Intro to computer music with Processing

Charles Martin
September 2014

Charles Martin (me)

things I do:
- HCI (w/ Henry Gardner)
- computer music
- NIME
- percussion

music:
- soundcloud.com/charlesmartin
- charlesmartin.bandcamp.com
- charlesmartin.com.au

why care about sound?

why care about sound?

- sound **demands** attention
  - think of banner ads compared to YouTube ads
  - easy to “filter out” visual things, very difficult to “filter out” sound.

why care about sound?

- interactive sound contributes to an illusion of reality
  - the click on the (original) iPod click wheel
  - “tap” sound on iPhone keyboards (fake haptics!)
  - audio cues help to communicate affordances

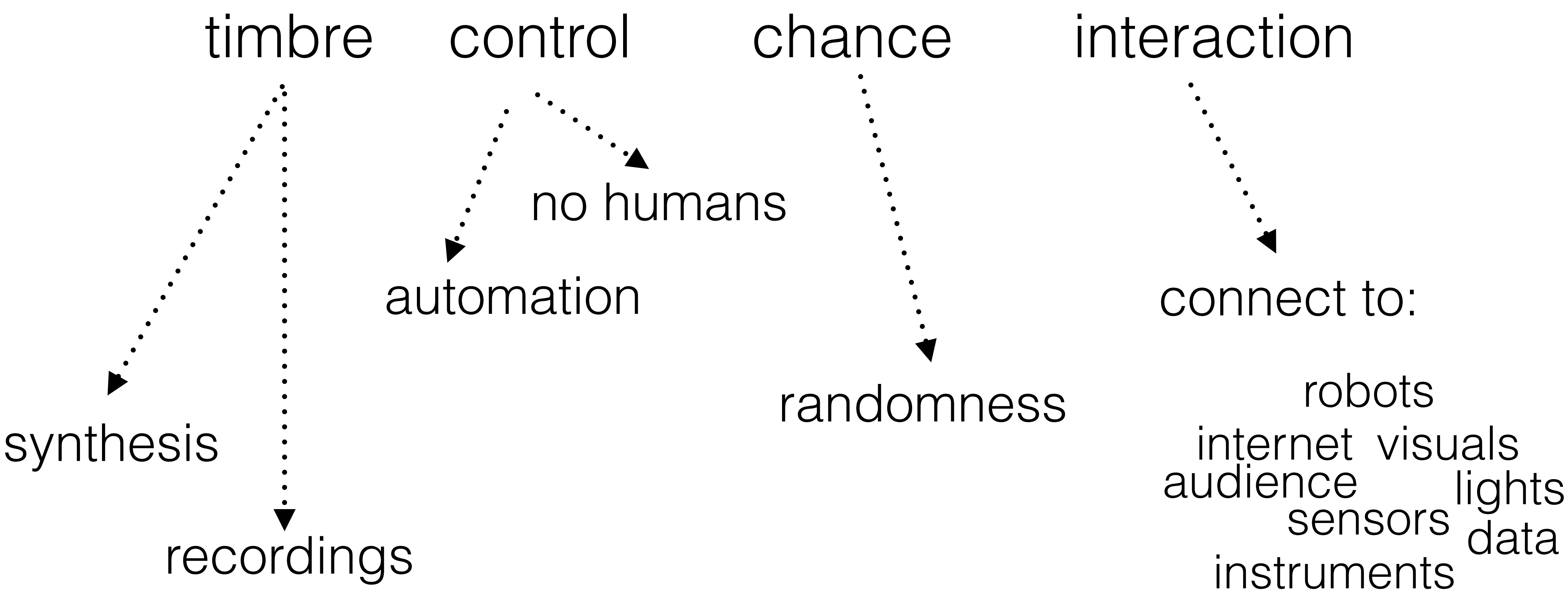
sound programming is a bit different than graphics!

- Graphics - frame rate is 25-60Hz
- Sound - frame rate is 44100-96000Hz
- No `draw()` loop to directly program frames of sound (it would be too slow).
- You connect low level sound generation objects together (Unit Generators - **UGens**).

minim - Processing Library for Sound

- minim is a Processing Library for making sound and music
- included in the current Processing distribution!
- there’s lots of other options in the Java world
- also lots of other environments more focussed on sound!

aspects of computer music?



timbre

synthesis

recordings

control

no humans

automation

chance

randomness

interaction

connect to:

