```
!Sa-Mathh8Mod.f95   Auxiliary         Stereo-3D Simulation Environment Vsn:1.00
!2025.05.24.1840dt JMS- Basic real(8) homog. vector math

!            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 hVecMath8Mod
! contains
!  Subroutine hAdd(Vouth,V1h,V2h)
!  Subroutine hSubtract(Vouth,V1h,V2h)
!  Subroutine hCross(Vouth,V1h,V2h)
!  Subroutine hDot(V1h,V2h,DotP)
!  Subroutine hVnorm(Vinh,Vouth,Vmag)
!  Subroutine hPointPolar(Ph,aRpyh,PDmag)
!End Module hVecMath8Mod

!-------------------------------------------------------------------------------7-9
Module hVecMath8Mod
!use hVecMath8Mod                                                    !2013.10.23
!use EnvJmsMod, only:FaultReport
!---
Implicit none
!---
public ::        &
! real(8) homogeneous math:
   hAdd          & !                                                  2013.09.30
  ,hSubtract     & !                                                  2013.09.30
  ,hCross        & !                                                  2013.10.03
  ,hDot          & !                                                  2013.10.03
  ,hVnorm        & !                                                  2013.10.03
  ,hPointPolar     !
!---
 contains
!-------------------------------------------------------------------------------7 9
!                    ///// real(8) homogeneous math: /////
!-------------------------------------------------------------------------------7 9
Subroutine hAdd(Vouth,V1h,V2h)
!2013.09.30.1650cdt JMS- Homogeneous vector addition:  Vouth=V1h+V2H
!                      - Traveler2/Athlon64/WinXPPro/APF9.0/Ogl1.2.1
 implicit none
  real(8)   ::Vouth(4)  !Output vector  (X,Y,Z,1.)                    !arguments
  real(8)   ::V1h(4)    !Input vector#1
  real(8)   ::V2h(4)    !Input vector#2
 !                                                                    !internals
  integer(4)::i
  real(8)   ::vTemph(4)    !Input vector#2
```

```
63     real(8)   ::V1Scale
64     real(8)   ::V2Scale
65     !-----
66      V1Scale=V1h(4) ;if(V1Scale.eq.0.d0) V1Scale=1.d0
67      V2Scale=V2h(4) ;if(V2Scale.eq.0.d0) V2Scale=1.d0
68      do i=1,3 ;vTemph(i)=V1h(i)/V1Scale+V2h(i)/V2Scale; enddo
69                vTemph(4)=1.d0
70      if(V1h(4)*V2h(4).eq.0.d0) then !~~~~~~~~~~~~~~~~
71       !call jPause("hAdd: V1h(4)*V2h(4)=0.")                  !Diagnostics
72        stop              "hAdd: V1h(4)*V2h(4)=0." !~~~
73        vTemph(4)=0.d0
74      endif !V1h(4)*V2h(4)=0.
75      do i=1,4 ;Vouth(i)=vTemph(i) ;enddo!i
76      return
77   End Subroutine hAdd
78   !-------------------------------------------------------------------------------7 9
79   Subroutine hSubtract(Vouth,V1h,V2h)
80   !2013.09.30.1700cdt JMS- Homogeneous vector Subtraction:  Vouth=V1h-V2H
81   !                     - Traveler2/Athlon64/WinXPPro/APF9.0/Ogl1.2.1
82    implicit none
83     real(8)   ::Vouth(4)  !Output vector  (X,Y,Z,1.)                    !arguments
84     real(8)   ::V1h(4)    !Input vector#1
85     real(8)   ::V2h(4)    !Input vector#2
86    !                                                                    !internals
87     integer(4)::i
88     real(8)   ::vTemph(4)    !Input vector#2
89     real(8)   ::V1Scale
90     real(8)   ::V2Scale
91     !-----
92      V1Scale=V1h(4) ;if(V1Scale.eq.0.d0) V1Scale=1.d0
93      V2Scale=V2h(4) ;if(V2Scale.eq.0.d0) V2Scale=1.d0
94      do i=1,3 ;vTemph(i)=V1h(i)/V1Scale-V2h(i)/V2Scale; enddo
95                           vTemph(4)=1.d0
96      if(V1h(4)*V2h(4).eq.0.d0) then !~~~~~~~~~~~~~~~~~~~
97       !call jPause("hSubtract: V1h(4)*V2h(4)=0.")              !Diagnostics
98        stop             "hSubtract: V1h(4)*V2h(4)=0."!~~
99        vTemph(4)=0.d0
100     endif !V1h(4)*V2h(4)=0.
101     do i=1,4 ;Vouth(i)=vTemph(i) ;enddo!i
102     return
103  End Subroutine hSubtract
104  !-------------------------------------------------------------------------------7 9
105  Subroutine hCross(Vouth,V1h,V2h)
106  !2013.10.03.2155cdt JMS- Cross product of homogeneous vectors: Vouth=V1h X V2h
107  !                     - Note: Vouth is isolated from V1h & V2h.
108  !                     - Traveler2/Athlon64/WinXPPro/APF9.0/Ogl1.2.1
109  implicit none                                                        !arguments
110    real(8)   ::V1h(4)      !Vector- input- #1
111    real(8)   ::V2h(4)      !             - #2
112    real(8)   ::Vouth(4)    !      - output cross product of = V1h X V2h
113   !---                                                               !internals
114    real(8)   ::Temph(4)    !Isolates Vouth from V1h & V2h.
