# PSP Ultimate Head Tracker

Par mcidclan 23 avril 2010

**Définition :**
Cette librairie codée en c++ permet la récupération d'informations relatives au tracking d'une partie du corps humain tel que la tête ou la main par une mise en évidence des pixels susceptibles d'appartenir à une couleur de peau.

**Important**
Pour de meilleurs résultats lors de l'utilisation d'une application faisant appel à cette librairie, veuillez, respectez au mieux les recommandations suivantes :

- Être dans un endroit bien éclairé par de la lumière blanche. (évitez les éclairages jaunâtre)
- Être placer dos à un mur blanc dégagé .
- Porter des vêtements de couleurs unis, de préférence blanc ou gris.
- Porter un col roulé, s'il s'agit d'un Head tracking.
- Avoir de longues manches, s'il s'agit d'un Hand tracking .

Uht.h permet l'utilisation d'un tracker déjà opérationnel.

Créez l'objet destiné à gérer le flux vidéo et permettant la récupération
des informations relatives au tracking comme ceci : **myUht** = **new** Uht();
Passez true en paramètre du constructeur pour indiquer l'utilisation du buffer RGB_565 .

Initialisez les buffers nécessaires au travail du tracker (WW = 160 et HH = 120) :

```
void *bwork = malloc(0x34D80);
myUht->initBuffer(bwork, true, true);
```

Pour un rendu direct, sans application du flou allouez 0x22240 et passez true au second paramètre sinon allouez 0x34D80 et passez false. Le dernier paramètre indique l'utilisation du buffer RGB_565. Informations sur les buffers :

**public**
FB_MPEG Pointe toujours sur le Direct Buffer correspondant au flux de la camera
FB_8888 pointe sur un  buffer de WW*HH*4 octets, type u32
FB_565 pointe sur un buffer de WW*HH*2 octets, type u16

**private**
B_XCONT un buffer de WW octets, type u8
B_YCONT un buffer de HH octets, type u8
B_TMP un buffer de WW*HH octets, type u8

Vous avez ensuite la possibilité de donner les dimensions minimales  en x et y du nuage de pixel à traquer : **myUht**->setMinDim(24, 24);

Ainsi que de définir une grille de pixelisation sur le buffer de traitement : **myUht**->setGrid(10, 50);
10 correspond à la largeur d'une case pixelsante, et 50 à la tolérance aux pixels externes au nuage.
Ici nous aurons pour  une case : largeur*hauteur = 10*10 = 100 pixels, tolérance = 100/2 = 50

Initialisez la camera. La méthode prend pour argument la priorité du thread associé au tracker .
**myUht**->initCam(0x27);

Vous avez la possibilité de récupérer l'id du thread ainsi : **myUht**->thid;

**myUht**->run; vous renverra false lorsque la récupération du flux échoue.
**myUht**->direction ; Correspond à sceUsbCamGetLensDirection()

**myUht**->pOut; Contient les 4 extremums du nuage de pixels appartenant à la couleur de peau.
Les extremums sont : X début, X fin et Y début, Y fin. (pOut.x, pOut.y, pOut.z, pOut.w)

**delete** **myUht**; Pour détruire le tracker dans son intégralité (libérer la mémoire, libération des drivers et déchargement des modules).Pour ne pas prendre en compte le déchargement des modules passez true à **myUht**->relativeClean.

*Le gestionnaire de frames*

> Activez le gestionnaire de frames (si besoin) . Ce dernier forcera le tracker
> à attendre un signal pour passer à la frame suivante.
> **myUht**->fm_activated();
>
> La méthode suivante renvoie true lorsque le tracker  attend un signal pour continuer
> **myUht**->fm_isWaiting();
>
> Envoi un signal de passage à la frame suivante
> **myUht**->fm_nextFrame();

mcidclan
m.cid.clan@gmail.com