robots

internet visuals

audience

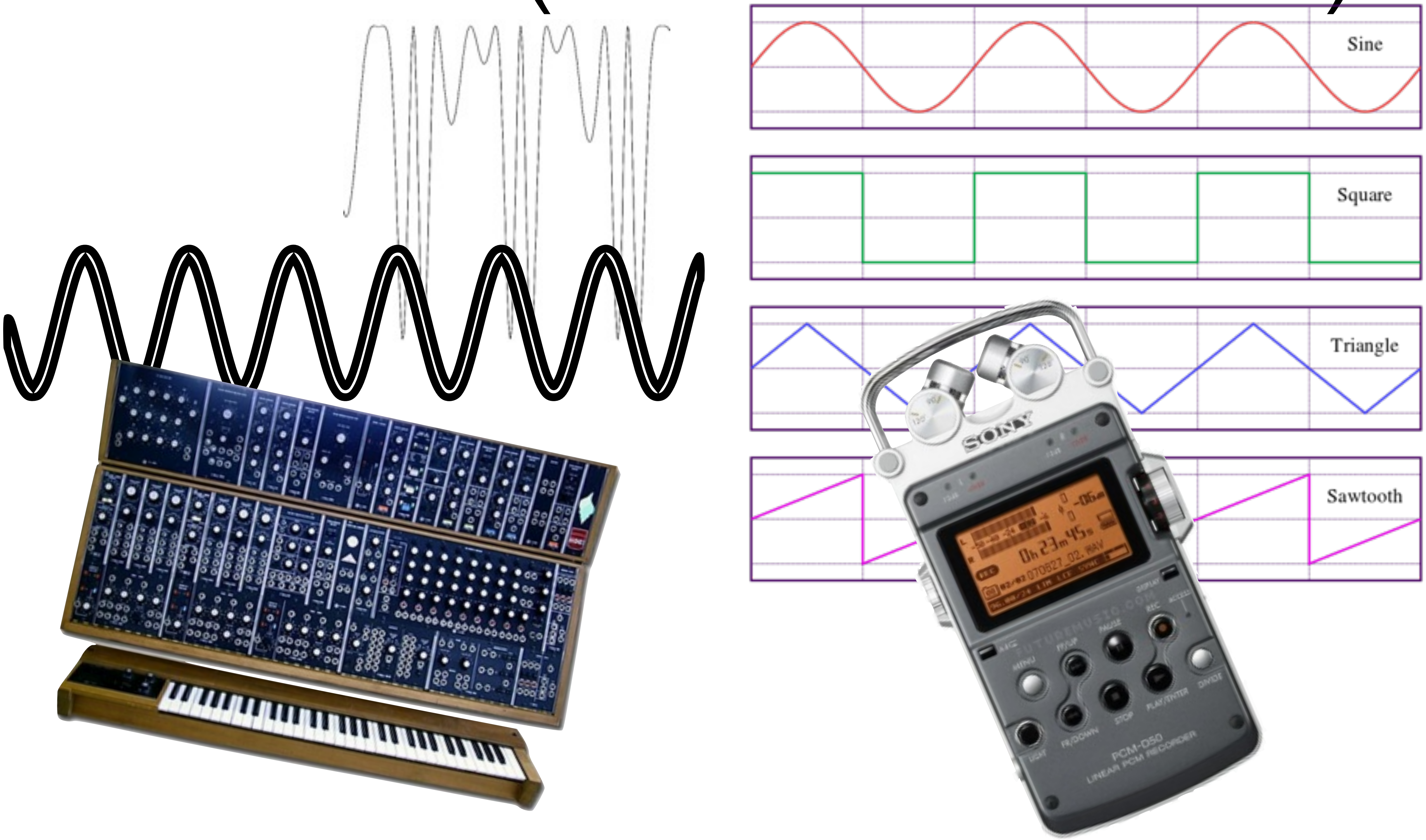
lights

sensors

data

instruments

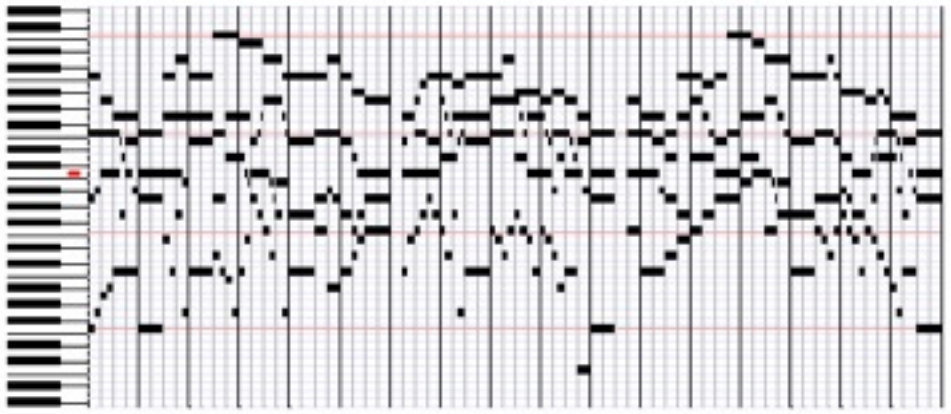
Timbre (Sound Colour)



synthesised sounds

recorded sounds

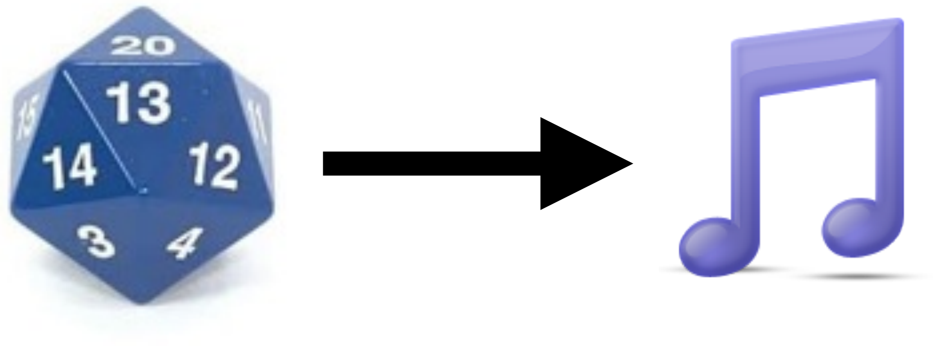
Control



playing notes (typical musical control)

connecting visual to sound

Chance



control sounds with a “random process”

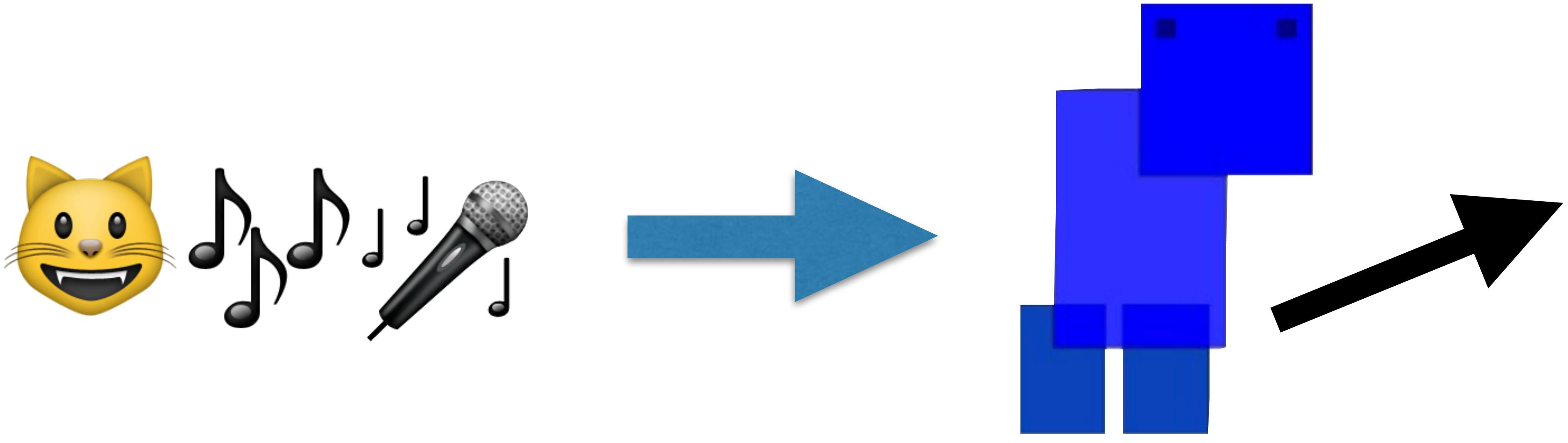
```
while (true) {
   playNote();
   delay(random(10000)+5000);
}
```

play a note

wait between 5s and 15s

repeat!

Interaction



- connect sound input to action on the screen
- connect keyboard/mouse to sounds
- connect camera to sounds

Don’t (over)-use others’ work without permission!

- It’s rude and probably a breach of copyright!
- Using an extract can be considered “Fair Dealing” - this still only means 10% of a song!
- This course is about *making your own interactive art works!*
- Make sure you have an interesting interactive/ artistic contribution the sonic parts of your project!

