```
/*3DEnv.c
  3D-Environment, Version c0.6 by  Jeffrey M. Setterholm
  Written in C and realized using Silicon Graphics' OpenGL & GLUT.

  2016.07.10 JMS- Updated the 'SeeFrustums' mathematical description.
                  Added App F2: "Mandelbrot Set"
  2016.06.29 JMS- Added MmMenuView & BufferMmMenuView.
                  Added 'bB' and 'mM" controls to'cbUserView'
                  in case the Right-Mouse-Button doesn't work.
                  See 'Selfie-A9-MenuViews.jpg' for details.
                  BufferMmMenuView is displayed in 'HindSight'.
  2016.06.24 JMS- This is the primary C source code file.
  2013.01.17 JMS- Traveler2/Athlon64/WinXPPro/APF9.0:C/OpenGL+CGlut

 Herein:
int main(void)
int GlutHandoff()       <-- Initializes & launches OpenGL's GLUT interface.
void cbUserView(void)   <-- Provides overall simulation control; calls HindSight.
void DaTime(void)
void Colors3D(int nCol)
void ColorStd(int nCol)

//Manual controls:
void cbKeyboard(unsigned char Key,int xCursor,int yCursor)
void cbSpecialFunctionKeys(int Key,int xCursor,int yCursor)
void cbMouseMotion(int iX,int iY)
void cbMouseButtons(int Button, int State, int iX,int iY)
void cbMousePassiveMotion(int iX,int iY)
void MenuInit(void)
void menuAppsUse(int Value)
void menuSimUse(int Value)
void menuHelpUse(int Value)
void menuMainUse(int Value)

void ProjOrtho(int iOrthoHdr)  <-- Orthogonal screen & boilerplate info.
void PrntOrtho(int nRow,int mColumn,int iColorFG,int iColorBg,char PText[80])
void PAhXR( double Xyzhin[4], double Rpyhin[4],double hXRout[4][4]) <-- 6dof
void Alpha6D(double Pxyzh[4],double ArpyDh[4],int ModeFlag,             ...etc.
void ScreenSelfie(void)

//Homogeneous vector & matrix functions:
void h4Fill( double h4[],double h00, double h01, double h02, double h03)
void h44Fill( double h44[4][4],                                         ...etc.
void Print4(double hin[4],char LabelIn[80])
void Print44(double hin[4][4],char LabelIn[80])
void MMply44(double hout[4][4],double hin1[4][4],double hin2[4][4])
void Invert44(double hin[4][4],double hinverse[4][4],int iPrint, int iView)
void Seeh4d(double hin[4],int nRow,int mCol,int Color,char LabelIn[80])
void Seeh44d(double hin[4][4],int nRow,int mCol,int Color,char LabelIn[80])

void Teapot(void)
void CubeGrid(int nCol)
int Brk(int LineNum, int n,int m)
void HindSight(void)       <-- *** Visualization geometry is resolved here. ***
void SeeFrustums(void)
void VecText7D( double Pxyzh[4],double ArpyDh[4]                        ...etc.
void AppF8(void)
void AccessYourActiveApp(void)  <-- Assign your new app to an unused F1 - F7.

Global Variables:                                                          */
#include <3DEnv.h>

int main(void) /*v:0.5-------------------------------------------------------
  2016.06.24.1600cdt JMS- Main program entry point                          */
{                                                                        //Dec
 char Label[80];

 sprintf(S.Banner  ,"3DEnv version 0.6\0");
 sprintf(S.DateTimeExe  ,"2016.07.10\0");
 sprintf(S.Author       ,"Jeffrey M. Setterholm\0");
```

```
  sprintf(S.YourAppOutput,"3DEnv.txt\0");

  // Writing to the DOS screen:
  printf("%-21s  %s  %-s\n",S.Banner   ,S.DateTimeExe,S.Author);
                                         DaTime();
  printf("Runtime                %s\0",S.DaTimeLabel);
  printf("  ...&   beep\a\n");

  printf("Opening %-35s...as output\n",S.YourAppOutput);
  S.fpt=fopen(S.YourAppOutput,"w");
  if(S.fpt==NULL) { printf("\a"); printf(" -FAILED TO OPEN.\n"); S.fpt=stdout;}
  S.fp=S.fpt;
  if(S.fpt!=stdout)
   { fprintf(S.fpt,"%-21s  %-35s   %s\n",S.Banner,S.Author,S.DateTimeExe);
     fprintf(S.fpt,"Runtime   output--> %-35s %23s\n",
                     S.YourAppOutput,S.DaTimeLabel);
   }
     printf(     "\nControl -> OpenGL ...\n");
// fclose( S.fp);
          //S.fp=stdout;

  S.Init3dv=0;
  S.AppNumber=8;  // Begin by viewing App#F8.
// S.NowPrint=1; //This activates print->.txt during the first iter.
//fprintf(S.fp,"\nFirst iteration printout... running App#%1i:\0",S.AppNumber);
//fprintf(S.fp," ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");

  GlutHandoff();  //... which follows
  return 0;
}//End main ----------------------------------------------------------------------


int GlutHandoff() /*v:0.5-------------------------------------------------------
  2016.06.24.1600cdt JMS- Initialization of the GLUT callbacks & handoff:    */
{                                                                        //NoDec
  S.MenuNew   = 0; S.BrkLim    =  1;
  S.MouseInit = 0; S.MouseUse  =  1;
  S.iScrnColor=15; S.FovYZzoom =  1.e0;
  //    DtoR      = (double)asin(1.0)/((double) 90.e0);

  //glutInit(&argc, argv);
  S.xWindowFull  =glutGet(GLUT_SCREEN_WIDTH );
  S.yWindowFull  =glutGet(GLUT_SCREEN_HEIGHT);
  //Text->screen character placement parameters:
  S.lCharX       = 8;
  S.lCharY       =14;  // !=13 ... slightly increases line spacing
  S.nCharMaxX    =S.xWindowFull/S.lCharX;
  S.nCharMaxY    =S.yWindowFull/S.lCharY;
  S.nCharCenX    =S.nCharMaxX/2+2;
  S.xyWindowRatio=(float)S.xWindowFull/(float)S.yWindowFull;
  S.cWidth       =2.00/S.nCharMaxX;
  S.cHeight      =2.00/S.nCharMaxY;

  //Firing up OpenGL:
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
  glutInitWindowSize(S.xWindowFull,S.yWindowFull);
  glutCreateWindow("3DEnv");
  //GLUT Callbacks:
  glutKeyboardFunc(cbKeyboard);
  glutSpecialFunc(cbSpecialFunctionKeys);
  glutDisplayFunc(cbUserView);
  glutIdleFunc(cbUserView);
  glutMouseFunc( cbMouseButtons);
  glutMotionFunc(cbMouseMotion );
                 cbMouseMotion(S.xWindowFull/2,S.yWindowFull/2); //does init.
  glutPassiveMotionFunc(cbMousePassiveMotion);
  MenuInit(); // sets the stage for screen 'menuMainUse', ~ a callback
                                                    //openglinfo(); //2015.08.17
  //Initially select full screen mode:
  glutFullScreen(); S.iFullScreen=1; glFlush();//'Escape' key will toggle this.
```

```
                    S.SimModeNew    =0;
 glutMainLoop();  //OpenGL takes over for good... & calls the shots henceforth.
 return 0;
}//End GlutHandoff ------------------------------------------------------------


void cbUserView(void) /*v:0.5-------------------------------------------------
  2016.06.26.1400cdt JMS- Added MmMenuView & BufferMmMenuView.
                          Added 'bB' and 'mM" controls to'cbUserView'
                          in case the Right-Mouse-Button doesn't work.
                          See 'Selfie-A9-MenuViews.jpg' for details.
  2016.06.24.1600cdt JMS- CallBack- User(i.e.your) View generator
                       - Welcome to center stage!                          */
{ double xyzh[4],rpyh[4];
  double DtoR = (double)asin(1.0)/((double) 90.e0);
  int i,j;

 // Turns off the MenuViewer when another keyboard key is pressed:          //
 if(S.KbdKey>0 || S.ArrowKey>0)                      //Added 2016.06.26.1400cdt
  { if(S.KbdKey!=77 && S.KbdKey!=109 && S.KbdKey!=126) S.MmMenuView=0;}

 S.IterTotal=S.IterTotal+1;           //Increment the total iteration counter
 if((S.KbdKey==67)||(S.KbdKey==99))             //if: Keyboard Key='C' or 'c'
   {S.iScrnColor=16-S.iScrnColor; S.KbdKey=0;}  // Toggle screen color:
 if(S.iScrnColor==15) glClearColor(0.,0.,0.,0.); //                      black
 if(S.iScrnColor== 1) glClearColor(1.,1.,1.,0.); //                      white
 glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
 glEnable(GL_DEPTH_TEST);
 glFlush();
 //Set up an Orthographic projection:-
 if(S.SimMode==1) ProjOrtho(4); //Extra on-screen information in 'Reset' mode
         //IP information ^(5)^ moved to individual apps             2016.07.08
 if(S.SimMode >1) ProjOrtho(1); //Decluttered in 'Hold','Run',& 'Stop' modes


 //F1-F12 Function Key useages:
 //                S.MenuNew=0;
 if(S.SimMode < 1) S.SimMode=1;
 if(S.ArrowKey>0){if(S.ArrowKey<13){S.MenuNew=S.ArrowKey; S.ArrowKey=0;}}
 //   "menuMainUse()" can also make S.MenuNew>0
                                  S.AppNumberNew=  0      ;
 if(S.MenuNew >0) {if(S.MenuNew<9) {S.AppNumber   =S.MenuNew;
                                  S.AppNumberNew=S.MenuNew; S.MenuNew =0;}}
 //Simulation mode switching:
 switch(S.MenuNew)
  {case( 9): S.SimMode=1      ; S.MenuNew=0;                    //F9 Reset
             printf("F_9  Reset  \n");                    break;
   case(10): S.SimMode=2      ; S.MenuNew=0;                    //F10 Hold
             printf("F10   Hold  Timer=%7.2f\n",S.RunTimer);break;
   case(11): S.SimMode=3      ; S.MenuNew=0;                    //F11  Run
             printf("F11    Run  Timer=%7.2f\n",S.RunTimer);break;
   case(12): S.SimMode=4      ; S.MenuNew=0;                    //F12 Stop
             printf("F12   Stop  Timer=%7.2f\n",S.RunTimer);break;}
 //'Real Time' Simulation mode control:
 switch(S.SimMode)
  {case( 1): S.RunTimer=0.             ;                      //F9 Reset
             S.IterRun =0              ;                 break;
   case( 2):                           ;                 break;//F10 Hold
   case( 3): S.RunTimer=S.RunTimer+.01 ;                      //f11  Run
             S.IterRun =S.IterRun +  1 ;                 break;
   case( 4):                           ;                 break;//F12 Stop
  }
 if(S.Scale==0.e0) S.Scale=1.e0;

 //KbdKey controls:
 switch(S.KbdKey)
  {
   //HindSight mode control  S.VuMode=[-1,0,1,2,3,4]:
   case( 69): S.KbdKey=0;  S.VuMode=S.VuMode+1;   //'E':includes RawGL & L|R
           if(S.VuMode> 4) S.VuMode=        -1;            break;//'E'
```

```
    case(101): S.KbdKey=0;  S.VuMode=S.VuMode+1;   //'e':accesses fewer VuMode's
            if(S.VuMode> 3) S.VuMode=         0;            break;//'e'
    //ModelView's Point of Interest (PoI) control
    //  PoI +X is controlled by the "F" key : Forward
    case(  6): S.KbdKey=0; S.PoIX=0.e0;                     break;//'ctl-F'
    case( 70): S.KbdKey=0; S.PoIX=S.PoIX+.01e0;             break;//'F'Fwd
    case(102): S.KbdKey=0; S.PoIX=S.PoIX-.01e0;             break;//'f'back
    //  S.PoIh +Y is controlled by the "R" key : Right
    case( 18): S.KbdKey=0; S.PoIY=0.e0;                     break;//'ctl-R'
    case( 82): S.KbdKey=0; S.PoIY=S.PoIY+.01e0;             break;//'R'Right
    case(114): S.KbdKey=0; S.PoIY=S.PoIY-.01e0;             break;//'r' left
    //  S.PoIh +Z is controlled by the "D" key : Down
    case(  4): S.KbdKey=0; S.PoIZ=0.e0;                     break;//'ctl-D'
    case( 68): S.KbdKey=0; S.PoIZ=S.PoIZ+.01e0;             break;//'D'Down
    case(100): S.KbdKey=0; S.PoIZ=S.PoIZ-.01e0;             break;//'d'up
    //Scale control:
    case( 19): S.KbdKey=0; S.Scale=1.e0;                    break;//'ctl-z'
    case( 83): S.KbdKey=0; S.Scale=1.01e0*S.Scale;                //'Z'
               if(fabs(S.Scale-1.e0)<.005e0) S.Scale=1.e0;  break;
    case(115): S.KbdKey=0; S.Scale= .99e0*S.Scale;                //'z'
               if(fabs(S.Scale-1.e0)<.005e0) S.Scale=1.e0;  break;
    //Model roll:
    case(44):
    case(60): S.KbdKey=0;  S.KbdRoll=S.KbdRoll-1.e0;        break;//',<'
    case(46):
    case(62): S.KbdKey=0;  S.KbdRoll=S.KbdRoll+1.e0;        break;//'.>'

    //Keyboard controls added: 2016.06.06.1400cdt                                  //
    //S.BrkOn: Toggles the Breaker viewer using "B" or "b"                         //
    case( 66):                                                  //'B'              //
    case( 98): S.KbdKey=0;                                      //'b'              //
     if(       S.BrkOn==0) S.BrkOn=1;                                             //
     else                  S.BrkOn=0;                           break;             //
                                                                                   //
    //S.MmMenuView: Toggles 'viewing' menu information "M" or "m"                  //
    //              Use this when your right mouse button does nothing.            //
    case( 77):                                                  //'M'              //
    case(109): S.KbdKey=0;                                      //'m'              //
     if(      S.MmMenuView==0) S.MmMenuView=1;                                     //
     else                      S.MmMenuView=0;                  break;             //

    //S.NowView: Toggles 'viewing' details on the screen using "v" or "V"
    case( 86): S.KbdKey=0;                                      //'V'
     if(      S.NowView==0) S.NowView=2;
     else if( S.NowView==1) S.NowView=2;
     else                   S.NowView=0;                        break;
    case(118): S.KbdKey=0;                                      //'v'
     if(      S.NowView==0) S.NowView=1;
     else if( S.NowView==2) S.NowView=1;
     else                   S.NowView=0;                        break;

    //S.NowPrint: One cycle Printout controlled by "p" or "P"
    case( 80): if(S.KbdKey== 80){S.fp=stdout;  //to screen      //'P'
                                 S.NowPrint=2;}
    case(112): if(S.KbdKey==112){S.fp=S.fpt;   //to your .txt file//'p'
                                 S.NowPrint=1;}
                                 S.KbdKey=0; printf("\a");
     if(S.fp!=NULL)
      {DaTime();
        fprintf(S.fp,"\nUser-requested printout running App#%1i\0",S.AppNumber);
        fprintf(S.fp,": ~~~~~~~~~~~~~~~~~~~~%19s\n",S.DaTimeLabel);
        if(S.RunTimer>0.e0)
        fprintf(S.fp,"Timer=%11.2f\n",S.RunTimer);
      }                                                         break;

    //GLUT Teapot viewing control by the "T" key:
    case( 20): S.KbdKey=0;  S.iTeapot=        0;            break;//'ctl-t'->0
    case( 84):                                                  //'T'
    case(116): S.KbdKey=0;  S.iTeapot=S.iTeapot+1;              //'t'
            if(S.iTeapot>7) S.iTeapot=          0 ;         break;
```

```
    //Nutation model view- toggle:                          using 'n'or'N'
    case( 78):                                                      //'N'
    case(110): S.KbdKey=0;  S.Nutate=1-S.Nutate;              break;//'n'
   }/*(S.KbdKey)*/

  //Update  model nutation:
  if(!S.Nutate) S.NutateAng=0.e0;
  if( S.Nutate)
   {   S.NutateAng = S.NutateAng+DtoR/1.e0;
    if(S.NutateAng>= 360.e0/DtoR) S.NutateAng=0.e0;}
  h4Fill(  xyzh   , 0.e0,   0.e0, 0.e0                  , 1.e0);
  h4Fill(S.Nutate4, 0.e0,   5.e0*cos(S.NutateAng)
                         ,  10.e0*sin(S.NutateAng)     , 1.e0);
  PAhXR(xyzh,S.Nutate4,S.Nutate44);

  //Transfer ModelView control parameters to S.HVu:
     S.PoIX=S.PoIX; //S.KbdRoll   =      0.e0  ;
     S.PoIY=S.PoIY; //S.MousePitch=S.rMouse[0] ;
     S.PoIZ=S.PoIZ; //S.MouseYaw  =S.rMouse[1] ;
     S.W2  =1.e0  ;   S.W1        =      1.e0  ;
  /*Other ArrowKey controls:*/
  switch(S.ArrowKey)
   {//Zero the S.PoIh using the "Home" key:
    case(106): S.ArrowKey=0;S.KbdRoll=0.e0;
               S.PoIX=0.e0 ;S.PoIY=0.e0;S.PoIZ=0.e0;S.Scale=1.e0; break;//Home
   }/*(S.ArrowKey)*/

  /* Call HindSight:  ...which in turn will exercise the App.'s */
    S.ThreePhase=1; HindSight();  //= 1 Raw initialization
    S.ThreePhase=2; HindSight();  //= 2 Update variables display 2D information
    S.ThreePhase=3; HindSight();  //= 3 Display the app.'s modelview graphics.
    S.ThreePhase=0;

  // Results -> screen:

  glutSwapBuffers(); glutPostRedisplay();  glFlush();

  //use "~" Tilda key to dump the full screen to SelfieN.bmp N=[0,1,...9]
  if(S.KbdKey==126) {ScreenSelfie(); S.KbdKey=0;}

  //use "`" key to toggle the screen selfie:
  if(S.KbdKey==96) {S.DepthSelfie=S.DepthSelfie+1; S.KbdKey=0;}

  //Clear the one-cycle 'NowPrint':
  if(S.NowPrint==1) S.fpt=S.fp;
     S.NowPrint= 0; S.fp=S.fpt; //defaulting to YourAppOutput.txt outputs.

}//End cbUserView ------------------------------------------------------------


 void DaTime(void) /*v:0.5-------------------------------------------------------
   2016.06.24.1600cdt JMS- Present date & time string -> S.DaTimeLabel [20].
                           Ref "C in a Nutshell" ** p.433 ** & also p.345     */
 {                                                                        //Dec
  time_t  rawtime;
  struct tm  local_tm,*ptr_tm;
                                time(&rawtime); // Get current time.
                     ptr_tm=localtime(&rawtime );
  memcpy( &local_tm, ptr_tm,sizeof(struct tm) );
  if( strftime( S.DaTimeLabel,sizeof(S.DaTimeLabel),
    "%Y.%m.%d.%H%M:%Z\0"   ,&local_tm) );
 }//End cbUserView ------------------------------------------------------------


 void Colors3D(int nCol)/*-------------------------------------------------------
   2016.06.24.1600cdt JMS- The 14 color palette presently in use in 3DEnv.exe
   2013.01.17.0815cst JMS- Traveler2/Athlon64/WinXPPro/APF9.0:C/OpenGL+CGlut
 Setup your color choices for a Black background screen.
 colors are modified here for viewing on a white background, or for Red/Cyan 3D.
```

```
A baseline of a few robust, vibrant drawing colors.

