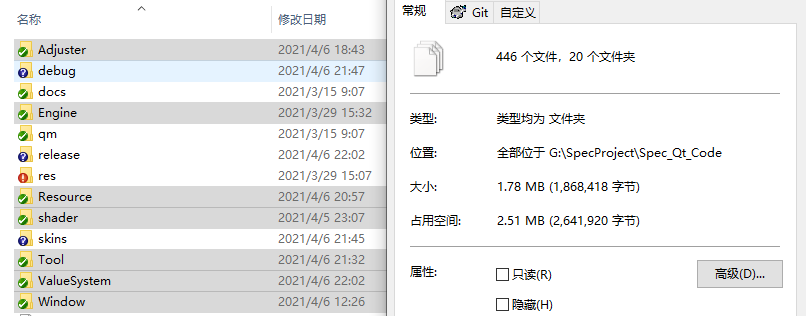
Spec的定位是能够完成实时渲染、可由用户自定义且极具艺术感的音频可视化图形。

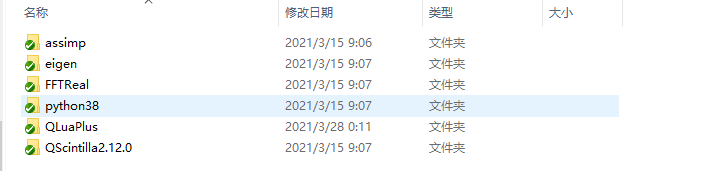
这个项目目前已维护一年半左右，除去第三方库外，仍具有10w+代码量，4w+的下载量，其中架构设计，UI设计，代码编写均由我一人实现，起因只是想做一个音频可视化效果，没想到越做越深，因为学过一段时间的UE4，所以逐渐往图形引擎这方面进行了扩展。

**其中主要的功能模块如下：**

* **音频处理：**利用底层WAS API 实时采集系统声音设备的数据，通过傅里叶运算库FFTW得到频谱数据，通过加窗滤波对数据进行平滑及节奏过滤，得到实时的频谱数据及时域节奏信息供Lua脚本使用。
* **基础类型封装：**由于图形引擎总是需要加载与存储和基础类型数据，和一些特定功能 ，这涉及到两个机制——序列化和反射。C++本身并不提供这样的机制，而Qt通过MOC机制完成了反射机制并提供了序列化方法，在我深入研究了Qt的Moc机制，并找到了Qt的moc工程文件后，通过修改 moc的词法分析机制，实现了一套自己的标记语法，并通过moc为这些标记生成相应的附加代码。（只此一家）
* **脚本引擎：**动画是图形的精髓，制作动画就需要能够动态变化的属性，内部通过代码很容易实现，要在外部实现就很麻烦，我一开始的做法是构建一棵表达式树，根据树来创建控件，树的节点能进行转化成一些运算操作或者基础值，这是参考了UE4 niagara的处理方式，后面重构发现这种方法的扩展性太差，因此想通过脚本进行实现，一开始选择的是Python，花了半月左右实现之后发现60帧运行有明显卡顿，然后又重新实现改用Lua，使用Lua第三方库 LuaPlus搭建Lua环境，并使用第三方开源库QsciScintilla实现了Lua的IDE（支持语法高亮，智能填充），对lua注册C++函数及类型以供Lua脚本调用。
* **Shader接口：**我使用的图形API是OpenGL，所以着色器使用GLSL进行编写，为了能够让使用者能够自己编写着色器，因此我也用QsciScintilla做了GLSL的IDE。结合lua脚本提供Uniform变量可以实现很多炫酷的shader。
* **模型系统：**借助开源模型加载库Assimp实现模型导入功能，Assimp是将大部分主流的3D模型格式转化为一种统一的数据结构，我做的部分是，解析这个数据结构，并将之用作绘图，另外我还写了骨骼动画的处理部分，通过骨骼节点以及动画数据，生成骨骼动画的变换矩阵，内部以定时器的方式播放动画。
* **粒子系统：**借鉴UE4 Niagara系统架构，在脚本引擎的基础上搭建了一个可编程的GPU粒子系统。原理是将粒子状态进行归纳，形状则由通过实例化完成，通过两个tranformFeedBack流水线处理分别完成粒子的生成及运行，这两个tranformFeedBack主要处理粒子的【位置，旋转，速度，颜色，尺寸】，处理完毕之后使用另一条tranformFeedBack流水线生成每个粒子的变换矩阵，最后绘制实例的时候利用得到的矩阵绘制出粒子。开发人员只需提供粒子实例顶点数据，粒子生成shader代码，粒子运动shader代码，可完成任意的粒子效果。

目前有446个代码文件。



使用了如下第三方库：