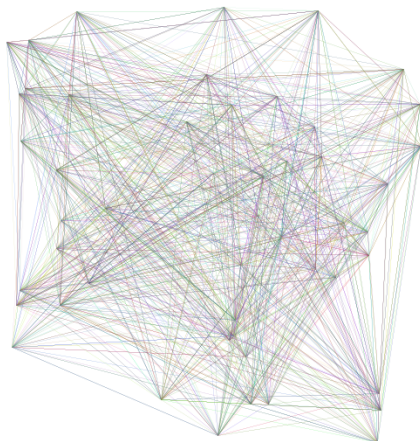


[processing](#)

# Ateliers Processing de l'OA

## Sketch 01



L'idée de ce premier atelier était d'implémenter un algorithme d'art contemporain proposé par [Sol LeWitt](#).

L'idée a été trouvée sur le site de [Pol Guezennec](#)

Bon, c'est vrai qu'on commence sur des chapeaux de roues, avec l'utilisation des boucles `for` et des listes, mais j'essaierai de garder un niveau de complexité constant, afin de ne pas pénaliser ceux-elles qui raccrocheraient le wagon en cours d'année. Si ce premier sketch vous semble compliqué (et il l'est lorsqu'on débute), les suivants devraient vous paraître de plus en plus simples, à force de répétition.

```
FloatList liste_x = new FloatList();
FloatList liste_y = new FloatList();

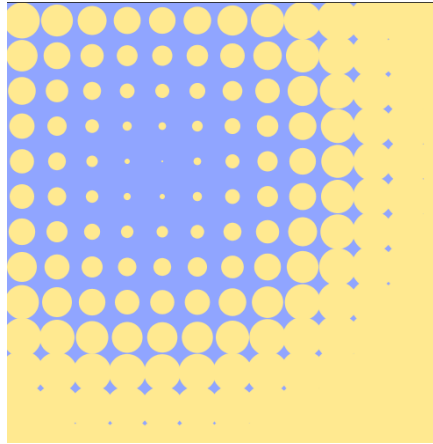
void setup() {
  size(500, 500);
  background(255);

  for (int i = 0; i < 50; i = i+1) {
    liste_x.append(random(width));
    liste_y.append(random(height));
  }
}

void draw() {
  stroke(random(255), random(255), random(255)); // Couleur des contours
  strokeWeight(0.1);                             // Épaisseur des contours
  int i0 = int(random(50));
  int i1 = int(random(50));
  line(liste_x.get(i0), liste_y.get(i0), liste_x.get(i1), liste_y.get(i1));
}
```

## Sketch 02

Ici nous abordons les boucles "for" pour répéter un bloc d'instructions. Nous imbriquons deux boucles "for" pour créer la grille sur deux dimensions.



```
int diametre = 40; // Diamètre des cercles

void setup() {
  size(500, 500);
  noStroke(); // Désactive le contour des formes
  fill(#FFE990); // Couleur de remplissage des cercles
}

void draw() {
  background(#90A5FF); // On repoint le fond

  for (int j = 0; j < height; j += diametre) {
    // A chaque tour de la boucle externe on descend d'une ligne
    for (int i = 0; i < width; i += diametre) {
      // A chaque tour de la boucle interne on décale d'une colonne
      int posx = i + diametre/2;
      int posy = j + diametre/2;
      // On calcule la distance entre le centre de chaque cercle et le curseur de la souris
      float d = dist(posx, posy, mouseX, mouseY);
      circle(posx, posy, d * 0.18);
    }
  }
}
```

[sketch\\_02.mp4](#)

## Sketch 03

Pour sortir de la monotonie des lignes droites, essayons-nous aux courbes !

### Première forme

Ce sketch est interactif. Cliquez dans la fenêtre pour rajouter des points d'ancrages à la courbe.

```
ArrayList<PVector> points = new ArrayList();

void setup() {
  size(500, 500);
  noFill();
}

void draw() {
  background(255);
  beginShape();
  curveVertex(0, 0); // On rajoute un premier point de contrôle aux mêmes coordonnées que le premier point d'ancrage de la courbe
  curveVertex(0, 0);

  for (PVector p : points) {
    p.x = p.x + random(-1,1)*2; // On modifie légèrement les coordonnées de chaque points pour l'effet de vibration
    p.y = p.y + random(-1,1)*2;
    curveVertex(p.x, p.y);
  }

  curveVertex(width, height);
  curveVertex(width, height); // Un dernier point de contrôle pour terminer la courbe
  endShape();

  for (PVector p : points) {
    circle(p.x, p.y, 10);
  }
}

void mousePressed() {
  points.add(new PVector(mouseX, mouseY)); // Chaque clique ajoute un nouveau points aux coordonnées du curseur de la souris
}
```

```
}
```

## Seconde forme

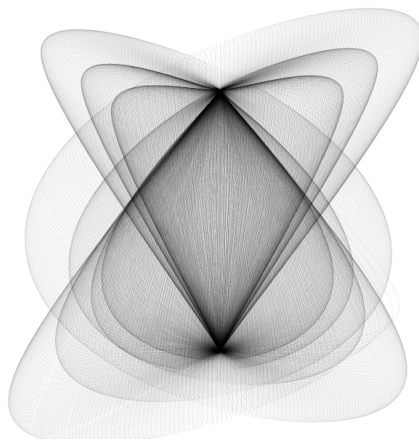
Dans le style de l'harmonographe.

```
ArrayList<PVector> list = new ArrayList();

void setup() {
  size(500, 500);
  background(255);
  strokeWeight(0.2);
}

void draw() {
  float x = 200 * cos(millis() * 0.005);
  float y = 200 * sin(millis() * 0.003);
  fill(0, 0);
  //background(255);
  beginShape();
  curveVertex(width*0.5, 50);
  curveVertex(width*0.5, 50);
  curveVertex(width*0.5 + x, height*0.5 + y);
  curveVertex(width*0.5, height-50);
  curveVertex(width*0.5, height-50);
  endShape();
}

void mouseClicked() {
  list.add(new PVector(mouseX, mouseY));
}
```



Article extrait de : <http://www.lesporteslogiques.net/wiki/> - **WIKI Les Portes Logiques**  
Adresse : <http://www.lesporteslogiques.net/wiki/atelier/processing/start?rev=1670853637>  
Article mis à jour: **2022/12/12 15:00**