```
!S7Motion7.F95       Group ID: #7
!2025.05.24.1840cdt- Modelview Matrix control using the mouse
!                  - 7DoF Point-of-Interest(PoI)
!                  - Standard Flight Simulation (FS) coordinates, i.e.:
!                       +X:forward, +y:right, +Z:down

!            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...
! Subroutine Motion7(nM7,iEye,iP)
! Subroutine hPoIGen(hPoI,XyzRpyM,Mode,iP)
! Subroutine hPoIDecode(hPoI,PoI7,iP)

! Subroutine hFS7Gen(hFS7,XyzRpyM,iP)
! Subroutine hFS7Decode(hFS7,FS7,iP)

! Subroutine DCfromRPY(DC,RPY,iP)
! Subroutine OddEven7DoF(ioe,ioein,ioeu,DoFsInUse,iP)

! Subroutine PackM7u
! Subroutine UnPackM7u

! Subroutine CubeGrid(nCol)
! Subroutine WirePlane(Sizeu,nCol)
! Subroutine VecFont7(XYZRPYwC,LineWidth,iCol,Label)

!--Quaternions:
! Subroutine QFromRpy(Q,Rpyh)
! Subroutine QoutFromQoQi(Qout,Qo,Qi)
! Subroutine QtoRpy(Q,Rpy)
! Subroutine QToQinverse(Q,Qinverse)
! Subroutine QtoVhatAngle(Q,Vhat,Angle,iP)      <-"JeffTernions"

!-------------------------------------------------------------------------------7-9

Subroutine Motion7(nM7,iEye,iPin)
!2020.06.30.0805cdt JMS- Updates Motion 7DoF's.
!////////////////////////////////////////////
!--Globals
 use ioDef       ,only: Us,Ut,UpVerb,NowView              !Files,Units,etc.
 use ScreenDef   ,only: nCharMaxY,nCharMaxXS,Side     &   !screen & colors
                       ,lCharX,lCharY,nCharMaxX,nCharCenX
 use KeyboardDef ,only: KbdKey,ArrowKey,NumLockOn         !Keyboard
 use MouseDef    ,only: MChan,MouseDirect,MXY,MXYdel,iMouse7DoF
 use ModelDef    ,only: c7DoFID,iColors,cDoFID         &  !Modelview matrix
       ,M7,M7zero,hPoIRotate,M7u,Motion7Rec            &
```

```
63          ,DoFL,H,DoFreset,Sens,nM7L,ID,iType,Locked,iec,ioe,DoFsInUse,M7Init
64     use simdef      ,only: IterTotal
65     use ViewDef     ,only: Key
66
67    !--End Globals
68     implicit none
69     !--Arguments- ^global variables in S1ModDef.f95:
70      integer(4)::nM7          !Index of the 7DoF object being controlled
71      integer(4)::iEye
72      integer(4)::iPin         ! >5: Printing enable, use: write(iP,...)
73     !--Internals
74      integer(4)::iP           ! >5: Printing enable, use: write(iP,...)
75      real(8)   ::DoF(7,0:3)   !(/ X,Y,Z,Roll,Pitch,Yaw,Mag /)
76      integer(4)::ioein(0:1)   !Axis change: Odd-or-even & DoF#
77      integer(4)::ioeu         !=ioe(ioe(0))
78      integer(4)::ie
79      real(8)   ::Mag
80      integer(4)::Ku,Au,i,j,i7u,IntExtMod(2),iDelta,iDeltaL,iUnlock(0:7)=1
81      integer(4)::iRepeat
82      !Screen reporting:
83      integer(4)::nRow,nCol,nCol2,iColor,McharXY(2),jPrev
84      character(len=80):: PText
85      !--EndDefs-------------------------------------------------
86       iP=iPin !; if(UpVerb==0) iP=0
87       if(iP>5) write(iP,"('Motion7:  7DoF',36('\'))")
88       M7u = M7(nM7); call UnPackM7u; ie=iec
89       if(M7Init==0) then
90        !--Initialize DoF's & Transform:
91         M7u                  = M7zero
92         nM7L                 = nM7
93         if(nM7L==0) iType =0 !Point-of-Interest control: <-Translate<-Rotate
94         if(nM7L> 0) iType =1 !Flight Simulation control: <-Rotate<-Translate
95         write(ID,"('M7(',i3,')')") nM7
96         DoFreset             =(/0.d0,0.d0,0.d0,0.d0,0.d0, 0.d0,+1.d0/)
97         DoFL                 = DoFReset
98         if(iType==0) call hPoIGen(H,DoFL,0,0)
99         if(iType==1) call hFS7Gen(H,DoFL,0,0)
100        Sens                 =(/.05d0,.05d0,.05d0,1.0d0,1.0d0,1.0d0,.01d0/)
101
102       !--Initialize control axes:
103        ioe            = (/2,5,6/)
104        MChan(3)%i7DoF = ioe(2) !Mouse channel assignment- lateral
105        MChan(4)%i7DoF = ioe(1) !                        - vertical
106        DoFsInUse      = (/0,0,0,0,0,2,1,0/)
107        Locked         =   0        !unlocked
108        ie             =   1        !interior control
109        M7Init         =   1
110        iec=ie; call PackM7u; M7(nM7) = M7u
111      endif! M7Init=0
112      if(iEye>0)                                                    goto 100
113     !--Each iteration's initialization:
114      DoF        = 0.d0
115      DoF(1:7,1) = DoFL
116      MChan(4)%i7DoF = ioe(1)
117      MChan(3)%i7DoF = ioe(2)
118
119      ioein      = 0
120      Mag        = DoF(7,1)
121      if (Mag<.000001d0) Mag=.001d0
122     !--Save the incoming 7DoF state (for locking):
123      DoF(1:7,0)=DoF(1:7,1)
124     !--exterior inputs, if any, are incremental:
125      DoF(1:7,2)=(/0.d0,0.d0,0.d0,0.d0,0.d0,0.d0,1.d0/)
```

```
125      DoF(1:7,2)=(/0.d0,0.d0,0.d0,0.d0,0.d0,0.d0,1.d0/)
126     !-------------------------------------------------
127     !ASCII NumLocked Key ID's:                    operators
128     !       /     *     -   |     k47  k42  k45      * k42
129     !  7    8     9     +   |k55  k56  k57  K43      + k43
130     !  4    5     6     "   |k52  k53  k54   "       - k45
131     !  1    2     3   Enter|k49  k50  k51  K13      / k47
132     !  0-----     .     "   |k48-----  k46   "
133     !-------------------------------------------------
134                   IntExtMod=0;  iDelta=0
135     !--Numeric keypad DoF selection:
136                   ku=kbdkey;KbdKey=0; ioeu=ioe(ioe(0))
137      select case(Ku)
138       case(46); Locked(ioeu)=1-Locked(ioeu) !'.' Lock
139       case(48); if(  ioeu >0) then          !'0' zero ~ reset one
140                   DoF(ioeu,1) = DoFreset(ioeu)
141                   DoF(ioeu,0) = DoF(ioeu,1)
142                   IntExtMod(1)=1
143                 endif!ioeu>0
144       case(49,51,53,55)          !Odds: '1':X, '3':Z, '5':Pitch, '7':Mag
145        ioein(0) = 1
146        ioein(1) = Ku-48
147        call OddEven7DoF(ioe,ioein,ioeu,DoFsInUse,0)
148        MChan(4)%i7DoF = ioe(1)
149        MChan(3)%i7DoF = ioe(2)
150        NumLockOn=1
151       case(50,52,54)             !Evens: '2':Y, '4':Roll, '6':Yaw
152        ioein(0) = 2
153        ioein(1) = Ku-48
154        call OddEven7DoF(ioe,ioein,ioeu,DoFsInUse,0)
155        MChan(4)%i7DoF = ioe(1)
156        MChan(3)%i7DoF = ioe(2)
157        NumLockOn=1
158       case(56)                   !'8':Reset modelview 7DoF entirely:
159        DoF(1:7,1) = DoFreset
160        IntExtMod(1)= 1
161        if(Ip>5) write(Ip,"('Reset all: X,Y,Z,Roll,Pitch,Yaw,& Mag:')")
162       case(57); ie=3-ie          !'9':INTERIOR|exterior flag:
163       case default; KbdKey=ku    !Keypresses ignored, e.g.:non-numeric:
164      end select !ku
165      i7u=ioeu
166
167      !--ArrowKey manual 7DoF control:
168                   Au=ArrowKey; ArrowKey=0
169      select case(Au)
170      !--Arrowkeys- lateral:
171       case(100); IntExtMod(ie)=1;  iDelta=-1 !Arrow-left:
172        i7u=MChan(3)%i7DoF
173       case(102); IntExtMod(ie)=1;  iDelta=+1 !Arrow-right:
174        i7u=MChan(3)%i7DoF
175      !--Arrowkeys- into:
176       case(101); IntExtMod(ie)=1;  iDelta=+1 !Arrow-forward:
177        i7u=MChan(4)%i7DoF
178       case(103); IntExtMod(ie)=1;  iDelta=-1 !Arrow-back:
179        i7u=MChan(4)%i7DoF
180      !--NumLock off warning:
181       case(104:108); write(6,"(a1)") char(7) !; ArrowKey=Au                  !Beep
182        NumLockOn=0
183      !--ArrowKeys ignored, e.g.:not arrows :
184       case default; ArrowKey=Au
185      end select!(Au)
186
```

```
187     iRepeat=0
188  50 continue
189    !--Increment/Decrement individual Point-of-Interest values:
190     if(iDelta/=0) then
191       select case(i7u)
192        case( 0 ) !
