PPEM 2020

Sonido

Repaso

Movie (play, time, duration, jump, speed)

Capture (read)

Sonido

Micrófono

Cargar archivo de audio

Generar sonido

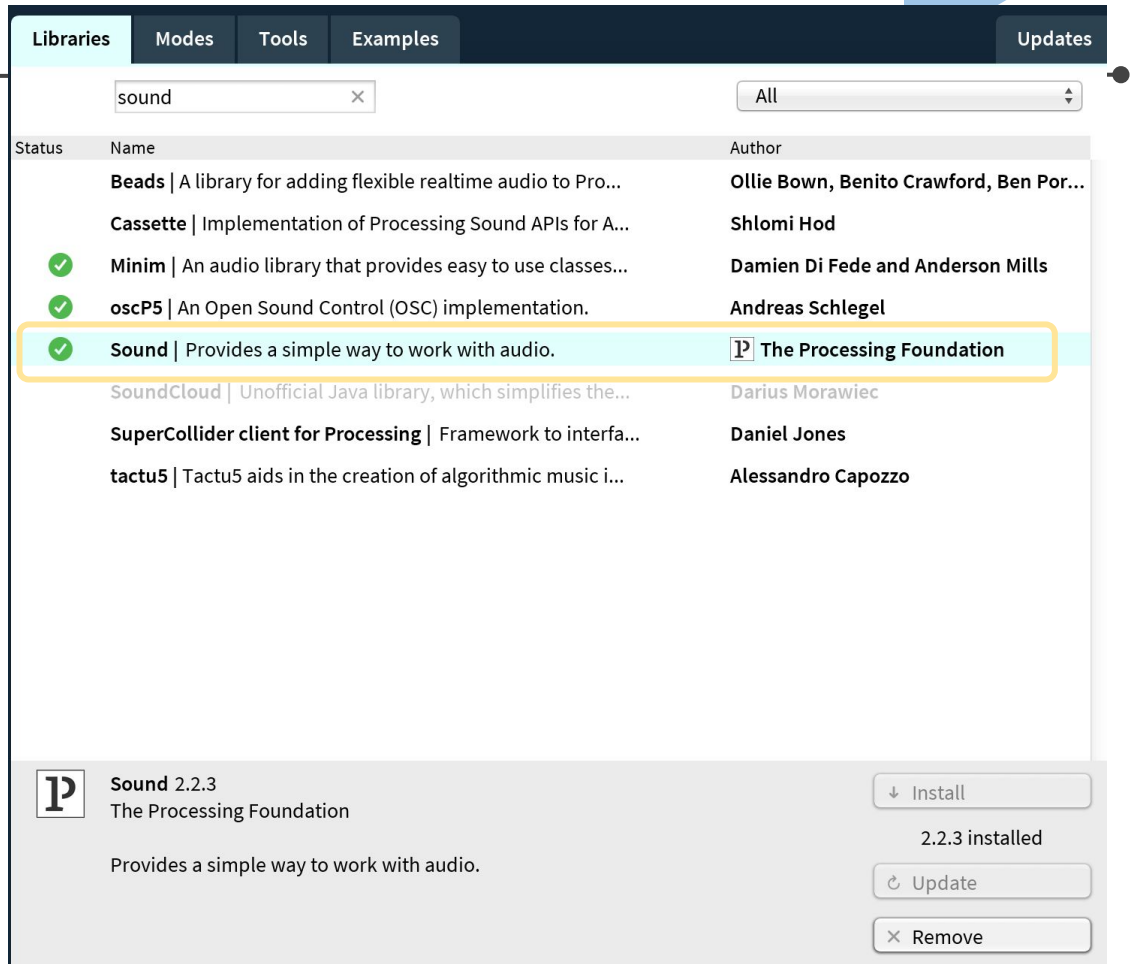
Analizar el sonido

Filtros

Efectos de audio

IMPORTANTE

Es importante instalar la librería **Sound** antes de correr los ejemplos.



