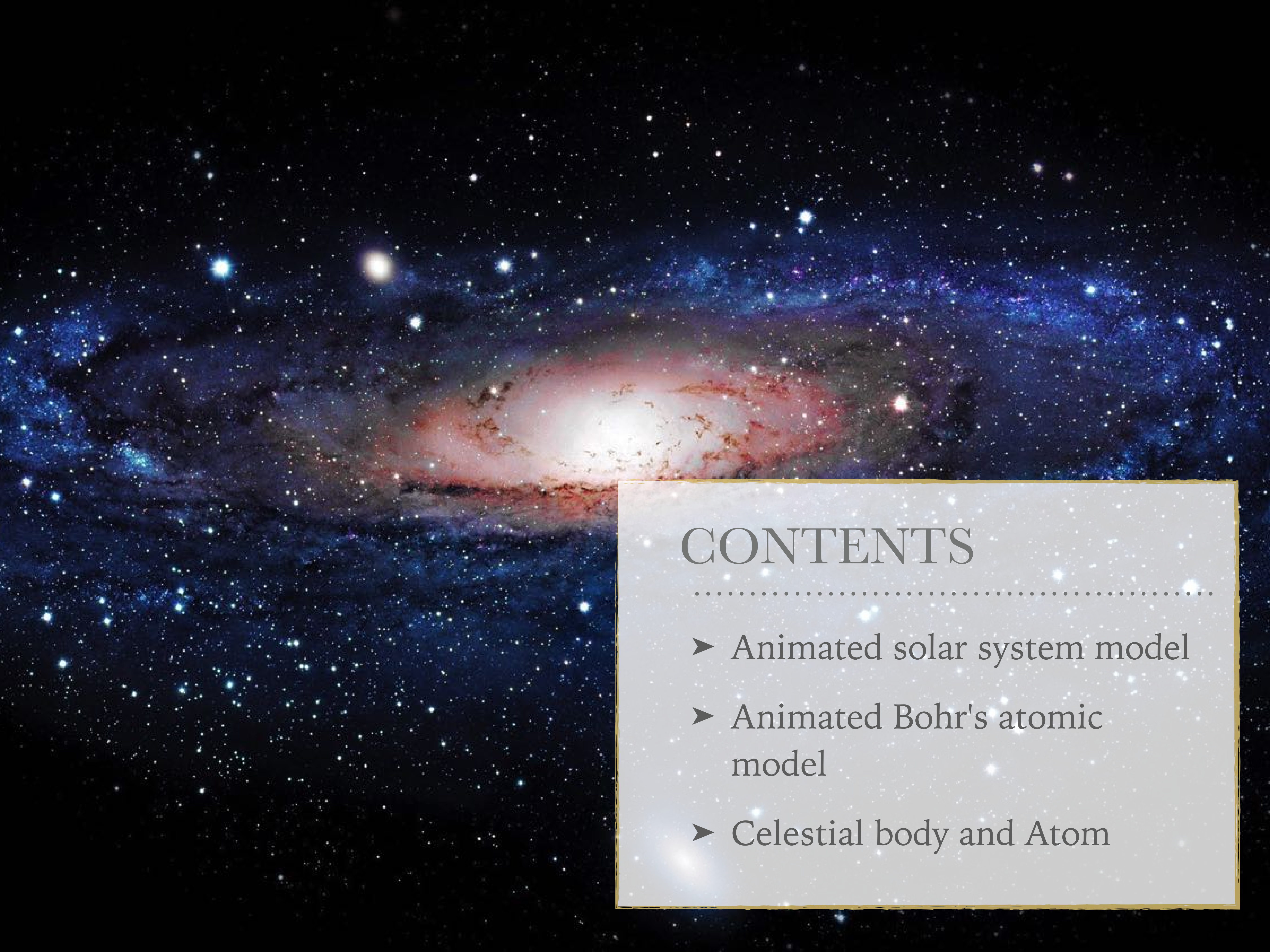
PROJECT OF COMPUTER ANIMATION ALGORITHMS AND TECHNIQUES

Wanru Zhao 17373240

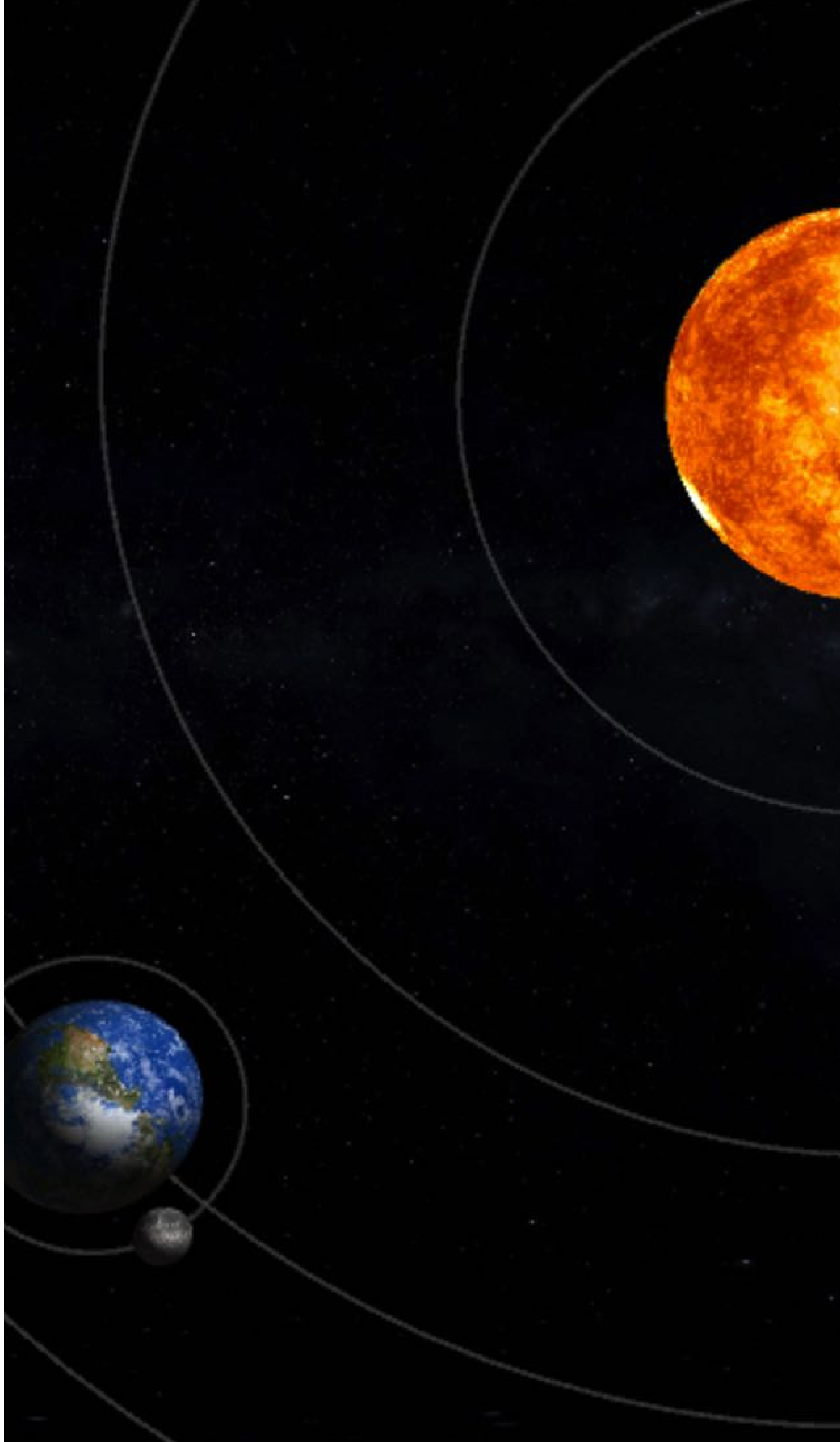
School of Computer Science and Engineering,BUAA

