Installation Documentation n°1

Abstract

Lugdunum is an open-source 3D engine using the Vulkan API as a backend. Lugudunum's goal is to provide a free, modern, cross-platform (mobile and desktop) 3D engine for everyone.

The team

 Corentin Chardeau

 Quentin Buathier

 Nicolas Comte

 Guillaume Labey

 Stuart Sulaski

 Antoine Bolvy

 Yoann Picquenot

 Alexandre Quivy

 Guillaume-Heritiana Sabatié

 Yoann Long

Document summary

This document is intended for every Lugdunum user who wants to install and build the project on their operating systems.

In this document you will find an overview of the Lugdunum project as well as the build and installation procedures for the different supported operating systems.

Each step will be detailed in order to let this document be as simple and straightforward as possible, for developers of all levels. It is however required for the developer reading this document to have some technical background about their own systems, as we will not cover the basics, that are usually well documented on other documents and do not enter in the scope of this manual. When appropriate, useful links and resources will be provided for your convenience.

In summary, when you finish reading this document, you should be able to compile and install the Lugdunum's libraries on your system, and it should be ready to use to develop an app using our 3D engine.
Should you have any question or concern, contact information is present at the end of this document.

Document description

Title	:	[2018][ID1] Lugdunum
Modification date	:	March 12, 2017
Accountant	:	Yoann Long
E-mail	:	lugdunum_2018@labeip.epitech.eu
Subjet	:	Lugdunum – Installation Documentation n°1
Document version	:	1.1

Revisions table

Date	Authors	Version	Modified Section(s)	Comment(s)
2017-03-07	Antoine Bolvy	1.0	All	Creation of the document
2017-03-11	Antoine Bolvy, Yoann Long, Nicolas Comte, Guillaume Sabatié	1.1	Build instruction and requirements	Added the section

Contents

About the scope of the EIP

The **E**pitech **I**nnovative **P**roject is the final step in a 5 year computer science program at the Paris Graduate School of Digital Innovation (also know as EPITECH), a program that distinguishes itself from others by its focus on practical projects rather than theory.

The EIP is a group project that spans the 4th and 5th year of the program which aims to have as a result a finished project that has actual lasting value, is well documented and is immediately usable in a professional environment. It is a chance for students to acquire skills and experience in accomplishing a task which aims to emulate the real world working environment.

Just like a real large scale project. The students are not only faced with technical problems but they are also tasked with handling marketing, communication, team management as well as writing and maintaining professional documentation, technical or otherwise.

The EIP is the amalgamation of EPITECH's project oriented vision. It aims to symbolize the transition between student and professional life.

Dependencies for Lugdunum

a. Introduction

Lugdunum depends on many different libraries / projects in order to work properly.
You can find here compiled versions, ready to use to compile Lugdunum and get started quickly.

b. List of dependencies

- Assimp: *Open Asset Import Library (short name: Assimp) is a portable Open Source library to import various well-known 3D model formats in a uniform manner.*
- Fmt: *fmt (formerly cppformat) is an open-source formatting library. It can be used as a safe alternative to printf or as a fast alternative to C++ IOStreams.*
- Vulkan: *Vulkan is a new generation graphics and compute API that provides high-efficiency, cross-platform access to modern GPUs used in a wide variety of devices from PCs and consoles to mobile phones and embedded platforms.*

c. Optional dependencies

All our code is covered by different tests through the Googletest / Googlemock framework.
You can find the sources of the framework at the following link or in our third party repository

d. How it works

All the dependencies can be found in our ThirdParty repository, which is added as a sub-module of the main repository.

It is planned to add an utility script to update and compile all the dependencies at once, but as of now, it is still a manual task.

How to build Lugdunum

a. Cloning the repository

First, clone the 3D engine repository:

```
git clone git@github.com:Lugdunum3D/Lugdunum.git
```

Now, to build Lugdunum, you'll need to either have some dependencies installed, or you can automatically pull them from the thirdparty submodule, that regroups their pre-compiled versions to set you up more quickly:

```
git submodule update --init --recursive
```

b. Linux

1. General prerequisites

Target	Toolchain
Linux	gcc >= 6
Linux	clang >= 3.8

2. Distribution specific prerequisites

Ubuntu (Instructions tested for 16.04)

A recent version of GCC (at least the version 6) is needed to compile Lugdunum. You can add the correct repository on an Ubuntu machine with the following commands:

```
sudo add-apt-repository ppa:ubuntu-toolchain-r/test
sudo apt update
```

You can now install the dependencies needed to build Lugdunum: gcc6, CMake (the tool we use to build Lugdunum), the development version of the X11 libraries:

```
sudo apt install gcc-6 cmake libxrandr-dev
```

There is not yet a Vulkan SDK package on Ubuntu, so you'll have to download and install it yourself. A very complete documentation is already available on the LunarG website, so we won't get into details here. Just make sure you have the VULKAN_SDK environment variable set, as described here, with the x86_64 architecture.

