IV – Interaction 2D via une WebCam

47

47

IV.1 – Principe de fonctionnement d'une caméra webCam

- Caractéristiques :
  - Nom de la caméra : « Logitech HD Webcam C310 »
  - Taille des images :
    - « Résolution » : plusieurs tailles possibles
    - Valeurs connues en interrogeant la caméra !
    - Mode VGA : 640x480 (standard d'une webcam)
  - Vitesse d'acquisition :
    - « FrameRate » : plusieurs valeurs possibles
    - Généralement : 30 images/seconde

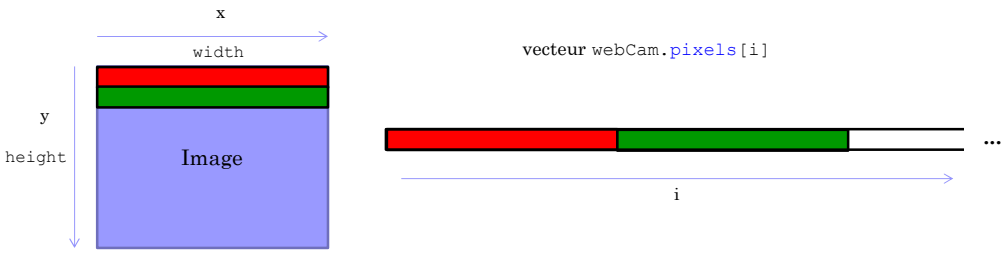
- Principe du traitement du flux vidéo :
  - Succession d'images ! (« Frame »)
  - Vérifier si une nouvelle frame est disponible pour récupérer l'image correspondante
  - Image = tableau de Pixels

48

48

IV.1 – Format d'image vidéo sur webCam

- Format d'une image vidéo :
  - « Tableau » de pixels, stocké sous la forme d'un **vecteur** de taille `width` x `height`



- Accès au contenu du vecteur (donc, des pixels de l'image) :
  - Encadrement par `loadPixels()` et `updatePixels()`

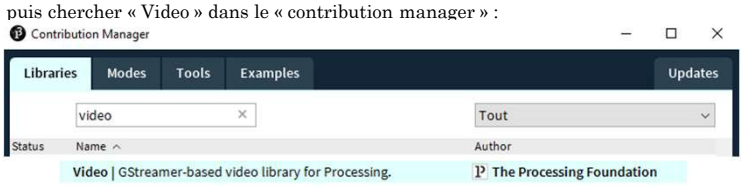
```
webCam.loadPixels();   // Chargement du tableau de pixels avant traitement
...
webCam.pixels[] =      // Analyse et/ou modification du vecteur (de l'image)
...
webCam.updatePixels(); // Mise a jour des pixels
```

49

49

IV.1 – Bibliothèque standard de gestion WebCam : « Video »

- Bibliothèque « video » basée sur « Gstreamer »
  - A ajouter explicitement : menu Sketch > Importer une librairie... > Ajouter une librairie
  - puis chercher « Video » dans le « contribution manager » :



- Intégration de cette bibliothèque dans un programme Processing :
  - Import de la bibliothèque, à faire **avant** `setup()` :

```
import processing.video.*;
```

  - Déclarer l'objet webCam, variable de type « Capture » :

```
Capture webCam;       // Declaration de la Capture par Camera
```

  - Déclarer une variable, chaîne de caractères, de liste des webcams disponibles :

```
String[] cameras;     // Liste textuelle des webCams disponibles sur l'ordi
```

50

50

IV.1 – Utilisation d'une WebCam de type « Capture »

- Dans `setup()` : Ouverture de la webCam, avec les bons paramètres
  - Recherche d'une webCam, par interrogation du système d'exploitation de la machine :
    ```
    String[] cameras = Capture.list();
    ```
  - Ouverture de la webCam, que si la recherche précédente a aboutie
    ```
    if (0 == cameras.length) {// pas de webCam... } else {
    webCam = new Capture(this, 640, 480, cameras[0], 30);
    ```
  - Mise en marche effective de la webCam, pour création d'images à la fréquence souhaitée
    ```
    webCam.start(); }
    ```
- Dans `draw()` : lecture d'une frame et traitement
  - Vérification au préalable de la présence effective d'une nouvelle « frame »
    ```
    if (webCam.available()) {
    ```
  - Récupération du vecteur de pixels :
    ```
    webCam.read(); }
    ```
  - Traitement des images
    ```
    webCam.loadPixels();
    ... webCam.pixels[] ...
    webCam.updatePixels();
    ```
  - Restitution (éventuelle) de l'image sur la fenêtre Processing
    ```
    image(webCam, 0, 0);
    ```
- *Eventuellement*, dans `exit()` (détourné), fermeture « propre »: `webCam.stop();`

