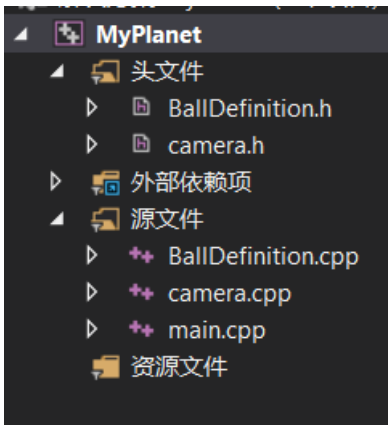
基于 OpenGL 实现太阳系模型——实验报告

一、 实验内容

根据计算机图形学课上所学知识，通过 OpenGL 工具，实现可视化的对于太阳系误差允许范围内的定量模拟。

二、 代码结构设计



其中 BallDefinition.h 中完成对于星球的定义，包括自转速度、公转速度、半径以及颜色等。

Camera.h 中完成相机的定义，最终实现视野的移动和缩放。

在 BallDefinition.cpp 和 camera.cpp 中具体实现类型中声明的公有函数。

Main.cpp 中实现最终的整合工作。

三、 数据结构设计

1、对于星球的定义：

```
class Ball {

public:
    Float4 Color;
    Float Radius;
    Float SelfSpeed;
    Float Speed;
    // ParentBall是本球绕行的球
    // Center是本球的中心点，当有ParentBall和Distance的时候可以不使用
    // Distance是本球中心与ParentBall中心的距离
    // Center暂时没有使用
    //Float2 Center;
    Float Distance;
    Ball * ParentBall;
    GLuint texture;
    virtual void Draw()
    {
        DrawBall();
    }
    virtual void Update(long TimeSpan);
    Ball(Float Radius, Float Distance, Float Speed, Float SelfSpeed, Ball * Parent, GLuint texture);       // 对普通的球体进行移动和旋转
    void DrawBall();

protected:
    Float AlphaSelf, Alpha;
};
```

据此，派生出两个类，分别是“发光球”（太阳）、不发光球（其它行星及卫星）

```
class MatBall : public Ball
{

public:
    virtual void Draw()
    {
        DrawMat();
        DrawBall();
    }
    MatBall(Float Radius, Float Distance, Float Speed, Float SelfSpeed, Ball * Parent, Float3 color, GLuint texture);
    // 对材质进行设置
    void DrawMat();
};
```

```
class LightBall : public MatBall {

public:
    virtual void Draw() { DrawLight(); DrawMat(); DrawBall(); }
    LightBall(Float Radius, Float Distance, Float Speed, Float SelfSpeed, Ball * Parent, Float3 color, GLuint texture);
    // 对光源进行设置
    void DrawLight();
};
```

2、对于相机的定义

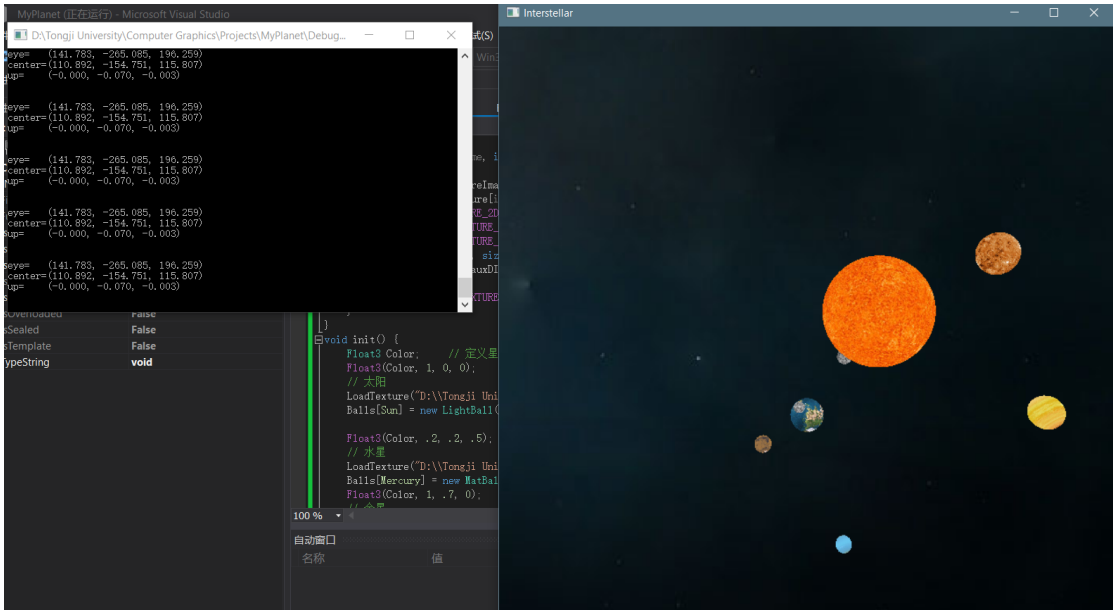
```
class Camera
{
public:
    static void bindCamera(Camera & camera);

    glm::vec3 eye;
    glm::vec3 center;
    glm::vec3 up;

    void setScreenSize(float w, float h);

    void translate(float x, float y, float z);
    void translate2(float x, float y, float z);
    void shift(float dx, float dy);

    void shiftMouse(float dx, float dy);

    void setlookAt(GLfloat x1, GLfloat y1, GLfloat z1, GLfloat x2, GLfloat y2, GLfloat z2, GLfloat x3, GLfloat y3, GLfloat z3);

    void updateView();

    float getR();

    double getEyeX();
    double getEyeY();
    double getEyeZ();

    Camera();
    ~Camera();
private:
    glm::mat4 _view;
    float screenW, screenH;
    glm::vec3 localXnorm();
};
```

3、星球的具体实现

```
enum STARS {
    Sun,
    Mercury,
    Venus,
    Earth,
    Moon,
    Mars,
    Jupiter,
    Saturn,
    Uranus,
    Neptune
};
Camera camera;
Ball * Balls[ARRAY_SIZE];
GLuint texture[ARRAY_SIZE + 1];
```

在主函数中定义枚举变量，分别代表各个星球，定义全局的星球数组以及纹理数组，其中纹理通过给定路径的.bnp 文件导入。

四、 结果展示



五、 心得体会

通过本次小组作业的编写，小组成员系统地运用了 OpenGL 涉及到的基础功能，同时可以比较好地结合其他学科知识并综合运用到代码设计及实现的过程当中。所有星球的自转周期和公转周期都是根据具体数据按比例缩放得到，但处于可是效果考虑，各个星球的半径并未完全按照比例，但可以通过方向键人为调整摄像机的位置来实现视野的缩放，近距离观察各个星球的运动状态。光源方面，实现了全局光和点光源结合，在不插入贴图时效果更加明显，但贴图亮度太高导致阴影效果欠佳。

六、 成员分工

摄像机模块：侯峥韬、贾梦婷

星球模块：柏炎、任秋宇

整合：高昕宇

贴图、报告及 PPT：刘禹辰