Dsimulator slit selection procedures

There are 3 procedures that are used to select and organize the targets based on the user priorities

* **selector**: Does an auto selection of slits.
* **sel\_rank**: Selects slits with priority ranking.
* **sel\_sort**: Sorts the selected/selection lists.

The procedures are defined in the script **dsim.x**.

The **selector** procedure is called in **dsimgraph.x**. The **sel\_rank** and **sel\_sort** procedures are called in the **selector** procedure.

The selection procedures require the auxiliary procedures:

* **data\_init**: Initialize data structure for telescope/background data.
* **targ\_init**: Initialize data structure for targets.

to define the pointers to data structures for the instrument/telescope and targets. The auxiliary procedures are defined in the script **dsim2.x**. The **data\_init** and **targ\_init** procedures are defined called in the **dsim.x** script.

# data\_init

Reads the Dsimulator input parameter file and stores the values in a structure that is pointed at by the pointer indat.

RA0\_FLD(indat) = DEGTORAD(15. \* clgetd ("ra0"))

DEC0\_FLD(indat) = DEGTORAD(clgetd ("dec0"))

HA\_FLD(indat) = DEGTORAD(15. \* clgetr ("ha0"))

PA\_ROT(indat) = DEGTORAD(clgetd ("PA0"))

TEMP(indat) = clgetr ("temp")

PRES(indat) = clgetr ("pressure")

WAVER(indat) = clgetr ("lambda\_cen") \* 1.e-4 # in microns

WAVEMN(indat) = clgetr ("blue") \* 1.e-4

WAVEMX(indat) = clgetr ("red") \* 1.e-4

SLIT\_GAP(indat) = clgetr ("sep\_slit") # sep. bet. slits, asec

DEF\_HLEN(indat) = 0.5 \* clgetr ("min\_slit") # min. length in arcsec

DEF\_BOXR(indat) = 0.5 \* clgetr ("box\_sz") # box 1/2-length in arcs

DEF\_SLWID(indat) = clgetr ("slit\_width") # slit width in arcsec

STD\_EQX(indat) = clgetr ("equinox")

if (clgetb ("proj\_len"))

PROJ\_LEN(indat) = YES

else

PROJ\_LEN(indat) = NO

if (clgetb ("no\_overlap"))

ADJ\_LEN(indat) = YES

else

ADJ\_LEN(indat) = NO

call clgstr ("maskid", DESNAME(indat), SZ\_LINE)

call clgstr ("maskid", BLUNAME(indat), SZ\_LINE)

call clgstr ("guiname", GUINAME(indat), SZ\_LINE)

call clgstr ("observer", BLUOBSR(indat), SZ\_LINE)

call clgstr ("author", DESAUTH(indat), SZ\_LINE)

call clgstr ("project", PROJNAME(indat), SZ\_LINE)

call clgstr ("dateobs", USEDATE(indat), SZ\_LINE)

call strcpy ("DEIMOS", INSTRUME(indat), SZ\_LINE)

call strcpy ("Dsimulator: Ver 0.0b", DESCREAT(indat), SZ\_LINE)

# targ\_init

Reads the input target file and the input telescope/background parameters and stores the result in a structure that is pointed at by the pointer tdat.

This is an example of a typical input catalog:

# Optional mask name, center and PA

DracoX 259.808203826 57.938868534 PA=0.0

# Name RA Dec Equinox Magnitude Band Pcode Sample (Select SlitPA Len1 Len2 SlitWidth)

611170 17:17:31.470 57:54:009.30 2000.0 21.670 I 1 2

654366 17:17:31.530 57:57:025.40 2000.0 23.764 I 1 2

568485 17:17:31.550 57:50:000.40 2000.0 24.025 I 1 2

667009 17:17:31.570 57:58:022.90 2000.0 23.843 I 1 2

616836 17:17:31.580 57:54:039.20 2000.0 22.517 I 1 2

677698 17:17:31.650 57:59:011.90 2000.0 23.706 I 1 2

730565 17:17:31.650 58:04:039.80 2000.0 24.298 I 1 2

First look for the mask center:

if (ndx == 0) { # check for mask center

call gargwrd (idstr, SZ\_ID)

call gargd (alpha)

call gargd (delta)

call gargr (equinox)

call gargwrd (workstr, SZ\_ID)

call strcpy (workstr, pastr, 3)

if (nscan() >= 5 && streq (pastr, "PA=")) {

call eprintf ("Field data from input file...\n")

RA0\_FLD(indat) = DEGTORAD(15. \* alpha)

DEC0\_FLD(indat) = DEGTORAD(delta)

STD\_EQX(indat) = equinox

eqnx\_std = STD\_EQX(indat) # TMP!

if (sscan (workstr[4]) != EOS) {

call gargr (pangle)

if (nscan() != 0) {

PA\_ROT(indat) = DEGTORAD(pangle)

next

} else {

call fatal (0, "Bad format on PA=")

}

}

}

call reset\_scan ()

}

Do some checks on the file format (number of fields, reasonable number of targets) and fill the structure with all the targets:

INDEX(tdat,ndx) = ndx

RA0(tdat,ndx) = DEGTORAD(alpha\*15.)

