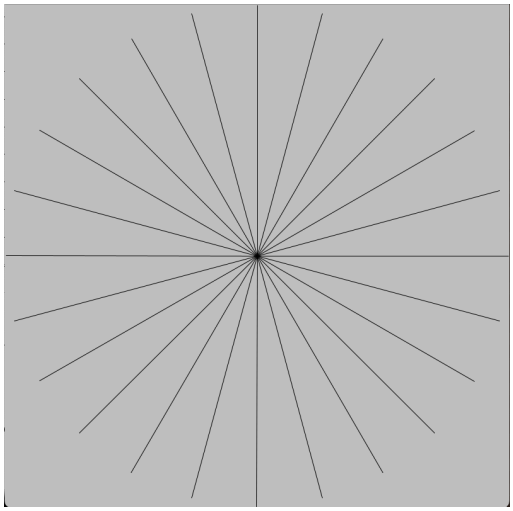
Licence MPI. Info 112 (IGOR)
F. Vernier

Examen 2023, session 1,5 (2h, barème /21)

1. Dessin (8 pts)

Dans ces exercices, 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.

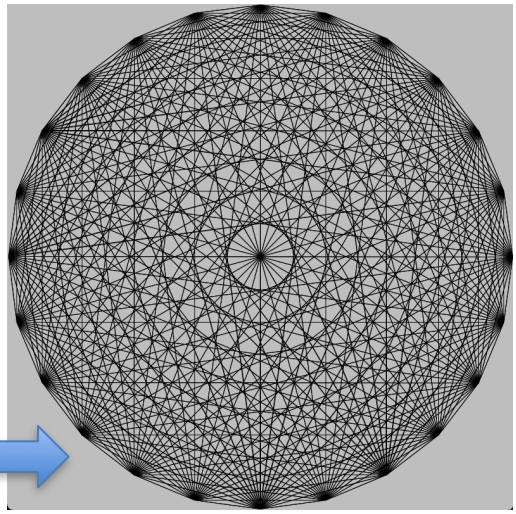
```
1.  void setup(){
2.    size(800, 800);
3.  }
4.  int N =24;
5.
6.  void draw(){
7.    background(192);
8.    noFill();
9.    float R = height/2-2;
10.   for(int i=0; i<N; i++){
11.     float a0 = i*PI*2/N;
12.     float x0 = width/2 +R*cos(a0);
13.     float y0 = height/2+R*sin(a0);
14.     line(x0,y0, width/2, height/2);
15.   }
16. }
```



1.1. ***(1pt)*** *Comment s'appelle le principe de math des lignes 12 et 13 ?*

La Trigonométrie

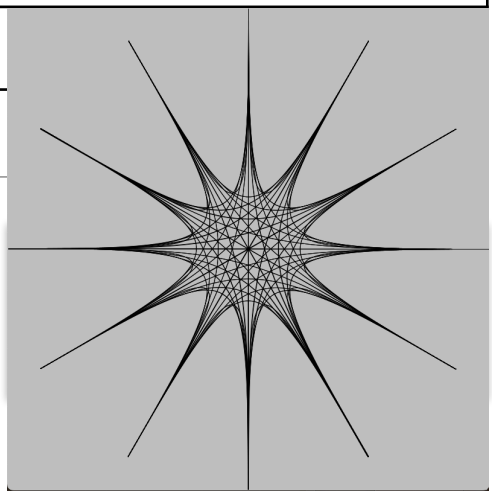
1.2. ***(3 pts)*** *Comment relier tous les points 2 à 2 par des lignes comme ci contre (vous écrirez le code qui remplace la ligne 14 par plusieurs lignes de code)?*



```
for(int j=i+1; j<i+N; j++){
  float a1 = j*PI*2/N;
  float x1 = width/2 +R*cos(a1);
  float y1 = height/2+R*sin(a1);
  line(x0,y0,x1,y1);
}
```

1.3. ***(1.5pt)*** *Remplacer le code de la question 1.2 qui relie les points 2 à 2 par des courbes de bezier dont la tangente est orientée vers le centre au début **et** a la fin de la courbe ?*

```
bezier(x0,y0, width/2, height/2,
width/2, height/2,  x1, y1);
```



1.4. ***(2.5pt)*** *Changer les points de contrôle pour être entre le centre et le bout de la courbe associé au point de contrôle selon une variable r(quand r=0.5 le point sera au milieu des 2, quand r=0.9 il sera beaucoup plus près du centre, quand r=0.0 il sera sur le bout de la courbe, etc.)*

```
bezier(x0,y0, (1-r)*x0+r*width/2, (1-r)*y0+r*height/2,
(1-r)*x1+r*width/2, (1-r)*y1+r*height/2,  x1, y1);
```

