**컴퓨터 그래픽스(SW) 과제 2**

**:** **Lighting & Texture**



**학번: 32193430**

**이름: 이재원**

**담당교수: 송 인 식 교수님**

**분반: 2분반**

**제출일: 2021. 11. 21**

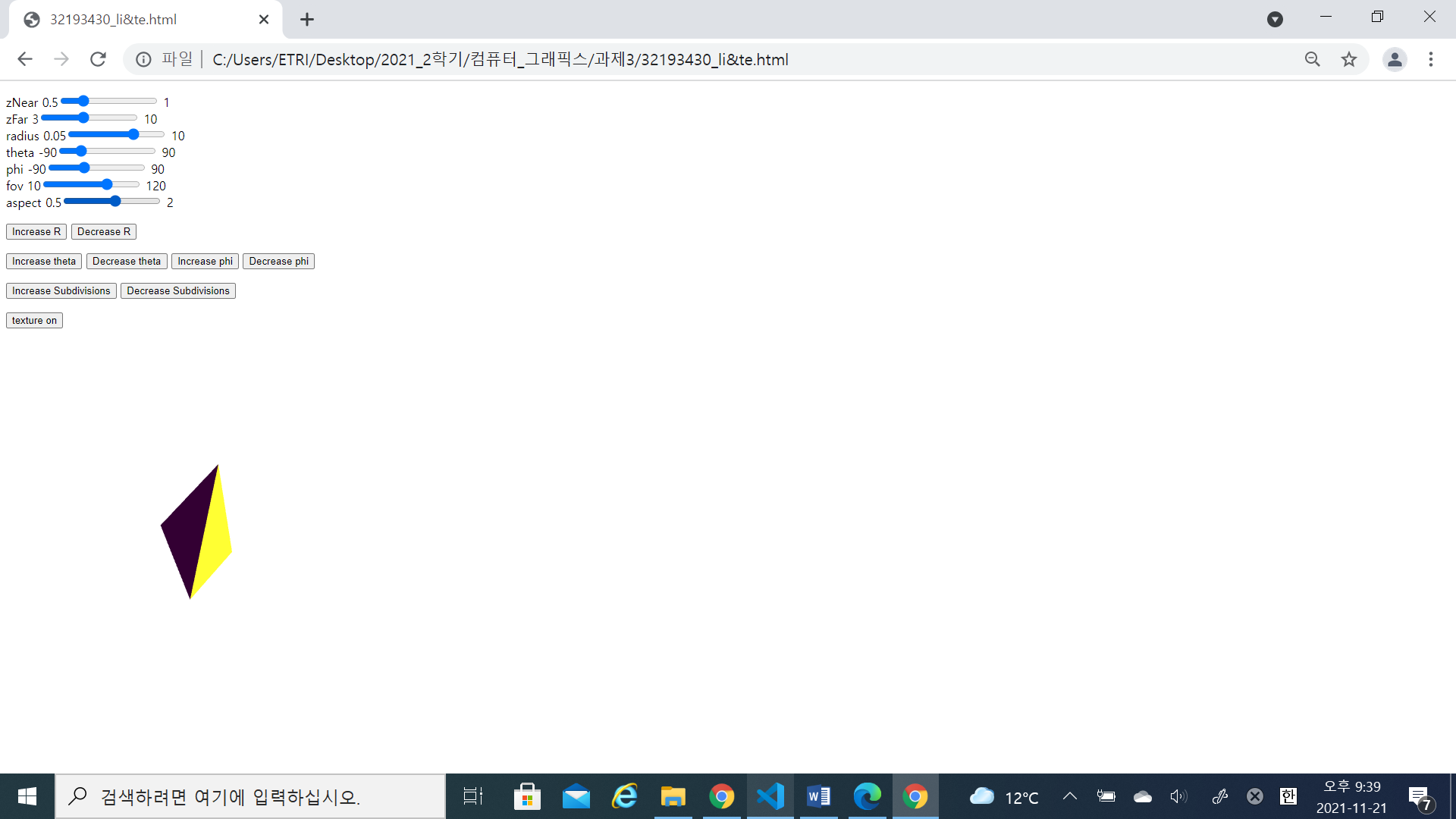
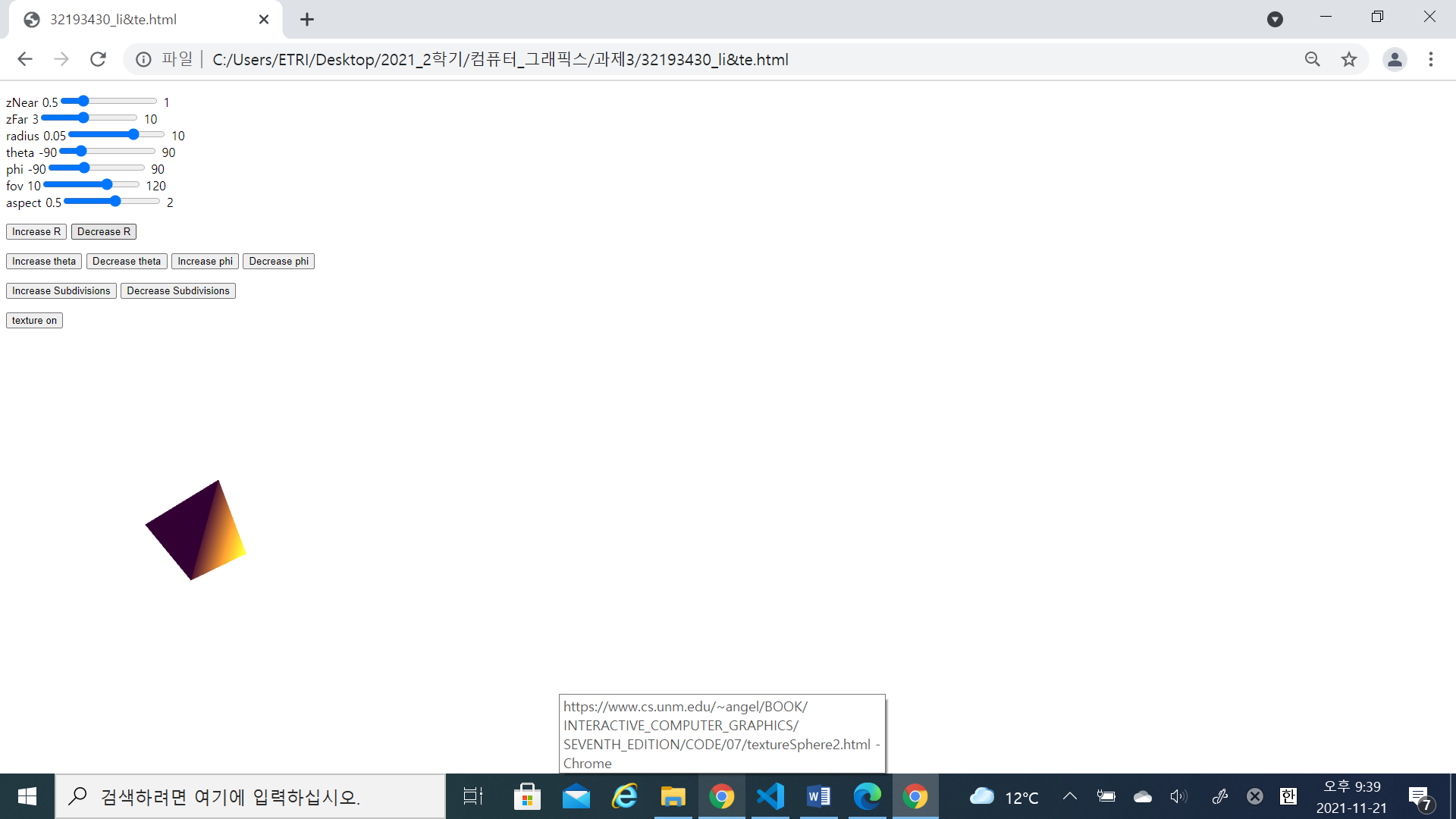
코드 편집은 Visual Studio Code 소프트웨어를 이용하였다.

