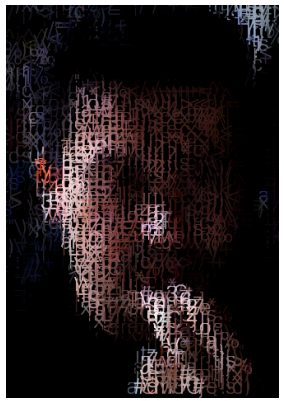
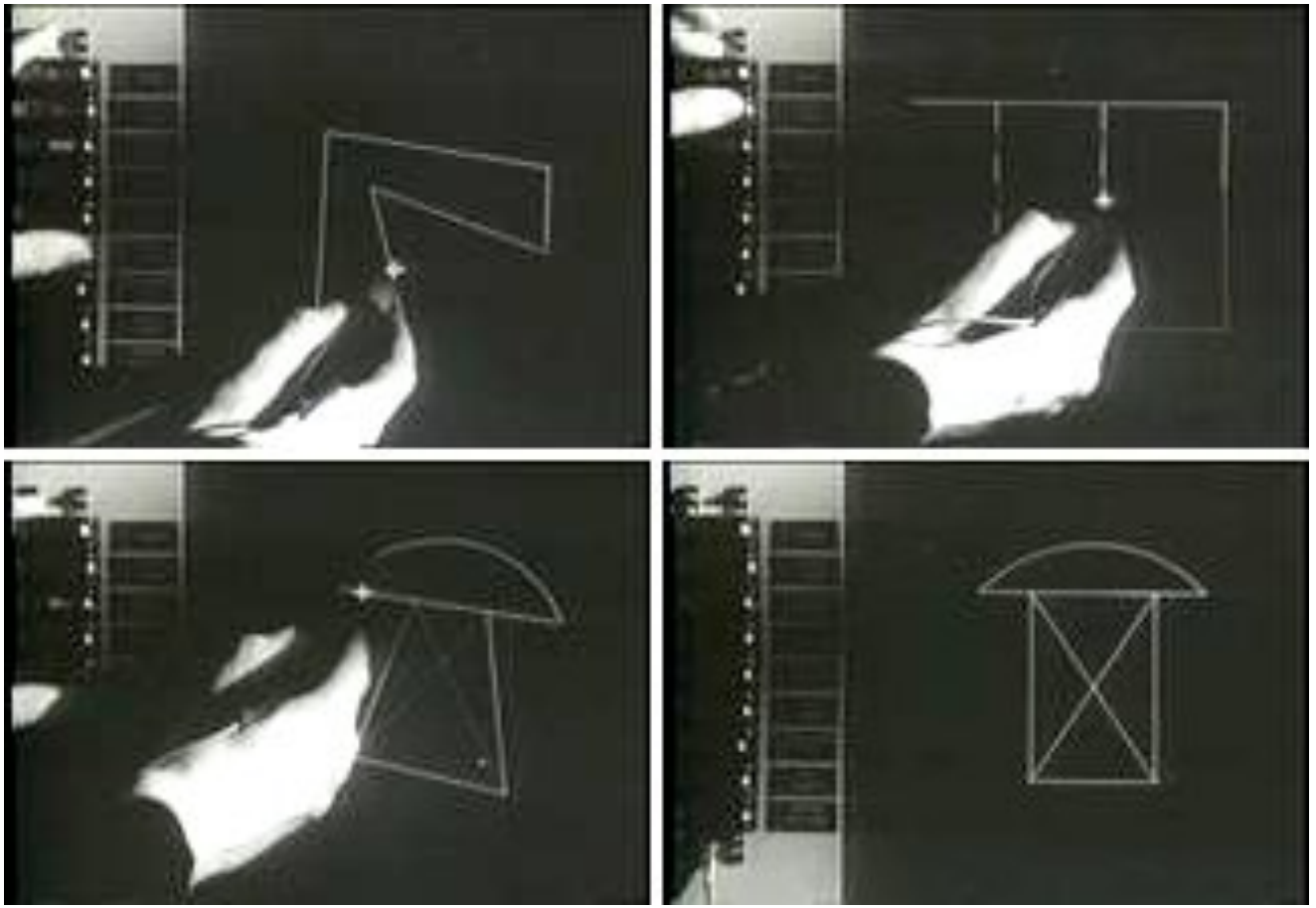
CNAM - dept. Informatique - DUT1 – USAL24

IHM(3) – Prog. interactive

Pierre Cubaud, CNAM
cubaud @ cnam.fr



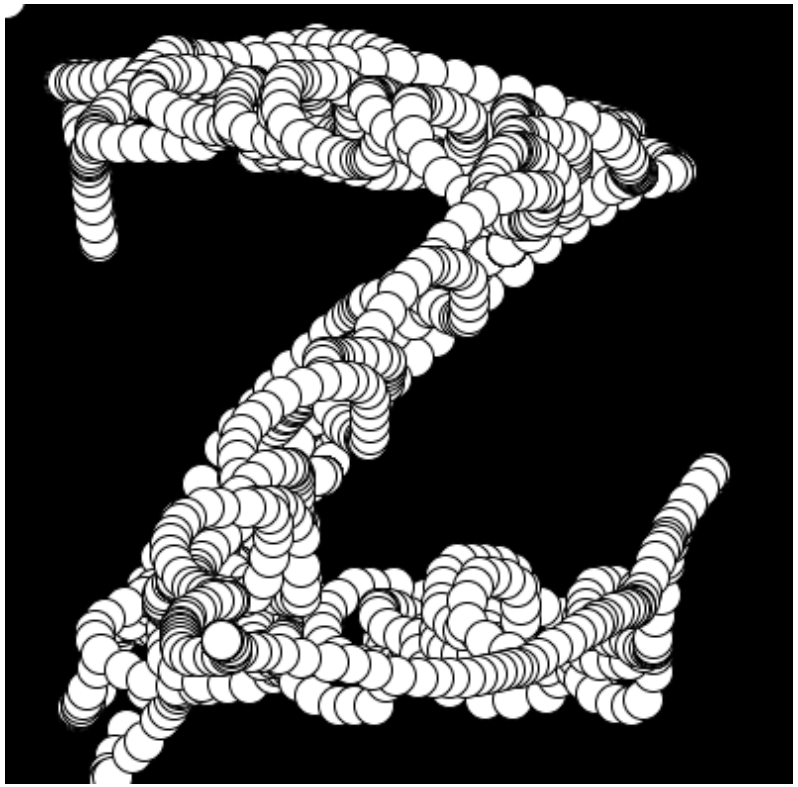
le cnam



Y. Sutherland
sketchpad

Souris, curseur

```
void setup(){
  size(400,400);
  smooth();
  background(0);
  noCursor();
}

void draw(){
//  background(0);
  x = mouseX;
  y = mouseY;
  ellipse(x,y,20,20);
}
```



(demosouris.pde)

