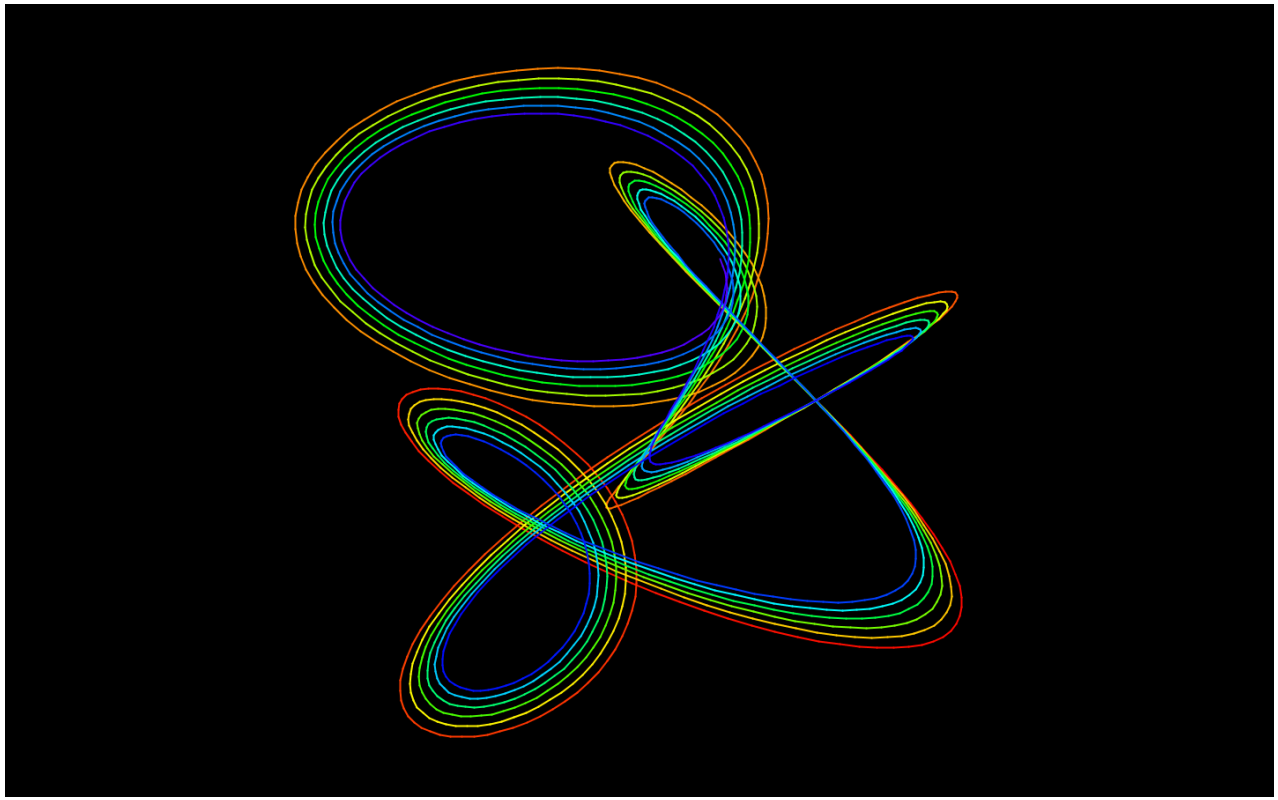
CreativeCodingParis

Atelier Processing

“ondulations et harmonies”

