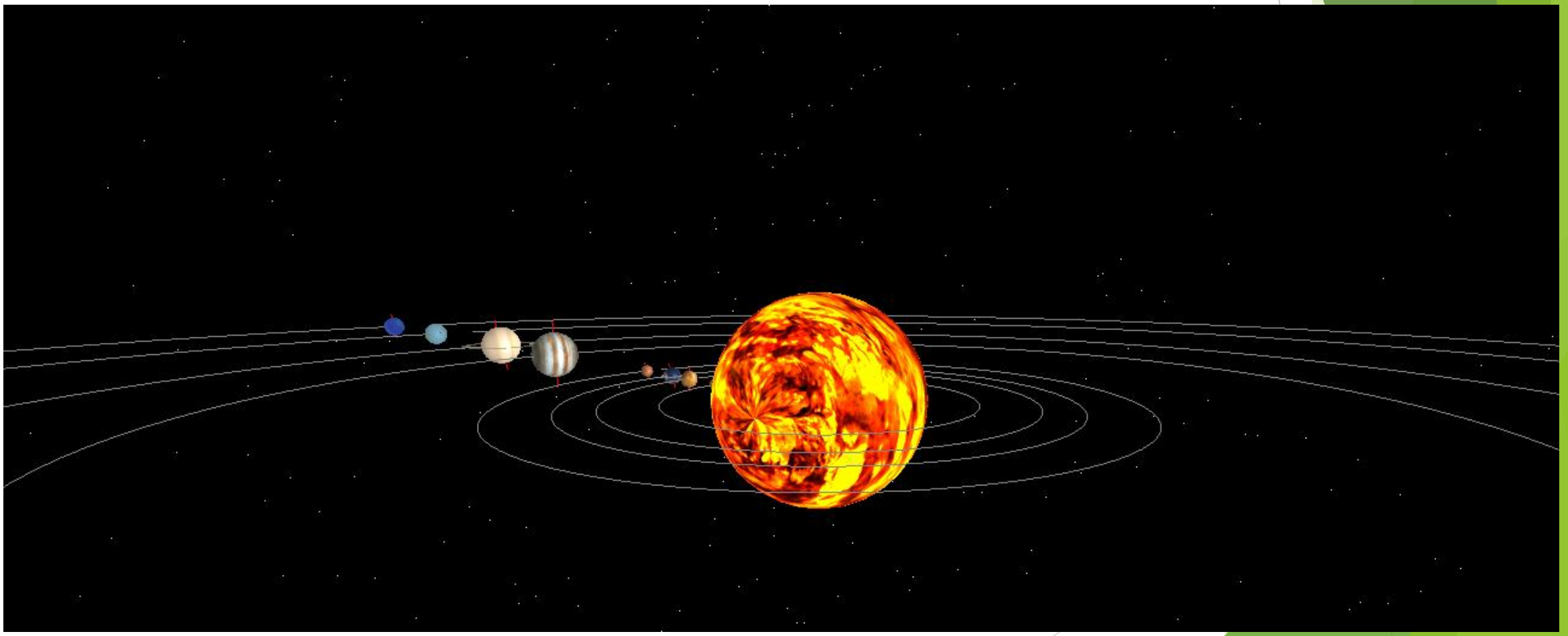
太阳系模拟系统

太阳系的模拟



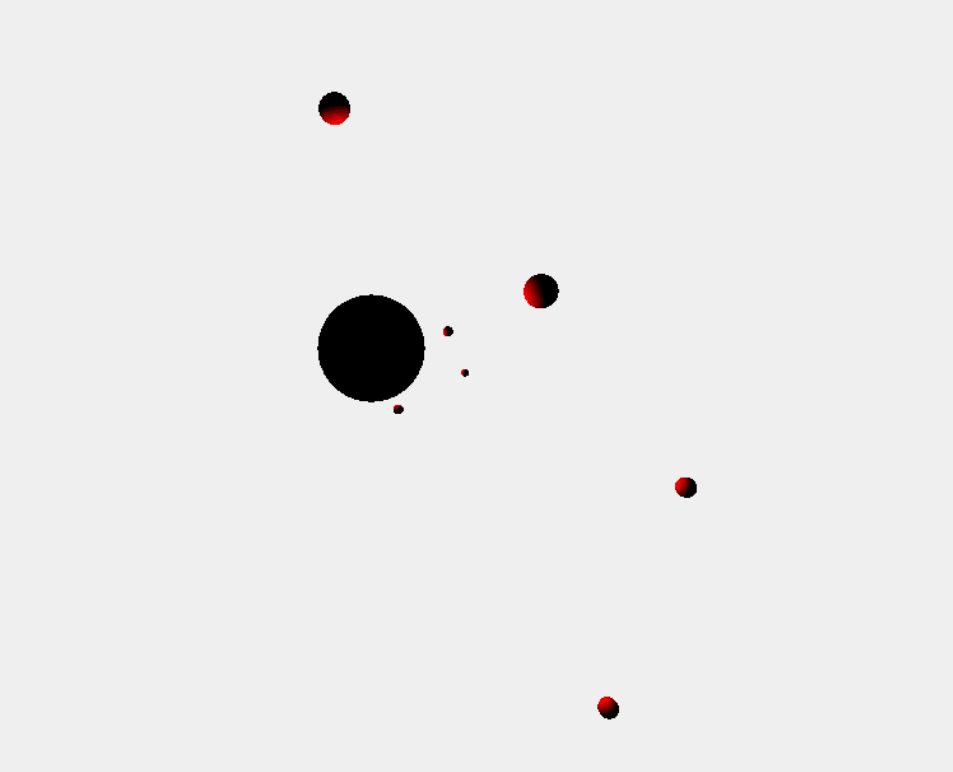
数据

- 距太阳的距离（月球是距地球的距离）
- 自转周期
- 公转周期
- 轨道倾角
- 星球半径
- 纹理（在一本书上找到了）
- 星球质量（没用上）

R = sqrt(R / 100km) px
D = sqrt(D / 100 0000 km) * 16 px

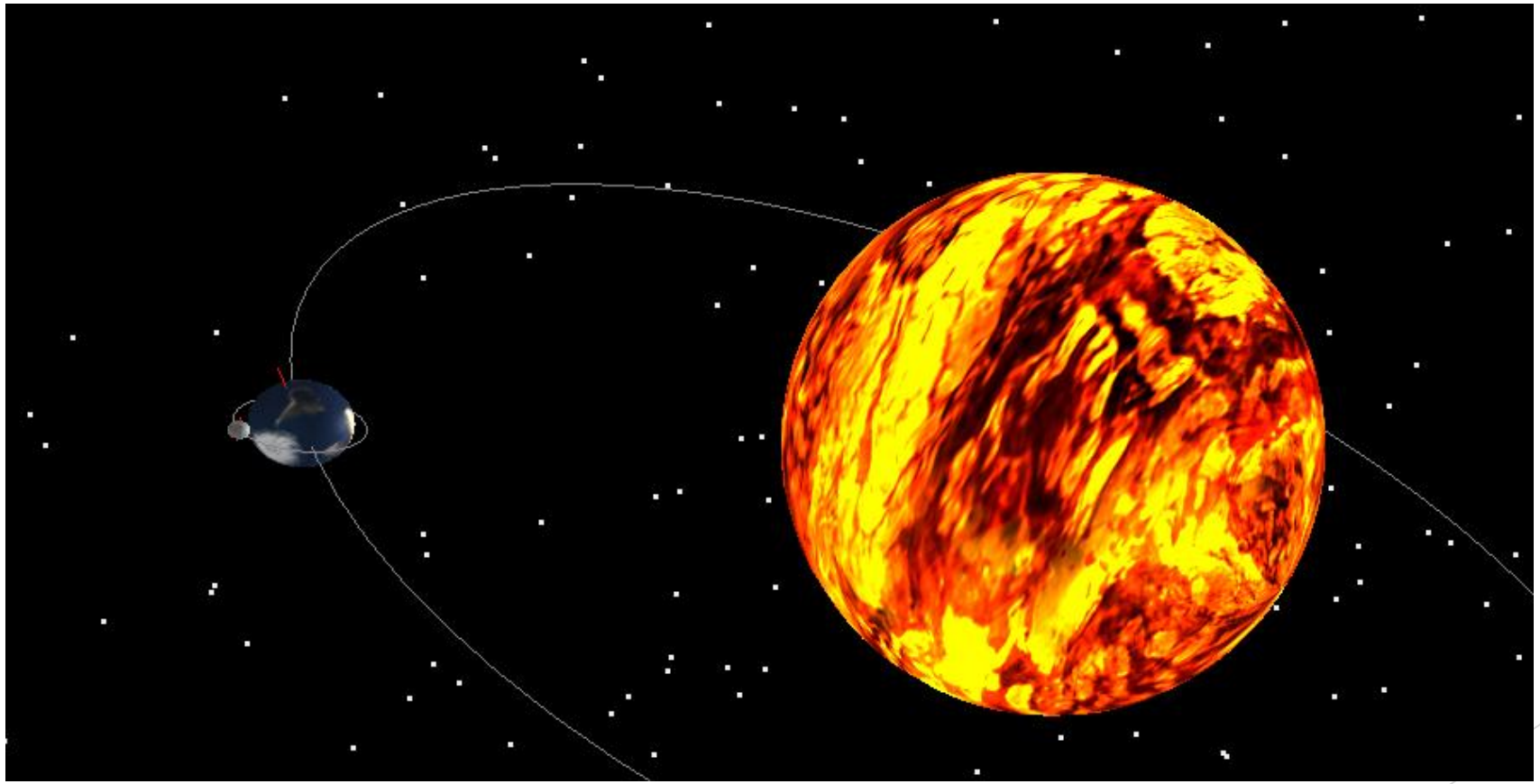
运动 层次化

- 行星运动由自传和公转两种运动组成
- 在太阳系中，月亮的运动是最为复杂的



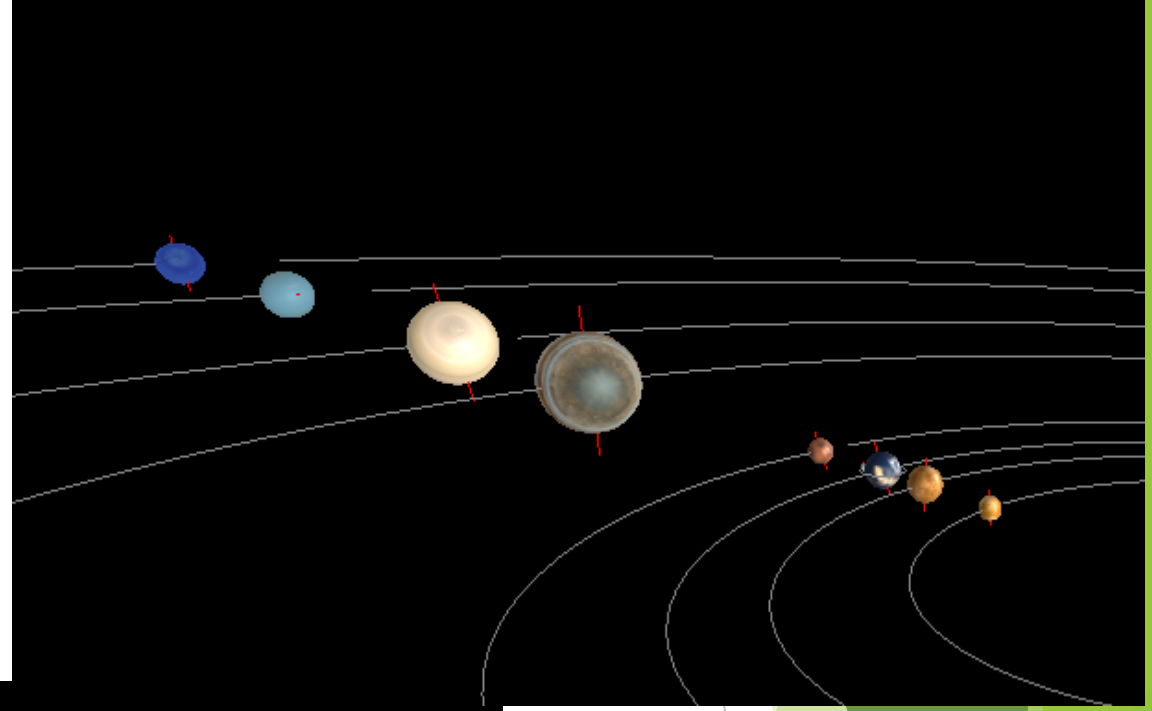
```
var earthGroup, earth, moonGroup, moon

moonGroup.add(moon);
earthGroup.add(earth);
earthGroup.add(moonGroup);

moon.rotate();
earth.rotate();
moonGroup.rotate();
earthGroup.rotate();
```

