```
;pro modstreamB

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;  Run packets by advancing packets dt at a time.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

endtime = options.endtime
runtime = dindgen(round(options.endtime)/dt+1)*dt
nt = n_elements(runtime)
options.at_once = 1
npack = options.packets

options.endtime = 0
source_distribution, geometry, spatialdist, speeddist, angulardist, PerturbVel, $
  options, seed, loc=loc, startloc=startloc
*startloc.TravelTime = runtime
options.endtime = endtime

;; Make arrays for the final values
x2 = dblarr(npack, nt) & x2[*,0] = *loc.x
y2 = dblarr(npack, nt) & y2[*,0] = *loc.y
z2 = dblarr(npack, nt) & z2[*,0] = *loc.z
vx2 = dblarr(npack, nt) & vx2[*,0] = *loc.vx
vy2 = dblarr(npack, nt) & vy2[*,0] = *loc.vy
vz2 = dblarr(npack, nt) & vz2[*,0] = *loc.vz
frac2 = dblarr(npack, nt) & frac2[*,0] = *loc.frac
time2 = dblarr(npack, nt)
if (options.trackloss) then begin
  loss2 = dblarr(npack,nt)
  hit2 = dblarr(npack,n_elements(*SystemConsts.objects), nt)
  ring2 = dblarr(npack,nt)
  left2 = dblarr(npack,nt)
  map2 = dblarr(360,180,nt)
endif

xcyc, xc, yc
plot, findgen(10), /nodata, xr=[-10,10], yr=[-10,10], /iso, /xst, /yst
polyfill, xc, yc, color=4

phi0 = *geometry.phi
locmoon, -runtime*options.motion, *geometry.phi, *SystemConsts.a, *SystemConsts.orbrate, $
  x=satx, y=saty, ang=ang

(*loc.fintime)[*] = dt
for i=1L,nt-1 do begin
  w = where(frac2[*,i-1] GT 0, nw, comp=c)
  if (nw NE 0) then begin
    options.packets = nw
    *loc.x = x2[w,i-1]
```

```
      *loc.y = y2[w,i-1]
      *loc.z = z2[w,i-1]
      *loc.frac = frac2[w,i-1]
      *loc.vx = vx2[w,i-1]
      *loc.vy = vy2[w,i-1]
      *loc.vz = vz2[w,i-1]
      *loc.fintime = replicate(dt, nw)
      *geometry.phi = reform(ang[i,*])

;     if (nw NE 10) then stop
      driver, loc, geometry, options, forces,plasma_info, loss_info, sticking_info, $
        deposition, seed=seed
      x2[w,i] = *loc.x
      y2[w,i] = *loc.y
      z2[w,i] = *loc.z
      frac2[w,i] = *loc.frac
      vx2[w,i] = *loc.vx
      vy2[w,i] = *loc.vy
      vz2[w,i] = *loc.vz
      time2[*,i] = runtime[i]
      if (options.trackloss) then begin
        loss2[w,i] = *loc.lossfrac
        hit2[w,*,i] = *loc.hitfrac
        ring2[w,i] = *loc.ringfrac
        left2[w,i] = *loc.leftfrac
     ;   if (max(*loc.leftfrac ) NE 0) then stop
        map2[*,*,i] = *deposition.map
        lon2 = *deposition.longitude
        lat2 = *deposition.latitude
        destroy_structure, deposition
      endif

      if (c[0] NE -1) then begin
        x2[c,i] = x2[c,i-1]
        y2[c,i] = y2[c,i-1]
        z2[c,i] = z2[c,i-1]
        frac2[c,i] = frac2[c,i-1]
        vx2[c,i] = vx2[c,i-1]
        vy2[c,i] = vy2[c,i-1]
        vz2[c,i] = vz2[c,i-1]
      endif

  ;   plots, *loc.x, *loc.y, psym=8, color=2
;     wait, .1
    endif
  endfor

  ;; Rotate packets backward to fixed satellite
  if (options.motion) then begin
    for i=0,nt-1 do begin
      aa = -(ang[i,s]-ang[0,s])
```

```
      x3 = x2[*,i] * cos(aa) - y2[*,i] * sin(aa)
      y3 = x2[*,i] * sin(aa) + y2[*,i] * cos(aa)

      vx3 = vx2[*,i] * cos(aa) - vy2[*,i] * sin(aa)
      vy3 = vx2[*,i] * sin(aa) + vy2[*,i] * cos(aa)

      x2[*,i] = x3 & y2[*,i] = y3
      vx2[*,i] = vx3 & vy2[*,i] = vy3
    endfor
  endif

  *geometry.phi = phi0
  *loc.x = x2
  *loc.y = y2
  *loc.z = z2
  *loc.frac = frac2
  *loc.vx = vx2
  *loc.vy = vy2
  *loc.vz = vz2
  *loc.fintime = time2
  if (options.trackloss) then begin
    *loc.lossfrac = loss2
    *loc.hitfrac = hit2
    *loc.ringfrac = ring2
    *loc.leftfrac = left2
    deposition.longitude = ptr_new(lon2)
    deposition.latitude = ptr_new(lat2)
    deposition.map = ptr_new(map2)
  endif

  options.packets = npack

```