Micrófono 

AudioIn

start() // Arranca a escuchar la entrada de sonido y NO rutea a la salida de audio.

play() // Arranca a escuchar la entrada de sonido y lo rutea a la salida de audio.

amp() // Cambia el volumen del micrófono. Recibe valores entre 0.0 y 1.0

add() // Offset the output of the input stream by given value (?!)

pan() // Mueve la fuente de sonido en stereo. Va de -1.0 (izquierda) a 1.0 (derecha).

set() // Fija amp, add y pan a la vez.

stop() // Para escuchar la entrada de sonido.

AudioIn.pan

```
import processing.sound.*;
AudioIn in;

void setup() {
  size(640, 360);
  in = new AudioIn(this, 0);
  in.play(); // play pasa la entrada de audio a la salida (parlantes, auriculares)
}

void draw() {
  // Mapeo mouseX a valores de -1.0 a 1.0 para cambiar donde se escucha (oreja izquierda o derecha)
  in.pan(map(mouseX, 0, width, -1.0, 1.0));
}
```

Análisis de audio: Amplitude

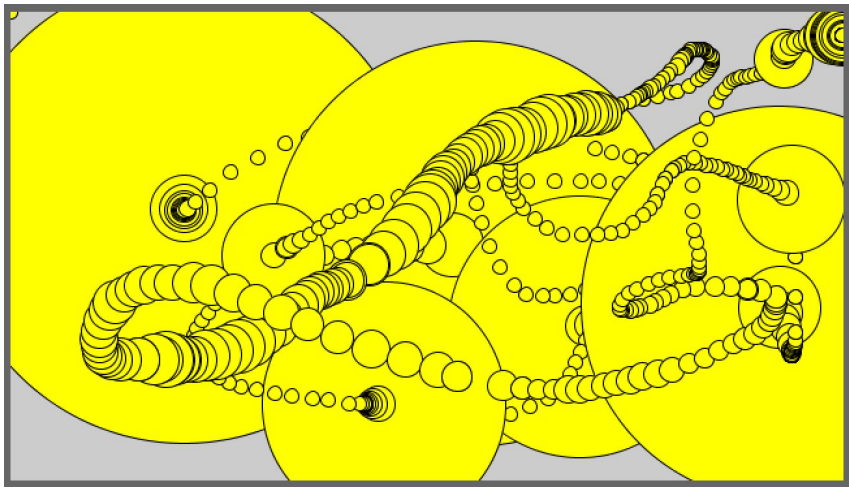
Para analizar el volumen.

input() // para definir la entrada de audio que se analizará.

analyze() // devuelve valores entre 0.0 y 1.0 que se corresponden al volumen.

