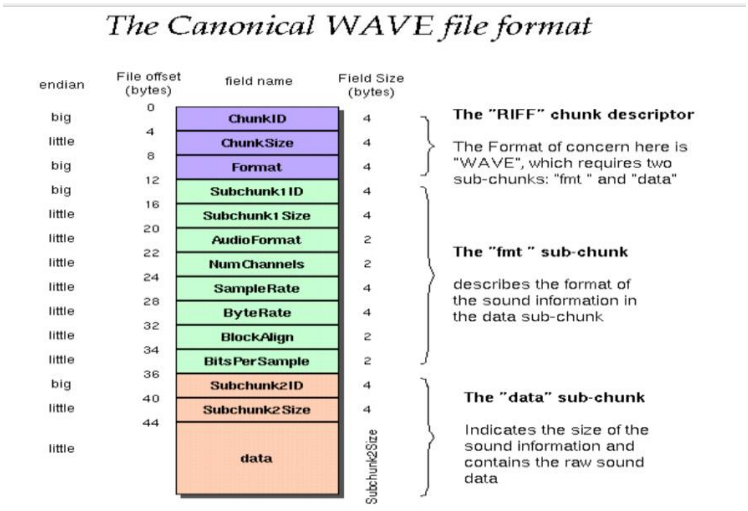
Project 1 音乐可视化报告

16300720005 周启辉

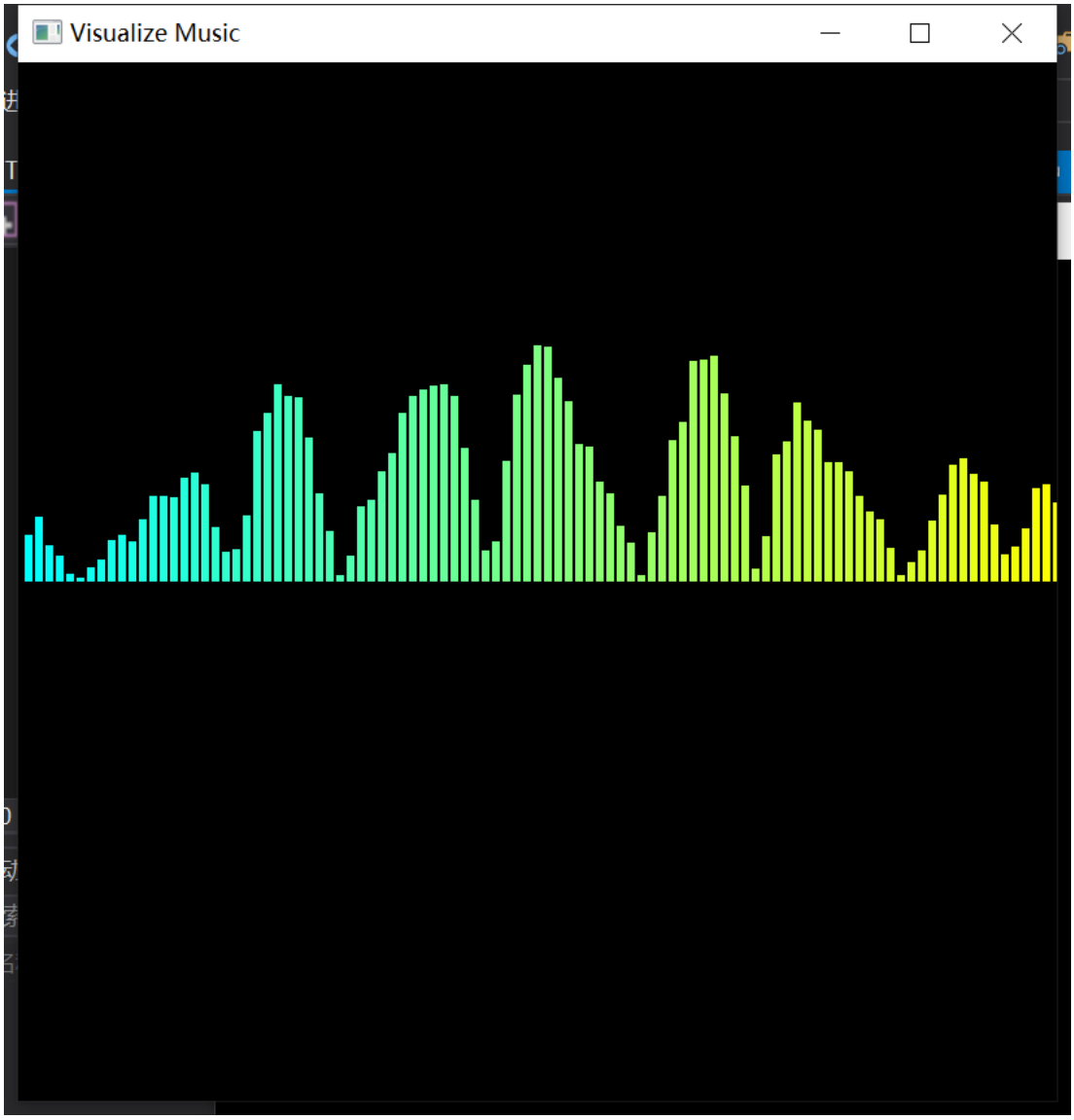
1. 基本思路：
   1) 处理输入：基本思路是使用 python 的 wave 和 audioop 包对 wav 格式的文件进行重采样，获得 16kHz 的 wav 文件（如果太高频率的话数据过多容易炸内存，太低频率的话音乐失真严重，不好听）。之后使用 C++手动编写解析 wav 文件格式的程序，把音频数据一次性读入内存，以便之后处理。
   2) 音乐播放：基于 MCI 用 C++写了一个播放音乐的类，理论上可以播放所有类型的音乐文件，可以随时停止播放，循环播放。
   3) 图形显示：使用 GLFW，这是一个 OpenGL 大家族里封装的比较好的一个库，提供各种接口。所以我做的就是一次性把 wav 文件中的所有数据点都读出来，然后存到一个 vector 数组里，调用里面的函数，显示一个画面，然后在上面画垂线，垂线的长度就代表了这个数据点的振幅。一次绘制一定数量的数据，然后采用滑动窗口的方式进行推进，产生动态的效果（我画的时序图，但是网上很多都画了频率图，我不明白是为什么，毕竟频域应该和时域正交，理论上不能反映我们对音乐的感受，但我还是实现了相关的快速傅里叶变换算法在 FFT.cpp 中，也实现了相关的绘制频域图的函数 drawSpectrum，但是效果不好，所以最后没有使用）。

