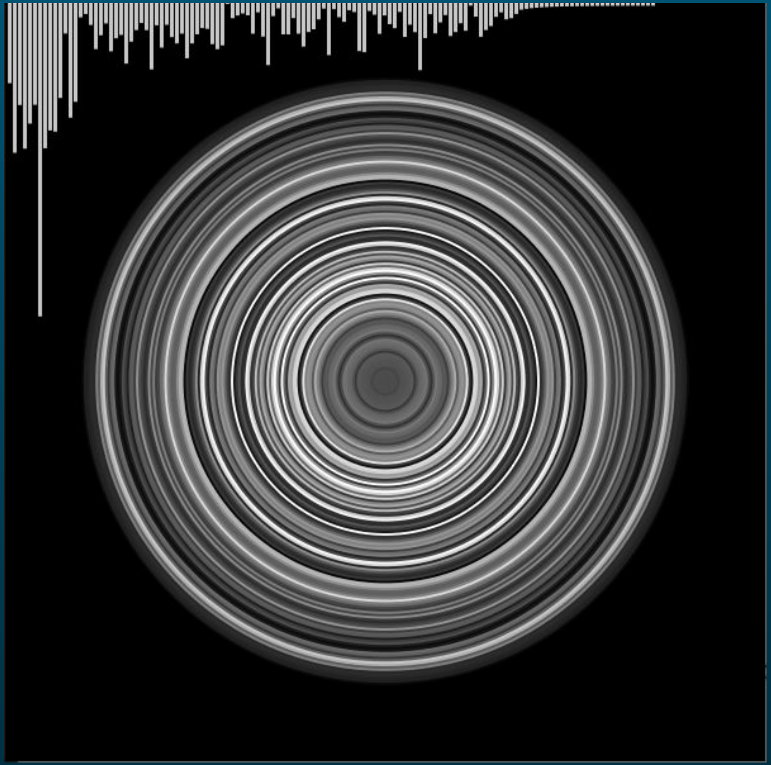
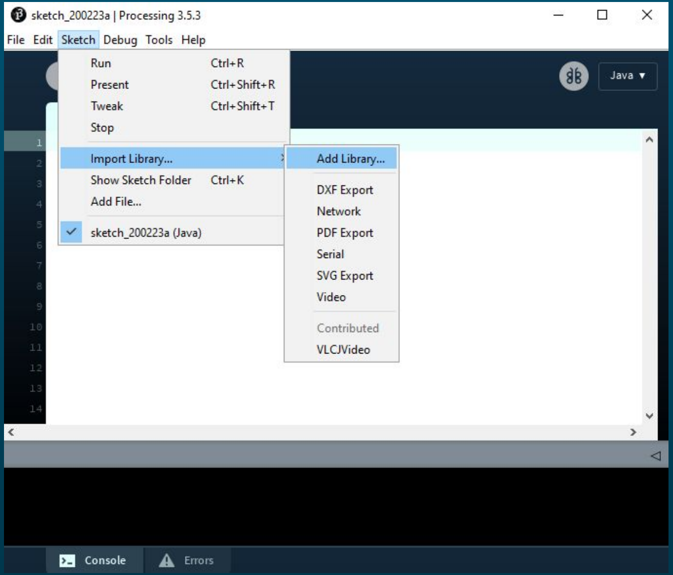
Week 09 - VJ Sonic

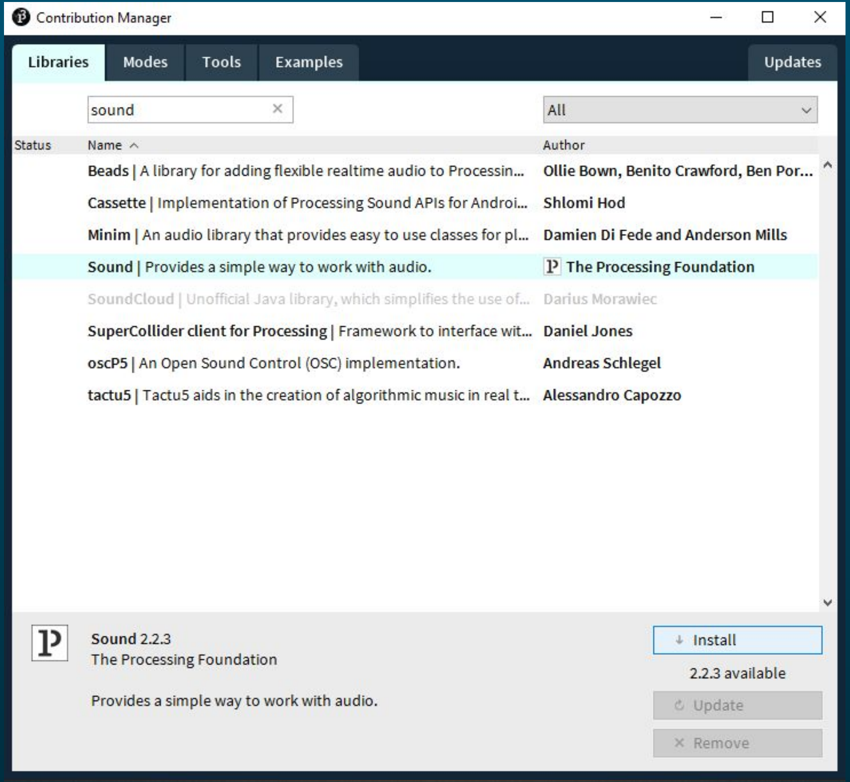
by Mike Wong. School of Creative Media



- Playback
- Analysis
- Synthesis

Install Processing Sound Library





sound ✕

All

Status	Name	Author
	Beads | A library for adding flexible realtime audio to Processin...	Ollie Bown, Benito Crawford, Ben Por...
	Cassette | Implementation of Processing Sound APIs for Androi...	Shlomi Hod
	Minim | An audio library that provides easy to use classes for pl...	Damien Di Fede and Anderson Mills
	Sound | Provides a simple way to work with audio.	The Processing Foundation
	SoundCloud | Unofficial Java library, which simplifies the use of...	Darius Morawiec
	SuperCollider client for Processing | Framework to interface wit...	Daniel Jones
	oscP5 | An Open Sound Control (OSC) implementation.	Andreas Schlegel
	tactu5 | Tactu5 aids in the creation of algorithmic music in real t...	Alessandro Capozzo

Sound 2.2.3
The Processing Foundation

Provides a simple way to work with audio.

SoundFile class of Sound Library

Methods	Description
`.channels()`	Returns # of channels
`.cue()`	Moves the playhead to the specified position
`.duration()`	Returns duration in second
`.frames()`	Returns number of frames
`.play()`	Playbacks once
`.jump()`	Jump to a specified position while continuing to play
`.pause()`	Pauses playing

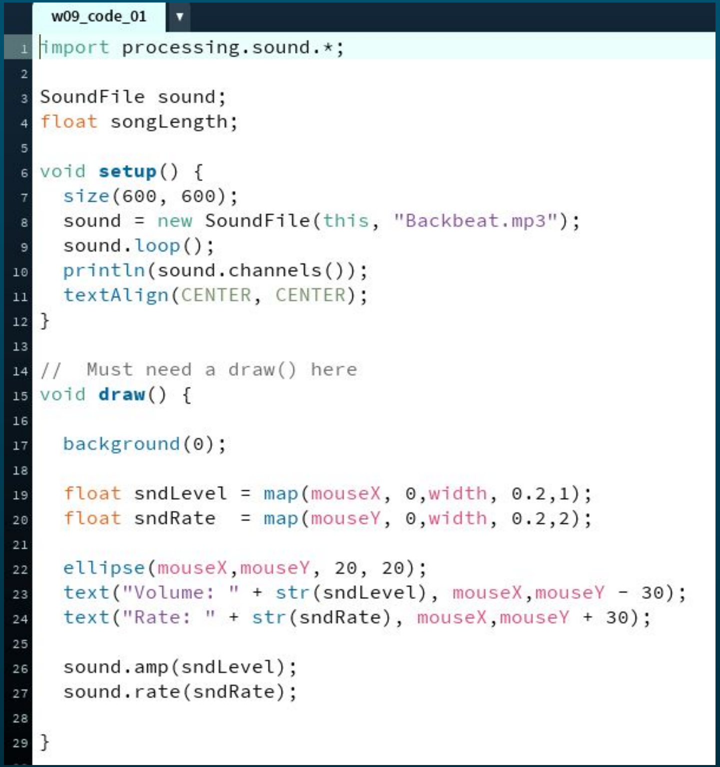
Methods	Description
`.isPlaying()`	Checks whether the file is playing
`.loop()`	Starts playback and loop
`.amp()`	Changes the volume
`.pan()`	Moves the sound in stereo panorama
`.rate()`	Sets the playback rate
`.stop()`	Stops the playback