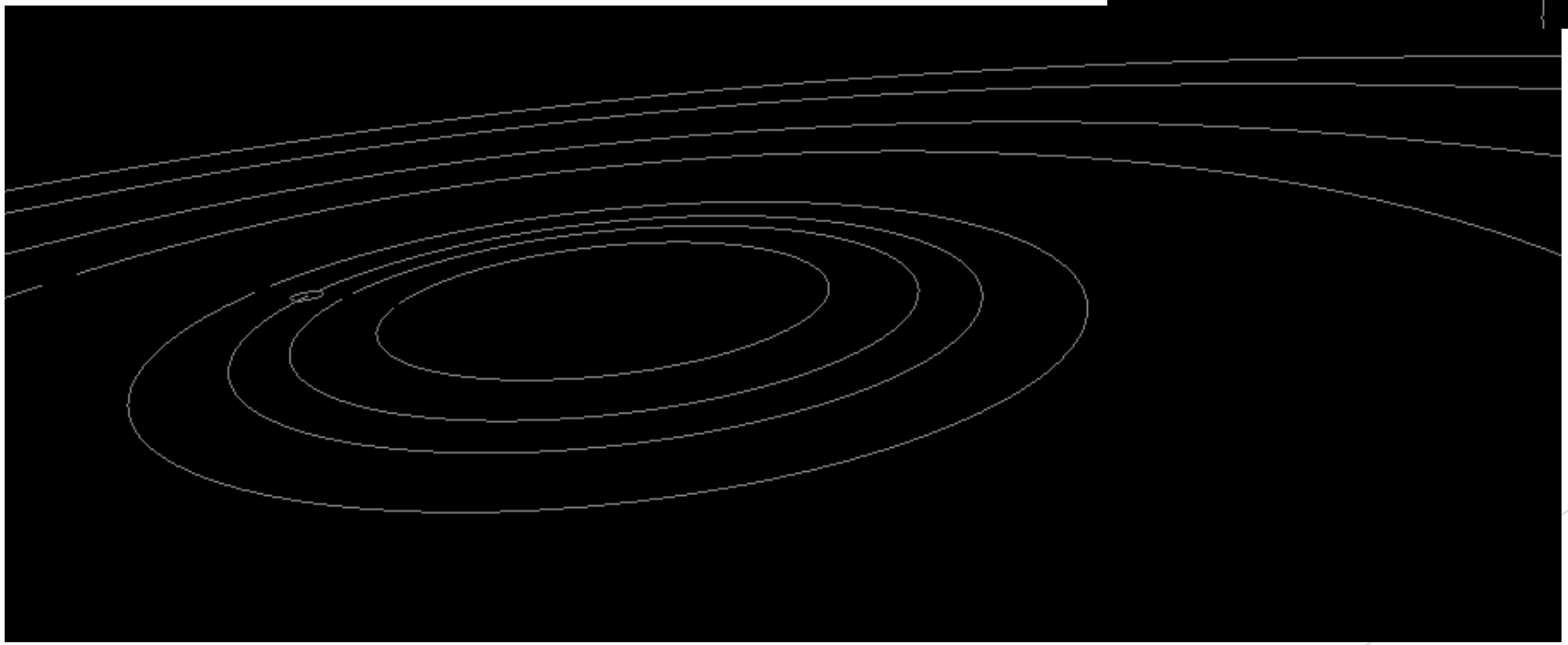
日地月三星系统



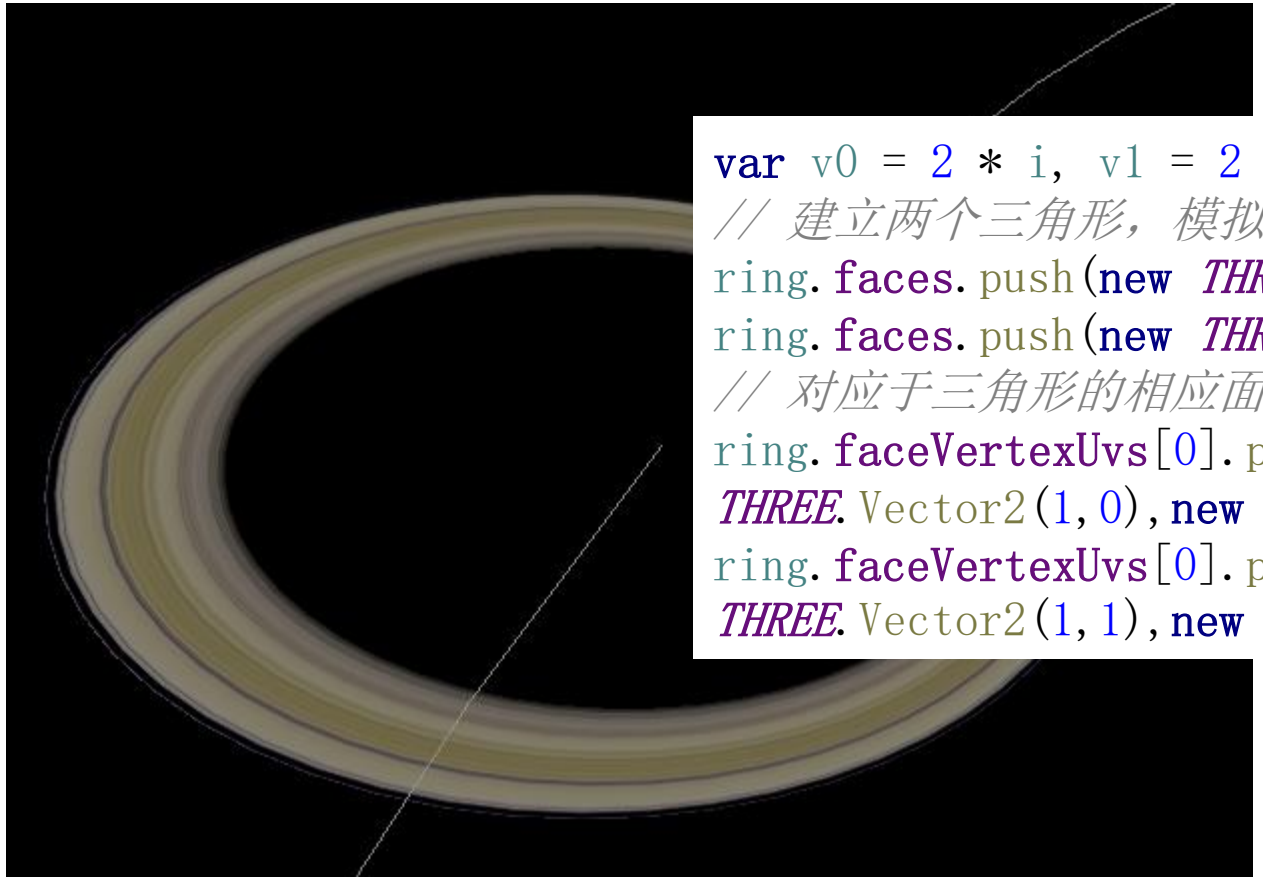
行星绘制

- 纹理
- 行星轨道与自转轴
- 行星运动



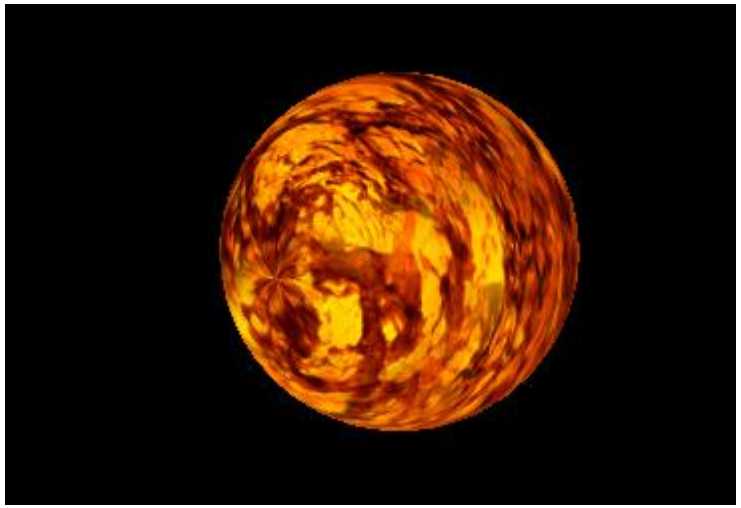


土星 土星环

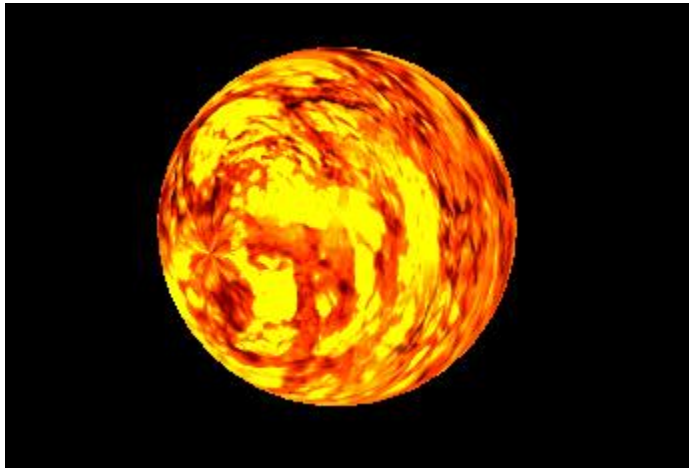


```