Les évenements en Processing

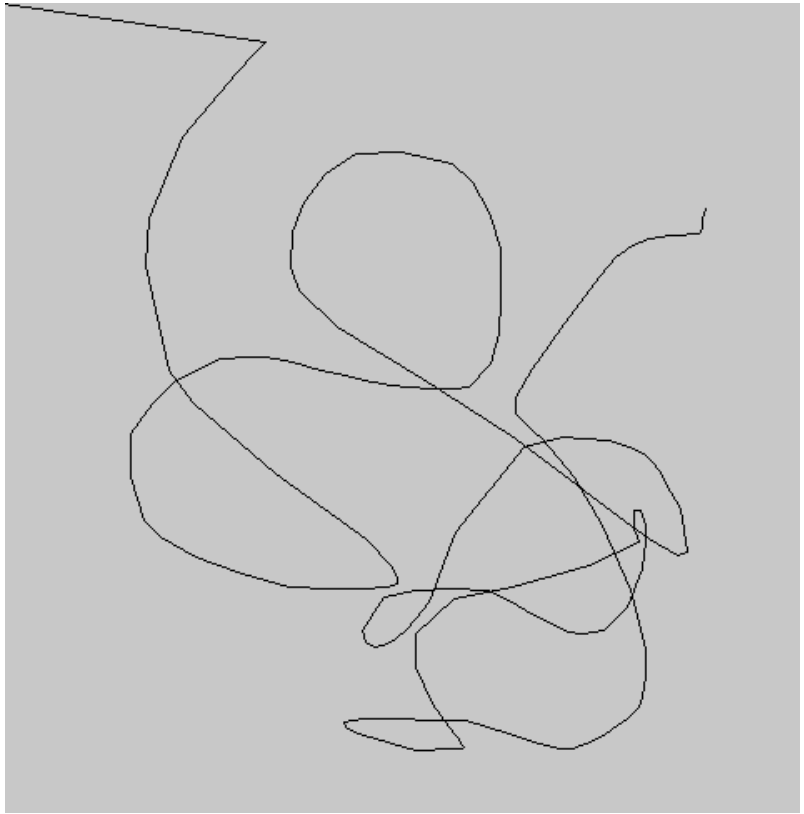
void mousePressed() {…}
mouseReleased()
mouseMoved()
mouseDragged()
keyPressed()
keyReleased()

loop(), noLoop(), redraw()

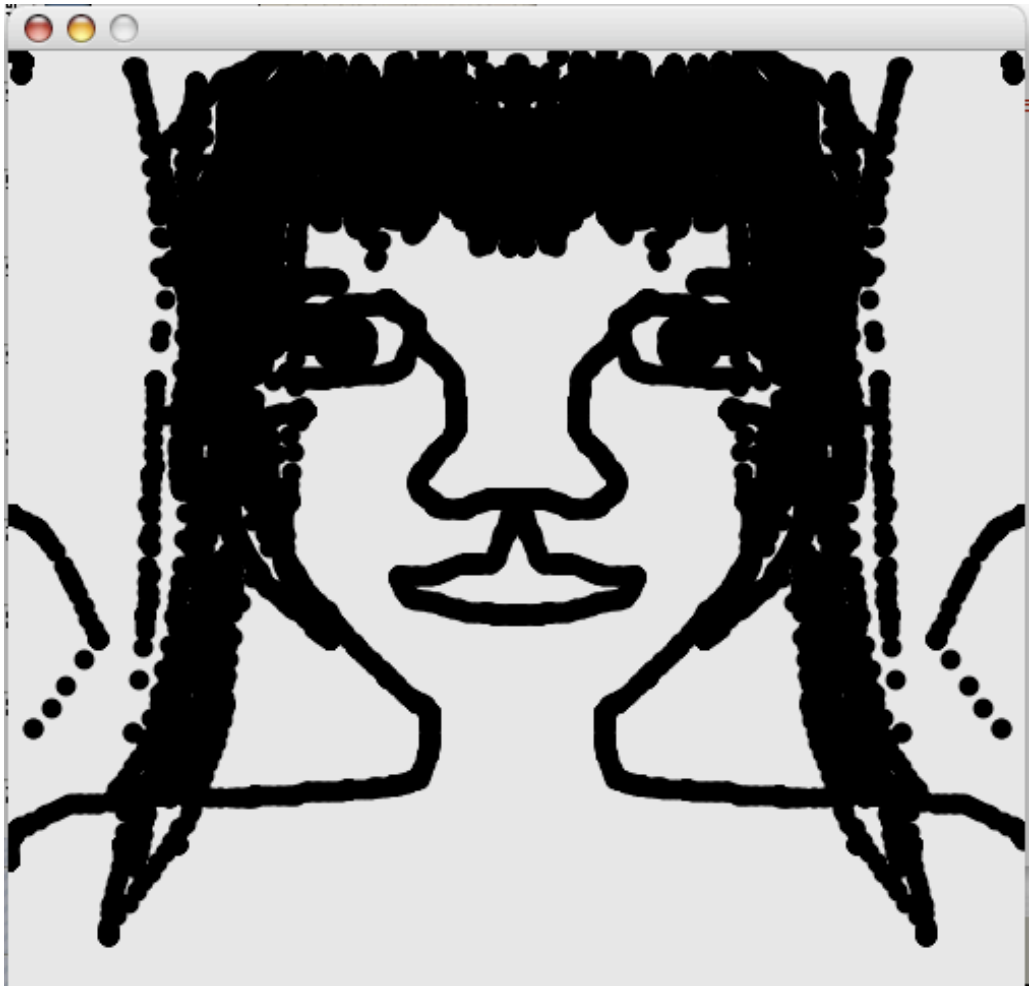
pmouseX, pmouseY

linepmouse

```
void setup(){
  size(600,600);
}
void draw(){
  line(pmouseX,pmouseY,mouseX,mouseY);
}
```