                  Cyan = 'not  Red'  = !Red
                Magenta = 'not Green' = !Green
                 Yellow = 'not  Blue' = !Blue

Custom character usages:
  My colors     [c][0] 'move without draw' - a no-op in this context
                        (clarifies & simplifies the content of .3dv files.)

                       IndCol[16][3]      IndColOnWhite[16][3]
Background:            Define on Black    Modified for White   |   Numeric
                       Red Green Blue       Red Green Blue     | Brightness
        White   [c][1] {255,  255, 255}     {  0,    0,   0}   |     0: none
  light Red        [2] {255,  159, 159}     {159,    0,   0}   |    63: 1/4
        Red        [3] {255,    0,   0}        no change       |    63: 1/4
!Green= Magenta    [4] {255,    0, 255}     {159,    0, 159}   |    95: 1/3
!Blue = Yellow     [5] {255,  255,   0}     {159,  159,   0}   |   127: 1/2
  light Green      [6] {159,  255, 159}     {  0,  159,   0}   |   159: 2/3
        Green      [7] {  0,  255,   0}        no change       |   191: 3/4
!Red  = Cyan       [8] {  0,  255, 255}     {  0,  159, 159}   |   255: full
  light Blue       [9] {159,  159, 255}     {  0,    0, 159}
        Blue       [a] {  0,    0, 255}        no change
  light Gray       [b] {191,  191, 191}     { 95,   95,  95}
        Gray       [c] {127,  127, 127}        no change
        Brown      [d] {255,  127,   0}     {159,  127,   0}
        Purple     [e] {255,  127, 255}     {159,   95, 159}
!White  Black      [f] {  0,    0,   0}     {255,  255, 255}                */

{
    /*       0 <begin line> 1 White        2~LtRed        3 Red
             4 Magenta      5 Yellow       6 LtGreen      7 Green
             8 Cyan         9 LtBlue      10 Blue        11 LtGray
            12 Gray        13 Brown       14 Purple      15 Black            */
int static IndCol[16][3]=
   {   //Red Green Blue , Red Green Blue , Red Green Blue , Red Green Blue
        {  0,    0,   0},{255,  255, 255},{255,   95,  95},{255,    0,   0}
       ,{255,    0, 255},{255,  255,   0},{159,  255, 159},{  0,  255,   0}
       ,{  0,  255, 255},{159,  159, 255},{  0,    0, 255},{191,  191, 191}
       ,{127,  127, 127},{255,  127,   0},{255,  127, 255},{  0,    0,   0} };
int static IndColOnWhite[16][3]=
   {    {  0,    0,   0},{  0,    0,   0},{191,    0,   0},{255,    0,   0}
       ,{159,    0, 159},{159,  159,   0},{  0,  159,   0},{  0,  255,   0}
       ,{  0,  159, 159},{  0,    0, 159},{  0,    0, 255},{ 95,   95,  95}
       ,{127,  127, 127},{159,  127,   0},{159,   95, 159},{255,  255, 255} };

/*Color Comparisons:                                                 approx.
 DOS# 'Colors3D' Red Grn Blu    'DOS'colors       Red Grn Blu   3D#    DXF#
   0<- 0 BeginLn{ na, na ,na}    0 gxBLACK       {  0,  0,  0}  =15~0  =  0
  15=  1 White  {255,255,255}    1 gxBLUE        {  0,  0,255}  =10    =  5
  12=  2 LtRed  {255,159,159}    2 gxGREEN       {  0,255,  0}  = 7    =  3
   4=  3 Red    {255,  0,  0}    3 gxCYAN        {  0,255,255}  = 8    =  4
   5=  4 Magenta{255,  0,255}    4 gxRED         {255,  0,  0}  = 3    =  1
  14=  5 Yellow {255,255,  0}    5 gxMAGENTA     {255,  0,255}  = 4    =  6
  10   6 LtGreen{159,255,159}    6 gxBROWN       {170, 85,  0}   13    = 62
   2=  7 Green  {  0,255,  0}    7 gxGRAY        {170,170,170}   11    =  9
   3=  8 Cyan   {  0,255,255}    8 gxDARKGRAY    { 85, 85, 85}   12    =  8
   9   9 LtBlue {159,159,255}    9 gxLIGHTBLUE   {127,127,255}    9    =161
   1= 10 Blue   {  0,  0,255}   10 gxLIGHTGREEN  {127,255,127}    6    = 81
   7  11 LtGray {191,191,191}   11 gxLIGHTCYAN   {127,255,255} X->8    =131
   8  12 Gray   {127,127,127}   12 gxLIGHTRED    {255,127,127}  = 2    = 11
   6  13 Brown  {255,127,  0}   13 gxLIGHTMAGENTA{255,127,255}  =14    =211
  13= 14 Purple {255,127,255}   14 gxYELLOW      {255,255,  0}  = 5    =  2
   0= 15 Black  {  0,  0,  0}   15 gxWHITE       {255,255,255}  = 1    =  7
                                                  No color passed = -1
                                                  ACAD decides    =255  */
//float ColorRGBA[4];  //={Red,Green,Blue,Alpha};

/* ----------------------------------
In version 0.52 The following color index conversions are located in VecText.h:
```

```
 //Converts DOS colors to Colors3D colors:  (...interprets black as 'move')
 int static ColorsDOSto3D[16]={15,10,7,8,3,4,13,11,12,9,6,8,2,14,5,1};
 //...for importing DOS colors in .3dv files.

 //Converts Colors3D colors to DOS colors:  (...interprets 'move' as black)
 int static Colors3DtoDOS[16]={0,15,12,4,5,14,10,2,3,9,1,7,8,6,13,0};
 //...for exporting DOS colors to .3dv files.

 //DOS -to- ACAD R12 Color conversion table (my wag-to-wag):
 //     Reference http://sub-atomic.com/~moses/acadcolors.html
 //Converts DOS colors to ACAD R12 colors:
 int static ColorsDOStoDXFR12[16]={0,5,3,4,1,6,62,9,8,161,81,131,11,211,2,7};
 //...for exporting ACADR12 colors to .dxf files.

-----------------------------------*/

 int n;                                                                  //DecEnd

 // S.ForceRGBA<0, causes your color 'S.ColorRGBA[4]' to be used:
 if(S.ForceRGBA<0) {glColor4fv(S.ColorRGBA); glFlush();                  return;}

                    n=nCol;
 // S.ForceRGBA>0 is used by Red/Cyan 3D for color override:
 if(S.ForceRGBA>0) n=S.ForceRGBA;
 if(n< 1 || n>15)  n=13;          //Default to Brown.

 //Red/Cyan color swap for White screen:
 if(S.ForceRGBA>0 && S.iScrnColor==15)
  { switch(n) { case(2): n=8; break;   // Red -> Cyan
                case(7): n=3; break; } //Cyan -> Red
  }
 if(S.iScrnColor==15){ S.ColorRGBA[0]=  IndCol[       n][0]/255.0;   //Red
                       S.ColorRGBA[1]=  IndCol[       n][1]/255.0;   //Green
                       S.ColorRGBA[2]=  IndCol[       n][2]/255.0;   //Blue
                       S.ColorRGBA[3]=     1.0            ; } //Alpha

 if(S.iScrnColor== 1){ S.ColorRGBA[0]=  IndColOnWhite[n][0]/255.0;   //Red
                       S.ColorRGBA[1]=  IndColOnWhite[n][1]/255.0;   //Green
                       S.ColorRGBA[2]=  IndColOnWhite[n][2]/255.0;   //Blue
                       S.ColorRGBA[3]=     1.0            ; } //Alpha

 glColor4fv(S.ColorRGBA);                                                glFlush();

}//End Colors3D ------------------------------------------------------------


void cbKeyboard(unsigned char Key,int xCursor,int yCursor) /*v:0.5-------------
  2016.06.24.1600cdt JMS- CallBack- keyboard:                                */
{int xScreen,yScreen;                                                    //Dec

 if(Key>0){S.KbdKey=Key;     //Save the key value for use elsewhere
         //printf("Key pressed: %i  ",Key);
          }
 switch(Key)
  {case(17): S.KbdKey=0; fprintf(stderr,"ctl-Q.\n"); exit(1);  //'ctl-Q'

   case(27): S.KbdKey=0;                                          //escape
     switch(S.iFullScreen)                                        //screen toggle
      {case(0): glutFullScreen()                  ; glFlush();  //fullscreen
                printf("Used Escape key.\n"); S.iFullScreen=1         ;return;
       case(1): xScreen=400; yScreen=200;
                glutReshapeWindow(xScreen,yScreen); glFlush();  //small window
                printf("Used Escape key.\n"); S.iFullScreen=0         ;return;
      }
   case  81:                                                     //'q'
   case 113: S.KbdKey=0; //fclose(S.fp);                         //'Q'
                         //fprintf(stderr,"Qq\n");      exit(1);
    DaTime();
  //if(S.fpt!=NULL)
```

```
   // { if(S.RunTimer>0.e0) fprintf(S.fpt,"\nTimer=%6.2f seconds\n",S.RunTimer);
   //   fprintf(S.fpt,"Quit (via the keyboard)%56s\n",S.DaTimeLabel);
   //   fclose( S.fpt);
   // }
     if(S.RunTimer>0.e0) fprintf(S.fpt,"\nTimer=%6.2f seconds\n",S.RunTimer);
     fprintf(S.fpt,"\nQuit (via the keyboard)%56s\n",S.DaTimeLabel);
     fclose( S.fpt);
     printf("\nQuit (via keyboard)\n");
     printf("%-21s  %s  ...done.\n",S.Banner,S.DaTimeLabel);
     if(S.iFullScreen==0) // <-- Intent: DOS window persists on exit.
      { printf("\nLeft-click on the DOS window & press 'enter' to exit.\0");
        getchar();
      }
     exit(0);                                                          return;

  //default: printf(" ...unassigned key index: %1c\n",(int)S.KbdKey)   ;return;
   }
}//End cbKeyboard ------------------------------------------------------------


void cbSpecialFunctionKeys(int Key,int xCursor,int yCursor) /*v:0.5------------
  CallBack- special function keys (- F1-F12, arrows, etc.):
                         -F9...F12 -> Reset,Hold,Operate,& Stop the timer.    */
{                                                                        //NoDec
 S.ArrowKey=Key;             //Save the "arrow keys" value for use elsewhere
 //if(Key<  9)printf("Arrow key pressed: %i\n",Key);
 //if(Key> 12)printf("Arrow key pressed: %i\n",Key);
}//End cbSpecialFunctionKeys -------------------------------------------------


void cbMouseMotion(int iX,int iY) /*v:0.5-----------------------------------
  2016.06.24.1600cdt JMS- CallBack- mouse motion:                          */
{int    n, m;                                                             //Dec
 long   MouseXcPrev,MouseYcPrev;
 double B[4][4],cp,cr,cy,sp,sr,sy,dRpy[3];                               //DecEnd
 double static DtoR = (double)asin(1.0)/((double) 90.e0);

 if(    S.MouseLmbWasPressed==0 )
       {S.MouseLmbWasPressed= 1;
        glutWarpPointer(S.Mouse[1].iX   , S.Mouse[1].iY    );   }/*( ==0)*/
  else {               S.Mouse[1].iX=iX; S.Mouse[1].iY=iY  ;   }/*(~==1)*/

 S.MouseXc=S.Mouse[1].iX; //=iX
 S.MouseYc=S.Mouse[1].iY; //=iY

 S.rMouse[0]= (S.MouseXc+.5e0)/S.xWindowFull;       //[   0.e0 ,       1.e0   ]
 S.rMouse[1]= (S.MouseYc+.5e0)/S.yWindowFull;       //[   0.e0 ,       1.e0   ]

 S.vMouse[0]= (1.e0-S.rMouse[0])*S.vScrnHVD[0][0]
             +(    S.rMouse[0])*S.vScrnHVD[1][0];
                                                    //[  -1.e0 ,       1.e0   ]
 S.vMouse[1]= (     S.rMouse[1])*S.vScrnHVD[0][1]
            +(1.e0-S.rMouse[1])*S.vScrnHVD[1][1];
 S.vMouse[2]=  0.e0;
 S.vMouse[3]=  1.e0;  //homogeneous vector scale factor.

  S.MousePitch= (S.MouseYc-S.yWindowFull/2.e0)/2.e0;
  S.MouseYaw  = (S.MouseXc-S.xWindowFull/2.e0)/2.e0;

 //The following code creates a 4x4 homogeneous matrix controlling Pitch & Yaw
 //    using the mouse only - which might be useful in an austere application.
 //       "Define the model's rotation:   (uses mouse & '<''>' inputs)"
 //       is the place where the matrix used by "HindSight" is generated.
                                    dRpy[0]= 0.e0;  //=DtoR*S.KbdRoll;
                                    dRpy[1]= DtoR*S.MousePitch;
                                    dRpy[2]= DtoR*S.MouseYaw  ;
 sr=sin(dRpy[0]); sp=sin(dRpy[1]); sy=sin(dRpy[2]);
 cr=cos(dRpy[0]); cp=cos(dRpy[1]); cy=cos(dRpy[2]);
 B[0][0]=       cy*cp   ; B[0][1]=       cy*cp   ; B[0][2]=-sp   ; B[0][3]= 0.;
 B[1][0]=-sy*cr+cy*sp*sr; B[1][1]= cy*cr+sy*sp*sr; B[1][2]= cp*sr; B[1][3]= 0.;
```

```
  B[2][0]= sy*sr+cy*sp*cr; B[2][1]=-cy*sr+sy*sp*cr; B[2][2]= cp*cr; B[2][3]= 0.;
  B[3][0]= 0.            ; B[3][1]= 0.            ; B[3][2]= 0.   ; B[3][3]= 1.;
  for(n=0; n<4; n++){for(m=0; m<4; m++){S.Bmouse[n][m]=B[n][m];}}
  //hInverse(*S.Bmouse[0][0][0],*S.Bmouse[0][0][1]);
   glutSetCursor(GLUT_CURSOR_INFO);
 //glutSetCursor(GLUT_CURSOR_CROSSHAIR);

 }//End cbMouseMotion ------------------------------------------------------


 void cbMouseButtons(int Button, int State, int iX,int iY) /*v:0.5--------------
   2016.06.24.1600cdt JMS- CallBack- mouse "button press":                    */
 {int iXLoc,iYLoc,JustOne=1;                                              //Dec
  float PixelDepth[1][1];                                                //DecEnd

  if( GLUT_DOWN) {S.MouseLmbWasPressed=0;
                  cbMouseMotion(       iX,iY);}
  if(!GLUT_DOWN) {S.MouseLmbWasPressed=1;
                  cbMousePassiveMotion(iX,iY);}
 }//End cbMouseButtons ------------------------------------------------------


 void cbMousePassiveMotion(int iX,int iY) /*v:0.5-----------------------------
   2016.06.24.1600cdt JMS- CallBack- mouse motion "without button press":
                          - Reads screen depth at cursor location.            */
 {//int iXLoc,iYLoc; //,JustOne=1;                                         //Dec
  GLint iXLoc,iYLoc;
  GLsizei JustOne=1;
  float PixelDepth[1][1];

  if(S.MouseInit==0)
   {S.Mouse[0].iX =S.xWindowFull/2;
    S.Mouse[0].iY =S.yWindowFull/2;
    S.Mouse[1].iX =S.xWindowFull/2;
    S.Mouse[1].iY =S.yWindowFull/2;
    S.MouseInit= 1; }/*(S.MouseInit==0)*/

  if(    S.MouseLmbWasPressed==1 )
        {S.MouseLmbWasPressed= 0;
     glutWarpPointer(S.Mouse[0].iX    , S.Mouse[0].iY    );   }
   else {             S.Mouse[0].iX=iX; S.Mouse[0].iY=iY  ;   }

  S.MouseXc=S.Mouse[0].iX; //=iX
  S.MouseYc=S.Mouse[0].iY; //=iY

    if(S.MouseInit==0) S.MouseInit=1;

  //Ref: Wright&Sweet P.384                                          2016.04.28
  //WINGDIAPI void APIENTRY glReadPixels (GLint x, GLint y, GLsizei width
  //        , GLsizei height, GLenum format, GLenum type, GLvoid *pixels);
  iXLoc=                S.MouseXc;
  iYLoc= S.yWindowFull-1-S.MouseYc;
     glReadPixels( iXLoc,iYLoc,JustOne,JustOne,
                   GL_DEPTH_COMPONENT,GL_FLOAT,&(PixelDepth));
  S.MouseScreenh[0]= (2.*iXLoc-S.xWindowFull)/S.xWindowFull;
  S.MouseScreenh[1]= (2.*iYLoc-S.yWindowFull)/S.yWindowFull;
  S.MouseScreenh[2]= PixelDepth[0][0];
  S.MouseScreenh[3]=          1.0;
   glutSetCursor(GLUT_CURSOR_CROSSHAIR);
 //glutSetCursor(GLUT_CURSOR_INFO);
 }//End cbMousePassiveMotion -----------------------------------------------


 void MenuInit(void) /*v:0.5----------------------------------------------------
   2016.06.28.0715cdt JMS- Menu Initialization, SubMenu access incl Keyboard  */
 {int menuMainId,menuAppsId,menuSimId,menuHelpId;                         //Dec

  menuAppsId = glutCreateMenu(menuAppsUse);
  //App-specific:
    glutAddMenuEntry("    Access Apps:   \0",71);
```

```
    glutAddMenuEntry("F1 Ortho Projection\0", 1);
    glutAddMenuEntry("F2 ...             \0", 2);
    glutAddMenuEntry("F3 ...             \0", 3);
    glutAddMenuEntry("F4 VecText7D Demo  \0", 4);
    glutAddMenuEntry("F5 3dv-Viewer      \0", 5);
    glutAddMenuEntry("F6 Invert44-Demo   \0", 6);
    glutAddMenuEntry("F7 ...             \0", 7);
    glutAddMenuEntry("F8 HindSight Demo  \0", 8);
    glutAddMenuEntry("     ^the 3D viewer\0",72);

  menuSimId = glutCreateMenu(menuSimUse);
  //Simulation control:
    glutAddMenuEntry("    Simulation Control:\0",73);
    glutAddMenuEntry("F9   Reset             \0", 9);
    glutAddMenuEntry("F10     Hold           \0",10);
    glutAddMenuEntry("F11        Run         \0",11);
    glutAddMenuEntry("F12     Stop           \0",12);

  menuHelpId = glutCreateMenu(menuHelpUse);
  //Keyboard & Mouse controls:
     glutAddMenuEntry("    ...via Keyboard only:\0",75);
    glutAddMenuEntry("eE  Eye Mode             \0",24);
    glutAddMenuEntry("fF  -+X  forward/ aft    \0",88);
    glutAddMenuEntry("rR  -+Y     right  /left\0",89);
    glutAddMenuEntry("dD  -+Z    down / up     \0",90);
    glutAddMenuEntry("sS  -+Scale down/up      \0",91);
    glutAddMenuEntry("<>  -+model roll angle   \0",99);
    glutAddMenuEntry("Home: resets X,Y,Z,&Scale\0",92);
    glutAddMenuEntry("nN  model Nutation       \0",93);
    glutAddMenuEntry("tT  Teapot      (0->7)   \0",93);
    glutAddMenuEntry("-+  Breaker index  control\0",94);
    glutAddMenuEntry("                         \0",76);
    glutAddMenuEntry("    ...via Mouse only:   \0",96);
    glutAddMenuEntry("    Yaw,Pitch    (LMB down)\0",97);
    glutAddMenuEntry("    Cursor use (LMB    up  )\0",98);

  menuMainId    = glutCreateMenu(menuMainUse);
  //AppNumber's access:
    glutAddSubMenu(  "Applications\0"             ,menuAppsId);
  //Simulation control access:
    glutAddSubMenu(  "Simulation Ctl.\0"          ,menuSimId );
    glutAddSubMenu(  "Help- Keyboard & Mouse\0"   ,menuHelpId);
    glutAddMenuEntry("                          \0",77);
    glutAddMenuEntry("`   Depth Selfie-> screen\0",15);
    glutAddMenuEntry("  ~           Selfie -> .bmp\0",14);
    glutAddMenuEntry("                          \0",74);
    glutAddMenuEntry("cC  ScreenColor           \0",21);
    glutAddMenuEntry("p   Print to - .txt file \0",25);
    glutAddMenuEntry("  P               - DOS screen\0",26);
    glutAddMenuEntry("vV  Viewing info.  0-1-2 \0",27);
    glutAddMenuEntry("bB  Breaker viewer toggle\0",22);     //2016.06.26.1130cdt
    glutAddMenuEntry("mM  Keyboard Menu toggle \0",28);     //2016.06.29.0700cdt
    glutAddMenuEntry("                          \0",78);
    glutAddMenuEntry("qQ  ***Quit***            \0",13);
    glutAddMenuEntry("                          \0",789);

  glutAttachMenu(GLUT_RIGHT_BUTTON);  //This activates the menu via g_r_b.
