Pre-requisites

* At the time of successful build, the Mac OS version is 10.11.6
* Cmake version used is 3.4.2
* If you have installed python on your mac (apart from the one that ships by default on OS X which is 2.7), I would suggest to do a clean install of the OS before as Panda 3D has been compiled to use the default version that ships with OS X. You might run into python errors while running the main.py if you have other python versions. An alternative is to point Panda3D to the python location it should use or keep the default python directories untouched. However, at the time I did not find a step by step guide to do so.
* Panda 3D version used is 1.9.2-MacOSX10.7.dmg. Using version 1.9.2-MacOSX 10.6 produced errors. [Here is the link](https://www.panda3d.org/download.php?platform=macosx&version=1.9.2&sdk).
* Xcode version used is 8.0.
* Python, Panda3d and Mac OS versions are crucial for successful build.

Compiling and Generating SMR with Cmake

* Download the latest SMR repository from [here](https://github.com/alexisheloir/smr-motion).
* Set the source directory in Cmake as smr-motion-master/smr as shown in Step 1 below.  
    
  
* Set the build directory as smr-motion-master/macBuild as shown in Step 2.  
    
  
* Click on Configure as shown in Step 3 and make sure the generator selected is Xcode.  
    
  
* Uncheck the build\_shared\_libs option as shown in Step 4.  
    
  
* Make sure X11 is installed on your Mac as it will be required to proceed. Here is the [link](https://www.xquartz.org/) to XQuartz. OS X comes installed with X11. So, there shouldn’t be an issue.
* Set the Executable output path as smr-motion-master/macbuild/bin.   
  Set the library output path as smr-motion-master/macbuild/lib as shown in Step 5.  
    
    
  
* You should get a number of unconfigured options for the X11. Scroll down to find them as shown in Step 6. If you do not see X11 options, make sure ‘**Advanced’ option** is checked in Cmake as shown. Also, make sure the settings are entered as mentioned in the section **Reference 1** below
* After entering the include paths and library paths, you will get a screen as shown in Steps 7-10.  
    
    
    
    
    
    
  
* Click on Configure again as shown in Step 10.
* The resulting screen should look like the one shown in Step 11.  
    
  
* Now, Click on Generate and ignore the warning if it states - MACOSX\_RPATH is not specified for Horde and Horde3dUtils as shown in Step 12.  
    
  
* Open Samsara.xcode project file which will be in macbuild/bin directory as shown in Step 13.  
    
  
* Build log4cplusS for Profiling Scheme (Under Product → Build for → Profiling) as shown in Steps 14 and 15.  
    
    
  
* Build core, then math, then io **in that order** as shown in Steps 16-21.  
    
    
    
    
    
    
    
    
  

  
  


**Note :**

* Unchecking the LOG4CPLUS\_ENABLE\_DECORATED\_LIBRARY\_NAME in SMR results in a number of issues. Therefore, dynamic library is necessary.
* I built all the libraries for both debug and release versions.
* It is important to build SMR correctly in order to compile EMBR without any issues.
* In case, you face any other issue while compiling SMR, please refer to a copy of the cache settings that have been included, here in this directory, on google drive.

**EMBR Compilation process**

Download the EMBR repository from [here](https://github.com/alexisheloir/embr-character-animation-engine).

* Follow the procedure for compiling boost libraries explained in section 5.2.1 and 5.2.4 in index.html file (Find this under boost\_1\_60\_0/more/getting started/ ). The process is illustrated in Steps 1 - 6 below.
*   
    
    
    
  
* Open Cmake and set the source as *embr-motion-master/trunk* as shown in Step 7.  
    
  
* Set the location to build binaries as *embr-motion-master/trunk/build/macbuild/bin* as shown in Step 8.  
    
  
* Click on Configure and Click OK if it says error in configuration files as shown in Step 9.  
    
  
* Set the boost\_dir and boost\_include\_dir as *embr-character-animation-engine/TRUNK/thirdParty/universal/boost\_1\_60\_0* as shown in Step 10.  
    
  
* Set the boost\_libs\_dir as *embr-character-animation-engine/TRUNK/thirdParty/universal/boost\_1\_60\_0/stage/lib* as shown in Step 11.(If you don’t have the stage directory, you have probably missed the steps in section 5.2.4 of the Boost library compilation process)  
  
* Set the game path as *embr-character-animation-engine-master/TRUNK/build/macBuild/game* as shown in Step 12.  
    
  
* Set SMR home directory as smr-motion-master/smr as shown in Step 13.



* Set Samsara Source\_Dir as smr-motion-master/smr/SRC as shown in Step 14 and click on Configure.  
    
  
* Uncheck the BUILD\_SHARED\_LIBS setting as shown in Step 15.
* Set the library output path as macbuild/lib.
* **Very important setting** - Uncheck the following -  
  LOG4CPLUS\_BUILD\_LOGGINGSERVER, LOG4CPLUS\_BUILD\_TESTING and LOG4CPLUS\_ENABLE\_DECORATED\_LIBRARY\_NAME.   
    
  
* Click on Configure. If you get an error saying configuration is invalid, make sure the paths for X11 libraries are properly configured as mentioned in **Reference 1**. Follow the exact steps as shown in Steps 7 - 10 of building SMR.
* You would get a warning that MacOSX\_RPATH is not specified for SMRPy but that does not prevent the xcode project from being generated as shown in Step 16.  
    
