電腦圖學

墨西哥帽子

作業一

邱奕達

2023/12/8

1. 班級學號姓名：

班級： 資工三乙　　學號： 410262119　　姓名： 邱奕達

2. 題目：墨西哥帽子

3. 討論：

本次作業利用 Webgl 的方式撰寫，執行程式的檔案是 html 檔。而這次作業所設計的墨西哥帽子總共有四頂，其中扮演太陽的角色為橘色墨西哥帽子、扮演地球的角色為綠色墨西哥帽子，扮演月球的角色為灰色墨西哥帽子，最後還有一頂帽子是紫色墨西哥帽子。

對於這四頂不同的墨西哥帽子，我設計了十個不同的按鈕來達到各式各樣的功能。其詳細功能如下方表格的說明。

按鈕名稱	操作說明以及功能介紹
增加月球速度	每按一次按鈕，月球繞地球旋轉速度提高
減少月球速度	每按一次按鈕，月球繞地球旋轉速度減少
增加地球速度	每按一次按鈕，地球繞太陽旋轉速度提高
減少地球速度	每按一次按鈕，地球繞太陽旋轉速度減少
增加太陽大小	每按一次按鈕，太陽會逐漸變大
減少太陽大小	每按一次按鈕，太陽會逐漸變小
重製模擬	回到初始狀態

切換背景顏色	每按一次按鈕會自動切換背景顏色
以地球為螢幕中心旋轉	畫面會以地球為中心點，顯示相對於其他星球運轉的模式
以太陽為螢幕中心旋轉	畫面會以太陽為中心點，顯示相對於其他星球運轉的模式

除了以上十個按鈕之外，還有另外兩種控制滑鼠的方式來觀察墨西哥帽子，其詳細的解釋如下方的表格說明。

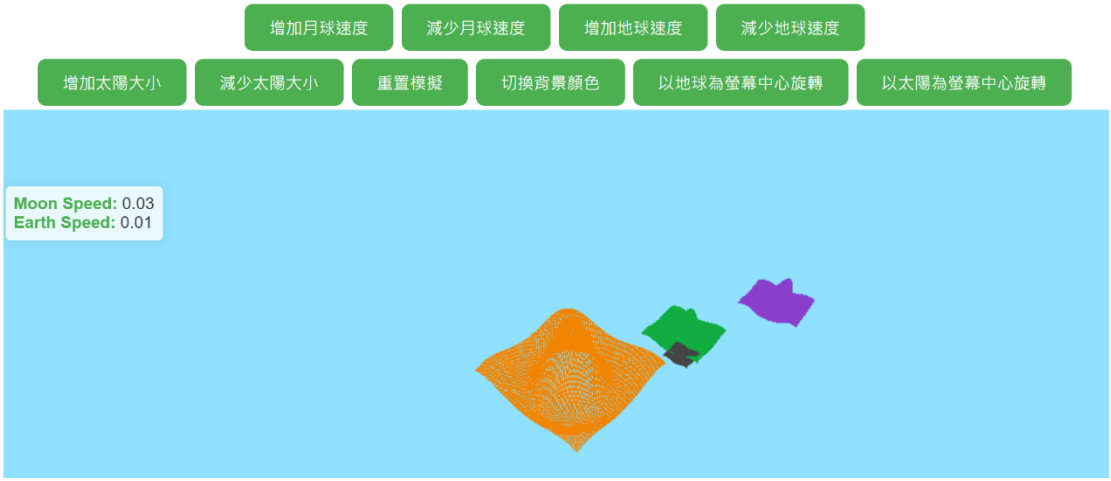
滑鼠操作	操作後的效果
滑鼠滾輪	可以同時控制四個墨西哥帽子的形狀變大或變小
長按滑鼠左鍵	長按左鍵來移動可以用不同視角觀看墨西哥帽子

這個作業讓我更深入理解了 3D 空間中物體運動的模擬和呈現。通過利用墨西哥帽子來模擬設置太陽、地球和月球的位置、尺寸和顏色，這讓我學會了如何使用 WebGL 函式庫來製作動態且具有層次感的場景。而在調整地球和月球繞行速度功能的過程中，我學會了如何控制物體的運動速度，這擴展了我對動態模擬和互動性程式的理解。另外像是透過撰寫調整視角和觀察點的功能，這些都讓我能夠親自體會和觀察場景中不同物體的運動。而最令我印象深刻的部分是我能夠獨立完成整個作

