The GL Utility Toolkit (GLUT)

Oregon State University

Mike Bailey
mjb@cs.oregonstate.edu

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License

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/

You don’t actually have to go out here. 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

glutDisplayMode( GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH );
// set the initial window configuration:
glutInitWindowSize( INIT_WINDOW_SIZE, INIT_WINDOW_SIZE );
// open the window and set its title:
MainWindow = glutCreateWindow( WINDOWTITLE );

All the GLUT_XXX constants are defined in glut.h

Using GLUT to Specify Event-driven Callback Functions

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

A NULL callback function means that this event will be ignored

The Keyboard Callback Function

Where the mouse was when the key was hit

The key that was hit

The MouseButton Callback Function

Where the button was when the mouse button was pressed

The button that was pressed
### The MouseMotion Callback Function

```c
void
MouseMotion(int x, int y)
{
  if (DebugOn != 0)
    fprintf(stderr, "MouseMotion: %d, %d\n", x, y);
  dx = x - Xmouse;            // change in mouse coords
  dy = y - Ymouse;
  if (ActiveButton & LEFT) != 0)
    Xrot += (ANGFACT*dy);
  if (ActiveButton & MIDDLE) != 0)
    Scale += SCLFACT * (float) (dx - dy);
    // keep object from turning inside-out or disappearing:
    if (Scale < MINSCALE)
      Scale = MINSCALE;
  Xmouse = x;                     // new current position
  Ymouse = y;
  glutSetWindow(MainWindow);
  glutPostRedisplay();
}
```

### The Animate Idle Callback Function

```c
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:
  // force GLUT to do a call to Display() next time it is convenient:
  glutSetWindow(MainWindow);
  glutPostRedisplay();
}
```

### 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();`
- `glutSolidIcosahedron();`
- `glutWireIcosahedron();`
- `glutSolidTeapot(size);`
- `glutWireTeapot(size);`

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.