**I. 핵심 아이디어**

**Pyramid Control과 Lighting과 Texture를 동시에 제어하는 방법**

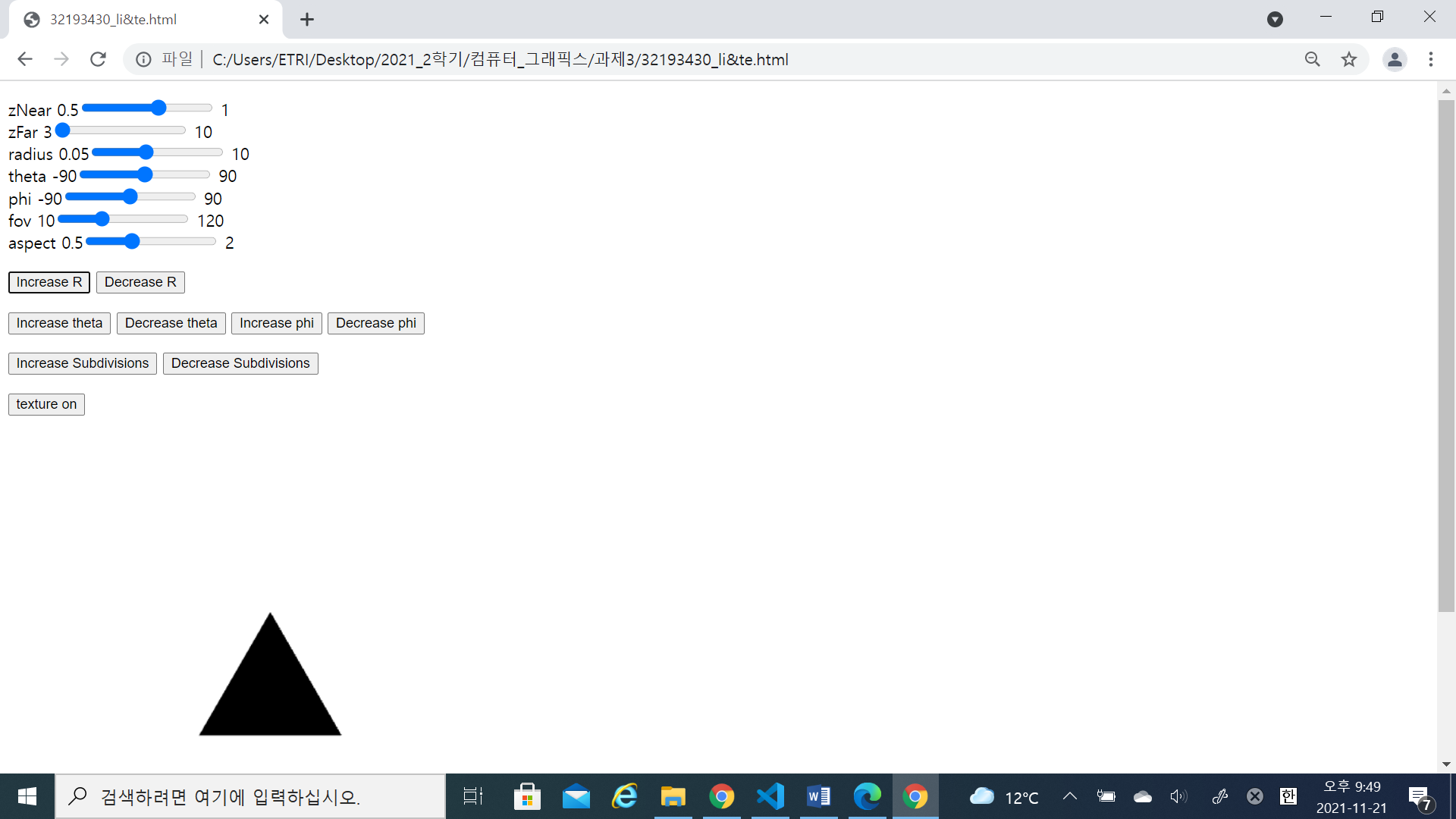
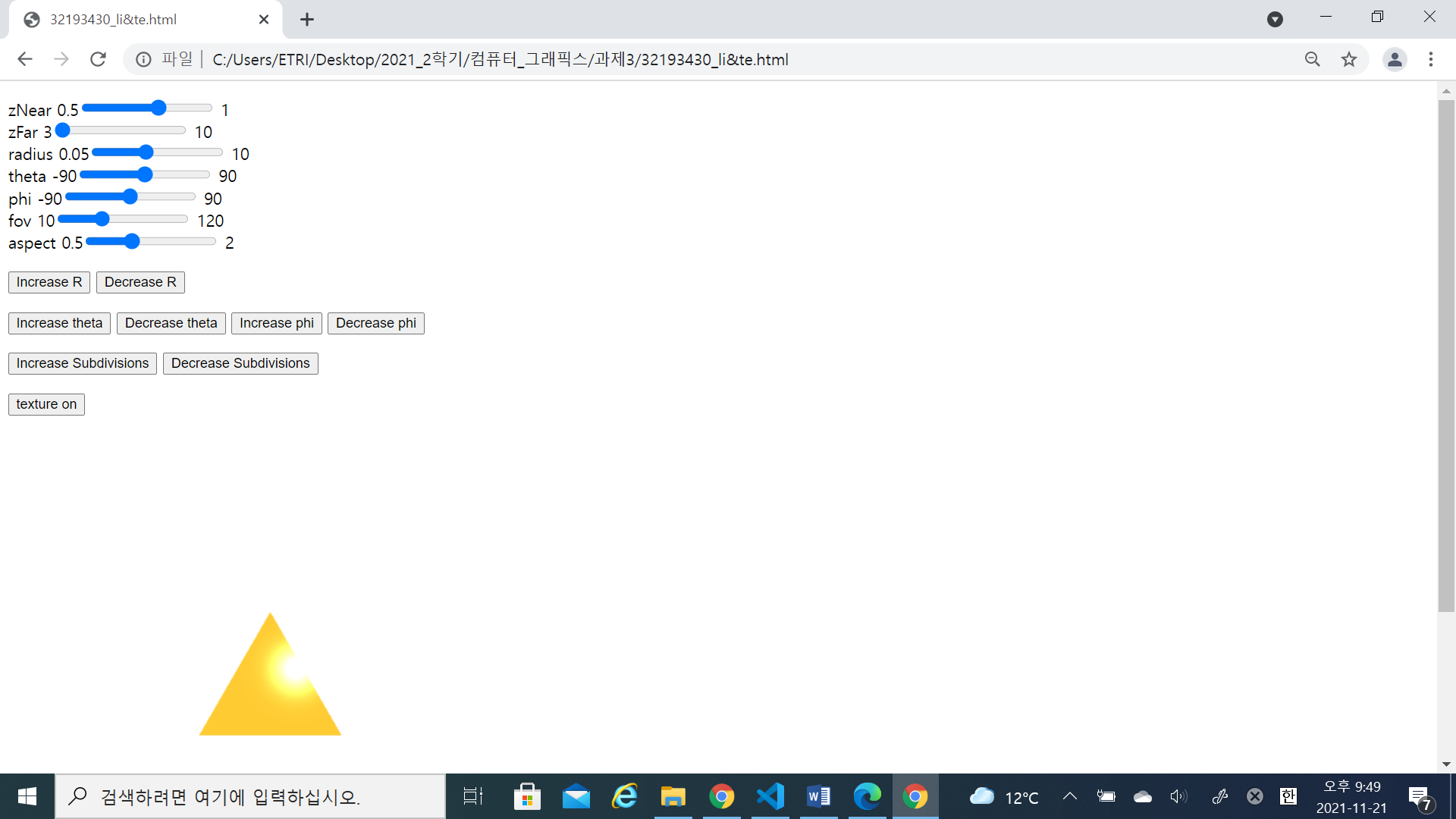
변수 c를 선언하고 초기값을 0으로 둔다. Pyramid Control과 Lighting 버튼을 눌렀을 때 c의 값이 0이나 1로 변한다. 1이면 True, 0이면 false인 것과 if/else문을 이용하여 projectionMatrix에 ortho(left, right, bottom, ytop, near, far) 함수를 대입하거나 perspective(fovy, aspect, near1, far1) 함수를 대입한다(자세한 내용은 II. 주요 코드 설명 참고)

