Tutorial : creating a Sensomusic Usine user module on MacOS X

Version 1.0 (29th April 2018) by Benoit Bouchez (alias iModularSynth)

Introduction

This document is a short tutorial explaining how to create "from scratch" a user module for Sensomusic Usine using Xcode on MacOS X platform. It is not meant to explain how Usine modules are working, the SDK User Manual made by Martin Fleurent is perfect for that.

This tutorial was made using Usine Hollyhock 3 SDK and Xcode 7. However, the process to create user modules for Usine Hollyhock 2 is exactly the same.

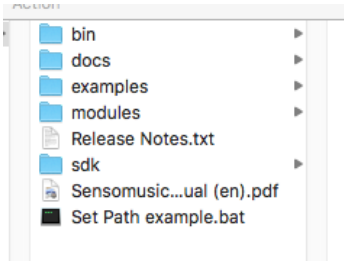
Preparing the computer

First of all, Xcode must be installed on your computer (and you are supposed to know how to use it...). Xcode is available for free on Apple website. We recommend you to use Xcode 7 as a minimum version (note that Xcode 6 is no more supported on MacOS 10.13 – aka High Sierra).

Download the Usine SDK on Sensomusic website and copy the whole directory on your hard disk. The SDK contains excellent examples to understand how Usine is working and what you have to write to make your own modules. We will concentrate here on the steps to follow to create modules from scratch (you can then copy source code from the projects in the SDK to have a good starting point).

As far as we know, the SDK version 7.xx for Usine HH3 works also to create modules for Usine HH2 (even if "official" SDK version for Usine HH2 is 6.xx)

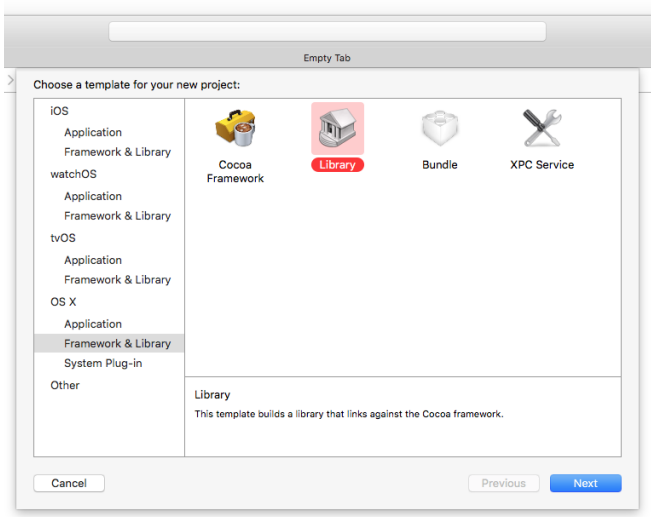
Once the SDK is copied on your hard disk, you should see the following structure in the SDK folder



Technically, Usine user modules are nothing else than a dylib (Dynamic Library), but you need to define some specific properties in the project to make sure that Usine can load and use them.

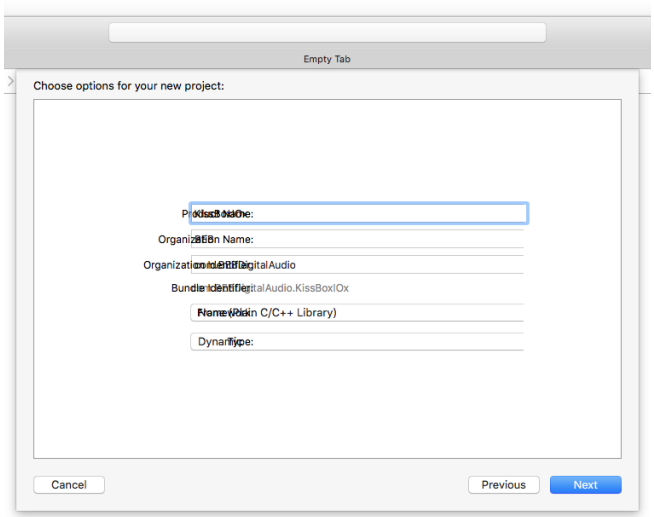
Creating the project in Xcode

Start Xcode. In File menu, select New then Project. The following window is displayed. Select "Framework & Library" in "OS X" section, then click on "Library". Click then on Next button.