var v0 = 2 * i, v1 = 2 *i + 1, v2 = (2 * i + 2), v3 = (2 * i + 3);
// 建立两个三角形，模拟圆环的一部分
ring.faces.push(new THREE.Face3(v0, v1, v2));
ring.faces.push(new THREE.Face3(v1, v3, v2));
// 对应于三角形的相应面的顶点的纹理坐标（）
ring.faceVertexUvs[0].push([new THREE.Vector2(0,0),new
THREE.Vector2(1,0),new THREE.Vector2(0,1)]);
ring.faceVertexUvs[0].push([new THREE.Vector2(1,0),new
THREE.Vector2(1,1),new THREE.Vector2(0,1)]);
```

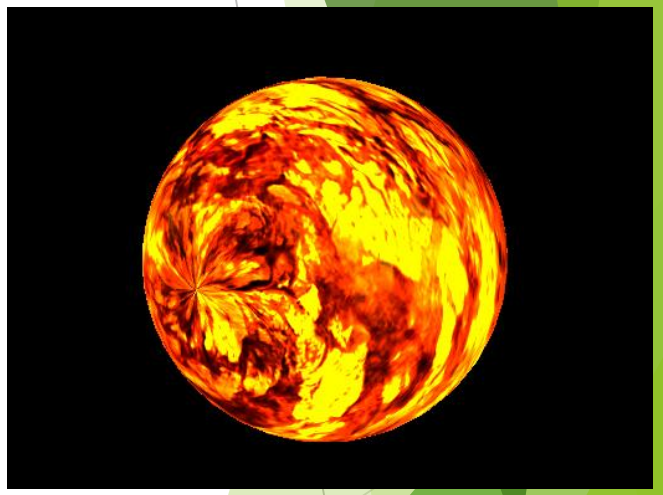
太阳 shader



```
gl_FragColor = temp
```



```
temp = temp * 2.0;
gl_FragColor = temp;
```

