Using the Leap Motion on Mac OS with Lazarus

Michaël Van Canneyt

December 27, 2013

Abstract

Lazarus is a cross platform IDE supporting (among others) Windows, Linux and Mac OS. The Leap Motion works on all of these OS-es. While not initially developed on the Mac, the intention was that the lazarus components for the Leap Motion were usable on all platforms that the Leap supports, and this includes Mac OS.

1 Introduction

The Leap Motion works on all major platforms: Windows, Mac OS and Linux. So does Lazarus. The workings of the Leap Motion controller on Linux and Windows were easily verified, as the component was developed on that platform. To check whether the component also works on Mac OS X, Lazarus was installed on a Mac (Macbook Pro, running OS X Lion 10.7.5) and compilation of the leap component and one of the demo applications is tested.

2 Installation of Lazarus

While cross-compilation is commonplace these days, there is nothing like native development. So, installing Lazarus on the Mac is the firststep. This can be easily done: From the Lazarus website, 3 disk image files need to be downloaded:

fpc-2.6.2.intel-macosx.dmg The Free Pascal compiler. The IDE calls the free pascal compiler when it needs to compile code.

fpcsrc-2.6.2-i386-macosx.dmg The Free Pascal Sources. The IDE needs this to provide code insight.

lazarus-1.0.14-20131116-i386-macosx.dmg The actual Lazarus IDE.

The version numbers may change as FPC and Lazarus evolve.

Each of these disk images contain an .pkg file with the same name, which can be installed by ctrl-click-ing it and selecting 'Open' (simply double-clicking will not necessarily work) from the menu that pops up. This will start an installation of the package. It is best to install the packages in the order listed here.

If the packages are installed in the default order and on default locations, a 'Lazarus' application will appear in the list of applications.

When first started, the Lazarus IDE will prompt for the location of the fpc compiler and the sources. For a default installation, this is `/usr/local/bin/fpc` and `/usr/local/share/fpcsrc`, respectively.

Once installed and started, the IDE is ready for use.

3 Verifying the LazLeap package

To check whether everything works as it should 3 packages must be installed:

laz_synapse The low-level TCP/IP socket support on which the websocket support builds.

bauglirwebsocket The websocket implementation needed to query the leap motion service.

lazleap The actual lazarus component.

These three packages can be downloaded and copied anywhere: After opening and compiling them, the lazarus IDE will remember where they are and will correctly resolve all references to them.

All 3 packages compiled out-of the box, with the exception of synapse: there the synaser unit gave an error. Since it is not needed for TC/IP support, it can simply be removed from the package.

4 Compiling the fingers and tap demo

After the packages have been compiled, the demo programs are next. The demo programs do not really rely on platform-specific things, and indeed, they compile without a glitch, the running tapdemo is shown e.g. in figure figure 1 on page 3.

The tap demo has been improved with a new 'magnetism' setting. The tap movement is very sensitive: while tapping, the tip of the finger (used to select a button) moves. The effect may be that the actual button that is under the finger cursor on the moment the tap gesture is received, is not the button for which the tap was intended: The downward movement of the finger during the tap may switch the focus to the button below the intended.

To prevent this from happening, a kind of magnetism is introduced: magnetism 'shrinks' the surface of a control, making it more difficult to be selected. It is in fact the number of pixels that the cursor must be inside the actual border, before the control is selected as the new focused control. The focus sticks to the previously selected control, unless the finger cursor is really centered on the new control, hence the name 'magnetism'.

This simple trick makes the tap demo a lot more easy to handle, as some experimenting will confirm.

5 Conclusion

In the case of the Leap Motion, Lazarus truly lives up to it's motto: Write once, compile anywhere. After verifying that the leap motion controller works on all platforms, it is time to start work on designing some components that help to drive the user interface with a Leap.

Figure 1: The Tap demo program in action on a Mac