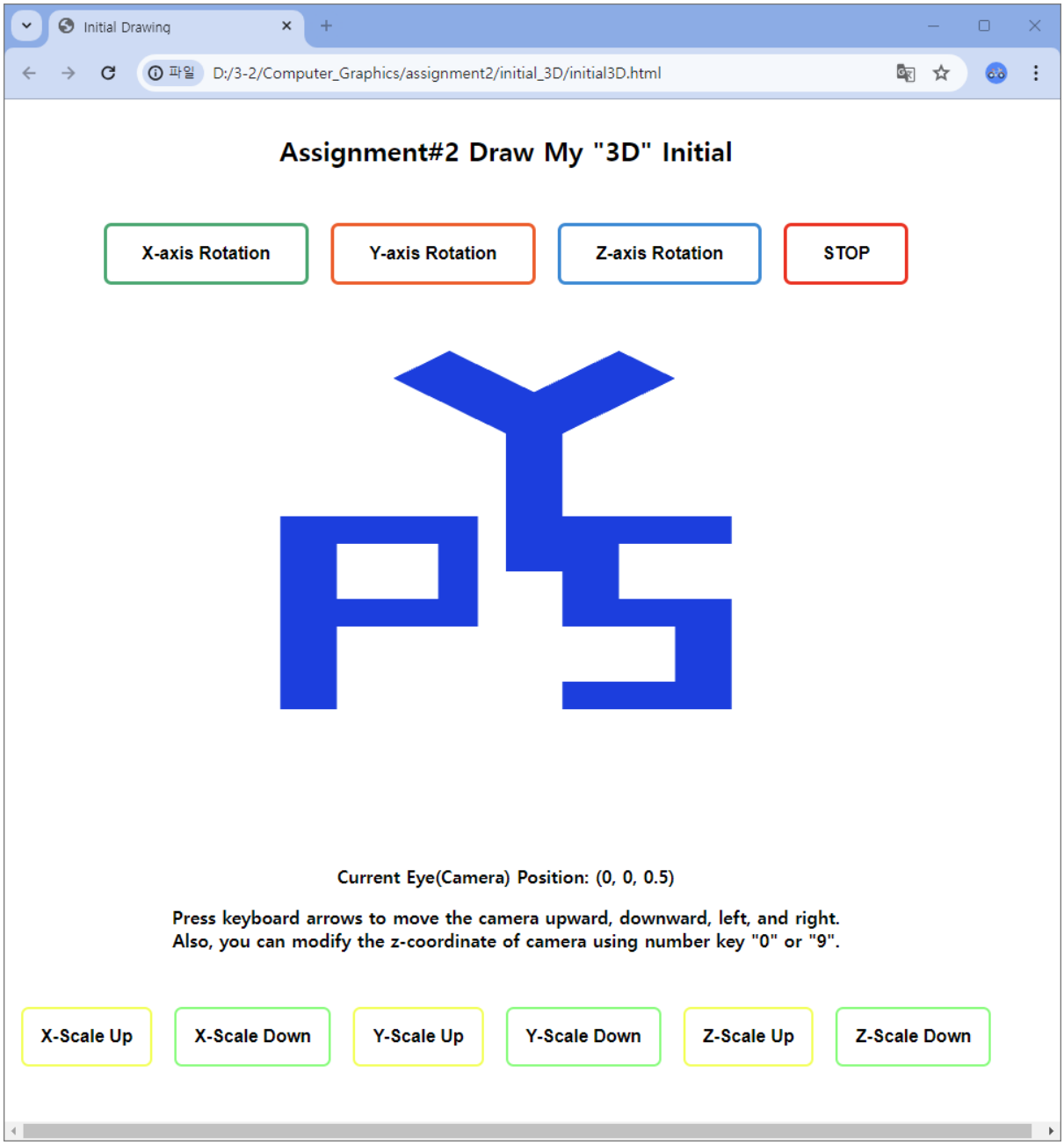
과제 2번. 이니셜 3d 모델 제작

202126894 소프트웨어학과 박승연

[결과물 시작 화면 스크린샷]



