FINAL PROJECT HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>FINAL PROJECT</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <meta name="Generator" content="https://callum.com">
        <link type="text/css" rel="stylesheet" href="css/style.css" />
        <link href='http://fonts.googleapis.com/css?family=PT+Sans'
rel='stylesheet' type='text/css'>
        <script type="text/javascript" src="js/three.min.js"></script>
        <script type="text/javascript" src="js/Detector.js"></script>
        <script type="text/javascript" src="js/stats.min.js"></script>
        <script type="text/javascript" src="js/TrackballControls.js"></script>
        <script type="text/javascript" src="js/dat.gui.min.js"></script>
        <script type="text/javascript" src="js/flights_one.js"></script>
        <script type="text/javascript" src="js/app.js"></script>
        <script type="x-shader/x-vertex" id="vertexshader">
            attribute float size;
            attribute vec3 customColor;
            varying vec3 vColor;
            void main() {
                vColor = customColor;
                vec4 mvPosition = modelViewMatrix * vec4( position, 1.0 );
                gl_PointSize = size * ( 300.0 / length( mvPosition.xyz ) );
                gl_Position = projectionMatrix * mvPosition;
            }
        </script>
        <script type="x-shader/x-fragment" id="fragmentshader">
            uniform vec3 color;
            uniform sampler2D texture;
            varying vec3 vColor;
            void main() {
                gl_FragColor = vec4( color * vColor, 0.5 );
                gl_FragColor = gl_FragColor * texture2D( texture,
gl_PointCoord );
            }
        </script>
        </head>
        <body>
                <script type="text/javascript">
                        document.addEventListener("DOMContentLoaded", start_app, false);
                </script>
                <div id="loading_overlay" class="hide"></div>

        </body>
</html>
```

FINAL PROJECT JavaScript

```
// Student ID:S0143046 Name:吳詮義 Date:2016/3/23 Major:地理系
var camera, scene, renderer, controls, stats;
var flight_path_splines = [];
var flight_point_cloud_geom;
var positions, sizes;
var flight_path_lines;
var flight_point_start_time = [];
var flight_point_end_time = [];
var flight_distance = [];
var start_flight_idx = 0;
var end_flight_idx = flights.length;
var flight_point_speed_changed = false;
var flight_point_speed_scaling = 5.0;
var flight_point_speed_min_scaling = 1.0;
var flight_point_speed_max_scaling = 25.0;
var flight_track_opacity = 0.02;
var flight_point_size = 0.015;
var earth_img = 0;
var elevation_img = 0;
var water_img = 0;
var is_loading = false;
var sphere;
var sphere1;

function start_app() {
	init();
	animate();
}

function init() {
	if (!Detector.webgl) {
		Detector.addGetWebGLMessage();
	}

	show_loading(true);

	renderer = new THREE.WebGLRenderer();
```

```
        renderer.setClearColor(0x000000, 1.0);
        renderer.setPixelRatio(window.devicePixelRatio);
        renderer.setSize(window.innerWidth, window.innerHeight);
        document.body.appendChild(renderer.domElement);

        scene = new THREE.Scene();

        camera = new THREE.PerspectiveCamera(45, window.innerWidth /
    window.innerHeight, 0.01, 100);
        camera.position.z = 1.5;

        scene.add(new THREE.AmbientLight(0x777777));

        var light1 = new THREE.DirectionalLight(0xffffff, 0.2);
        light1.position.set(5, 3, 5);
        scene.add(light1);

        var light2 = new THREE.DirectionalLight(0xffffff, 0.2);
        light2.position.set(5, 3, -5);
        scene.add(light2);

        var radius2 = 5, segemnt2 = 64;

        var galaxy_img = new THREE.MeshPhongMaterial(
        {
            map: THREE.ImageUtils.loadTexture('images/galaxy.jpg' ),
            side    : THREE.BackSide
         } );