Symétries

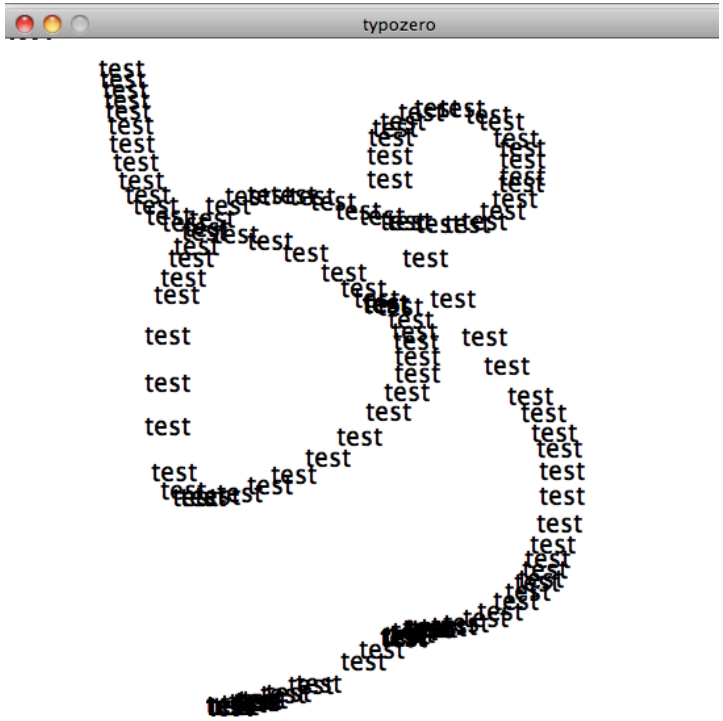


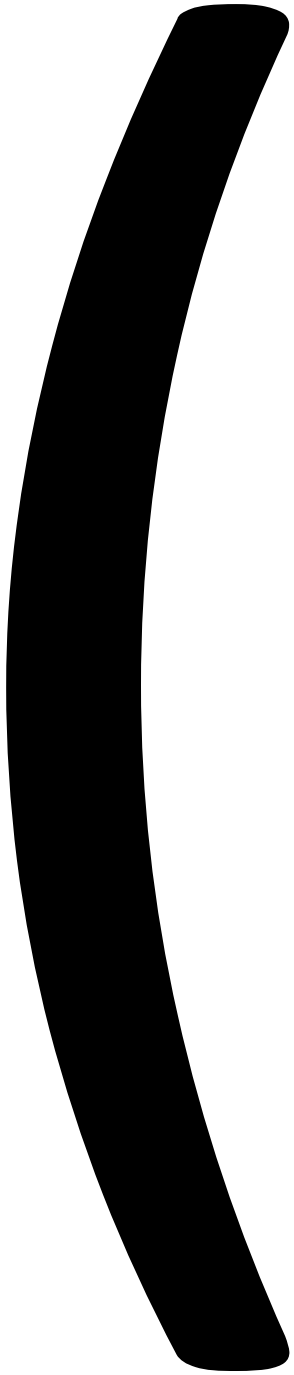
```
boolean dessin = false;

void setup() {
  size(500, 500);
  background(226);
  smooth();
  strokeWeight(20); stroke(0, 100);
}

void draw() {
  point(mouseX,mouseY);
  point(width-mouseX, mouseY);

  //// variante pour symetrie centrale
  //point(mouseX,height-mouseY);
  //point(width-mouseX,height-mouseY);

}
```

(symetries.pde)

typozero

```
  background(255);
  fill(0);
  smooth();
  font = createFont("Bauhaus93-48",10);
  textFont(font,20);
  textAlign(LEFT);
}

void draw() {
  int x = mouseX;
  int y = mouseY;
  text("test", x, y);
}
```





Les images avec Processing

```
Pimage img = loadImage("gnagna.jpg");
image(img, 30,40, 640, 480);
```

Coloriage de l'image : tint(…) noTint()

Accès aux pixels de l'écran : get(…) set(…) copy(…)
Pour une image : ima.get(…) ima.set(…)

filter(…) blend(…) mask(…)

loadPixels(…) Pixel[] updatePixels()

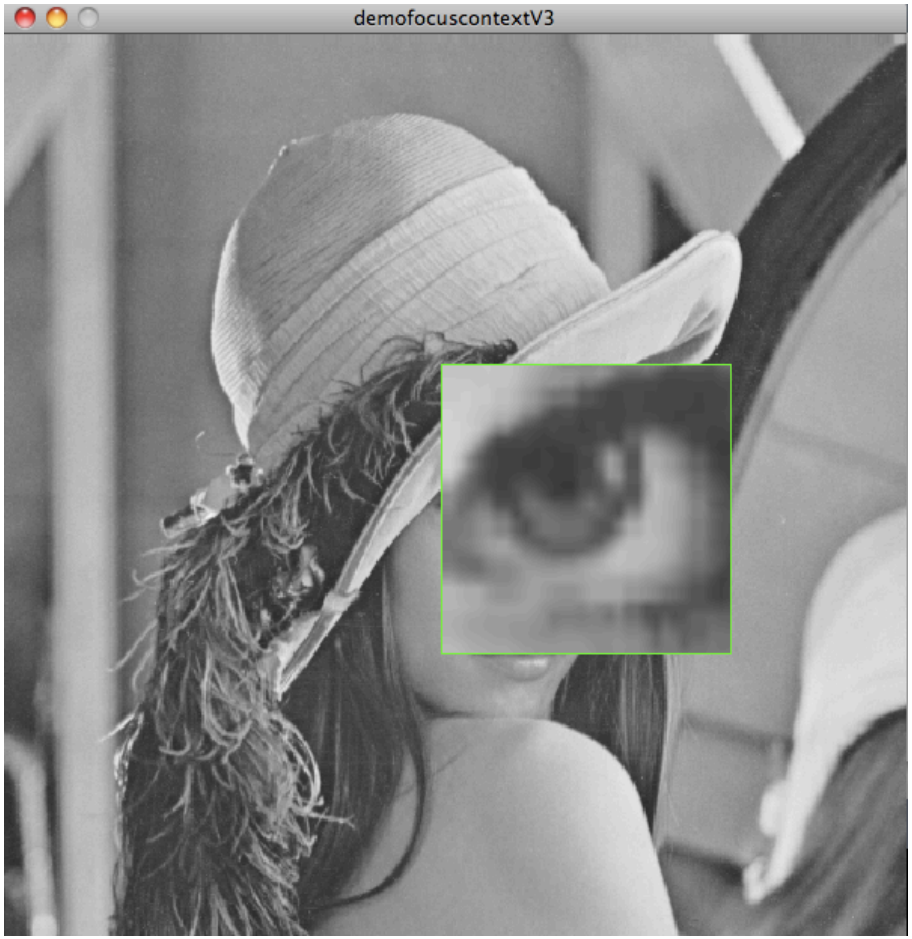
Hommage à Akaklikè

Processing - 0125 Beta

akaklike

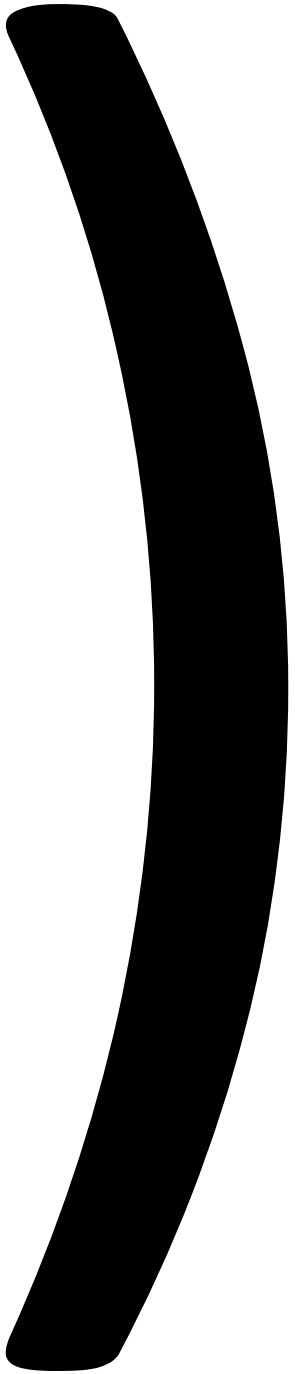
```
PImage ima;

void setup(){
  ima=loadImage("Photo.jpg");
  size(ima.width,ima.height);
}

void draw(){
  copy(ima,mouseX,mouseY,30,30,mouseX,mouseY,30,30);
}
```

