audio, processing, chuck, em

Séquences étranges

(Notes... 1er juin 2020)

Jouets sonores.

séquence étrange trois

Boucles de samples type granulaire : à partir du même fichier sonore, boucler des petits fragments rapidement.

En vidéo, ça donne

exemple vidéo

code

Un script processing pour l'interface graphique et un script chuck pour le granulateur, les deux communiquent par OSC

sequence_etrange_trois_001.pde (cliquer pour afficher le code)

sequence_etrange_trois_001.pde

```
/*
  Discussions entre processing et chuck
  Lire en boucle plusieurs grains de sons d'un même fichier
 Quimper, Dour Ru, 20200601 / pierre@lesporteslogiques.net
 Processing 3.5.3 + ControlP5 2.2.6 + OscP5 0.9.9
 chuck version: 1.4.0.1 linux (pulse) 64-bit
 @ kirin / Debian Stretch 9.5
*/

import oscP5.*;
import netP5.*;
OscP5 oscP5;
NetAddress myRemoteLocation;

ArrayList<Grain> grains = new ArrayList<Grain>();

int id = 0;
float duree_morceau = 58.16; // en secondes
float dur;

PImage wave;

void setup() {
  size (800, 400);
  oscP5 = new OscP5(this, 12000);
  myRemoteLocation = new NetAddress("127.0.0.1", 12012);
  wave = loadImage("wf.png");
}

void draw() {

  background(0);
  image(wave,0,0);

  // Montrer la position et le fragment bouclé
  fill(128, 50);
  stroke(128, 50);
  strokeWeight(0.5);
  dur = (((duree_morceau * 44100) / width) / ((mouseY / (float)height) * (3 * 44100)));
  float pixels_par_seconde = (float)width / duree_morceau; // x pixel = 1 seconde
  float duree_extrait = (1 - (mouseY / (float)height)) * 3 ; // 3 secondes max
  dur = pixels_par_seconde * duree_extrait;

  line(0, mouseY, width, mouseY);
  rect(mouseX, 0, dur, height);
  for (int i = grains.size() - 1; i >= 0; i--) {
    Grain g = grains.get(i);
    g.miseajour();
```

```
  }
}

void keyPressed() {
  if (key == 'p') ajouterGrain();
  if (key == 'd') supprimerGrain(mouseX, mouseY);
  if (key == 'c') resetGrain();
}

void ajouterGrain() {
  id ++;
  if (id < 100) {
    grains.add( new Grain(mouseX, mouseY, id) );

    OscMessage myMessage = new OscMessage("/sequence_etrange_trois");
    float xn = mouseX / (float)width;  // normaliser les valeurs
    float yn = mouseY / (float)height; // normaliser les valeurs
    myMessage.add(1);
    myMessage.add(id);
    myMessage.add(xn);
    myMessage.add(1 - yn); // inverser, c'est plus intuitif, le zéro est en bas!
    oscP5.send(myMessage, myRemoteLocation);
  } else {
    background(255, 0, 0);
    println("trop de fragments!");
  }
}

void supprimerGrain(float x, float y) {
  for (int i = grains.size() - 1; i >= 0; i--) {
    Grain g = grains.get(i);
    if (dist(x, y, g.x, g.y) < 6) {
      grains.remove(i);
      OscMessage myMessage = new OscMessage("/sequence_etrange_trois");
      myMessage.add(0);
      myMessage.add(g.id);
      myMessage.add(0);
      myMessage.add(0);
      oscP5.send(myMessage, myRemoteLocation);
    }
  }
}

void resetGrain() {
  for (int i = grains.size() - 1; i >= 0; i--) {
    Grain g = grains.get(i);
    grains.remove(i);
    OscMessage myMessage = new OscMessage("/sequence_etrange_trois");
    myMessage.add(0);
    myMessage.add(g.id);
    myMessage.add(0);
    myMessage.add(0);
    oscP5.send(myMessage, myRemoteLocation);
  }
  id = 0;
}

class Grain {

  float x, y;                       // coordonnées à l'écran (en pixels)
  int id;

  Grain(float x_, float y_, int id_) {
    x = x_;
    y = y_;
    id = id_;
  }

  void miseajour() {
    noFill();
    stroke(255, 200);
    strokeWeight(1);
    line(x, y-5, x, y+5);
    line(x-5, y, x+5, y);
  }
}
```

sequence_etrange_trois_recepteur.ck (cliquer pour afficher le code)

```
// Debian 9.5 @ kirin
// Chuck 1.4.0.1 linux (pulse) 64-bit
// 20200601 / pierre@lesporteslogiques.net

OscIn oin;             // définir le récepteur OSC
12012 => oin.port;     // port de réception
OscMsg msg;

oin.addAddress( "/sequence_etrange_trois" );

int fragments[100]; // Pour stocker les id et les numéros de shred...
```

```
while(true) {
  oin => now;

  while (oin.recv(msg) != 0) {
    msg.getInt(0)   => int action;
    msg.getInt(1)   => int index;
    msg.getFloat(2) => float start;
    msg.getFloat(3) => float dur;

    if (action == 0) { // Supprimer le shred associé à cet id
      <<< "supprimer index " , index, "fragments : ", fragments[index] >>>;
      Machine.remove(fragments[index]);
    }

    if (action == 1) { // Démarrer un nouveau shred
      Shred s;
      spork ~ grain(start, dur) @=> s;
      <<< "Index : " , index, " > nouveau shred : " , s.id() >>>;
      s.id() => fragments[index];
    }
  }
}

fun void grain(float start, float dur) {

  SndBuf gra => dac;
  me.dir() + "/kabuki_sound_effects.wav" => gra.read;
  while(1) {

    (dur * 3 * 44100) $ int => int gradur;
    (start * gra.samples()) $ int => gra.pos;
    if (gra.pos() < 0) 0 $ int => gra.pos;
    gradur :: samp => now;
  }
}
```

petites choses utiles