Simple Audio playback of SoundFile

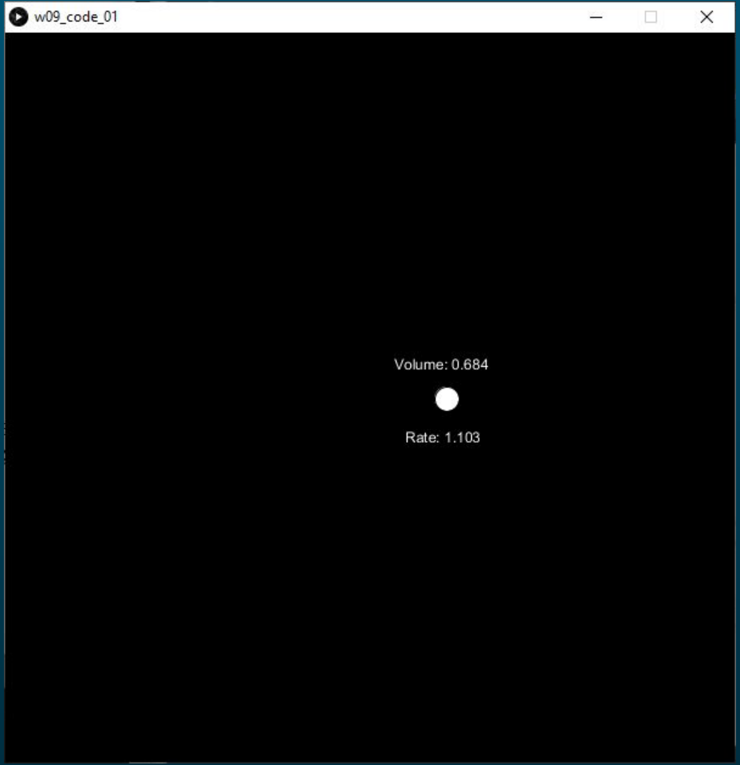
The following example illustrates how to load and playback an audio file. WAV/AIFF/MP3 file formats are supported.

```
import processing.sound.*;
SoundFile sound;
void setup() {
  size(600, 600);
  background(255);
  sound = new SoundFile(this, "sample.mp3");
  sound.play();
}
void draw() {  // the draw() function is required for playback
}
```

Example 1 - DJ LoFi

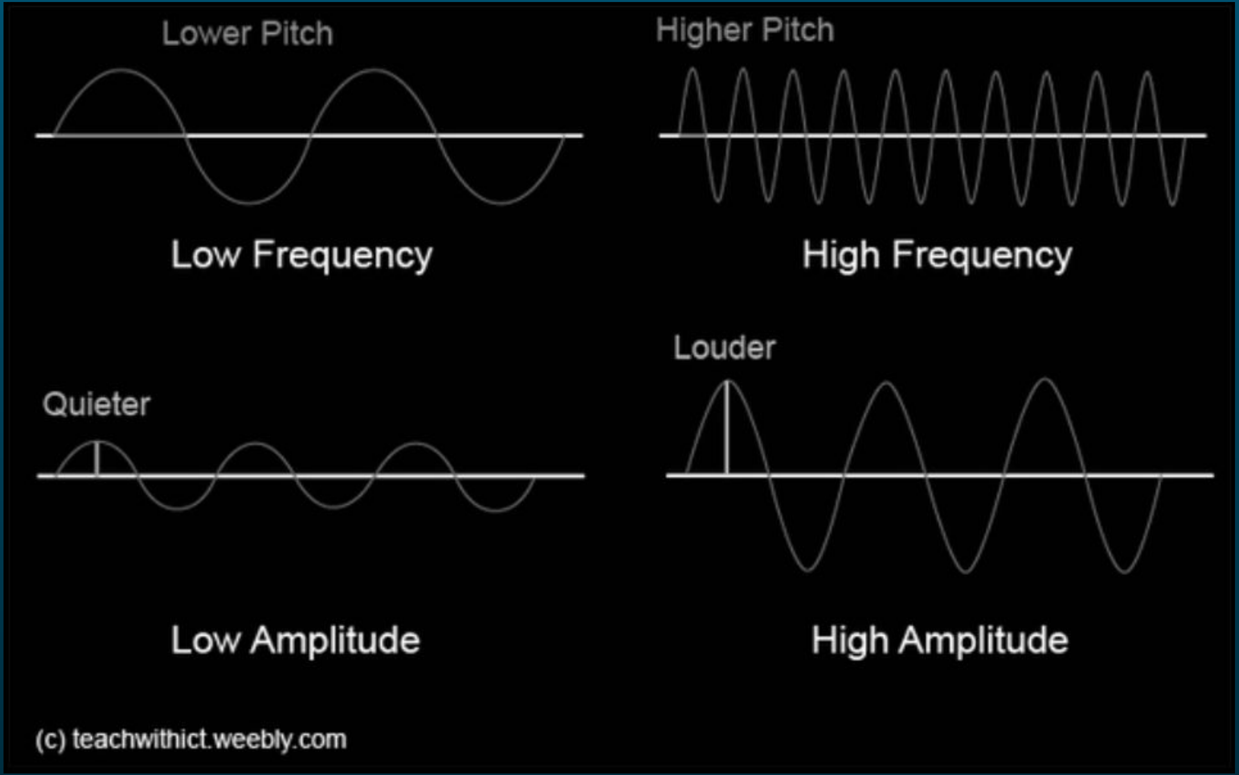


```
import processing.sound.*;

SoundFile sound;
float songLength;

void setup() {
  size(600, 600);
  sound = new SoundFile(this, "Backbeat.mp3");
  sound.loop();
  println(sound.channels());
  textAlign(CENTER, CENTER);
}

//  Must need a draw() here
void draw() {

  background(0);

  float sndLevel = map(mouseX, 0,width, 0.2,1);
  float sndRate  = map(mouseY, 0,width, 0.2,2);

  ellipse(mouseX,mouseY, 20, 20);
  text("Volume: " + str(sndLevel), mouseX,mouseY - 30);
  text("Rate: " + str(sndRate), mouseX,mouseY + 30);

  sound.amp(sndLevel);
  sound.rate(sndRate);

}
```



```
Volume: 0.684

Rate: 1.103
```

