Introduction to Graphics Programming and its Applications

繪圖程式設計與應用

GLUT

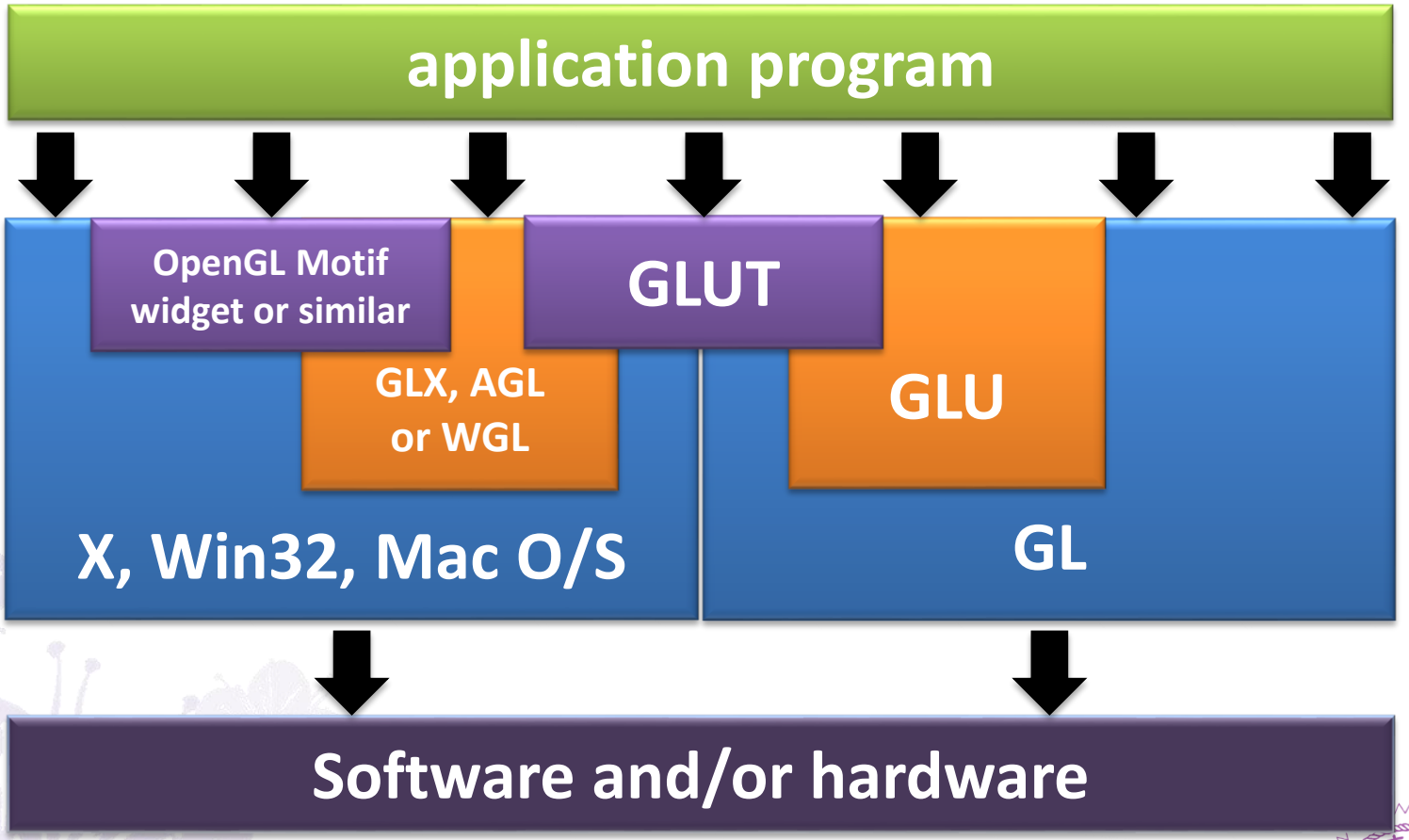
Instructor: Hung-Kuo Chu

Department of Computer Science

National Tsing Hua University

CS4595

Software Organization



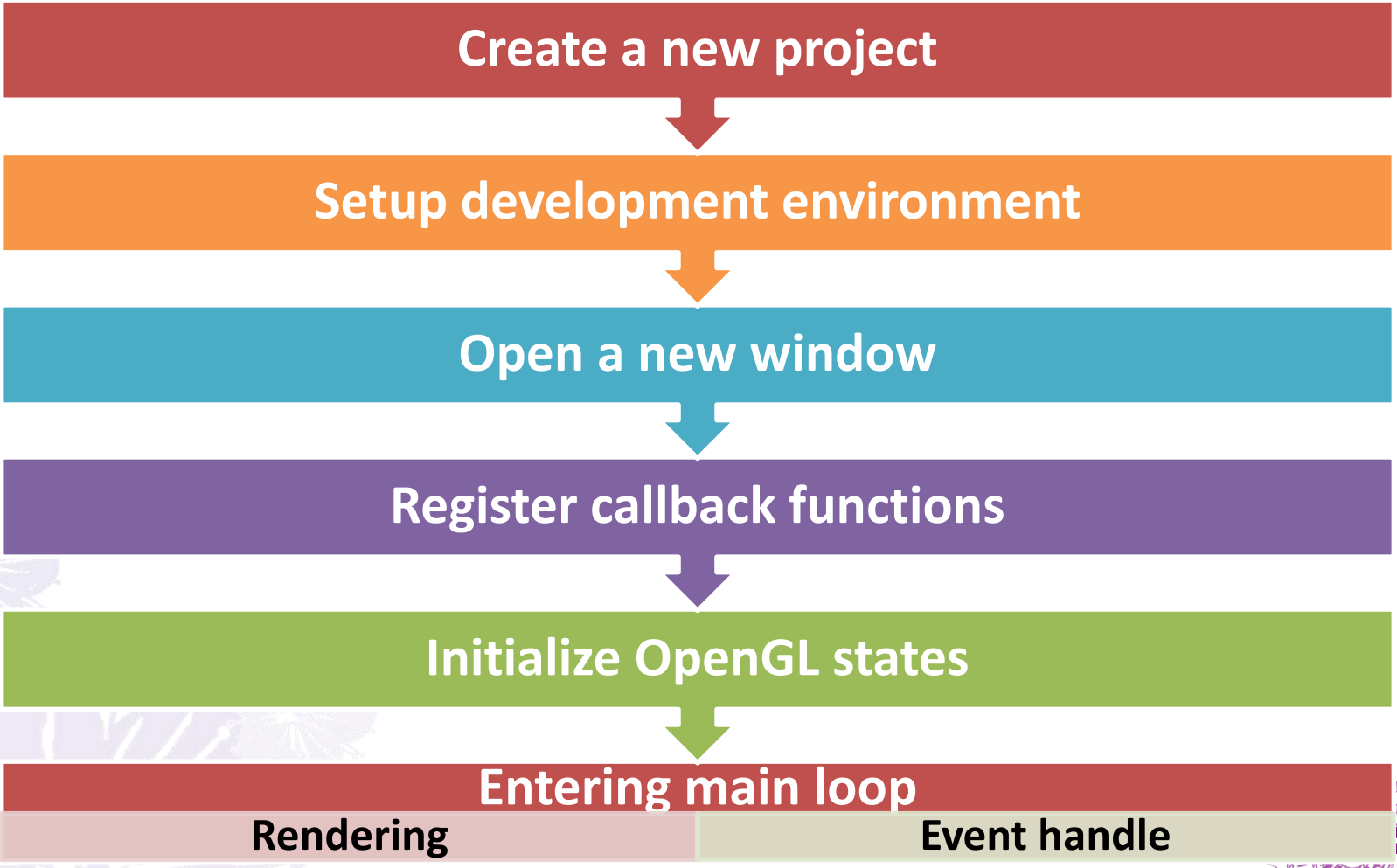
GLUT (OpenGL Utility Toolkit)

- Provides functionality common to all window systems
  - Open a window
  - Get input from mouse and keyboard
  - Menus
  - Event-driven

GLUT (OpenGL Utility Toolkit)

- GLUT was originally written by *Mark Kilgard* to support the sample programs in the second edition OpenGL 'RedBook‘
- The original GLUT project is no longer maintained; We use FreeGLUT nowadays.
- Latest version: FreeGLUT 3.0.0
- You may also want to use: GLFW

Using GLUT : Step-by-Step



Preliminaries

- GLUT library (FreeGLUT) files
  - Header files
    - freeglut.h
    - freeglut_ext.h
    - freeglut_std.h
    - glut.h
  - Library file
    - freeglut.lib
  - Binary file
    - freeglut.dll

OPEN A NEW WINDOW

Code Snippet

```
int main(int argc, char* argv[])
{
    /// Create a new window
    ////////////////////////////////////////////////////////////
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowPosition(100,100);
    glutInitWindowSize(600,600);
    glutCreateWindow("SimpleApp");
    ////////////////////////////////////////////////////////////
    /// Initialize OpenGL
    InitGL();

    /// Entering main loop
    glutMainLoop();

    return 0;
}
```

glutInitDisplayMode()

- Specify required data for each pixel in frame buffer
- GLUT_RGBA
  - RGBA color mode
- GLUT_DOUBLE
  - A double-buffered window
- GLUT_DEPTH
  - Allocate depth information

Double Buffering

- The drawing commands are actually executed on an off-screen buffer and then quickly swapped into view on the window later.
- Avoid flashing effect when doing animation

Double Buffering cont.

