URL: http://cxc.harvard.edu/ciao3.4/pixlib.html
Last modified: December 2006

AHELP for CIAO 3.4

pixlib

Context: modules

Jump to: Description Example An example of how to use the pixlib module CHANGES IN CIAO 3.2 Bugs See Also

Synopsis

The S−Lang interface to the CXC pixlib library

Description

The pixlib module is the interface between the S−Lang interpreter (see "ahelp slang") and the CXC pixlib library (see "ahelp dmcoords" for the more−limited tool−based interface to the pixlib library). This document provides an overview of the features of the pixlib module, and tips for using it efficiently in a S−Lang program. Detailed descriptions of each function are provided by individual ahelp entries.

The pixlib module is not available by default; to use it in a S−Lang program, it must be loaded using the S−Lang require() function:

```
require("pixlib");
```

Functions provided by the module

The following functions are provided by the module; use "ahelp <function>" to get a detailed description of a function:

Function name with arguments
pix_init_pixlib([String_Type, String_Type])
pix_close_pixlib()
pix_disp_config()
pix_set_detector(String_Type)
pix_set_tdetsys(String_Type)
pix_set_fpsys(String_Type)
pix_set_gdpsys(String_Type)
pix_set_aimpoint(String_Type)
pix_set_aimpoint(Double_Type, Double_Type, Double_Type)
pix_set_grating(String_Type)

pix_set_grating(Integer_Type)
pix_set_grating(String_Type, Integer_Type)
pix_set_gzo(Double_Type, Double_Type)
pix_set_simoffset(Double_Type, Double_Type, Double_Type [,Double_Type])
Double_Type pix_get_flength()
Double_Type pix_get_rowland()
Double_Type pix_get_grating_period()
Double_Type pix_get_grating_angle()
Array_Type pix_chip_to_fpc(Integer_Type, Double_Type, Double_Type)
(Integer_Type,Array_Type) pix_fpc_to_chip(Double_Type, Double_Type)
Array_Type pix_chip_to_tdet(Integer_Type, Double_Type, Double_Type)
(Integer_Type, Array_Type) pix_tdet_to_chip(Double_Type, Double_Type)
Array_Type pix_fpc_to_msc(Double_Type, Double_Type)
Array_Type pix_chip_to_gdp (Integer_Type, Double_Type, Double_Type)
Array_Type pix_fpc_to_gdp(Double_Type, Double_Type)
Array_Type pix_gdp_to_gac(Double_Type, Double_Type)
Array_Type pix_gac_to_gdp(Double_Type, Double_Type)
Double_Type pix_get_energy(Double_Type, Double_Type)
Double_Type pix_get_grating_wavelength(Double_Type, Double_Type)
Array_Type pix_apply_aspect(Double_Type, Double_Type, Double_Type, Double_Type, Double_Type)
Array_Type pix_deapply_aspect(Double_Type, Double_Type, Double_Type, Double_Type, Double_Type)
Array_Type pix_dmTanPixToWorld(Array_Type, Array_Type, Array_Type, Array_Type)
Array_Type pix_dmTanWorldToPix(Array_Type, Array_Type, Array_Type, Array_Type)

Pixlib and Chandra Coordinate Systems

A number of coordinate systems have been defined for use with Chandra. The pixlib module makes conversion between these systems much easier, but the user must first familiarize themselves with the various systems, which are described in Coordinate Systems Paper I: Imaging.

Setting up pixlib geometry

The pixlib module must be initialized, after it is loaded, with the pix_init_pixlib call. This sets the default geometry for Chandra in flight. Most of these values (those set by the pix_set_tdetsys, pix_set_fpsys, or pix_set_gdpsys calls) should not be changed. The Science Instrument Module (SIM) position, however, must be set correctly for each observation and aspect solution in order to properly convert between sky and detector coordinates. This is done using the pix_set_aimpoint() call (see example below) which uses the nominal SIM_X, SIM_Y, and SIM_Z values (available in the event file header) and the aspect solution DY and DZ values as inputs.

Restarting pixlib

Pixlib should not be closed and re−initialized within a single session, as this causes internal variables to be mis−set and incorrect results produced. This bug will be addressed in a future release; the simple workaround is to call pix_init_pixlib only once, immediately after the call to require("pixlib").

Example

```
chips> require("pixlib")
chips> pix_init_pixlib()
chips> pix_disp_config()
****** PIXLIB System Configuration ******
Telescope = flight
Focal Length (mm) = 10070.000
Detector = ACIS
Focal Plane Sys. = FP−1.1
Tiled Detector Plane Sys. = ACIS−2.2
SIM Offset (mm) = (0.684 0.75 236.552)
Aim Point(AI1) (mm) = (0 0 −237.5)
Grating Arm = HEG
Grating Order = 1
Dispersion Plane Sys. = ASC−GDP−1.1
Rowland Circle (mm) = 8632.48
```

