

OpenGL Utility ToolKit (GLUT)

```
glutInit (&argc, argv);
glutInitDisplayMode (GLUT_DEPTH | GLUT_DOUBLE |
GLUT_RGBA);
glutInitWindowPosition (100,100);
glutInitWindowSize (320,320);
glutCreateWindow ("GLUT Tutorial");

glutDisplayFunc (renderScene);
glutMainLoop ();
```

1

Reshape

- ❑ void glutReshapeFunc(void (*func)(int width, int height));
 - *func* – The name of the function that will be responsible for setting the correct perspective when the window changes size

```
void changeSize(int width, int height)
{
    // Set the viewport to be the entire window
    glViewport(0, 0, width, height);

    // Use the Projection Matrix
    glMatrixMode(GL_PROJECTION);

    // Reset Matrix
    glLoadIdentity();

    gluOrtho2D (0, width, 0, height);
}
```

2

Animation

- ❑ void glutIdleFunc(void (*func)(void));
 - *func* – The name of the function that will be called whenever the application is idle

```
int main(int argc, char **argv) {
    ...
    // register callbacks
    glutDisplayFunc(renderScene);
    glutIdleFunc(renderScene);
    // here is our new entry in the main function
    glutReshapeFunc(changeSize);
    // here are the new entries
    glutKeyboardFunc(processNormalKeys);
    glutSpecialFunc(processSpecialKeys);

    // enter GLUT event processing cycle
    glutMainLoop();
}
```

3

Keyboard

- ❑ void glutKeyboardFunc(void (*func)(unsigned char key, int x, int y));
 - *func* – The name of the function that will process the “normal” keyboard events. Passing NULL as an argument causes GLUT to ignore “normal” keys
- ❑ void glutSpecialFunc(void (*func)(int key, int x, int y));
 - *func* – The name of the function that will process the special keyboard events. Passing NULL as an argument causes GLUT to ignore the special keys

4

Keyboard (cont'd)

```
void processNormalKeys (unsigned char key, int x, int y) {
    if (key == 27)
        exit(0);
}

void processSpecialKeys(int key, int x, int y) {
    switch(key) {
        case GLUT_KEY_F1 :
            red = 1.0; green = 0.0;
            blue = 0.0; break;
        case GLUT_KEY_F2 :
            red = 0.0; green = 1.0;
            blue = 0.0; break;
        case GLUT_KEY_F3 :
            red = 0.0; green = 0.0;
            blue = 1.0; break;
    }
}
```

5

Mouse

- ❑ void glutMouseFunc(void (*func)(int button, int state, int x, int y));
 - *func* – The name of the function that will handle mouse click events
 - Which button
 - ❖ GLUT_LEFT_BUTTON
 - ❖ GLUT_MIDDLE_BUTTON
 - ❖ GLUT_RIGHT_BUTTON
 - Pressed or release
 - ❖ GLUT_DOWN
 - ❖ GLUT_UP

6

Mouse (cont'd)

- ❑ Detecting motion
- ❑ `void glutMotionFunc(void (*func) (int x,int y));`
- ❑ `void glutPassiveMotionFunc(void (*func) (int x, int y));`
 - `func` – the function that will be responsible for the respective type of motion
 - Active motion occurs when the mouse is moved and a button is pressed
 - Passive motion is when the mouse is moving but no buttons are pressed

7

Mouse (cont'd)

- ❑ Detecting when the mouse enters or leaves the window
- ❑ `void glutEntryFunc(void (*func)(int state));`
 - `func` – the function that will handle these events.
 - ❖ `GLUT_LEFT`
 - ❖ `GLUT_ENTERED`

8

Popup Menu

- ❑ `int glutCreateMenu(void (*func)(int value));`
 - `func` – the function that will handle the menu events for the newly created menu
- ❑ `void glutAddMenuEntry(char *name, int value);`
 - `name` – the string that will show up in the menu
 - `value` – this is the value that will be returned to the callback function when the menu entry is selected
- ❑ `void glutAttachMenu(int button);`
 - `button` – an integer that specifies which button the menu will be attached to
 - ❖ `GLUT_LEFT_BUTTON`
 - ❖ `GLUT_MIDDLE_BUTTON`
 - ❖ `GLUT_RIGHT_BUTTON`

9

Popup Menu (cont'd)

```
...
#define RED 1
#define GREEN 2
#define BLUE 3
#define ORANGE 4
...
void createGLUTMenus() {
    int menu = glutCreateMenu(processMenuEvents);
    //add entries to our menu
    glutAddMenuEntry("Red",RED);
    glutAddMenuEntry("Blue",BLUE);
    glutAddMenuEntry("Green",GREEN);
    glutAddMenuEntry("Orange",ORANGE);
    // attach the menu to the right button
    glutAttachMenu(GLUT_RIGHT_BUTTON);
}
```

10

Popup Menu (cont'd)

```
void processMenuEvents(int option) {
    switch (option) {
        case RED :
            red = 1.0f; green = 0.0f;
            blue = 0.0f; break;
        case GREEN :
            red = 0.0f; green = 1.0f;
            blue = 0.0f; break;
        case BLUE :
            red = 0.0f; green = 0.0f;
            blue = 1.0f; break;
        case ORANGE :
            red = 1.0f; green = 0.5f;
            blue = 0.5f; break;
    }
}
```

11

Sub Menu

- ❑ `void glutAddSubMenu(char *entryName, int menuIndex);`
 - `entryName` – The name of the submenu entry in the menu
 - `menuIndex` – The index of the submenu, this is the value that we get as a return value when calling `glutCreateMenu` for the submenu.

12

Sub Menu (con'd)

```
void createPopupMenu() {
    shrinkMenu = glutCreateMenu(processShrinkMenu);
    glutAddMenuEntry("Shrink", SHRINK);
    glutAddMenuEntry("NORMAL", NORMAL);

    fillMenu = glutCreateMenu(processFillMenu);
    glutAddMenuEntry("Fill", FILL);
    glutAddMenuEntry("Line", LINE);

    colorMenu = glutCreateMenu(processColorMenu);
    glutAddMenuEntry("Red", RED);
    glutAddMenuEntry("Blue", BLUE);
    glutAddMenuEntry("Green", GREEN);
    glutAddMenuEntry("Orange", ORANGE);

    mainMenu = glutCreateMenu(processMainMenu);
    glutAddSubMenu("Polygon Mode", fillMenu);
    glutAddSubMenu("Color", colorMenu);

    // attach the menu to the right button
    glutAttachMenu(GLUT_RIGHT_BUTTON);
}
```

13

glutPostRedisplay

- ❑ void glutPostRedisplay(void);
 - Mark the normal plane of *current window* as needing to be redisplayed. The next iteration through glutMainLoop, the window's display callback will be called to redisplay the window's normal plane.

14