Interaction avec l'utilisateur

Daniel Porumbel

nombreaux slides dus à
Pierre Cubaud

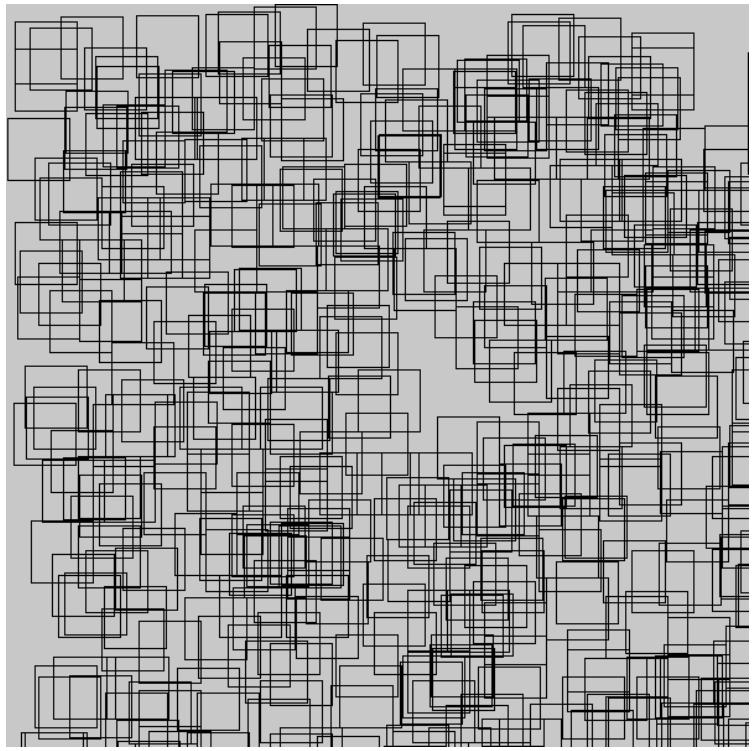
Processing en mode réactif

il faut renseigner deux fonctions prédéfinies :

```
void setup() {
  // ici du code pour l'initialisation du programme
}

void draw() {
 // là du code exécuté pour chaque trame (frame)
}
```

void signifie que la fonction ne retourne rien après son exécution

Un premier draw()



en hommage à Vera Molnar

modereactif1

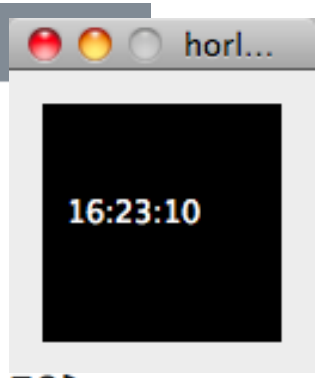
```
void setup() {
  size(600,600);
  noFill();stroke(0);
  background(200);
  frameRate(10);
}

void draw() {
  float x=random(0,width);
  float y=random(0,height);
  rect(x,y,50,50);
}
```

- jouer avec la valeur de frameRate()
- déplacer background() au début de draw()

Une horloge

horloge

```
void draw() {
  background(0);
  int s = second();
  int m = minute();
  int h = hour();
  text(str(h)+":"+str(m)+":"+nf(s,2),10,50);
}
```



