

iOS XCode compilation and export to devices

Those are the steps you need to follow to be able to build your game make with GDE for any iOS device.

Download the necessary elements to start.

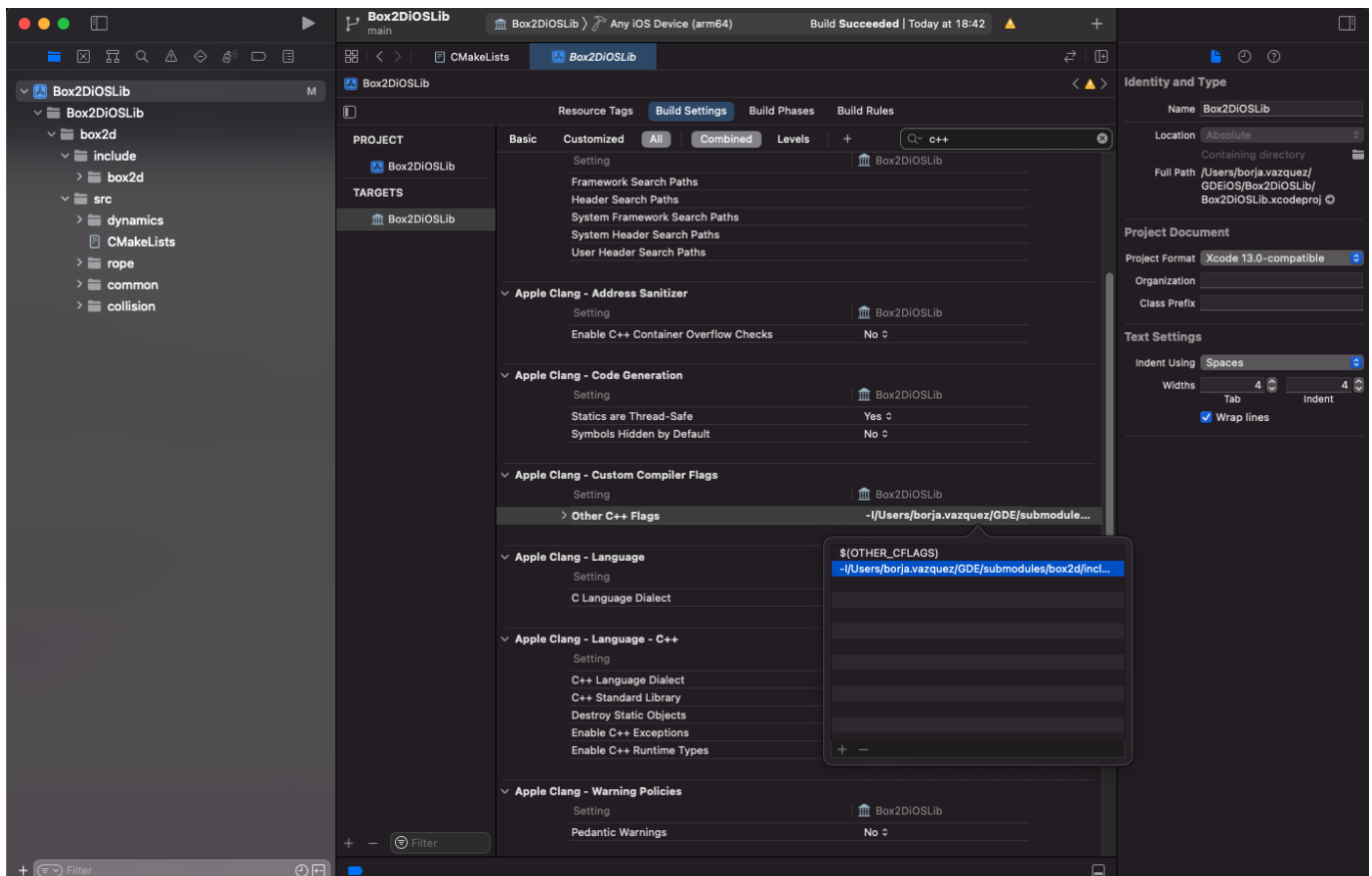
Clone <https://github.com/bovacu/GDEiOS> in the same directory as your GDE core.

Submodules

Now we need to reconfigure all of the include paths of the submodules. It will be the same thing for all of them. Enter to GDEiOS.

Box2D

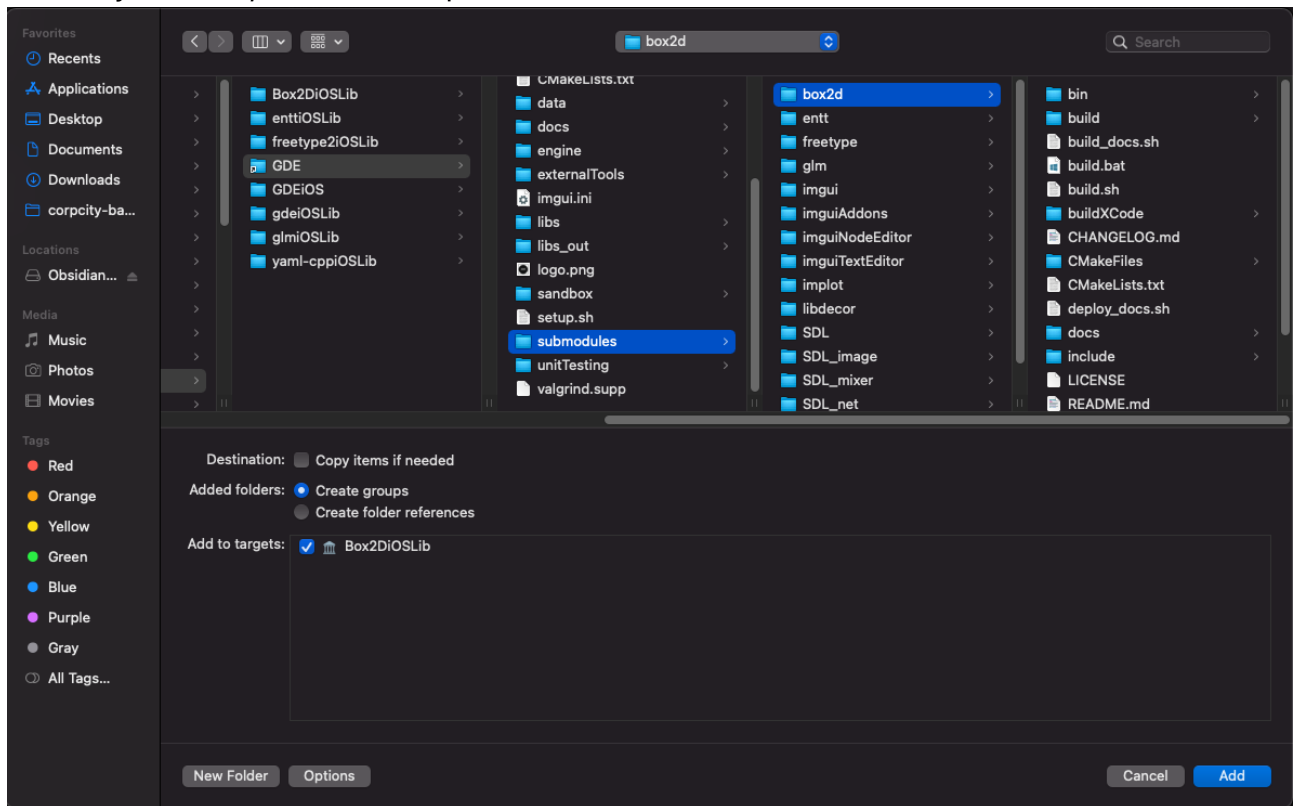
Enter to Box2DiOSLib and open the XCode project, you will see this:



If any folder is in red:

- Right click on box2d folder and delete -> remove reference.

