Examen de 2nde session 2013-2014 (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.

```
1.   void setup (){
2.     size(800, 800);
3.   }

4.   void draw(){
5.     background(255, 255, 196);
6.     stroke(128, 0, 128);
7.     strokeWeight(3);
8.     for (int i=0; i<640; i++){
9.       float angle1 = (i*PI*8)/640.0;
10.      float x1 = i*cos(angle1);
11.      float y1 = i*sin(angle1);
12.      float angle2 = ((i+1)*PI*8)/640.0;
13.      float x2 = (i+1)*cos(angle2);
14.      float y2 = (i+1)*sin(angle2);
15.      line (width/2+x1, height/2+y1,
               width/2+x2, height/2+y2);
16.    }
17.  }
```

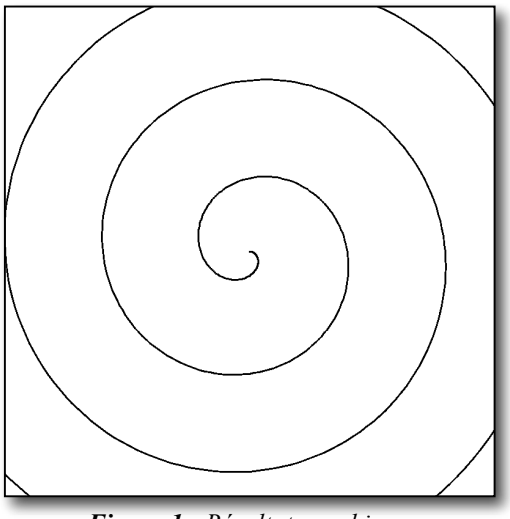


Figure 1 : *Résultat graphique*

1.1 ***(2pt)*** *Quel sont les deux vraies couleurs (en français) du programme ci-dessus ?*

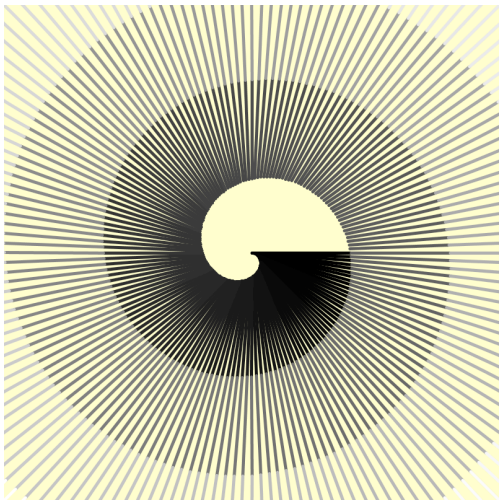
Jaune clair pour le fond
Magenta foncé pour le trait

1.2 ***(1.5pt)*** *A partir de quelle valeur de i le programme dessine t-il un tour complet ?*

160

1.3 ***(2.5pts)*** *Proposez une modification du programme pour obtenir le nouveau dessin ci-contre (noir et blanc !).*

Enlever le stroke avant la boucle
Mettre stroke(i/2, i/2, i/2);
dans la boucle
aux lignes 13 et 14 remplacer
(i+1) par (i+160)



1.4 ***(1pts)*** *Par quelle autre primitive de dessin faudrait-il remplacer* ***line(…)*** *pour bien remplir tout l'escargot (le fond ne soit plus visible sauf au debut)?*

par le dessin d'un polygone à 4 points.

2 points sur le même tour de boucle (i et i+1) et deux points sur le tour suivant (i+160 et i+161)

1.5 ***(1pts)*** *Que faut-il modifier pour que l'escargot soit dessiné dans l'autre sens (la spirale tourne dans l'autre sens).*

height/2-y1 au lieu de height/2+y1

height/2-y2 au lieu de height/2+y2

1.6 ***(1pts)*** *Que faut-il modifier dans le programme original pour que l'escargot soit dessiné en pointillé.*

i=i+2 dans la boucle for à la ligne 8

2 Traitement d'image (5 pts)

Soit la fonction suivante qui copie une version plus **rouge** d'une image (src) dans une autre (dst) de même taille sans tenir compte de la transparence de l'image src :

```
void blitimg(PImage dst, PImage src) {
  dst.loadPixels();
  for (int j=0; j<src.height; j++) {
    for (int i=0; i<src.width; i++) {
      int c = src.pixels[i+j*src.width];
      float r = red(c);
      float g = green(c);
      float b = blue(c);
      float r2= 255-(255-r)/2; // C'est ICI qu'on augmente le rouge
      dst.pixels[i+j*dst.width] = color(r2,g,b, 255) ;
    }
  }
  dst.updatePixels();
}
```