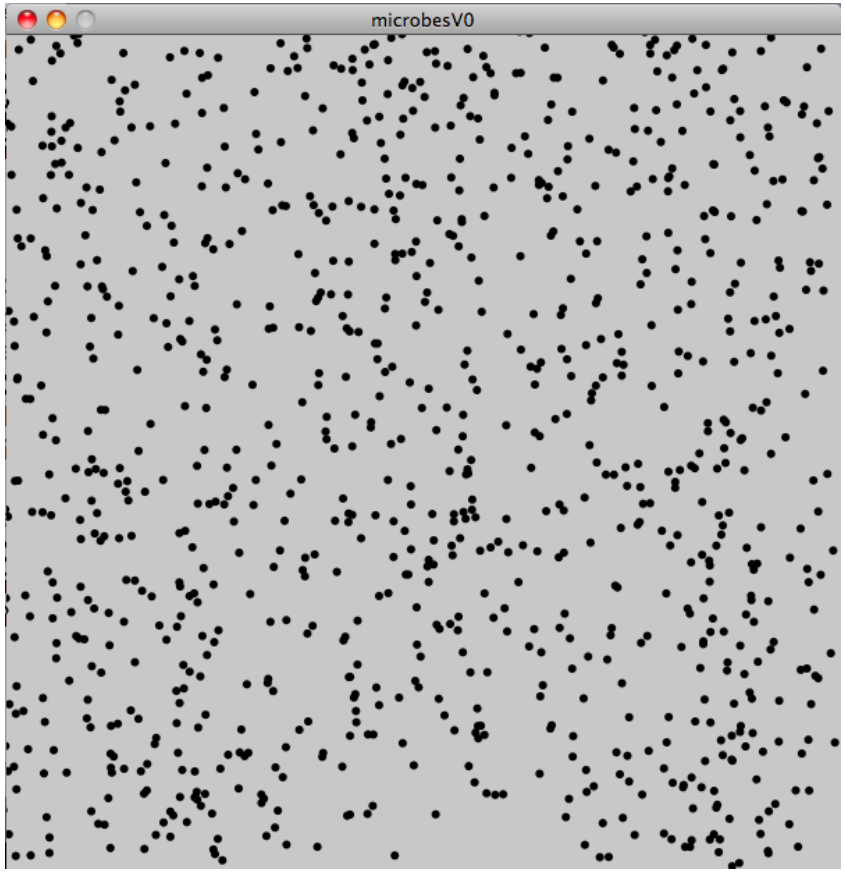
nf() = ???

horlogeMilli

```
void setup() {
  frameRate(10);
}

void draw() {
  background(0);
  text(millis(),10,50);
}
```

essayer plusieurs variantes

Un classique : le mouvement brownien



microbesV0

```
final int NBMICROBES = 1000;
float[] mX = new float[NBMICROBES];
float[] mY = new float[NBMICROBES];

void setup() {
 size(600,600);
 smooth();
 for (int i=0;i<NBMICROBES;i++) {
  mX[i] = random(0,width);
  mY[i] = random(0,height);
 }
}
```

```
void draw() {
 background(200);
 // dessins
 for (int i=0;i<NBMICROBES;i++) {
  fill(0);noStroke();
  ellipse(mX[i],mY[i],6,6);
 }
 // deplacements
 for (int i=0;i<NBMICROBES;i++) {
  mX[i] = (mX[i]+random(-1,+1))%width;
  mY[i] = (mY[i]+random(-1,+1))%height;
 }
}
```

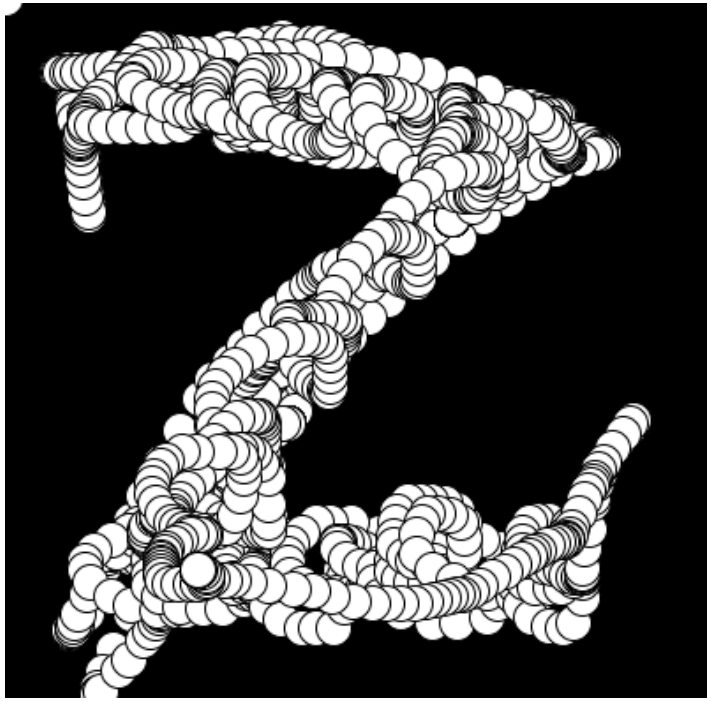
on contraint
les objets à rester
dans la zone de dessin

Interaction

C’est facile de récupérer la position de la souris :

- variable `mouseX`
- variable `mouseY`

Interaction !



