```
/*App-F2-MandelbrotSet.c
  C Opengl(GLUT) Example User Application F2:
  2016.07.10 JMS
  2013.01.17 JMS- Traveler2/Athlon64/WinXPPro/APF9.0:C/OpenGL+CGlut

 Herein: void AppF1(void)
*/
#include <3DEnv.h>   //<- Environment's variables & functions become available.

void AppF2(void) /*-----------------------------------------2016.07.10 JMS
          You've entered this application reached by pressing 'F2':          */
{                  //    "F_: Title_____________ yyyy.mm.dd\0";      <-e.g.
         char AppName[]="F2: Mandelbrot Set     2016.07.10\0";

            //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 FrustCoes[8]={ 1.2,-11.994, 24000.,-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.
            //A suggestion: make L,R,T,& B the +- dimensions of your screen.

  int i,j,k;
  double Xyzh[4],Rpyh[4];
  double static DtoR= asin(1.0e0)/90.e0;
  char Label[80];

  double   DistSq;
  int static Niter=0;
  struct
   {double C[3];
    double Zn[3];
    double Znp1[3];
    int    n;
    int    nDone;
    int    color;
   } static Z[400][200],ZL;
  //Track a mouse-controlled starting point.
  double static MyTrack[1001][2],MyC[2];
  int    static MyDone;
  double zHeight;
  double static zUse=-1.e0; //Initially heights are displayed in the -Z.

 switch(S.ThreePhase) // <- driven by the 3D viewer -------
  {        /*-----------------------------------*/
   case(1):
    if(S.AppInit[S.AppNumber]<1)
     { S.iTeapot               = 0;
       S.AppInit[S.AppNumber] = 1;
       S.Scale                = 5.e0;
     }/*S.AppInit*/
       S.FovYZzoom            = 1.e0;

    sprintf(S.AppName,"%s\0",AppName);
                                                          //VuMode
    for(i=0;i<8;i++) { S.FrustCoes[i]=FrustCoes[i]; } //Frustum coefficients

    if(S.KbdKey==50) //Using keyboard key '2' to toggle Z axis content:
     { S.KbdKey=0;   //Zero the zey input so the effect happens only once.
       zUse=-1.e0-zUse; }

    if(Niter>=1000)  Niter=0; //Progress to 1000 interations & start over.
    Niter=Niter+1;
    //Advance the trajectories of 80,000 points by one iteration:
    for(i=   0;i<400;i++)
     { for(j=0;j<200;j++)
        { //On the first iteration: initialize the point:
          if(Niter==1){ Z[i][j].C[0]   = double(i-200)/100.e0; //real
```

```
                        Z[i][j].C[1]   = double(j    )/100.e0; //imaginary
                //The Mandelbrot equation uses 'complex variables' ^
                        Z[i][j].C[2]   = 0.e0;
                        Z[i][j].Znp1[0]= 0.e0;
                        Z[i][j].Znp1[1]= 0.e0;
                        Z[i][j].Znp1[2]= 0.e0;
                        Z[i][j].n      = 0;
                        Z[i][j].color  = 0;
                        Z[i][j].nDone  = 0;
                                DistSq = Z[i][j].C[0]*Z[i][j].C[0]
                                        +Z[i][j].C[1]*Z[i][j].C[1];
                        if(DistSq>4.e0)
                        Z[i][j].nDone  =-1; //Ignore points beyond R=2.0
                      }//Niter==1
          //The rest of the iterations proceed as follows:
          ZL        = Z[i][j]; //Make a local copy with no indices
                               //  so the equations are easier to read.
          if(ZL.nDone==0)
           { //Make the output of the previous iteration
             //  the beginning of this iteration:
             ZL.Zn[0]  = ZL.Znp1[0];
             ZL.Zn[1]  = ZL.Znp1[1];
             ZL.Zn[2]  = ZL.Znp1[2];
             ZL.n      =    Niter;

             //Height above the complex plane (greatly compressed):
             ZL.C[2]   =log10(log10(double(Niter))+1.e0);

             //Implement Z(n+1)=Z(n)*Z(n)+C ...the complex Mandelbrot equation:
             ZL.Znp1[0]= ZL.Zn[0]*ZL.Zn[0] //real part
                        -ZL.Zn[1]*ZL.Zn[1]
                        +ZL.C[ 0];
             ZL.Znp1[1]= ZL.Zn[0]*ZL.Zn[1] //imaginary part
                        +ZL.Zn[1]*ZL.Zn[0]
                        +ZL.C[ 1];
             //Check for divergence:
             DistSq    = ZL.Znp1[0]*ZL.Znp1[0]
                        +ZL.Znp1[1]*ZL.Znp1[1];
             if(DistSq>4.e0) ZL.nDone=Niter;
             //Save the iteration update back in the Z[i][j] array:
             Z[i][j]   = ZL;
           }//ZL.nDone==0
        }//j
      }//i

     //For 1000 iterations: compute the trajectory of one mouse-controlled 'C':
     MyC[0]       = 2.e0*S.MouseScreenh[0];  //real      ...mouse controlled.
