**컴퓨터 그래픽스 중간고사 대체 프로젝트**

**Space Ship Simulator**

**최종보고서**

**32142858**

**원규호**

**소프트웨어학과**

**목적:**

자유로운 주제를 선정하여 Web 상에서 실행 가능한 간단한 그래픽스 응용 프로그램을 작성

**조건:**

- WebGL을 이용할 것

- 모델링 데이터는 모델링 소프트웨어를 이용하여 직접 제작하거나 공개 데이터를 사용할 수 있음

- 프로그램의 제공 기능을 구체적으로 포함하는 제안서를 제출할 것(5/22)

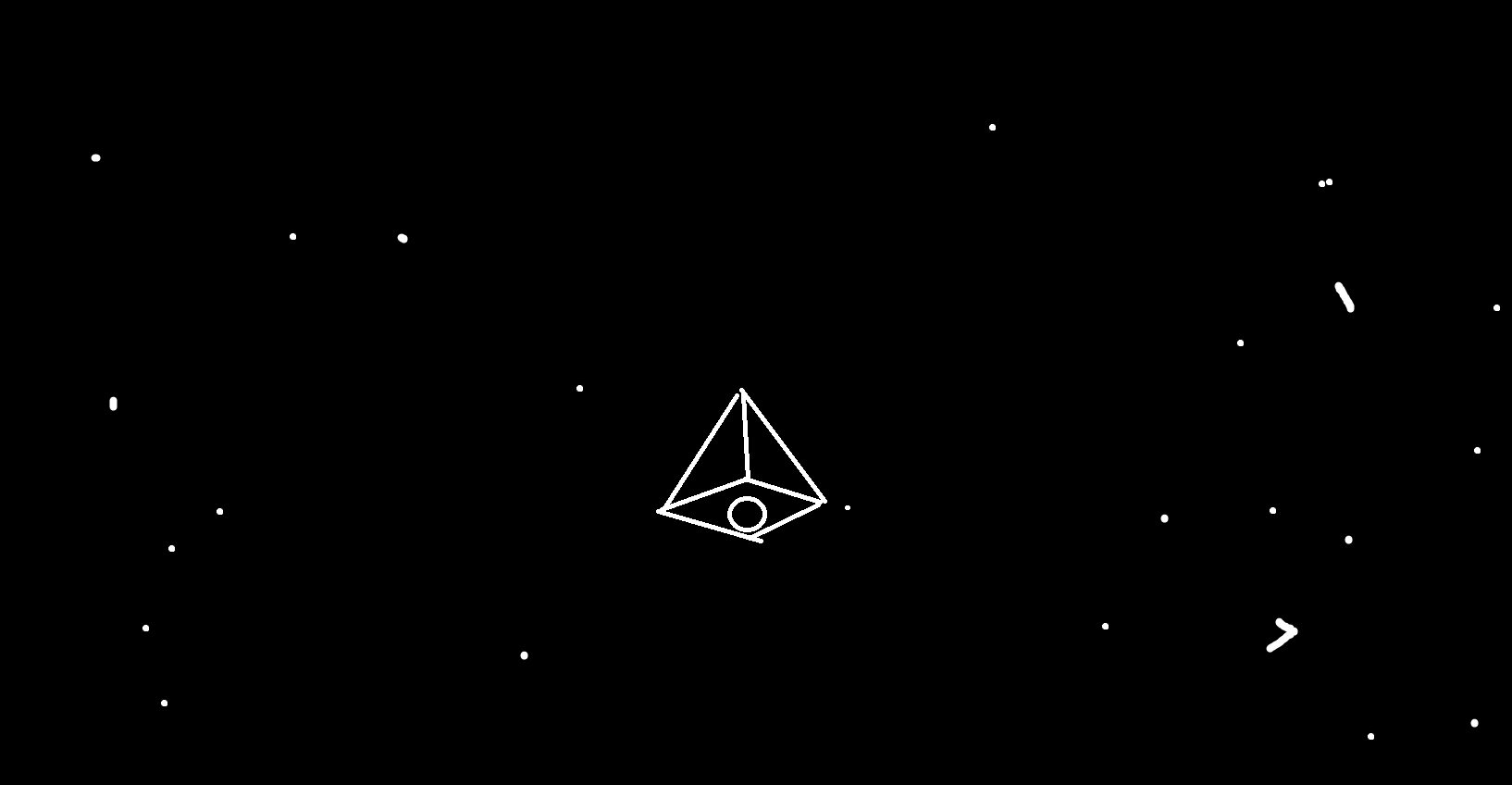
- 프로그램의 설계와 테스트 결과를 포함하는 최종 보고서를 제출할 것(6/19)

