Get started with PS Move Api on MacOS X

The PS Move API (psmoveapi) is a cross-platform C library for the Sony Playstation Move Motion Controller. It includes bindings for Python, Java, Qt, Processing and others.

The library is available for Mac OS X, Windows and Linux (USB and Bluetooth).

This tutorial will guide you through the installation process on MacOS X (10.6 and up) and explain how to set up a PS Move API project in Xcode. I tried to keep it really straightforward and simple to follow. If you're new to this, don't be put off by all the command line stuff. Just follow along and you'll be fine.

Updates
News about the API are posted there: http://thp.io/2010/psmove/
Sources are on Github: https://github.com/thp/psmoveapi
In case you are reading this document elsewhere, the latest revision of the present tutorial can be found there on google docs: http://goo.gl/cahbJ (if you see me editing, say hi!)

I need help!
Subscribe to the PS Move API mailing list: https://lists.ims.tuwien.ac.at/mailman/listinfo/psmove

I want to help!
Subscribe to the PS Move API mailing list: https://lists.ims.tuwien.ac.at/mailman/listinfo/psmove

I want to help but I cannot code!
No problem! You can offer Thomas a beer :) http://flattr.com/thing/147084/PS-Move-API

There's an error in the tutorial!
OMG! Please write about it in the mailing list (copy the title of this tutorial in the title of your mail). I'll correct the mistake as soon as I can.

There's an error in the API
If you spotted an error in the API codebase or see something that could be improved, please file an issue on Github: https://github.com/thp/psmoveapi/issues

1) Installation and pairing

You will need...
... a PS Move controller (obviously)
... a mini-usb cable

... to create a root password: http://youtu.be/10sKJQuXq6A
... to install Homebrew: http://mxcl.github.com/homebrew/ (see step 0.1)
... to install cmake, and git from Homebrew (see step 0.2)

Step 0.1: Installing Homebrew

Open a Terminal window (the Terminal.app is located in /Applications/Utilities/), type in:

```
ruby -e "$(curl -fsSkL raw.github.com/mxcl/homebrew/go)"
```

and press [enter].

Step 0.2: Installing Cmake

Likewise...

```
brew install cmake git
```

Optional

- Download and install Doxygen for mac (to generate documentation)

Installing and pairing instructions

Step 1: Getting ready

Connect the controller via USB.
Enable Bluetooth in MacOS.

The following steps will be done in the Terminal.

Step 2: Choosing install folder

We open the folder where we want the API installed. For the purpose of this tutorial:

```
cd /Developer/Library/
```

Note : the API will install in a sub-folder named /psmoveapi/.

Step 3: Grabbing binaries

We download the source code for the PS Move API and external libraries:

```
git clone git://github.com/thp/psmoveapi.git
cd psmoveapi
git submodule init
git submodule update
```

Step 4: Building

```
bash -e -x contrib/build-osx-snapshot
```

This script installs OpenCV from Git and builds the API itself. It can take a while.

If you feel comfortable with bash commands, you can read the script file to understand what gets built where.

Note: If the script returns some errors, you might need to build the API step-by-step. See “Building the API manually” for the alternative procedure.

Step 5: Checking install

Make sure the controller you are pairing is plugged via USB to your computer then type in:

```
cd build
./example
```

The controller vibrates and the sphere blinks showing that the installation was successful.

Note: If this did not work, you might need to build the API step-by-step. See “Building the API manually” for the alternative procedure.

Step 6: Automatic pairing

We now proceed to the pairing of the controller so that the system recognizes it as a bluetooth device.

```
./psmovepair
```

Repeat with other Move controllers if needed.

Note: Some MacOS 10.7 users have trouble with the automated pairing. If this is your case, see the “Manual pairing” instructions below.

Note 2: Mountain Lion (Mac OS X 10.8) introduced changes related to bluetooth. A fix is underway. You can get more info in the comments of this blog post by Douglas Wilson (http://gutefabrik.com/blog/?p=1843) or asking on the mailing list.

Note 3: Douglas Wilson created an easy pairing tool distributed with Unimove. Get it by downloading uniMove: http://www.copenhagengamecollective.org/projects/unimove/
Extract the package and get to the folder called "Pairing Utility" to find the app.

Step 7: Handshake

Once the pairing is done, press the PS button on the controller and wait until the LED stops blinking.

Click on the Bluetooth icon and you should now see a line reading “Motion Controller” for each connected controller.

Open the “example” file in the /build folder and enjoy the flow of data!

You’re done, congratulations!

Now you can start using the API in your language and environment of choice.

