REPORT

중간대체과제 결과 – Solar System 구현

교 과 목	컴퓨터그래픽스
분 반	3
담 당 교 수	송인식 교수님
학 과	소프트웨어학과
학 번	32141868
이 름	박유현
제 출 일	2020.06.19

목 차

1. 코드 구현

Three.js를 사용하여 구현하였으며 코드분량이 많아 중요 코드 일부만을 설명하겠습니다.

1.1 변수의 초기화

```
var pointLight, sun, mercury, venus, moon, earth, mars, jupiter, saturn, uranus, neptune,
mercuryOrbit, venusOrbit, earthOrbit, marsOrbit, jupiterOrbit, saturnOrbit, uranusOrbit, neptuneOrbit,
controls, scene, camera, renderer, scene;
var planetSegments = 48;
var mercuryData = constructPlanetData(87.969, 0.00025, 4, "mercury", "img/mercury.jpg", 0.0383, planetSegments);
var venusData = constructPlanetData(224.7, 0.00006, 7, "venus", "img/venus.jpg", 0.0949, planetSegments);
var earthData = constructPlanetData(365.2564, 0.015, 10, "earth", "img/earth.jpg", 0.1, planetSegments);
var moonData = constructPlanetData(29.5, 0.01, 0.28, "moon", "img/moon.jpg", 0.05, planetSegments);
var marsData = constructPlanetData(686.971, 0.014, 15, "mars", "img/mars.jpg", 0.0533, planetSegments);
var jupiterData = constructPlanetData(4332.59, 0.036, 52, "jupiter", "img/jupiter.jpg", 1.1209, planetSegments);
var saturnData = constructPlanetData(10756.199, 0.033, 95, "saturn", "img/saturn.jpg", 0.9449, planetSegments);
var uranusData = constructPlanetData(30707.489, 0.02, 192, "uranus", "img/uranus.jpg", 0.4007, planetSegments);
var neptuneData = constructPlanetData(60223.353, 0.022, 301, "neptune", "img/neptune.jpg", 0.3883, planetSegments);
var orbitData = {value: 100, runOrbit: true, runRotation: true};
var clock = new THREE.Clock();
```

필요한 각 행성의 공전 주기, 자전 주기, 반지름, 텍스쳐 이미지 등 필요한 데이터를 초기화하는 부분입니다. 각 행성의 물리량은 최대한 실제 태양계 물리량의 스케일을 그대로 구현하였습니다.

1.2 init 메서드

```
function init() {
    // Create the camera that allows us to view into the scene.
    camera = new THREE.PerspectiveCamera(
            34, // field of view
            window.innerWidth / window.innerHeight, // aspect ratio
            1, // near clipping plane
            1000 // far clipping plane
            );
    camera.position.z = 300;
    camera.position.x = -300;
    camera.position.y = 300;
    camera.lookAt(new THREE.Vector3(0, 0, 0));
```

가장 먼저 카메라를 초기화하는 코드입니다.

```
316     // Create the scene that holds all of the visible objects.
317     scene = new THREE.Scene();
318
319     // Create the renderer that controls animation.
320     renderer = new THREE.WebGLRenderer();
321     renderer.setSize(window.innerWidth, window.innerHeight);
322
323     // Attach the renderer to the div element.
324     document.getElementById('webgl').appendChild(renderer.domElement);
```

Three.js의 화면 개념인 scene과 Three.js의 WebGL 부분인 renderer를 초기화하는 코드입니다.

```
340     // Attach the background cube to the scene.
341     scene.background = reflectionCube;
342
343     // Create light from the sun.
344     pointLight = getPointLight(0.3, "rgb(255, 220, 180)");
345     scene.add(pointLight);
346
347     // Create light that is viewable from all directions.
348     var ambientLight = new THREE.AmbientLight(0x777777);
349     scene.add(ambientLight);
350
351     // Create the sun.
352     var sunMaterial = new THREE.MeshBasicMaterial({color: 'rgb(255, 255, 255)'});
353     sun = getSphere(sunMaterial, 1.03, 48);
354     scene.add(sun);
355
356     // Create the glow of the sun.
357     var spriteMaterial = new THREE.SpriteMaterial(
358             {
359                 map: new THREE.ImageUtils.loadTexture("img/glow.png")
360                 , useScreenCoordinates: false
361                 , color: 0xffffee
362                 , transparent: false
363                 , blending: THREE.AdditiveBlending
364             });
365     var sprite = new THREE.Sprite(spriteMaterial);
366     sprite.scale.set(7, 7, 1.0);
367     sun.add(sprite); // This centers the glow at the sun.
```

