

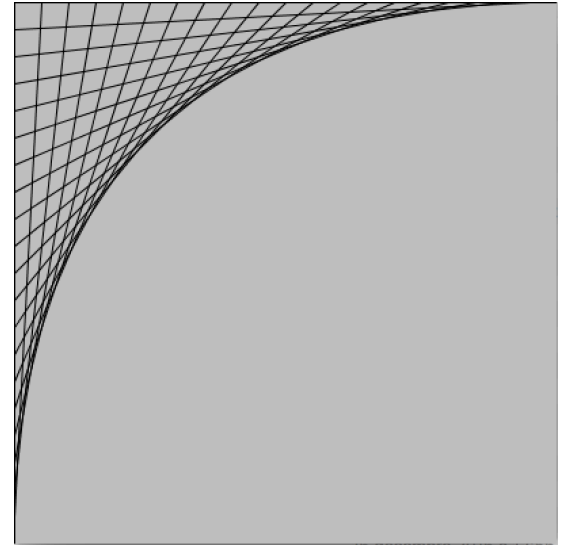
Examen 2017-2018, session 2 (2h, barème /20)

1. Dessin (9 pts)

Dans cet exercice, il vous est demandé de fournir un certain nombre d'éléments nécessaires à la réalisation d'une application écrite en langage Processing. Nous partirons du programme ci-dessous qui donne le résultat ci-contre. (Un corrigé sera distribué à la toute fin des 2 heures)

```

1. void setup(){
2.   size(400, 400);
3. }
4.
5. void draw(){
6.   background(255, 192, 192);
7.   strokeWeight(1);
8.   for (int i=0; i<=20; i++){
9.     float x0 = 0 + i*width/20;
10.    float y0 = 0 ;
11.    float x1 = 0 ;
12.    float y1 = height - i*height/20;
13.    line (x1, y1, x0, y0);
14.  }
15. }
```



1.1. (1pt) de quelle couleur était le fond avant la photocopie en noir et blanc ?

Rose

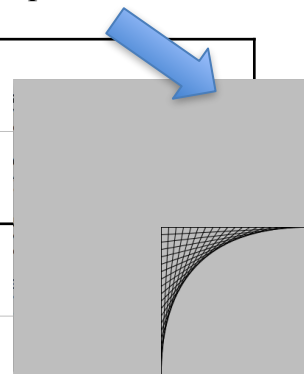
1.2. (3 pts) Changer les calculs de x_0 , y_0 , x_1 et y_1 afin d'obtenir le motif plus petit et décalé

```
float x0 = width/2 + i*width/40;
```

```
float y0 = height/2;
```

```
float x1 = width/2;
```

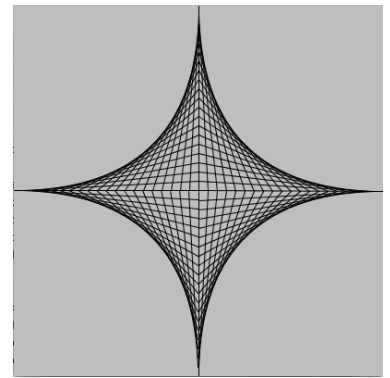
```
float y1 = height - i*height/40;
```



1.2. (1 pts) Comment modifier le code le moins possible pour ne pas afficher la ligne horizontale (celle qui est tout en haut du motif) ?

Il suffit de remplacer le \leq de la ligne 8 par un $<$

1.3. (4pt) En supposant qu'on a mis le résultat de la question 1.2 et 1.3 dans une fonction `drawQ()`, le code suivant affiche donc la même chose qu'après 1.2



```
5. void draw(){
6.     background(255, 192, 192);
7.     strokeWeight(1);
8.     drawQ();
9. }
```

Comment obtenir le résultat ci-dessus en utilisant 4 appels à `drawQ()` et des transformations ?

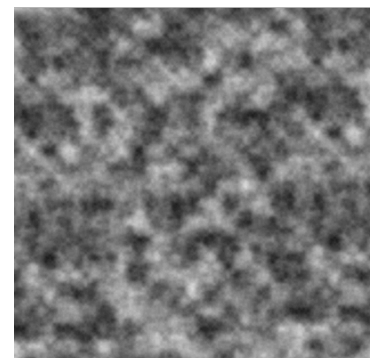
<code>drawQ()</code>
<code>translate(width, 0)</code>
<code>rotate(PI/2)</code>
<code>drawQ()</code>
<code>translate(width, 0)</code>
<code>rotate(PI/2)</code>
<code>drawQ()</code>
<code>translate(width, 0)</code>
<code>rotate(PI/2)</code>
<code>drawQ()</code>
<code>// P.S. on peut aussi faire une boucle !</code>

ATTENTION : après la question 1.2 le repère ne touche plus le coin du motif.