Pairing more controllers

If you need to pair another controller later, connect it via USB then just point to the build folder and run the psmovepair script once again.

```
cd /Developer/Library/psmoveapi/build
./psmovepair
```

Updating

The API gets frequent updates. Be sure to use the latest version. This is how to update:

```
cd /Developer/Library/psmoveapi
git pull
cd build
cmake ..
make -j4
```

Building the API manually

Step 4.1: Build folder

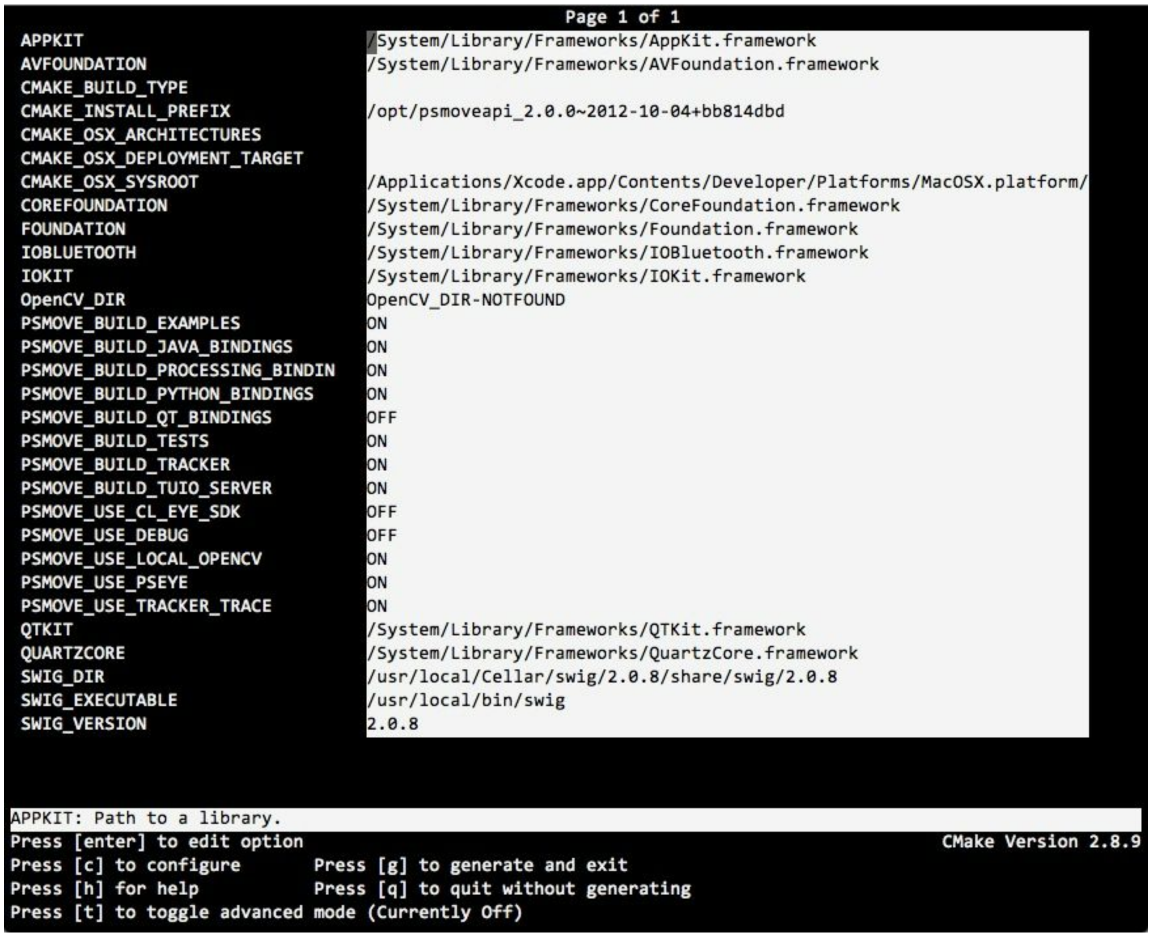
Create the “build” folder and open it:

```
mkdir build
cd build
```

Step 4.2: Build configuration

We configure the build using the ccmake command (careful: two c’s):

```
ccmake ..
```



```
                                                   Page 1 of 1
 APPKIT                           /System/Library/Frameworks/AppKit.framework
 AVFOUNDATION                     /System/Library/Frameworks/AVFoundation.framework
 CMAKE_BUILD_TYPE
 CMAKE_INSTALL_PREFIX             /opt/psmoveapi_2.0.0~2012-10-04+bb814dbd
 CMAKE_OSX_ARCHITECTURES
 CMAKE_OSX_DEPLOYMENT_TARGET
 CMAKE_OSX_SYSROOT                /Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/
 COREFOUNDATION                   /System/Library/Frameworks/CoreFoundation.framework
 FOUNDATION                       /System/Library/Frameworks/Foundation.framework
 IOBLUETOOTH                      /System/Library/Frameworks/IOBluetooth.framework
 IOKIT                            /System/Library/Frameworks/IOKit.framework
 OpenCV_DIR                       OpenCV_DIR-NOTFOUND
 PSMOVE_BUILD_EXAMPLES            ON
 PSMOVE_BUILD_JAVA_BINDINGS       ON
 PSMOVE_BUILD_PROCESSING_BINDIN   ON
 PSMOVE_BUILD_PYTHON_BINDINGS     ON
 PSMOVE_BUILD_QT_BINDINGS         OFF
 PSMOVE_BUILD_TESTS               ON
 PSMOVE_BUILD_TRACKER             ON
 PSMOVE_BUILD_TUIO_SERVER         ON
 PSMOVE_USE_CL_EYE_SDK            OFF
 PSMOVE_USE_DEBUG                 OFF
 PSMOVE_USE_LOCAL_OPENCV          ON
 PSMOVE_USE_PSEYE                 ON
 PSMOVE_USE_TRACKER_TRACE         ON
 QTKIT                            /System/Library/Frameworks/QTKit.framework
 QUARTZCORE                       /System/Library/Frameworks/QuartzCore.framework
 SWIG_DIR                         /usr/local/Cellar/swig/2.0.8/share/swig/2.0.8
 SWIG_EXECUTABLE                  /usr/local/bin/swig
 SWIG_VERSION                     2.0.8



APPKIT: Path to a library.
Press [enter] to edit option                                            CMake Version 2.8.9
Press [c] to configure       Press [g] to generate and exit
Press [h] for help           Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

To modify the building configuration, navigate with the up and down arrows of your keyboard and press [enter] to edit an option.

If you want to use the API with Processing, activate `PSMOVE_BUILD_JAVA_BINDINGS` and `PSMOVE_BUILD_PROCESSING_BINDIN`

To get camera tracking, activate:
`PSMOVE_BUILD_TRACKER` and `PSMOVE_USE_TRACKER_TRACE`

Then press [c] to configure, then [e] to come back to the configuration screen and finally [g] to generate the configuration settings (if the option does not show up on the first try, retry [c] then [e]).

Step 4.3: Building and checking

We are now back to the regular shell command. Type in:

```
cmake ..
make -j4
```

Make sure the controller you are pairing is plugged via USB to your computer then type in:

```
./example
```

The controller vibrates and the sphere blinks showing that the installation was successful.

Manual Pairing

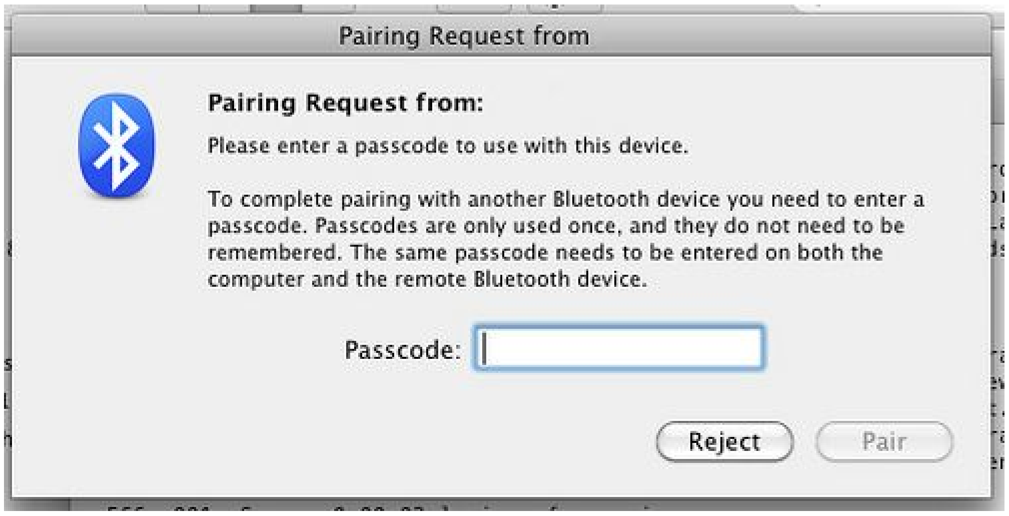
On some versions of MacOS 10.7, automated pairing may not work, but fear not! I'll show you how you can pair your controller manually.

Step 8.1: Retrieving MAC address

Right after you run ./psmovepair write down the adress you find after "`controller address:`" in the form "`aa:bb:cc:dd:ee:ff`"

Step 8.2: Trying connection

Check that the controller can't connect. Press the PS button, you get a PIN entry:



Pairing Request from

Pairing Request from:

Please enter a passcode to use with this device.

To complete pairing with another Bluetooth device you need to enter a passcode. Passcodes are only used once, and they do not need to be remembered. The same passcode needs to be entered on both the computer and the remote Bluetooth device.

Passcode:

Close the pin entry (click on "Reject").

Step 8.3: Shutting down BT

Disable Bluetooth (or the modifications that follow won't work).

Step 8.4: Waiting for BT process to quit

In the terminal, run :

```
pgrep blued
```

again and again until it does not print anything anymore (e.g. the process "blued" has quit) - this can take a few seconds up to a minute.

Step 8.5: Authorizing the controller's MAC address

Type in the following (this is just one line). Replace `"aa-bb-cc-dd-ee-ff"` by the controller address you wrote down earlier (note: lowercase and hyphen "-" as a separator).