業並且成功地呈現出一個動態的太陽系模擬場景，這份作業不僅能讓我學習如何畫出圖形，還增進了我解決問題的能力。

4. 執行畫面：

在執行畫面中除了有十個控制按鈕和四個墨西哥帽子外，還有一個小方塊在畫面的左手邊。用來顯示當前月亮跟地球各別的速度，好讓操作者可以清楚了解現在月亮跟地球的即時速度。詳細的執行畫面如下方的截圖。



5. 程式碼：

以下為完整 html 檔的程式碼

```
<!DOCTYPE html>
<html>
<head>
    <title>墨西哥帽子_邱奕達</title>
    <style>

        canvas { width: 80%; height: 70% }
        /* 按鈕樣式 */
        .btn {
```

```
            background-color: #4CAF50;  /* 按鈕背景色 */
            border: none;               /* 去除邊框 */
            color: white;               /* 文字顏色 */
            padding: 12px 24px;
            text-align: center;         /* 文字置中 */
            text-decoration: none;      /* 文字無裝飾 */
            display: inline-block;
            font-size: 16px;            /* 字體大小 */
            margin: 4px 2px;
            transition-duration: 0.4s;  /* 過渡效果時間 */
            cursor: pointer;             /* 指標類型 */
            border-radius: 8px;         /* 圓角 */
        }
        /* 按鈕懸停效果 */
        .btn:hover {
            background-color: #45a049;
        }
        /* 按鈕區域樣式 */
        .btn-group {
            text-align: center; /* 將按鈕置中 */
            margin-top: 20px;   /* 與上方的間距 */
        }
        .btn-row {
            display: inline-block; /* 讓按鈕每四個一行 */
        }

    </style>
</head>
<body>
     <!-- 按鈕區域 -->
    <div class="btn-group">
        <!-- 每一行按鈕 -->
        <div class="btn-row">
            <button id="increaseMoonSpeed" class="btn">增加月球速度
</button>
            <button id="decreaseMoonSpeed" class="btn">減少月球速度
</button>
            <button id="increaseEarthSpeed" class="btn">增加地球速度
```

```
</button>
            <button id="decreaseEarthSpeed" class="btn">減少地球速度
</button>
        </div>
        <br> <!-- 換行 -->
        <div class="btn-row">
            <button id="increaseSunSize" class="btn">增加太陽大小
</button>
            <button id="decreaseSunSize" class="btn">減少太陽大小
</button>
            <button id="resetSimulation" class="btn">重置模擬</button>
            <button id="toggleBackground" class="btn">切換背景顏色
</button>
            <button id="focusEarth" class="btn">以地球為螢幕中心旋轉
</button>
            <button id="focusSun" class="btn">以太陽為螢幕中心旋轉
</button>
        </div>
    </div>

    <div id="stats"></div>

    <!-- 引入需要的 JavaScript 檔案 -->
    <script src="js/three.js"></script>
    <script src="js/TrackballControls.js"></script>

    <script>
        var stats = document.getElementById('stats');

        var camera, scene, renderer, controls;
        var moonSpeed = 0.025, earthSpeed = 0.01;
        var sun, earth, moon;
        var angle = 0, angle2 = 0;

        var nRows = 50;
        var nColumns = 50;
        var pointsArray = [];
        var pointsArray1 = [];
```

```
        var pointsArray2 = [];
        var data = new Array(nRows);

        // 初始化資料陣列
        for (var i = 0; i < nRows; i++) {
            data[i] = new Array(nColumns);
            for (var j = 0; j < nColumns; j++) {
                var x = Math.PI * (4 * i / nRows - 2.0);
                var y = Math.PI * (4 * j / nRows - 2.0);
                var r = Math.sqrt(x * x + y * y);
                if (r) data[i][j] = Math.sin(r) / r;
                else data[i][j] = 1;
            }
        }

        init();
        animate();

        function init() {
            // 創建相機
            camera = new THREE.PerspectiveCamera(75, window.innerWidth /
window.innerHeight, 1, 1000);
            camera.position.z = 500;

            // 創建場景
            scene = new THREE.Scene();

            // 創建渲染器
            renderer = new THREE.WebGLRenderer();
            renderer.setSize(window.innerWidth, window.innerHeight);
            renderer.setClearColor(0x8fe1ff);
            document.body.appendChild(renderer.domElement);

            // 創建太陽 sun