193        case(1:3) !X,Y,Z:
194         iDeltaL=iDelta; if(i7u==3) iDeltaL=-iDelta
195         if(ie==1) DoF(i7u,ie)=DoF(i7u,ie)+.05d0*iDeltaL/Mag
196         if(ie==2) DoF(i7u,ie)=DoF(i7u,ie)+.05d0*iDeltaL
197        case(4:6 ) !Roll.Pitch,Yaw:
198         DoF(i7u,ie)=DoF(i7u,ie)+1.0d0*iDelta
199         if(DoF(i7u,ie)>   180.d0) DoF(i7u,ie) = DoF(i7u,ie)-360.d0
200         if(DoF(i7u,ie)<= -180.d0) DoF(i7u,ie) = DoF(i7u,ie)+360.d0
201        case( 7 ) !Mag
202         if(abs(iDelta)>5) iDelta=sign(5,iDelta)
203         if(iDelta>0) DoF(i7u,ie) = DoF(i7u,ie)*1.010d0     **abs(iDelta)
204         if(iDelta<0) DoF(i7u,ie) = DoF(i7u,ie)*0.99009901d0**abs(iDelta)
205         if(dabs(     DoF(i7u,ie)  -1.d0) < +.005d0) &
206                      DoF(i7u,ie) = 1.d0
207        !--Bound magnitude:
208         if(          DoF(i7u,ie) <   0.001d0) &
209                      DoF(i7u,ie) =   0.001d0
210         if(          DoF(i7u,ie) > 1000.0d0 ) &
211                      DoF(i7u,ie) = 1000.0d0
212        case default; write(6,"('i7u=',i2,' ie=',i1,': out of range @L143.')") &
213                                  i7u     ,  ie
214       end select! i7
215     endif!iDelta/=0
216
217    !--LMB-pressed + Mouse motion manual 7DoF control:
218     if(MXYDel(3)/=2) then; iRepeat        =3 !LMB- not pressed
219      else;                 iRepeat=iRepeat+1 !   -     pressed
220       select case(iRepeat)
221        case(1) !Mouse X-axis:
222         i7u=MChan(3)%i7DoF;   iDelta = MXYDel(1)
223         if(iDelta/=0) IntExtMod(ie)=1; MXYDel(1)= 0
224         goto 50
225        case(2) !Mouse Y-axis:
226         i7u=MChan(4)%i7DoF;   iDelta =-MXYDel(2)
227         if(iDelta/=0) IntExtMod(ie)=1; MXYDel(2)= 0
228         goto 50
229       end select !(iRepeat)
230     endif !MXYDel(3)/=2
231
232     if(MXY(3,1)==2) then !Left mouse button- pressed:
233      !--LMB-pressed - check for mouse-screen-based DoF changes:
234       nRow  = nCharMaxY-17; nCol = nCharMaxXS(Side)-33
235       McharXY(1)=MXY(1,1)/lCharX
236      !--Interpret mouse X location in the split screen, if necessary:
237       if((Key%SplitScreen/=0).and.(McharXY(1)>nCharCenX)) &
238       McharXY(1)=McharXY(1)-(nCharCenX-1)
239       McharXY(1)=McharXY(1)-nCol
240       McharXY(2)=MXY(2,1)/lCharY-nRow; j=McharXY(2)-2
241       if(j/=jPrev) then
242         select case(j)
243          case(-1) !Change the controlled object
244           if(McharXY(1)>0) then
245             iMouse7Dof=1-iMouse7Dof                                    ;jPrev=j
246           endif!McharXY(1)>0
247          case(1:7)
248           select case(McharXY(1))
```

```
249             case(16 ); if(IterTotal-MXY(4,1)<2) Locked(j)=1-Locked(j) ;jPrev=j
250             case(18:28)                                                ;jPrev=j
251  !           if(Ut>5) write(Ut,"('LMB DoF update:',4i5)") nM7,j,McharXY
252              select case(j)
253               case(1,3,5,7); ioein = (/1,j/)
254               case(2,4,6  ); ioein = (/2,j/)
255              end select!(J)
256              call OddEven7DoF(ioe,ioein,ioeu,DoFsInUse,0)
257              MChan(4)%i7DoF= ioe(1)
258              MChan(3)%i7DoF= ioe(2)
259            end select!(MpiXY(1))
260          end select!j
261        endif!j/=jPrev
262       else; jPrev=0
263      endif!MXY(3,1)==2
264
265      if(nM7>0) then   !For Flight Simulation 6DoF:
266        Locked(7)   =1 !  No magnification
267        ie          =1 !  & internal control only.
268        IntExtMod(2)=0
269      endif!nM7>0
270
271      i7u=ioeu
272      if((IntExtMod(1)/=0).or.((iType==0).and.(IntExtMod(2)/=0)) ) then
273       !--Check for locked DoF's
274        Locked(0)=0; do i=1,7; if(Locked(i)>0) Locked(0)=1; enddo!i
275       !--Point-of-Interest transform update:
276       !--Update ModelView's interior 7DoF Transform:
277        if(IntExtMod(1)>0) then
278          do i=1,7; if(Locked(i)==1) DoF(i,1)=DoF(i,0); enddo!i
279  !       if(iP>5) write(iP,"('INTERIOR 7DoF transform update:' )")
280          if(iType==0)call hPoIGen(H,DoF(1,1),0,iP)
281          if(iType==1)call hFS7Gen(H,DoF(1,1),iP)
282        endif!IntExtMod(1)>0
283        if(IntExtMod(2)>0) then
284       !--Update ModelView's exterior 7DoF Transform:
285          DoF(1:7,1)=DoF(1:7,2)
286          if(iType==0)call hPoIGen(H,DoF(1,1),1,iP)
287          if(Locked(0)>0) then
288            do i=1,7; if(Locked(i)==1) DoF(i,1)=DoF(i,0); enddo!i
289            if(iP>5) write(iP,"('& with locked state(s):' )")
290          if(iType==0)call hPoIGen(H,DoF(1,1),0,iP)
291          endif!Locked(0)>0
292        endif!IntExtMod()>0                    !Use DoF(_,1) for screen reporting.
