Doubling the amplitude

def double( sound ) :
    for sample in getSamples(sound):
        value = getSample(sample)
        setSample(sample, value * 2)
Normalizing

A few ways to think about “normalizing”:

- use the whole enchilada (don’t waste any bits...)
- make everything use the same scale (0 to 100%)
- need 2 loops -- one to find largest and one to reset

```python
def normalize( sound ) :
    largest = 0
    for sample in getSamples(sound):
        largest = max( largest, getSample(sample) )
    multiplier = 32767.0 / largest
    print “Largest”, largest, “multiplier is”, multiplier
    for sample in getSamples(sound):
        setSample(sample, getSample(sample) * multiplier)
```
def normalize(sound):
    largest = 0
    for sample in getSamples(sound):
        largest = max(largest, abs(getSample(sample)))
    if largest > 32766:
        return sound
    multiplier = 32768.0 / largest
    print "Largest", largest, "multiplier is", multiplier
    for sample in getSamples(sound):
        setSample(sample, getSample(sample) * multiplier)
    return sound
def sineWave( freq, amplitude ) :
    mySound = getMediaPath("sec1silence.wav")
    buildSin = makeSound(mySound)
    sr = getSamplingRate(buildSin)  # sampling rate
    interval = 1.0 / freq  # interval of sample
    samplesPerCycle = interval * sr  # samples / cycle
    maxCycle = 2 * pi
    for pos in range(1, getLength(buildSin) + 1):
        rawSample = sin((pos / samplesPerCycle) * maxCycle)
        sampleVal = int(amplitude * rawSample)
        setSampleValueAt(buildSin, pos, sampleVal)
    return buildSin
Square wave

recipe 72

def squareWave( freq, amplitude ) :
    mySound = getMediaPath(“sec1silence.wav”)  
    square = makeSound(mySound)
    samplingRate = getSamplingRate(square)  # sampling rate
    seconds = 1  
    interval = 1.0 * seconds / freq  # interval of sample
    samplesPerCycle = interval * samplingRate  # samples / cycle
    samplesPerHalfCycle = int(samplesPerCycle / 2)
    sampleVal = amplitude
    i = 1
    for s in range( 1, getLength( square ) + 1 ) :
        if (i > samplesPerHalfCycle):
            sampleVal = sampleVal * -1
            i = 0
            setValueAt( square,s, sampleVal )
        i = i + 1
    return square


**Triangular wave**

*recipe 73, modified*

```python
def triangleWave( freq ):
    amplitude = 6000
    samplingRate = 22050  # sampling rate
    seconds = 1
    triangle = makeEmptySound( seconds )  # create a sound object (the book uses “sec1silence.wav”)
    interval = 1.0 * seconds / freq  # interval of sample
    samplesPerCycle = interval * samplingRate  # samples / cycle
    samplesPerHalfCycle = int(samplesPerCycle / 2)
    increment = int( amplitude / samplesPerHalfCycle )
    sampleVal = -amplitude
    i = 1
    for s in range( 1, samplingRate + 1 ) :
        if (i > samplesPerHalfCycle):
            increment = increment * -1
            i = 0
        sampleVal = sampleVal + increment
        setSampleValueAt( triangle, s, sampleVal )
        i = i + 1
    return triangle  # return the sound (the book says play)
```
Adding

- recipe 71 (part 2)

```python
def addSounds( sound1, sound2 ) :
    for index in range( 1, getLength(sound1) + 1 ) :
        s1Sample = getSampleValueAt( sound1, index )
        s2Sample = getSampleValueAt( sound2, index )
        setSampleValueAt( sound2, index, s1Sample + s2Sample )
    return sound2
```
def echoes( soundFile, delay, num ):
    s1 = makeSound( soundFile )
    ends1 = getLength( s1 )
    ends2 = ends1 + (delay - num)
    s2 = makeEmptySound(1 + int( ends2 / getSamplingRate(s1) ) )
    echoAmplitude = 1.0
    for echoCount in range( 1, num + 1 ) :
        echoAmplitude = echoAmplitude * 0.6  # each echo is 60% of previous
        for posn1 in range( 1, ends1 ) :
            posn2 = posn1 + (delay* echoCount )
            values1 = getSampleValueAt( s1, posn1 ) * echoAmplitude
            values2 = getSampleValueAt( s2, posn2 )
            setSampleValueAt( s2, posn2, values1 + values2 )
    return s2
Shifting the frequency

- recipe 68, modified
- how sampling keyboards work....

```python
def shift( soundFile, factor ) :
    source = makeSound( soundFile )
    target = makeSound( soundFile )
    sourceIndex = 1
    sourceLength = getLength( source )

    for targetIndex in range( 1, sourceLength + 1 ) :
        setValueAt( target, targetIndex, getValueAt( source, int( sourceIndex )))
        sourceIndex = sourceIndex + factor
        if sourceIndex > sourceLength :
            sourceIndex = 1

    return target
```
Exquisite Corpse 2: sound

- What should we do?
- Need to have everyone contribute a sound
- What rules should we create?
- Consider our experience with the visual exquisite corpse.
- We’ll decide on Friday.