Processing
P05

1- Image

Cédric MASSART

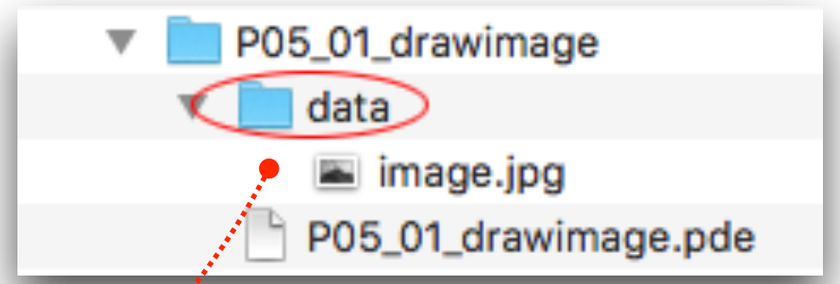
Image

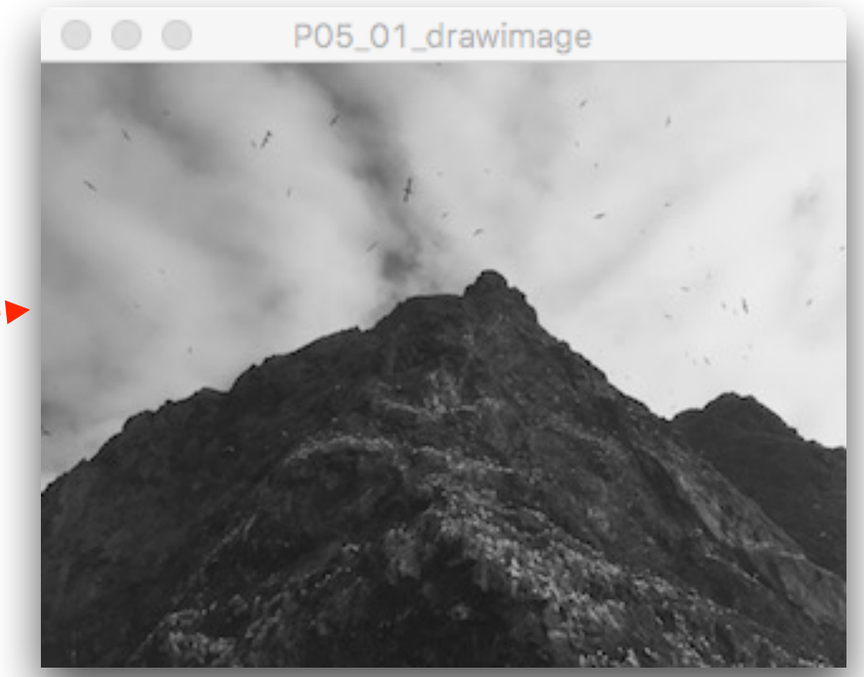
loadImage() et image()

chargement de l’image

```
PImage img;

void setup() {
  size(320, 240);
  img = loadImage("image.jpg");
}

void draw() {
  background(0);
  image(img, 0, 0, width, height);
}
```

dessine l’image





P05_01_drawimage.pde

Animation (type Sprite)

```
PImage img;
float x,y;
float rot;

void setup() {
  size(500,500);
  img = loadImage("image.jpg");
  x = 0.0;
  y = width/2.0;
  rot = 0.0;
  imageMode(CENTER);
}
void draw() {
  background(255);
  translate(x,y);
  rotate(rot);
  image(img,0,0);

  x += 1.0;
  rot += 0.02;

  if (x > width+img.width) {
    x = -img.width;
  }
}
```