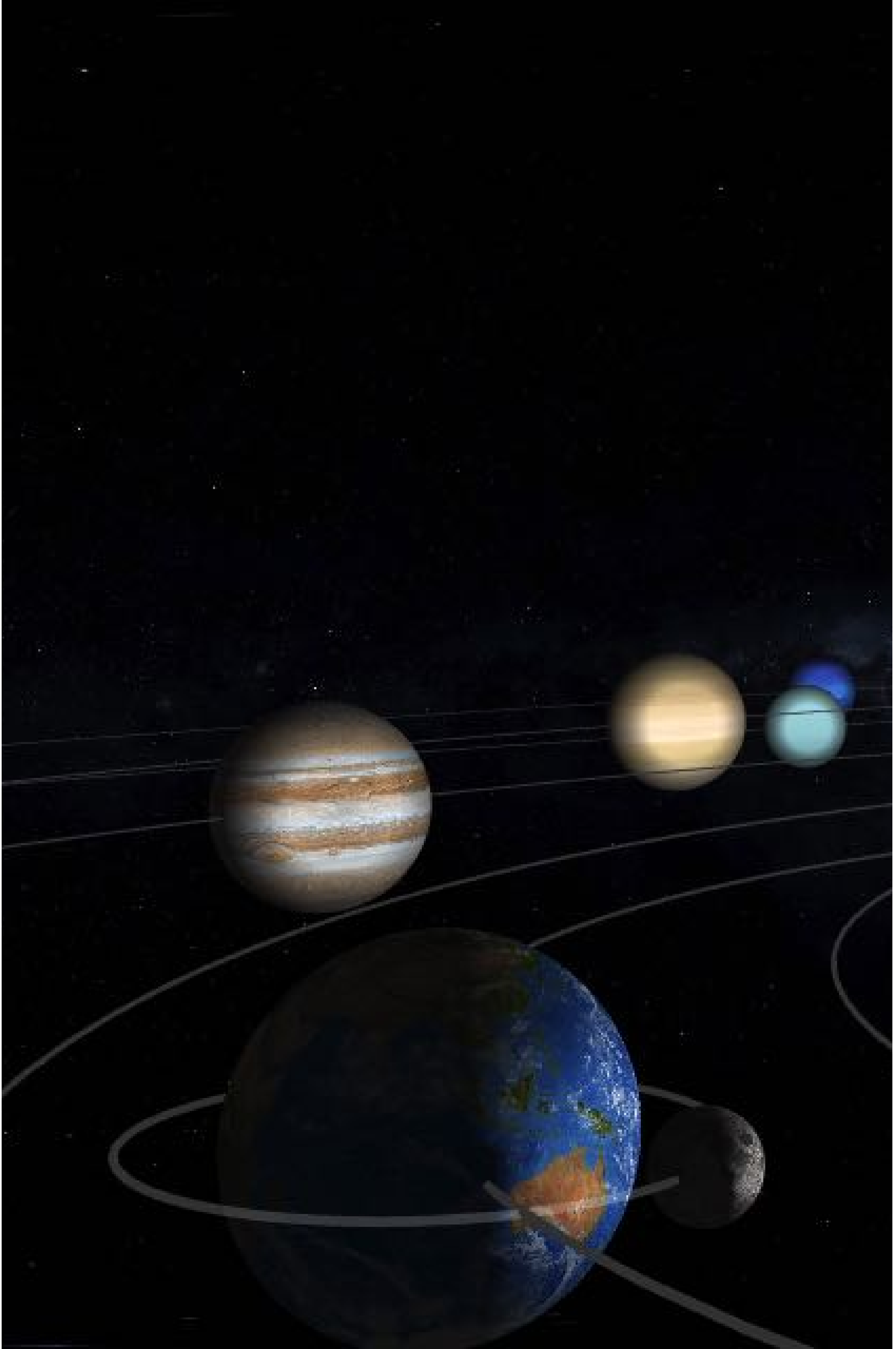


CONTENTS

- Animated solar system model
- Animated Bohr's atomic model
- Celestial body and Atom

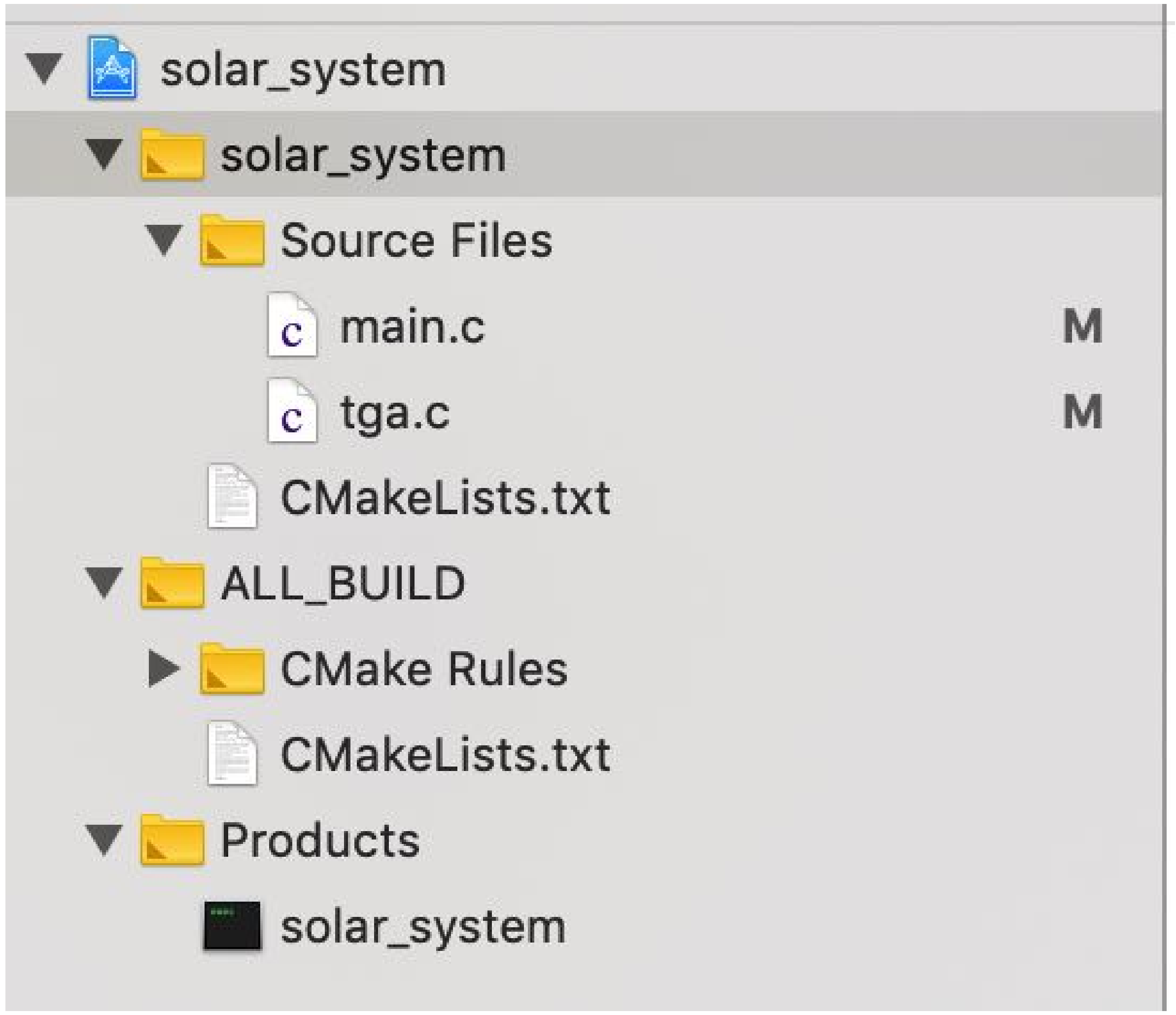
ANIMATED SOLAR SYSTEM MODEL





ANIMATED SOLAR SYSTEM MODEL

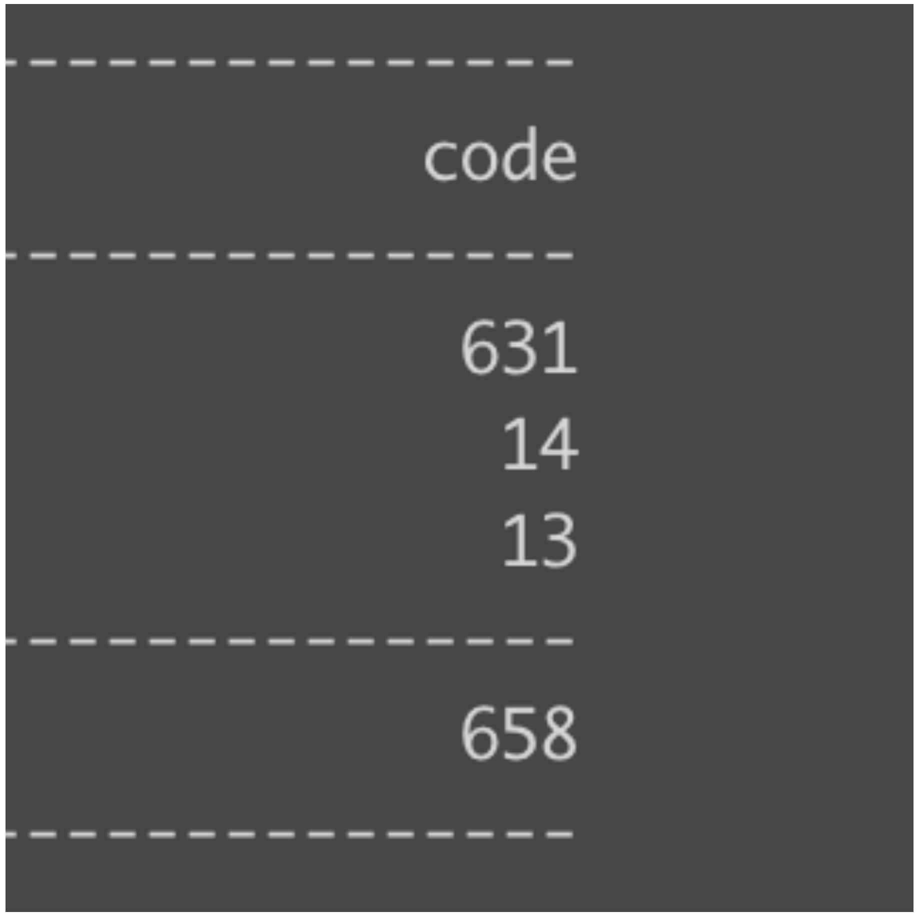
- platform: macOS
- ide: Xcode-beta
- written in C++
- OpenGL Libraries



```
▼ solar_system
  ▼ solar_system
    ▼ Source Files
        main.c                 M
        tga.c                  M
      CMakeLists.txt
  ▼ ALL_BUILD
    ▶ CMake Rules
      CMakeLists.txt
  ▼ Products
      solar_system
```