Gregor Schneider



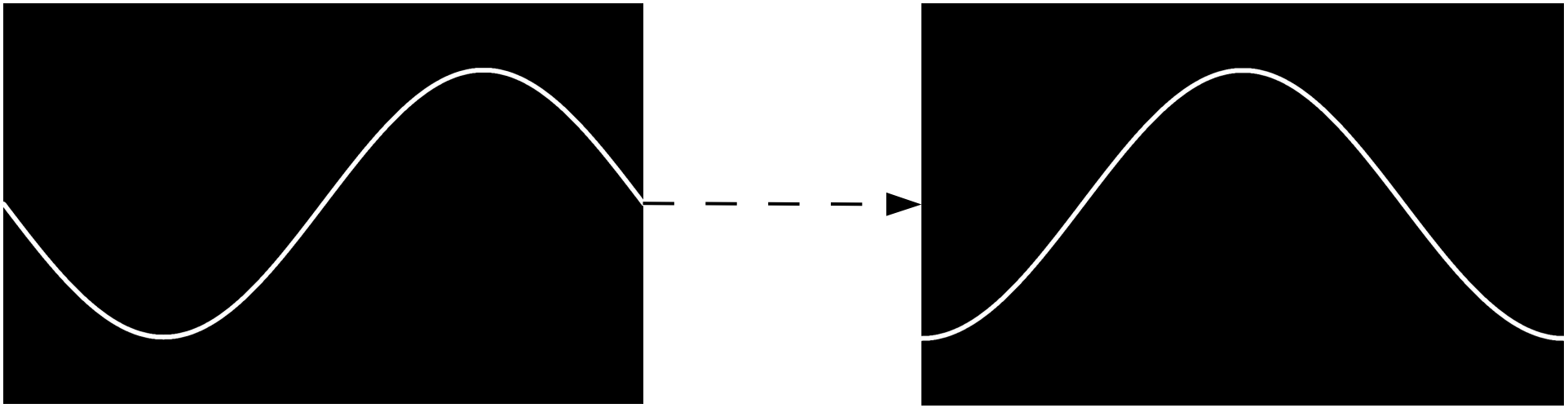
Console Errors

1.a. CreativeCodingParis_Ondulations.pde

```
//Ce code est écrit en Java pour le logiciel de graphisme libre Processing,
//vous pouvez le télécharger gratuitement sur internet.
//Le sketch ci-dessous vous permet de créer des courbes d'ondulations variées,
//tout simplement en changeant les paramètres a, f, p et d.
//Afin d'avoir plus d'interaction, remplacez une valeur avec “mouseX” ou “mouseY”.


void setup() {                      //fonction de configuration globale
  fullScreen();                          //taille de la fenêtre
  frameRate(25);                         //cadence d'images
  background(0);                         //couleur de fond
  stroke(255);                           //couleur du trait
  strokeWeight(10);                      //épaisseur du trait
}
void draw() {                       //fonction d'affichage d'images
  background(0);                         //efface l'image précédente
  translate(0,height/2);                 //coordonnées du point zéro
  float a = height/3;                    //amplitude de l'ondulation
  float f = 1;                           //fréquence de l'ondulation
  float p = 0;                           //phase de l'ondulation
  float d = 0;                           //déclin de l'ondulation
  for (int x = 0; x <= width; x++) {     //boucle décrivant le déplacement sur l'abscisse
    float t = x*TWO_PI/width;                 //fraction de la période 2
    float y = a*sin(f*t+p)*exp(-d*t);         //sinus décrivant le déplacement sur l'ordonnée
    point(x,y);                               //attribution des variables au point à dessiner
  }
}
```

1.b. CreativeCodingParis_Ondulations.pde

```
y = a*sin(f*t+p)*exp(-d*t);   //Le changement de phase p décale le point de départ de la période
```

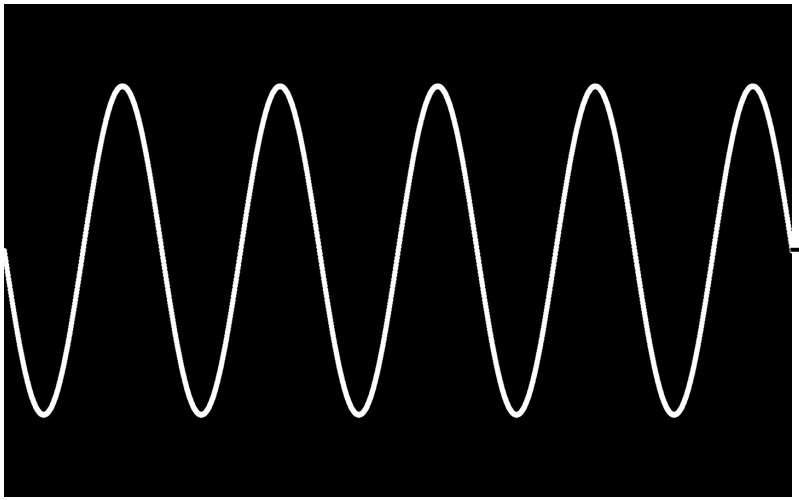


```
a = height/3;
f = 1;
p = 0;
d = 0;
```

