Processing
P07

1- Son

Cédric MASSART

Son

Gestion du son dans processing

- Deux solutions :
  - en avec OSC (Open Sound Control)
    - avec d'autre logiciel :
      - PureData (http://www.puredata.org/)
      - Max/MSP (http://www.cycling74.com/)
  - ou d'autres matériels
- ou directement avec la librairie 'sound'.

OSC : Format de transmission de données entre ordinateurs, synthétiseurs, robots ou tout autre matériel ou logiciel compatible, conçu pour le contrôle en temps réel. Il utilise le réseau au travers des protocoles UDP ou TCP et apporte des améliorations en termes de rapidité et flexibilité par rapport à l'ancienne norme MIDI.

Exemple de Contrôleur OSC (phone/tablet)

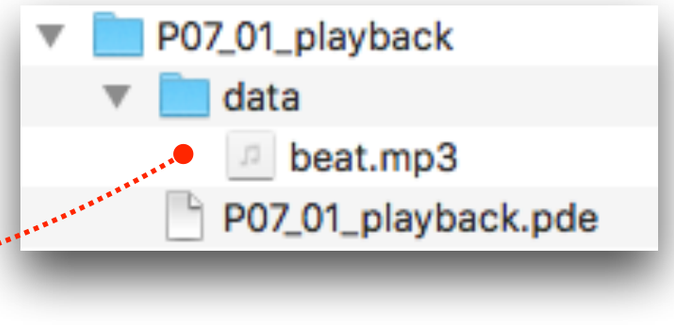
TouchOSC

http://hexler.net/software/touchosc

http://hexler.net/software/touchosc-android

Lecture d'un fichier son

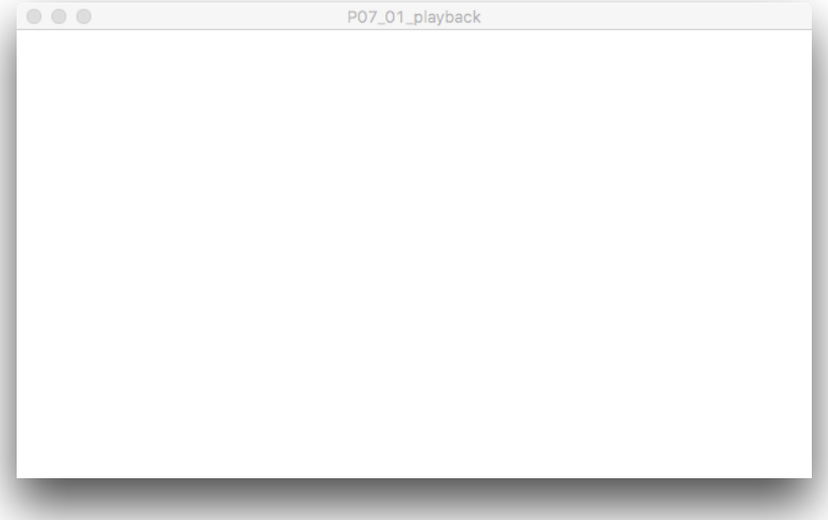
```
import processing.sound.*;

SoundFile song;

void setup() {
  size(640, 360);
  song = new SoundFile(this, "beat.mp3");
  song.play();
}

void draw() {
  background(255);
  noLoop();
}

boolean playing = true;

void mousePressed() {
  if (playing) {
    song.stop();
    playing = false;
  } else {
    song.play();
    playing = true;
  }
}
```



chargement et lecture du fichier son

draw ne boucle pas

contrôle de l'arrêt et la lecture du fichier son avec la souris



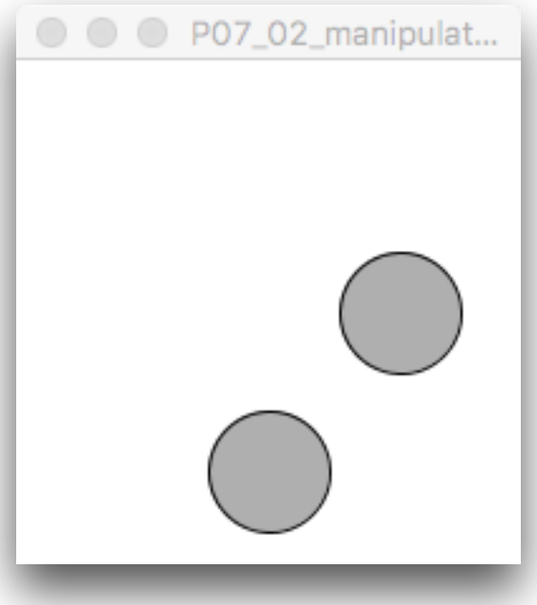
Contrôle d'un fichier son

```
import processing.sound.*;

SoundFile song;

void setup() {
  size(200, 200);

  song = new SoundFile(this, "beat.mp3");
  song.loop();
}

void draw() {
  background(255);

  float volume = map(mouseX, 0, width, 0, 1);
  song.amp(volume);

  float speed = map(mouseY, 0, height, 0, 2);
  song.rate(speed);

  stroke(0);
  fill(51, 100);
  ellipse(mouseX, 100, 48, 48);
  stroke(0);
  fill(51, 100);
  ellipse(100, mouseY, 48, 48);
}
```

gestion du volume sonore

vitesse de lecture

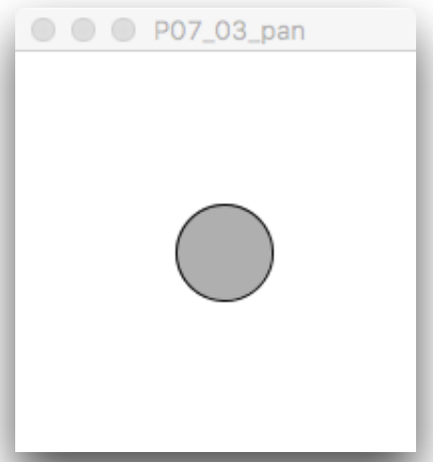
retour graphique



P07_02_manipulate_sound.pde

Contrôle du Panoramique

```
import processing.sound.*;

SoundFile soundFile;

void setup() {
  size(200, 200);
  soundFile = new SoundFile(this, "beat.mp3");
  soundFile.loop();
}

void draw() {
  background(255);
  float panning = map(mouseX, 0., width, -1.0, 1.0);
  soundFile.pan(panning);

  stroke(0);
  fill(51, 100);
  ellipse(mouseX, 100, 48, 48);
}
```



conversation des coordonnées souris x de -1.0 à 1.0 suivant la largeur de la fenêtre

retour graphique

Réverbération

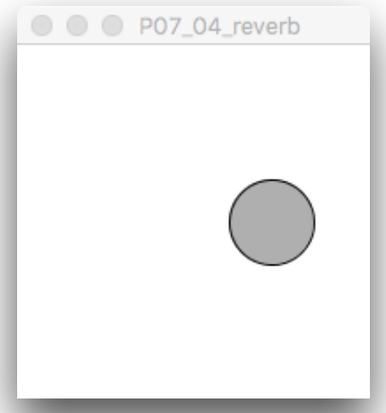
```
import processing.sound.*;

SoundFile song;
Reverb reverb;

void setup() {
  size(200, 200);

  song = new SoundFile(this, "beat.mp3");
  song.loop();

  reverb = new Reverb(this);
  reverb.process(song);
}

void draw() {
  background(255);

  float room = map(mouseX, 0, width, 0, 1);
  reverb.room(room);

  stroke(0);
  fill(51, 100);
  ellipse(mouseX, 100, 48, 48);
}
```

création de l'objet « Reverb »

association du fichier son
à l'objet « Reverb »