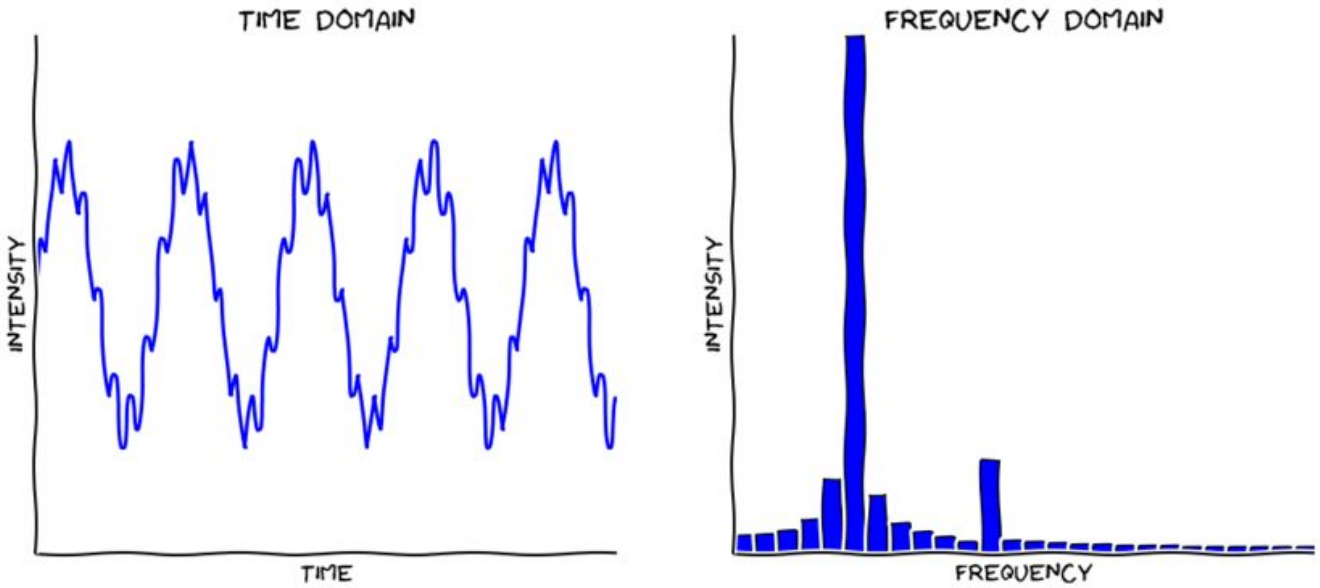
Ejemplo: tamaño y volumen (micrófono)

```
import processing.sound.*;
Amplitude amp;
AudioIn in;
void setup() {
  size(640, 360);
  fill(255,255,0);
  in = new AudioIn(this, 0); // entrada de audio -> micrófono
  amp = new Amplitude(this); // para analizar el volumen
  in.start(); // empiezo a escuchar
  amp.input(in); // defino que debe analizar
}
void draw() {
  float radio = map(amp.analyze(),0,1,10,height); // mapeo el volumen al tamaño
  ellipse(mouseX,mouseY, radio, radio);
}
```



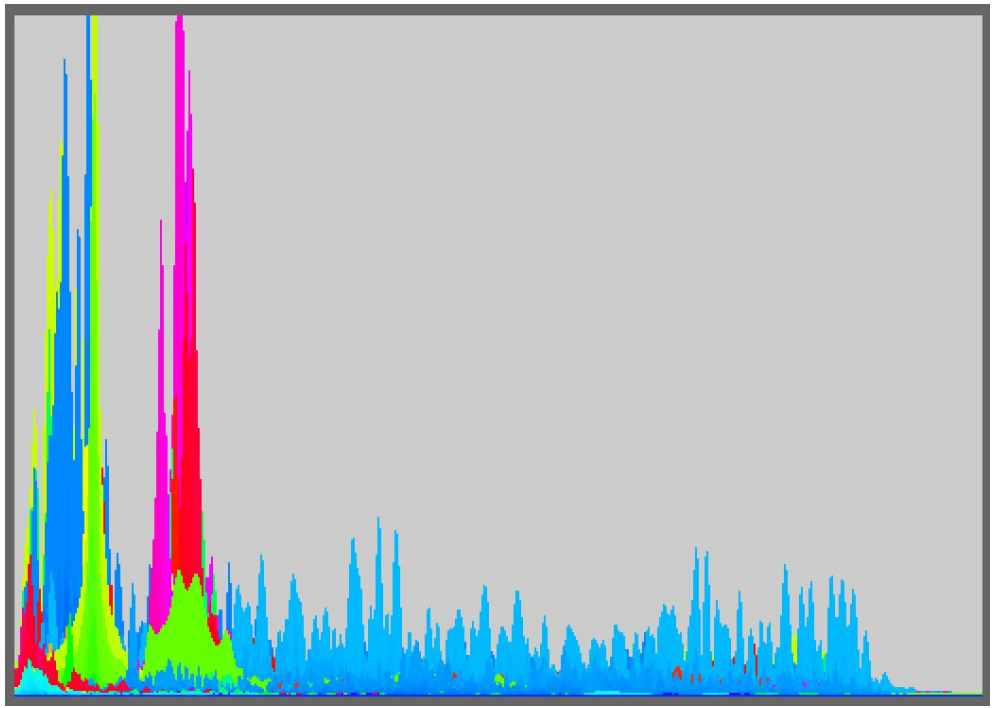
Análisis de audio: Fast Fourier Transformation

Análisis del espectro. Divide la señal de audio en bandas de frecuencia. La altura se corresponde con la intensidad de cada frecuencia en la señal.