```
sudo defaults write /Library/Preferences/com.apple.Bluetooth HIDDevices
-array-add "aa-bb-cc-dd-ee-ff"
```

The terminal asks for your root password.

Note: For security reasons, the terminal will not display what you type when entering passwords. Just type your password and press [enter]

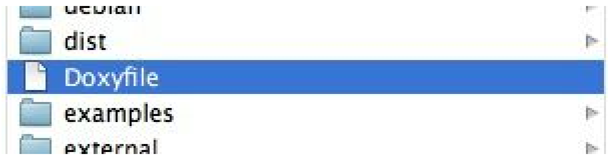
Step 8.6: No more PIN

Enable Bluetooth again then press the PS button on the controller. The PIN request should not pop up this time and the Move should now appear in the bluetooth devices as "Motion Controller".

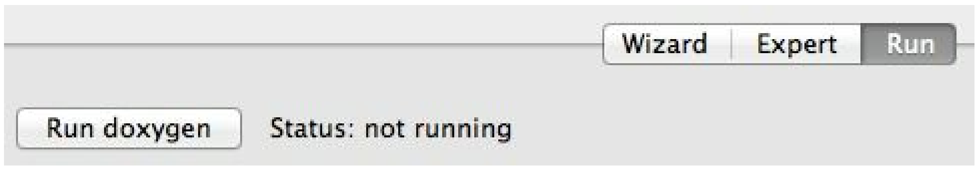
Generate documentation with Doxygen

Doxygen is a documentation system that takes the code from the api and automatically generate an html reference for all its functions. Download the latest binary distribution of Doxygen at: http://www.stack.nl/~dimitri/doxygen/download.html#latestsrc

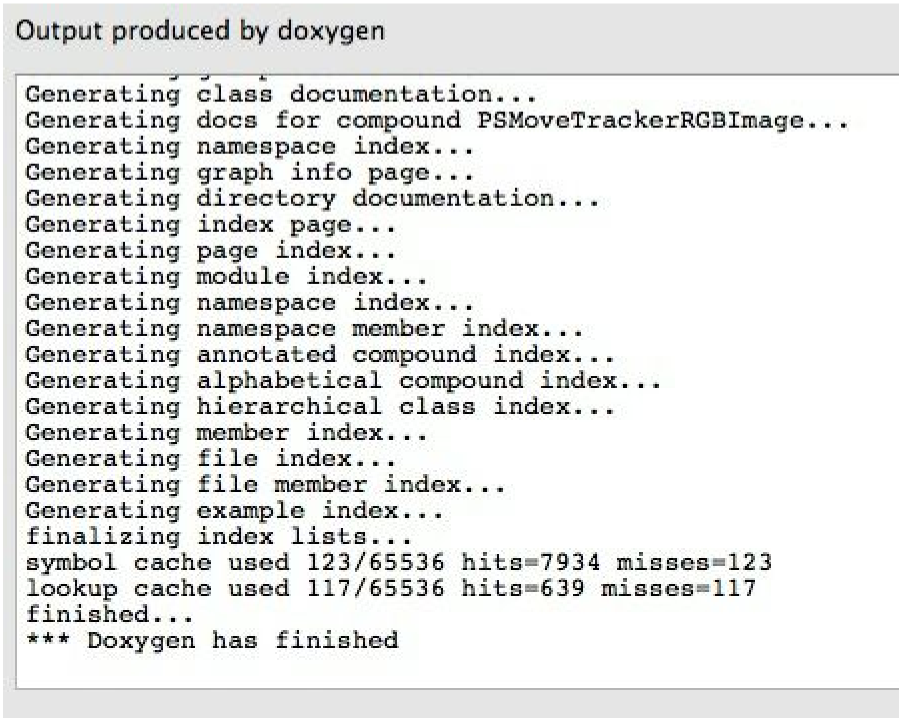
Run Doxygen.



Click File>Open and select the Doxyfile item in the psmoveapi folder which will load the configuration for the build.



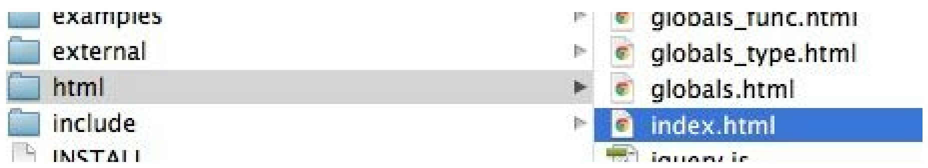
Click on the "Run" tab then on "Run Doxygen".



Output produced by doxygen