 }//End MenuInit ------------------------------------------------------------

 void menuAppsUse(int Value) /*v:0.5-------------------------------------------
   2016.06.24.1600cdt JMS- SubMenu use:                                      */
 {
  switch(Value)
   {//Application access... via function keys F1-F8 or via the menu alone:
    case( 1): S.MenuNew= 1; break; //F1  Lines & Text
    case( 2): S.MenuNew= 2; break; //F2  2D/3D Exerciser
    case( 3): S.MenuNew= 3; break; //F3  OverWriter
    case( 4): S.MenuNew= 4; break; //F4  Generic/unused
    case( 5): S.MenuNew= 5; break; //F5  Generic/unused
    case( 6): S.MenuNew= 6; break; //F6  Generic/unused
```

```
    case( 7): S.MenuNew= 7; break; //F7  Generic/unused
    case( 8): S.MenuNew= 8; break; //F8  Generic/unused
   }/*(Value)*/
 }//end menuAppsUse


 void menuSimUse(int Value) /*v:0.5-------------------------------------------
   2016.06.24.1600cdt JMS- SubMenu use:                                      */
 {
  switch(Value)
   {//Simulation control... via function keys F9-F12:
    case( 9): S.MenuNew= 9               ; break; //F9   Reset
    case(10): S.MenuNew=10               ; break; //F10   Hold
    case(11): S.MenuNew=11               ; break; //F11    Oper8
    case(12): S.MenuNew=12               ; break; //F12   Stop

   }/*(Value)*/
 }//end menuSimUse


 void menuHelpUse(int Value) /*v:0.5-------------------------------------------
   2016.06.24.1600cdt JMS- SubMenu use:                                      */
 {
 }//end menuHelpUse


 void menuMainUse(int Value) /*v:0.5-------------------------------------------
   2016.06.28.0715cdt JMS- Menu use:                                         */
 {                                                                          //Dec
  switch(Value)
   {   //i/o control:
    case(13): S.MenuNew=13; // **Quit**
     DaTime();
     if(S.fpt!=NULL)
      { if(S.RunTimer>0.e0) fprintf(S.fpt,"\nTimer=%6.2f seconds\n",S.RunTimer);
        fprintf(S.fpt,"Quit (via the menu)    %56s\n",S.DaTimeLabel);
         fclose(S.fpt);
      }
     printf("Quit (via the menu)\n");
     printf("%-21s  %s  ...done.\n",S.Banner,S.DaTimeLabel);
     if(S.iFullScreen==0)
      { printf("\nLeft-click on the DOS window & press 'enter' to exit.\0");
        getchar();       // <-- Intent: DOS window persists on exit.
      }
     exit(0);                                                                return;

    case(15): S.DepthSelfie=S.DepthSelfie+1 ;break;  // DepthSnapshot Toggle
    case(21): S.iScrnColor =16-S.iScrnColor  ;break;  // ChangeScreen

    case(22): S.BrkOn=1-S.BrkOn             ;break;  //F12 'Breaker' viewer

       //One cycle Printout- like using keyboard key "pP" in cbUserView:
    case(25): if(Value==25){S.fp=S.fpt ;S.NowPrint=1;}  //to your .txt file//'p'
    case(26): if(Value==26){S.fp=stdout;S.NowPrint=2;}  //to screen        //'P'
    DaTime();
    fprintf(S.fp,"\nUser-requested printout running App#%1i\0",S.AppNumber);
    fprintf(S.fp,": ~~~~~~~~~~~~~~~~~~~~%19s\n",S.DaTimeLabel);
    if(S.RunTimer>0.e0) fprintf(S.fp,"Timer=%11.2f\n",S.RunTimer);        break;

    case(27): if( S.NowView==0) S.NowView=1;              //Viewing info toggle
         else if( S.NowView==1) S.NowView=2;
         else                   S.NowView=0;                                 break;

    case(28): S.MmMenuView=1-S.MmMenuView; break;         //KeyBoard's menu

   }/*(Value)*/
 }//End menuMainUse ------------------------------------------------------------


 void ProjOrtho(int iOrthoHdr) /*v:0.5-----------------------------------------
```

```
   2016.06.24.1600cdt JMS- Setup an orthographic projection:
                          ...includes screen boilerplate.
   OpenGL Screen:  +X:right,+Y:up,+Z:into screen (left-hand coordinates)      */
{int nRow,nCol,Pass;                                                          //Dec
 char *cPointer, PText[80];
 float XScreenRow,YScreenCol,Zdepth,DtoR = (float)asin(1.0)/(90.);
 double Xyzh[4],Rpyh[4];

 //Set up an orthographic projection:
 glMatrixMode(GL_TEXTURE   ); glLoadIdentity();
 glMatrixMode(GL_MODELVIEW ); glLoadIdentity();
 glMatrixMode(GL_PROJECTION); glLoadIdentity();
 glRasterPos3d( 0.e0,0.e0,0.e0);                                      glFlush();

 glMatrixMode(GL_PROJECTION);
 //Setup a normalized orthographic screen:
 S.vScrnHVD[0][0]=-S.xyWindowRatio; S.vScrnHVD[1][0]=-S.vScrnHVD[0][0];
 S.vScrnHVD[0][1]=-1.000;          ; S.vScrnHVD[1][1]= 1.000;
 S.vScrnHVD[0][2]= 1.000;          ; S.vScrnHVD[1][2]=-1.000;
 glOrtho(S.vScrnHVD[0][0],S.vScrnHVD[1][0]    //  Left-to-right
        ,S.vScrnHVD[0][1],S.vScrnHVD[1][1]    //bottom-to-top
        ,S.vScrnHVD[0][2],S.vScrnHVD[1][2] ); //  near-to-far
 glMatrixMode(GL_MODELVIEW ); glLoadIdentity();glFlush();
                                                      if(iOrthoHdr<1)return;
 //Extra on-screen information begins here:
 // At upper right of screen:
                            nRow = 1;              nCol = S.nCharMaxX-21;
 cPointer=strncpy(PText,S.Banner,sizeof(S.Banner));
                                    nRow++; PrntOrtho(nRow,nCol, 1, 0,PText);
 cPointer=strncpy(PText,S.DateTimeExe,sizeof(S.DateTimeExe));
                                    nRow++; PrntOrtho(nRow,nCol, 1, 0,PText);
 DaTime(); cPointer=strncpy(PText,S.DaTimeLabel,sizeof(S.DaTimeLabel));
                                    nRow++; PrntOrtho(nRow,nCol, 1, 0,PText);
 switch(S.SimMode)
 {case(0):sprintf(PText,"       Timer undefined\0"       ); break;
  case(1):sprintf(PText,"Reset  Timer=%8.2f\0",S.RunTimer); break;
  case(2):sprintf(PText," Hold  Timer=%8.2f\0",S.RunTimer); break;
  case(3):sprintf(PText,"  Run  Timer=%8.2f\0",S.RunTimer); break;
  case(4):sprintf(PText," Stop  Timer=%8.2f\0",S.RunTimer); break;
  case(5):sprintf(PText," Drop  Timer=%8.2f\0",S.RunTimer); break;
 }                                  nRow++; PrntOrtho(nRow,nCol, 1, 0,PText);
 sprintf(PText,"AppNumber =  F:%1.1i\0",S.AppNumber);
                                    nRow++; PrntOrtho(nRow,nCol, 5, 0,PText);
 sprintf(PText,"Qq:->QUIT,   RMB:menu\0 ");
                                    nRow++; PrntOrtho(nRow,nCol, 8, 0,PText);
 switch(S.VuMode)
  {
   case(-1):sprintf(PText,"Ee:%2.1i 2D rawGL L-hand\0",S.VuMode); break;
   case( 0):sprintf(PText,"Ee:%2.1i 2D Orthographic\0",S.VuMode); break;
   case( 1):sprintf(PText,"Ee:%2.1i 2D Perspective\0" ,S.VuMode); break;
   case( 2):sprintf(PText,"Ee:%2.1i 3D  Red & Cyan\0"  ,S.VuMode); break;
   case( 3):sprintf(PText,"Ee:%2.1i 3D Right|Left\0"  ,S.VuMode); break;
   case( 4):sprintf(PText,"Ee:%2.1i 3D  Left|Right\0" ,S.VuMode); break;
  }                                 nRow++; PrntOrtho(nRow,nCol, 6, 0,PText);
 sprintf(PText,"~ :Screen Selfie->.bmp file\0");
                                    nRow++; PrntOrtho(nRow,nCol, 8, 0,PText);
 sprintf(PText,"` :Depth  Selfie\0");
                                    nRow++; PrntOrtho(nRow,nCol, 8, 0,PText);
 sprintf(PText,"on\0");if(S.DepthSelfie>0) PrntOrtho(nRow,nCol+17, 1, 0,PText);

 sprintf(PText,"Escape:toggles screen\0");
                                    nRow++; PrntOrtho(nRow,nCol, 8, 0,PText);
                                                      if(iOrthoHdr<2)return;
 // -- Author info:
                                  nRow++;
 sprintf(PText,"Email ServerName\0");nRow++;PrntOrtho(nRow,nCol  , 1, 0,PText);
 sprintf(PText,"@\0");                      PrntOrtho(nRow,nCol+ 5, 6, 0,PText);
 sprintf(PText,".com\0");                   PrntOrtho(nRow,nCol+16, 6, 0,PText);
 sprintf(PText,"(nnn)nnn-nnnn \0"); nRow++;PrntOrtho(nRow,nCol+ 2, 1, 0,PText);
                                                      if(iOrthoHdr<3)return;
```

```
  // -- KeyPress info:
  sprintf(PText,"KeyChar =  %3.1i   %1c\0",(int)S.KbdKey,S.KbdKey);
                                      nRow++; PrntOrtho(nRow,nCol,11, 0,PText);
  sprintf(PText,"ArrowKey=  %3.1i\0",S.ArrowKey);
                                      nRow++; PrntOrtho(nRow,nCol,11, 0,PText);
      if(iOrthoHdr<4)return;
  // -- Current mouse location info:
  sprintf(PText,"Pix     Hor   &  Ver\0",S.vMouse[0],S.vMouse[1]);
                                      nRow++; PrntOrtho(nRow,nCol, 6, 0,PText);
  sprintf(PText,"Screen%5.1i  %7.1i\0",S.xWindowFull,S.yWindowFull);
                                      nRow++; PrntOrtho(nRow,nCol, 6, 0,PText);
  sprintf(PText,"Mouse %5.1i    %5.1i\0",S.MouseXc,S.MouseYc);
                                      nRow++; PrntOrtho(nRow,nCol, 6, 0,PText);
  sprintf(PText,"Vec  %6.3f%9.3f\0",S.vMouse[0],S.vMouse[1]);
                                      nRow++; PrntOrtho(nRow,nCol, 1, 0,PText);
  sprintf(PText,"Txt     Col   &  Row\0",S.vMouse[0],S.vMouse[1]);
                                      nRow++; PrntOrtho(nRow,nCol, 6, 0,PText);
  sprintf(PText,"    ->%5.1i    %5.1i\0",S.MouseXc/S.lCharX+1,
                                        S.MouseYc/S.lCharY+1);
                                      nRow++; PrntOrtho(nRow,nCol, 6, 0,PText);
  sprintf(PText,"      %5.1i  <-\0",S.MouseXc/S.lCharX+1-S.nCharMaxX);
                                      nRow++; PrntOrtho(nRow,nCol, 6, 0,PText);
                                                        if(iOrthoHdr<4)return;
  sprintf(PText,"Tt: iTeapot =%5.1i\0",S.iTeapot);
                                      nRow++; PrntOrtho(nRow,nCol, 4, 0,PText);

  sprintf(PText,"(X,Y,Depth) GL-screen:\0",S.iTeapot);
                                      nRow++; PrntOrtho(nRow,nCol,11, 0,PText);
  sprintf(PText,"%6.3f %6.3f %6.3f\0",S.MouseScreenh[0],S.MouseScreenh[1],
                 S.MouseScreenh[2]);  nRow++; PrntOrtho(nRow,nCol,11, 0,PText);

  sprintf(PText,"nN Nutation    :on\0");
                       if(S.Nutate>0){ nRow++; PrntOrtho(nRow,nCol, 1, 0,PText);}

 //if(S.MouseLmbWasPressed==1)
   { sprintf(PText,"Model 7dof:\0");
                                      nRow++; PrntOrtho(nRow,nCol, 6, 0,PText);
     sprintf(PText,"X    =%8.3f\0",S.PoIX);
                                      nRow++; PrntOrtho(nRow,nCol, 6, 0,PText);
     sprintf(PText,"Y    =%8.3f\0",S.PoIY);
                                      nRow++; PrntOrtho(nRow,nCol, 6, 0,PText);
     sprintf(PText,"Z    =%8.3f\0",S.PoIZ);
                                      nRow++; PrntOrtho(nRow,nCol, 6, 0,PText);
     sprintf(PText,"Roll =%8.3f\0",S.Attitudeh[0]);
                                      nRow++; PrntOrtho(nRow,nCol, 6, 0,PText);
     sprintf(PText,"Pitch=%8.3f\0",S.Attitudeh[1]);
                                      nRow++; PrntOrtho(nRow,nCol, 6, 0,PText);
     sprintf(PText,"Yaw  =%8.3f\0",S.Attitudeh[2]);
                                      nRow++; PrntOrtho(nRow,nCol, 6, 0,PText);
     sprintf(PText,"Scale=%8.3f\0",S.Scale);
                                      nRow++; PrntOrtho(nRow,nCol, 6, 0,PText);
   }//(S.MouseLmbWasPressed==1)

                                                        if(iOrthoHdr<5)return;
  // At upper center of screen - IP information:
  sprintf(PText,"Copyright 2016 -- Your name here. --\0");
                                                         nCol = S.nCharCenX;
                                    nRow=1; PrntOrtho(nRow,nCol-18, 1, 0,PText);
  sprintf(PText,"Your street here.\0");

                                    nRow++; PrntOrtho(nRow,nCol- 8,11, 0,PText);
  sprintf(PText,"   City  ,    State    ZipCode \0");
                                    nRow++; PrntOrtho(nRow,nCol-15,11, 0,PText);
  sprintf(PText,"(nnn) nnn-nnnn\0");
                                    nRow++; PrntOrtho(nRow,nCol- 7,11, 0,PText);
  sprintf(PText,"Personal/Proprietary R&D\0");
                                    nRow++; PrntOrtho(nRow,nCol-12, 2, 0,PText);
 }//End ProjOrtho ------------------------------------------------------------
```

```
void PrntOrtho(int nRow,int mColumn,int iColorFG,int iColorBg,char PText[80])
               /*v:0.5 ------------------------------------------------------
  Print text on a raw OpenGL screen (Proj. & ModelView matrices: identities)
    using an OpenGL bitmap font.
  2016.06.24.1600cdt JMS                                                   */
{int n,nChar,MtxMode[1];                                                 //Dec

 //Save MatrixMode,PROJECTION,MODELVIEW & load Identities:
//ColorStd(iColorFG); glFlush(); //<- This works - I don't know why!!!
  Colors3D(iColorFG);
 glGetIntegerv(GL_MATRIX_MODE,MtxMode);   //GL_MATRIX_MODE=ba0=2976 dec.
 glMatrixMode(GL_PROJECTION); glPushMatrix(); glLoadIdentity();
 glMatrixMode(GL_MODELVIEW ); glPushMatrix(); glLoadIdentity();      glFlush();
 //Position the first character:
 S.xPrint=       -1.00     +(mColumn-1)*S.cWidth ;
 S.yPrint=        1.00     - nRow      *S.cHeight;
 S.zPrint=       -0.10                          ;
 glRasterPos3d( S.xPrint,S.yPrint,S.zPrint);

 //Colors3D(iColorFG);  glFlush(); <-This call made here lagged 1 iteration!!!
 //Write the string as individual character 8x13 bitmaps.
             nChar=strlen(PText);
 for(n=0; n<nChar; n++) glutBitmapCharacter(GLUT_BITMAP_8_BY_13,PText[n]);
                                                                  glFlush();
 //Restore PROJECTION, MODELVIEW, & MatrixMode:
 glMatrixMode(GL_PROJECTION); glPopMatrix();
 glMatrixMode(GL_MODELVIEW ); glPopMatrix();
 glMatrixMode(MtxMode[1])   ;                                glFlush();
}//End PrntOrtho ------------------------------------------------------------


void PAhXR( double Xyzhin[4], double Rpyhin[4],double hXRout[4][4]) /*v:0.5----
/*                Positionh &       Attitudeh to      hXR[4][4] <-6DoF:
                   (X,Y,Z,w),    (Roll,Pitch,Yaw,w) -> 6doF Motion Control
        in right-handed Flight Simulation (Fs) coordinates.
        Externally - all angles are in degrees.
        Internally - the angle computations are in radians.
  2016.06.24.1600cdt JMS-                                                  */
{int n,m;                                                                //Dec
 double Temp,Rpy[3],Xyz[3],H[4][4];      /* double DtoR = asin(1.0e0)/90.e0; */
 double cp,cr,cy,sp,sr,sy,DtoR;

 /* Normalize the homogeneous inputs & convert angles to radians: */
 DtoR= asin(1.0e0)/90.e0;                Temp=Xyzhin[3];if(Temp==0.) exit(-1);
 for(n=0;n<3;n++) Xyz[n]=     Xyzhin[n]/Temp;
                                         Temp=Rpyhin[3];if(Temp==0.) exit(-2);
 for(n=0;n<3;n++) Rpy[n]=DtoR*Rpyhin[n]/Temp;
 /* Compute the sines & cosines of the angles:*/
 sr=(double)sin(Rpy[0]); sp=(double)sin(Rpy[1]); sy=(double)sin(Rpy[2]);
 cr=(double)cos(Rpy[0]); cp=(double)cos(Rpy[1]); cy=(double)cos(Rpy[2]);
 /* Compute the resulting 6DoF matrix: */
 H[0][0]=cy*cp;H[0][1]=-sy*cr+cy*sp*sr;H[0][2]= sy*sr+cy*sp*cr;H[0][3]=Xyz[0];
 H[1][0]=sy*cp;H[1][1]= cy*cr+sy*sp*sr;H[1][2]=-cy*sr+sy*sp*cr;H[1][3]=Xyz[1];
 H[2][0]=  -sp;H[2][1]= cp*sr         ;H[2][2]= cp*cr         ;H[2][3]=Xyz[2];
 H[3][0]= 0.  ;H[3][1]= 0.            ;H[3][2]= 0.            ;H[3][3]= 1.   ;
 /* Return the 6DoF solution: */
 for(n=0; n<4; n++){for(m=0; m<4; m++){hXRout[n][m]=H[n][m];}}
}//End PAhXR ---------------------------------------------------------------


void Alpha6D(double Pxyzh[4],double ArpyDh[4],int ModeFlag, //v:0.5-----------
             double SizeH   ,float LineWidth ,int iCol    , char Label[80])  /*
  Vector-based character string writing in 6dof using an OpenGL vector font:
  2016.06.24.1600cdt JMS- Writes Label[] ~scaled to ModelView coordinates
           with 6DOF: Pxyz[4], ArpyD[4],
                      size:SizeH, color:iCol, LineWidth in pixels

    ModeFlag=0: Unrotated characters are written parallel-to the +X axis,
                          readable from the -Z direction
                  in the lefthanded native OpenGL screen coordinates.