2. Pixels (5 pts)

Nous partons du programme ci-dessous qui donne le résultat ci-contre.

```
1. void draw(){
2.     loadPixels();
3.     for (int j=0; j<height; j++) {
4.         for (int i=0; i<width; i++) {
5.             float d = dist(1*width/2, 1*height/2, i, j) ;
6.             float a = atan2(j-1*height/2.0, i-1*width/2.0);
7.             float x = 1*width/2.0+d*cos(a);
8.             float y = 1*height/2.0+d*sin(a);
9.             float v = noise(x/20.0, y/20.0);
10.            pixels[i+j*width] = color(v*255);
11.        }
12.    }
13.    updatePixels();
14. }
```



2.1. (1pt) Que font mathématiquement les lignes 5 et 6 ?

Elles passent d'un repère cartésien vers un repère polaire

2.2. (1pt) Que font mathématiquement les lignes 7 et 8 ?

Elles passent d'un repère polaire vers un repère cartésien

2.3. (1 pts) Comment modifier la ligne 9 pour se débarrasser de ces 4 lignes ?

Remplacer x par i et y par j

2.4. (2 pts) Center les lignes 6 et 7 on rajoute une des lignes suivante :

- 6. `a=a+d/100.0;`
- 6. `a=a+d/100.0-frameCount/20.0;`

Quel effet a la première ligne sur le graphisme ?

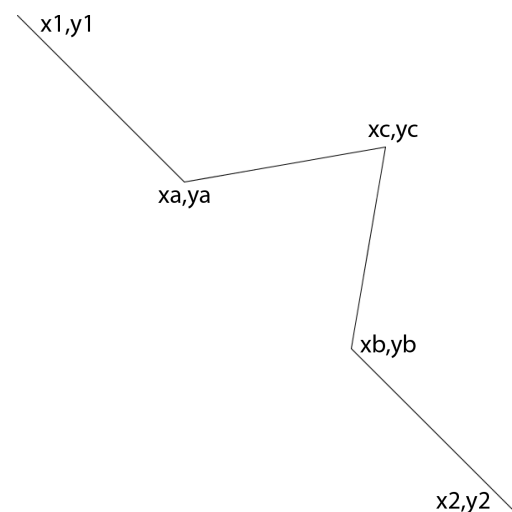
Elle donne un effet de spirale

Quel effet a EN PLUS la deuxième ligne sur le graphisme ?

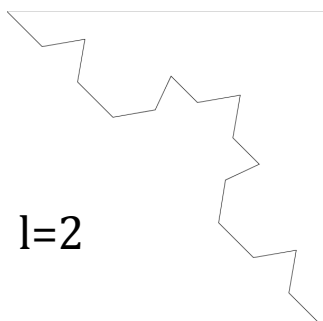
Elle anime l'effet dans le temps (spirale qui tourne)

3. Dessin récursif (6 pts) Courbe de Koch

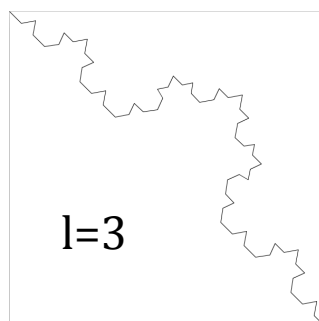
```
1. void drawR(float x1, float y1, float x2, float y2, float l){
2.     ...
3. }
4.
5. void setup(){
6.     size(600, 600);
7. }
8.
9. void draw(){
10.    background(255);
11.    drawR(0, 0, width, height, 4);
12. }
```



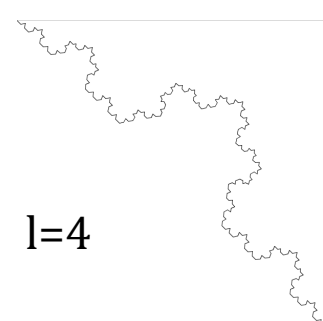
3.1. (6pt) Ecrivez complètement une fonction récursive drawR qui dessine une courbe de Koch entre les points $x1, y1$ et $x2, y2$ pour un nombre de niveau l



$l=2$



$l=3$



$l=4$

Explication : Une étape(un niveau) d'une courbe de Koch consiste à calculer les points intermédiaires x_a, y_a puis x_b, y_b puis x_c, y_c en fonction des 2 points donnés en entrée puis à s'appeler récursivement sur les 4 segments appropriés (sauf au niveau 0 !)

```
void drawR(float x1
  if (l==0){
    line(x1, y1, x2, y2);
  } else {
    float xa = (x1*2+x2*1)/3;
    float ya = (y1*2+y2*1)/3;

    float xb = (x1*1+x2*2)/3;
    float yb = (y1*1+y2*2)/3;

    float xc = (x1+x2)/2 + (yb-ya)/sqrt(2);
    float yc = (y1+y2)/2 - (xb-xa)/sqrt(2);

    drawR(x1, y1, xa, ya, l-1);
    drawR(xa, ya, xc, yc, l-1);
    drawR(xc, yc, xb, yb, l-1);
    drawR(xb, yb, x2, y2, l-1);
  }
}
```