VRAC|HCI

HCI 557

Augmented Reality

XCode Introduction

Rafael Radkowski
rafael@iastate.edu

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Content

- Creating Command Line Code with XCode

Prerequisites

This introduction requires the following software packages

OpenSceneGraph 3.x x86

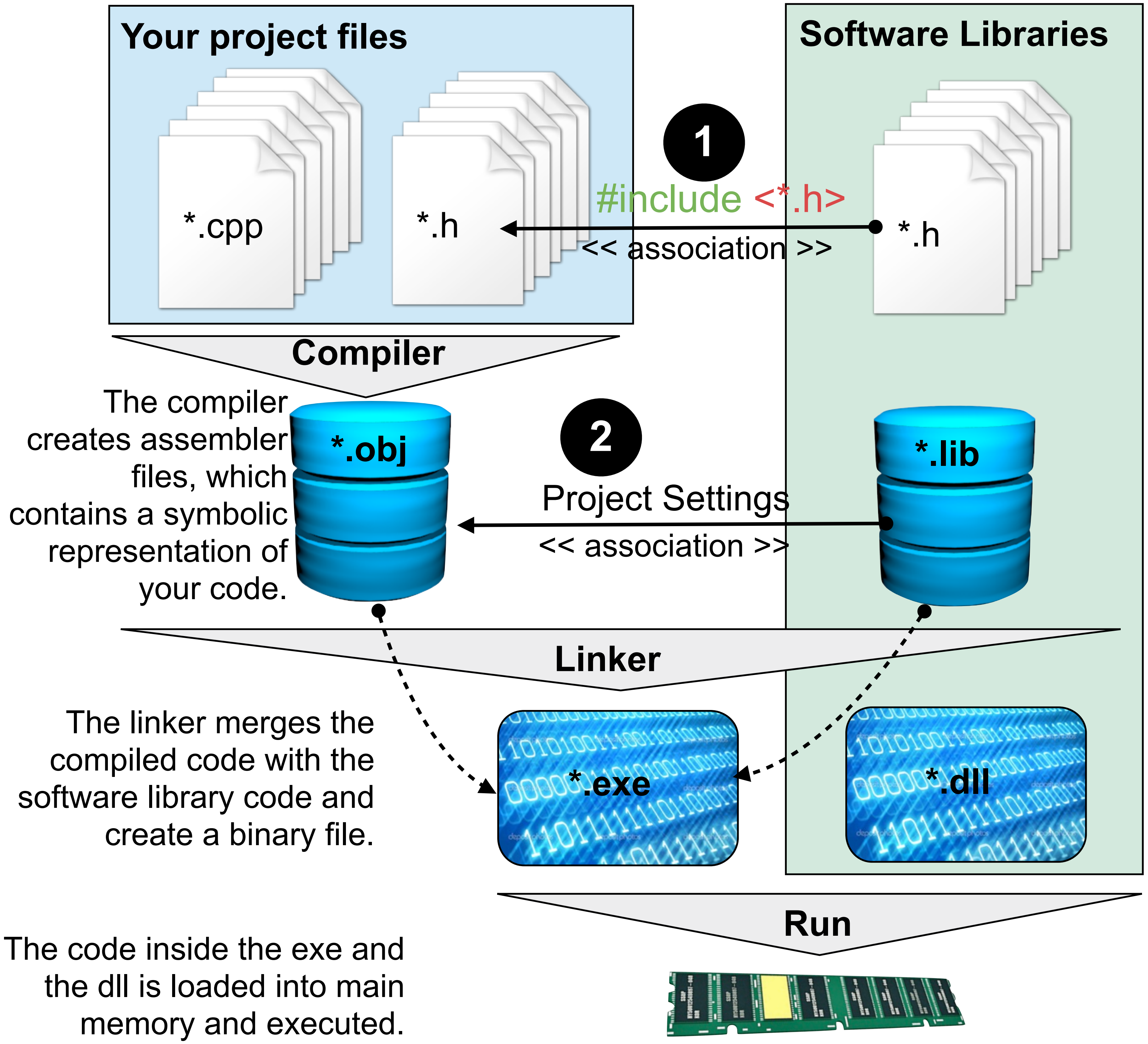
CMake

XCode 5.0,x

VRAC|HCI

Creating Command Line Code with XCode

C/C++ Compiler


Your project files
*.cpp
*.h
1
#include <*.h>
<< association >>
Software Libraries
*.h
Compiler
The compiler creates assembler files, which contains a symbolic representation of your code.
*.obj
2
Project Settings
<< association >>
*.lib
Linker
The linker merges the compiled code with the software library code and create a binary file.
*.exe
*.dll
Run
The code inside the exe and the dll is loaded into main memory and executed.

Every software project consists of two set of code: your own code and code from software libraries.

Your project code incorporates a set of cpp-files and header files.

The software library incorporates a set of header files, a library (multiple library files), and a binary file (dll), which contains the executables.

C/C++ code is generated in two steps.

First, a compiler compiles your project files and generates object files (obj). The contain assembler code. During this step, your code needs to know all the libraries and the provided function. This association is established using the #include command in your header files. The obj files contain a symbolic link to each library function.

Secondly, the Linker merges the generated obj files to one binary file. During this process, the Linker searches the lib files for the binary code, related to the symbolic links.

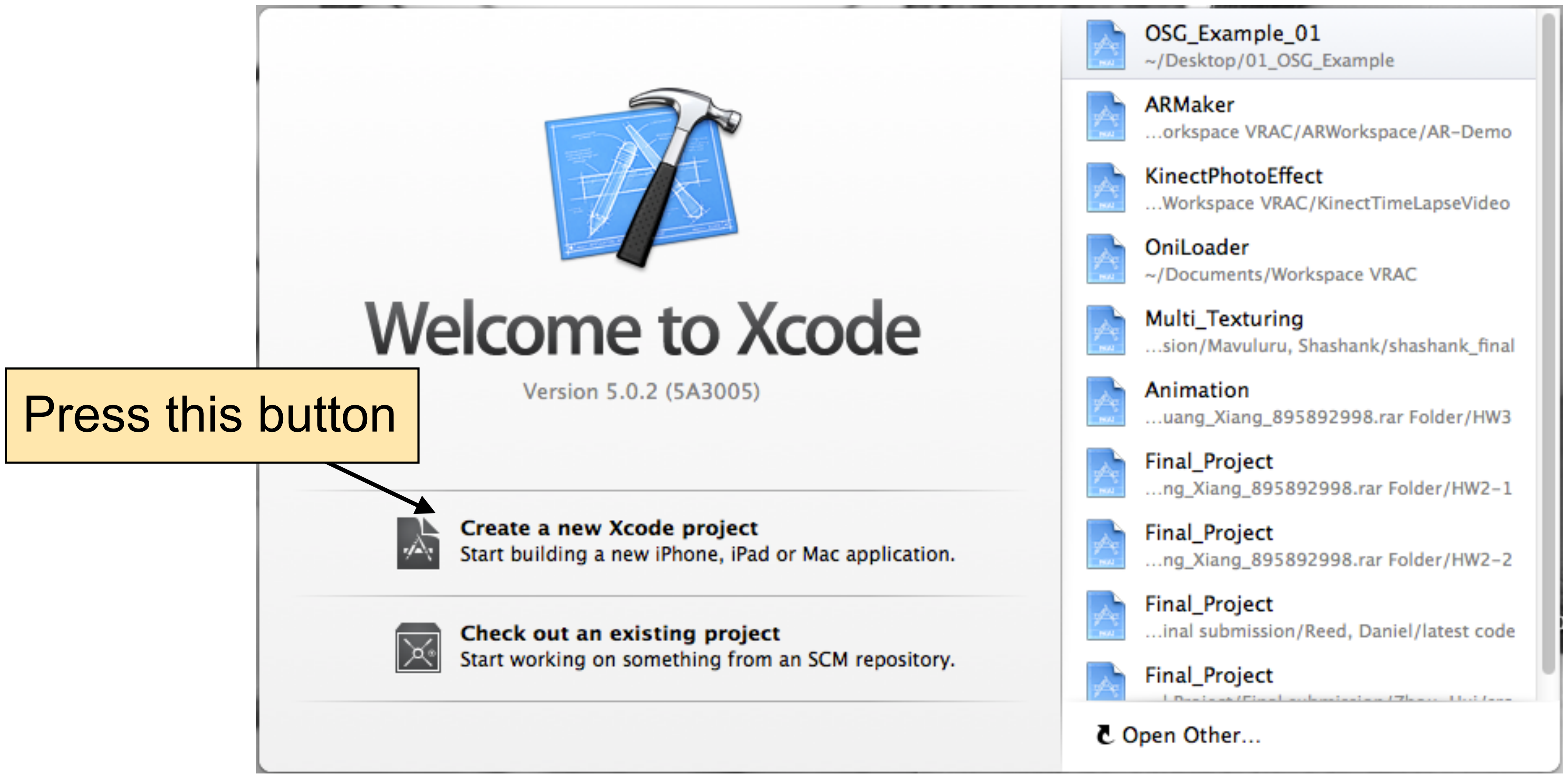
The result is an executable file containing machine code.

During program start, the machine code from the exe and the dll are loaded into computer's main memory. Thus, the program runs.

XCode Command Line Tool

This introduction explains the steps necessary to manually create an XCode command line tool. A command line tool is a Unix / Mac OS X tool that starts from a terminal application.

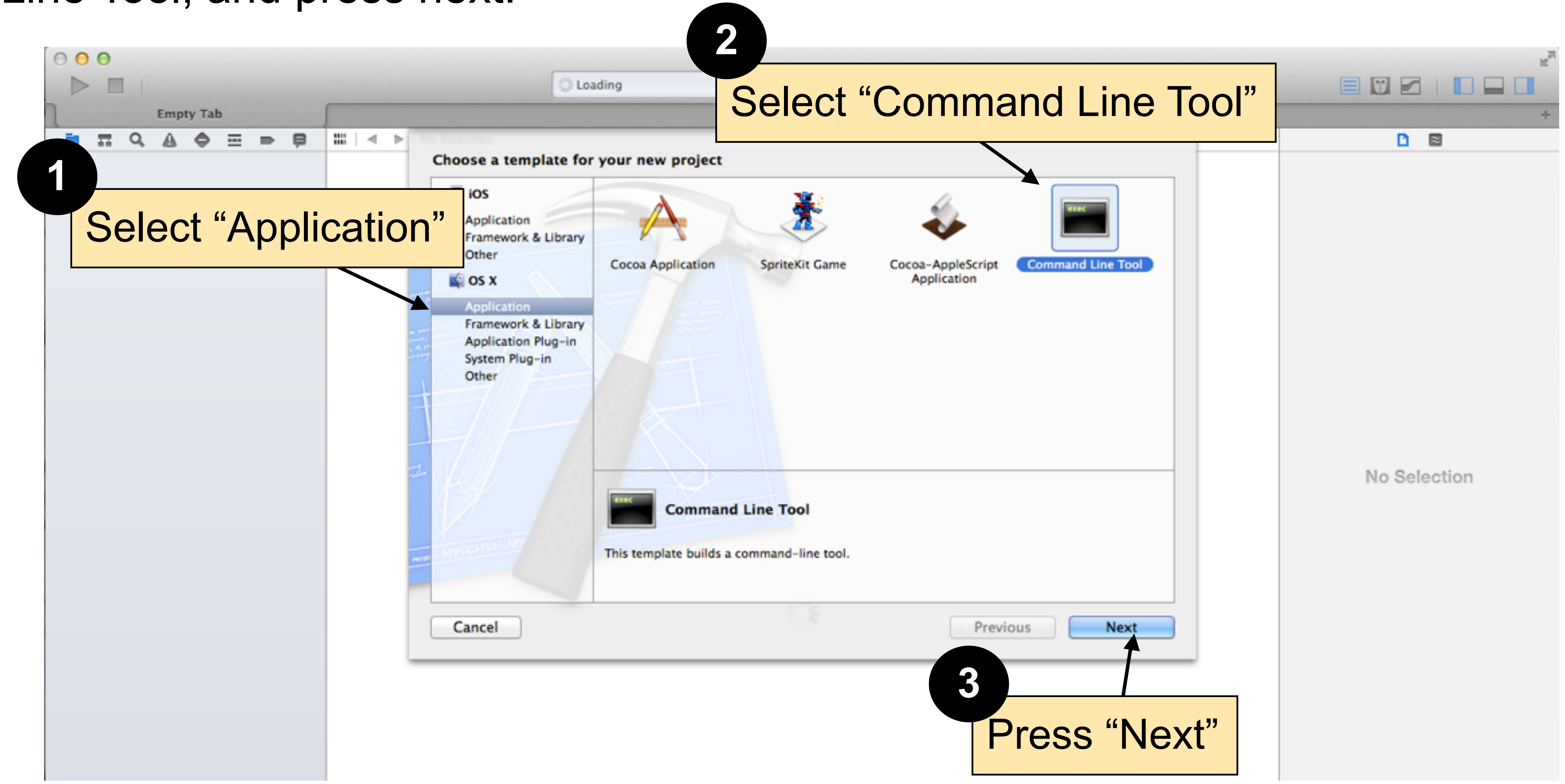
Step 1: open XCode and press “Create a new XCode project”


Press this button
Welcome to Xcode
Version 5.0.2 (5A3005)
Create a new Xcode project
Start building a new iPhone, iPad or Mac application.
Check out an existing project
Start working on something from an SCM repository.
OSG_Example_01
~/Desktop/01_OSG_Example
ARMaker
...orkspace VRAC/ARWorkspace/AR-Demo
KinectPhotoEffect
...Workspace VRAC/KinectTimeLapseVideo
OniLoader
~/Documents/Workspace VRAC
Multi_Texturing
...sion/Mavuluru, Shashank/shashank_final
Animation
...uang_Xiang_895892998.rar Folder/HW3
Final_Project
...ng_Xiang_895892998.rar Folder/HW2-1
Final_Project
...ng_Xiang_895892998.rar Folder/HW2-2
Final_Project
...inal submission/Reed, Daniel/latest code
Final_Project
Open Other...

