Matérialité du Son

"Le son est une vibration mécanique d'un fluide, qui se propage sous forme d'ondes longitudinales grâce à la déformation élastique de ce fluide" Source Wikipedia

Dans l'air, elle est créée par une variation de pression émise par la source sonore, les particules d'air se mettent à osciller, mais tout ceci reste invisible à nos yeux.
Une représentation visuelle de ce phénomène serait la chute un objet verticale dans l'eau. Les vagues créées autour de ce point de chute se déplacent en s'éloignant de ce point, mais l'eau reste au même au endroit en ne se déplacent que verticalement et non en suivant les vagues.
Pour le cas des fluides, l'onde sonore étant longitudinale, la différence est juste qu'elle se déplace donc parallèlement à la direction de propagation.
La turbulence ainsi observer est la déformation, vibration du fluide de propagation de l'onde, variable suivant le milieu et la source d'émission.

Un de mes premier travaux en première année d'architecture a été de composer une mélodie, musique uniquement à partir de sections topographiques calqué sur une partition. La pente ainsi que la hauteur de la courbe donnant respectivement la durée et hauteur de la note.
Ce premier travail, qui m'avait beaucoup intéressé, mettant en scène la création de son à partir de visuels. Ma question pour ce développement serait donc l'inverse, cette fois-ci, comment produire un visuel à partir de son ? Comment représenter, quantifier visuellement ces émissions sonores ?

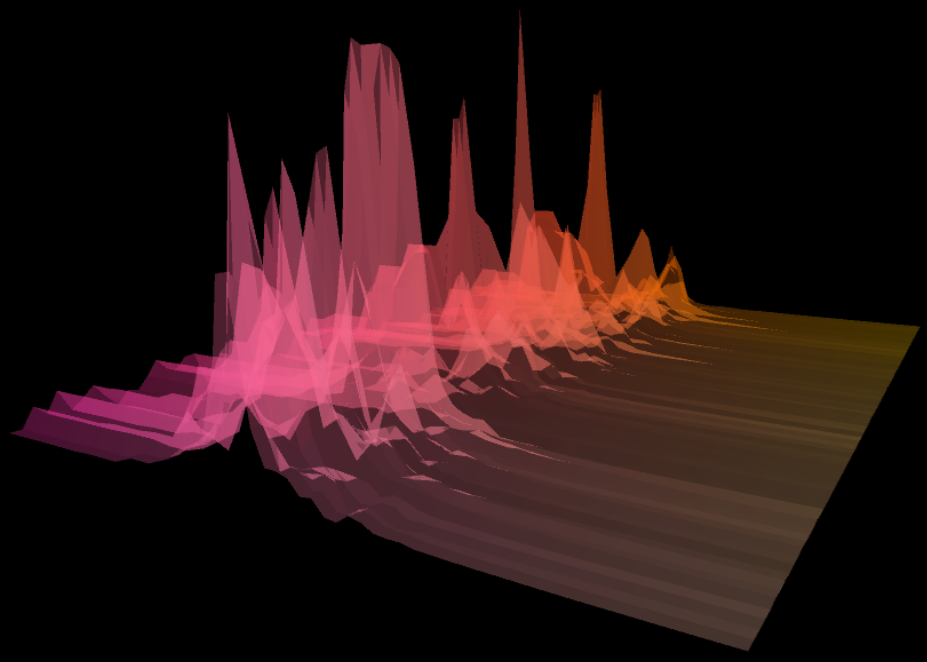
De nombreux modes de représentation visuels ont été développé en conséquence :
- la Partition : Temps (x) et Notes de gamme (y)
- le Spectrogramme : Temps (x) et Fréquences (y)
- le Sonagramme : Temps (x), Fréquences (y) et Intensités (couleur)
- la MAO : Interfaces graphiques (Ableton Live, FL Studio, Iannix, MAX/MSP)
- le Visualiseur : Animations (iTunes, Winamp, Mapping)