FFT

```
import processing.sound.*;
FFT fft;
AudioIn in;
int bands = 512;
float[] spectrum = new float[bands];
void setup() {
 size(512, 360);
 colorMode(HSB);
 fft = new FFT(this, bands);
 in = new AudioIn(this, 0);
 in.start();
 fft.input(in);
}
void draw() {
 stroke(frameCount%255,255,255);
 fft.analyze(spectrum); // normaliza los valores (0-1)
 for(int i = 0; i < bands; i++){
  line( i, height, i, height - spectrum[i]*height*5);
 }
}
```



Archivos de audio -> SoundFile

Formatos: WAV, AIF/AIFF, MP3

frames() // Devuelve el número frames/samples del archivo.

sampleRate() // Devuelve la velocidad de muestreo del archivo.

channels() // Devuelve el número de canales.

duration() // Devuelve la duración.

play() // Empieza a reproducir (una vez sola) el archivo.

loop() // Empieza a reproducir en loop.

jump() // Salta a la posición especificada (en segundos) y empieza a reproducir el audio.

SoundFile

cue() // Salta a la posición especificada (en segundos, int) y NO reproduce el audio.

pan() // Mueve la fuente de sonido en estéreo (solo con archivos Mono!). Va de -1.0 (izquierda) a 1.0 (derecha).

rate() // Cambia la velocidad de la reproducción.

amp() // Cambia el volumen. Recibe valores entre 0.0 y 1.0.

add() // Offset the output of the player by given value

set() // Setea rate, pan, amp y add de una.

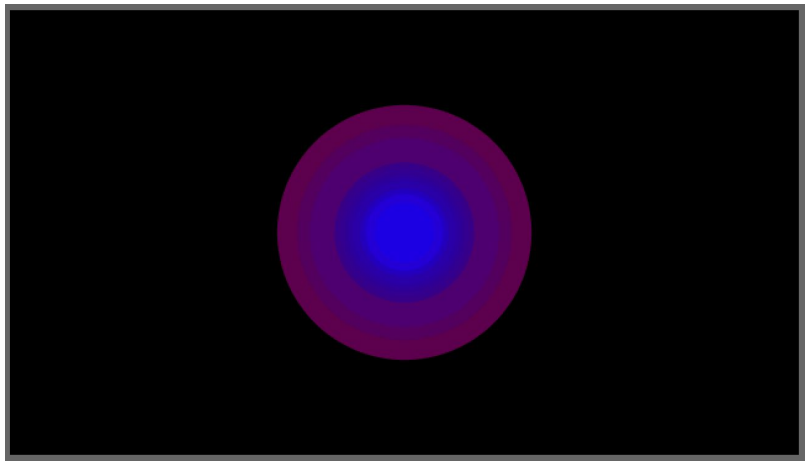
stop() // detiene la reproducción

Cambio de la velocidad de la reproducción

```
import processing.sound.*;
SoundFile file;

void setup() {
  size(640, 360);
  file = new SoundFile(this, "yujuM.mp3");
  file.play();
  file.loop();
}

void draw() {
  file.rate(map(mouseX,0,width,0.5,3));
}
```