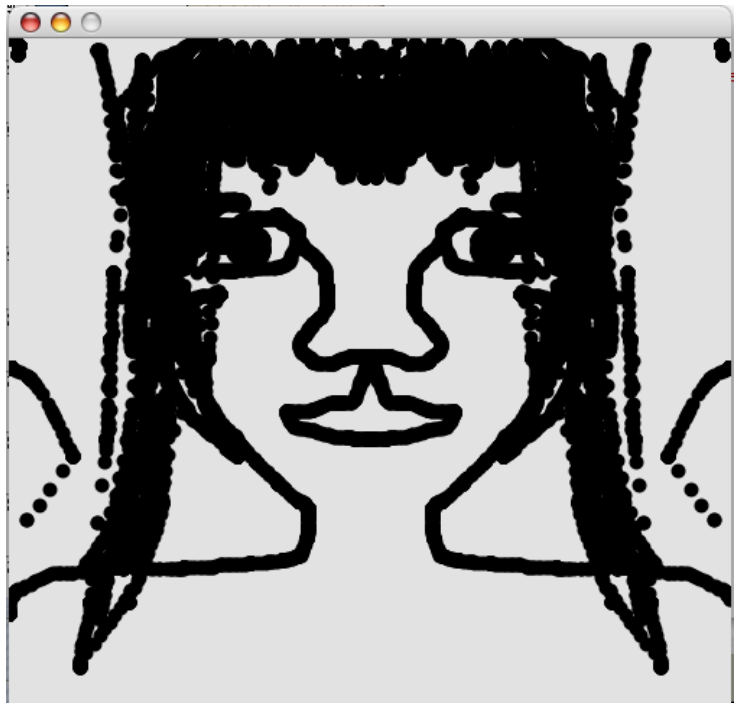
en hommage à Douglas Engelbart

souris

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

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

- ajouter d'autres tracés
- jouer avec la transparence

symétrie



sourisSYMETRIE

```
void setup(){
  size(600,600);
  smooth();
  background(200);
  fill(0);noStroke();
  noCursor();
}

void draw(){
  int x = mouseX;
  int y = mouseY;
  ellipse(x,y,20,20);
  ellipse(width-x,y,20,20);
}
```

justifier l'équation pour la symétrie sur l'axe x

Gestion d'évènements

Pour que le code réagisse aux actions de l'utilisateur, il faut renseigner les fonctions appropriées :

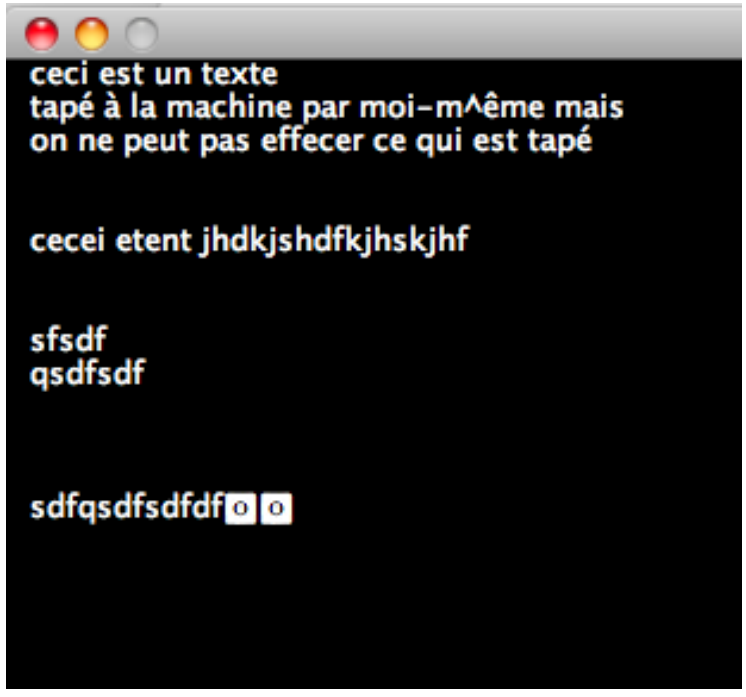
void mousePressed()
void mouseReleased()
void mouseMoved()
void mouseDragged()
+ utiliser les variables pré-définies mouseX mouseY