Ceci dit, toutes ces techniques de représentation précédemment citées ne sont que virtuelles, seul un affichage numérique ou analogique sur un écran ou support 2D permet de les voir.
Mon sujet est donc, comment réaliser un objet 3D représentatif du son, de manière la plus fidèle possible au niveau des informations audio (spectre de fréquences principalement) ?
L'approche se base sur une comparaison avec les instruments de musique mécaniques (automatophones), particulièrement les boites à musique et les orgues de barbarie.
La musique est ici créée à partir de supports physiques et matériels, picots sur un cylindre qui entrainent la vibration de lamelles métalliques pour la boite à musique, papier perforé ne laissant passer l'air que dans certains tuyaux pour de l'orgue de barbarie.

Mon travail se caractérise donc par un programme d'analyse sonore, permettant soit la sélection d'un fichier audio, soit d'un microphone comme source d'entrée, et affichant visuellement le défilement du spectre de fréquences (FFT) dans un environnement 3D (Temps x, Fréquence y et Intensité z).
Cette analyse génère ensuite un fichier géométrique 3D des 5 dernières secondes, puis une impression 3D vient donc finalement révéler cette représentation de manière physique et matérielle.
Cet objet 3D montre donc visuellement la perturbation et turbulences sonores invisibles dans l'air qui nous entoure. Il crée un paysage artificiel, une topographie éphémère, voire une ville imaginaire symbolisant ces perturbations, où la notion de temps et timeline disparait.
Différents sons sont analysés de domaines et bandes différentes, tels que les infrasons et ultrasons (inaudibles par l'oreille humaine), mais aussi certaines musiques ou bruits (audibles).

Références :
- http://rhizomesonore.free.fr/index.html
- http://www.home-of-penda.com/Home/Index/showWorkInfo?work_info_id=5ef097eef99c24b13ff4212cc2cf4581
- http://www.gillesazzaro.com/pages/en/printing3D.htm