- 소스 코드 및 문서를 GitHUb에 등록할 것

- 최종 결과 중 일부는 화상 회의 수업 시간에 발표 예정임 (발표 불참 시 페널티 적용)

**기본 기획:**

우주 공간을 이동하는 우주선을 묘사해보자

****

우주의 배경을 만들기 위해 배경은 검은색 그리고 하얀색 점들로 별을 표현

화면의 가운데는 우주선 모델링을 사용해서 우주를 여행하는 모습을 표현

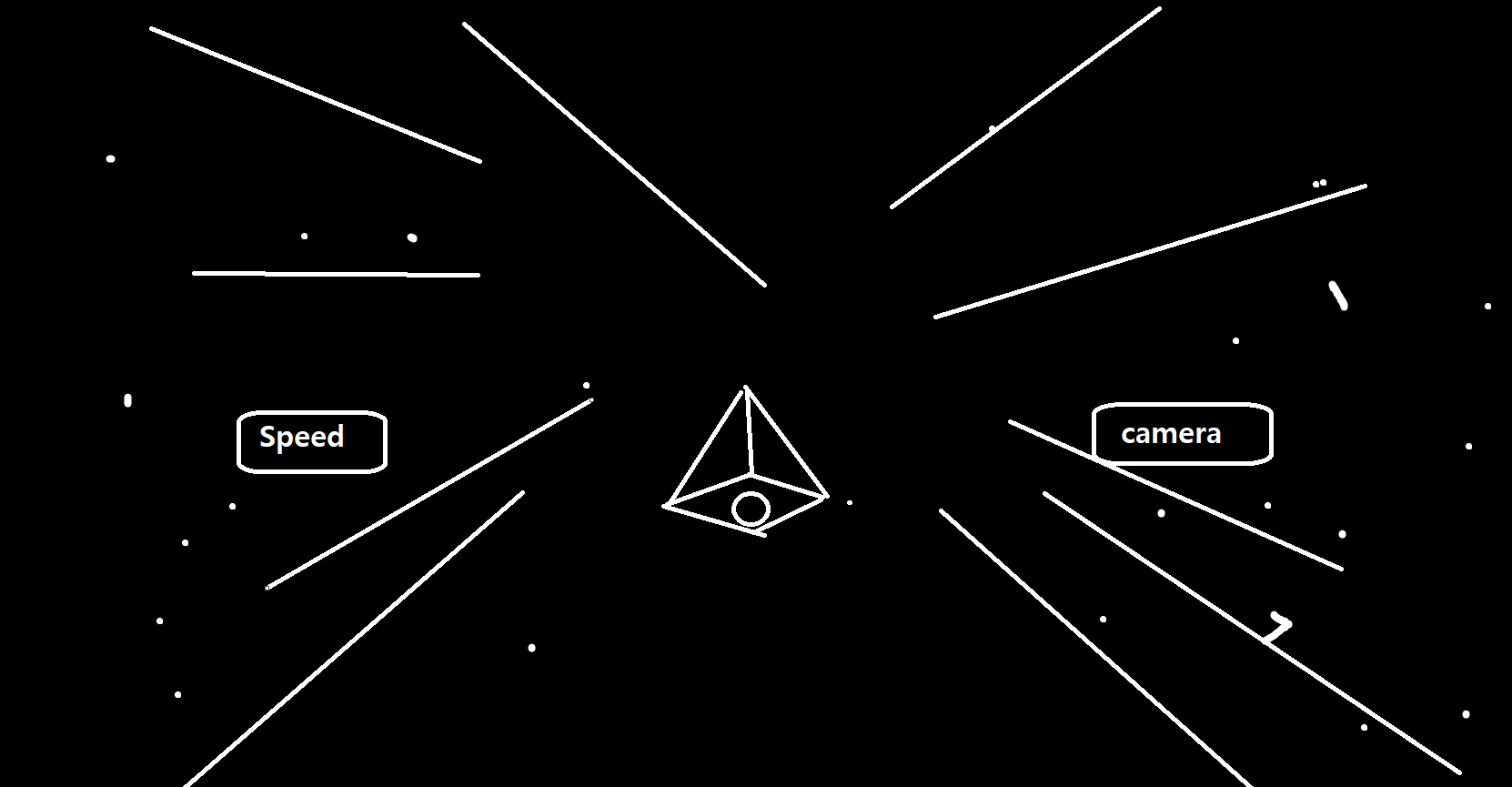
**추가 기획:**

>버튼을 추가해 우주선의 카메라 시점을 변경해보기

>별들을 움직이게 해서 속도감을 표현해보기

>우주선이 움직이는 애니메이션 추가해보기

**예상 모습:**

****

**사용 툴 및 모델링:**

**>Three.js를 사용 예정 :**

Three.js는 [웹 브라우저](https://ko.wikipedia.org/wiki/%EC%9B%B9_%EB%B8%8C%EB%9D%BC%EC%9A%B0%EC%A0%80)에서 애니메이션 [3차원 컴퓨터 그래픽스](https://ko.wikipedia.org/wiki/3%EC%B0%A8%EC%9B%90_%EC%BB%B4%ED%93%A8%ED%84%B0_%EA%B7%B8%EB%9E%98%ED%94%BD%EC%8A%A4)를 만들고 표시하기 위해 사용되는 [크로스 브라우저](https://ko.wikipedia.org/w/index.php?title=%ED%81%AC%EB%A1%9C%EC%8A%A4_%EB%B8%8C%EB%9D%BC%EC%9A%B0%EC%A0%80&action=edit&redlink=1) [자바스크립트 라이브러리](https://ko.wikipedia.org/wiki/%EC%9E%90%EB%B0%94%EC%8A%A4%ED%81%AC%EB%A6%BD%ED%8A%B8_%EB%9D%BC%EC%9D%B4%EB%B8%8C%EB%9F%AC%EB%A6%AC)이자 [API](https://ko.wikipedia.org/wiki/API)이다

현 상황에서 가장 쉽게 3D모델링을 넣는 것이 가능한 API.

**>모델링 소스:**

<https://sketchfab.com/feed>

Three.js에서 권장되는 모델링은 gltf형식이므로 gltf형식의 모델링을 무료로 받을 수 있는 사이트에서 우주선의 모델링을 따온다.

**사용 모델링:**



<https://sketchfab.com/3d-models/x-wing-starfighter-t-65c-a2-2397144aa2314e8fb03bf131d8f56af4>

위 사이트에 존재하는 무료 모델링을 사용하였다.

**코드:**

**HTML:**

<!DOCTYPE html>

<html>

  <head>

    <meta charset=UTF-8 />

