```
function determine_image_rotation, input, format

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; There are lots of ways to specify what the proper viewing geometry should be.
;; Need to break that down into three angles: rotations about the x, y, and z axis
;; in model coordinates.
;;
;; Tags that can be specified:
;;   a) SubObsLongitude, SubObsLatitude, PolarAngle
;;      * SubObsLong and Lat give the intersection point on the surface in
;;        longitude and latitude relative to the sub-solar point
;;      * On a planet, define the SubObsLongitude positive in the ccw direction when
;;        looking down from above.
;;        * long=0 -> looking down over sub-solar meridian
;;        * long=!pi/2 -> looking down over dusk meridian
;;        * long=!pi -> looking down over midnight meridian
;;        * long=3*!pi/2 -> looking down over dawn meridian
;;      * Latitude defined positive is north
;;        * lon = -!pi/2 -> looking down over south pole
;;        * lon = 0 -> looking down over equator
;;        * lon = !pi/2 -> looking down over north pole
;;   b) Observer (e.g. Earth, MESSENGER, ...), time
;;
;; Version History
;;   4.0: 25 Jan 2011
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

tags = strlowcase(tag_names(format.geometry))

q0 = total(strmatch(tags, 'SubObsLongitude', /fold))
w0 = total(strmatch(tags, 'SubObsLatitude', /fold))

q1 = total(strmatch(tags, 'observer', /fold))
w1 = total(strmatch(tags, 'time', /fold))

case (1) of
  (q0+w0 EQ 2): begin
    pSun = [0.,-1.,0.]
    pObs = [sin(format.geometry.SubObsLongitude)*cos(format.geometry.SubObsLatitude), $
      -cos(format.geometry.SubObsLongitude)*cos(format.geometry.SubObsLatitude), $
      sin(format.geometry.SubObsLatitude)]
    end
  (q1+w1 EQ 2): begin
    relative_position, 0., 0., 0., format.geometry.observer, input.geometry.planet, $
      frame='J2000', pos=pObs, havetime=utc2et(time)
    relative_position, 0., 0., 0., 'Sun', input.geometry.planet, frame='J2000', $
      pos=pSun, havetime=utc2et(time)
    end
endcase
```

```
52
53  M = rotationmatrix(pSun, pObs) ;; This is the rotation in space relative to the
54          ;; model coordinate system
55
56  ;; Determine rotation of FOV -- rotation about new y-axis
57  q = rotation([0,0,0], [0,1.,0], format.geometry.PolarAngle, R=R)
58  M = R # M
59
60  return, M
61
62  end
```