ANIMATED SOLAR SYSTEM MODEL

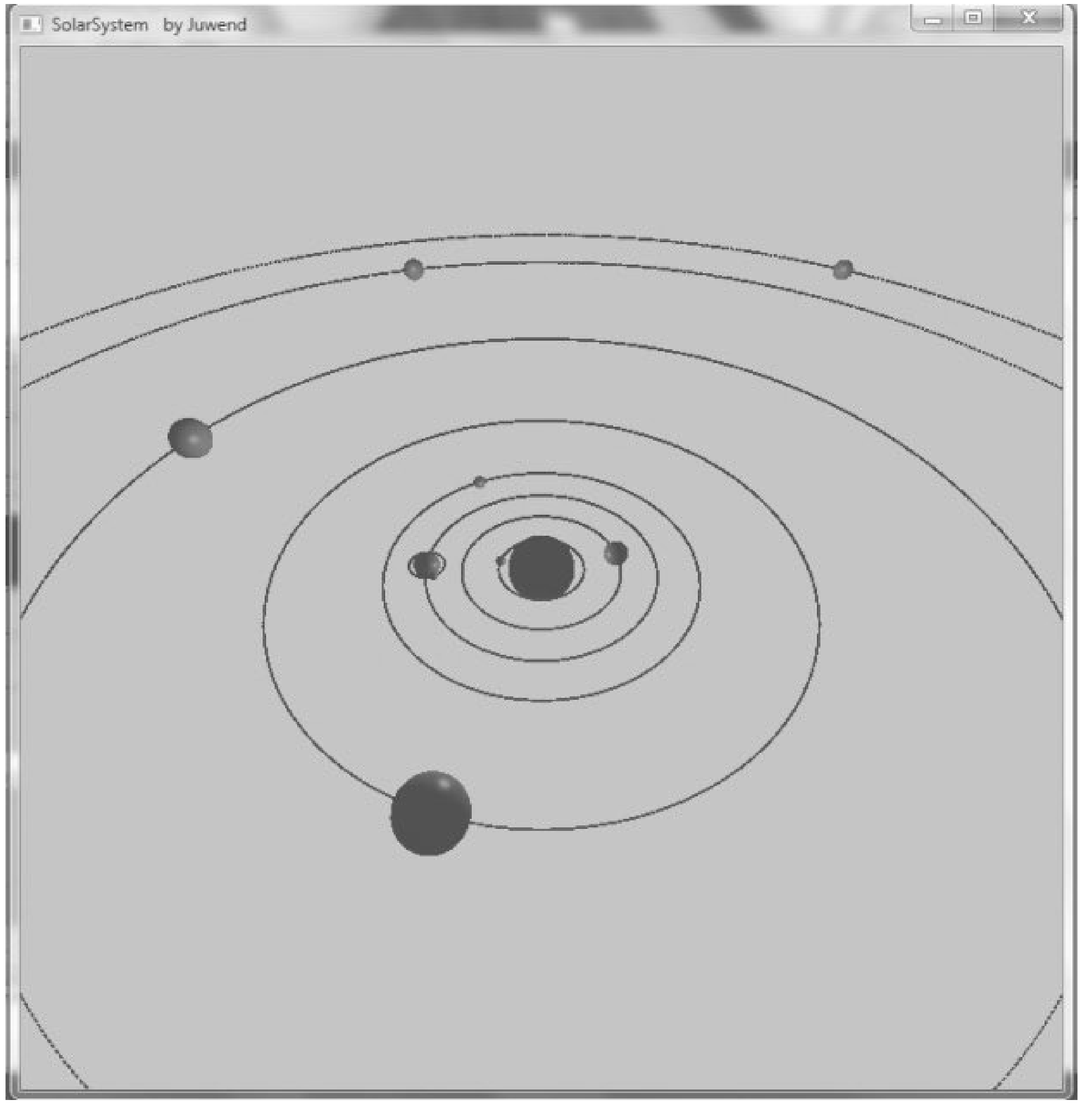
- Glitter(http://polytonic.github.io/Glitter/)
- GLUT
- GLAD
- GLFW

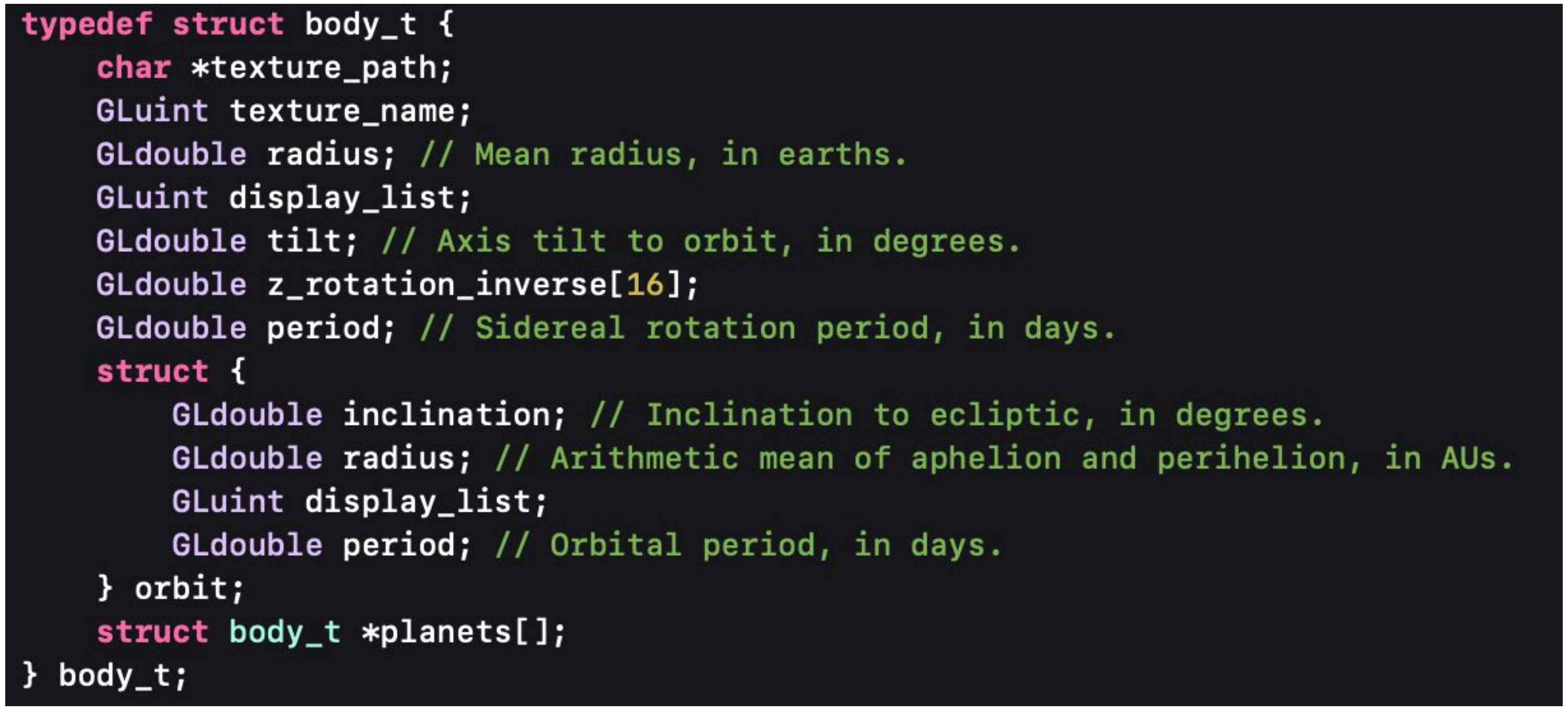


```
--------------
          code
--------------
           631
            14
            13
--------------
           658
--------------
```

NECESSARY ELEMENTS

- ➤ Sun, eight planets and moon.
- ➤ Their Orbits

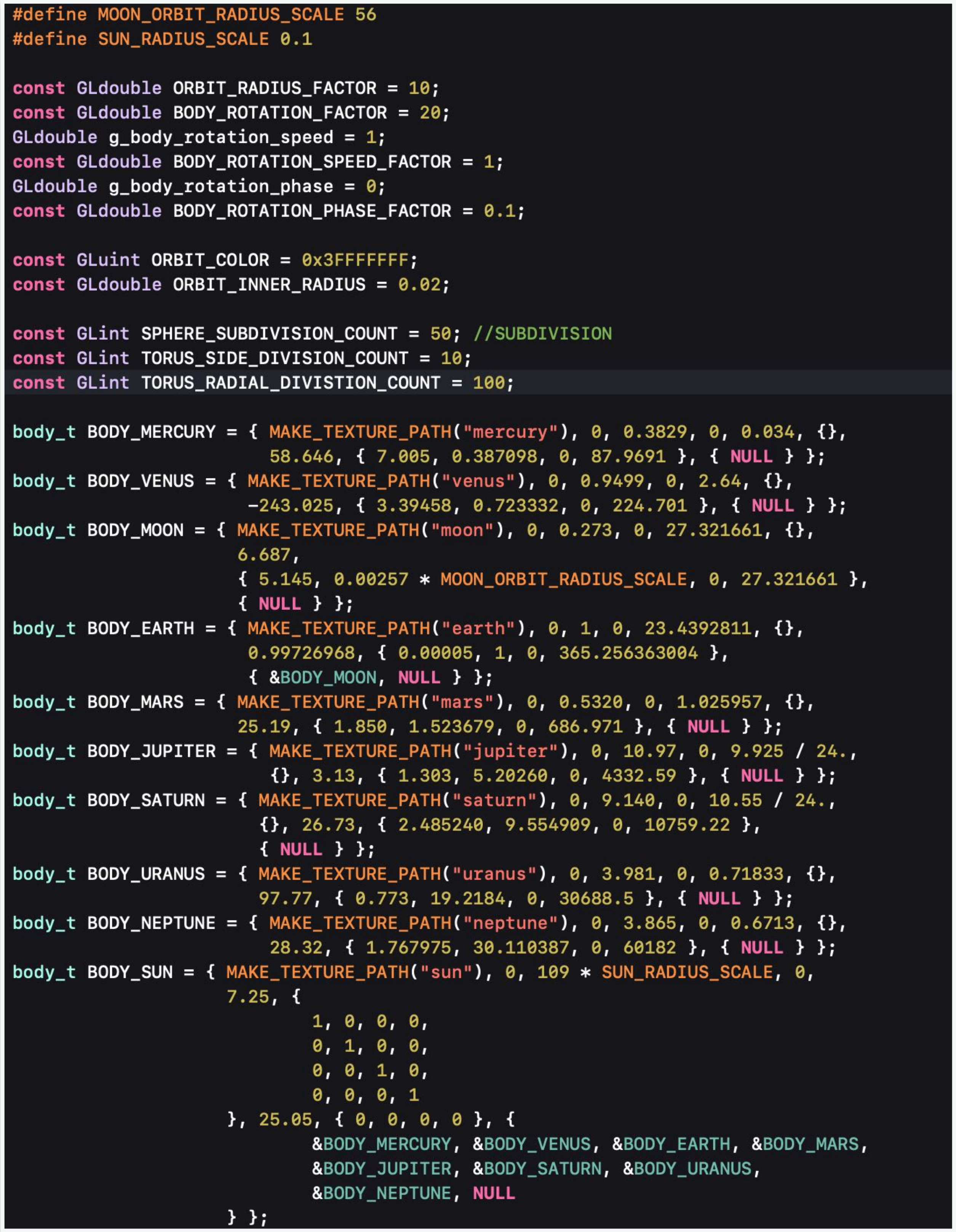




```
typedef struct body_t {
    char *texture_path;
    GLuint texture_name;
    GLdouble radius; // Mean radius, in earths.
    GLuint display_list;
    GLdouble tilt; // Axis tilt to orbit, in degrees.
    GLdouble z_rotation_inverse[16];
    GLdouble period; // Sidereal rotation period, in days.
    struct {
        GLdouble inclination; // Inclination to ecliptic, in degrees.
        GLdouble radius; // Arithmetic mean of aphelion and perihelion, in AUs.
        GLuint display_list;
        GLdouble period; // Orbital period, in days.
    } orbit;
    struct body_t *planets[];
} body_t;
```

