sound(y, rate)
plot(1:length(y), y)
```

Usually playback at a rate equal to the sampling rate

Possible solution

```
playList = {'austin',...
            'sp_beam',...
            'sp_oz6'};

for k=1:length(playList)
    [y,rate] = wavread(playList{k});
    sound(y,rate)
end
```

Problem: will start playing sp_beam before austin finishes playing

Compute a pause!

```
for k=1:length(playList)
    [y,rate] = wavread(playList{k});
    sound(y,rate)

    % How long does it take
    % to play one file?

end
```

A: $\text{rate} \times \text{length}(y)$

B: $\text{rate} / \text{length}(y)$

C: $\text{length}(y) / \text{rate}$

Example: store the data from wav files as a struct array for play back later

```
function SA = wavSegments(wnames)
% Build a struct array SA such that
% SA(k).data stores the data of wnames{k}
% SA(k).rate stores the sampling rate of
% wav file wnames{k}

for k= 1:length(wnames)
    [y,rate] = wavread(wnames{k});
    SA(k)= struct('data', y, 'rate', rate);
end
```

```
function playSegments(SA)
% Play sound data stored in struct array SA.
% SA(k).data stores the k-th segment of
% sound data (from wavread)
% SA(k).rate is sampling rate of k-th seg.

for k= 1:length(SA)
    theData= SA(k).data;
    theRate= SA(k).rate;
    sound(theData,theRate)
    pause(length(theData)/theRate + 1)
end
```

Example

- An interactive graphical environment to visualize sound data
 - Allows user to zoom in and out
 - Allows user to look forward and back

[lookAtSound.m](#)

- As an exercise, add sound to the environment!