        //var sphereMaterial1 = new THREE.MeshLambertMaterial({ color: 0xCC0000 });


        sphere1 = new THREE.Mesh(
        new THREE.SphereGeometry(radius2,segemnt2,segemnt2),
        galaxy_img
        );
        sphere1.position.set(0, 0, 0);
        sphere1.geometry.verticesNeedUpdate = true;
```

```
  sphere1.geometry.normalsNeedUpdate = true;

  scene.add(sphere1);


var radius1 = 0.1, segemnt1 = 64;

var moon_img = new THREE.MeshPhongMaterial(
{
      map: THREE.ImageUtils.loadTexture('images/moonmap1k.jpg' ),
      bumpMap: THREE.ImageUtils.loadTexture('images/moonbump1k.jpg' ),
      bumpScale: 0.005

  } );


sphere = new THREE.Mesh(
new THREE.SphereGeometry(radius1,segemnt1,segemnt1),
moon_img
);

sphere.position.set(-0.8, 0, 0);
sphere.geometry.verticesNeedUpdate = true;
  sphere.geometry.normalsNeedUpdate = true;


  scene.add(sphere);

var pointLight = new THREE.PointLight(0xFFCC66, 0.5);

pointLight.position.x = 10;
pointLight.position.y = 50;
pointLight.position.z = 150;

scene.add(pointLight);


var radius = 0.5;
```

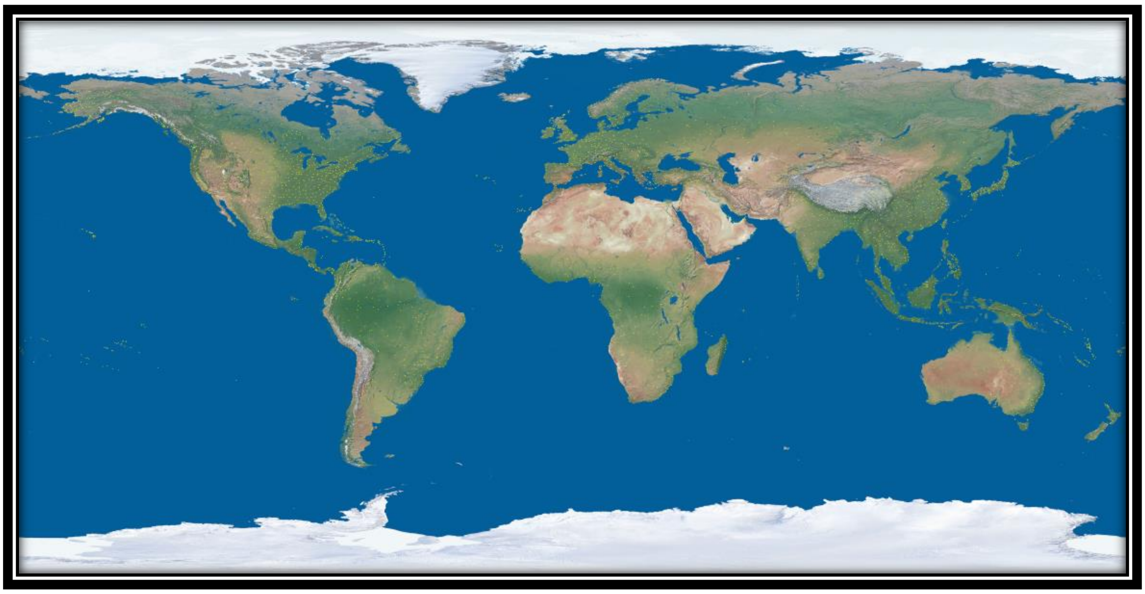
```
    var segments = 64;

    earth_img = THREE.ImageUtils.loadTexture('images/earth_airports.png',
THREE.UVMapping, function() {
        elevation_img = THREE.ImageUtils.loadTexture('images/elevation.jpg',
THREE.UVMapping, function() {
            water_img = THREE.ImageUtils.loadTexture('images/water.png',
THREE.UVMapping, function() {
                scene.add(new THREE.Mesh(
                    new THREE.SphereGeometry(radius, segments, segments),
                    new THREE.MeshPhongMaterial({
                            map: earth_img,
                            bumpMap: elevation_img,
                            bumpScale: 0.01,
                            specularMap: water_img,
                            specular: new THREE.Color('grey')
                        })
                    )
                );