TO MAKE IT MORE REALISTIC & SCIENTIFIC…

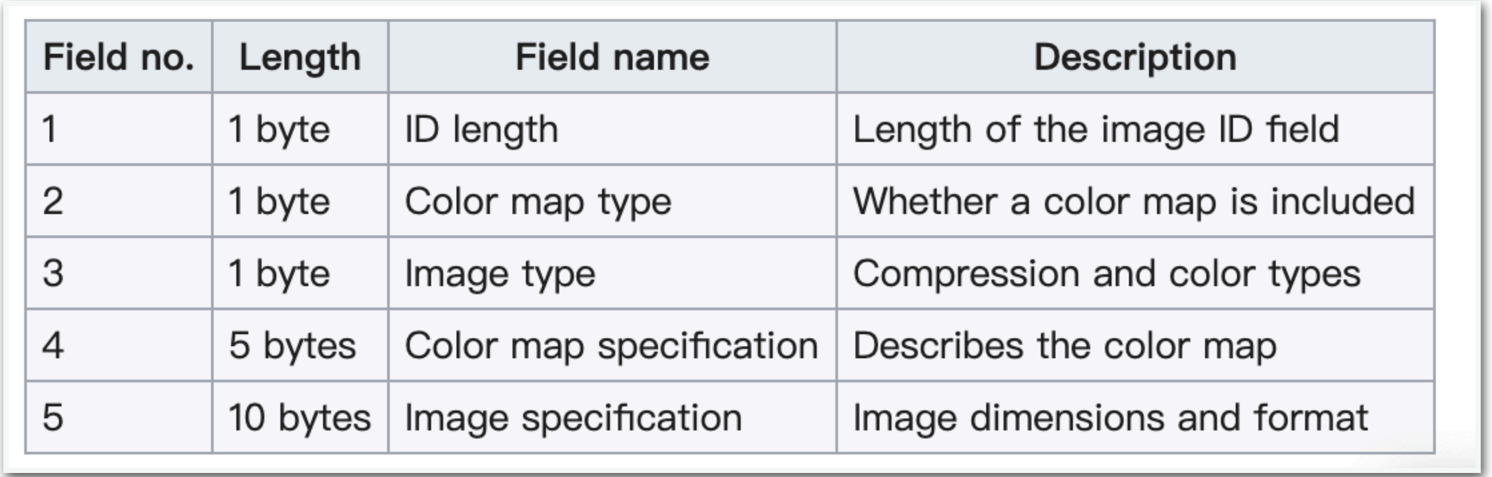
- reliable data
  - from Wikipedia
- texture
  - TGA image
    - transformation
  - from
- lighting



```c
#define MOON_ORBIT_RADIUS_SCALE 56
#define SUN_RADIUS_SCALE 0.1

const GLdouble ORBIT_RADIUS_FACTOR = 10;
const GLdouble BODY_ROTATION_FACTOR = 20;
GLdouble g_body_rotation_speed = 1;
const GLdouble BODY_ROTATION_SPEED_FACTOR = 1;
GLdouble g_body_rotation_phase = 0;
const GLdouble BODY_ROTATION_PHASE_FACTOR = 0.1;

const GLuint ORBIT_COLOR = 0x3FFFFFFF;
const GLdouble ORBIT_INNER_RADIUS = 0.02;

const GLint SPHERE_SUBDIVISION_COUNT = 50; //SUBDIVISION
const GLint TORUS_SIDE_DIVISION_COUNT = 10;
const GLint TORUS_RADIAL_DIVISTION_COUNT = 100;

body_t BODY_MERCURY = { MAKE_TEXTURE_PATH("mercury"), 0, 0.3829, 0, 0.034, {},
                        58.646, { 7.005, 0.387098, 0, 87.9691 }, { NULL } };
body_t BODY_VENUS = { MAKE_TEXTURE_PATH("venus"), 0, 0.9499, 0, 2.64, {},
                      -243.025, { 3.39458, 0.723332, 0, 224.701 }, { NULL } };
body_t BODY_MOON = { MAKE_TEXTURE_PATH("moon"), 0, 0.273, 0, 27.321661, {},
                     6.687,
                     { 5.145, 0.00257 * MOON_ORBIT_RADIUS_SCALE, 0, 27.321661 },
                     { NULL } };
body_t BODY_EARTH = { MAKE_TEXTURE_PATH("earth"), 0, 1, 0, 23.4392811, {},
                      0.99726968, { 0.00005, 1, 0, 365.256363004 },
                      { &BODY_MOON, NULL } };
body_t BODY_MARS = { MAKE_TEXTURE_PATH("mars"), 0, 0.5320, 0, 1.025957, {},
                     25.19, { 1.850, 1.523679, 0, 686.971 }, { NULL } };
body_t BODY_JUPITER = { MAKE_TEXTURE_PATH("jupiter"), 0, 10.97, 0, 9.925 / 24.,
                        {}, 3.13, { 1.303, 5.20260, 0, 4332.59 }, { NULL } };
body_t BODY_SATURN = { MAKE_TEXTURE_PATH("saturn"), 0, 9.140, 0, 10.55 / 24.,
                       {}, 26.73, { 2.485240, 9.554909, 0, 10759.22 },
                       { NULL } };
body_t BODY_URANUS = { MAKE_TEXTURE_PATH("uranus"), 0, 3.981, 0, 0.71833, {},
                       97.77, { 0.773, 19.2184, 0, 30688.5 }, { NULL } };
body_t BODY_NEPTUNE = { MAKE_TEXTURE_PATH("neptune"), 0, 3.865, 0, 0.6713, {},
                        28.32, { 1.767975, 30.110387, 0, 60182 }, { NULL } };
body_t BODY_SUN = { MAKE_TEXTURE_PATH("sun"), 0, 109 * SUN_RADIUS_SCALE, 0,
                    7.25, {
                            1, 0, 0, 0,
                            0, 1, 0, 0,
                            0, 0, 1, 0,
                            0, 0, 0, 1
                    }, 25.05, { 0, 0, 0, 0 }, {
                            &BODY_MERCURY, &BODY_VENUS, &BODY_EARTH, &BODY_MARS,
                            &BODY_JUPITER, &BODY_SATURN, &BODY_URANUS,
                            &BODY_NEPTUNE, NULL
                    } };
```

