Examen 2020-2021 (2h, barème /20)

Vous pouvez répondre sur cette feuille que vous glisserez dans votre copie d'examen. Vous prendrez soin de recopier votre numéro d'étudiant dans les cases ci-dessus.
Nous vous conseillons d'utiliser le crayon de papier sur votre copie d'examen.

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 draw(){
2   noFill();
3   strokeWeight(2);
4   beginShape();
5   for (int i=0; i<=40; i++){
6     vertex(i*10+random(-5,5),
             i*10+random(-5,5));
7   }
8   endShape();
9 }
```

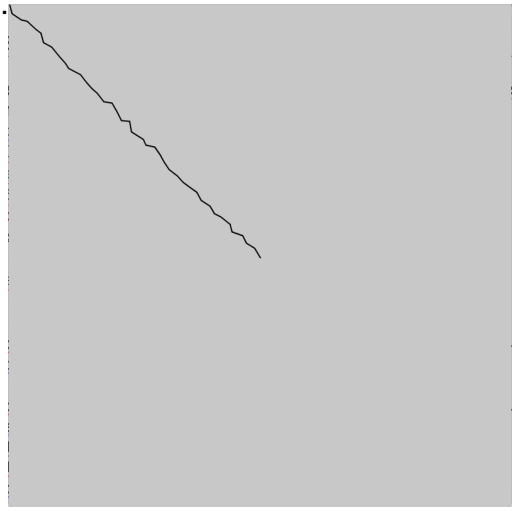


Figure 1 : *Résultat graphique*

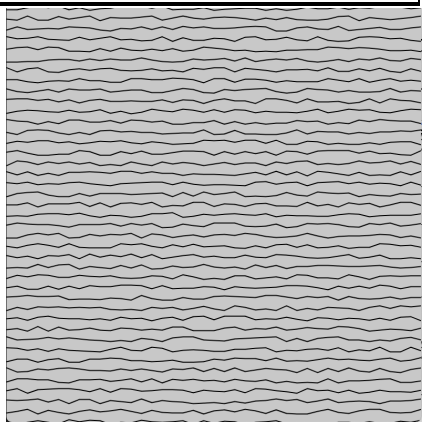
1. (1pt) Quelle est la taille de la fenêtre carrée pour que le trait la traverse à moitié ?

800x800

1.1 ***(1.5pt)*** *Proposez une règle de proportion (règle de 3) qui fasse en sorte que le trait traverse toute la fenêtre quelle que soit sa taille spécifiée dans setup()?*

```
vertex(  // à la place de la ligne 6
    i*width/40+random(-5,5),
    i*height/40+random(-5,5)
);
```

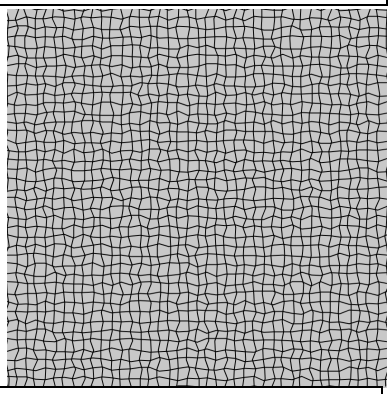
1.2 ***(2.5pts)*** *Proposez une modification du programme pour obtenir le nouveau dessin ci-contre qui affiche 40 lignes horizontales.*



```
void draw(){
  noFill();
  strokeWeight(2);
```

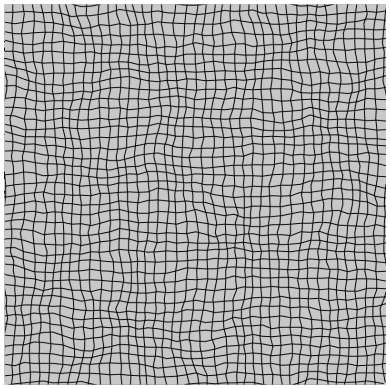
```
for (int j=0; j<=40; j++){
  beginShape();
  for (int i=0; i<=40; i++){
    vertex(i*width/40, j*height/40+random(-5,5));
  }
  endShape();
  //*
}
```

1.3 ***(3pts)*** *Complétez votre code en mettant un astérisque dans votre code au dessus puis en proposant ci-dessous ce qu'il faut rajouter pour obtenir la grille de 40x40 lignes comme cela :*