293    !--Update the PoI rotation-only transform:
294       if(iType==0) then
295         DoF(1:7,3) =(/    0.d0      ,   0.d0      ,   0.d0      &
296                          ,DoF(4,1),DoF(5,1),DoF(6,1) &
297                          ,   1.d0                                 /)
298         if(iP>5) write(iP,"('PoI rotation only:' )")
299         call hPoIGen(hPoIRotate,DoF(1,3),0,iP)
300       endif!iType==0
301      else
302       if(iP>5) then
303         write(iP,"('M7(',i3,') transform:' )") nM7
304         if(iType==0)call hPoIGen(H,DoF(1,1),0,iP)
305         if(iType==1)call hFS7Gen(H,DoF(1,1),iP)
306        else
307         if(iType==0)call hPoIGen(H,DoF(1,1),0,0)   !WAG add 2021.03.08
308         if(iType==1)call hFS7Gen(H,DoF(1,1),0)     ! "   "       "
309       endif!iP>1\5
310      endif! IntExtMod(1)/=0 or IntExtMod(2)/=0
```

```
311
312     !--Identify the controlled 7DoF's:
313      MChan(3)%ID = cDoFID(ioe(2))
314      MChan(4)%ID = cDoFID(ioe(1))
315      DoFL=DoF(1:7,1); iec=ie; call PackM7u; M7(nM7) = M7u
316
317   100 continue
318       if(NowView/=1)                                                    return
319     !--On-screen report:
320                             nRow =nCharMaxY-16;          nCol=nCharMaxXS(Side)-33
321     !write(Ptext,"('Units=    ',a12)") S.cFrustumUnits           !LGreen
322        !                     nRow=nRow+1; call PrntOrtho(nRow,nCol,10, 0,PText)
323
324      do i=1,15; PText=char(0); iColor=iColors(i)
325        select case(i)
326         case(1:2,10,12:15) !7DoF generic info:
327                           write(Ptext,"(a32)") c7DoFID(i)
328          if(i==1) then
329            if(iType==0) write(Ptext,"('Seven DoF Point-of-Interest Ctl:')")
330            if(iType >0) write(Ptext,"('Seven DoF Object#',i3,' 6DoF Ctl: ')") nM7
331          endif;             nRow=nRow+1; call PrntOrtho(nRow,nCol,iColor,0,PText)
332         case(3:9); j=i-2   !7DoF states info:
333          if(j==ioe(1)) iColor= 1
334          if(j==ioe(2)) iColor= 1
335          write(Ptext,"(a32)") c7DoFID(i)
336                             nRow=nRow+1; call PrntOrtho(nRow,nCol,iColor,0,PText)
337          write(PText,"(f16.6)") DoFL(j)
338                                          call PrntOrtho(nRow,nCol,iColor,0,PText)
339          PText="L"                                        !White
340          if(Locked(j)>0)                 call PrntOrtho(nRow,nCol+17, 1 ,0,PText)
341          if(iColor==1) then
342            if(j==ioe(ioe(0))) then;  PText="1"
343                               else;  PText="2"
344            endif!j==ioe(io1e2)
345                           nCol2=nCol+27
346            if(mod(j,2)==1) nCol2=nCol+29
347                                          call PrntOrtho(nRow,nCol2,iColor,0,PText)
348          endif!iColor==1
349         case(11)            !Numlock& INT|EXT
350          if(NumLockOn==0) then; PText="NumLock?" ;                iColor=7!Red
351           else;                 PText="NumLocked";                iColor=3!LLWhit
352          endif;              nRow=nRow+1;;call PrntOrtho(nRow,nCol,iColor,0,PText)
353          if(ie==1) Ptext="INTERIOR control"
354          if(ie==2) Ptext="EXTERIOR control"                         !White
355                                          call PrntOrtho(nRow,nCol+15, 1 ,0,PText)
356        end select!(i)
357       enddo!i
358       if(Ip>5) write(Ip,"(14x,'Display on-screen M7(',i3,') 7DoF controls.'))")nM7
359       if(Ip>5) write(Ip,"('End Motion7:  ',36('/'))")                      ;return
360   End Subroutine Motion7
361   !-------------------------------------------------------------------------7-9
362
363   Subroutine hPoIGen(hPoI,XyzRpyM,Mode,iP)
364   !2020.06.18.1210cdt JMS- Generates the Point-of-Interest control matrix
365   !                       in the INTERIOR or the exterior concatenated form.
366    implicit none
367    !--Arguments:
368     real(8)    ::hPoI(4,4)  !<-- Rotate+Magnify <-- Translate <--
369     real(8)    ::XyzRpyM(7) !7DoF rigid body motion & magnification:
370    !                        !X    Y    Z  Roll Pitch Yaw  Mag
371    !                        !1    2    3    4    5    6    7
372     integer(4)::Mode        !=0: no incoming matrix - defining the INTERIOR matrix.
```

```
373                          !=1: hPoIin defined; concat. the exterior matrix.
374    integer(4)::iP         !>5: Printing enable, use: write(iP,...)
375   !--Internals
376    real(8)   ::V(7)       !internal copy of XyzRpyM(7)
377    real(8)   ::Hin(4,4)   !internal copy of hPoI input
378    real(8)   ::H(4,4),Hout(4,4)
379    real(8)   ::DC(3,3)                  !Direction cosines of (Roll,Pitch,Yaw)
380    real(8)   ::XYZ(3)                   !rotated translations
381    integer(4)::i,j
382    real(8)   ::XyzRpyML(7)
383    !--EndDefs-------------------------------------------------
384     V   = XyzRpyM
385     Hin = hPoI !Used by the exterior PoI computation
386    !--Compute the direction cosines:
387     call DCfromRPY(DC,V(4:6),iP)
388    !--Rotate(X,Y,Z)
389     do i=1,3; XYZ(i)=DC(i,1)*V(1)+DC(i,2)*V(2)+DC(i,3)*V(3); enddo!i
390    !--Compute the PoI transform of XyzRpyM:
391     H(1,1:4) = (/ DC(1,1), DC(1,2), DC(1,3),   XYZ(1) /)
392     H(2,1:4) = (/ DC(2,1), DC(2,2), DC(2,3),   XYZ(2) /)
393     H(3,1:4) = (/ DC(3,1), DC(3,2), DC(3,3),   XYZ(3) /)
394     H(4,1:4) = (/  0.d0   ,  0.d0   ,  0.d0   ,+1.d0/V(7)/)
395     select case(Mode)
396      case(0) !INTERIOR PoI:
397       if(iP>5) then !FYI printout:
398         write(iP,"('hPoIGen: INTERIOR.  XyzRpyM=')")
399         write(iP,"(7f10.4,'     :DoF')") V
400         write(iP,"( '          yields hPoI:')")
401         write(iP,"( 4f20.9)") ((H(i,j),j=1,4),i=1,4)
402       endif!iP>5
403       hPoI = H                                                          ;return
404
405      case(1) !Concatenate the exterior PoI:
406       do i=1,4; do j=1,4; Hout(i,j)=  H(i,1)*Hin(1,j) &
407                                     + H(i,2)*Hin(2,j) &
408                                     + H(i,3)*Hin(3,j) &
409                                     + H(i,4)*Hin(4,j)   ;enddo; enddo!i
410       hPoI   = Hout
411       if(iP>5) then !FYI printout:
412         write(iP,"('hPoIGen: EXTERIOR. Incoming XyzRpyM= :')")
413         write(iP,"(7f10.4,'     :DoF')") V
414         write(iP,"('     & incoming hPoI:')")
415         write(iP,"(4f20.9)") ((Hin(i,j),j=1,4),i=1,4)
416         call hPoIDecode(Hin,XyzRpyML,iP)
417         write(iP,"( '       exterior tweak:')")
418         write(iP,"( 4f20.9)") ((H(i,j),j=1,4),i=1,4)
419         write(iP,"( '       Resulting hPoI:')")
420         write(iP,"( 4f20.9)") ((Hout(i,j),j=1,4),i=1,4)
421         write(iP,"('  & DoF XyzRpyM=')")
422       endif!iP>5
423       call hPoIDecode(Hout,XyzRpyM,iP)                                  ;return
424      !--Decode hPoI's 7DoF states:
425     end select!Mode
426   End Subroutine hPoIGen
427   !-------------------------------------------------------------------------------7-9
428
429   Subroutine hPoIDecode(hPoI,PoI7,iP)
430   !2020.06.10.0500cdt JMS- !Decodes hPoI7's 7DoF states.
