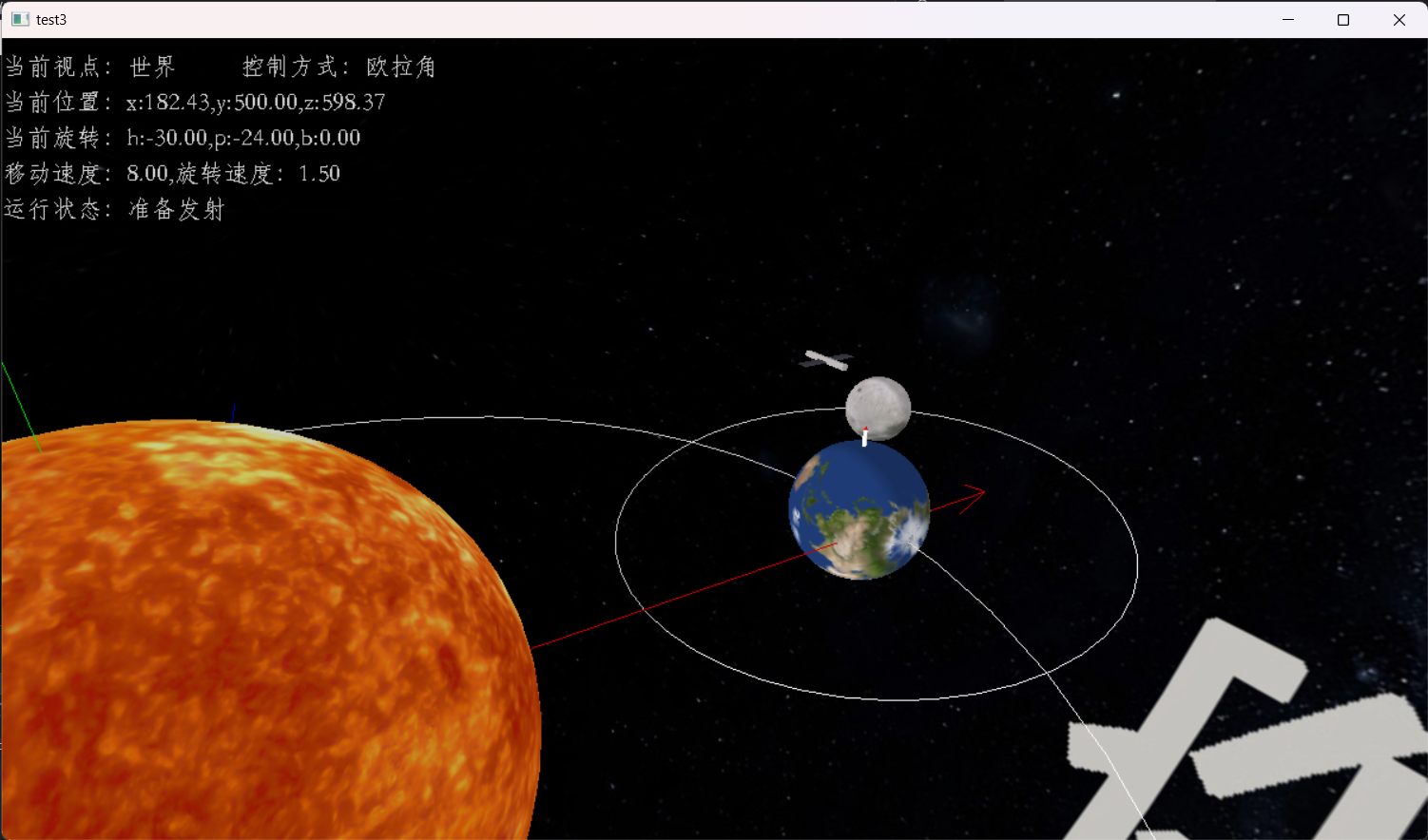
