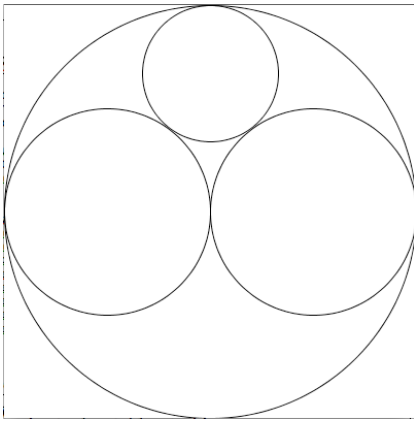
Examen 2014-2015 (2h, barème /20)

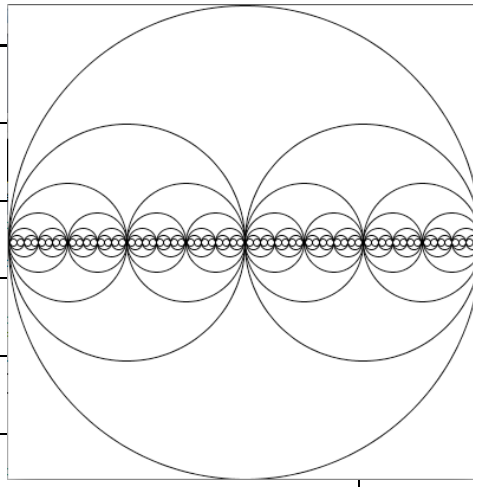
1 Dessin Récursif (5 pts)

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.



```
1.   void setup(){
2.     size(400, 400);
3.     noFill();
4.     stroke(0);
5.     background(255);
6.     ellipseMode(RADIUS);
7.     noLoop();
8.   }
9.   void draw(){
10.    ellipse(width/2, height/2, width/2, width/2);
11.    ellipse(1*width/4, height/2, width/4, width/4);
12.    ellipse(3*width/4, height/2, width/4, width/4);
13.    ellipse(width/2, height/6, width/6, width/6);
14.  }
```

1.1 ***(2 pts)*** *Donnez le code d'une fonction récursive qui dessine un cercle à partir des paramètres et selle récursivement 2 fois comme ci-contre.*

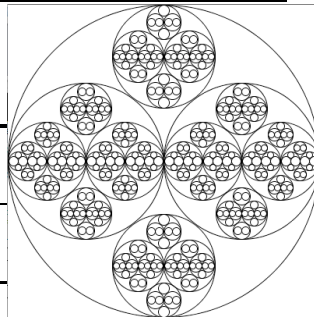


```
void drawR(int x, int y, int r){
ellipse(x, y, r, r);
if (r>1){
   drawR(x-r/2,  y, r/2);
   drawR(x+r/2, y, r/2);
 }
}
```

1.2 ***(1 pts)*** *Donnez l'appel initial qui affiche le dessin ci-dessus dans tout l'écran.*

```
drawR(width/2, height/2, width/2);
```

1.3 ***(2pt)*** *Quel lignes de code faut-il ajoutera la fin de la fonction récursive pour des cercle plus petit en dessus et au dessous ?*



```
drawR(x, y-r+r/3, r/3);
drawR(x, y+r-r/3, r/3);
```

2 Glamouflage = camouflage glamour (5 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 de la Figure 2.1..

```
1.  PImage src;
2.  void setup(){
3.    size(400, 400);
4.    src = createImage(width, height, RGB);
5.    src.loadPixels();
6.    for (int j=0; j<height; j++)
7.      for (int i=0; i<width; i++){
8.        float val = noise(i/25.0, j/25.0)*255;
9.        if (3*i+2*j<3.5*height && 2*j-3*i<height/2 &&  j>=height/4)
10.         src.pixels[i+j*width] = color(val, val,
     val);
11.     }
12.   src.updatePixels();
13. }
14. void draw(){
15.   image(src, 0, 0);
16. }
```

Fig. 2.1

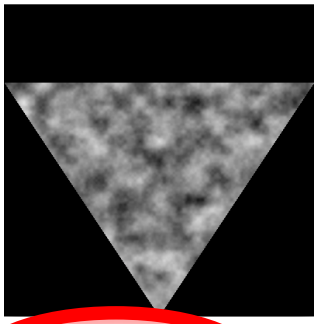
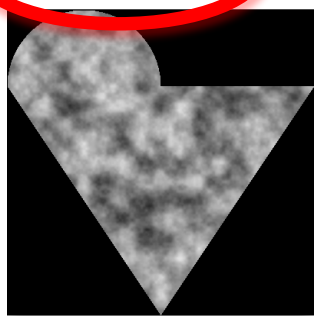


Fig. 2.2



2.1 (1 pts) *Le bruit de Perlin (noise) est affiché dans un triangle. Entourez dans le code ci-dessus la partie responsable du coté horizontal du triangle qui est à un quart de la hauteur.*

2.2 (2 pts) *Réécrivez les lignes 9 à 10 pour que le bruit de Perlin s'affiche dans le triangle OU dans un demi-cercle supérieur de centre (width/4, height/4) et de rayon width/4(Fig.2.2)*

```
    float d1 = dist(i,j, width/4, height/4);
 if ((3*i+2*j<3.5*height && 2*j-3*i<height/2 &&
j>=height/4) || (d1<width/4  && j<height/4))
```

2.3 (2 pts) *Réécrivez la ligne 11 pour remplacer le dégradé de gris du bruit de Perlin par trois variantes de rose/rouge que vous choisirez et qui formeront un motif de glamouflage ! (Fig. 2.3 avec un autre demi cercle).*