You will need to enter project specific data in the next window. By default, Xcode fills Organization Name and Organization Prefix. You need to enter :

- the Product Name : it is normally the name of your module on the hard disk
- the Framework type : you have to select "None (Plain C/C++ Library)
- the Type : you have to select "Dynamic"

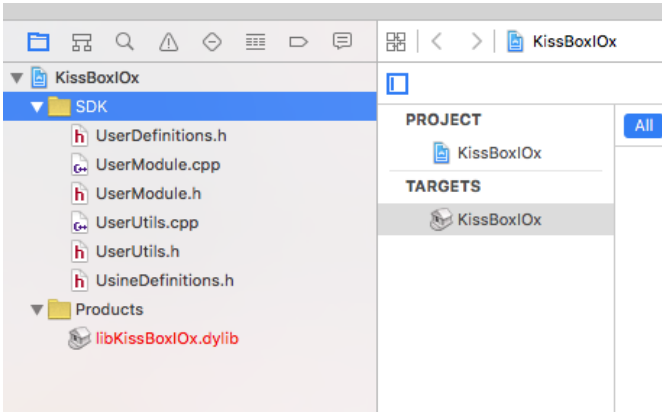


Click on Next button. Xcode will then ask you to choose in which folder the project will be stored. Once the folder is selected, click on Create button.

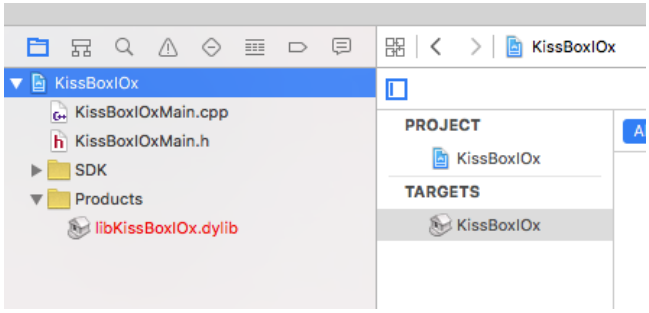
Xcode will then return to the main window.

Adding files to the project
First of all, you need to add the SDK files to your project. These files are mandatory to create the user modules. You can put the files wherever you want in your project, as long as the files are part of the project. For our project, the files are added in a specific group, called SDK.

Right click on the group in which to place the files ("SDK" in our example), then select "Add files to (project name)". Select all the files that are in Sensomusic SDK folder. Your project should then look like the screenshot below (except the project and solution name, of course)



You can now add the project specific source files (the ones that you have to write to make your own module). Detailed explanation about these files are given in SDK User Manual from Sensomusic. Of course, the file names given in the screenshot below are pure examples, your files will be different.



DO NOT TRY TO COMPILE YOUR PROJECT NOW !

The project properties must be set first, and you may experience strange results if you try to compile the module at this step (and the module would be totally unrecogized by Usine and then, it will be useless)

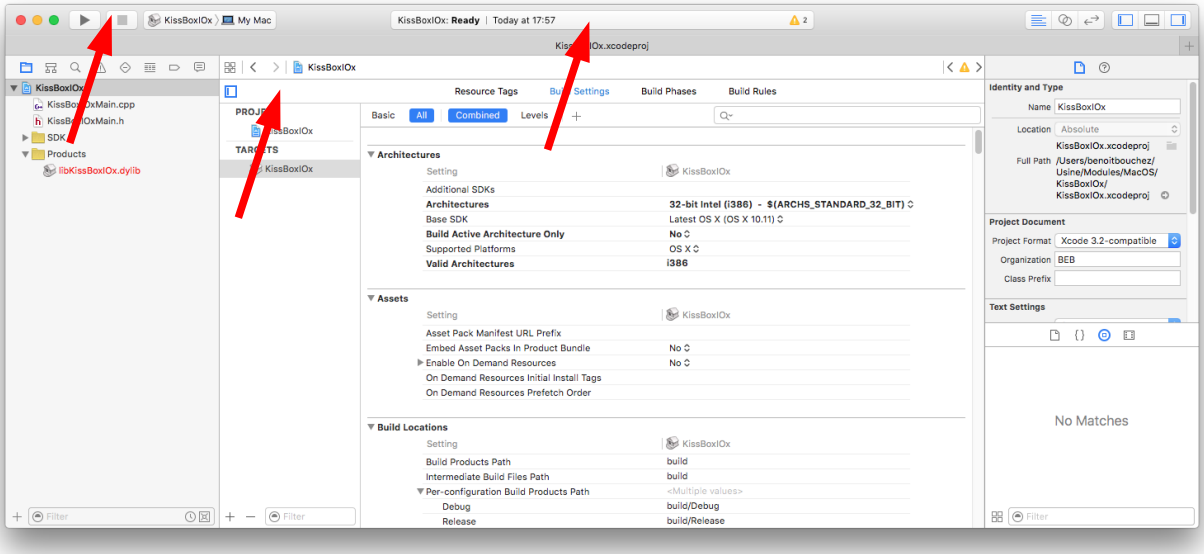
Defining Xcode project properties for the Usine modules

