```
!Sa-TubeMod.f95      Auxiliary        Stereo-3D Simulation Environment Vsn:1.00
!2025.05.24.1840cdt- Lines drawn as tubes with hemispherical ends.
!                  - Herein My understanding of lighting & materials is very
!                     limited; any expert could improve on the code.

!            Author- Jeffrey M. Setterholm, Lakeville,MN 55044 USA
!         IP Status- Free source code (e.g.: post copyright)
!
!          Computer- "T3"/Dell Precision T3500/Intel i5 E5520/Win10Pro-21H2
!                     ^name ^Mfgr.Id           ^chipset       ^OS
!                    /Absoft Pro Fortran 21.0.2/GeForce GTX 1050/f90gl~Glut3.7
!                     ^compiler ~Fortran 95     ^graphics card   ^graphics

!    f90gl bindings- public domain; see "https://math.nist.gov/f90gl/"

!Disclaimer:
! ************************************************************
! **********   Individual cognition is always flawed,  **********
! **********         including yours and mine.         **********
! **********                 - So: -                   **********
! **********       Use this code at your own risk.     **********
! ************************************************************

!Table of Contents: ...use to search...
!Module TubeDef
!  type,public::TubeRec:Tu(:),TuIn,TuZero
!  type,public::LightMaterialRec  ;sequence
!  contains
!  Subroutine Lighting(iOffOn)
!  Subroutine LightShow
!  Subroutine Material
!  Subroutine Tubes(iDone,nPoint,pXyzh,Radius,iP)
!  Subroutine Sphere(SphRad,pXyz)
!End Module TubeDef

!////////////////////////////////////////////////////////////////////////7 9
Module TubeDef
!use TubeDef
!-------------------------------------------------------------------------
 type,public::TubeRec ;sequence                                  !2013.11.22
  integer(4)::n           !Number
  integer(4)::iSpare      !
  real(8)   ::C(4)        !Center
  real(8)   ::BangleD     !Bend angle (degrees)
  real(8)   ::iSphere     !>0:draw a sphere also... when BangelD >10. deg
  real(8)   ::Rdown       !Radius-(transverse to plane of bend, or of sphere)
  real(8)   ::Rout        !      -(in plane of bend) =Rdown/cosd(BangleD)
  real(8)   ::Vcln(1:5)   !Unit vector- C/L vector toward Tu(n+1).C- (5)= mag
  real(8)   ::Vdn( 1:5)   !           - down (transverse)          -
  real(8)   ::Von( 1:5)   !           - outward                    -
  real(8)   ::Ring(4,0:36)!Ring of points- around Tu(n).C
  real(8)   ::RingN(4,0:36)!             - vertex normals
 end type     TubeRec ;type(TubeRec),public,allocatable::Tu(:) !(0:LM%nTubes)
                       type(TubeRec),public            ::TuIn,TuZero

 !-------------------------------------------------------------------------
  !----- Light & material characterization:
 type,public::LightMaterialRec  ;sequence                        !2013.11.18
  integer(4):: nTubes !Number of allocated tube sections Tu(0:nTubes)

  real(4)   :: lit_Null(4)   = (/   .0,   .0,   .0,   .0 /)
 !-- Light#0
```

```
63     integer(4):: lit2_On       =1
64     real(4)   :: lit2_Amb(4)   = (/   .0,   .0,   .0,   .0 /)
65     real(4)   :: lit2_Dif(4)   = (/   .7,   .7,   .7,  1.  /)
66     real(4)   :: lit2_Spec(4)  = (/  1.0,  1.0,  1.0,  1.  /)
67     real(4)   :: lit2_Pos(4)   = (/ -3.0,  3.0, -3.0,  0.  /)
68
69    !-- Light#1
70     integer(4):: lit3_On       =1
71     real(4)   :: lit3_Amb(4)   = (/   .0,   .0,   .0,   .0 /)
72     real(4)   :: lit3_Dif(4)   = (/   .7,   .7,   .7,  1.  /) !<-- use S.StdColor
73     real(4)   :: lit3_Spec(4)  = (/  1.0,  1.0,  1.0,  1.  /)
74     real(4)   :: lit3_Pos(4)   = (/ -3. , -3.0, -3. ,  1.  /)
75
76    !-- Material#0
77     real(4)   :: mat0_AmbDif(4) = (/   .5,   .5 ,   .5 ,   .5  /)
78     real(4)   :: mat0_EmisF(4)  = (/  .1 ,   .1 ,   .1 ,   .0  /) !Outside
79    !real(4)   :: mat0_EmisF(4)  = (/  .0 ,   .0 ,   .0 ,   .0  /) !Outside
80     real(4)   :: mat0_EmisB(4)  = (/  .0 ,   .1 ,   .1 ,   .0  /) !Inside
81    !real(4)   :: mat0_Spec(4)   = (/  1.0,  1.0 ,  1.0 ,  1.0  /)
82     real(4)   :: mat0_Spec(4)   = (/   .5,   .5 ,   .5 ,   .5  /)
83     real(4)   :: mat0_Shin(1)   = (/ 40.  /)                      ![0.,128.]