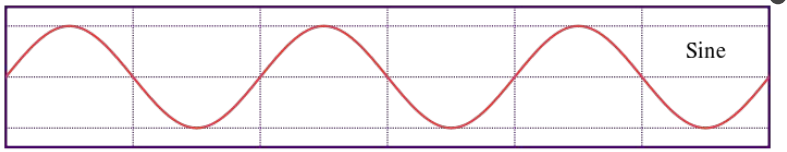
Ejemplo: tamaño y volumen (mp3)

```
import processing.sound.*;
Amplitude amp;
SoundFile in;
void setup() {
  size(640, 360);
  noStroke();
  in = new SoundFile(this,"jazz.mp3");// entrada de audio -> mp3
  amp = new Amplitude(this); // para analizar el volumen
  in.loop(); // empiezo a reproducir
  amp.input(in); // defino que debe analizar
  background(0); //empezamos con fondo negro
}
void draw() {
  fill(0,4);
  rect(0,0,width,height);
  float widthHeight = map(amp.analyze(),0,1,10,height); // mapeo el volumen (0-1) al tamaño (10-height)
  fill(lerpColor(color(0, 0, 255),  color(255, 0, 0), amp.analyze()));  // mapeo de azul a rojo
  ellipse(width/2,height/2, widthHeight, widthHeight);
}
```

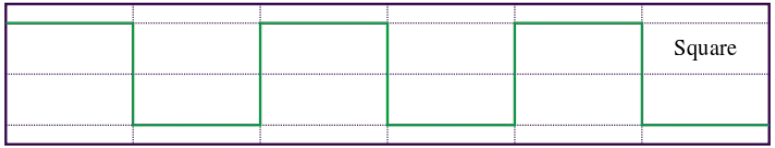


Osciladores

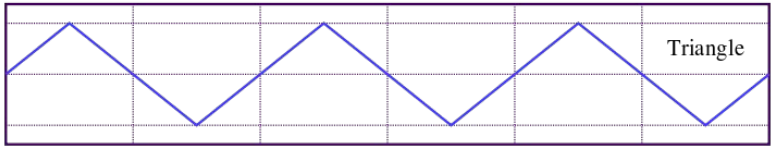
SinOsc - Oscilador de onda sinusoidal.



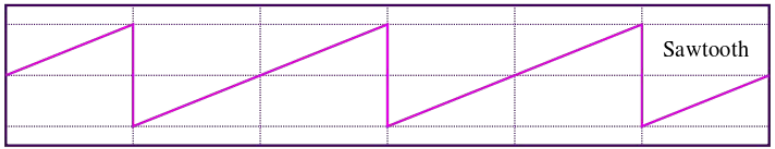
SqrOsc - Oscilador de onda cuadrada.



TriOsc - Oscilador de onda triangular.



SawOsc - Oscilador de onda de sierra.



Pulse - Oscilador de onda de pulso.

SinOsc, SawOsc, SqrOsc, TriOsc

play() // Arranca el oscilador.

freq() // Setea la frecuencia del oscillator en Hz.

amp() // Setea el volumen, recibe valores entre 0.0 y 1.0.

add() Offset the output of the oscillator by given value

pan() // Mueve la fuente de sonido en stereo. Va de -1.0 (izquierda) a 1.0 (derecha).

set() // Setea freq, amp, add, pan a la vez.

stop() // Detiene el oscilador.

Pulse

Las funciones mencionados en la transparencia anterior

+

width() // Setea el ancho del pulso.