431    implicit none
432    !--Arguments:
433     real(8)   ::hPoI(4,4) !Point-of-Interest- homogeneous transform
434     real(8)   ::PoI7(7)   !                  - seven degrees of freedom:
```

```
435                          !                   (X,Y,Z,Roll,Pitch,Yaw,Magnification)
436    integer(4)::iP         !>5: Printing enable, use: write(iP,...)
437   !--Internals
438    real(8)   ::H(4,4)     !internal copy of hPoI7
439    real(8)   ::XYZ(3)
440    real(8)   ::X,Y,Z,Roll,Pitch,Yaw,Mag  !Individual 7DoF states
441    real(8)   ::sP         !sine of Pitch
442    integer(4)::i
443    !--EndDefs-------------------------------------------------
444     H=hPoI
445    !--Decode Hout's 7DoF states:
446     do i=1,3; XYZ(i)=H(1,i)*H(1,4)+H(2,i)*H(2,4)+H(3,i)*H(3,4); enddo!i
447     Mag = 1.d0/H(4,4)
448     X   = XYZ(1)
449     Y   = XYZ(2);              sP    = -H(3,1)
450     Z   = XYZ(3);              Pitch =  dasind(sP)
451       if(dabs(sP)<1.d0) then; Yaw   =  datan2d(H(2,1),H(1,1))
452                               Roll  =  datan2d(H(3,2),H(3,3))
453                          else; Roll  =  0.d0
454                               Yaw   =  datan2d(-H(1,2),H(2,2))
455       endif!|sP|<1.
456     PoI7=(/ X,Y,Z,Roll,Pitch,Yaw,Mag /)
457     if(iP>5) write(iP,"(7f10.4,'     :PoI7')") PoI7                          ;return
458  End Subroutine hPoIDecode
459  !-------------------------------------------------------------------------------7-9
460
461  Subroutine hFS7Gen(hFS7,XyzRpyM,iP)
462  !2020.06.10.0705cdt JMS- Generates 6DoF rigid body motion (+magnification)
463  !                         using Standard Flight Simulation coordinates.
464   implicit none
465   !--Arguments:
466    real(8)   ::hFS7(4,4)  !<-- Translate <-- Magnify+Rotate <--
467                           ! C11 , C12 , C13 ,    +X/Mag
468                           ! C21 , C22 , C23 ,    +Y/Mag
469                           ! C31 , C32 , C33 ,    +Z/Mag
470                           ! 0.d0, 0.d0, 0.d0, +1.d0/Mag
471
472    real(8)   ::XyzRpyM(7) !7DoF rigid body motion & magnification:
473   !                       !X    Y    Z  Roll Pitch Yaw  Mag
474   !                       !1    2    3    4    5    6    7
475    integer(4)::iP         !>5: Printing enable, use: write(iP,...)
476   !--Internals
477    real(8)   ::V(7)       !internal copy of XyzRpyM(7)
478    real(8)   ::H(4,4)     !internal copy of hFS input
479    real(8)   ::DC(3,3)    !Direction cosines of (Roll,Pitch,Yaw)
480    integer(4)::i,j
481    !--EndDefs-------------------------------------------------
482     V=XyzRpyM
483     if(iP>5) write(iP,"('hFS7Gen: XyzRpyM=')")
484     if(iP>5) write(iP,"(7f10.4,'      :FS7')") V
485    !--Compute the direction cosines:
486     call DCfromRPY(DC,V(4:6),iP)
487    !--Compute the FS transform of XyzRpyM:
488     H(1,1:4) = (/ DC(1,1), DC(1,2), DC(1,3), V(1)/V(7) /)
489     H(2,1:4) = (/ DC(2,1), DC(2,2), DC(2,3), V(2)/V(7) /)
490     H(3,1:4) = (/ DC(3,1), DC(3,2), DC(3,3), V(3)/V(7) /)
491     H(4,1:4) = (/  0.d0   ,  0.d0   ,  0.d0   ,+1.d0/V(7) /)
492     hFS7     = H                                                 ;if(iP<6) return
493     write(iP,"( '  yields hFS7:')")
494     write(iP,"( 4f20.9)") ((H(i,j),j=1,4),i=1,4)                          ;return
495  End Subroutine hFS7Gen
496  !-------------------------------------------------------------------------------7-9
```

```
497
498  Subroutine hFS7Decode(hFS7,FS7,iP)
499  !2020.06.10.0700cdt JMS- !Decodes hFS7's 7DoF states.
500   implicit none
501   !--Arguments:
502    real(8)   ::hFS7(4,4)  !<-- Translate <-- Magnify+Rotate <--
503    real(8)   ::FS7(7)     !Decoded seven degrees of freedom:
504                          !  (X,Y,Z,Roll,Pitch,Yaw,Magnification)
505    integer(4)::iP         !>5: Printing enable, use: write(iP,...)
506   !--Internals
507    real(8)   ::H(4,4)     !internal copy of hFS7
508    real(8)   ::X,Y,Z,Roll,Pitch,Yaw,Mag  !Individual 7DoF states
509    real(8)   ::sP         !sine of Pitch
510    !--EndDefs-----------------------------------------------
511     H=hFS7
512    !--Decode hFS7's 7DoF states:
513                    Mag = 1.d0/H(4,4)
514     X   = H(1,4)*Mag
515     Y   = H(2,4)*Mag;          sP    = -H(3,1)
516     Z   = H(3,4)*Mag;          Pitch =  dasind(sP)
517       if(dabs(sP)<1.d0) then; Yaw   =  datan2d(H(2,1),H(1,1))
518                               Roll  =  datan2d(H(3,2),H(3,3))
519                          else; Roll  =  0.d0
520                               Yaw   =  datan2d(-H(1,2),H(2,2))
521       endif!|sP|<1.
522     FS7 = (/ X,Y,Z,Roll,Pitch,Yaw,Mag /)
523     if(iP>5) write(iP,"(7f10.4,'      :FS7')") FS7                          ;return
524  End Subroutine hFS7Decode
525  !-------------------------------------------------------------------------------7-9
526
527  Subroutine DCfromRPY(DC,RPY,iP)
528  !2020.06.10.0420cdt JMS- !Computes Direction Cosines from FS (Roll,Pitch,Yaw).
529   implicit none
530   !--Arguments:
531    real(8)   ::DC(3,3)    !Direction Cosine output.
532    real(8)   ::RPY(3)     !(Roll,Pitch,Yaw) - in Flight Simulation coordinates.
533    integer(4)::iP         !>5: Printing enable, use: write(iP,...)
534   !--Internals
535    real(8)   ::Roll,Pitch,Yaw      !Individual rotation angles
536    real(8)   ::sr,cr, sp,cp, sy,cy !Sines & Cosines of Roll,Pitch,& Yaw
537    real(8)   ::C11,C21,C31,C12,C22,C32,C13,C23,C33 !DC coefficients;
538    !--EndDefs-----------------------------------------------
539     Roll  = RPY(1); sR=dsind(Roll ); cR=dcosd(Roll )
540     Pitch = RPY(2); sP=dsind(Pitch); cP=dcosd(Pitch)
541     Yaw   = RPY(3); sY=dsind(Yaw  ); cY=dcosd(Yaw  )
542     C11 = cY*cP ; C12 = (cY*sP*sR-sY*cR) ; C13 = (cY*sP*cR+sY*sR)
543     C21 = sY*cP ; C22 = (sY*sP*sR+cY*cR) ; C23 = (sY*sP*cR-cY*sR)
544     C31 =   -sP ; C32 =     cP*sR        ; C33 =     cP*cR
545     DC(1,1:3)=(/ C11,C12,C13 /)
546     DC(2,1:3)=(/ C21,C22,C23 /)
547     DC(3,1:3)=(/ C31,C32,C33 /)                                           ;return
548  End Subroutine DCfromRPY
549  !-------------------------------------------------------------------------------7-9
550
551  Subroutine OddEven7DoF(ioe,ioein,ioeu,DoFsInUse,iP)
552  !2020.06.20.1530cdt JMS- Odd|Even DoF management
553   implicit none
554   !--Arguments
555    integer(4)::ioe(0:2)      !(0)- =1: Primary control- vertical- X,Z,Pitch,Mag
556                             !     =2:                  - lateral - Y,Roll,Yaw
557                             !(1)    :Controlled DoF- vertical
558                             !(2)    :              - lateral
```

```
559     integer(4)::ioein(0:1)   !Axis change: Odd-or-even & DoF#
560     integer(4)::ioeu         !=ioe(ioe(0))
561     integer(4)::DoFsInUse(0:7)!=0 not under control
562                              !=1 primary   #1 -onscreen
563                              !=2 secondary #2 -
564     integer(4)::iP           ! >5: Printing enable, use: write(iP,...)