위와 같이 하면 Pyramid Control과 lighting을 동시에 제어할 수 있다.



Increase R을 하고 aspect를 줄인 모습

마찬가지로, 변수 o를 선언하고 초기값을 0으로 둔다. 이후 texture on을 눌렀을 때 변수 o에 1을 대입하고 이후 if/else문을 이용하여 texture을 입힐 지 여부를 결정한다(이 부분은 완벽하게 구현하지 못하였다. 자세한 내용은 IV. 한계 참고)



Texture를 입히기 전후 모습

**II. 주요 코드 설명**

* Javascript

1) Lighting

- Pyramid Control과 Lighting을 동시에 제어

Lighting Control 버튼을 누를 때 c에는 1이 대입되고, Pyramid Control 버튼을 누를 때 c에는 0이 대입된다. 변수 c = 1일 때 if문 안의 코드가 작동하고, 변수 c = 0일 때 else문 안의 코드가 작동한다.

var c = 0

// (중략)

document.getElementById("Button0").onclick = function(){radius \*= 2.0;

c = 1;};

    document.getElementById("Button1").onclick = function(){radius \*= 0.5;

c = 1;};

// (중략)

document.getElementById("zFarSlider").onchange = function(event) {

        far1 = event.target.value;

        c = 0;

    };

 document.getElementById("zNearSlider").onchange = function(event) {

        near1 = event.target.value;

        c = 0;

    };

// (중략)

if(c == 1){

projectionMatrix = ortho(left, right, bottom, ytop, near, far);

       // c = 0;

    }

    else{

        projectionMatrix = perspective(fovy, aspect, near1, far1);

       //c = 1;

- Pyramid와 Subdivision을 위한 함수

function triangle(a, b, c) {

     var t1 = subtract(b, a);

     var t2 = subtract(c, a);

     var normal = normalize(cross(t2, t1));

     normal = vec4(normal);

     normal[3]  = 0.0;

     normalsArray.push(normal);

     normalsArray.push(normal);

     normalsArray.push(normal);

     pointsArray.push(a);

     pointsArray.push(b);

     pointsArray.push(c);

     index += 3;

}

function divideTriangle(a, b, c, count) {

    if ( count > 0 ) {

        var ab = mix( a, b, 0.5);

        var ac = mix( a, c, 0.5);

        var bc = mix( b, c, 0.5);

        ab = normalize(ab, true);

        ac = normalize(ac, true);

        bc = normalize(bc, true);

        divideTriangle( a, ab, ac, count - 1 );

        divideTriangle( ab, b, bc, count - 1 );

        divideTriangle( bc, c, ac, count - 1 );

        divideTriangle( ab, bc, ac, count - 1 );

    }

    else {

        triangle( a, b, c );

    }

}

function tetrahedron(a, b, c, d, n) {

    divideTriangle(a, b, c, n);

    divideTriangle(d, c, b, n);

    divideTriangle(a, d, b, n);

    divideTriangle(a, c, d, n);

}

- 조명 속성 & 물질 속성

여기서 vec4의 마지막 요소는 불투명도를 나타낸다.

var lightPosition = vec4(1.0, 1.0, 1.0, 0.0 );

var lightAmbient = vec4(0.2, 0.2, 0.2, 1.0 );

var lightDiffuse = vec4( 1.0, 1.0, 1.0, 1.0 );

var lightSpecular = vec4( 1.0, 1.0, 1.0, 1.0 );

var materialAmbient = vec4( 1.0, 0.0, 1.0, 1.0 );

var materialDiffuse = vec4( 1.0, 0.8, 0.0, 1.0 );

var materialSpecular = vec4( 1.0, 1.0, 1.0, 1.0 );

var materialShininess = 20.0;

2) Texture

- Texture 객체

function configureTexture( image ) {

    texture = gl.createTexture();

    gl.bindTexture( gl.TEXTURE\_2D, texture );

    gl.pixelStorei(gl.UNPACK\_FLIP\_Y\_WEBGL, true);

    gl.texImage2D( gl.TEXTURE\_2D, 0, gl.RGB,

         gl.RGB, gl.UNSIGNED\_BYTE, image );

    gl.generateMipmap( gl.TEXTURE\_2D );

    gl.texParameteri( gl.TEXTURE\_2D, gl.TEXTURE\_MIN\_FILTER,

                      gl.NEAREST\_MIPMAP\_LINEAR );

    gl.texParameteri( gl.TEXTURE\_2D, gl.TEXTURE\_MAG\_FILTER, gl.NEAREST );

    gl.uniform1i(gl.getUniformLocation(program, "texture"), 0);

}

- 무늬 생성

 Convert floats to ubytes for texture

var image2 = new Uint8Array(4\*texSize\*texSize);

    for ( var i = 0; i < texSize; i++ )

        for ( var j = 0; j < texSize; j++ )

           for(var k =0; k<4; k++)

                image2[4\*texSize\*i+4\*j+k] = 255\*image1[i][j][k];

- Texture 적용

var texture = gl.createTexture();

    gl.bindTexture( gl.TEXTURE\_2D, texture );

    gl.pixelStorei(gl.UNPACK\_FLIP\_Y\_WEBGL, true);

    gl.texImage2D(gl.TEXTURE\_2D, 0, gl.RGBA, texSize, texSize, 0,

            gl.RGBA, gl.UNSIGNED\_BYTE, image2);

    gl.generateMipmap( gl.TEXTURE\_2D );

    gl.texParameteri( gl.TEXTURE\_2D, gl.TEXTURE\_MIN\_FILTER,

                    gl.NEAREST\_MIPMAP\_LINEAR );

    gl.texParameteri( gl.TEXTURE\_2D, gl.TEXTURE\_MAG\_FILTER, gl.NEAREST );

    gl.uniform1i(gl.getUniformLocation(program, "texture"), 0)

* HTML

1) lighting

- Lighting과 Texture을 동시에 제어

mediump int o = 0; // 이 값이 0이면 texture가 입혀지고, 1이면 벗겨진다.