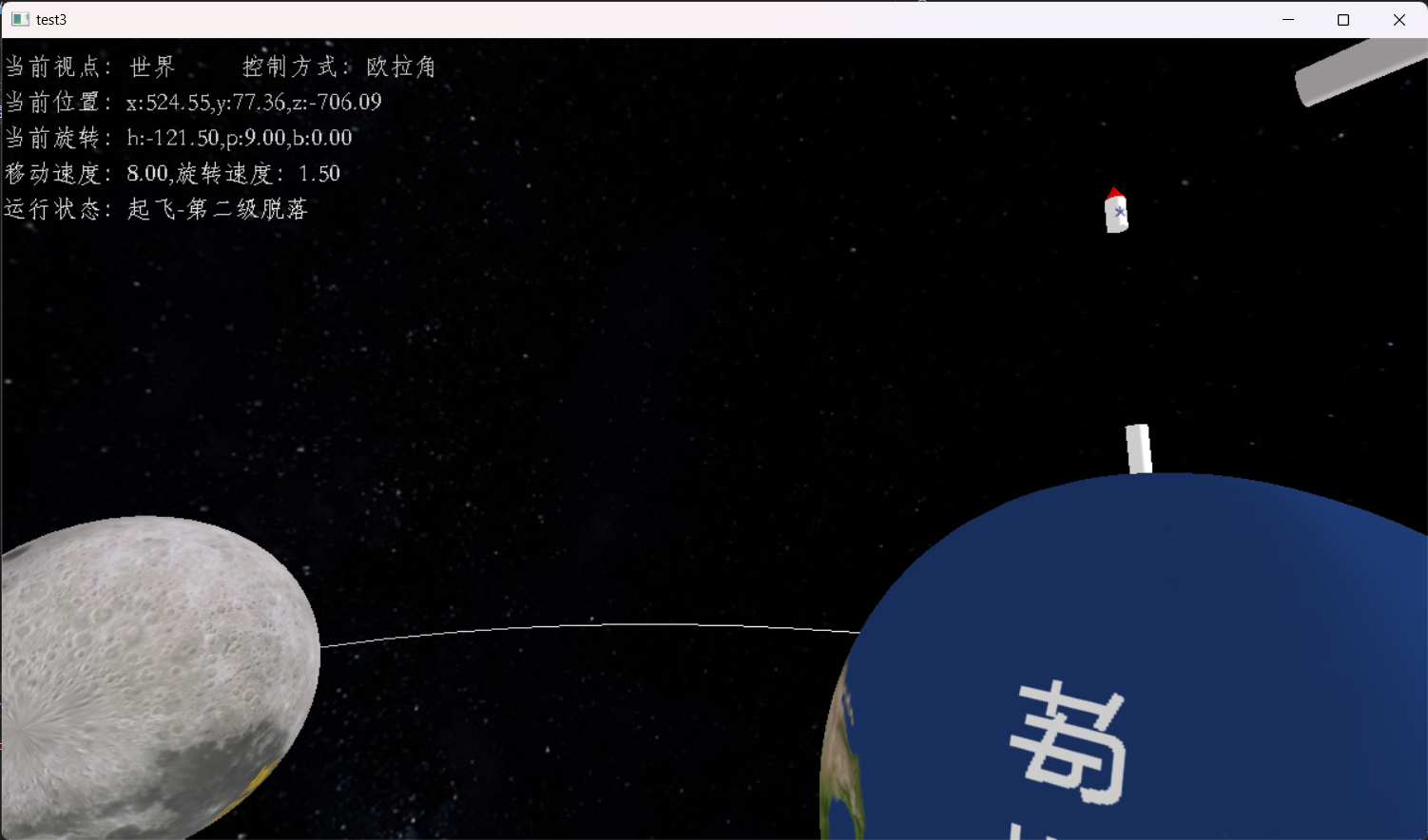
**操作说明**

