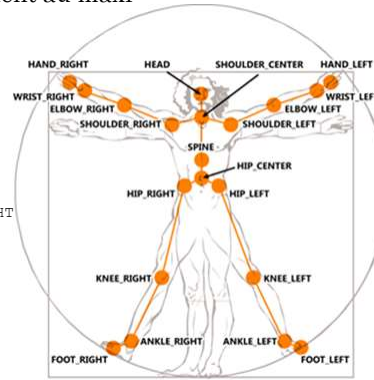


## IV.5 – Flux des squelettes : « Skeletons »

- Objectif de ce flux
  - NUI : « Natural User Interface »
- Principe de repérage Kinect
  - 6 personnes (humains !) repérées simultanément au maxi
  - Reconnaissance automatique du squelette ...
    - uniquement sur les 2 premiers humains détectés
  - 20 articulations par squelette (« joints »)
    - Modèle « Skeleton » dans J4K

HEAD, NECK  
 SPINE\_BASE, SPINE\_MID  
 SHOULDER\_LEFT, ELBOW\_LEFT, WRIST\_LEFT, HAND\_LEFT  
 SHOULDER\_RIGHT, ELBOW\_RIGHT, WRIST\_RIGHT, HAND\_RIGHT  
 HIP\_LEFT, KNEE\_LEFT, ANKLE\_LEFT, FOOT\_LEFT  
 HIP\_RIGHT, KNEE\_RIGHT, ANKLE\_RIGHT, FOOT\_RIGHT



69

## IV.5 – Flux des squelettes : « Skeletons »

- Fonctions de « Flags » (drapeaux) utiles :
  - `isTracked()` :  
humain traqué, donc visible (en partie)
  - `isJointTracked()` :  
articulation visible
  - `isJointTrackedOrInferred()` :  
articulations visibles  
+ articulations non visibles, mais déduites
- Fonctions de récupération des positions
  - `get2DJoint()` :  
informations de position, en coordonnées image écran (i,j)
  - `get3DJoint()` :  
informations de position, en coordonnées réelles (x,y,z)
  - `get3DJointX()`, `get3DJointY()`, `get3DJointZ()` :  
idem, en version séparées

70

### IV.5 – Exemple 7 : Kinect en mode Squelette

```

/*****
/*  MESNARD Emmanuel                                ISIMA      */
/*                                          */
/*      Exemple 8: Exploitation du flux squelette de la Kinect    */
/*                                          */
/* Exemple_7_Kinect_Skeleton_2D.pde                                Processing 3.0 */
/*****/
...
// Declaration des variables globales
PKinect kinect;    // Declaration de la Kinect
Skeleton[] s;      // Tableau des Squelettes des personnes detectees
...
void setup() {
    ...
    kinect = new PKinect(this); // Initialisation Objet Kinect
    if (kinect.start(PKinect.SKELETON) == false) { // Ouverture du flux "SKELETON"
        ...
    }

    void draw() {
        s = kinect.getSkeletons(); // Recuperation d'eventuelles donnees sur la kinect...
    }
}

```

71

### IV.5 – Exemple 7 : Kinect en mode Squelette

```

...
// Traitement du Flux "Skeletons"
for (i=0; i<sMax; i++) {
    if (s[i]!=null) { // Des donnees sont disponibles
        if (s[i].isTracked()==true) { // Cet humain est actuellement visible
            traceSquelette(i); // Dessin effectif du squelette
        }
    }
}

// Trace du squelette du personnage
void traceSquelette(int userId) {
    // Tete
    dessinMembre(userId, Skeleton.HEAD, Skeleton.NECK);

    // Bras Gauche
    dessinMembre(userId, Skeleton.NECK, Skeleton.SHOULDER_LEFT);
    dessinMembre(userId, Skeleton.SHOULDER_LEFT, Skeleton.ELBOW_LEFT);
    dessinMembre(userId, Skeleton.ELBOW_LEFT, Skeleton.WRIST_LEFT);
    dessinMembre(userId, Skeleton.WRIST_LEFT, Skeleton.HAND_LEFT);

    ...
}

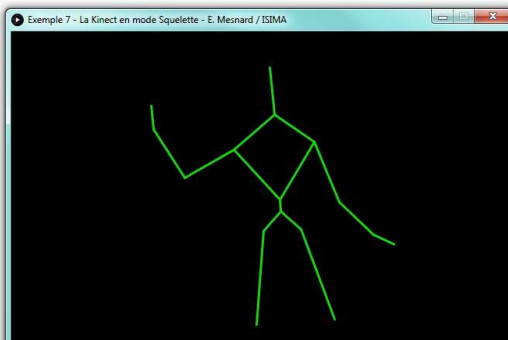
```

72

### IV.5 – Exemple 7 : Kinect en mode Squelette

```
// Trace d'un seul membre du corps
void dessinMembre(int userId, int jointType1, int jointType2) {
    int[] jointPos1, jointPos2 ; // Coordonnees des membres

    // Verification de la presence effective du membre
    if ( (s[userId].isJointTracked(jointType1)==true) &&
        (s[userId].isJointTracked(jointType2)==true) ) {
        // Recuperation des coordonnees 2D, proportionnelles a la taille de la fenetre
        jointPos1 = s[userId].get2DJoint(jointType1, width, height);
        jointPos2 = s[userId].get2DJoint(jointType2, width, height);
        // Trace du trait
        line(jointPos1[0], jointPos1[1], jointPos2[0], jointPos2[1]);
    }
}
```



73

### TD 5 Réalité Augmentée : KinectShow



- Travail demandé :
  - Faire apparaître une image virtuelle entre les mains d'un utilisateur
  - Dimensionner en permanence l'image pour rester cohérent
  - Lorsque l'écartement est trop faible, ne plus afficher l'image « augmentée »
  - Optionnel :
    - gestion de plusieurs images
    - changement d'image lors du rapprochement des 2 mains



74