III.2 – Gestion du clavier par Processing

- **Interactions « classiques »** : souris et clavier
- Principales informations et fonctions associées au clavier :
  - Informations sur le « code » de la touche
    - `key` : nom de la touche, par exemple 'a' pour la première lettre de l'alphabet
    - `keyCode` : même chose, sous la forme d'un code ASCII (valeur 65 pour la lettre a)
    - Certaines touches « spéciales » ne sont repérables que par leur code ASCII :
      - LEFT : 37, UP : 38 , RIGHT : 39, DOWN : 40
      - SHIFT : 16
      - CONTROL : 17
      - ALT : 18
      - PAGE UP (page haute) : 33
      - PAGE DOWN (page basse) : 34
  - Information d'état d'une touche
    - `keyPressed` : vaut `true` si une des touches est enfoncée
  - Fonctions **événementielles**
    - `keyPressed()` : fonction invoquée automatiquement à l'instant où une touche est enfoncée
    - `keyReleased()` : invoquée lorsque la touche est relâchée
    - `keyTyped()` : invoquée lors d'un cycle d'appui suivi d'un relâchement
    - Attention au Focus :la fenêtre doit être au **premier plan** (« Windows focus ») pour que les événements d'appui des touches soient envoyés aux fonctions événementielles de Processing !

38

38

III.2 – Exemple 3 : image virtuelle par clavier (1/3)

```
/*****************************************************************************/
/*          Exemple 3 : Recuperation des informations "clavier"              */
/*****************************************************************************/
...
int X,Y;   // Coordonnees des points a tracer
int Delta; // Le "pas" d'incrementation
boolean flagDebug = true; // Booleen pour afficher ou non les infos de debug

void setup() {
  size(400,200);
  surface.setTitle("Exemple 3 - Le Clavier - E. Mesnard / ISIMA");
  colorMode(RGB, 255,255,255); // fixe format couleur R G B pour fill, stroke, etc...
  noFill(); // pas de remplissage
  stroke(rouge); // couleur pourtour RGB - noStroke() si pas de pourtour
  background(blanc); // couleur fond fenetre
  smooth(); // Activation de l'anti-aliasing (smooth)

  // Initialisation des variables
  X=0;
  Y=0;
  Delta=1;
} // Fin de Setup

void draw() {
  // Rien de particulier ! : tout est dans la gestion des evenements
}
```

39

39

III.2 – Exemple 3 : image virtuelle par clavier (2/3)

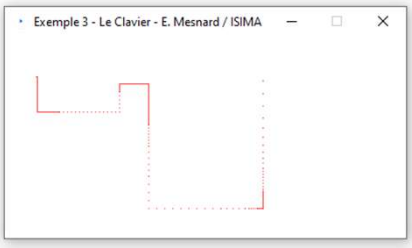
```
// Fonction de gestion des evenements du clavier

void keyPressed() {
  // Analyse des caracteres "classiques"
  switch (key) {
    case 'd' : flagDebug = !flagDebug; // Debug on/off
               break;
    case ' ' : background(blanc); // Appui sur la Barre d'espace
               break;             // pour effacer la fenetre

    // Appui sur les touches plus et moins ...
    case '+' : Delta = (Delta<15) ? (Delta+1) : 15; // ... pour incrementer le pas
               break;
    case '-' : Delta = (Delta>1) ? (Delta-1) : 1; // ... pour decrementer le pas
               break;
  }

  // Analyse des caracteres "etendus"
  switch (keyCode) {
    // Fleches : LEFT, RIGHT, UP, DOWN
    case LEFT    : X-=Delta;
                   X = (X<0) ? 0 : X;
                   break;
    case RIGHT   : X+=Delta;
                   X = (X>width) ? width : X;
                   break;
```

40

40

III.2 – Exemple 3 : image virtuelle par clavier (3/3)

```
    case UP      : Y-=Delta;
                   Y = (Y<0) ? 0 : Y;
                   break;
    case DOWN    : Y+=Delta;
                   Y = (Y>height) ? height : Y;
                   break;
    // Autres Touches speciales : CONTROL, ALT et SHIFT
  }
  if (flagDebug) { // information en fenetre de console
    println("Touche " + key + " avec le code " + keyCode);
  }
  // Trace d'un point a l'endroit indique
  point(X,Y);
}
```