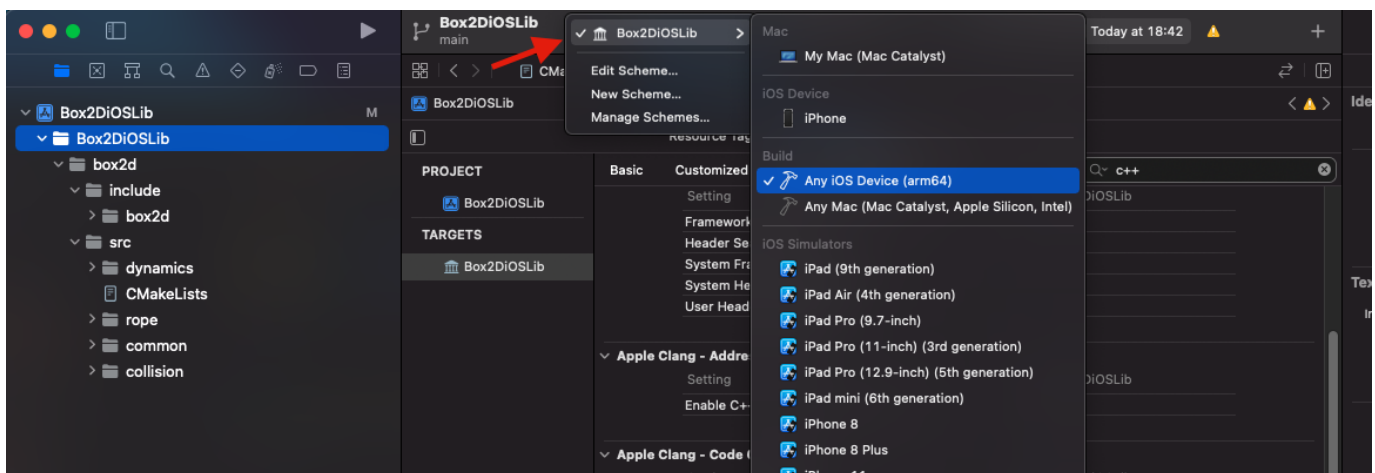
- Right click on Box2DiOSLib (gray folder) -> Add files to "Box2DiOSLib".
- Locate your GDE/submodules path and then select box2d.



- Remember to tick Add to Target and select Create Groups.
- Click on Add.
- Now a lot of things have been imported, but we only need the *.c and .h*
- Remove EVERYTHING else (CMakeLists.txt files can stay, but are optional)

Set compilation target

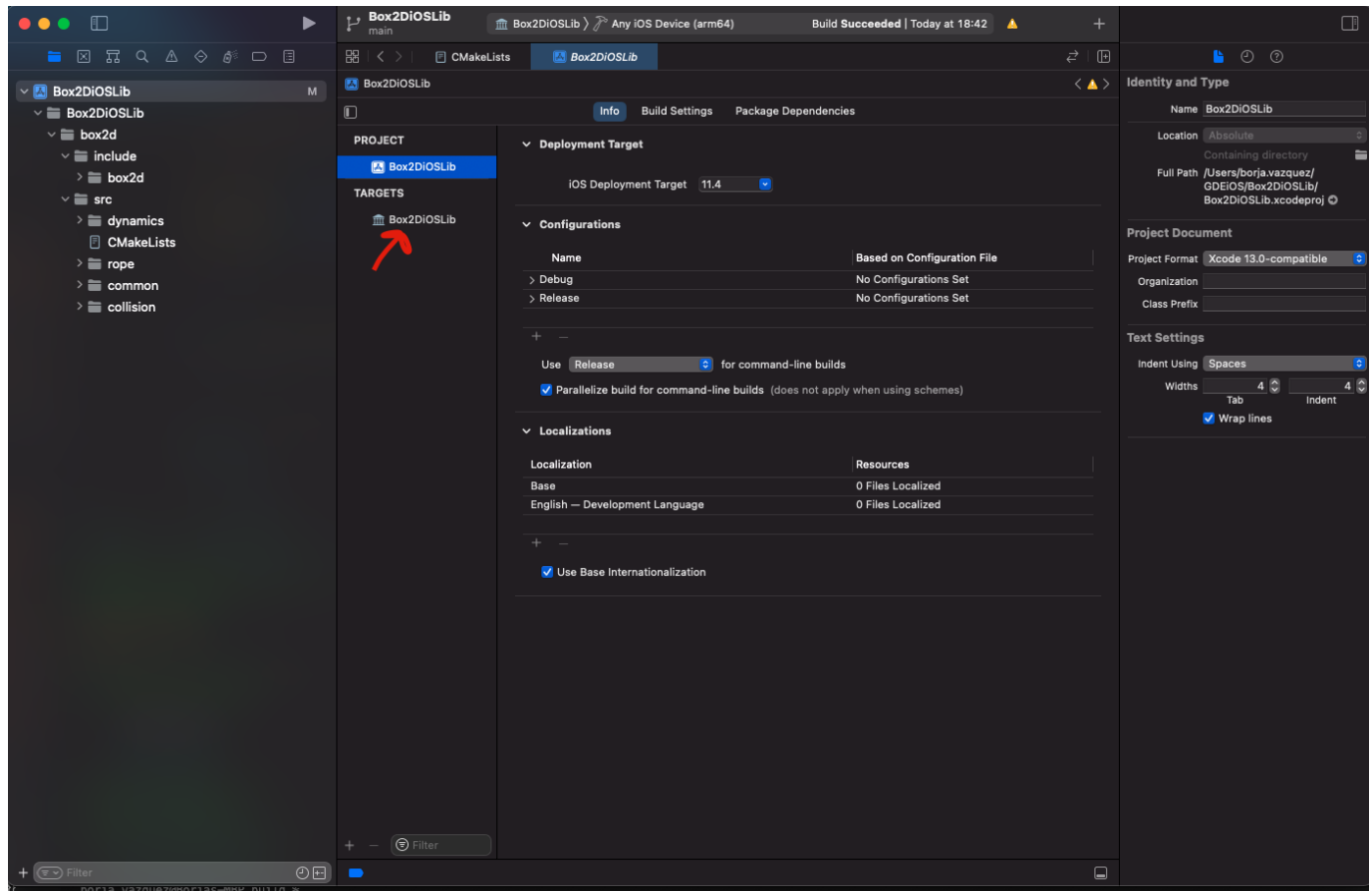
Now we will tell XCode what to compile exactly:



