



## V – Interactions 2D et 2D ½ via une Kinect

70

### V.1 – Présentation de la Kinect

#### ■ Dispositifs de la Microsoft Kinect (V1 ou 360) :

- microphones (au pluriel !)
- caméra RGB
  - flux possible : 640x480 à 30 images/seconde
- caméra InfraRouge
  - filme la grille infrarouge émise par le laser
  - flux : 640x480 à 30 images/seconde
  - Utilisé comme capteur de profondeur 3D
  - flux : 320x240 à 30 « images »/seconde

#### ■ Utilisation sous Processing

- Bibliothèque de fonctions J4K/UFDW :
  - « Java For Kinect / University of Florida – The Digital Worlds institute »



71

## V.2 – Flux de l'image couleur : ColorFrame

- Flux produit par la caméra RGB : « ColorFrame »
- Récupération en tableau des couleurs : « colorMap »
  - 4 octets (« byte ») séparés pour chaque pixel : B, G, R et A

B
G
R
A
B
G
R
A
...

- Conversion du tableau en une image en couleur : « colorImage »

R
G
B
R
G
B
R
G
B
...

- Algorithme de conversion :

```

PImage colorImage; // Format d'image classique sous Processing
byte[] colorMap = kinect.getColorFrame(); // Recuperation du flux de couleur
int j = 0;
for (i = 0; i < colorMap.length; i+=4) {
    int BB = (int) (colorMap[i]&0x0000FF); // Extraction des 3 composantes
    int GG = (int) (colorMap[i+1]&0x0000FF); // en ne conservant que l'octet
    int RR = (int) (colorMap[i+2]&0x0000FF); // de poids faible
    colorImage.pixels[j] = color(RR, GG, BB);
    j++;
}

```

- Résultat alors identique à celui d'une caméra de type « WebCam »

72

## V.2 – Exemple 9 : Kinect en mode vidéo RGB

```

/*****
/* MESNARD Emmanuel                               ISIMA */
/*                                                    */
/*      Exemple 9 : Utilisation de la Kinect V1 en mode RGB      */
/*                                                    */
/* Exemple_9_Kinect_RGB.pde                               Processing 3.5.4 */
*****/
// Declaration des variables globales
PKinect kinect; // Declaration de la Kinect
byte[] colorMap; // Carte des valeurs des couleurs = Flux "COLOR"
PImage colorImage; // Image RGB reconstruite equivalente
...
void setup() {
    ...
    kinect = new PKinect(this); // Initialisation Objet Kinect
    if (kinect.start(PKinect.COLOR) == false) { // Ouverture du flux "COLOR"
        println("Pas de kinect connectee !");      exit();    return;
    } else if (kinect.isInitialized()) {
        println("Kinect initialisee avec : ");
        colorW = kinect.getColorWidth();
        colorH = kinect.getColorHeight();
        println(" * Largeur image couleur : " + colorW);
        println(" * Hauteur image couleur : " + colorH);
    } else {
        println("Probleme d'initialisation de la kinect"); exit(); return;
    }
}

```

73

## V.2 – Exemple 9 : Kinect en mode vidéo RGB

```
void draw() {
    // Recuperation d'eventuelles donnees sur la kinect...
    colorMap = kinect.getColorFrame();

    // Traitement du Flux "COLOR"
    if (colorMap!=null) { // Des donnees sont disponibles
        // Conversion du tableau en une image en couleur
        colorImage.loadPixels(); // Lecture du tableau de pixels

        j = 0;

        for (i = 0; i < colorMap.length; i+=4) {
            int BB = (int) (colorMap[i]&0x0000FF); // Extraction des 3 composantes
            int GG = (int) (colorMap[i+1]&0x0000FF); // en ne conservant que l'octet
            int RR = (int) (colorMap[i+2]&0x0000FF); // de poids faible
            colorImage.pixels[j] = color(RR, GG, BB);
            j++;
        }

        colorImage.updatePixels(); // Forçage mise a jour du tableau de pixels (obligatoire)
    }
    // Affichage de l'image RGB
    image(colorImage,0,0);
}
```

74

## V.3 – Flux infrarouge : InfraredFrame et DepthFrame

- Image infrarouge :
  - Déduite du flux « InfraredFrame »
  - même principe que l'image couleur...
  - ... mais en niveaux de gris (R=G=B= gris)
  - peu intéressant !
- Carte de profondeur :
  - DepthMap déduite du flux « DepthFrame »
  - Calculé par la Kinect à partir des informations d'infrarouge
  - Profondeur Z :
    - 11 bits : de 0 à 2047
    - Coordonnées DepthMap = (i,j,Z)
    - image possible (i,j)... en niveaux de gris (couleur Z)
    - blanc = proche, noir = loin
- 2D et demie !



75

### V.3 – Flux des profondeurs : DepthFrame

- Flux produit par le capteur de profondeur : « DepthFrame »

- Récupération en tableau des distances : « depthMap »

- 1 entier (« short », car sur 11 bits) pour chaque pixel :

d d d d d d d d ...

- Conversion du tableau en une image (grise) : « depthImage »

- Gris, donc R=G=B=Z

Z Z Z Z Z Z Z Z ...

- Algorithme de conversion :

```
PImage depthImage; // Format d'image classique sous Processing
short[] depthMap = kinect.getDepthFrame(); // Recuperation du flux de profondeur
for (i = 0; i < depthMap.length; i++) {
    if (depthMap[i] == 0) { // Hors champ, traitement particulier
        depthImage.pixels[i] = 0; // en Noir !
    } else {
        // Valeur proportionnelle, limitee a des valeurs variant de 0 a 255
        // A noter que mini et maxi sont des valeurs calculees au prealable
        ValZ = (int) map((float)depthMap[i], mini, maxi, 255, 0);
        depthImage.pixels[i] = color(ValZ, ValZ, ValZ); // En gris
    }
}
```

76

### V.3 – Exemple 10 : Kinect en mode Depth

```
/* *****
/* MESNARD Emmanuel ISIMA */
/* *****
/* Exemple 10 : Utilisation de la Kinect V1 en mode Depth */
/* *****
/* Exemple_10_Kinect_Depth.pde Processing 3.5.4 */
/* *****
// Declaration des variables globales
PKinect kinect; // Declaration de la Kinect
short[] depthMap; // Carte des profondeurs = flux "DEPTH"
PImage depthImage; // Image equivalente aux profondeurs, en niveaux de gris

...
void setup() {
    ...
    kinect = new PKinect(this); // Initialisation Objet Kinect
    if (kinect.start(PKinect.DEPTH) == false) { // Ouverture du flux "DEPTH"
        println("Pas de kinect connectee !"); exit(); return;
    }
    ...
    void draw() {
        // Recuperation d'eventuelles donnees sur la kinect...
        depthMap = kinect.getDepthFrame();
        // Traitement du Flux "DEPTH"
        ...
    }
}
```

77