84     real(4)   :: mat0_Null(4)   = (/   .0,   .0 ,   .0 ,   .0  /)
85    end type     LightMaterialRec ;type(LightMaterialRec),public::LM
86    !-------------------------------------------------------------------------------
87    contains
88    !-------------------------------------------------------------------------7 9
89
90    Subroutine Lighting(iOffOn)                                  !2013.11.18.0720
91    !2013.11.18.0720cst JMS- Creates two call lists (#1 & #2) to toggle lighting.
92    !                     - Supports viewer-anchored lighting.
93    !2025.04.14.1445cdt JMS- Using lights 2 & 3.
94     use OpenGLRec,only:  & !Ref: OpenGL GL/GLU/GLUT docs
95          glnewlist,GL_COMPILE,glDisable,GL_LIGHTING,GL_BLEND,glendlist   &
96         ,glShadeModel,GL_SMOOTH,glEnable,glLightModeli                   &
97         ,GL_LIGHT_MODEL_TWO_SIDE,GL_TRUE,GL_LIGHT_MODEL_LOCAL_VIEWER     &
98         ,GL_LIGHT0,glLightfv,GL_AMBIENT,GL_DIFFUSE,GL_SPECULAR           &
99         ,GL_POSITION,GL_DEPTH_TEST,GL_AUTO_NORMAL,glcalllist,glFlush     &
100        ,GL_LIGHT0,GL_LIGHT1,GL_LIGHT2,GL_LIGHT3
101   !use TubeDef, only:LM
102
103    implicit none                                                 !arguments
104     integer(4)::iOffOn
105     !--                                                          !internals
106     integer(4):: Init
107
108     !-- Initialization:
109      if(Init.eq.0) then
110
111       call glnewlist(1, GL_COMPILE)  !-- Turn off lighting
112         call glEnable( GL_LIGHTING)
113         call glDisable(GL_LIGHT2)
114         call glDisable(GL_LIGHT3)
115         call glLightModeli(GL_LIGHT_MODEL_TWO_SIDE    ,0)
116         call glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER,0)
117
118         call glDisable(GL_LIGHTING)
119       call glendlist !(1)
120
121       call glnewlist(2, GL_COMPILE)    !<-if ( LM%lit2_on > 0 )
122         !-- Draws the light locations as a small spheres:
123        !call LightShow
124         !-- Set the light color:
125         call glShadeModel(GL_SMOOTH)
```

```
125         call glShadeModel(GL_SMOOTH)
126         !-- Turn on lighting:
127         call glEnable(GL_LIGHTING   )
128         !-- Set the light color:
129        !call glColor4fv(LM%lit2_Dif)
130        !call glLightModeli(GL_LIGHT_MODEL_TWO_SIDE    ,GL_TRUE)
131        !call glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER,GL_TRUE) !Flawed.