```
Generating class documentation...
Generating docs for compound PSMoveTrackerRGBImage...
Generating namespace index...
Generating graph info page...
Generating directory documentation...
Generating index page...
Generating page index...
Generating module index...
Generating namespace index...
Generating namespace member index...
Generating annotated compound index...
Generating alphabetical compound index...
Generating hierarchical class index...
Generating member index...
Generating file index...
Generating file member index...
Generating example index...
finalizing index lists...
symbol cache used 123/65536 hits=7934 misses=123
lookup cache used 117/65536 hits=639 misses=117
finished...
*** Doxygen has finished
```

Once Doxygen has finished, quit the app and go to your psmoveapi folder. You will find a folder called html which contains the doc.



Open index.html in your favorite web browser to access the documentation.

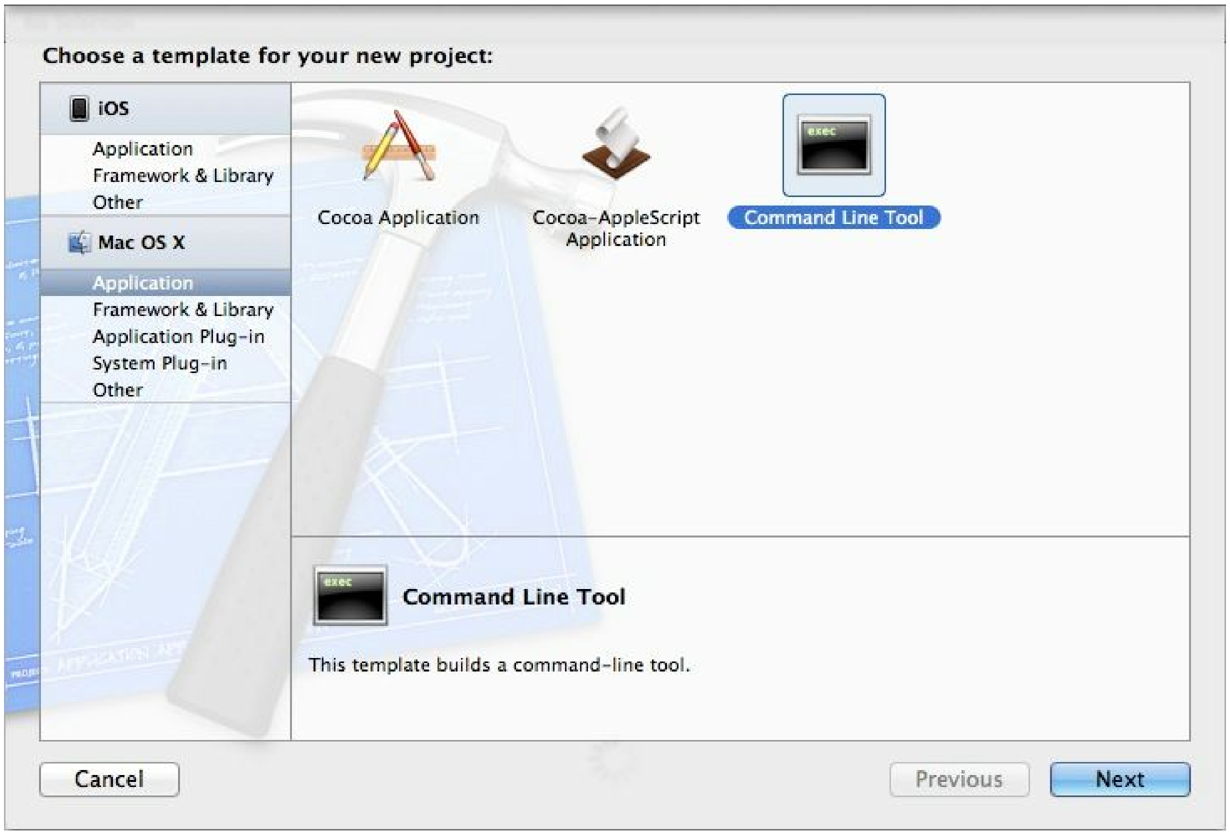
---

2) Creating a C project linked to the API in Xcode

Note: this part has not been updated in a while and might be slightly outdated. It will eventually be moved to an independent document. You are welcome to fork this and improve on it. Please post on the mailing list if you do.

