Installation Documentation

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.

This document is split in two parts: the first is focused on Lugdunum, the 3D rendering engine, and on the other hand, the second is focused on LugBench, the benchmarking product.

In the first part of the 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 that you have some background in 3D rendering, and a working knowledge of your own system (git, CMake, etc.) 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 part, 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 part.

In the second part of the document you will find an overview of LugBench, as well as instruction on how to build and install the API, the website, and the desktop application on your own machine.

Document description

Title	:	Lugdunum Installation Documentation
Modification date	:	July 12, 2017
Accountant	:	Yoann Long
E-mail	:	lugdunum_2018@labeip.epitech.eu
Subjet	:	Lugdunum – Installation Documentation
Document version	:	2.0

Revisions table

Date	Authors	Version	Modified Section(s)	Comment(s)
2017-03-07	Antoine Bolvy	1.0	All Sections	Creation of the document
2017-03-11	Antoine Bolvy, Yoann Long, Nicolas Comte, Guillaume Sabatié	1.1	Build instruction and requirements	Added the section
2017-05-07	Antoine Bolvy, Yoann Long, Nicolas Comte, Guillaume Sabatié, Yoann Picquenot, Corentin Chardeau	2.0	All Sections	Added LugBench and general improvements for the 2.0 version of the document

Contents

Part. 1

Lugdunum

Contents

I. Dependencies for Lugdunum

1. Introduction

Lugdunum depends on many different libraries / projects in order to work properly.
You can find here[1] compiled versions, ready to use to compile Lugdunum and get started quickly.

2. List of dependencies

- Assimp[2]: *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[3]: *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[4]: *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.*

3. 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[5] or in our third party repository[6]

4. How it works

All the dependencies can be found in our ThirdParty repository[7], which is added as a submodule 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.

II. How to build Lugdunum

1. Cloning the repository

First, clone the 3D engine repository:

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

[1] here: https://github.com/Lugdunum3D/Lugdunum-ThirdParty
[2] Assimp: http://assimp.org/
[3] Fmt: http://fmtlib.net/latest/index.html
[4] Vulkan: https://www.khronos.org/vulkan/
[5] following link: https://github.com/google/googletest
[6] third party repository: https://github.com/Lugdunum3D/Lugdunum-ThirdParty
[7] ThirdParty repository: https://github.com/Lugdunum3D/Lugdunum-ThirdParty

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
```

2. Linux

2.a. General prerequisites

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

2.b. Distribution specific prerequisites

Ubuntu

These instructions were tested for Ubuntu 16.04 LTS. 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[8], so we won't get into details here. Just make sure you have the `VULKAN_SDK` environment variable set, as described here[9], with the x86_64 architecture.

[8] on the LunarG website: https://vulkan.lunarg.com/doc/sdk/latest/linux/getting_started.html

[9] as described here: https://vulkan.lunarg.com/doc/sdk/latest/linux/getting_started.html#user-content-set-up-the-runtime-environment

Arch Linux

On Arch Linux, nice people packaged the Vulkan SDK and provided it at vulkan-validation-layers[10], 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
```

2.c. 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
```

3. Windows

3.a. General prerequisites

Target	Toolchain
Windows 10	Visual Studio 2017[11]

Note: Visual Studio 2015 is **NOT** supported anymore, Visual Studio 2017 is the only supported toolchain.

3.b. Building

To build Lugdunum on Windows, you'll need CMake[12]. 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"
   ../
```

[10] vulkan-validation-layers: https://www.archlinux.org/packages/extra/i686/vulkan-validation-layers/
[11] Visual Studio 2017: https://www.visualstudio.com/downloads/
[12] CMake: https://cmake.org/download/

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 recent support of CMake[13] in Visual Studio 2017, building and installing CMake projects 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"
     }
    ]
  }
 ]
}
```

4. Android

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

4.a. General prerequisites

- Android NDK r14+[14]
- Android Studio 2.2+[15]

[13] recent support of CMake: https://blogs.msdn.microsoft.com/vcblog/2016/10/05/cmake-support-in-visual-studio/
[14] Android NDK r14+: https://developer.android.com/ndk/index.html
[15] Android Studio 2.2+: https://developer.android.com/studio/index.html

Please note that arm64-v8a[16] is the only supported ABI and that we only support Android N (android-24) and up.

4.b. 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[17] from Android NDK 14.

4.c. 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[18]

```
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.d. 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.

[16] arm64-v8a: https://developer.android.com/ndk/guides/abis.html#arm64-v8a
[17] Unified Headers: https://github.com/android-ndk/ndk/wiki/Changelog-r14
[18] official NDK documentation: https://developer.android.com/ndk/guides/cmake.html#cmake-variables

5. Apple macOS & iOS

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

III. Contact us

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

1. Github

You can find our repositories on Github, at Lugdunum3D[19], and report specific problems or questions directly by filing a new issue.

2. Mailing list

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

[19] Lugdunum3D: `https://github.com/Lugdunum3D`

Part. 2

Lugbench

Contents

I. LugBench API

1. Dependencies for Lugbench API

- NodeJS[1]: a JavaScript runtime built on Chrome's V8 JavaScript engine
- NPM[2]: a package manager for JavaScript
- MongoDB[3]: a scalable and flexible document database

2. Cloning the repository

First, clone the front-end repository:

```
1 git clone git@github.com:Lugdunum3D/LugBench-API.git
```

3. Prerequisites

3.a. MongoDB

First, you may have to create a local database to test on using the following command:

```
1 mongod --dbpath <wanted_path> --smallfiles
```

Note: `27017` is the default port but you can set it by running `mongod` with the `--port <port_number>` argument.