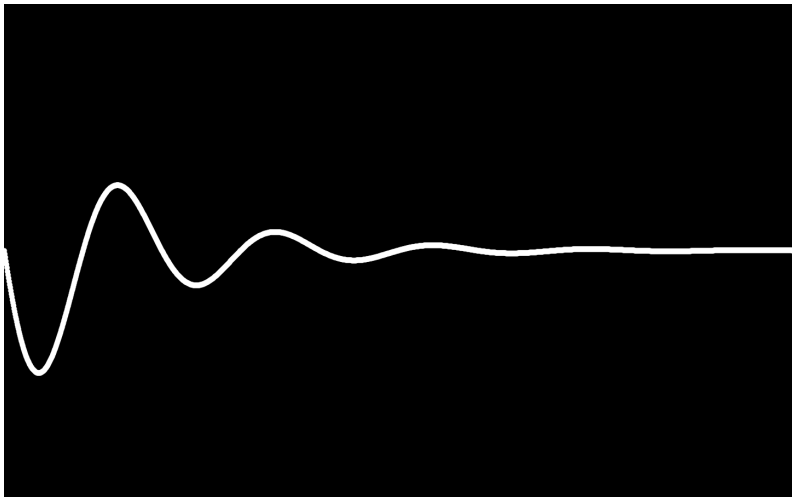
```
a = height/3;
f = 1;
p = HALF_PI;
d = 0;
```

1.c. CreativeCodingParis_Ondulations.pde

```
y = a*sin(f*t+p)*exp(-d*t);   //Le changement du déclin d influence l'amplitude en fonction de t
```



```
a = height/3;
f = 5;
p = 0;
d = 0;
```



```
a = height/3;
f = 5;
p = 0;
d = 1;
```

CreativeCodingParis

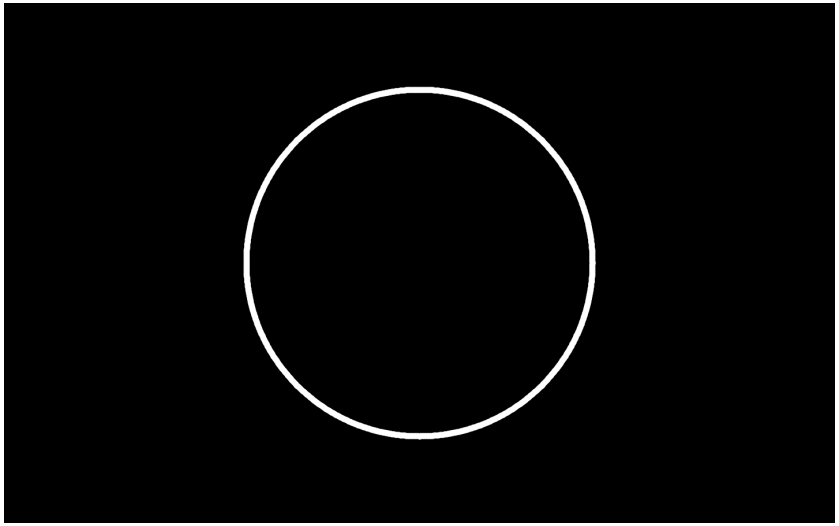
2.a. CreativeCodingParis_Lissajous.pde

```
//Ce sketch est basé sur le principe des figures Lissajous,
//nommées d'après un scientifique français du 19ème siècle.
//Les courbes qui en résultent, naissent de la superposition
//de deux fonctions sinusoïdales sur l'axe x et y.

void setup() {

  fullScreen();
  frameRate(25);
  background(0);
  stroke(255);
  strokeWeight(10);
}
void draw() {
  background(0);
  translate(width/2, height/2);
  float a = height/3;
  float f1 = 1;                          //fréquence de la première fonction trigonométrique
  float f2 = 1;                          //fréquence de la deuxième fonction trigonométrique
  float p2 = 1*HALF_PI;                  //phase de la deuxième fonction trigonométrique
  for (int i=0; i <= width; i++) {
    float t = i*TWO_PI/width;
    float x = a*sin(f1*t);               //sinus décrivant le déplacement sur l'abscisse
    float y = a*sin(f2*t+p2);            //sinus décrivant le déplacement sur l'ordonnée
    point(x, y);                         //attribution des variables au point à dessiner
  }
}
```