XCode welcome window

XCode Command Line Tool

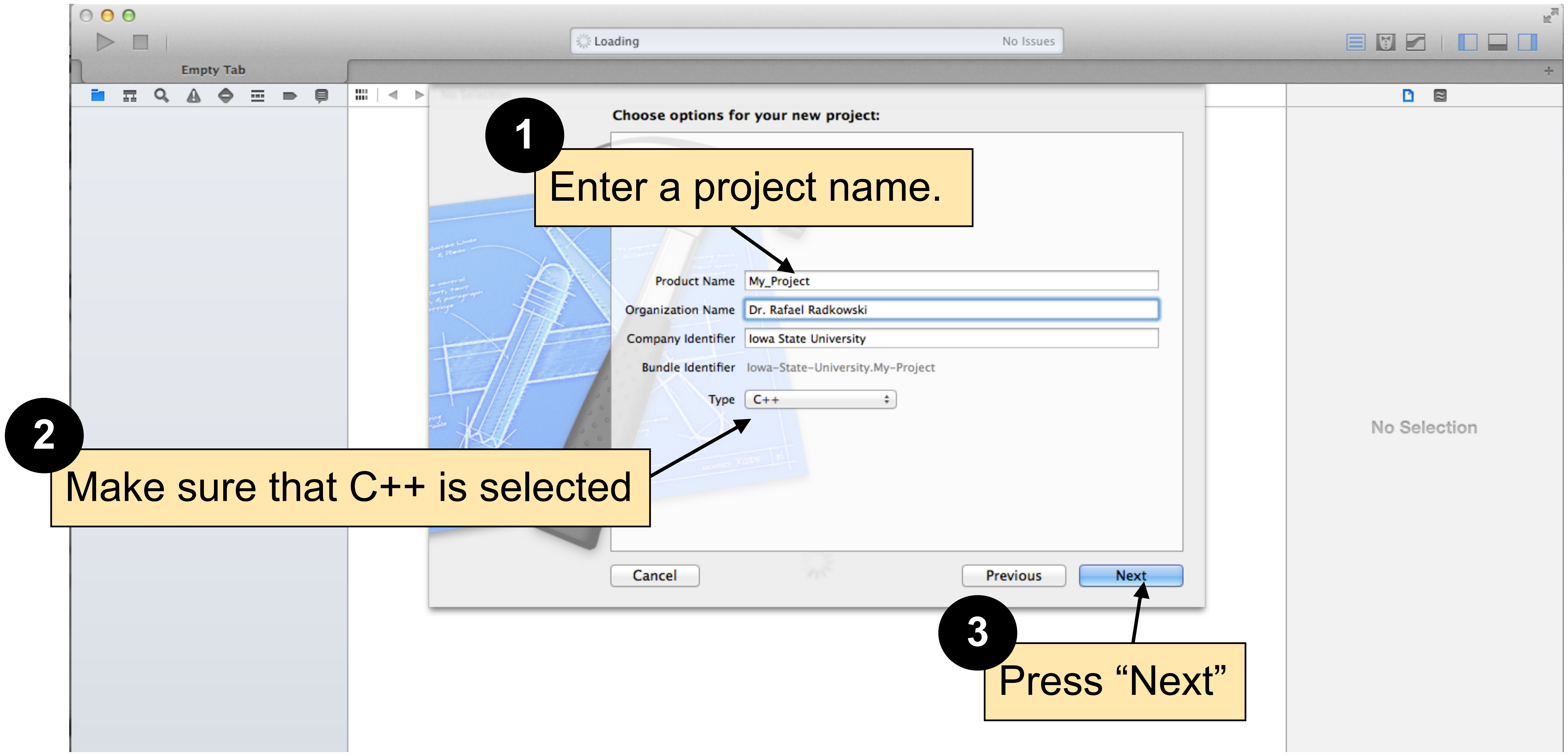
The following window will appear. Switch to OS X - Application, select Command Line Tool, and press next.



XCode main window

XCode Command Line Tool

Now enter a project name. You can select a project name on your own.



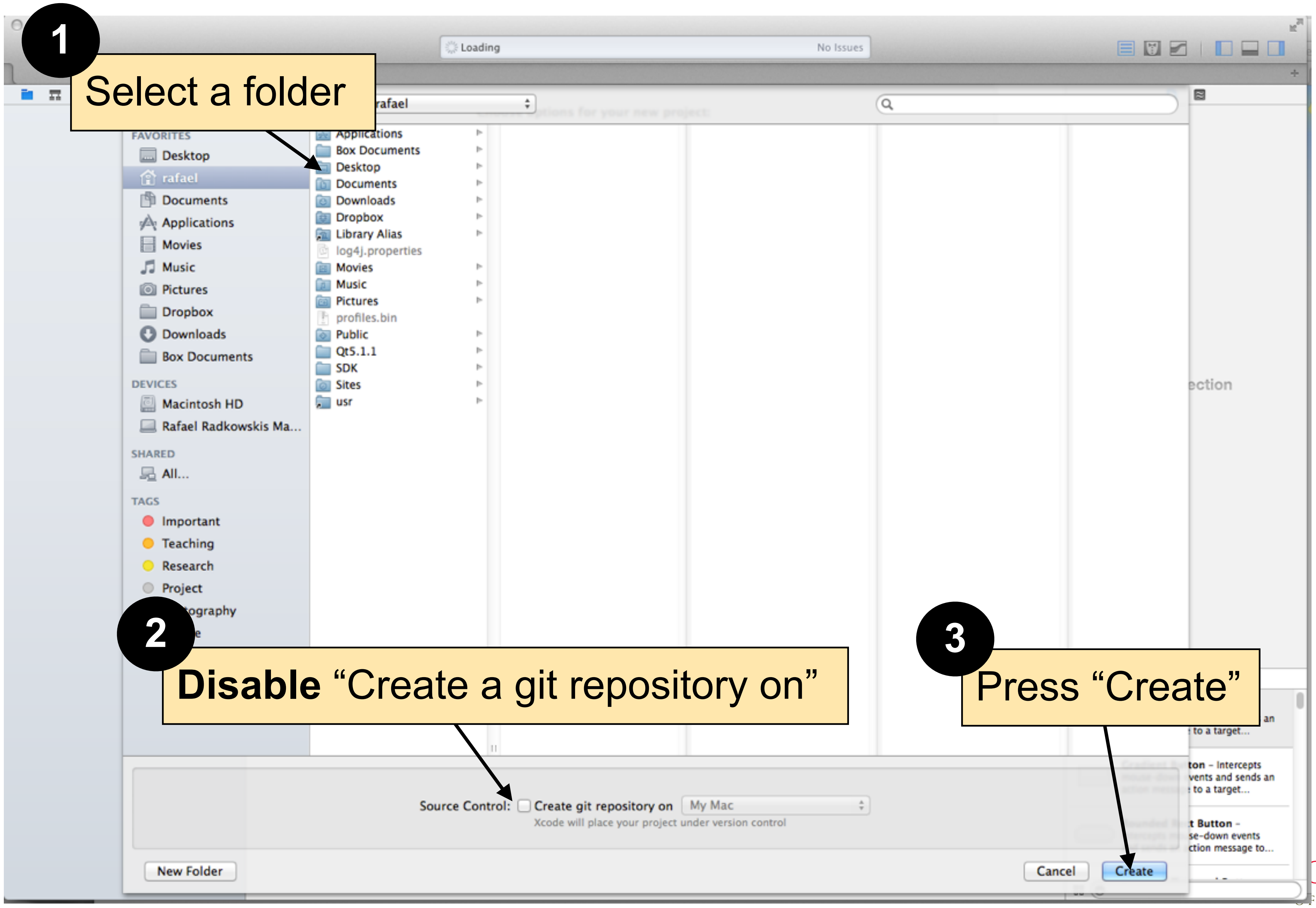
Choose options for your new project:

Product Name: My_Project
Organization Name: Dr. Rafael Radkowski
Company Identifier: Iowa State University
Bundle Identifier: Iowa-State-University.My-Project
Type: C++

XCode main window

XCode Command Line Tool

Select a location on your hard disc. XCode will save the project and all files at this location. You can decide on your own where to save the project files.

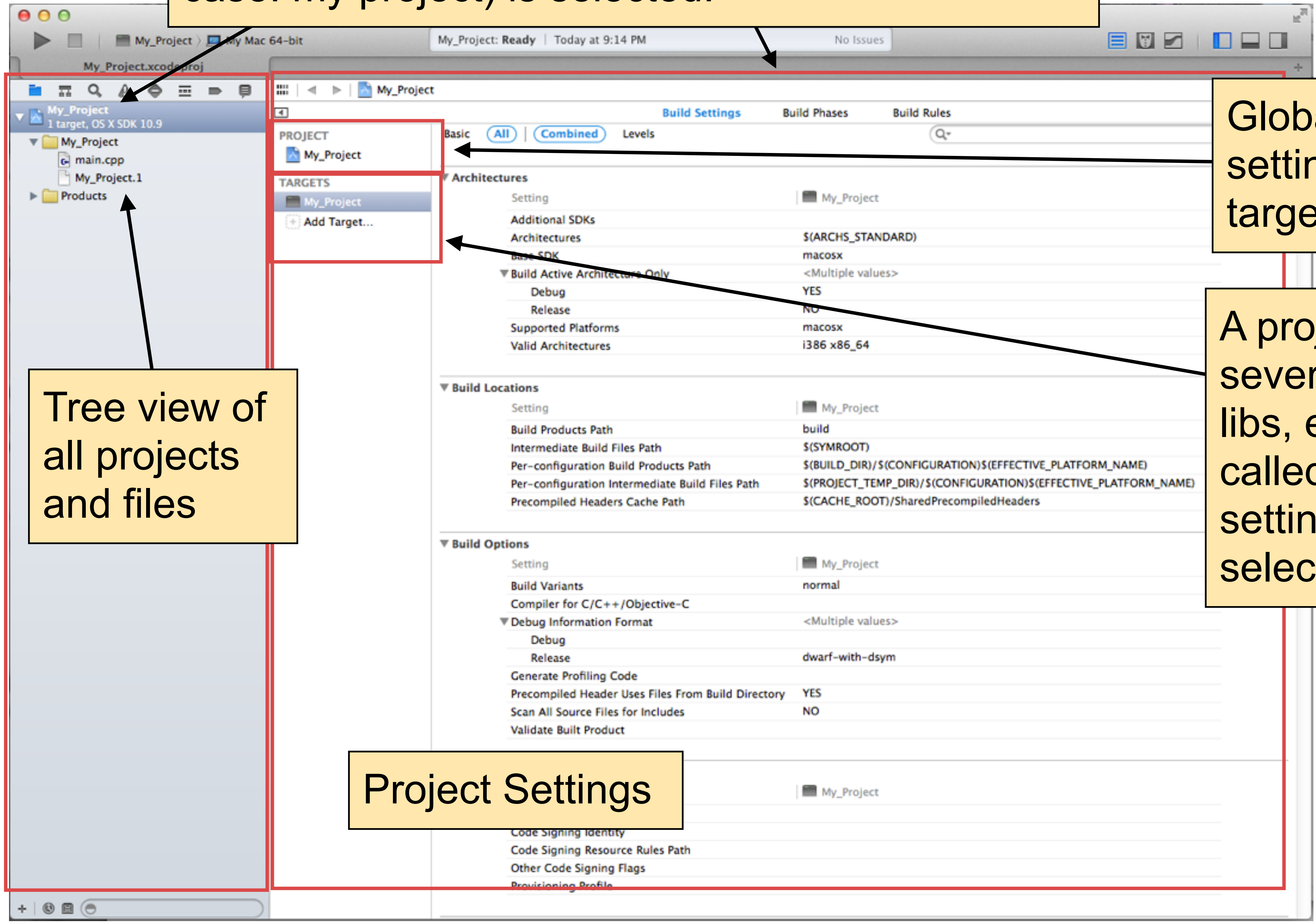


Source Control: ☐ Create git repository on My Mac
Xcode will place your project under version control

XCode Command Line Tool

The XCode main window will appear. The project is ready for code development.

This view appear if the main project (in this case: my project) is selected.



