```
!S4Callbacks.f95       Group ID: #4   Stereo-3D Simulation Environment Vsn:1.00
!2025.05.24.1840cdt JMS- CallBack referenced routines*, & utilities->OpenGL.

!            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 UserView
!*Subroutine Keyboard(Key,xCursor,yCursor)
!*Subroutine SpecialFunctionKeys(Key,xCursor,yCursor)
!*Subroutine MouseButtons(Button,State,iX,iY)
!*Subroutine MouseMotion(iX,iY)
! Subroutine MouseRecUpdate(md)    -used by MouseButtons() & MouseMotion()
!*...directly called by the OpenGL S2Callback subroutine passthroughs.
!-------------------------------------------------------------------------------7 9
Subroutine UserView
!2025.04.18.1520cdt JMS- CallBack- User(i.e.your) View generator:
!                      - ThreePhase=3 bypassed & removed.
!--Globals
 use OpenGLRec,only:  &      !Ref: OpenGL GL/GLU/GLUT docs
      glClear,glClearColor,glEnable,glFlush,glLoadIdentity,glMatrixMode &
     ,glMultMatrixd,glOrtho,glutPostRedisplay,GLUTSWAPBUFFERS           &
     ,GL_COLOR_BUFFER_BIT,GL_DEPTH_BUFFER_BIT,GL_DEPTH_TEST             &
     ,GL_MODELVIEW,GL_PROJECTION
!use S2Callback,only:CheckGL !OpenGL CB's, GlutHandoff, & glGetError
 use ioDef     ,only: &      !Files,Units,TimeStamp,Selfies,Flags
      ExeFileOut,ExeFileIn,EnvIni,EnvNml,Uupdate,Ur,Us,Ut,DaTimeLabel &
     ,NowView,Up,NowPrint,iTeapot
 use ScreenDef ,only: &      !screen & colors
      xWindowFull,yWindowFull,xyWindowRatio,vScrnHVD,iScrnColor &
     ,nCharCenX,nCharCenY,nCharMaxY,ForceRGBA,Side,nCharCenXS
 use KeyboardDef,only:&      !Keyboard
      KbdKey,ArrowKey
 use MouseDef   ,only: &     !Mouse
      Mouse,iMouse7DoF,MChanX,MChanY,MouseDirect,MouseDirectPrev,MXYdel
 use ModelDef   ,only: &     !Modelview Projection variables
      nRowOffset,nRowIn,nColOffset,nColIn,Nnutate,NutateAng,Nutate7,Nutateh
 use ViewDef,only: ThreePhase,Key,V,jVG,cVuModeName,cEyeName,cSplitName
 use SimDef     ,only: &     !Simulation F9-F12
      SimMode,dT,RunTimer,Fnew,IterTotal,IterRun
 use HelpDef    ,only: &     !Help text block
      HhHelpView,nHelpLines,BufferHhHelpView
 use AppsDef    ,only: &     !User Apps F1-F8
      Appnumber,AppNumberNew
 use BreakPtDef ,only: BrkOn !BreakPoint & scrolling tracker
!--End Globals
 implicit none
```

```
63    !--Internals
64     !--EndDefs-----------------------------------------------
65     Up=0
66     !--Increment the total iteration counter & update RunSecs (.nnn)
67      IterTotal=IterTotal+1;                          call RunSeconds
68     !--Screen color selection
69      if((KbdKey==67).or.(KbdKey==99)) then
70        iScrnColor=16-iScrnColor; KbdKey=0   ! toggle screen color
71      endif !KbdKey='C'or'c'
72      if(iScrnColor==15) call glClearColor(0.,0.,0.,0.) ! toggle->black
73      if(iScrnColor== 1) call glClearColor(1.,1.,1.,0.) ! toggle->white
74      call glMatrixMode(GL_PROJECTION) ;call glLoadIdentity
75      call glMatrixMode(GL_MODELVIEW)  ;call glLoadIdentity
76      call glClear(ior(GL_COLOR_BUFFER_BIT,GL_DEPTH_BUFFER_BIT))
77      call glEnable(GL_DEPTH_TEST)                                    ;call glFlush
78
79     !-- Define the coefficients of an orthographic screen, vertical [-1.:1.]:
80      vScrnHVD(1,1)=-xyWindowRatio; vScrnHVD(2,1)=-vScrnHVD(1,1) !X Left -to-right
81      vScrnHVD(1,2)=-1.000;         ; vScrnHVD(2,2)= 1.000       !Y bottom-to-top
82      vScrnHVD(1,3)=-1.000;         ; vScrnHVD(2,3)=+1.000       !Z far  -to-near
83                                                                 !(right handed)
84     !--The projection matrix will be (re)defined while calling the apps.