    <link rel="stylesheet" type="text/css" href="styles.css" />

    <link href="https://fonts.googleapis.com/css?family=Raleway&display=swap" rel="stylesheet">

  </head>

  <body>

      <script src="three.min.js"></script>

      <script src="GLTFLoader.js"></script>

    <script>

    let scene, camera, renderer, stars, starGeo, speed=0 , front=0;

    function init() {

      scene = new THREE.Scene();

      camera = new THREE.PerspectiveCamera(60,window.innerWidth/window.innerHeight,1, 5000);

      camera.position.z = 10;

      camera.rotation.x = Math.PI/2;

      camera.position.y= 0;

      hlight = new THREE.AmbientLight (0x404040,10);

      scene.add(hlight);

      directionalLight = new THREE.DirectionalLight(0xffffff,10);

      directionalLight.position.set(0,1,0);

      directionalLight.castShadow = true;

      scene.add(directionalLight);

      renderer = new THREE.WebGLRenderer();

      renderer.setSize(window.innerWidth, window.innerHeight);

      document.body.appendChild(renderer.domElement);

      starGeo = new THREE.Geometry();

      for(let i=0;i<6000;i++) {

        star = new THREE.Vector3(

          Math.random() \* 600 - 300,

          Math.random() \* 600 - 300,

          Math.random() \* 600 - 300

        );

        star.velocity = 0;

        star.acceleration = 0.01;

        starGeo.vertices.push(star);