     MyC[1]       = 2.e0*S.MouseScreenh[1];  //imaginary ...mouse controlled.
     MyDone       = 0;                       //Iterations to divergence
     MyTrack[0][0]= 0.e0;
     MyTrack[0][1]= 0.e0;
     for(i=   1;i<1001;i++)
      { if(MyDone>0) continue; //if diverged... stop iterating.
        //Z[i+1]=Z[i]*Z[i]+C  complex square & add:
        MyTrack[i][0]=  MyTrack[i-1][0] * MyTrack[i-1][0]  //real
                      - MyTrack[i-1][1] * MyTrack[i-1][1]
                      + MyC[ 0];
        MyTrack[i][1]=  MyTrack[i-1][0] * MyTrack[i-1][1]  //imaginary
                      + MyTrack[i-1][1] * MyTrack[i-1][0]
                      + MyC[ 1];
        //Test for divergence:
        DistSq       =  MyTrack[i  ][0] * MyTrack[i  ][0]
                      + MyTrack[i  ][1] * MyTrack[i  ][1];
        if(DistSq>4.e0) MyDone=i; //Record the divergence iteration#.
      }/*i*/
     if(MyDone==0) MyDone=1000;

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

    case(2)://Phase Two- 2D orthographic screen graphics:
```

```
     //Author & IP information:
     h4Fill(Xyzh, -.25e0*(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., 11  ,Label);
     sprintf(Label,"        jeff.setterholm@gmail.com  \0");
     Xyzh[1]=.89e0; VecText7D(Xyzh,Rpyh, .02e0, 1.,  6  ,Label);
     sprintf(Label,"This 'App-F2' is freely distributable.\0");
     Xyzh[1]=.86e0; VecText7D(Xyzh,Rpyh, .02e0, 1.,  8  ,Label);

     sprintf(Label,"Iteration:%4i\0",Niter);
     h4Fill(Xyzh ,(-S.xyWindowRatio),  .85e0, 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, 2.       ,  1  ,Label  );

     sprintf(Label,"   MyDone:%4i\0",MyDone);
     h4Fill(Xyzh ,(-S.xyWindowRatio),  .80e0, 0.e0, 1.e0);
     Alpha6D(Xyzh,Rpyh, 0      , .03e0, 2.       ,  5  ,Label  );

     sprintf(Label," key '2' toggles depth.\0",MyDone);
     h4Fill(Xyzh ,(-S.xyWindowRatio),  .75e0, 0.e0, 1.e0);
     Alpha6D(Xyzh,Rpyh, 0      , .02e0, 2.       , 11  ,Label  );

     sprintf(Label," Move the mouse & use the LMB.\0",MyDone);
     h4Fill(Xyzh ,(-S.xyWindowRatio),  .70e0, 0.e0, 1.e0);
     Alpha6D(Xyzh,Rpyh, 0      , .02e0, 2.       , 12  ,Label  );

     sprintf(Label," Press v for trajectory values.\0",MyDone);
     h4Fill(Xyzh ,(-S.xyWindowRatio),  .65e0, 0.e0, 1.e0);
     if(S.NowView==0)
       Alpha6D(Xyzh,Rpyh, 0      , .02e0, 1.       , 5  ,Label  );
     else
      { PrntOrtho(15,2, 1,0,"Trajectory (complex values):\0");
        PrntOrtho(16,2, 1,0,"Iter     Real     Imag:\0");
        for(i=0;i<50;i++)
         {if(i>MyDone) break;
          sprintf(Label,"%4i %9.5f %9.5f\0"
              ,i ,float(MyTrack[i][0]),float(MyTrack[i][1]) );
          if(i<2)PrntOrtho(17+i,2, 1,0,Label);
          else   PrntOrtho(17+i,2, 5,0,Label);
         }/*i*/
      }
           /*---------------------------------*/                        break;

    case(3)://Phase Three- 2D/3D Ee-key-controlled screen graphics:
     Teapot();
     h4Fill(Xyzh ,  0.e0, 0.e0, 0.e0, 1.e0);
     h4Fill(Rpyh ,-90.e0, 0.e0, 0.e0, 1.e0);
     VecText7D(Xyzh,Rpyh, .05e0, 2.,  1  ,"  Imag + >\0" );
     h4Fill(Rpyh ,-90.e0, 0.e0,90.e0, 1.e0);
     VecText7D(Xyzh,Rpyh, .05e0, 2.,  1  ,"  Real + >\0" );
     h4Fill(Rpyh ,  0.e0,90.e0,90.e0, 1.e0);
     if(zUse<0.e0)
     VecText7D(Xyzh,Rpyh, .05e0, 2.,  1  ,"  Iter + >\0" );
     h4Fill(Xyzh ,MyTrack[1][1],MyTrack[1][0], 0.e0, 1.e0);

     h4Fill(Rpyh ,-90.e0, 0.e0,90.e0, 1.e0);
     VecText7D(Xyzh,Rpyh, .05e0, 2.,  1  ,"C\0" );


     // Outside R=2. the solution has diverged.