// (중략)

if (o < 1) {gl\_FragColor = vec4(1.0, 0.0, 0.0, 1.0)\*texture2D( texture, vec2(s, t));} // o가 0일 때 lighting

    else {gl\_FragColor = fColor;} // o가 1일 때 texture

- 단편 조명 Fragment Shader: 빛을 비추는 모습 구현

uniform vec4 ambientProduct;

uniform vec4 diffuseProduct;

uniform vec4 specularProduct;

uniform float shininess;

varying vec3 N, L, E;

#ifdef GL\_ES

precision highp float;

#endif

void main()

{

    vec4 fColor;

    vec3 H = normalize( L + E );

    vec4 ambient = ambientProduct;

    float Kd = max( dot(L, N), 0.0 );

    vec4  diffuse = Kd\*diffuseProduct;

    float Ks = pow( max(dot(N, H), 0.0), shininess );

    vec4  specular = Ks \* specularProduct;

    if( dot(L, N) < 0.0 ) specular = vec4(0.0, 0.0, 0.0, 1.0);

    fColor = ambient + diffuse +specular;

    fColor.a = 1.0;

- 단편 조명 Vertex Shader: 광원의 위치 구현

<script id="vertex-shader" type="x-shader/x-vertex">

attribute vec4 vPosition;

attribute vec4 vNormal;

varying vec3 N, L, E;

uniform mat4 modelViewMatrix;

uniform mat4 projectionMatrix;

uniform vec4 lightPosition;

uniform mat3 normalMatrix;

varying float s, t;

void main()

{

    vec3 light;

    vec3 pos = (modelViewMatrix \* vPosition).xyz;

    if(lightPosition.z == 0.0)  L = normalize(lightPosition.xyz);

    else  L = normalize(lightPosition).xyz - pos;

    E =  -normalize(pos);

    N = normalize( normalMatrix\*vNormal.xyz);

    float pi = acos(0.0);

    s= 0.5\*acos(vPosition.x)/pi;

    t = 0.5\*asin(vPosition.y/sqrt(1.0-vPosition.x\*vPosition.x))/pi;

    gl\_Position = projectionMatrix \* modelViewMatrix \* vPosition;

}

</script>

2) Texture

- Texture vertex Shading

<script id="vertex-shader" type="x-shader/x-vertex">

// assume both position and normal are in homogeneous form

attribute vec4 vPosition;

uniform mat4 modelViewMatrix;

uniform mat4 projectionMatrix;

varying float s, t;

void

main()

{

    float pi = acos(0.0);

    s= 0.5\*acos(vPosition.x)/pi;

    t = 0.5\*asin(vPosition.y/sqrt(1.0-vPosition.x\*vPosition.x))/pi;

    gl\_Position = projectionMatrix \* modelViewMatrix \* vPosition;

}

</script>

- Texture Fragment Shader

<script id="fragment-shader" type="x-shader/x-fragment">

precision mediump float;

varying float s, t;

uniform sampler2D texture;

void

main()