2. 难点坑点：
   1) Wav 读入：读入 wav 的时候碰到很多问题，我一开始使用著名的 librosa 库来重采样，但是发现采样之后的 wav 文件竟然无法用 wave 库来解析。这是应该因为 librosa 解析之后重写并不是严格按照 wav 文件格式来保存的，有一些字段比如 format 可能就无效了。还有就是如果直接用 librosa 来重采样的话，不会区分声道（也可能是我设置有问题），就是说他会把不同声道的数据直接暴力的合在一起变成一个声道，而且最后文件的大小和格式信息里保存的大小不符合。这时候就显现出 audioop 的优势了，这个库用起来比较麻烦，但是可以自定义如何整合声道的信息转化为单声道，然后可以手动设置 wav 的格式，使得可以顺利解析。



2) 同步播放：由于读入的音频是 16kHz 的，也就意味着 1s 的音乐实际对应了 16000 个数据点，所以我采用的方式很简单就是使得 openGL 在 1s 内扫过 16000 个数据点，这样就可以保证音乐的播放和显示之间的同步。当然，这也需要利用时间函数来计算程序运行的时间，再配合 sleep 函数来实现毫秒级的精确同步。

3. 效果展示：



4. 感想：虽然我只做出了最基础的效果，而且看起来有点丑，但是从零开始自己搭建所有的组件还是给了我很多收获，让我对 wav 文件格式有了深刻的认识，同时也熟悉了 openGL 的使用。虽然我知道有很多 js 的接口是直接可以进行音乐可视化的，而且十分好看，但是我觉得如果使用那种方法的话，可能获得的收获不会很大。