▼ Architectures

Setting	My_Project
Additional SDKs	
Architectures	$(ARCHS_STANDARD)
Base SDK	macosx
▼Build Active Architecture Only	<Multiple values>
Debug	YES
Release	NO
Supported Platforms	macosx
Valid Architectures	i386 x86_64

▼ Build Locations

Setting	My_Project
Build Products Path	build
Intermediate Build Files Path	$(SYMROOT)
Per-configuration Build Products Path	$(BUILD_DIR)/$(CONFIGURATION)$(EFFECTIVE_PLATFORM_NAME)
Per-configuration Intermediate Build Files Path	$(PROJECT_TEMP_DIR)/$(CONFIGURATION)$(EFFECTIVE_PLATFORM_NAME)
Precompiled Headers Cache Path	$(CACHE_ROOT)/SharedPrecompiledHeaders

▼ Build Options

Setting	My_Project
Build Variants	normal
Compiler for C/C++/Objective-C	
▼Debug Information Format	<Multiple values>
Debug	
Release	dwarf-with-dsym
Generate Profiling Code	
Precompiled Header Uses Files From Build Directory	YES
Scan All Source Files for Includes	NO
Validate Built Product	

	My_Project
Code Signing Identity	
Code Signing Resource Rules Path	
Other Code Signing Flags	
Provisioning Profile	

Global project settings: valid for all targets.

A project can contain several applications, libs, etc. They are called *targets*. Target settings can be set by selecting a target.

Tree view of all projects and files

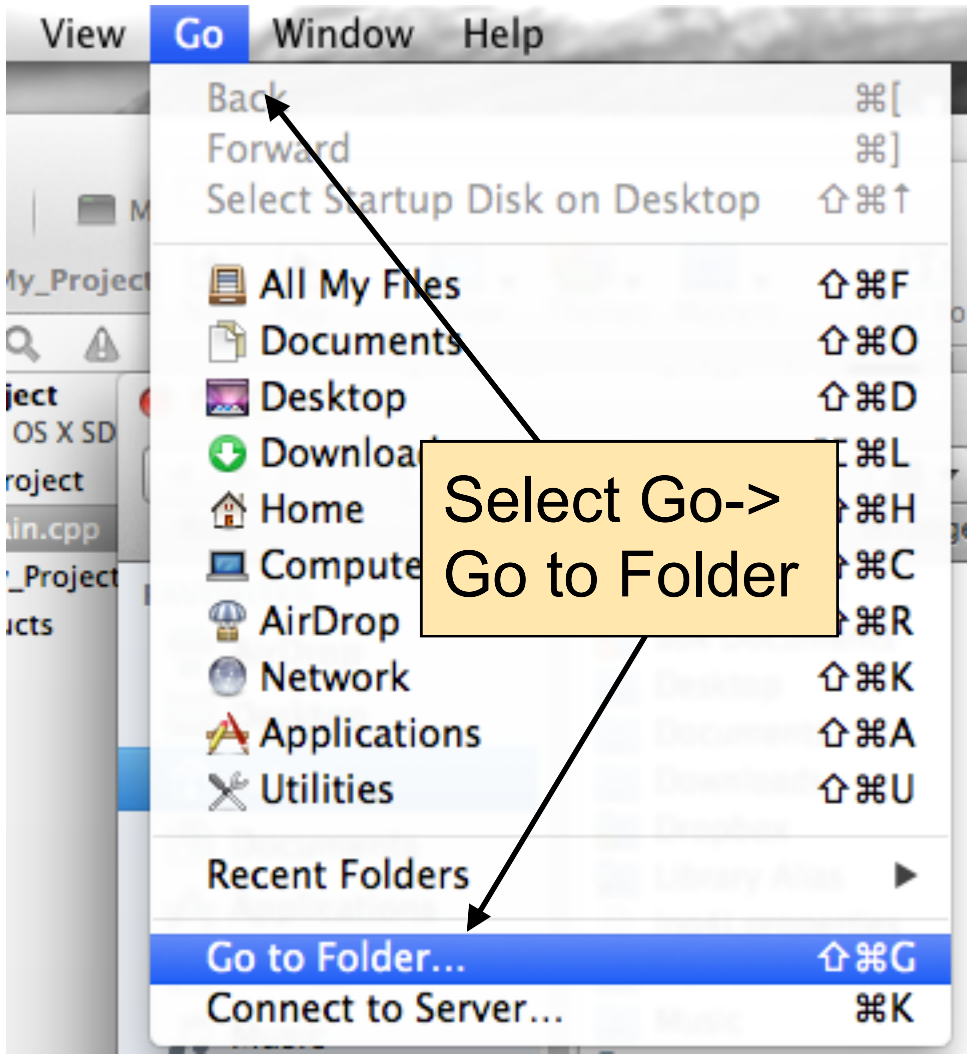
Project Settings

Header and Library Paths

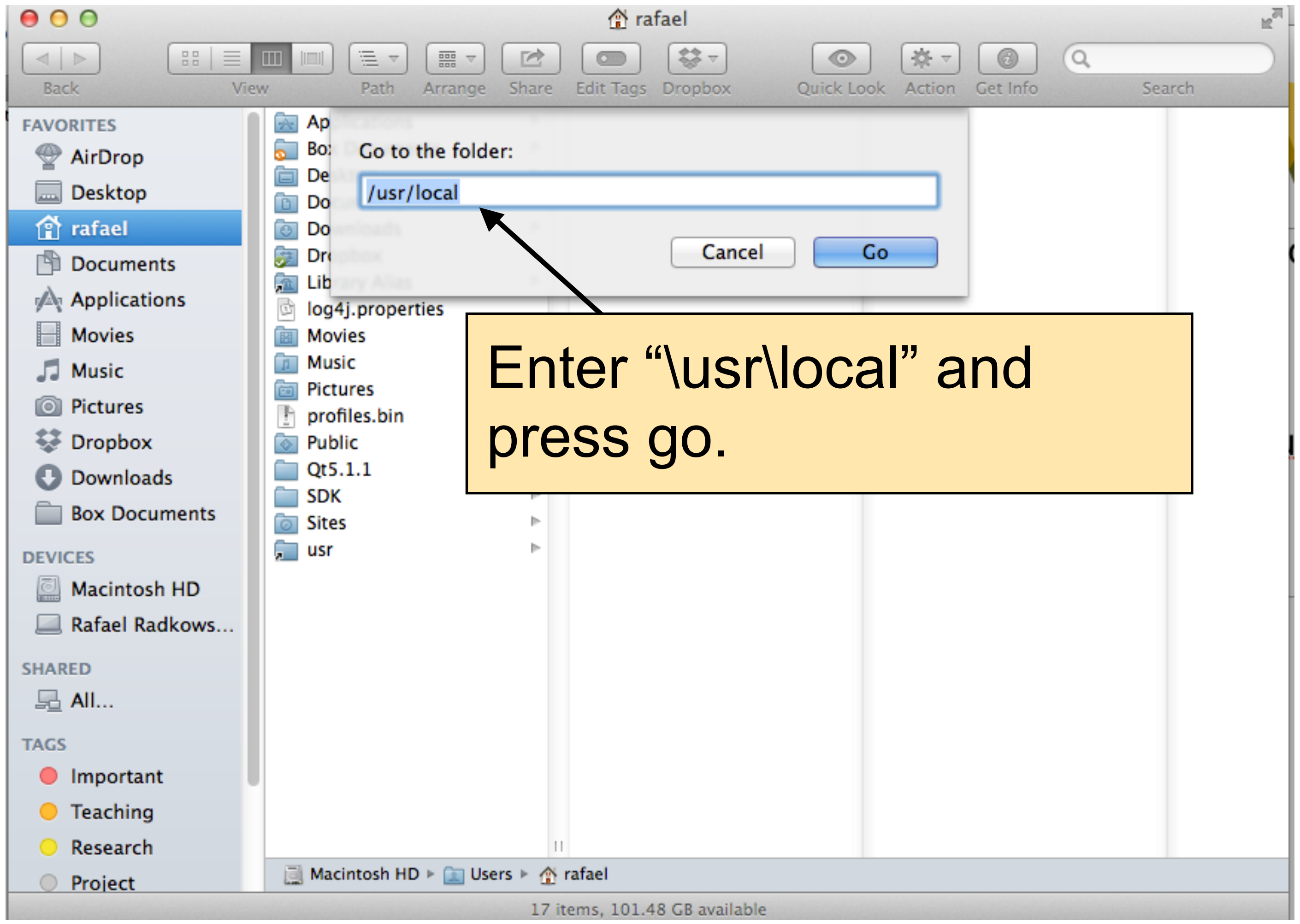
XCode need to know where to find the header files of a toolkit like OpenSceneGraph as well as where to find the library files (.dylib / .so / .a).

First, locate your header and library files.

- If correct installed, they should be in \usr\local\lib and \usr\local\include



Select Go-> Go to Folder

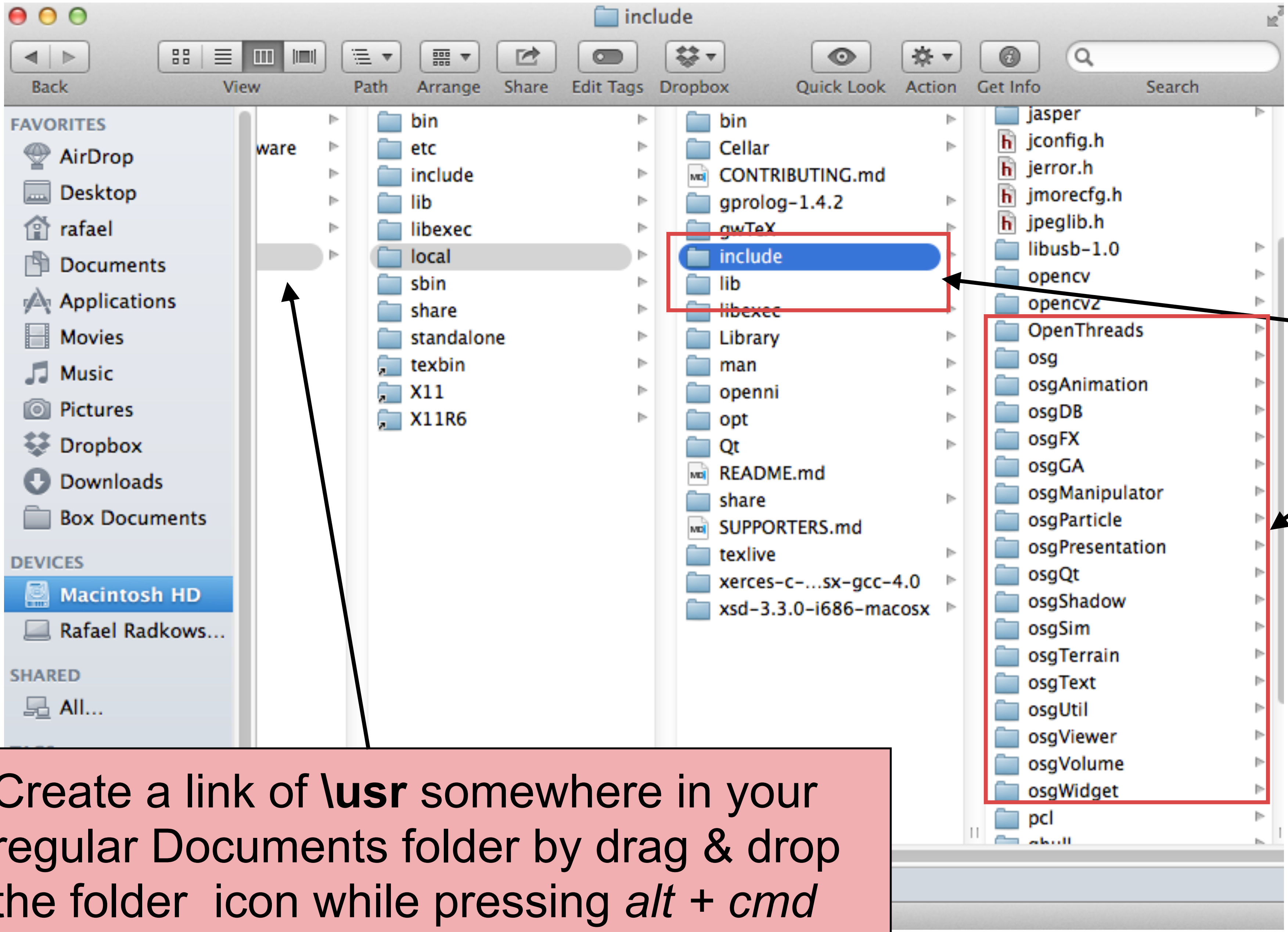


Enter "\usr\local" and press go.

Header and Library Paths

A Finder window will appear that shows the selected folder.

If not, OSG is installed at a different location. That's no problem, but I recommend to install it in this folder (if the install function is called, it installs itself in this folder).

