**20307110315 周训哲**

**编程实现音乐节奏或旋律的可视化**

【程序说明】

本次实验主要运用了pygame对音频的分贝进行识别并对其进行柱状可视化，同时更改了背景，使其个性化，然后通过RGB调色实现可视化颜色的变化，采用pygame内部自带的计时函数对时间进行统计并输出。

然后使用了pygame.event.get()用来获取键盘信息，如果识别到空格输入则切换暂停和播放模式，并在显示框内输出，使用回车键可以从头开始播放，再次点击便开始播放。

对于柱状可视化显示，则使用了librosa自带的fft（傅里叶级数变换实现）。

【算法原理】

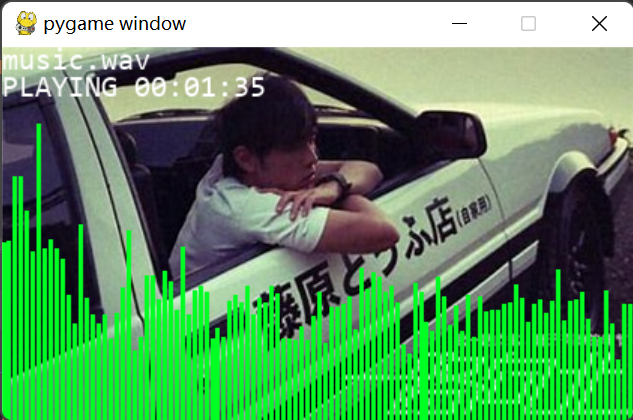
FFT的基本思想是把原始的N点序列，依次分解成一系列的短序列。充分利用DFT计算式中指数因子 所具有的对称性质和周期性质，进而求出这些短序列相应的DFT并进行适当组合，达到删除重复计算，减少乘法运算和简化结构的目的。

（[www.cnblogs.com/lyush/articles/3219196.html](http://www.cnblogs.com/lyush/articles/3219196.html)）

【使用说明】

本项目使用python语言编写，在使用前首先需要安装numpy、pygame、librosa三个包，然后直接在项目文件夹中运行【my\_project.py】文件即可。

【使用效果】



**编程画一个真实感静态或动画景物**

【程序说明】

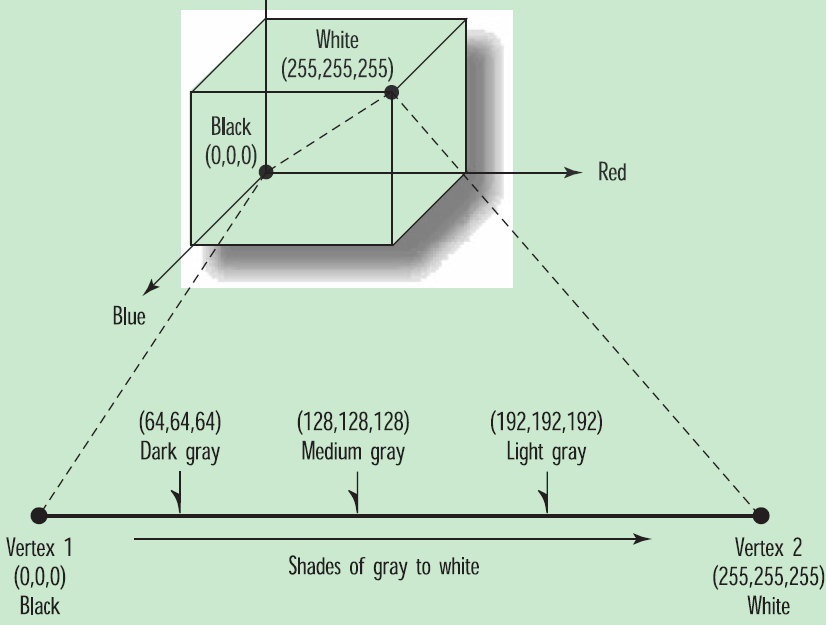
这个项目主要使用opengl库和pygame库实现了太阳系行星运动模拟，其中模拟参数主要源自nasa官网（<https://ssd.jpl.nasa.gov/>）给出的观测数据，利用行星大小、行星质量、运动速度、轨道半径等数据大致模拟了太阳及八大行星之间的运动关系（为了方便计算，假设所有行星运动的轨道近似为圆，并且轨道都在同一个平面内）。