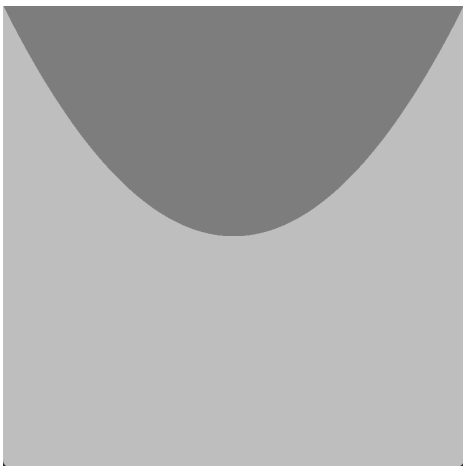
Que rajouter au début de la fonction draw (declaration et calcul de r) pour faire une animation qui oscille(fonction de trigonométrie) entre le résultat de 1.2 et celui de 1.3

```
float r = (cos(frameCount/20.0)+1)/2;
```

2. Pixels (8 pts)

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

```
17. void draw(){
18.   background(192);
19.   loadPixels();
20.   for (int j=0; j<height; j++) {
21.     for (int i=0; i<width; i++) {
22.       float x = (i-width/2.0)/(width/2);
23.       float y = - (j-height/2.0)/(height/2);
24.        if(x*x<y)
25.         pixels[i+j*height] = color(128);
26.      }
27.    }
28.   updatePixels();
29. }
```



Dessinez le repère x,y ci dessus

2.1. ***(2.5pt)*** *Les lignes 22 et 23 font un changement de repère. Dessinez d'abord le repère x,y sur la figure grise ci-dessus. Pour chacune des 3 parties de la ligne 23 détaillez à quoi sert ce morceau de calcul pour le nouveau repère ?*

```
23.float y =  -    (j-height/2.0)  / (height/2);
```

A B C

B : la soustraction de height/2 sert a quoi ?

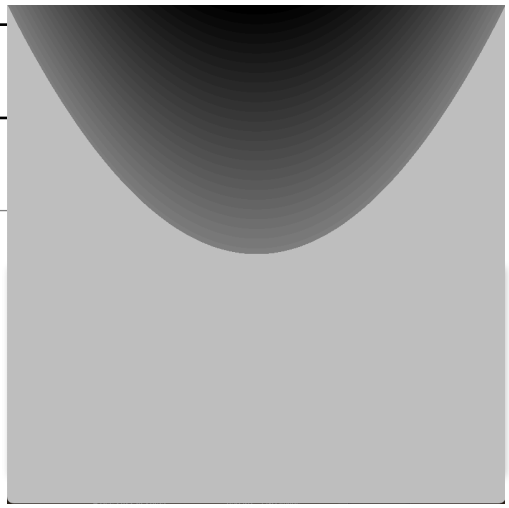
A déplacer le repère au centre

C : la division par height/2 sert a quoi ?

A normaliser le repère entre -1 et +1

A : le signe - (moins) global sert à quoi ?

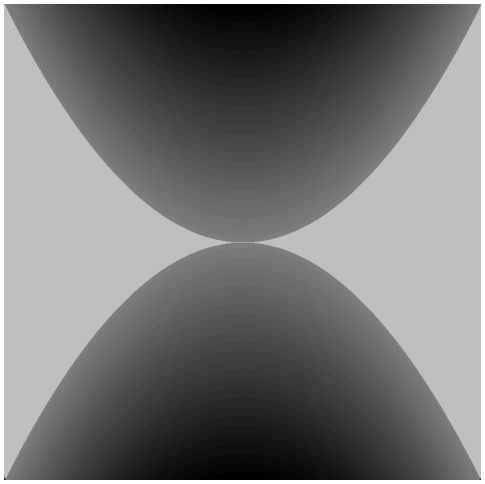
A retourner l'axe des y vers le haut



2.2. ***(1.5 pts)*** *plutôt que faire des pixels gris moyen (128) comment faire que plus les pixels s'éloignent de la courbe y=x*x plus les pixels soient foncés (et on voudrait qu'ils soient complètement noir(0 au milieu au centre). Re-ecrire la ligne 25*

```
pixels[i+j*height] = color(128-(y-x*x)*128);
```

2.3. ***(2 pts)*** *en utilisant la fonction abs() ou bien en utilisant un "ou logique" (||) coloriez aussi la partie basse. changer le if (ligne 24). On ne vous demande pas de modifier le dégradé de la question précédente*

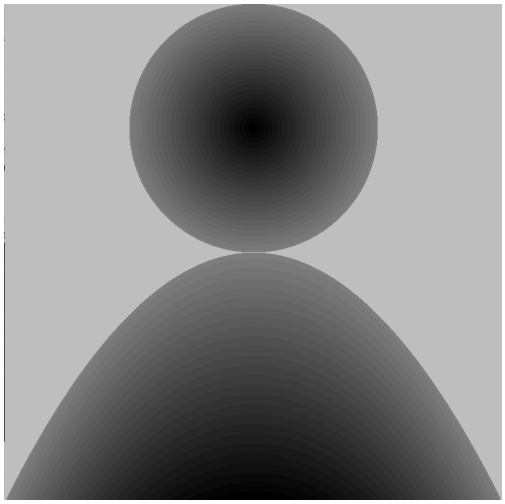


```
if(x*x<abs(y) )
```

Ou

```
if(x*x<y  || x*x<-y )
```

2.4. ***(2 pts)*** *comment faire maintenant une forme de personnage comme ci-contre avec le meme dégradé vers le noir quand les pixels s'éloignent du bord*