85     ! (PrntOrtho uses its own Identity projection matrix.)
86
87     !F1-F12 Function Key useages (Apps access + simulation controls):
88      if(SimMode< 1) SimMode=0
89      if((ArrowKey> 0).and.(ArrowKey< 13)) then
90        Fnew=ArrowKey; ArrowKey=0
91      endif !ArrowKey >0 & <13
92          AppNumberNew=  0
93      if((Fnew> 0).and.(Fnew< 9)) then !F1-F8 App selection):
94          AppNumber   =Fnew
95          AppNumberNew=Fnew; Fnew =0
96      endif !Fnew >0 & <9
97     !--Simulation mode switching:
98      select case(Fnew) !----------------------
99       case( 9); SimMode=1      ; Fnew=0;                      !F9 Reset
100       write(Us,"(   a18,' Reset(F 9) dT   =',f8.3)" )  &
101                        DaTimeLabel,dT
102      case(10); SimMode=2      ; Fnew=0                       !F10 Hold
103       write(Us,"(13x,a5,' Hold( F10) Timer=',f8.3)" )  &
104                        DaTimeLabel(14:18),RunTimer
105      case(11); SimMode=3      ; Fnew=0                       !F11  Run
106       write(Us,"(13x,a5,' Run(  F11) Timer=',f8.3)" )  &
107                        DaTimeLabel(14:18),RunTimer
108      case(12); SimMode=4      ; Fnew=0                       !F12 Stop
109       write(Us,"(13x,a5,' Stop( F12) Timer=',f8.3,i10,' iterations.')" ) &
110                        DaTimeLabel(14:18),RunTimer,IterRun
111     end select !Fnew ----------------------^
112
113    !--'Real Time' Simulation mode control:
114     select case(SimMode) !----------------------
115      case( 1); RunTimer=0.d0                                  !F9 Reset
116                IterRun =0
117      case( 2);                                                !F10 Hold
118      case( 3); RunTimer=RunTimer+dT                           !f11  Run
119                IterRun =IterRun +  1
120      case( 4);                                                !F12 Stop
121     end select !S.SimMode ----------------------^
122
123    !--KbdKey controls:
124     select case(KbdKey) !----------------------
125     !--Eye configuration  =-1:Right|Left  =0:Full screen +1:Left|Right
```

```
125     !--Eye configuration: =-1:Right|Left, =0:Full-screen,+1:Left|Right
126      case( 69); KbdKey   =0; Key%SplitScreen =Key%SplitScreen +1      !'E'
127            if(Key%SplitScreen> 1) Key%SplitScreen =          -1
128     !--Orthographic|Projective control. =0:orthographic,=1:persp,=2:3D
129      case(101); KbdKey   =0; Key%VuMode=Key%VuMode+1                  !'e'
130            if(Key%VuMode> 2) Key%VuMode=           0
131
132     !--Breakpoint scrolling control toggle:
133      case(66,98); KbdKey=0; BrkOn=1-BrkOn                             !'Bb'
134
135     !--Mouse output redirect toggle (when==2):
136      case(  77); KbdKey=0; if(iMouse7Dof <2) then; iMouse7DoF=2       !'M'
137                                              else; iMouse7DoF=0
138                            endif!iMouse7Dof <2
139     !--PoI / 6DoF toggle (when==1):
140      case( 109); KbdKey=0; if(iMouse7Dof==0) then; iMouse7DoF=1       !'m'
141                                              else; iMouse7DoF=0
142                            endif!iMouse7Dof==0
143     !--Help screen toggle:                                  changed 2024.04.02
144      case( 72); KbdKey=0                                              !'H':2