                generateControlPoints(radius);

                flight_path_lines = flightPathLines();
                scene.add(flight_path_lines);

                scene.add(flightPointCloud());

                show_loading(false);
            })
        })
    })

    var gui = new dat.GUI();
    gui.add(this, 'flight_point_speed_scaling', flight_point_speed_min_scaling,
flight_point_speed_max_scaling).name("速度").onFinishChange(function(value) {
        flight_point_speed_changed = true;
        update_flights();
        flight_point_speed_changed = false;
    });
    gui.add(this, 'flight_point_size', 0.01, 0.2).name("大小
").onChange(function(value) {
        flight_point_cloud_geom.attributes.size.needsUpdate = true;
        for (var i = start_flight_idx; i < end_flight_idx; ++i) {
            sizes[i] = flight_point_size;
        }
    });
    gui.add(this, 'flight_track_opacity', 0, 1.0).name("航線透明度
").onChange(function(value) {
        flight_path_lines.material.opacity = value;
    });

    controls = new THREE.TrackballControls(camera, renderer.domElement);
    controls.rotateSpeed = 0.4;
    controls.noZoom = false;
    controls.noPan = true;
    controls.staticMoving = false;
    controls.minDistance = 0.75;
    controls.maxDistance = 3.0;

    stats = new Stats();
    stats.domElement.style.position = 'absolute';
    stats.domElement.style.top = '0px';
    document.body.appendChild(stats.domElement);

    window.addEventListener('resize', onWindowResize, false);
}

function generateControlPoints(radius) {
    for (var f = start_flight_idx; f < end_flight_idx; ++f) {

        var start_lat = flights[f][0];
        var start_lng = flights[f][1];
        var end_lat = flights[f][2];
        var end_lng = flights[f][3];
        var max_height = Math.random() * 0.04;

        var points = [];
        var spline_control_points = 8;
        for (var i = 0; i < spline_control_points + 1; i++) {
            var arc_angle = i * 180.0 / spline_control_points;
            var arc_radius = radius + (Math.sin(arc_angle * Math.PI / 180.0)) * max_height;
            var latlng = latlngInterPoint(start_lat, start_lng, end_lat, end_lng, i / spline_control_points);

            var pos = xyzFromLatLng(latlng.lat, latlng.lng, arc_radius);

            points.push(new THREE.Vector3(pos.x, pos.y, pos.z));
        }

        var spline = new THREE.SplineCurve3(points);

        flight_path_splines.push(spline);

        var arc_length = spline.getLength();
        flight_distance.push(arc_length);

        setFlightTimes(f);
    }
}

function xyzFromLatLng(lat, lng, radius) {
    var phi = (90 - lat) * Math.PI / 180;
    var theta = (360 - lng) * Math.PI / 180;