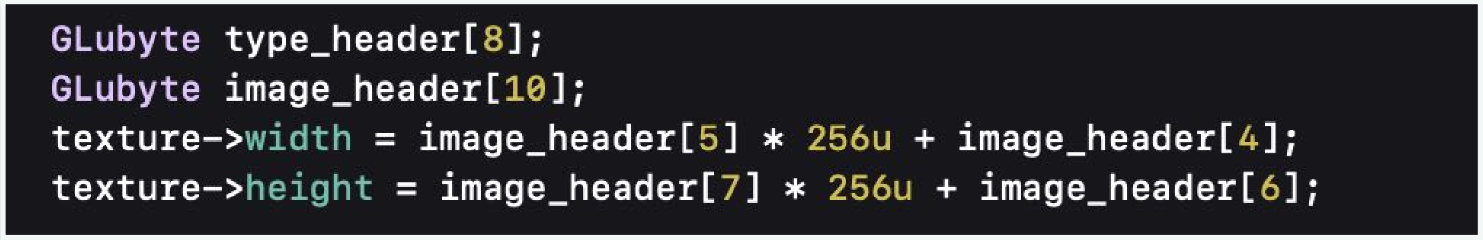
TO MAKE IT MORE REALISTIC & SCIENTIFIC…

- reliable data
  - from Wikipedia



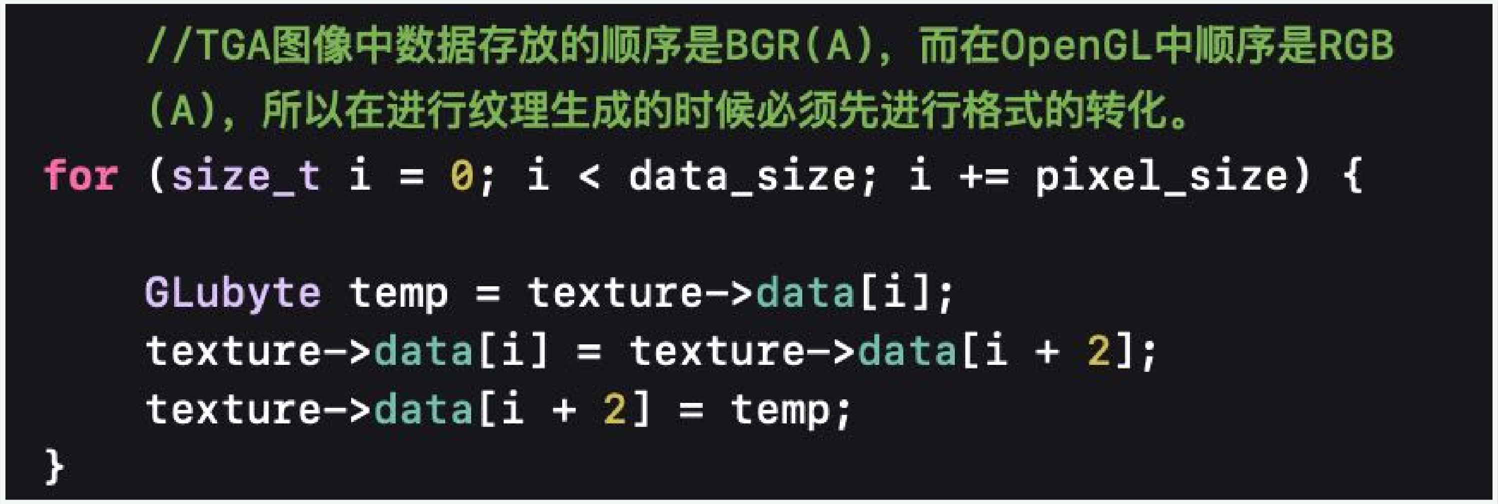
Field no.	Length	Field name	Description
1	1 byte	ID length	Length of the image ID field
2	1 byte	Color map type	Whether a color map is included
3	1 byte	Image type	Compression and color types
4	5 bytes	Color map specification	Describes the color map
5	10 bytes	Image specification	Image dimensions and format

- texture
  - TGA image(TARGA)
    - read in binary format



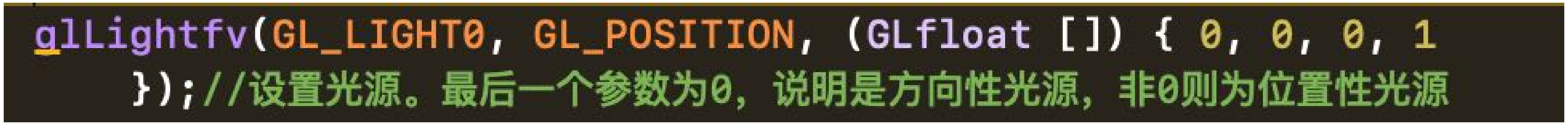
```
GLubyte type_header[8];
GLubyte image_header[10];
texture->width = image_header[5] * 256u + image_header[4];
texture->height = image_header[7] * 256u + image_header[6];
```

  - pictures from solarsystemscope.com
- lighting

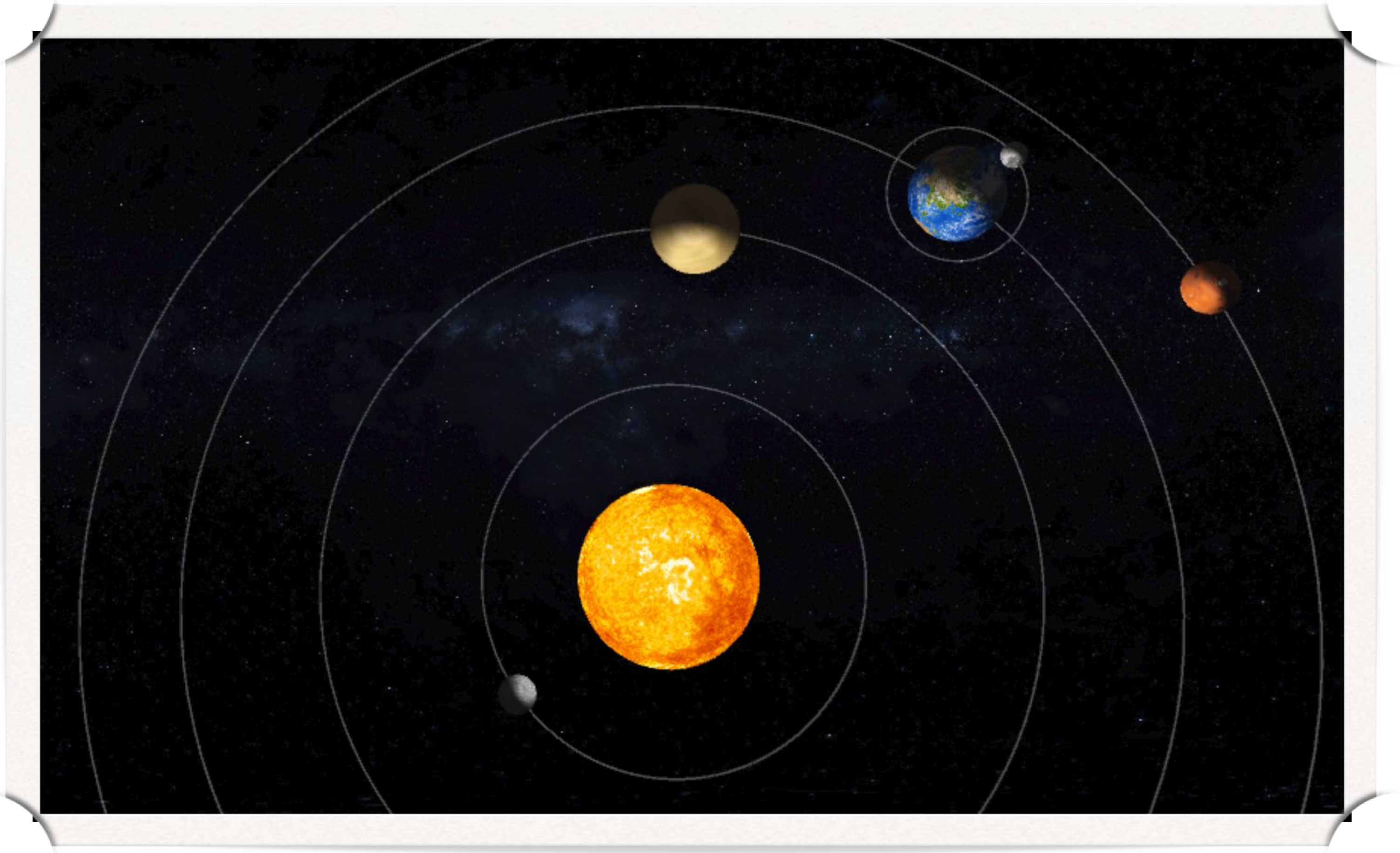


```
    //TGA图像中数据存放的顺序是BGR(A)，而在OpenGL中顺序是RGB(A)，所以在进行纹理生成的时候必须先进行格式的转化。
for (size_t i = 0; i < data_size; i += pixel_size) {

    GLubyte temp = texture->data[i];
    texture->data[i] = texture->data[i + 2];
    texture->data[i + 2] = temp;
}
```

TO MAKE IT MORE REALISTIC & SCIENTIFIC…

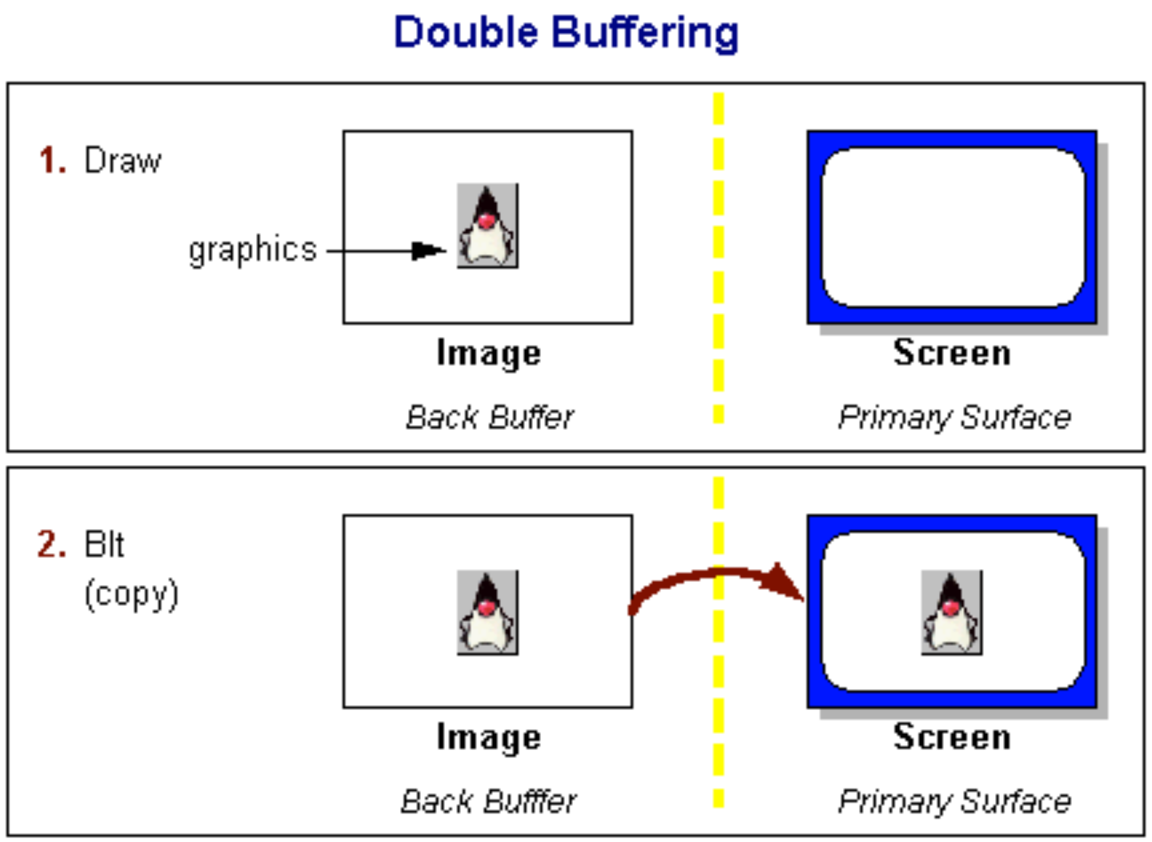
- reliable data
  - from Wikipedia
- texture
  - TGA image
    - transformation
  - from
- lighting
  - 