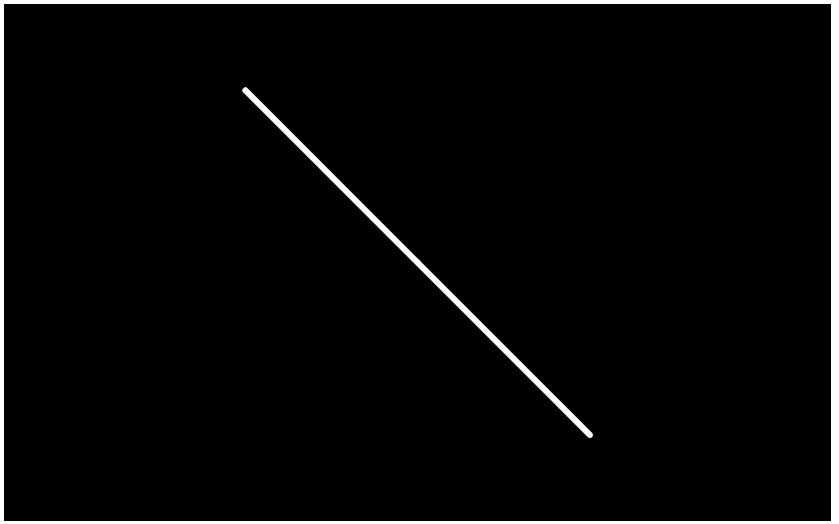
Console Errors

2.b. CreativeCodingParis_Lissajous.pde

```
x = a*sin(f1*t);
y = a*sin(f2*t+p2);  //Ici la phase p permet d'obtenir un cercle, une diagonale, ou une ellipse
```



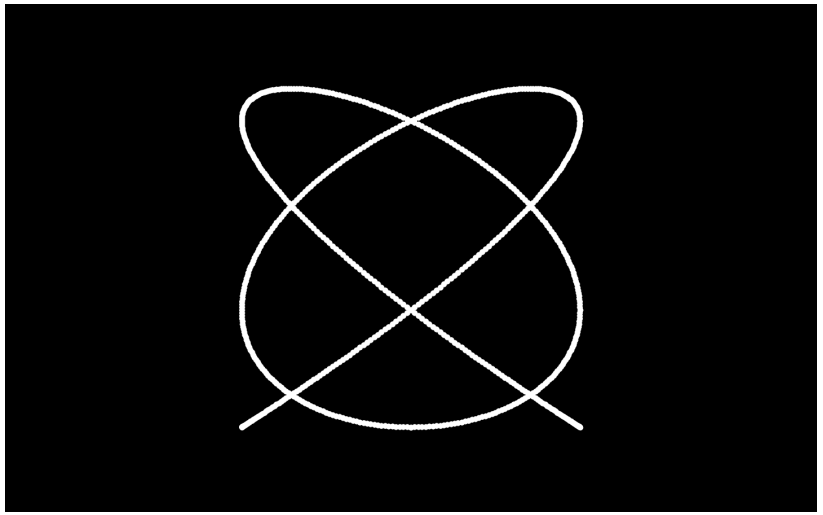
```
f1=1;
f2=1;
p2=PI/2;
```



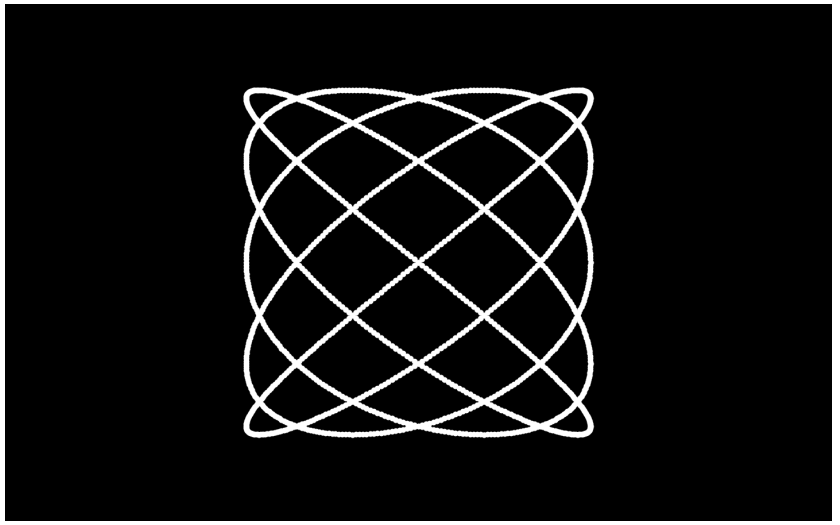
```
f1=1;
f2=1;
p2=0;
```

2.c. CreativeCodingParis_Lissajous.pde

```
x = a*sin(f1*t);          //Les fréquences augmentées donnent des figures intéressantes
y = a*sin(f2*t+p2);       //La fréquence f2 est souvent inférieure à celle de f1
```



```
f1=5;
f2=4;
p2=PI/2;
```



```
f1=5;
f2=4;
p2=0;
```