[Initial3D.js 코드 코멘트]

p, s, y 이니셜을 생성하는 데에 필요한 vertex의 3d coordinate 정보를 32FloatArray형의 vertices에 저장합니다. primitive로 삼각형을 사용하여 전체 vertex의 개수만큼 한 번에 모든 vertex를 생

성할 것이므로 numVertices 변수를 선언하여 전체 vertices 변수의 길이를 3으로 나눈(3d coordinate을 배열에 나열하므로) 값을 저장한 후 drawArrays의 인수로 값을 넘겨줍니다.

```
var vertices = new Float32Array([ ···
]);
```

```
gl.drawArrays(gl.TRIANGLES, 0, numVertices);
```

vertex shader에 uniform mat4 형의 modelViewMatrix를 선언하여 Rotation, translate, Viewing에 대한 정보를 coordinate system에 추가합니다. (mat4 -> homogeneous vector를 포함) modelViewMatrix는 위치 정보와 벡터 곱 연산을 수행하여 위치 정보를 업데이트할 수 있습니다.

```
uniform mat4 modelViewMatrix;

void main()
{
    gl_PointSize = 3.0;
    gl_Position = modelViewMatrix * vPosition;
```

Application 영역에서는 vPositionLoc과 modelViewMatrixLoc 변수를 선언하여 program으로부터 shader에 선언된 형식을 가져온 후 buffer에 바인딩하고, 데이터를 삽입하여 렌더링 합니다. 이때 vPosition이 3차원 좌표를 가리키고 modelViewMatrix는 4차원 행렬이므로 vertexAttribPointer에는 3차원임을 표지하고, modelViewMatrixLoc에는 uniformMatrix4fv 함수를 사용하여 modelViewMatrix를 맵핑합니다.

```
vPositionLoc = gl.getAttribLocation(program, "vPosition");

modelViewMatrixLoc = gl.getUniformLocation(program, "modelViewMatrix");
```

```
gl.enableVertexAttribArray(vPositionLoc);
gl.vertexAttribPointer(vPositionLoc, 3, gl.FLOAT, false, 0, 0);
```

```
gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(modelViewMatrix));
```

다음으로 (x, y, z)별 회전 각도를 3차원 벡터인 theta 변수에 저장하여 업데이트하고, x, y, z축 정보를 가리키는 _Axis 이름의 변수들을 생성합니다. _축 회전 버튼 입력이 들어오면 theta의 _Axis index value를 0.1 degree만큼 증가시키고, animation frame을 생성하여 theta 값 업데이트를 반복합니다. (stop을 누르면 frame을 해제)

```
var theta = [0, 0, 0];
var xAxis = 0, yAxis = 1, zAxis = 2;
```

```
document.getElementById("Stop").onclick = function () {···
}
// Rotate with 3 axis when the button is pressed.
document.getElementById("X").onclick = function () {···
}
document.getElementById("Y").onclick = function () {···
}
document.getElementById("Z").onclick = function () {···
}
```

이렇게 업데이트 되는 theta값을 가지고 rotate_ 함수를 수행한 결과물(4차원 행렬)들을 모두 곱하여 Rotation matrix를 생성합니다.

```
modelViewMatrix = mult(modelViewMatrix, rotateX(theta[xAxis]));
modelViewMatrix = mult(modelViewMatrix, rotateY(theta[yAxis]));
modelViewMatrix = mult(modelViewMatrix, rotateZ(theta[zAxis]));
```

(no translation operation is used)

