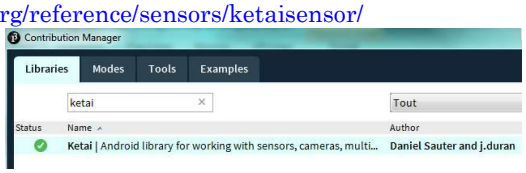
VIII.5 – Gestion des capteurs

- Utilisation de Ketai pour les capteurs (« Ketai » signifie « mobile » en japonais)

http://ketai.org/reference/sensors/ketaisensor/



- accéléromètre, gyroscope, magnétomètre, lumière, pression, température, orientation...
- Import de la bibliotheque :

```
import ketai.sensors.*; // Import de la bibliotheque des capteurs
```

- Declaration du gestionnaires du ou des capteurs :

```
KetaiSensor sensor;
```

- Mise en marche du gestionnaire sur les capteurs, dans le `setup` :

```
sensor = new KetaiSensor (this);
sensor.start();
```

- Lecture des valeurs du capteur, dans la fonction callback associée :

```
void onAccelerometerEvent(float x, float y, float z) {
  accelerometerX = x;
  accelerometerY = y;
  accelerometerZ = z;
}
```

X: -6.73076
Y: 4.91701
Z: 5.17262

http://ketai.org/examples/accelerometer/

135

VIII.6 – Gestion du « Touch » par Ketai

- Utilisation de Ketai pour le « Touch gesture »
  - Import de la bibliotheque :

```
import ketai.ui.*; // Import de la bibliotheque des touch gesture
```

  - Declaration du gestionnaires de gesture :

```
KetaiGesture gesture;
```

  - Mise en marche du gestionnaire dans le `setup` :

```
gesture = new KetaiGesture(this);
```

  - Fonctions événementielles, callback associées de « Touch gesture »

```
onTap(float x, float y) // x et y sont les positions du debut de gesture
onDoubleTap(float x, float y)
onFlick(float x, float y, float px, float py, float v)
// px et py = fin du mouvement ; v est la vitesse en pixels par seconde
onScroll(int x, int y)
onLongPress(float x, float y)
onPinch(float x, float y, float d)  // distance d'ecartement des doigts
onRotate(float x, float y, float a) // angle du mouvement
```

Tap | Double Tap | Flick | Scroll | Long Press | Pinch | Rotate

136

VIII.7 – Gestion de la caméra par Ketai

- Obtention du flux vidéo... toujours par Ketai !
  - Caméra : ce n'est qu'un capteur, mais qu'il faut autoriser à utiliser...
    - Attention : Android > Sketch Permissions > Camera
  - Eventuellement, plusieurs caméras disponibles :
    - `getNumberOfCameras()`, `getCameraID()` et `setCameraID()`
  - Ouverture de la caméra (par défaut), avec les bons paramètres :

```
import ketai.camera.*;  // Importation de la bibliotheque de controle camera Ketai
cam = new KetaiCamera(this, 640, 480, 30); // ou autre : 1280 x 720, en 24 fps
```

    - Mise en marche effective de la caméra

```
void mousePressed() {
  if (mouseX < 100 && mouseY < 100) { // Bouton On / Off en haut a gauche
    if (cam.isStarted()) cam.stop();
    else cam.start();
  }
}
```

  - Lecture d'une image sur le flux vidéo :
    - Si présence effective d'une nouvelle « frame » ...

```
void onCameraPreviewEvent() { // Fonction appelee a chaque nouvelle frame
  cam.read(); // Recuperation du vecteur de pixels
  refreshCam = true; // Gestion d'un booleen pour informer d'une nouvelle frame
```

137

VIII.7 – Gestion de la caméra par Ketai

- Traitement des images :
  - Analyse et/ou modification du vecteur, donc, de l'image

```
... cam.loadPixels();   // Encadrement load/update obligatoire !
...   cam.pixels[] ...  // Utilisation / modification du tableau de pixels
... cam.updatePixels();
...
```

- Restitution (éventuelle) de l'image

```
if (cam != null && cam.isStarted() && refreshCam) { // Verification
  image(cam, 0, 0, width, height);
  refreshCam = false;
```

138

VIII.8 – Exemple 13 : lecture caméra par Ketai

https://www.mobileprocessing.org/cameras.html

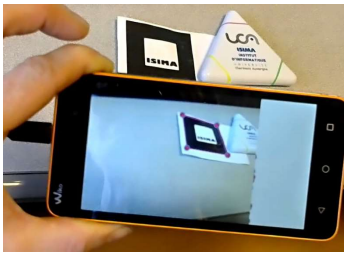
http://ketai.org/examples/camerageettingstarted/

```
import ketai.camera.*;  // Importation de la bibliotheque de controle camera
KetaiCamera cam;
float resolutionX = 30; float resolutionY = 30;
Boolean refreshCam = false;
void setup() {
  fullScreen(P2D); orientation(LANDSCAPE);
  imageMode(CENTER);  // Mode centre pour garantir que l'image sera visible
  cam = new KetaiCamera(this, 1280, 720, 24); // Gerer eventuellement cam == null
}
void onCameraPreviewEvent(){ // Fonction appelee a chaque nouvelle frame
  cam.read(); // Recuperation effective de l'image
  refreshCam = true; // Le traitement principal se fera dans la boucle draw
}
void draw() {
  if (cam.isStarted()) {
    if (refreshCam) { // Traitement principal car nouvelle frame disponible
       refreshCam = false;
       image(cam, width/2, height/2);  // Placement de l'image au centre
       for(int y=0; y < height; y += resolutionY) {
        for(int x=0; x < width; x += resolutionX) {
           color c = cam.get(x,y); // Lecture couleur du point
           fill(c); noStroke(); ellipse(x,y,resolutionX, resolutionY); // Un cercle a la place
  }}} else {
    cam.start();
  }
}
```

139

TD 10 Réalité Augmentée sous Android

- Travail demandé :
  - Faire une application Android, qui ouvre le flux vidéo (par Ketai) pour faire de la Réalité Enrichie / Réalité Augmentée
    (attention : pas de NFT sous Android...)
- Intégrer ensuite diverses options :
  - Afficher le clavier virtuel
  - Afficher l'état des capteurs
  - Réagir au touch
  - ...



A noter : `.setARClipping` sur Nyar4psg pour éviter les troncatures d'image

140