VENOT Mathieu

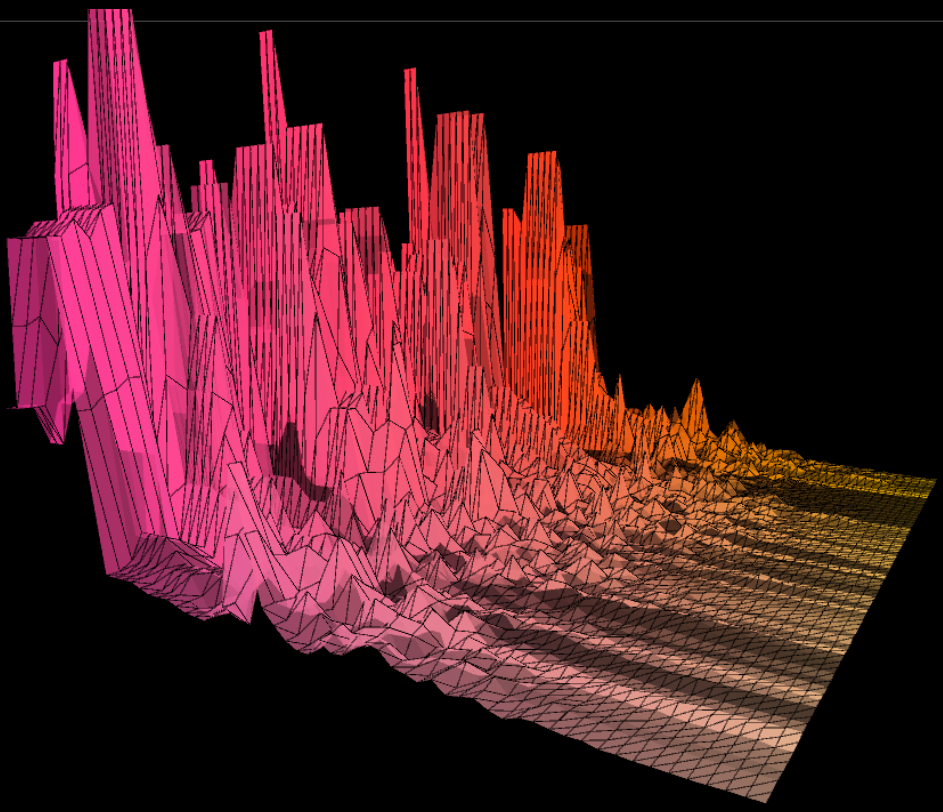


Line : Timeline Bar

0.65 SCALE

Slider : Intensity Scale
1/2/3 : Display Modes
S : Display Strokes
&/é/" : Strokes Modes
T : File Input
I : Micro Input
W : DXF Export
X : OBJ Export
- - - - -
Name : Endeavors.mp3
Length : 3min41s
Position : 0min8s
FPS : 18
Display Modes : 1
Strokes : false
Strokes Display : 1

// SOUND MATERIALITY
VENOT Mathieu



Line : Timeline Bar

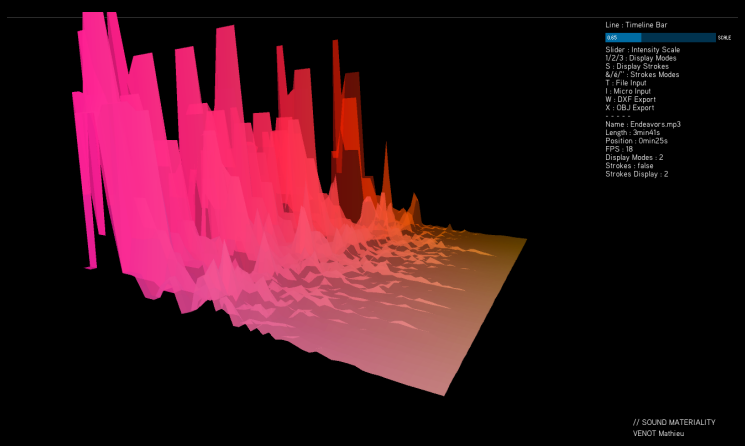
0.65 SCALE

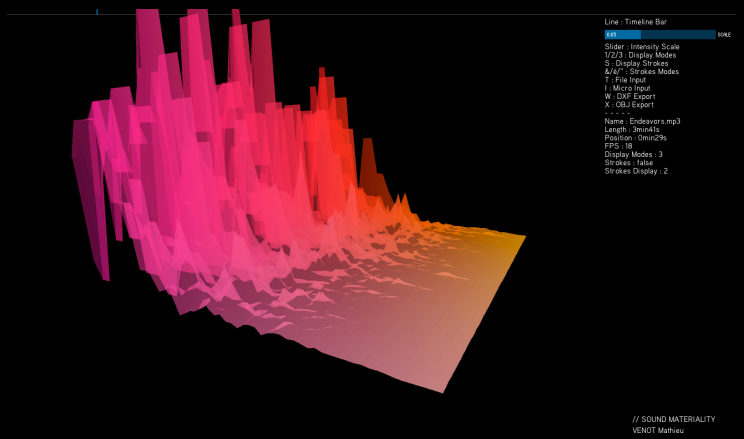
Slider : Intensity Scale
1/2/3 : Display Modes
S : Display Strokes
&/é/" : Strokes Modes
T : File Input
I : Micro Input
W : DXF Export
X : OBJ Export
- - - - -
Name : Endeavors.mp3
Length : 3min41s
Position : 0min37s
FPS : 15
Display Modes : 1
Strokes : true
Strokes Display : 1