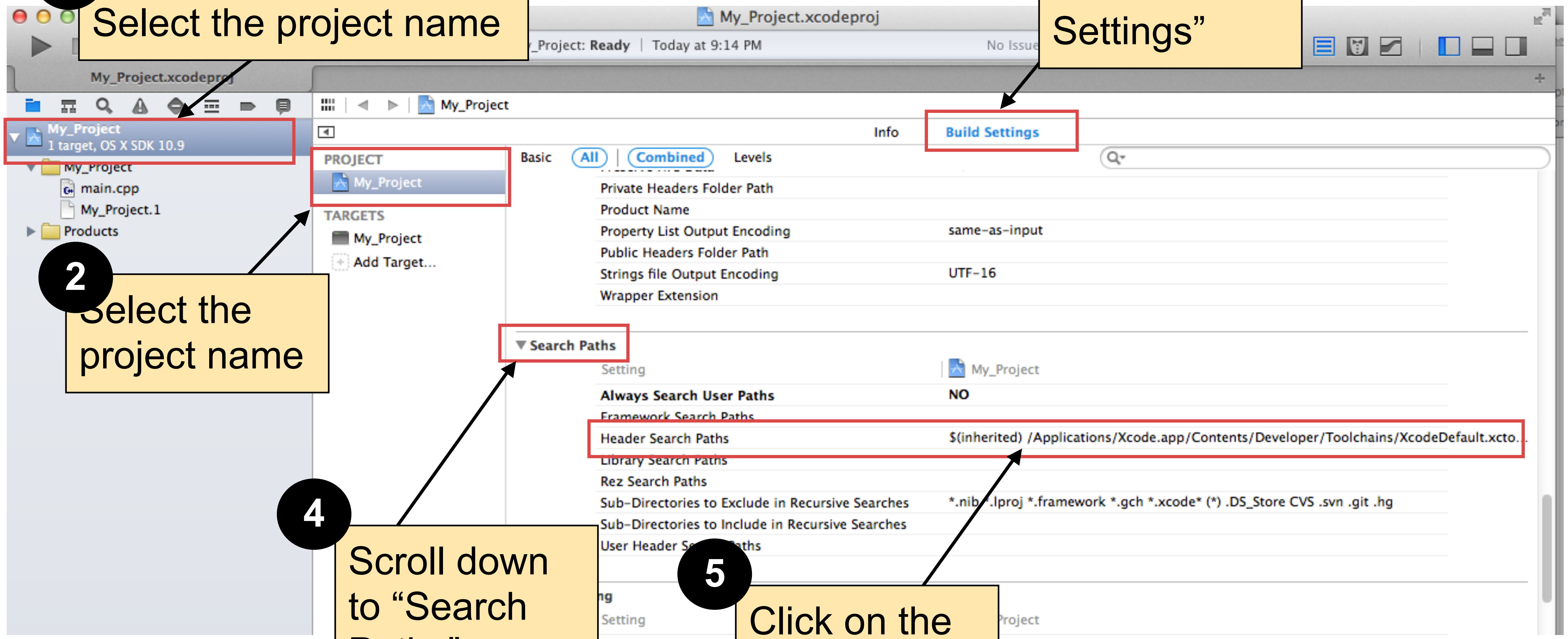


\usr\local\lib and \usr\local \include should contain the OSG files.

Create a link of **\usr** somewhere in your regular Documents folder by drag & drop the folder icon while pressing *alt* + *cmd*

XCode: add header search path

XCode must know where to find the header files.



Setting	My_Project
Private Headers Folder Path	
Product Name	
Property List Output Encoding	same-as-input
Public Headers Folder Path	
Strings file Output Encoding	UTF-16
Wrapper Extension	

Search Paths

Setting	My_Project
Always Search User Paths	**NO**
Framework Search Paths	
Header Search Paths	$(inherited) /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xcto...
Library Search Paths	
Rez Search Paths	
Sub-Directories to Exclude in Recursive Searches	*.nib *.lproj *.framework *.gch *.xcode* (*) .DS_Store CVS .svn .git .hg
Sub-Directories to Include in Recursive Searches	
User Header Search Paths	

1 Select the project name

2 Select the project name

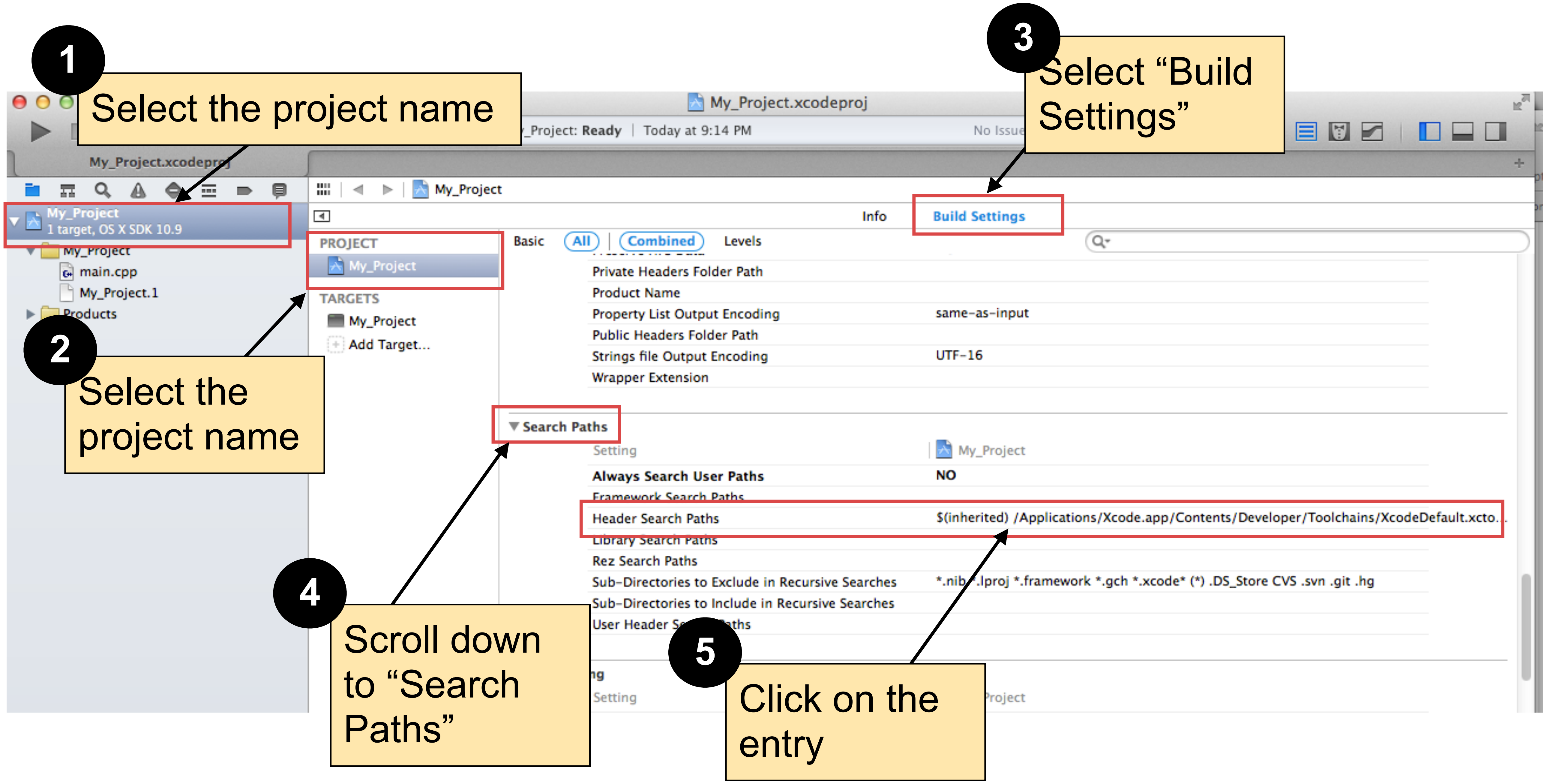
3 Select "Build Settings"

4 Scroll down to "Search Paths"

5 Click on the entry

XCode: add header search path

XCode must know where to find the header files.

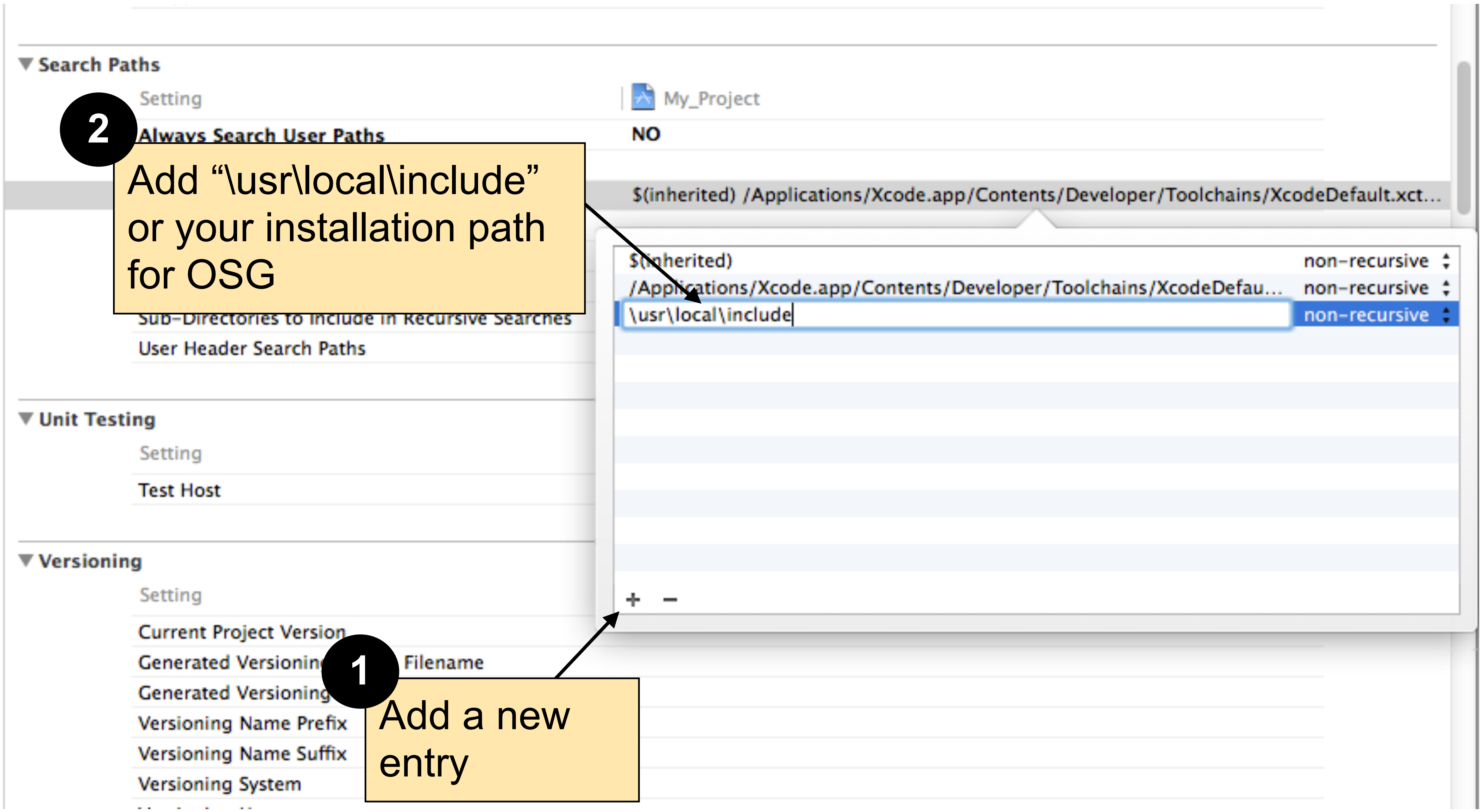


Setting	My_Project
Private Headers Folder Path	
Product Name	
Property List Output Encoding	same-as-input
Public Headers Folder Path	
Strings file Output Encoding	UTF-16
Wrapper Extension	

Setting	My_Project
Always Search User Paths	**NO**
Framework Search Paths	
Header Search Paths	$(inherited) /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xcto...
Library Search Paths	
Rez Search Paths	
Sub-Directories to Exclude in Recursive Searches	*.nib *.lproj *.framework *.gch *.xcode* (*) .DS_Store CVS .svn .git .hg
Sub-Directories to Include in Recursive Searches	
User Header Search Paths	