```

```
                  Use when S.ThreePhase=2. no rotation writes horizontally;
                                           +yaw (Rrpyh[2]) rotates text CCW.

    ModeFlag=1: Unrotated characters are written parallel-to the +X axis,
                        readable from the +Y direction
             in righthanded "Flight Simulation" (abbrev. 'Fs') coordinates:
                  Use when S.ThreePhase=3.                                  */
{                                                                          //Dec
 int MtxModeL[1],k,m,n,nChar;
 double Xyzh[4],Rpyh[4],hXR[4][4],hSerial[16],Temp;

 //Adapt ModelView
 glGetIntegerv(GL_MATRIX_MODE,MtxModeL);
 glMatrixMode(GL_MODELVIEW ); glPushMatrix(); //glLoadIdentity();
 //Translate
 Temp=Pxyzh[3] ; if(Temp==0.) exit(-1);
 for(n=0; n<4; n++){Xyzh[ n]=Pxyzh[n]/Temp; }
 h4Fill( Rpyh, 0.e0, 0.e0, 0.e0, 1.e0 );

  PAhXR(Xyzh,Rpyh,hXR);
   k=0;for(m=0; m<4; m++){for(n=0; n<4; n++){hSerial[k]=hXR[n][m];k=k+1;}}
    glMultMatrixd(hSerial);
 //Scale
 for(n=0; n<4; n++){for(m=0; m<4; m++){hXR[n][m]=0.;   }}
//for(n=0; n<3; n++)                {hXR[n][n]=SizeH/104. ;}  hXR[3][3]=1.;
//for(n=0; n<3; n++)                {hXR[n][n]=SizeH/100. ;}  hXR[3][3]=1.;
  hXR[0][0]=SizeH/110.;
  hXR[1][1]=SizeH/100.;
  hXR[2][2]=SizeH/100.;
  hXR[3][3]=   1.     ;

   k=0;for(m=0; m<4; m++){for(n=0; n<4; n++){hSerial[k]=hXR[n][m];k=k+1;}}
  glMultMatrixd(hSerial);
 //Rotate
 Temp=ArpyDh[3]; if(Temp==0.) exit(-1);
 for(n=0; n<4; n++){Rpyh[n]=ArpyDh[n]/Temp; }
 h4Fill( Xyzh, 0.e0, 0.e0, 0.e0, 1.e0 );

 if(ModeFlag==1) Rpyh[0]=Rpyh[0]-90.e0;       //<-Fs coords.
  PAhXR(Xyzh,Rpyh,hXR);
   k=0;for(m=0; m<4; m++){for(n=0; n<4; n++){hSerial[k]=hXR[n][m];k=k+1;}}
 //k=0;for(n=0; n<4; n++){for(m=0; m<4; m++){hSerial[k]=hXR[n][m];k=k+1;}}
  glMultMatrixd(hSerial);
 //Render the text
 glLineWidth(LineWidth);glShadeModel(GL_FLAT);
 if(iCol>0) Colors3D(iCol);
 nChar=strlen(Label);
 for(n=0;n<nChar;n++)glutStrokeCharacter(GLUT_STROKE_MONO_ROMAN,Label[n]);
 //Restore PROJECTION, MODELVIEW, & MatrixMode:
 glFlush; glMatrixMode(GL_MODELVIEW ); glPopMatrix();
 glFlush; glMatrixMode(  MtxModeL[1]); glFlush();
}//End Alpha6D ------------------------------------------------------------


void ScreenSelfie(void) /*v:0.5-------------------------------------------------
  Screen dump:
  2016.06.24.1600cdt JMS- Works. Max screen width: 3000 pixels.
                        - Choosing a white background for selfies
                            saves a lot of ink when printing hardcopies.

  This routine dumps the OpenGL "console application" screen as  "Selfie.Bmp"
   which is formatted as a 24 bit-per-pixel full color image.
  The "~" tilda key triggers the dump.                                      */
{                                                                          //Dec
//typedef struct{          // Bitmap Header (54 bytes)
  char  BM[2];             // "BM" mandatory
  int   nSizeTot;        //54+nClrUsed*4+(((nWidth*(nBitsPP/8)+3)/4)*4)*nHeight
  short nReserv1,nReserv2; //  0, 0 not used
  int   nOffBit,nSizeH;    // 54,40
  int   nWidth,nHeight;    // ___,___(pixels,pixels)
```

```
   short nPlanes,nBitsPP;   //   1,___  (24:TC,  8:Grey Scales)
   int   nCompres,nSizeC;   //   0,___ Orthos use in error -> BHFix
   float XPpM,YPpM;         //   ___,___ scaling (USGS float)
   int   nClrUsed,nClrImpo; //   ___, 0 ( 0:TC,256:Grey Scales)
 //}  BmpHdrL;

  FILE   *fpdump;           //output file "SelfieN.Bmp"
  float   fImage[    9000]; // 1920*3=5760 ... sized for 3000 pixels max.
  char     Image2[   9000]; //                "    "  3000 pixels max
  GLsizei iWidth,jHeight;   //integer(kind=GLsizei)
  GLint   i3WH;             //integer(kind=GLint)
    int   i,iHr3,iHr4,j,k;  //integer*4
  GLint   iw0=0, jh0=0;
  static int nDump=-1;      //integer*4
  char    DumpFile[20];
  char    PText[   80];
  int     PixelRowPadBites;
  char    PadBite[1];

  nDump=nDump+1;
  if(nDump>9){sprintf(PText,"10 screen dumps max.\0");PrntOrtho(3,2, 3,0,PText);
              printf(PText);                                          return;}
  //Size the output image
  iWidth =S.xWindowFull;
  jHeight=S.yWindowFull;

  //Define the .Bmp header:
  sprintf(BM,"BM");
  nBitsPP  = 24;
  nSizeH   = 40;
  nOffBit  = 14 + nSizeH;
  iHr3=iWidth*3;
 //iHr4=int((iHr3+3)/4)*4;
  iHr4=(iHr3+3)/4;
  iHr4=iHr4*4;
  PixelRowPadBites=iHr4-iHr3;
  nSizeTot = nOffBit+iHr4*jHeight;
  nWidth   = iWidth;
  nHeight  = jHeight;
  nPlanes  = 1;
  nSizeC   = 0;
  XPpM     = .5296908e-41;
  YPpM     = .5296908e-41;
 //Allocate memory for the dump:
  i3WH=3*iWidth*jHeight;
  sprintf(         DumpFile ,"Selfie%i.bmp\0",nDump);
  printf(          DumpFile);
  strcpy(PText,DumpFile);         PrntOrtho(3,2, 3,0,PText);
 //Export a fairly-standard .bmp file:
   fpdump=fopen(DumpFile,"wb");
    fwrite(BM       ,sizeof(char ),2,fpdump); //"BM" mandatory
    fwrite(&nSizeTot,sizeof(int  ),1,fpdump);
                     /*54+nClrUsed*4+(((nWidth*(nBitsPP/8)+3)/4)*4)*nHeight*/
    fwrite(&nReserv1,sizeof(short),1,fpdump); //  0  not used
    fwrite(&nReserv2,sizeof(short),1,fpdump); //  0  not used
    fwrite(&nOffBit ,sizeof(int  ),1,fpdump); // 54
    fwrite(&nSizeH  ,sizeof(int  ),1,fpdump); // 40
    fwrite(&nWidth  ,sizeof(int  ),1,fpdump); // pixels
    fwrite(&nHeight ,sizeof(int  ),1,fpdump); // pixels
    fwrite(&nPlanes ,sizeof(short),1,fpdump); //  1
    fwrite(&nBitsPP ,sizeof(short),1,fpdump); // 24:TC,8:gray scale
    fwrite(&nCompres,sizeof(int  ),1,fpdump); //  0
    fwrite(&nSizeC  ,sizeof(int  ),1,fpdump); //  ? Orthos use in error
    fwrite(&XPpM    ,sizeof(float),1,fpdump); // scaling: (USGS float)
    fwrite(&YPpM    ,sizeof(float),1,fpdump); // scaling: (USGS float)
    fwrite(&nClrUsed,sizeof(int  ),1,fpdump); // 0: for true color
    fwrite(&nClrImpo,sizeof(int  ),1,fpdump); // 0: for true color

    glReadBuffer(GL_FRONT_LEFT);             /*Dump the left-front buffer*/
    for(j=0; j<jHeight; j++)
```

```
    {jh0=j; iw0=0;
     glReadPixels( iw0,jh0,iWidth,1,GL_RGB,GL_FLOAT,fImage); glFlush();
     for(i=0; i<iWidth; i++)
      {       k=i*3                ;
        Image2[k  ]=fImage[k+2]*255.;
        Image2[k+1]=fImage[k+1]*255.;
        Image2[k+2]=fImage[k  ]*255.;
      }//i
     fwrite(Image2,sizeof(char),iHr3,fpdump);
     for(i=0; i<PixelRowPadBites; i++) fwrite(PadBite,sizeof(char),1,fpdump);
    }//j
 fclose(fpdump  );
 printf("Beep!\a\n");
}//End ScreenSelfie ------------------------------------------------------


void h4Fill( double h4[],double h00, double h01, double h02, double h03)/*v:0.5
  Fills a 4 element vector with values
  2016.06.24.1600cdt JMS                                                  */
{                       h4[0]= h00; h4[1]= h01; h4[2]= h02; h4[3]= h03;
}//End h4Fill ------------------------------------------------------------


void h44Fill( double h44[4][4], //v:0.5-----------------------------------
              double h00,    double h01,    double h02,    double h03,
              double h10,    double h11,    double h12,    double h13,
              double h20,    double h21,    double h22,    double h23,
              double h30,    double h31,    double h32,    double h33 )
/*Fills a 4x4 matrix with values
  2016.06.24.1600cdt JMS                                                  */
{          h44[0][0]=h00; h44[0][1]=h01; h44[0][2]=h02; h44[0][3]=h03;
           h44[1][0]=h10; h44[1][1]=h11; h44[1][2]=h12; h44[1][3]=h13;
           h44[2][0]=h20; h44[2][1]=h21; h44[2][2]=h22; h44[2][3]=h23;
           h44[3][0]=h30; h44[3][1]=h31; h44[3][2]=h32; h44[3][3]=h33;
}//End h44Fill ----------------------------------------------------------


void Print4(double hin[4],char LabelIn[80]) /*v:0.5---------------------------
void   Seeh4d(...
  2016.06.24.1600cdt JMS- Prints h4d[4] followed by LabelIn:              */
{int i;                                                                 //Dec
 for(i=0;i<4;i++) { fprintf(S.fp,"%14.6f\0",hin[i] ); }/*i*/
                    fprintf(S.fp,"%-80s\n" ,LabelIn);
}//End Print4 -----------------------------------------------------------


void Print44(double hin[4][4],char LabelIn[80]) /*v:0.5-----------------------
void   Seeh44d(...
  2016.06.24.1600cdt JMS- Prints LabelIn & h44d[4][4]:                    */
{int i,j;                                                               //Dec
                                fprintf(S.fp,"\n%-80s\n" ,LabelIn  );
   for(i=0; i<4; i++) { for(j=0; j<4; j++)
                         {      fprintf(S.fp,"%19.13f\0",hin[i][j]);
                         }/*j*/ fprintf(S.fp,"\n"                 );
                      }/*i*/
//                             fprintf(S.fp,"=\n");
// for(i=0; i<4; i++) { for(j=0; j<4; j++)
//                         {      fprintf(S.fp,"%19.9e\0",hin[i][j]);
//                         }/*j*/ fprintf(S.fp,"\n"                 );
//                      }/*i*/
}//End Print44 ----------------------------------------------------------


void MMply44(double hout[4][4],double hin1[4][4],double hin2[4][4]) /*v:0.5----
  2016.06.24.1600cdt JMS- Solves: hout[4][4]=hin1[4][4]*hin2[4][4]        */
{ int i,j,k;
 for(i=0;i<4;i++)
  { for(j=0;j<4;j++)  {   hout[i][j]=0.e0;
     for(k=0;k<4;k++)   { hout[i][j]=hout[i][j]+hin1[i][k]*hin2[k][j]; }/*k*/
  }/*j*/             }/*i*/
```

```
 }//End MMply44 ------------------------------------------------------------


 void Invert44(double hin[4][4],double hinverse[4][4],int iPrint, int iView) /*-
   2016.06.24.1600cdt JMS- Inverts a 4x4 (e.g. homogeneous) matrix.
                          - Self-reports via S.iPrint & S.iView flags. Demo:F6.*/
 { double h[5][9],hMax,Noise,ValMax;
   double hinSave[4][4],hinversehin[4][4];
   int iExit,Iter,n, nu,nRowUsed[5],m,mu,mColUsed[9],color,iDone;
   char PText[80];

  if(S.NowPrint>0) Print44(hin,"Invert44 input: hin[4][4]\0");

                                hMax=0.e0;
  for(n=0;n<4;n++) {for(m=0;m<4;m++) { hinSave[n][m]=hin[n][m]; }/*m*/ }/*n*/
  for(m=0;m<9;m++) {mColUsed[m]=-m; if(m>4) mColUsed[m]=-(m-4);} /*m*/
  for(n=0;n<5;n++)
   {nRowUsed[n]=-n;//unused row indices
    for(m=0;m<9;m++)
     {          h[n][m]  =0.e0;                    //clear the h[][] array
      if((n>0)&&(m>0)&&(m<5))
       {        h[n][m]  =hin[n-1][m-1];           //Importing hin
        if(abs(h[n][m])>hMax) hMax=abs(h[n][m]); //Finding the largest abs value
       }/*((n>0)&&(m>0))*/
     }/*m*/
    if(n>0)    h[n][n+4]=1.e0; //Appending the identity matrix on the right side
   }/*n*/
  //Set the noise floor at a billionth of the largest matrix coefficient:
                         Noise=hMax/1.e9;

  //Iterate the inversion process: ------------------------
                   iDone=0;
  for(Iter=0;Iter<6;Iter++)
   {for(n=0;n<5;n++) {h[n][0]=0.e0;}
    for(m=0;m<9;m++) {h[0][m]=0.e0;}
    //Find the largest remaining coefficient >Noise, if any:
    nu=0;mu=0;ValMax=Noise;
    if(Iter==0)           goto NowReport;
    if(Iter==5) {iDone=1; goto NowReport;}
    for(m=1;m<5;m++)
     {for(n=1;n<5;n++)
       {if((nRowUsed[n]<0)&&(mColUsed[m]<0))
         {if(fabs(ValMax)<fabs(h[n][m])) {ValMax=h[n][m];nu=n;mu=m;}
         }/*(nRowUsed[n]<0)&&(mColUsed[n]<0)*/
       }/*n*/                                                          }/*m*/
    if(nu==0){iDone=1;goto InverterDone;} //<- Exits when only noise is left.
    //Swap rows:
    for(m=1;m<9;m++) {h[ 0][m]=h[nu][m]/ValMax;
                      h[nu][m]=h[mu][m];
                      h[mu][m]=h[ 0][m];                             }/*m*/
    nRowUsed[ 0]  =    nRowUsed[nu]; //& swap indices & change signs to >0
    nRowUsed[nu]  =    nRowUsed[mu];
    nRowUsed[mu]  =abs(nRowUsed[ 0]);
    mColUsed[mu]  =abs(mColUsed[mu]);
        mColUsed[nRowUsed[mu]+4]=
    abs(mColUsed[nRowUsed[mu]+4]);
    //Perform the row reductions:
    for(n=1;n<5;n++) {if(n!=mu)
                      {h[n][0]=h[n][mu];
                       for(m=1;m<9;m++){h[n][m]=h[n][m]-h[n][0]*h[0][m];}/*m*/
                      }/*(n!=nu)*/                                   }/*n*/
    if((iDone==0)&&(Iter<4)) goto NowReport;

  InverterDone:
    //Eliminate linear independencies
    for(n=1;n<5;n++)
     {if(nRowUsed[n]<0) {for(m=1;m<9;m++) {h[n][m]=0.e0;}/*m*/
                         }/*(nRowUsed[n]<0)*/                        }/*n*/
    for(m=1;m<5;m++)
     {if(mColUsed[m]<0) {for(n=1;n<5;n++) {h[n][m]=0.e0; h[n][m+4]=0.e0;}/*n*/
```

```
                    }/*(mColUsed[n]<0)*/                                  }/*m*/

  //Return the inverse:
    for(n=1;n<5;n++) {for(m=1;m<5;m++){hinverse[n-1][m-1]=h[n][m+4];}/*m*/}/*n*/

  NowReport:
    //Opional printed output:
    if((iPrint>0)&&(S.NowPrint>0))
      {if(Iter==0)
         fprintf(S.fp,"\nInvert44- h[][] initialized as follows:\n");
       else if((Iter<5)&&(iDone>0))
         fprintf(S.fp,"\nInvert44- Linearly dependent:\n");
       else if(Iter==5) fprintf(S.fp,"\nInvert44- final result:\n");
       else fprintf(S.fp
         ,"\nEnd of Iteration#%1.1i nu=%1.1i mu=%1.1i ValMax=%10.4f\n"
                           ,Iter          ,nu      ,mu     ,ValMax);color=15;

                fprintf(S.fp,"            \0");
       for(  m=1; m<9; m++) {      fprintf(S.fp,"%5.1i     \0",mColUsed[m]);
                           }/*j*/ fprintf(S.fp,"\n");
       for(  n=0; n<5; n++) {//if(n==0) fprintf(S.fp,"   \0");
                             fprintf(S.fp,"%2.1i \0",nRowUsed[n]);
         for(m=0; m<9; m++) {      fprintf(S.fp,"%10.4f\0",h[n][m]);
                           }/*m*/ fprintf(S.fp,"\n");                    }/*n*/
       if(iDone>0)
        { Print44(hinverse,"hinverse=\0");
          MMply44(hinversehin,hinverse,hin);
          Print44(hinversehin,"hinverse*hin=\0");
          for(n=0;n<4;n++){hinversehin[n][n]=hinversehin[n][n]-1.e0;}
          Print44(hinversehin,"hinverse*hin-I=\0");
        }/*(iDone>0)*/
     }/*((iPrint>0)&&(S.NowPrint>0))*/

    //Optional screen output:
    if((iView>0)&&(S.NowView>0))
     {//Iteration header information:
      if(Iter==0) Seeh44d(hin,5,20, 1,"Invert44 matrix input = hin[4][4]:\0");
      sprintf(PText,"End of Iteration#%1.1i nu=%1.1i mu=%1.1i ValMax=%10.4f\0"
                           ,Iter          ,nu      ,mu      ,ValMax );color=1;
      if(Iter==0) sprintf(PText,"Invert44- h[][] initialized as follows:\0");
      if(Iter==5) sprintf(PText,"Invert44- final result:\0");
      if((Iter<5)&&(iDone>0)){sprintf(PText,"Invert44- Linearly dependent:\0");
                                                              color=9;}
                                 PrntOrtho(Iter*8+11,2,color,0,PText);
      //Matrix & indices numerics:
      for(m=1; m<9; m++)  {sprintf(PText,"%5i\0",mColUsed[m]);
                           PrntOrtho(Iter*8+12,m*14+8, 1,0,PText);}/*m*/
      for(n=0; n<5; n++)  {sprintf(PText,"%5i\0",nRowUsed[n]);
                           PrntOrtho(Iter*8+13+n,2   , 1,0,PText);
       for(m=0; m<9; m++) {
        if     (m==0)                                        color=12;
        else if(m<5 ){                                       color= 1;
                      if((mColUsed[m]>0)||(nRowUsed[n]>0)) color= 5;}
        else         {                                       color= 5;
                      if((mColUsed[m]>0)||(nRowUsed[n]>0)) color= 6;}
             if(n==0)                                        color=12;
             if((n==nu)&&(m==mu))                            color= 6;
        if((iDone>0)&&(n>0)&&(m>0))
         {if((mColUsed[m]<0)||(nRowUsed[n]<0))               color=10;
          if((mColUsed[m]<0)&&(nRowUsed[n]<0))               color= 3;}
        if((iDone>0)&&((n==0)||(m==0)))                      color=12;    //=0?7?