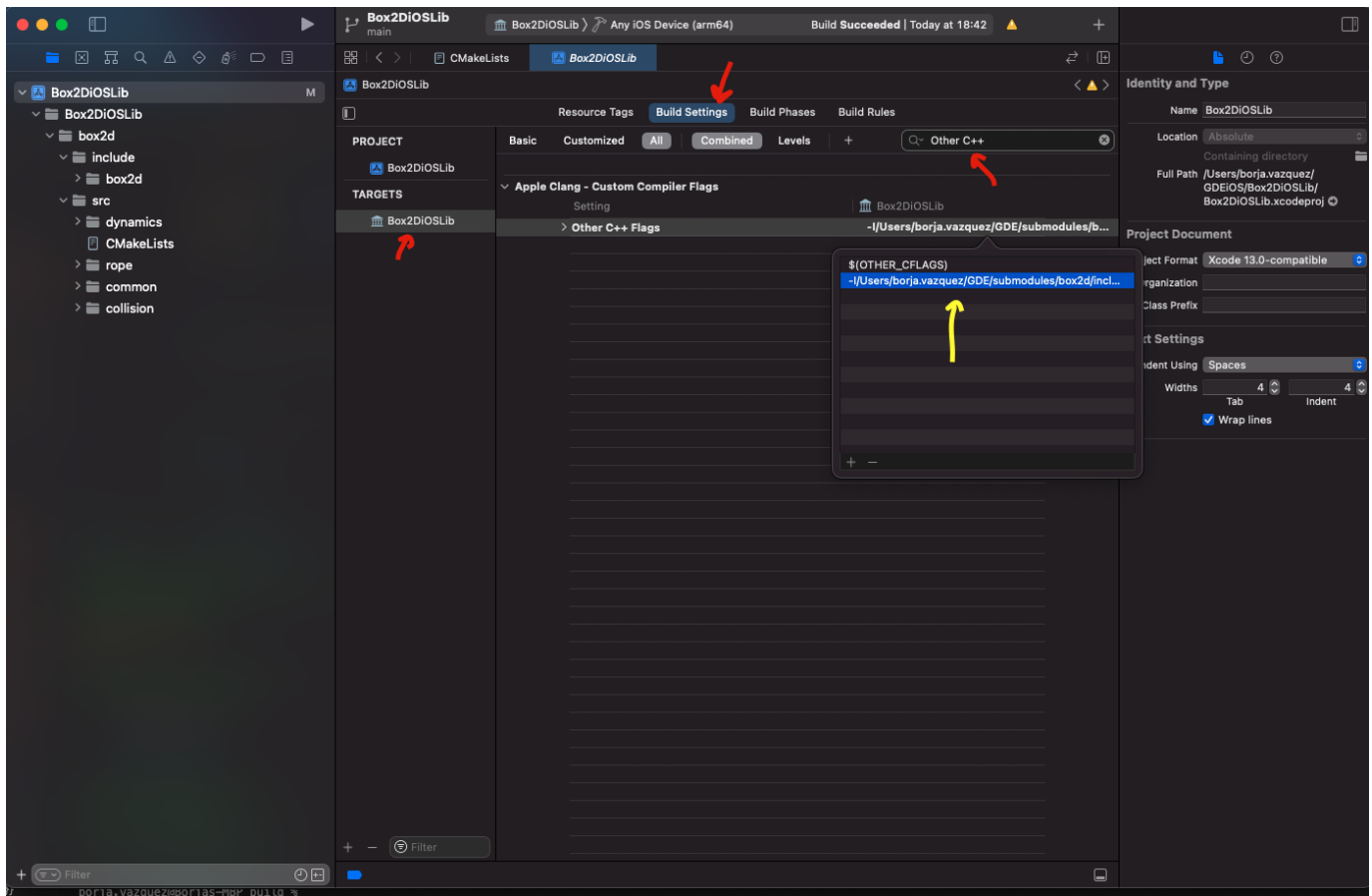
In this list, select any option on the Simulator section. I always use iPhone 8 Plus, but is irrelevant.

Now, if you press Command + B it will start compiling (or press the play button on top left), but it will fail, this is because XCode doesn't know where to find the includes of the library (eventhough we have put them in the directory... anyways).

Press on the blue Box2DiOSLib, this will open this panel:



Select the target that is marked with the arrow. And on the above bar, select 'Building Settings'. Then search for 'Other C++ Flags':

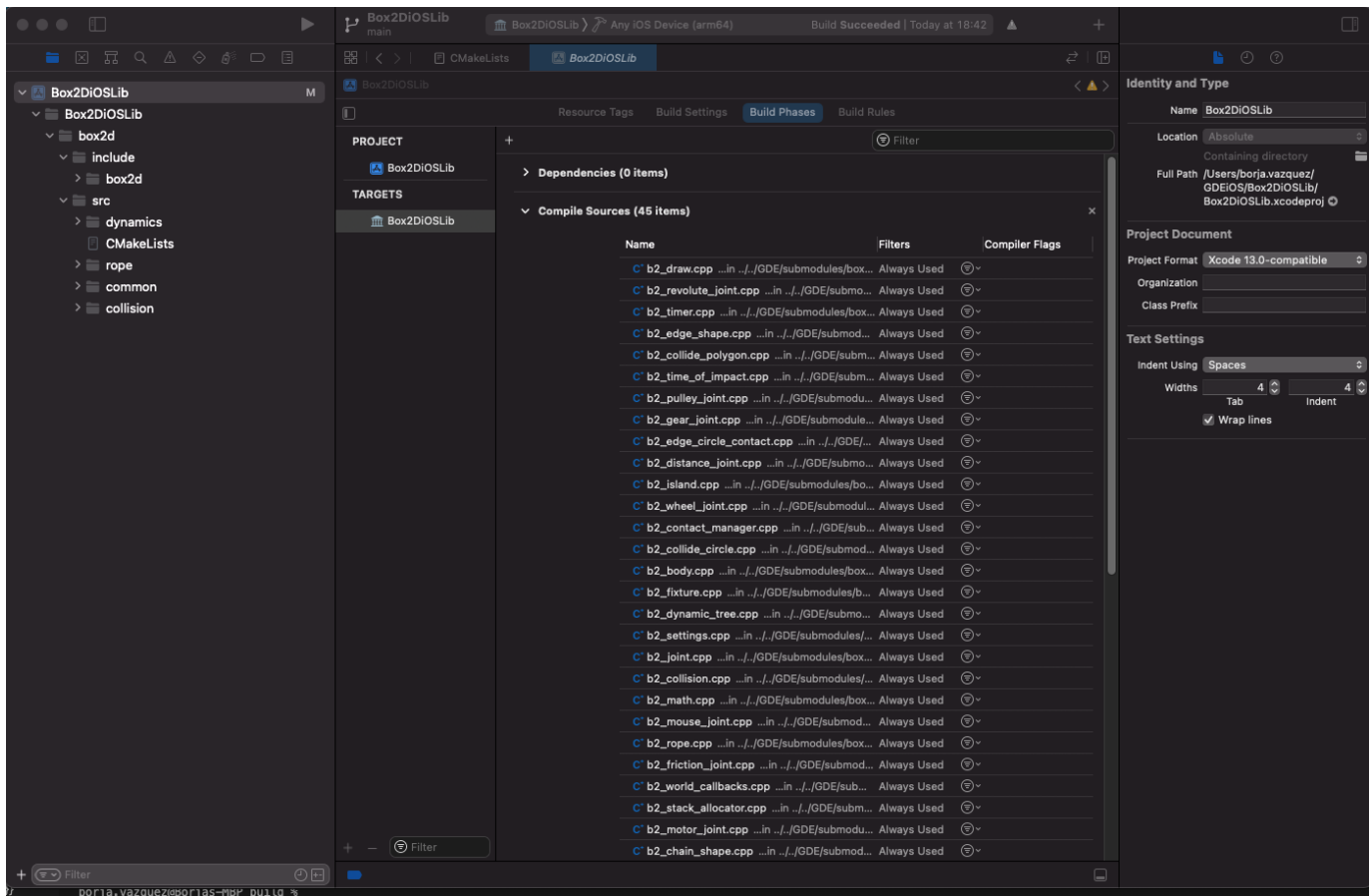


Now, the most important arrow here is the yellow one. Here you need to write

```
-I PATH_TO_GDE_SUBMODULES_BOX2D_INCLUDE
```

In my case it was `-I/Users/borja.vazquez/GDE/submodules/box2d/include`.

Now as you went to 'Building Settings', go to 'Building Phases' and check that Compile Sources has all of the sources we add in the first steps, under Box2DiOSLib/box2d



Now if you compile again, it will say compilation success or something similar. And with this, we are done with box2d.

Freetype

For this one follow the same exact steps as for Box2D. In this case, in 'Build Settings' -> Other C++ Flags, the path must point to freetype/include.