Loupe

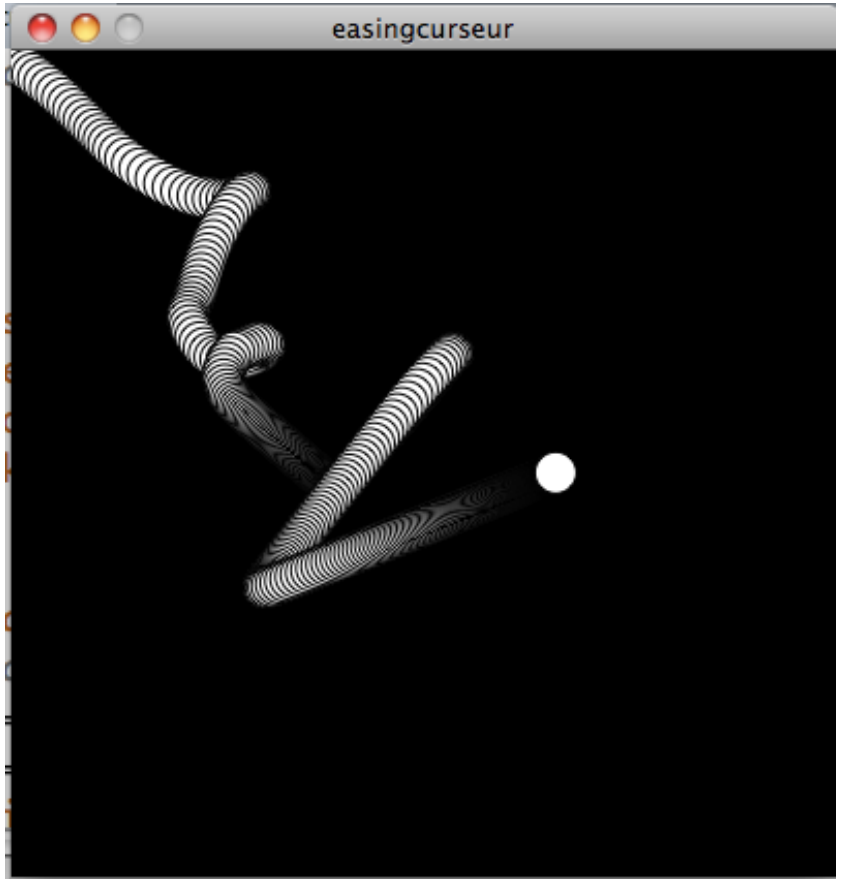


```
PImage pgmat;
float ar;

void setup(){
 pgmat = loadImage("lenna.gif");
 ar = pgmat.width/float(pgmat.height);
 size(int(600*ar), 600);
}

void draw(){
  int x = mouseX;
  int y = mouseY;
  image(pgmat, 0,0, width,height);
  copy(x-16,y-16,32,32, x-96, y-96, 192,192);
  stroke(0,255,0);noFill();rect(x-96, y-96, 192,192);
}
```

(demofocuscontextV3.pde)



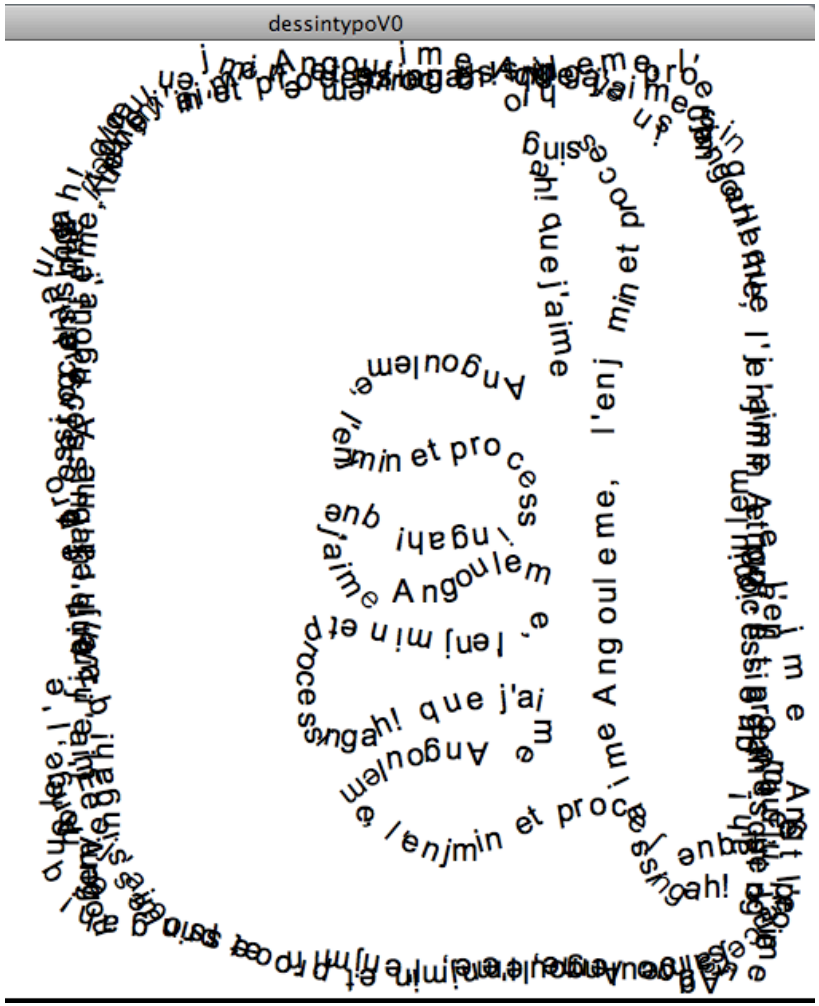
Easing

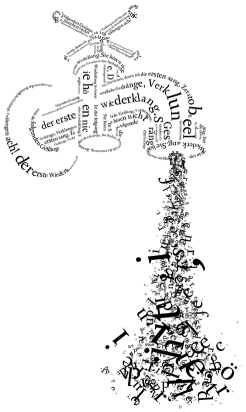


easingcurseur

```
// d'apres Geridan & Lafargue "Processing"
float x=0;
float y=0;
float ease=0.1;

void setup(){
  size(400,400);
  smooth();
  background(0);
}

void draw(){
  background(0);
  x += (mouseX -x)*ease;
  y += (mouseY -y)*ease;
  ellipse(x,y,20,20);
}
```

plusieurs autres
méthodes (fonction sinus etc)





d'après programme P.2.3.3 de "Design génératif"