Examples:

- Synthesis:
  1. “Hello World” - sine tone
  2. Additive Synthesis sine Tones
  3. FM Synthesis
- Recordings:
  4. Playing back a sound file
  5. Performing with a sound file
- Control
  6. Envelope Generator
- Not-Control
  7. Random Values
  8. Composing with Randomness
- Interaction
  9. Interacting with a Processing Sketch
  10. Interacting with Sound Input?

1. Hello Soundworld!

```
import ddf.minim.*;
import ddf.minim.ugens.*;

Minim minim;
AudioOutput out;
Oscil wave;

void setup() {
  size(200, 200);
  minim = new Minim(this);
  out = minim.getLineOut();
  wave = new Oscil(440, 0.5f, Waves.SINE);
  wave.patch(out);
}

void draw() {}
```

- Playing a sine tone is the “Hello World” of computer music!

What’s a sine tone?

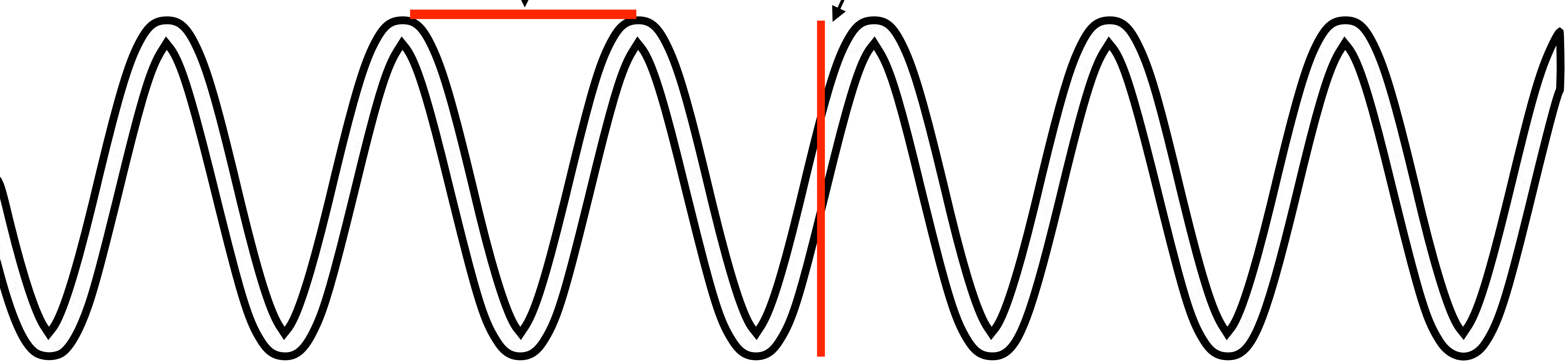
```
wave = new Oscil(440, 0.5f, Waves.SINE);
```

(short for “oscillator”)

frequency (pitch)

amplitude (volume)

type of wave (timbre)



Connecting sounds together

```
wave.patch(out);
```

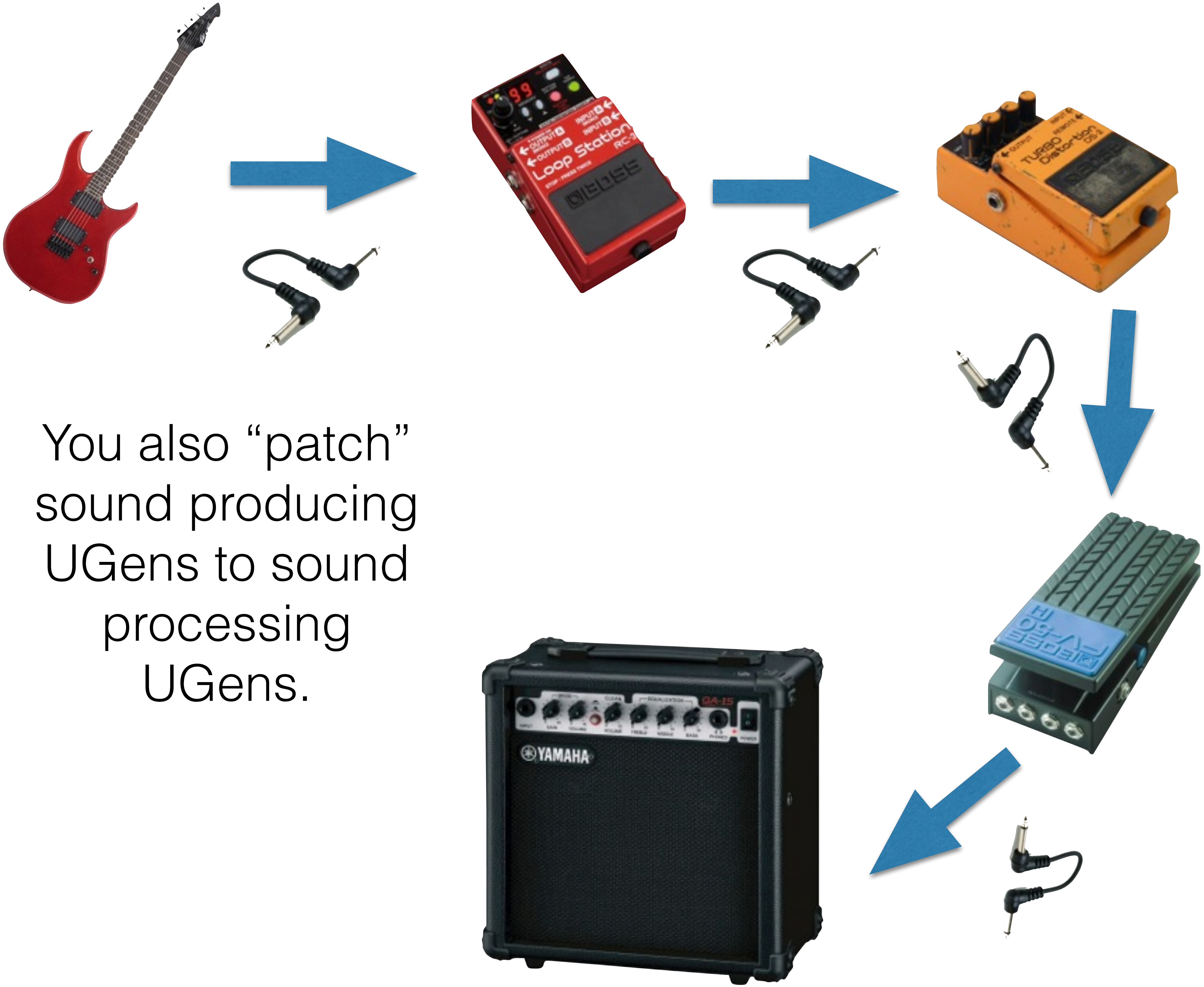
- The model for putting sounds where you want them in minim is “patching”

- You need to “patch” a sound producing object to an output.

- Sound producing objects are called “UGens”, short for “unit generators”.

You “patch” sound producers to an output!

A long one of these is a “guitar lead”, but short ones are usually called “patch leads”!