GLM

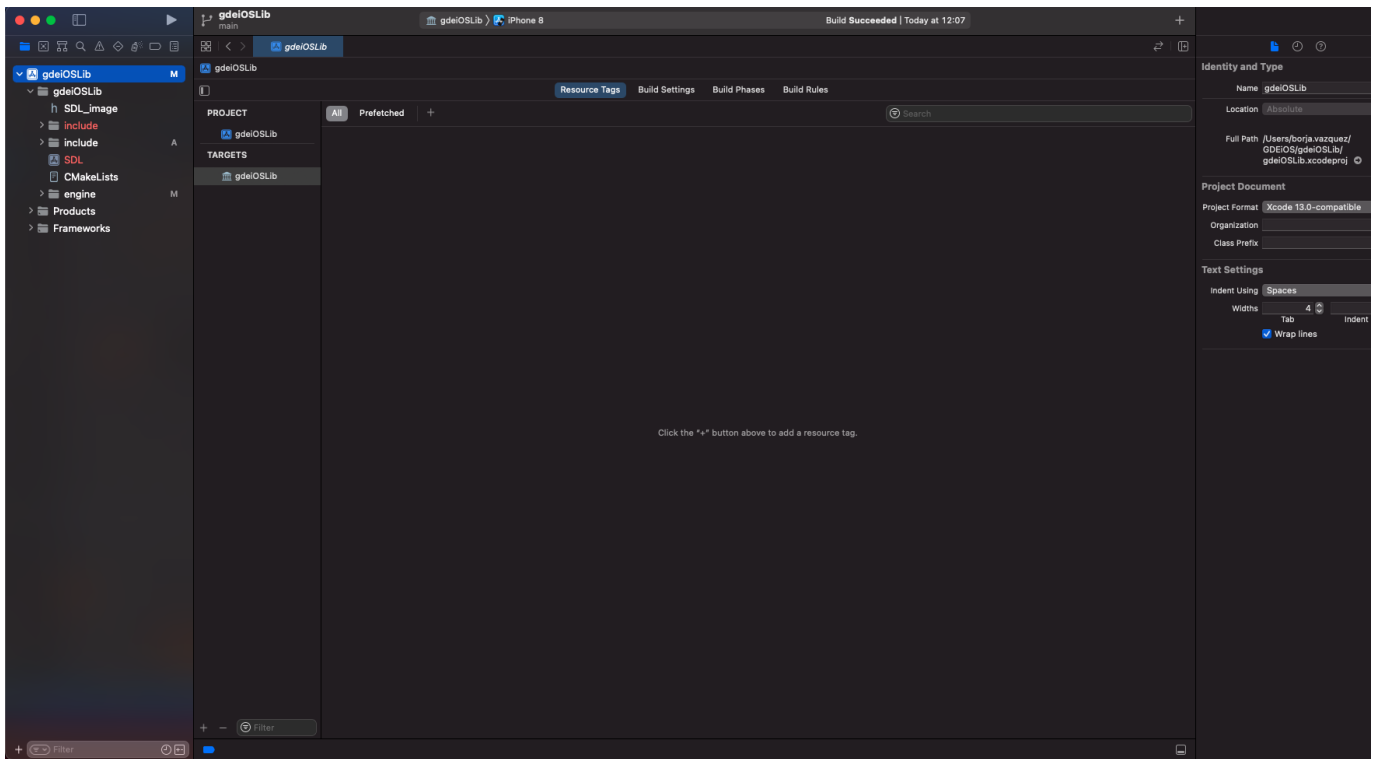
For this one follow the same exact steps as for Box2D. In this case, in 'Build Settings' -> Other C++ Flags, the path must point to glm. There is no include in this one.

YAML-CPP

For this one follow the same exact steps as for Box2D. In this case, in 'Build Settings' -> Other C++ Flags, the path must point to yaml-cpp/include.

GDE

This is the trickiest one to get right, but I think won't return any errors, hopefully.



You will probably have as much or more errors than I do. So let's fix them.

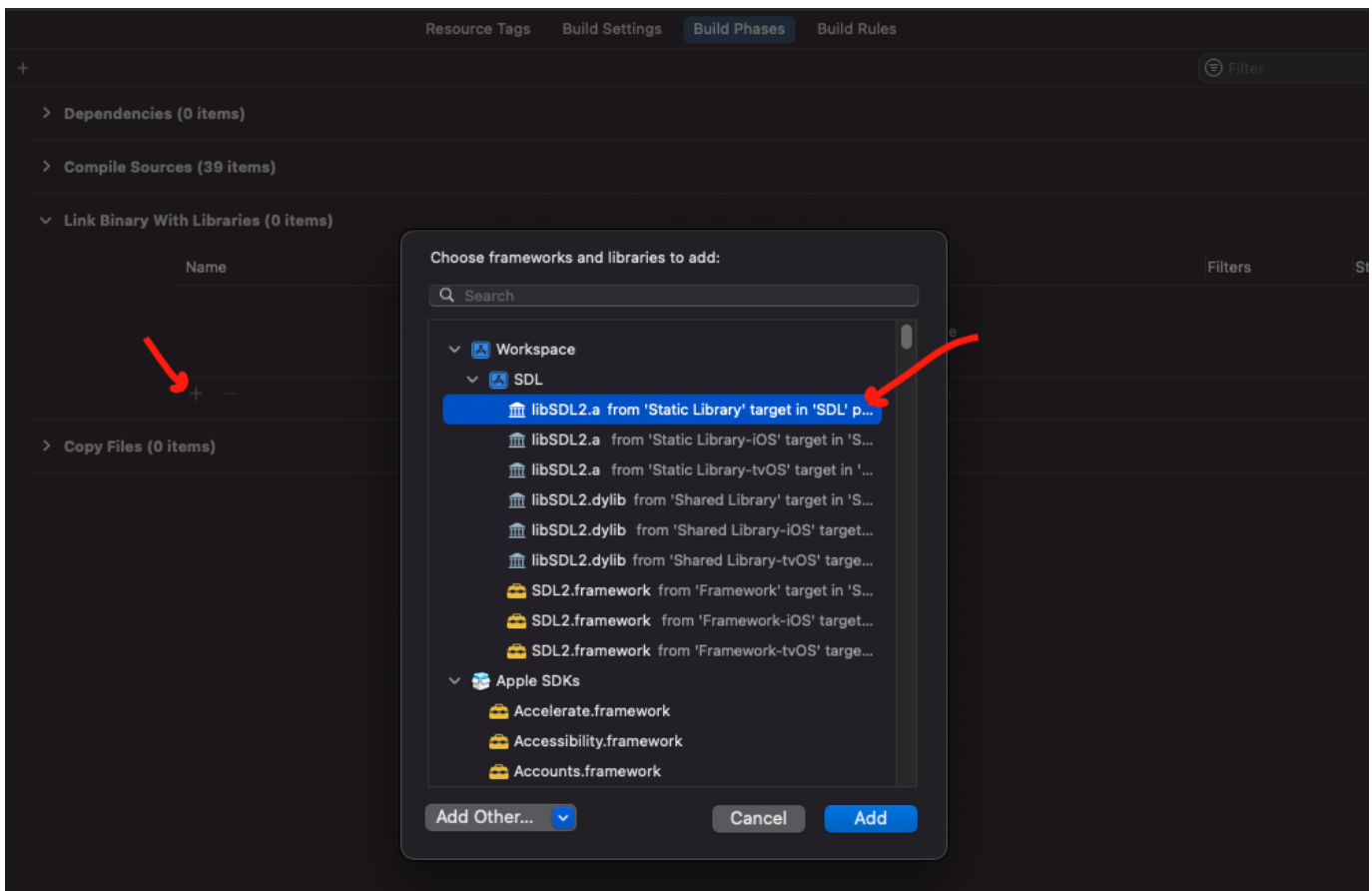
- If engine is in red:
Delete engine reference. Then gdeiOSLib -> Add Files to 'gdeiOSLib' and search for GDE/engine. For the options same thing as with Box2D.
- If SDL_image is in red:
Delete SDL_image.h. Then gdeiOSLib -> Add Files to 'gdeiOSLib' and locate GDE/submodules/SDL_image/include/SDL_image.h. Add it.
- If include of SDL_mixer is in red:
Delete include. Then gdeiOSLib -> Add Files to 'gdeiOSLib' and locate GDE/submodules/SDL_mixer/include. Add it.
- If include of SDL is in red:
Delete include. Then gdeiOSLib -> Add Files to 'gdeiOSLib' and locate GDE/submodules/SDL/include. Add it.
- If SDL is in red:
Delete SDL reference. Then gdeiOSLib -> Add Files to 'gdeiOSLib' and locate GDE/submodules/SDL/XCode/SDL/SDL.xcodeproj

Okay, now we should good to go.