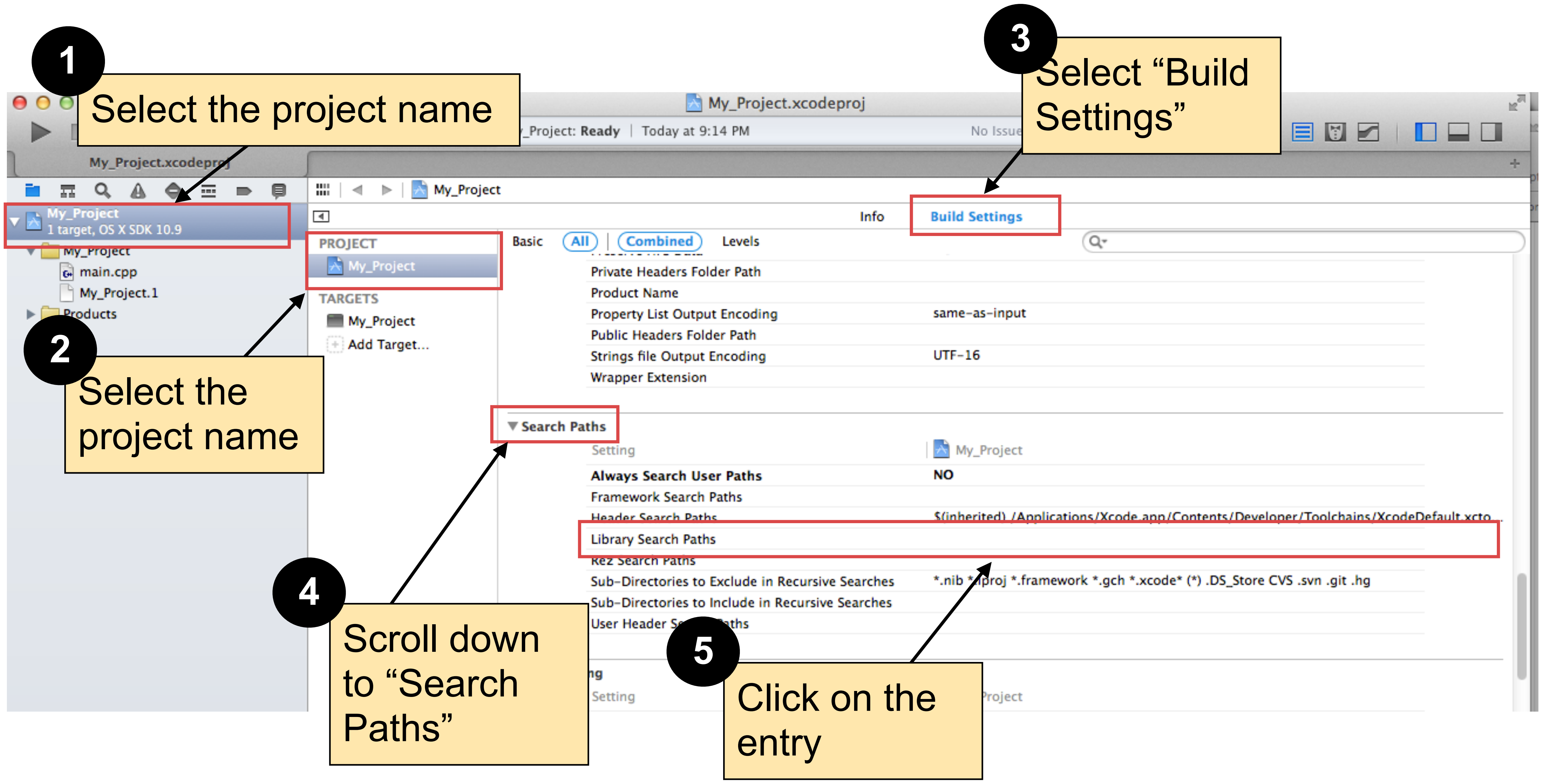
XCode: add header search path

XCode must know where to find the header files.



XCode: add library search path

XCode must also know where to find the library files. They should be at \usr\local \lib. But they can also be at every path you picked during installation.

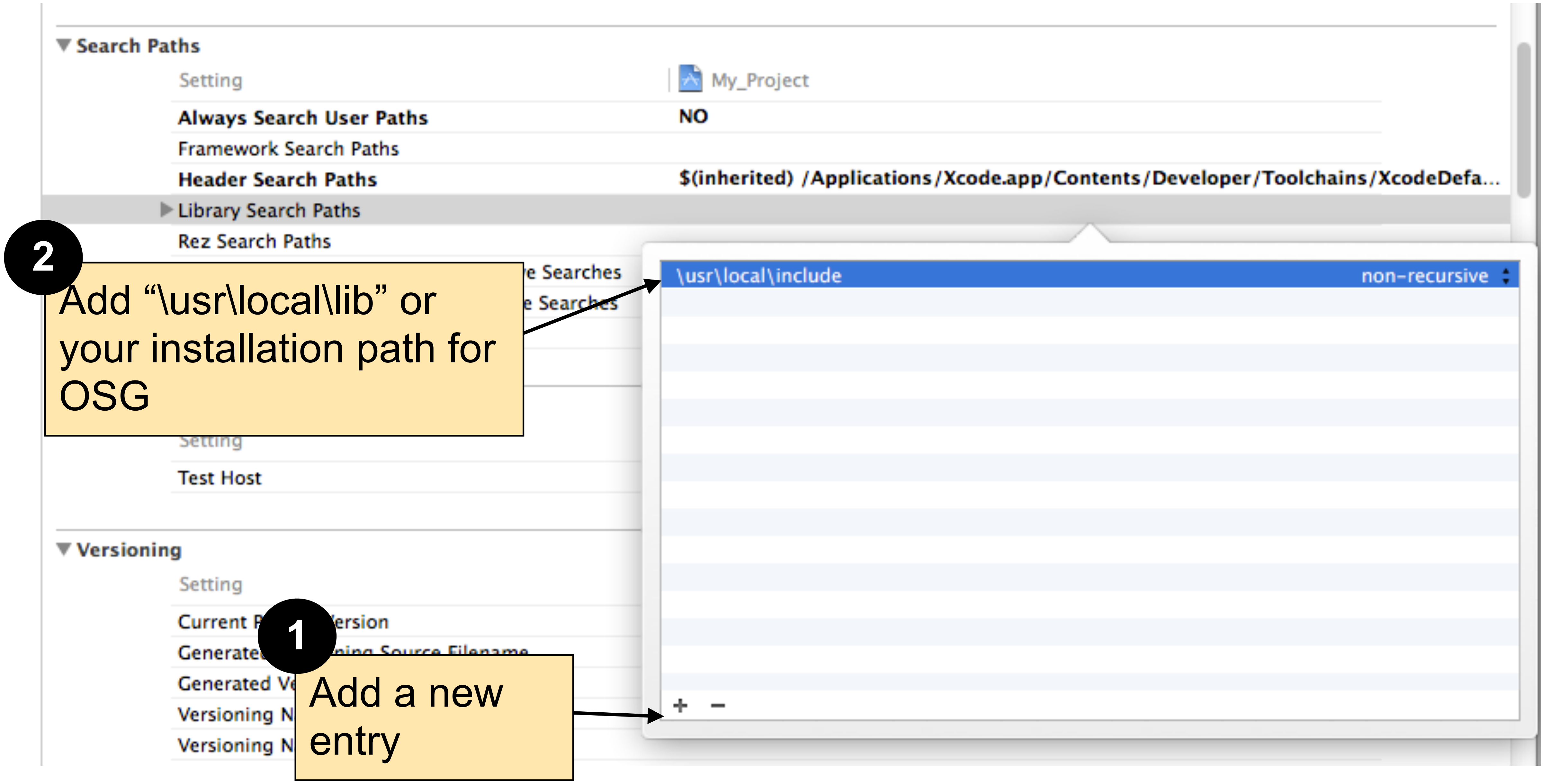


Setting	My_Project
Private Headers Folder Path	
Product Name	
Property List Output Encoding	same-as-input
Public Headers Folder Path	
Strings file Output Encoding	UTF-16
Wrapper Extension	

Setting	My_Project
Always Search User Paths	NO
Framework Search Paths	
Header Search Paths	$(inherited) /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xcto...
Library Search Paths	
Rez Search Paths	
Sub-Directories to Exclude in Recursive Searches	*.nib *.lproj *.framework *.gch *.xcode* (*) .DS_Store CVS .svn .git .hg
Sub-Directories to Include in Recursive Searches	
User Header Search Paths	

XCode: add library search path

XCode must also know where to find the library files. They should be at \usr\local \lib. But they can also be at every path you picked during installation.



Search Paths

Setting	My_Project
Always Search User Paths	NO
Framework Search Paths	
Header Search Paths	$(inherited) /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefa...
Library Search Paths	
Rez Search Paths	