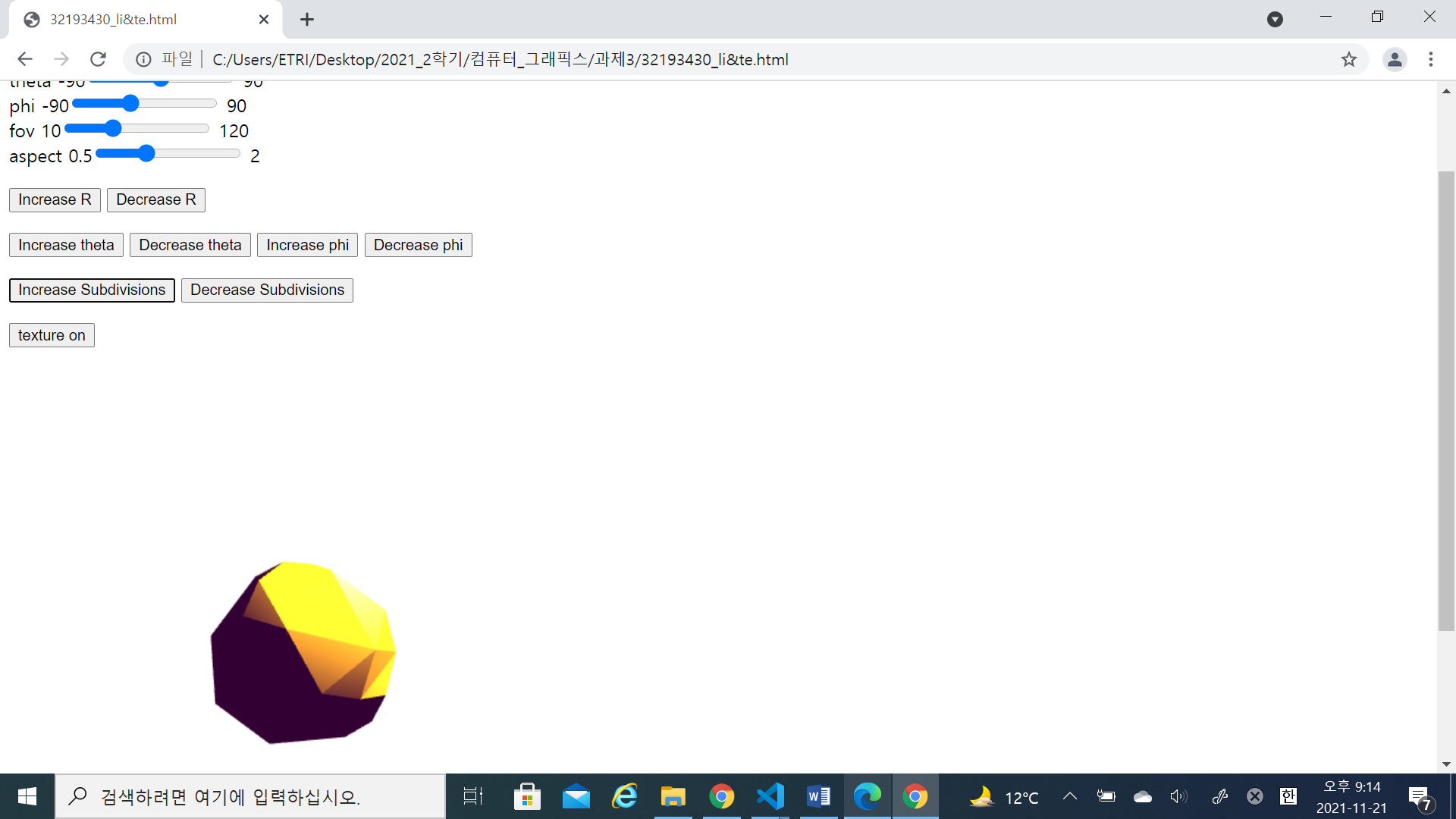
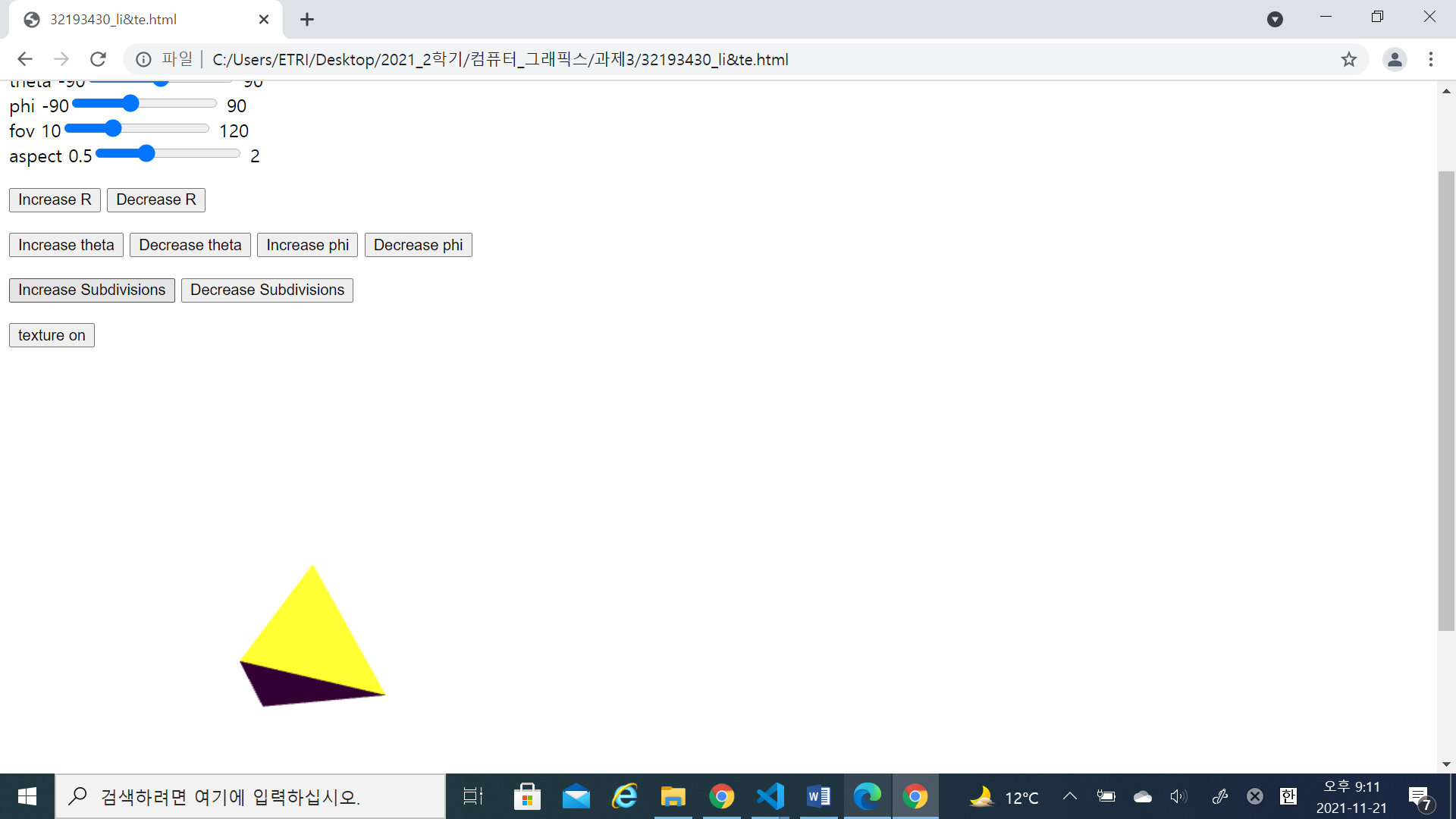
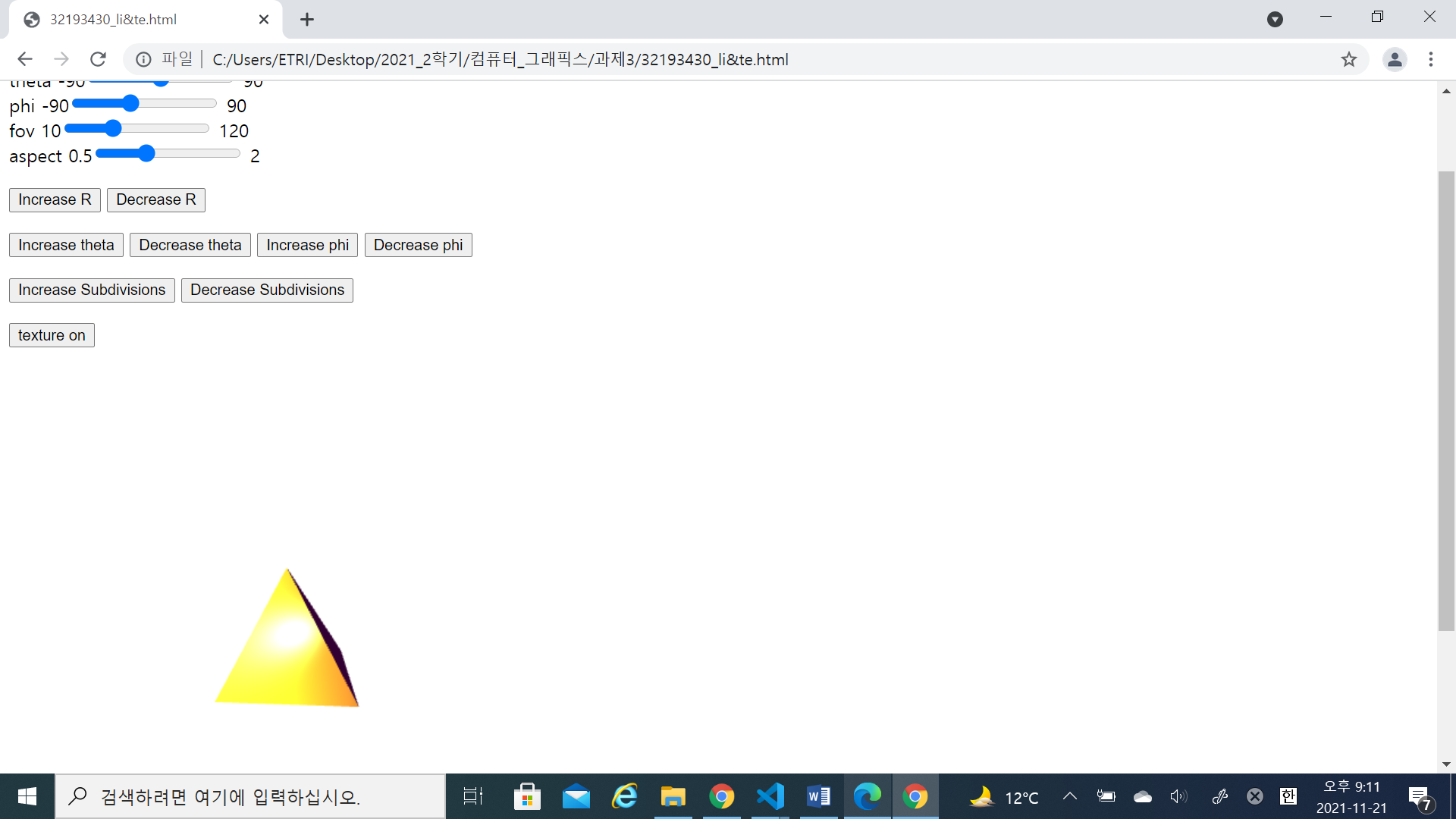
{

