le cnam

Médias interactifs numériques

interaction

Dessine-moi un bouton

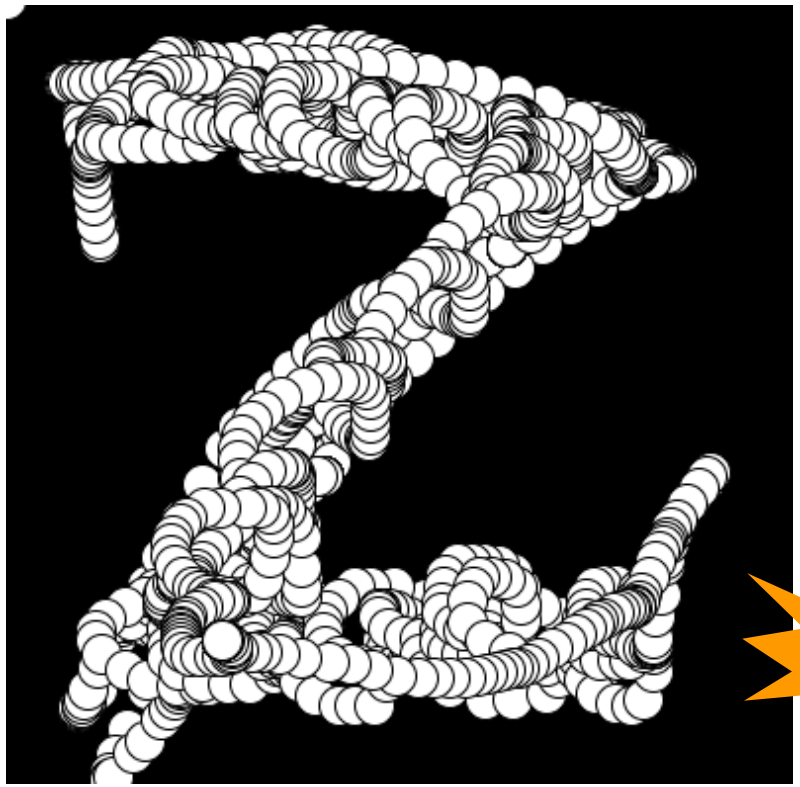
Pierre Cubaud

Département d'informatique

CNAM

Interaction !