    return {
        x: radius * Math.sin(phi) * Math.cos(theta),
        y: radius * Math.cos(phi),
        z: radius * Math.sin(phi) * Math.sin(theta)
    };
}
```

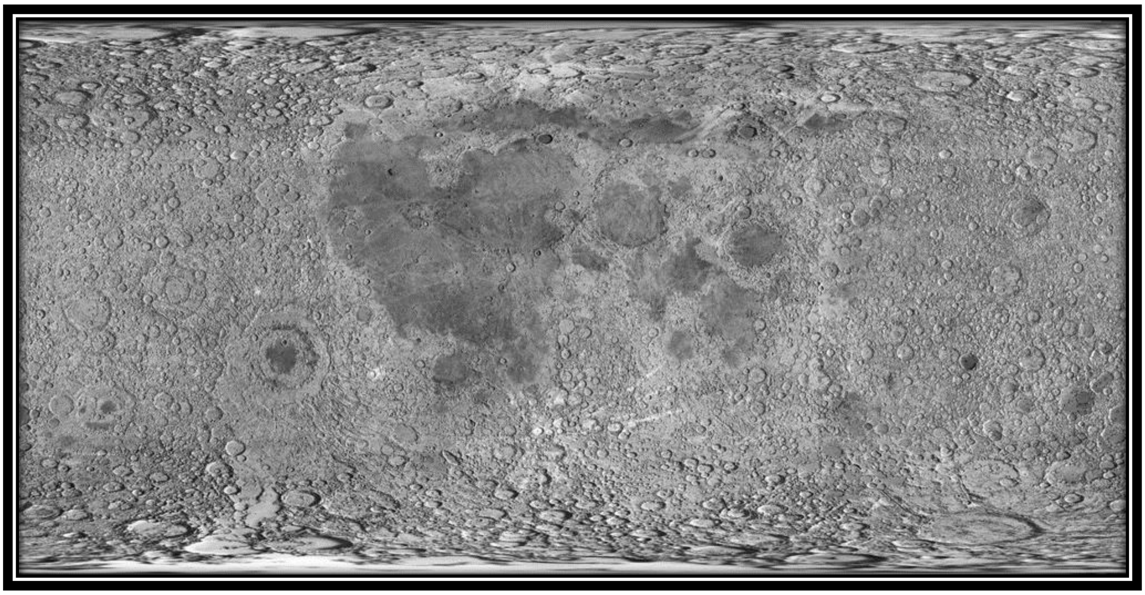
```
function latlngInterPoint(lat1, lng1, lat2, lng2, offset) {
        lat1 = lat1 * Math.PI / 180.0;
        lng1 = lng1 * Math.PI / 180.0;
        lat2 = lat2 * Math.PI / 180.0;
        lng2 = lng2 * Math.PI / 180.0;

        d = 2 * Math.asin(Math.sqrt(Math.pow((Math.sin((lat1 - lat2) / 2)), 2) +
                Math.cos(lat1) * Math.cos(lat2) * Math.pow(Math.sin((lng1 - lng2) / 2),
2)));
        A = Math.sin((1 - offset) * d) / Math.sin(d);
        B = Math.sin(offset * d) / Math.sin(d);
        x = A * Math.cos(lat1) * Math.cos(lng1) + B * Math.cos(lat2) * Math.cos(lng2);
        y = A * Math.cos(lat1) * Math.sin(lng1) + B * Math.cos(lat2) * Math.sin(lng2);
        z = A * Math.sin(lat1) + B * Math.sin(lat2);
        lat = Math.atan2(z, Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2))) * 180 /
Math.PI;
        lng = Math.atan2(y, x) * 180 / Math.PI;

        return {
                lat: lat,
                lng: lng
        };
}

function flightPointCloud() {
        flight_point_cloud_geom = new THREE.BufferGeometry();

        num_points = flights.length;

        positions = new Float32Array(num_points * 3);
        var colors = new Float32Array(num_points * 3);
        sizes = new Float32Array(num_points);

        for (var i = 0; i < num_points; i++) {
                positions[3 * i + 0] = 0;
                positions[3 * i + 1] = 0;
                positions[3 * i + 2] = 0;
        colors[3 * i + 0] = Math.random();
        colors[3 * i + 1] = Math.random();
        colors[3 * i + 2] = Math.random();

        sizes[i] = 0.03;
    }

    flight_point_cloud_geom.addAttribute('position', new
THREE.BufferAttribute(positions, 3));
    flight_point_cloud_geom.addAttribute('customColor', new
THREE.BufferAttribute(colors, 3));
    flight_point_cloud_geom.addAttribute('size', new THREE.BufferAttribute(sizes,
1));
    flight_point_cloud_geom.computeBoundingBox();

    var attributes = {
        size: {
            type: 'f',
            value: null
        },
        customColor: {
            type: 'c',
            value: null
        }
    };

    var uniforms = {
        color: {
            type: "c",
            value: new THREE.Color(0xffffff)
        },
        texture: {
            type: "t",
            value: THREE.ImageUtils.loadTexture("images/point.png")
        }
    };