实现思路主要为使用opengl库建立不同大小的球模型，然后使用math库中的参数对轨道半径以及行星大小进行计算，并生成相应的3d模型的位置。然后根据旋转速度进行位置的更新。其中，为了更加模拟现实光源，采用opengl库中的自然光，设定其位于恒星的位置，以模拟太阳光。

除此以外，还使用pygame库对鼠标及键盘输入进行检测，如果检测到【滑轮】滚动，则放大或缩小视图，【X】键是使行星按原轨道加速运行，【Z】是使行星按原轨道方向的反方向加速运行，实现为旋转方向上的速度加减。【Q】键是退出。会根据鼠标位置对观测摄像机位置进行移动。（除了视觉方面，在听觉方面还特地增加电影《星际穿越》的原声以增加宇宙氛围感，音频文件在【music.mp3】中）

【算法原理】。

本项目主要采用了opengl的库函数，调用了其中很多的库算法和库参数，

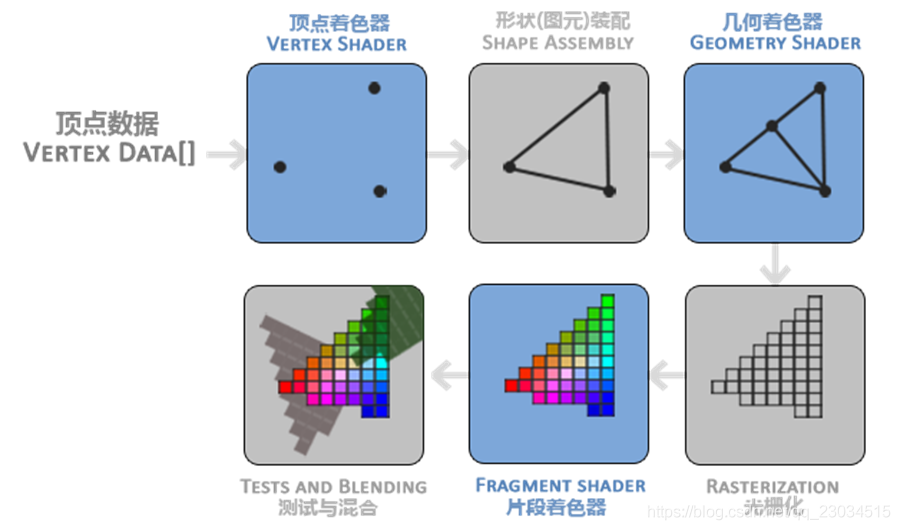
【opengl的光照算法】：OpenGL通过设置着色模式来达到计算指定顶点间颜色的插值。SMOOTH插值使用的是一种线性的插值方式，如下图所示：  
  
vertex1和vertex2之间点的颜色是两者颜色的线性插值。  
FLAT模式仅仅使用所绘制几何图元最后一点的颜色来给图元内部着色

OpenGL光照计算的时候也是使用这么一种设定，我们并不把物体描述为具有一种特定的颜色，而是认为它由一些具有某种反射属性的材料所组成。在指定反射属性的时候，我们同样也指定这种材料对于环境光、散射光以及镜面光的反射属性。（正好和入射光的三种成分相对应）。（<https://blog.csdn.net/csxiaoshui/article/details/53332647>）

【物理引擎】：主要就是利用物理的万有引力公式对于行星轨道的计算，套用公式后将行星实时位置输出给函数，然后对行星位置进行渲染输出。

【Opengl渲染流程】：OpenGL利用GPU进行图形绘制，主要做了两件事，1.配置图元的位置，2.配置图元的颜色，这两者分别通过顶点着色器和图元着色器这一对方法，操作GPU进行像素的位置和色彩数值计算，从而实现图形绘制。大体流程如下：