132         call glLightModeli(GL_LIGHT_MODEL_TWO_SIDE    ,1)
133         call glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER,1)
134         !-- Turn on Light#0
135         call glEnable(GL_LIGHT2     )
136         call glLightfv(GL_LIGHT2,GL_AMBIENT  ,LM%lit2_Amb )
137         call glLightfv(GL_LIGHT2,GL_DIFFUSE  ,LM%lit2_Dif )
138         call glLightfv(GL_LIGHT2,GL_SPECULAR ,LM%lit2_Spec)
139         call glLightfv(GL_LIGHT2,GL_POSITION ,LM%lit2_Pos )
140         !-- Turn on Light#1
141         call glEnable(GL_LIGHT3     )
142         call glLightfv(GL_LIGHT3,GL_AMBIENT  ,LM%lit3_Amb )
143         call glLightfv(GL_LIGHT3,GL_DIFFUSE  ,LM%lit3_Dif )
144         call glLightfv(GL_LIGHT3,GL_SPECULAR ,LM%lit3_Spec)
145         call glLightfv(GL_LIGHT3,GL_POSITION ,LM%lit3_Pos )
146         call glEnable(GL_DEPTH_TEST )
147         call glEnable(GL_AUTO_NORMAL)
148
149        !-- if lighting is disabled, this colors it:
150        !call glColor4fv(LM%lit2_Dif)
151         call glFlush
152       call glendlist !(2)
153
154       Init=1
155     endif !Init=0
156
157    !-- Recurring processing:
158     if(iOffOn.le.0)    call glcalllist(1)  !Turns off lighting
159     if(iOffOn.ge.1)    call glcalllist(2)  !Turns on  lighting
160     return
161 End Subroutine Lighting
162 !-----------------------------------------------------------------------------7-9
163 Subroutine LightShow                                          !2013.11.15.0830
164 !2013.11.15.0830cst JMS- Shows where Lignt#0 & Light#1 are in 3D.
165 !                      - Traveler2/Athlon64/WinXPPro/APF9.0/Ogl1.2.1
166  use OpenGLRec,only:  & !Ref: OpenGL GL/GLU/GLUT docs
167       glEnable,GL_COLOR_MATERIAL,glColor4fv,glTranslatef,glutSolidSphere &
168      ,glDisable
169 !use TubeDef, only:LM
170
171  implicit none                                                !arguments
172   !--                                                         !internals
173    !---
174     if(LM%lit2_on.gt.0) then
175        !-- Set the light color:
176 !      call glEnable(GL_COLOR_MATERIAL) !This uses ColorStd()
177 !      call glColor4fv(LM%lit2_Dif)
178        !-- Draw the light location as a small sphere:
179        call glTranslatef( LM%lit2_Pos(1), LM%lit2_Pos(2), LM%lit2_Pos(3))
180        call glutSolidSphere(.1d0, 20, 20)
181        call glTranslatef(-LM%lit2_Pos(1),-LM%lit2_Pos(2),-LM%lit2_Pos(3))
182 !      call glDisable(GL_COLOR_MATERIAL)
183     endif !LM%lit2_on>0
184
185     if(LM%lit3_on.gt.0) then
186        !-- Set the light color:
```

```
!      call glEnable(GL_COLOR_MATERIAL) !This uses ColorStd()
!      call glColor4fv(LM%lit3_Dif)
       !-- Draw the light location as a small sphere:
       call glTranslatef( LM%lit3_Pos(1), LM%lit3_Pos(2), LM%lit3_Pos(3))
       call glutSolidSphere(.1d0, 20, 20)
       call glTranslatef(-LM%lit3_Pos(1),-LM%lit3_Pos(2),-LM%lit3_Pos(3))
!      call glDisable(GL_COLOR_MATERIAL)
    endif !LM%lit3_on>0
   return
End Subroutine LightShow
!-------------------------------------------------------------------------------7 9
Subroutine Material                                                  !2013.11.20.0640
!2013.11.20.0640cst JMS- Material colored defined by sub ColorStd's output.