You also “patch” sound producing UGens to sound processing UGens.

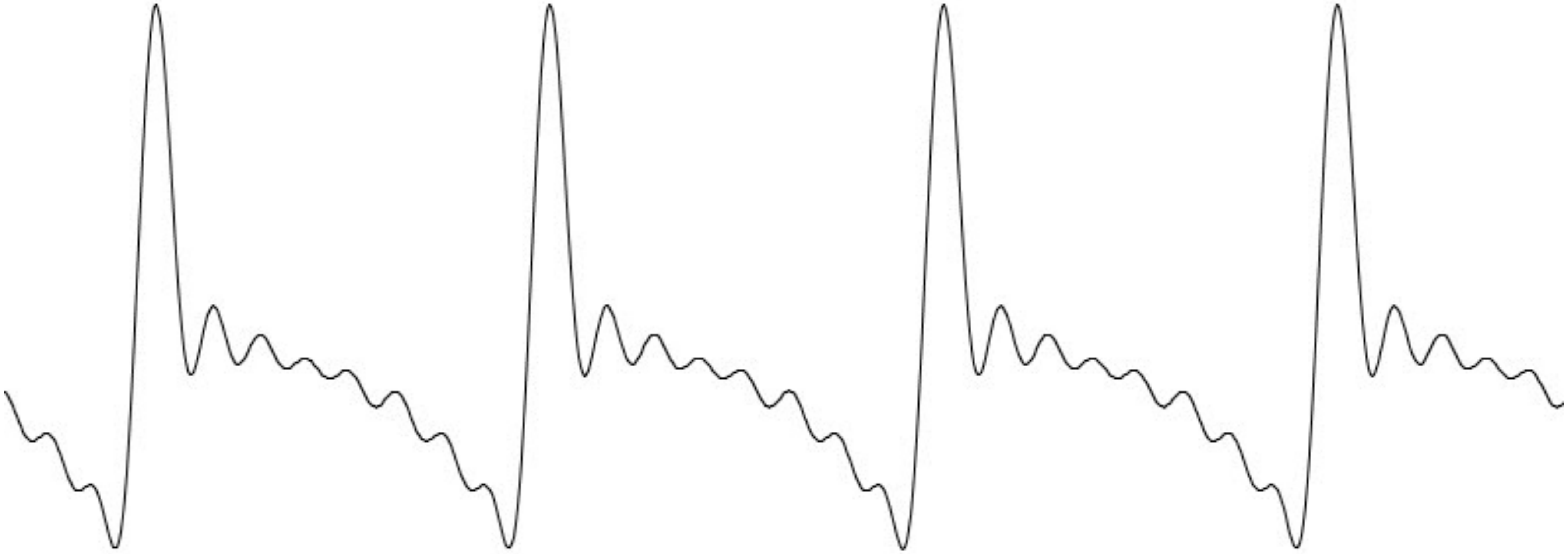
2. Additive Synthesis

```
float[] partials = {1, 2, 3, 4, 5, 6, 7, 8}; // harmonic
float[] amps = {1, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3};

wave0 = new Oscil(partials[0] * baseFrequency, waveAmplitude * amps[0], Waves.SINE);
wave1 = new Oscil(partials[1] * baseFrequency, waveAmplitude * amps[1], Waves.SINE);
wave2 = new Oscil(partials[2] * baseFrequency, waveAmplitude * amps[2], Waves.SINE);
wave3 = new Oscil(partials[3] * baseFrequency, waveAmplitude * amps[3], Waves.SINE);
wave4 = new Oscil(partials[4] * baseFrequency, waveAmplitude * amps[4], Waves.SINE);
wave5 = new Oscil(partials[5] * baseFrequency, waveAmplitude * amps[5], Waves.SINE);
wave6 = new Oscil(partials[6] * baseFrequency, waveAmplitude * amps[6], Waves.SINE);
wave7 = new Oscil(partials[7] * baseFrequency, waveAmplitude * amps[7], Waves.SINE);
```

- play sine tones at the same time - wave form is added together!
- add different frequencies to make complex waveforms

- the lowest sound is called the *fundamental* frequency
- the other sounds are called *partials*
- *harmonic* partials are natural number multiples of the fundamental frequency.
- *inharmonic* partials are not multiples of the fundamental
- usually the partials are quieter than the fundamental
- the frequency *and* amplitude of the partials determine the *timbre* of the sound.

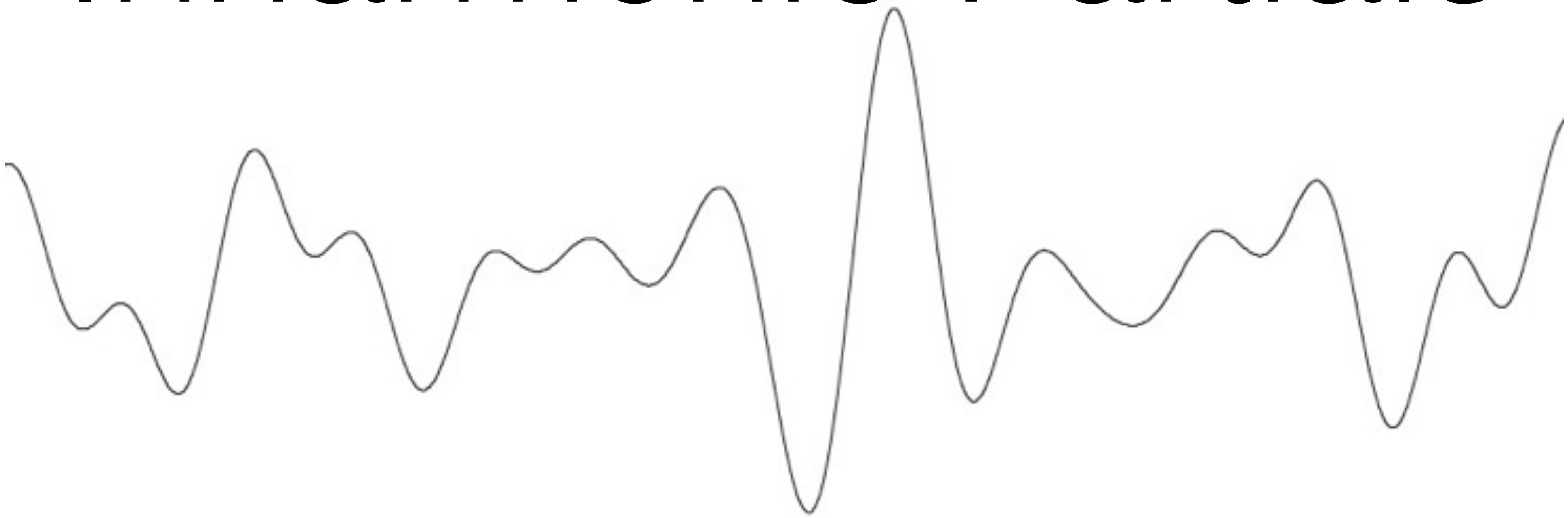
Harmonic Partials



```
float[] partials = {1, 2, 3, 4, 5, 6, 7, 8}; // harmonic
```