      }

      let sprite = new THREE.TextureLoader().load( 'star.png' );

      let starMaterial = new THREE.PointsMaterial({

        color: 0xaaaaaa,

        size: 0.7,

        map: sprite

      });

      stars = new THREE.Points(starGeo,starMaterial);

      scene.add(stars);

      window.addEventListener("resize", onWindowResize, false);

      renderer = new THREE.WebGLRenderer({antialias:true});

      renderer.setSize(window.innerWidth,window.innerHeight);

      document.body.appendChild(renderer.domElement);

      let loader = new THREE.GLTFLoader();

      loader.load('scene.gltf', function(gltf){

      spaceShip = gltf.scene.children[0];

      spaceShip.scale.set(30,30,30);

      spaceShip.position.x=-21;

      spaceShip.position.y=110;

      spaceShip.position.z=-50;

      spaceShip.rotation.x=-0.12;

      spaceShip.rotation.y=-0.12;

      spaceShip.rotation.z=0.22;

      scene.add(spaceShip);

      });

      animate();

    }

    function button1\_click() { // 속도 올리는 버튼/ 최대 3까지

      if(speed<3){

      animate();

      speed++;

      }

      if(speed==3){

        alert("최대 속도 입니다.");

      }

    }

    function button2\_click() { //속도 내리는 버튼/ 최소 -1까지

      if(speed==-1){

        alert("최소 속도 입니다.");

      }

      if(speed>-1){

      animate2();

      speed--;

      }

    }

    function button3\_click() { //앞면으로 시점 변환해주는 버튼, 다시 한번 클릭 시 돌아간다.

      if(front==0){

        camera.position.y= 100;

        front++;

      }else if(front == 1){

        camera.position.y= 0;

        front--;

      }

    }

    function button4\_click() {

      camera.rotation.z += 0.3;

    }

    function onWindowResize() {

        camera.aspect = window.innerWidth / window.innerHeight;

        camera.updateProjectionMatrix();

        renderer.setSize(window.innerWidth, window.innerHeight);

      }

    function animate() {

      starGeo.vertices.forEach(p => {

        p.velocity += p.acceleration

        p.y -= p.velocity;

        if (p.y < -200) {

          p.y = 200;

          p.velocity = 0;

        }

      });

      starGeo.verticesNeedUpdate = true;

      stars.rotation.y +=0.002;

      renderer.render(scene, camera);

      requestAnimationFrame(animate);

    }

     function animate2() {

      starGeo.vertices.forEach(p => {

        p.velocity -= p.acceleration

        p.y += p.velocity;

        if (p.y < -200) {

          p.y = 200;

          p.velocity = 0;

        }

      });

      starGeo.verticesNeedUpdate = true;

      stars.rotation.y -=0.002;

      renderer.render(scene, camera);

      requestAnimationFrame(animate2);

    }

    init();

    </script>

    <div id="container">

      <header>

       <h1>Space Ship Simulator </h1>

      </header>

      <input type="button" id="button1" onclick="button1\_click();" value="Speed Up" class="button" />

      <input type="button" id="button2" onclick="button2\_click();" value="Speed Down" class="button2"/>

      <input type="button" id="button3" onclick="button3\_click();" value="Front View" class="button3" />

      <input type="button" id="button4" onclick="button4\_click();" value="Turn Space ship" class="button4"/>

    </div>

  </body>

</html>

**Three.js를 사용하였다.**

**사용된 js는**

>기본 베이스인 three.min.js

>GLTF 모델을 불러오기 위한 GLTFLoader를 사용하였다.