- Instead of one color buffer, we use two
  - Front Buffer: one that is displayed
  - Back Buffer: one that is written to
- Program then requests a double buffer in
  - glutInitDisplayMode(GL_RGB | GL_DOUBLE)
  - At the end of the display callback buffers are swapped using ***glutSwapBuffers()*** command

INITIALIZE OPENGL STATES

Code Snippet

```
int main(int argc, char* argv[])
{
    /// Create a new window
    ////////////////////////////////////////////////////////////
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowPosition(100,100);
    glutInitWindowSize(600,600);
    glutCreateWindow("SimpleApp");
    ////////////////////////////////////////////////////////////
```

```
    /// Initialize OpenGL
    InitGL();
```

```
    ////////////////////////////////////////////////////////////

    /// Create GLUT menu system
    My_Menu();

    /// Initialize OpenGL
    InitGL();

    /// Entering main loop
    glutMainLoop();

    return 0;
}
```

InitGL() - Code Snippet

– Setup OpenGL initial states

```
////////////////////////////////////////////////////////////////////////
// Setup the rendering state
void InitGL(void)
{
    /// Setup background color
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

    /// Enable depth testing
    glEnable(GL_DEPTH_TEST);

    /// Disable lighting
    glDisable(GL_LIGHTING);
}
```

CREATE GLUT CONTEXT MENU

Menus

- GLUT supports pop-up menus
  - A menu can have submenus
- Three steps
  - Define entries for the menu
  - Define action for each menu item
    - Action carried out if entry selected
  - Attach menu to a mouse button

Defining a Simple Menu

entries that appear when right button depressed

menu callback

identifiers

```
menu_id = glutCreateMenu(menu_main_func);
glutAddMenuEntry("clear Screen", 1);

glutAddMenuEntry("exit", 2);

glutAttachMenu(GLUT_RIGHT_BUTTON);
```

clear screen
exit

specify the action that triggers the menu

Menu Actions

– Menu callback

```
void menu_main_func(int id)
{
        if(id == 1) glClear();
        if(id == 2) exit(0);
}
```

– Note each menu has an id that is returned when it is created

– Add submenus by

- glutAddSubMenu(char *submenu_name, submenu id)

entry in parent menu

Code Snippet

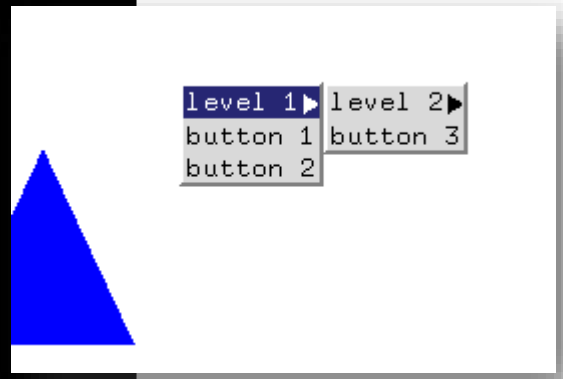
```
////////////////////////////////////////////////////////////
/// Create GLUT menu system
void My_Menu( void )
{
    int menu_main, menu_lvl1, menu_lvl2;

    /// Setup menu callback functions
    menu_main = glutCreateMenu(menu_main_func);
    menu_lvl1 = glutCreateMenu(menu_lvl1_func);
    menu_lvl2 = glutCreateMenu(menu_lvl2_func);

    glutSetMenu( menu_main );
    glutAddSubMenu( "level 1", menu_lvl1 );
    glutAddMenuEntry("button 1",1);
    glutAddMenuEntry("button 2",2);

    glutSetMenu( menu_lvl1 );
    glutAddSubMenu( "level 2", menu_lvl2 );
    glutAddMenuEntry("button 3",1);

    /// Bind menu to right mouse button
    glutSetMenu( menu_main );
    glutAttachMenu(GLUT_RIGHT_BUTTON);
}
```



menu_main_func() - Code Snippet

– Callback function for main menu

```
////////////////////////////////////////////////////////////
/// Main menu callback function
void menu_main_func(int value)
{
    switch(value)
    {
    case 1:
        cout << "Main menu : button 1" << endl;
        break;
    case 2:
        cout << "Main menu : button 2" << endl;
        break;
    }
}
```