    var shaderMaterial = new THREE.ShaderMaterial({
            uniforms: uniforms,
            attributes: attributes,
            vertexShader: document.getElementById('vertexshader').textContent,
            fragmentShader:
document.getElementById('fragmentshader').textContent,
            blending: THREE.AdditiveBlending,
            depthTest: true,
            depthWrite: false,
            transparent: true
      });

      return new THREE.PointCloud(flight_point_cloud_geom, shaderMaterial);
}

function flightPathLines() {

      var num_control_points = 32;

      var geometry = new THREE.BufferGeometry();
      var material = new THREE.LineBasicMaterial({
            color: 0x0099FF,
            vertexColors: THREE.VertexColors,
            transparent: true,
            opacity: flight_track_opacity,
            depthTest: true,
            depthWrite: false,
            linewidth: 0.001
      });
      var line_positions = new Float32Array(flights.length * 3 * 2 *
num_control_points);
      var colors = new Float32Array(flights.length * 3 * 2 * num_control_points);

      for (var i = start_flight_idx; i < end_flight_idx; ++i) {

            for (var j = 0; j < num_control_points - 1; ++j) {

                  var start_pos = flight_path_splines[i].getPoint(j / (num_control_points
- 1));
```

```
            var end_pos = flight_path_splines[i].getPoint((j + 1) /
(num_control_points - 1));

            line_positions[(i * num_control_points + j) * 6 + 0] = start_pos.x;
            line_positions[(i * num_control_points + j) * 6 + 1] = start_pos.y;
            line_positions[(i * num_control_points + j) * 6 + 2] = start_pos.z;
            line_positions[(i * num_control_points + j) * 6 + 3] = end_pos.x;
            line_positions[(i * num_control_points + j) * 6 + 4] = end_pos.y;
            line_positions[(i * num_control_points + j) * 6 + 5] = end_pos.z;

            colors[(i * num_control_points + j) * 6 + 0] = 1.0;
            colors[(i * num_control_points + j) * 6 + 1] = 0.4;
            colors[(i * num_control_points + j) * 6 + 2] = 1.0;
            colors[(i * num_control_points + j) * 6 + 3] = 1.0;
            colors[(i * num_control_points + j) * 6 + 4] = 0.4;
            colors[(i * num_control_points + j) * 6 + 5] = 1.0;
        }
    }

    geometry.addAttribute('position', new THREE.BufferAttribute(line_positions,
3));
    geometry.addAttribute('color', new THREE.BufferAttribute(colors, 3));

    geometry.computeBoundingSphere();

    return new THREE.Line(geometry, material, THREE.LinePieces);
}

function onWindowResize() {
    camera.aspect = window.innerWidth / window.innerHeight;
    camera.updateProjectionMatrix();
    renderer.setSize(window.innerWidth, window.innerHeight);
}