145                    if(HhHelpView/=2) then; HhHelpView=2
146                                      else; HhHelpView=0
147                    endif !HhHelpView=0
148      case(104); KbdKey=0                                              !'h':1
149                    if(HhHelpView/=1) then; HhHelpView=1
150                                      else; HhHelpView=0
151                    endif !HhHelpView=0
152
153      case(78    ); KbdKey=0 !Linear nutation                          !'N '
154                    if(Nnutate==0) then; Nnutate=2
155                                   else; Nnutate=0
156                    endif !Nnutate=0
157
158      case(   110); KbdKey=0  !Rotational nutation                   !' n'
159                    if(Nnutate==0) then; Nnutate=1
160                                   else; Nnutate=0
161                    endif !Nnutate=0
162
163 !    !--NowView: Toggles 'viewing' details on the screen using "v" or "V"
164      case(86)    ; KbdKey=0                                           !'V' uc
165       if(     NowView<=0) then; Nowview=2
166        elseif(NowView==1) then; Nowview=2
167        else                   ; Nowview=0
168       endif  !NowView==0
169      case(118)   ; KbdKey=0                                           !'v' lc
170       if(     NowView<=0) then; Nowview=1
171        elseif(NowView==2) then; Nowview=1
172        else                   ; Nowview=0
173       endif  !NowView==0
174
175     !--NowPrint: One cycle Printout controlled by "p" or "P" (beeps)
176      case( 80);   Up=  Ut; NowPrint=2; KbdKey=0 !to your .txt file     'P' uc
177                                              call DaTime18
178        write(Up,"(/a18,' Hardcopy: `P` ',46('-'))") DaTimeLabel
179      case(112);   Up=  Us; NowPrint=1; KbdKey=0 !to your screen        'p' lc
180                                               call DaTime18
181        write(Us,"(/a18,' DOS screen: `p` ',44('-'))") DaTimeLabel
182
183      !--GLUT Teapot viewing control by the "T" key:
184      case(20);      KbdKey=0;  iTeapot=          0                 !'ctl-t'->0
185      case(84,116); KbdKey=0;  iTeapot=iTeapot+1                     !'Tt'
186             if(iTeapot>7) iTeapot=             0
```

```
187     end select !KbdKey ----------------------^
188
189    !--Update model nutation:
190     if(Nnutate==0) then; NutateAng=0.d0
191      else
192           if(Nnutate==1)       NutateAng = NutateAng+1.d0
193           if(Nnutate==2)       NutateAng = NutateAng+2.d0
194           if(NutateAng>359.d0) NutateAng = 0.d0
195           if(Nnutate==1) & !Rotational nutation
196           Nutate7=(/0.d0,0.d0,0.d0,0.d0, 5.d0*dcosd(NutateAng)     &
197                                         ,10.d0*dsind(NutateAng),1.d0 /)
198           if(Nnutate/=1) & !Translational nutation
199           Nutate7=(/ 0.d0,dcosd(NutateAng)*.5d0,0.d0,0.d0,0.d0,0.d0,1.d0 /)
200           call hFS7Gen(Nutateh,Nutate7,0)
201     endif!Nnutate>0
202
203    !--Visualization- 14DoF //////////////////////
204     call View
205
206     if(NowView==2) call ShowProjectAndModel(-19,2,12,'S4Callbacks: @L202')
207                                                    !^ a20           !2021.10.04
208
209    !-- Results -> screen:
210  100 continue                                  !<-HhHelpView==1.'s goto (above).