3.b. Installing dependencies

Using npm, just run:

```
1 npm install
```

This command will install the dependencies from the `package.json` file.

4. Environnement variables

Add the `MONGODB_URI` environment variable to set the MongoDB url, with the port being the port you set in the above step, or the default port, `27017`.

```
1 export MONGODB_URI="mongodb://localhost:27017/lugbench-dev"
```

[1] NodeJS: https://nodejs.org/en/
[2] NPM: https://www.npmjs.com/
[3] MongoDB: https://www.mongodb.com/what-is-mongodb

Here the name is completely up to you to choose; Mongo will automatically create the database if it doesn't exist yet.

You can also define a custom port for the API to run on by setting the PORT environment variable.

5. Launch the project

In command line, you can launch the project with:

```
npm start
```

The API will listen on the port 5000 by default. You can then send requests to the server, e.g.:

```
GET http://localhost:5000/api/v1/gpus
```

II. LugBench's website (the front-end)

1. Dependencies for building LugBench's website

- NPM[4]: a package manager for JavaScript
- TypeScript[5]: TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.
- Gulp[6]: Gulp is a toolkit for automating painful or time-consuming tasks in your development workflow, so you can stop messing around and build something.
- Angular2[7]: Angular2 is a JavaScript framework.

2. Cloning the repository

First, clone the front-end repository:

```
git clone git@github.com:Lugdunum3D/LugBench-Front.git
```

Then, navigate to the folder `LugBench—Front`

3. Launch the project

You will need NPM[8] (Node Packet Manager) installed on your computer. NPM will install all the dependances of the project.

In command line, you can launch the project with:

[4] NPM: https://www.npmjs.com/
[5] TypeScript: https://www.typescriptlang.org/
[6] Gulp: http://gulpjs.com/
[7] Angular2: http://angular.io/
[8] NPM: https://www.npmjs.com/

```
npm install
npm run serve
```

Then start any web browser go to `http://localhost:3000`

4. Use NPM scripts

Command	Description
`npm run build`	Build an optimized version of your application in /dist
`npm run serve`	Launch a browser sync server on your source files
`npm run serve:dist`	Launch a server on your optimized application
`npm run test`	Launch your unit tests with Karma
`npm run test:auto`	Launch your unit tests with Karma in watch mode

III. How to build the LugBench application

1. Dependencies for the LugBench application

1.a. Introduction

Lugbench depends on many different libraries / projects in order to work properly.
You can find on our ThirdParty repository[9] all the compiled versions, ready to use to compile Lugbench and get started quickly.

1.b. List of dependencies

- Lugdunum[10]: Lugdunum is an open-source 3D engine using Vulkan as backend. Lugudunum's goal is to provide a free, modern, cross-platform (mobile and desktop) 3D engine for everyone.
- Json *(from nlohmann)*[11] is a header-only Json library for Modern C++.
- libcurl[12] is a free and easy-to-use client-side URL transfer library
- Restclient-cpp[13] This is a simple REST client for C++. It wraps libcurl for HTTP requests.

Note: libcurl and restclient are not needed to build Lugbench on Android.

[9] ThirdParty repository: `https://github.com/Lugdunum3D/LugBench-ThirdParty`
[10] Lugdunum: `https://github.com/Lugdunum3D/Lugdunum`
[11] Json (from nlohmann): `https://github.com/nlohmann/json`
[12] libcurl: `https://curl.haxx.se/libcurl/`
[13] Restclient-cpp: `https://github.com/mrtazz/restclient-cpp`

2. Cloning the repository

First, clone Lugbench repository:

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

Now, to build Lugbench, 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
```

Note: You must first compile the Lugdunum libraries, as shown earlier in this document

3.  Linux

3.a. General prerequisites

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

3.b. 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 and the location of the Lugdunum library.

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

Note: Of course, CMAKE_C_COMPILER and CMAKE_CXX_COMPILER can be set to clang and clang++

4.  Windows

4.a. General prerequisites

Target	Toolchain
Windows 10	Visual Studio 2015
Windows 10	Visual Studio 2017[14]

4.b. Building

To build Lugbench on Windows, you'll need CMake[15]. CMake will generate a Visual Studio solution that you can then open, and build the project from.

In command line, you can generate the solution with:

```
mkdir build
cd build
cmake
   -G"Visual Studio 2017 15 Win64"
   -DLUG_ROOT=PATH_TO_LUGDUNUM_LIBRARY
   ../
```

`LUG_ROOT` designates the location of the Lugdunum library, which is required to build Lugbench. Steps for building the Lugdunum libraries were describes in the first part of this document.

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

Visual studio 2017

With the recent support of CMake[16] in Visual Studio 2017, building and installing CMake projects is now possible directly within Visual Studio.
Just modify the CMake configuration file called `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": "LUG_ROOT",
```

[14] Visual Studio 2017: https://www.visualstudio.com/downloads/
[15] CMake: https://cmake.org/download/
[16] recent support of CMake: https://blogs.msdn.microsoft.com/vcblog/2016/10/05/cmake-support-in-visual-studio/

```
    "value": "PATH_TO_LUGDUNUM_LIBRARY"
   }
  ]
 }
]
}
```

5. Android

5.a. General prerequisites

- You must compile and install Lugdunum[17] for Android.

Note: We suppose that Lugdunum libraries for Android are built in *ANDROID_NDK/sources/lugdunum*
In case you specified a different path with CMAKE_INSTALL_PREFIX, you must modify the build.gradle accordingly.

5.b. Compiling

Open the folder Lugbench/android with Android Studio and let gradle configure the project.

Note: 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

The project should now be available as a target and be buildable from Android Studio.

6. Apple macOS & iOS

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

[17] Lugdunum: https://lugdunum3d.github.io