Open Xcode.

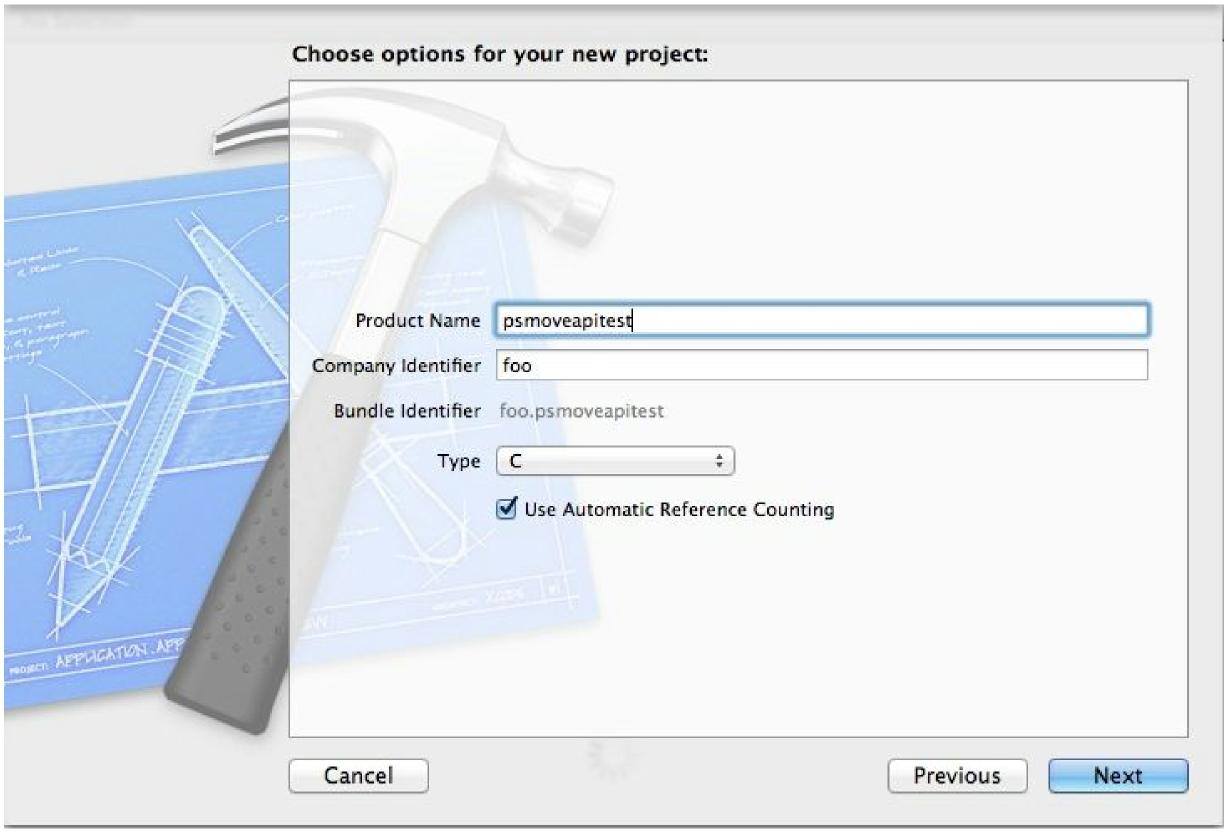
Go to File>New>New Project...


Choose a template for your new project:

Command Line Tool

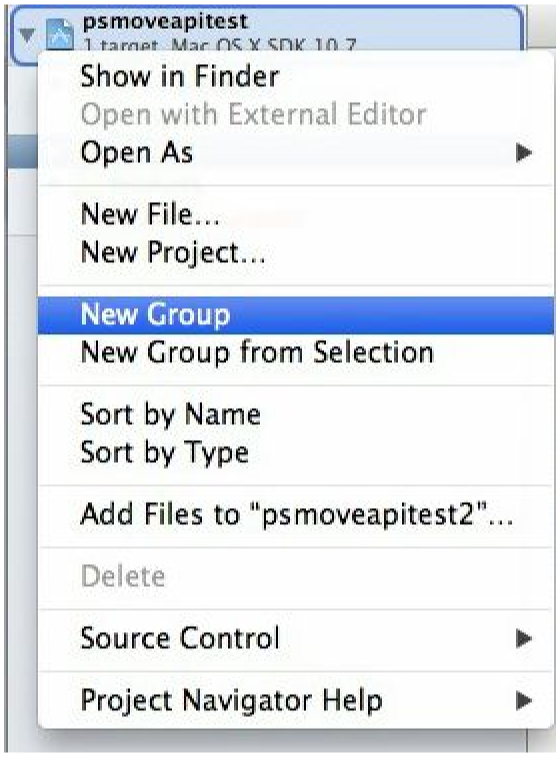
This template builds a command-line tool.

Under Mac OS X, click on “Application”.
From the list of templates, choose “Command Line Tool” and click “Next”.


Choose options for your new project:

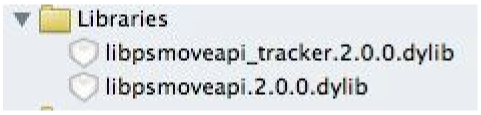
Product Name: psmoveapitest
Company Identifier: foo
Bundle Identifier: foo.psmoveapitest
Type: C
[✓] Use Automatic Reference Counting

Give your project any name you want. For example “psmoveapitest”.
Chose the language of the project from the drop-down menu : **C**

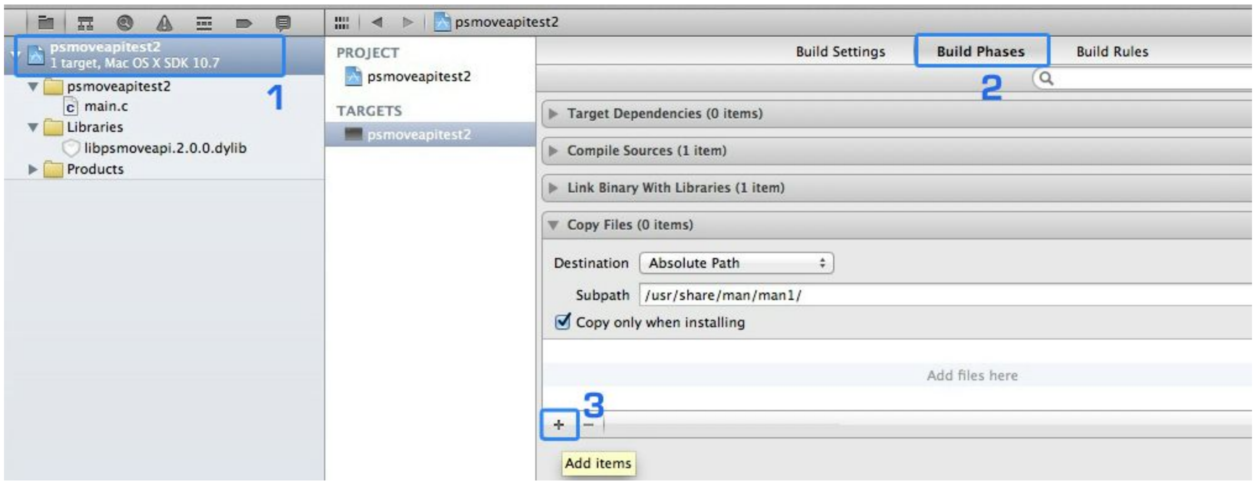


Create a new group in the left panel and name it "Libraries" for example

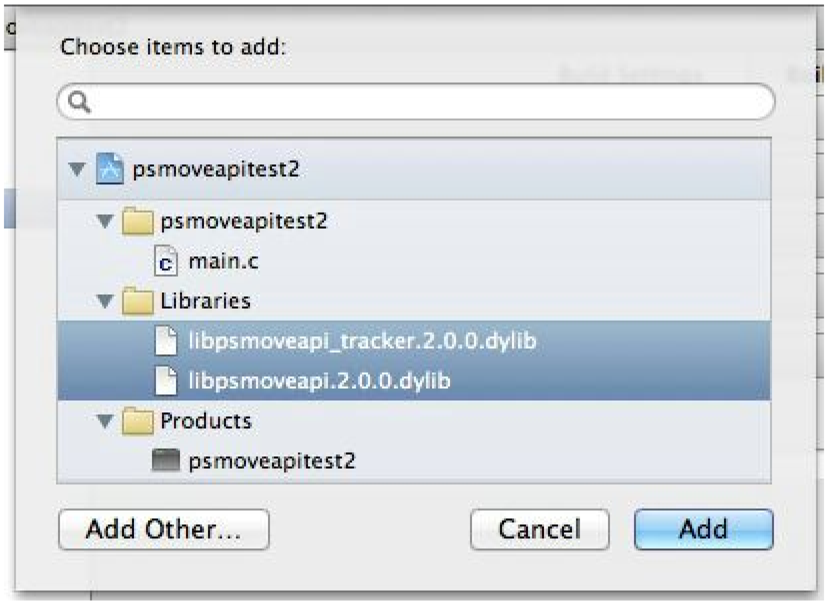
In the finder, open the /psmoveapi folder and look in /build for two dynamic library files named "libpsmoveapi.2.0.0.dylib" and "libpsmoveapi_tracker.2.0.0"



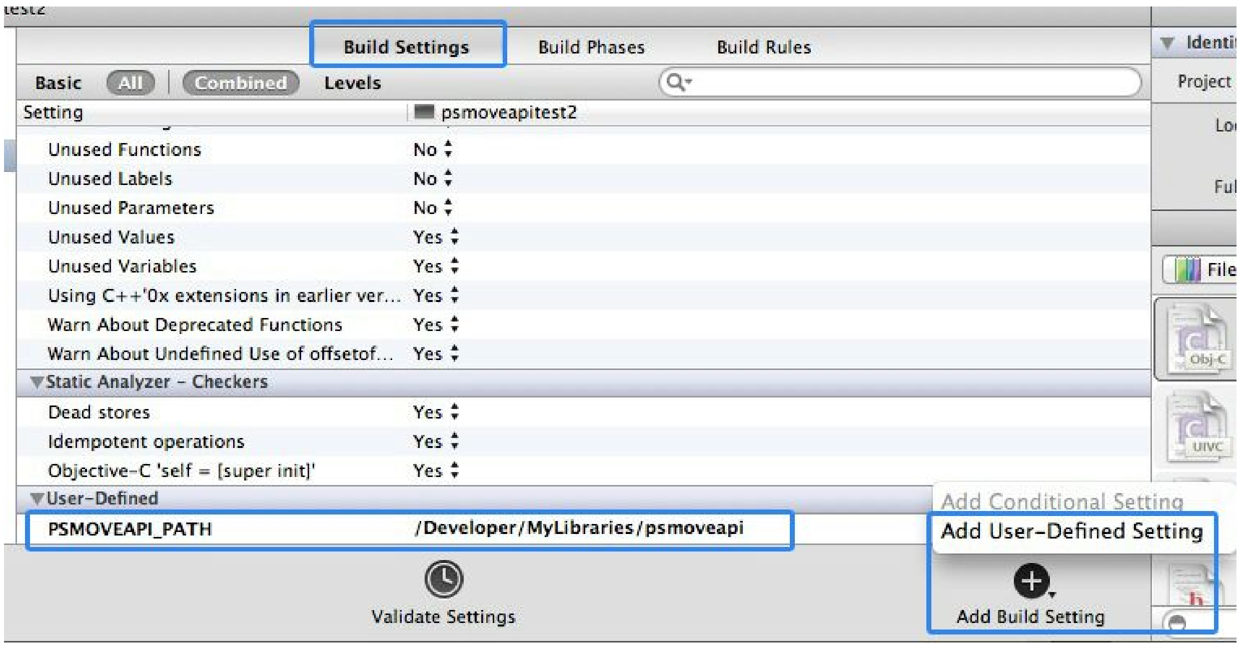