CreativeCodingParis

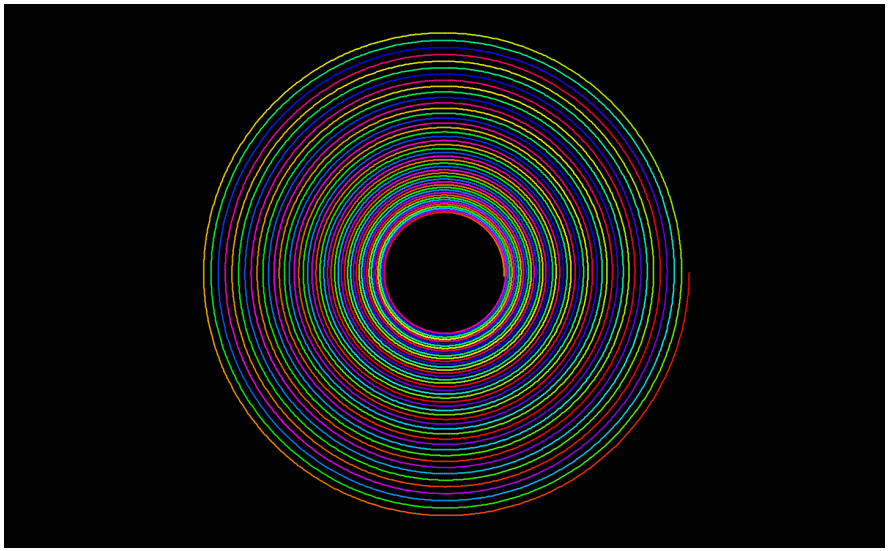
3.a. CreativeCodingParis_Harmonographe.pde

```
//Ce sketch utilise quatre fonctions sinusoïdales pour créer des dessins plus complexes.
float a = 2*height;
float d = 0.005;
float hu,t,x,y,px,py;
void setup() {
  fullScreen();
  smooth();
  noCursor();
  colorMode(HSB);                                    //changement du mode couleur RGB en HSB
  background(0);
  strokeWeight(2);
}
void draw() {
  translate(width/2, height/2);
  x = a*cos(1*t)*exp(-d*t)+a*cos(1*t)*exp(-d*t);      //deux fonctions sinusoïdales horizontales
  y = a*sin(1*t)*exp(-d*t)+a*sin(1*t)*exp(-d*t);      //deux fonctions sinusoïdales verticales
  if (frameCount == 1) {                             //initialisation du point de départ
    px = x;
    py = y;
  }
  stroke(hu,255,255);                                //couleur du trait
  line(px,py,x,y);                                   //ligne à dessiner
  px = x;
  py = y;
  t += 0.01;                                         //incrémentation de l'angle
  hu += 0.1;                                         //incrémentation de la couleur
    if (hu > 255) {
     hu = 0;
   }
}
void mousePressed() {                                //arrêt du dessin en cours
  noLoop();
}
```