565    !--Internals
566     !--EndDefs-------------------------------------------------
567      if(iP>5) write(iP,"('OddEven7DoF: ioein=',2i2)") ioein
568                    ioe(0)     = ioein(0)
569                ioe(ioe(0))    = ioein(1) !Activate the DoF
570      DoFsInUse                = 0
571      DoFsInUse(ioe(  ioe(0))) = 1
572      DoFsInUse(ioe(3-ioe(0))) = 2
573                ioeu           = ioe(0)
574                ioein          = 0
575      if(iP>5) write(iP,"(13x,          'ioe  =',3i2,3x,8i2)") ioe,DoFsInUse
576      return
577   End Subroutine OddEven7DoF
578   !-------------------------------------------------------------------------------7-9
579
580   Subroutine PackM7u
581   !2020.06.21.0900cdt JMS- Unpacks a Unit Cube vector into its 14 components.
582   !--Globals
583    use ModelDef    ,only: &
584      M7u,DoFL,H,DoFreset,Sens,nM7L,ID,iType,Locked,iec,ioe,DoFsInUse,M7Init
585    implicit none
586     !--EndDefs-------------------------------------------------
587      M7u%DoF       =DoFL
588      M7u%H         =H
589      M7u%DoFreset  =DoFreset
590      M7u%Sens      =Sens
591      M7u%n         =nM7L
592      M7u%ID        =ID
593      M7u%iType     =iType
594      M7u%Locked    =Locked
595      M7u%ie        =iec
596      M7u%ioe       =ioe
597      M7u%DoFsInUse =DoFsInUse
598      M7u%M7Init    =M7Init                                                ;return
599   End Subroutine PackM7u
600   !-------------------------------------------------------------------------------7-9
601
602   Subroutine UnPackM7u
603   !2020.06.21.0900cdt JMS- Unpacks a Unit Cube vector into its 14 components.
604   !--Globals
605    use ModelDef    ,only: &
606      M7u,DoFL,H,DoFreset,Sens,nM7L,ID,iType,Locked,iec,ioe,DoFsInUse,M7Init
607    implicit none
608     !--EndDefs-------------------------------------------------
609      DoFL      =M7u%DoF
610      H         =M7u%H
611      DoFreset  =M7u%DoFreset
612      Sens      =M7u%Sens
613      nM7L      =M7u%n
614      ID        =M7u%ID
615      iType     =M7u%iType
616      Locked    =M7u%Locked
617      iec       =M7u%ie
618      ioe       =M7u%ioe
619      DoFsInUse =M7u%DoFsInUse
620      M7Init    =M7u%M7Init                                                ;return
```

```
621  End Subroutine UnPackM7u
622  !-------------------------------------------------------------------------------7-9
623
624  Subroutine CubeGrid(nCol)
625  !2024.04.28.2015cdt JMS- Using +X:Orange, +Y:Green, +Z:White for Rubik's cube
626  !2020.04.05.1315cdt JMS- Draws X,Y,Z axes/6dof Origin, and an enclosing cube.
627  !--Globals                                                S1ModDef.f95: 2024.03.18
628   use OpenGLRec,only:  & !Ref: OpenGL GL/GLU/GLUT docs
629        glVertex3f,glShadeModel,GL_SMOOTH,GL_FLAT,glBegin ,GL_LINES,glLineWidth &
630       ,glFlush,glEnd
631   use S2Callback, only:CheckGL
632  !--End Globals
633   implicit none
634   !--Arguments
635    integer(4)::nCol
636   !--Internals
637    integer(4)::i,j,k
638    real(4)   ::Frac,gi,gj,gk
639   !real(8)   ::Xyzh(4),Rpyh(4)
640    real(8)   ::XYZRPYwC(7) !(X,Y,Z,Roll,Pitch,Yaw,height of Character)
641    character(len=80)::PText
642
643
644    !--EndDefs-------------------------------------------------
645    !--draw corners
646     Frac=.9
647     call glLineWidth(1.)
648     call glShadeModel(GL_SMOOTH) !(GL_FLAT) !(GL_SMOOTH)
649     call Colors3D(nCol)
650     if(nCol>0) then
651       call glBegin(GL_LINES)
652         do     i=-1,1; gi=i
653           do   j=-1,1; gj=j
654             do k=-1,1; gk=k
655               call glVertex3f(gi     ,gj     ,gk      )
656               call glVertex3f(gi*Frac,gj     ,gk      )
657               call glVertex3f(gi     ,gj     ,gk      )
658               call glVertex3f(gi     ,gj*Frac,gk      )
659               call glVertex3f(gi     ,gj     ,gk      )
660               call glVertex3f(gi     ,gj     ,gk*Frac )
661         enddo; enddo; enddo!i
662       call glEnd;                                                    call glFlush
663     endif!nCol>0
664                                                  call glFlush;call CheckGL(+70628)
665   !--Draw colored axes:
666     call glLineWidth(5.)
667    !--                                                            mod 2020.04.23
668
669     if(1>0)then !Use +X:Green, +Y:Yellow,+Z:Red------------
670     call Colors3D(11) !Green
671     call glBegin(GL_LINES)
672       call glVertex3f(0.,0.,0.); call glVertex3f(1.,0.,0.)
673     call glEnd
674    !--
675     call Colors3D( 9) !Yellow
676     call glBegin(GL_LINES)
677       call glVertex3f(0.,0.,0.); call glVertex3f(0.,1.,0.)
678     call glEnd
679    !--
680     call Colors3D( 7) !Red
681     call glBegin(GL_LINES)
682       call glVertex3f(0.,0.,0.); call glVertex3f(0.,0.,1.)
```

```
683      call glEnd
684     !--
685                                                  call glFlush;call CheckGL(+70646)
686      if(nCol>0) then
687       !--Axis labels:
688                     !XYZRPYwC=(/   X ,   Y  , Z   , Roll, Pitch, Yaw,width Ch/)
689                      XYZRPYwC=(/ .8d0, .0d0,-.02d0, 0.d0,  0.d0,  0.d0 ,.1d0 /)
690                         !VecFont7(XYZRPYwC,LineWidth,iCol,Label)
691        PText="+X" ; call VecFont7(XYZRPYwC,  5.     , 11 ,PText) !Green
692
693                      XYZRPYwC=(/ .0d0, .8d0,-.02d0, 0.d0,  0.d0, 90.d0 ,.1d0 /)
694        PText="+Y" ; call VecFont7(XYZRPYwC,  5.     ,  9 ,PText) !Yellow
695
696                      XYZRPYwC=(/ .0d0, .02d0, .8d0, 0.d0,-90.d0,  0.d0 ,.1d0 /)
697        PText="+Z" ; call VecFont7(XYZRPYwC,  5.     ,  7 ,PText) !Red
698       !              XYZRPYwC=(/   X ,   Y  , Z   , Roll, Pitch, Yaw,width Ch/)
699                      XYZRPYwC=(/ .8d0, .02d0, .0d0, 0.d0,  0.d0,  0.d0 ,.1d0 /)
700      endif
701
702      else !Use rubik's cube face colors: --------------------
703     !call Colors3D(11) !Green
704      call Colors3D(15) !Orange
705      call glBegin(GL_LINES)
706        call glVertex3f(0.,0.,0.); call glVertex3f(1.,0.,0.)