     Colors3D( 1);
     glLineWidth(2.0f);
     glShadeModel(GL_FLAT);
     glBegin(GL_LINE_LOOP);
     for(i=  0;i<360;i=i+5)
      {glVertex3d(  2.e0*cos(double(i)*DtoR )    //imaginary -> +X
```

```
                , 2.e0*sin(double(i)*DtoR )    //real      -> +Y
                , 0.e0                       ); //Iterations-> -Z
     }//i
    glEnd(); glFlush();

    // Outside this ellipse the solution diverges in one more iteration.
      zHeight=log10(log10(double(2))+1.e0)*zUse;
    glBegin(GL_LINE_LOOP);
    for(i=  0;i<360;i=i+5)
     {glVertex3d(  1.333e0*cos(double(i)*DtoR)
                , 1.500e0*sin(double(i)*DtoR)-0.5e0
                , zHeight                              );
     }//i
    glEnd(); glFlush();

    // Draw the mouse-selected trajectory:
    Colors3D( 1); //Make the first line white
    glBegin(GL_LINE_STRIP);
        //Log10(0) blows up:
        zHeight=0.e0;                                  //The starting point
        glVertex3d( MyTrack[0][1], MyTrack[0][0], zHeight );
        //log10(i>0) is defined:
        zHeight=log10(log10(double(1))+1.e0)*zUse;
        glVertex3d( MyTrack[0][1], MyTrack[0][0], zHeight );
        glVertex3d( MyTrack[1][1], MyTrack[1][0], zHeight ); //C
        zHeight=log10(log10(double(1000))+1.e0)*zUse; //Vertical spike @ C:
        glVertex3d( MyTrack[1][1], MyTrack[1][0], zHeight );
     //For the rest of the trajectory:
     for(i=  2;i<MyDone+1;i++)
      { zHeight=log10(log10(double(i))+1.e0)*zUse;
        glVertex3d( MyTrack[i-1][1], MyTrack[i-1][0], zHeight); //Move up
        if(i==2) Colors3D( 5); //Make the rest of the lines yellow.
        glVertex3d( MyTrack[i  ][1], MyTrack[i  ][0], zHeight); //Move over
      }//i
    glEnd(); glFlush();

    // Draw all the 'C' starting points that have diverged
    glPointSize(2.0f);
    glShadeModel(GL_FLAT);
    glBegin(GL_POINTS);
    for(i=  0;i<400;i++)
     {for(j=0;j<200;j++)
       {
        if(Z[i][j].nDone<1) continue; //Skips points that didn't diverge.
        ZL=Z[i][j];
        if(     ZL.nDone<  2) Colors3D(15);
        else if(ZL.nDone<  3) Colors3D(12);
        else if(ZL.nDone<  4) Colors3D(11);
        else if(ZL.nDone<  5) Colors3D(12);
        else if(ZL.nDone< 10) Colors3D( 9);
        else if(ZL.nDone< 30) Colors3D( 7);
        else if(ZL.nDone< 50) Colors3D( 6);
        else if(ZL.nDone<100) Colors3D( 3);
        else if(ZL.nDone<200) Colors3D( 2);
        else                  Colors3D( 1);
        //Set pixel color: ^ number of iterations to reach divergence.

         glVertex3d( ZL.C[1], ZL.C[0], ZL.C[2]*zUse );
         glVertex3d(-ZL.C[1], ZL.C[0], ZL.C[2]*zUse );//Mirrors imaginary data.
       }//j
     }//i
    glEnd();                                                        glFlush();


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

}//End AppF2 ----------------------------------------------------------------
```

```
//-------------------------------------------------------------------------7 9
```