REGISTER CALLBACK FUNCTIONS

Callbacks

- Programming interface for event-driven input
- Define a *callback function* for each type of event the graphics system recognizes
- This user-supplied function is executed when the event occurs

GLUT Callbacks

- GLUT recognizes a subset of the events recognized by any particular window system (Windows, X, Macintosh)
  - glutDisplayFunc
  - glutMouseFunc
  - glutReshapeFunc
  - glutKeyboardFunc
  - glutTimerFunc
  - glutIdleFunc

Code Snippet

```
int main(int argc, char* argv[])
{
    /// Create a new window
```

```
/// Register callback functions
////////////////////////////////////////////////////////////
glutReshapeFunc ( My_Reshape );
glutDisplayFunc ( My_Display );
glutKeyboardFunc( My_Keyboard );
glutMouseFunc   ( My_Mouse );
glutSpecialFunc ( My_SpecialKeys );
glutTimerFunc   (timer_speed, My_Timer, timer_flag);
////////////////////////////////////////////////////////////
```

```
    /// Initialize OpenGL
    InitGL();

    /// Entering main loop
    glutMainLoop();

    return 0;
}
```

The Reshape Callback

- glutReshapeFunc(myreshape)
- void myreshape(int w, int h)
  - Returns width and height of new window (in pixels)
  - A redisplay is posted automatically at end of execution of the callback
  - GLUT has a default reshape callback but you probably want to define your own
- The reshape callback is good place to put viewing functions because it is invoked when the window is first opened

My_Reshape() - Code Snippet

– Whenever the window is resized glut calls this function

```
//////////////////////////////////////////////////////////////
// Called by GLUT library when the window has changed size
void My_Reshape(int w, int h)
{
    GLfloat aspectRatio;

    // Prevent a divide by zero
    if(h == 0)
        h = 1;

    // Set Viewport to window dimensions
    glViewport(0, 0, w, h);

    // Reset coordinate system
    glMatrixMode(GL_PROJECTION);
```

Width and height of new window

The Display Callback

- The display callback is executed whenever GLUT determines that the window should be refreshed, for example
  - When the window is first opened
  - When the window is reshaped
  - When a window is exposed
  - When the user program decides it wants to change the display
- Every GLUT program must have a display callback

My_Display() - Code Snippet

– Call ***glutSwapBuffers*** to swap back/front buffers

```
////////////////////////////////////////////////////////
// Called to draw scene
void My_Display(void)
{
    /// Clear the window with current clearing color
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    /// Draw a triangle
    glBegin(GL_TRIANGLES);
    glColor3ub(timer_cnt, 0, 255-timer_cnt);
    glVertex3fv(tri_v1);
    glColor3ub(255, timer_cnt, 255-timer_cnt);
    glVertex3fv(tri_v2);
    glColor3ub(255-timer_cnt, 0, timer_cnt);
    glVertex3fv(tri_v3);
    glEnd();

    /// Flush drawing commands
    glutSwapBuffers();
}
```

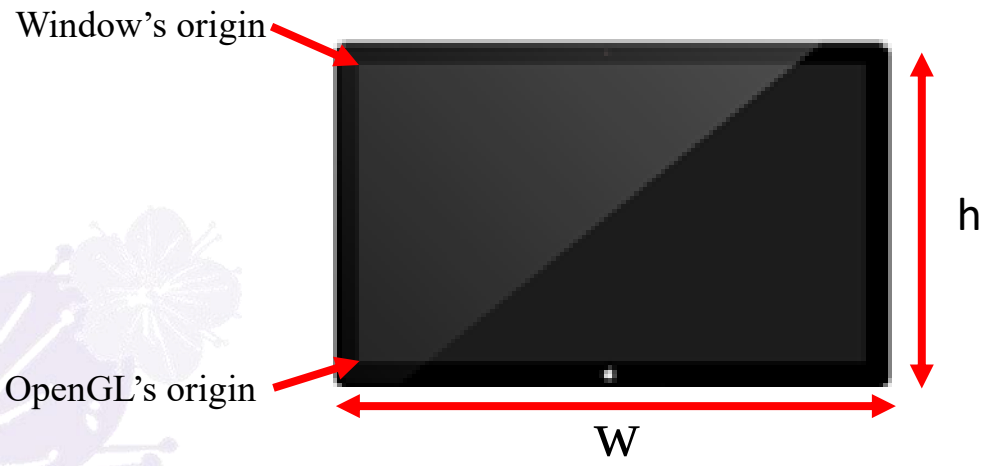
The Mouse Callback

```
glutMouseFunc(mymouse)
void mymouse(GLint button, GLint
  state, GLint x, GLint y)
```