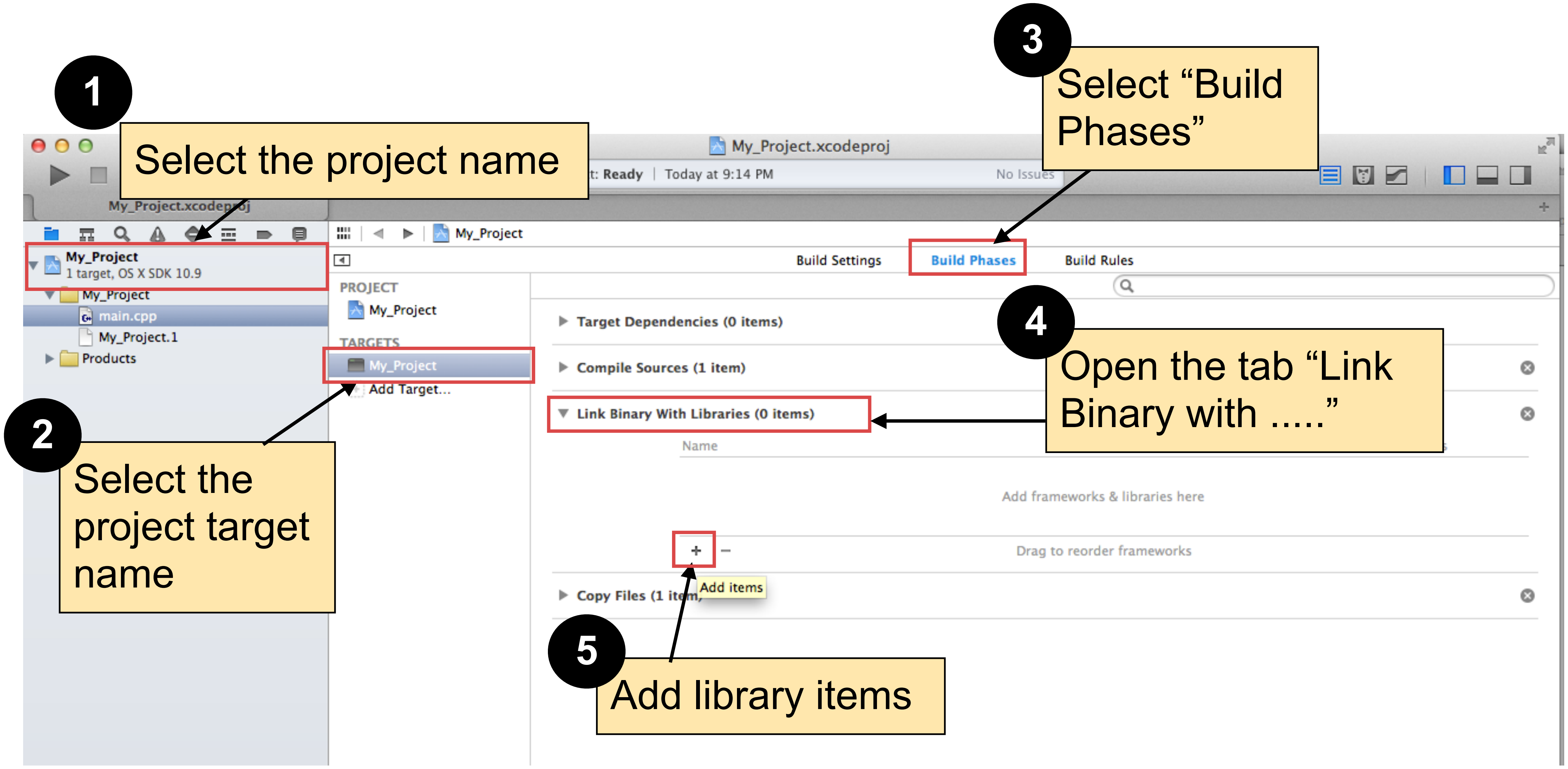
\usr\local\include non-recursive

Test Host

Versioning

XCode: add library files

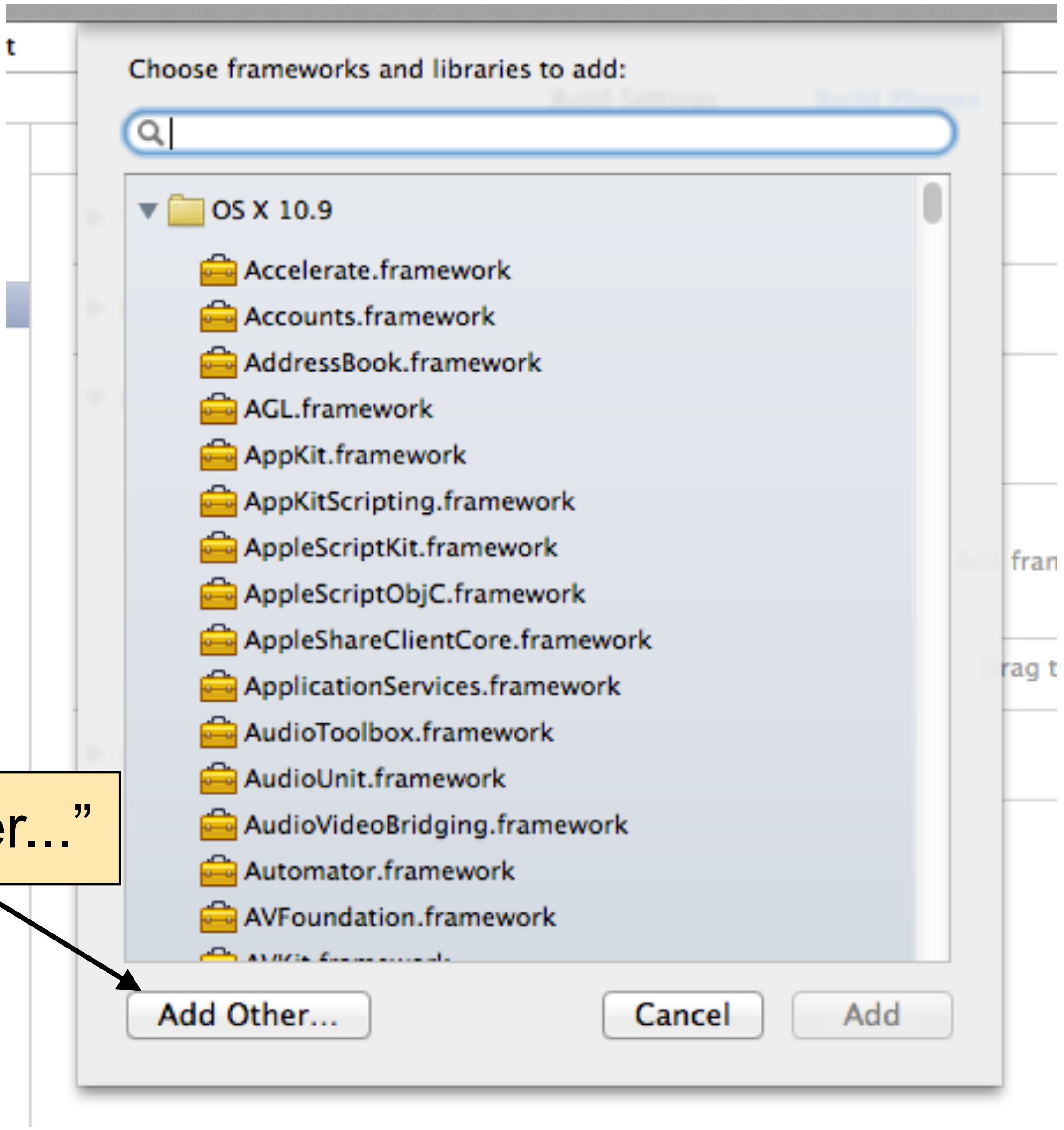
Next, all the library files that you want to use in your project must be added to this project. This is a build setting that belong to a particular target.



XCode: add library files

Next, all the library files that you want to use in your project must be added to this project. This is a build setting that belong to a particular target.

A window pops up that shows all available Mac OS X system libraries.

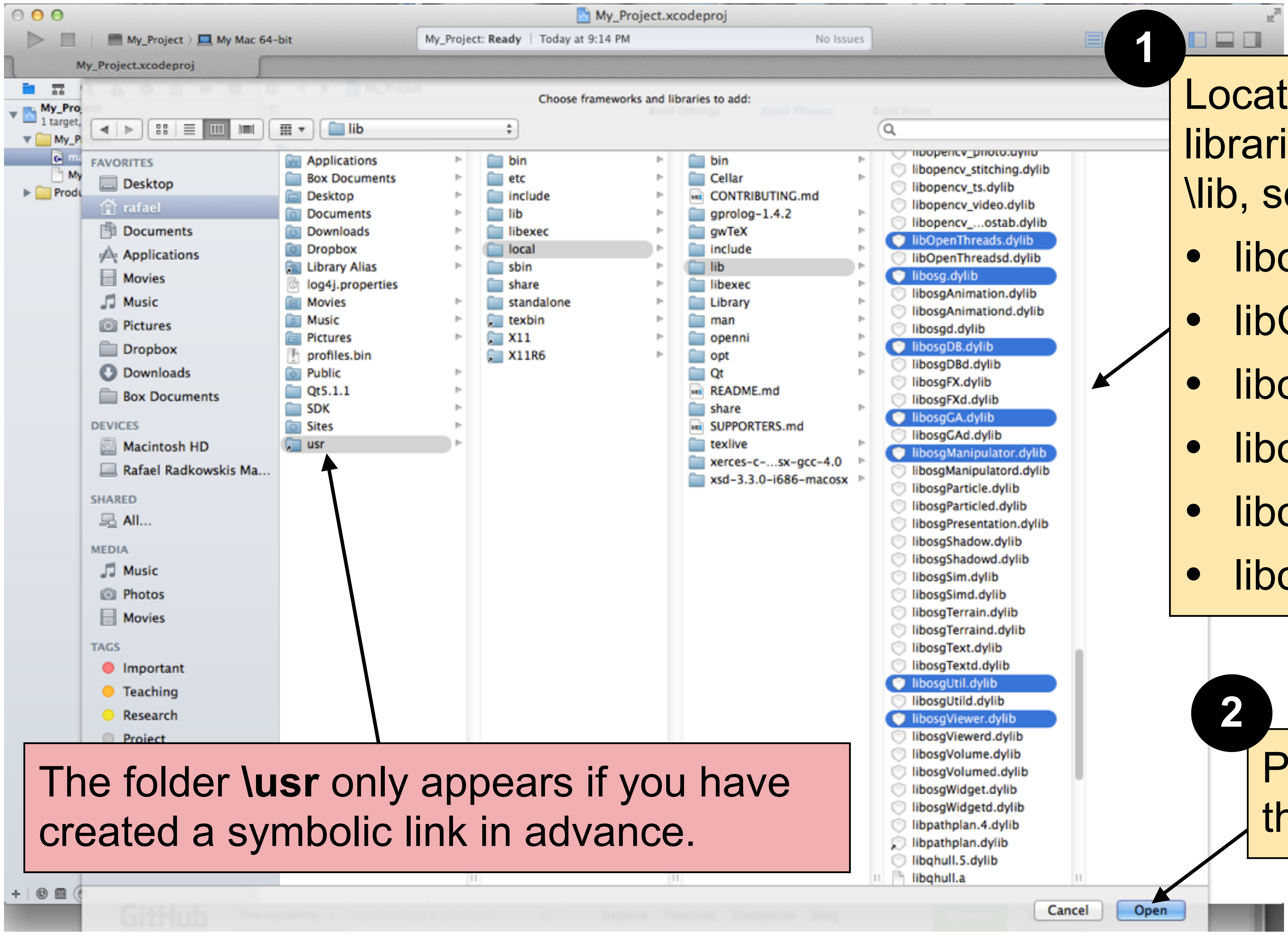

Choose frameworks and libraries to add:

OS X 10.9
- Accelerate.framework
- Accounts.framework
- AddressBook.framework
- AGL.framework
- AppKit.framework
- AppKitScripting.framework
- AppleScriptKit.framework
- AppleScriptObjC.framework
- AppleShareClientCore.framework
- ApplicationServices.framework
- AudioToolbox.framework
- AudioUnit.framework
- AudioVideoBridging.framework
- Automator.framework
- AVFoundation.framework

1 Press “Add Other...”

XCode: add library files

A Finder window will appear. Locate your link to \usr and navigate to \usr\local\lib. Add the OSG libraries which are necessary for your project.



1 Locate you osg libraries at \usr\local \lib, select them i.e.,

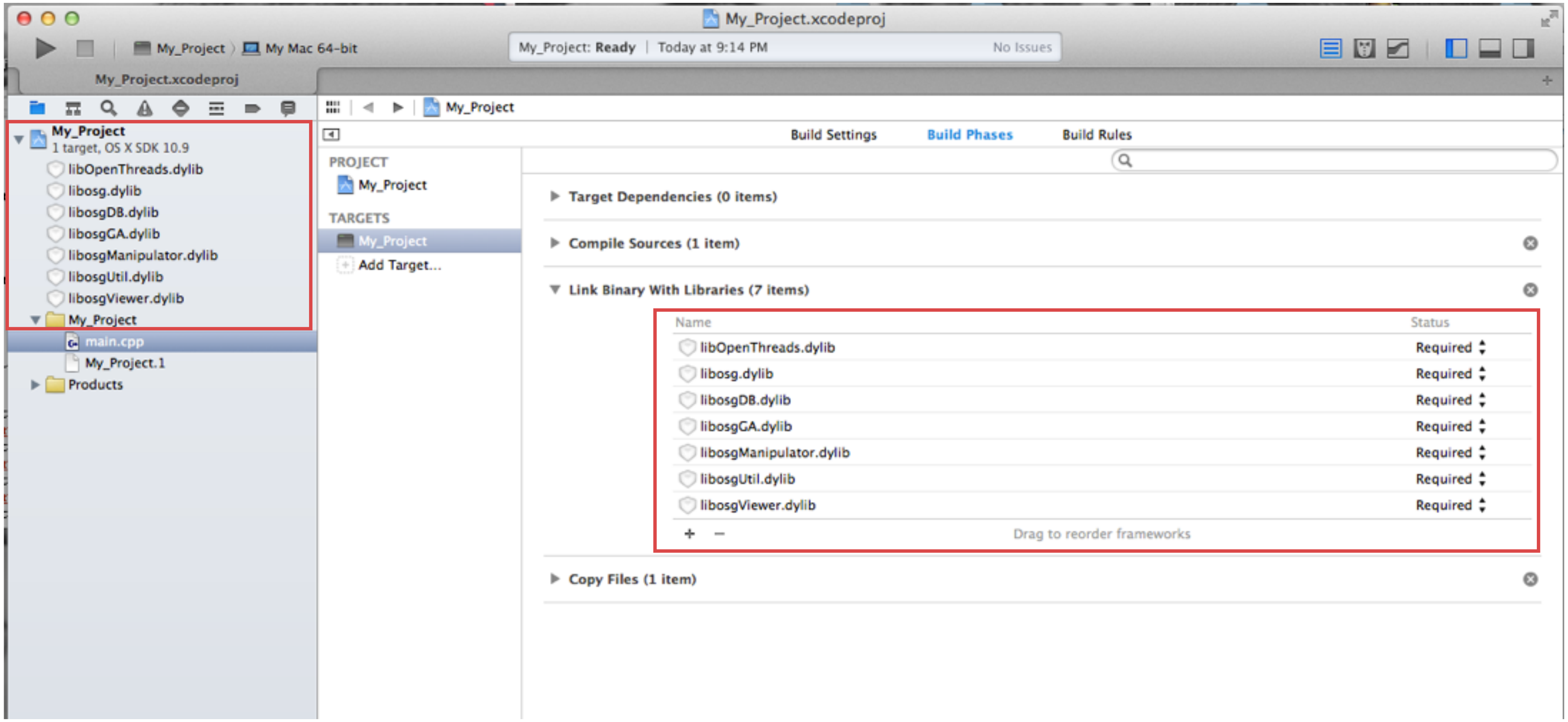
- libosg.dylib
- libOpenThread.dylib
- libosgDB.dylib
- libosgGA.dylib
- libosgUtil.dylib
- libosgViewer.dylib

The folder **\usr** only appears if you have created a symbolic link in advance.

2 Press “Open” to add them

XCode: add library files

The library files should appear in the tree view at the left as well as in the "Link Binary with ...." tab.