conversation des coordonnées
souris x de 0 à 1 suivant
la largeur de la fenêtre

retour graphique



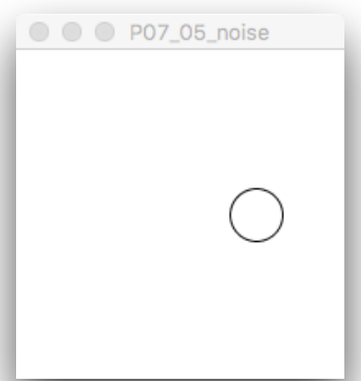
Bruit

```
import processing.sound.*;

WhiteNoise noise;

void setup() {
  size(200, 200);
  noise = new WhiteNoise(this);
  noise.play();
}

void draw() {
  background(255);

  float vol = map(mouseX, 0, width, 0, 1);
  noise.amp(vol);
  ellipse(mouseX, 100, 32, 32);
}
```

création de l'objet « WhiteNoise »

conversation des coordonnées
souris x de 0 à 1 suivant
la largeur de la fenêtre

retour graphique



P07_05_noise.pde

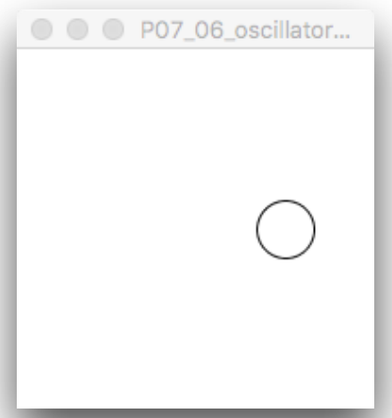
Oscillation de fréquence

```
import processing.sound.*;

SinOsc osc;

void setup() {
  size(200, 200);
  osc = new SinOsc(this);
  osc.play();
}

void draw() {
  background(255);

  float freq = map(mouseX, 0, width, 150, 880);
  osc.freq(freq);
  ellipse(mouseX, 100, 32, 32);
}
```

création de l'objet « SinOsc »

conversation des coordonnées souris x de 150 à 880 suivant la largeur de la fenêtre

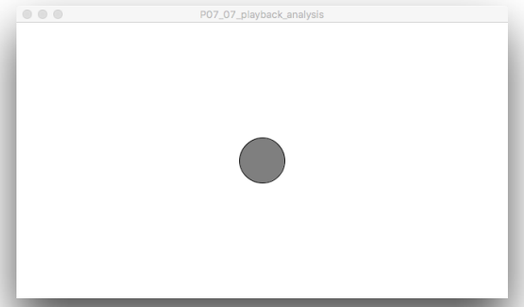
retour graphique



P07_06_oscillator_frequency.pde

Lecture du volume

```
import processing.sound.*;

SoundFile song;
Amplitude analyzer;

void setup() {
  size(640, 360);
  song = new SoundFile(this, "beat.mp3");
  song.loop();

  analyzer = new Amplitude(this);
  analyzer.input(song);
}

void draw() {
  background(255);

  float vol = analyzer.analyze();
  fill(127);
  stroke(0);

  ellipse(width/2, height/2, 10+vol*200, 10+vol*200);
}
```



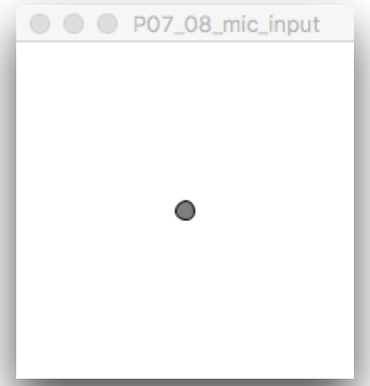
création de l'objet « Amplitude »

association du fichier son
à l'objet « Reverb »

lecture du volume sonore

retour graphique

Lecture du micro




```
import processing.sound.*;

AudioIn input;
Amplitude analyzer;

void setup() {
  size(200, 200);

  input = new AudioIn(this, 0);

  input.start();

  analyzer = new Amplitude(this);
  analyzer.input(input);
}

void draw() {
  background(255);

  float vol = analyzer.analyze();
  fill(127);
  stroke(0);

  ellipse(width/2, height/2, 10+vol*200, 10+vol*200);
}
```

création de l'objet « Audio Line »