Now for the includes you will need to add (with your path, in 'Building Settings') all of this:

```
$(OTHER_CFLAGS)
-I/Users/borja.vazquez/GDE/engine/include
-I/Users/borja.vazquez/GDE/submodules/entt/src
-I/Users/borja.vazquez/GDE/submodules/yaml-cpp/include
-I/Users/borja.vazquez/GDE/submodules/glm
-I/Users/borja.vazquez/GDE/submodules/freetype/include
-I/Users/borja.vazquez/GDE/submodules/box2d/include
-I/Users/borja.vazquez/GDE/engine/submodules/SDL/include
-I/Users/borja.vazquez/GDE/engine/submodules/SDL_image
-I/Users/borja.vazquez/GDE/engine/submodules/SDL_mixer/include
-I/Users/borja.vazquez/GDE/engine/submodules/SDL/src
```

Last but not least, head to 'Build Phases' and go to Link Binary with Libraries. As far as I know I don't know if this step is 100% necessary, but I'm doing it just in case.



Lastly, go to gdeiOSLib /engine and remove reference for main.cpp.

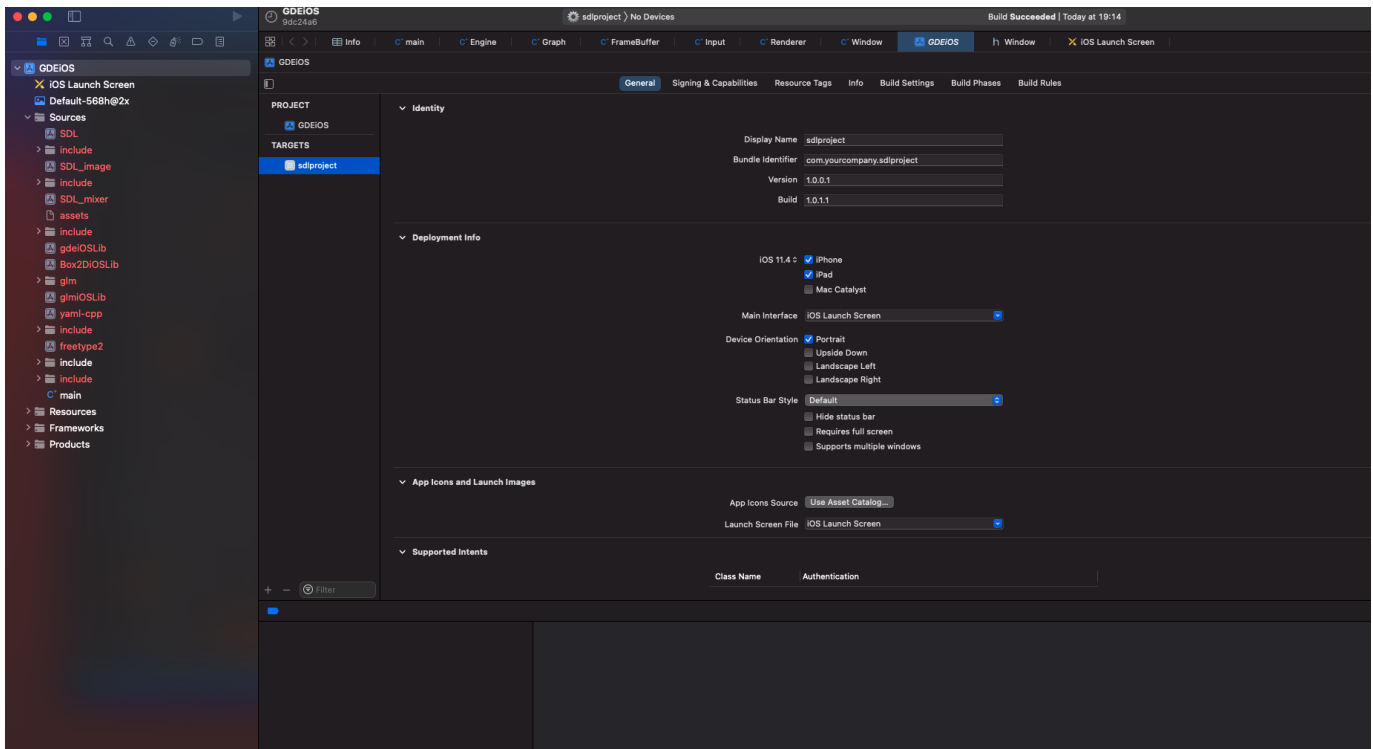
Now compile it and you should get no errors.

iOS project

Now that all our submodules have been setup, lets head to GDEiOS/GDEiOS.xcodeproj and open it. Before you panic, if you noticed, entt module is not needed to be set as the other

ones.

Now you will panic as everythin is like me or worse:



Don't worry, this is just a matter of reimporting everything, is a pain in the neck, but very straightforward, so, let's begin:

- If SDL is in red:
Delete SDL reference. Then Sources -> Add Files to 'Sources' and then look for GDE/submodules/SDL/XCode/SDL/SDL.xcodeproj. Import it as we have import everything else.
- If SDL include is in red:
Delete SDL include reference. Then Sources -> Add Files to 'Sources' and then look for GDE/submodules/SDL/include. Import it as we have import everything else.
- If SDL_image is in red:
Delete SDL_image reference. Then Sources -> Add Files to 'Sources' and then look for GDE/submodules/SDL_image/XCode/SDL_image.xcodeproj. Import it as we have import everything else.
- If SDL_image include include is in red:
Delete SDL_image include reference. Then Sources -> Add Files to 'Sources' and then look for GDE/submodules/SDL_image/SDL_image.h. Import it as we have import everything else.
- If SDL_mixer is in red:
Delete SDL_image reference. Then Sources -> Add Files to 'Sources' and then look for

GDE/submodules/SDL_mixer/XCode/SDL_mixer.xcodeproj. Import it as we have import everything else.