    gl\_FragColor = vec4(1.0, 0.0, 0.0, 1.0)\*texture2D( texture, vec2(s, t));

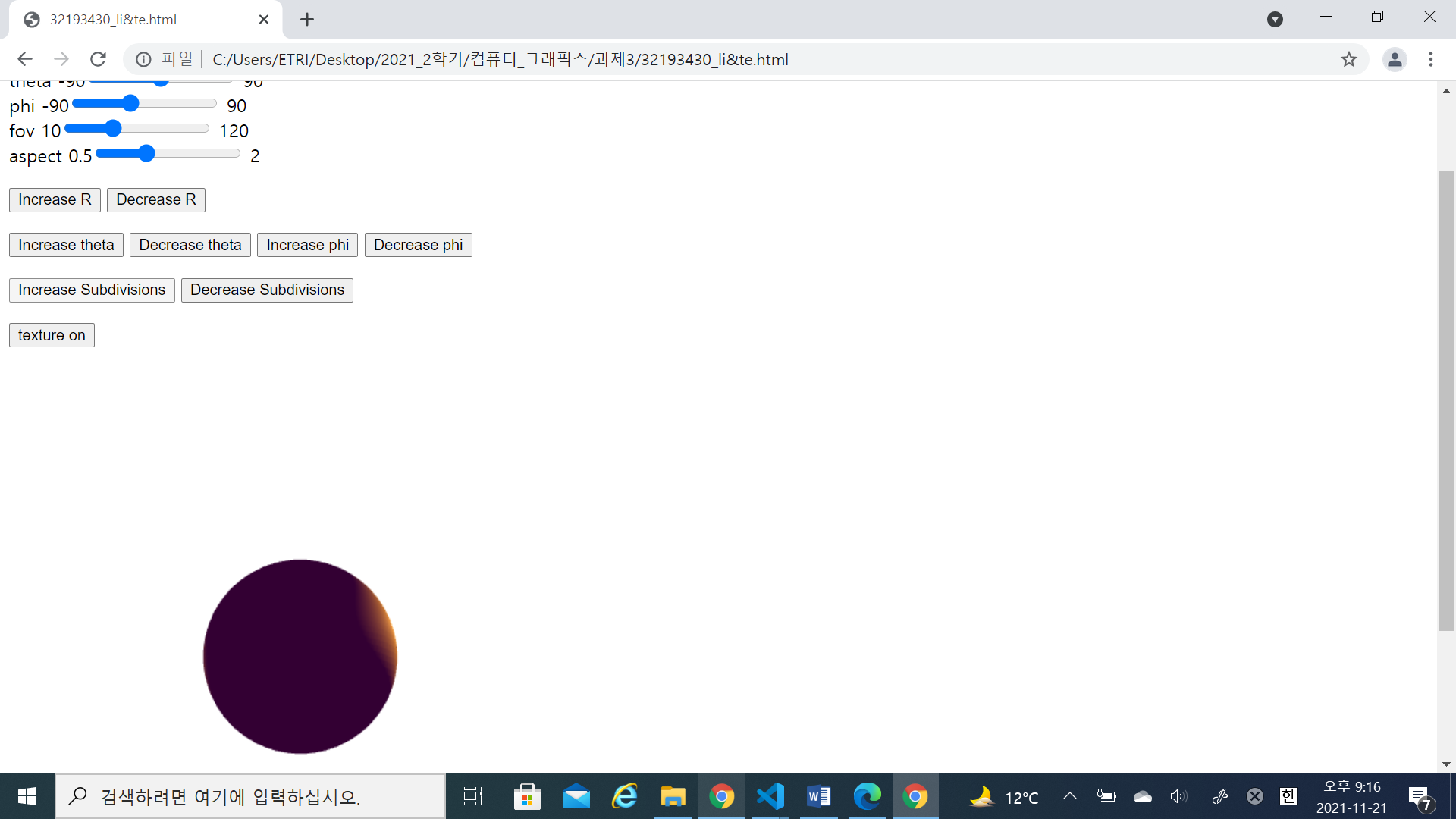
}

**III. 브라우저 테스트 결과(Chrome, Edge)**

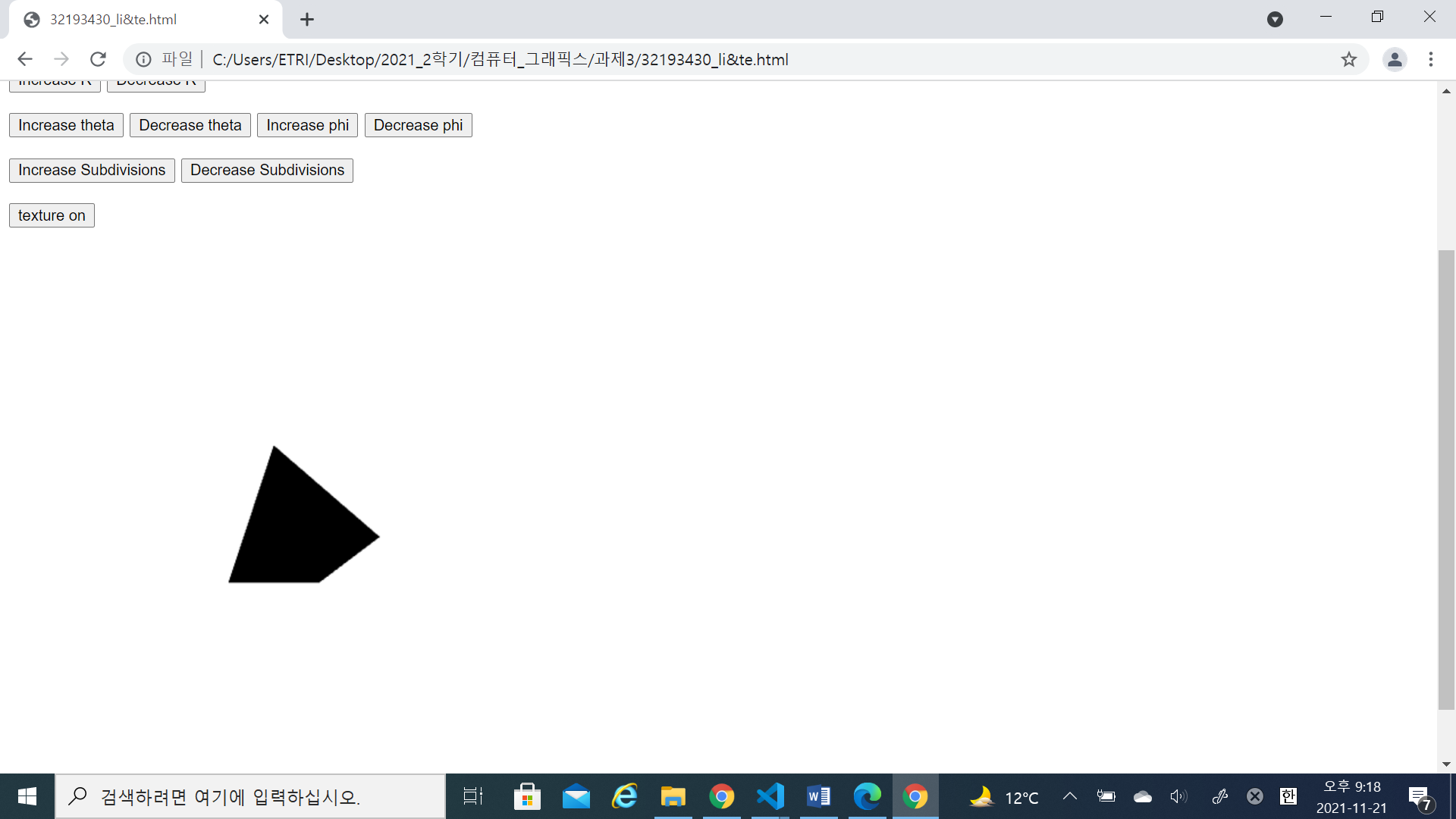