!2013.11.18.0555cst JMS- !GLUT Fortran 90 program to draw red light sphere.
 use OpenGLRec,only:                                                          &
      glEnable,GL_DEPTH_TEST,GL_NORMALIZE,GL_AUTO_NORMAL,glShadeModel   &
     ,GL_SMOOTH,glMaterialfv,GL_FRONT,GL_AMBIENT,GL_DIFFUSE,GL_EMISSION &
     ,GL_BACK,GL_SPECULAR,GL_SHININESS,glFlush
!use TubeDef   ,only:LM
 use ScreenDef ,only:ColorRGBA
 implicit none                                                          !arguments
  !--                                                                   !internals
 !integer*4:: Init

  !-- Initialization:
  !-- Lighting must be enabled (done elsewhere):
      !call glEnable(GL_LIGHTING)
      !call Lighting(1)

   call glEnable(GL_DEPTH_TEST)
  !call glEnable(GL_CULL_FACE)
  !call glCullFace(GL_BACK)
   call glEnable(GL_NORMALIZE)
   call glEnable(GL_AUTO_NORMAL)
   call glShadeModel(GL_SMOOTH)

  !call glEnable(GL_COLOR_MATERIAL) !This uses glColor4fv/ColorStd()
  !call glColor4fv(LM%mat0_AmbDif)
 !!call glColorMaterial(GL_FRONT_AND_BACK, GL_DIFFUSE)
  !call glColorMaterial(GL_FRONT,GL_AMBIENT_AND_DIFFUSE)
  !call glDisable(GL_COLOR_MATERIAL)

   call glMaterialfv(GL_FRONT, GL_AMBIENT  , LM%mat0_Null  )
  !call glMaterialfv(GL_FRONT, GL_DIFFUSE  , LM%mat0_AmbDif)
   call glMaterialfv(GL_FRONT, GL_DIFFUSE  , ColorRGBA     ) !S.StdColor)
   call glMaterialfv(GL_FRONT, GL_EMISSION , LM%mat0_EmisF ) !Outside
   call glMaterialfv(GL_BACK , GL_EMISSION , LM%mat0_EmisB ) !Inside
   call glMaterialfv(GL_FRONT, GL_SPECULAR , LM%mat0_Spec  )
   call glMaterialfv(GL_FRONT, GL_SHININESS, LM%mat0_Shin  )

  !-- Blending can be turned on as follows:
      !call glenable(GL_BLEND)
      !call glblendfunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)

  !-- If lighting is turned on, surfaces should display as materials.
      !call glutSolidSphere(.5d0, 100, 100)
   call glFlush

  !-- Recurring processing:
  !call glTranslated( pXyz(1), pXyz(2), pXyz(3))
  !call glcalllist(3)
  !call glutSolidSphere(SphRad, 100, 100)
  !call glDisable(GL_COLOR_MATERIAL)
```

```
249    !call glDisable(GL_COLOR_MATERIAL)
250    !call glTranslated(-pXyz(1),-pXyz(2),-pXyz(3))
251
252     return
253  End Subroutine Material
254  !-------------------------------------------------------------------------7-9
255
256  Subroutine Tubes(iDone,nPoint,pXyzh,Radius,iP)
257  !2025.04.15.0720cdt JMS- Used by Sn3D for Dodecahedron visualization.
258  !2013.11.22.0820cst JMS- Generating a solid cylinder(tube) from its centerline.
259  !                       - nPoints: produce nPL[1:[-nPoint]] -> [0:[-nPoint]+1]
260
261  !Each tube segment is located and "bends" at its beginning end.