707      call glEnd
708     !--
709      call Colors3D( 9) !Yellow
710      call Colors3D(11) !Green
711      call glBegin(GL_LINES)
712        call glVertex3f(0.,0.,0.); call glVertex3f(0.,1.,0.)
713      call glEnd
714     !--
715     !call Colors3D( 7) !Red
716      call Colors3D( 1) !White
717      call glBegin(GL_LINES)
718        call glVertex3f(0.,0.,0.); call glVertex3f(0.,0.,1.)
719      call glEnd
720     !--
721      call Colors3D( 7) !Red
722      call glBegin(GL_LINES)
723        call glVertex3f(0.,0.,0.); call glVertex3f(-1.,0.,0.)
724      call glEnd
725     !--
726      call Colors3D(14) !Blue
727      call glBegin(GL_LINES)
728        call glVertex3f(0.,0.,0.); call glVertex3f(0.,-1.,0.)
729      call glEnd
730     !--
731      call Colors3D( 9) !Yellow
732      call glBegin(GL_LINES)
733        call glVertex3f(0.,0.,0.); call glVertex3f(0.,0.,-1.)
734      call glEnd
735                                                  call glFlush;call CheckGL(+70646)
736      if(nCol>0) then
737       !--Axis labels:
738                     !XYZRPYwC=(/   X ,   Y  , Z   , Roll, Pitch, Yaw,width Ch/)
739                      XYZRPYwC=(/ .8d0, .0d0,-.02d0, 0.d0,  0.d0,  0.d0 ,.1d0 /)
740                         !VecFont7(XYZRPYwC,LineWidth,iCol,Label)
741       !PText="+X" ; call VecFont7(XYZRPYwC,  5.     , 11 ,PText) !Green
742        PText="+X" ; call VecFont7(XYZRPYwC,  2.     , 15 ,PText) !Orange
743
744                      XYZRPYwC=(/ .0d0, .8d0,-.02d0, 0.d0,  0.d0, 90.d0 ,.1d0 /)
```

```
745       !PText="+Y" ; call VecFont7(XYZRPYwC,  5.     ,  9 ,PText) !Yellow
746        PText="+Y" ; call VecFont7(XYZRPYwC,  2.     , 11 ,PText) !Green
747
748                     XYZRPYwC=(/ .0d0, .02d0, .8d0, 0.d0,-90.d0,  0.d0 ,.1d0 /)
749       !PText="+Z" ; call VecFont7(XYZRPYwC,  5.     ,  7 ,PText) !Red
750        PText="+Z" ; call VecFont7(XYZRPYwC,  2.     ,  1 ,PText) !White
751       !              XYZRPYwC=(/   X ,   Y  , Z   , Roll, Pitch, Yaw,width Ch/)
752                     XYZRPYwC=(/ .8d0, .02d0, .0d0, 0.d0,  0.d0,  0.d0 ,.1d0 /)
753       !call VecFont7(XYZRPYwC,LineWidth,iCol,Label)
754
755                     XYZRPYwC=(/-1.d0, .0d0,-.02d0, 0.d0,  0.d0,  0.d0 ,.1d0 /)
756        PText="-X" ; call VecFont7(XYZRPYwC,  2.     ,  7 ,PText) !Red
757
758                     XYZRPYwC=(/.0d0,-1.d0,-.02d0, 0.d0,  0.d0, 90.d0 ,.1d0 /)
759        PText="-Y" ; call VecFont7(XYZRPYwC,  2.     , 14 ,PText) !Blue
760
761                     XYZRPYwC=(/ .0d0, .02d0,-1.d0, 0.d0,-90.d0,  0.d0 ,.1d0 /)
762        PText="-Z" ; call VecFont7(XYZRPYwC,  2.     ,  9 ,PText) !Yellow
763      endif
764      endif!(1>0)then !-----------------------------------------
765                                                call glFlush;call CheckGL(+70658)
766                                                                    ;return
767   End Subroutine CubeGrid
768   !-------------------------------------------------------------------------------7-9
769
770   Subroutine WirePlane(Sizeu,nCol)
771   !2023.09.07.1200cdt JMS-  Draws wire plane model.
772   !--Globals                                          S1ModDef.f95: 2024.03.18
773    use OpenGLRec,only:  & !Ref: OpenGL GL/GLU/GLUT docs
774         glBegin      ,glEnd       ,glFlush                          &
775        ,glShadeModel ,GL_SMOOTH   ,GL_FLAT                          &
776        ,GL_POINTS    ,glPointSize                                   &
777        ,GL_LINES     ,glLineWidth                                   &
778        ,GL_LINE_STRIP                                               &
779        ,GL_Triangles                                                &
780        ,GL_QUADS                                                    &
781        ,glPolygonMode,GL_LINE       ,GL_FILL                        &
782        ,glCullFace   ,GL_CULL_FACE  ,GL_Front   ,GL_BACK            &
783   !    !,GL_FRONT_AND_GL_BACK                                       &
784        ,glVertex3f   ,glVertex3fv                                   &
785        ,GL_Lighting  ,glEnable    ,glDisable
786
787    use S2Callback, only:CheckGL
788   !--End Globals
789    implicit none
790    !--Arguments
791     real(4)   ::sizeu
792     integer(4)::nCol
793    !--Internals
794     real(8)   ::XYZRPYwC(7) !(X,Y,Z,Roll,Pitch,Yaw,height of Character)
795     character(len=80)::PText
796     real(4)   ::Po(3) = (/  0.    , 0.    , 0.     /)!Origin
797     real(4)   ::Px(3) = (/  1.    , 0.    , 0.     /)!Axes
798     real(4)   ::Py(3) = (/  0.    ,  .5   , 0.     /)
799     real(4)   ::PZ(3) = (/  0.    , 0.    ,  .5    /)
800     real(4)   ::P1(3) = (/ +1.    , 0.    , 0.     /)!Fuselage
801     real(4)   ::P2(3) = (/ -1.    , -.667 , 0.     /)
802     real(4)   ::P3(3) = (/ -1.    , +.667 , 0.     /)
803     real(4)   ::P4(3) = (/  0.    , 0.    , 0.     /)!Tail
804     real(4)   ::P5(3) = (/ -1.    , 0.    , 0.     /)
805     real(4)   ::P6(3) = (/ -1.    , 0.    , -.667  /)
806      !--EndDefs-----------------------------------------------
```

```
807      call glLineWidth(1.)
808      call glShadeModel(GL_SMOOTH) !(GL_FLAT)(GL_SMOOTH)
809      call Colors3D(nCol)                          ;call glFlush;call CheckGL(+70628)
810     !--
811   !  call Colors3D( 1) !White
812   !  call Colors3D( 7) !Red
813   !  call Colors3D( 9) !Yellow
814   !  call Colors3D(11) !Green
815
816      call glPolygonMode(GL_FRONT            ,GL_FILL)
817      call glPolygonMode(GL_BACK             ,GL_FILL)
818
819      call glEnable(GL_CULL_FACE)   !-----
820      call glCullFace(GL_BACK)
821      call Colors3D(2)
822      call glBegin(GL_Triangles)
823       !Fuselage:
824        call glVertex3fv( P1*SizeU)
825        call glVertex3fv( P2*SizeU)
826        call glVertex3fv( P3*SizeU)
827      call glEnd
828      call glDisable(GL_CULL_FACE)                                    ;call glFlush
829                                  !-----
830     !call Colors3D(nCol)
831     !call Colors3D( 12 )
832      call Colors3D(  4 )
833      call glBegin(GL_Triangles)
834       !Tail:
835        call glVertex3fv( P4*SizeU)
836        call glVertex3fv( P5*SizeU)
837        call glVertex3fv( P6*SizeU)
838      call glEnd
839
840      call Colors3D(3)
841     !call Colors3D(5)  !...didn't help RC clarity.
842      call glLineWidth(4.)