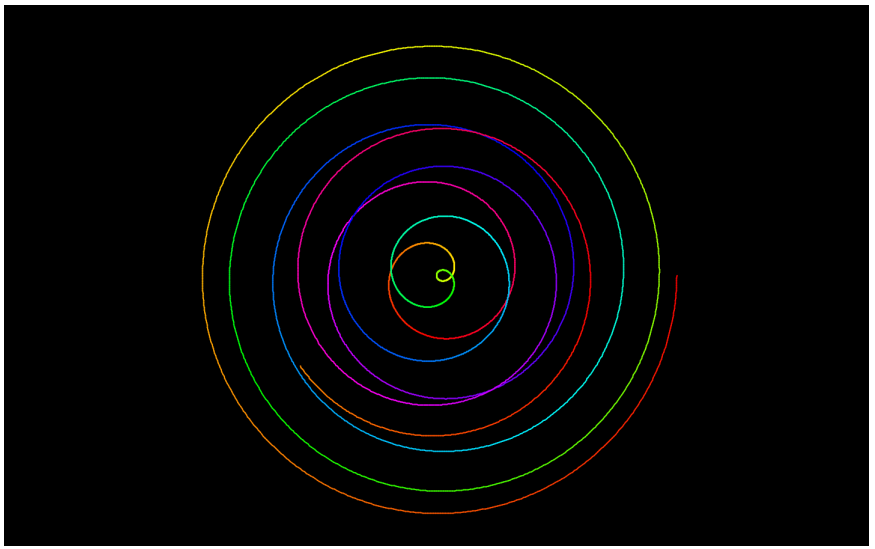
Console Errors

3.b. CreativeCodingParis_Harmonographe.pde

```
x = a1*cos(f1*t)*exp(-d*t)+a2*cos(f2*t)*exp(-d*t);   //Addition de valeurs horizonatles de cercle
y = a1*sin(f1*t)*exp(-d*t)+a2*sin(f2*t)*exp(-d*t);   //Addition de valeurs verticales de cercle
```



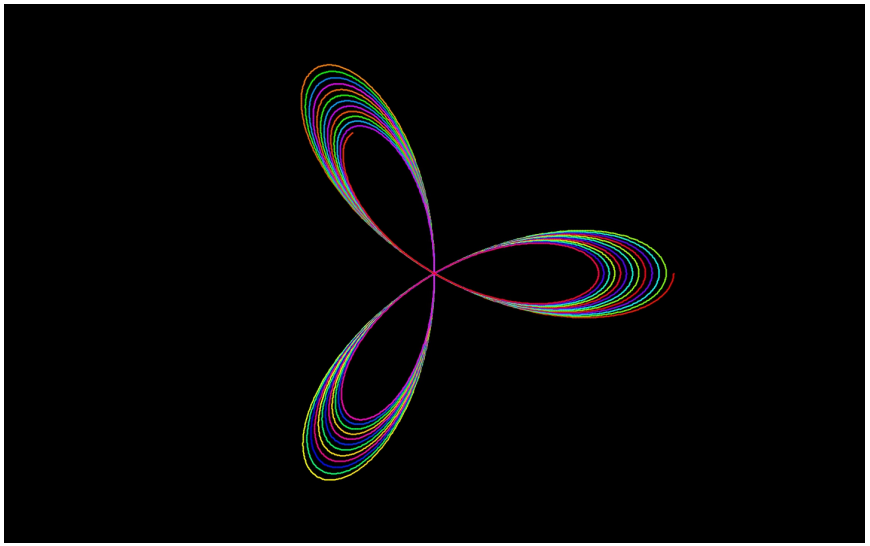
```
f1 = 1;
f2 = 1;
a1 = 1;
a2 = 1;
```



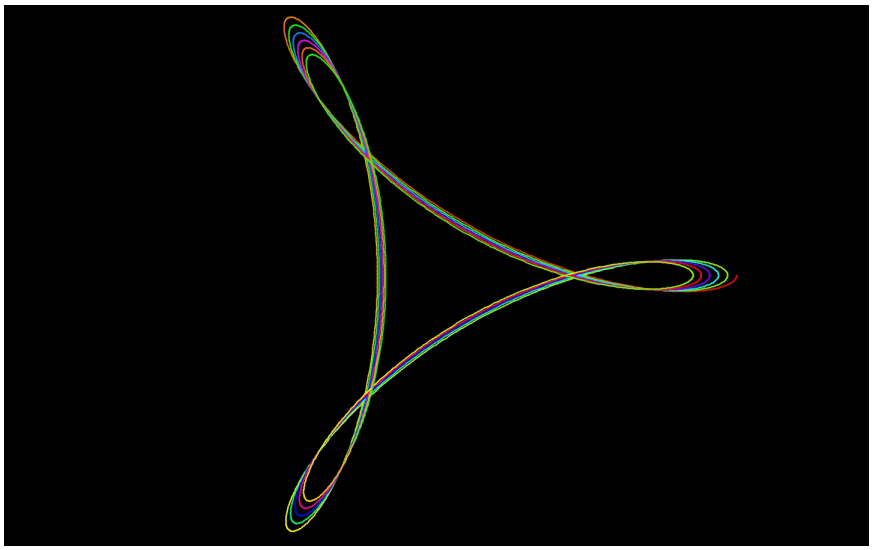
```
f1 = 1.1;
f2 = 1;
a1 = 1;
a2 = 1;
```