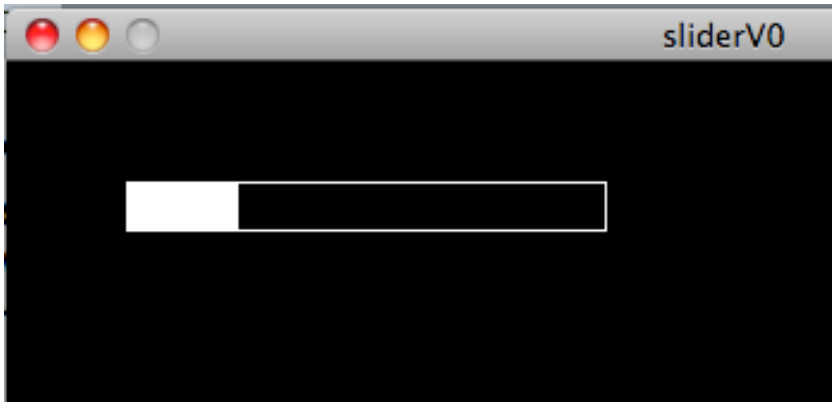
```
float x, y, ox, oy, d;
float stepSize;
PFont font;
String letters = "ah! que j'aime Angouleme, l'enjmin et processing";
int counter = 0;

void setup() {
  size(600,600);
  background(255);
  fill(0);
  smooth();
  cursor(CROSS);
  font = createFont("Arial",10);
  textFont(font,20);
  textAlign(LEFT);
  stepSize = 0;
  x = mouseX;
  y = mouseY;
  ox = x;
  oy = y;
}

void draw() {
  x = mouseX;
  y = mouseY;
  d = dist(x,y, ox,oy);

  if (d >= stepSize) {
   float angle = atan2(y-oy, x-ox);
   pushMatrix();
   translate(x, y);
   rotate(angle);
   //textFont(font,fontSizeMin+d);
   char newLetter = letters.charAt(counter);
   text(newLetter, 0, 0);
   counter = (counter+1)%letters.length();
   popMatrix();

   stepSize = textWidth(newLetter);
   ox = x;
   oy = y;
  }
}
```

(dessintypoV0.pde)

Exemple d'un slider

sliderV0 §

```
// un slider minimal
int sx=50; // coordonnees du slider
int sy=50;
int sw=200; // longueur
int sh=20; // hauteur

void setup(){
  size(600,600);
  stroke(255);fill(255);
}

void draw(){
  background(0);
  float sv = map(mouseX,0,width,0,1);
  fill(255);noStroke();
  rect(sx,sy,sw*sv,sh);
  stroke(255);noFill();
  rect(sx,sy,sw,sh);
}
```



problème :
il faut modifier la valeur seulement quand le curseur entre dans la zone du slider

Mieux :

slider

```
}

void draw(){
  background(0);
  fill(255);noStroke();
  rect(sx,sy,sw*sv,sh);
  stroke(255);noFill();
  rect(sx,sy,sw,sh);
}

void mousePressed(){
  boolean okx = (mouseX >= sx) && (mouseX <= sx+sw);
  boolean oky = (mouseY >= sy) && (mouseY <= sy+sh);
  if (okx && oky) sv = map(mouseX-sx,0,sw,0,1);
}
```

début du code inchangé
(sauf déclaration de sv à faire)

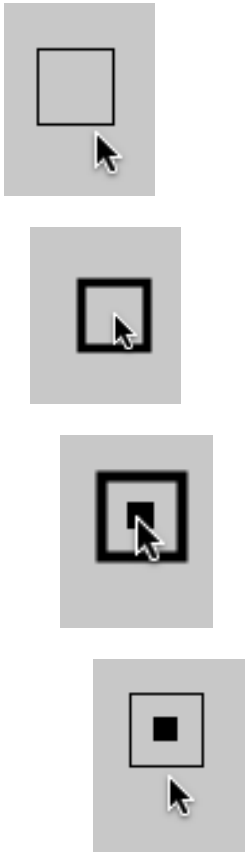
Encore mieux : un retour à l'utilisateur

sliderV2

```
void draw(){
  background(0);
  fill(255);noStroke();
  rect(sx,sy,sw*sv,sh);
  if (mouseIN) strokeWeight(3); else strokeWeight(1);
  stroke(255);noFill();
  rect(sx,sy,sw,sh);
}

void mousePressed(){
  boolean okx = (mouseX >= sx) && (mouseX <= sx+sw);
  boolean oky = (mouseY >= sy) && (mouseY <= sy+sh);
  if (okx && oky) sv = map(mouseX-sx,0,sw,0,1);
}

void mouseMoved(){
  boolean okx = (mouseX >= sx) && (mouseX <= sx+sw);
  boolean oky = (mouseY >= sy) && (mouseY <= sy+sh);
  mouseIN = (okx && oky);
}
```

début du code inchangé
+ booléen mouseIN initialisé false

Dessine-moi un bouton (radio)



état 1 : non sélectionné, non désigné

entrée de zone : "roll over"

état 2 : non sélectionné, désigné

clic

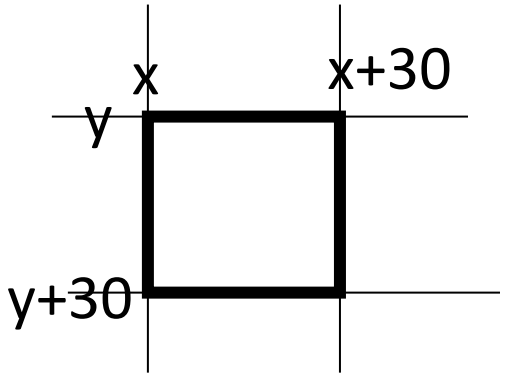
état 3 : sélectionné, désigné

sortie de zone

état 4 : non sélectionné, désigné

Détection du rollover :

bouton = un carré
en coord (x,y) de largeur
30 pixels




```
si    (mouseX > x) et (mouseX < x+30)
      et (mouseY >y) et (mouseY < y+30)
alors le curseur est dans la boite du bouton
sinon il est dehors
```

⇒une variable booléenne ("boolean")
pour le rollover + une autre pour la selection

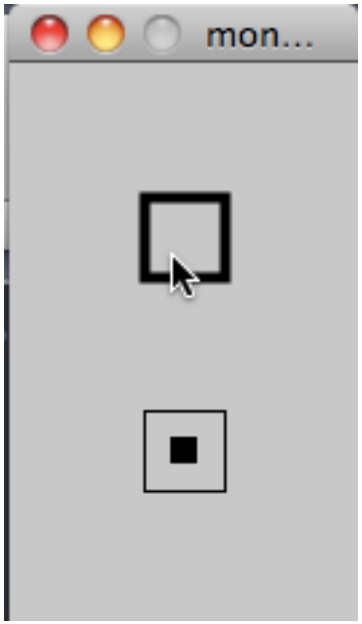
Utilisation dans le programme Processing

monboutonCLASSE | Processing 1.5.1

STANDARD

monboutonCLASSE | Bouton

```
Button b1,b2;

void setup() {
  size(130,210);
  smooth();
  b1 = new Button(50,50,false);
  b2 = new Button(50,130,true);
}

void draw() {
  background(250);
  b1.display();
  b2.display();
}

void mouseMoved() {
  b1.rollover(mouseX,mouseY);
  b2.rollover(mouseX,mouseY);
}

void mousePressed() {
  b1.clic(mouseX,mouseY);
  b2.clic(mouseX,mouseY);
}
```



