SolarSystem

3200105085 丛箫言 A4

1. 实验内容：
    1. 绘制一个太阳
    2. 绘制至少两个行星
    3. 绘制至少一个卫星
    4. 实现各星球的自转or公转
    5. 实现键盘&鼠标控制视角变换
2. 实验思路：
    1. 定义星球类GLstar：以“卫星”作为基类，行星和恒星分别为该基类的派生类
        - 星球半径
        - 自转半径，公转半径
        - 自转速度、公转速度
        - 控制自转、公转位置的角度（定位星球位置）
        - 颜色RGBA
        - 绕转星球
        - 类定义如下

```
class GLstars{
public:
    GLfloat selfRadius, aroundRadius;
    GLfloat selfSpeed, aroundSpeed;
    GLfloat RGBAColor[4] = {1.0f,1.0f,1.0f,1.0f};
    GLfloat selfAlpha, aroundAlpha;

    GLstars* father;

    GLstars();
    GLstars(GLfloat selfRadius, GLfloat aroundRadius, GLfloat
selfSpeed, GLfloat aroundSpeed, GLstars* father);
    void GLstars_Draw();
    virtual void GL_Update();
    virtual void GLDraw();
    virtual ~GLstars();
};
```

    2. 定义太阳系控制类GLsolarsystem，关键属性：
        - 存储太阳系所有星球（恒星+行星+卫星）的指针数组
        - 决定观测视角的相机变量GLcamera
        - 在OpenGL的glutMainLoop()控制下反复执行的
            - Update：更新当前太阳系（各星球）的信息
            - Display：实时显示变化，响应Update
            - Keyboard：监测键盘信息

- Mousehit：监测鼠标点击信息（左键or中键or右键）
- Mousemove：监测鼠标移动信息（任何一个键在屏幕上移动的像素信息）

- 类定义如下

```
class GLsolarsystem{
public:
    GLsolarsystem();
    GLsolarsystem(GLfloat centerX, GLfloat centerY, GLfloat centerZ,
GLfloat upX, GLfloat upY, GLfloat upZ);
    void GLsolarsystem_Update();
    void GLsolarsystem_Display();
    void GLsolarsystem_Keyboard(unsigned char key,int x,int y);
    void GLsolarsystem_Mousehit(int button, int state, int x, int
y);
    void GLsolarsystem_Mousemove(int x, int y);
    ~GLsolarsystem();


    GLstars* Stars[Stars_N];
    GLcamera Camera;

    GLfloat centerX, centerY, centerZ, upX, upY, upZ;
};
```

3. 代码架构：

1. GLmain.cpp

- 主程序，利用glut特性搭建程序运行整体框架，完成初始化、创建所需要的窗口，并在glutMainloop()控制下实现实时的显示、更新、键盘&鼠标信息监测

```
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
glutInitWindowPosition(WINDOW_POS_X, WINDOW_POS_Y);
glutInitWindowSize(WIDTH, HEIGHT);
glutCreateWindow("SolarSystem created by XiaoyanCong");
glutDisplayFunc(My_SolarSystem_Display);
glutIdleFunc(My_SolarSystem_Update);
glutKeyboardFunc(My_SolarSystem_Update_Keyboard);
glutMouseFunc(MY_SolarSystem_Mouse_Hit);
glutMotionFunc(MY_SolarSystem_Mouse_Move);
glutMainLoop();
```

2. GLstars.hpp

- GLstars.cpp

  GLstars完成了所有星球（恒星、行星、卫星）的属性定义，并实现了对应每一个星球的绘制函数，以供GLmain.cpp和GLsolarsystem.cpp调用。在glutMainloop()控制下，实时执行整体的update函数，遍历所有星球即可实现所有星球属性的实时更新，随后实时执行的display函数可以遍历调用每一个星球的绘制函数实现所有星球的实时绘制。

3. GLsolarsystem.hpp

- GLsolarsystem.cpp

GLsolarsystem完成了整个太阳系的架构定义，存储着太阳系中所有的星球信息，以及观察该太阳系的照相机信息GLcamera，关键部分是在其中定义了GLmain.cpp的glut架构中由glutMainloop()控制的反复调用的更新、显示、键盘&鼠标信息监测函数，即

```
void GLsolarsystem_Update();
void GLsolarsystem_Display();
void GLsolarsystem_Keyboard(unsigned char key,int x,int y);
void GLsolarsystem_Mousehit(int button, int state, int x, int y);
void GLsolarsystem_Mousemove(int x, int y);
```

