Handling Messages

Important to study from this code. Essential steps.

▶ Register window class
▶ Create window
▶ Enter message loop
▶ Handle messages via `WndProc()`

The code starts on `WinMain()` which has to have a specific signature or otherwise the linker will complain about overload.

The `WinMain()` function calls `initWindow()` which in order initializes the data members of the `WNDCLASSEX` , notably `WndProc` ,

Then the WNDCLASSEX is registered as a reference.
then it is finally created

From here we enter the msg loop, and call `GetMessage()` with a reference for the message and the windows handle we populated with `initwindow()`

this will run forever until I provide VM_DESTROY by clicking the [x]

WndProc

```
//Function Prototype
LRESULT CALLBACK WndProc(HWND  hwnd, UINT msg, WPARAM wparam, L
PARAM lparam);


//Function definition
LRESULT CALLBACK WndProc(HWND  hwnd, UINT msg, WPARAM wParam, L
PARAM lParam) {
    switch (msg){
    case  WM_CLOSE:
    {
```

```
        DestroyWindow(hwnd);
        break;
    }
    case WM_DESTROY:
        PostQuitMessage(0);
        break;
    default:
        return DefWindowProc(hwnd, msg, wParam, lParam);
    }
}
```

I've modules with the functions specifically separating

1. Windows Registration and creation
2. The message loop

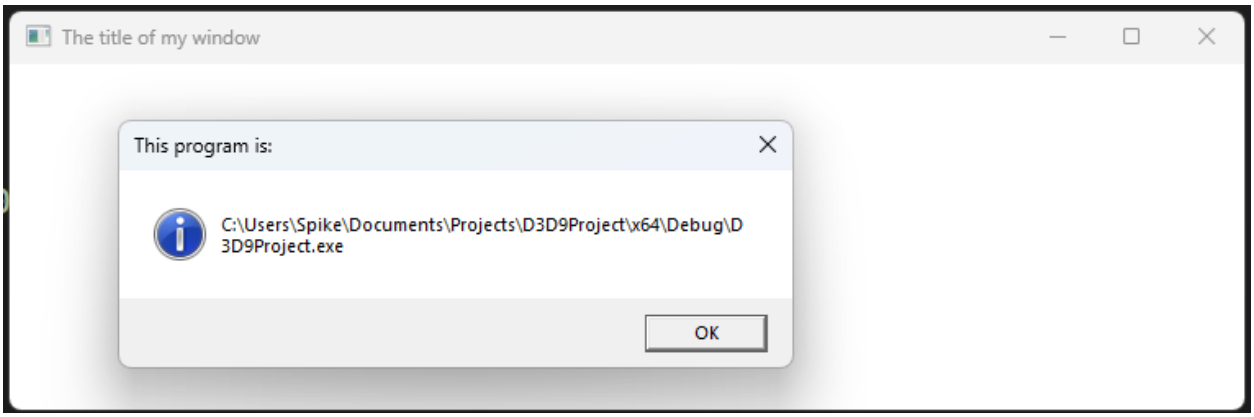
All handled within a single `run()` function that keeps `WinMain` clean.

```
int run(HINSTANCE hInstance, int nCmdShow) {
    if (initWindow(hInstance, nCmdShow) != 0)
        return 1;
    int wParam = messageLoop(hInstance, nCmdShow);

    return  wParam;
}
```

I'm not immediately sure of its advantages but code looks cleaner and if I were to add more functionalities I can keep it separated from windows initialization.

I've also learned that the message loop is stateful, the program is not mean to exit it as long as `VM_QUIT` or `VM_CIOSE` is not called.



The code is getting complex now with so many functions in main.cpp maybe I can start making some interface classes, so far this does not feel like C++ with all the procedural structure of it.