Drag the files to the "Libraries" folder you created in Xcode.
In the confirmation window, **do not** tick the "Copy items into destination's group folder" option and click "Finish"



Click on the project name and find the "Build Phases" tab.
Click on the triangle next to "Copy Files" then on the **+** sign at the bottom left.

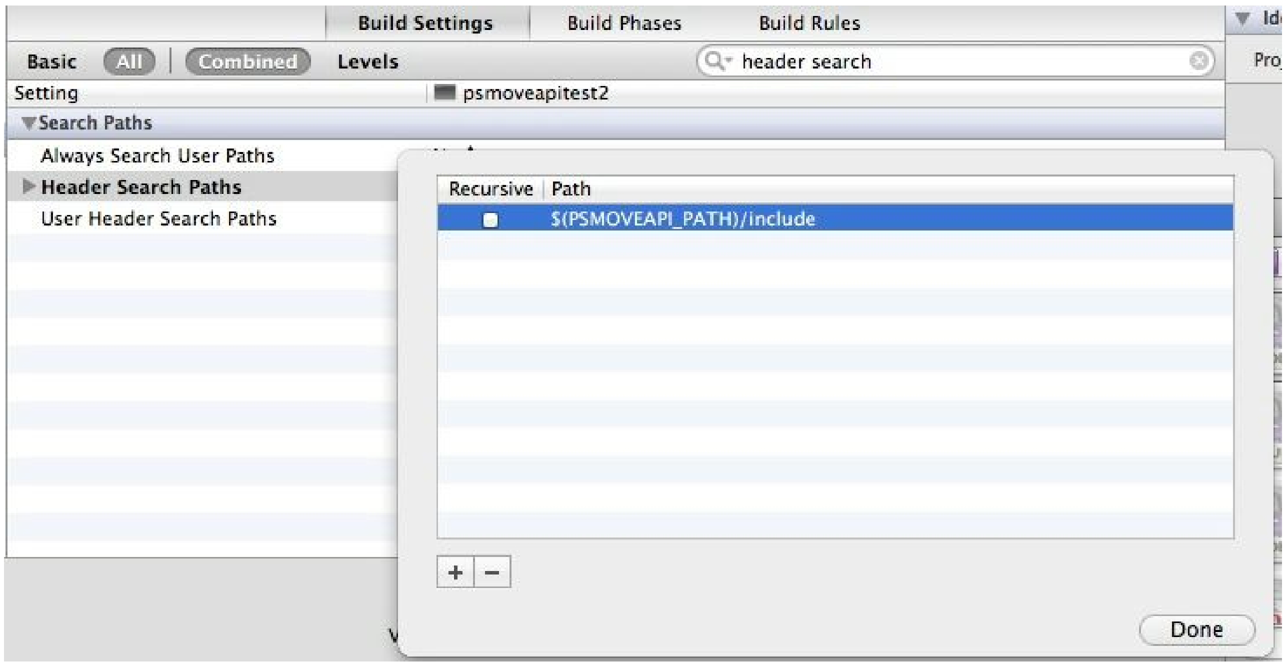


A window opens and asks to "Choose items to add"
Select all the dylibs in “Libraries” then click "Add".



Setting	psmoveapitest2
Unused Functions	No
Unused Labels	No
Unused Parameters	No
Unused Values	Yes
Unused Variables	Yes
Using C++'0x extensions in earlier ver...	Yes
Warn About Deprecated Functions	Yes
Warn About Undefined Use of offsetof...	Yes
▼Static Analyzer – Checkers	
Dead stores	Yes
Idempotent operations	Yes
Objective-C 'self = [super init]'	Yes
▼User-Defined	
PSMOVEAPI_PATH	/Developer/MyLibraries/psmoveapi

Now click on the "Build Settings" tab.
Click on "Add Build Settings" (in earlier versions of Xcode, click on the gear icon at the bottom left)
Then click on "Add User-Defined Setting".
Type in "PSMOVEAPI_PATH" (without the quotes) for the setting's name and the path of the folder where you installed the lib. In my case : /Developer/MyLibraries/psmoveapi



Recursive	Path
☐	$(PSMOVEAPI_PATH)/include

In the search field, type "header search"
Add "$(PSMOVEAPI_PATH)/include" to the "Header Search Paths" setting.

If you’re going to use OpenCV, also add “$(PSMOVEAPI_PATH)/opencv/build/install/include”.
For TUIO, add “$(PSMOVEAPI_PATH)/external/TUIO_CPP” and select “recursive”.
Click “Done”. (in Xcode 4.5, just click outside of the popup window)

In the "main.c", add the lines

```
#include "psmove.h"
#include "psmove_tracker.h"
```

Build and run to check everything is ok.

You can start by pasting the contents of example.c or tracker.c over the contents of your main.c and compile it then play around with the code.

You’re done!

Get some inspiration from the files in the /psmoveapi/examples/c/ folder on how to use the API and read the doc for details on the methods included. Some of the examples will require a little tweaking in Xcode before they compile. I’ll update this tutorial as soon as I figure it out. Feel free to share your experience on the mailing list (https://lists.ims.tuwien.ac.at/mailman/listinfo/psmove).