Arch Linux

On Arch Linux, nice people packaged the Vulkan SDK and provided it at vulkan-validation-layers, and it depends on all the right things to make things easier. So all you have to do is:

```
pacman -S vulkan-validation-layers base-devel cmake
```

3. Building

The commands below should be distribution independant, hopefully. What we do is create a "build" directory (out-of-source build), cd in it and run cmake with the appropriate compiler versions.

```
mkdir build
cd build
cmake ../ -DCMAKE_C_COMPILER=gcc-6 -DCMAKE_CXX_COMPILER=g++-6
make
```

c. Windows

1. General prerequisites

Target	Toolchain
Windows 10	Visual Studio 2015
Windows 10	Visual Studio 2017

2. Building

To build Lugdunum on Windows, you'll need CMake. CMake will generate a Visual Studio solution that you can then open, and build.

In command line, you can generate the solution with:

```
mkdir build
cd build
cmake
   -G"Visual Studio 15 2017 Win64"
   -DCMAKE_INSTALL_PREFIX="Path/To/Install"
   ../
```

Caution: As Windows doesn't have a default path to install libraries, CMAKE_INSTALL_PREFIX is mandatory

Then, open the generated Lugdunum.sln with Visual Studio and compile it.

Visual studio 2017

With the support of CMake in Visual Studio 2017, building and installing a CMake project is now possible directly within Visual Studio.
Just modify the CMake configuration file CMakeSettings.json to change the install path.

```
{
  "configurations": [
   {
    "name": "my-config",
    "generator": "Visual Studio 15 2017",
    "buildRoot": "${env.LOCALAPPDATA}\\CMakeBuild\\${workspaceHash}\\build\\${name}",
    "cmakeCommandArgs": "",
    "variables": [
     {
      "name": "CMAKE_INSTALL_PREFIX",
      "value": "Path/To/Install"
     }
    ]
   }
 ]
}
```

d. Android

Target	Toolchain
Android	NDK >= r14 + clang + Gradle >= 2.2

1. General prerequisites

- Android NDK r14+
- Android Studio 2.2+

Please note that arm64-v8a is the only supported ABI and that we only support Android N (android-24) and up.

2. About the Android NDK

As the gcc toolchain is now deprecated by Android's developers, the clang toolchain will be the only one supported in this project. Please note that we're also using Unified Headers from Android NDK 14.

3. Compiling

The following commands should work on a Linux environment, and should give you an idea of what's necessary to build Lugdunum for Android in another environment.

For better understanding of Android NDK CMake variables, visit official NDK documentation

```
mkdir build
cd build
~/Android/Sdk/cmake/3.6.3155560/bin/cmake \
```

```
    -G "Android Gradle - Unix Makefiles" \
    -DANDROID=true \
    -DANDROID_PLATFORM=android-24 \
    -DANDROID_STL=c++_shared \
    -DCMAKE_BUILD_TYPE=Debug \
    -DCMAKE_TOOLCHAIN_FILE=PATH_TO_ANDROID_NDK/build/cmake/android.toolchain.cmake \
    -DANDROID_UNIFIED_HEADERS=ON
    ../
make install
```

Note: Here the CMake path might be different of the one displayed in the command above, please double-check before executing the command and/or filing a bug report.

4. Samples

Open the folder samples/compiler/android with Android Studio, let gradle configure the project.
If the NDK isn't configured properly, you'll have to tell Android Studio where to find it:

File > *Project Structure* > *SDK Location* > *Android NDK Location*

Let the gradle configure and sync the project.

The samples should now be available as targets and be buildable from Android Studio.

e. Apple macOS & iOS

These platforms are not yet supported, but they might be one day if Apple decides to release Vulkan on their systems.

Contact us

The development team is available through a wide range of channels if you want to reach out to us:

a. Github

You can find our repositories on Github, at Lugdunum3D, and report specific problems or questions directly by filing a new issue.

b. Mailing list

If you want to write us an email, you can totally do so at `lugdunum_2018@labeip.epitech.eu`.