                        sprintf(PText,"%14.6f\0",h[n][m]);
                           PrntOrtho(Iter*8+13+n,m*14+6,color,0,PText);
                       }/*m*/                                          }/*n*/
      if(iDone>0) Seeh44d( hinverse,Iter*8+19,76, 6
                          ,"Invert44 matrix output = hinverse[4][4]:\0");
     }/*((iView>0)&&(S.NowView>0))*/
    if(iDone>0) break;
   }//Iter ------------------------------------------------
```

```
}// End Invert44  ------------------------------------------------------------


void Seeh4d(double hin[4],int nRow,int mCol,int Color,char LabelIn[80]) /*v:0.5
  2016.06.24.1600cdt JMS- Displays h44d[4][4] at screen location nRow,mColumn,
                            followed by LabelIn.                              */
{int  i; char PText[80];                                                      //Dec
                        PrntOrtho(nRow  ,mCol     ,  1  ,0,LabelIn);
 for(i=0;i<4;i++)
  {     sprintf(PText,"%14.6f\0",hin[i]);
                        PrntOrtho(nRow+1,mCol+i*14,Color,0,PText  );
  }/*i*/
                                /*LabelIn: at the right end of the text row.*/
}//End Seeh4d ---------------------------------------------------------------


void Seeh44d(double hin[4][4],int nRow,int mCol,int Color,char LabelIn[80])
             /*v0.5-----------------------------------------------------------
  2016.06.24.1600cdt JMS-Displays h44d[4][4] at screen location nRow,mColumn.*/
{int i,j;  char PText[80];                                                    //Dec

                        PrntOrtho(nRow    ,mCol     ,  1  ,0,LabelIn);
 for(i=0; i<4; i++)
  { for(j=0; j<4; j++)
         { if(fabs(hin[i][j])< 100000.) sprintf(PText,"%14.6f\0",hin[i][j]);
           if(fabs(hin[i][j])>=100000.) sprintf(PText,"%14.6e\0",hin[i][j]);
                        PrntOrtho(nRow+1+i,mCol+j*14,Color,0,PText  );
  }/*j*/ }/*i*/
}//End Seeh44d --------------------------------------------------------------


void Teapot(void) /*v:0.5-------------------------------------------------------
/* The GLUT Teapot:
  2016.06.24.1600cdt JMS- Ref:william.mitchell@nist.gov  "module view_modifier"
                          - Coloring fairly successful.                       */
{int iT;                                                                      //Dec
//Redbook teapot's surface optical properties for bronze:
//     Material surface optical coefficients:
 float    ambient[4] = { 0.2125  , 0.1275  , 0.054   , .02};
 float    diffuse[4] = { 0.714   , 0.4284  , 0.181   ,1.0 };
 float   specular[4] = { 0.393548, 0.271906, 0.166721,1.0 };
//     Light source location & color:
 float        pos[4] = { 1.0     , 1.0     , 1.0     , 1.0 };
 float      white[4] = { 1.0     , 1.0     , 1.0     ,  .9};
 float       gray[4] = { 0.5     , 0.5     , 0.5     ,  .9};
//float L0ambient[4] = { 0.2     , 0.2     , 0.2     , 1.0};
 float  L0ambient[4] = { 0.6     , 0.6     , 0.6     , 1.0};
 float noemission[4] = { 0.0     , 0.0     , 0.0     , 0.0 };

 if(S.iTeapot<1) return;
 iT=S.iTeapot-1;

 if(!(iT&1)) glShadeModel(GL_SMOOTH);  //Smooth facets.
 if(  iT&1 ) glShadeModel(GL_FLAT);    // Flat  facets.
 //if(  iT&4 ) CubeGrid(13);

 //Roll ModelView by -90. degrees so the teapot draws top-up (toward -Z):
 if(S.VuMode>-1) glRotated(-90.0e0, 1.0e0, 0.0e0, 0.0e0);
 //-----------------------------------
 /*Define Light0's position, color, & ambient contribution:
   Display LIGHT0 as an emitting 400-facet wire sphere .5 units in diameter:*/
                   glLightfv(     GL_LIGHT0,GL_POSITION ,  pos        );
                   glLightModelfv(GL_LIGHT_MODEL_AMBIENT,  noemission );
 if(S.ForceRGBA>=0){glMaterialfv(  GL_FRONT ,GL_EMISSION ,  white      );
                  //if(S.iScrnColor== 1)
                  //glMaterialfv(  GL_FRONT ,GL_EMISSION ,  gray      );
                   }
 if(S.ForceRGBA< 0) glMaterialfv(  GL_FRONT ,GL_EMISSION ,S.ColorRGBA  );

                   glEnable(GL_LIGHTING); glEnable(GL_LIGHT0);
```

```
                      glTranslatef( pos[1], pos[2]    , pos[3]    );
                        glutWireSphere( .05e0, 20, 20 );       //View the light
                      glTranslatef(-pos[1],-pos[2]    ,-pos[3]    );
                    glDisable(GL_LIGHT0); glDisable(GL_LIGHTING);

                    glMaterialfv(GL_FRONT,GL_EMISSION  , noemission);
 if(S.ForceRGBA>=0) glLightfv(     GL_LIGHT0        , GL_DIFFUSE ,  white    );
 if(S.ForceRGBA< 0) glLightfv(     GL_LIGHT0        , GL_DIFFUSE ,S.ColorRGBA);
                    glLightModelfv(GL_LIGHT_MODEL_AMBIENT,          L0ambient);
 //---------------------------------------
 //Enable material surface optical properties:
                         //glEnable(GL_COLOR_MATERIAL); <- isn't working here.
 if(S.ForceRGBA>=0)
  {glMaterialfv(GL_FRONT          , GL_AMBIENT  , ambient    ); //Bronze.
   glMaterialfv(GL_FRONT          , GL_DIFFUSE  , diffuse    );
   glMaterialfv(GL_FRONT          , GL_SPECULAR , specular   );
   glMaterialf( GL_FRONT          , GL_SHININESS, 25.6       );      glFlush();
  }/*(S.ForceRGBA>=0)*/
 if(S.ForceRGBA<0)
  {glMaterialfv(GL_FRONT          , GL_AMBIENT  , S.ColorRGBA);
   glMaterialfv(GL_FRONT          , GL_DIFFUSE  , S.ColorRGBA);
   glMaterialfv(GL_FRONT          , GL_SPECULAR , noemission );
   glMaterialf( GL_FRONT          , GL_SHININESS, 25.6       );      glFlush();
  }/*(S.ForceRGBA<0)*/

 if(S.ForceRGBA< 0) glLightfv(GL_LIGHT0,GL_DIFFUSE,S.ColorRGBA);
 glEnable( GL_LIGHTING); glEnable( GL_LIGHT0  ); //Enable lighting
   if(!(iT&2)) glutSolidTeapot(1.0e0);           //draw the Teapot:
   if(  iT&2 ) glutWireTeapot( 1.0e0);
 glDisable(GL_LIGHT0); glDisable(GL_LIGHTING);   //Disnable lighting
 glShadeModel(GL_FLAT);                  //Revert to the unsmoothed glShadeModel
 //UnRoll (restore) ModelView  & continue in standard Flight Sim coordinates.
 if(S.VuMode>-1) glRotated( 90.0e0, 1.0e0, 0.0e0, 0.0e0);
}//End Teapot ---------------------------------------------------------------


void CubeGrid(int nCol) /*v:0.5-------------------------------------------------
  A 6dof Origin:
  2016.06.24.1600cdt JMS- Draws X,Y,Z axes and an enclosing cube.           */
{int i,j,k;                                                              //Dec
 float Frac,gi,gj,gk;
 double Xyzh[4],Rpyh[4];

 Frac=.9;
 glLineWidth(1.f);
 glShadeModel(GL_FLAT); //glShadeModel(GL_SMOOTH)
 glBegin(GL_LINES);
 Colors3D(nCol);
 for(     i=-1;i<2;i++) {gi=i;
   for(   j=-1;j<2;j++) {gj=j;
     for( k=-1;k<2;k++) {gk=k;
       glVertex3f(gi     ,gj     ,gk      ); //draw corners
       glVertex3f(gi*Frac,gj     ,gk      );
       glVertex3f(gi     ,gj     ,gk      );
       glVertex3f(gi     ,gj*Frac,gk      );
       glVertex3f(gi     ,gj     ,gk      );
       glVertex3f(gi     ,gj     ,gk*Frac );
   }/*k*/ }/*j*/} /*i*/
 glEnd();

 glLineWidth(3.f);
 glBegin(GL_LINES);
   Colors3D( 7); glVertex3f(0.,0.,0.); glVertex3f(1.,0.,0.);
   Colors3D( 5); glVertex3f(0.,0.,0.); glVertex3f(0.,1.,0.);
   Colors3D( 3); glVertex3f(0.,0.,0.); glVertex3f(0.,0.,1.);
 glEnd();
 glFlush();
 //Axis labels:
 h4Fill(    Xyzh,         .8e0   , 0.e0,-.01e0   , 1.e0          );
 h4Fill(          Rpyh,  0.e0   , 0.e0,  0.e0   , 1.e0          );
```

```
    Alpha6D( Xyzh, Rpyh,    1    , .1e0,  2.     ,  7     ,"+X\0");
  //Alpha6D(Pxyzh,ArpyDh,ModeFlag,SizeH,LineWidth,iCol    ,Label );

  h4Fill( Xyzh,  0.e0,  .8e0,-.01e0, 1.e0);
  h4Fill( Rpyh,  0.e0,  0.e0, 90.e0, 1.e0);
    Alpha6D( Xyzh, Rpyh,    1    , .1e0,  2.     ,  5     ,"+Y\0");

  h4Fill(Xyzh , .01e0,  0.e0,  .8e0, 1.e0);
  h4Fill(Rpyh ,  0.e0,-90.e0,  0.e0, 1.e0);
    Alpha6D( Xyzh, Rpyh,    1    , .1e0,  2.     ,  3     ,"+Z\0");
    }
}//End CubeGrid ------------------------------------------------------------


int Brk(int LineNum, int n,int m) /*v:0.5---------------------------------------
  "Breaker" - a progressive visualization tool - rev. A.
  When Brk is turned on - by setting S.BrkOn <>0,
    then each call to Brk increments S.BrkCur and returns zero
      until S.BrkCur>=S.BrkLim at which point Brk returns one.
                    S.BrkLim is incremented by '+' & decremented by '-'.
  Thus,insert:"if(Brk(Linenum,n,m)) return;" at desired incremental breakpoints

  "App7-OverWriter.c" uses this tool to step through matrix inversions.
  2016.06.24.1600cdt JMS                                                    */
{
 if(S.BrkOn==0)                                                  return 0;
 if(S.BrkLim <1)                    S.BrkLim     =           1;
 if((S.KbdKey==43)||
    (S.KbdKey==61)  ) {   S.KbdKey=0; S.BrkLim     =  S.BrkLim+1;}  // '+'
 if (S.KbdKey==45   ) {   S.KbdKey=0; S.BrkLim     =  S.BrkLim-1;
                     if(S.BrkLim<1) S.BrkLim     =           1;}  // '-'

                                    S.BrkCur     =  S.BrkCur+1;
                                    S.BrkLineNum =    LineNum;
                                    S.BrkN       =    n;
                                    S.BrkM       =    m;
 if(S.BrkCur < S.BrkLim){           S.BrkVal     =    0;           return 0;}
                                    S.BrkCur     =  S.BrkLim;
  sprintf(S.BrkLabelOut,"Line#=%3.1i, BrkCur=%5.1i, [%3.1i ] [%3.1i ]\0",
                          S.BrkLineNum,S.BrkCur,S.BrkN,S.BrkM);
                                    S.BrkVal     =    1;           return 1;
}//End Brk -----------------------------------------------------------------


void HindSight(void) /*v:0.5---------------------------------------------------
  2016.06.26.1400cdt JMS- Added S.MmMenuView & BufferMmMenuView use.
  2016.06.24.1600cdt JMS- Visualization Control: Projection, ModelView, etc. */
{                                                                           /*
 Hindsight variables:
 FrustumDefIn[  8] breaks out as follows:                                  */
 double E,N,F,L,R,T,B,D,
        e,    l,r,t,b,d; //Intermediates:
 double Mnf,Bnf,Xs0;     //  -Depth
 double Mlr,Blr;         //  -Lateral
 double Mtb,Btb;         //  -Vertical

           //Arbitrarily using inches as the unit of measurement...
           //   and assuming an eye separation of 2.4 inches...
           //         E   ,   N     ,      F  ,   L   ,   R   ,  T   ,  B   ,   D
 double Bq[8]       ={ 1.2, -8.000, 24.00 ,-10.4, 10.4,-6.5, 6.5, 24.0 };
           //       { 1.2, -8.000, 24.00 ,-10.4, 10.4,-6.5, 6.5, 24.0 }
           //                              for more near-field depth resolution.
           //       { 1.2,-11.994, 24000.,-10.4, 10.4,-6.5, 6.5, 24.0 }
           //                              far depth cutoff is ~.4 miles away.
 double h44[4][4];
 int    i,iEye,j,k,iSide,iColor,MtxMode[1],m,n;
 double Xyzh[4],Rpyh[4],hSerial[16];
 float  Xyz[3];
 char PText[80];
 int  nRow;
```

```
  switch(S.ThreePhase) //-----------------------------------
   {case(1):/*Phase One of ThreePhase: Transfers, init.'s, & non-screen stuff:*/
     if(S.AppInit[0]<1)
      {
       for(i=0;i<8;i++) { S.FrustCoes[i]=Bq[i]; } //Set default Frustum coeff.'s
        S.VuMode               = 1;
        S.Scale                 = 5.e0;
//      S.iEye=0;
        S.AppInit[0]=1;

      }/*(S.AppInit[0])*/

       //User numeric computation BeZumes->  ~~~~~~~~~~~~~~ S.ThreePhase=1 ~~~//
                                                                              //
       AccessYourActiveApp();                                                 //
                                                                              //
       //User numeric computation <- BrEnds  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~//

    /*-------------------------------------------------------*/            break;

    case(2):/*Phase Two of ThreePhase- : Two- 2D screen graphics:             */

     h4Fill(Xyzh ,(-S.xyWindowRatio),  .95e0, 0.e0, 1.e0);
     h4Fill(Rpyh,   0.e0            , 0.e0  , 0.e0, 1.e0);  /*
     Alpha6D(    ,    ,ModeFlag, SizeH, LineWidth, iCol,Label[~38] );*/
     Alpha6D(Xyzh,Rpyh, 0      , .03e0, 3.       ,  1  ,S.AppName  );

     if(S.MmMenuView==1)                              //Added 2016.06.26.1400cdt
      {h4Fill(Xyzh, -1.e0, .5e0 , 0.e0, 1.e0);                                //
       h4Fill(Rpyh, 90.e0, 0.e0 , 0.e0, 1.e0);                                //
       VecText7D(Xyzh,Rpyh,.03e0, 2.,1,BufferMmMenuView );          return;//
      }                                                                       //

       //User 2D drawing BeZumes->  ~~~~~~~~~~~~~~~~~~~~~~ S.ThreePhase=2 ~~~//
                                                                              //
       AccessYourActiveApp();                                                 //
                                                                              //
       //User 2D drawing <- BrEnds  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~//

//   switch(S.VuMode)
//   {
     /* The geometry of a one-eye (2D) perspective view is called a "frustum".
          3D involves two-eyes and hence calls for two "frustums".
        In what follows, if 'E' = "Eye Offset" = 0.
          then the two 3D frustums become a single 2D frustum.

                             //You start out with your nose...
                  [the point on/in your nose that is exactly half way between
                       the optical centers of your right and left eyeballs]
                           ...at the origin of "your 3D world"
                               looking in the +X. direction -with-
                                              +Y to your right -and-
                                              +Z  down.
      which is THE right-handed "Standard Flight Simulation Coordinate system"*/

      //                          3D'ness:

       E     = S.FrustCoes[0]; /*'E' = "Eye Offset" -the optical center of your
                                       right eyeball in the +Y direction.
                                            Default value: E = D / 20.

      /*             Orthographic & 2D Field-of-view cutoffs:

                                 Forward:                                     */
        N   = S.FrustCoes[1]; /*'N' = "Near" - your near view cutoff
                                   with respect to the origin
                                   on the X axis.      -D < N  */
          F = S.FrustCoes[2]; /*'F' = "Far"  - your far-field cutoff
                                   with respect to the origin
```

```
                                        on the X axis.       N < F
                          Lateral:                                  */

      L   = S.FrustCoes[3]  //'L' = "Left" - your left-field cutoff
           /S.FovYZzoom;    /*                with respect to the origin
                                              on the Y axis.*/
        R = S.FrustCoes[4]  //'R' = "Right"- your right-field cutoff
           /S.FovYZzoom;    /*                with respect to the origin
                                              on the Y axis.       L < R
                          Vertical:                                 */
      T   = S.FrustCoes[5]  //'T' = "Top"  - your upper-field cutoff
           /S.FovYZzoom;    /*                with respect to the origin
                                              on the Z axis.*/
        B = S.FrustCoes[6]  //'B' ="Bottom"- your lowerr-field cutoff
           /S.FovYZzoom;    /*                with respect to the origin
                                              on the Z axis.       T < B  */

//                    Projection depth reference:

  D     = S.FrustCoes[7]; /*You push the origin forward by distance 'D'
                          ... hence, in a spatial sense, you now have
                                        ***Hindsight*** !!!
                          & I've chosen the sign convention  D > 0.   */

S.E=E; S.N=N; S.F=F; S.L=L; S.R=R; S.T=T; S.B=B; S.D=D;
/*                    *** Compute the frustums: ***
  Morph your (F-N) (R-L) (B-T) world into OpenGL's signed unit screen cube.

 "Intermediate variables"
  Compute the frustum's intermediate variables (glorified: y=m*x+b):     */

for(iSide=-1; iSide<2; iSide=iSide+2) //For the Left eye, then the right:
 {S.iSide=iSide ;    e=E ;l=L;    r=R;
                     d=D ;t=T;    b=B;    iColor=-1;
  if((S.VuMode<2) && (iSide>0)) break;
  switch(S.VuMode) // Juggle each eye's lateral location (frustum bashing):
   {
    case(-1):     /* 2D LeftHand rawGL: skip the 2nd pass when e=0.
                     ...an identity matrix.
                      'S.Frustum3h' will be loaded with the identity matrix
                         at the end S.ThreePhase case(2) below.
                      OpenGL automatically clips the signed unit cube.   */
                                                  goto RawGLOverwrite;

    case( 0):     /* 2D Orthographic  : skip the 2nd pass when e=0.
                       The Exact answer is below at:                    */
                                               if(1>0) goto OrthoOverwrite;
                   /*The clipping planes are the boundaries of
                     OpenGL automatically clips the signed unit cube.