- strong “pitched” sound
- because we have odd and even harmonic partials, the sound is quite “nasal”

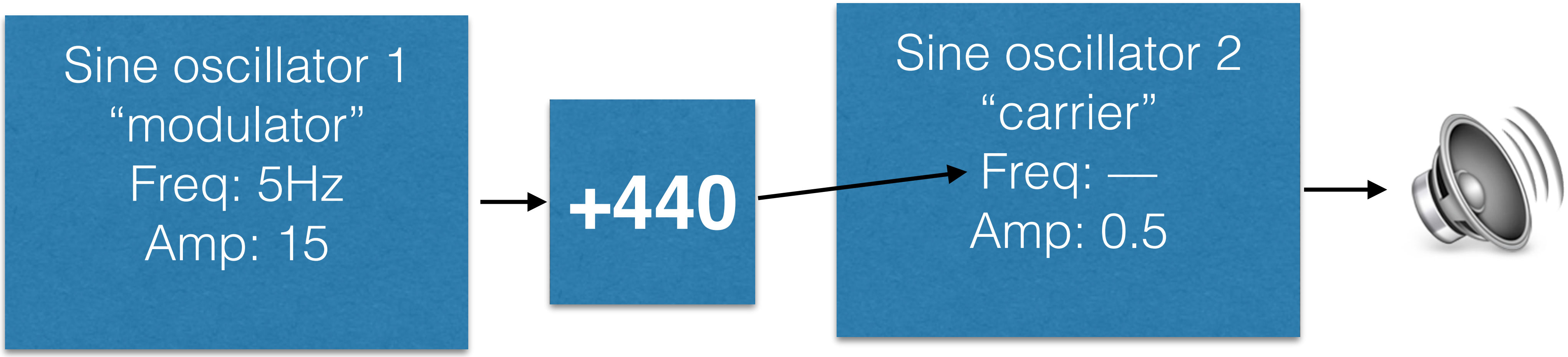
Inharmonic Partials



```
float[] partials = {1,1.5,1.75,2,2.25,2.4,2.6,2.9}; // inharmonic
```

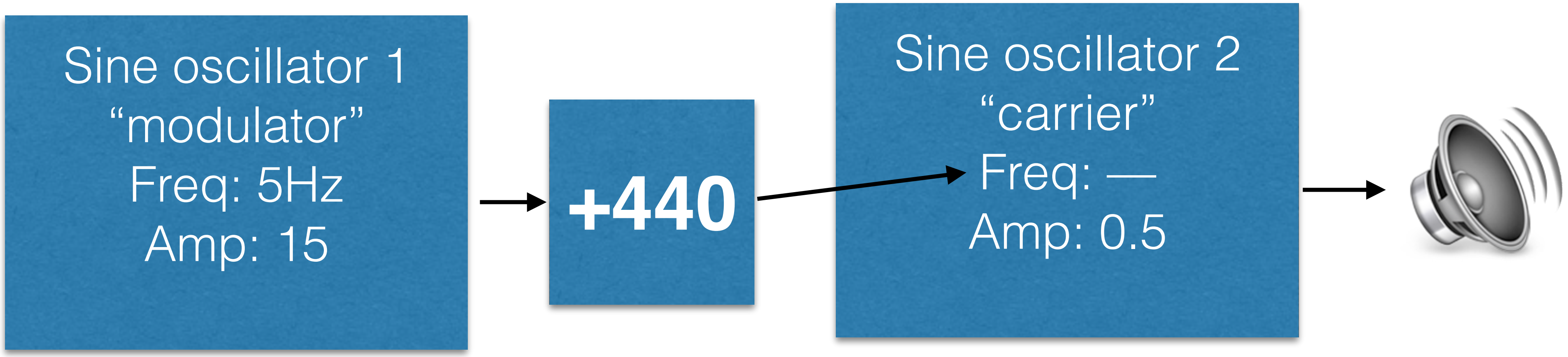
- with inharmonic partials there isn’t a strong pitch
- waveform doesn’t repeat regularly at the fundamental frequency
- try creating inharmonic partials randomly for lots of interesting variation of the timbre

3. FM Synthesis



- simple way to create complex sound colours!

- use a sine oscillator to control another sine oscillator!



Sine oscillator 1
“modulator”
Freq: 5Hz
Amp: 15

+440

Sine oscillator 2
“carrier”
Freq: —
Amp: 0.5

Output of Sine Osc. 1:

moves between 15 and -15,
repeats 5 times per second

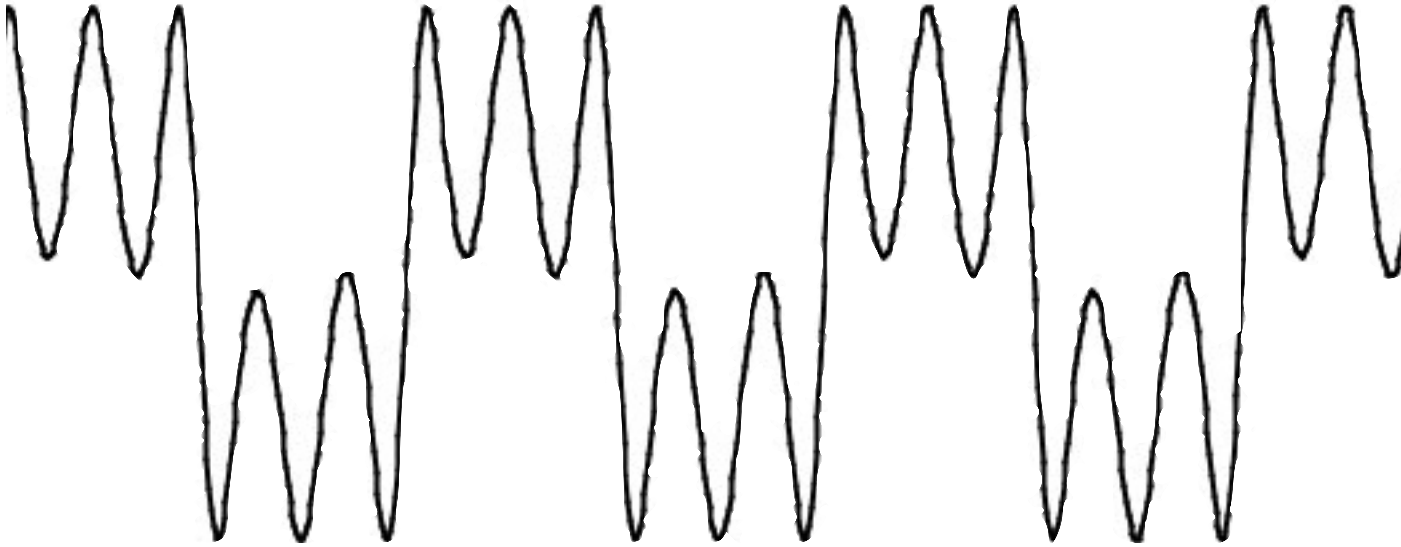
1. Add 440 to the output, so it ranges from 425 to 455.