다음으로 3차원 벡터 scaleMatrix를 생성하여 회전과 같이 _up 또는 _down 버튼이 눌렸을 때 해당 축에 지정된 index(i.e., xAxis=0)의 scale 값을 0.1만큼 증가 또는 감소시켜 scalem 함수의 인수로 업데이트된 scale 값을 넘겨줍니다. render 함수 내부에서 이를 Rotation Matrix에 곱하여 Scale Matrix를 반영합니다. *(physical computational order는 scale, rotation 순서이지만 graphic 연산 order는 역순으로 진행된다는 이야기를 보아서 rotate 다음에 scalem을 배치했습니다)*

```
document.getElementById("xup").onclick = function () {···
}
document.getElementById("xdown").onclick = function () {···
}
document.getElementById("yup").onclick = function () {···
}
document.getElementById("ydown").onclick = function () {···
}
document.getElementById("zup").onclick = function () {···
}
document.getElementById("zdown").onclick = function () {···
}
```

```
modelViewMatrix = mult(modelViewMatrix, scalem(scale));
```

마지막으로 lookAt 함수를 사용하여 Viewing Matrix를 modelViewMatrix의 마지막 요소로 추가했습니다.

카메라의 위치 coordinate을 저장하는 eyePosition 변수, 카메라가 가리키는 coordinate을 저장하는 atPoint 변수를 선언하고, 보편적인 [0, 1, 0] upDirection 벡터를 생성하여 y축의 증가 방향이 카메라의 위쪽 방향이 되도록 초기 lookAt 변수들을 설정했습니다.

```
var eyePosition = [0., 0., 0.5]; // Camera Position
var atPoint = [0., 0., 0];       // The point camera is looking at
var upDirection = [0, 1, 0];
```

차례대로 lookAt function에 대입하여 Viewing matrix까지 생성해 준 다음 Rotation Matrix, Scale Matrix, Viewing Matrix들의 곱으로 modelViewMatrix를 구성하여 uniformMatrix4fv를 통해 최종 coordinate 정보를 완성했습니다. lookAt function의 파라미터들 또한 키보드 입력을 통해 변경 가능하도록 EventListener를 추가하여 입력에 따라 eyePosition의 x, y, z 값을 증가 또는 감소시킬 수 있게 했습니다.

```
document.addEventListener("keydown", function (event) {
    switch (event.key) {
        case "ArrowUp":
            eyePosition[yAxis] += 0.1;  // Move up
            break;
        case "ArrowDown":
            eyePosition[yAxis] -= 0.1;  // Move down
            break;
        case "ArrowLeft":
            eyePosition[xAxis] -= 0.1;  // Move left
            break;
        case "ArrowRight":
            eyePosition[xAxis] += 0.1;  // Move right
            break;
        case "0":
            eyePosition[zAxis] += 0.1;
            break;
        case "9":
            eyePosition[zAxis] -= 0.1;
            break;
    }
    updateEyePosition();
    render();
});
```

```
function updateEyePosition() {
    modelViewMatrix = lookAt(eyePosition, atPoint, upDirection);
    gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(modelViewMatrix));
}
```

또한 업데이트되는 eyePosition 정보를 사용자가 확인할 수 있도록 information device를 html에 추가하여 업데이트된 eyePosition을 Text 형식으로 표시하고, 키보드 입력을 통해 eyePosition value를 바꿀 수 있음을 고지했습니다.

```
document.getElementById("eyePosition").innerText = `(${eyePosition[0]}, ${eyePosition[1]}, ${eyePosition[2]})`
```

```
<div class="info">
    <p>Current Eye(Camera) Position: <span id="eyePosition"></span></p>
    <div id="lookAt-info">
        Press keyboard arrows to move the camera upward, downward, left, and right.
        <br>
        Also, you can modify the z-coordinate of camera using number key "0" or "9".
    </div>
</div>
```

이외에는 1번 과제의 initial.html에서 html의 <style>과 <body> 내용을 과제에 맞게 수정 및 추가하거나, 이니셜의 초기 색상 정보를 1번 과제와 다르게 변동 사항 없이 Math.random()함수를 사용하여 지정한 후에 그대로 사용하도록 변경하거나 했습니다.