4. GLcamera.hpp
    - GLcamera.cpp

      GLcamera完成了观察整个太阳系所需要的所有视角信息，即EYEX，EYEY，EYEZ，VIEW，结合观察目标点center_X, center_Y, center_Z就可以唯一确定观测视角，通过GLmove和GLRotate函数实现了视角的变化，每次读取到鼠标和键盘的信息后，都会根据该信息去更新视角，实现鼠标和键盘对视角的控制

```
class GLcamera{
public:
    GLcamera();
    GLcamera(GLfloat EYEX, GLfloat EYEY, GLfloat EYEZ, GLfloat
VIEW);
    void GLmove(GLfloat move_up_and_down, GLfloat
move_right_and_left, GLfloat move_forward_and_backward);
    void GLRotate(GLfloat VIEW_ROTATE);
    ~GLcamera(){};

    GLfloat EYEX, EYEY, EYEZ, VIEW;
};
```

5. GLparameters.hpp

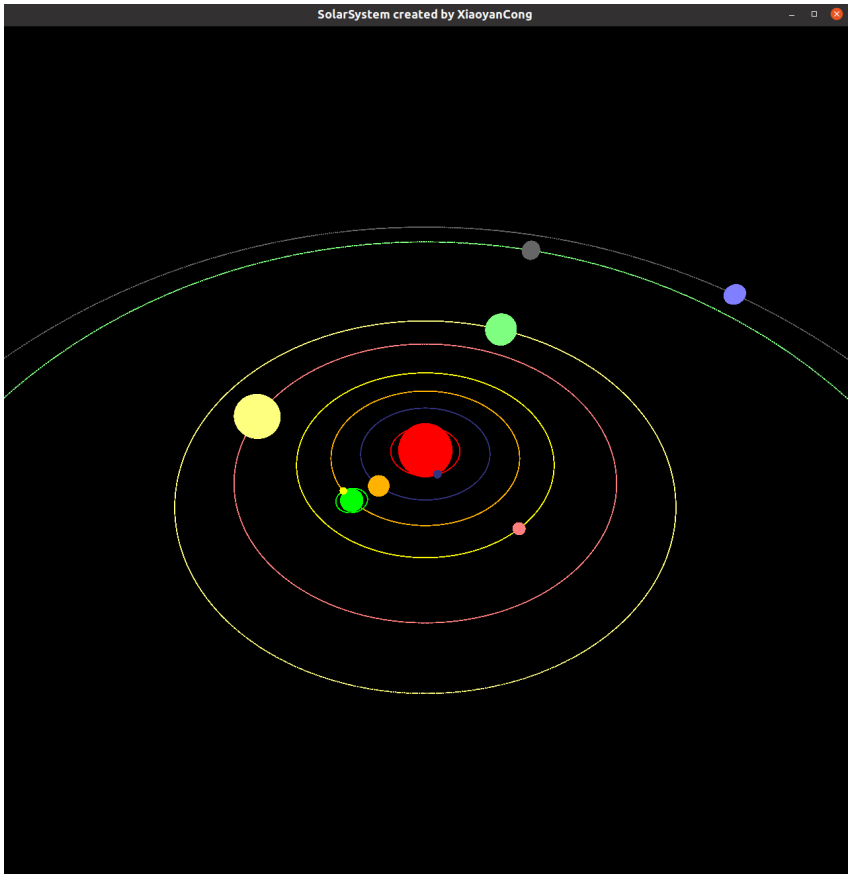
   该头函数定义了my_SolarSystem所需要的一些基本且重要的全局变量，比如恒星、行星和卫星的自身半径，公转半径，自转速度，公转速度，显示窗口的大小，在屏幕上的显示位置等。