2. *Patch* this value into the frequency of Sine Osc. 2

3. *Patch* Sine Osc. 2 to the speakers.

FM Synthesis Example

```
import ddf.minim.*;
import ddf.minim.ugens.*;
Minim minim;
AudioOutput out;
Oscil carrier, modulator;

void setup() {
  size(400, 200);
  minim = new Minim(this);
  out = minim.getLineOut();

  carrier = new Oscil(220,0.5,Waves.SINE);
  modulator = new Oscil(440,1000,Waves.SINE);
  modulator.offset.setLastValue(220);

  modulator.patch(carrier.frequency);
  carrier.patch(out);
}
void draw() {}
```



this bit adds 220!

Use mouse to change parameters!

```
void mouseMoved() {
  float modulateAmount = map( mouseY, 0, height, 1200, 1 );
  float modulateFrequency = map( mouseX, 0, width, 0.1, 800 );
  modulator.setFrequency( modulateFrequency );
  modulator.setAmplitude( modulateAmount );
}
```

Sound Files

- more complex than synthesised sound
- sometimes have cultural and narrative meanings?
- try recording some “field recordings” with your phone

4. Playing a Sound File

```
import ddf.minim.*;

Minim minim;
AudioPlayer player;

void setup() {
  size(200, 200);
  minim = new Minim(this);
  player = minim.loadFile("unilodge.mp3");
  player.play();
}
```

- this is the “simple” way to play a sound file in Minim.

Controlling Playback

```
player.pause();

player.rewind();

player.play();

player.loop(5);
```

5. Performing with a Sound File

- IMHO “performing” with sound means changing parameters…

- for sound files:

  - speed

  - direction

  - volume

  - rhythm (starting and stopping)

Sampler Object

```
Minim minim;
AudioOutput out;
MultiChannelBuffer sampleBuffer;
Sampler unilodge;
```

- *Sampler* object can play back sounds and we can control it like an oscillator!

- You'll need a *MultiChannelBuffer* object as well to store the sound data.

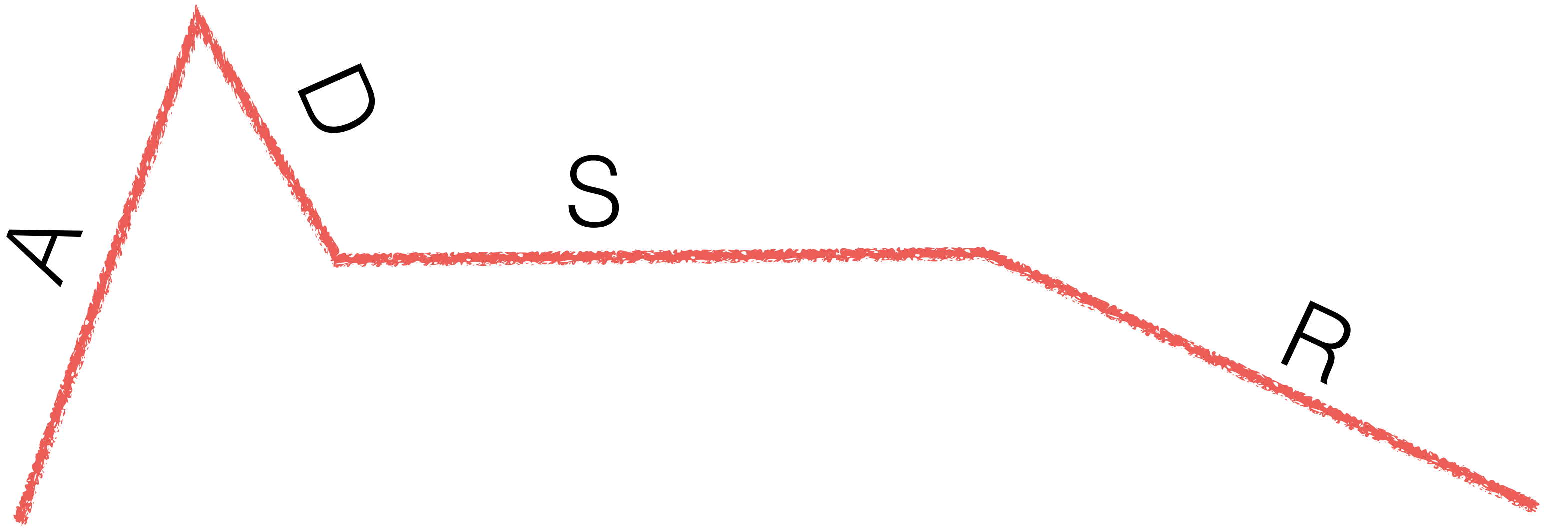
```
void setup() {
  size(800, 400);
  minim = new Minim(this);
  out = minim.getLineOut();

  sampleBuffer = new MultiChannelBuffer(1,1024);
  minim.loadFileIntoBuffer("unilodge.wav",sampleBuffer);

  unilodge = new Sampler(sampleBuffer, 44100, 1);
  unilodge.patch(out);
}

void mouseClicked() {
 unilodge.rate.setLastValue(mouseX/(1.0 * width));
 unilodge.trigger();
}
```