51

51

IV.1 – Exemple 5 : Webcam Capture

```
/*************************************************************************/
/*          Exemple 5 : Gestion d'une Webcam via le mode "Capture"       */
/*************************************************************************/
import processing.video.*; // Bibliotheque de controle camera
Capture webCam;      // Declaration de la Capture par Camera
String[] cameras;    // Liste textuelle des webCams disponibles

void setup() {
  size(640,480);
  cameras = Capture.list();   // Recherche des webCams disponibles
  if (0 == cameras.length) {
    println("Pas de Webcam sur cet ordinateur !"); exit();
  } else {
    webCam = new Capture(this, 640, 480); // Webcam par defaut
    webCam.start(); // Mise en marche de la webCam
} // Fin de Setup

void draw() {
  if (webCam.available() == true) { // Verification de presence d'une nouvelle frame
    webCam.read();                  // Lecture du flux sur la camera... lecture d'une frame
    image(webCam, 0, 0);            // Restitution de l'image
 }
} // Fin de draw

// ATTENTION : voir les differentes variantes dans le fichier Exemple_5
```

52

52

IV.2 – Seconde possibilité : Bibliothèque Java « Sarxos »

- Bibliothèque externe à Processing
  - A ajouter **manuellement** car non disponible dans le « contribution manager »
  - Dézipper « Webcam_Sarxos » directement dans C:\Users\xxx\Documents\Processing\libraries
- Intégration de cette bibliothèque dans un programme Processing :
  - Import de la bibliothèque, à faire **avant** `setup()` :

```
import com.github.sarxos.webcam.*;    // Bibliotheque SARXOS de gestion webcam
import java.awt.image.BufferedImage; // Biblio pour conversion BufferedImage
import java.awt.Dimension;           // en PImage, a la bonne taille
import java.util.List;    // Pour recuperer la liste des webcams disponibles
```

- Dans `setup()` : Ouverture de la webCam, avec les bons paramètres
  - Recherche d'une webCam, par interrogation du système d'exploitation de la machine :

```
List<Webcam> cameras = Webcam.getWebcams();
```

  - Ouverture de la webCam, que si la recherche précédente a aboutie

```
if (cameras.isEmpty()) {// pas de webCam... } else {
webCam = Webcam.getDefault(); // Recuperation de la camera par defaut
```

  - Configuration de la webCam : choix de la résolution

```
webCam.setViewSize(new Dimension(640, 480)); // Choix de la resolution
PImgWebCam = createImage(640, 480, ARGB); // Creation de l'image de reception
```

  - Mise en marche effective de la webCam

```
webCam.open(); }
```

53

53

IV.2 – Seconde possibilité : Bibliothèque Java « Sarxos »

- Dans `draw()` : lecture d'une frame et traitement
  - Vérification au préalable de la présence effective d'une nouvelle « frame »

```
if (webCam.isImageNew() && webCam.isOpen()) {
```

  - Récupération du vecteur de pixels, au format « BufferedImage » de Java :

```
BImgWebCam = webCam.getImage(); }
```

  - **Conversion** en PImage, format standard de Processing

```
BImgWebCam.getRGB(0, 0, 640, 480, PImgWebCam.pixels, 0, 640);
PImgWebCam.updatePixels();
```

  - Analyse et/ou modification du vecteur, donc, de l'image

```
PImgWebCam.loadPixels();
... PImgWebCam.pixels[] ...
PImgWebCam.updatePixels();
```

  - Restitution (éventuelle) de l'image sur la fenêtre Processing

```
image(PImgWebCam, 0, 0);
```

- *Eventuellement*, dans `exit()` (détourné), fermeture « propre »: `webCam.close();`

54

54

IV.2 – Exemple 6 : Webcam Sarxos (1/2)

```
/*************************************************************************/
/*          Exemple 6 : Gestion d'une Webcam via le driver "Sarxos"      */
/*************************************************************************/
import com.github.sarxos.webcam.*;   // Bibliotheque SARXOS de gestion webcam
import java.awt.image.BufferedImage; // Biblio pour conversion BufferedImage
import java.awt.Dimension;           // en PImage, a la bonne taille
import java.util.List;    // Pour recuperer la liste des webcams disponibles

Webcam webCam;            // Declaration de la webCam
List<Webcam> cameras ;    // Liste des webCams disponibles
int nbCam;                // Nombre de cameras dans la liste
BufferedImage BImgWebCam; // Image fournie par la webCam
PImage PImgWebCam;        // Meme chose, convertie au format Processing PImage

void setup() {
  size(640,480);
  cameras = Webcam.getWebcams(); // Recherche des webCams disponibles
  if (cameras.isEmpty()) {
    println("Pas de Webcam sur cet ordinateur !"); exit();
  } else {
    webCam = Webcam.getDefault();// Webcam par defaut
    webCam.setViewSize(new Dimension(640, 480)); // Choix de la resolution pour cette webCam
    webCam.open(); // Mise en marche de la webCam
} // Fin de Setup
```