chargement de l'image

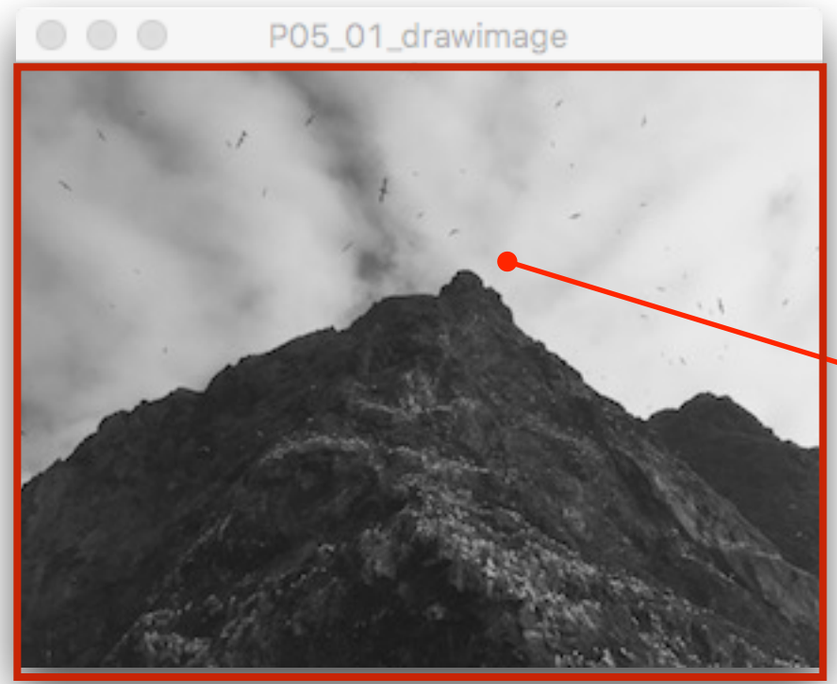
déplacement et rotation de l'image

dessine l'image

incrémentation des variables

condition pour le retour de l'image à droite de la fenêtre

Liste de pixels



0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

Affichage des pixels

Stockage des pixels dans la mémoire

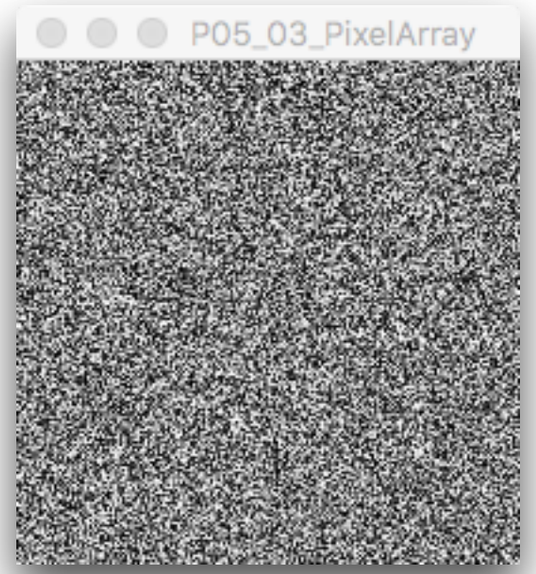
0	1	2	3	4	5	6	7	8	9	.	.	.		

Liste de pixel (Tableau)

chargement des pixels de la fenêtre dans une liste

```
size(200, 200);

loadPixels();

for (int i = 0; i < pixels.length; i++ ) {
  float rand = random(255);
  color c = color(rand);
  pixels[i] = c;
}
updatePixels();
```