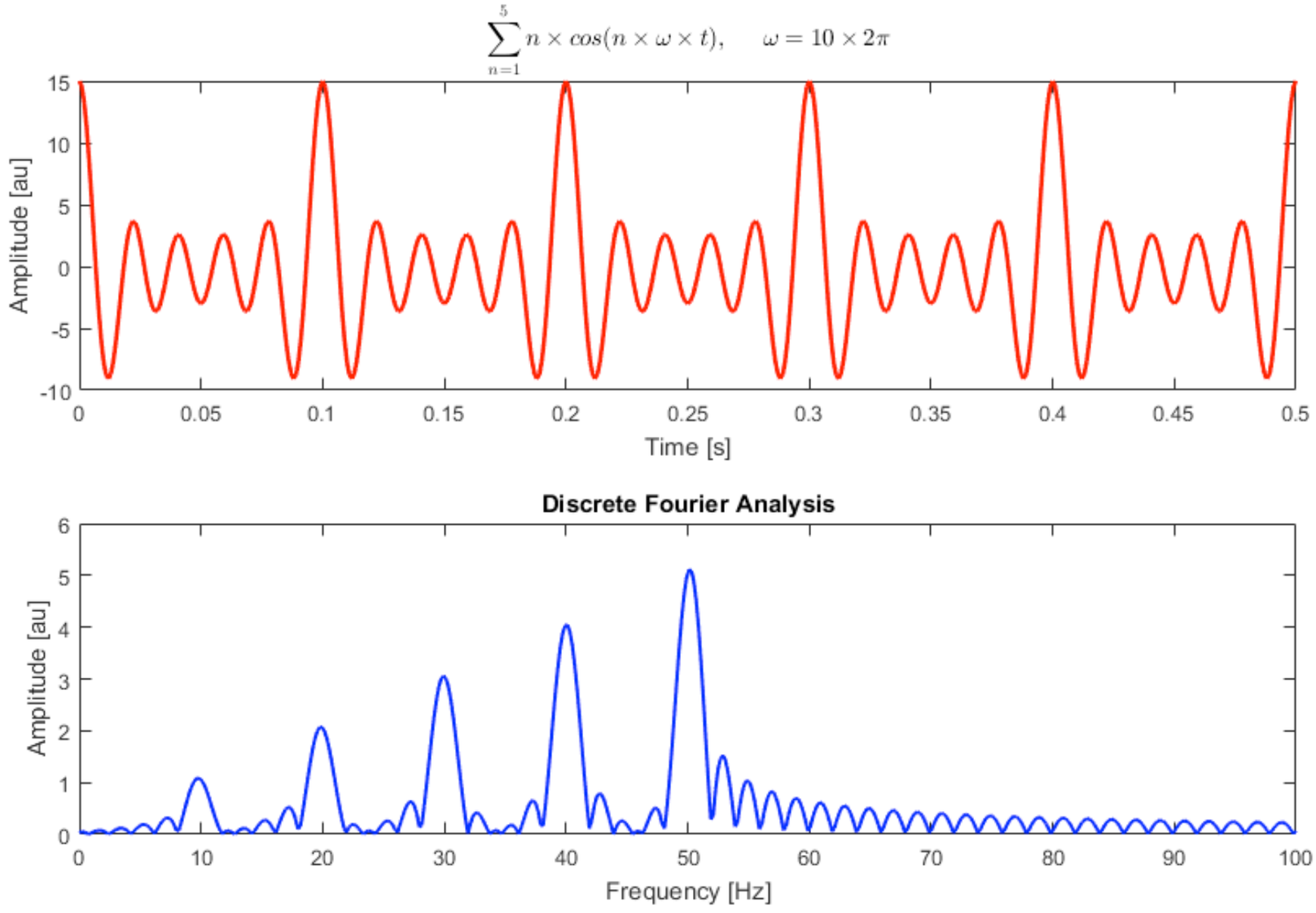
création de l'objet « Amplitude »

association du fichier son
à l'objet « Reverb »

lecture du volume sonore

retour graphique

Transformation de Fourier rapide (FFT)



