A 3D music visualisation in OpenGL using the Discrete Fourier Transform

Matt Young
s4697249
m.young2@uqconnect.edu.au

March 2024

Abstract

Constructing computer graphics from music has important implications in the field of live entertainment. The Discrete Fourier Transform (DFT), often computed via the Fast Fourier Transform (FFT), is the typical method to convert time domain audio signals to a frequency domain spectrum. With this spectral data comes an almost unlimited number of ways to interpret it and construct a visualisation. In this paper, I investigate applying the DFT to construct a semi real-time audio visualisation using OpenGL. The visualisation consists of offline spectral data that is rendered in real-time in the form of "bars" with emissive lighting, and a set of pre-programmed camera moves computed via spherical linear interpolation.

Contents

1 Introduction

2 Methodology

2.1 Toolchain and environment

The visualiser itself is developed and tested in a Linux environment, and is split into two sub-applications: the visualiser itself, and the analysis script. Signal processing is a very complex subject, and real-time ("online") signal processing is even more so. Python generally has simpler tools to address signal processing problems, such as NumPy and SciPy, so the signal processing part was moved offline into a Python script.

The visualiser is what we see on the screen, the real-time rendering system that displays the bars, once the spectral data has been computed. It is written in C++20 using OpenGL, and is built using industry standard tools CMake, Ninja and the Clang compiler. It uses a number of open-source libraries:

- SDL2: Platform window management, keyboard/mouse inputs, OpenGL context creation
- glad: For OpenGL function loading and feature queries
- glm: The OpenGL maths library, used for computing transforms and its matrix/vector types

- Cap'n Proto: An extremely fast data serialisation format, used to transport data between Python and C++.

The analysis script is written in Python, and computes the spectral data itself. It takes a FLAC file, and computes the spectral data necessary to render the bars. It also uses a number of open source libraries:

- NumPy
- SciPy
- spectrum.py
- Cap'n Proto

2.2 Signal processing

3 References