基本的流程如下图所示：

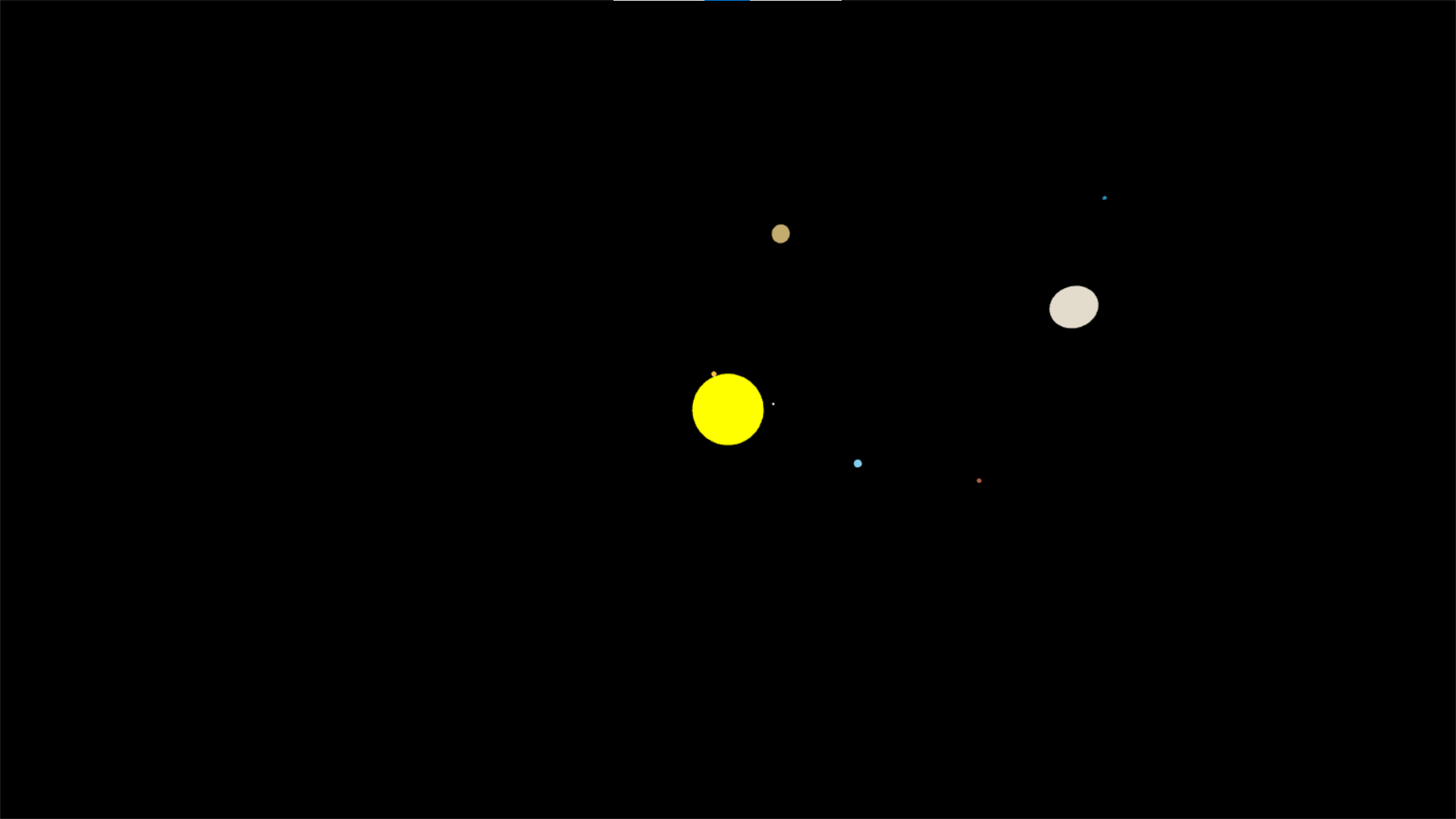


（<https://blog.csdn.net/qq_23034515/article/details/108132191>）

【使用说明】

本项目使用python语言编写，在使用前首先需要安装OpenGL、pygame、numpy、sys、math等库，然后直接在项目文件夹中运行【main.py】文件即可。

【实现效果】



**创作一个Flash动画**

【程序说明】

本次项目主要实现思路就是从网络上寻找一张图片，然后对其进行关键信息进行捕捉，进行平移变换，然后对多张图片进行插帧处理，最后实现了一个眼睛可以转动的表情包。

【实现效果】

