How to use premake to create a build configuration for your own project

Orx now provides a tool to do this for you. If you want a hassle free way of spinning up your own Orx-based projects, see the article: Creating your own Orx-based Project using 'init'

If you wish to do this manually yourself, continue with the article below.

This article will help you to create a project using the premake command supplied with Orx. These instructions largely replace the more manual IDE setup instructions by Grey:Setup tutorials by Grey

What is a build project?

When you clone a copy of the very latest orx, look in the \orx\code\build folder. In here you will see folders to Windows, Linux, Android etc. And in these folders, you'll find subfolders to VisualStudio2010, Codelite, CodeBlocks, XCode, jni, etc.

These are build projects. With previous releases of orx, there were several build projects for some IDEs but not all. Premake now builds these Projects for us, for your chosen operating system and editor.

Your build may not exist in these folders, but you can create it. Then you can build orx for your own project.

Download the latest Orx from Github

See here for instructions on getting Orx downloaded.

Build the Orx project from Github

You downloaded Orx, but now you need to build it. And your project needs to use compiled files from that Orx build. These steps are covered in this article. Follow steps: **What is a build project?**, **Creating your build** and **Compiling orx from scratch**. Then you will have a compiled version of orx for your chosen IDE / OS.

Gather parts for your project

Ensure you have the following suggested folder structure:

- **myproject**

  - **bin** (an empty bin folder)
    - **windows** (windows folder for windows exe(s) and dll(s). copy the *.dll files from orx/code/lib/dynamic subfolder to here)
  - **build** (empty build folder - ***copy the premake4.lua file from the orx/tutorial/build folder to here***)
  - **include** (copy the contents of orx/code/include subfolder to here)
  - **lib** (an empty lib folder)
    - **windows** (copy the *.a files from orx/code/lib/dynamic subfolder to here)
  - **src** (your source files)

In the windows folders above, if you are using linux, name your folders linux. Or have both if you are doing multiplatform.

Make sure this folder structure is sitting next to the downloaded orx folder. If it is not, then copy paste your myproject there now, so that orx and myproject are on the same level, ie:

```
/myproject/
/orx/
```

This will be required when the script is run. Parts of orx will be accessed to build out your project(s). Once your project is built, you won't need orx there anymore if you don't want it.

Editing the premake4.lua file in order to make a build project

- Edit the premake4.lua file.
- Scroll to the line: “– Project: 02_Clock”
- Select from that line and remove all lines from here down to the very bottom, leaving just a single project (Project: 01).
- Rename '– Solution: orx' to '– Solution: MySolution'
- Rename 'Solution “Tutorial”' to 'solution “MySolution”'
- Change 'language (“C”)' to 'language (“C++”)' if your project is written in c++. Otherwise leave it as it is.
- Change the includedirs to just reflect your one includes folder:

```
  includedirs
  {
    "../include",
  }
```

- Change the libdirs to just reflect your one lib folder:

```
  libdirs
  {
     "../lib/windows" --or linux, whatever you're target platform is.
  }
```

- Down into the project area, Rename '– Project: 01_Object' to '– Project: MyProject'
- Change 'project “01_Object”' to 'project “Your Project”'
- Change 'files {“../src/01_Object.c”}' to be one or more c, cpp or h files:

```
    files {
        "../src/file1.cpp",
        "../src/file1.h",
        "../src/file2.cpp",
        "../src/file2.h",
    }

    or
    files {
        "../src/*.cpp",
        "../src/*.h"
    }
```

- Under files, add a target name for your executable:

```
targetname ("windows/myproject")
```

Generating the build project

- cd to your build folder (eg myproject/build)
- Call the premake4.exe from within your build folder with:

```
    ..\..\orx\extern\premake\bin\windows\premake4.exe --os=windows codelite
    or just:
    ..\..\orx\extern\premake\bin\windows\premake4.exe codelite
```

(..\..\orx\extern\premake\bin in this example is the folder where you downloaded orx, we need to get to the premake4.exe to do the work against our .lua file, and to save the result in our current folder)

- Your myproject\build\windows folder will contain a codelite folder containing your new project's files.

If you wanted to build for linux (from within windows) you could do a:

```
    ..\..\orx\extern\premake\bin\windows\premake4.exe --os=linux codelite
```

- Open codelite and load your newly created workspace from your myproject\build\windows\codelite folder.
- Add your data folder with your game assets, create your configuration files, etc.
- Build your project and play your game!

Notes

I have used codelite for windows as the example throughout. However there are many more IDE/OS combos available. You can find a complete list from the latest premake here: http://industriousone.com/what-premake or see what the current list is from the premake that ships with orx by checking the “All operating systems, architectures, build projects” section here

Premake Template

myproject.zip

All of the above has been created in a pre-zipped folder structure if you just want to try it out. However, you will still need to copy the correct files in place, ie the libraries, the includes, your source and any ini configs / assets.

Troubleshooting

1. When running premake, you get: [string “local codelite = premake.codelite…”]:13: attempt to index field 'cfg' (a nil value)

Your premake4.lua file contains references to source files, but you didn't add any source files yet.

2. When compiling: *fatal error: orx.h: No such file or directory*

You forgot to copy orx's includes to your includes.

3. When compiling, if you get errors like… *undefined reference to `vtable for* cxxabiv1::*si_class_type_info*

You set the wrong language. Check if you want C++, your haven't set your solution to C.

4. When compiling, if you get errors like… *c:/mingw-4.6.1/bin/../lib/gcc/mingw32/4.6.1/../../../../mingw32/bin/ld.exe: cannot find -lorxd*

Adjust your libfolder path in the premake4.lua file. You probably aren't pointing to the correct spot (myproject/lib can't be found). Or you have nothing in your lib folder at all. Did you build orx and copy the files from that lib folder to yours?