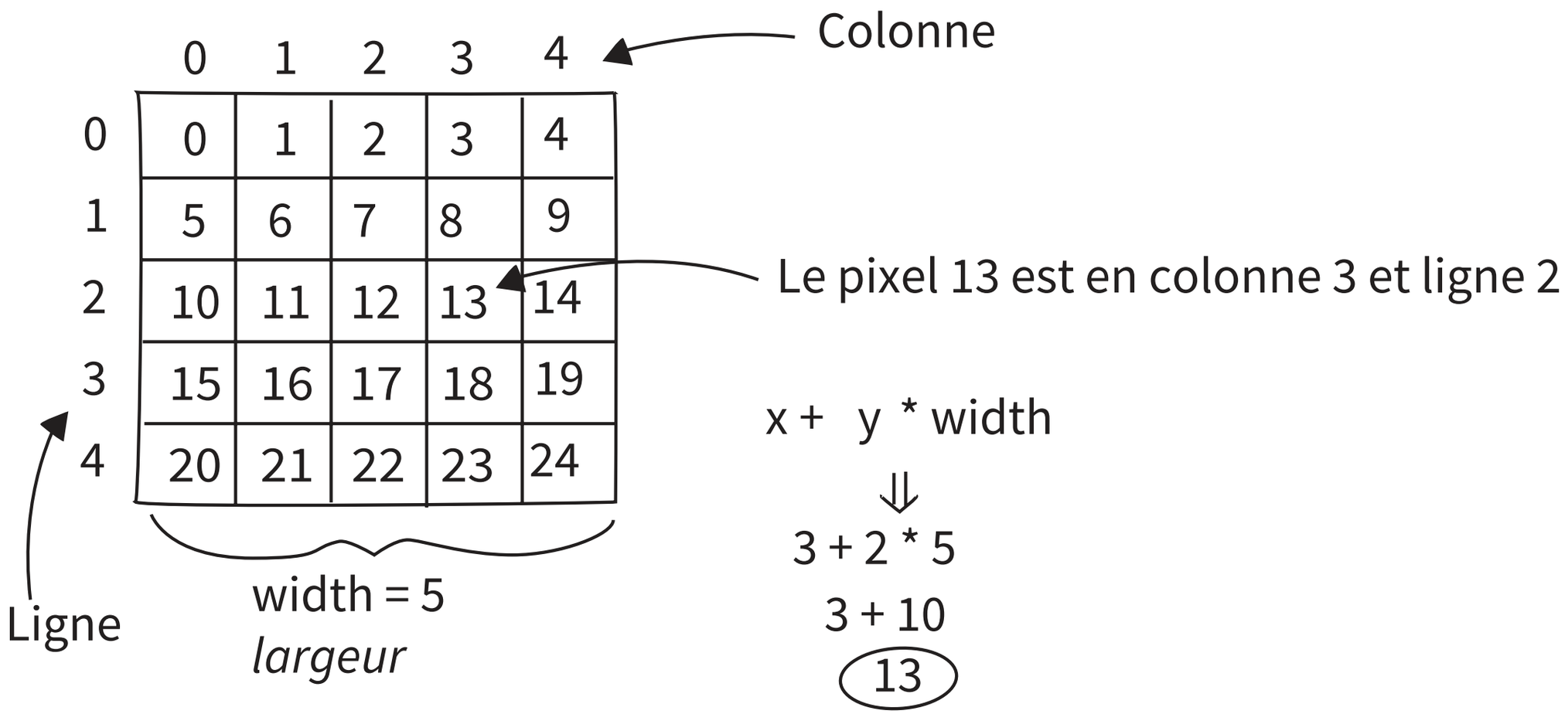
parcours de tous les pixels de la liste



mettre la couleur à la position du pixel

affichage dans la fenêtre de la liste de pixels

Calcul de position du pixel en 2D



Traitement en 2D

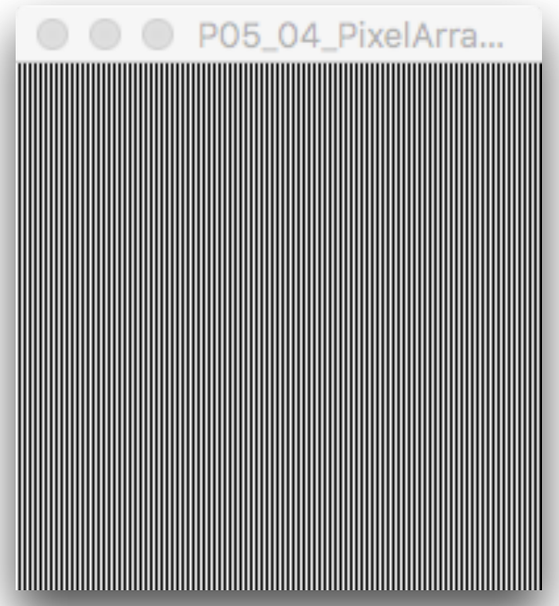
```
size(200, 200);
loadPixels();

for (int x = 0; x < width; x++ ) {
  for (int y = 0; y < height; y++ ) {
      int loc = x + y * width;
      if (x % 2 == 0) {
      pixels[loc] = color(255);
    } else {
      pixels[loc] = color(0);
    }
  }
}

updatePixels();
```

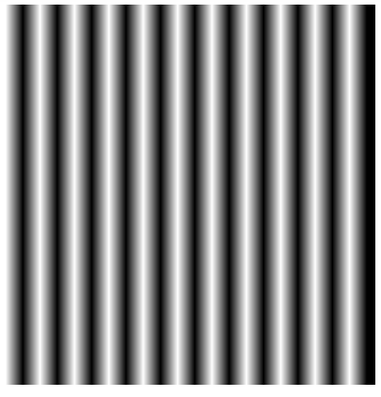
double boucle
horizontale (x)
et verticale (y)

calcul de la position

une colonne sur deux

affectation de la couleur
du pixel dans la liste





P05_04_PixelArray2D.pde

Afficher une image

```
PImage img;

void setup() {
  size(200, 200);
  img = loadImage("sunflower.jpg");
}

void draw() {
  loadPixels();

  img.loadPixels();
  for (int y = 0; y < height; y++ ) {
    for (int x = 0; x < width; x++ ) {