1、操作与ppt的要求一致

2、火箭、空间站等物体的视角，由于物体本身位置，初始时效果不佳，建议按2切换到自由漫游模式（见提示信息），然后按q向前一段距离，再按j回头，即可看到物体。

3、操作截图





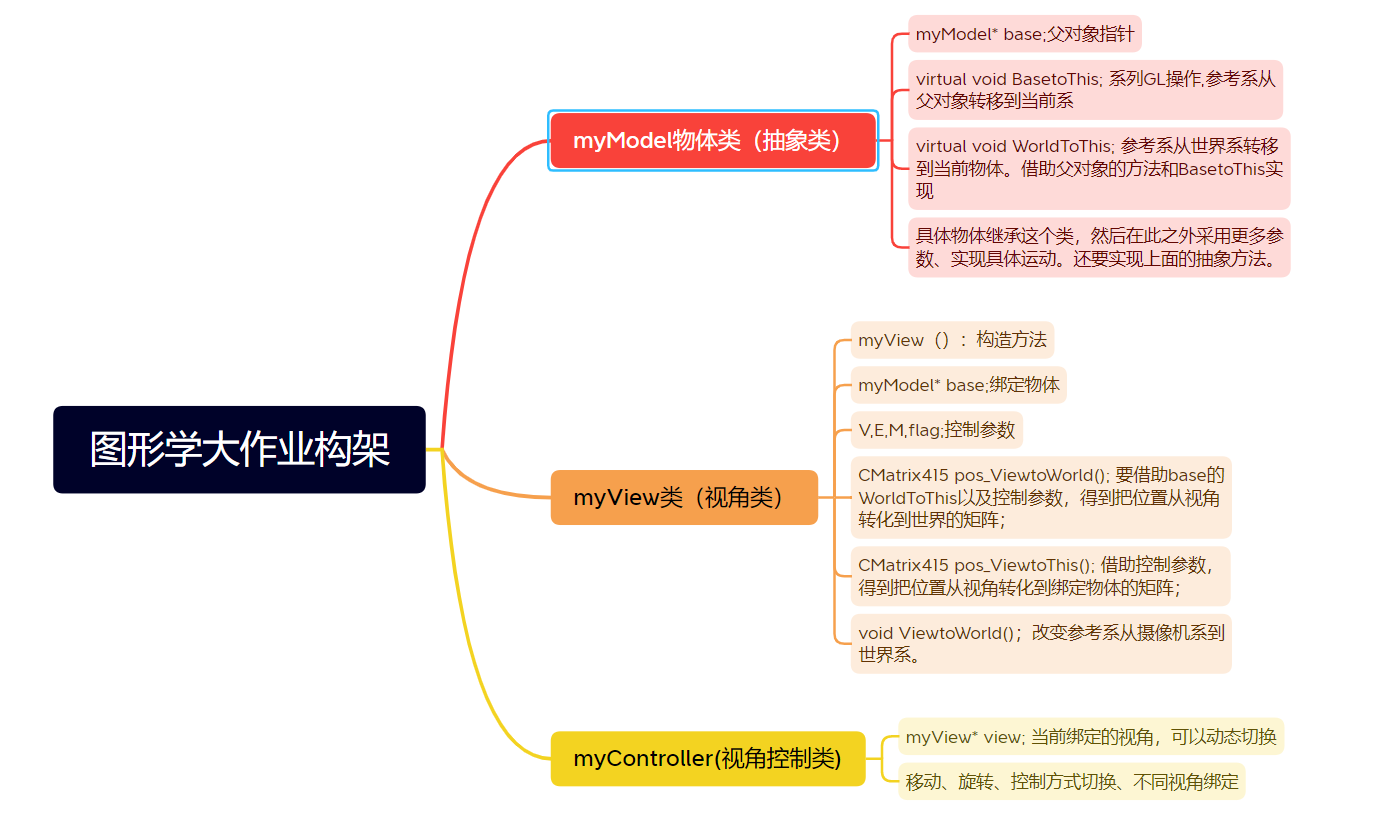
**设计说明**

**视角控制**

上次作业已经确定了用插值+控制变量的方式来实现视角切换。

需要改进的地方是，上次是再controller中直接设置了两组变量手动切换维护，但此次视点较多，手动维护不方便。

于是考虑抽象出controller控制类，View视点类，model物体模型类。View绑定在Model上，然后可以动态的注入到controller中，进而controller可以用通用的方法维护视角。