In order to define project properties, you need to click on project name in Project Navigator, then click on corresponding Target (in case your Xcode project contains projects for multiple modules), then select Build Settings.

Note that Xcode will try to generate 64-bits dylib by default. Usine HH2 and HH3 are 32-bits application and they require 32-bits dylibs. Xcode must then be adjusted accordingly.

IMPORTANT: don't forget to set the parameters both for Debug and Release configurations

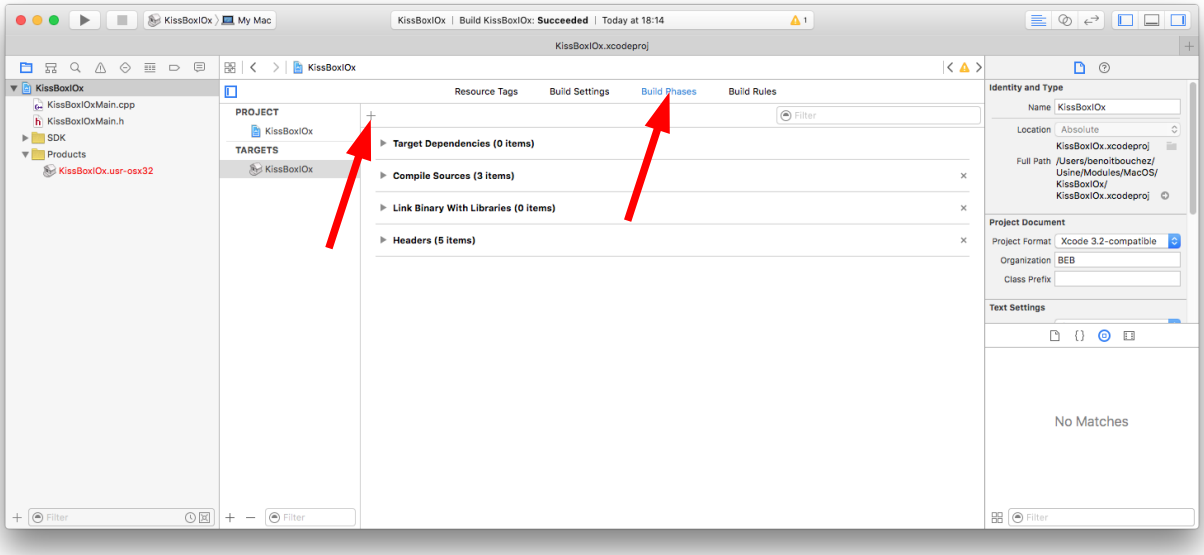
The following parameters must be set :