843      call glBegin(GL_LINE_STRIP)
844       !Fuselage:
845        call glVertex3fv( P1*SizeU)
846        call glVertex3fv( P2*SizeU)
847        call glVertex3fv( P3*SizeU)
848        call glVertex3fv( P1*SizeU)
849      call glEnd
850      call glBegin(GL_LINE_STRIP)
851       !Tail:
852        call glVertex3fv( P4*SizeU)
853        call glVertex3fv( P5*SizeU)
854        call glVertex3fv( P6*SizeU)
855        call glVertex3fv( P4*SizeU)
856      call glEnd
857
858    !--Draw colored axes:
859      call glLineWidth(3.)
860     !--
861      call Colors3D(11) !Green
862      call glBegin(GL_LINES)
863        call glVertex3fv( Po*SizeU); call glVertex3fv( Px*SizeU)
864      call glEnd
865     !--
866      call Colors3D( 9) !Yellow
867      call glBegin(GL_LINES)
868        call glVertex3fv( Po*SizeU); call glVertex3fv( Py*SizeU)
```

```
869     call glEnd
870    !--
871     call Colors3D( 7) !Red
872     call glBegin(GL_LINES)
873       call glVertex3fv( Po*SizeU); call glVertex3fv( Pz*SizeU)
874     call glEnd
875
876     if(nCol>0) then
877      !--Axis labels:
878      !              XYZRPYwC=(/   X ,   Y  , Z   , Roll, Pitch, Yaw,width Ch/)
879              !XYZRPYwC     =(/ .8d0, .02d0, .0d0,   0.d0,  0.d0,  0.d0 ,.1d0 /)
880               XYZRPYwC     =(/1.2d0, .02d0, .0d0,   0.d0,  0.d0,180.d0 ,.1d0 /)
881               XYZRPYwC(1:3)=XYZRPYwC(1:3)*SizeU
882      !call VecFont7(XYZRPYwC,LineWidth,iCol,Label)
883       PText="+X" ; call VecFont7(XYZRPYwC,  2.     , 11 ,PText) !Green
884               XYZRPYwC     =(/-.02d0, .4d0, .0d0,   0.d0,  0.d0,+90.d0 ,.1d0 /)
885               XYZRPYwC(1:3)=XYZRPYwC(1:3)*SizeU
886       PText="+Y" ; call VecFont7(XYZRPYwC,  2.     ,  9 ,PText) !Yellow
887              !XYZRPYwC     =(/ .00d0, .02d0, .4d0,   0.d0,-90.d0,  0.d0 ,.1d0 /)
888              !XYZRPYwC     =(/ .08d0, .00d0,1.2d0,+180.d0,+90.d0,  0.d0 ,.1d0 /)
889               XYZRPYwC     =(/-.20d0,-.50d0, .6d0, -90.d0,  0.d0,+90.d0 ,.1d0 /)
890               XYZRPYwC(1:3)=XYZRPYwC(1:3)*SizeU
891       PText="+Z" ; call VecFont7(XYZRPYwC,  2.     ,  7 ,PText) !Red
892                                                call glFlush;call CheckGL(+70658)
893     endif                                                                ;return
894  End Subroutine WirePlane
895  !-------------------------------------------------------------------------7-9
896
897  Subroutine VecFont7(XYZRPYwC,LineWidth,iCol,Label)
898  !2020.06.12.0630cdt JMS- Vector-based character string writing in ~7DoF.
899  !                       using an OpenGL vector font.
900  !A standard 6DoF + scale operation with an unusual scale.
901  !XYZRPYwC(7)=(X, Y, Z, Roll, Pitch, Yaw, width of Character)
902  !  LineWidth: r*4 pixels, color index:i*4 iCol, & the Label*80.
903
904   use OpenGLRec,only:  & !Ref: OpenGL GL/GLU/GLUT docs
905        glGetIntegerv,GL_MATRIX_MODE,glMatrixMode,GL_MODELVIEW,glPushMatrix   &
906       ,glMultMatrixd,glLineWidth,glShadeModel,GL_FLAT,glBegin,glEnd,GL_LINES &
907       ,glutStrokeCharacter,GLUT_STROKE_MONO_ROMAN,glPopMatrix,glFlush        &
908       ,glLoadIdentity,GL_PROJECTION,glOrtho,glGetDoublev,GL_MODELVIEW_MATRIX
909   use ioDef     ,only: Up
910   use ScreenDef ,only: vScrnHVD !screen & colors
911  !-----
912   implicit none                                                      !arguments
913    real(8)   ::XYZRPYwC(7) !(X,Y,Z,Roll,Pitch,Yaw,height of Character)
914    real(4)   ::LineWidth  !Width of character strokes, in pixels
915    integer(4)::iCol       !Text color
916    character::Label*80   !Text to be written
917    !--                                                                !internals
918    integer(4)::i,iCharL,MtxMode(1),n,nChar
919    real(8)   ::XyzRpySL(7)
920    real(8)   ::Model7h(4,4)
921    character::A*80,A80(80)*1;   equivalence(A,A80)
922  ! real(8)   ::XyzRpySDecode(7) !(X,Y,Z,Roll,Pitch,Yaw,height of Character)
923    !-----                                                              !end defs
924
925    !--Adapt ModelView:
926    !if(Up>5) write(Up,"(/'VecFont7: ',80a1)") (Label(i:i),i=1,len_trim(Label))
927     call glGetIntegerv(GL_MATRIX_MODE,MtxMode)
928     call glMatrixMode(GL_MODELVIEW ); call glPushMatrix !; call glLoadIdentity
929     XyzRpySL   = XYZRPYwC
930    !XyzRpySL(4)=0.d0              !no roll - 2021.10.03
```

```
931     XyzRpySL(4)=XyzRpySL(4)-90.d0 !From: +X:along text +Y:up
932                                   ! To : +X:along text -Z:up          2022.05.10
933     XyzRpySL(7)= XYZRPYwC(7)/110.d0 !scales size to user units in modelspace.
934     call hFS7Gen(Model7h,XyzRpySL,0) !,Up)
935
936  !  if(Up>5) call Decode7DoF(     Model7h,0,XyzRpySDecode, 0)!,Up)
937                                                  call glMultMatrixd(Model7h)
938    !--Render the text
939     call Colors3D(iCol)
940     if(LineWidth>0.) call glLineWidth(LineWidth)                    !2021.10.03
941     call glShadeModel(GL_FLAT)
942     if(iCol>0) then;call glBegin(GL_LINES);call glEnd;call glFlush !Quirk fix?
943     endif !iCol>0)
944     A=char(0); A=Label; nChar=len_trim(A)
945     do n=1,nChar;          i=ichar(A80(n))              ;if((n<1).or.(n> 80)) exit
946       call PrintableIchar(i,iCharL)                     ;if(iCharL==0) exit
947       call glutStrokeCharacter(GLUT_STROKE_MONO_ROMAN,iCharL)
948     enddo                                                ;call glFlush
949    !Restore MODELVIEW, & MatrixMode:
950     call glMatrixMode(GL_MODELVIEW ); call glPopMatrix   ;call glFlush
951     call glMatrixMode(  MtxMode(1))                      ;call glFlush
952     return
953  End Subroutine VecFont7
954  !-------------------------------------------------------------------------7 9
955
956  Subroutine QFromRpy(Q,Rpyh)
957  !2020.05.29.1010cdt JMS- Computes the quaternion of an SFS rotation. (McAir)
958   implicit none
959   !--Arguments
960    real(8)  ::Q(4)    !Quaternion rotation
961    real(8)  ::Rpyh(4) !homogeneous (Roll,Pitch,Yaw,W) SFS coordinates        deg
962   !--Internals
963    real(8)  ::RollD2,PitchD2,YawD2 !Normalized angles, /2.
