Licence MPI. Info 112 (IGOR)
F. Vernier, Y. Gaffary

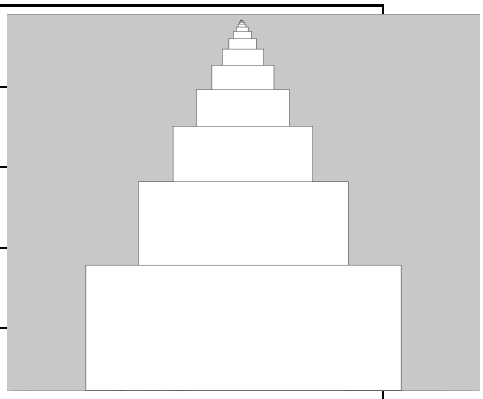
Examen de Rattrapage 2014-2015 (2h, barème /20)

1 Dessin Récursif (7 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(640, 480);
3.     noLoop();
4.   }
5.   void draw(){
        rect(width/6, 2*height/3,
             2*width/3, height/3);
6.   }
```

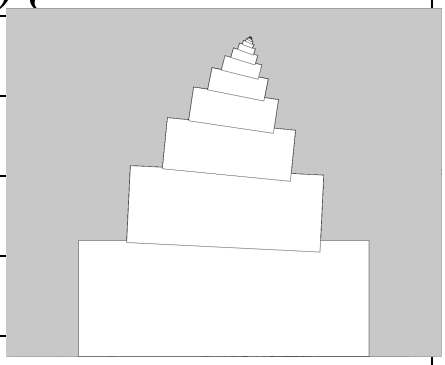
1.1 ***(4 pts)*** *Donnez le code d'une fonction récursive qui dessine un empilement de rectangles comme ci-contre (facteur de taille de 2/3 entre deux rectangles successifs).*



```
void setup(){
  size(1280, 1024);
}


void draw(){
  dr(width/6, 2*height/3, 2*width/3, height/3, 15);
}


void dr(int x, int y, int w, int h, int n){
  rect(x,y,w,h);
  if (n>0)
    dr(x+w/6, y-2*h/3, 2*w/3, 2*h/3, n-1);
}
```

1.2 ***(3 pts)*** *Modifiez la fonction draw() pour que l'empilement soit penché (c.a.d avec une rotation de PI/64 autour du coin haut-gauche a chaque pas recursif)*

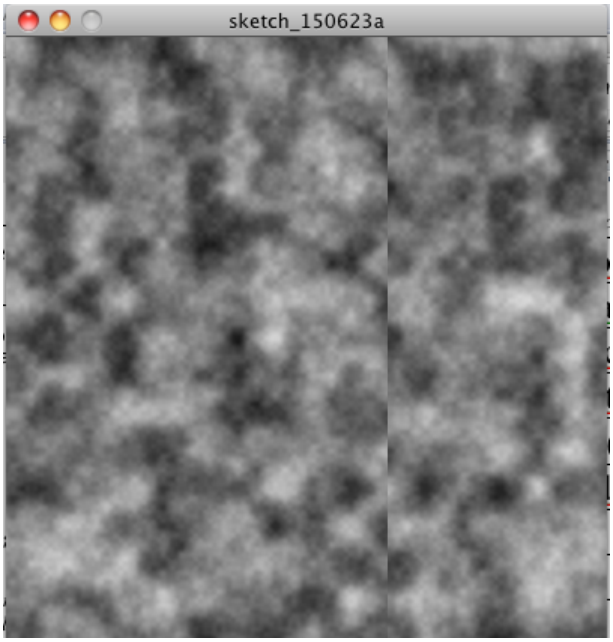
```
void dr(int x, int y, int w, int h, int n){
  translate(x, y);
  rect(0,0,w,h);
  rotate(PI/64);
  if (n>0)
    dr(w/6, -2*h/3, 2*w/3, 2*h/3, n-1);
}
```



2 Bruit de Perlin (8 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 ci contre