태양계의 기준이 되는 태양을 렌더링 하는 과정입니다. 다른 행성과 행동패턴이 다르기에 따로 코드를 구현하였고, 태양으로부터 전방위로 조명이 비치도록 설정하였습니다.

```
    // Load the images used in the background.
    var path = 'cubemap/';
    var format = '.jpg';
    var urls = [
        path + 'px' + format, path + 'nx' + format,
        path + 'py' + format, path + 'ny' + format,
        path + 'pz' + format, path + 'nz' + format
    ];
    var reflectionCube = new THREE.CubeTextureLoader().load(urls);
    reflectionCube.format = THREE.RGBFormat;

    // Attach the background cube to the scene.
    scene.background = reflectionCube;
```

우주의 배경이 될 큐브형의 배경을 구현하는 코드입니다.

```
    // Create planets.
    mercury = loadTexturedPlanet(mercuryData, mercuryData.distanceFromAxis, 0, 0);
    venus = loadTexturedPlanet(venusData, venusData.distanceFromAxis, 0, 0);
    earth = loadTexturedPlanet(earthData, earthData.distanceFromAxis, 0, 0);
    moon = loadTexturedPlanet(moonData, moonData.distanceFromAxis, 0, 0);
    mars = loadTexturedPlanet(marsData, marsData.distanceFromAxis, 0, 0);
    jupiter = loadTexturedPlanet(jupiterData, jupiterData.distanceFromAxis, 0, 0);
    saturn = loadTexturedPlanet(saturnData, saturnData.distanceFromAxis, 0, 0);
    uranus = loadTexturedPlanet(uranusData, uranusData.distanceFromAxis, 0, 0);
    neptune = loadTexturedPlanet(neptuneData, neptuneData.distanceFromAxis, 0, 0);

    // Create the visible orbit that the Earth uses.
    createVisibleOrbits();

    // Create the GUI that displays controls.
    var gui = new dat.GUI();
    var folder = gui.addFolder('speed');
    folder.add(orbitData, 'value', 0, 500);
    folder.add(orbitData, 'runOrbit', 0, 1);
    folder.add(orbitData, 'runRotation', 0, 1);

    // Start the animation.
    update(renderer, scene, camera, controls);
}
```

행성들의 텍스쳐를 로딩하고 공전, 자전 애니메이션을 시작하는 코드입니다.

1.3 movePlanet 메서드

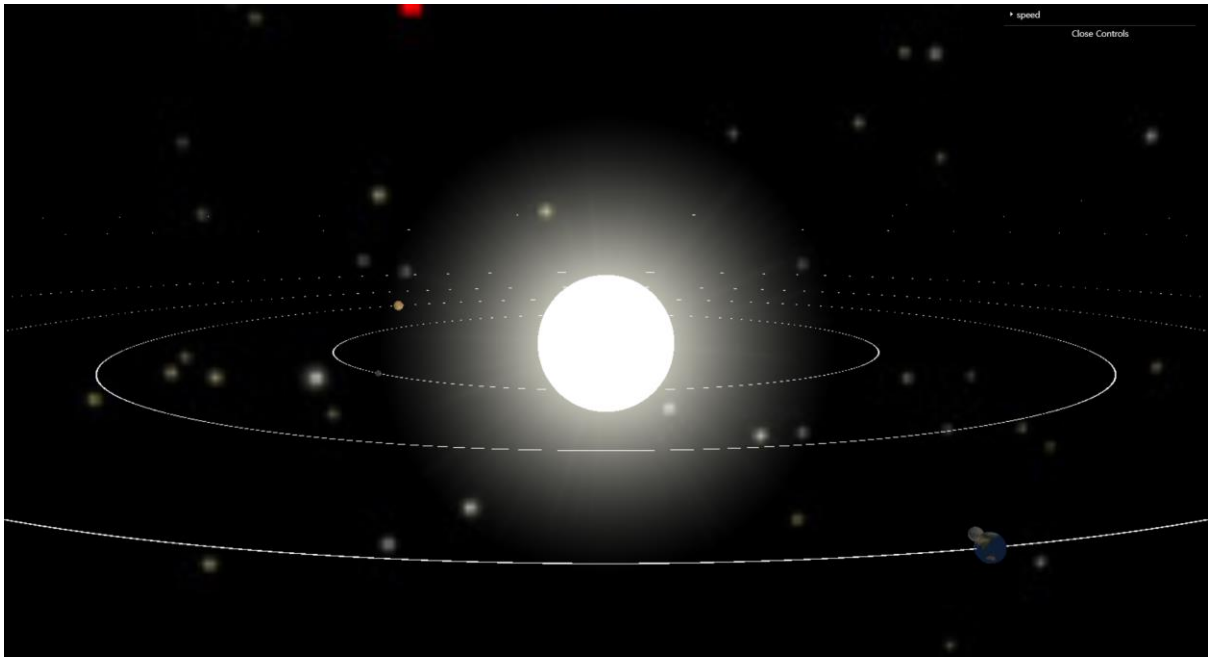
```
239 function movePlanet(myPlanet, myData, myTime, stopRotation) {
240     if (orbitData.runRotation && !stopRotation) {
241         myPlanet.rotation.y += myData.rotationRate;
242     }
243     if (orbitData.runOrbit) {
244         myPlanet.position.x = Math.cos(myTime
245                 * (1.0 / (myData.orbitRate * orbitData.value)) + 10.0)
246                 * myData.distanceFromAxis;
247         myPlanet.position.z = Math.sin(myTime
248                 * (1.0 / (myData.orbitRate * orbitData.value)) + 10.0)
249                 * myData.distanceFromAxis;
250     }
251 }
```