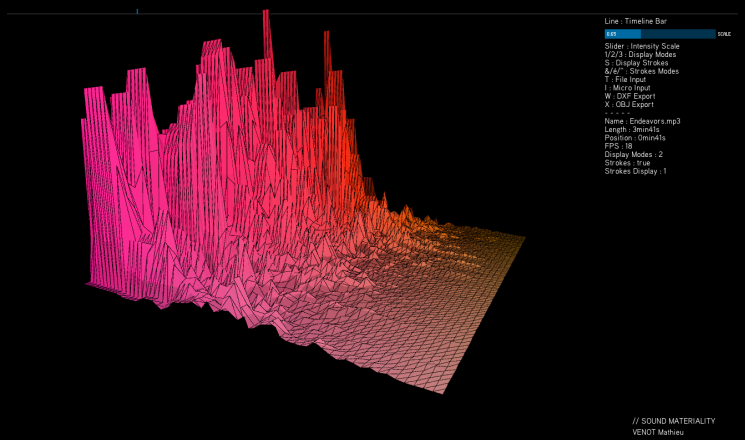
// SOUND MATERIALITY
VENOT Mathieu

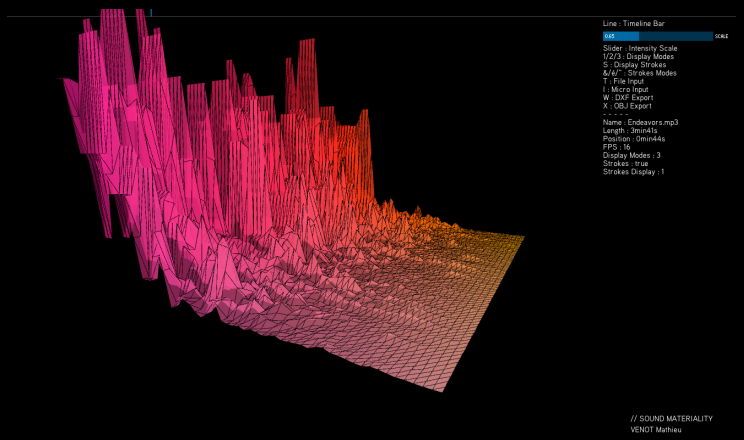
D8 - Turbulences

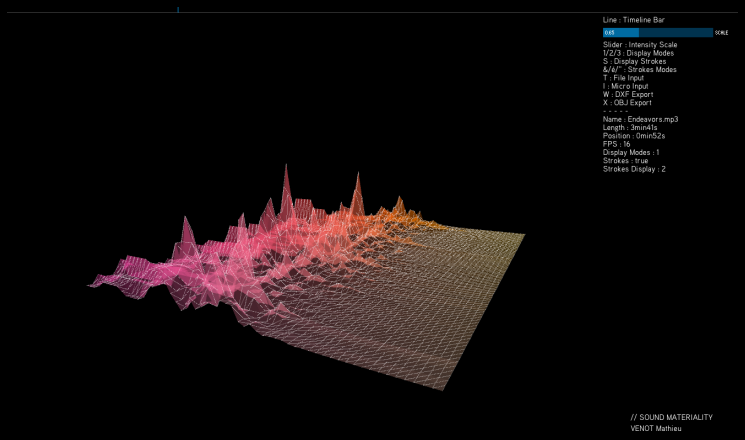
VENOT Mathieu

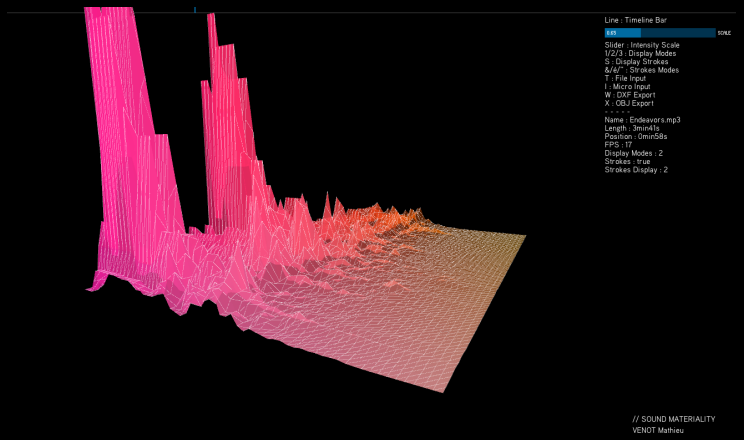


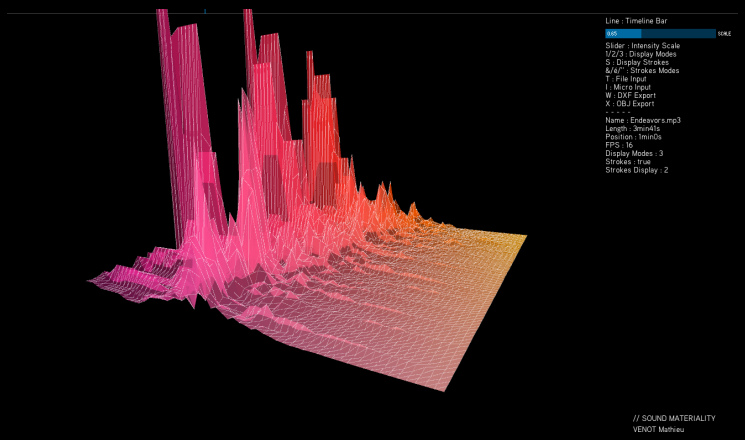


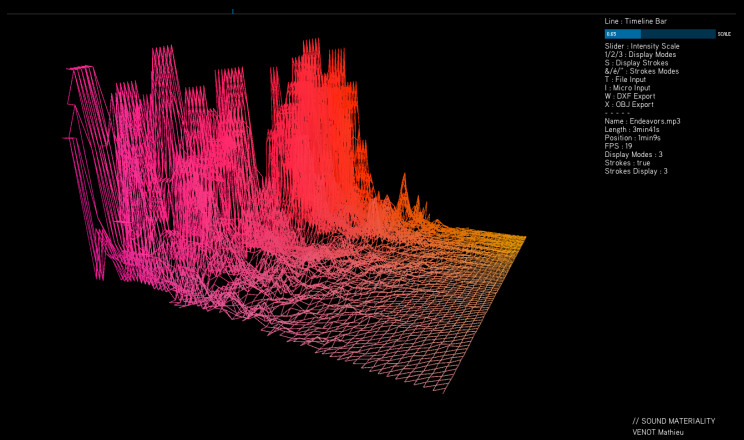




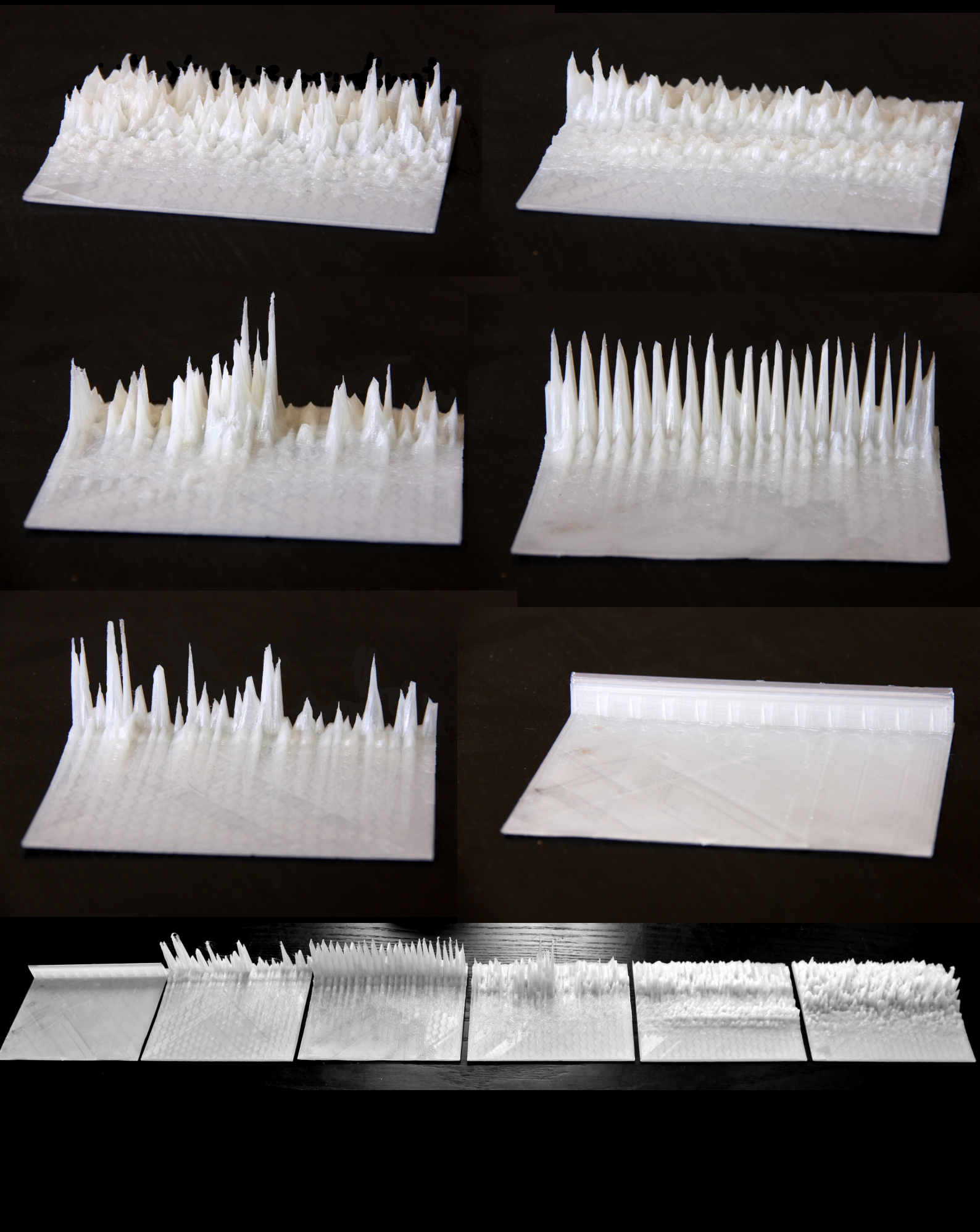








VENOT Mathieu



PROCESSING Code

```
/*
© 2015 VENOT Mathieu
*/

import controlP5.*;
import nervoussystem.obj.*;
import processing.dxf.*;
import ddf.minim.analysis.*;
import ddf.minim.*;


Minim minim;
AudioInput in;
AudioPlayer player;
AudioMetaData meta;
FFT fft;
ControlP5 cp5;
Slider sliders;


// Texts display
int ys = 15;
int yi = 15;