                     ...For a close approximation, you could move hindsight
                          a million times farther back instead.         */
                          d=D*1.e6;      e=0.e0;break;

    case( 1):     // 2D Perspective: skip the 2nd pass when e=0.        */
                                         e=0.e0;break;

    case( 2):  // 3D  Red&Cyan
     switch(iSide) {case(-1):             e=-E;break; //Left  eye - Red
                    case( 1):             e= E;break; //Right eye - Cyan
                   }                          ;break;

    case( 3):  /* 3D Right|Left: double the lateral FoV on the other side
                                & double the vertical FoV:              */
     switch(iSide) {case(-1):l=2.*L-R   ;e=-E  ;break;//Left  eye -> right
                    case( 1):r=2.*R-L   ;e= E  ;break;//Right eye <-  left
                   }          t=2.*T     ;b=2.*B;break;

    case( 4):  /* 3D Left|Right: double the lateral FoV on the same side
                                & double the vertical FoV:              */
```

```
         switch(iSide) {case(-1):r=2.*R-L   ;e=-E  ;break;//Left  eye <-  left
                        case( 1):l=2.*L-R   ;e= E  ;break;//Right eye -> right
                       }        ;t=2.*T     ;b=B*2.;break;
       }//S.VuMode)

      Mnf = 2.*(N+d)*(F+d)/(N-F);Bnf=-((N+F)+2.*d)/(N-F);//~y=Mx+B for depth*
      Mlr = 2.            /(r-l);Blr=- (r+l)      /(r-l);// y=Mx+B for lateral
      Mtb = 2.            /(b-t);Btb=- (b+t)      /(b-t);// y=Mx+B for vertical
                                    // (*Note: depth isn't really linear.)

      /*Each eye's frustum (a homogeneous matrix) is defined
        in the next five lines of code:                                    */
      h44Fill(   h44      ,                                 //Visually: this is
             (+Mlr*e+Blr),(+d*Mlr),( 0    ),(+d*Blr    ), // the exact
             (      -Btb),( 0    ),(-d*Mtb),(-d*Btb    ), //  symbolic
             (      +Bnf),( 0    ),( 0    ),(+d*Bnf+Mnf), //   homogeneous
             (+1         ),( 0    ),( 0    ),(+d        ));//    solution.
      //                        ... and saved:
                                                      goto FillFrustum3h;

  RawGLOverwrite: //<- target of a goto above (from/when: S.VuMode==-1)

       //2D LeftHand rawGL: Is an identity matrix:
       h44Fill( h44 ,(1.e0),(0.e0),(0.e0),(0.e0),
                     (0.e0),(1.e0),(0.e0),(0.e0),
                     (0.e0),(0.e0),(1.e0),(0.e0),
                     (0.e0),(0.e0),(0.e0),(1.e0) );        goto FillFrustum3h;

  OrthoOverwrite:  //2D Orthographic... in Flight Simulation Coordinates:
      h44Fill(   h44      ,
             ( 0.e0      ),( 2.e0/(R-L)),( 0.e0      ),(-(R+L)/(R-L)),
             ( 0.e0      ),( 0.e0      ),(-2.e0/(B-T)),( (B+T)/(B-T)),
             (-2.e0/(N-F)),( 0.e0      ),( 0.e0      ),( 0.e0       ),
             ( 0.e0      ),( 0.e0      ),( 0.e0      ),( 1.e0       ));
                                                      goto FillFrustum3h;

  FillFrustum3h: //<- target of a goto above.
      for(i=0;i<4;i++) { for(j=0;j<4;j++)
       { S.Frustum3h[i][j][iSide+1] = h44[i][j];
       }/*j*/          }/*i*/

      S.e[iSide+1]=e;

 //HindSight self-visualization of Projection: -------------------------
      if((S.AppNumber==8)&&(S.NowView==1))
       {                                                              nRow=9;
        sprintf(PText,"Frustum coefficients modified by FovYZzoom:\0");
        if(iSide==-1) PrntOrtho(nRow,53+iSide*51   , 5,0,PText);      nRow++;
        sprintf(PText,"E[0]=%14.6f N[1]=%14.6f L[3]=%14.6f\0"
                     , E          ,N          ,L            );
        if(iSide==-1) PrntOrtho(nRow,53+iSide*51   , 5,0,PText);
        sprintf(PText," T[5]=%14.6f D[7]=%14.6f\0"
                     , T          ,D          );
        if(iSide==-1) PrntOrtho(nRow,53+iSide*51+59, 5,0,PText);      nRow++;
        sprintf(PText,"                    F[2]=%14.6f R[4]=%14.6f\0"
                     ,                     F          ,R            );
        if(iSide==-1) PrntOrtho(nRow,53+iSide*51   , 5,0,PText);
        sprintf(PText," B[6]=%14.6f YZzoom:%12.6f\0"
                     , B          ,S.FovYZzoom     );
        if(iSide==-1) PrntOrtho(nRow,53+iSide*51+59, 5,0,PText);      nRow++;

        sprintf(PText,"e   =%14.6f                    l   =%14.6f\0"
                     , e                           ,l            );
        if(S.VuMode>0)PrntOrtho(nRow,53+iSide*51   , 5,0,PText);
        sprintf(PText," t   =%14.6f d   =%14.6f\0"
                     , t          ,d          );
        if(S.VuMode>0)PrntOrtho(nRow,53+iSide*51+59, 5,0,PText);      nRow++;
        sprintf(PText,"r   =%14.6f b   =%14.6f\0"
                     , r          ,b          );
        if(S.VuMode>0)PrntOrtho(nRow,53+iSide*51+40, 5,0,PText);      nRow++;
```

```
        sprintf(PText,"Mnf =%14.6f Mlr =%14.6f Mtb =%14.6f\0"
                     , Mnf        ,Mlr        ,Mtb          );
        if(S.VuMode>0)PrntOrtho(nRow,53+iSide*51+20, 5,0,PText);        nRow++;
        sprintf(PText,"Bnf =%14.6f Blr =%14.6f Btb =%14.6f\0"
                     , Bnf        ,Blr        ,Btb          );
        if(S.VuMode>0)PrntOrtho(nRow,53+iSide*51+20, 5,0,PText);        nRow++;
        switch(S.VuMode)
         {case(-1):  // 2D LeftHand rawGL: skip the 2nd pass when e=0.
          case( 0):  // 2D Orthographic  :  "
          case( 1):  // 2D Perspective   :  "
                        sprintf(PText,"Projection Matrix\0");           break;
          case( 2):  // 3D   Red&Cyan
          case( 3):  // 3D Right|Left:
          case( 4):  // 3D  Left|Right:
           if(iSide==-1){sprintf(PText,"Projection Matrix- Left  Eye\0");}
           if(iSide!=-1){sprintf(PText,"Projection Matrix- Right Eye\0");}
                                                                        break;
         }/*(S.VuMode)*/
        Seeh44d(  h44,nRow,53+iSide*51, 9,PText); nRow=nRow+5;
       }//((S.AppNumber==8)&&(S.NowView>0)) -------------------------


     }//(iSide)-----------------------------------------

    Xs0 = 2.*(F*N-D*D)/(F+N+2.*D)+D; /*Advises you of Where
                                       screen depth encoding = 0.
                                       in your modelling volume.*/

    //--- Define the OpenGL lateral "clipping planes" for 3D use:
    h4Fill(S.ClipPlane1,(-L-E),( D),(0.),(-D*L)); // Left - far  field left
    h4Fill(S.ClipPlane2,( R+E),(-D),(0.),( D*R)); //      - near field right
    h4Fill(S.ClipPlane3,(-L+E),( D),(0.),(-D*L)); // Right- near field left
    h4Fill(S.ClipPlane4,( R-E),(-D),(0.),( D*R)); //      - far  field right
    //---  ... and inform OpenGL of the results:
    glClipPlane(GL_CLIP_PLANE1,S.ClipPlane1);
    glClipPlane(GL_CLIP_PLANE2,S.ClipPlane2);
    glClipPlane(GL_CLIP_PLANE3,S.ClipPlane3);
    glClipPlane(GL_CLIP_PLANE4,S.ClipPlane4);                          glFlush();
    //--- Define the model's rotation:   (uses mouse & '<''>' inputs)
    h4Fill(Xyzh , 0.e0, 0.e0, 0.e0, 1.e0);
    h4Fill(S.Attitudeh,S.KbdRoll,S.MousePitch,S.MouseYaw,1.e0);
    PAhXR(Xyzh,S.Attitudeh,S.Attitudeh44);//<-augmented Mouse 3DoF rotation
    //--- Compute the 3D Panning and Scaleing homogeneous matrix;
    h44Fill(S.PoIScaleh44,
      ( S.Scale), ( 0.e0), ( 0.e0), (-S.Scale*S.PoIX),
      ( 0.e0), ( S.Scale), ( 0.e0), (-S.Scale*S.PoIY),
      ( 0.e0), ( 0.e0), ( S.Scale), (-S.Scale*S.PoIZ),
      ( 0.e0), ( 0.e0), ( 0.e0), ( 1.e0       )  );
    //   ...and prepare the 1D version that glMultMatrixd() will accept;
    for(m=0;m<4;m++){for(n=0;n<4;n++)
                      {S.PoIScaleh16[4*m+n]=S.PoIScaleh44[n][m]; } }

//HindSight self-visualization of ModelView: --------------------------
      if(S.AppNumber==8 && S.NowView==1)
       {sprintf(PText,"Screen depth zero is at X=%14.6f\0",Xs0);
                    PrntOrtho(nRow,2,11,0,PText);                       nRow++;
        if(S.VuMode>1)
       {sprintf(PText,"Clipping Planes:\0");
                    PrntOrtho(nRow,2  ,1,0,PText);                      nRow++;
        Seeh4d( S.ClipPlane1  ,nRow,2, 5
               ,"     -L-E          D             0.         -D*L"
                "   :Left - far  field left\0");                        nRow=nRow+2;
        Seeh4d( S.ClipPlane2  ,nRow,2, 5
               ,"      R+E         -D             0.          D*R"
               "          - near field right\0");                      nRow=nRow+2;
        Seeh4d( S.ClipPlane3  ,nRow,2, 5
               ,"     -L+E          D             0.         -D*L"
                "   :Right- near field left :\0");                      nRow=nRow+2;
        Seeh4d( S.ClipPlane4  ,nRow,2, 5
               ,":     R-E          -D             0.          D*R"
```

```
                "         - far  field right\0");                    nRow=nRow+2;
       }
                                                                     nRow=31;
        if( S.Nutate)
         {
        //Seeh4d( S.NutateAng  ,nRow,2,11
        //      ,"     Roll          Pitch         Yaw    1.e0\0");nRow=nRow+2;
          Seeh44d(S.Nutate44   ,nRow,2,11,
      "Model  Nutation Matrix: (provides monocular depth cues\0"); nRow=nRow+5;
         }

        Seeh4d( S.Attitudeh  ,nRow,2, 6
               ,"S.KbdRoll,S.MousePitch,S.MouseYaw,1.e0\0");       nRow=nRow+2;
        Seeh44d(S.Attitudeh44,nRow,2, 6,
                "Model  Rotation Matrix (3dof):\0");               nRow=nRow+5;
        h4Fill(Xyzh ,S.PoIX,S.PoIY,S.PoIZ,S.Scale);
        Seeh4d(Xyzh  ,nRow,2, 9
               ,"S.PoIX,S.PoIY,S.PoIZ,S.Scale\0");                 nRow=nRow+2;
        Seeh44d(S.PoIScaleh44,nRow,2, 9,
                "Model Pan/Scale Matrix (4dof):\0");               nRow=nRow+5;
       }//(S.AppNumber==8 && S.NowView==1) ---------------------------

   /*-------------------------------------------------------*/            break;

   case(3):/*Phase Three of ThreePhase: 3D screen graphics:                  */
    if(S.MmMenuView==1)                       return; //Added 2016.06.26.1400cdt

    //Taking direct control of OpenGL's Projection 7 ModelView matrices:
    glGetIntegerv(GL_MATRIX_MODE,MtxMode);
    //Save the Texture matrix
    glMatrixMode(GL_TEXTURE   ); glPushMatrix(); glLoadIdentity();
    //Save the Modelview matrix:                  //glPushAttrib(___);
    glMatrixMode(GL_MODELVIEW ); glPushMatrix(); glLoadIdentity();     glFlush();
    //Save the Projection matrix: &
    glMatrixMode(GL_PROJECTION); glPushMatrix(); glLoadIdentity();     glFlush();

    S.ForceRGBA=0;
    h4Fill(Rpyh, 0., 0., 0., 1.);
    switch(S.VuMode)
     {
      case(-1):
       h4Fill(Xyzh , -.18, -.95, -1., 1.);
       Alpha6D(Xyzh,Rpyh,0,.03e0, 2.,11,"LeftHand rawGL\0"); break;
      case( 0):
       h4Fill(Xyzh , -.18, -.95, -1., 1.);
       Alpha6D(Xyzh,Rpyh,0,.03e0, 2.,11,"2D Orthographic\0"); break;
      case( 1):
       h4Fill(Xyzh , -.18, -.95, -1., 1.);
       Alpha6D(Xyzh,Rpyh,0,.03e0, 2.,11,"2D Perspective\0"); break;
      case( 2):
       h4Fill(Xyzh , -.179e0, -.949e0, -.98e0, 1.e0                    );
       Alpha6D(Xyzh, Rpyh   , 0      ,  .03e0, 5.  ,15, "3D Red/Cyan\0");   //=0?
       h4Fill(Xyzh , -.18e0, -.95e0, -.99e0, 1.e0                      );
       Alpha6D(Xyzh, Rpyh   , 0      , .030e0, 2.  , 1, "3D Red/Cyan\0");  break;
    /*case( 2):
       h4Fill(Xyzh , -.06, -.95, -1., 1.);
       Alpha6D(Xyzh,Rpyh,0,.03e0, 2.,5,"3D Red\0");
       h4Fill(Xyzh , -.08, -.95, -1., 1.);
       Alpha6D(Xyzh,Rpyh,0,.03e0, 2.,4,"3D Cyan\0"); break;*/
      case( 3):
       h4Fill(Xyzh , -.66, -.95, -1., 1.);
       Alpha6D(Xyzh,Rpyh,0,.03e0, 2.,11,"3D Right View\0");
       h4Fill(Xyzh ,  .34, -.95, -1., 1.);
       Alpha6D(Xyzh,Rpyh,0,.03e0, 2.,11,"3D  Left View\0"); break;
      case( 4):
       h4Fill(Xyzh , -.66, -.95, -1., 1.);
       Alpha6D(Xyzh,Rpyh,0,.03e0, 2.,11,"3D  Left View\0");
       h4Fill(Xyzh ,  .34, -.95, -1., 1.);
       Alpha6D(Xyzh,Rpyh,0,.03e0, 2.,11,"3D Right View\0"); break;
     }/*(S.VuMode)*/
```

```
    for(iEye=0;iEye<3;iEye=iEye+2) //...load each eye's view:
     { S.iSide=iEye-1;
      glMatrixMode(GL_PROJECTION); glLoadIdentity();                  glFlush();
     //Load the Projection Matrix with a single 4x4 matrix:----------
      k=0; for(m=0; m<4; m++){ for(n=0;n<4;n++)
            {hSerial[k]=S.Frustum3h[n][m][iEye];k=k+1; }/*n*/ }/*m*/
      glMultMatrixd(hSerial);

      glMatrixMode(GL_MODELVIEW ); glLoadIdentity();                  glFlush();
      //Mark the lateral viewing bounds with vertical lines:
      glLineWidth(2.f); glShadeModel(GL_FLAT); Colors3D(11);
      glBegin(GL_LINES); glVertex3f(.0,L     ,T );glVertex3f(.0,L    ,B);
                         glVertex3f(.0,R     ,T );glVertex3f(.0,R,B);glEnd();
      //Enable the lateral clipping planes:
      if(S.VuMode>1) { glEnable(GL_CLIP_PLANE1);
                       glEnable(GL_CLIP_PLANE2);
                       glEnable(GL_CLIP_PLANE3);
                       glEnable(GL_CLIP_PLANE4); }
      //Mark the screen center (i.e. model Yaw=0. ,Pitch=0.) with a point:
      glPointSize(3.f); Colors3D(10);
      glBegin(GL_POINTS);glVertex3f(0.,0.    ,0.);                      glEnd();

     //Model Nutation:-----------------------------------------------
      //Concatinate the nutation matrix:
      if( S.Nutate)
      {k=0;for(m=0;m<4;m++)
            {for(n=0;n<4;n++){hSerial[k]=S.Nutate44[n][m];k=k+1; }/*n*/}/*m*/
       glMultMatrixd(hSerial);
      }/*( S.Nutate)*/
     //Setup Modelview's 7DoF:---------------------------------------
      //Concatinate model 3DoF rotation:
      k=0;for(m=0;m<4;m++)
           {for(n=0;n<4;n++){hSerial[k]=S.Attitudeh44[n][m];k=k+1; }/*n*/}/*m*/
      glMultMatrixd(hSerial);
      //Concatinate model 4DoF pan& Scale:
      glMultMatrixd(S.PoIScaleh16);

      if(S.VuMode==2) //Color control when using Red/Cyan 3D glasses...
       {                S.ForceRGBA= 0;
         if(iEye==0) {  Colors3D(3);           //Color Red...
                        glColorMask(1,0,0,1);} //Mask  Left eye - Red
         if(iEye==2) {  Colors3D(8);           //Color Cyan...
                        glColorMask(0,1,1,1);} //Mask Right eye - Cyan
                      S.ForceRGBA=-1;          //Locks in the color value.
                             glEnable(GL_COLOR_LOGIC_OP);
         if(S.iScrnColor==15) glLogicOp(GL_OR);           //for Black screen
         if(S.iScrnColor== 1) glLogicOp(GL_COPY_INVERTED); //for White screen
         //Right & left eye 3D views have depth coexistence in both colors.
                             glClear(GL_DEPTH_BUFFER_BIT);            glFlush;
       }/*(S.VuMode==2)*/


      //User 3D drawing BeZumes-> ~~~~~~~~~~~~~~~~~~~~~~~ S.ThreePhase=3 ~~~//
                                                                          //
      AccessYourActiveApp();                                              //
                                                                          //
      //Teapot();                                                         //
                                                                          //
      //User 3D drawing <- BrEnds ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~//

      //Visualize the depth Array in 3D:
      if(S.DepthSelfie==2)
       {glPointSize(1.f); glShadeModel(GL_FLAT); Colors3D( 5);
        glBegin(GL_POINTS);
          for(i=0;i<1200;i++) {for(j=0;j<1920;j++)
           {Xyz[0]=2.*S.ScreenDepthMap[i][j]-1.;
            Xyz[1]= ( 2.*j-1920.)/1920.;
            Xyz[2]= (-2.*i+1200.)/1200.;
            if(Xyz[0]<1.) glVertex3fv(Xyz);
```

```
           }/*j*/              }/*i*/
        glEnd();
        // Label the depth values:
        h4Fill( Rpyh,-90.e0, 0.e0,90.e0, 1.e0);
        h4Fill( Xyzh, -1.e0, 1.e0, 1.e0, 1.e0);
        Alpha6D(Xyzh,Rpyh,0,.06e0,1.,11," Depth  0.0\0");
        h4Fill( Xyzh,  0.e0, 1.e0, 1.e0, 1.e0);
        Alpha6D(Xyzh,Rpyh,0,.06e0,1.,11," Depth +0.5\0");
        h4Fill( Xyzh,  1.e0, 1.e0, 1.e0, 1.e0);
        Alpha6D(Xyzh,Rpyh,0,.06e0,1.,11," Depth +1.0\0");

       }/*(S.DepthSelfie==2)*/

      //<-End - User 3D drawing.  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      if(S.VuMode==2) { glDisable(GL_COLOR_LOGIC_OP);
                        glColorMask(1,1,1,1);
                        S.ForceRGBA=0;              }

      //Disable the lateral clipping planes:
      glMatrixMode(GL_PROJECTION);
      glDisable(GL_CLIP_PLANE1);
      glDisable(GL_CLIP_PLANE2);
      glDisable(GL_CLIP_PLANE3);
      glDisable(GL_CLIP_PLANE4);                                        glFlush();
                                                         if(S.VuMode<2) break;
     }/*(iEye)*/

 switch(S.DepthSelfie)
  {case(1)://Create the depth Array for every screen pixel
     glReadPixels(     0,     0,    1920,1200,
                   GL_DEPTH_COMPONENT,GL_FLOAT,&(S.ScreenDepthMap));
     S.DepthSelfie=2;                                                       break;
   case(2)://Visulization is handled inside the 3D drawing loop above.