Basic Sound Analysis



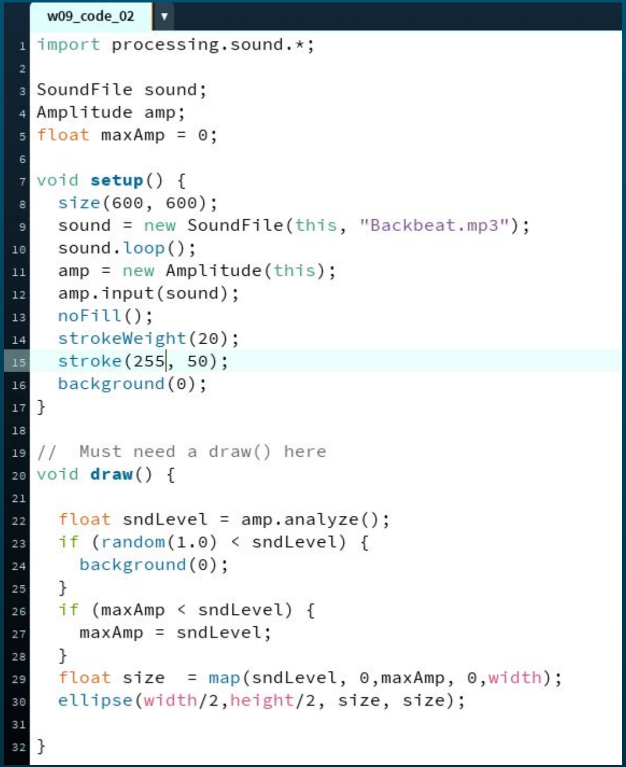
(c) teachwithict.weebly.com

Amplitude class of Sound Library

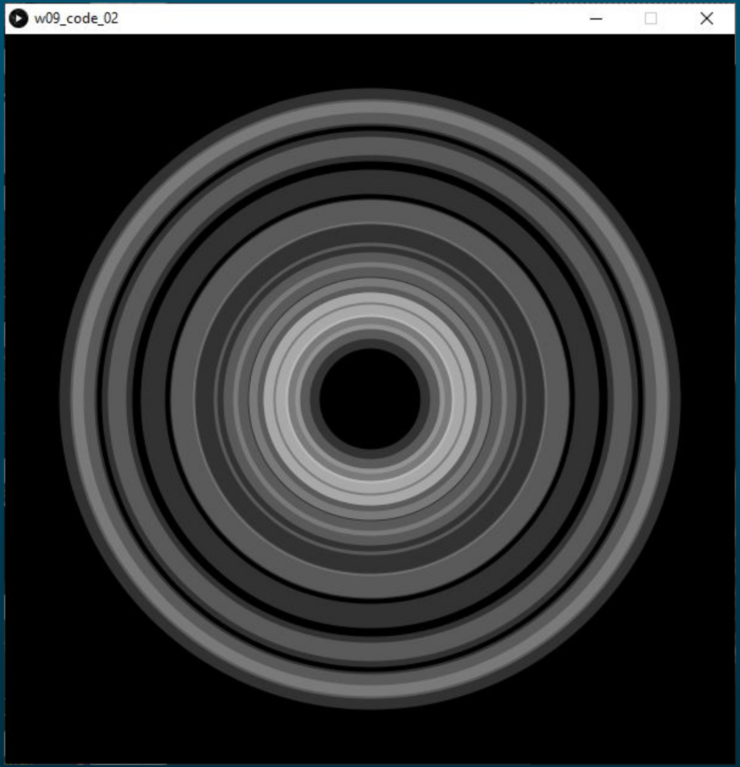
The Amplitude class analyzes the audio input, and outputs the root mean square of the amplitude of each audio block.

```
import processing.sound.*;
SoundFile sound;
Amplitude amp;
void setup() {
  size(600, 600);
  sound = new SoundFile(this, "Backbeat.mp3");
  sound.loop();
  amp = new Amplitude(this);      // New Amplitude
  amp.input(sound);               // Amplitude .input() to define audio source
}
void draw() {
  println(amp.analyze());         // Amplitude .analyze() to analyze the audio block
}
```

Example 2 - Rings of Amplitude



```
import processing.sound.*;

SoundFile sound;
Amplitude amp;
float maxAmp = 0;

void setup() {
  size(600, 600);
  sound = new SoundFile(this, "Backbeat.mp3");
  sound.loop();
  amp = new Amplitude(this);
  amp.input(sound);
  noFill();
  strokeWeight(20);
  stroke(255, 50);
  background(0);
}

//  Must need a draw() here
void draw() {

  float sndLevel = amp.analyze();
  if (random(1.0) < sndLevel) {
    background(0);
  }
  if (maxAmp < sndLevel) {
    maxAmp = sndLevel;
  }
  float size  = map(sndLevel, 0,maxAmp, 0,width);
  ellipse(width/2,height/2, size, size);

}
```

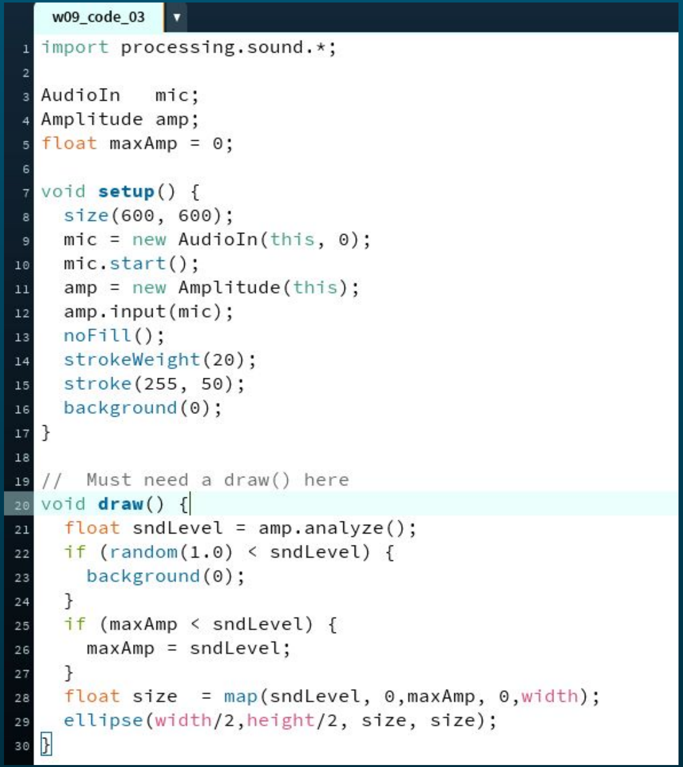


AudioIn class of Sound Library

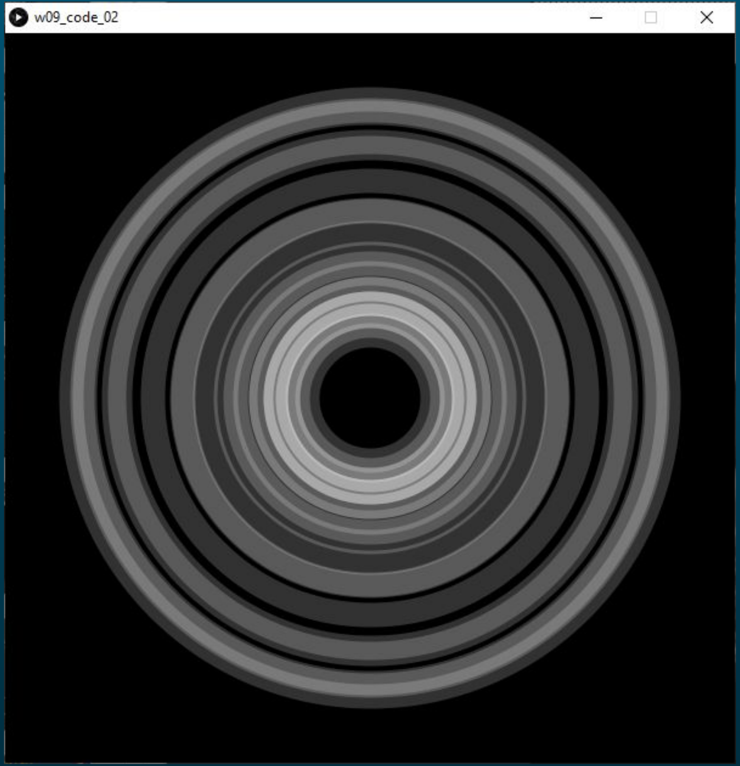
The AudioIn class captures audio input of your soundcard.

```
import processing.sound.*;
AudioIn in;
void setup() {
  size(600, 600);
  in = new AudioIn(this, 0);  // defines new AudioIn source
  in.play();                  // Captures & streams to speaker
}
void draw() {
}
```