1. **Chrome**



**기본 모습 Increase R & Theta & phi Increase Subdivisions**

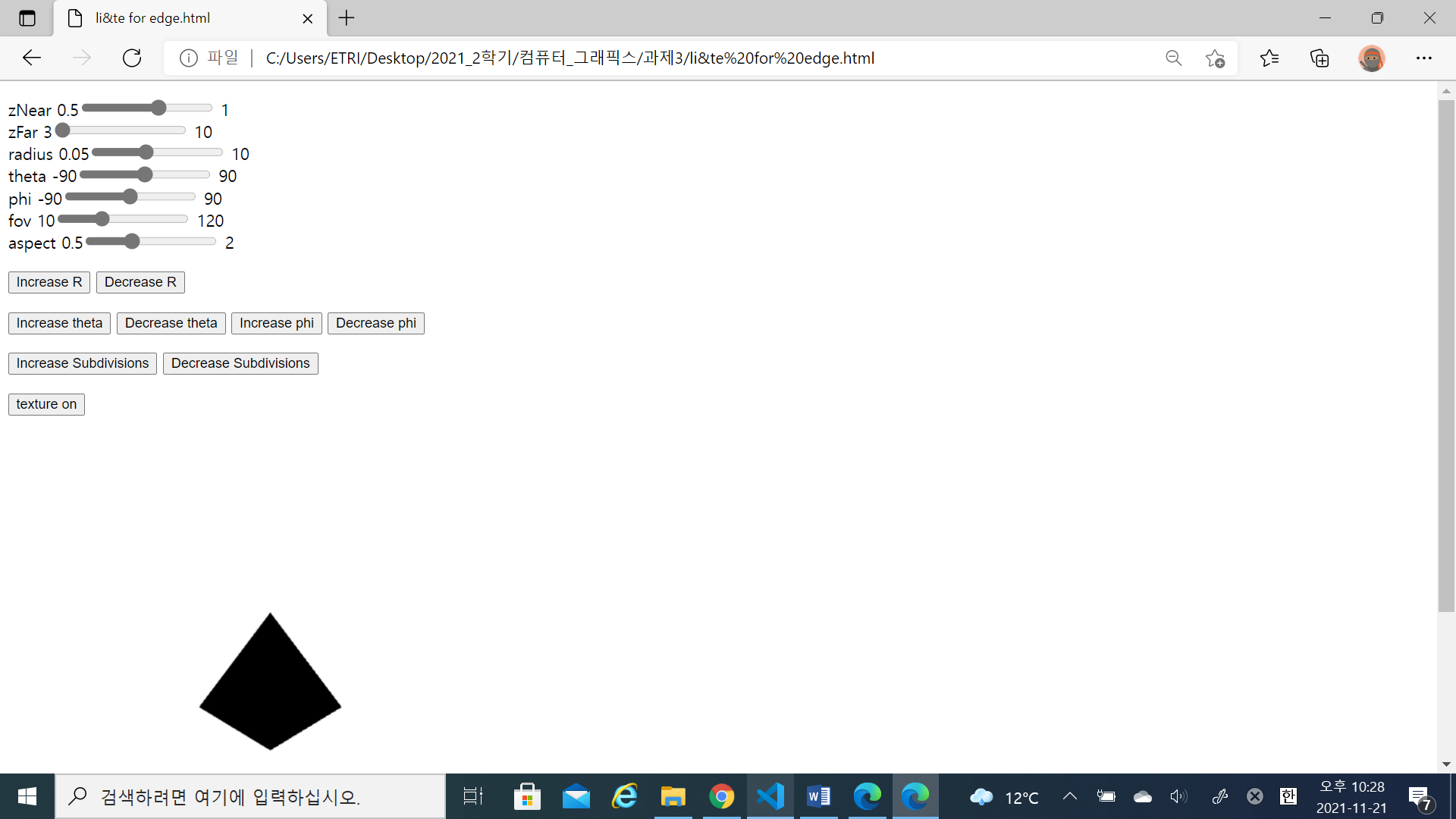
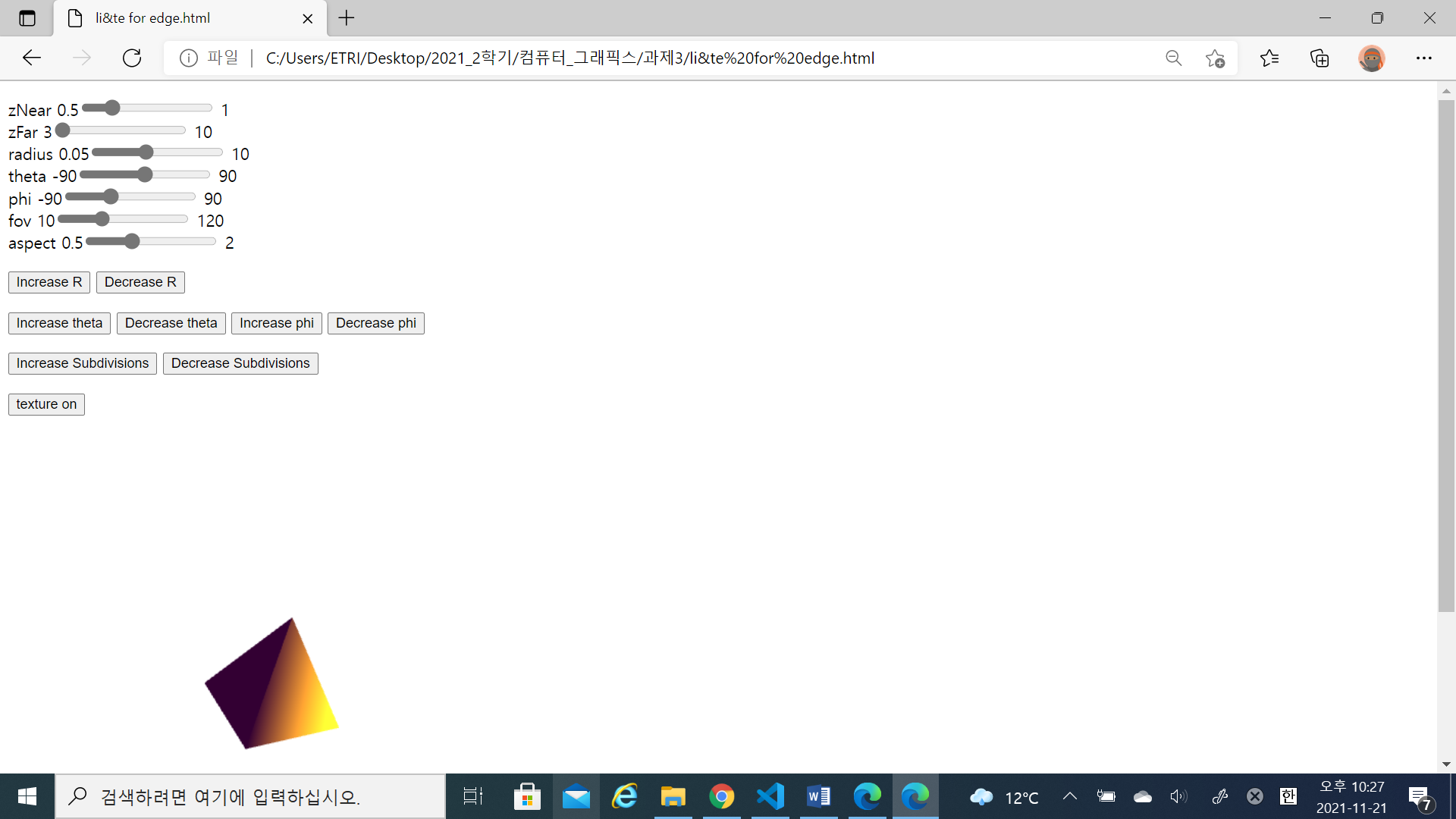


