```
function produce_image, files, savefile=savefile

common constants
common results

;;;;;;;;;;;;;;;;
;; Determine the image origin
s = (where(strcmp(*SystemConsts.objects, format.geometry.origin, /fold), ns))[0]
if (ns NE 1) then stop

;;;;;;;;;;;;;;;
;; Determine image field of view and rotation
geometry = format.geometry

image = dblarr((geometry.dims)[0],(geometry.dims)[1])
immin = geometry.center - geometry.width/2.
immax = geometry.center + geometry.width/2.

scale = geometry.width/(geometry.dims-1)   ;; [xscale,zscale] in Rplan/pix
Apix = (scale[0]*scale[1])*((*SystemConsts.radius)[s]*SystemConsts.rplan*1e5)^2
   ;; cm^2/pix

;; xaxis and zaxis in Robj measured from center of object
xaxis = findgen((geometry.dims)[0])*scale[0] + immin[0]
zaxis = findgen((geometry.dims)[1])*scale[1] + immin[1]

;; Determine frame rotation
M = determine_image_rotation(input, format)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
for ff=0,n_elements(files)-1 do begin
  ;; restore output file and extract useful packets
  ;; pts_sun is in solar reference frame with origin=Object center, units R_obj
  ;; vels_sun in km/s
  results_loadfile, files[ff], pts_sun, vels_sun, frac, /keepall
  radvel_sun = vels_sun[*,1] + stuff.vrplanet  ;; for g-value
     ;; note -- want to keep the ones with frac = 0 to make sure those regions
     ;; are counted as not contributing

  ;; Rotate the packets to observer frame
  pts_obs = M ## pts_sun    ;; observer along -y axis
  vels_obs = M ## vels_sun

  ;; Determine which packets are not blocked by the planet
  rhosqr_obs = pts_obs[*,0]^2 + pts_obs[*,2]^2  ;; rho in observer's frame
  inview = ((rhosqr_obs GT 1) or (pts_obs[*,1] LT 0))
  frac *= inview

  rhosqr_sun = pts_sun[*,0]^2 + pts_sun[*,2]^2
  out_of_shadow = ((rhosqr_sun GT 1) or (pts_sun[*,1] LT 0))

  ;; Determine which packets are in the FOV
  h = where((pts_obs[*,0] GE immin[0]) and (pts_obs[*,0] LE immax[0]) and $
    (pts_obs[*,2] GE immin[1]) and (pts_obs[*,2] LE immax[1]), nh)
  if (nh GT 0) then begin
    out = {x:ptr_new(pts_obs[h,0]), y:ptr_new(pts_obs[h,1]), z:ptr_new(pts_obs[h,2]), $
      frac:ptr_new(frac[h]), radvel_sun:ptr_new(radvel_sun[h])}

    ;; Packet weighting
    weight = results_packet_weighting(out, out_of_shadow)

    ;; Additional factors:
    case (format.quantity) of
      'column': weight /= Apix
      'intensity': weight /= Apix
      'density': weight /= Vpix
      else: stop
    endcase

    ;; Now make the image
    newh = where(weight GT 0, nh)
    if (nh GT 0) then begin
      qx = round(interpol(findgen((geometry.dims)[0]), xaxis, (*out.x)[newh]))
      qz = round(interpol(findgen((geometry.dims)[1]), zaxis, (*out.z)[newh]))
      for j=0,nh-1 do image[qx[j],qz[j]] += weight[newh[j]]
    endif
    ;tv, bytscl(image)
  endif ;; (nh GT 0)
  print, 'Completed image ' + strint(ff) + ' of ' + strint(n_elements(files))
endfor

result = {image:ptr_new(image), xaxis:ptr_new(xaxis), zaxis:ptr_new(zaxis), $
  format:format}
return, result

end
```