Rque : mettre le code
de la classe dans un fichier
séparé est plus commode

Code complet de la classe

```
class Button {
  boolean selected = false;
  boolean rollover = false;
  float x,y;

  Button(float Px, float Py, boolean Pselected) {
    x = Px;
    y = Py;
    selected = Pselected;
  }

  void display() {
    stroke(0);noFill();
    if (rollover) strokeWeight(4);
    else strokeWeight(1);
    rect(x,y,30,30);
    if (selected) {
      noStroke();fill(0);
      rect(x+10,y+10,10,10);
    }
 }
```

```
  void rollover(int mx, int my) {
    if (mx > x && mx < x + 30 && my > y && my < y + 30)
      rollover = true;
    else
      rollover = false;
  }

  void clic(int mx, int my) {
    if (rollover) selected = ! selected;
  }

}
```

début du code

monboutonV0

```
int x,y;
boolean rollover, selected;

void setup() {
  size(200,200);
  x = 50; y = 50;
  rollover = false; selected = false;
}

void draw() {
  background(200);
  stroke(0);noFill();
  if (rollover) strokeWeight(4); else strokeWeight(1);
  rect(x,y,30,30);
  if (selected) {
    noStroke();fill(0);
    rect(x+10,y+10,10,10);
  }
}
```

Suite du code :

```
void mouseMoved() {
 int mx = mouseX;
 int my = mouseY;
 if (mx > x && mx < x + 30 && my > y && my < y + 30)
   rollover = true;
 else
   rollover = false;
}

void mousePressed() {
 if (rollover)
   selected = ! selected;
}
```

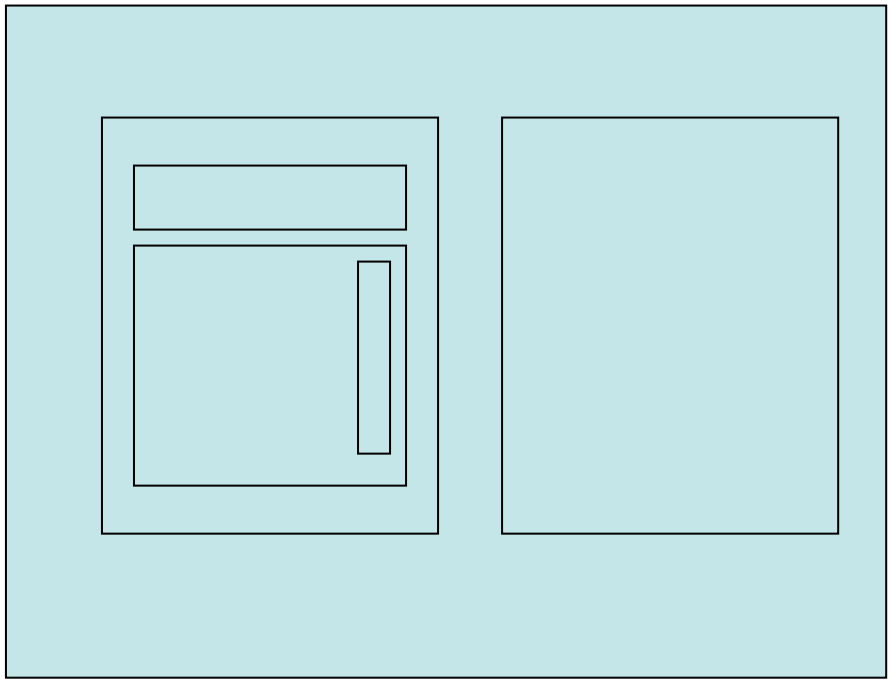
essayer d'autres
formes de boutons
et d'autres feedbacks

Conclusion à ce stade :

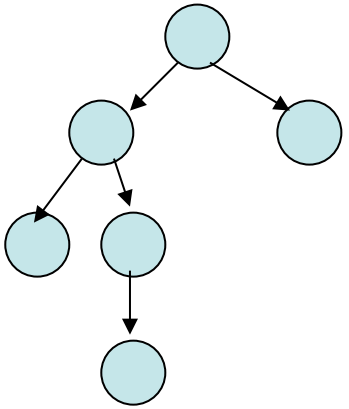
pour faire une IHM pour une application
complète, il faut des mécanismes plus avancés que ceux
vu ici

⇒ bibliothèques de fonctions (API) pour les interfaces
⇒ programmation orientée-objet
⇒ les Widgets