void keyPressed()
void keyReleased()
+ utiliser la variable pré-définie key

Autre exemple : une mauvaise machine à écrire

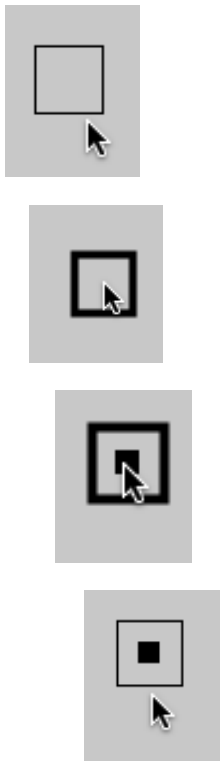
machineecrire

```
String letexte="";

void setup(){
  size(800,600);
  stroke(255);fill(255);
}

void draw(){
  background(0);
  text(letexte,10,10);
}

void keyPressed() {
 letexte += key;
}
```



pas de retour (feedback) utilisateur !

Dessine-moi un bouton (radio)



état 1 : non sélectionné, non désigné

entrée de zone : "roll over"

état 2 : non sélectionné, désigné

clic

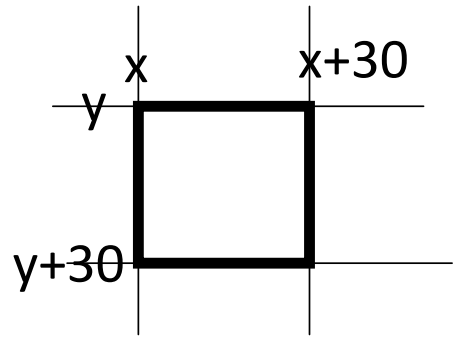
état 3 : sélectionné, désigné

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

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

Quelques fonctions

1. Écrire un fonction qui renvoie la somme de deux arguments

Quelques fonctions

1. Écrire un fonction qui renvoie la somme de deux arguments
2. Écrire une fonction qui renvoie le cube d'un argument

Quelques fonctions

1. Écrire un fonction qui renvoie la somme de deux arguments
2. Écrire une fonction qui renvoie le cube d'un argument
3. Écrire une fonction qui renvoie la valeur minimale d'un tableau

Quelques fonctions

1. Écrire un fonction qui renvoie la somme de deux arguments
2. Écrire une fonction qui renvoie le cube d'un argument
3. Écrire une fonction qui renvoie la valeur minimale d'un tableau
4. Écrire une fonction qui renvoie la somme des valeurs d'un tableau

Quelques fonctions

1. Écrire un fonction qui renvoie la somme de deux arguments
2. Écrire une fonction qui renvoie le cube d'un argument
3. Écrire une fonction qui renvoie la valeur minimale d'un tableau
4. Écrire une fonction qui renvoie la somme des valeurs d'un tableau
5. Écrire une fonction avec un seul argument qui renvoie :
   - `true` si l'argument est un nombre premier
   - `false` sinon

Quelques fonctions

1. Écrire un fonction qui renvoie la somme de deux arguments
2. Écrire une fonction qui renvoie le cube d'un argument
3. Écrire une fonction qui renvoie la valeur minimale d'un tableau
4. Écrire une fonction qui renvoie la somme des valeurs d'un tableau
5. Écrire une fonction avec un seul argument qui renvoie :
   - `true` si l'argument est un nombre premier
   - `false` sinon
6. Afficher tous les nombres premiers inférieurs à 100