**또한 Geometry를 통해 별들의 움직임을 재현하였다.**

**실행화면:**

****

**초기 기획 목적이었던 별들의 바다 사이로 비행을 한다는 장면에 구현에 성공하였다.**

**추가 기획이었던,**

>버튼을 추가해 우주선의 카메라 시점을 변경해보기

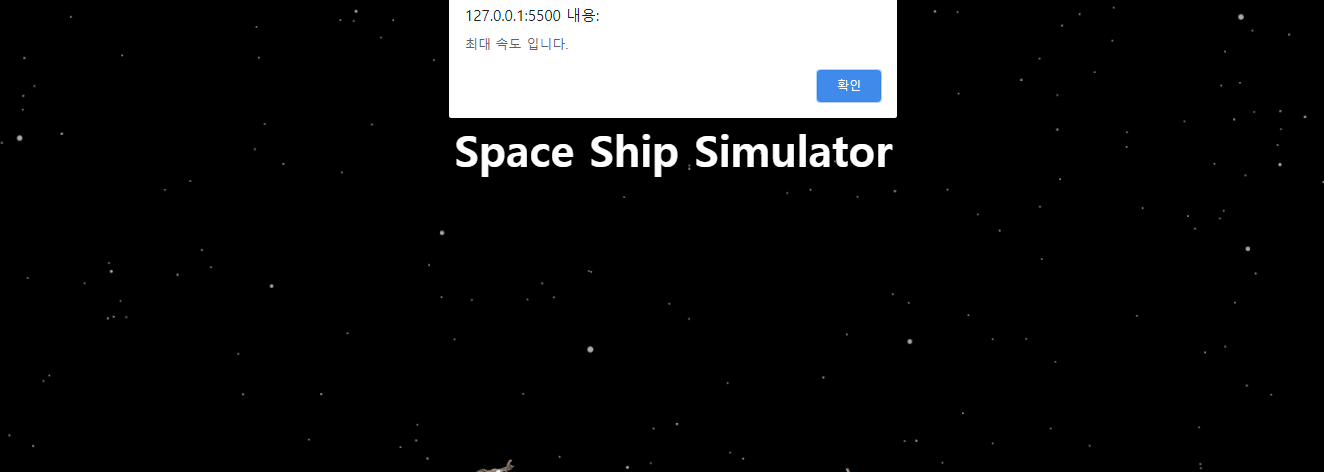
>별들을 움직이게 해서 속도감을 표현해보기

>우주선이 움직이는 애니메이션 추가해보기

**들은 각각 4개의 버튼으로 구현을 해보았다.**

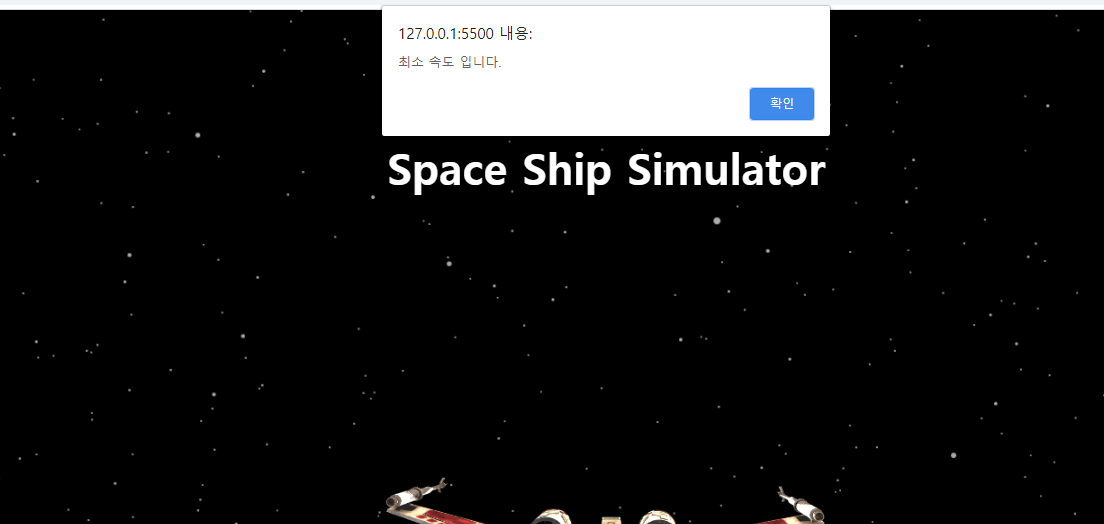
**Speed Up버튼**

**Speed Up 버튼을 누를 시 별들의 별 들의 geometry 수치에 속도를 추가하는 것으로 마치 우주선에 속도가 빨라진 것처럼 연출하게 된다. 하지만 너무 빨라 질 경우 별들의 움직임이 너무 규칙적으로 바뀌어서 부자연스럽게 바뀌는 것을 확인, 속도의 최대 제한을 걸었다.**