```
1.  PImage src;
2.  int dec = 0;
3.
4.  void setup(){
5.    size(400, 400);
6.    src = createImage(width, height, RGB);
7.    src.loadPixels();
8.    for (int j=0; j<height; j++) {
9.      for (int i=0; i<width; i++){
10.       float val = noise(i/25.0, j/25.0, 0)*255;
11.       src.pixels[i+j*width] =
12.            color(val,val,val);
13.     }
14.   }
15.   src.updatePixels();
16. }
17.
18. void draw(){
19.   dec++;
20.   image(src, (dec%width), 0);
21.   image(src, (dec%width)-width, 0);
22. }
```



2.1 ***(1 pts)*** *A quoi servent les division par 2.05 a la ligne 10 ?* **.**

la fonction de perlin prend de petits nombres
alors que i et j sont dans le repère de l'image qui est
beaucoup plus grand

2.2 ***(2 pts)*** *Décrivez en 3 lignes maximums l'animation que fait ce programme (que voit l'utilisateur ?).*

l'image de bruit de perlin qui est pre-calculée dans
setup **défile de gauche à droite** avec une couture
quand l'image boucle sur elle-meme

2.3 ***(3 pts)*** *Afin d'éviter la « couture » au milieu de l'image la solution consiste delà ne pas prendre la valeur de perlin sur un plan* `(i/25.0, j/25.0, 0)*` *mais sur un cylindre car un cylindre boucle sur lui –même et les pixels a un bout de la texture correspondront a ceux de l'autre bout. Proposez plusieurs lignes de code qui remplaceront la ligne 10.*

INDICATION : *quelle doit-être le rayon du cylindre pour que la texture fasse toujours 400x400 ?*

```
float angle = i*PI/width;
float dist  = width/PI/50.0;
float val   = noise(dist*cos(angle),
                    dist*sin(angle), j/25.0)*255;
```

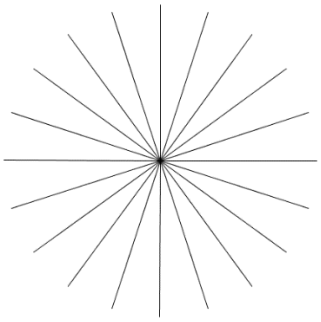
2.4 ***(2 pts)*** *L'autre possibilité, moins uniforme mais plus simple a mettre en œuvre consiste a créer la texture a partir des valeurs du bruit de perlin dans un sens sur une moitie de la texture puis l'autre moitié dans l'autre sens. Une telle stratégie ajoute toutefois une symétrie visible. Proposez plusieurs lignes de code qui remplaceront la ligne 10.*

```
float val = noise(i/25.0, j/25.0, 0)*255;
if (i>width/2)
    val = noise((width-i)/25.0, j/25.0, 0)*255;
```

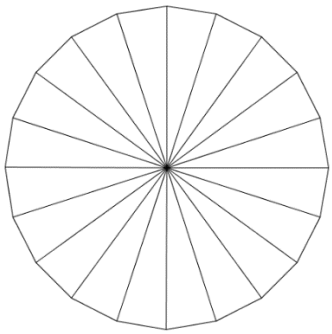
3 La toile d'araignée (OUI C'EST BIEN LE MEME EXERCICE) (5 pts)

Soit le code suivant :

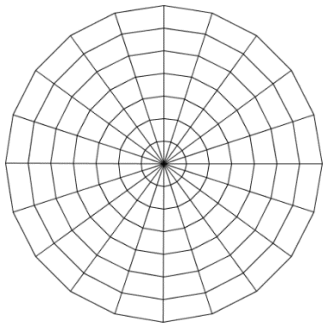
```
void setup() {
  size(600, 600);
  background(255);
  noLoop();
}

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)

3.1 (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)) ;

3.2 (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));

3.3 (2pts) *Remplacer le code de la question 4.2 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));
}