```
//
//  CS75301 System Modeling and Performance Eval
//
//  File:     3d.cpp
//  Author:   Kenneth W Schmidt
//  Abstract: This program reads an input file containing three fields
//            per tuple, the x, y, and z components of points in space.
//            The format is float, space, float, space, float, endline.
//            The data is read into an array (small data size and faster)
//            Max tuples = 1000. The data in the array is normalized to
//            between 0 and 1 for consistant display. The initial display
//            is 2-D and is rotatible around all three axes with the arrow
//            keys and mouse. The display can be switched to 3-D which
//            requires red-blue 3D glasses.
//
//  Revision History:   Date           Who      Description
//                -----------------------------------------------
//                       12/7/03        KWS      Initial Release


#include <fstream.h>  //to write to a file
#include <math.h>
#include <GL/glut.h>  //open GL
#include <stdio.h>
#include <windows.h>  //for message box

ifstream readFile;    // to read from a file

struct data
  {
    double x; //point location in x plane
    double y; //point location in y plane
    double z; //point location in z plane
  };

char readFileName [ 36 ];   // name of the file to be read from,
                            //31 char plus . plus 3 char ext plus "\0" (end of line)
data dataArray [ 1000 ];    //array for data read from input file
int n = 0;                  //array length used counter

void getUserInput ( void );
void display2 ( void );
void display3 ( void );
void help ( void );
void keyboard ( unsigned char, int, int );
void arrowKey ( int, int, int );
void mouse ( int, int, int, int );
void init2 ();
void init3 ();

////////////////////////////////////////////////////////////////////
//  main
//--------------------------------------------------------------------
//  Purpose:  main section of code
////////////////////////////////////////////////////////////////////

int main(int argc, char** argv)
{
  getUserInput ();
  glutInit(&argc, argv);
  init2 ();
  help ();            //display the help box
  glutMainLoop();     //waiting for an event - click of mouse, etc.

  return 0;
}

////////////////////////////////////////////////////////////////////
//  init2
//--------------------------------------------------------------------
//  Purpose:  creates a single buffer drawing window for the input
//            points for the 2-D display
////////////////////////////////////////////////////////////////////

void init2 ()
{
```

```
  glutInitDisplayMode ( GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH );
                                    //GLUT_DOUBLE is 2 buffer, prevents jittery display
                                    //GLUT_RGB means we are using colors
  glutInitWindowPosition(25,25);    //position of upper left corner of window
  glutInitWindowSize(625,625);      //size in pixels
  glutCreateWindow ( readFileName );//create window with readFileName title of window

  glClearColor(0.0, 0.0, 0.0, 0.0); //open a blank GL command window,
                                    //choose the background color
                                    //R, G, Blu, Transparancy alpha coefficient

  glMatrixMode(GL_PROJECTION);  //project the image to the front plane
  glLoadIdentity();             //loads a 4x4 identity matrix for normal initial projection
  glOrtho(-1.5, 1.5, -1.5, 1.5, -1.5, 1.5); //visible world, projection mode
        // x     x    y     y     z    z, lower and upper limits of x,y,z
        //glOrtho says take the image from the 0 in the Z plane and project it
        //to the front plane (which would be the max pos limit of Z

  glutDisplayFunc ( display2 );//draw the initial image, display is another function
  glutMouseFunc ( mouse );//activate the mouse function
  glutKeyboardFunc ( keyboard );//activate the keyboard function
  glutSpecialFunc ( arrowKey );//activate the keyboard arrow keys
}


//////////////////////////////////////////////////////////////////////
//  init3
//--------------------------------------------------------------------
//  Purpose:  creates a double buffer drawing window for the input
//            points for the 3-D display
//////////////////////////////////////////////////////////////////////

void init3 ()
{
  glutInitDisplayMode ( GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH );
                                    //GLUT_DOUBLE is 2 buffers to eliminate flicker
                                    //GLUT_RGB means we are using colors
  glutInitWindowPosition(25,25);    //position of upper left corner of window
  glutInitWindowSize(625,625);      //size in pixels
  glutCreateWindow ( readFileName );//create window with readFileName title of window

  glClearColor(0.0, 0.0, 0.0, 0.0); //open a blank GL command window,
                                    //choose the background color
                                    //R, G, Blu, Transparancy alpha coefficient

