Java sonido

import ddf.minim.\*;

Minim minim;

AudioInput audioInput;

int numCircles = 4;

float[] volumes = new float[numCircles];

float[] angles = {0, PI, PI/2, -PI/2}; // front, back, right, left

float radius = 50;

void setup() {

size(800, 600);

minim = new Minim(this);

audioInput = minim.getLineIn(Minim.STEREO);

}

void draw() {

background(0);

// get the audio input levels

volumes[0] = audioInput.left.level(); // front

volumes[1] = audioInput.right.level(); // back

volumes[2] = (audioInput.left.level() + audioInput.right.level()) / 2; // right

volumes[3] = (audioInput.left.level() + audioInput.right.level()) / 2; // left

// constrain the volumes to be between 0 and 1

for (int i = 0; i < numCircles; i++) {

volumes[i] = constrain(volumes[i], 0, 1);

}

// draw the circles

translate(width/2, height/2);

noStroke();

for (int i = 0; i < numCircles; i++) {

fill(255);

float r = radius \* volumes[i];

float x = r \* cos(angles[i]);

float y = r \* sin(angles[i]);

ellipse(x, y, r\*2, r\*2);

}

}

void stop() {

audioInput.close();

minim.stop();

super.stop();

}

Anotaciones:

solo presenta un sólo sensor y detecta bien los rangos de sonidos

import ddf.minim.\*;

Minim minim;

AudioInput audioInput;

int numCircles = 4;

float[] volumes = new float[numCircles];

float[] angles = {0, PI, PI/2, -PI/2}; // front, back, right, left

float radius = 50;

float separation = 200; // separation between circles

void setup() {

size(800, 600);

minim = new Minim(this);

audioInput = minim.getLineIn(Minim.STEREO);

}

void draw() {

background(0);

// get the audio input levels

volumes[0] = audioInput.left.level(); // front

volumes[1] = audioInput.right.level(); // back

volumes[2] = (audioInput.left.level() + audioInput.right.level()) / 2; // right

volumes[3] = (audioInput.left.level() + audioInput.right.level()) / 2; // left

// constrain the volumes to be between 0 and 1

for (int i = 0; i < numCircles; i++) {

volumes[i] = constrain(volumes[i], 0, 1);

}

// draw the circles

translate(width/2, height/2);

noStroke();

for (int i = 0; i < numCircles; i++) {

fill(255);

float x = separation \* cos(angles[i]);

float y = separation \* sin(angles[i]);

ellipse(x, y, radius\*2, radius\*2);

// draw a small point at the center of each circle

fill(255, 0, 0); // red color

ellipse(x, y, 5, 5);

}

}

void stop() {

audioInput.close();

minim.stop();

super.stop();

}

Anotaciones:

Presenta los 4 sensores en las cuatro direcciones pero no detecta el ruido

import ddf.minim.\*;

Minim minim;

AudioInput audioInput;

int numCircles = 4;

float[] volumes = new float[numCircles];

float[] angles = {0, PI, PI/2, -PI/2}; // front, back, right, left

float radius = 50;

float separation = 200; // separation between circles

void setup() {

size(800, 600);

minim = new Minim(this);

audioInput = minim.getLineIn(Minim.STEREO);

}

void draw() {

background(0);

// get the audio input levels

volumes[0] = audioInput.left.level(); // front

volumes[1] = audioInput.right.level(); // back

volumes[2] = audioInput.left.level(); // right

volumes[3] = audioInput.right.level(); // left

// constrain the volumes to be between 0 and 1

for (int i = 0; i < numCircles; i++) {

volumes[i] = constrain(volumes[i], 0, 1);

}

// draw the circles

translate(width/2, height/2);

noStroke();

for (int i = 0; i < numCircles; i++) {

fill(255);

float x = separation \* cos(angles[i]);

float y = separation \* sin(angles[i]);

float r = radius \* volumes[i]; // radius proportional to volume

ellipse(x, y, r\*2, r\*2);

// draw a small point at the center of each circle

fill(255, 0, 0); // red color

ellipse(x, y, 5, 5);

}

}

void stop() {

audioInput.close();

minim.stop();

super.stop();

}

Anotación:

Presenta los 4 sensores y las cuatro direcciones pero detecta solo sonidos fuertes y reaccionan todas al mismo tiempo (no sabemos si es por el sonido fuerte activa todos por estar cerca o si todos reaccionan juntos por la progra)

import ddf.minim.\*;

import ddf.minim.analysis.\*;

Minim minim;

AudioInput audioInput;

FFT fft;

int numCircles = 4;

float[] volumes = new float[numCircles];

float[] angles = {0, PI, PI/2, -PI/2}; // front, back, right, left

float radius = 50;

float separation = 200; // separation between circles

void setup() {

size(800, 600);

minim = new Minim(this);

audioInput = minim.getLineIn(Minim.STEREO, 1024);

fft = new FFT(audioInput.bufferSize(), audioInput.sampleRate());

}

void draw() {

background(0);

// analyze the audio input

fft.forward(audioInput.mix);

// get the volume for each direction

volumes[0] = getVolume(fft, 0, 100); // front

volumes[1] = getVolume(fft, 100, 200); // back

volumes[2] = getVolume(fft, 200, 300); // right

volumes[3] = getVolume(fft, 300, 400); // left

// constrain the volumes to be between 0 and 1

for (int i = 0; i < numCircles; i++) {

volumes[i] = constrain(volumes[i], 0, 1);

}

// draw the circles

translate(width/2, height/2);

noStroke();

for (int i = 0; i < numCircles; i++) {

fill(255);

float x = separation \* cos(angles[i]);

float y = separation \* sin(angles[i]);

float r = radius \* volumes[i]; // radius proportional to volume

ellipse(x, y, r\*2, r\*2);

// draw a small point at the center of each circle

fill(255, 0, 0); // red color

ellipse(x, y, 5, 5);

}

}

// get the volume for a given frequency range

float getVolume(FFT fft, int start, int end) {

float sum = 0;

for (int i = start; i < end; i++) {

sum += fft.getBand(i);

}

return sum / (end - start);

}

void stop() {

audioInput.close();

minim.stop();

super.stop();

}

Anotaciones