211     side=0
212     call glutSwapBuffers; call glutPostRedisplay; call glFlush
213
214    !--Use "~" Tilda key to dump the full screen to SelfieN.bmp N=[0,1,...9]
215     if(KbdKey==126) then
216       call ScreenSelfie
217        KbdKey=0
218     endif !=126
219    !-- Use "`" key to toggle the screen depth selfie:
220   !!if(KbdKey== 96) then; DepthSelfie=1-DepthSelfie; KbdKey=0; endif
221    !-- Clear the one-cycle 'NowPrint':
222  !  if(NowPrint>0)   Up =0
223        NowPrint=0
224  !  MXYdel=0
225     return
226  End Subroutine UserView
227  !-------------------------------------------------------------------------7 9
228
229  Subroutine Keyboard(aKey,xCursor,yCursor)
230  !2020.06.03.1545cdt JMS-- CallBack- keyboard:
231  !--Globals
232   use OpenGLRec,only:  & !Ref: OpenGL GL/GLU/GLUT docs
233        glubyte,glcint,glutFullScreen,glFlush,glutReshapeWindow,glutGetModifiers
234   use ioDef   ,only:   & !Files,Units,TimeStamp,Selfies,Flags
235        Us,Ut,DaTimeLabel
236   use ScreenDef ,only: & !screen & colors
237        xWindowFull,yWindowFull,iFullScreen,iScrnColor,WinId,ctlKbd
238   use KeyboardDef,only:& !Keyboard
239        KbdKeyIn,KbdKey,ArrowKey,iKeyMods,nKpress,KeyHist
240   use SimDef    ,only: & !Simulation F9-F12
241        IterTotal
242  !--End Globals
243   implicit none
244   !--Arguments
245    character(kind=glubyte),intent(inout)::akey
246    integer(kind=glcint)   ,intent(inout)::xCursor
247    integer(kind=glcint)   ,intent(inout)::yCursor
248    integer(4)::xScreen,yScreen
249    !--EndDefs-------------------------------------------------
```

```
249   !--Endbers-------------------------------------------------
250    iKeyMods=glutGetModifiers()
251    if(ichar(aKey)> 0) KbdKeyIn=ichar(aKey)!Persistent/not erased when used
252    if(ichar(aKey)> 0) KbdKey  =ichar(aKey)!Zeroed after use
253   !--Update Keypress history:
254    nKpress=nKpress+1; KeyHist(1:3,3)=KeyHist(1:3,2)
255                       KeyHist(1:3,2)=KeyHist(1:3,1)
256                       KeyHist(1:3,1)=(/ichar(aKey),1,iKeyMods/)
257   !--ctl-Q hard quit:
258    if(KbdKey==17) then; if(Ut>9) close(Ut); stop 'ctl-Q.'//char(7) ;endif
259   !--ctl-K Keyboard redirection toggle:
260    if(KbdKey==11) then
261         ctlKbd =1-ctlKbd; KbdKey  =0
262      if(ctlKbd==1)          ArrowKey=0
263    endif!ctlKbd>0
264    if(ctlKbd>0) then !Silence KbdKey and ArrowKey
265       KbdKey=0                                                    ;return
266    endif!KbdKey==11
267
268    select case(KbdKey) !----------------------------
269     case(17); KbdKey=0; stop 'ctl-Q.'                          !'ctl-Q'
270     case(27); KbdKey=0                    ;call DaTime18 !escape
271      select case(iFullScreen) !-----------                    !screen toggle
272       case(0); call glutFullScreen; call glFlush                !fullscreen
273                write(Us,"(a18,' DOS screen: escape-end')") DaTimeLabel
274                iFullScreen=1                                    ;return
275       case(1)!;xScreen=400          ; yScreen=200
276                xScreen=xWindowFull*.95; yScreen=yWindowFull*.95
277                call glutReshapeWindow(xScreen,yScreen); call glFlush
278                                                              !small window
279                write(Us,"(/a18,' DOS screen: escape')") DaTimeLabel
280                iFullScreen=0                                    ;return
281      end select !iFullScreen
282
283     case(81,113); KbdKey=0                    ;call DaTime18 !'Q,q'
284      if(Ut>Us) then
285 !       Closing message to output file:
286          write(Ut,"(/a18,' Quit(Q/q)',14x,i15,' cbUserView iterations')")  &
287                       DaTimeLabel   ,IterTotal
288      !!if(RunTimer> 0.d0) &
289      !!  write(Ut,"(57x,'Timer=',f8.3,' seconds')") RunTimer
290      !!  write(Ut,"(58a1:)") (ExeBanner(i:i),i=1,len_trim(ExeBanner))
291          write(Ut,"(70x,' ...done.')")