41

41

III.3 – Gestion d'une souris par Processing

- Principales informations et fonctions de la souris :
  - Informations de position de la souris
    - `mouseX` et `mouseY` : position courante
    - `pmouseX` et `pmouseY` : position **précédente**
  - Information d'état des boutons de la souris (valeur booléenne)
    - `mousePressed` : vaut `true` si un des boutons est enfoncé
    - `mouseButton` : nom du bouton enfoncé : `LEFT`, `CENTER`, `RIGHT`
  - Fonctions événementielles
    - `mousePressed()` : fonction invoquée à l'instant où un bouton de la souris est appuyé
    - `mouseReleased()` : même chose, lorsque le bouton de la souris est relâché
    - `mouseClicked()` : invoquée après un cycle d'appui suivi d'un relâchement
    - `mouseMoved()` : invoquée chaque fois que la souris a bougé
    - `mouseDragged()` : même chose avec clic maintenu
    - `mouseWheel()` : invoquée lors d'une rotation de la molette centrale

42

42

III.3 – Exemple 4 : gestion de la souris (1/2)

```
/*****************************************************************************/
/*          Exemple 3 : Creation d'une image virtuelle via la souris         */
/*****************************************************************************/
...
// Declaration d'une variable globale
int totalMolette; // Gestion de la roulette centrale

// Fonction d'initialisation de l'application - executee une seule fois
void setup() {
  // Initialisation des parametres graphiques utilises
  size(640,480,P2D);  // Avec acceleration graphique 2D
  surface.setTitle("Exemple 4 - La souris - E. Mesnard / ISIMA");

  colorMode(RGB, 255,255,255); // fixe format couleur R G B pour fill, stroke, etc...
  noFill(); // pas de remplissage
  stroke(rouge); // couleur pourtour RGB - noStroke() si pas de pourtour
  background(blanc); // couleur fond fenetre
  smooth(); // Activation de l'anti-aliasing (smooth)

totalMolette = 0;
} // Fin de Setup
```

43

43

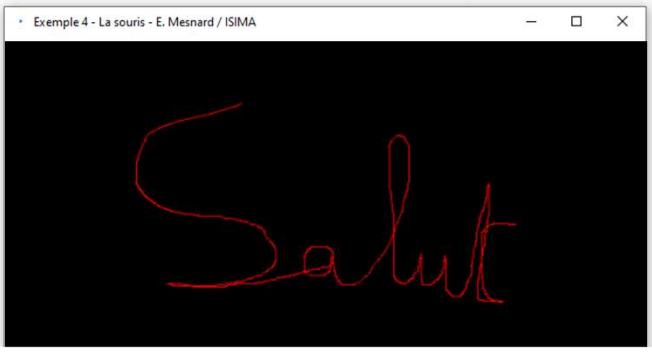
III.3 – Exemple 4 : gestion de la souris (2/2)

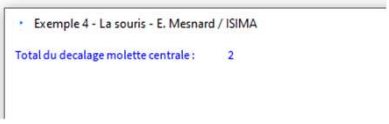
```
void draw() {
  if (mousePressed && (mouseButton == RIGHT)) { // Test clic droit de la souris...
    stroke(vert);    // ... mauvaise methode => fonctions evenementielles
    circle(mouseX, mouseY, 5); // pour tracer un rond vert
  }
}
// Fonctions de gestion des evenements de la souris
void mousePressed() { // Fonction invoquee lors de l'appui sur un des deux boutons
  background(noir);   // Equivaut a effacer la fenetre et mettre tout en noir
}
void mouseReleased() { // Fonction invoquee lors du relachement d'un bouton
  background(blanc);   // Equivaut a effacer la fenetre et mettre tout en blanc
}
void mouseDragged() { // Fonction invoquee tant que le bouton est maintenu appuye
  stroke(rouge);      // Trace d'une ligne rouge entre la nouvelle position de
  line(pmouseX, pmouseY, mouseX, mouseY); // la souris et l'ancienne
}
void mouseWheel(MouseEvent event) {
  totalMolette += event.getCount(); // Prise en compte de l'evenement
  println(totalMolette); // Affichage retour console pour debug
  background(blanc); // Effacement de la fenetre
  // Affichage des informations sur la fenetre de l'application
  text("Total du decalage molette centrale : ",10,20);
  text(str(totalMolette),222,20);
}
```

44

44

III.3 – Exemple 4 : exécution





45

45