            var sungeometry=new THREE.Geometry();
            for(var i=0; i<nRows-1; i++) {
                for(var j=0; j<nColumns-1;j++) {
                    // 將太陽表面點座標添加至 sungeometry.vertices
```

```
                    sungeometry.vertices.push(new
THREE.Vector4(100*(2*i/nRows-1), 100*(data[i][j]), 100*(2*j/nColumns-
1),1.0));
                    sungeometry.vertices.push(new
THREE.Vector4(100*(2*(i+1)/nRows-1), 100*(data[i+1][j]),
100*(2*j/nColumns-1), 1.0));
                    sungeometry.vertices.push(new
THREE.Vector4(100*(2*(i+1)/nRows-1), 100*(data[i+1][j+1]),
100*(2*(j+1)/nColumns-1), 1.0));
                    sungeometry.vertices.push(new
THREE.Vector4(100*(2*i/nRows-1), 100*(data[i][j+1]),
100*(2*(j+1)/nColumns-1), 1.0) );
                }
            }
            // 創建太陽的表面三角形面
            for(var i=0; i<nRows*nColumns*5; i+=4) {
                sungeometry.faces.push( new THREE.Face3(i,i+1,i+2));
                sungeometry.faces.push( new THREE.Face3(i,i+3,i+2));
            }
            // 創建太陽材質與物件
            var sunmaterial = new THREE.MeshBasicMaterial({wireframe:
true, color: 0xF28500});
            sun = new THREE.Mesh(sungeometry, sunmaterial);
            sun.position.set(0, 0, 0);
            scene.add(sun);

            // 創建地球 earth
            var earthgeometry=new THREE.Geometry(); // 創建地球的幾何體
（Geometry）
            // 迴圈建立地球表面的點座標
            for(var i=0; i<nRows-1; i++) {
                for(var j=0; j<nColumns-1;j++) {
                    earthgeometry.vertices.push(new
THREE.Vector4(50*(2*i/nRows-1), 50*(data[i][j]), 50*(2*j/nColumns-
1),1.0));
                    earthgeometry.vertices.push(new
THREE.Vector4(50*(2*(i+1)/nRows-1), 50*(data[i+1][j]),
50*(2*j/nColumns-1), 1.0));
```

```
                    earthgeometry.vertices.push(new
THREE.Vector4(50*(2*(i+1)/nRows-1), 50*(data[i+1][j+1]),
50*(2*(j+1)/nColumns-1), 1.0));
                    earthgeometry.vertices.push(new
THREE.Vector4(50*(2*i/nRows-1), 50*(data[i][j+1]),
50*(2*(j+1)/nColumns-1), 1.0) );

                }
            }
            // 創建地球表面的三角形面
            for(var i=0; i<nRows*nColumns*5; i+=4) {
                earthgeometry.faces.push( new THREE.Face3(i,i+1,i+2));
                earthgeometry.faces.push( new THREE.Face3(i,i+3,i+2));
            }
            // 創建地球的材質（Material）
            var earthmaterial   = new
THREE.MeshBasicMaterial({wireframe: true, color: 0x11AD42});
            earth = new THREE.Mesh(earthgeometry, earthmaterial);
            angle=0;    // 初始化地球的旋轉角度為 0
            scene.add(earth);   // 將地球添加到場景中

            // 創建 另一個墨西哥帽子
            var earthgeometry2 =new THREE.Geometry(); // 創建墨西哥帽子的
幾何體（Geometry）
            // 迴圈建立墨西哥帽子表面的點座標
            for(var i=0; i<nRows-1; i++) {
                for(var j=0; j<nColumns-1;j++) {
                    earthgeometry2.vertices.push(new
THREE.Vector4(50*(2*i/nRows-1), 50*(data[i][j]), 50*(2*j/nColumns-
1),1.0));
                    earthgeometry2.vertices.push(new
THREE.Vector4(50*(2*(i+1)/nRows-1), 50*(data[i+1][j]),
50*(2*j/nColumns-1), 1.0));
                    earthgeometry2.vertices.push(new
THREE.Vector4(50*(2*(i+1)/nRows-1), 50*(data[i+1][j+1]),
50*(2*(j+1)/nColumns-1), 1.0));
                    earthgeometry2.vertices.push(new
THREE.Vector4(50*(2*i/nRows-1), 50*(data[i][j+1]),
```

```
50*(2*(j+1)/nColumns-1), 1.0) );

                }
            }
            // 創建墨西哥帽子表面的三角形面
            for(var i=0; i<nRows*nColumns*5; i+=4) {
                earthgeometry2.faces.push( new THREE.Face3(i,i+1,i+2));
                earthgeometry2.faces.push( new THREE.Face3(i,i+3,i+2));
            }
            // 創建墨西哥帽子的材質（Material）
            var earthmaterial   = new
THREE.MeshBasicMaterial({wireframe: true, color: 0x8a3fcc});