  glMatrixMode(GL_PROJECTION); //project the image to the front plane
  glLoadIdentity();           //loads a 4x4 identity matrix for normal initial projection
  glOrtho(-1.5, 1.5, -1.5, 1.5, -1.5, 1.5); //visible world, projection mode
        // x     x    y     y     z    z, lower and upper limits of x,y,z
        //glOrtho says take the image from the 0 in the Z plane and project it
        //to the front plane (which would be the max pos limit of Z

  glutDisplayFunc ( display3 ); //draw the initial image, display is another function
  glutMouseFunc ( mouse );      //activate the mouse function
  glutKeyboardFunc ( keyboard );//activate the keyboard function
  glutSpecialFunc ( arrowKey ); //activate the keyboard arrow keys
}

//////////////////////////////////////////////////////////////////////
//  getUserInput
//--------------------------------------------------------------------
//  Purpose:  get input file name from user, read file, up to 1000
//            tuples, store in an array, normalize array to 0-1 range
//  Input:    nothing
//  Output:   tuples of data in a global array variable
//////////////////////////////////////////////////////////////////////

void getUserInput ()
{
  int i = 0;      //loop counter
  double x = 0;   //temp variable used for max value
  double y = 0;   //temp variable used for max value
  double z = 0;   //temp variable used for max value
  double maxX = 0;//max value of x for normalization
  double maxY = 0;//max value of y for normalization
  double maxZ = 0;//max value of z for normalization
```

```
  cout << "enter the file name to be read, max 31 char name plus 3 char ext\n1000 tuples max (stop
s reading after 1000)" << endl;
  cin >> readFileName;
  cout << endl;

  readFile.open ( readFileName, ios::in );

  if ( !readFile )     //did input file open successfully
  {
    cout << "can't open the file to be read from" << endl;
  }

  while ( !readFile.eof() && n < 999 )  //read the input file into memory
  {
    readFile >> dataArray [ n ].x >> dataArray [ n ].y >> dataArray [ n ].z;

    n++;
  }

  readFile.close ();

  n--;                       //decrement total length for end of file

  for ( i = 0; i < n; i++ ) //find max values for x, y, z in array
  {
    x = fabs ( dataArray [ i ].x );//floating point absolute value
    y = fabs ( dataArray [ i ].y );//floating point absolute value
    z = fabs ( dataArray [ i ].z );//floating point absolute value

    if ( x > maxX )
      maxX = x;       //store max value of x

    if ( y > maxY )
      maxY = y;       //store max value of y

    if ( z > maxZ )
      maxZ = z;       //store max value of z

  }

  for ( i = 0; i < n; i++ )//normalize x, y, z to 0 to +- 1 range for fitting display on screen
  {
    dataArray [ i ].x = dataArray [ i ].x / maxX;
    dataArray [ i ].y = dataArray [ i ].y / maxY;
    dataArray [ i ].z = dataArray [ i ].z / maxZ;
  }
}

////////////////////////////////////////////////////////////////////
//  display2
//--------------------------------------------------------------------
//  Purpose:  sets the color to white, point size to 2 pixels, plots
//            the points from the array on the grid, plots a three
//            axis grid
//  Input:    nothing
//  Output:   the points have been plotted on the screen
////////////////////////////////////////////////////////////////////

void display2 ( )
{
  int i = 0;                    //loop counter
  glClear( GL_COLOR_BUFFER_BIT);//buffer for background, colors the background
                                //using color black
  glColor3f(1.0, 1.0, 1.0);     //R, G, Blu, 1,1,1 is white, color of object
  glPointSize( 2.0 );           //size of points in pixels
  glBegin(GL_POINTS);           //draw a point for every glVertex3f until glEnd()

  for ( i = 0; i < n; i++ )     //draw points on screen from array
  {
    glVertex3f ( dataArray [ i ].x, dataArray [ i ].y, dataArray [ i ].z );
  }

  glEnd();                      //end of glBegin()

  glBegin(GL_LINES);            //draw 3 axes
```

```
    glVertex3f(-1.0, 0.0, 0.0);
    glVertex3f(1.0, 0.0, 0.0);
    glVertex3f(0.0, -1.0, 0.0);
    glVertex3f(0.0, 1.0, 0.0);
    glVertex3f(0.0, 0.0, -1.0);
    glVertex3f(0.0, 0.0, 1.0);
  glEnd();

  glutSwapBuffers ();     //swap to buffer that is finished with calculating its dispaly
}

////////////////////////////////////////////////////////////////////
//  display3
//-----------------------------------------------------------------
//  Purpose:  Display the points from the array twice. One set the color
//            the color is red, point size is 2 pixels, rotated 1 deg
//            around the y axis, and translated 2% of the screen width
//            in the x direction.
//            The other set the color is blue, point size is 2 pixels
//            rotated -1 deg around the y axis, and translated -2% of
//            the screen width in the x direction.
//            A three axis grid is also plotted.