2.1 ***(3pt)*** *En Processing la fonction dist(x1, y1, x2, y2) permet de calculer la distance euclidienne entre deux points. Modifier le code précédent pour calculer la distance d au centre de l'image (src.width/2, src/height/2) puis appliquer un dégradé radiale de l'augmentation du rouge (au centre l'image sera plus rouge et sur les bord elle sera normale entre les deux les pixels seront de moins en moins modifiés au fur et à mesure qu'on s'éloignera du centre)*

```
float d = dist(i, j, src.width/2, src.height/2)/(src.height/2);
// ATTENTION ON NOMALISE LA DISTANCE
dst.pixels[i+j*dst.width] =
  color(r2*(1.0-d)+r*d,g,b, 255);

ou bien

float r2= 255-(255-r)/(2-d);
```

2.2 ***(2pt)*** *Que faut-il modifier pour que la transparence du résultat soit inversement proportionnelle à la distance au bord du haut (attention le 4ieme paramètre de la fonction* ***color*** *est l'opacité qui est l'inverse de la transparence)*

```
dst.pixels[i+j*dst.width] =
  color(r2,g,b, 255*(dst.height-j)/dst.height) ;
```

3 Animation (6 pts)

Soit la fonction suivante qui anime une centaine de petites bestioles grouillantes de façon aléatoire à l'écran.

```
void setup (){
  size(800, 800);
}
void drawBug(){
  fill(0);
  stroke(0);
  ellipse(-16/2-1, 0, 16, 6);
  ellipse(1, 0, 6, 6);
  line(-0*16/3, 0, 0*16/3-3, -6);
  line(-1*16/3, 0, 1*16/3-3, -6);
  line(-2*16/3, 0, 2*16/3-3, -6);
  line(-0*16/3, 0, 0*16/3-3, 6);
  line(-1*16/3, 0, 1*16/3-3, 6);
  line(-2*16/3, 0, 2*16/3-3, 6);
}
```

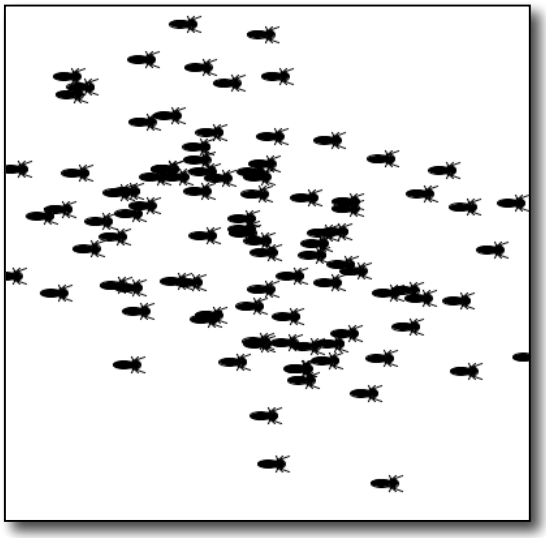


Figure 2 *: résultat du code*

```
void draw(){
  background(255);
  t=t+0.005;
  for (int i=0; i<100; i++){
    float x= noise(i,t-0.005)*width;
    float y= noise(i-0.5, t-0.005)*height;
    pushMatrix();
    translate(x2, y2);
    drawBug();
    popMatrix();
  }
}
```

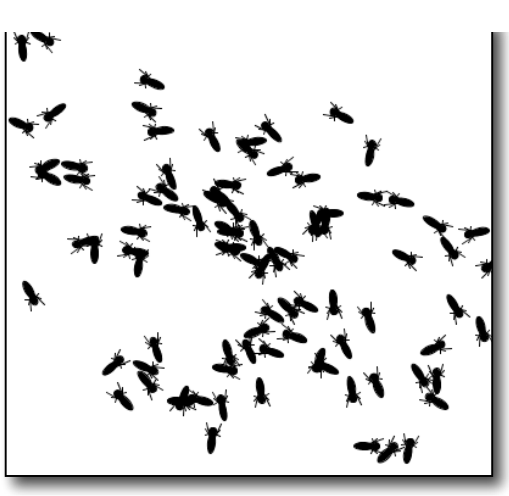


Figure 3 *: résultat de la question 3.2*

3.1 ***(1pts)*** *Proposer une modification du code pour les bestioles aient la même trajectoire mais deux fois plus rapide*

t=t+0.01 au lieu de t=t+0.005 ;

3.2 ***(3pts)*** *Proposer une modification du code pour les bestioles aient une orientation qui suivent leur trajectoire (pour l'instant elles sont toutes horizontales) cf figures 2 et 3*

on calcule la position a l'instant d'avant
float x1= noise(i,t-0.005)*width;
float y1= noise(i-0.5, t-0.005)*height;
et on rajoute un rotate en plus du translate
rotate(atan2(y-y1, x-x1));

3.3 ***(2pts)*** *Proposer une modification du code initiale pour que chaque bestiole ait sa propre vitesse (la plus lente sera immobile et la plus rapide se déplacera aussi vite qu'a la question 3.1)*

on ralentit toutes les bestioles en jouant encore
une fois sur l'incrément de t
t+=0.0001;
puis on remplace tous les t par t*i qui accélèrera
les dernières affichées mais pas les premières