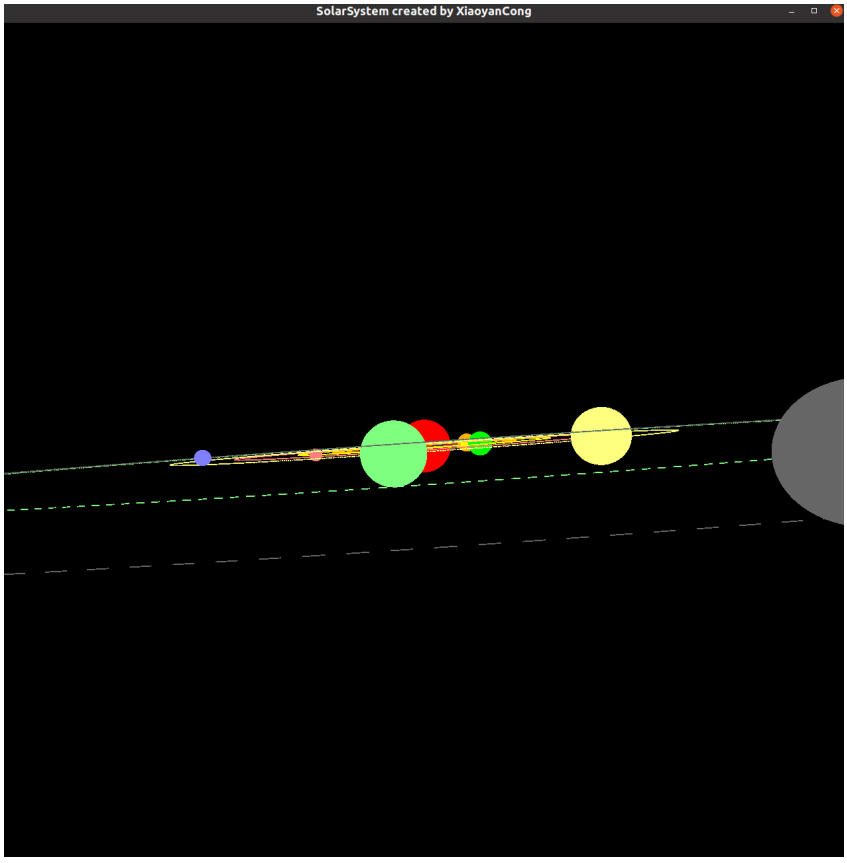
4. 已实现功能

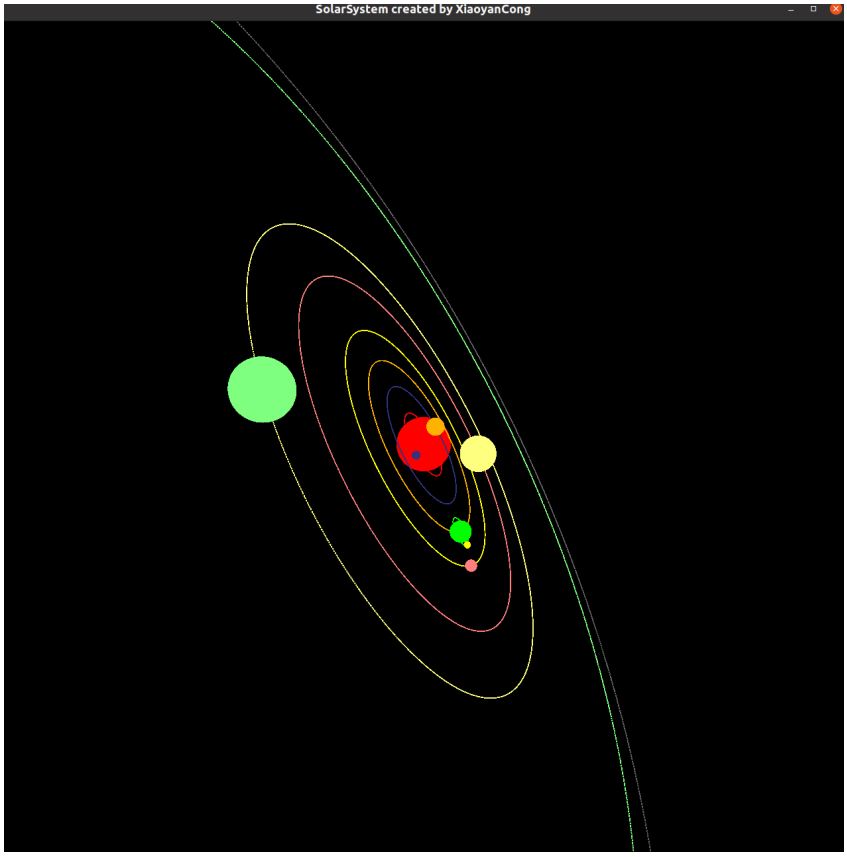
   Achieve：

   1. The revolution of eight planets in the solar system
   2. The moon orbits the earth
   3. The rotation of the sun and the eight planets
   4. View control:
      1. q-a : view -- up & down
      2. w-s : view -- right & left
      3. e-d : view -- forward & backward
      4. "esc" : Exit(0)
      5. mouse button : control the camera flexibly
         1. left mouse button : EYEX / EYEY / EYEZ (rotate around axes)
         2. right mouse button : upX / upZ

5. 实验效果图：


SolarSystem created by XiaoyanCong


SolarSystem created by XiaoyanCong



6. 可优化方向
    1. 纹理贴图：
        1. 可以给各个星球增加纹理贴图，并且贴图可以跟随星球的转动而转动，实现更加真实的太阳系建模
        2. 用星空背景取代纯黑背景，并且可以随着视角变换变化星空的外观，实现更加逼真的太阳系建模
    2. 增加光源，设置光源位置，增加环境光、镜面反射光等。
    3. 可以给每个星球添加材质
    4. 添加阴影 & 深度检测等
    5. 增加键盘按键和鼠标控制功能
        1. 如基于某视角的直接放大、缩小
        2. 某星球某个属性的实时可视化
        3. 以及各种键盘+鼠标配合使用的功能