function easeOutQuadratic(t, b, c, d) {
    if ((t /= d / 2) < 1)
        return c / 2 * t * t + b;
    return -c / 2 * ((--t) * (t - 2) - 1) + b;
```

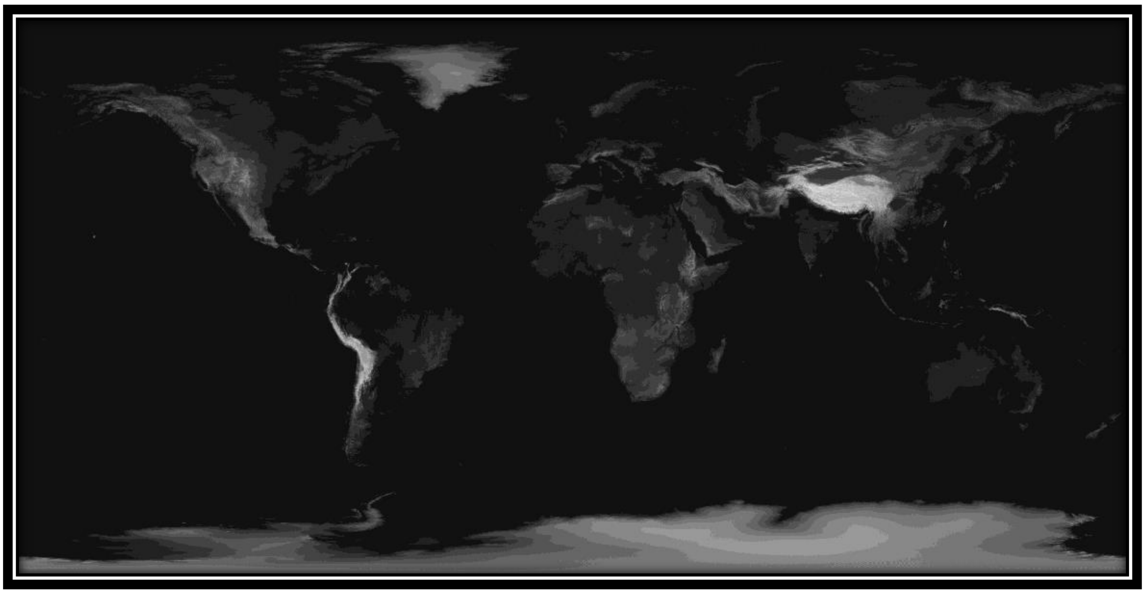
```
}

function setFlightTimes(index) {
    var scaling_factor = (flight_point_speed_scaling -
flight_point_speed_min_scaling) /
                            (flight_point_speed_max_scaling -
flight_point_speed_min_scaling);
    var duration = (1-scaling_factor) * flight_distance[index] * 80000;

    var start_time = Date.now() + Math.random() * 5000
    flight_point_start_time[index] = start_time;
    flight_point_end_time[index] = start_time + duration;
}

function update_flights() {
    flight_point_cloud_geom.attributes.position.needsUpdate = true;

    for (var i = start_flight_idx; i < end_flight_idx; ++i) {

        if ( Date.now() > flight_point_start_time[i] ) {
            var ease_val = easeOutQuadratic(Date.now() -
flight_point_start_time[i], 0, 1, flight_point_end_time[i] - flight_point_start_time[i]);

            if (ease_val < 0 || flight_point_speed_changed) {
                ease_val = 0;
                setFlightTimes(i);
            }

            var pos = flight_path_splines[i].getPoint(ease_val);
            positions[3 * i + 0] = pos.x;
            positions[3 * i + 1] = pos.y;
            positions[3 * i + 2] = pos.z;
        }
    }
}

function show_loading(visible) {
    if (visible) {
        is_loading = true;
        document.getElementById("loading_overlay").className = "show";
        document.getElementById("loading_overlay").style.pointerEvents = "all";
    } else {
        is_loading = false;
        document.getElementById("loading_overlay").className = "hide";
        document.getElementById("loading_overlay").style.pointerEvents =
"none";
    }
}


function animate(time) {
    requestAnimationFrame(animate);

    var timer = Date.now() * 0.0001;

                for ( var i = 0, l = scene.children.length; i < l; i ++ ) {

                    var radius = scene.children[ i ];

                    radius.rotation.x = timer * 2;
                    radius.rotation.y = timer * 1.5;

