```
pro modstreamA, loc, startloc, geometry, spatialdist, speeddist, angulardist, $
  PerturbVel, options, dt, seed

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Run streamlines - the result of this version is a set of packets in which
;; a packet with traveltime = t has been ejected from the object t seconds ago
;;
;; The satellites end up at their specified positions and
;; each packet has travelled for (runtime) seconds
;;
;; Version history:
;;   2.3: 1/14/2010
;;    - added ability to track packet loss and surface deposition
;;   2.1: created from a section in modeldriver_2.0 (2 Dec 2008)
;;     - Tries to do everything with a single call to the driver. If there are too
;;       many packets or time steps, then it will iterate
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

if (Sticking_info.stickcoef NE 1) then stop

maxsteps = 200000L ;; do everything at once if fewer points than maxsteps

endtime = options.endtime
runtime = dindgen(round(options.endtime/dt)+1)*dt
nt = n_elements(runtime)
options.at_once = 1
options.endtime = 0.

;; Find the initial conditions for t=tfinal
source_distribution, geometry, spatialdist, speeddist, angulardist, PerturbVel, $
 options, seed, loc=loc, startloc=startloc
*startloc.TravelTime = runtime
npack = options.packets
options.endtime = endtime

;; Save the starting values
x0 = *loc.x
y0 = *loc.y
z0 = *loc.z
vx0 = *loc.vx
vy0 = *loc.vy
vz0 = *loc.vz

;; Make arrays for the final values
x2 = dblarr(npack, nt)
y2 = dblarr(npack, nt)
z2 = dblarr(npack, nt)
vx2 = dblarr(npack, nt)
vy2 = dblarr(npack, nt)
```

```
vz2 = dblarr(npack, nt)
frac2 = dblarr(npack, nt)
time2 = dblarr(npack, nt)
*startloc.phi = dblarr(npack, nt)

nsteps = long(nt)*long(npack)
print, nsteps
if ((s NE 0) and (options.motion)) then locmoon, runtime, (*geometry.phi)[s], $
  (*SystemConsts.a)[s], (*SystemConsts.orbrate)[s], x=satx, y=saty, ang=ang

for i=0L,npack-1 do begin
  ;; Run one packet at a time for all the times

  x1 = replicate(x0[i], nt)
  y1 = replicate(y0[i], nt)
  z1 = replicate(z0[i], nt)
  vx1 = replicate(vx0[i], nt)
  vy1 = replicate(vy0[i], nt)
  vz1 = replicate(vz0[i], nt)

  ;; Determine the local starting point for each packet - only need to do this
  ;; if the starting position is a function of time
  if ((s NE 0) and (options.motion)) then begin
    (*startloc.phi)[i,*] = ang

    ;; Rotate packets to their proper start position
    aa = reform((*startloc.phi)[i,*] - (*geometry.phi)[s])

    xtemp = *loc.x & ytemp = *loc.y & ztemp = *loc.z
    x2[i,*] = x1 * cos(aa) - y1 * sin(aa)
    y2[i,*] = x1 * sin(aa) + y1 * cos(aa)
    z2[i,*] = z1

    vxtemp = *loc.vx &  vytemp = *loc.vy & vztemp = *loc.vz
    vx2[i,*] = vx1 * cos(aa) - vy1 * sin(aa)
    vy2[i,*] = vx1 * sin(aa) + vy1 * cos(aa)
    vz2[i,*] = vz1

    frac2[i,*] = 1d
    time2[i,*] = runtime
  endif else begin
    x2[i,*] = x1 & y2[i,*] = y1 & z2[i,*] = z1
    vx2[i,*] = vx1 & vy2[i,*] = vy1 & vz2[i,*] = vz1
    frac2[i,*] = 1d & time2[i,*] = runtime
  endelse
endfor

if (nsteps LT maxsteps) then begin
  options.packets = nsteps

  *loc.x = x2[*]
```

```
    *loc.y = y2[*]
    *loc.z = z2[*]
    *loc.vx = vx2[*]
    *loc.vy = vy2[*]
    *loc.vz = vz2[*]
    *loc.frac = frac2[*]
    *loc.finTime = time2[*]

    driver, loc, geometry, options, forces, plasma_info, loss_info, sticking_info, $
      deposition, seed=seed

    *loc.x = reform(*loc.x, npack, nt)
    *loc.y = reform(*loc.y, npack, nt)
    *loc.z = reform(*loc.z, npack, nt)
    *loc.vx = reform(*loc.vx, npack, nt)
    *loc.vy = reform(*loc.vy, npack, nt)
    *loc.vz = reform(*loc.vz, npack, nt)
    *loc.frac = reform(*loc.frac, npack, nt)
    *loc.fintime = reform(*loc.fintime, npack, nt)

    if (options.trackloss) then begin
      *loc.lossfrac = reform(*loc.lossfrac, npack, nt)
      *loc.hitfrac = reform(*loc.hitfrac, npack, nt, n_elements(*SystemConsts.objects))
      *loc.ringfrac = reform(*loc.ringfrac, npack, nt)
      *loc.leftfrac = reform(*loc.leftfrac, npack, nt)
    endif
  endif else begin
    ;; Make arrays for the final values
    x1 = dblarr(npack, nt)
    y1 = dblarr(npack, nt)
    z1 = dblarr(npack, nt)
    vx1 = dblarr(npack, nt)
    vy1 = dblarr(npack, nt)
    vz1 = dblarr(npack, nt)
    frac1 = dblarr(npack, nt)
    if (options.trackloss) then begin
      loss1 = dblarr(npack, nt)
      hit1 = dblarr(npack, n_elements(*SystemConsts.objects), nt)
      ring1 = dblarr(npack, nt)
      left1 = dblarr(npack, nt)
      map1 = dblarr(360, 180, n_elements(*SystemConsts.objects), nt)
    endif
    options.packets = npack

    for i=0L,nt-1 do begin
      *loc.x = x2[*,i]
      *loc.y = y2[*,i]
      *loc.z = z2[*,i]
      *loc.vx = vx2[*,i]
      *loc.vy = vy2[*,i]
      *loc.vz = vz2[*,i]
```

```
        *loc.frac = frac2[*,i]
        *loc.fintime = time2[*,i]

        if (i GT 0) then driver, loc, geometry, options, forces, plasma_info, $
          loss_info, sticking_info, deposition, seed=seed

        x1[*,i] = *loc.x
        y1[*,i] = *loc.y
        z1[*,i] = *loc.z
        vx1[*,i] = *loc.vx
        vy1[*,i] = *loc.vy
        vz1[*,i] = *loc.vz
        frac1[*,i] = *loc.frac
        if (options.trackloss) then begin
          loss1[*,i] = *loc.lossfrac
          hit1[*,*,i] = *loc.hitfrac
          ring1[*,i] = *loc.ringfrac
          left1[*,i] = *loc.leftfrac
          if (i GT 0) then begin
            map1[*,*,*,i] = *deposition.map
            lon1 = *deposition.longitude
            lat1 = *deposition.latitude
            destroy_structure, deposition
          endif
        endif
      endfor
      *loc.x = x1
      *loc.y = y1
      *loc.z = z1
      *loc.vx = vx1
      *loc.vy = vy1
      *loc.vz = vz1
      *loc.frac = frac1
      if (options.trackloss) then begin
        *loc.lossfrac = loss1
        *loc.hitfrac = hit1
        *loc.ringfrac = ring1
        *loc.leftfrac = left1
        deposition.longitude = ptr_new(lon1)
        deposition.latitude = ptr_new(lat1)
        deposition.map = ptr_new(map1)
      endif
    endelse
    options.packets = npack
```