292        close(Ut)
293      endif !Ut>6
294 !       Closing message to screen:
295          write(Us,"(/a18,' Quit(Q/q)',14x,i15,' cbUserView iterations')")  &
296                       DaTimeLabel   ,IterTotal
297      !!if(RunTimer> 0.d0) &
298      !!  write(Us,"(57x,'Timer=',f8.3,' seconds')") RunTimer
299      !!  write(Us,"(58a1:)") (ExeBanner(i:i),i=1,len_trim(ExeBanner))
300          write(Us,"(70x,' ...done.')")
301 !    Pause the DOS window from vanishing immediately at run's end:
302      if(iFullScreen==0) &
303      pause 'Left-click on the DOS window & press `enter` to exit.'
304      stop
305    end select !key
306    return
307 End Subroutine Keyboard
308 !-------------------------------------------------------------------------------7 9
309
310 Subroutine SpecialFunctionKeys(Key,xCursor,yCursor)
```

```
311 !2020.06.03.1545cdt JMS- CallBack- special function keys (F1-F12, arrows, etc.)
312 !                     - F9...F12 -> Reset,Hold,Operate,& Stop the timer.
313 !--Globals
314  use OpenGLRec  ,only:glcint,glutGetModifiers !Ref: OpenGL GL/GLU/GLUT Docs.
315  use KeyboardDef,only:ArrowKey,ArrowKeyIn,iKeyMods,nKpress,KeyHist !Keyboard
316  use ScreenDef  ,only:ctlKbd                  !screen & colors
317 !--End Globals
318  implicit none
319  !--Arguments
320   integer(glcint),intent(inout)::Key,xCursor,yCursor
321   !--EndDefs-----------------------------------------------
322    iKeyMods=glutGetModifiers()
323    if(Key   > 0) ArrowKeyIn = Key!Persistent/not erased when used
324    if(Key   > 0) ArrowKey   = Key!Zeroed after use
325   !--Update Keypress history:
326    nKpress=nKpress+1; KeyHist(1:3,3)=KeyHist(1:3,2)
327                       KeyHist(1:3,2)=KeyHist(1:3,1)
328                       KeyHist(1:3,1)=(/Key,2,iKeyMods/)
329   !--ctk-K silences KbdKey and ArrowKey
330    if(ctlKbd> 0) ArrowKey  =  0                                          ;return
331 End Subroutine SpecialFunctionKeys
332 !-------------------------------------------------------------------------------7 9
333
334 Subroutine MouseButtons(Button,State,iX,iY)
335 !2020.06.19.1100cdt JMS- CallBack- mouse button(s) press/release.
336
337 !Mouse buttons                    <- Persists after most recent button press.