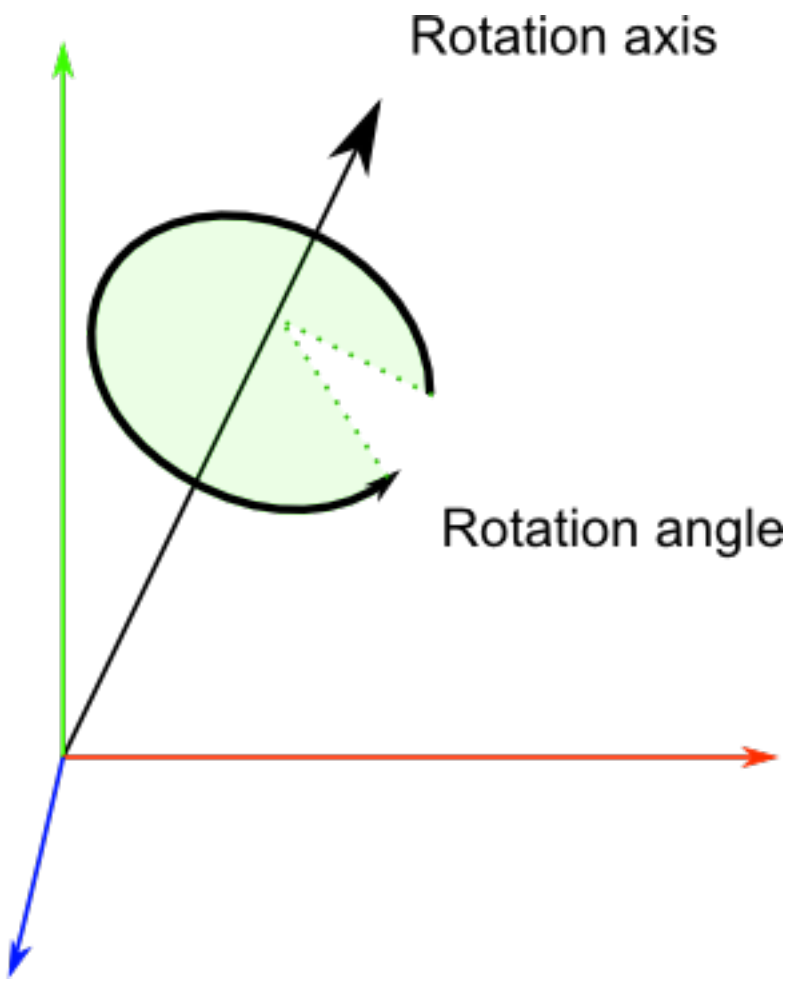
```
glLightfv(GL_LIGHT0, GL_POSITION, (GLfloat []) { 0, 0, 0, 1
    });//设置光源。最后一个参数为0，说明是方向性光源，非0则为位置性光源
```



THEN MAKE IT DYNAMIC !

- Double Buffering
- Rotation and Revolution
- Transformation and Viewing
  - camera moving



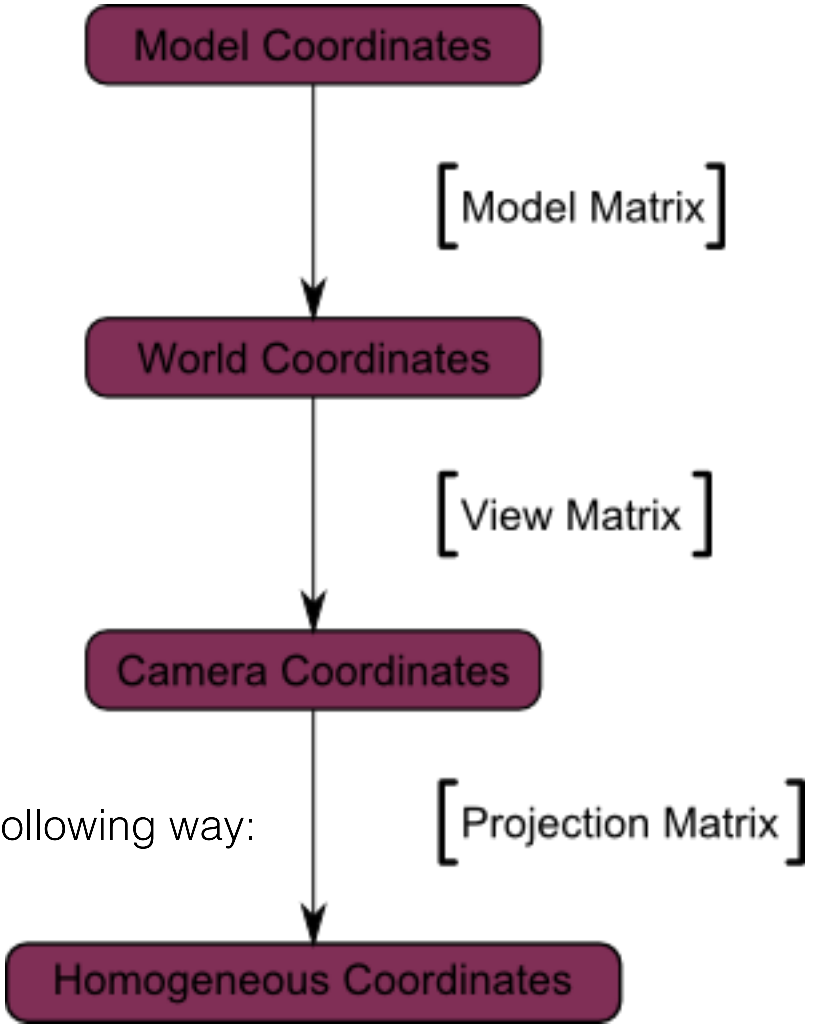


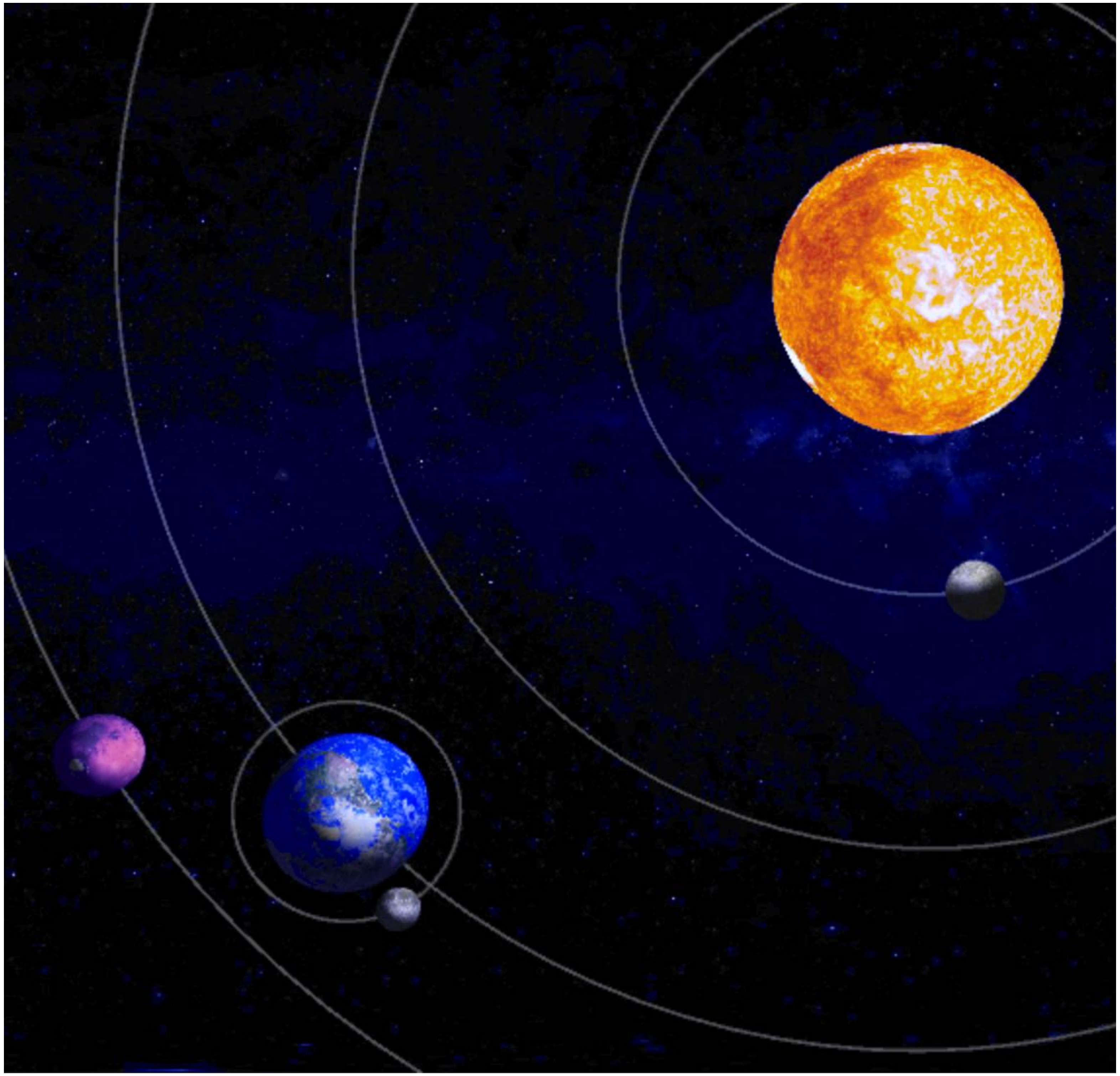
THEN MAKE IT DYNAMIC !