262   use OpenGLRec ,only:  & !Ref: OpenGL GL/GLU/GLUT docs
263        glDisable,GL_COLOR_MATERIAL,glEnable,GL_LIGHTING,glEnable   &
264       ,GL_DEPTH_TEST,GL_AUTO_NORMAL,glShadeModel,GL_SMOOTH,glbegin &
265       ,GL_QUAD_STRIP,glNormal3dv,glVertex4dv,glEnd,glFlush
266   use hVecMath8Mod                                                  !2013.10.23
267
268   implicit none                                                     !arguments
269    integer(4):: iDone        !<0:-nPMax;=0:next point;>0:nPmax to finish&display
270    integer(4):: nPoint       !Current point number [1:nPMax]
271    real(8)   :: pXyzh(4)
272    real(8)   :: Radius
273    integer(4):: iP
274    !--                                                              !internals
275    integer(4):: i,iBend,iFault,nP,nP2,nPMax
276    real(8)   :: Vdown(4)
277    real(8)   :: Temp1
278
279   !real(8)   :: R=.2d0,H=.4d0
280   !character::cW*120
281
282  10 select case(iDone) !////////////////////////////////////////////////////
283     !//////////
284      case(:-1) !Initialization & first point:
285       if(LM%nTubes.gt.0)      deallocate(Tu               )
286          LM%nTubes=iabs(iDone) ;allocate(Tu(0:LM%nTubes+1)) ;Tu=TuZero
287          nP   =0       ;iDone=0  ;nPMax=nP
288       nPoint=1                                                     ;goto 10
289     !//////////
290      case( 0 ) !Point accumulation:
291          nP=nP+1                  ;nPMax=nP
292       if(nP.ne.nPoint) call jPause("Tubes: nPoint #'s not sequential.")
293       Tu(nP)      = TuZero
294       Tu(nP)%n    = nPoint
295       Tu(nP)%C    = pXyzh
296       Tu(nP)%Rdown= Radius
297       Vdown=Tu(nP)%C
298       call hVnorm(Vdown,Vdown,Temp1)
299       if(nP.eq.LM%nTubes) then; iDone=1 ; goto 10 ;endif !nP=LM%nTubes
300       nPoint=nPoint+1
301     !//////////
302      case( 1 )  !Process points for display:
303      !-- Compute the interpoint spacings:
304       do nP=2,nPMax
305         call hSubtract(Tu(nP-1)%Vcln,Tu(nP  )%C,Tu(nP-1)%C)
306         call hVnorm(   Tu(nP-1)%Vcln,Tu(nP-1)%Vcln,Tu(nP-1)%Vcln(5))
307       enddo!nP
308      !-- Use the first point with non-zero spacing to back-out the 0th point:
309       Tu(0)=Tu(1)                                    ;iFault=1
310       do nP=1,nPMax-1
```

```
311         if(Tu(nP)%Vcln(5).gt.0.d0) then
312            Tu(0)%C(1:3)=Tu(0)%C(1:3)-Tu(nP)%Vcln(1:3);iFault=0
313            exit
314         endif !Tu(nP).Vcln(0)>0.
315       enddo!nP
316       if(iFault.gt.0) then          ![Note: ...could draw a sphere instead.]
317         call jPause("Tubes: pointset has no extent. Hence`ends`are undef.")
318        !stop             "Tubes: pointset has no extent. Hence`ends`are undef."
319         return !For debugging only
320       endif !iFault>0
321      !-- Use the last point with non-zero spacing to extend the nPMax+1 point:
322       Tu(nPMax+1)=Tu(nPMax)
323       do nP=nPMax-1,1,-1
324         if(Tu(nP)%Vcln(5).gt.0.d0) then
325            Tu(nPmax+1)%C(1:3)=Tu(nPmax+1)%C(1:3)+Tu(nP)%Vcln(1:3)
326           exit
327         endif !Tu(nP)%Vcln(0)>0.