964    real(8)  ::Sy,cY,sP,cP,sR,cR    !Sines & cosines of the normalized angles
965    !--EndDefs-----------------------------------------------
966     RollD2 = Rpyh(1)/(2.d0*Rpyh(4)); sR=dsind(RollD2 ); cR=dcosd(RollD2 )
967     PitchD2= Rpyh(2)/(2.d0*Rpyh(4)); sP=dsind(PitchD2); cP=dcosd(PitchD2)
968     YawD2  = Rpyh(3)/(2.d0*Rpyh(4)); sY=dsind(YawD2  ); cY=dcosd(YawD2  )
969    !--
970     Q(1)=cY*cP*cR+sY*sP*sR
971     Q(2)=sY*cP*cR-cY*sP*sR
972     Q(3)=cY*sP*cR+sY*cP*sR
973     Q(4)=cY*cP*sR-sY*sP*cR
974
975  !
976  !  Q(1)=cY*cP*cR+sY*sP*sR  !same
977  !  Q(2)=cY*cP*SR-sY*sP*cR  !prev Q(4)
978  !  Q(3)=cY*sP*cR+sY*cP*sR  !same
979  !  Q(4)=SY*cP*sR-cY*sP*sR  !prev Q(2)
980                                                                    return
981  End Subroutine QFromRpy
982  !-------------------------------------------------------------------------7-9
983
984  Subroutine QoutFromQoQi(Qout,Qo,Qi)
985  !2020.05.29.1010cdt JMS- Sequential gimbal multiplication.
986  !                     - Qout <-hodpodge(Qouter <- Qinner)
987   implicit none
988   !--Arguments
989    real(8)  ::Qout(4) !Quaternion- composite (~concatenated) rotation
990    real(8)  ::Qo(4)   !          - gimbal- outer
991    real(8)  ::Qi(4)   !          -       - inner
992   !--Internals
```

```
993     real(8)  ::Q(4)     !Isolates the output from the inputs.
994     !--EndDefs-----------------------------------------------
995      Q(1) = -Qo(4)*Qi(4) -Qo(3)*Qi(3) -Qo(2)*Qi(2) +Qo(1)*Qi(1)
996      Q(2) = -Qo(3)*Qi(4) +Qo(4)*Qi(3) +Qo(1)*Qi(2) +Qo(2)*Qi(1)
997      Q(3) = +Qo(2)*Qi(4) +Qo(1)*Qi(3) -Qo(4)*Qi(2) +Qo(3)*Qi(1)
998      Q(4) = +Qo(1)*Qi(4) -Qo(2)*Qi(3) +Qo(3)*Qi(2) +Qo(4)*Qi(1)
999     !--
1000     Qout(1) = Q(1)
1001     Qout(2) = Q(2)
1002     Qout(3) = Q(3)
1003     Qout(4) = Q(4)                                                  ;return
1004 End Subroutine QoutFromQoQi
1005 !-------------------------------------------------------------------------------7-9
1006
1007 Subroutine QtoRpy(Q,Rpyh)
1008 !2023.05.03.1405cdt JMS- Sets roll=0. deg. within .1 deg. of "up" or "down"
1009 !2020.05.29.1010cdt JMS- Updates Manual Modelspace PoI 7DoF.
1010  implicit none
1011  !--Arguments
1012   real(8)  ::Q(4)    !Quaternion rotation input
1013   real(8)  ::Rpyh(4) !homogeneous (Roll,Pitch,Yaw,W), SFS coordinates       deg
1014  !--Internals
1015   real(8)  ::A11,A21,A31,A12,A22,A32,A13,A23,A33
1016   real(8)  ::sP
1017
1018   !--EndDefs-----------------------------------------------
1019  !These are (agree with) the SFS Euler direction cosines:           2023.05.05
1020    A11=       Q(1)*Q(1)-Q(2)*Q(2)-Q(3)*Q(3)+Q(4)*Q(4)
1021    A21=2.D0*( Q(1)*Q(2)           +Q(3)*Q(4)          )
1022    A31=2.D0*(-Q(1)*Q(3)+Q(2)*Q(4)                     )
1023    !--
1024    A12=2.D0*(-Q(1)*Q(2)           +Q(3)*Q(4)          )
1025    A22=       Q(1)*Q(1)-Q(2)*Q(2)+Q(3)*Q(3)-Q(4)*Q(4)
1026    A32=2.D0*(           Q(2)*Q(3)          +Q(4)*Q(1) )
1027    !--
1028    A13=2.D0*( Q(1)*Q(3)+Q(2)*Q(4)                     )
1029    A23=2.D0*(           Q(2)*Q(3)          -Q(4)*Q(1) )
1030    A33=       Q(1)*Q(1)+Q(2)*Q(2)-Q(3)*Q(3)-Q(4)*Q(4)
1031    !--
1032    !Quaternion direction cosine printout:
1033    !write(iP,"(3f20.9)") A11,A12,A13
1034    !write(iP,"(3f20.9)") A21,A22,A23
1035    !write(iP,"(3f20.9)") A31,A32,A33
1036
1037    !Previous solution:
1038 !   Rpyh(1)=datan2d(A32,A33)
1039 !   Rpyh(2)=dasind(-A31)
1040 !   Rpyh(3)=datan2d(A21,A11)
1041 !   Rpyh(4)=1.d0
1042
1043    !Revised solution: Within .1 degree of "Up" or "Down"... Roll=0. degrees
1044                               sP = -A31
1045             RPYh(2) =  dasind(sP)        !Pitch
1046
1047     if(dabs(sP)<.999998476d0)  &              !sin(89.9deg)=.999998476...
1048       then; RPYh(1) =  datan2d(A32,A33)  !Roll
1049             RPYh(3) =  datan2d(A21,A11)  !Yaw
1050       else; RPYh(1) =  0.d0              !Roll
1051             RPYh(3) =  datan2d(-A12,A22) !Yaw
1052     endif!|sP|<1.
1053             Rpyh(4)=1.d0                                            ;return
1054 End Subroutine QtoRpy
```

```
!-------------------------------------------------------------------------------7-9

Subroutine QToQinverse(Q,Qinverse)
!2020.05.29.1010cdt JMS- UnRotates Q.
 implicit none
 !--Arguments
  real(8)  ::Q(4)         !Quaternion- input
  real(8)  ::Qinverse(4)  !          - inverse
 !--Internals
  !--EndDefs-------------------------------------------------
   Qinverse(1) = +Q(1)
   Qinverse(2) = -Q(2)
   Qinverse(3) = -Q(3)
   Qinverse(4) = -Q(4)
End Subroutine QToQinverse
!-------------------------------------------------------------------------------7-9

Subroutine QtoVhatAngle(Q,Vhat,Angle,iP)
!2023.08.22.1545cdt JMS- Conversion: quaternion -> "Jeffternion"
!2020.05.29.1010cdt JMS- Computes a quaternion's unit vector & rotation angle.
!                       - Ref,: D.H.Eberle 3D Game Engine Design P.16-17
 implicit none
 !--Arguments
  real(8)   ::Q(4)    !Quaternion rotation
  real(8)   ::Vhat(3) !Rotation axis unit vector
  real(8)   ::Angle   !Rotation angle around Vhat
  integer(4)::iP      ! >5: Printing enable, use: write(iP,...)
 !--Internals
  real(8)  ::sA
  !--EndDefs-------------------------------------------------
   if(     Q(1)==+1.d0) then !No rotation
     Angle   = 0.d0
     Vhat    = 0.d0
    else                      !Compute the rotation axis unit vector & angle.
     Angle   = 2.d0*dacosd(Q(1));  sA=dsind(Angle/2.d0)!Q(1)=    cos(a/2)
    !if(dabs(Angle-180.d0)<1.d-20) sA=1.d0
     Vhat(3) =             Q(2) / sA               !Q(2)=Vhat(1)*sin(a/2)
     Vhat(2) =             Q(3) / sA               !Q(3)=Vhat(2)*sin(a/2)
     Vhat(1) =             Q(4) / sA               !Q(4)=Vhat(3)*sin(a/2)
   endif! Q(1)==+1.d0
   if(iP<6)                                                              return
   write(iP,"(4f15.9,' Q(1:4) Quaternion')") Q
   write(iP,"(4f15.9,' cw:X,Y,Z,Angle>=0.',/,66x,'= Jeffternion')") Vhat,Angle
                                                                         return
End Subroutine QtoVhatAngle
!-------------------------------------------------------------------------------7-9


```