  
* Open EMBR.xcode project file and you should be able to compile debug version of SMRPy as shown in Step 17. Debug version can be compiled by hitting the button marked 2 in the figure below. The Release version can also be compiled by going to Product → Build for → Profiling option. However, the release version has threading issues which would cause the application to terminate. At this point, the issue still exists with the release version of SMRPy.  
  
* Once libSMRPy.dylib is compiled, it can be used as a Python library provided it is renamed as SMRPy.so as shown in Step 18.
* Double click on libSMRPy.dylib in the macbuild/lib/debug folder to compile the library.
* After this, copy the libSMRPy.dylib in BenSignLanguage directory. Rename it as SMRPy.so and make sure to change the file extension.
* Open up the terminal. Navigate to BenSignLanguage directory. Run the main.py using the command - *python main.py.*
* Debug version works fine but the release version crashes when the realizer thread initializes.
* If you face any issues while building EMBR, please look at the Cmake cache settings that are included in this directory on google drive.

Reference 1 (will be used for both SMR and EMBR)

//Path to a file.  
X11\_ICE\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_ICE\_LIB:FILEPATH=/usr/X11R6/lib/libICE.dylib  
  
//Path to a file.  
X11\_SM\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_SM\_LIB:FILEPATH=/usr/X11R6/lib/libSM.dylib  
  
//Path to a file.  
X11\_X11\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_X11\_LIB:FILEPATH=/usr/X11R6/lib/libX11.dylib  
  
//Path to a file.  
X11\_XRes\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_XRes\_LIB:FILEPATH=/usr/X11R6/lib/libXRes.dylib  
  
//Path to a file.  
X11\_XShm\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a file.  
X11\_XSync\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a file.  
X11\_XTest\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_XTest\_LIB:FILEPATH=/usr/X11R6/lib/libXtst.dylib  
  
//Path to a file.  
X11\_Xaccessrules\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a file.  
X11\_Xaccessstr\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a file.  
X11\_Xau\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_Xau\_LIB:FILEPATH=/usr/X11R6/lib/libXau.dylib  
  
//Path to a file.  
X11\_Xcomposite\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_Xcomposite\_LIB:FILEPATH=/usr/X11R6/lib/libXcomposite.dylib  
  
//Path to a file.  
X11\_Xcursor\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_Xcursor\_LIB:FILEPATH=/usr/X11R6/lib/libXcursor.dylib  
  
//Path to a file.  
X11\_Xdamage\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_Xdamage\_LIB:FILEPATH=/usr/X11R6/lib/libXdamage.dylib  
  
//Path to a file.  
X11\_Xdmcp\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_Xdmcp\_LIB:FILEPATH=/usr/X11R6/lib/libXdmcp.dylib  
  
//Path to a library.  
X11\_Xext\_LIB:FILEPATH=/usr/X11R6/lib/libXext.dylib  
  
//Path to a file.  
X11\_Xfixes\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_Xfixes\_LIB:FILEPATH=/usr/X11R6/lib/libXfixes.dylib  
  
//Path to a file.  
X11\_Xft\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_Xft\_LIB:FILEPATH=/usr/X11R6/lib/libXft.dylib  
  
//Path to a file.  
X11\_Xi\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_Xi\_LIB:FILEPATH=/usr/X11R6/lib/libXi.dylib  
  
//Path to a file.  
X11\_Xinerama\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_Xinerama\_LIB:FILEPATH=/usr/X11R6/lib/libXinerama.dylib  
  
//Path to a file.  
X11\_Xinput\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_Xinput\_LIB:FILEPATH=/usr/X11R6/lib/libXi.dylib  
  
//Path to a file.  
X11\_Xkb\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a file.  
X11\_Xkbfile\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_Xkbfile\_LIB:FILEPATH=/usr/X11R6/lib/libxkbfile.dylib  
  
//Path to a file.  
X11\_Xkblib\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a file.  
X11\_Xlib\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a file.  
X11\_Xmu\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_Xmu\_LIB:FILEPATH=/usr/X11R6/lib/libXmu.dylib  
  
//Path to a file.  
X11\_Xpm\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_Xpm\_LIB:FILEPATH=/usr/X11R6/lib/libXpm.dylib  
  
//Path to a file.  
X11\_Xrandr\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_Xrandr\_LIB:FILEPATH=/usr/X11R6/lib/libXrandr.dylib  
  
//Path to a file.  
X11\_Xrender\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_Xrender\_LIB:FILEPATH=/usr/X11R6/lib/libXrender.dylib  
  
//Path to a file.  
X11\_Xscreensaver\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_Xscreensaver\_LIB:FILEPATH=/usr/X11R6/lib/libXss.dylib  
  
//Path to a file.  
X11\_Xshape\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a file.  
X11\_Xt\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_Xt\_LIB:FILEPATH=/usr/X11R6/lib/libXt.dylib  
  
//Path to a file.  
X11\_Xutil\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a file.  
X11\_Xv\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a library.  
X11\_Xv\_LIB:FILEPATH=/usr/X11R6/lib/libXv.dylib  
  
//Path to a library.  
X11\_Xxf86misc\_LIB:FILEPATH=/usr/X11R6/lib/libXxf86misc.dylib  
  
//Path to a library.  
X11\_Xxf86vm\_LIB:FILEPATH=/usr/X11R6/lib/libXxf86vm.dylib  
  
//Path to a file.  
X11\_dpms\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a file.  
X11\_xf86misc\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a file.  
X11\_xf86vmode\_INCLUDE\_PATH:PATH=/usr/X11R6/include  
  
//Path to a file.  
ZLIB\_INCLUDE\_DIR:PATH=/usr/include