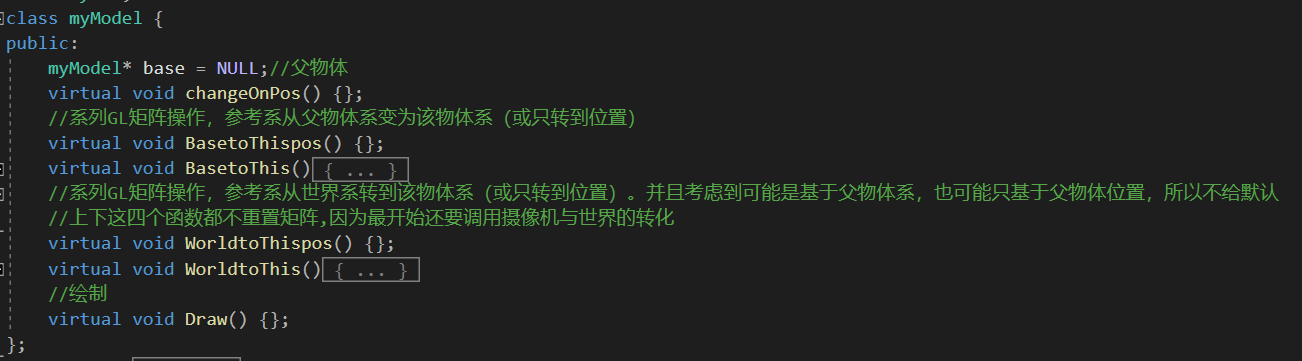


**Model类**

Model为抽象类，但具有一些通用名称的方法，以便View调用。如图，包含把参考系从基准系转移到当前系的gl操作，以及把参考系从世界系转移到当前系的操作。

实际上，后续实现时，将函数进一步拆分细化了，以便重用。

举个例子，空间站和月球都以地球为参照，但空间站要跟随地球自转，是相对于地球系而言；但月球公转的参照实际上是地球的球心，并不随地球自转。这在重用时引起不便，于是拆分出位置（有pos后缀）-系（无pos后缀）两类方法，由changeOnPos（在物体位置上进一步变化）关联起来。



这样，一是可以在重写WorldtoThispos()中具体指定如何参照基准，二是绘制时调整坐标系有更多选择，减少代码重写次数。

**View类**

原本controller中的位置参数被移到View类中，可以用统一的形式调用，不用手动控制。

然后，通过Model指针，View对Model中的void方法进行了包装，将gl操作，变为对应的矩阵或逆矩阵形式，方便controller进行位置计算。

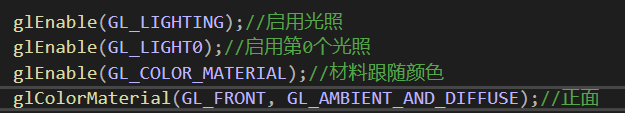
**controller类**

各种操作直接沿用作业三即可，然后将作业三主函数中的插值过程包装成函数，放到这个类里面，作为新增的操作，方便使用。

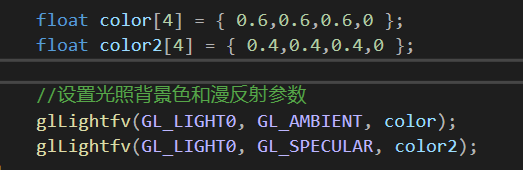
此外，多出一个View\*指针，指示当前的视角。并通过注入的方式让不同视角用同样的语句调用，减少代码重复。