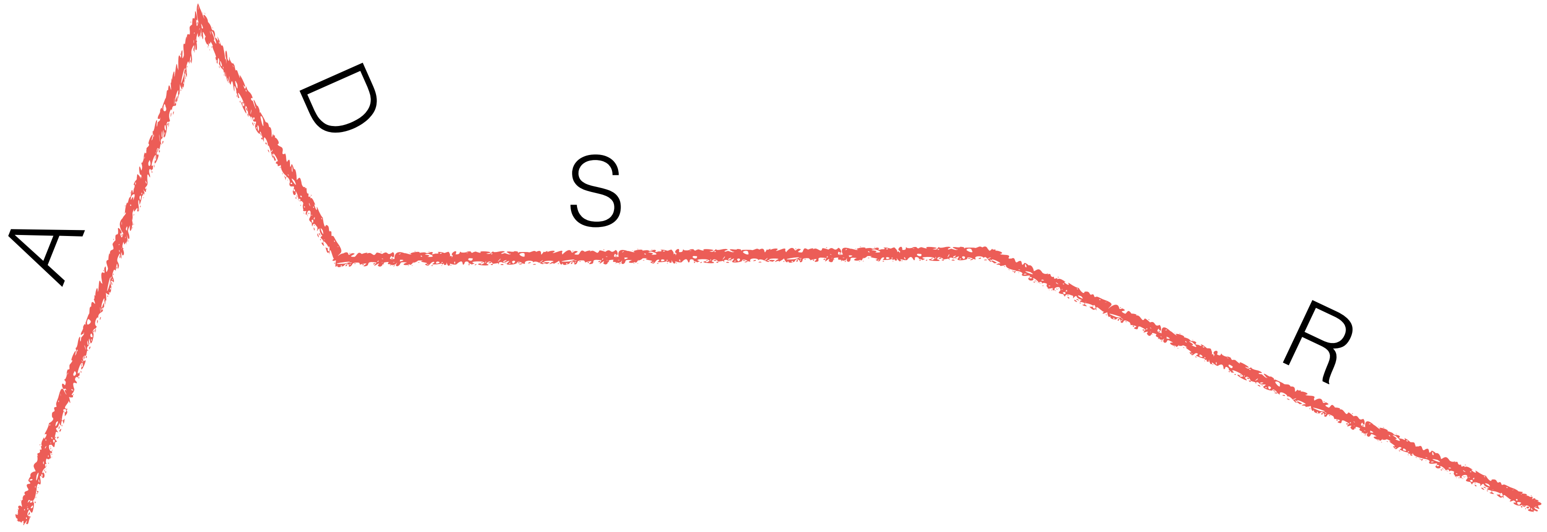
start playback!

changing playback rate!

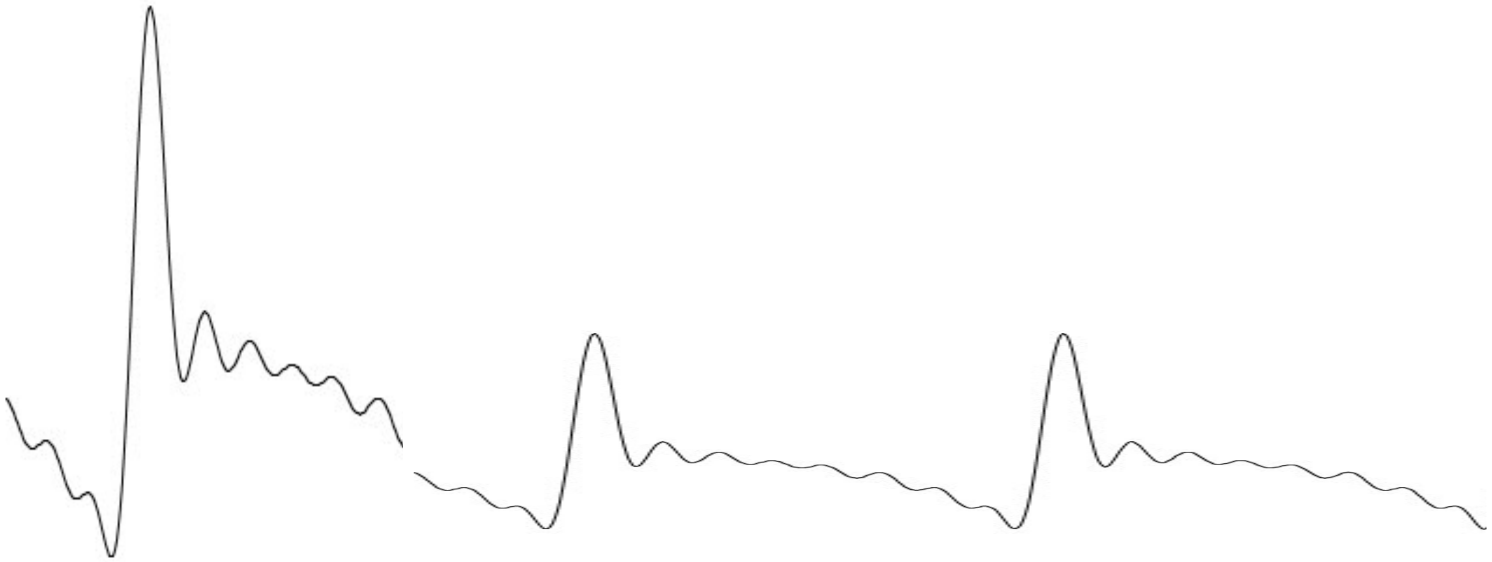
Envelopes

- what's the shape of a note?
  - attack
  - decay
  - sustain
  - release

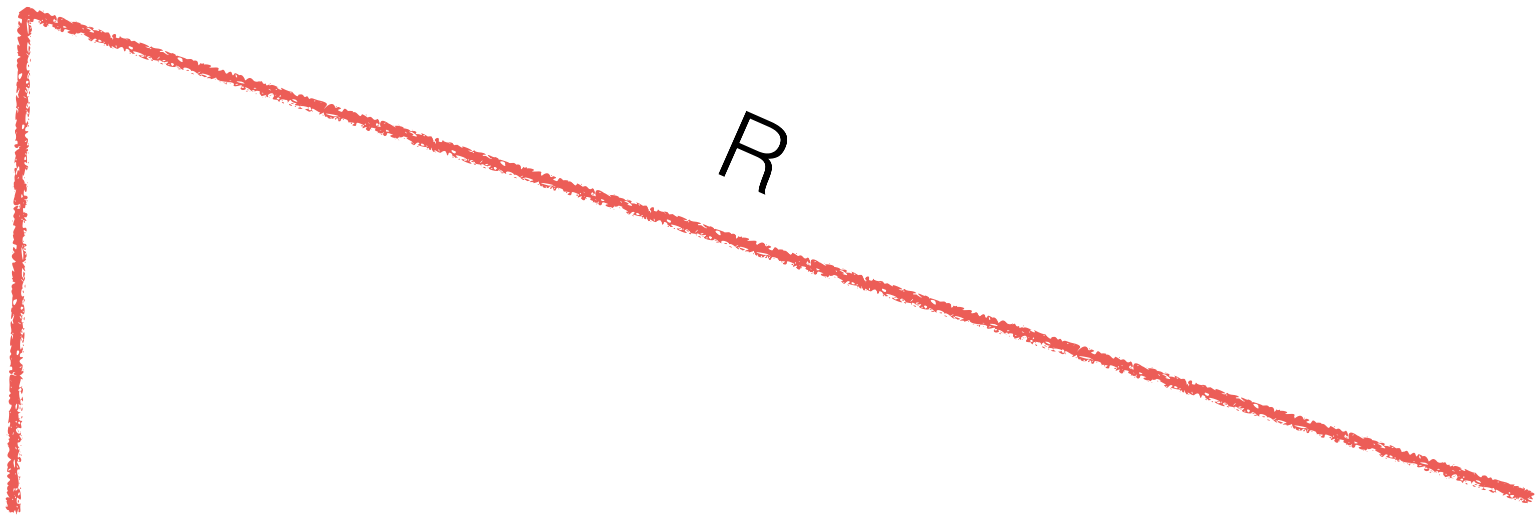




Patch an envelope curve to the amplitude of your sound.