```
  beginShape();
  for (int i=0; i<=40; i++){
    vertex(j*width/40+random(-5,5), i*height/40);
  }
  endShape();
```

1.4 ***(1pts)*** *Par quelle autre fonction a t-on remplacé random() pour obtenir le résultat ci-contre ?*



Par la fonction noise car le
Résultat est beaucoup plus continu

2 Traitement d'image (4 pts)

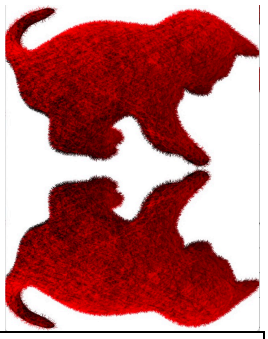
Soit la fonction suivante qui produit une image miroir horizontale :

```
1  PImage fonction(PImage img1){
2    PImage img2 = createImage(img1.width, img1.height, ARGB);
3    for(int j=0; j < img1.height; j++) {
4      for(int i=0; i < img1.width; i++) {
5        color p = img1.pixels[(img1.width-1-i)+j*img1.width];
6        img2.pixels[i+j*img2.width] = p;
7      }
8    }
9    return img2;
10 }
```

Dont l'utilisation ci-dessous produit cela :

```
PImage img0, imgR;
void setup(){
  size(623, 800);
  img0 = loadImage("img.jpg");
  imgR = fonction(img0);
  noLoop();
}
void draw(){
  background(255);
  image(img0, 0, 0);
  image(imgR, 0, height/2);
}
```

2.1 ***(2pt)*** *Proposez une modification de ce code pour faire une image miroir dans l'autre sens(verticale).*



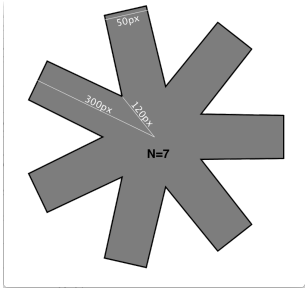
color p = img1.pixels[
i+(img1.height-1-j)*img1.width];
//en modifiant juste la ligne 5

2.2 ***(2pt)*** *Que faut-il modifier pour que la transparence soit proportionnelle à la distance au bord du haut (opacité à 128 en haut et à 0 en bas) ATTENTION : vous pourrez extraire les composantes RGB du pixel p avec les fonction red(), green() et blue()*

float r = red(p); //
float g = green(p);
float b = blue(p);
img2.pixels[i+j*img2.width] = color(r,g,b, 128-j*128/(height/2));

3 Etoile (7 pts)

Écrire une fonction draw() en Processing qui dessine une étoile épaisse à partir de 4 paramètres N, INNER, OUTER et TOPER. Voici un exemple ou N=7, INNER=120, OUTER=300 et TOPER=50 ATTENTION : vous utiliserez la même fonction de dessin de polygone que l'exercice 1. Il vous faudra sans doute dessiner les



points 3 par trois …

```
int TOPER = 50;
int INNER = 120;
int OUTER = 300;
int N = 7;

void draw(){
  background(255);
  strokeWeight(3);
  fill(128);
  beginShape();
  for(int i=0; i<N; i++){
    float xF = width/2 + OUTER *cos(i*PI*2/N);
    float yF = width/2 + OUTER *sin(i*PI*2/N);
    float xN = width/2 + INNER *cos((i+0.5)*PI*2/N);
    float yN = width/2 + INNER *sin((i+0.5)*PI*2/N);
    float dx = -TOPER*sin(i*PI*2/N);
    float dy = TOPER*cos(i*PI*2/N);
    vertex(xF-dx, yF-dy);
    vertex(xF+dx, yF+dy);
    vertex(xN, yN);
  }
  endShape(CLOSE);
```