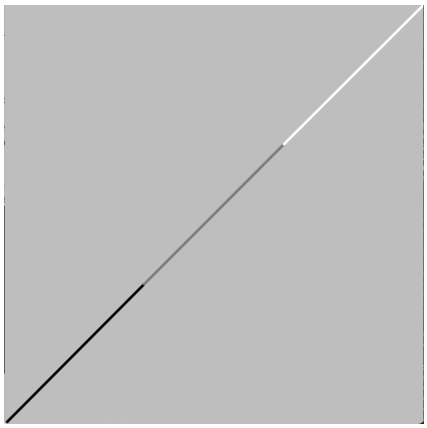


```
if(x*x<-y )
  pixels[i+j*height] = color(128-(abs(y)-x*x)*128);
    else if (dist(x,y,0,0.5)<0.5)
  pixels[i+j*height] = color(dist(x,y,0,0.5)*255);
```

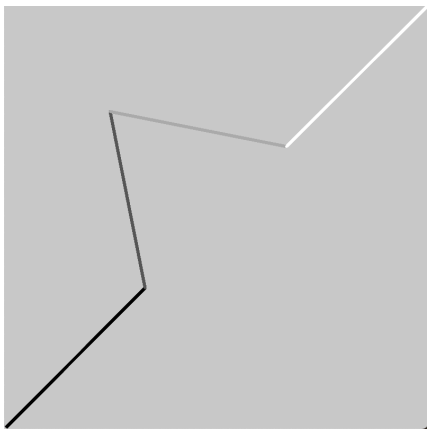
3. Dessin récursif (5 pts)

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

```
30. void draw(){
31.   strokeWeight(5);
32.   truc(0,height,  width,0,   0, 255);
33. }
34.
35. void truc(float x0, float y0, float x1, float y1,
                    float minb, float maxb){
36.   float xa = (x0*2+x1*1)/3;
37.   float ya = (y0*2+y1*1)/3;
38.   float xb = (x0*1+x1*2)/3;
39.   float yb = (y0*1+y1*2)/3;
40.   stroke(minb);
41.   line(x0,y0, xa, ya);
42.   stroke((minb+maxb)/2);
43.   line(xa,ya, xb, yb);
44.   stroke(maxb);
45.   line(xb,yb, x1, y1);
46. }
```



3.1. ***(2 pts)*** *Dans un premier temps nous souhaitons découper le segment en 4 au lieu de 3 comme ci-contre, avec 4 niveaux de couleur entre minb et maxb. Le nouveau point est sur la droite perpendiculaire à partir du milieu des 2 points (x1,y1 et x0,y0)*

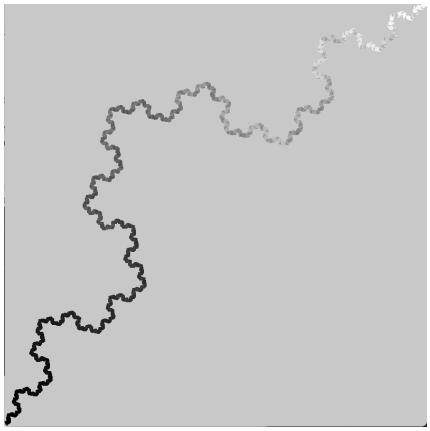


```
float yc = (y0+y1)/2- (x1-x0)/4;
float xc = (x0+x1)/2+ (y1-y0)/4;
```

Et a la place des lignes 42 et 43 :

```
  stroke((minb*2+maxb)/3);
  line(xa,ya, xc, yc);
  stroke((minb+maxb*2)/3);
  line(xc,yc, xb, yb);
```

3.2. ***(3 pts)*** *Dans un second temps nous souhaitons transformer* ***truc()*** *en fonction récursive.* ***truc()*** *doit s'appeler récursivement 4 fois à la place des 4 dessins de lignes si la distance entre x0, y0 et x1,y1 est supérieur à 5*



```
float yc = (y0+y1)/2- (x1-x0)/4;
float xc = (x0+x1)/2+ (y1-y0)/4;
if(dist(x0,y0, x1,y1)>5){
  truc(x0,y0, xa, ya, minb, (minb*2+maxb)/4);
  truc(xa,ya, xc, yc, (minb*3+maxb*1)/4, (minb*2+maxb*2)/4);
  truc(xc,yc, xb, yb, (minb*2+maxb*2)/4, (minb*1+maxb*3)/4);
  truc(xb,yb, x1, y1, (minb*1+maxb*3)/4, maxb);
} else{
   stroke((minb+maxb)/2);
  line(x0,y0, x1, y1);
}
```