Transformation de Fourier rapide (FFT)

```
import processing.sound.*;

SoundFile song;

FFT fft;
int bands = 512;

void setup() {
  size(512, 360);

  song = new SoundFile(this, "beat.mp3");
  song.loop();

  fft = new FFT(this, bands);
  fft.input(song);
}

void draw() {
  background(255);

  fft.analyze();

  for (int i = 0; i < fft.size(); i++) {
    stroke(0);
    float y = map(fft.spectrum[i], 0, 1, height * 0.75, 0);
    line(i, height * 0.75, i, y);
  }
}
```

nombre de bandes

création de l'objet « Audio Line »

analyse FFT

affichage

P07_09_soundfile_FFT.pde

Transformation de Fourier rapide (FFT) v1

```
import processing.sound.*;

SoundFile song;

FFT fft;
int bands = 512;

void setup() {
  size(512, 360);

  song = new SoundFile(this, "beat.mp3");
  song.loop();

  fft = new FFT(this, bands);
  fft.input(song);
}

void draw() {
  background(255);

  fft.analyze();

  for (int i = 0; i < fft.size(); i++) {
    stroke(0);
    float y = map(fft.spectrum[i], 0, 1, height * 0.75, 0);
    line(i, height * 0.75, i, y);
  }
}
```

nombre de bandes

création de l'objet « FFT »

analyse FFT

affichage

P07_09_soundfile_FFT.pde

Transformation de Fourier rapide (FFT) v2

```
import processing.sound.*;

AudioIn input;
FFT fft;
int bands=512;
float[] spectrum = new float[bands];

void setup() {
  size(512, 360);

  input = new AudioIn(this, 0);

  input.play();

  fft = new FFT(this, bands);
  fft.input(input);
}

void draw() {
  background(255);

  fft.analyze(spectrum);

  for (int i = 0; i < spectrum.length; i++) {
    stroke(0);
    float y = map(spectrum[i], 0, 1, height * 0.75, 0);
    line(i, height * 0.75, i, y);
  }
}
```

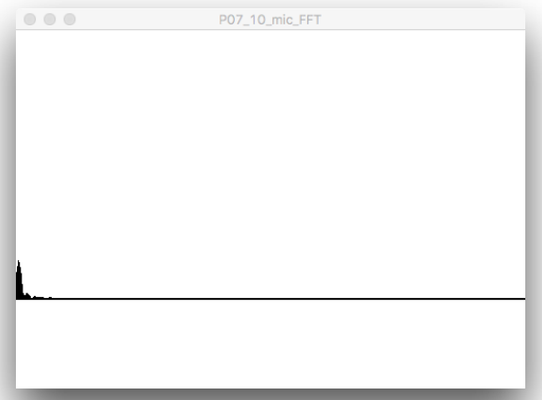
nombre de bandes

création de l'objet « Audio Line »

création de l'objet « FFT »

analyse FFT

affichage



P07_09_soundfile_FFT.pde

Transformation de Fourier rapide (FFT) v3

```
import processing.sound.*;

SoundFile song;

FFT fft;

void setup() {
  size(514, 360);

  song = new SoundFile(this, "beat.mp3");
  song.loop();

  fft = new FFT(this, 64);
  fft.input(song);
}

void draw() {

  fft.analyze();

  float w = width / (fft.size()/2);

  background(255);
  for (int i = 0; i < fft.size()/2; i++) {
    stroke(0);
    float x = i * w;
    float h = map(fft.spectrum[i], 0, 1, 0, height);
    //fill(i*20 % 255);
    rect(x+2, height - h - 2, w-2, h);
  }
}
```

nombre de bandes

création de l'objet « FFT »

analyse FFT

affichage

P07_11_graphic_equalizer.pde