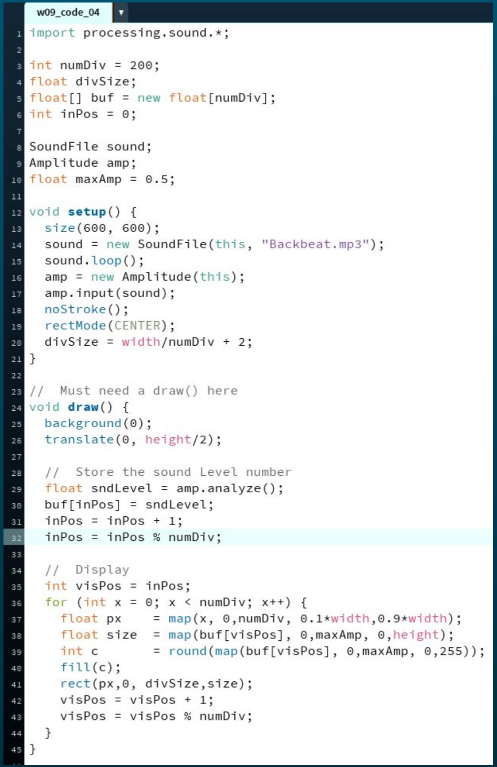
Example 3 - MC Circles



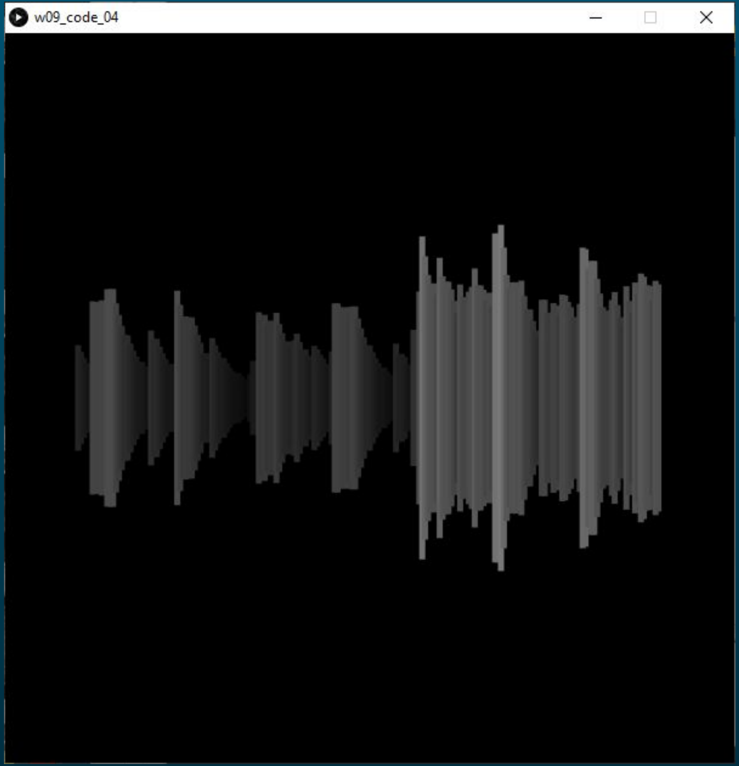
```
import processing.sound.*;

AudioIn   mic;
Amplitude amp;
float maxAmp = 0;

void setup() {
  size(600, 600);
  mic = new AudioIn(this, 0);
  mic.start();
  amp = new Amplitude(this);
  amp.input(mic);
  noFill();
  strokeWeight(20);
  stroke(255, 50);
  background(0);
}

//  Must need a draw() here
void draw() {
  float sndLevel = amp.analyze();
  if (random(1.0) < sndLevel) {
    background(0);
  }
  if (maxAmp < sndLevel) {
    maxAmp = sndLevel;
  }
  float size  = map(sndLevel, 0,maxAmp, 0,width);
  ellipse(width/2,height/2, size, size);
}
```



Example 4 - Array of Amplitudes

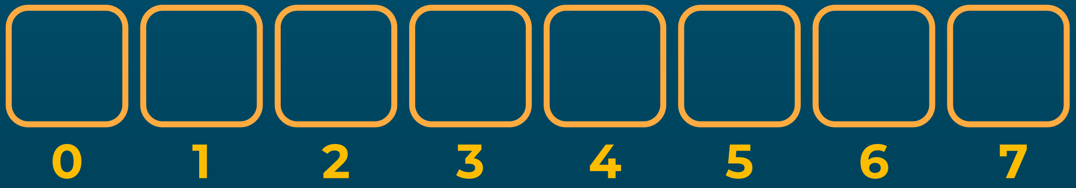


```
import processing.sound.*;

int numDiv = 200;
float divSize;
float[] buf = new float[numDiv];
int inPos = 0;

SoundFile sound;
Amplitude amp;
float maxAmp = 0.5;

void setup() {
  size(600, 600);
  sound = new SoundFile(this, "Backbeat.mp3");
  sound.loop();
  amp = new Amplitude(this);
  amp.input(sound);
  noStroke();
  rectMode(CENTER);
  divSize = width/numDiv + 2;
}

//  Must need a draw() here
void draw() {
  background(0);
  translate(0, height/2);

  //  Store the sound Level number
  float sndLevel = amp.analyze();
  buf[inPos] = sndLevel;
  inPos = inPos + 1;
  inPos = inPos % numDiv;

  //  Display
  int visPos = inPos;
  for (int x = 0; x < numDiv; x++) {
    float px    = map(x, 0,numDiv, 0.1*width,0.9*width);
    float size  = map(buf[visPos], 0,maxAmp, 0,height);
    int c       = round(map(buf[visPos], 0,maxAmp, 0,255));
    fill(c);
    rect(px,0, divSize,size);
    visPos = visPos + 1;
    visPos = visPos % numDiv;
  }
}
```



Example 4 - Array of Amplitudes

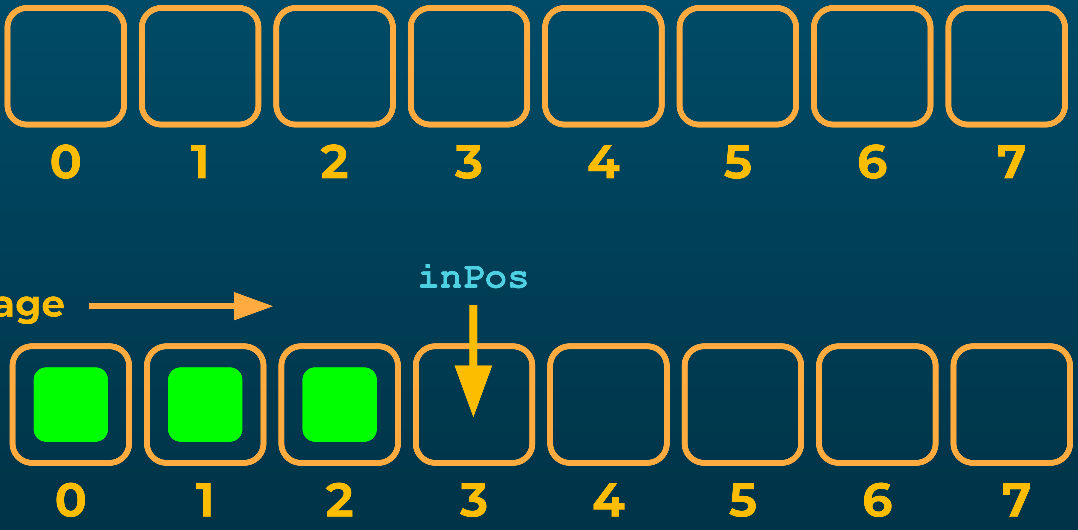
Array for Data Storage (capacity: 8)



Example 4 - Array of Amplitudes

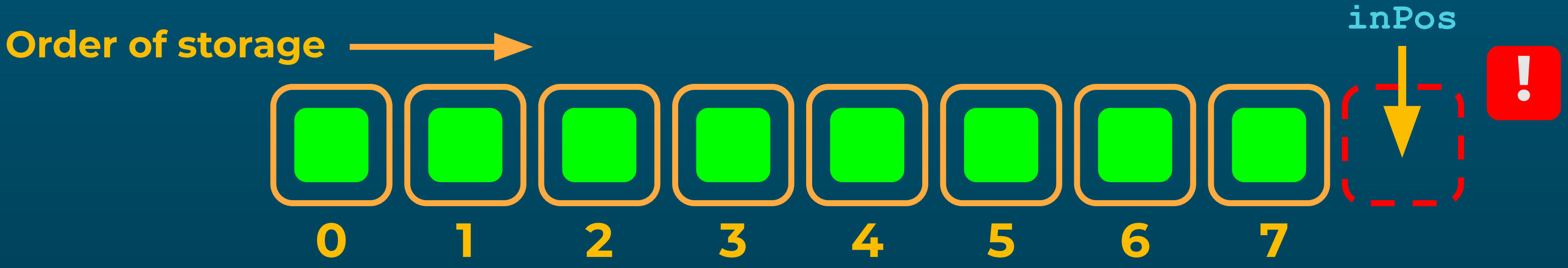
Array for Data Storage (capacity: 8)