            earth2 = new THREE.Mesh(earthgeometry2, earthmaterial);
            angle=0;    // 初始化墨西哥帽子的旋轉角度為 0
            scene.add(earth2);   // 將墨西哥帽子添加到場景中

            // 創建月球 moon
            var moongeometry=new THREE.Geometry();  // 創建月球的幾何體
（Geometry）
            moongeometry.verticesNeedUpdate = true;     // 標記需要更新頂
點位置
            // 迴圈建立月球表面的點座標
            for(var i=0; i<nRows-1; i++) {
                for(var j=0; j<nColumns-1;j++) {
                    // 將月球表面點的座標加入到 moongeometry.vertices 中
                    moongeometry.vertices.push(new
THREE.Vector4(20*(2*i/nRows-1), 20*(data[i][j]), 20*(2*j/nColumns-
1),1.0));
                    moongeometry.vertices.push(new
THREE.Vector4(20*(2*(i+1)/nRows-1), 20*(data[i+1][j]),
20*(2*j/nColumns-1), 1.0));
                    moongeometry.vertices.push(new
THREE.Vector4(20*(2*(i+1)/nRows-1), 20*(data[i+1][j+1]),
20*(2*(j+1)/nColumns-1), 1.0));
                    moongeometry.vertices.push(new
THREE.Vector4(20*(2*i/nRows-1), 20*(data[i][j+1]),
20*(2*(j+1)/nColumns-1), 1.0) );
                }
```

```
            }
            // 創建月球表面的三角形面
            for(var i=0; i<nRows*nColumns*5; i+=4) {
                moongeometry.faces.push( new THREE.Face3(i,i+1,i+2));
                moongeometry.faces.push( new THREE.Face3(i,i+3,i+2));
            }
            // 創建月球的材質（Material）
            var moonmaterial = new THREE.MeshBasicMaterial({wireframe:
true, color: 0x444444});
            moon = new THREE.Mesh(moongeometry, moonmaterial);
            angle2=0;   // 初始化月球的旋轉角度為 0
            scene.add(moon);    // 將月球添加到場景中


            // 增加月球速度的按鈕
            var increaseMoonSpeedBtn =
document.getElementById('increaseMoonSpeed');
            // 減少月球速度的按鈕
            var decreaseMoonSpeedBtn =
document.getElementById('decreaseMoonSpeed');
            // 增加地球速度的按鈕
            var increaseEarthSpeedBtn =
document.getElementById('increaseEarthSpeed');
            // 減少地球速度的按鈕
            var decreaseEarthSpeedBtn =
document.getElementById('decreaseEarthSpeed');
            // 增加太陽大小的按鈕
            var increaseSunSizeBtn =
document.getElementById('increaseSunSize');
            // 減少太陽大小的按鈕
            var decreaseSunSizeBtn =
document.getElementById('decreaseSunSize');
            // 重置模擬的按鈕
            var resetSimulationBtn =
document.getElementById('resetSimulation');
            // 以地球為中心旋轉
            var focusEarthButton =
document.getElementById('focusEarth');
            // 以太陽為中心旋轉
```

```
var focusSunButton = document.getElementById('focusSun');

// 監聽按鈕的點擊事件，改變月球和地球的速度
increaseMoonSpeedBtn.addEventListener('click', function() {
    moonSpeed += 0.025; // 增加月球速度
});

decreaseMoonSpeedBtn.addEventListener('click', function() {
    if ( moonSpeed > 0.025 ) {
        moonSpeed -= 0.025; // 減少月球速度
    }
    else {
        moonSpeed = 0; // 如果速度已經是 0 或負值，將速度設置為 0
    }
});

increaseEarthSpeedBtn.addEventListener('click', function() {
    earthSpeed += 0.025; // 增加地球速度
});

decreaseEarthSpeedBtn.addEventListener('click', function() {
    if ( earthSpeed > 0.025 ) {
        earthSpeed -= 0.025; // 減少地球速度
    }
    else {
        earthSpeed = 0; // 如果速度已經是 0 或負值，將速度設置為
0

    }
});

// 讓背景顏色隨著頁面大小改變
window.addEventListener('resize', function () {
    var width = window.innerWidth;
    var height = window.innerHeight;

    // 調整渲染器的大小
    renderer.setSize(width, height);
```

```
    // 根據新的視窗大小更新背景顏色區塊
    if (backgroundToggle) {
        renderer.setClearColor(0x8fe1ff);
    } else {
        renderer.setClearColor(0x3ce5ba);
    }

    // 更新相機的長寬比例
    camera.aspect = width / height;
    camera.updateProjectionMatrix();
});

resetSimulationBtn.addEventListener('click', function() {
    // 重置月球和地球速度