115    !-----
116     Temph(1)=V1h(2)*V2h(3)-V1h(3)*V2h(2)
117     Temph(2)=V1h(3)*V2h(1)-V1h(1)*V2h(3)
118     Temph(3)=V1h(1)*V2h(2)-V1h(2)*V2h(1)
119     Temph(4)=V1h(4)*V2h(4)
120     if(Temph(4).ne.0.d0) then
121       Temph(1:3)=Temph(1:3)/Temph(4) ;Temph(4)=1.d0
122      else !~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
123      !call jPause("hCross: Temph(4)<>0.")                     !Diagnostics
124       stop              "hCross: Temph(4)=0."
125     endif !Temph(4)<>0.d0
```

```
125     endif !Temph(4)<>0.d0 ~~~~~~~~~~~~~~~~~~~
126     Vouth=Temph
127     return
128   End Subroutine hCross
129   !-----------------------------------------------------------------------7 9
130   Subroutine hDot(V1h,V2h,DotP)                                 !2013.11.22.0850
131   !2013.11.22.0850cdt JMS- zero (4)th components are permitted.
132   !2013.10.03.2155cdt JMS- Cross product of homogeneous vectors: Vouth=V1h X V2h
133   !                     - Note: DotP is isolated from V1h & V2h.
134   !                     - Traveler2/Athlon64/WinXPPro/APF9.0/Ogl1.2.1
135    implicit none                                                  !arguments
136     real(8)   ::V1h(4)     !Vector- input- #1
137     real(8)   ::V2h(4)     !              - #2
138     real(8)   ::DotP       !      - Dot product = V1h .dot. V2h
139     !---                                                          !internals
140     real(8)   ::Temp       !Isolates DotP from V1h() & V2h().
141     !-----
142     Temp=V1h(1)*V2h(1)+V1h(2)*V2h(2)+V1h(3)*V2h(3)
143     if(V1h(4).ne.0.d0) Temp=Temp/v1h(4)
144     if(V2h(4).ne.0.d0) Temp=Temp/v2h(4)
145    !if(V1h(4)*V2h(4).eq.0.d0)then !~~~~~~~~~~~~~
146    !  call jPause("hDot: V1h(4)*V2h(4)=0.")                      !Diagnostics
147    ! !stop              "hDot: V1h(4)*V2h(4)=0."
148    !endif !V1h(4)*V2h(4)=0. ~~~~~~~~~~~~~~~~~~~
149     DotP=Temp
150     return
151   End Subroutine hDot
152   !-----------------------------------------------------------------------7 9
153   Subroutine hVnorm(Vinh,Vouth,Vmag)                            !2013.11.22.0715
154   !2013.11.22.0715cst JMS- Improved the fault logic to pass Vinh(4)=0. inputs.
155   !2013.11.18.0840cst JMS- Comment corrections
156   !2013.10.03.2315cdt JMS- Normalizes a homogeneous vector: Vouth=norm(Vinh)
157   !                     - Traveler2/Athlon64/WinXPPro/APF9.0/Ogl1.2.1
158    implicit none                                                  !arguments
159     real(8)   ::Vinh(4)    !Vector- input
160     real(8)   ::Vouth(4)   !       - output- normalized
161     real(8)   ::Vmag       !Length of Vinh (homogeneous)
162     !---                                                          !internals
163     real(8)   ::Temph(4)   !Isolates Vouth from Vinh.
164     real(8)   ::TempMag
165     !-----
166                               Temph=Vinh
167     if(Vinh(4).ne.0.d0) Temph=Temph/Vinh(4)
168     TempMag=Temph(1)*Temph(1)+Temph(2)*Temph(2)+Temph(3)*Temph(3)
169     if(TempMag.gt.0.d0) then
170       TempMag=dsqrt(TempMag)
171       Temph(1:3)=Temph(1:3)/TempMag
172      else
173       Temph=(/0.d0,0.d0,0.d0,1.d0/); TempMag=0.d0
174     endif !TempMag>0.
175     if((TempMag.eq.0.d0).and.(Temph(4).eq.0)) then !~~~~~
176      !call jPause("hVnorm: Vinh is a null vector.")              !Diagnostics
177       stop              "hVnorm: Vinh is a null vector."
178     endif !TempMag>0 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
179     Vouth=TempH ;Vmag=TempMag
180     return
181   End Subroutine hVnorm
182   !-----------------------------------------------------------------------7 9
183   Subroutine hPointPolar(Ph,aRpyh,PDmag)
184   !2013.10.01.2155cdt JMS- Expresses a 3D point in FS ~polar coordinates.
185   !                     - Traveler2/Athlon64/WinXPPro/APF9.0/Ogl1.2.1
186    !-----
```

```
 Implicit none                                                          !arguments
  real(8)   ::Ph(4)    !Input - Point#1
  real(8)   ::aRpyh(4) !Output- att.(Roll,Pitch,Yaw)-                     [degrees]
  real(8)   ::PDmag    !      - Distance to point
  !---                                                                   !internals
  real(8)   ::Ptemph(4)
  !-----
   Ptemph=Ph
   call hVnorm(Ph,Ptemph,PDmag)
   aRpyh=(/0.d0,0.d0,0.d0,1.d0/)
   aRpyh(2)=-dasind(Ptemph(3))
   if((Ptemph(1).ne.0.d0).or.(Ptemph(2).ne.0.d0))  &
   aRpyh(3)=datan2d(Ptemph(2),Ptemph(1))
   return
End Subroutine hPointPolar
!-------------------------------------------------------------------------------7 9
End Module hVecMath8Mod
!-------------------------------------------------------------------------------7 9


```