Building XOPs with Xcode 3.2 or Later

If you are using Xcode 3.2 or later, the instructions in Chapter 2 of the XOP Toolkit 5 manual are out-of-date. This document contains corrections and replacement instructions for the sections that are out-of date.

Building SimpleFit

This section of the XOP manual is OK except for one thing.

When you get to the step of building the SimpleFit XOP, if you get an error, follow these instructions:

1. Double-click the SimpleFit icon in the Targets group to open the Target Info window.

2. Click the Build Tab.

3. Change the "Base SDK" setting in the Architectures section to "Mac OS X 10.5".

4. Close the Info window.

5. Click the Build button.

The following section is new. Follow these instructions before proceeding.

Using Xcode 3.2 or Later

To create an XOP project in Xcode 3.2 or later you need to add a template to your Xcode installation. Before we explain how to do that, here is some background information.

Xcode uses "templates" as the starting point for new projects.

An Igor XOP is a "Carbon bundle". "Carbon" means that it uses Apple's Carbon framework which is a set of operating-system supplied routines and the supporting header files and libraries. "Bundle" means that it is a loadable module (a.k.a., "extension" or "plug-in"). Consequently, you need the "Carbon Bundle" template to create a new XOP project in Xcode.

In Xcode 3.2, Apple stopped shipping the Carbon project templates. This invalidated the instructions in the XOP Toolkit 5 manual for creating an XOP project. To fix that, we need to add the Carbon bundle project templates back to Xcode 3.2 by copying them from another version of Xcode.

The project templates are located in different directories depending on your version of Xcode. For example:

Xcode 2.x: /Library/Application Support/Apple/Developer Tools/Project Templates

Xcode 3.x: /Developer/Library/Xcode/Project Templates

For versions prior to Xcode 3.2, the Carbon bundle templates are in the "Bundle/Carbon Bundle" subfolder of the Project Templates folder.

Adding the Carbon Bundle Template to Xcode 3.2 or Later

Find the Carbon Bundle subfolder in Xcode 3.1 or earlier or download them from:
ftp://ftp.wavemetrics.net/IgorXOPToolkit/Xcode/CarbonBundleTemplates.zip

If you downloaded, unzip the archive. Either way you now have a folder named "Carbon Bundle" that contains the missing templates.

Copy the Carbon Bundle folder to the Other folder in the Xcode 3.2 or later project templates folder. You should now have a Carbon Bundle folder at this location on your hard disk:
/Developer/Library/Xcode/Project Templates/Other

Restart Xcode.

The following section is new. Follow this instead of the section "Creating the New Project in Xcode" in the XOP Toolkit 5 manual.

Creating the New Project In Xcode 3.2

Do the steps in this section to create the project in Xcode 3.2 or later.

NOTE: For Xcode 3.2 or later you must add a template to your Xcode installation in order to create an XOP project. See **Using Xcode 3.2 or Later** in Chapter 3 before proceeding.

1-Xcode 3.2.

On the desktop, create a folder named Xcode32 inside the SimpleGaussFit folder.

This folder will hold files specific to the Xcode development system.

2-Xcode 3.2.

In Xcode, choose File->New Project.

In the resulting New Project window, click the Other category in the lefthand pane.

Click "Carbon Bundle" in the righthand pane.

If you don't see "Carbon Bundle", see the NOTE above.

Click the Choose button. A Save File dialog appears.

Enter “SimpleGaussFit” in the Save As box.

Navigate to the SimpleGaussFit/Xcode32 folder and click Save.

You now have a SimpleGaussFit project.

Close the project by clicking the close box on the project window.

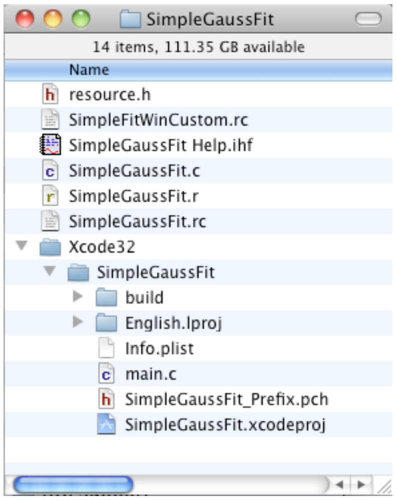
You now have the following hierarchy on disk:

/XOP Toolkit 5/IgorXOPs5/SimpleGaussFit/Xcode32/SimpleGaussFit/

The last SimpleGaussFit is extraneous so we will now get rid of it.