**최대 속도에 도달한 후 다시 Speed Up 버튼을 누르면 경고창이 뜬다.**

**Speed Down버튼**

**Speed up 버튼과 마찬가지로 버튼을 누를 시 별 들의 geometry 수치의 속도를 낮춘다. 단 기본 속도 인 상태에서 한번 이상 버튼을 누를 시 별들이 반대로 이동하는 것을 확인, 최소 속도를 기본에서 한 단계 아래까지로 제한하였다.**

****

**최소 속도에 도달한 후 다시 Speed Down 버튼을 누르면 경고창이 뜬다.**

**Front View 버튼**

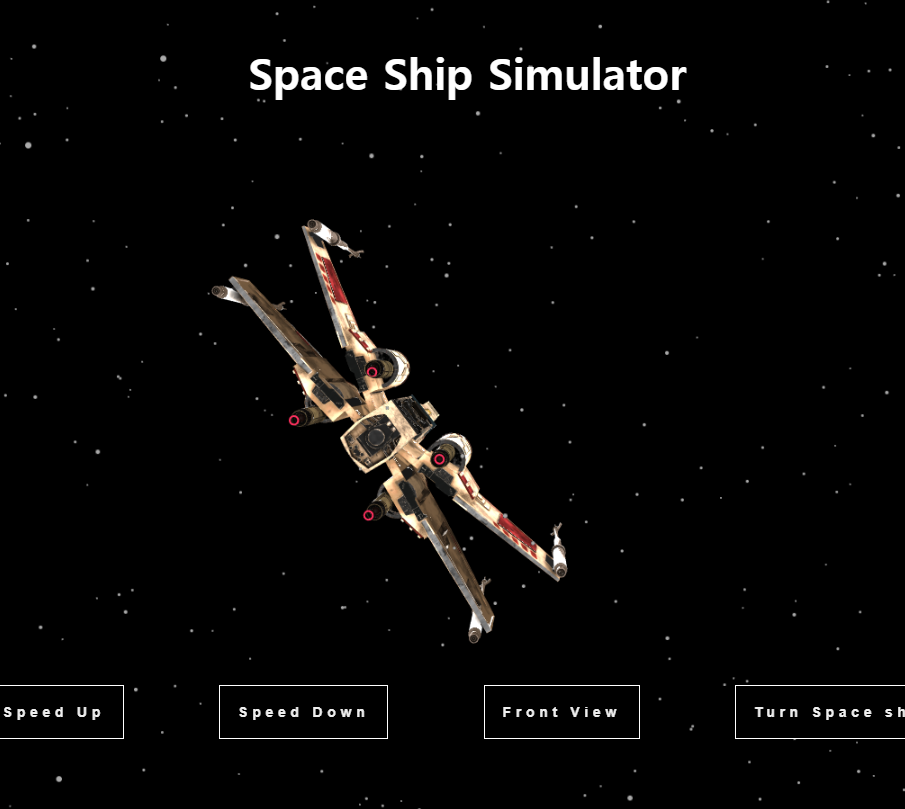
Front View 버튼을 누를 시 카메라 시점 조절을 통해 우주선 앞으로 카메라를 이동 시킨다. 다시 버튼을 누를 경우 다시 카메라를 원위치로 돌아가게 하였다.



**Front View 상태, 다시 버튼을 누르면 원 상태로 되돌아간다.**

**Turn Space ship 버튼**

**Turn Space ship 버튼의 겨우 누를 시 우주선이 왼쪽 방향으로 회전한다. 카메라를 회전 시키는 것으로 우주선의 움직임을 연출하였다. 카메라가 변경된 뒤에도 다른 버튼들과 연동이 가능하다.**

****

**옆으로 비행 중인 우주선**

****

**Front View일 때도 그대로 유지된다.**

**아쉬웠던 점**

**아직 Three.js의 모든 기능을 파악하지는 못하였기에 orbit control.js를 사용해 마우스를 통해 우주 안에서의 우주선을 다방면으로 보게 만들어 보고 싶었지만, 실패하였다. 또한 우주선 자체에게 애니메이션을 주고 싶었지만 너무 어려웠기에 버튼을 통해 카메라 워크를 조절하는 것으로 대신 연출해봤다.**

**Github 주소**

<https://github.com/wonkh11/Space-Project->