ImGui + GLFW/GLAD + Boost.Signals2 Setup Guide

Prerequisites

- Visual Studio 2022 (recommended)
- vcpkg package manager
- Administrator privileges

1. Environment Setup

1.1 VCPKG Setup

1. Locate your vcpkg installation using Command Prompt:

```
where vcpkg
```

2. Set VCPKG_ROOT environment variable:
    - Open System Properties → Advanced → Environment Variables
    - Add new System Variable:

```
Name: VCPKG_ROOT
Value: Your vcpkg path (e.g., D:\vcpkg)
```

3. Verify setup:

```
echo %VCPKG_ROOT%
```

> 🚨 **IMPORTANT**: Restart Visual Studio after setting environment variables

2. Install Dependencies

2.1 Install Required Packages

Run in Command Prompt as administrator:

```
%VCPKG_ROOT%\vcpkg install glfw3:x64-windows
%VCPKG_ROOT%\vcpkg install glad:x64-windows
%VCPKG_ROOT%\vcpkg install boost-signals2:x64-windows
%VCPKG_ROOT%\vcpkg install imgui[glfw-binding,opengl3-binding]:x64-windows
```

2.2 Verify Libraries

Check these paths:

GLFW Debug Files

```
%VCPKG_ROOT%\installed\x64-windows\debug\bin\glfw3.dll
%VCPKG_ROOT%\installed\x64-windows\debug\lib\glfw3dll.lib
```

GLFW Release Files

```
%VCPKG_ROOT%\installed\x64-windows\bin\glfw3.dll
%VCPKG_ROOT%\installed\x64-windows\lib\glfw3.lib
```

ImGui Debug Files

```
%VCPKG_ROOT%\installed\x64-windows\debug\lib\imguid.lib
%VCPKG_ROOT%\installed\x64-windows\debug\bin\imgui.dll
```

ImGui Release Files

```
%VCPKG_ROOT%\installed\x64-windows\lib\imgui.lib
%VCPKG_ROOT%\installed\x64-windows\bin\imgui.dll
```

3. Project Setup

3.1 Create New Project

1. Open Visual Studio
2. File → New → Project
3. Select "Empty Project" (C++)
4. Set Platform to x64

5. Choose project name/location

3.2 Project Properties Configuration

Right-click project → Properties, then set:

General Settings (All Configurations)

- Platform: x64
- C++ Language Standard: C++17
- Character Set: Use Multi-Byte Character Set
- Windows SDK Version: Latest
- Platform Toolset: Visual Studio 2022 (v143)

Include Directories (All Configurations)

C/C++ → General → Additional Include Directories:

```
$(VCPKG_ROOT)\installed\x64-windows\include
```

Library Directories

Debug Configuration:

```
$(VCPKG_ROOT)\installed\x64-windows\debug\lib
```

Release Configuration:

```
$(VCPKG_ROOT)\installed\x64-windows\lib
```

Additional Dependencies

Debug Configuration:

```
glfw3dll.lib
imguid.lib
opengl32.lib
```

Release Configuration:

```
glfw3.lib
imgui.lib
opengl32.lib
```

Runtime Library

- Debug: Multi-threaded Debug DLL (/MDd)
- Release: Multi-threaded DLL (/MD)

Subsystem

Linker → System → SubSystem:

```
Windows (/SUBSYSTEM:WINDOWS)
```

3.3 Environment PATH Setup

Debugging → Environment:

Debug Configuration:

```
PATH=$(VCPKG_ROOT)\installed\x64-windows\debug\bin;%PATH%
```

Release Configuration:

```
PATH=$(VCPKG_ROOT)\installed\x64-windows\bin;%PATH%
```