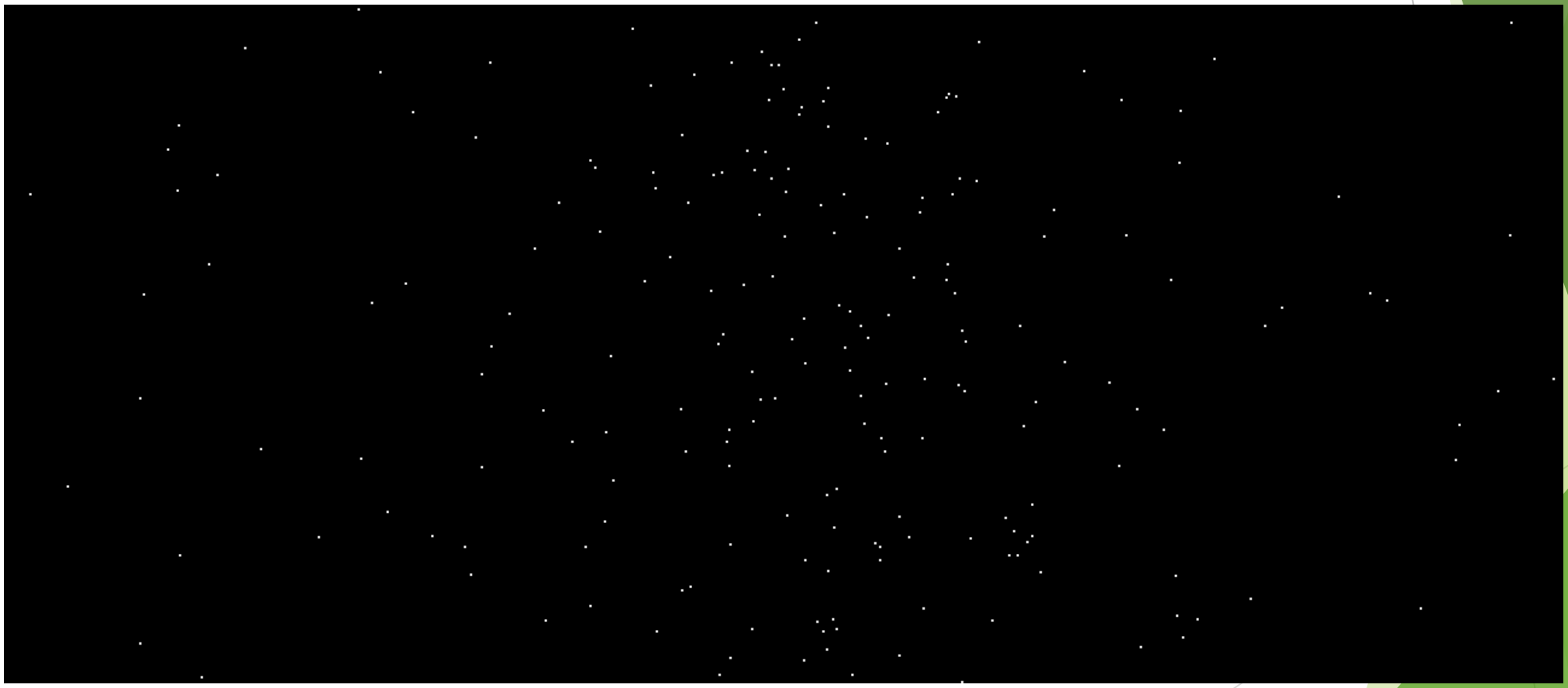


来自网络代码

引入太阳黑子运动的shader

```
vec4 noise = texture2D( texture1, texCoord );

vec2 T1 = texCoord + vec2( 1.5, -1.5 ) * time  * 0.01;
vec2 T2 = texCoord + vec2( -0.5, 2.0 ) * time *  0.01;

T1.x -= noise.r * 2.0;
T1.y += noise.g * 4.0;
T2.x += noise.g * 0.2;
T2.y += noise.b * 0.2;

float p = texture2D( texture1, T1 * 2.0 ).a + 0.25;

vec4 color = texture2D( texture2, T2 );
vec4 temp = color * 2.0 * ( vec4( p, p, p, p ) ) + ( color * color );
gl_FragColor = temp;
```

群星



交互

- Three.js官方的traceback相机，并进行了些许改动
- 多视角切换(虚实两套结构的切换)

问题

- 最大的问题就是学习的书中用的three.js和开发使用的版本不同，很多函数不同
- TextureLoader 代替了 ImageUtil
- 自定义的几何体去除了vextex的定义
- Shader编程的uniform的参数发生变化