//  Input:    nothing
//  Output:   the points have been plotted on the screen twice, different
//            colors, rotated and translated with respect to each other
////////////////////////////////////////////////////////////////////

void display3 ( )
{
  int i = 0;                        //loop counter
  glClear( GL_COLOR_BUFFER_BIT ); //buffer for background, colors the background
                                  //using color black

  glPushMatrix ();  //glPushMatrix and glPopMatrix required so you start again at 0, 0, 0 for each
 display
  glColor3f(1.0, 0.0, 0.0);         //R, G, Blu, 1,0,0 is red, color of points
  glPointSize( 2.0 );               //size of points in pixels
  glRotatef ( 1.0, 0.0, 1.0, 0.0 ); //rotate around y axis for 3D
  glTranslatef ( 0.02, 0.0, 0.0 );  //translate on x axis for 3D
  glBegin(GL_POINTS);               //draw a point for every glVertex3f until glEnd()

  for ( i = 0; i < n; i++ )         //draw points on screen from array
  {
    glVertex3f ( dataArray [ i ].x, dataArray [ i ].y, dataArray [ i ].z );
  }

  glEnd();                          //end of glBegin()
  glPopMatrix ();

  glPushMatrix ();
  glColor3f(0.0, 0.0, 1.0);         //R, G, Blu, 0,0,1 is blue, color of points
  glPointSize( 2.0 );               //size of points in pixels
  glRotatef ( -1.0, 0.0, 1.0, 0.0 );  //rotate around y axis for 3D
  glTranslatef ( -0.02, 0.0, 0.0 ); //translate on x axis for 3D
  glBegin(GL_POINTS);               //draw a point for every glVertex3f until glEnd()

  for ( i = 0; i < n; i++ )         //draw points on screen from array
  {
    glVertex3f ( dataArray [ i ].x, dataArray [ i ].y, dataArray [ i ].z );
  }

  glEnd();                          //end of glBegin()
  glPopMatrix ();

  glPushMatrix ();
  glColor3f(1.0, 1.0, 1.0);         //R, G, Blu, 1,1,1 is whilte, color of axes
  glBegin(GL_LINES);                //draw 3 axes
    glVertex3f(-1.0, 0.0, 0.0);
    glVertex3f(1.0, 0.0, 0.0);
    glVertex3f(0.0, -1.0, 0.0);
    glVertex3f(0.0, 1.0, 0.0);
    glVertex3f(0.0, 0.0, -1.0);
    glVertex3f(0.0, 0.0, 1.0);
  glEnd();
  glPopMatrix ();
```

```
  glutSwapBuffers ();     //swap to buffer that is finished with calculating its dispaly
}

////////////////////////////////////////////////////////////////////
//  help
//----------------------------------------------------------------
//  Purpose:  displays a help message box
//  Input:    nothing
//  Output:   pops up a help message box
////////////////////////////////////////////////////////////////////

void help ()
{
  MessageBox (NULL, "'h' key:                  displays this help screen\nleft mouse:
 rotates the display counterclockwise around the z axis\nright mouse:           rotates the displa
y clockwise around the z axis\n'right arrow' key:    rotates the display to the right around the y
 axis \n'left arrow' key:      rotates the display to the left around the y axis \n'up arrow' key:
       rotates the display up around the x axis \n'down arrow' key    rotates the display down aro
und the x axis \n'escape' key            to quit the program \n'2' key                      swtiche
s to 2D\n'3' key                     switches to 3D", "Help", MB_SETFOREGROUND );
}

////////////////////////////////////////////////////////////////////
//  GLUT callback function keyboard
//----------------------------------------------------------------
//  Purpose:  activates keyboard, exits program for escape key, calls
//            the help message box if 'h' is pressed, changes to 2-D
//            display if the number 2 is pressed, changes to 3-D display
//            if the number 3 is pressed.
////////////////////////////////////////////////////////////////////

void keyboard ( unsigned char key, int x, int y )
{
  switch ( key )
  {
    case 27:  //escape key
      exit ( 0 );
      break;

    case 'h':
      help ();
      break;

    case '2':
      init2 ();
      break;

    case '3':
      init3 ();
      break;

    default: break;
  }
}

////////////////////////////////////////////////////////////////////
//  GLUT callback function arrowKey
//----------------------------------------------------------------
//  Purpose:  The keyboard arrow keys do not have ASCII values so cannot
//            be activated by the keyboard function.  The arrow keys rotate
//            the display around the x and y axes
////////////////////////////////////////////////////////////////////

void arrowKey ( int key, int x, int y )//special case of keyboard where no ASCII character is assi
gned
{
  switch ( key )
  {
    case GLUT_KEY_LEFT:   //roatate around the y axis to the left
      glRotatef ( -2.0, 0.0, 1.0, 0.0 );
      glutPostRedisplay ();
      break;

    case GLUT_KEY_RIGHT:  //roatate around the y axis to the right
      glRotatef ( 2.0, 0.0, 1.0, 0.0 );
      glutPostRedisplay ();
```

```
      break;

    case GLUT_KEY_UP:     //roatate around the x axis in the up direction
      glRotatef ( -2.0, 1.0, 0.0, 0.0 );
      glutPostRedisplay ();
      break;

    case GLUT_KEY_DOWN:   //roatate around the x axis in the down direction
      glRotatef ( 2.0, 1.0, 0.0, 0.0 );
      glutPostRedisplay ();
      break;

    default:
      break;
  }
}

//////////////////////////////////////////////////////////////////////
//  GLUT callback function mouse
//--------------------------------------------------------------------
//  Purpose:  rotates the display around the z axis when left and right
//            mouse buttons are clicked
//////////////////////////////////////////////////////////////////////

void mouse ( int btn, int state, int x, int y )
{
  if ( btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN )
  {             //roatate around the z axis in the counter clockwise direction
    glRotatef ( 2.0, 0.0, 0.0, 1.0 );
    glutPostRedisplay ();
  }
  if ( btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN )
  {             //roatate around the z axis in the clockwise direction
    glRotatef ( -2.0, 0.0, 0.0, 1.0 );
    glutPostRedisplay ();
  }

}
```