4. Sample Code

Create main.cpp in your project and add this code:

```
#include <glad/glad.h>
#include <GLFW/glfw3.h>
#include <imgui.h>
#include <imgui_impl_glfw.h>
#include <imgui_impl_opengl3.h>
#include <boost/signals2.hpp>
#include <iostream>
#include <string>
#include <Windows.h>
```

```cpp
// Signal for window events
boost::signals2::signal<void(const std::string&)> onWindowEvent;

// Global variables for theming
float backgroundColor[3] = { 0.45f, 0.55f, 0.60f }; // RGB values for background color
bool isDarkTheme = true;

void printEvent(const std::string& msg) {
    std::cout << "Event: " << msg << std::endl;
}

// Windows Entry Point
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
{
    // Connect signal
    onWindowEvent.connect(&printEvent);

    // Initialize GLFW
    if (!glfwInit()) {
        std::cerr << "Failed to initialize GLFW" << std::endl;
        return -1;
    }

    // Configure GLFW
    glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
    glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
    glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);

    // Create window
    GLFWwindow* window = glfwCreateWindow(1280, 720, "ImGui + GLFW Example", nullptr, nullptr);
    if (!window) {
        std::cerr << "Failed to create GLFW window" << std::endl;
        glfwTerminate();
        return -1;
    }

    glfwMakeContextCurrent(window);
    glfwSwapInterval(1); // Enable vsync

    // Initialize GLAD
    if (!gladLoadGLLoader((GLADloadproc)glfwGetProcAddress)) {
        std::cerr << "Failed to initialize GLAD" << std::endl;
        return -1;
    }

    // Setup Dear ImGui
    IMGUI_CHECKVERSION();
    ImGui::CreateContext();
```

```
    ImGuiIO& io = ImGui::GetIO(); (void)io;
    io.ConfigFlags |= ImGuiConfigFlags_NavEnableKeyboard;

    ImGui::StyleColorsDark();

    // Setup Platform/Renderer backends
    ImGui_ImplGlfw_InitForOpenGL(window, true);
    ImGui_ImplOpenGL3_Init("#version 330");

    // Emit window creation event
    onWindowEvent("Window created successfully");

    // Main loop
    while (!glfwWindowShouldClose(window)) {
        glfwPollEvents();

        // Start the Dear ImGui frame
        ImGui_ImplOpenGL3_NewFrame();
        ImGui_ImplGlfw_NewFrame();
        ImGui::NewFrame();

        // Create ImGui window
        {
            ImGui::Begin("Example Window");

            static float value = 0.0f;
            if (ImGui::SliderFloat("Slider", &value, 0.0f, 1.0f)) {
                onWindowEvent("Slider value changed: " + std::to_string(value));
            }

            // Color Preview
            ImGui::ColorEdit3("Background Color", backgroundColor);

            // Enhanced button functionality
            if (ImGui::Button("Click Me!")) {
                // Toggle between dark and light theme
                if (isDarkTheme) {
                    ImGui::StyleColorsLight();
                    backgroundColor[0] = 0.9f; // Light gray background
                    backgroundColor[1] = 0.9f;
                    backgroundColor[2] = 0.9f;
                }
                else {
                    ImGui::StyleColorsDark();
                    backgroundColor[0] = 0.45f; // Original dark blue-gray
                    backgroundColor[1] = 0.55f;
                    backgroundColor[2] = 0.60f;
                }
                isDarkTheme = !isDarkTheme;
```

```
                onWindowEvent("Theme changed to: " + std::string(isDarkTheme ? "Dark" : "Light"))
            }

            ImGui::Text("Click the button to toggle between dark and light theme!");

            // Display current theme status
            ImGui::TextColored(
                isDarkTheme ? ImVec4(1.0f, 1.0f, 0.0f, 1.0f) : ImVec4(0.0f, 0.0f, 1.0f, 1.0f),
                "Current Theme: %s",
                isDarkTheme ? "Dark" : "Light"
            );

            // Add Demo Window for reference
            if (ImGui::Button("Toggle Demo Window")) {
                static bool showDemo = false;
                showDemo = !showDemo;
                if (showDemo)
                    ImGui::ShowDemoWindow(&showDemo);
            }

            ImGui::End();
        }

        // Rendering
        ImGui::Render();
        int display_w, display_h;
        glfwGetFramebufferSize(window, &display_w, &display_h);
        glViewport(0, 0, display_w, display_h);
        glClearColor(backgroundColor[0], backgroundColor[1], backgroundColor[2], 1.00f);
        glClear(GL_COLOR_BUFFER_BIT);
        ImGui_ImplOpenGL3_RenderDrawData(ImGui::GetDrawData());
        glfwSwapBuffers(window);
    }

    // Cleanup
    ImGui_ImplOpenGL3_Shutdown();
    ImGui_ImplGlfw_Shutdown();
    ImGui::DestroyContext();
    glfwDestroyWindow(window);
    glfwTerminate();

    return 0;
}
```

5. Build and Run

1. Make sure configuration is set to "Debug" and platform to "x64"

2. Build Solution (F7)
3. Run (F5)

Expected result:

- Window appears with ImGui interface
- Interactive UI elements working
- Theme toggle functioning
- Background color changeable
- Demo window available

6. Troubleshooting

Common Build Errors

LNK1168: Cannot open exe for writing

```
Error LNK1168: cannot open ... .exe for writing
```

Fix:

1. Close any running instances of your application
2. Close Visual Studio
3. Check Task Manager and end related processes
4. Delete the executable manually if needed
5. Clean and Rebuild solution

Missing WinMain Entry Point

```
Error LNK2019: unresolved external symbol WinMain referenced
```

Fix:

1. Verify SubSystem is set to "Windows"
2. Ensure WinMain function signature is correct
3. Include <Windows.h>

Library Not Found

```
Error LNK2019: unresolved external symbol ... referenced
```

Fix:

1. Check library paths
2. Verify debug/release library names
3. Confirm platform (x64) consistency

Runtime DLL Issues

```
The application was unable to start correctly (0xc000007b)
```

Fix:

1. Verify PATH environment variable
2. Check debug/release DLL matching
3. Rebuild solution

Tips and Reminders

- Always build in x64 configuration
- Match debug/release libraries correctly
- Restart Visual Studio after environment changes
- Check Output window for detailed error messages

7. Features Overview

UI Elements

1. Main window with:
    - Slider control
    - Theme toggle button
    - Color picker
    - Demo window toggle
    - Status display

Functionality

1. Theme Switching:
    - Dark/Light theme toggle
    - Background color changes
    - Dynamic UI updates
2. Event System:
    - Slider value changes
    - Theme changes
    - Window creation events
3. Graphics:
    - OpenGL context
    - ImGui rendering
    - Vsync support

Additional Notes

- Uses Boost.Signals2 for event handling
- GLFW for window management
- GLAD for OpenGL loading
- Full ImGui demo window available for reference