Building Pragma

You can also find these build instructions on the Pragma repository on GitHub.

Build Requirements

- ~50 GiB of disk space
- CMake 3.21.4 or newer
- Python 3.9.5 or newer

Windows

- Visual Studio 2022 or newer

Linux

- clang-14 or newer (Pragma is *not* compatible with gcc!)

Build Instructions

Launch a command-line interface and clone the pragma repository (with submodules) into a directory of your choice:

```
git clone https://github.com/Silverlan/pragma.git --recurse-submodules
```

After that you can simply navigate to the pragma directory and run the python build-script, which will automatically download all dependencies, configure CMake, and build and install the project (this will take several hours):

```
python build_scripts/build.py --with-pfm --with-all-pfm-modules --with-vr
```

> If you don't need the filmmaker, you can omit the `--with-pfm --with-all-pfm-modules` arguments, which will significantly reduce the build time and the required amount of disk space.

Linux

> Before running the build script, you will have to install the following packages:

```
# Required for the build script
sudo apt-get install python3

# Required for Pragma core
sudo apt install build-essential
sudo add-apt-repository ppa:savoury1/llvm-defaults-14
sudo apt update
sudo apt install clang-14
sudo apt install libstdc++-12-dev
sudo apt install libstdc++6
sudo apt-get install patchelf

# Required for Vulkan
sudo apt-get -qq install -y libwayland-dev libxrandr-dev

sudo apt-get install libxcb-keysyms1-dev
sudo apt-get install xcb libxcb-xkb-dev x11-xkb-utils libx11-xcb-dev libxkbcommon-x11-dev

# Required for GLFW
sudo apt install xorg-dev

# Required for OIDN
sudo apt install git-lfs

# Required for Cycles
sudo apt-get install subversion

# Required for Curl
sudo apt-get install libssl-dev
sudo apt install libssh2-1

# Required for OIIO
sudo apt-get install python3-distutils
```

Once the build script has been completed, you should find the build files in `pragma/build`, and the install files in `pragma/build/install`. The `install` directory should contain everything you need to run

Pragma.

If you make any code changes to the core engine code, you can build the `pragma-install` target to build them. This will also re-install the binaries.

If you make any code changes to a module, you will have to build the module build target first, and then build `pragma-install` afterwards.

Build Customization

Running the build-script with the arguments above will build and install Pragma and the Pragma Filmmaker with all dependencies. Alternatively you can also configure the build to your liking with the following parameters:

Parameter	Description	Default
`--help`	Display this help	
`--generator <generator>`	The generator to use.	`Visual Studio 17 2022` (On Windows) `Unix Makefiles` (On Linux)
`--c-compiler`	[Linux only] The C-compiler to use.	`clang-14`
`--cxx-compiler`	[Linux only] The C++-compiler to use.	`clang++-14`
`--with-essential-client-modules <1/0>`	Include essential modules required to run Pragma.	1
`--with-common-modules <1/0>`	Include non-essential but commonly used modules (e.g. audio and physics modules).	1
`--with-pfm <1/0>`	Include the Pragma Filmmaker.	0
`--with-core-pfm-modules <1/0>`	Include essential PFM modules.	1

`--with-all-pfm-modules <1/0>`	Include non-essential PFM modules (e.g. chromium and cycles).	0
`--with-vr <1/0>`	Include Virtual Reality support.	0
`--build <1/0>`	Build Pragma after configurating and generating build files.	1
`--build-config <config>`	The build configuration to use.	`RelWithDebInfo`
`--build-directory <path>`	Directory to write the build files to. Can be relative or absolute.	`build`
`--deps-directory <path>`	Directory to write the dependency files to. Can be relative or absolute.	`deps`
`--install-directory <path>`	Installation directory. Can be relative (to build directory) or absolute.	`install`
`--verbose <1/0>`	Print additional debug information.	0
`--module <moduleName>:<gitUrl>`	Custom modules to install. Use this argument multiple times to use multiple modules.	

Example for using the `--module` parameter:

```
--module pr_physx:"https://github.com/Silverlan/pr_physx.git"
```

If you want to create your own binary module, please check out the article on binary modules.

Revision #15
Created 5 December 2022 08:43:43 by Silverlan
Updated 5 December 2022 16:11:33 by Silverlan