Lecture notes: GUI Event Driven Programming

A **GUI (graphical user interface)** is typically composed of a window, menu bar and system icons for minimizing/maximizing/closing the window. User input to a GUI program is typically given through mouse motion, button presses and key presses. Common elements in a GUI include menus, dialog boxes, message boxes and window scroll bars.

An **event** is a type of input given to a GUI program, typically by a human user. For example, a user may press a key, move a mouse, or minimize a window. These types of input occur in any order and with any amount of time in-between each input. This can be contrasted with structured synchronous input such as data read from a file, where the order and timing of input is sequential, predictable, and can be repeated each time the program is executed.

A GUI is programmed around the processing of events, called **event handling**, and is typically coded using an **event loop**:

```
function event_loop()
  {
  initialize()
  loop
    message = get_event()
    process_event(message)
  until message = QUIT
  }
```

The function get_event() is like a “super-sized” scanf() function call. A scanf() **blocks** program execution and waits until the user types in a string and presses ENTER. A get_event() function call blocks program execution and waits until the user does anything with the GUI.

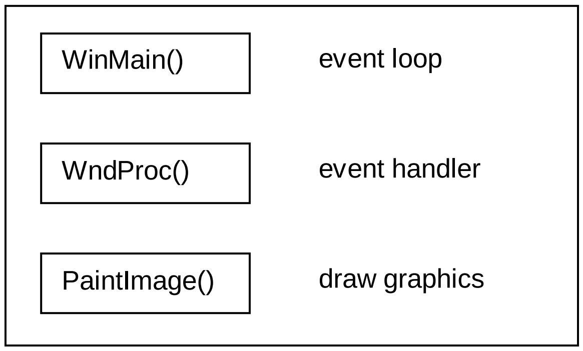
The **operating system (O/S) controls which messages** get sent to a program. Typically events are only sent to a program when one of its windows is active. The program can also use a mask to request that the O/S block one or more message types from coming to the program. This is analogous to a firewall, but for events rather than packets.

An event loop is often coded in the main function of a GUI program. Because the get_event() function is blocking, GUI programs often use **multiple threads**. For example, if the program is intended to display video or other evolving graphics while still being responsive to GUI events, then it will be multi-threaded. The event loop will run in the main thread, while other operations run in child threads.

An event loop can also be coded in a sub-function used for a secondary purpose such as a dialog box or form. In this case, the function is also typically multi-threaded to avoid blocking the main GUI and other operations of the program.

Programming support for a GUI is provided by default in some programming languages such as Java. In other programming languages, it is provided by one or more libraries. Using C, the most common GUI libraries are Win32 (under Microsoft Windows), Xlib (under UNIX/linus), and GTK (cross-platform, built on top of Win32 and Xlib). In either case, the language or library provides a set of functions that interact with GUI elements.

For this class, we will use the Win32 library to examine how to program a GUI and handle events. The following shows a common organization of a Win32 GUI program.



The WinMain() function is the main function of the program. It initializes the program, such as creating a window and setting up default variable values. It then goes into an event loop. One of the variables initialized in WinMain() is the name of the event handling function, which in this example is named WndProc(). The WndProc() function is called to process any event sent to the program by the O/S. Some of these events require no update of the contents of the program window, while others require parts or the entire window be redrawn. This is typically handled by grouping together the code that redraws window content in a third function, which in this example is named PaintImage().

The program **plus** given at the course website will be used for further demonstrations. It was coded using Microsoft Visual Studio 6.0. The project file must be converted for use in later versions. Note that Clemson University has unlimited license to use all versions of the compiler. The latest can be obtained from CCIT. A free but less comprehensive version of the software can also be obtained from Microsoft at http://www.microsoft.com/visualstudio/eng/products/visual-studio-express-products.

In the WinMain() function in the plus program, the first block of code creates a **window class**. This is a set of GUI elements affecting a window such as its icon, border width, background color, etc. A window class can be used to create multiple windows so that they all share the same basic properties. The next blocks of code create a window, set some default variable values, and go into an event loop.

The WndProc() function is the event handler. The plus program demonstrates several example events. The first is the WM_COMMAND which is a command from a menu. Each menu choice generates a message with a unique ID. The plus program demonstrates 3 examples. Other types of events demonstrated include WM_SIZE, which is caused by resizing the window; WM_PAINT which is caused by part of the window being uncovered; WM_LBUTTONDOWN and WM_RBUTTONDOWN which are caused by mouse button presses; WM_MOUSEMOVE which occurs whenever the mouse moves; and WM_KEYDOWN which is caused by the pressing of a keyboard key. There are numerous others, including UP versions of all the down events demonstrated, scrolling events, and other window notifications.

The code for handling a WM_KEYDOWN event demonstrates how a program can send itself a message. This is useful in many instances, for example when a program wants to trigger one of its event handlers in a loop.

The PaintImage() function draws the content in the window. For the plus program, this function draws the loaded image in the window assuming it is an 8-bit grayscale image. This function could also be used to draw text, instructions, coordinates, or other graphics or information.