Order of storage

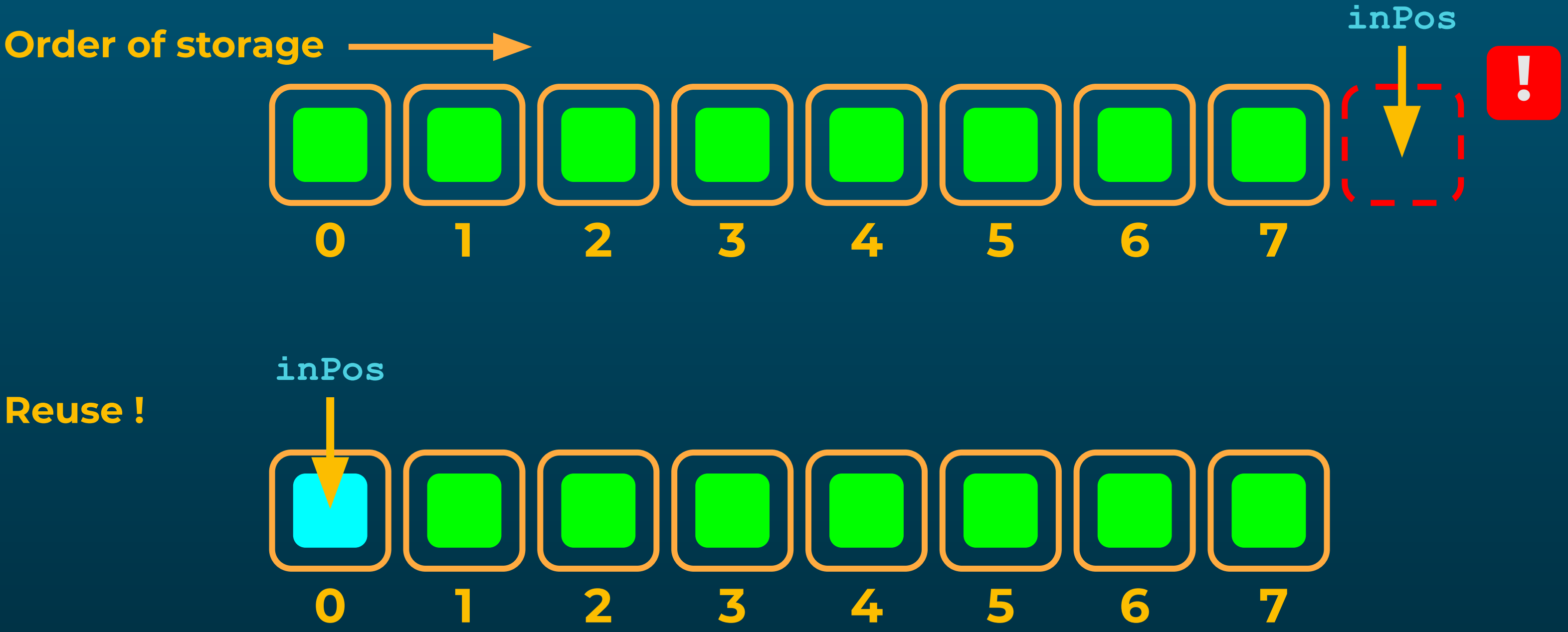


by Mike Wong. School of Creative Media

Example 4 - Array of Amplitudes



Example 4 - Array of Amplitudes



Order of storage

inPos

0 1 2 3 4 5 6 7

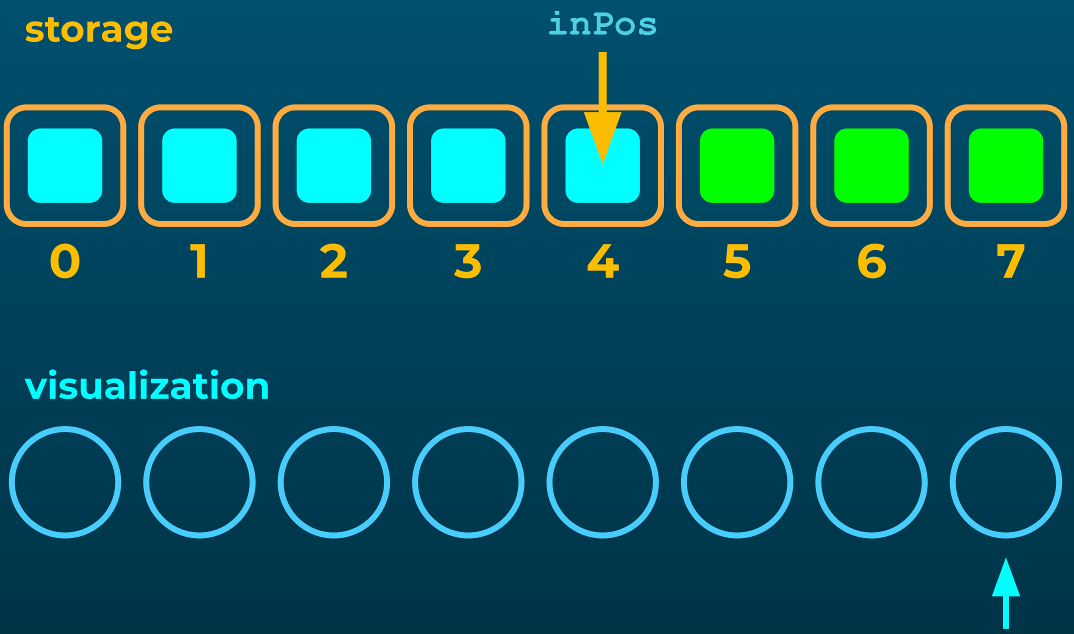
!

Reuse !

inPos

0 1 2 3 4 5 6 7

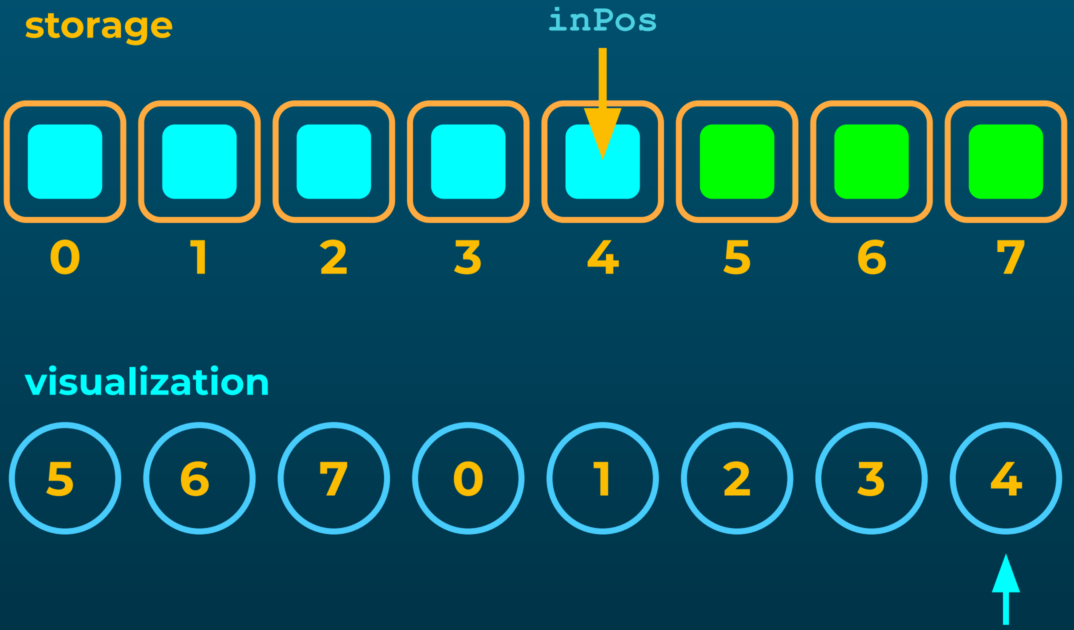
Example 4 - Array of Amplitudes



This one ALWAYS shows the latest input.

