Sound, Part 3

Multiple Echoes

- Here is a recipe to create multiple echoes:

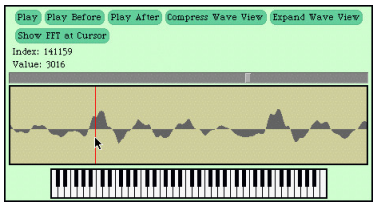
```
def echoes(sndfile, delay, num):
  s1 = makeSound(sndfile)
  ends1 = getLength(s1)
  ends2 = ends1 + (delay * num)
  #ends2 is in samples – convert to seconds and make empty sound that long
  s2 = makeEmptySound(1 + int(ends2 / getSamplingRate(s1)))

  echoAmplitude = 1.0
  for echoCount in range(1, num + 1):
    echoAmplitude = echoAmplitude * 0.6
    for pos1 in range(1, ends1):
      pos2 = pos1 + (delay * echoCount)
      value1 = getSampleValueAt(s1, pos1) * echoAmplitude
      value2 = getSampleValueAt(s2, pos2)
      setSampleValueAt(s2, pos2, value1 + value2)
  play(s2)
  return s2
```

Splicing Sounds

- Say we want to splice pieces of speech together extracting words **from the same** sound file:
  - Find the end points of words
  - Copy the samples into the right places to make the words come out as we want them
  - (We can also change the volume of the words as we move them, to increase or decrease emphasis and make it sound more natural.)

Finding the word endpoints

- Using *MediaTools* and play before/after cursor, can figure out the index numbers where each word ends



Word	Ending index
We	15730
the	17407
People	26726
of	32131
the	33413
United	40052
States	55510

Change to:
"We the United People of the United States…"

Now, it's all about copying

- We have to keep track of the source and target indices, **srcIndex** and **destIndex**

```
destIndex = Where-the-incoming-sound-should-start
for srcIndex in range(startingPoint, endingPoint):
    sampleValue = getSampleValueAt(source, srcIndex)
    setSampleValueAt(dest, destIndex, sampleValue)
    destIndex = destIndex + 1
```

How to do it

- First, set up a source and target.
- Next, we copy "United" (samples 33414 to 40052) after "We the" (sample 17408)
  - That means that we end up at 17408+(40052-33414) = 17408+6638=24046
  - Where does "People" start?
- Next, we copy "People" (17408 to 26726) immediately afterward
  - 24047 + (26726-17408) = 33365
  - Do we have to copy "of" to?
  - Or is there a pause in there that we can make use of?
- Finally, we insert a little (1/441-th of a second) of space – 0's

Word	Ending index
We	15730
the	17407
People	26726
of	32131
the	33413
United	40052
States	55510

The Whole Splice

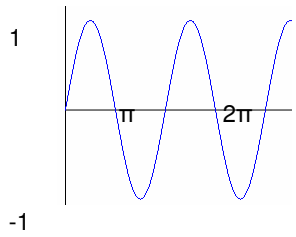
```
def splicePreamble():
  file = "/Users/sweat/1315/MediaSources/preamble10.wav"
  source = makeSound(file)
  dest = makeSound(file)   # This will be the newly spliced sound
  destSample = 17408         # targetIndex starts after "We the" in the new sound
  for srcSample in range(33414, 40052):  # Where the word "United" is in the sound
    setSampleValueAt(dest, destSample,  getSampleValueAt( source, srcSample))
    destSample = destSample + 1
  for srcSample in range(17408, 26726):   # Where the word "People" is in the sound
    setSampleValueAt(dest, destSample, getSampleValueAt( source,  srcSample))
    destSample = destSample + 1
  for index in range(1, 1000):                  #Stick some quiet space after that
    setSampleValueAt(dest,  destSample, 0)
    destSample = destSample + 1
  play(dest)                                    #Let's hear and return the result
  return dest
```

What if we didn't do that second copy? Or the pause?

```
def splicePreamble():
  file = "/Users/sweat/1315/MediaSources/preamble10.wav"
  source = makeSound(file)
  dest = makeSound(file)   # This will be the newly spliced sound
  destSample = 17408         # targetIndex starts after "We the" in the new sound
  for srcSample in range(33414, 40052):  # Where the word "United" is in the sound
    setSampleValueAt(dest, destSample,  getSampleValueAt( source, srcSample))
    destSample = destSample + 1
  #for srcSample in range(17408, 26726):   # Where the word "People" is in the sound
  #setSampleValueAt(dest, destSample, getSampleValueAt( source,  srcSample))
    #destSample = destSample + 1
  #for index in range(1, 1000):                  #Stick some quiet space after that
    #setSampleValueAt(dest,  destSample, 0)
    #destSample = destSample + 1
  play(dest)                                    #Let's hear and return the result
  return dest
```