My_Project
1 target, OS X SDK 10.9
libOpenThreads.dylib
libosg.dylib
libosgDB.dylib
libosgGA.dylib
libosgManipulator.dylib
libosgUtil.dylib
libosgViewer.dylib

Target Dependencies (0 items)

Compile Sources (1 item)

Link Binary With Libraries (7 items)

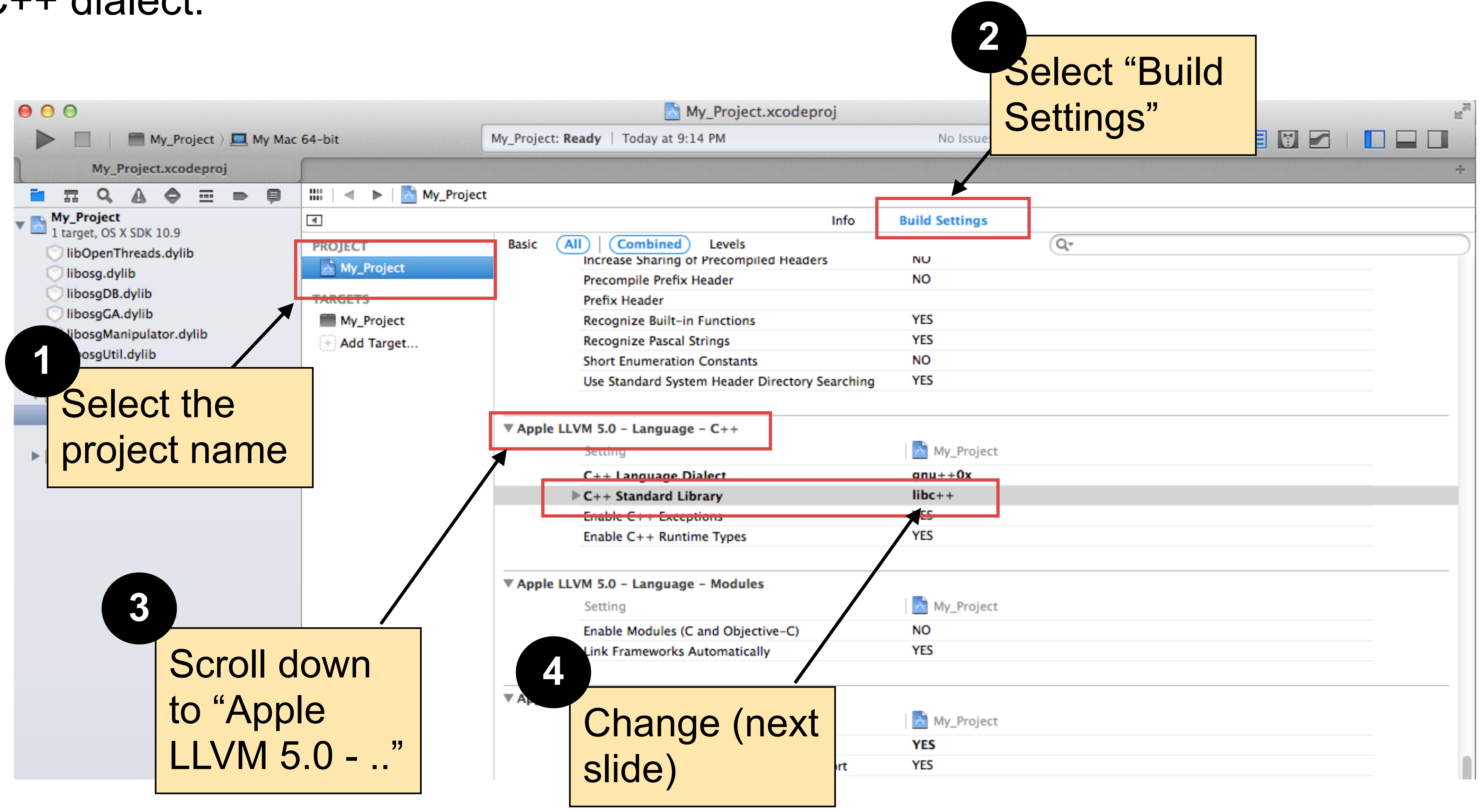
Name	Status
libOpenThreads.dylib	Required
libosg.dylib	Required
libosgDB.dylib	Required
libosgGA.dylib	Required
libosgManipulator.dylib	Required
libosgUtil.dylib	Required
libosgViewer.dylib	Required

Drag to reorder frameworks

Copy Files (1 item)

XCode: change the C++ Language

XCode 5 uses by default a C++ language dialect that does not comply with C++ policies necessary for OpenSceneGraph applications. It is necessary to change the C++ dialect.



1 Select the project name

2 Select "Build Settings"

3 Scroll down to "Apple LLVM 5.0 - .."

4 Change (next slide)

Increase Sharing of Precompiled Headers	NO
Precompile Prefix Header	NO
Prefix Header	
Recognize Built-in Functions	YES
Recognize Pascal Strings	YES
Short Enumeration Constants	NO
Use Standard System Header Directory Searching	YES

▼ Apple LLVM 5.0 – Language – C++

Setting	My_Project
C++ Language Dialect	gnu++0x
▶ C++ Standard Library	libc++
Enable C++ Exceptions	YES
Enable C++ Runtime Types	YES

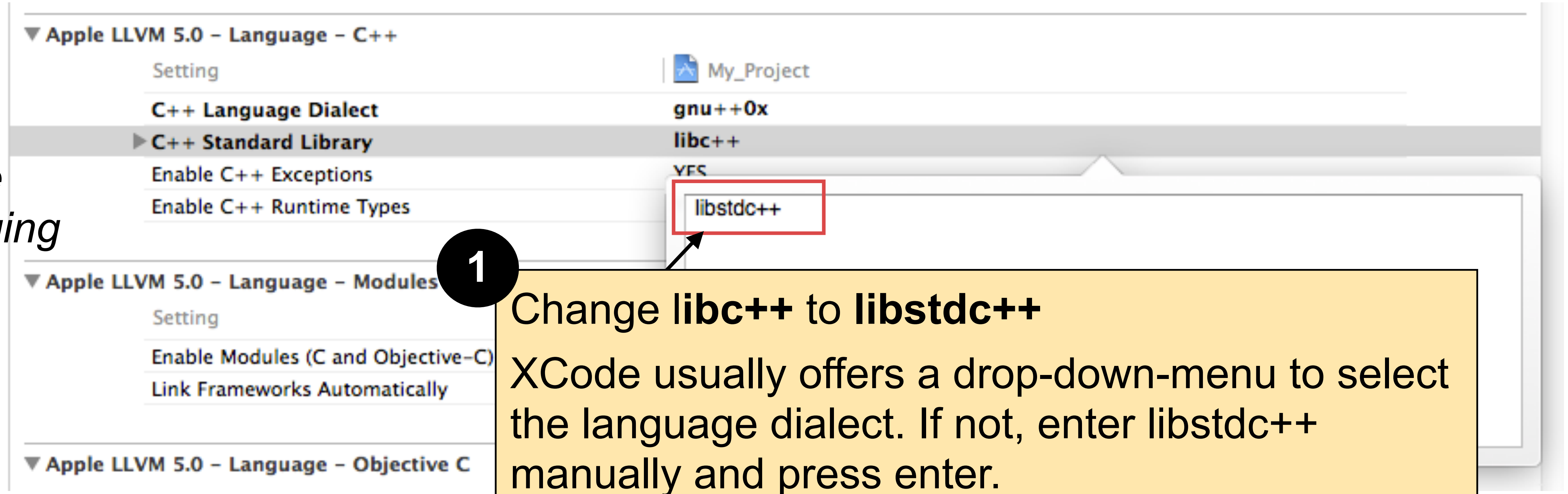
▼ Apple LLVM 5.0 – Language – Modules

Setting	My_Project
Enable Modules (C and Objective-C)	NO
Link Frameworks Automatically	YES

XCode: change the C++ Language

XCode 5 uses by default a C++ language dialect that does not comply with C++ policies necessary for OpenSceneGraph applications. It is necessary to change the C++ dialect.

Before Changing



▼ Apple LLVM 5.0 – Language – C++

Setting	My_Project
C++ Language Dialect	gnu++0x
▶ C++ Standard Library	libc++
Enable C++ Exceptions	YES
Enable C++ Runtime Types	libstdc++

▼ Apple LLVM 5.0 – Language – Modules

Setting	
Enable Modules (C and Objective-C)	
Link Frameworks Automatically	

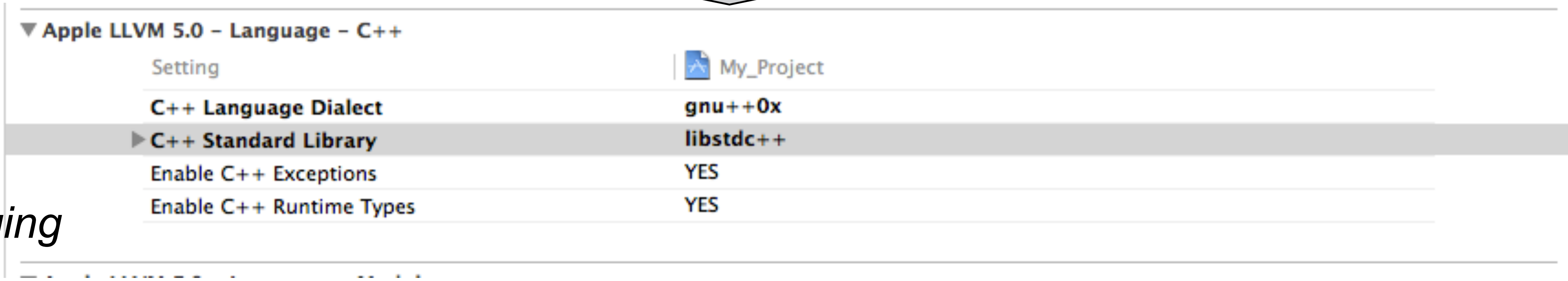
▼ Apple LLVM 5.0 – Language – Objective C

1

Change **libc++** to **libstdc++**

XCode usually offers a drop-down-menu to select the language dialect. If not, enter libstdc++ manually and press enter.

After changing

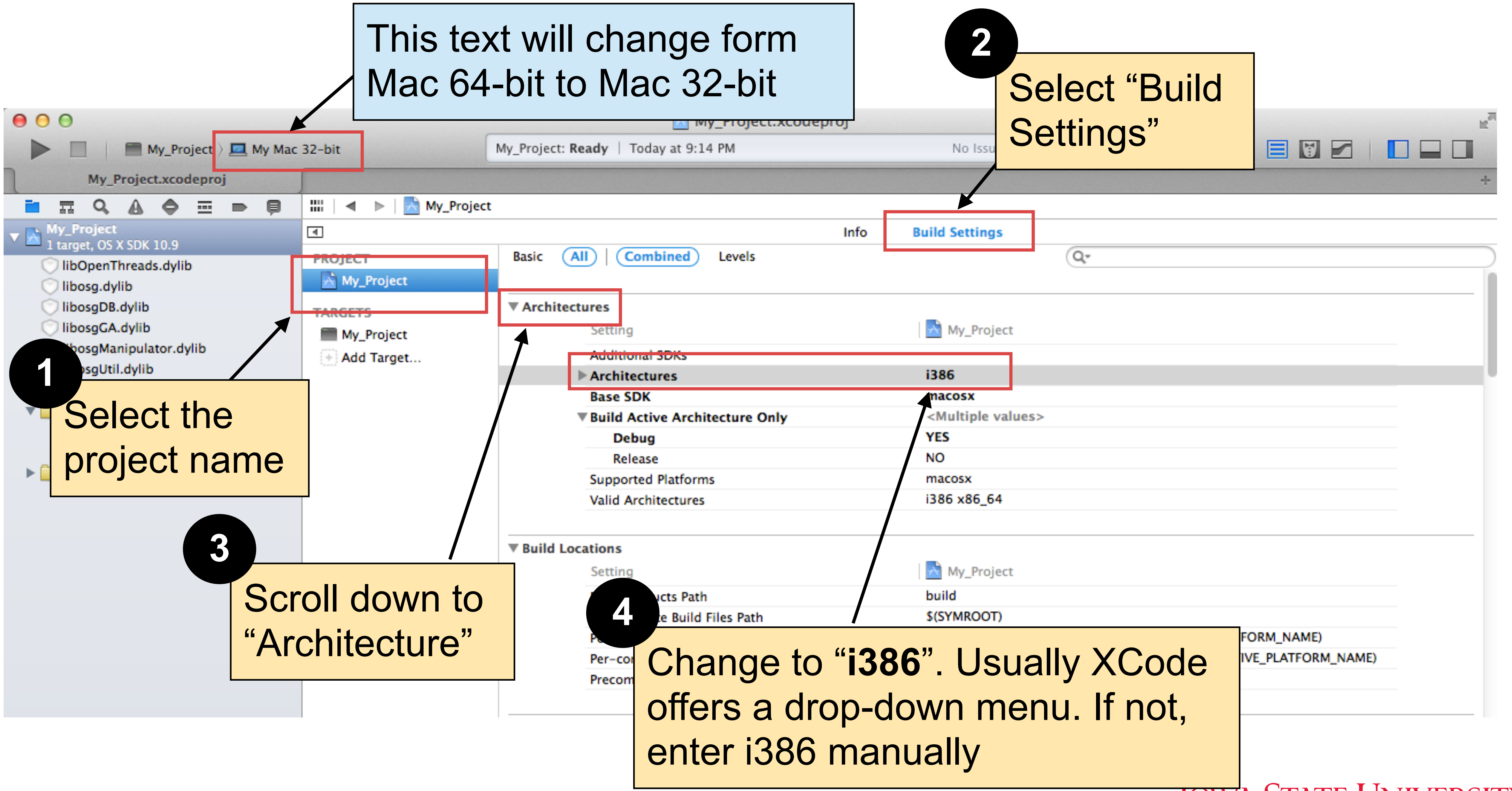


▼ Apple LLVM 5.0 – Language – C++

Setting	My_Project
C++ Language Dialect	gnu++0x
▶ C++ Standard Library	libstdc++
Enable C++ Exceptions	YES
Enable C++ Runtime Types	YES

XCode: set 32 bit build version

Most of the libraries that we have to use in HCI 571X are 32 bit libraries. To make sure XCode can use them, change the build architecture to 32 bit.