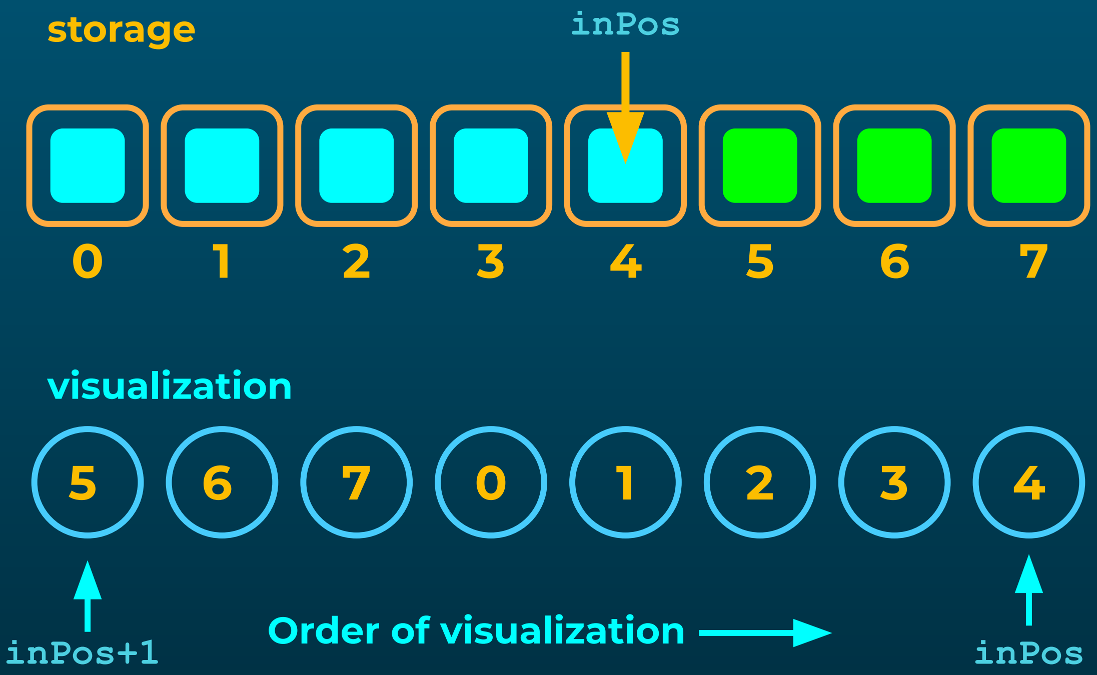
by Mike Wong. School of Creative Media

Example 4 - Array of Amplitudes

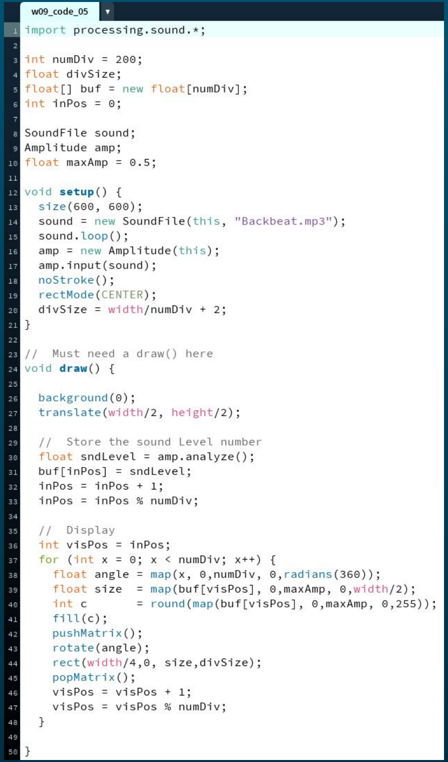


This position ALWAYS shows the latest input.