                                                                            break;
   default://Vanish the depth Array:
      S.DepthSelfie=0;                                                      break;
  }/*(S.DepthSelfie)*/

    // ~ Restore the pre-call visual environment:
    glMatrixMode(GL_PROJECTION); glPopMatrix();
    glMatrixMode(GL_TEXTURE    ); glPopMatrix();
    glMatrixMode(MtxMode[1]    );                                       glFlush();

   /*----------------------------------------------------------*/           break;
  }/*(S.ThreePhase)*/

}//End HindSight --------------------------------------------------------------


void SeeFrustums(void) /*v:0.5-----------------------------------------------
  2016.07.06.1530cdt JMS- Update of Buffer0 mathematics description & scale ctl
  2016.06.24.1600cdt JMS- Draws frustum outlines in 3D                       */
{ int i,iC,iCu,iSide,j,k,n;
  double x,y,FrustP[4],H[4][4],Hinv[4][4],Clip[4],ClipK,TBmult;
  double static Vertex[16][4] =
    {   0.e0,  0.e0, 0.e0,1.e0    // 0
     ,  0.e0,  0.e0, 1.e0,1.e0    // 1:+depth ?
     ,  1.e0,  0.e0, 0.e0,1.e0    // 2:+right ?
     ,  0.e0, -1.e0, 0.e0,1.e0    // 3:+down  ?
                                  //Near
     , -1.e0,  1.e0,-1.e0,1.e0    // 4:upper- left
     ,  1.e0,  1.e0,-1.e0,1.e0    // 5:      - right
     , -1.e0, -1.e0,-1.e0,1.e0    // 6:lower- left
     ,  1.e0, -1.e0,-1.e0,1.e0    // 7:      -right
                                   //Middle
     , -1.e0,  1.e0, 0.e0,1.e0    // 8:upper- left
     ,  1.e0,  1.e0, 0.e0,1.e0    // 9:      - right
     , -1.e0, -1.e0, 0.e0,1.e0    //10:lower- left
     ,  1.e0, -1.e0, 0.e0,1.e0    //11:      -right
                                  //Far
```

```
      , -1.e0,  1.e0, 1.e0,1.e0    //12:upper- left
      ,  1.e0,  1.e0, 1.e0,1.e0    //13:     - right
      , -1.e0, -1.e0, 1.e0,1.e0    //14:lower- left
      ,  1.e0, -1.e0, 1.e0,1.e0    //15:     -right
    };// L/R, -T/-B,  N/F,1.       //Unit screen cube.
  int static iConn[19][2] =
    { 0, 1,  0, 2,  0, 3,
      4, 5,  6, 7,  4, 6,  5, 7,
      8, 9, 10,11,  8,10,  9,11,
     12,13, 14,15, 12,14, 13,15,
      4,12,  5,13,  6,14,  7,15};

  double Xyzh[4],Rpyh[4];
  char static buffer0[8100]=
{"2D/3D Visualization Transforms:      (Public Domain)\n\n"
"Projection Matrix \"Concatenation\" Sequence (six matrices) :\n\n"

"#1. Offset model by X=X+D & Y=Y-E*LR :                      <-- Welcome  to\n"
"| +1  ,  0  ,  0  , +D    |      LR =-1. for left eye          HindSight!!\n"
"|  0  , +1  ,  0  , -E*LR |         = 0. for perspective\n"
"|  0  ,  0  , +1  ,  0    |         =+1. for right eye\n"
"|  0  ,  0  ,  0  , +1    |\n\n"

"#2. Project YZ onto plane X=D & map X->[-1,+1] :\n"
"| +Bnf,  0  ,  0  , +Mnf  |\n"
"|  0  , +D  ,  0  ,  0    |\n"
"|  0  ,  0  , +D  ,  0    |\n"
"| +1  ,  0  ,  0  ,  0    |\n\n"

"#3. Offset model by Y=Y+E*LR :\n"
"| +1  ,  0  ,  0  ,  0    |\n"
"|  0  , +1  ,  0  , +E*LR |\n"
"|  0  ,  0  , +1  ,  0    |\n"
"|  0  ,  0  ,  0  , +1    |\n\n"

"#4.  Y->[-1,+1], Z->[-1,+1] :\n"
"| +1  ,  0  ,  0  ,  0    |\n"
"|  0  , +Mlr,  0  , +Blr  |\n"
"|  0  ,  0  , +Mtb, +Btb  |\n"
"|  0  ,  0  ,  0  , +1    |\n\n"

"#5. Screen subsetting for split-screen 3D modes :\n"
"| +1  ,  0  ,  0  ,  0    |\n"
"|  0  , +Ms ,  0  , +Bsh  |  <-- Screen- horizontal- scale factors\n"
"|  0  ,  0  , +Ms , +Bsv  |  <--       -  vertical\n"
"|  0  ,  0  ,  0  , +1    | Adjusting L,R,T,& B turns this into\n"
"                            an identity matrix... the factors go away.\n\n"
"#6. Convert to OpenGL(Y,-Z,X) left-handed coordinates :\n"
"|  0  , +1  ,  0  ,  0    |\n"
"|  0  ,  0  , -1  ,  0    |\n"
"| +1  ,  0  ,  0  ,  0    |\n"
"|  0  ,  0  ,  0  , +1    |\n\n"

"The concatenated projection matrix is #6<#5<#4<#3<#2<#1 :\n"
"|+Ms*(+Mlr*E*LR+Blr)+Bsh ,+Ms*Mlr*D , 0        ,+Ms *D*(Blr+Bsh)|\n"
"|-Ms*           Btb -Bsv , 0        ,-Ms*Mtb*D ,-Ms *D(*Btb-Bsv)|\n"
"|+Bnf                    , 0        , 0        ,    +D *Bnf+Mnf |\n"
"|+1                      , 0        , 0        ,    +D          |\n"
"       ... per my symbolic matrix concatenator & inverter;\n"
"           such a tool saves a lot of time & error.\n"
"          (Permutations can be evaluated symbolically.)\n\n"

"As implemented in function 'HindSight':\n"
" Substitute: 'e'=LR*E\n"
"'d'= D, but in the orthographic chase 'd'=infinity ...very large is close.\n"
" In split screen 3D,the left and right sides of the viewing frustums\n"
" are fudged. L,R,T,& B are modified to l,r,t,& b, centering & shrinking the\n"
" individual eye views onto their respective sides of the screen.\n"
"     Peripheral dissimilarities, large or small, are clipped away.\n\n"
```

```
"#2/#4's bias & scaleing factors map the viewing volume into a +-1 unit cube:"
"\n"
" Mnf = 2.*(N+d)*(F+d)/(N-F);Bnf=-((N+F)+2.*d)/(N-F);//~y=Mx+B for depth **\n"
" Mlr = 2.           /(r-l);Blr=- (r+l)      /(r-l);// y=Mx+B for lateral\n"
" Mtb = 2.           /(b-t);Btb=- (b+t)      /(b-t);// y=Mx+B for vertical\n"
"\n"
"    ** Note: depth isn't really linear. If N=-D/2. and F=+infinity,\n"
"   XS0 would still exactly coincide with the modelspace origin, & the\n"
"ability to resolve depth in the near field of view would remain excellent.\n"
"  For zero screen depth at X=0.,  use:\n"
"    N = -D*F/(D+2.*F)     when: F & D are predefined.\n"
"    F = -D*N/(D+2.*N)     when: N & D  \"       \"        & if: N= -D/ a\n"
"    D = -2._16*F*N/(F+N)  when: F & N  \"       \"        then: F= +D/(a-2.)\n"
"        Thus 'Depth Selfie's become a little easier to interpret.'\n"
"\n"
" Each eye's frustum (a homogeneous matrix) is defined\n"
"  in the next five lines of code:\n"
"h44Fill(   h44      ,                                    //Visually: this is\n"
"       (+Mlr*e+Blr),(+d*Mlr),( 0    ),(+d*Blr    ),  // the exact\n"
"       (      -Btb),( 0    ),(-d*Mtb),(-d*Btb    ),  //  symbolic\n"
"       (      +Bnf),( 0    ),( 0    ),(+d*Bnf+Mnf),  //   homogeneous\n"
"       (+1        ),( 0    ),( 0    ),(+d        ) );//    solution.\n\n"

"... except in the ORTHOGRAPHIC Case...\n"
"                when D=Infinity, the matrix terms blow up.\n"
"Algebraically dividing all the terms of the above matrix by 'd' yields:\n"
"h44Fill(   h44      ,\n"
"       ( 0.e0      ),( 2.e0/(R-L)),( 0.e0      ),(-(R+L)/(R-L)),\n"
"       ( 0.e0      ),( 0.e0      ),(-2.e0/(B-T)),( (B+T)/(B-T)),\n"
"       (-2.e0/(N-F)),( 0.e0      ),( 0.e0      ),( 0.e0       ),\n"
"       ( 0.e0      ),( 0.e0      ),( 0.e0      ),( 1.e0       ) );\n\n"

"The projection matrix is 'Zoomed' by variable S.FovYZzoom which divides\n"
"L,R,T,& B at the outset; zooming does not affect the screen depth range.\n\n"

"Use function 'PAhXR' for 6dof control of your model, which HindSight uses\n"
"- with position zeroed - to generate the model rotation matrix (3dof).\n\n"
//Added 2016.07.06:
"The rotation concatenation sequence in Flight Simulation coordinates is:\n\n"
"#1. Roll:    positive  Roll rotates +Y toward +Z ,   ~inner gimbal\n"
"| +1  ,  0  ,  0  ,  0    |\n"
"|  0  , +cR , -sR ,  0    |          sR=  sine(Roll)\n"
"|  0  , +sR , +cR ,  0    |          cR=cosine(Roll)\n"
"|  0  ,  0  ,  0  , +1    |\n\n"

"#2. Pitch:   positive pitch rotates +Z toward +X,    ~middle gimbal\n"
"| +cP ,  0  , +sP ,  0    |\n"
"|  0  , +1  ,  0  ,  0    |          sP=  sine(Pitch)\n"
"| -sP ,  0  , +cP ,  0    |          cP=cosine(Pitch)\n"
"|  0  ,  0  ,  0  , +1    |\n\n"

"#3. Yaw:     positive  yaw  rotates +X toward +Y     ~outer gimbal\n"
"| +cY , -sY ,  0  ,  0    |\n"
"| +sY , +cY ,  0  ,  0    |          sY=  sine(Yaw)\n"
"|  0  ,  0  , +1  ,  0    |          cY=cosine(Yaw)\n"
"|  0  ,  0  ,  0  , +1    |\n\n"

"The concatenated pure rotation matrix #3<#2<#1 is:\n"
"| cy*cp,  cy*sp*sr-sy*cr,  cy*sp*cr+sy*sr,  0 |\n"
"| sy*cp,  sy*sp*sr+cy*cr,  sy*sp*cr-cy*sr,  0 |\n"
"|   -sp,     cp*sr      ,     cp*cr      ,  0 |\n"
"|    0 ,        0       ,        0       , +1 |\n\n"

"Each of the upper-left 3x3 sub-matrices in the four matrices above\n"
"is a 'Direction Cosine Matrix', because the numerical values are the\n"
"cosines of the projection of each input axis onto each output axe...\n"
"which is why the result is a rigid rotation rather than a warp/'morph'.\n"
"For pure rotation matrices- the transpose is the inverse.\n"
"Direction Cosine matrices, once populated with numbers, are independent of\n"
"the 'angles' used to compute them. But if you don't know in which\n"
```

```
 "directions +X,+Y,& +Z are point, you've got a problem! \n\n"

 "In real-world processes - like manufacturing or navigating - not knowing\n"
 "the Six Degree-Of-Freedom (6dof) coordinate frame you're working in\n"
 "puts you on perilous ground. Questions to ask:\n"
 "    Where is the origin?\n"
 "    Where do +X, +Y, & +Z point, & what is the unit of measurement? Meters?\n"
 "    How are rotations defined & what is the unit of measurement? Degrees?\n\n"

 "HindSight's 3D viewer uses: \"Standard Flight Simulation Coordinates\"\n"
 " in ModelView space, described at/in:\n"
 "              www.setterholm.com in the /Geodesy subdirectory:\n"
 "               'qVPMath12-AppendixA-20091211.pdf'\n\n"

 "Note: 'Quaternions' provide another way of implementing model rotation\n"
 "       which has no specific gimbal sequence, but instead exactly\n"
 "       rotates around an arbitrarily-chosen axis. (The math is complex.)\n\n"

 "The Model Translation & Scaleing Sequence (two matrices,4dof):\n\n"
 // ~ End of additions 2016.07.06
 "#1. ReCenter on (i.e. translate to) the 'Point of interest':\n"
 "| +1    ,  0    ,  0    , -PoIX |\n"
 "|  0    , +1    ,  0    , -PoIY |\n"
 "|  0    ,  0    , +1    , -PoIZ |\n"
 "|  0    ,  0    ,  0    , +1    |\n\n"

 "#2.\"Scale\" (i.e. 3D Magnify):\n"
 "| +Scale,  0    ,  0    ,  0    |\n"
 "|  0    , +Scale,  0    ,  0    |\n"
 "|  0    ,  0    , +Scale,  0    |\n"
 "|  0    ,  0    ,  0    , +1    |\n\n"
 "Which concatinates to:\n"
 "    h44Fill(PoIScaleh44,\n"
 "      ( Scale), ( 0.e0 ), ( 0.e0 ), (-Scale*PoIX),\n"
 "      ( 0.e0 ), ( Scale), ( 0.e0 ), (-Scale*PoIY),\n"
 "      ( 0.e0 ), ( 0.e0 ), ( Scale), (-Scale*PoIZ),\n"
 "      ( 0.e0 ), ( 0.e0 ), ( 0.e0 ), ( 1.e0      )  );\n\n"

 "The Clipping Planes:\n\n"
 "Geometric planes are defined by a surface 'normal' (= 'perpendicular')\n"
 "vector and a distance. In the frustum(s) viewer - the directions of\n"
 "the four clipping plane normals are displayed & scaled to exactly touch\n"
 "their respective clipped planes.\n\n"

 "Press 'v' to see the numerical values live.\n\n"

 "Press 'b' to view: the eye viewpoints(s)                              -&-\n"
 " ...from here      the projection frustum(s) -mapped by inversion  -&-\n"
 "      & 's'        the clipping plane normals.\n\n"

 "3DEnv/HindSight, Version 0.6,  July 10,2016 Author: Jeffrey M. Setterholm\n"
 "                                                    8095 230th St. E.\n"
 "                                                Lakeville, Minnesota 55044\n"

 "\n                  Use these results at your own risk.\n"
 };
  PrntOrtho(4,2, 1,0,
  "Menu item 'Viewer Breaker' ='b' or 'B' toggles projection matrix info.\0");
  i=Brk(0,0,0); //This allows the '+' & '-' keys to change S.BrkLim
                // ... which will be used to scroll the text.
  if(S.BrkLim<   1) S.BrkLim=  1;
  if(S.BrkLim>1150) S.BrkLim=1150;  //Limits how far you can scroll.
  if(S.BrkOn>0)
   { if(S.Scale<.5e0) S.Scale=1.0;       //2016.07.06 allows some
     if(S.Scale>2.e0) S.Scale=1.0;       //2016.07.06   scale change
      PrntOrtho(5,2, 6,0,
     "Use the +- keys to scroll the text.\0");

     h4Fill(Xyzh,  0.e0, -4.e0 , -2.e0-double(S.BrkLim)/50.e0, 1.e0);
     h4Fill(Rpyh,  0.e0,  0.e0 , 90.e0               , 1.e0);
```

```
    VecText7D(Xyzh,Rpyh, .1e0 ,    1.    ,   6 ,  buffer0  );           return;
  }

                                                    k=5;
  for(n=0;n<16;n++)
   {//Seeh4d(&Vertex[n][0],k,2,15,"FrustP\0");    k=k+1;
   }/*j*/

  for(iSide=-1;iSide<2;iSide=iSide+2)
   { for(i=0;i<4;i++){for(j=0;j<4;j++){H[i][j]=S.Frustum3h[i][j][iSide+1];}}
    Invert44(H,Hinv,0,0);
    glLineWidth(2.0f);
    glShadeModel(GL_FLAT);
                  Colors3D(10);
    //Draw the viewing frustums:
    glBegin(GL_LINES);
      for(n=0;n<19;n++)
       {if(n==0) Colors3D( 7);
        if(n==1) Colors3D( 5);
        if(n==2) Colors3D( 3);
        for(iC=0;iC<2;iC++)
         {iCu=iConn[n][iC];
          for(i=0;i<4;i++)
           {FrustP[i]=0.e0;
            for(j=0;j<4;j++){FrustP[i]=FrustP[i]+Vertex[iCu][j]*Hinv[i][j];
                            }/*j*/
           }/*i*/
          glVertex4dv(FrustP);
         }/*iC*/
        if(iSide==-1) Colors3D(3);
        if(iSide== 1) Colors3D(8);
       }/*n*/
     glEnd();
     glLineWidth(1.0f);                                                glFlush();
     glTranslated(-S.D, S.e[iSide+1],0.e0 );
     if(S.VuMode>0) glutWireSphere( .5e0, 20, 20 ); //Show the eye location(s).
