Window programming with GLUT
The Windows Programming Model

- **Traditional Programs: Procedural programming model.**
  - Programs execute top-to-bottom.
  - Main function handles program flow.

- **Windows programs: Event-driven programming model.**
  - Applications respond to *events*, by processing *messages*.
  - Message loop waits for messages.

![Diagram of Windows programming model](image-url)
while (GetMessage(&msg, NULL, 0, 0))
{
    if (!TranslateAccelerator(
            msg.hwnd, hAccelTable, &msg))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
}

The Message Loop
Many Years Ago ... Windows Applications written in C:

- Knowing the ins and outs of the operating system
- Knowing hundreds of different API functions
- Knowing hundreds of different messages

LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam) {
    switch (msg) {
    case WM_LBUTTONDOWN:
        char szFileName[MAX_PATH];
        HINSTANCE hInstance = GetModuleHandle(NULL);
        GetModuleFileName(hInstance, szFileName, MAX_PATH);
        MessageBox(hwnd, szFileName, "This program is:", MB_OK |
            MB_ICONINFORMATION);
        break;
    case WM_CLOSE: DestroyWindow(hwnd); break;
    case WM_DESTROY: PostQuitMessage(0); break;
    default: return DefWindowProc(hwnd, msg, wParam, lParam);
    }
    return 0;
}
<table>
<thead>
<tr>
<th>Message</th>
<th>Sent When</th>
</tr>
</thead>
<tbody>
<tr>
<td>WM_CHAR</td>
<td>A character is input from the keyboard.</td>
</tr>
<tr>
<td>WM_COMMAND</td>
<td>The user selects an item from a menu, or a control sends a notification to its parent.</td>
</tr>
<tr>
<td>WM_CREATE</td>
<td>A window is created.</td>
</tr>
<tr>
<td>WM_DESTROY</td>
<td>A window is destroyed.</td>
</tr>
<tr>
<td>WM_LBUTTONDOWN</td>
<td>The left mouse button is pressed.</td>
</tr>
<tr>
<td>WM_LBUTTONUP</td>
<td>The left mouse button is released.</td>
</tr>
<tr>
<td>WM_MOUSEMOVE</td>
<td>The mouse pointer is moved.</td>
</tr>
<tr>
<td>WM_PAINT</td>
<td>A window needs repainting.</td>
</tr>
<tr>
<td>WM_QUIT</td>
<td>The application is about to terminate.</td>
</tr>
<tr>
<td>WM_SIZE</td>
<td>A window is resized.</td>
</tr>
</tbody>
</table>
The **MSG** struct contains message info:

```c
typedef struct MSG
{
    HWND hwnd;          // Uniquely identifies a window.
    UINT message;      // Specifies the message type.
    WPARAM wParam;     // Additional information
    LPARAM lParam;     // about the message.
    ...
} MSG;
```

For **wParam** and **lParam**, the exact meaning depends on the value of the message member. For WM_LBUTTONDOWN it is the state of the Ctrl or Shift keys, and the mouse coordinates.
OpenGL Utility Toolkit (GLUT)

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

- Code is portable but GLUT lacks the functionality of a good toolkit for a specific platform
  - No slide bars
GLUT was created long ago and has been unchanged
- Amazing that it works with OpenGL 3.1
- Some functionality can’t work since it requires deprecated functions

freeglut updates GLUT
- Added capabilities
- Context checking
#include <GL/glut.h>
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    // need to fill in this part
    // and add in shaders
}
int main(int argc, char** argv) {
    glutCreateWindow("simple");
    glutDisplayFunc(display);
    glutMainLoop();
}
Event Loop

- Note that the program specifies a display *callback* function named `display`
  - Every glut program must have a display callback
  - The display callback is executed whenever Windows decides the display must be refreshed, for example when the window is opened
  - The main function ends with the program entering an event loop
Event Types

- Window: resize, expose, iconify
- Mouse: click one or more buttons
- Motion: move mouse
- Keyboard: press or release a key
- Idle: nonevent

Define what should be done if no other event is in queue
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 example: glutMouseFunc(mouse)
GLUT callbacks

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

- The last line in main.c for 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
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
Posting redisplay

- Many events may invoke the display callback function
  - Can lead to multiple executions of the display callback on a single pass through the event loop

- We can avoid this problem by instead using
  - `glutPostRedisplay();`
  - which sets a flag.

- GLUT checks to see if the flag is set at the end of the event loop

- If set then the display callback function is executed