338 !  #define GLUT_LEFT_BUTTON   0                                Reference: glut.h
339 !  #define GLUT_MIDDLE_BUTTON 1
340 !  #define GLUT_RIGHT_BUTTON  2
341 !Mouse button  state              <- Either button controls this
342 !  #define GLUT_DOWN          0
343 !  #define GLUT_UP            1
344 !--Globals
345  use OpenGLRec,only:  &           !Ref: OpenGL GL/GLU/GLUT docs
346                      glcint,glutWarpPointer,glutSetCursor &
347                     ,GLUT_CURSOR_CROSSHAIR        &  ! +
348                     ,GLUT_CURSOR_CYCLE            &  !~not
349                     ,GLUT_CURSOR_FULL_CROSSHAIR   &
350                     ,GLUT_CURSOR_INFO             &  ! crossed arrows
351                     ,GLUT_CURSOR_NONE             &
352                     ,GLUT_CURSOR_TOP_RIGHT_CORNER &
353                     ,GLUT_CURSOR_UP_DOWN          &  ! both ways
354                     ,GLUT_CURSOR_WAIT                ! hourglass
355  use ioDef      ,only: Us                  !Print unit#
356  use MouseDef   ,only: &                   !Mouse
357       iMouseButton,iMouseState,MouseDirect,Mouse,MouseDirectPrev,iMouse7DoF &
358      ,MouseU,MChan,MChanX,MChanY,nMpress,mdHist,MXY,MXYdel
359  use simdef     ,only: IterTotal
360 !--End Globals
361  implicit none
362  !--Arguments
363   integer(kind=glcint),intent(inOut)::Button,State,iX,iY
364  !--Internals
365   integer(4)   ::md
366   !--EndDefs-----------------------------------------------
367                         iMouseButton = Button
368                         iMouseState  = State
369                      MouseDirectPrev =   MouseDirect
370                      MouseDirect     =2*iMouseButton-iMouseState+2
371    if(iMouse7Dof==2) MouseDirect     =6 +MouseDirect
372                  md = MouseDirect
```

```
373    !--Update the mouse button use history:
374     nMpress=nMpress+1
375     mdHist( 3)=  mDHist( 2); mdHist( 2)=mDHist( 1); mdHist( 1)=md
376     MXY(1:4,3)=  MXY(1:4,2); MXY(1:4,2)=MXY(1:4,1)
377     MXY(1:4,1)= (/ iX,iY,MouseDirect,IterTotal /)
378     MXYdel    =  0
379
380    !--Mouse mode switching is done with the 'M' key.
381     if(iMouse7Dof==2) then  !Reposition the cursor at its previous location
382       call glutWarpPointer(Mouse(md)%iX   , Mouse(md)%iY   )
383                     else  !Use the present mouse position:
384             md        = MouseDirect
385       Mouse(md)%iX    = iX
386       Mouse(md)%iY    = iY
387     endif!iMouse7Dof==2
388
389    !--Process mouse motion:
390
391     call MouseRecUpdate(md)
392     MouseU = Mouse(md)
393     MChanX = MChan(MouseU%mcx)
394     MChanY = MChan(MouseU%mcy)
395
396     if(0>1) then
397    !--Change the mouse's screen cursor:
398     select case(md)
399      case(1); call glutSetCursor(GLUT_CURSOR_CYCLE)            !Lup
400      case(2); call glutSetCursor(GLUT_CURSOR_UP_DOWN)          !Ldown
401      case(3); call glutSetCursor(GLUT_CURSOR_WAIT)             !Mup
402      case(4); call glutSetCursor(GLUT_CURSOR_WAIT)             !Mdown
403      case(5); call glutSetCursor(GLUT_CURSOR_INFO)             !Rup
404      case(6); call glutSetCursor(GLUT_CURSOR_TOP_RIGHT_CORNER) !Rdown
405      case(7); call glutSetCursor(GLUT_CURSOR_WAIT)             !TBDup
406      case(8); call glutSetCursor(GLUT_CURSOR_WAIT)             !TBDdown
407     end select!im
408     endif!0>1 ////////////////////////////////////
409 !             call glutSetCursor(GLUT_CURSOR_NONE)
410    !--Middle Button: beep!