// Time conversion
int cmin, csec, pmin, psec;
// Switch
boolean Source;
boolean Lights = false;
boolean strokes = false;
boolean recordOBJ;
boolean recordDXF;
float scale = 1;
int Display = 1;
int strokesDisplay = 1;

float[] storedAvg = new float[63]; //63
float[] storedVols = new float[80]; //80

float[][] storedFFT = new float[80][63]; //80 63

PImage fade;

float volume = 0;
float combinedValues = 0;


void setup() {
  background(0);
  size(displayWidth-100, displayHeight-100,
P3D);
  frameRate(20);
  if (frame != null) {
    surface.setResizable(true);
  }
  smooth();

  cp5 = new ControlP5(this);

  cp5.addSlider("scale")
    .setPosition((width-250), ys+30)
      .setSize(200, 17)
        .setRange(0.00, 2.00)
          .setValue(1.00);
  cp5.getTooltip().register("scale", "Change the
FFT Scale");

  minim = new Minim(this);
  player = minim.loadFile("Endeavors.mp3",
512);
  meta = player.getMetaData();
  in = minim.getLineIn(Minim.STEREO, 512);
  fft = new FFT(in.bufferSize(),
in.sampleRate());

  fft.logAverages(60, 7);
  //fft.linAverages(63);

  textFont(loadFont("AauxProRegular-14.vlw"));

  fade = get(0, 0, width, height);
}


void draw() {
  background(0);


  pushStyle();
  stroke(255, 75);
  strokeWeight(0.75);
  line(0, 17.5, width, 17.5);
  popStyle();


  if (recordOBJ) {

beginRecord("nervoussystem.obj.OBJExport",
"mesh.obj");
  }
  if (recordDXF) {
    beginRaw(DXF, "mesh.dxf");
  }
```