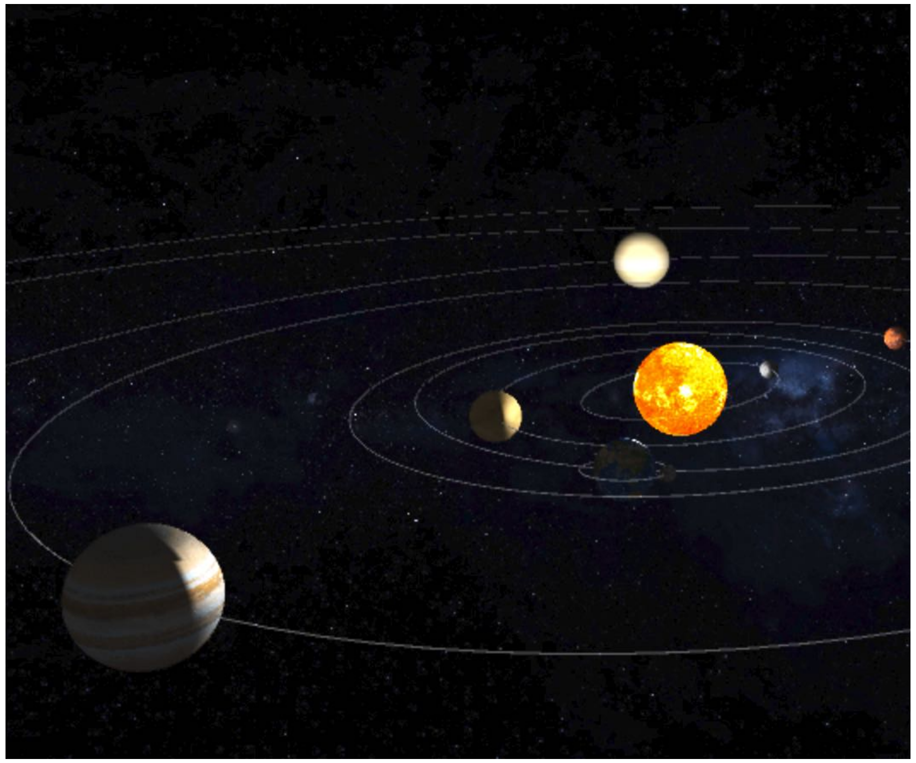
- Double Buffering
- Rotation and Revolution
- Transformation and Viewing
  - Camera Moving
  - Quaternions

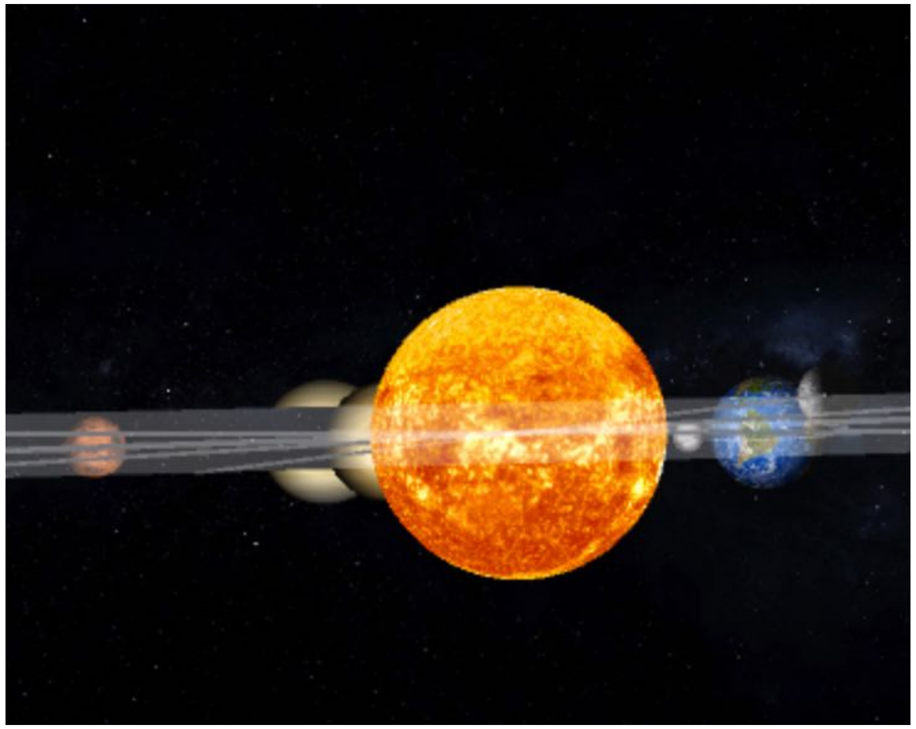
A quaternion is a set of 4 numbers, [x y z w], which represents rotations the following way:

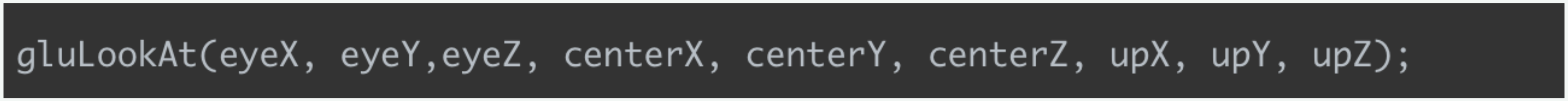
```
// RotationAngle is in radians
x = RotationAxis.x * sin(RotationAngle / 2)
y = RotationAxis.y * sin(RotationAngle / 2)
z = RotationAxis.z * sin(RotationAngle / 2)
w = cos(RotationAngle / 2)
```





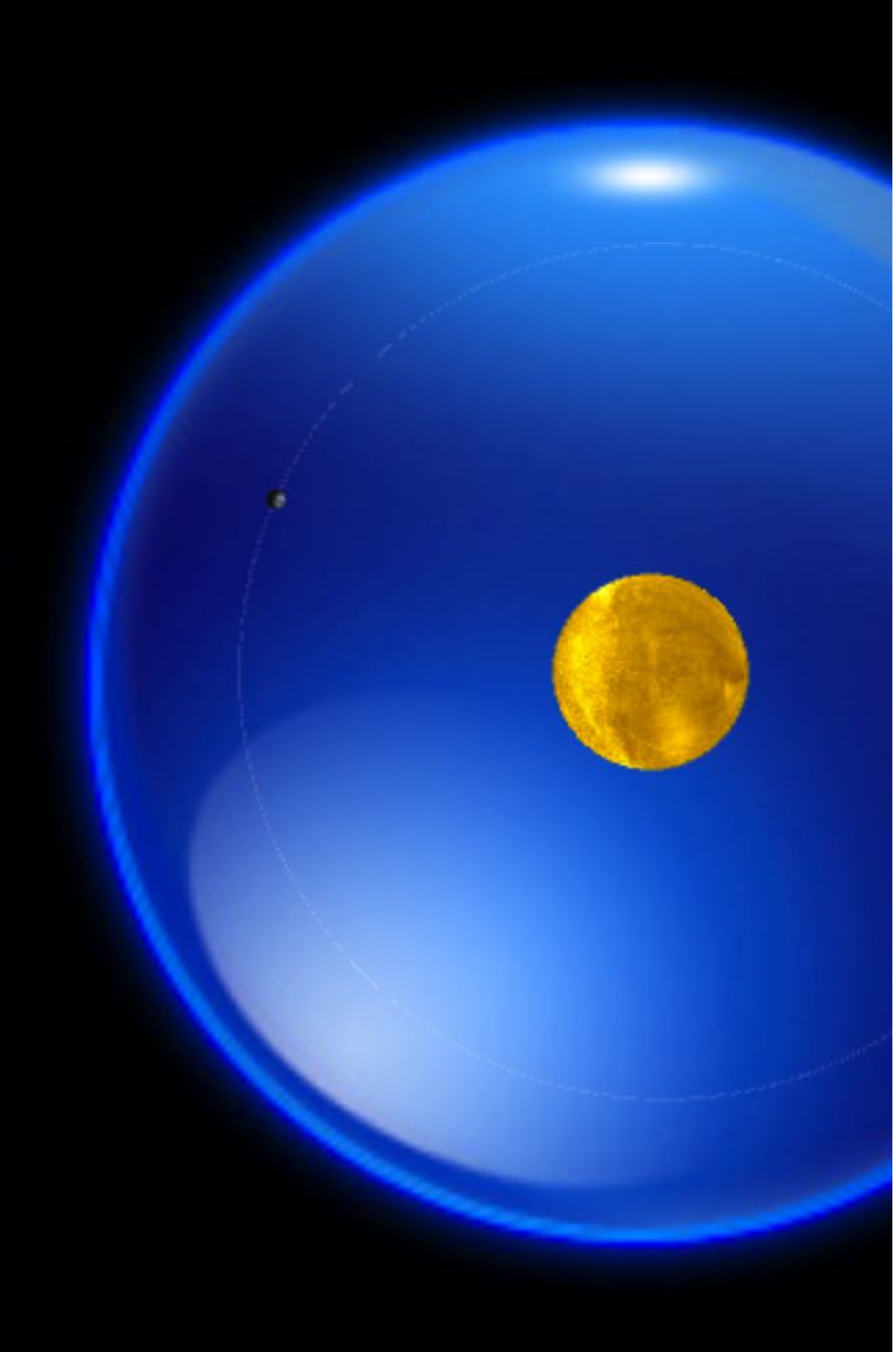






```
gluLookAt(eyeX, eyeY,eyeZ, centerX, centerY, centerZ, upX, upY, upZ);
```