                }

    if ( ! is_loading ) {
        controls.update();
        update_flights();
    }

    stats.update();
    renderer.render(scene, camera);
}
```

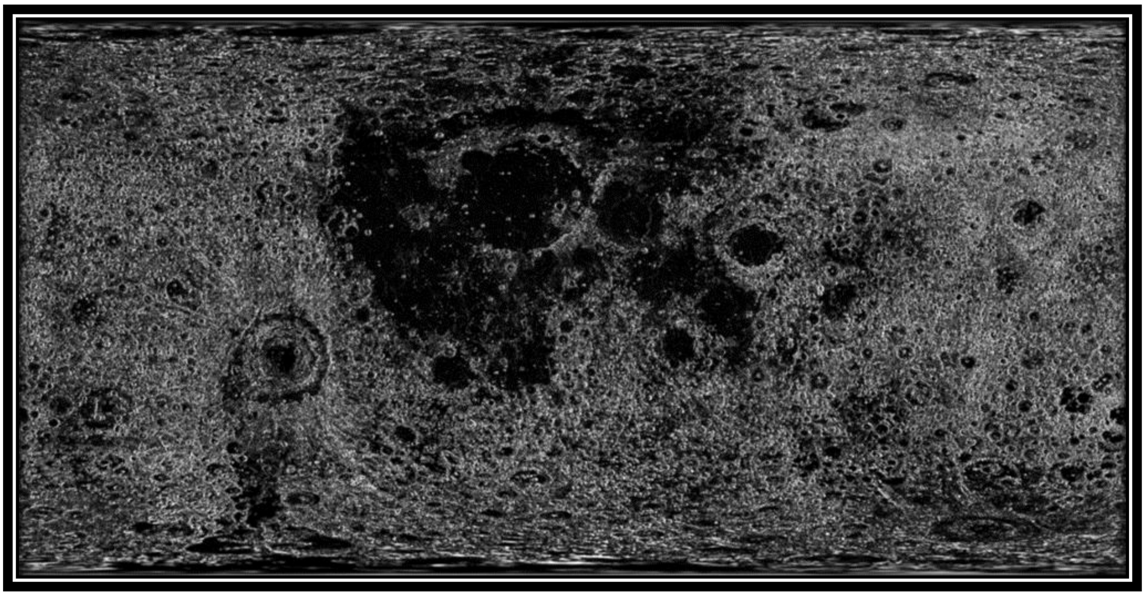
使用貼圖





凹凸貼圖

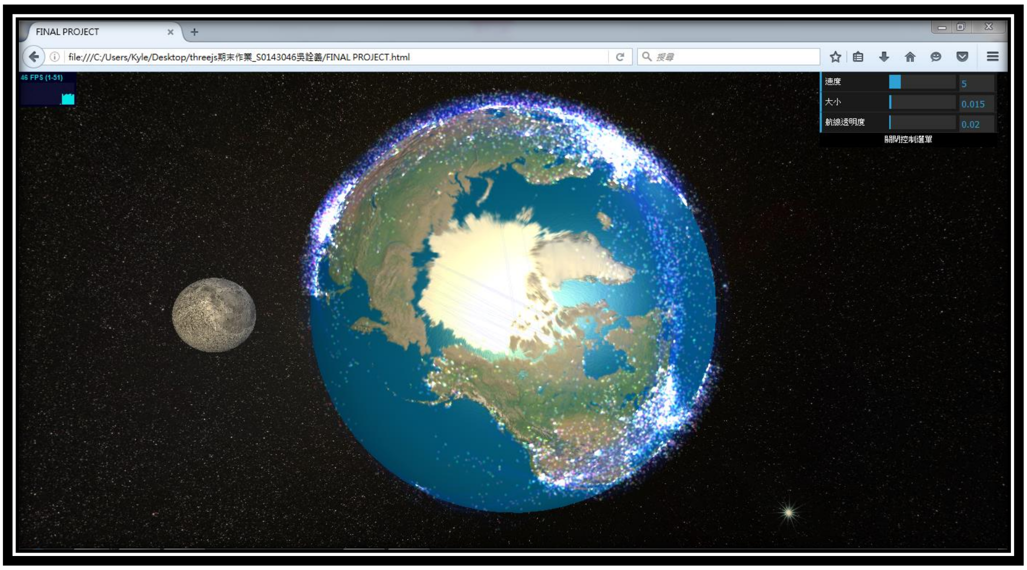




鏡面貼圖

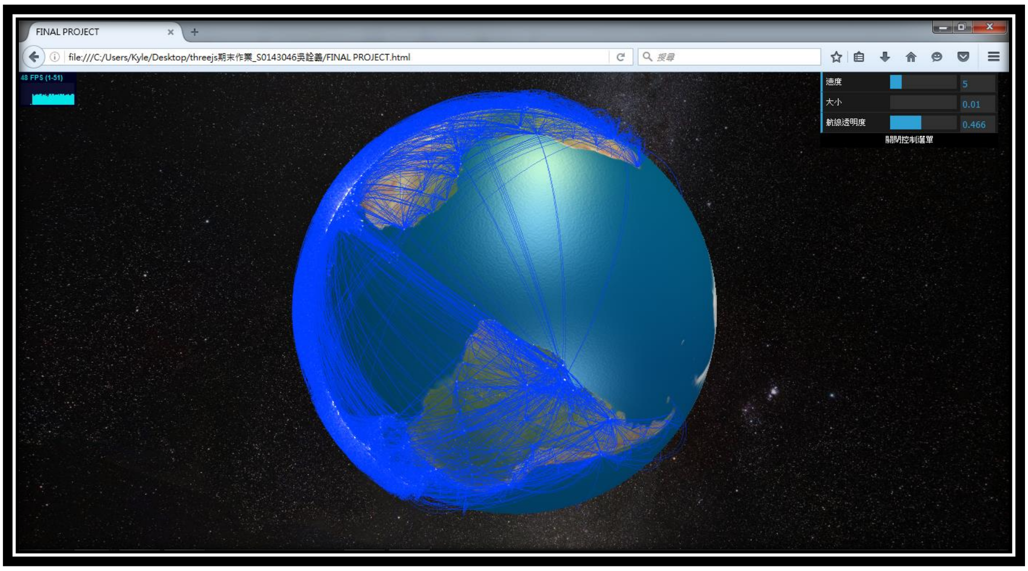
背景貼圖

展示成果