PROCESSING Code (suite)

```
  if (Source == true) {
    fft.forward(player.mix);
    pushStyle();
    stroke( 0, 175, 255, 175 );
    strokeWeight(1.75);
    float position = map( player.position(), 0,
player.length(), 0, width );
    line( position, 0, position, 17.5 );
    popStyle();
  } else {
    fft.forward(in.mix);
  }

  combinedValues = 0;
  for (int i = 0; i < fft.avgSize (); i++) {
    combinedValues += fft.getAvg(i);
  }
  volume = combinedValues / fft.avgSize();
  if ( volume > 1 ) {
    for (int i = 0; i < fft.avgSize (); i++) {
      storedAvg[i] = fft.getAvg(i);
    }
    //store last 100 volumes over the threshold
    storedVols[storedVols.length-1] = volume;
    for (int i = 0; i < storedVols.length-1; i++) {
      storedVols[i] = storedVols[i+1];
    }
  }
  for (int i = 0; i < storedFFT[0].length; i++) {
    storedFFT[storedFFT.length-1][i] =
fft.getAvg(i);
  }
  for (int i = 0; i < storedFFT.length-1; i++) {
    for (int j = 0; j < storedFFT[0].length; j++) {
      storedFFT[i][j] = storedFFT[i+1][j];
    }
  }

  pushMatrix();
  translate(width/2-25, height/2-5, 425);
  rotateX(radians(169));
  rotateY(radians(117));

  for (int i = 0; i < storedFFT.length-1; i++) {
    for (int j = storedFFT[0].length-2; j >= 0; j--) {
      pushMatrix();
      translate(i*2, 0, j*2);
      float r = map(storedFFT[i][j], 0, 40, 225,
255);
      float b = map(i, 0, 80, 50, 175);
      float g = map(j, 0, 63, 50, 175);
      float c = map(storedFFT[i][0], 0, 100, 50,
255);
      //float aa = map(storedFFT[i][j],0,100,0,50);
      //box(2, storedFFT[i][j]*2, 2);
      if (strokes == false) {
        noStroke();
      } else if (strokes == true) {
        if (strokesDisplay == 1) {
          stroke(0);
          strokeWeight(0.75);
        } else if (strokesDisplay == 2) {
          stroke(255,127);
          strokeWeight(0.5);
        } else if (strokesDisplay == 3) {
          noFill();
          stroke(r, g-25, b-25, 225);
          strokeWeight(1.25);
        }
      }
      if (Display == 1 && strokesDisplay != 3) {
        fill(r, g, b-20, c+25);
      } else if (Display == 2 && strokesDisplay !=
3) {
        fill(r, g-25, b-25, b+50);
      } else if (Display == 3 && strokesDisplay !=
3) {
        fill(r, g-25, b-25, g+50);
      }

      beginShape();
      normal(0, 0, 1);
      vertex(0, storedFFT[i][j]*scale, 0);
      vertex(2, storedFFT[i+1][j]*scale, 0);
      vertex(0, storedFFT[i][j+1]*scale, 2);
      endShape();

      beginShape();
      normal(0, 0, 1);
      vertex(2, storedFFT[i+1][j+1]*scale, 2);
      vertex(0, storedFFT[i][j+1]*scale, 2);
      vertex(2, storedFFT[i+1][j]*scale, 0);
      endShape();

      popMatrix();
    }
  }

  // for (int i = 0; i < storedVols.length; i++) {
  //   pushMatrix();
  //   fill(255, 0, 0, 50);
  //   translate(i*2, 0, 140);
```

