Akarin Engine

Hanif Bin Ariffin

September 7, 2019

Contents

0.1 Introduction

A demo of some version of the engine.

This project was supposed to be a simple calculator. Somehow, it became a freak-of-nature ala Frankenstein of an attempt at making a 3d engine. My current goal is to be able to implement all kinds of rendering techniques (mostly lighting, shadows, etc etc).

After that, I might implement other stuff like physics, animation, sounds, scripts etc. etc. for it to become a proper game engine.

Many terminologies will be used in this book without regard to its proper usage. I am bad at this, so if you find me using inaccurate word to describe things then feel free to comment.

0.2 Motivation

Figure 1: If Nenecchi from *New Game!!* can write a game engine from scratch by herself, what's stopping you?

Honestly, this engine does not have a specific motivation. It is an amalgamation of several sparks of interest that lead me to writing it.

First, while I was contributing to Godot Engine I realize how hard it is to manage large codebases. Upon reading Data-Oriented Design by Richard Fabian, I see why using OOP doesn't help when maintaining an ever growing project. If you want to know why, just read the book. The first few chapters are especially instructive to software architecture in general while the later chapters tackle game engine architecture specifically.

Second, I am interested 3D graphics in general. Whenever I play games, I will always think of how they do it, and how I would (hypothetically) do them myself. However, without any knowledge of writing 3d renderer I didn't have anything to look forward to. That is until I decided to start picking up Game Engine Architecture, OpenGL Redbook and LearnOpenGL

Thus, this project can be thought of as my feeble attempt at implementing what I have learnt so far. Therefore, any decisions made in this project *is going to be dumb*. As such, any comments, criticisms, and suggestions are always helpful.

0.3 Installation

Before we proceed, you will need to install *cmake* and a C++17 compliant compiler. Therefore, if you are in Linux, GCC 7 and above is required.

If you are in Windows, then you will need to use *Visual Studio 2019*. When installing *Visual Studio*, be sure to select the option to install the `cl` compiler. Otherwise, you will not be able to compile them easily – you will need extra steps that involve opening Visual Studio *shivers*.

This project uses *cmake* as its build system. The following steps should work for Linux and Windows.

I don't have a *Mac* so you are on your own. Feel free to try to compile it there and submit a PR of how you did it.

1. Recursively clone the git repo.

   `git clone --recursive https://github.com/hbina/akarin_engine.git`

   We have to recursively clone because the project uses a lot of third party libraries. The list can be found in `.gitmodules`. It's honestly a pain in the buttocks to develop these from the ground up, so that's why I use them :3

2. Get inside the cloned repository folder.

   `cd akarin_engine`

3. Create a folder. This is where the binary will be generated. Therefore, you might want to name it `build` because that folder is specified by `.gitignore`.

   So, execute:

   `mkdir build`.

4. Get inside the folder you just created.

   `cd build`

5. We will now generate the build files.

   If you are using Linux, you can simply execute the following.

   `cmake ..`

   If you are using Windows however, you can skip this step.

6. If you are using Linux, you can simply execute:

   `make all -j$(nproc)`.

   The argument `${nproc}` here will use all the cores in your computer to compile the project.

   Your computer will likely freeze by executing this and possibly make a lot of noise. Therefore, if you want to reduce its usage, then replace it with however many cores you want it to use.

   In Windows, I have wrangled together a `tasks.json` file for use with Visual Studio Code. You can find the file in `./documentation/.source_code`. Copy the file and paste it in `.vscode/` folder and press `CTRL + SHIFT + B`.

   The source code is provided also provided in the appendix A.

TODO :: Describe the project further ...

0.4 Appendices

0.4.1 Appendix A

Content of `tasks.json`.

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "label": "cmake build",
            "type": "shell",
            "command": "cmake",
            "args": [
                "--build",
                "${workspaceFolder}/build",
                "--config",
                "Debug",
                "--target",
                "akarin_engine"
            ],
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "options": {
                "cwd": "${workspaceFolder}/build"
            },
            "dependsOn": "cmake generate"
        },
        {
            "label": "cmake generate",
            "type": "shell",
            "command": "cmake",
            "args": [
                "..",
                "-G",
                "'Visual Studio 16 2019'"
            ],
            "options": {
                "cwd": "${workspaceFolder}/build"
            }
        }
    ]
}
```