55

55

IV.2 – Exemple 6 : Webcam Sarxos (2)

```
/*************************************************************************/
/*          Exemple 6 : Gestion d'une Webcam via le driver "Sarxos"      */
/*************************************************************************/

void draw() {
   // Verification de presence d'une nouvelle frame
   if (webCam.isImageNew() && webCam.isOpen()) {

    // Lecture du flux sur la camera... lecture d'une frame
    BImgWebCam = webCam.getImage();

    // Conversion de BufferedImage au format PImage
    PImgWebCam.loadPixels();
    BImgWebCam.getRGB(0, 0, 640, 480, PImgWebCam.pixels, 0, 640);
    PImgWebCam.updatePixels();

    // Restitution de l'image
    image(PImgWebCam, 0, 0);
 }
} // Fin de draw

// ATTENTION : voir les differentes variantes dans le fichier Exemple_6
```

56

56

IV.3 – Interaction 2D basique : détecteur « simpliste »

- Interaction 2D
  - Utilisation de la webcam pour détecter des « mouvements » de la part de l'utilisateur

  - La caméra est utilisée comme « Capteur »

- Nécessité de faire du « Traitement d'images » !
  - Acquérir l'image
  - Simplifier l'image : filtrer, changer la résolution, changer les couleurs (N&B), …
  - Traiter l'image : rechercher des POI et ROI (« Point/Region Of Interest »)
  - Modifier l'image en conséquence

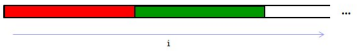
57

57

IV.3 – Traitement de l'image

- Traitement d'image = analyse des pixels, donc, de leurs couleurs
- Deux modes de parcours : vecteur (1D) ou matrice (2D)

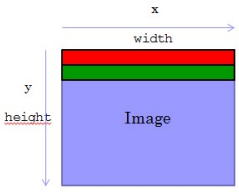
  - Parcours sur le vecteur (index i) :




```
int i;            // index
color currColor; // Couleur du pixel courant
webCam.loadPixels();
for (i = 0; i < (webCam.height*webCam.width); i++) {
  currColor = webCam.pixels[i]; // Recuperation d'un pixel...
  // Traitements sur les pixels...
}
webCam.updatePixels();
```

  - Parcours sur la matrice (index x et y) :

```
int i, x, y, yPos;
webCam.loadPixels();
for (y = 0; y < webCam.height; y++) { // lignes y
  yPos = y * webCam.width; // adresse debut de ligne
  for (x = 0; x < webCam.width; x++) { // colonnes x
    i = yPos + x; // Index = numero ligne + decalage
    currColor = webCam.pixels[i]; // Lecture pixel
    // Traitements sur les pixels...
  }
}
webCam.updatePixels();
```



58

58

IV.3 – Exemple 7 : recherche d'un point vert sur webCam

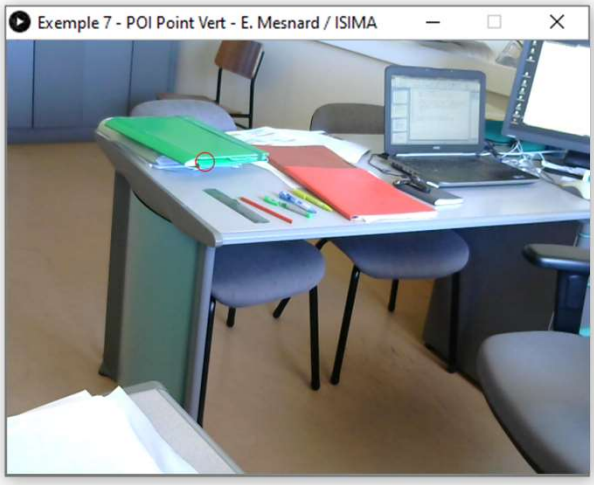
```
    // Recherche du point le plus proche de la couleur de reference
    poidsPOI = 360; // Valeur la plus grande possible
    xPOI = 0; yPOI = 0;   // Par defaut, le POI est en (0,0)
    // Analyse de l'image
    webCam.loadPixels();
    for (yy = 0; yy < heightCapture; yy++) { // abscisse yy
      yPos = yy * widthCapture;
      for (xx = 0; xx < widthCapture; xx++) { // ordonnees xx
        i = xx + yPos;
        currColor = webCam.pixels[i]; // recuperation couleur
        teinte = hue(currColor);      // et de la teinte

        // Calcul de l'ecart de teinte par rapport au vert pur
        poids = abs(120-teinte); // car vert pur = 120 degre
        if (poids < poidsPOI) {  // Le POI est le point qui a le moins de difference...
          poidsPOI = poids; // Mise a jour des informations et coordonnees
          xPOI = xx;
          yPOI = yy;
        }
      }
    } // A la sortie de cette boucle, le POI est (xPOI,yPOI)
    webCam.updatePixels();
    image(webCam, 0, 0); // Restitution de l'image captee sur la webCam
    ellipse(xPOI,yPOI,20,20); // Trace d'un cercle rouge autour du POI
  }
}
```

59

59

IV.3 – Exemple 7 : Résultat d'exécution !



60

60