

ITEC 120

Lecture 18
Sound

Review

- Arrays
 - Capabilities
 - Sort
 - Grow / Shrink

Sound

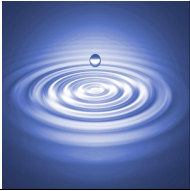
Objectives

- See a real world usage of arrays
 - Sound

Sound

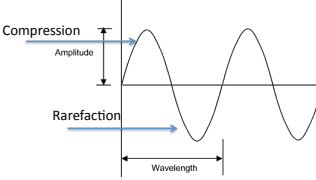
Sound

- Pressure in the air
- Sounds are made up of rises and falls in pressure
 - Compressions / rarefactions
- Waves of sound data
 - Frequency
 - Amplitude
 - Shape



Sound

Waves



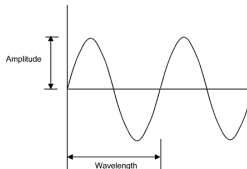
The diagram shows a sine wave on a horizontal axis. A vertical double-headed arrow indicates the height from the center line to the peak, labeled 'Amplitude'. A horizontal double-headed arrow indicates the distance between two consecutive peaks, labeled 'Wavelength'. The top of the wave is labeled 'Compression' and the bottom is labeled 'Rarefaction'.

- Simple wave
 - Sine wave
- Cycle
 - One compression and rarefaction
- Amplitude
 - Distance from 0 point to greatest pressure
 - Volume
 - Formula for converting to decibels (db)

Sound

Waves

- Pitch
 - How often the cycle occurs
 - Longer= lower
 - Shorter= higher
- Chipmunk effect
- Real sound
 - Several waves combined
 - Sine waves don't work



The diagram shows a sine wave on a horizontal axis. A vertical double-headed arrow indicates the height from the center line to the peak, labeled 'Amplitude'. A horizontal double-headed arrow indicates the distance between two consecutive peaks, labeled 'Wavelength'.

Sound

Computers

hz = cycles per second

- Analog (real world) / Digital (0s and 1s)
 - Pick a ceiling / floor for amplitudes
 - How many cycles a second are there?
- Record X times a second (sampling)
 - CD quality = 44.1khz recording means 44,100 samples per second
 - 44,100 ints per second (10 megs a minute)
 - CDs hold 650 MB or ~70 minutes of audio

Sound

Code

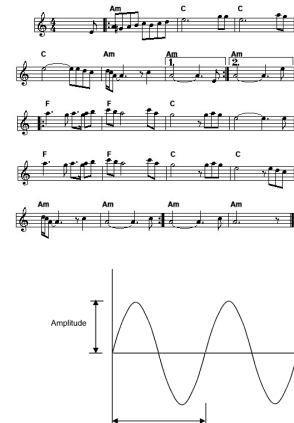
- Java has lots of APIs for sounds
 - Custom function container that is simplified

```
int sampleRate=22050; //Use for all your code
int duration=1; // How long to play the sound (seconds)
int numSamples = sampleRate*duration;
Sound mySound = new Sound(duration*sampleRate);
//To access the array length
int numPlaces = mySound.getLength();
//To set a sound value
mySound.setSampleValueAt(0, 440);
//To play the sound
mySound.play();
//To play and make sure it is unique
mySound.blockingPlay();
```

Sound

Problem

- People are used to
- Computers deal with




Sound

Transformation

- Each note is mapped to a specific frequency
- Multiply by the 12th root of 2 to goto the next note

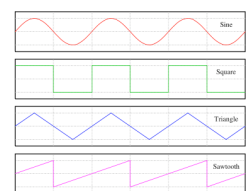
— $261.626 * 2^{(1/12)} = 277.183 = C\#$



Sound

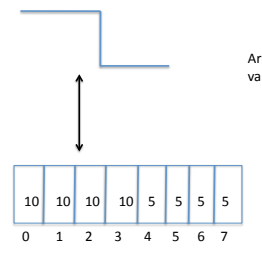
Problem

- Just knowing the frequency isn't enough
 - Compression / rarefaction
- Need to generate a wave
 - Sound different



Sound

Array tie in



Array holds the sampled values for the wave

This array is small compared to a regular CD.

For example, a crumb compared to the 16ft party sub that is an audio CD

Sound

Square

- Need both rarefaction and compression
- How many array values for the top / bottom
- Cycles in a sample (frequency)
- Half a cycle on top
- Half on bottom

Sound

Play A4 for 1 second

Code

```
int SamplingRate = 22050;
double amp = 4000;
Sound example = new Sound((int)(SamplingRate*1));
double interval = 1.0/440;
double samplesPerCycle = interval*SamplingRate;
int halfCycle = (int)samplesPerCycle/2;
double value=440;
int num=1;
for (int i=0; i<example.getLength(); i++)
{
    if (num > halfCycle)
    {
        num=0;
        value*=-1;
    }
    num++;
    double raw = value * amp;
    example.setSampleValueAt(i, (int)raw);
}
```

Sound

Triangle

- Shape analysis
- Start at amplitude, add increment until half cycle
- Decrement for other half

Sound

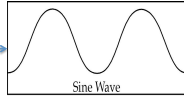
Code

```
int SamplingRate = 22050;
double amp = 4000;
Sound example = new Sound((int)(SamplingRate*duration));
double interval = 1.0/getFreq(name);
double samplesPerCycle = interval*SamplingRate;
int halfCycle = (int)samplesPerCycle/2;
double value= getFreq(name);
double increment = amp / samplesPerCycle;
int num=1;
for (int i=0; i<example.getLength(); i++)
{
    if (num > halfCycle)
    {
        num=0; increment*=-1; }
    num++;
    value+=increment;
    example.setSampleValueAt(i, (int)value);
}
```

Sound

Do the wave

- Sine wave
 - `Math.sin(value); +` for loop
- Figure out duration (1 second)
- Determine interval (space between samples)
 - $1/\text{frequency}$
- Figure out how many samples per cycle
 - $\text{Interval} * \text{Sampling Rate}$
- Feed `Math.sin` a value for each sample



Sound

Code

Creates the note A4 for 1 second

```
int SamplingRate = 22050;
double amp = 4000; //For volume
Sound example = new Sound((int)(SamplingRate*1));
double interval = 1.0/440;
double samplesPerCycle = interval*SamplingRate;
double maxCycle = 2*3.14159;
for (int i=0; i<example.getLength(); i++)
{
    double raw = Math.sin( (i/samplesPerCycle) *maxCycle);
    raw = raw * amp;
    example.setSampleValueAt(i, (int)raw);
}
```

Note: Values will always range from 0 to 2Pi
The base sine wave is then adjusted for volume (multiplied by the amplitude)

Sound

Demo

- Process
 - Note->Frequency->Sample
- Program
 - Creates X sample notes
 - Plays based on what is typed in
 - Simulator loop

Sound

Auto-tune

- What is it?
- Example
- How?
 - Examine frequency
 - Increase or decrement to right frequency
- Makes horrible singers sound like pop singers...

Sound

Summary

- Music on computers
 - Arrays are crucial
- Note->frequency
- Shape of wave
 - Square
 - Triangle
 - Sine

Sound