Cloning Orx from Github and Initial Setup

In order to clone the Orx repo, you will need to download and install git. Once installed, you can proceed with the instructions on this page.

```
git clone https://github.com/orx/orx.git
```

Expected output:

```
Cloning into 'orx'...
remote: Counting objects: 67954, done.
remote: Total 67954 (delta 0), reused 0 (delta 0), pack-reused 67954
Receiving objects: 100% (67954/67954), 51.54 MiB | 155.00 KiB/s, done.
Resolving deltas: 100% (49335/49335), done.
Checking connectivity... done.
```

Orx has a few external dependencies. There is a setup script in the root of the folder which does three things:

1. Automatically download the dependencies.
2. Set up projects for various IDEs and compilers, allowing you to compile the orx library.
3. Set an environment variable ($ORX).

For Linux or Mac this is the `setup.sh` command.

For Windows, the command is `setup.bat`.

The first time you clone Orx, you need to run the setup script. Any subsequent updates of Orx with `git pull`, setup will run automatically for you.

When the script automatically downloads the dependencies, the output will look something like:

```
== Checking version: [ extern/ ]
== [ 9d5a692e924e ] needed, current [ ]
== [ 9d5a692e924e ] not in cache
== Fetching [ https://codeload.github.com/orx/orx-extern/zip/9d5a692e924e ]
== Please wait!
== [ 9d5a692e924e ] cached!
== Decompressing [ cache/9d5a692e924e.zip ] => [ extern/ ]
== [ 9d5a692e924e ] installed!
```

Next, the script will generate some projects for several Visual Studio versions, gmake (mingw32), Code::Blocks and Codelite for Windows developers. These projects are for compiling the orx library.

For Mac developers: gmake, Codelite, Code::Blocks and Xcode projects will be built.

Finally for Linux: gmake, Code::Blocks and Codelite projects will be build for you.

The output for the projects being created will be something like:

```
== Generating build files for [ windows ]
== Generating [ gmake ]
Building configurations...
Running action 'gmake'...
Generating windows/gmake/Makefile...
Generating windows/gmake/orx.make...
Generating windows/gmake/orxLIB.make...
Generating windows/gmake/Bounce.make...
Done.
== Generating [ vs2017 ]
Building configurations...
Running action 'vs2017'...
Generating windows/vs2017/orx.sln...
Generating windows/vs2017/orx.vcxproj...
Generating windows/vs2017/orx.vcxproj.user...
Generating windows/vs2017/orx.vcxproj.filters...
Generating windows/vs2017/orxLIB.vcxproj...
Generating windows/vs2017/orxLIB.vcxproj.user...
Generating windows/vs2017/orxLIB.vcxproj.filters...
Generating windows/vs2017/Bounce.vcxproj...
Generating windows/vs2017/Bounce.vcxproj.user...
Generating windows/vs2017/Bounce.vcxproj.filters...
Done.

... etc
```

The final part of the output will show something like:

```
== You can now build orx in [ code/build/windows ]
== For more details, please refer to [ https://orx-project.org/wiki/ ]
== Installing git hook [ post-checkout ]
== Installing git hook [ post-merge ]

== IMPORTANT - New environment detected: Please restart your favorite IDE
before using orx.
== [ 0:00:58 ] Setup successful!
```

This indicates two things:

1. … The Orx library is ready to be built by you, and
2. … that the script has created an `ORX` environment variable in your system.

This makes it easy for your own projects to refer to your Orx library without copying libraries or includes to each project you make.

Your `$ORX` variable should be pointing to your `orx/code` folder.

Now that you have the Orx source, dependencies, and newly created build project, you can compile Orx with your favourite IDE to get a working orx.dll and .lib, .dylib or .so (depending on platform).

This process is fairly bullet proof, but if you encounter any issues, let us know on the forum or in the chat room.