328       enddo!nP
329      !-- Complete computing the interpoint spacings:
330       do nP=1,nPMax+1 ;if((nP.gt.1).and.(nP.lt.nPMax)) cycle
331         call hSubtract(Tu(nP-1)%Vcln,Tu(nP  )%C,Tu(nP-1)%C)
332         call hVnorm(   Tu(nP-1)%Vcln,Tu(nP-1)%Vcln,Tu(nP-1)%Vcln(5))
333       enddo!nP
334      !-- Define the tube bends:
335                                                        iBend=0
336     do nP=1,nPMax
337       if(Tu(nP-1)%Vcln(5).eq.0.d0) cycle
338       if(Tu(nP  )%Vcln(5).eq.0.d0) cycle
339       call hDot(Tu(nP-1)%Vcln,Tu(nP)%Vcln,Tu(nP)%BangleD)
340         if(Tu(nP)%BangleD.gt. 1.d0) Tu(nP)%BangleD= 1.d0
341         if(Tu(nP)%BangleD.lt.-1.d0) Tu(nP)%BangleD=-1.d0
342       Tu(nP)%BangleD=dacosd(Tu(nP)%BangleD)              ![0.,180.]
343       if(Tu(nP)%BangleD.lt.1.d-4) Tu(nP)%BangleD=0.d0
344       if(Tu(nP)%BangleD.gt.0.d0 ) then;                    iBend=iBend+1
345         call hCross( Tu(nP  )%Vdn, Tu(nP-1)%Vcln, Tu(nP  )%Vcln  )
346         call hVnorm( Tu(nP  )%Vdn, Tu(nP  )%Vdn , Tu(nP  )%Vdn(5))
347                                                    Tu(nP  )%Vdn(5) =1.d0
348         call hCross( Tu(nP  )%Von, Tu(nP  )%Vcln, Tu(nP  )%Vdn   )
349         call hCross( Tu(nP+1)%Von, Tu(nP-1)%Vcln, Tu(nP  )%Vdn   )
350         call   hAdd( Tu(nP  )%Von, Tu(nP  )%Von , Tu(nP+1)%Von   )
351         call hVnorm( Tu(nP  )%Von, Tu(nP  )%Von , Tu(nP  )%Von(5))
352                                                    Tu(nP  )%Von(5) =1.d0
353        !-- Define default "down" vectors for straight segments
354         do nP2=nP-1,0,-1 ;if(Tu(nP2)%BangleD.gt.0.d0) exit
355                            Tu(nP2)%Vdn=Tu(nP)%Vdn      ;enddo!nP2
356         do nP2=nP+1,nPMax+1 ;Tu(nP2)%Vdn=Tu(nP)%Vdn    ;enddo!nP2
357       endif !BangleD>0.
358      !write(cW,"('nP=',i4,' Bangle=',f8.3,5f10.3 )") &
359      !            nP     ,  Tu(nP)%BangleD ,Tu(nP)%Vcln     ;call WriteSF(cW, 6)
360     enddo!nP
361    !-- If there are no bends (the tube is straight,e.g. 1 segment) assign "down"
362     if(iBend.eq.0) then                              ! X    Y    Z    h    Mag
363       if(dabs(Tu(0)%Vcln(3)).lt..9d0) then; vDown=(/0.d0,0.d0,1.d0,1.d0/)
364                                      else; vDown=(/1.d0,0.d0,0.d0,1.d0/)
365       endif !|Tu(0).Vcln(3)|<.9
366       call hCross( Tu(0)%Von   , Tu(0)%Vcln ,       vDown   )
367       call hVnorm( Tu(0)%Von   , Tu(0)%Von  , Tu(0)%Von( 5) )
368       call hCross( Tu(0)%Vdn   , Tu(0)%Von  , Tu(0)%Vcln    )
369                    Tu(0)%Vdn(5)=1.d0
370       do nP=1,nPMax+1            ;Tu(nP)%Vdn=Tu(0)%Vdn        ;enddo!nP
371     endif !iBend=0
372
373    !-- Compute the remaining Tu() Von's:
```

```
373   !-- Compute the remaining Tu().Von's:
374                  !Tu(nPMax ).Vcln(1:4)=Tu(nPMax-1)%Vcln(1:4)
375    do nP=0,nPMax    ;if(Tu(nP  )%BangleD.gt.0.d0) cycle
376      call hCross( Tu(nP)%Von, Tu(nP)%Vcln, Tu(nP)%Vdn     )
377      call hVnorm( Tu(nP)%Von, Tu(nP)%Von , Tu(nP)%Von( 5) )
378    enddo!nP
379   !-- Create the ring of points at point nP:
380    do nP=0,nPMax
381      if(Tu(nP)%BangleD.le.0.d0) then
382             Tu(nP)%Rout=Tu(nP)%Rdown
383       else; Tu(nP)%Rout=Tu(nP)%Rdown/dcosd(Tu(nP)%BangleD/2.d0)
384      endif !Tu(nP).BangleD<=0.