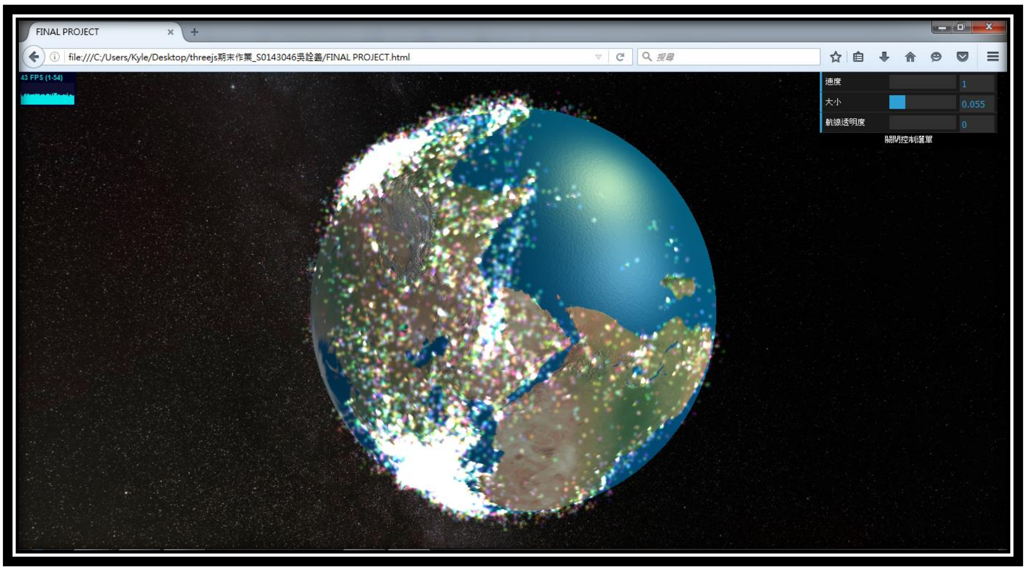
file:///C:/Users/Kyle/Desktop/threejs期末作業_S0143046吳詮義/FINAL PROJECT.html

速度 5
大小 0.015
航線透明度 0.02



file:///C:/Users/Kyle/Desktop/threejs期末作業_S0143046吳詮義/FINAL PROJECT.html

速度 5
大小 0.01
航線透明度 0.466



file:///C:/Users/Kyle/Desktop/threejs期末作業_S0143046吳詮義/FINAL PROJECT.html

速度 1
大小 0.055
航線透明度 0