Crear un oscilador de onda sinusoidal

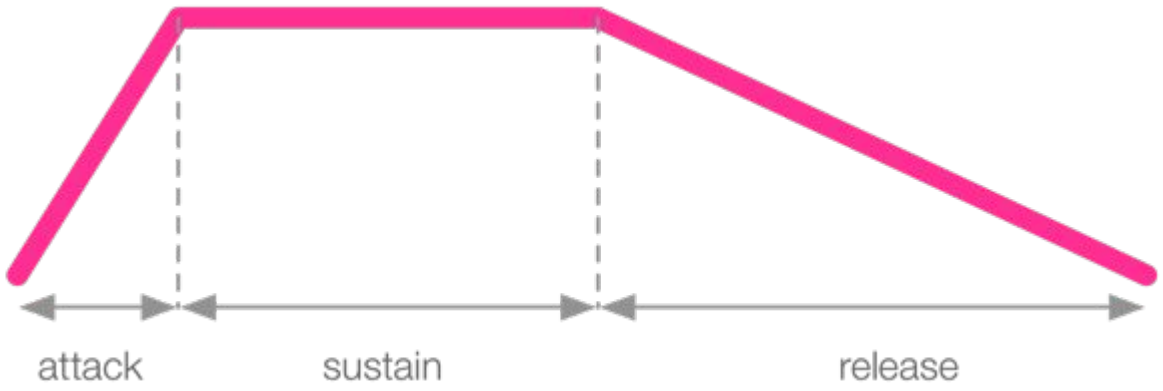
```
import processing.sound.*;
SinOsc sine;

void setup() {
  size(640, 360);
  sine = new SinOsc(this);
  sine.play();
}

void draw() {
  sine.freq(map(mouseX,0,width,200,2000));
}
```

Cambio de frecuencia

Env

```
Env.play(input, attackTime, sustainTime, sustainLevel, releaseTime)
```

```
import processing.sound.*;
SinOsc sine;
Env env;
void setup() {
  size(640, 360);
  sine = new SinOsc(this);
  env  = new Env(this);
}
void draw() {
}
void mousePressed() {
  sine.play();
  env.play(sine, 0.001, 0.004, 0.3, 0.4);
}
```



Ruidos, filtros y efectos

Ruidos: WhiteNoise, PinkNoise, BrownNoise

Filtros: BandPass, LowPass, HighPass

Efectos: Delay, Reverb

Ruido blanco

```
import processing.sound.*;
WhiteNoise noise;
void setup() {
  noise = new WhiteNoise(this);
  noise.play();
}

void draw() {
}
```

Filtros

LowPass -> lowPass.process(noise, 800);

HighPass -> highPass.process(noise, 5000);

BandPass -> bandPass.process(noise, 100, 50);

Retraso = Delay, repetición = feedback

```
import processing.sound.*;
AudioIn in;
Delay delay;
void setup() {
  in = new AudioIn(this, 0);
  delay = new Delay(this);
  in.play(); // arranca el microfono
  delay.process(in,1); // el retraso va a ser de 1 segundo
  delay.feedback(0.66); // el feedback va disminuyendo x 0.66
}
void draw() {
}
```

Reverberación = Reverb

```
import processing.sound.*;
AudioIn in;
Reverb reverb;

void setup() {
  in = new AudioIn(this, 0);
  reverb = new Reverb(this);
  in.play();
  reverb.process(in);
  //posibles modificadores: reverb.wet(0.5); reverb.room(0.5); reverb.damp(0.5);
}
void draw() {
}
```