This text will change form Mac 64-bit to Mac 32-bit

1 Select the project name

2 Select “Build Settings”

3 Scroll down to “Architecture”

4 Change to “**i386**”. Usually XCode offers a drop-down menu. If not, enter i386 manually

▼ Architectures

Setting	My_Project
Additional SDKs	
▶ Architectures	i386
Base SDK	macosx
▼ Build Active Architecture Only	<Multiple values>
Debug	YES
Release	NO
Supported Platforms	macosx
Valid Architectures	i386 x86_64

▼ Build Locations

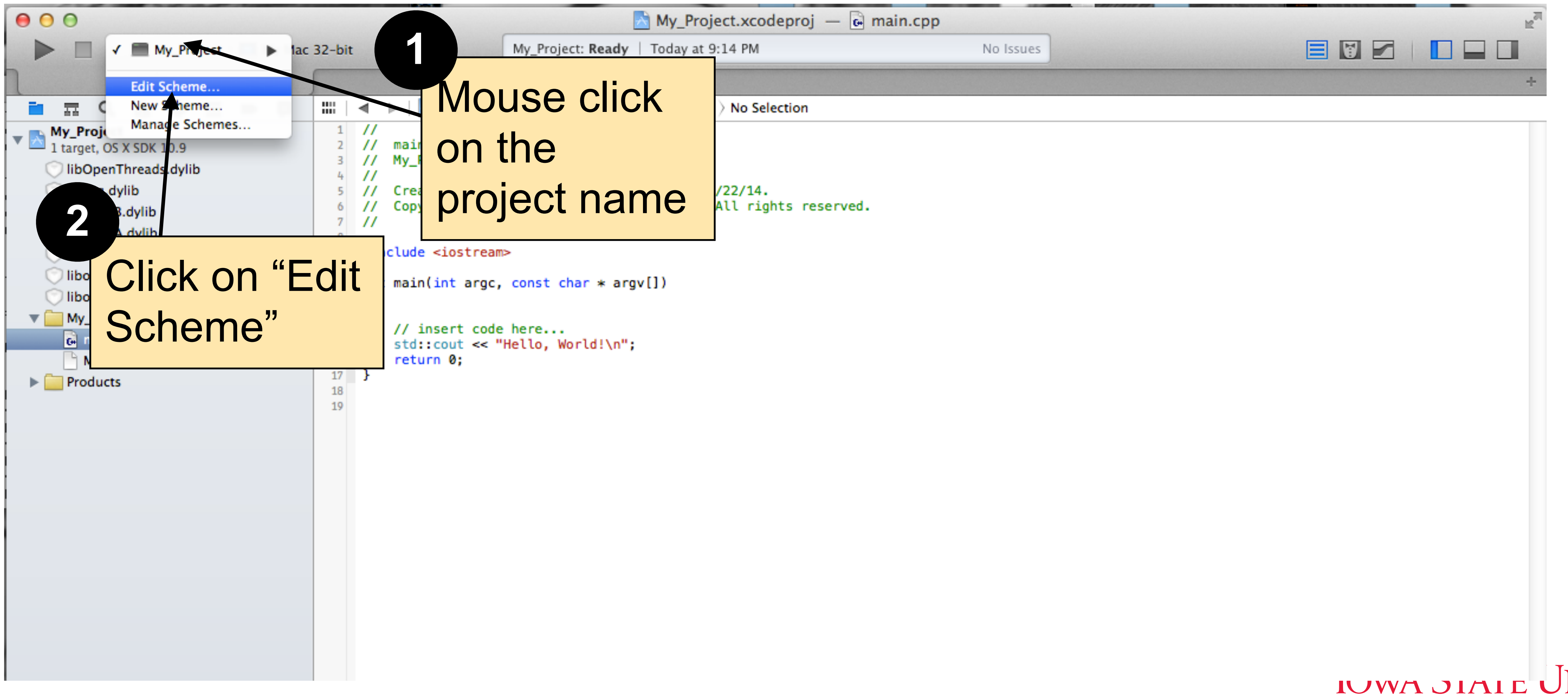
Setting	My_Project
ucts Path	build
e Build Files Path	$(SYMROOT)
P	FORM_NAME)
Per-co	IVE_PLATFORM_NAME)
Precom	

Debug and Release

Development environments distinguish Debug and Release versions of an application.

Debug: this version keeps the link between the main memory and your code during runtime. It enables the development environment to read variable values, which helps to debug code and to find error. But, debug versions run slow.

Release: this version should be used to deploy an application. Debugging of code (reading memory values) is not possible. But the application runs faster.



```
//
//  mai
//  My_P
//
//  Crea                 /22/14.
//  Copy                 All rights reserved.
//

 clude <iostream>

 main(int argc, const char * argv[])

 // insert code here...
 std::cout << "Hello, World!\n";
 return 0;
}
```

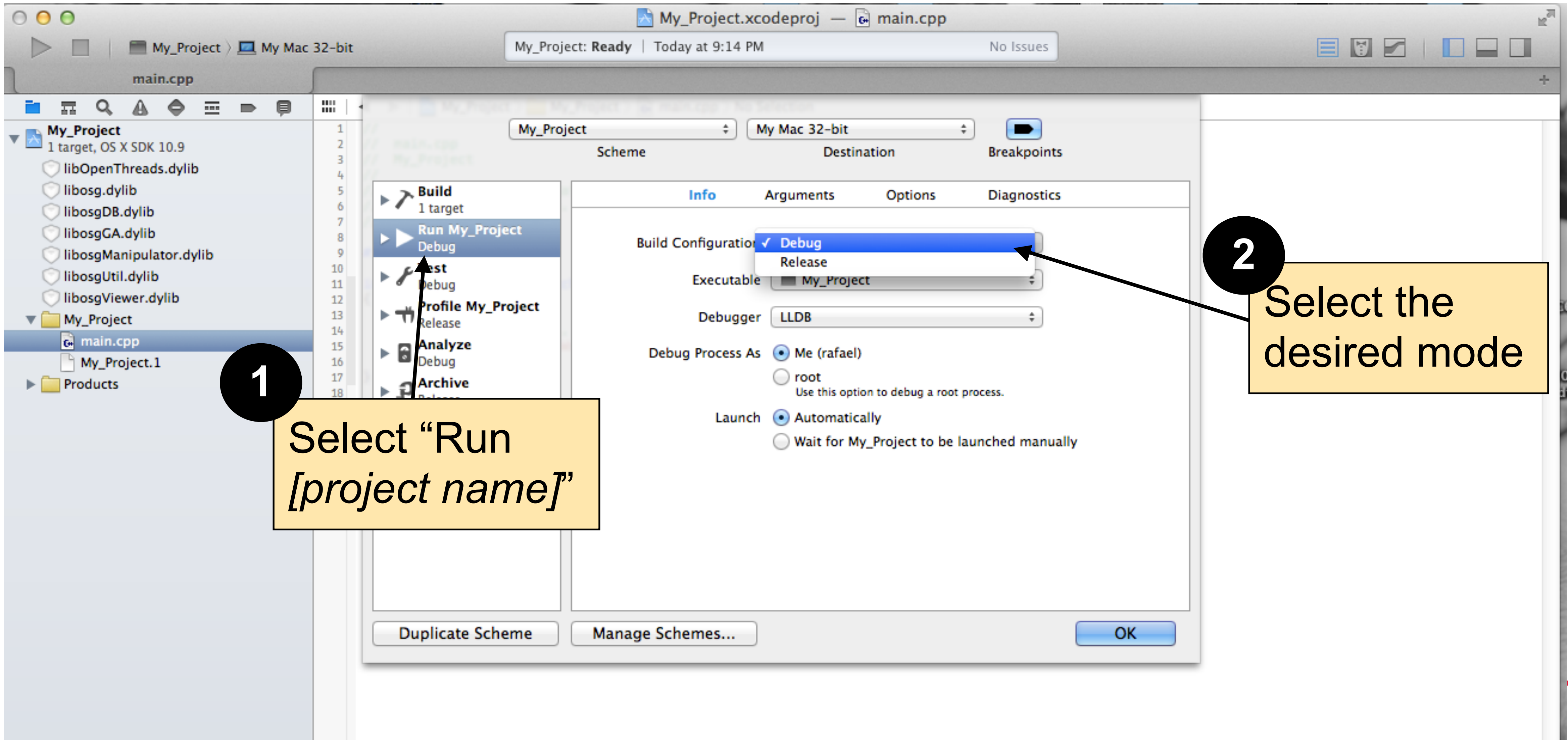
Debug and Release

VRAC|HCI

Development environments distinguish Debug and Release versions of an application.

Debug: this version keeps the link between the main memory and your code during runtime. It enables the development environment to read variable values, which helps to debug code and to find error. But, debug versions run slow.

Release: this version should be used to deploy an application. Debugging of code (reading memory values) is not possible. But the application runs faster.



My_Project.xcodeproj — main.cpp

My_Project: Ready | Today at 9:14 PM | No Issues

Scheme: My_Project | Destination: My Mac 32-bit | Breakpoints

- Build — 1 target
- Run My_Project — Debug
- Test — Debug
- Profile My_Project — Release
- Analyze — Debug
- Archive — Release

Info | Arguments | Options | Diagnostics

Build Configuration: [✓ Debug] Release

Executable: My_Project

Debugger: LLDB

Debug Process As: [✓ Me (rafael)] root — Use this option to debug a root process.

Launch: [✓ Automatically] Wait for My_Project to be launched manually

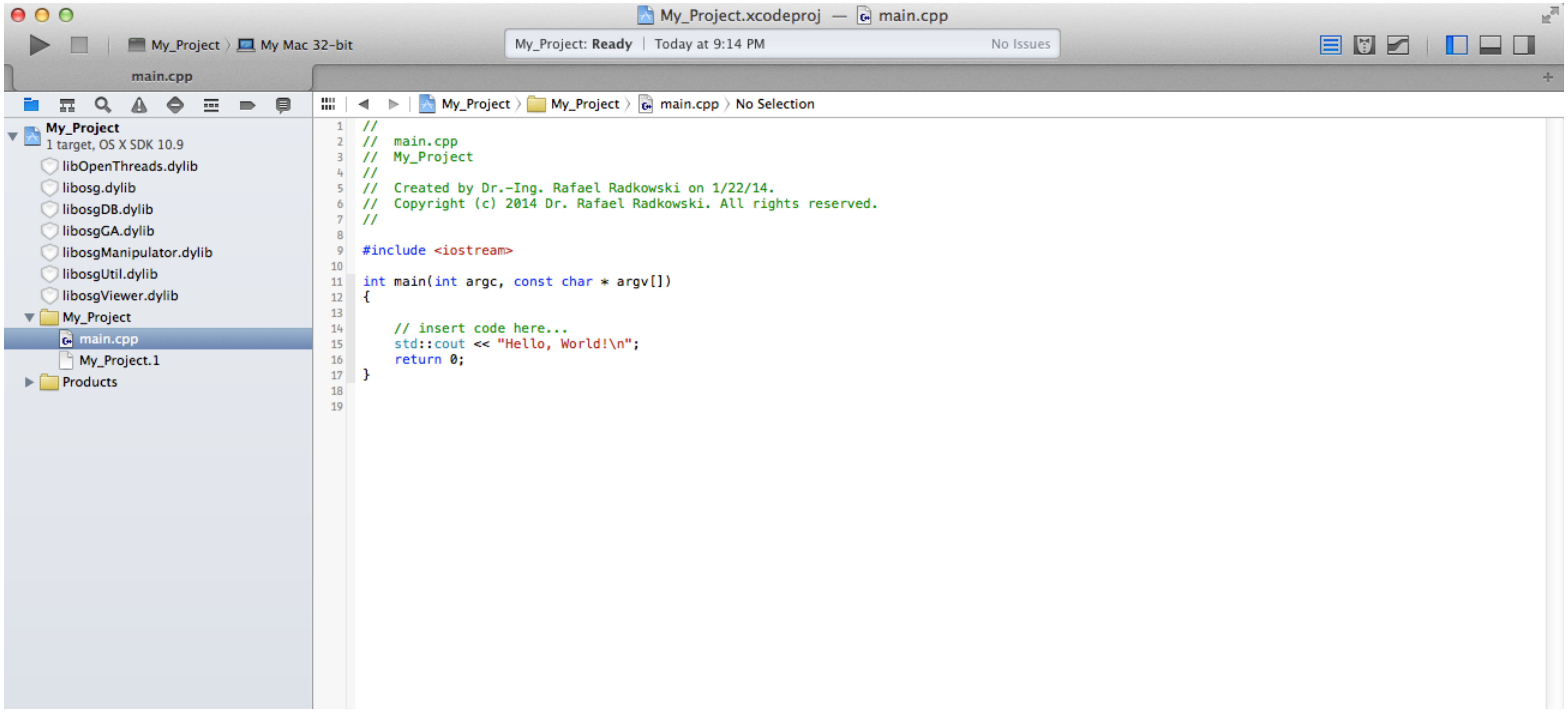
1 Select “Run *[project name]*”

2 Select the desired mode

NIVERSITY
OLOGY

Ready to develop code

XCode 5 is ready for development now.



```
//
//  main.cpp
//  My_Project
//
//  Created by Dr.-Ing. Rafael Radkowski on 1/22/14.
//  Copyright (c) 2014 Dr. Rafael Radkowski. All rights reserved.
//

#include <iostream>

int main(int argc, const char * argv[])
{

    // insert code here...
    std::cout << "Hello, World!\n";
    return 0;
}
```