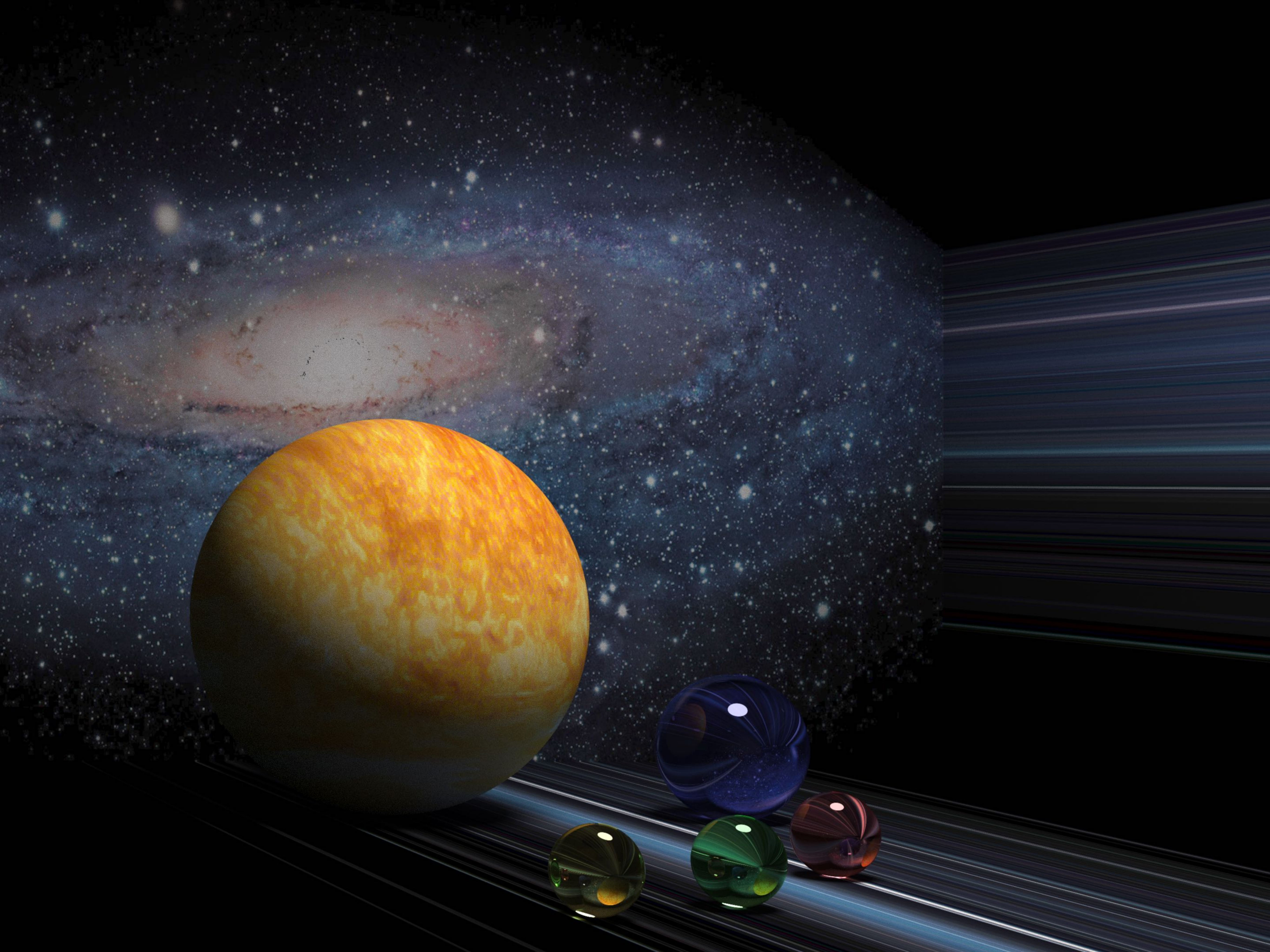
ANIMATED BOHR'S ATOMIC MODEL

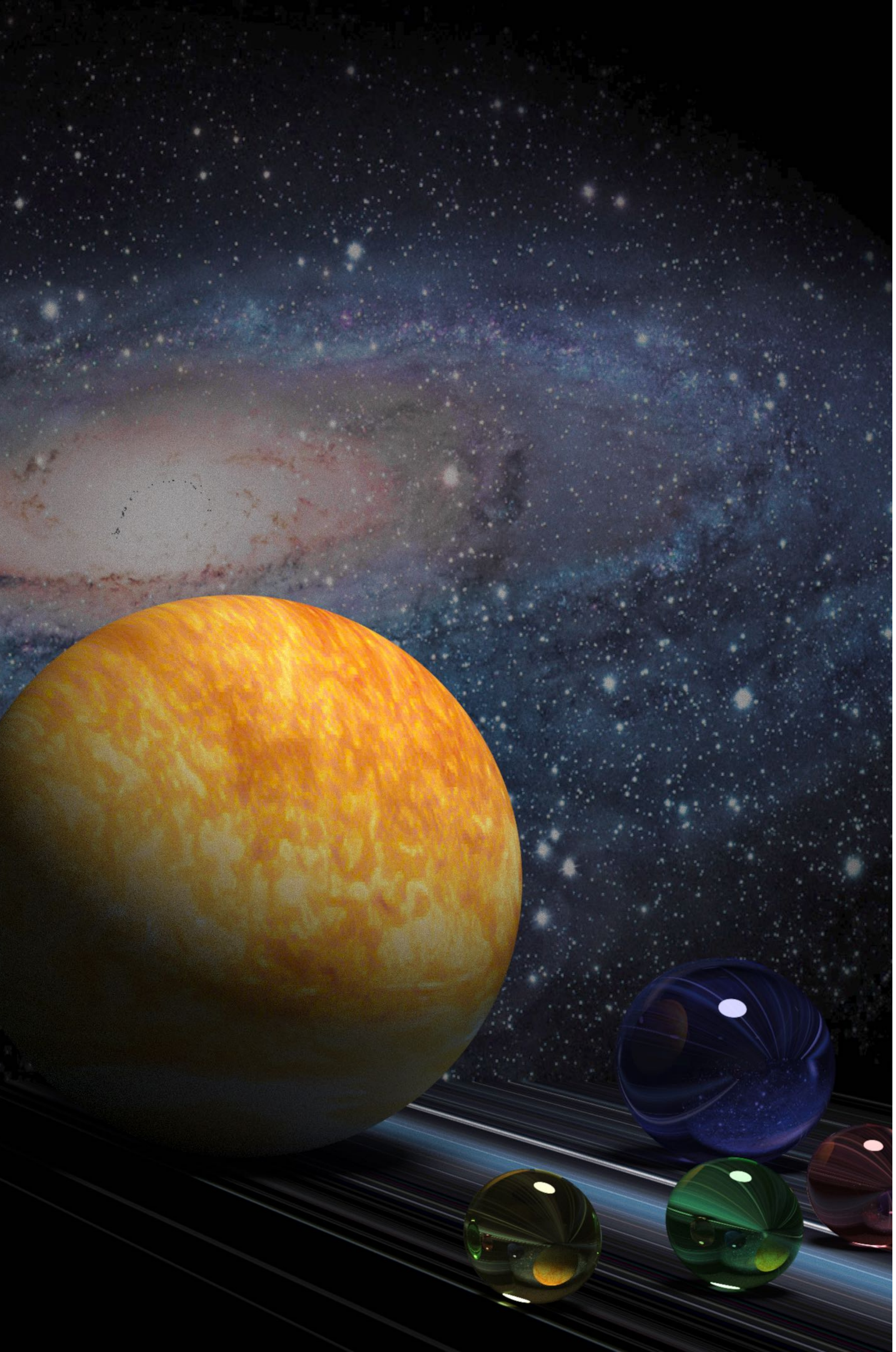


ANIMATED BOHR'S ATOMIC MODEL

- ➤ Glitter
- ➤ GLUT
- ➤ GLAD
- ➤ GLFW
- ➤ GLM(OpenGL Mathematics)

CELESTIAL BODY AND ATOM





CELESTIAL BODY AND ATOM

- written in the OpenGL Shading Language (GLSL)
- Run on GPU

- Color Blending
- Path Tracing
- Texture Mapping
- Depth of Field

REFERENCES

- learn OpenGL (https://learnopengl.com/)
- Wikipedia (https://en.wikipedia.org/)
- OpenGL实现太阳系模型(https://www.juwends.com/tech/opengl/opengl-solar-system.html)
- Are Atoms Solar Systems? (https://www.youtube.com/watch?v=SjTk7OGNxqg)
- Computational Graphics(https://github.com/Trinkle23897/Computational-Graphics-THU-2018)

THANKS FOR YOUR WATCHING

Wanru Zhao 17373240

School of computer science and engineering,BUAA