     glTranslated( S.D,-S.e[iSide+1],0.e0 );                           glFlush();

    //Show N,F,L,R,T,B,-D,+E, & -E:        added 2016.07.06
                  TBmult=1.e0;
    if(S.VuMode>2) TBmult=2.e0;
    h4Fill(Xyzh,  S.N   , 0.0e0 ,  0.e0       , 1.e0 );
    h4Fill(Rpyh,  0.0e0, 0.0e0 , 90.e0        , 1.e0 );
    VecText7D(Xyzh,Rpyh, 1.0e0 ,  3.   , 1    ,"Near\0" );
    h4Fill(Xyzh,  S.F   , 0.0e0 ,  0.e0       , 1.e0 );
    h4Fill(Rpyh,  0.0e0, 0.0e0 , 90.e0        , 1.e0 );
    VecText7D(Xyzh,Rpyh, 1.0e0 ,  3.   , 1    ,"Far\0" );
    h4Fill(Xyzh,  0.0e0, S.L    ,  0.e0       , 1.e0 );
    h4Fill(Rpyh,  0.0e0, 0.0e0 ,  0.e0        , 1.e0 );
    VecText7D(Xyzh,Rpyh, 1.0e0 ,  3.   , 1    ,"Left\0" );
    h4Fill(Xyzh,  0.0e0, S.R    ,  0.e0       , 1.e0 );
    h4Fill(Rpyh,  0.0e0, 0.0e0 ,  0.e0        , 1.e0 );
    VecText7D(Xyzh,Rpyh, 1.0e0 ,  3.   , 1    ,"Right\0" );

    h4Fill(Xyzh,  0.0e0, 0.0e0 ,  S.T*TBmult , 1.e0 );
    h4Fill(Rpyh,-90.0e0, 0.0e0 , 90.e0        , 1.e0 );
  if(S.VuMode<3)
    VecText7D(Xyzh,Rpyh, 1.0e0 ,  3.   , 1    ,"Top\0" );
  else
    VecText7D(Xyzh,Rpyh, 1.0e0 ,  3.   , 1 ,"Top*2.\0" );
    h4Fill(Xyzh,  0.0e0, 0.0e0 ,  S.B*TBmult , 1.e0 );
    h4Fill(Rpyh,-90.0e0, 0.0e0 , 90.e0        , 1.e0 );
  if(S.VuMode<3)
    VecText7D(Xyzh,Rpyh, 1.0e0 ,  3.   , 1 ,"Bottom\0" );
  else
    VecText7D(Xyzh,Rpyh, 1.0e0 ,  3.   , 1 ,"Bottom*2.\0");

     if(S.VuMode<1) return;

    h4Fill(Xyzh, -S.D   , 0.0e0 ,  0.e0       , 1.e0 );
```

```
    h4Fill(Rpyh,  0.0e0, 0.0e0 , 90.e0       , 1.e0 );
    VecText7D(Xyzh,Rpyh, 1.0e0 ,  3.  , 1 ,"    -Dist\0");
    h4Fill(Xyzh, -S.D  , S.E   ,  0.e0       , 1.e0 );
    h4Fill(Rpyh,  0.0e0, 0.0e0 ,  0.e0       , 1.e0 );
    if(S.VuMode>1)
    VecText7D(Xyzh,Rpyh, 1.0e0 ,  3.  , 1 ,"   +Eye\0");
    h4Fill(Xyzh, -S.D  ,-S.E   ,  0.e0       , 1.e0 );
    h4Fill(Rpyh,  0.0e0, 0.0e0 ,  0.e0       , 1.e0 );
    if(S.VuMode>1)
    VecText7D(Xyzh,Rpyh, 1.0e0 ,  3.  , 1 ,"   -Eye\0");

    //At-plane clipping plane normal vectors:
     glBegin(GL_LINES);
       glLineWidth(4.0f);
       if(iSide==-1 )
        { Colors3D(3);
          //Left eye- far field left:
          ClipK=-S.ClipPlane1[3]/( S.ClipPlane1[0]*S.ClipPlane1[0]
                                  +S.ClipPlane1[1]*S.ClipPlane1[1]
                                  +S.ClipPlane1[2]*S.ClipPlane1[2]);
          h4Fill(Clip ,S.ClipPlane1[0]*ClipK
                      ,S.ClipPlane1[1]*ClipK
                      ,S.ClipPlane1[2]*ClipK
                      ,  1.e0                );   glVertex4dv(Clip);
          h4Fill(Clip ,0.e0,0.e0,0.e0,1.e0);      glVertex4dv(Clip);
          //Left eye- near field right:
          ClipK=-S.ClipPlane1[3]/( S.ClipPlane2[0]*S.ClipPlane2[0]
                                  +S.ClipPlane2[1]*S.ClipPlane2[1]
                                  +S.ClipPlane2[2]*S.ClipPlane2[2]);
          h4Fill(Clip ,S.ClipPlane2[0]*ClipK
                      ,S.ClipPlane2[1]*ClipK
                      ,S.ClipPlane2[2]*ClipK
                      ,  1.e0                               );
          glVertex4dv(Clip);
          h4Fill(Clip ,0.e0,0.e0,0.e0,1.e0);
          glVertex4dv(Clip);
       }/*(iSide==-1)*/
       if(iSide== 1 && S.VuMode>1 )
        { Colors3D(8);
          //Right eye- near field left:
          ClipK=-S.ClipPlane3[3]/( S.ClipPlane3[0]*S.ClipPlane3[0]
                                  +S.ClipPlane3[1]*S.ClipPlane3[1]
                                  +S.ClipPlane3[2]*S.ClipPlane3[2]);
          h4Fill(Clip ,S.ClipPlane3[0]*ClipK
                      ,S.ClipPlane3[1]*ClipK
                      ,S.ClipPlane3[2]*ClipK
                      ,  1.e0                );   glVertex4dv(Clip);
          h4Fill(Clip ,0.e0,0.e0,0.e0,1.e0);      glVertex4dv(Clip);
          //Right eye- far field right:
          ClipK=-S.ClipPlane4[3]/( S.ClipPlane4[0]*S.ClipPlane4[0]
                                  +S.ClipPlane4[1]*S.ClipPlane4[1]
                                  +S.ClipPlane4[2]*S.ClipPlane4[2]);
          h4Fill(Clip ,S.ClipPlane4[0]*ClipK
                      ,S.ClipPlane4[1]*ClipK
                      ,S.ClipPlane4[2]*ClipK
                      ,  1.e0                               );
          glVertex4dv(Clip);
          h4Fill(Clip ,0.e0,0.e0,0.e0,1.e0);
          glVertex4dv(Clip);
       }/*(iSide== 1)*/
     glLineWidth(1.0f);
    glEnd();

    if(!S.iTeapot|S.iTeapot>4) CubeGrid(11);
                               Teapot(  );
    if(S.VuMode<2) return;
   }/*iSide*/
 }//End SeeFrustums ------------------------------------------------------------

 //void VecText7D( double xyz[4],double rpy[4],double SizeH,char Label[5000])
```

```
 void VecText7D( double Pxyzh[4],double ArpyDh[4] //v:0.5-----------------------
                ,double SizeH   ,float LineWidth ,int iCol   ,char Label[5000])/*
 /* 7D vector-based characters; the individual character shapes are specified.
          This supports both: font modification and extension  -and-
                        exporting characters as 3D lines.
   2016.06.24.1600cdt JMS- Works - but - color of first character is
                                        "kluged-to-correctness".

   Unrotated characters are written parallel-to the +X axis,
                  readable from the +Y direction
   in righthanded "Flight Simulation" (abbrev. 'Fs') coordinates.

   Function 'HindSight'- with S.ThreePhase=3: uses Fs-based screen coordinates
                               "2D/3D"      for the various projection matrices
                                            +X:forward,+Y:right,+Z:down.
                     - with S.ThreePhase=2: uses glOrtho (which is non-FS).
                               "2D ortho"   +X:right,+Y:up, +Z:forward,
                                            a left-handed coordinate frame.

   Use 'PrntOrtho' to write stationary color 2D text on the screen.

 Function 'Colors3D' defines the colors

 VecText.h has the following c* & d* characters encoded,
    ... however, they presently have no useful effect.
 The objective is to enrich the formatting within a text file
    by repurposeing some relatively unused characters.
 The key question is: How to easily insert hex code characters in text?

 Custom character usages:
  My colors:        [c][0] 'move without draw' - a no-op in this context
                             (clarifies & simplifies the content of .3dv files.)
            White   [c][1]
            Red     [c][2]
    !Green= Magenta [c][3]
    !Blue = Yellow  [c][4]
      light Green   [c][5]
            Green   [c][6]
    !Red  = Cyan    [c][7]
      light Blue    [c][8]
            Blue    [c][9]
      light Gray    [c][a]
            Gray    [c][b]
            Brown   [c][c]
            Purple  [c][d]
    !White  Black   [c][e]
        user-defined[c][f]

   Shape modifiers  [d][0]    toggles-> off
      Line width    [ ][1] =1.
                    [ ][2] =2.
                    [ ][3] =3.
                    [ ][4] =4.
                    [ ][5] =5.
      Italics       [ ][6]    toggle on/off
      Underlined    [ ][7]      "
      Subscript     [ ][8]      "
      Superscript   [ ][9]      "
                    [ ][a]
                    [ ][b]
                    [ ][c]
      Show controls [ ][d]      "
                    [ ][e]
                    [ ][f]                                              */
 {                                                                     //Dec
  int i,j,j1,j2,k,m,n,nCh,nChar,nPts;
  double hXR[4][4],Pin[4],Pout[4];
  union iAndc {char Char1;short Num1;} Try;
  float LineWidthPrev[1],colors[4];
```

```
  //char PText[80];

 //if(ModeFlag==0) i=0;

  nChar=strlen(Label); if(nChar<=0) return;
  glGetFloatv( GL_CURRENT_COLOR,colors);     //Save the current color.
  Colors3D(iCol);                            //Use the character's color.
  glGetFloatv( GL_LINE_WIDTH,LineWidthPrev); //Save the line width.

  PAhXR(Pxyzh,ArpyDh,hXR);    //<- X,Y,Z,Roll,Pitch, & Yaw : 6 dof's.

  m= 0;  //Zero the line counter. 0 is the index of the first line.
  n=-1;  //Set the character offset to 'before the first character' of the line.

  //Process the incoming label... character by character... starting at [0]:
  for(nCh=-1;nCh<nChar;nCh++)
   {
        // For reasons unknown to me... text color is initially incorrect.
        // a workaround:
          if(nCh<=0) n=-1;                  //
          if(nCh<0 )Try.Char1 = 46;         //...this adds a 'point artifact'
            else    Try.Char1 = Label[nCh]; //   at the origin of model space
        //                                  //   as a preceeding character.
        // Recognize & utilise some of the traditonal control characters
        //   to easily write paragraphs in 7dof space:
        if( nCh>=0 && Try.Char1<32 )
           switch(Try.Num1)
            { case( 8): n= n-1     ;continue; //BackSpace                  '\b'
              case( 9): n=(n+1)/8  ;          //Horizontal Tab- 8 wide    '\t'
                        n=(n+1)*8-1;continue;
              case(11): m=(m  )/5  ;          // Vertical  Tab- 5 high    '\v'
                        m=(m+1)*5  ;continue;
              case(10): m= m+1     ;          //Line Feed                 '\n'
              // Here: line feed also adds a carriage return...
              case(13): n=  -1     ;continue; //Carriage Return           '\r'

              case(12): m= m+80    ;continue; //Form Feed                 '\f'
            }/*( nCh>=0 && Try.Char1<32 )*/

                 n=n+1;    //  move right by one character.
                if(Try.Num1< 32) Try.Num1= 32;
                if(Try.Num1>255) Try.Num1=255;
    j1   = CharLoc[Try.Num1 ];
    if(  j1==0) continue;//if the character isn't defined- go to the next one.
    j1   = j1-1;         //This is the beginning of the character's information
    nPts = PointLoc[j1+2];
    if(nPts==0) continue;//if the charcter has no points- go to the next one.
    j2   = j1+(1+nPts)*3;//This is the beginning of the following character
                                    k=  0;
      for(j=j1+3;j<j2;j=j+3) //Step through the character's points,
       { switch(PointLoc[j])                          // in groups of 3:
          {case(1):                 if(k>0) glEnd(); //ends a begun line.
            glLineWidth(LineWidth);   k=k+1;
            glBegin(GL_LINE_STRIP); //Move without drawiing
           case(2):                  //Draw (or move):
            // Resize first: ...the 7th dof:
            h4Fill(Pin, ( (PointLoc[j+1]+15)/30.e0 +n      )*SizeH
                      ,   0.e0
                      , (-(PointLoc[j+2]+10)/20.e0 +m*1.5e0)*SizeH
                      ,   1.e0);
            // Then translate & rotate... the other 6 dof's:
            for(i=0;i<4;i++)
             {Pout[i]= hXR[i][0]*Pin[0]
                      +hXR[i][1]*Pin[1]
                      +hXR[i][2]*Pin[2]
                      +hXR[i][3]*Pin[3]; }

             if(nCh<0) h4Fill(Pout,0.e0,0.e0,0.e0,1.e0); //a 'point artifact'

             //Move-to or draw-to the point:
```

```
             glVertex3f(Pout[0],Pout[1],Pout[2]);                          break;

           casedefault: goto Oops;  //Oops- misaligned vectext.h data:
         }/*(PointLoc[j])*/
       }/*j*/
Oops:          //Cease to try to draw more characters.

     //Normal termination:
     glEnd();  //end the last line

    //Restore previous line width and line color.
    glLineWidth(LineWidthPrev[1]);
  //glColor4fv(colors);                                                 glFlush;
   }/*n*/
}//End VecText7D ------------------------------------------------------------

void AppF8(void) /*v:0.5---------------------------------------------------
  2016.07.10 JMS- This is the opening application of 3DEnv.exe
          You've entered this application reached by pressing 'F8':          */
{                 //    "F_: Title__________________ yyyy.mm.dd\0";      <-e.g.
        char AppName[]="F8: HindSight Demo        2016.07.10\0";
  //                    E   ,   N    ,     F   ,  L   ,  R   , T   , B   ,  D
    double FrustCoes[8]={ 1.2, -4.0  ,     6.0,-10.4, 10.4,-6.5, 6.5, 24.0 };
  //double FrustCoes[8]={ 1.2,-11.994, 24000.0,-10.4, 10.4,-6.5, 6.5, 24.0 };
  //                                    far depth cutoff is ~5 miles away.
  //double FrustCoes[8]={ 1.2, -8.0  ,    24.0,-10.4, 10.4,-6.5, 6.5, 24.0 };
  //                                         for more near-field depth res.
  int i;
  char Buffer[400];
  char Label[80];
  double Xyzh[4],Rpyh[4];

//float fogColor[4]={.5,.5,.5,1.0};
//int fogmode=GL_EXP2;


 switch(S.ThreePhase) // <- driven by the 3D HindSight ------
  {       /*---------------------------------*/
   case(1):// Phase One- Transfers, initializations, & non-screen computations:
    if(S.AppInit[S.AppNumber]<1)
     { S.iTeapot              = 7;
       S.AppInit[S.AppNumber] = 1;
       S.Scale                 = 2.e0; //S.Scale alters depth bounds
       S.VuMode               = 1;
       S.MousePitch           = 0.e0;
       S.MouseYaw             = 0.e0;
     }/*S.AppInit*/
       S.FovYZzoom            = 2.e0; //S.FovYZzoom doesn't alter depth bounds

    sprintf(S.AppName,"%s\0",AppName);
    // HindSight: Load your frustum coefficients:
    for(i=0;i<8;i++) { S.FrustCoes[i]=FrustCoes[i]; }

          /*---------------------------------*/                          break;
   case(2):// Phase Two- 2D orthographic screen graphics:
    //Author & IP information:
    h4Fill(Xyzh, -.40e0*(S.xyWindowRatio),  .95e0, 0.e0, 1.e0);
    h4Fill(Rpyh,90.00e0                   , 0.00e0, 0.e0, 1.e0);
    sprintf(Label,
"      \xe0\x08 2016 Jeffrey M. Setterholm\0");
    Xyzh[1]=.95e0; VecText7D(Xyzh,Rpyh, .03e0, 1.,  1  ,Label);
    sprintf(Label,
"          8095 230th St. E., Lakeville, MN 55044\0");
    Xyzh[1]=.92e0; VecText7D(Xyzh,Rpyh, .02e0, 1.,  1  ,Label);
    sprintf(Label,
"                jeff.setterholm@gmail.com  \0");
    Xyzh[1]=.89e0; VecText7D(Xyzh,Rpyh, .02e0, 1.,  6  ,Label);
    sprintf(Label,
"   This Visualization Environment is freely distributable.\0");
    Xyzh[1]=.86e0; VecText7D(Xyzh,Rpyh, .02e0, 1.,  8  ,Label);
```

```
    sprintf(Label,
"You are welcome to utilize your own Apps in the environment.\0");
    Xyzh[1]=.83e0; VecText7D(Xyzh,Rpyh, .02e0, 1.,  8  ,Label);
    sprintf(Label,
"  I place my 2D/3D vision transforms in the public domain.\0");
    Xyzh[1]=.80e0; VecText7D(Xyzh,Rpyh, .02e0, 1.,  1  ,Label);


    if(S.IterTotal<600)
     { sprintf(Buffer,
"     3DEnv.exe - June 24,2016+\n\n"

"   My goal is to provide insights\n"
"       into the mathematics of\n"
"3D-stereo homogeneous vector graphics\n"
"   to help synergize & standardize\n"
"      the visual component of\n"
"tomorrow's human/computer interfaces.\n\n"
);

       h4Fill(Xyzh,   .6e0,  -.6e0 , -.8e0, 1.e0);
       h4Fill(Rpyh,  90.e0,  0.e0  ,  0.e0, 1.e0);
     //VecText7D(    ,     ,  SizeH, LineWidth, iCol,Label[~38] );
       VecText7D(Xyzh,Rpyh, .025e0,    1.    ,   5 ,  Buffer);          break;
     }/*S.IterTotal<600*/

    if(S.NowView==0)
     {
      PrntOrtho( 4,2, 1,0,"PRESS 'm'/'M' for the menu of control options.\0");



      PrntOrtho( 7,2, 6,0,"Press 'v' to view HindSight's numerical values.\0");
      PrntOrtho( 8,2, 6,0,"Press 'V' and Scale down to see the frustums.\0");
      PrntOrtho( 9,2,11,0,"Interactively HindSight 'Scales',not 'Zooms'\0");
      PrntOrtho(10,2,11,0,
                  " ...expands/contracts about your 'Point of Interest'.\0");

      PrntOrtho(12,2, 6,0,"Press '`' to view the 'depth selfie'.\0");
      PrntOrtho(13,2, 6,0,"Press '~' to save the screen as 'selfie0.bmp'.\0");

      PrntOrtho(15,2,11,0,"No teapot in sight? Press 'Home' &/or 't'.\0");
     }
    if(S.BrkOn>0)                                                       break;
    if(S.NowView==2 )
      PrntOrtho( 6,2, 6,0,"Scale down to see the frustum(s).\0");
    if(S.NowView==2 && S.Scale>.75e0)
      PrntOrtho( 7,2, 1,0,"  ... holding down 's' decreases model scale\0");
    if(S.NowView==2 && S.Scale<.75e0 && S.VuMode>0 && S.Scale>.1e0)
      PrntOrtho( 7,2,11,0
            ,"  ... Scale down far enough to see the eye location sphere.\0");
          /*---------------------------------*/                         break;
   case(3):Phase Three- 2D/3D Ee-key-controlled screen graphics:

    if(S.NowView==2) SeeFrustums();
    else
     {if(!S.iTeapot|S.iTeapot>4) CubeGrid(11);
                                 Teapot(  );
     }
          /*---------------------------------*/                         break;
  }//S.ThreePhase-----------------------------------------

}//End AppF8 ---------------------------------------------------------------


void AccessYourActiveApp(void) /*v:0.5-----------------------------------------
  The main Switchboard for calling applications:
  2016.06.24.1600cdt JMS                                               */
{  char PText[80];
   double Xyzh[4],Rpyh[4];
```

```
  switch(S.AppNumber)
  {  case(1): AppF1(); break;  //F1 "F1: Ortho Projection"
     case(2): AppF2(); break;  //F2 "F2:Mandelbrot Set"
   //case(3): AppF3(); break;  //F3
     case(4): AppF4(); break;  //F4 "F4 VecText7D Demo"
     case(5): AppF5(); break;  //F5 "F5: 3dv-Viewer"
     case(6): AppF6(); break;  //F6 "F6: Demo-Invert44"
   //case(7): AppF7(); break;  //F7 "F7: OverWriter & Demo"
     case(8): AppF8(); break;  //F8 "F8 HindSight & Demo" -the opening application
     default:
      h4Fill(Xyzh,-1.e0, 0.e0, 0.e0, 1.e0);
      h4Fill(Rpyh,90.e0, 0.e0, 0.e0, 1.e0);
      sprintf( PText,"...Application F%1i isn't presently accessible.\0"
                              ,S.AppNumber);
      if(S.ThreePhase==2)
      VecText7D(Xyzh,Rpyh, .04e0, 2.,  1, PText );


  }//(S.AppNumber)
 }//End AccessYourActiveApp-------------------------------------------------------
```