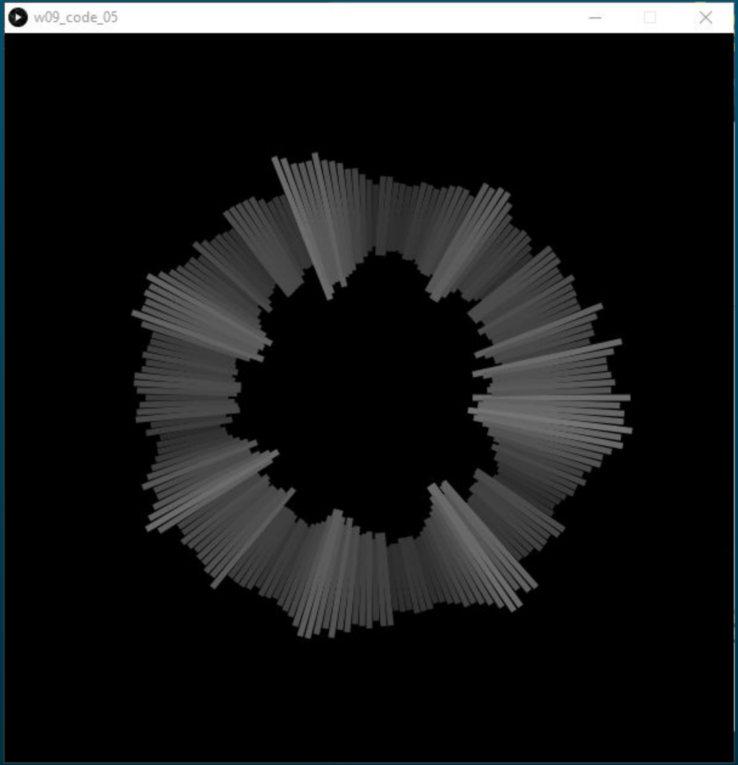
Example 4 - Array of Amplitudes



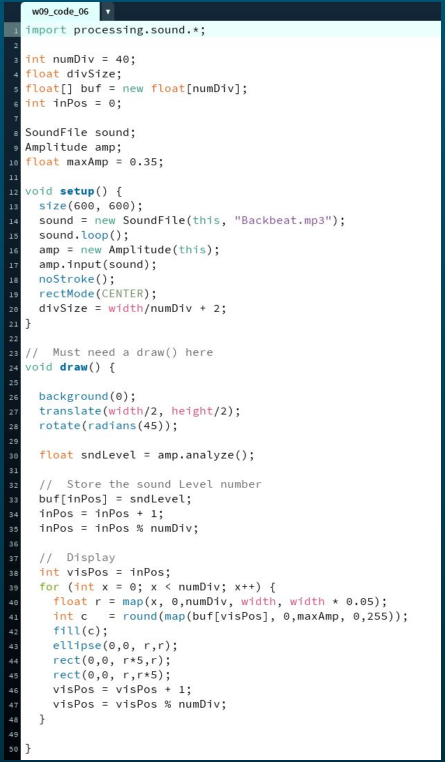
Example 5 - Amplitude Donut



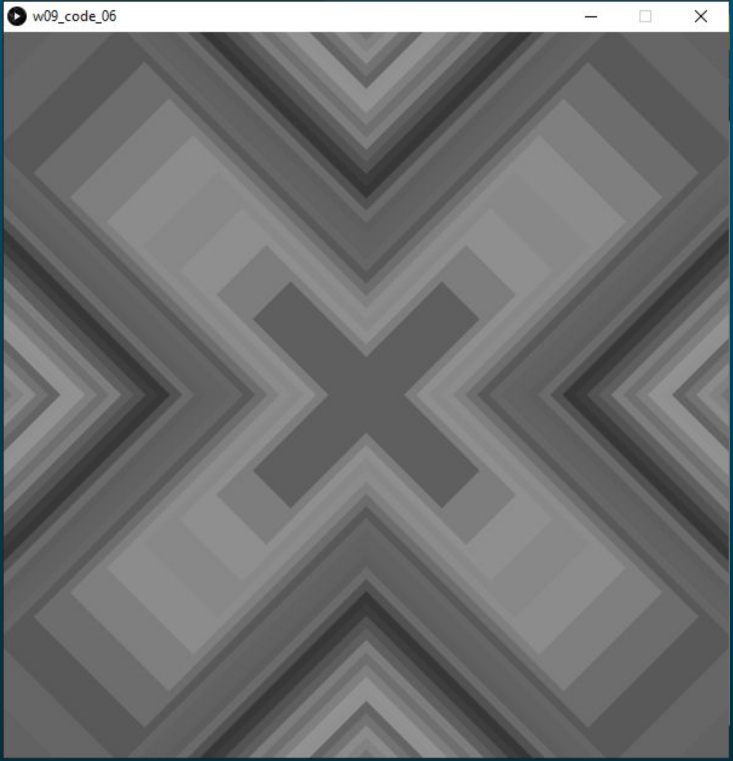
```
import processing.sound.*;

int numDiv = 200;
float divSize;
float[] buf = new float[numDiv];
int inPos = 0;

SoundFile sound;
Amplitude amp;
float maxAmp = 0.5;

void setup() {
  size(600, 600);
  sound = new SoundFile(this, "Backbeat.mp3");
  sound.loop();
  amp = new Amplitude(this);
  amp.input(sound);
  noStroke();
  rectMode(CENTER);
  divSize = width/numDiv + 2;
}

//  Must need a draw() here
void draw() {

  background(0);
  translate(width/2, height/2);

  //  Store the sound Level number
  float sndLevel = amp.analyze();
  buf[inPos] = sndLevel;
  inPos = inPos + 1;
  inPos = inPos % numDiv;

  //  Display
  int visPos = inPos;
  for (int x = 0; x < numDiv; x++) {
    float angle = map(x, 0,numDiv, 0,radians(360));
    float size  = map(buf[visPos], 0,maxAmp, 0,width/2);
    int c       = round(map(buf[visPos], 0,maxAmp, 0,255));
    fill(c);
    pushMatrix();
    rotate(angle);
    rect(width/4,0, size,divSize);
    popMatrix();
    visPos = visPos + 1;
    visPos = visPos % numDiv;
  }

}
```



Example 6 - Amplitude X



```
import processing.sound.*;

int numDiv = 40;
float divSize;
float[] buf = new float[numDiv];
int inPos = 0;

SoundFile sound;
Amplitude amp;
float maxAmp = 0.35;

void setup() {
  size(600, 600);
  sound = new SoundFile(this, "Backbeat.mp3");
  sound.loop();
  amp = new Amplitude(this);
  amp.input(sound);
  noStroke();
  rectMode(CENTER);
  divSize = width/numDiv + 2;
}

//  Must need a draw() here
void draw() {

  background(0);
  translate(width/2, height/2);
  rotate(radians(45));

  float sndLevel = amp.analyze();

  //  Store the sound Level number
  buf[inPos] = sndLevel;
  inPos = inPos + 1;
  inPos = inPos % numDiv;

  //  Display
  int visPos = inPos;
  for (int x = 0; x < numDiv; x++) {
    float r = map(x, 0,numDiv, width, width * 0.05);
    int c    = round(map(buf[visPos], 0,maxAmp, 0,255));
    fill(c);
    ellipse(0,0, r,r);
    rect(0,0, r*5,r);
    rect(0,0, r,r*5);
    visPos = visPos + 1;
    visPos = visPos % numDiv;
  }

}
```



FFT for Frequency Analysis

Fast Fourier Transform (FFT) analyzes the audio input in terms of frequency, and outputs the normalized power spectrum.

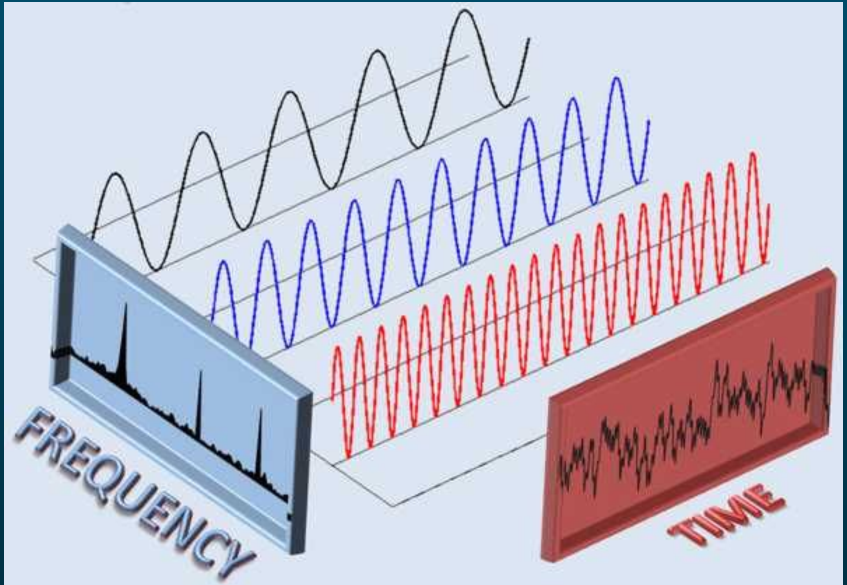


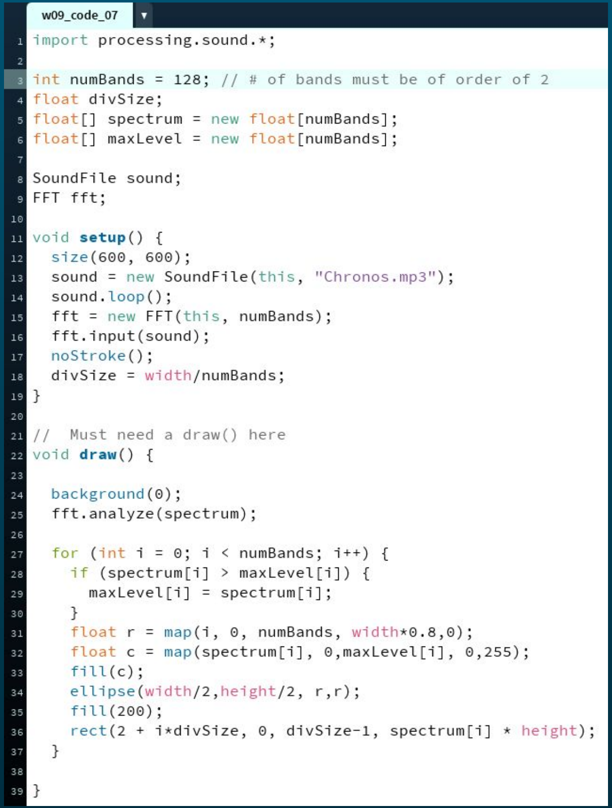
Image URL: https://www.i-programmer.info/news/181-algorithms/3644-a-faster-fourier-transform.html