      int loc = x + y*width;

      float r = red(img.pixels [loc]);
      float g = green(img.pixels[loc]);
      float b = blue(img.pixels[loc]);

      pixels[loc] = color(r, g, b);
    }
  }

  updatePixels();
}
```

double boucle
horizontale (x)
et verticale (y)

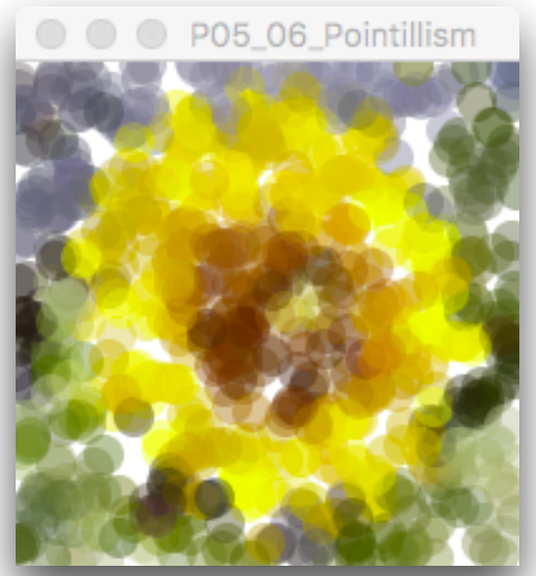
calcul de la position

lecture des composantes RVB

affectation de la couleur
du pixel dans la liste

Principe (1/5)

```
PImage img;
int pointillize = 16;

void setup() {
  size(200,200);
  img = loadImage("sunflower.jpg");
  background(255);
  smooth();
}

void draw() {
  int x = int(random(img.width));
  int y = int(random(img.height));
  int loc = x + y*img.width;

  loadPixels();
  float r = red(img.pixels[loc]);
  float g = green(img.pixels[loc]);
  float b = blue(img.pixels[loc]);
  noStroke();

  fill(r,g,b,100);
  ellipse(x,y,pointillize,pointillize);
}
```



tirage aléatoire sur l'axe des x et y

calcul de la position

lecture des composantes RVB

couleur de remplissage

dessine une ellipse