In the Finder, open IgorXOPs5/SimpleGaussFit and set the view to List (using the Finder's View menu).

Click the disclosure triangle on the Xcode32 subfolder so you see this:

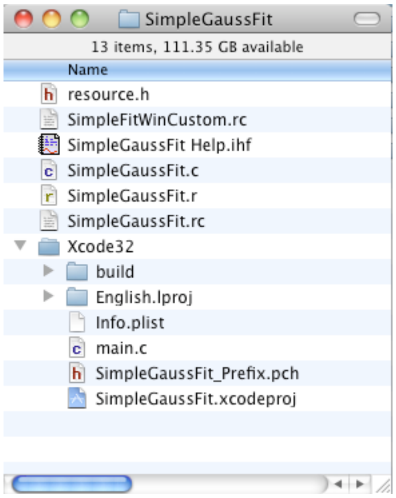


```
SimpleGaussFit
14 items, 111.35 GB available
Name
resource.h
SimpleFitWinCustom.rc
SimpleGaussFit Help.ihf
SimpleGaussFit.c
SimpleGaussFit.r
SimpleGaussFit.rc
▼ Xcode32
    ▼ SimpleGaussFit
        ▶ build
        ▶ English.lproj
          Info.plist
          main.c
          SimpleGaussFit_Prefix.pch
          SimpleGaussFit.xcodeproj
```

Drag the *contents* of IgorXOPs/SimpleGaussFit/Xcode32/SimpleGaussFit into IgorXOPs/SimpleGaussFit/Xcode.

Delete the now empty IgorXOPs/SimpleGaussFit/Xcode32/SimpleGaussFit folder.

You now have the following hierarchy:



```
SimpleGaussFit
13 items, 111.35 GB available
Name
resource.h
SimpleFitWinCustom.rc
SimpleGaussFit Help.ihf
SimpleGaussFit.c
SimpleGaussFit.r
SimpleGaussFit.rc
▼ Xcode32
    ▶ build
    ▶ English.lproj
      Info.plist
      main.c
      SimpleGaussFit_Prefix.pch
      SimpleGaussFit.xcodeproj
```

3-Xcode 3.2.

Double-click SimpleGaussFit.xcodeproj to open it in Xcode.

4-Xcode 3.2.

In Xcode, in the project window, open the SimpleGaussFit group at the top of the Groups & Files list, then open the Source folder in the SimpleGaussFit group.

5-Xcode 3.2.

Select the main.c sample file that was put there by Xcode.

Choose Edit->Delete.

In the resulting dialog, click the Also Move to Trash button.

This removes main.c from the project and moves it to the trash.

6-Xcode 3.2.

Click the Source group icon and choose Project->Add To Project.

In the resulting dialog, select the following files:

SimpleGaussFit.c
SimpleGaussFit.r

Now click the Add button.

Xcode now presents another dialog.

7-Xcode 3.2.

Click the Add button.

File references will be project-relative so that if you move your IgorXOPs5 folder, the project references will remain valid.

8-Xcode 3.2.

In the Groups & Files list, select the SimpleGaussFit->External Frameworks & Libraries icon.

Choose Project->Add To Project and add the file:

IgorXOPs5/XOPSupport/Xcode/libXOPSupport.a.

Xcode now presents another dialog.

9-Xcode 3.2.

Click the Add button to add the library.

10-Xcode 3.2

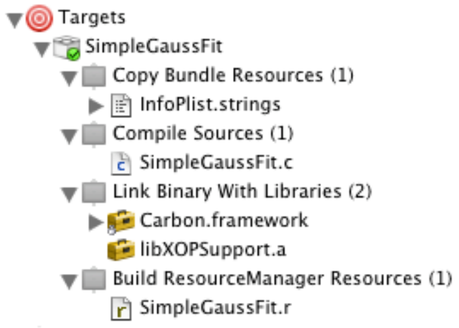
Open the Targets icon, select the Targets->SimpleGaussFit icon, control-click and choose Add->New Build Phase->New Build ResourceManager Resources Build Phase.

Close the blank window opened by Xcode.

Open the Targets->SimpleGaussFit icon.

Drag the SimpleGaussFit.r icon from the Source group into the Build ResourceManager Resources icon in the Targets->SimpleGaussFit icon.