행성들의 자전과 공전을 구현한 코드입니다. 자전은 행성의 y 데이터를 갱신하는것으로 비교적 쉽게 구현하였고, 공전은 태양을 기준으로 각속도를 구하여 구현하였습니다.

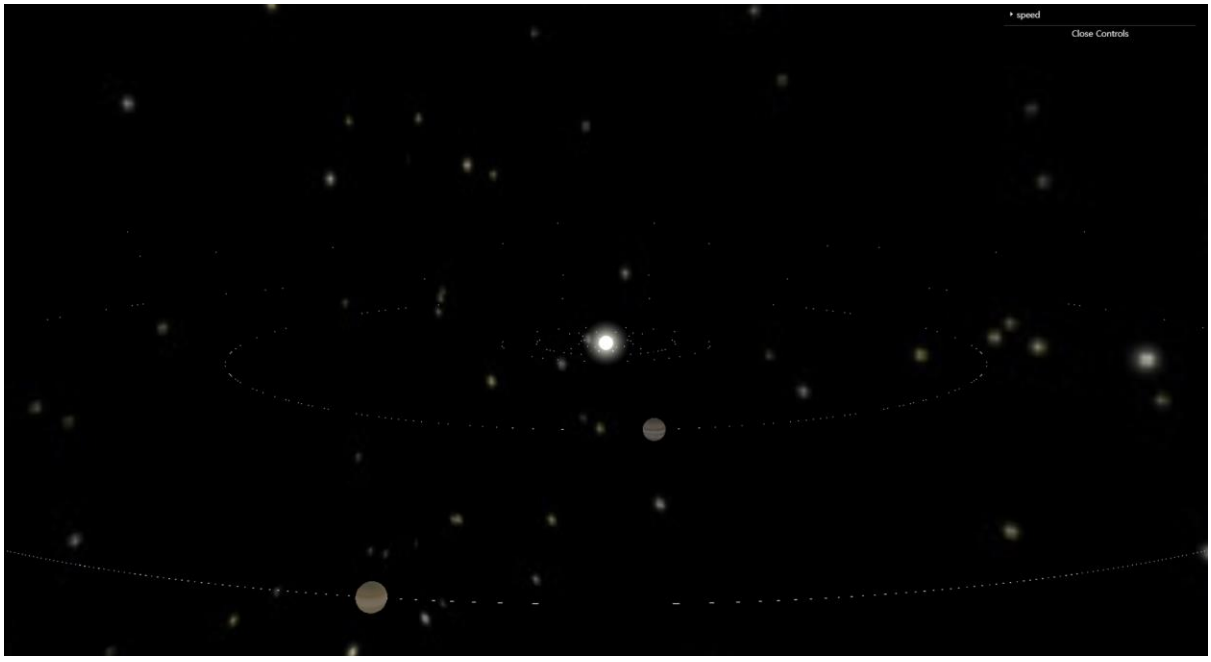
2. 결과

원하던 바와 같이 공전 및 자전 운동을 하게 구현되었습니다. 결과 화면은 실제 물리량과 동일하게 구현했기 때문에 한 화면에 모든 행성을 잡기는 힘들었습니다.

2.1 결과화면 - 수성, 금성, 지구 (내행성계)



2.2 결과 화면 - 목성, 토성 (외행성계)



3. 참고

- 참고 소스코드

https://www.youtube.com/watch?v=7VaAIVqu_P0&t=1s

- 직접 구현한 소스코드 Github

https://github.com/kevin0309/ComputerGraphics_SolarSystem