Minim

http://code.compartmental.net/tools/minim/quickstart/

Reproducir audio

Acceder a la entrada de audio

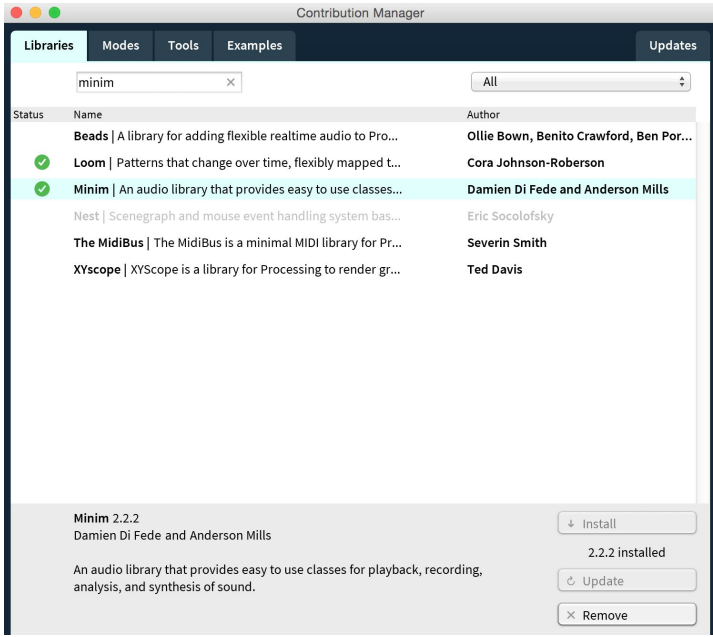
Salida audio

Grabar audio

Metadatos de archivos

Osciladores

Tocar notas http://code.compartmental.net/minim/audiooutput_method_playnote.html



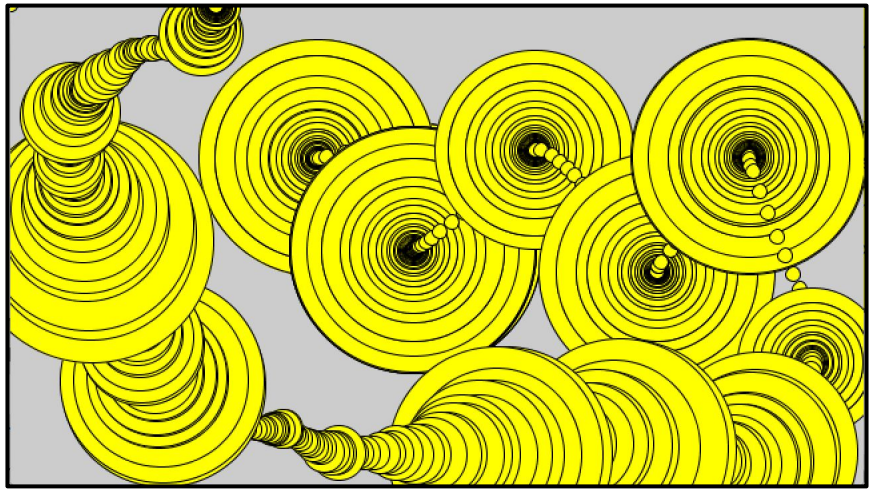
Resumiendo

Sabemos analizar la entrada de micrófono.

Sabemos reproducir y analizar archivos de audio.

Sabemos generar sonido y aplicarle efectos y filtros.

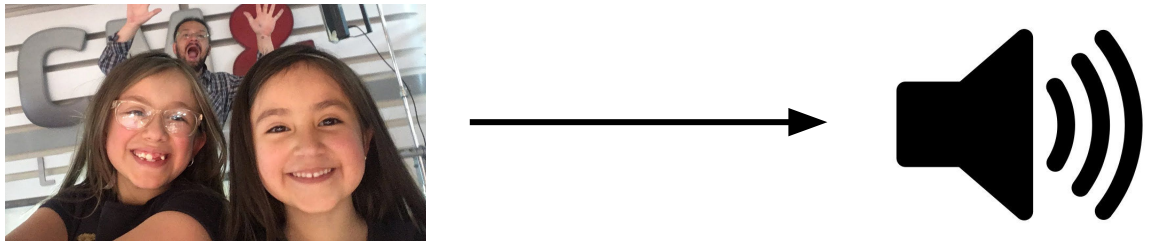
Sabemos que hace Minim.



Deberes - Entrega 2 - convertir vídeo a sonido

Mandar:

1. Sketch que en función de la **imágen captada** por la cámara (¿Cantidad de píxeles blancos?, ¿Cantidad de movimiento?, ¿Posición de la cara detectada?) **genera sonido** (¿Cambios en los valores de un oscilador?, ¿Cambio en velocidad de la reproducción de un archivo de audio?, ¿Reproducción de un sonido particular con Env?, ¿Suena una nota particular?, ...).
2. Un **vídeo** (puede ser grabado con celular) que muestra el funcionamiento del sketch. En el vídeo debería verse lo que capta la cámara y debería escucharse el sonido generado.



Fecha de la entrega: **06.03.22**, 23:59 hora Uruguay, por Eva