**光照**

启用光照，设置材料跟随颜色，并且绘制除了坐标轴以外的图形都使用白色，就可以让阴影效果与纹理结合。



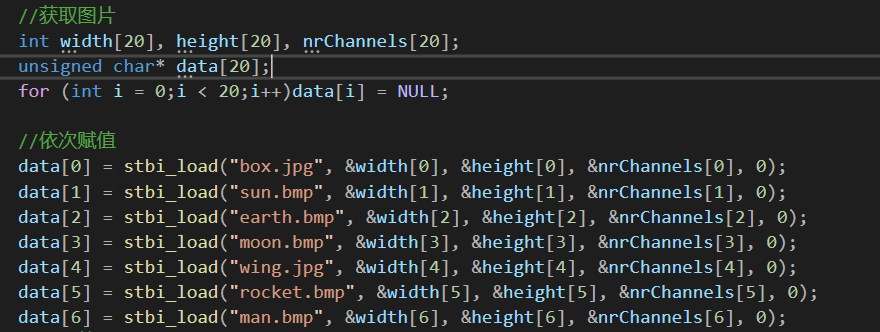
然后调整光照参数



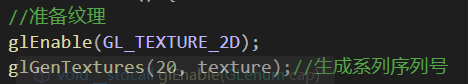
之后每次绘制前，将光源放在太阳中心即可。

**纹理**

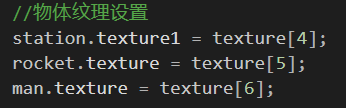
首先，使用了stb\_image.h图片处理库来导入纹理图片数据。



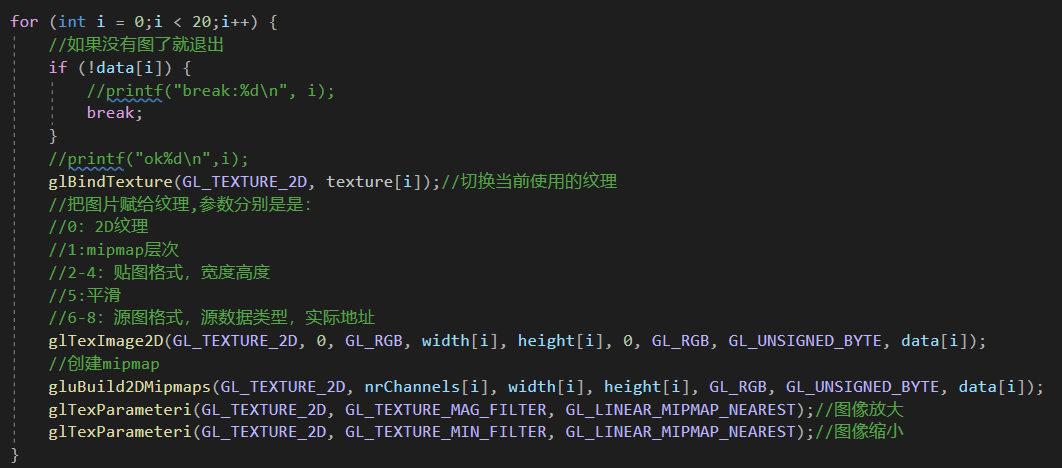
然后，获取一系列纹理标识id



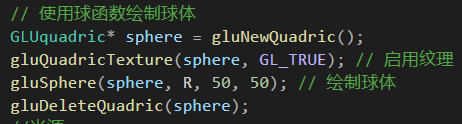
把需要在Model类中调用的部分纹理设置好



然后分别绑定每个id，把图片设置给纹理，并设置mipmap，和一些参数。之后就可以使用了



使用时，可以通过GLUquadric类的纹理开关自动设置纹理坐标。也可以自己指定每个点。



**火箭运动说明**

朝某个固定方向移动的过程比较简单，略过。主要说明一下对接的处理。

跟空间站对接时，有一个旋转部分，但由于角度是知道的，所以每帧调整角度即可。

与月亮对接：

由于月亮在不断移动，如果一直朝着月亮非，可能出现穿模。于是选择了先回到地球正空，然后固定500帧后相遇，直接朝着月亮500帧后的位置飞。

在这个过程中需要两次旋转火箭方向，并且角度不固定。我们需要改为以地球球心为参照，获取二者相对位置，进而知道前进方向。然后参考视角切换，根据原本方向和前进方向，用插值+控制变量的形式，获得火箭的系列旋转角度。

对接后，火箭基准会变为月亮，跟随月亮移动。

**2D文字**

这一部分的方法来自网络。概括地说，使用了windows.h来创建字体和获取DC设备上下文。将上下文中设置的字体替换。利用宽字符处理中文，然后再根据字符通过DC获得位图。再利用显示列表输出。

在绘制完世界后，将投影改为2d，然后设置光标位置，再调用。

具体如下，注释为学习理解该方法时查阅的资料。