Even Simpler Version: the *Line* object



```
Line envelope;
envelope = new Line();
envelope.patch(unilodge.amplitude);
envelope.activate( 3.0 * mouseY / 100.0, 0.5f, 0 );
```

```
//Syntax:
// Line.activate(float duration, float beginAmp, float endingAmp)
```

Example:

- click in the window to play notes
- mouseX - speed of playback
- mouseY - length of note
- playback position - random!

```
import ddf.minim.*;
import ddf.minim.ugens.*;

Minim minim;
AudioOutput out;
MultiChannelBuffer sampleBuffer;
Sampler unilodge;
Line envelope;
int sampleBufferLength;

void setup() {
  size(800, 400);
  minim = new Minim(this);
  out = minim.getLineOut();

  sampleBuffer = new MultiChannelBuffer(1,1024);
  minim.loadFileIntoBuffer("unilodge.wav",sampleBuffer);
  sampleBufferLength = sampleBuffer.getBufferSize();

  unilodge = new Sampler(sampleBuffer, 44100, 1);
  unilodge.patch(out);

  envelope = new Line();
  envelope.patch(unilodge.amplitude);
}

void draw() {}

void mouseClicked() {
 unilodge.rate.setLastValue(mouseX/(1.0 * width));
 unilodge.begin.setLastValue(random(sampleBufferLength));
 envelope.activate( 3.0 * mouseY / 100.0, 0.5f, 0 );
 unilodge.trigger();
}
```

7. Sound Input?

```
import ddf.minim.*;

Minim minim;
AudioInput in;

void setup()
{
  size(200, 200, P3D);
  minim = new Minim(this);
  in = minim.getLineIn();
}
```

- *AudioInput* object
- there’s ways to choose which input you use with the *getLineIn()* method

Analysing Sound Input

- “Onset detection” is a simple and effect interaction with sound
- that means searching for the start of sounds
- we’ll use Minim’s *BeatDetect* ugen
- look in *ddf.minim.analysis* library for other ideas!

```
import ddf.minim.*;
import ddf.minim.analysis.*;

Minim minim;
BeatDetect beat;
AudioInput in;

void setup()
{
  size(200, 200, P3D);
  minim = new Minim(this);
  in = minim.getLineIn();
  beat = new BeatDetect();
}

void draw()
{
  background(255); // white bg
  beat.detect(in.mix);
  if ( beat.isOnset() ) {
    background(255, 0, 0); // red bg
  }
}
```

1. set up a BeatDetect object
2. each frame, call *detect()*
3. check *isOnset()* method
4. use this information!
5. $$$!

Sounds like Eno

- let’s make some generative music!
- we’ll make a class that plays random notes at random times
- in every *draw()* loop, we’ll ask the class to play a note if it’s ready!
- we can instantiate the class several times to make a dense texture of ambient sounds!

```
class PhraseGenerator {
  float lastTimePlayed;
  float timeToNextNote;
  float minPlayTime;
  float maxPlayTime;
  Oscil wave;
  Damp envelope;

  PhraseGenerator(AudioOutput out) {
    lastTimePlayed = 0;
    minPlayTime = 2000;
    maxPlayTime = 10000;
    timeToNextNote = 1000;
    wave = new Oscil(0, 0.5, Waves.SINE);
    envelope = new Damp(0.1, 4.0, 0.5);
    wave.patch(envelope);
    envelope.patch(out);
  }

  void checkPlayTime() {
    if (millis() > lastTimePlayed + timeToNextNote) {
      wave.setFrequency(random(1000) + 50);
      envelope.activate();
      timeToNextNote = minPlayTime + random(minPlayTime, maxPlayTime);
      lastTimePlayed = millis();
    }
  }
}
```

new kind of envelope “Damp”

patching

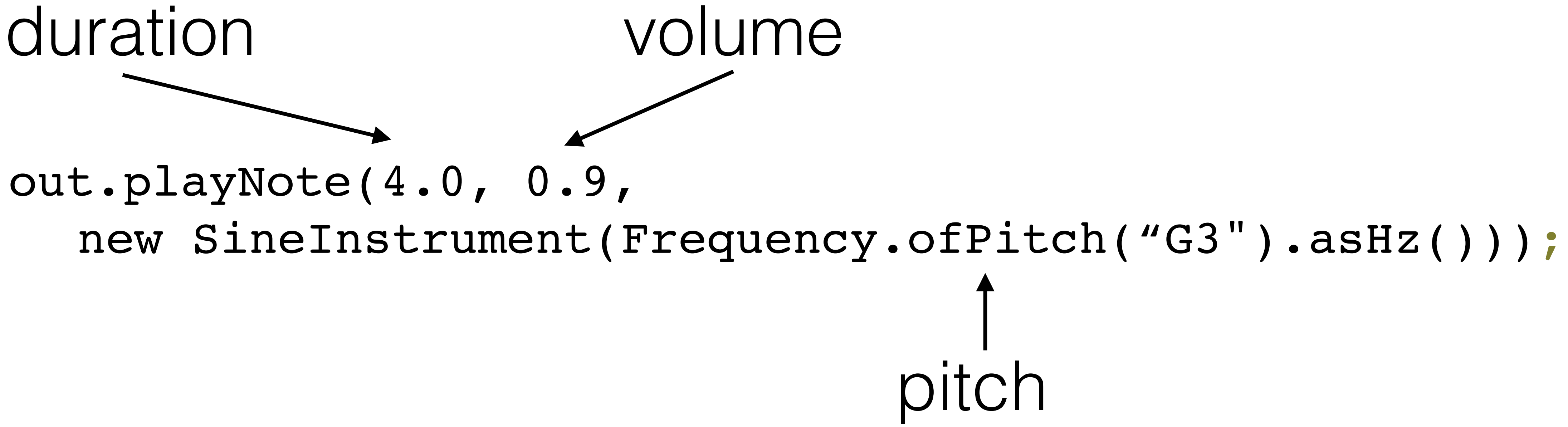
this gets called in *draw()*

random freq!

Other stuff in minim!

- *Instrument* interface - a standard way of controlling sound making classes!

- *Frequency* object - convert from MIDI pitch notation to frequency easily!



duration

volume

```
out.playNote(4.0, 0.9,
  new SineInstrument(Frequency.ofPitch(“G3").asHz()));
```

pitch

links:

- Minim website: code.compartmental.net/minim/
- Minim JavaDoc: code.compartmental.net/minim/javadoc/
- My examples: github.com/cpmpercussion/MusicInProcessingLecture
- Extra: Computer Music with Examples in SuperCollider (Cottle) www.mat.ucsb.edu/275/CottleSC3.pdf

Lab this Friday

- use a sketch to control synthetic sounds
- use a sketch to control sound files
- use BeatDetect to control a sketch