- Returns
  - which button
    - **GLUT_LEFT_BUTTON**
    - **GLUT_MIDDLE_BUTTON**
    - **GLUT_RIGHT_BUTTON**
  - state of that button
    - **GLUT_UP**
    - **GLUT_DOWN**
  - Position in window

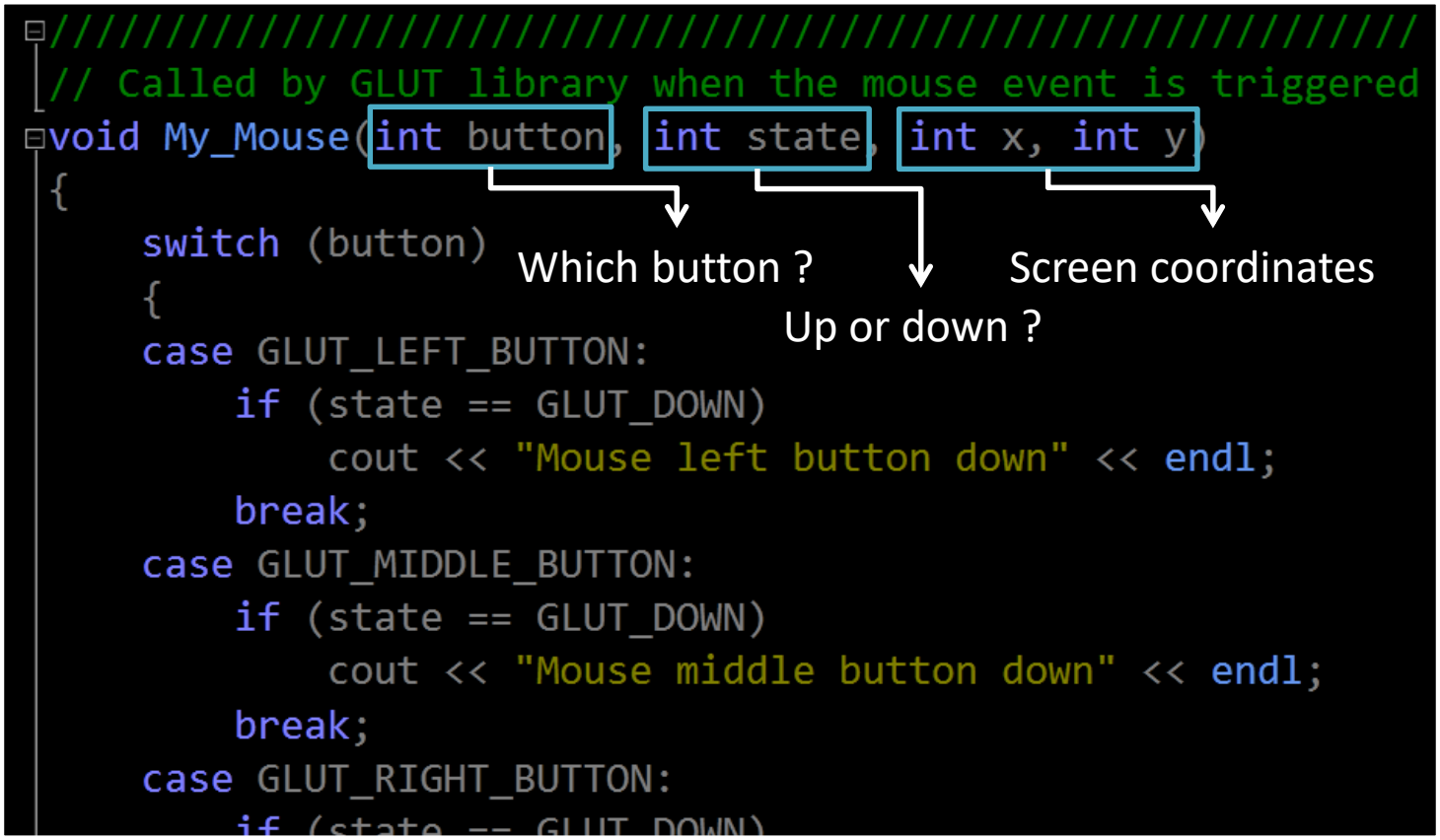
Positioning

- The position in the screen window is usually measured in pixels with the origin at the top-left corner
  - Consequence of refresh done from top to bottom
- OpenGL uses a world coordinate system with origin at the bottom left
  - Must invert y coordinate returned by callback using $y_{ogl} = h - y_{win}$;