Fig 2.3

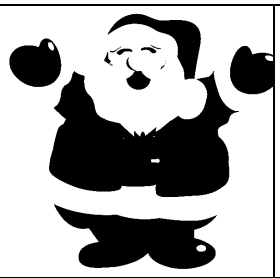
```
if (val<96)
    src.pixels[i+j*width] = color(255, 90, 90);
else if (val<160)
    src.pixels[i+j*width] = color(255, 180, 180);
else
    src.pixels[i+j*width] = color(255, 220, 220);
```

3 Traitement d'image (5 pts)

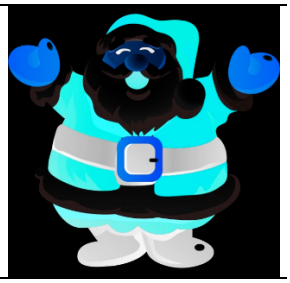
Le code ci-dessous permet d'afficher à l'écran l'image de la Figure 3.

```
1.  PImage perenoel;
2.  void setup() {
3.    size(600, 600);
4.    perenoel = loadImage("pere-noel.jpg");
5.    perenoel.loadPixels();
6.    for (int j=0; j < perenoel.height; j=j+1) {
7.      for (int i=0; i < perenoel.width; i=i+1) {
8.        int c = perenoel.pixels[i+j*perenoel.width];
9.        int r = int(red(c));
10.       int g = int(green(c));
11.       int b = int(blue(c));
12.       // Variable à utiliser uniquement pour les questions 1 et 2
13.       int l = …;
14.       // A changer pour les questions
15.       perenoel.pixels[i+j*perenoel.width] = color(r,g,b);
16.     }
17.   }
18.   perenoel.updatePixels();
19.   noLoop();
20. }
21. void draw() {
22.   image(perenoel,     0,
    0);
23. }
```

3.1. Niveaux de gris



3.2. Noir et blanc



3.3. Couleurs inversées

3.1 ***(1 pts) Niveaux de gris*** *Modifier la couleur affectée aux lignes 12-15 pour transformer l'image en niveau de gris, c'est-à-dire où composante rouge = composante bleu = composante verte (figure 3.1).*

```
int l = (r+g+b)/3 ;
perenoel.pixels[i+j*perenoel.width] = color(l,l,l);
```

3.2 ***(2 pts) NOIR OU BLANC*** *En utilisant votre réponse à la question 3.1, passez l'image en noir et blanc. Fixer le seuil entre noir et blanc à 180 (figure 3.2).*

```
int l = (r+g+b)/3 ;
if (l<180 ) perenoel.pixels[….] = color(0,0,0) ;
else perenoel.pixels[….] = color(255,255,255) ;
```

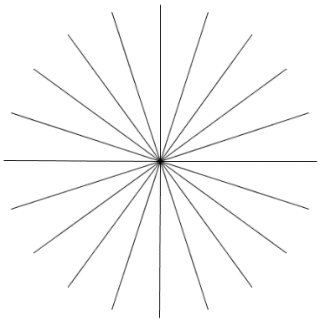
3.3 ***(2 pts) INVERSE VIDEO*** *Inversez les couleurs : un composante de couleur à 255 devient 0, à 254 devient 1 etc. un 0 devient 255 (figure 3.3).*

```
r=255-r ; g=255-g ;b=255-b ;
perenoel.pixels[i+j*perenoel.width] = color(r,g,b);
```

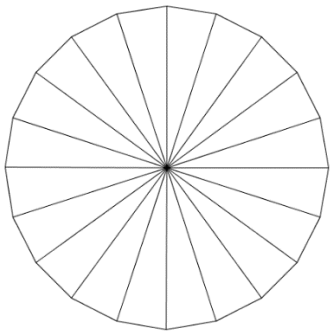
4 La toile d'araignée (5 pts)

Soit le code suivant :

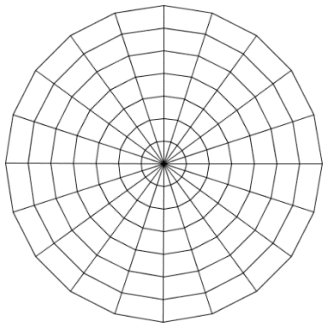
```
void setup() {
  size(600, 600);
  background(255);
  noLoop();
}

void draw() {
  int nombre_fils = 20;
  int r = 250;
  int nombre_cercles = 7;
  for(float i = 0; i < 2*PI; i+=(2*PI)/nombre_fils) {
    …
  }
}
```



4.1



4.2



4.3

On rappelle la fonction processing :
`line(x1,y1,x2,y2) : trace une ligne du point (x1, y1) au point (x2, y2)`

4.1 (1.5 pt) *Compléter la boucle `for` d'un `line(…)` pour obtenir la figure 4.1, c'est-à-dire une ligne partant du centre de la fenêtre au bout du rayon de longueur `r`.*

```
line(width/2, height/2, width/2+r*cos(i), height/2+r*sin(i)) ;
```

4.2 (1.5 pts) *Rajouter un autre `line(…)` pour obtenir la figure 4.2, où chaque extrémité d'un fil est reliée à la suivante.*

```
float i2 = i+(2*PI)/nombre_fils;
line(width/2+r*cos(i), height/2+r*sin(i)) , width/2+r*cos(i2),
height/2+r*sin(i2));
```

4.3 (2pts) *Remplacer le code de la question 4.3 pour obtenir la figure 4.3 avec nbc cercles.*

```
float i2 = i+(2*PI)/nombre_fils;
for (float r2 = 0; r2<=r; r2 += r/nbc) {
  line(width/2+r2*cos(i), height/2+r2*sin(i)) ,
       width/2+r2*cos(i2), height/2+r2*sin(i2));
}
```