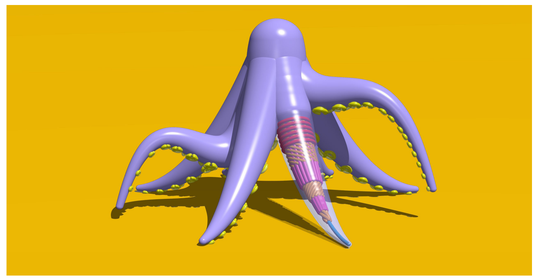
3D Graphics and Build Systems for Elastica

Ian Dailis

Department of Mechanical Science and Engineering, College of Engineering, University of Illinois at Urbana-Champaign

Introduction

Elastica is a software simulator under development by the Gazzola Lab at UIUC. It is used to model all sorts of slender and flexible objects.



The developers make modularity, portability, and cross-platform support a top priority.

To run the program at high resolutions, a user configures their project then deploys clusters to crunch the numbers. After, the data is sent to an offsite rendering service that creates a high resolution visualization.

Aim

Due to the large amount of dependencies, Elastica requires a lot of effort to manage and build.

Since Elastica is primarily maintained by mechanical engineering majors, digging into the dense documentation of CMake (the build system) wasn't something they particularly enjoyed doing.

My task was to learn how to use CMake, along with its packaging and installation capabilities. Then, I would apply what I learned to Elastica to make it installable. It was paramount that the process for installation and linking against the installed binaries was as seamless as possible.

In addition, the team wanted me to create a simple 3D visualization component for the Elastica simulator. High resolution calculations take a long time, so it is difficult to catch mistakes early on. A built-in 3D visualization tool would allow researchers to run the simulation at low resolutions to verify their configuration before deploying the simulation on more expensive hardware.

Following suit with the rest of the project, this 3D simulator also needed to be lightweight but powerful, cross-platform, and packagable.

Method

To begin learning about how to use CMake (and how to program in C++ as a whole), I started by creating the 3D visualization engine.

To fufill the performance demands while keeping the engine packagable and compatible across platforms, I decided to use Khronos Group's newest graphics API: Vulkan.

Vulkan™

Vulkan is supported by all major platforms, and can link to a project using CMake.

The engine only draws triangles. These triangles are stored in two arrays: one with the coordinates of the vertices, and the other with the indices into the vertex array. To draw a triangle, the engine takes three vertices at a time from the index array, and simply fills in a triangle with a given texture. These triangles can be combined to create complex polygons and 3D objects.

Once the engine is created, it needs to be built into a package and installed with CMake.

For CMake to be able to find a package and link against it, the installation must include the following:

1. The compiled binary
2. The public header files
3. A Config.cmake file

The binary is needed to actually run the library code in a user's personal project. The public header files are needed to tell the user's source code that these library functions will exist at runtime. The Config.cmake file is used to easily locate the binaries and do all of the linking automatically using CMake.

First, a binary is created by linking the engine with all of its dependencies. The binary is then installed along with its public headers using a CMake installation command, specifying the paths using the GNU default install directories.

Next, all of the internal targets that were linked together in the project are exported into a Targets.cmake file.

Finally, the Config.cmake file is created. It is generated using a template file that just includes the exported targets and the commands needed to find external package dependencies.

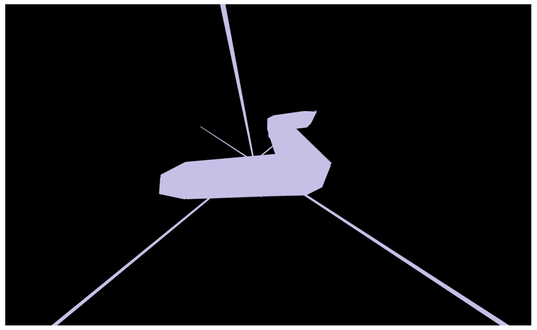
Results

The 3D engine works and can be easily installed and linked with by a user.

There are four shapes that could be used: right triangle, rectangle, cube, and cylinder. These shapes can be generated with any dimensions and at any position.

All of the shapes and the viewer perspective could be individually moved and rotated.

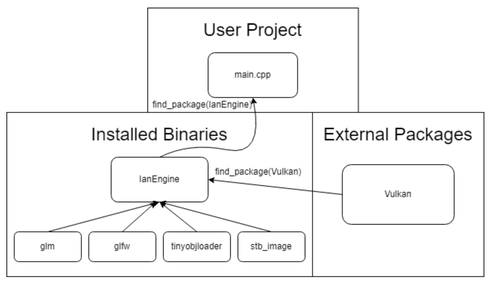
Once the objects are generated, the user uses the provided API to start the engine and begin rendering the objects, while also polling for user input.



Rendering of a snake-like object using 1000 cubes

The installation of this rendering library is also relatively simple. A user would first install the Vulkan library. Then, they would download the source code for my engine and run two commands, one to generate the CMake configuration, and the second to compile the code into binaries and install it to the system.

From there, a user would add one instruction to their own CMake configuration to link their project against the 3D engine, and they can begin using the 3D engine.



Dependency flow chart of the 3D visualization engine

Future Work

The next step for me is to use what I learned about CMake packaging and installation and apply it to the Elastica simulator. The Elastica project is even more complex than my engine with many more dependencies, so it needs an easy installation system so the creators can begin sharing their work.

To add the 3D engine to the Elastica project, it will still need some polishing.

This was my first experience rendering 3D graphics, using CMake, and programming in C++ as a whole. Therefore, I made a lot of mistakes and poor decisions along the way, which makes maintaining this engine very difficult.

I will need to learn the stages of the graphics pipeline to a higher degree so that I could organize the project into a manageable piece of software, and add some basic features such as the ability to use different textures.

Conclusion

A 3D visualization engine for the Elastica project was created using Vulkan then packaged and installed using CMake.

The experience will be applied to Elastica's build system to add packaging and installation features, which will allow the project to be shared with others.

Acknowledgements

Thank you to my graduate mentor, Tejaswin Parthasarathy, for being incredibly helpful and accommodating throughout the past three months. He made this a great first research experience.

PURE Research: Fall 2021

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN