**Application Programs using Windows API**

I am Alvin Alexander, eighth standard student of Salvation Army HSS, Kowdiar, Thiruvananthapuram.

In the previous session, we learned about the window procedure that is needed for processing certain window messages sent by the system, like WM\_PAINT for painting our window. This session, we will delve into the message loop, which sends messages to the window procedure to be subsequently handled.

This is a conventional message loop:

MSG msg;

ZeroMemory(&msg, sizeof(msg));

while (GetMessage(&msg, NULL, 0, 0) > 0) {

TranslateMessage(&msg);

DispatchMessage(&msg);

}

A very small piece of code, but is the most crucial piece of code for our window. The message loop works as follows: The GetMessage(…) function gets messages from the system deposited in our thread message queue (Here, it is the main thread. Multithreading is possible). The last three parameters let us filter for a particular message. We do not want this; we need all the messages sent by the system. Once a message is received, it is filled into the *msg* variable, after which this message is ran through processing by TranslateMessage(…) and then sent to the destination window procedure by DispatchMessage(…) (Here, the destination window procedure is our window procedure. For multiple windows, the function automatically sends the message to the correct window). TranslateMessage(…) function converts virtual key-code messages like WM\_KEYDOWN or WM\_KEYUP to WM\_CHAR or WM\_DEADCHAR. This is not necessary for our window, as we are not handling keystrokes or mouse clicks, but we are handling it with the mindset for extension (If we want to make this window a keystroke-receiving- and handling window). In WM\_DESTROY message handling in our window procedure, we sent WM\_QUIT message to kill the program. This works because WM\_QUIT message is defined as zero. The *while* loop will end when we receive a message like WM\_QUIT, therefore killing the program after the window is closed.

In the next session, we will work with components to make the GUI more functional.