The Targets part of your Groups & Files pane should now look like this:



- Targets
  - SimpleGaussFit
    - Copy Bundle Resources (1)
      - InfoPlist.strings
    - Compile Sources (1)
      - SimpleGaussFit.c
    - Link Binary With Libraries (2)
      - Carbon.framework
      - libXOPSupport.a
    - Build ResourceManager Resources (1)
      - SimpleGaussFit.r

11-Xcode 3.2.

In the desktop, find the Exports.exp file in the SimpleFit/Xcode folder and *copy* it to the SimpleGaussFit/Xcode32 folder.

The Exports.exp file is a plain text file that tells the Xcode what symbols the project exports. In the case of an XOP, the only exported symbol is “main” since this is the only routine in the XOP that Igor must be able to locate by name.

12-Xcode 3.2.

Back in Xcode, in the Groups & Files list, open the Targets icon.

Double-click the Targets->SimpleGaussFit icon.

This displays the info window for the SimpleGaussFit target.

Click the Build tab.

Choose All Configurations from the Configurations popup menu.

Choose All Settings from the Show popup menu.

Find the Architectures section and enter the following build settings:

Architectures	32-bit Universal
Base SDK	Mac OS X 10.5

If you elected to install OS X 10.4 support when you installed Xcode then you can choose "Mac OS X 10.4" for "Base SDK". This will allow your XOP to run with OS X 10.4. Otherwise your XOP will require OS X 10.5 or later.

13-Xcode 3.2.

Scroll to the Search Paths section and enter the following build settings:

Header Search Paths	../../XOPSupport
Library Search Paths	../../XOPSupport/Xcode

To edit these settings you must click directly on the text already in the settings and wait a second until Xcode switches to edit mode. You can then edit the settings as text. Clicking in blank space in the text area for the setting that contains text does not put you into edit mode - you must click directly on the existing text.

14-Xcode 3.2.

Scroll to the Linking section and enter these settings:

Exported Symbols File	./Exports.exp

Scroll to the User-Defined section, select the ZERO_LINK setting and delete it by clicking the gear icon at the bottom/left corner of the window and choosing "Delete Definition at this Level".

15-Xcode 3.2.

Scroll to the Packaging section and enter the following setting:

Wrapper Extension	xop

16-Xcode 3.2.

Scroll to the GCC Warnings section and enter the following setting:

Warn About Deprecated Functions	Unchecked

If you ever decided to modernize your code you can turn this warning on.

17-Xcode 3.2.

Scroll to the Rez - Language section and enter the following setting:

OTHER_REZ_FLAGS	-i ../../XOPSupport -d TARGET_RT_MAC_MACHO

The last letter of TARGET_RT_MAC_MACHO is “oh”, not zero. If you spell it wrong, your XOP’s XOPI resource will be wrong and Igor will display an error when you try to run your XOP.

18-Xcode 3.2.

Click the Properties tab at the top of the info window and enter the following settings:

Identifier	com.wavemetrics.xops.curvefitting.simplegaussfit
Type	IXOP (third character is “oh”, not zero)
Creator	IGR0 (last character is zero, not “oh”)

The identifier is a unique string associated with an executable. By convention it uses Java package notation. For your own XOPs you can make up an identifier of your choice.

This is also the place where you would enter the version number of your XOP. We will leave it at the default value of 1.0.

19-Xcode 3.2.

Close the info window.

20-Xcode 3.2.

Choose Build->Build.

Xcode will build the project. You may get some warnings about items where the Xcode compiler is pickier than most. When Xcode is finished, it should display a “Build succeeded” message in the lower left corner of the project window.

21-Xcode 3.2.

In the desktop, verify that you have a package folder named SimpleGaussFit.xop in IgorXOPs5/SimpleGaussFit/Xcode32/build/Debug.

Debug is the name of the Xcode configuration currently being built so the output goes into the /build/Debug directory.

The SimpleGaussFit.xop package folder is your compiled XOP. It should look like a file in the Finder and you should have to Control-click it and choose Show Package Contents to see what it contains. Inside the package you will find a file named SimpleGaussFit. This is the executable file and contains both PowerPC and Intel code.

The PowerPC code will execute when you run with a PowerPC version of Igor. The Intel code will execute when you run with an Intel version of Igor.

22-Xcode 3.2.

Make an alias for the SimpleGaussFit.xop package folder and put the alias in your Igor Extensions folder.