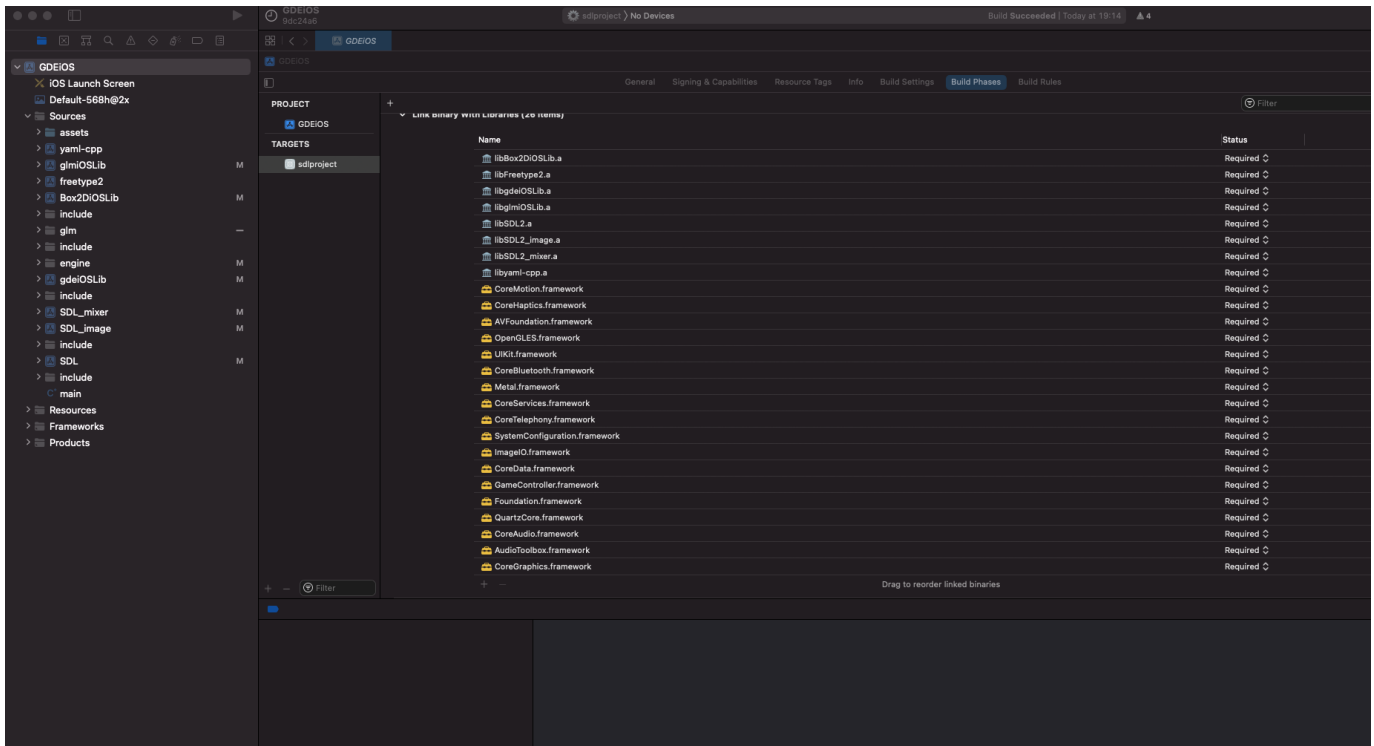
- If SDL_mixer include is in red:
Delete SDL_mixer include reference. Then Sources -> Add Files to 'Sources' and then look for GDE/submodules/SDL_mixer/include. Import it as we have import everything else.
- If gdeiOSLib is in red:
Delete gdeiOSLib reference. Then Sources -> Add Files to 'Sources' and then look for in this same directory for gdeiOSLib/gdeiOSLib.xcodeproj.
- If gdeiOSLib include is in red:
Delete SDL_mixer include reference. Then Sources -> Add Files to 'Sources' and then look for GDE/engine. Import it as we have import everything else.
- If Box2DiOSLib is in red:
Delete Box2DiOSLib reference. Then Sources -> Add Files to 'Sources' and then look for in this same directory for Box2DiOSLib/Box2DiOSLib.xcodeproj.
- If Box2DiOSLib include is in red:
Delete Box2DiOSLib include reference. Then Sources -> Add Files to 'Sources' and then look for GDE/submodules/box2d/include. Import it as we have import everything else.
- If yaml-cpp is in red:
Delete yaml-cpp reference. Then Sources -> Add Files to 'Sources' and then look for in this same directory for yaml-cppiOSLib/yaml-cpp.xcodeproj.
- If yaml-cppiOSLib include is in red:
Delete yaml-cppiOSLib include reference. Then Sources -> Add Files to 'Sources' and then look for GDE/submodules/yaml-cpp/include. Import it as we have import everything else.
- If glmiOSLib is in red:
Delete glmiOSLib reference. Then Sources -> Add Files to 'Sources' and then look for in this same directory for glmiOSLib/glmiOSLib.xcodeproj.
- If glmiOSLib include is in red:
Delete glmiOSLib include reference. Then Sources -> Add Files to 'Sources' and then look for GDE/submodules/glm/glm. Import it as we have import everything else.
- If freetype2 is in red:
Delete freetypeiOSLib reference. Then Sources -> Add Files to 'Sources' and then look for in this same directory for freetypeiOSLib/freetype2.xcodeproj.
- If freetypeiOSLib include is in red:
Delete freetypeiOSLib include reference. Then Sources -> Add Files to 'Sources' and then look for GDE/submodules/freetype/include. Import it as we have import everything else.
- If assets is in red:
Delete assets reference. THIS ONE HAS A DIFFERENT OPTION. Then Sources -> Add

Files to 'Sources' and then look for GDE/assets and add it but with the option 'Create Folder References', you will see it is blue and not as the others that are gray.

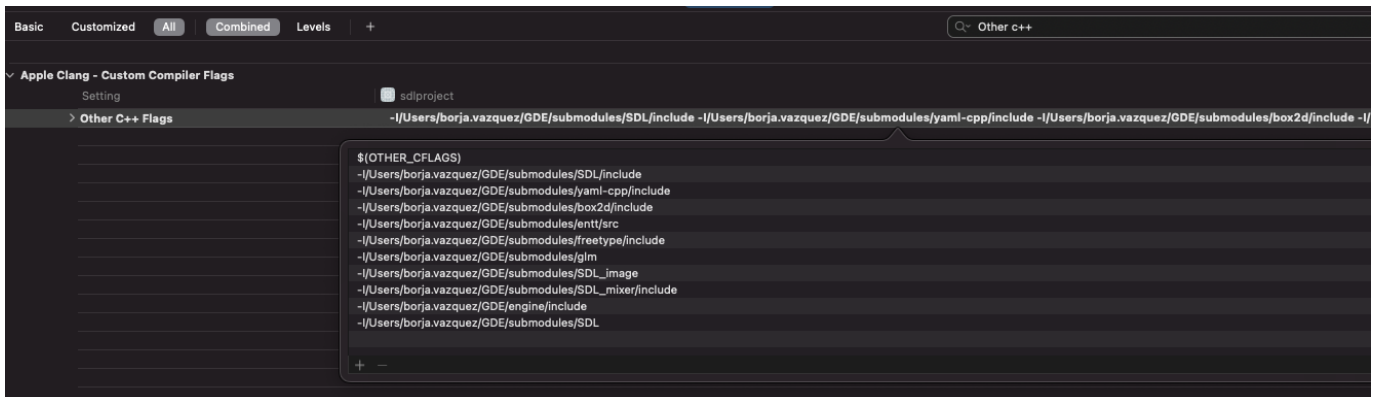
Now select on the hierarchy GDEiOS and then go to 'Build Phases' it is time to add a whole bunch of libraries.

When importing the static libraries (.a) if there is option, choose always the iOS version.

