The GL Utility Toolkit (GLUT)

The GL Utility Toolkit (GLUT) serves two major purposes:
1. It interfaces with your operating system and window system
2. It provides various application utilities, such as drawing 3D shapes for you

You can find GLUT (actually freeGLUT) at:
http://freeglut.sourceforge.net/
although we will give you some binaries that are ready-to-use.

Using GLUT to Setup the Window

```c
// set the initial window configuration:
int initWindow = GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH;
// open the window and set its title:
MainWindow = glutCreateWindow(WINDOWTITLE);
// force a call to Display( ):
glutSetWindow(MainWindow);
```

Using GLUT to Specify Event-driven Callback Functions

```c
void Keyboard(unsigned char c, int x, int y)
{
    if (DebugOn != 0)
        fprintf(stderr, "Keyboard: '%c' (0x%0x)", c, c);
    switch (c)
    {
    case 'o': case 'O':
        WhichProjection = ORTHO;
        break;
    case 'p': case 'P':
        WhichProjection = PERSP;
        break;
    case 'q': case 'Q':
    case ESCAPE:
        DoMainMenu(QUIT);     // will not return here
        break;                          // happy compiler
    default:
        fprintf(stderr, "Don't know what to do with keyboard hit: '%c' (0x%0x)", c, c);
    }
    // force a call to Display( ):
    glutSetWindow(MainWindow);
    glutPostRedisplay();
}
```

The MouseButton Callback Function

```c
void MouseButton(int button, int state, int x, int y)
{
    int b = 0;                      // LEFT, MIDDLE, or RIGHT
    if (DebugOn != 0)
        fprintf(stderr, "MouseButton: %d, %d, %d, %d
", button, state, x, y);
    // get the proper button bit mask:
    switch (button)
    {
    case GLUT_LEFT_BUTTON:
        b = LEFT;               break;
    case GLUT_MIDDLE_BUTTON:
        b = MIDDLE;             break;
    case GLUT_RIGHT_BUTTON:
        b = RIGHT;              break;
    default:
        b = 0;
        fprintf(stderr, "Unknown mouse button: %d
", button);
    }
    // button down sets the bit, up clears the bit:
    if (state == GLUT_DOWN)
    {
        Xmouse = x;
        Ymouse = y;
        ActiveButton |= b;              // set the proper bit
    }
    else
    {
        ActiveButton &= ~b;             // clear the proper bit
    }
}
```
```c
void MouseMotion ( int x, int y )
{
  if ( DebugOn != 0 )
    fprintf ( stderr, "MouseMotion: %d, %d\n", x, y );
  if ( ActiveButton & LEFT )
    Xrot += ( ANGFACT * dy );
  if ( ActiveButton & MIDDLE )
    Scale += SCLFACT * (float) ( dx - dy );
  if ( Scale < MINSCALE )
    Scale = MINSCALE;
  Xmouse = x;  // new current position
  Ymouse = y;
  glutPostRedisplay();
}
```

**The MouseMotion Callback Function**

```c
void Animate ( )
{
  int ms = glutGet ( GLUT_ELAPSED_TIME );
  ms %= MS_IN_ANIMATION_CYCLE;
  Time = (float)ms / (float)MS_IN_ANIMATION_CYCLE;        // [ 0., 1. )
  // put animation stuff in here -- change some global variables
  // for Display( ) to find:
  glutSetWindow ( MainWindow );
  glutPostRedisplay();
}
```

**The Animate Idle Callback Function**

```c
void InitMenus ( )
{
  glutSetWindow ( MainWindow );
  int numColors = sizeof ( Colors ) / (3*sizeof(int));
  int colormenu = glutCreateMenu ( DoColorMenu );
  for ( int i = 0; i < numColors; i++ )
    glutAddMenuEntry ( ColorNames[i], i );
  int axesmenu = glutCreateMenu ( DoAxesMenu );
  glutAddMenuEntry ( "Off", 0 );
  glutAddMenuEntry ( "On", 1 );
  int depthcuemenu = glutCreateMenu ( DoDepthMenu );
  glutAddMenuEntry ( "Off", 0 );
  glutAddMenuEntry ( "On", 1 );
  int debugmenu = glutCreateMenu ( DoDebugMenu );
  glutAddMenuEntry ( "Off", 0 );
  glutAddMenuEntry ( "On", 1 );
  int projmenu = glutCreateMenu ( DoProjectMenu );
  glutAddMenuEntry ( "Orthographic", ORTHO );
  glutAddMenuEntry ( "Perspective", PERSP );
  int mainmenu = glutCreateMenu ( DoMainMenu );
  glutAddSubMenu ( "Axes", axesmenu );
  glutAddSubMenu ( "Colors",        colormenu);
  glutAddSubMenu ( "Depth Cue",     depthcuemenu);
  glutAddSubMenu ( "Projection",    projmenu );
  glutAddMenuEntry ( "Reset", RESET );
  glutAddSubMenu ( "Debug", debugmenu);
  glutAddMenuEntry ( "Quit", QUIT );
  glutAttachMenu ( GLUT_RIGHT_BUTTON );
}
```

**Pop-up Menus are easy to Create with GLUT**

```
//Without lighting, the GLUT solids don't look very cool. I'd recommend you stick with the wireframe versions of the GLUT 3D Objects for now! We will get to lighting soon.
```

**The GLUT 3D Objects**

```c
void glutSolidSphere( radius, slices, stacks );
glutWireSphere( radius, slices, stacks );
glutSolidCube( size );
glutWireCube( size );
glutSolidCone( base height, slices, stacks );
glutWireCone( base height, slices, stacks );
glutSolidTeapot( size );
glutWireTeapot( size );
```

**The GLUT 3D Objects**

```c
// Without lighting, the GLUT solids don't look very cool. I'd recommend you stick with the wireframe versions of the GLUT 3D Objects for now! We will get to lighting soon.
```