3.c. CreativeCodingParis_Harmonographe.pde

```
x = a1*cos(f1*t)*exp(-d*t)+a2*cos(f2*t)*exp(-d*t); //Addition de valeurs horizonatles de cercle
y = a1*sin(f1*t)*exp(-d*t)-a2*sin(f2*t)*exp(-d*t); //Soustraction de valeurs verticales de cercle
```



```
f1 = 2;
f2 = 1;
a1 = 1;
a2 = 1;
```

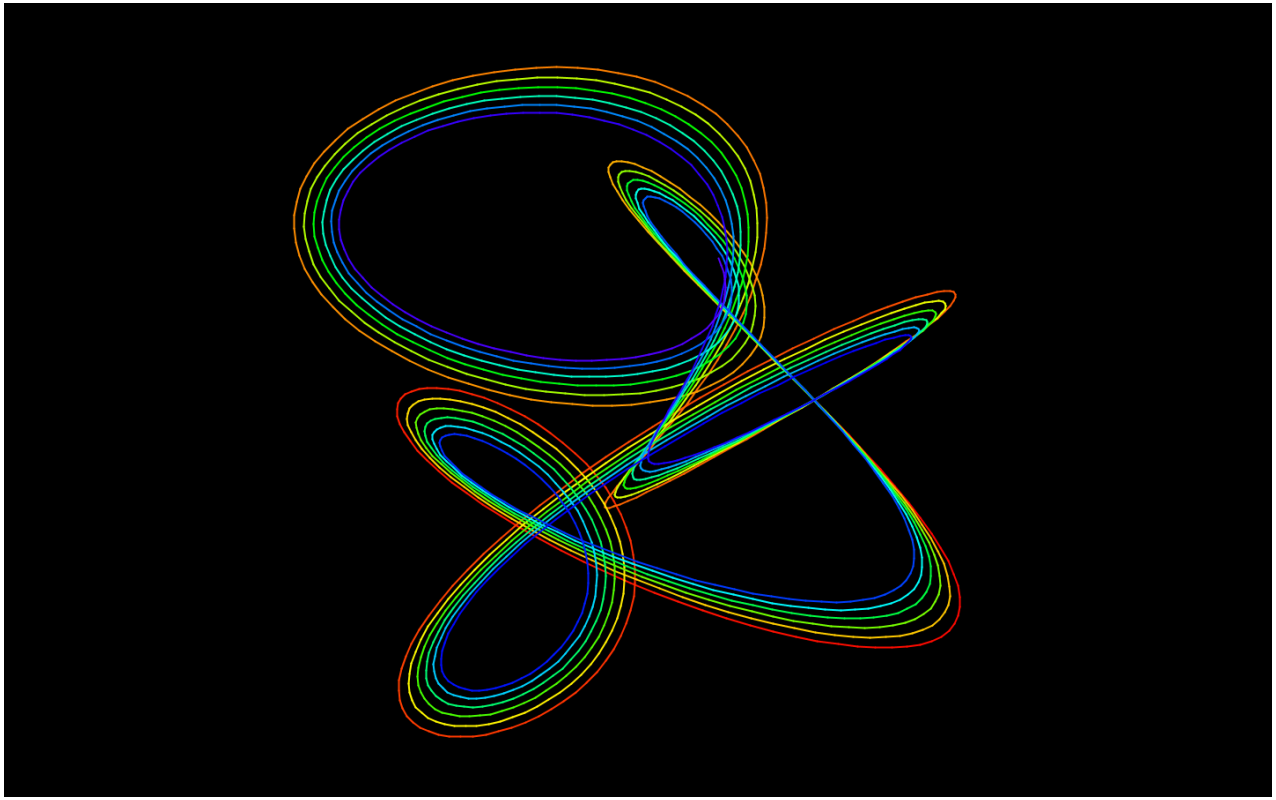


```
f1 = 2;
f2 = 1;
a1 = 1;
a2 = 1.5;
```

CreativeCodingParis

À vous de jouer avec les “ondulations et harmonies”

(ci-dessous: x:sin5+cos2,y:sin4+sin1)



Console Errors