

CSCI 128 Lab 10

Chris Greencorn

Lab 10

1. Write a function to create a cos sound wave.

```
def cosWave(freq,amplitude):
    buildCos = makeEmptySoundBySeconds(2)
    sr = getSamplingRate(buildCos)
    interval = 1.0/ freq
    samplesPerCycle = interval * sr
    maxCycle = 2 * pi
    for pos in range (0, getLength(buildCos)):
        rawSample = cos((pos/samplesPerCycle) * maxCycle)
        sampleVal = int(amplitude*rawSample)
        setSampleValueAt(buildCos,pos,sampleVal)
    play(buildCos)
```

2. Write a function to create that combines square wave and triangle wave.

```
def SquareAndTriangleWave(freq,amplitude,seconds):
```

```
#### SQUARE WAVE
```

```
square = makeEmptySoundBySeconds(seconds)
sqSamplingRate = getSamplingRate(square)
interval = 1.0 * seconds / freq
samplesPerCycle = interval * sqSamplingRate
samplesPerHalfCycle = int(samplesPerCycle / 2)
sampleVal = int(amplitude)
s = 1
i = 1
for s in range(0, getLength(square)):
    # if end of a half-cycle
    if (i > samplesPerHalfCycle):
        # reverse the amplitude every half-cycle
        sampleVal = sampleVal * -1
        # and reinitialize the half-cycle counter
        i = 0
    setSampleValueAt(square,s,sampleVal)
    i = i + 1
```

```
## TRIANGLE WAVE
```

```
triangle = makeEmptySoundBySeconds(seconds)
triSamplingRate = getSamplingRate(triangle)
interval = 1.0 * seconds / freq
samplesPerCycle = interval * triSamplingRate
samplesPerHalfCycle = int(samplesPerCycle / 2)
increment = int(amplitude/samplesPerHalfCycle)
sampleVal = -(amplitude)
i = 1
```

```

for s in range(1, getLength(triangle):
    # if end of a half-cycle
    if (i > samplesPerHalfCycle):
        # reverse the amplitude every half-cycle
        increment = increment * -1
        # and reinitialize the half-cycle counter
        i = 0
    sampleVal= sampleVal + increment
    setSampleValueAt(triangle,s,sampleVal)
    i = i + 1

## BLEND WAVES
squareAndTriangle = makeEmptySoundBySeconds(seconds)
for index in range(0, getLength(squareAndTriangle)):
    sqSample = getSampleValueAt(square, index)
    triSample = getSampleValueAt(triangle, index)
    setSampleValueAt(squareAndTriangle, index, sqSample + triSample)

play(squareAndTriangle)

```