My_Mouse() - Code Snippet

– Handle the mouse events



The Keyboard Callback

- glutKeyboardFunc(mykey)
- void mykey(unsigned char key, int x, int y)
  - Returns ASCII code of key depressed and mouse location

- Can also check of one of the modifiers is depressed by glutGetModifiers()
  - GLUT_ACTIVE_SHIFT
  - GLUT_ACTIVE_CTRL
  - GLUT_ACTIVE_ALT
  - Allows emulation of three-button mouse with one- or two-button mice

My_Keyboard() - Code Snippet

– Handle the keyboard events

```
////////////////////////////////////////////////////////////
// Called by GLUT library when the keyboard event is triggered
void My_Keyboard( unsigned char key, int x, int y )
{
    switch( key ) {
    case 'q' : case 'Q' :
        exit(0); /// quit the program
        break;
    case 'f' : case 'F' :
        /// enter/leave full-screen mode
        glutFullScreenToggle();
        break;
    case 'p' : case 'P':
        /// stop/resume timer
        if(timer_flag == 0)
```

My_SpecialKeys() - Code Snippet

– Handle the special keyboard events

```
//////////////////////////////////////////////////////////
// Called by GLUT library when the special keyboard event is triggered
void My_SpecialKeys( int key, int x, int y )
{
    switch( key ) {
    case GLUT_KEY_F1 :
        cout << "This is F1 key" << endl;
        break;
    case GLUT_KEY_PAGE_UP :
        cout << "This is PageUp key" << endl;
        break;
    case GLUT_KEY_LEFT :
        cout << "This is Left key" << endl;
        break;
    }
}
```

Animation in GLUT

– glutTimerFunc

- Register a timer callback to be triggered in a specified time period using milliseconds (Only once!).
- Multiple timer callbacks at same or differing times

*void glutTimerFunc(unsigned int msecs , void (*func)(int value), value);*

– Call ***glutPostRedisplay*** to refresh the screen

Animation in GLUT alternative

– glutIdleFunc

- Sets the global idle callback.
- Only one idle function

```
void glutIdleFunc ( void (*func)());
```

- Can be easily stopped by

```
glutIdleFunc ( NULL );
```

– Call ***glutPostRedisplay*** to refresh the screen

My_Timer() - Code Snippet

```
//////////////////////////////////////////////////////////////
// Called by GLUT library when the special keyboard event is triggered
void My_Timer( int value )
{
    if(value == 0) return;

    timer_cnt++;
    timer_cnt = timer_cnt % 256;

    glutPostRedisplay();
    glutTimerFunc(timer_speed, My_Timer, timer_flag);
}
```

Stop / Resume Timer

```
case 'p' : case 'P':
    /// stop/resume timer
    if(timer_flag == 0)
    {
        timer_flag = 1;
        glutTimerFunc(timer_speed, My_Timer, timer_flag);
    }
    else
        timer_flag = 0;
    break;
```

ENTERING MAIN LOOP

GLUT Event Loop

- Recall that the last line in a program using GLUT must be
  - ***glutMainLoop();***
- which puts the program in an infinite event loop
- In each pass through the event loop, GLUT looks at the events in the queue for each event in the queue, GLUT executes the appropriate callback function if one is defined if no callback is defined for the event, the event is ignored

Code Snippet

```
int main(int argc, char* argv[])
{
    /// Create a new window
    ///////////////////////////////////////////////////////
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowPosition(100,100);
    glutInitWindowSize(600,600);
    glutCreateWindow("SimpleApp");
    ///////////////////////////////////////////////////////

    /// Register callback functions
    ///////////////////////////////////////////////////////
    glutReshapeFunc ( My_Reshape );
    glutDisplayFunc ( My_Display );
    glutKeyboardFunc( My_Keyboard );
    glutMouseFunc   ( My_Mouse );
    glutSpecialFunc ( My_SpecialKeys );
```

```
    /// Entering main loop
    glutMainLoop();

    return 0;
}
```