le cnam



démo

en hommage à Douglas Engelbart

souris

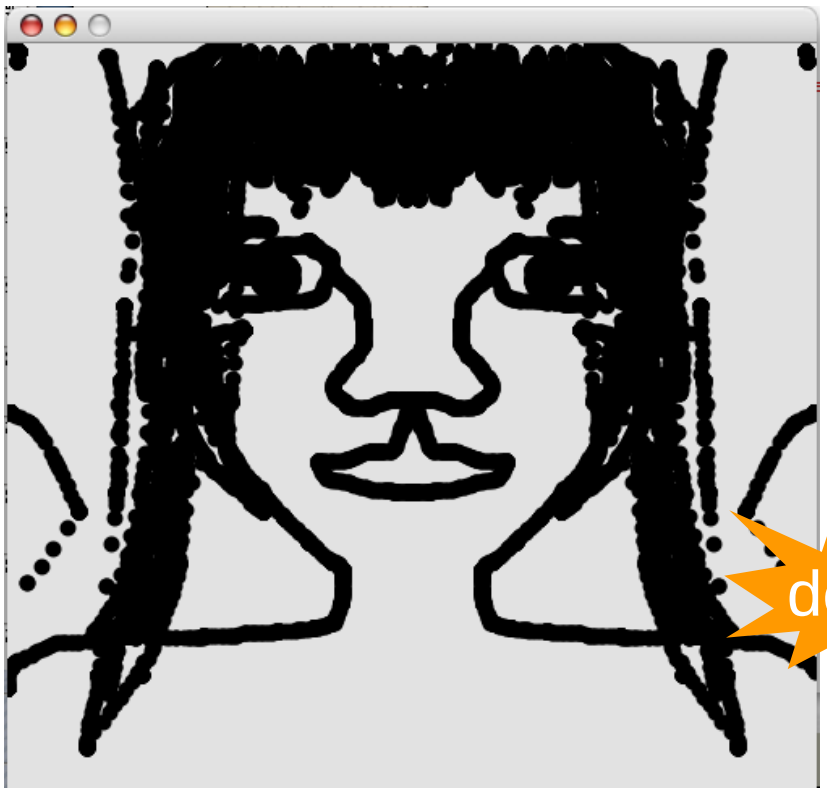
```
void setup(){
  size(400,400);
  smooth();
  background(0);
  noCursor();
}

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

- ajouter d'autres tracés
- jouer avec la transparence

le cnam

symétrie



démo

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'évenements

le cnam

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

mousePressed()
mouseReleased()
mouseMoved()
mouseDragged()

keyPressed()
keyReleased()

Dessine-moi un bouton (radio)

état 1 : non sélectionné, non désigné

entrée de zone : "roll over"

état 2 : non sélectionné, désigné

clic

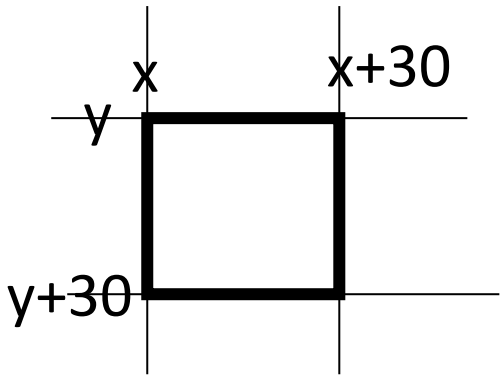
état 3 : sélectionné, désigné

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

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);
  }
}
```

le cnam

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

le cnam

BONUS : approche objet

Comment faire si on veut 2 (ou 100) boutons ???

- 4 paramètres par boutons : x, y, rollover, selected
- lourdeur de la boucle draw()
- test de detection de zone ??

⇒avantage décisif de la programmation "objets"

on va créer une **classe** Bouton qui regroupe les traitements de dessin et la gestion des événements et les paramètres propres à chaque bouton

monboutonCLASSE | Processing 1.5.1

STANDARD

monboutonCLASSE | Bouton

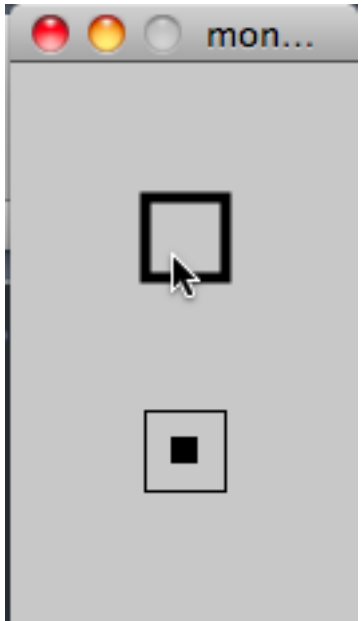
```
Button b1,b2;

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);
}
```



le cnam

Squelette du code de la classe :

le cnam

```
class Button {
  /// ici declaration des parametres x,y, selected, rollover

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

  void display() {
   /// ici code d'affichage
  }

  void rollover(int mx, int my) {
   /// ici code detection de rollover
  }

  void clic(int mx, int my) {
   //// ici code gestion du booleen selected
  }
}
```

constructeur pour les objets Button

les autres méthodes

code complet de la classe

le cnam

```
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;
  }

}
```

Conclusion sur l'exercice :

le cnam

- le passage aux classes se justifie si on a plusieurs "objets" de même comportement
- si ce nombre devient grand, il faut inventer en plus des classes "gestionnaires"
- profiter aussi de la notion d'héritage, qui permet de créer des objets au comportement proche (non traité dans ce cours)

- il reste à faire que ces boutons servent à quelque-chose (donc déclenchent des actions)

=> embryon d'une bibliothèque d'interface graphique