Structure

```
    |- .forge = for now it is quite empty. But you can check REY_LoggerNUtils/.forge to
understand what this really is for
    |- .install = `cmake install`
    |- CMakeFiles
        |- REY_FetchV4 from REY_LoggerNUtils
    |- intern
    |- REY_LoggerNUtils:- [GIT-SUBMODULE] /see ## libraries section in this doc
    |
    |- amGHOST_logWIN32.hh = ☺ [wrapper around REY_LoggerNUtils]
    |- amGHOST_System.hh = Like an Platform Agnostic "INTERFACE"
    |- amGHOST_Window.hh = same as above
    |- amGHOST_<smth>.hh = more like the above two
```

`amGHOST_<smth>.hh` :- e.g. `amGHOST_System.hh`

- These are "INTERFACE" objects.
    - i.e. `class amGHOST_System` has `pure virtual` functions.
- under the hood `class amGHOST_SystemWIN32/X11` or `XLIB/WAYLAND/cocoa` gets created.
    - check files inside `./intern/`
- same kinda thingy happens to all other `amGHOST_<smth>.hh`
- These files serve as both INTERFACE + DOCUMENTATION ☺

Tutorial

- One of my 2025 goal is to create a LIVE Video on this, 😊> where I show the creation of amGHOST from ground up / void / nada / null ☺.

ex. 1

```cpp
#include "amGHOST/amGHOST_System.hh"
#include <iostream>

int main(int argumentCount, char* argumentVector[]) {
    std::cout << "\n";

    amGHOST_System::create_system(); // Static Func, saves the created system into `amG_HEART`

    amGHOST_Window* W = amG_HEART->new_window_interface();
    W->create(L"Whatever", 0, 0, 500, 600);

    std::cin.get();     // wait for terminal input

    W->destroy();

    std::cout << "\n";
    return 0;
}
```

docs

- Treat `amGHOST_<smth>.hh` files as INTERFACE + DOCUMENTATION ☺!
- Everything that you can do with `amGHOST` will be listed inside these files. That is, basically functions and documentation for them.

Libraries / Modules / External Stuffs [.forge]

1. `REY_LoggerNUtils` :- [GIT-SUBMODULE]
    - even tho it's a git-submodule. we fetch/grab/do-shits using CMAKE Scripts like `.forge/CMakeFiles/REY_FetchV4_REY_LoggerNUtils.cmake` instead of `git submodule --update --init`

Common Principles I Followed

1. Logs are better than RETURN VALUES.
    - The way that we need to check RETURN VALUES of every single VULKAN FUNCTION. Wrapping every vulkan function call around with a RESULT/VK_CHECK wrapper.... [all of it felt really frickin hectic >_<>] .... is exactly what led me to take this decision.