Une hiérarchie de boites …



...décrite par un arbre

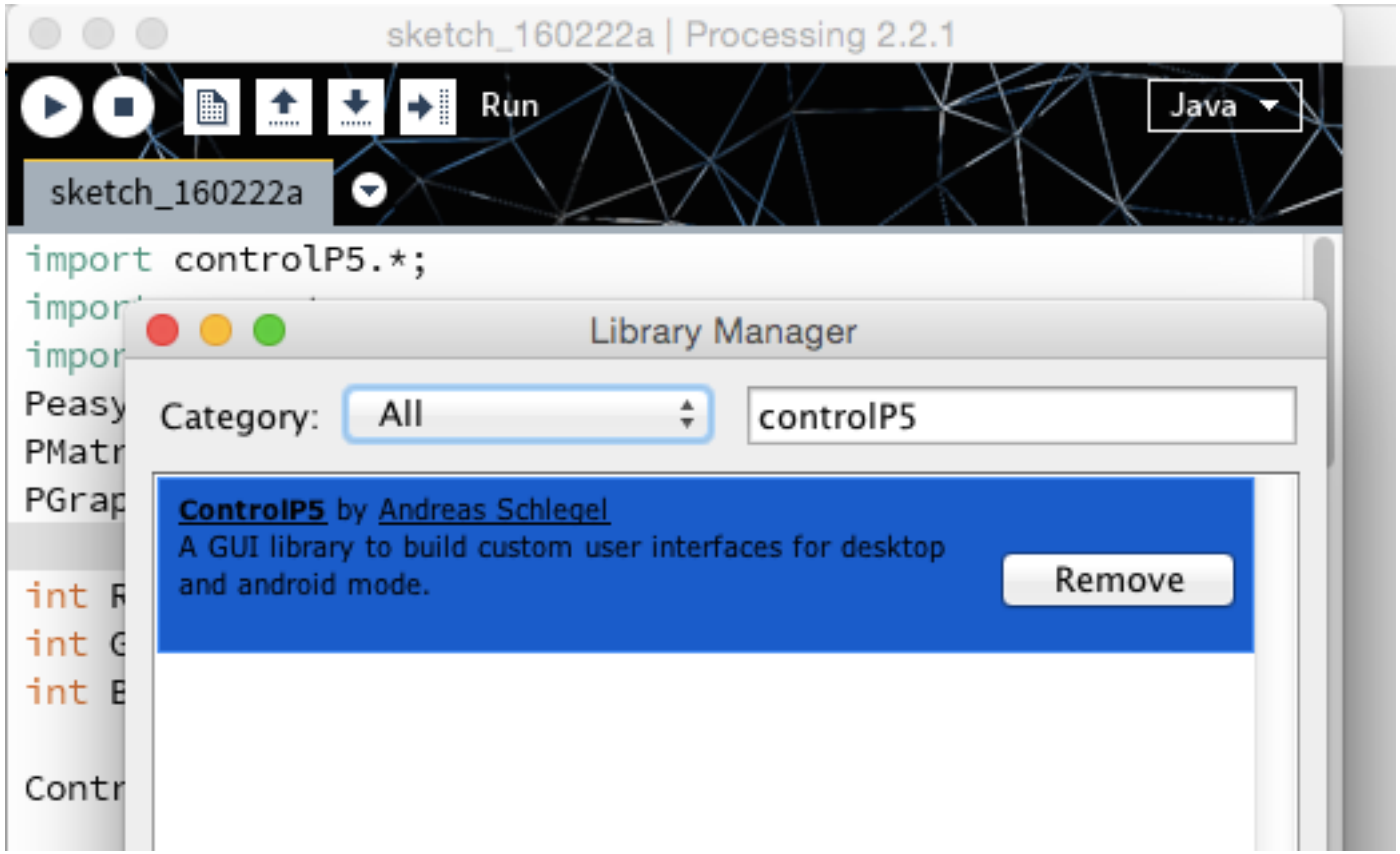


Le tracé des objets graphiques se fait dans des zones mémoire spécifiques ≠ écran (coût !)

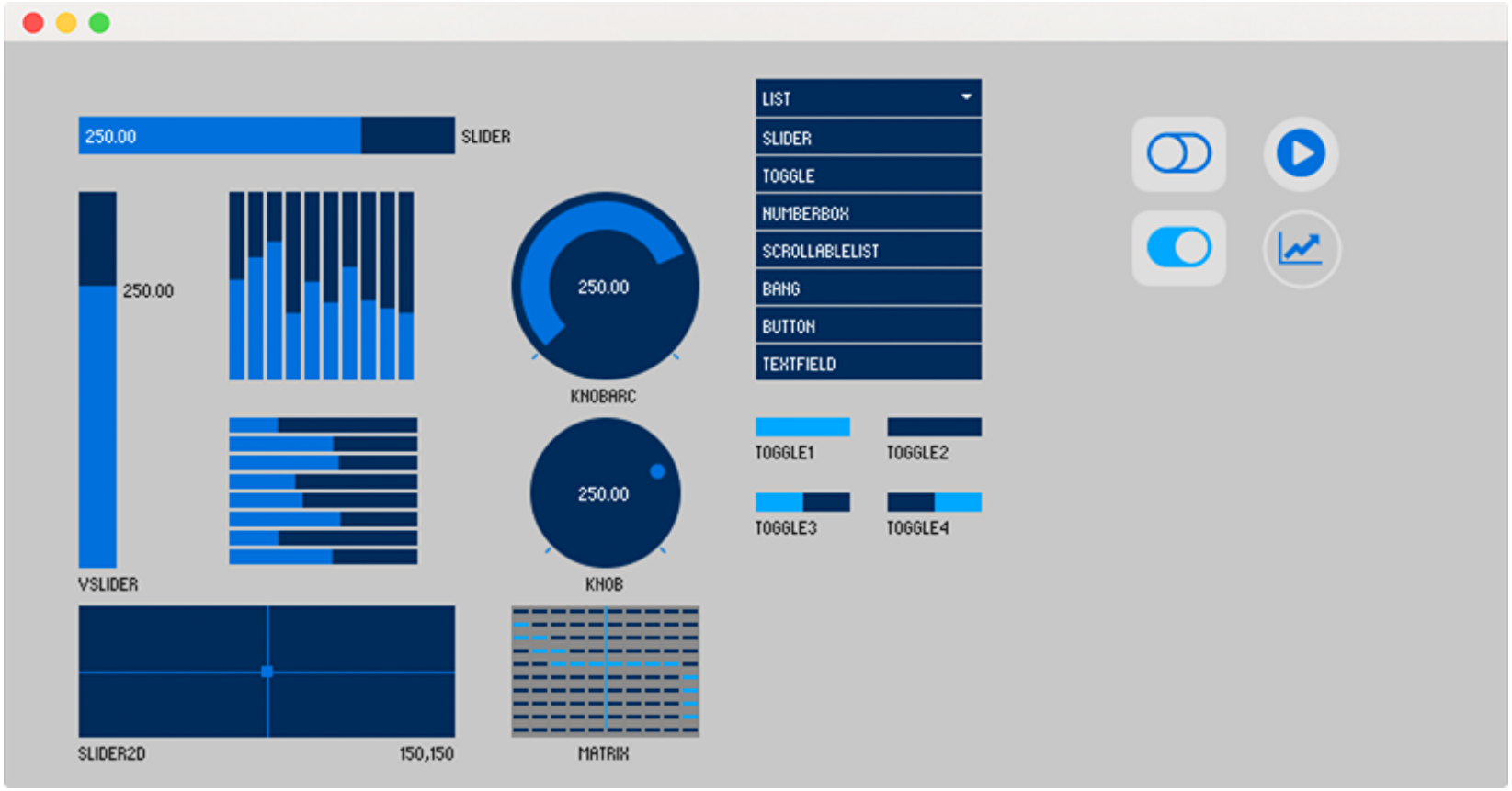
La géométrie des boites est spécifiée de manière abstraite (par contraintes, souvent)

La librairie controlP5

Pour installer la librairie : commande « import library » du menu « Sketch », puis « add library »