385      do i=0,36
386        Tu(nP)%Ring(1:3,i)=  Tu(nP)%Rdown*Tu(nP)%Vdn(1:3)*dcosd(dble(i*10)) &
387                           +Tu(nP)%Rout *Tu(nP)%Von(1:3)*dsind(dble(i*10))
388        Tu(nP)%Ring( 4 ,i)=1.d0  !<-Without this, the object was a no-show!
389        call hVnorm(Tu(nP)%Ring(1,i) ,Tu(nP)%RingN(1,i) , Temp1 )
390                                     Tu(nP)%RingN(4,i)=1.d0
391        Tu(nP)%Ring(1:3,i)=Tu(nP)%Ring(1:3,i)+Tu(nP)%C(1:3)
392      enddo!i
393    enddo!nP
394    iDone=nPMax ;goto 10
395    !//////////
396     case(2: )  !Draw the tubes:
397    call glEnable(GL_LIGHTING)
398    call lighting(1)                                                  !20250414
399    call LightShow
400   !-----
401    call glEnable(GL_DEPTH_TEST)
402    call glEnable(GL_AUTO_NORMAL)
403    call glShadeModel(GL_SMOOTH)
404
405    do nP=1,nPMax-1
406      call glbegin(GL_QUAD_STRIP)
407      do i=0,36
408        call glNormal3dv(Tu(nP+1)%RingN(1:3,i))
409        call glVertex4dv(Tu(nP+1)%Ring( 1:4,i))
410        call glNormal3dv(Tu(nP  )%RingN(1:3,i))
411        call glVertex4dv(Tu(nP  )%Ring( 1:4,i))
412      enddo!i
413      call glEnd
414      call glFlush
415    enddo!nP
416    call lighting(0)                                                  !20250414
417
418   !-----
419    end select !iDone ////////////////////////////////////////////////////////
420    return
421 End Subroutine Tubes
422 !-------------------------------------------------------------------------------7-9
423
424 Subroutine Sphere(SphRad,pXyz)
425 !2025.04.18.1605cdt JMS- !GLUT Fortran 90 program to draw a sphere.
426
427  use OpenGLRec ,only:                                                 &
428       glDisable,GL_COLOR_MATERIAL,glEnable,GL_LIGHTING,glEnable       &
429      ,GL_AUTO_NORMAL,glShadeModel,GL_SMOOTH,glTranslated,glutSolidSphere
430
431  implicit none                                                    !arguments
432   real(8)   ::SphRad
433   real(8)   ::pXyz(4)
434   !--                                                             !internals
```

```
435   real(8)   ::H=.1d0
436   real(8)   ::R=.25d0
437
438   !-- Initialization:
439    call glDisable(GL_COLOR_MATERIAL)
440    call glEnable(GL_LIGHTING)
441    call lighting(1)                                                    !20250414
442
443    call Material
444    call glEnable(GL_AUTO_NORMAL)
445    call glShadeModel(GL_SMOOTH)
446   !-- Recurring processing:
447    call glTranslated( pXyz(1), pXyz(2), pXyz(3))
448    call glutSolidSphere(SphRad, 20, 20)
449
450    call lighting(0)                                                    !20250414
451    call glTranslated(-pXyz(1),-pXyz(2),-pXyz(3))
452
453    return
454 End Subroutine Sphere
455 !-------------------------------------------------------------------------7-9
456 End Module TubeDef
457 !////////////////////////////////////////////////////////////////////////7 9
458
459
```