DEC0(tdat,ndx) = DEGTORAD(delta)

MAG(tdat,ndx) = magn

if (strlen (passband) > 1)

call eprintf ("WARNING: passband truncated to 1 char\n")

PBAND(tdat,ndx) = passband[1]

PCODE(tdat,ndx) = prior

SAMPL(tdat,ndx) = nlist

SEL(tdat,ndx) = selcode

if (pangle != INDEF)

PA(tdat,ndx) = DEGTORAD(pangle)

else

PA(tdat,ndx) = INDEF

LEN1(tdat,ndx) = l1 # Requested length above object (in direction of PA) What is this?

LEN2(tdat,ndx) = l2 # Requested length below object (opposite to PA) What is this?

# XXX Assign slit-width

if (PCODE(tdat,ndx) == CODE\_AS) {

PA(tdat,ndx) = INDEF

SLWID(tdat,ndx) = 2.\*DEF\_BOXR(indat)

} else {

SLWID(tdat,ndx) = DEF\_SLWID(indat)

}

Finally, simply define ntarg as the number of targets read from the catalog file.

# selector

The call to selector from within **dsimgraph.x** is as follows:

do nlist = 1, NLIST\_MAX

call selector (indat, tdat, ntarg, nlist, 2.\*(DEF\_HLEN(indat)+SLIT\_GAP(indat)), psum) # XXX

It does NOT include a weighting function to keeps things toward the center.

Callable parameters:

* indat: pointer to instrument/telescope data structure
* tdat: pointer to data structure for targets
* ntarg: number of targets
* nlist: working list. The external loop runs through all possible list codes defined in the input catalog file.
* minsep: minimum separation
* psum: sum of selected priorities

Internal variables:

int nlist # list to work on

real minsep # XXX min. separation -- probably should be in DEFDAT

int psum # sum of selected priorities (returned)

int nopt, npre # number of options, prev. selected objects

int i, ndx

int ix # starting index for search (saves time)

int nselect # Number of selected slits

real xlow, xupp, xskip

pointer bufx1, bufx2 # TMP? buffers for pre-sel. objs

pointer bufn, bufx, bufp, bufsel # TMP, for now

Run through the target list. Count the pre-selected targets (**npre**) and the objects in each active list that are inside the mask (**nopt**).

nopt = 0

npre = 0

do i = 0, ntarg-1 {

if (SEL(tdat,i) == YES) {

npre = npre + 1

} else if (SAMPL(tdat,i) == nlist && STAT(tdat,i) == YES) {

nopt = nopt + 1

}

}

* **SEL(tdat,i) == YES**: Pre-selected targets in the input file. These have an input selection code other than 0. The default value in the input file is 0.
* **SAMPL(tdat,i) == nlist**: Sample to which the object belongs defined in the input file. When auto-selecting, objects in sample 1 are selected first. The remaining space is then filled with sample 2, then sample 3, etc. The default value in the input file is 1.
* **STAT(tdat,i) == YES**: Targets that are inside the mask. STAT is the output from the check\_stat procedure.

Allocate memory for different temporary vectors. This memory will be freed at the end of the selector procedure:

call malloc (bufx1, npre, TY\_INT)

call malloc (bufx2, npre, TY\_REAL)

call malloc (bufn, nopt, TY\_INT)

call malloc (bufx, nopt, TY\_REAL)

call malloc (bufp, nopt, TY\_INT)

call malloc (bufsel, nopt, TY\_INT)

Run through the target list again and fill the vectors:

nopt = 0

npre = 0

do i = 0, ntarg-1 {

if (PCODE(tdat,i) == CODE\_GS) # GS's don't take space

next

if (SEL(tdat,i) == YES) {

# Memr[bufx1+npre] = XARCS(tdat,i) - LEN1(tdat,i)

# Memr[bufx2+npre] = XARCS(tdat,i) + LEN2(tdat,i)

Memr[bufx1+npre] = X1(tdat,i)

Memr[bufx2+npre] = X2(tdat,i)

npre = npre + 1

} else if (SAMPL(tdat,i) == nlist && STAT(tdat,i) == YES && PCODE(tdat,i) > 0) {

Memi[bufn+nopt] = i # INDEX(tdat,i) XXX

Memr[bufx+nopt] = XARCS(tdat,i)

Memi[bufp+nopt] = PCODE(tdat,i)

nopt = nopt + 1

}

}

* **PCODE(tdat,i) == CODE\_GS**: Skip guide stars.
* **SEL(tdat,i) == YES**: Pre-selected targets with input code non 0. The vector is filled with the left and right edges of the corresponding slit. These are calculated in the procedure **tel\_coords** (defined inside **dsim2.x**) that converts the refracted RA,Dec into offsets from telescope center.
* **SAMPL(tdat,i) == nlist && STAT(tdat,i) == YES and PCODE(tdat,i) > 0**: Targets in the current working list that are inside the mask and that have a priority code > 0. The vector is filled with a running index starting with 0 at the top of the input file, the X coordinate of the target and the priority code. The X coordinate of the target is calculated in the procedure **tel\_coords** (defined inside **dsim2.x**) that converts the refracted RA,Dec into offsets from telescope center.