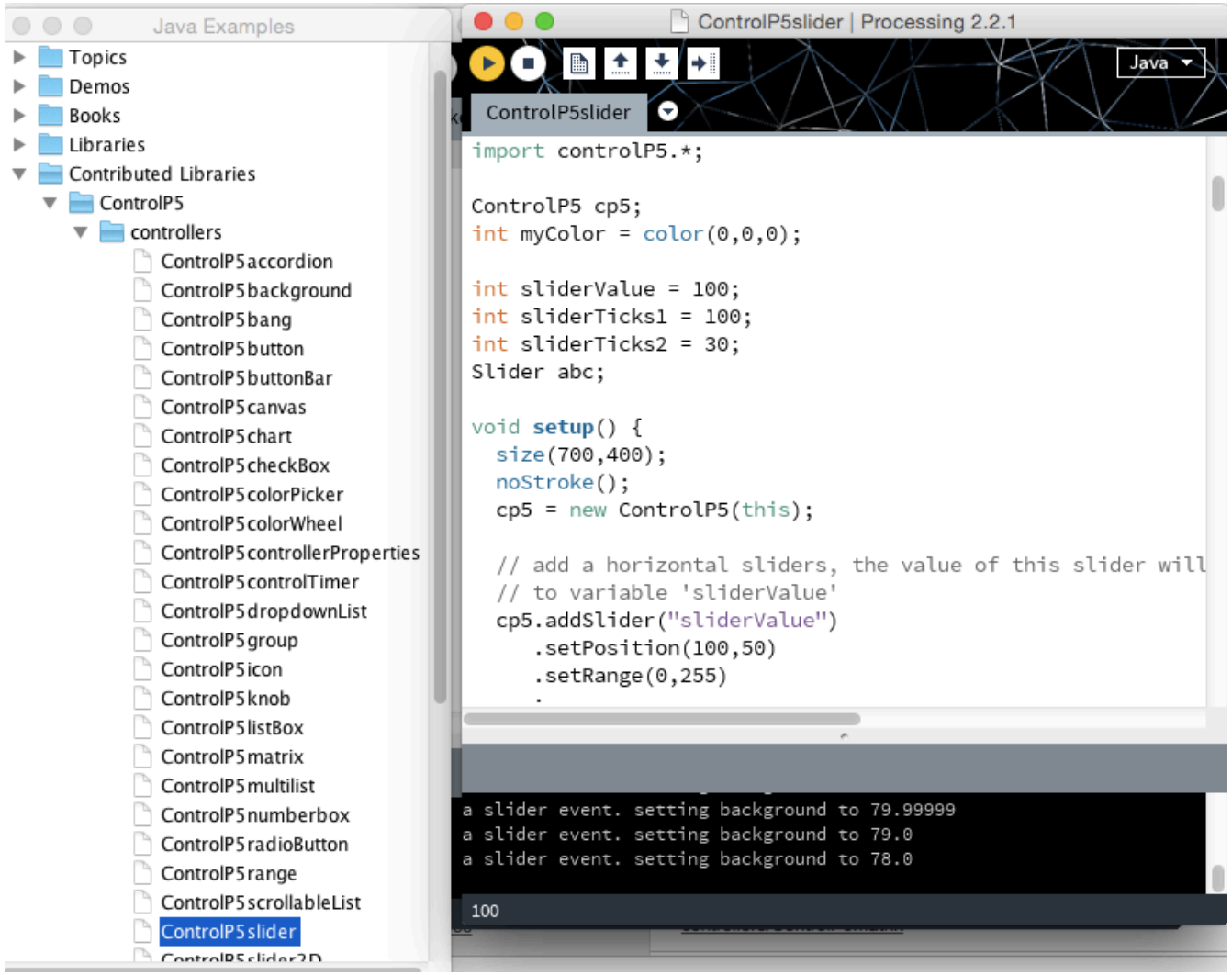
Les principaux widgets pré-définis dans ControlP5



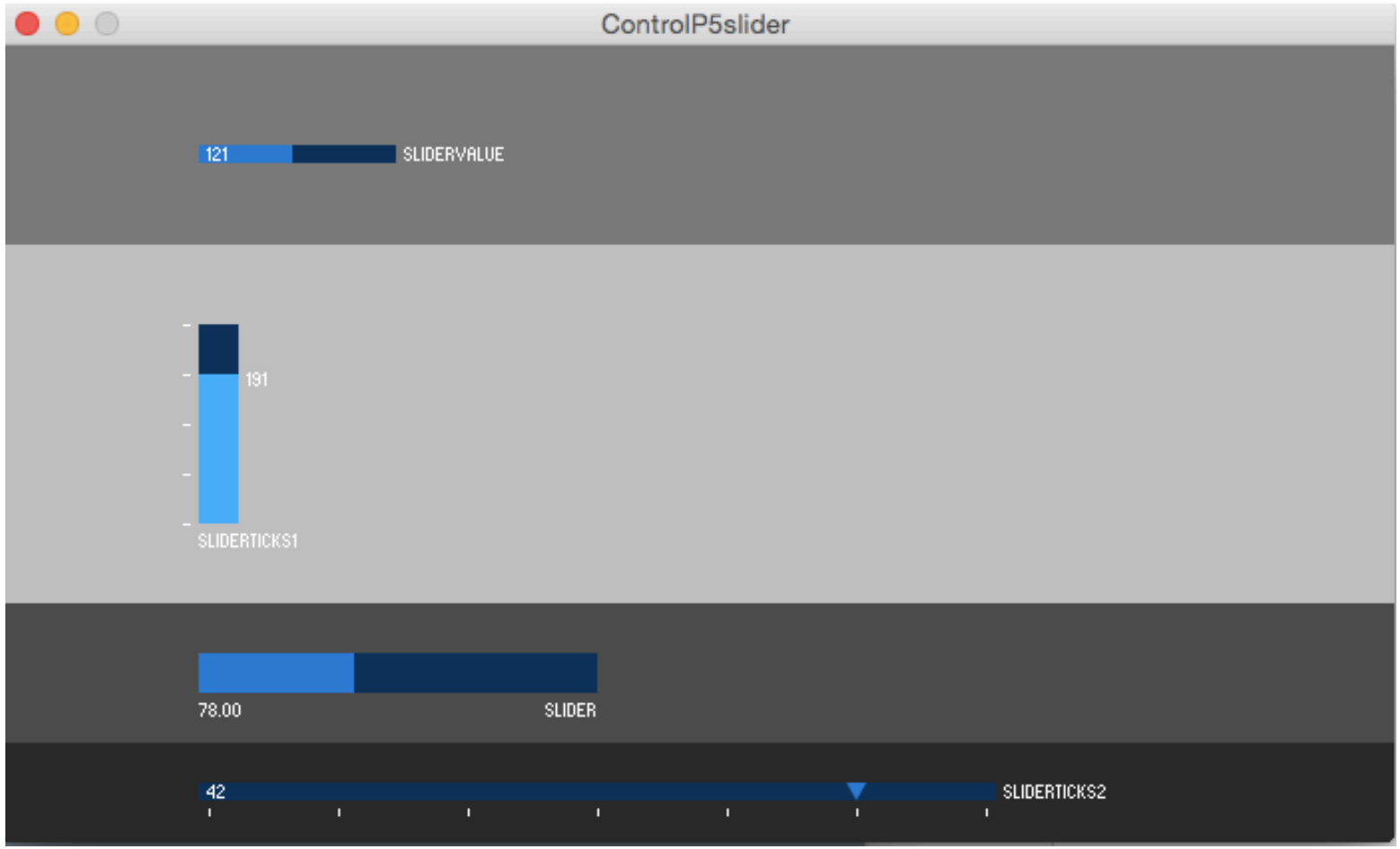
http://www.sojamo.de/libraries/controlP5/

Tester un exemple de la librairie (menu File->Examples)

Ici, le code ControlP5slider



A l’exécution :



Etudier le code !

Remarques :

- controlP5 est limitée mais suffisante pour notre cours

- Il existe d'autres librairies de GUI pour Processing.
En particulier, G4P permet de faire des interfaces multi-fenêtres et inclut un logiciel de dessin des interfaces pour le protypage rapide de celles-ci.

- Processing est incompatible avec la librairie standard AWT de Java. Pour Swing, en théorie ça marche.