411     if((md==3).or.(md==4)) write(Us,"(a1)") char(7)                        ;return
412 End Subroutine MouseButtons
413 !-------------------------------------------------------------------------------7 9
414
415 Subroutine MouseMotion(iX,iY)
416 !2020.06.19.1100cdt JMS- CallBack- mouse motion:
417 !--Globals
418  use OpenGLRec,only:  &          !Ref: OpenGL GL/GLU/GLUT docs
419       glReadPixels,glutWarpPointer,glutSetCursor        &
420      ,GL_FLOAT,GL_DEPTH_COMPONENT,glcint,GLint,GLsizei
421  use ScreenDef  ,only: &               !screen & colors
422       xWindowFull,yWindowFull
423  use MouseDef   ,only: &               !Mouse         ///Consolidate///
424       MouseDirect,Mouse,mbID,MouseU,MChan,MChanX,MChanY,MouseDirectPrev &
425      ,MouseInit,MouseScreenh,mdTot,MXY,MXYdel
426  use simdef     ,only: IterTotal
427 !--End Globals
428  implicit none
429  !--Arguments
430   integer(kind=glcint ),intent(inout)::iX,iY
431   integer(kind=GLint  )              ::iXLoc,iYLoc
432   integer(kind=GLsizei)              ::JustOne=1
433  !--Internals
434   integer(4)::md !,iDoF
```

```
435    real(4) PixelDepth(1)
436    !--EndDefs-------------------------------------------------
437
438    !--Mouse initialization, called from GlutHandoff:
439     if(MouseInit==0) then
440      !--Screen-enter the cursor positions:
441       do md=1,mdTot; Mouse(md)%iX  =xWindowFull/2
442                      Mouse(md)%iY  =yWindowFull/2
443                      Mouse(md)%md  =md
444                      Mouse(md)%mcx =2*md-1
445                      Mouse(md)%mcy =2*md
446                      Mouse(md)%ID  =mbID(md)
447                      call MouseRecUpdate(md)
448       enddo!md
449      !--Position the cursor at Mouse(1):
450       MouseDirect    =1
451       call glutWarpPointer(Mouse(1)%iX   , Mouse(1)%iY   )
452       MouseInit      =1                                                ;return
453     endif !MouseInit==0
454
455    !--Update the mouse motion history:
456     MXY(1:4,3)=MXY(1:4,2); MXY(1:4,2)=MXY(1:4,1)
457     MXY(1:4,1)=(/iX,iY,MouseDirect,IterTotal/)
458    !--Compute this iteration's mouse motion change:
459                            MXYdel    =MXY(1:4,1)-MXY(1:4,2)
460     if(MXYdel(3)== 0)      MXYdel(3) =MouseDirect
461     if(MXYdel(4)<=120) then; MXYdel(4) =IterTotal !...within 120 iter. ~2. sec
462                        else; MXYdel    =(/0,0,MouseDirect,0/)
463     endif!MXYdel(4)==1
464
465    !--Process mouse motion:
466     md=MouseDirect
467     Mouse(md)%iX=iX; Mouse(md)%iY=iY
468     call MouseRecUpdate(md)
469
470    !--Simplify read-only access to the two active mouse channels:
471     MouseU = Mouse(md)
472     MChanX = MChan(MouseU%mcx)
473     MChanY = MChan(MouseU%mcy)
474
475    !--Screen depth of the single pixel under the active mouse cursor:
476    !Ref: Wright&Sweet P.384                                          2016.04.28
477    !WINGDIAPI void APIENTRY glReadPixels (GLint x, GLint y, GLsizei width
478    !         , GLsizei height, GLenum format, GLenum type, GLvoid *pixels)
479     iXLoc=               MouseU%iX
480     iYLoc= yWindowFull-1-MouseU%iY
481     call glReadPixels( iXLoc,iYLoc,JustOne,JustOne  &
482                       ,GL_DEPTH_COMPONENT,GL_FLOAT,PixelDepth)
483     MouseScreenh(1)= (2.*iXLoc-xWindowFull)/xWindowFull
484     MouseScreenh(2)= (2.*iYLoc-yWindowFull)/yWindowFull
485     MouseScreenh(3)= PixelDepth(1)*2.-1.                          !2020.05.25
486     MouseScreenh(4)=            1.0
487 !   call glutSetCursor(GLUT_CURSOR_CROSSHAIR)
488     return
489 End Subroutine MouseMotion
490 !-------------------------------------------------------------------------7 9
491
492 Subroutine MouseRecUpdate(md)
493 !2020.06.18.0925cdt JMS- Mouse(iMD) -> MChan(iMD*2-1) & MChan(iMD*2)
494 !--Globals
495  use ScreenDef,only: xWindowFull,yWindowFull,vScrnHVD & !screen & colors
496                     ,lCharX,lCharY
```

```
 use MouseDef ,only: Mouse,MChan,PpX,PpY !Mouse
 use SimDef   ,only: IterTotal,RunSecs   !Simulation control,F9-F12
!--End Globals
 implicit none
 !--Arguments
  integer(4)   ::md      !Mouse() raw data to use.                          [1:6]
 !--Internals
  integer(4)   ::ic1,ic2 !MChan() rescaled outputs.                        [1:12]

  !--EndDefs-------------------------------------------------
   if((md<1).or.(md>12))          stop'MouseRecUpdate: md outside [1:12]. Halt.'