- Architectures : set to **32-bit Intel (i386) - $(ARCHS_STANDARD_32_BIT)**
- Base SDK : select the SDK you want to use. In most cases, you can use **Latest OS X**.
- Build Active Architecture : set to **No**
- Valid Architectures : set to **i386**
- Code signing entity : set to **Don't Code Sign**
- OS X Deployment Target : set to **OS X 10.6**
- Skip Install : set to **YES**
- Strip Debug Symbol During Copy : set to No for Debug, and set to Yes for Release
- Executable Extension : set to **usr-osx32**
- Executable Prefix : clear the line to remove "lib"
- Product Module Name : set the filename you want for the user module
- Product Name : set the filename you want for the user module
- Strings file Output Encoding : set to **UTF-8**
- C++ Language Dialect : set to **Compiler Default**
- C++ Standard Library : set to **Compiler Default**
- Preprocessor Macros (Debug) : set to **DEBUG=1 MACOSX=1 USINE_OSX_1068=1**
- Preprocessor Macros (Release) : set to **NDEBUG=1 MACOSX=1 USINE_OSX_1068=1**

To finish, I recommend to make an automatic copy of your module in a "user module" folder. You can basically copy the file in any directory as long as you configure Usine to find it.

Click on "Build phases" for the project, then click on the "+" symbol. Choose "New Copy Files Phase".



Once the Copy Files phase is created, you need to specify :

- Destination : set to **Absolute Path**
- Path : set the destination path
- Name : select the **usr-osx32** file created in the project

You can now compile your module and test it in Usine.

How to test and debug your module

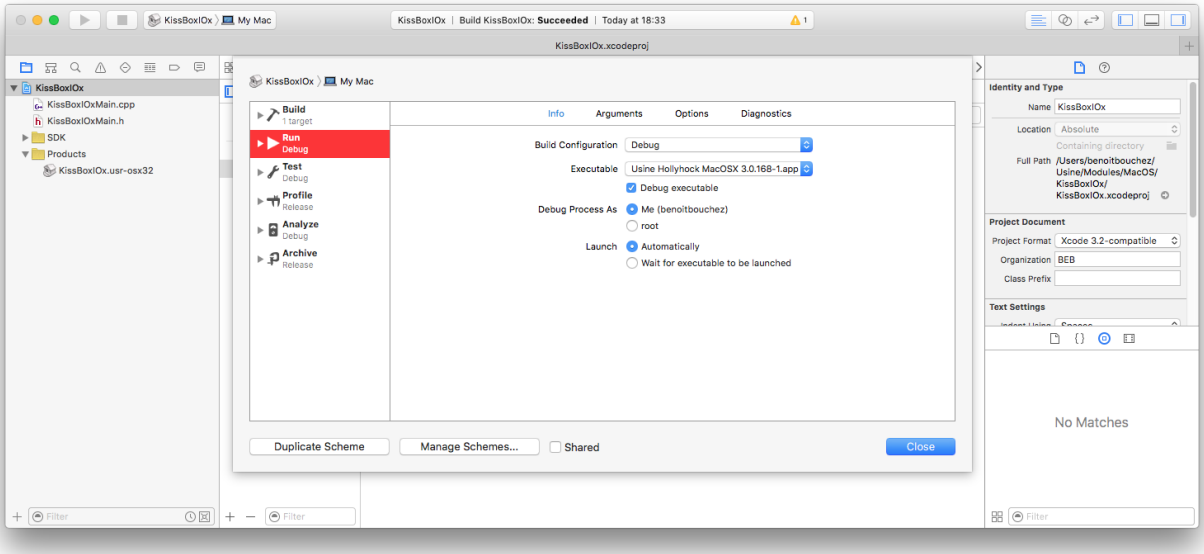
Usine has plenty of cool features, one of them being that it does not rely on a painful copy protection scheme that forces you to use a runtime to debug your creation (some developers will understand me quickly... °-)

You can compile your module and debug it directly in Xcode while Usine is running. You can then put breakpoints, go step-by-step, etc... in your code and debug it efficiently.

In order to use Xcode debugger, you need to declare the application used to start your module (don't forget that a user module in Usine is nothing than a dylib, it then can't run by itself)

Go to Product menu, then select Scheme, then Edit Scheme. Click on Run tab on the left side. Click on Executable list, then select "Other". A file selection dialog will open. Locate the Usine application on the hard disk and click on "Choose" button when the application is selected.

Note that you can define a different application for Debug and Release configurations.



Once your module is compiled in Xcode, click on the "Run" arrow on top of Xcode window. Usine will then start. You can use all debugging features of Xcode (breakpoints, step by step, etc...)