FFT class of Sound Library

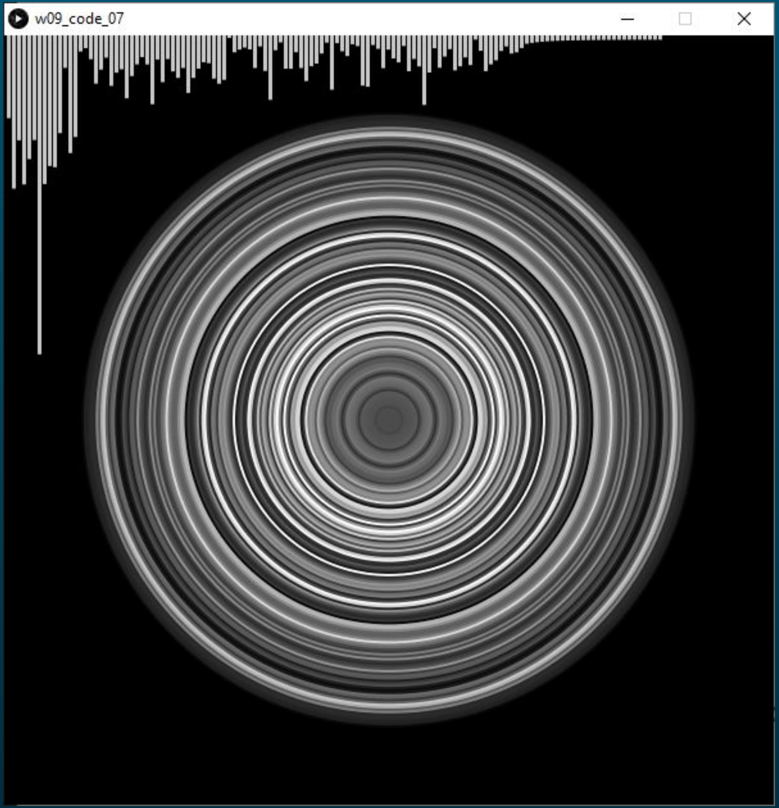
The FFT class analyzes the audio input in terms of frequency, and outputs the normalized power spectrum.

```
import processing.sound.*;
SoundFile sound;
FFT fft;
float[] spectrum = new float[128];    // To store the results returned by FFT.analyze()
void setup() {
  size(600, 600);
  sound = new SoundFile(this, "Backbeat.mp3");
  sound.loop();
  fft = new FFT(this, 128);      // New FFT
  fft.input(sound);              // FFT .input() to define audio source
}
void draw() {
  fft.analyze(spectrum);         // FFT .analyze() returns output into the array
}
```

Example 7 - FFT visualization

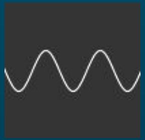


```
import processing.sound.*;

int numBands = 128; // # of bands must be of order of 2
float divSize;
float[] spectrum = new float[numBands];
float[] maxLevel = new float[numBands];

SoundFile sound;
FFT fft;

void setup() {
  size(600, 600);
  sound = new SoundFile(this, "Chronos.mp3");
  sound.loop();
  fft = new FFT(this, numBands);
  fft.input(sound);
  noStroke();
  divSize = width/numBands;
}

//  Must need a draw() here
void draw() {

  background(0);
  fft.analyze(spectrum);

  for (int i = 0; i < numBands; i++) {
    if (spectrum[i] > maxLevel[i]) {
      maxLevel[i] = spectrum[i];
    }
    float r = map(i, 0, numBands, width*0.8,0);
    float c = map(spectrum[i], 0,maxLevel[i], 0,255);
    fill(c);
    ellipse(width/2,height/2, r,r);
    fill(200);
    rect(2 + i*divSize, 0, divSize-1, spectrum[i] * height);
  }

}
```

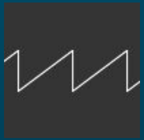


Simple Sound Synthesis

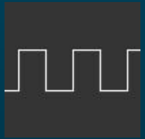
The following oscillators are available from the sound library for simple synthesis.



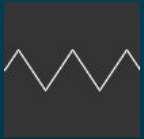
SinOsc



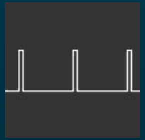
SawOsc



SqrOsc



TriOsc



Pulse

The common methods available in each oscillator.

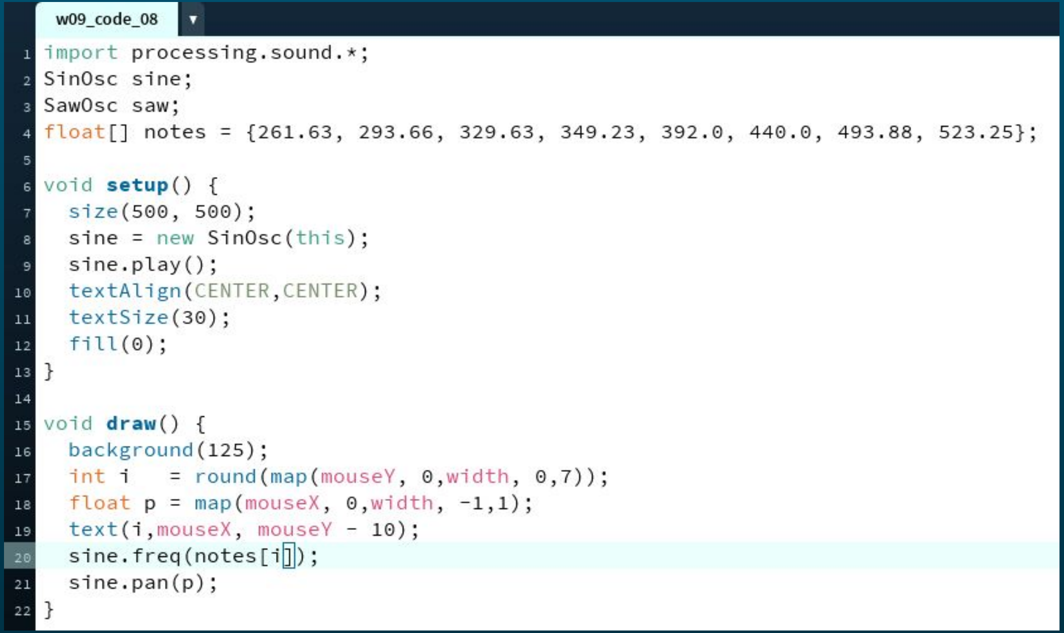
Methods	Description
`.play()`	Starts the oscillator
`.freq()`	Sets the frequency in Hz
`.amp()`	Sets the amplitude
`.add()`	Offset the output
`.pan()`	Moves the sound in stereo
`.stop()`	Stops the playback

SinOsc class of Sound Library

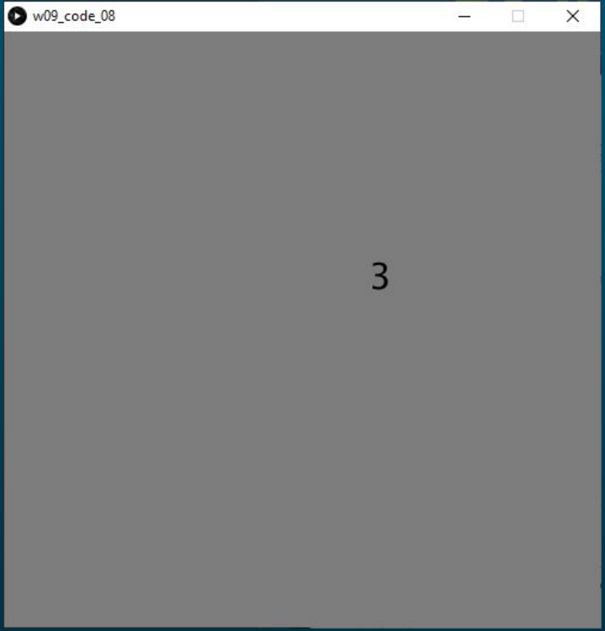
The SinOsc class generates Sinusoidal wave based tone.

```
import processing.sound.*;
SinOsc sine;

void setup() {
  size(640, 360);
  background(255);

  // Create the sine oscillator.
  sine = new SinOsc(this);
  sine.play();
}

void draw() {
}
```

Example 8 - Simple Notes



```
import processing.sound.*;
SinOsc sine;
SawOsc saw;
float[] notes = {261.63, 293.66, 329.63, 349.23, 392.0, 440.0, 493.88, 523.25};

void setup() {
  size(500, 500);
  sine = new SinOsc(this);
  sine.play();
  textAlign(CENTER,CENTER);
  textSize(30);
  fill(0);
}

void draw() {
  background(125);
  int i    = round(map(mouseY, 0,width, 0,7));
  float p = map(mouseX, 0,width, -1,1);
  text(i,mouseX, mouseY - 10);
  sine.freq(notes[i]);
  sine.pan(p);
}
```


w09_code_08
3