Making Sine Waves

- We can build our own waves using trigonometry functions like sine or cosine



Let's say that we want the sound to play at 440 hz ; that is one cycle in 1/440 sec = 0.00227 seconds

If our sampling rate is 22,050 samples per second, then for one cycle we need: 0.00227 (seconds) * 22050 (samples/second) = 50 samples

Get a sample from the sine wave every (2 * pi) / 50 and repeat

Generating a Sine Wave

- To build the sine wave
  - Compute sample interval from the sine function
    - samplesPerCycle = (1/freq) * samplingRate
    - samplingInterval = (2 * pi) / samplesPerCycle
  - Generate blank sound of desired length
  - Loop through each sample of the blank sound
    - Set it to the next sample from the sine wave

Sine Wave Function

```
def sineWave(frequency, amplitude, duration):
  snd = makeEmptySound(duration)

  interval = 1.0 / frequency
  samplesPerCycle = interval * getSamplingRate(snd)
  samplingInterval = (2 * 3.14159) / samplesPerCycle

  sampleValue = 0
  for pos in range (1, getLength(snd) + 1):
    rawSample = sin(sampleValue)
    sampleVal = int(amplitude * rawSample)
    setSampleValueAt(snd, pos, sampleVal)
    sampleValue = sampleValue + samplingInterval
  play(snd)
  return snd
```

Did it work?

- Test with different frequencies, durations, amplitudes
- Save to a file and examine with media tools
  - writeSoundTo(snd, “filename.wav”)
- Getting fancier
  - Can add sine waves together, just like we did with .wav files, to generate chords and more interesting sounds
- Early computers used sine waves, square waves, and triangle waves to make sound

MP3

- Today, many audio files are stored using the MP3 format
- Data is compressed so it requires less space
- Lossless compression
  - Instead of storing each sample, what if we only stored the difference from the last sample?
  - The difference is usually much smaller than 32767 to -32768. It might only be +/- 100, which would require fewer bits to store
- Lossy compression
  - Throws away some of the sound, especially at higher frequencies, that you can't hear
    - E.g. soft sound simultaneously played with a loud sound
- WAV files also compressed, using a lossless compression technique

MIDI

- Musical Instrument Digital Interface
  - Standard for synthesizers so different musical hardware can interoperate with a computer
  - Can specify notes and instruments
  - JES has a built-in MIDI player using a piano
- The musical scale, starting at middle C, proceeds as follows:

  C
  C sharp
  D
  D sharp
  E
  F
  F sharp
  G
  A flat
  A
  B …

MIDI

- The MIDI format assigns a numerical value to each note. The table below shows some notes and their numeric values.

Musical Note	Numeric Value
C (middle C)	60
C Sharp	61
D	62
D Sharp	63
E	64
F	65
F sharp	66
G	67
A flat	68
A	69
...	...

playNote

- The playNote function is:
  - playNote(note, duration, velocity)

Note - This is the numeric value corresponding to the note using the table above. The value can actually range between 0 and 127, so you can use values below 60 for lower octaves below Middle C, and larger values for higher octaves.

Duration. The duration is how long to play the note in milliseconds. A duration of 1000 would play the note for one second, while a duration of 250 would play the note for a quarter of a second.

Volume or Velocity. In a metaphor to a piano, the velocity is how hard you hit the key and thus corresponds to the volume of the note. A velocity of 0 is silent while the highest velocity is 127.

Sample Programs

```
def playScale():
  for note in range(60, 71):
    playNote(note, 1000, 127)

def playSong():
  playNote(60, 500, 127)  # C
  playNote(60, 500, 127)  # C
  playNote(67, 500, 127)  # G
  playNote(67, 500, 127)  # G
  playNote(69, 500, 127)  # A
  playNote(69, 500, 127)  # A
  playNote(67, 1000, 127) # G
  playNote(65, 500, 127)  # F
  playNote(65, 500, 127)  # F
  playNote(64, 500, 127)  # E
  playNote(64, 500, 127)  # E
  playNote(62, 500, 127) # D
  playNote(62, 500, 127) # D
  playNote(60, 1000, 127) # C
```