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

All the GLUT_XXX constants are #defined in glut.h

```c
glutDisplayMode( GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH );
```

// set the initial window configuration:
```c
glutInitDisplayMode( GLUT_RGBA );
glutInitWindowSize( INIT_WINDOW_SIZE, INIT_WINDOW_SIZE );
```

// open the window and set its title:
```c
MainWindow = glutCreateWindow( WINDOWTITLE );
glutSetWindowTitle( WINDOWTITLE );
```

Constants not beginning with GL_ or GLUT_ are user-defined

Using GLUT to Specify Event-driven Callback Functions

For example, the `Keyboard()` function gets called when a keyboard key is hit

```c
void Keyboard( unsigned char c, int x, int y )
{
...}
```

The `Keyboard` Callback Function

Where the mouse was when the key was hit
The key that was hit

```c
void MouseButton( int button, int state, int x, int y )
{
...}
```

The `MouseButton` Callback Function
void MouseMotion( int x, int y )
{
    if( DebugOn != 0 )
        fprintf( stderr, "MouseMotion: %d, %d
", x, y );
    int dx = x - Xmouse;            // change in mouse coords
    int dy = y - Ymouse;
    if( ( ActiveButton & LEFT ) != 0 )
    { 
        Xrot += ( ANGFACT * dy );
        Yrot += ( ANGFACT * dx );
    }
    if( ( ActiveButton & MIDDLE ) != 0 )
    { 
        Scale += SCLFACT * (float) ( dx - dy );
        if( Scale < MINSCALE )
            Scale = MINSCALE;
    }
    Xmouse = x;                     // new current position
    Ymouse = y;
    glutSetWindow( MainWindow );
    glutPostRedisplay( );
}

The MouseMotion Callback Function

The Animate Idle Callback Function

The Idle Function gets called when the GLUT event handler has nothing else to do

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

We’ll talk about this later. This is a good way to control your animations!

void glutIdleFunc( Animate );

The GLUT 3D Objects

glutSolidSphere( radius, slices, stacks );
glutWireSphere( radius, slices, stacks ) ;
glutSolidCube( size );
glutWireCube( size );
glutSolidCone( base, height, slices, stacks );
glutWireCone ( base, height, slices, stacks );
glutSolidTorus( innerRadius, outerRadius, nsides, nrings );
glutWireTorus( innerRadius, outerRadius, nsides, nrings );
glutSolidDodecahedron( );
glutWireDodecahedron( );
glutSolidOctahedron( );
glutWireOctahedron( );
glutSolidTetrahedron( );
glutWireTetrahedron( );
glutSolidTeapot( size );
glutWireTeapot( size );
glutSolidIcosahedron( );
glutWireIcosahedron( );
glutSolidTeapot( size );
glutWireTeapot( size );

The GLUT 3D Objects

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.

Without lighting

With lighting

Pop-up Menus are easy to Create with GLUT

This is the color menu’s callback function. When the user selects from this pop-up menu, its callback function gets executed. Its argument is the integer ID of the menu item that was selected. You specify that integer ID in glutAddMenuEntry().

This is how you create hierarchical sub-menus

Finally, tell GLUT which mouse button activates the entire menu hierarchy

The GLUT 3D Objects

In case you have a hard time remembering which direction "slices" are, think of this:

This is how you create hierarchical sub-menus

Without lighting

With lighting

The GLUT 3D Objects