An example of how to use the pixlib module

The following routines can be used to calculate sky pixel coordinates from the chip coordinates along with the aspect solution. After applying the aspect solution, the chipx, chipy values are converted to Focal Plane Coordinates (in the event file, detx, dety) and then to Celestial coordinates (in degrees) and finally to sky pixels (in the event file, x,y).

```
define GetKey(file, key) {

   variable result=NULL;
   punlearn("dmkeypar");
   pset("dmkeypar","infile",file);
   pset("dmkeypar","keyword",key);
   pset("dmkeypar","mode","h");
   () = system("dmkeypar");

   if (strcmp(pget("dmkeypar","exist"),"yes")==0) {
      result = pget("dmkeypar","value");
   }
   return result;
}

define calc_sky_pixel(evtfile, asolfile) {

   variable iE, iA, gp, dy, dz, fpc, tdet;
   variable skydeg, skypix, ra, dec, roll, time, pointing;
   % Reading input files
   variable evt = readfile(evtfile);
   variable asp = readfile(asolfile);
   %  First, get the setup values
   variable ra_nom = atof(GetKey(evtfile,"RA_NOM"));
   variable dec_nom = atof(GetKey(evtfile,"DEC_NOM"));
   variable roll_nom= atof(GetKey(evtfile,"ROLL_NOM"));
   variable crpix = [4096.5,4096.5]; % center pixels for ACIS
   variable crdelt= [−0.492,0.492]/3600;  % pixelsize in degrees
   variable sim_x = atof(GetKey(evtfile,"SIM_X"));
   variable sim_y = atof(GetKey(evtfile,"SIM_Y"));
   variable sim_z = atof(GetKey(evtfile,"SIM_Z"));
   %  Set up pixlib for ACIS
   pix_set_detector("ACIS");
```

```
   if ((asp.time[0] > evt.time[0]) or
       (asp.time[asp._nrows-1] < evt.time[evt._nrows-1])) {
      message("Aspect solution does not cover entire observation.");
      return NULL;
   }
   for (iE=0;iE<evt._nrows;iE++) {
      gp = where(asp.time < evt.time[iE]);
      iA = gp[length(gp)-1];
      dy = asp.dy[iA];
      dz = asp.dz[iA];
      ra = asp.ra[iA];
      dec= asp.dec[iA];
      roll=asp.roll[iA];
      pix_set_aimpoint(sim_x,sim_y+dy,sim_z+dz);
      fpc=pix_chip_to_fpc(evt.ccd_id[iE],evt.chipx[iE],evt.chipy[iE]);
      pointing = [ra, dec, roll];  % current pointing
      skydeg = pix_dmTanPixToWorld(fpc, pointing, crpix, crdelt);
      pointing = [ra_nom, dec_nom, 0];  % nominal pointing
      skypix = pix_dmTanWorldToPix(skydeg, pointing, crpix, crdelt);
      vmessage("%12.6f, %12.6f",skypix[0],skypix[1]);
   }
} % calc_sky_pixel
```

The above function shows how to convert between the major coordinate systems of interest to the observer. Also provided is a short routine that returns header keys in S−lang, using the paramio library (see "ahelp paramio"). This routine could, for example, be easily converted to perform the same function as the CIAO reproject_events tool.

```
sherpa> require("pixlib");
sherpa> pix_init_pixlib();
sherpa> require("paramio");
sherpa> evalfile("calc_sky_pixel.sl");
1
sherpa> calc_sky_pixel("acis_evt2.fits","pcad_asol1.fits")
1864.309455, 3218.957142
1976.600734, 3319.221975
2145.482788, 2906.326701
2291.051471, 3308.732751
2518.960912, 3042.806528
2525.190971, 3409.252333
2606.418985, 2942.680818
1780.086398, 4941.340299
...
```

CHANGES IN CIAO 3.2

The module can now be loaded by using the

```
require("pixlib");
```

statement, although the previous method (loading with the import command) still works.

pix_set_simoffset()

An optional argument "dtheta" has been added to the pix_set_simoffset() routine. This is used to set the DTHETA aspect solution for calculating focal−plane coordinates (prior to CIAO 3.2 this was documented as the missing pix_set_mirror function which could lead to small offsets when compared to the results from acis_process_events

and hrc_process_events).

pix_fpc_to_msc()

The pix_fpc_to_msc() routine has been added as a convenience function for people interested in calculating off−axis angles.

Removed conflict with the caldb module

Prior to CIAO 3.2 the caldb and pixlib modules had to be started in a particular order (caldb then pixlib) when used together, otherwise a warning message was generated. This restriction has been removed.

Bugs

See the bugs page for the pixlib library on the CIAO website for an up−to−date listing of known bugs.

See Also

calibration
ardlib, caldb
chandra
coords, guide, isis, level, pileup, times
chips
chips
concept
autoname, parameter, stack, subspace
dm
dm, dmbinning, dmcols, dmfiltering, dmimages, dmimfiltering, dmintro, dmopt, dmregions, dmsyntax
gui
gui
modules
caldb, paramio, stackio
pixlib
pix_apply_aspect, pix_chip_to_fpc, pix_chip_to_gdp, pix_chip_to_tdet, pix_close_pixlib, pix_deapply_aspect, pix_disp_config, pix_dmtanpixtoworld, pix_dmtanworldtopix, pix_fpc_to_chip, pix_fpc_to_gdp, pix_fpc_to_msc, pix_gac_to_gdp, pix_gdp_to_gac, pix_get_energy, pix_get_flength, pix_get_grating_angle, pix_get_grating_period, pix_get_grating_wavelength, pix_get_rowland, pix_init_pixlib, pix_set_aimpoint, pix_set_detector, pix_set_fpsys, pix_set_gdpsys, pix_set_grating, pix_set_gzo, pix_set_simoffset, pix_set_tdetsys, pix_tdet_to_chip
slang
overview, slang, tips
tools
quizcaldb

60 Garden Street, Cambridge, MA 02138 USA.
Last modified: December 2006