    moonSpeed = 0.05;   // 重置月球速度為初始值
    earthSpeed = 0.01;  // 重置地球速度為初始值

    // 重置太陽、地球和月亮的位置
    sun.rotation.y = 0;          // 太陽的旋轉歸零
    sun.scale.set(1, 1, 1);      // 太陽的尺寸還原為初始值
    sun.position.set(0, 0, 0);   // 太陽的位置歸零

    earth.rotation.y = 0;        // 地球的旋轉歸零
    earth.position.set(0, 0, 0);// 地球的位置歸零
    angle = 0;                   // 地球的旋轉角度歸零

    moon.rotation.y = 0;         // 月亮的旋轉歸零
    moon.position.set(50, 0, 0);// 月亮的位置重置
    angle2 = 0;                  // 月亮的旋轉角度歸零
});

increaseSunSizeBtn.addEventListener('click', function () {
    sun.scale.multiplyScalar(1.1); // 增加太陽大小
});

decreaseSunSizeBtn.addEventListener('click', function () {
    sun.scale.multiplyScalar(0.9); // 減少太陽大小
});
```

```
            controls = new THREE.TrackballControls(camera);
            controls.addEventListener('change', render);    // 監聽控制器
的變化事件，當發生變化時渲染場景

            focusEarthButton.addEventListener('click', function() {
                camera.lookAt(earth.position);
                // 將控制器綁定到地球上
                controls.target = earth.position;
            });
            focusSunButton.addEventListener('click', function() {
                camera.lookAt(sun.position);
                // 將控制器綁定到太陽上
                controls.target = sun.position;
            });

            var stats = document.getElementById('stats');
            var backgroundToggle = false; // 背景開關狀態

            // 找到切換背景的按鈕元素
            var toggleBackgroundBtn =
document.getElementById('toggleBackground');

            // 監聽切換背景按鈕的點擊事件
            toggleBackgroundBtn.addEventListener('click', function () {
                // 切換背景色
                if (backgroundToggle) {
                    renderer.setClearColor(0x8fe1ff); // 如果開關為 true，
設置為預設的背景色
                } else {
                    renderer.setClearColor(0x3ce5ba); // 如果開關為
false，設置為淺綠色
                }
                backgroundToggle = !backgroundToggle; // 切換開關狀態
            });
        }

        function animate() {
```

```
            // 更新顯示月球速度和地球速度的資訊
            stats.innerHTML = "<span style='font-weight: bold; color:
#4CAF50;'>Moon Speed:</span> " + moonSpeed.toFixed(2) + "<br><span
style='font-weight: bold; color: #4CAF50;'>Earth Speed:</span> " +
earthSpeed.toFixed(2);

            // 設置資訊顯示的樣式
            stats.style.position = "absolute";
            stats.style.top = "200px";
            stats.style.left = "10px";
            stats.style.fontFamily = "Arial, sans-serif";
            stats.style.fontSize = "16px";
            stats.style.color = "#333";
            stats.style.backgroundColor = "rgba(255, 255, 255, 0.8)";
            stats.style.padding = "8px";
            stats.style.borderRadius = "5px";
            stats.style.boxShadow = "0 0 10px rgba(0, 0, 0, 0.1)";

            // 進行動畫更新
            requestAnimationFrame(animate);
            controls.update();  // 更新控制器
            render();    // 渲染畫面
        }

        // 事件監聽器設定
        function render() {
            angle += earthSpeed;    // 更新地球運動的角度
            angle2 += moonSpeed;    // 更新月球運動的角度

            // 更新太陽的旋轉
            sun.rotation.y += 0.01;

            // 更新地球的旋轉，並根據角度設置其位置
            earth.rotation.y += 0.01;
            earth.position.set(200 * Math.cos(angle), 0, 200 *
Math.sin(angle));

            // 更新墨西哥帽子的旋轉，並根據角度設置其位置
```

```
            earth2.rotation.y += 0.01;
            earth2.position.set(400 * Math.cos(angle), 0, 400 *
Math.sin(angle));

            // 更新月球的旋轉，並根據相對於地球的角度設置其位置
            moon.rotation.y += 0.01;
            moon.position.set(200 * Math.cos(angle) + 50 *
Math.cos(angle2), 0, 200 * Math.sin(angle) + 50 * Math.sin(angle2));

            renderer.render(scene, camera);// 渲染場景
        }

    </script>
</body>
</html>
```