class PlanImageHealpix (nécessite healpixtools.jar, voir s'il existe une version plus récente) extends PlanImage

avec une projection AITOF et centrée sur 0,0 avec tout le ciel par défaut
utilise type = IMAGEHEALPIX;

cacheImageFits(MyInputStream dis)
lit le fichier et le stocke dans une « map » en bitpix à -32
copie les pixels en 8 bits avec la calib demandée (hpx2Fits)

taille max 1024*1024 des pixels

Visualisation

ViewSimple : **createZoomPixels** (fonction de cropping d'Aladin)

```
if( p.type==Plan.IMAGEHEALPIX )
((PlanBG)p).getVisiblePixels(pixelBG, x1,y1,w,h,w*zoom);
```

class HealpixKey

cacheImageFits : lit le fichier au format CDS-HEALPIX
contenant les 4 coins dans les tags :
"C"+(i+1)+"_RA"
"C"+(i+1)+"_DEC"

getNpixList

```
/**
 * Depuis une position et une résolution, cherche la liste des numéros
 * de pixels dans la région
 * @param order
 * @param center
 * @param radius
 * @return
 */
```

(utilise cds.tools.pixtools.Pixtools)

getNeighbours

```
/**
 * Cherche la liste des pixels voisins concernés par une vue
 * @param order ordre des fichiers
 * @param npixList la liste des pixels actuellement affichés
 * @param center centre de la vue
 * @param radius demi diagonale de la vue
 * @return
 */
```

(utilise cds.tools.pixtools.Pixtools)

class PlanBG : extends PlanImage

construit une projection à une position et taille angulaire dépendant du survey dans un tableau de pixels de taille 1024*1024

askServer

```
/**
 * Cherche la liste des pixels concernés autour d'une position Si pixview
 * contient déjà  des pixels, cherche uniquement dans le voisinage Ecrit la
 * nouvelle liste de pixels dans pixview Lit chacun des fichiers et les
 * enregitre dans pixlist (liste globale des pixels lus)
 *
 * @param pstock
 *          structure dans laquelle sera stockée les infos de projection
 *          de l'image produite
 * @param zoom
 */
```

getProj

```
/**
 * Définit a propri la nouvelle projection
 * @param pstock pixelStock dans lequel sera stocké la nouvelle projection
 * @param center centre la vue (et de la nouvelle proj)
 * @param zoom zoom qui est appliqué (qui change la résolution angulaire, par rapport
à projd)
 */
```

getVisiblePixels

```
/**
 * Recherche un bloc de pixels qui couvre la zone décrite. Si celui existe
 * déjà, le retourne via pixelBG, sinon va le générer dans un thread séparé
 * @param widthMax
 */
```

startProduct : Calcule la nouvelle projection en fonction du zoom et demande l'affichage de chacun des losanges.

class PixelStock (Stockage des pixels disponibles dans les différentes résolutions)

```
public Vector<Long> pixlist = new Vector<Long>();
int width; // Longueur en pixel d'une ligne
int height;
byte[] pixels; // Le bloc de pixels
Coord c[]; // Les coord RA,DEC et X,Y des 4 coins (HG->HD->BD->BG)
Projection proj; // Projection associée au bloc de pixel (TAN
                                        // central)
int status; // Etat du buffer
long time; // Date de la derniére utilisation
```

class PixelBG {

```
   byte []pixels;  // Tableau de pixels 8 bits issu d'une mappin de losanges Healpix
   int width;      // Nbre de pixels d'une ligne
   int x,y,w,h;    // Portion à extraire
   Projection proj;// Projection à utiliser
   double zoom;    // Facteur de zoom manuel à appliquer a posteriori pour obtenir une vue
   boolean theBest;// true si les pixels ne peuvent être améliorer (résolution...)
   int status;
}
```