         ic1    = 2*md-1                !Channel index    - X
         ic2    = 2*md                  !                 - Y
   PpX          = MChan(ic1)%Pp         !Channel          - X
   PpY          = MChan(ic2)%Pp         !                 - Y
   PpX(2)       = PpX(1)                !Previous<-Present- X
   PpY(2)       = PpY(1)                !                 - Y
   PpX(1)%iPix  = Mouse(md)%iX          ![0:ScreenPix]    - X
   PpY(1)%iPix  = Mouse(md)%iY          !                 - y
   PpX(1)%nChar = PpX(1)%iPix/lCharX+1  !Screen character - column
   PpY(1)%nChar = PpY(1)%iPix/lCharY+1  !                 - row
   PpX(1)%iUsed = PpX(1)%iUsed+1        !Iteration duration
   PpY(1)%iUsed = PpY(1)%iUsed+1
   PpX(1)%Iter  = IterTotal             !@iteration#_
   PpY(1)%Iter  = IterTotal
   PpX(1)%RunSecs= RunSecs              !@runtime_
   PpY(1)%RunSecs= RunSecs
   MChan(ic1)%mc  = ic1
   MChan(ic2)%mc  = ic2
   MChan(ic1)%Pp  = PpX; MChan(ic1)%mdParent=md; MChan(ic1)%iX1Y2=1
   MChan(ic2)%Pp  = PpY; MChan(ic2)%mdParent=md; MChan(ic2)%iX1Y2=2
   MChan(ic1)%r01 = Mouse(md)%iX/dble(xWindowFull)      !x          [0.:1.]
   MChan(ic2)%r01 = Mouse(md)%iY/dble(yWindowFull)      !y
   MChan(ic1)%rm11 =MChan(ic1)%r01*2.d0-1.d0            !x         [-1.:1.]
   MChan(ic2)%rm11 =MChan(ic2)%r01*2.d0-1.d0            !y
   MChan(ic1)%v    =(1.d0-MChan(ic1)%r01)*vScrnHVD(1,1) &!x  e.g.:[-1.6:1.6]
                    +MChan(ic1)%r01 *vScrnHVD(2,1)
   MChan(ic2)%v    =(1.d0-MChan(ic2)%r01)*vScrnHVD(1,2) &!y typical:[-1.:1.]
                    +MChan(ic2)%r01 *vScrnHVD(2,2)

   if(PpX(1)%RunSecs-PpX(2)%RunSecs<3.d0) then
    !--Less than 3 seconds elapsed, apply the mouse motion:
     MChan(ic1)%iDeltaPix=PpX(1)%iPix-PpX(2)%iPix
     MChan(ic2)%iDeltaPix=PpY(1)%iPix-PpY(2)%iPix
    else
     MChan(ic1)%iDeltaPix=0
     MChan(ic2)%iDeltaPix=0
   endif !PpX(1)%RunSecs-PpX(2)%RunSecs<3.
   return
End Subroutine MouseRecUpdate
!-------------------------------------------------------------------------------7 9

```