PROCESSING Code (suite)

```
  //   box(2, storedVols[i]*2, 10);
  //   popMatrix();
  // }
  popMatrix();


  if (recordDXF) {
    endRaw();
    recordDXF = false;
  }
  if (recordOBJ) {
    endRecord();
    recordOBJ = false;
  }


  // Texts display
  int y = ys+185+15;
  fill (255);
  //Controls
  text("Line : Timeline Bar", width-250, ys+20);
  text("Slider : Intensity Scale", width-250, ys
+65);
  text("1/2/3 : Display Modes", width-250, ys
+80);
  text("S : Display Strokes", width-250, ys+95);
  text("&/é/' ' : Strokes Modes", width-250, ys
+110);
  text("T : File Input", width-250, ys+125);
  text("I : Micro Input", width-250, ys+140);
  text("W : DXF Export", width-250, ys+155);
  text("X : OBJ Export", width-250, ys+170);
  text("- - - - -", width-250, ys+185);

  if (Source == true) {
    text("Name : " + meta.fileName(), width-250,
y);
    cmin = (meta.length()%(60*60*1000))/
(1000*60);
    csec = (((meta.length()%(1000*60*60))
%(1000*60))/(1000));
    pmin = (player.position()%(60*60*1000))/
(1000*60);
    psec = (((player.position()%(1000*60*60))
%(1000*60))/(1000));
    text("Length : " + cmin + "min" + csec + "s",
width-250, y+=yi);
    text("Position : " + pmin + "min" + psec + "s",
width-250, y+=yi);
    text("FPS : " + round(frameRate), width-250,
y+=yi);
  }
  text("Display Modes : " + Display, width-250, y
+=yi);
  text("Strokes : " + strokes, width-250, y+=yi);
  text("Strokes Display : " + strokesDisplay,
width-250, y+=yi);

  text("// SOUND MATERIALITY", width-200,
height-45);
  text("VENOT Mathieu", width-200, height-25);
}


void keyReleased() {
  if ( key == 't' || key == 'T')
  {
    Source = true;
    if (Source == true) {
      player.play();
    }
  }

  if ( key == 'i' || key == 'I' )
  {
    Source = false;
    player.pause();
  }

  if (key == '1') {
    Display = 1;
  } else if (key == '2') {
    Display = 2;
  } else if (key == '3') {
    Display = 3;
  }

  if (key == '&') {
    strokesDisplay = 1;
  } else if (key == 'é') {
    strokesDisplay = 2;
  } else if (key == '"') {
    strokesDisplay = 3;
  }

  if (key == 's' || key == 'S') {
    strokes =! strokes;
  }

  if (key == 'x' || key == 'X') {
    recordOBJ = true;
    println("Export OBJ : " + recordOBJ);
```

PROCESSING Code (suite)

```
  }
  if (key == 'w' || key == 'W') {
    recordDXF = true;
    println("Export DXF : " + recordDXF);
  }
}


void mousePressed()
{
  if (Source == true) {
    if (mouseY > 0 && mouseY < 17.5) {
      int position = int( map( mouseX, 0, width, 0,
player.length() ) );
      player.cue( position );
    }
  }
}


void stop() {
  minim.stop();
}
```