You will need all of this:

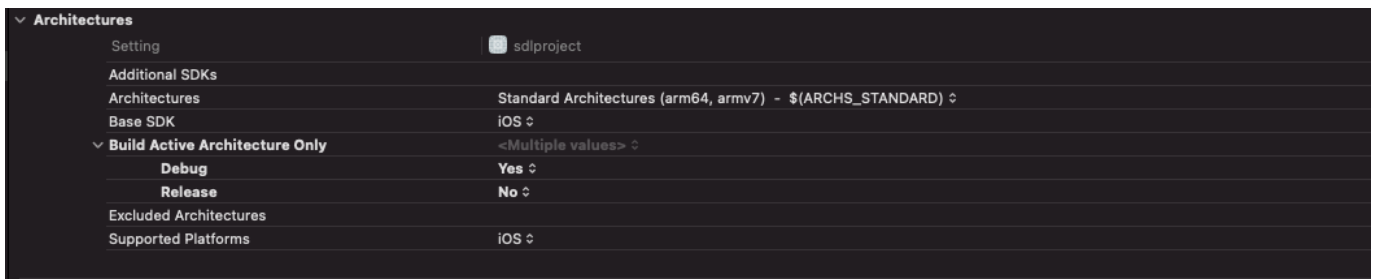
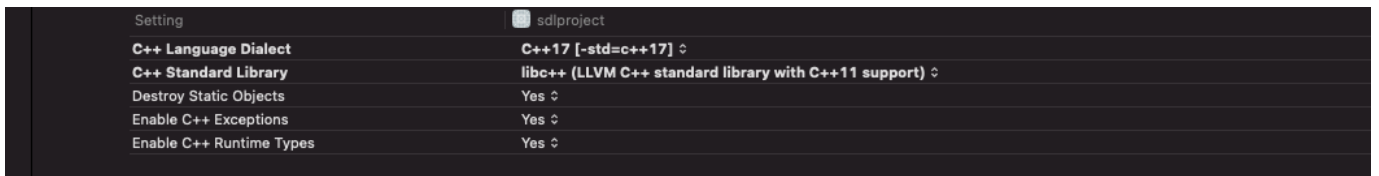
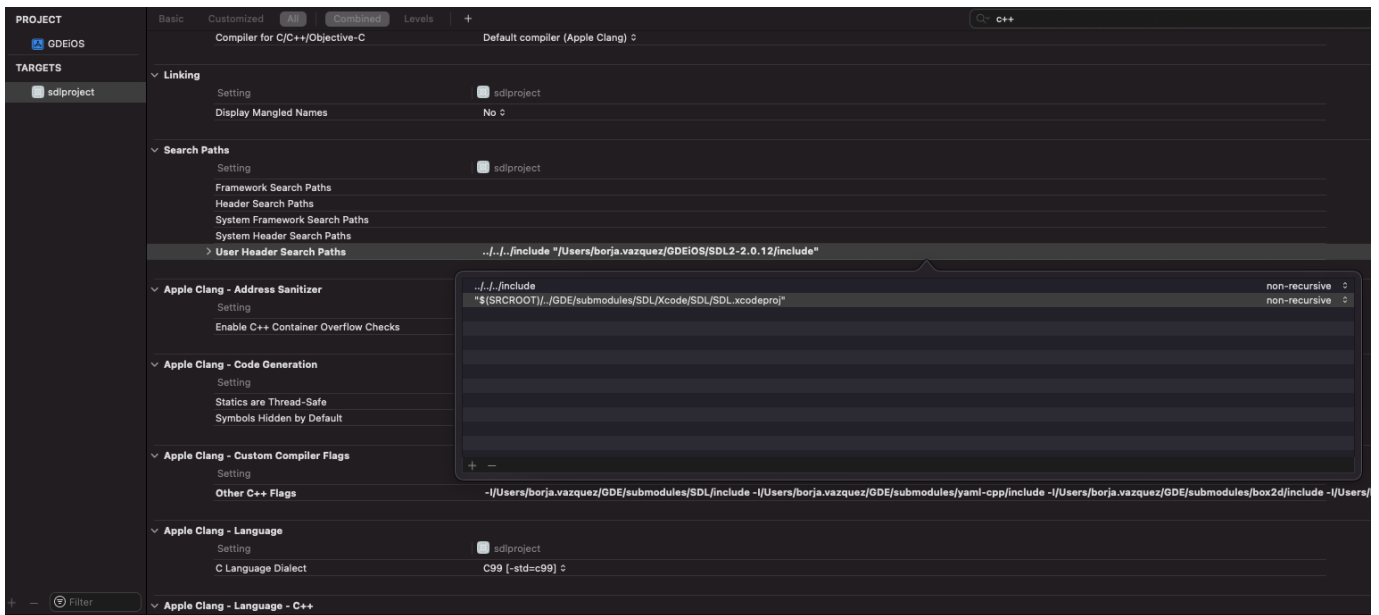


Now let's go to 'Build Settings' and look for Other C++ Flags:

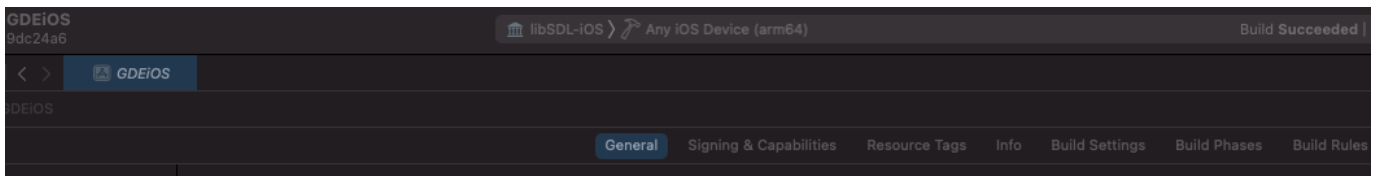


As always, fill in with your paths.

Now just search for C++ and modify the following options:

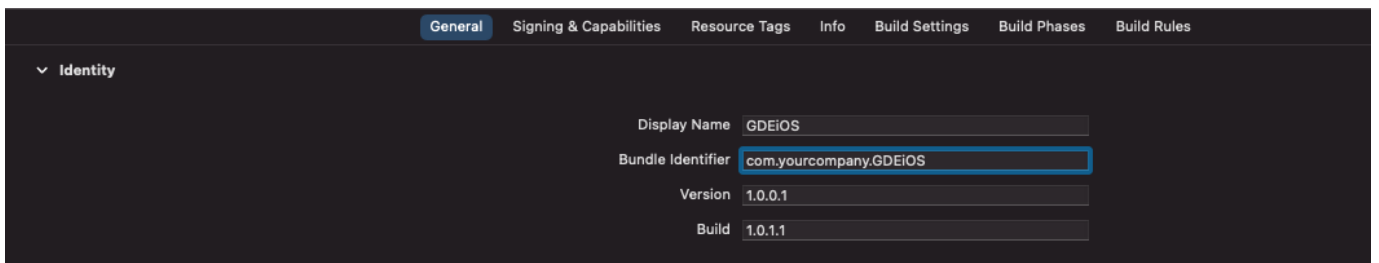


Now select the following and build:



If this is not done, an error 'no entry point found' is going to pop.

In the 'General' tab remember to fill all fields in Identity:

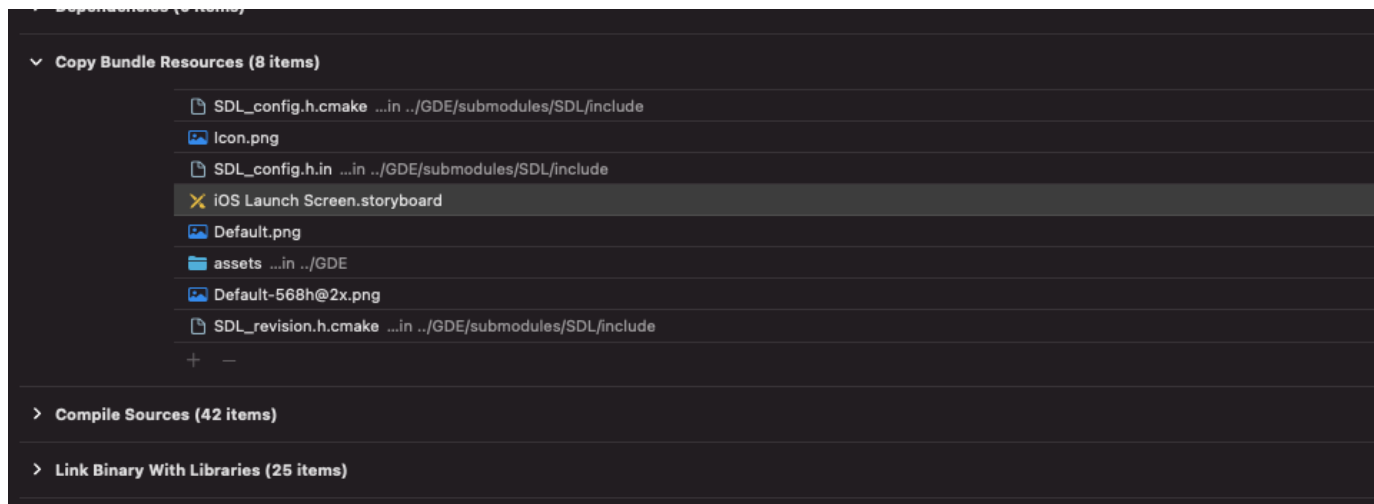


Now go to the hierarchy and open the 'Resources' folder and click on info, check you have those marked values or otherwise add them:

Key	Type	Value
Information Property List	Dictionary	(17 items)
Localization native development region	String	en
Bundle display name	String	GDEiOS
Executable file	String	\$(EXECUTABLE_NAME)
Icon file	String	Icon
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	\$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle version string (short)	String	1.0.0.1
Bundle creator OS Type code	String	????
Bundle version	String	\$(CURRENT_PROJECT_VERSION)
App Category	String	
Application requires iPhone environment	Boolean	NO
Privacy - Bluetooth Always Usage Description	String	\$(PRODUCT_NAME) Because SDL asks for it
Launch screen interface file base name	String	iOS Launch Screen
Main storyboard file base name	String	iOS Launch Screen
Supported interface orientations	Array	(1 item)

Now download the file `simd_vec4.h` of GLM from the internet and add it to glm/gtx.

Now try to compile. If you get an error of type 'Multiple commands produce' then select the blue 'gdeiOSLib' and go to 'Build Phases'. There head to 'Copy Bundles Resources' and you should only have this:



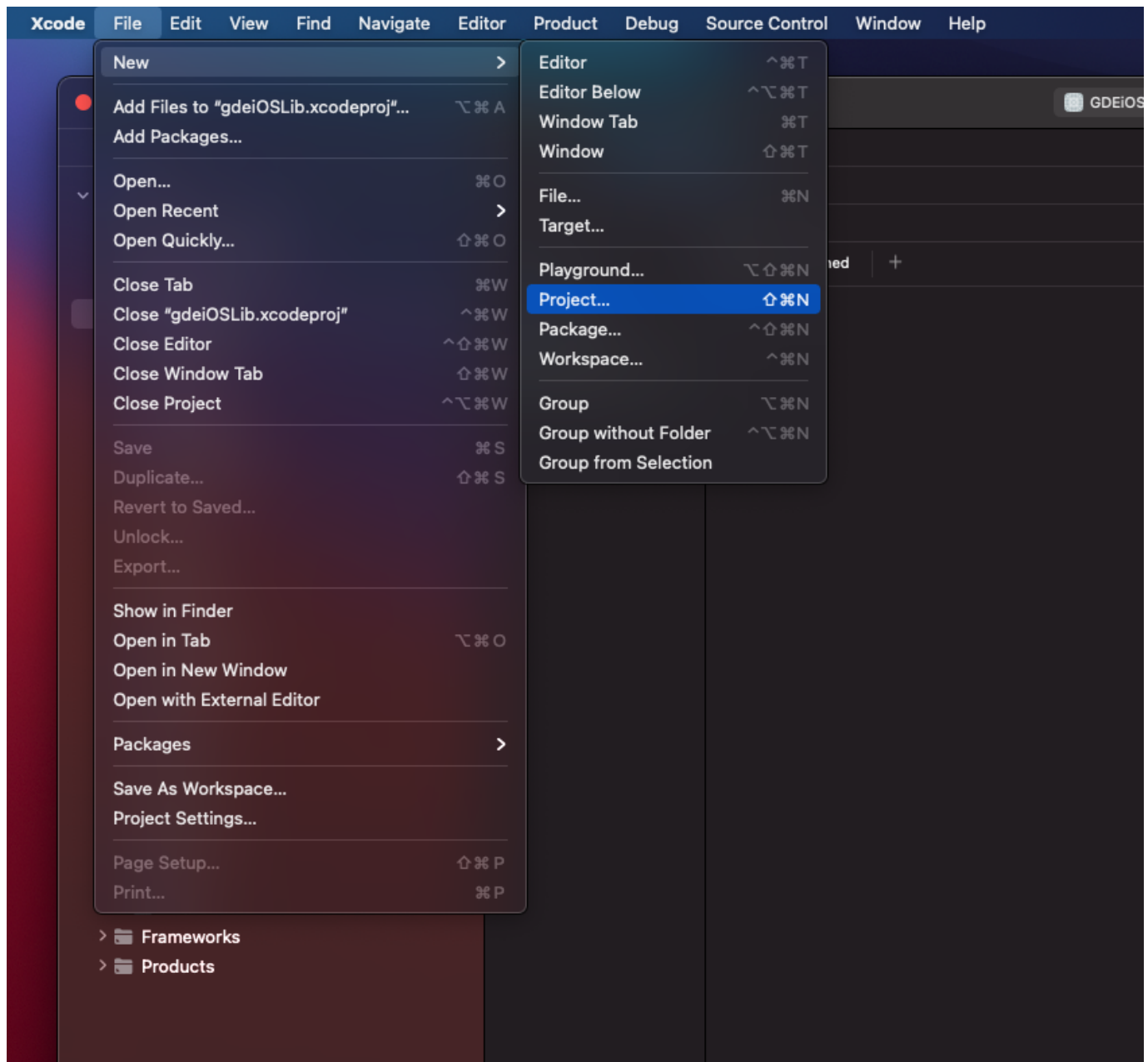
Any other file must be deleted.

Compile again. If you get an error of 'No type named 'Sandbox' in namespace 'GDE"' head to engine and remove main.cpp reference.

Now clicl run (white play button) and a simulator should appear to run your app. To test on device connect it to the Mac and then instead of iPhone 8 select your device.

Add new libraries.

It is quite common to add more libraries to your games, and they may or may not work directly with the iOS version. So if it does not directly, this is how you can create a module like for yaml-cpp, GDE, Box2D or any other:



Choose a template for your new project:

Multiplatform

ios

macOS

watchOS


tvOS

DriverKit


Other

Filter


Application




App




Document App




Game




Augmented Reality App




Swift Playgrounds App



Sticker Pack App




iMessage App




Safari Extension App


Framework & Library



Framework



Static Library



Metal Library

Cancel

Previous

Next

Choose options for your new project:

Product Name:

Team:

Organization Identifier:

Bundle Identifier:

Language:

Then just save it wherever you want (I recommend to do it near the other ones) and then you will see that it creates 2 files, delete them, they are useless.

To configure it just follow the steps done for Submodules and then add it to GDEiOS as we did it in the previous section.

Change final app name.

The screenshot shows the Xcode interface with the 'Info' tab selected for the 'GDEiOS' project. The 'Bundle display name' is highlighted in red. The table below lists the project's configuration details.

Key	Type	Value
Localization native development region	String	en
Bundle display name	String	GDEiOS_Test
Executable file	String	\$(EXECUTABLE_NAME)
Icon file	String	Icon
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	\$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle version string (short)	String	1.0.0.1
Bundle creator OS Type code	String	???
Bundle version	String	\$(CURRENT_PROJECT_VERSION)
App Category	String	
Application requires iPhone environment	Boolean	NO
Privacy - Bluetooth Always Usage Description	String	\$(PRODUCT_NAME) Because SDL asks for it
Launch screen interface file base name	String	iOS Launch Screen
Main storyboard file base name	String	iOS Launch Screen
Supported interface orientations	Array	(1 item)