Sort both lists:

call sel\_sort (Memr[bufx1], Memr[bufx2], npre, Memi[bufn], Memr[bufx], Memi[bufp], nopt)

Rank the selected targets

# The number of "gaps" to search is npre+1

ndx = 0

xlow = XLOW\_LIM

xskip = 0.

nselect = 0 # triggers init in sel\_rank

if (nopt > 0) {

do i = 0, npre {

if (i < npre) {

xupp = Memr[bufx1+i]

xskip = Memr[bufx2+i] - Memr[bufx1+i]

} else {

xupp = XUPP\_LIM

}

## old ...

# if (xupp <= xlow)

# next

# call sel\_rank (Memr[bufx], Memi[bufp], Memi[bufn],

# Memi[bufsel], nopt, ix, xlow, xupp, minsep, nselect)

# xlow = xupp + xskip

if (xupp > xlow) {

call sel\_rank (tdat, indat, Memi[bufn],

Memi[bufsel], nopt, ix, xlow, xupp, minsep, nselect)

}

xlow = xupp + xskip

}

}

XLOW\_LIM and XUPP\_LIM are the lower and upper limits of the mask dimensions in the X axis.

When the number of optional targets is > 0, run through all the pre-selected targets. Initially define **xlow=XLOW\_LIM**. For all pre-selected targets except for the last one (rightmost one):

1. Define **xupp** as the left edge of the current pre-selected slit.
2. Define **xskip** as the length of the current pre-selected slit.
3. If **xupp > xlow** rank it. Rank pre-selected slits whose left edge is located to the right of the right edge of the previous pre-selected slit.
4. **xlow = xupp + xskip**. Jump in X by the slit width of the current pre-selected slit.

Select the slits

#...select the mask slits

if (nselect > 0) {

do i = 0, nselect-1 {

SEL(tdat,Memi[bufsel+i]) = YES

}

}

psum = 0

do i = 0, ntarg-1 {

if (SEL(tdat,i) == YES)

psum = psum + max (PCODE(tdat,i), 0) # NO GS, AS

}

# sel\_rank

Select slits with priority ranking. Find the next possible slit and then look up to one min-slit width away for higher priority objects. The higher priorities are down-weighted depending on their distances.

pointer tdat

pointer indat

int tndex[npt] # Index of selectable slits

int sel[npt] # selected objects

int npt # Number of objects

int isel # starting index

real xlow, xupp # xrange to fill

real minsep # minimum separation (arcsec)

int nsel # Number of selected objects

int i, j, ndx

real x, xj, xnext, xlook, xstop, xlast

real len

real prisel, prinorm

Compare the right edge of one slit to the left edge of the next. Return if the separation is less than the minimum separation:

# Can we fit a minimum slit in here?

if (xupp - xlow < minsep) # probably too restrictive

return

Check the last target on the list:

# Start at half a slit length; stop inside half slit length

ndx = tndex[npt]

x = XARCS(tdat,ndx)

xstop = min (x, xupp-0.5\*minsep) # last target or upper limit

xnext = xlow + 0.5 \* minsep

xlast = xlow

Loop through all the selectable targets (between pre-selected targets, I think) up to the last one on the list:

# Loop through to end

for (i = isel + 1; i <= npt; i = i + 1) {

ndx = tndex[i]

x = XARCS(tdat,ndx)

if (x < xnext)

next

if (X1(tdat,ndx) < xlast)

next

if (x > xstop) {

isel = i - 1

break

}

isel = i

len = X2(tdat,ndx) - X1(tdat,ndx)

prisel = PCODE(tdat,ndx) / (x - xlast) / len

# Now look for higher priority to win out, over range (xlast,xlook)

xlook = min (x+minsep, xstop)

if (isel < npt) {

do j = isel+1, npt {

ndx = tndex[j]

if (X1(tdat,ndx) >

X2(tdat,isel)+SLIT\_GAP(indat)) {

next # There is no conflict

} # XXX but prisel gets higher?

if (X2(tdat,ndx) > x\_upp)

next # XXX Can't use

xj = XARCS(tdat,ndx)

if (xj >= xlook)

break

len = X2(tdat,ndx) - X1(tdat,ndx)

prinorm = PCODE(tdat,ndx) / (xj - xlast) / len

if (prinorm > prisel) {

x = xj

isel = j

prisel = prinorm

}

}

}

nsel = nsel + 1

ndx = tndex[isel]

sel[nsel] = ndx

xlast = X2(tdat,ndx)

xnext = xlast + 0.5 \* minsep

i = isel # Reset search start point

}

# sel\_sort

Sort the pre-selected (**npre**) and optional (**nopt**) lists separately in X coordinate, from lower to higher.