**Max Increase Subdivisions & Decrease R**



**Texture를 입힌 모습**

1. **Edge**



**IV. 한계**

* texture on 버튼을 완벽하게 구현하지 못하였다.

원래 texture on 버튼을 만들어 사용자가 texture를 원하는 대로 입히고 벗길 수 있게 구현할 계획이었다. 그러나 지금 현재 texture on 버튼은 작동하지 않는다. 대신 html 파일의 fragment shader에서

mediump int o = 1; // 이 값이 0이면 texture가 입혀지고, 1이면 벗겨진다.

위 코드로 texture을 입히고 벗길 수 있다. 변수 o 값에 따라 if문이나 else문이 작동하기 때문이다.

if (o < 1) {gl\_FragColor = vec4(1.0, 0.0, 0.0, 1.0)\*texture2D( texture, vec2(s, t));} // o가 0일 때 lighting

    else {gl\_FragColor = fColor;} // o가 1일 때 texture

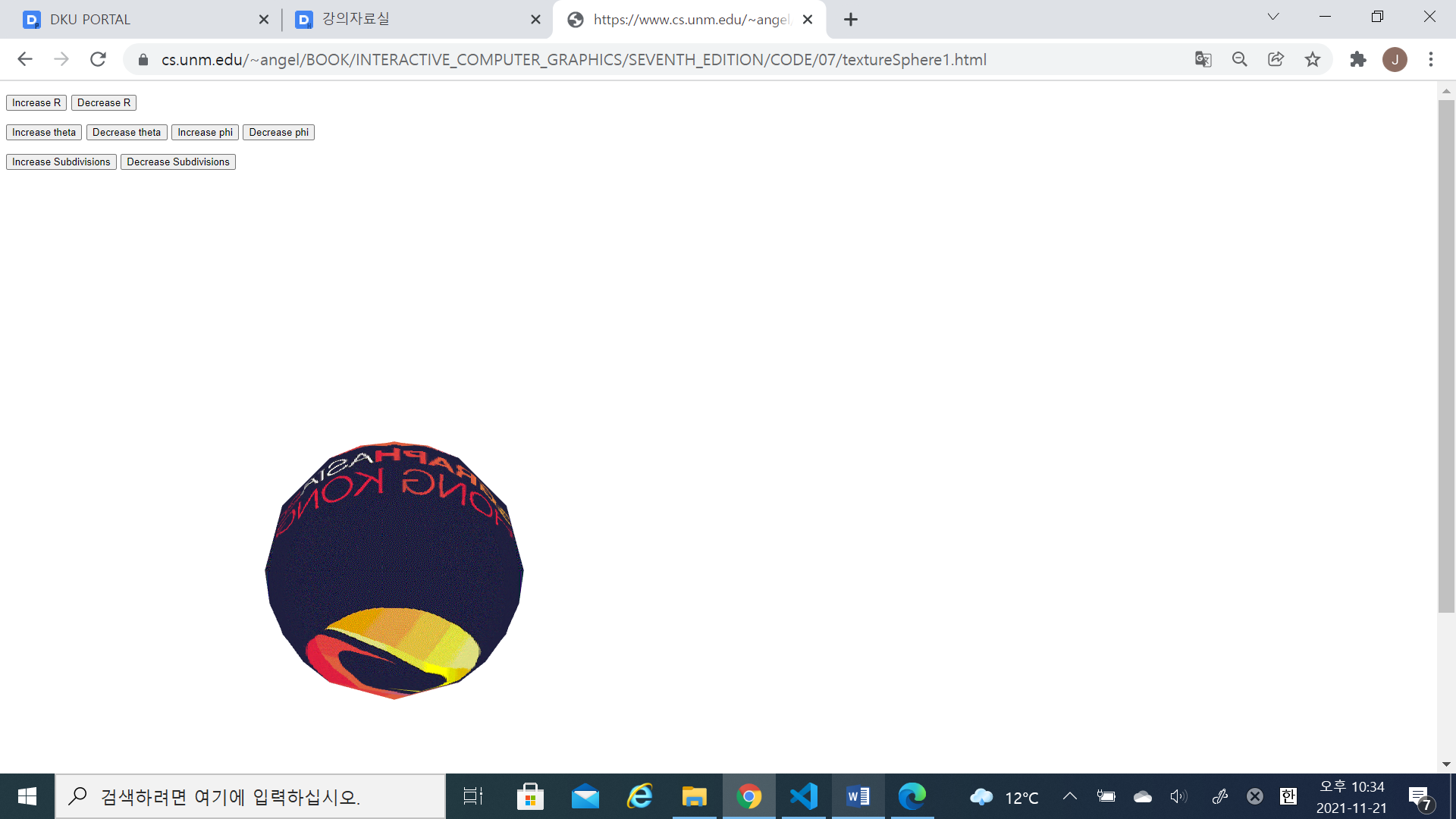
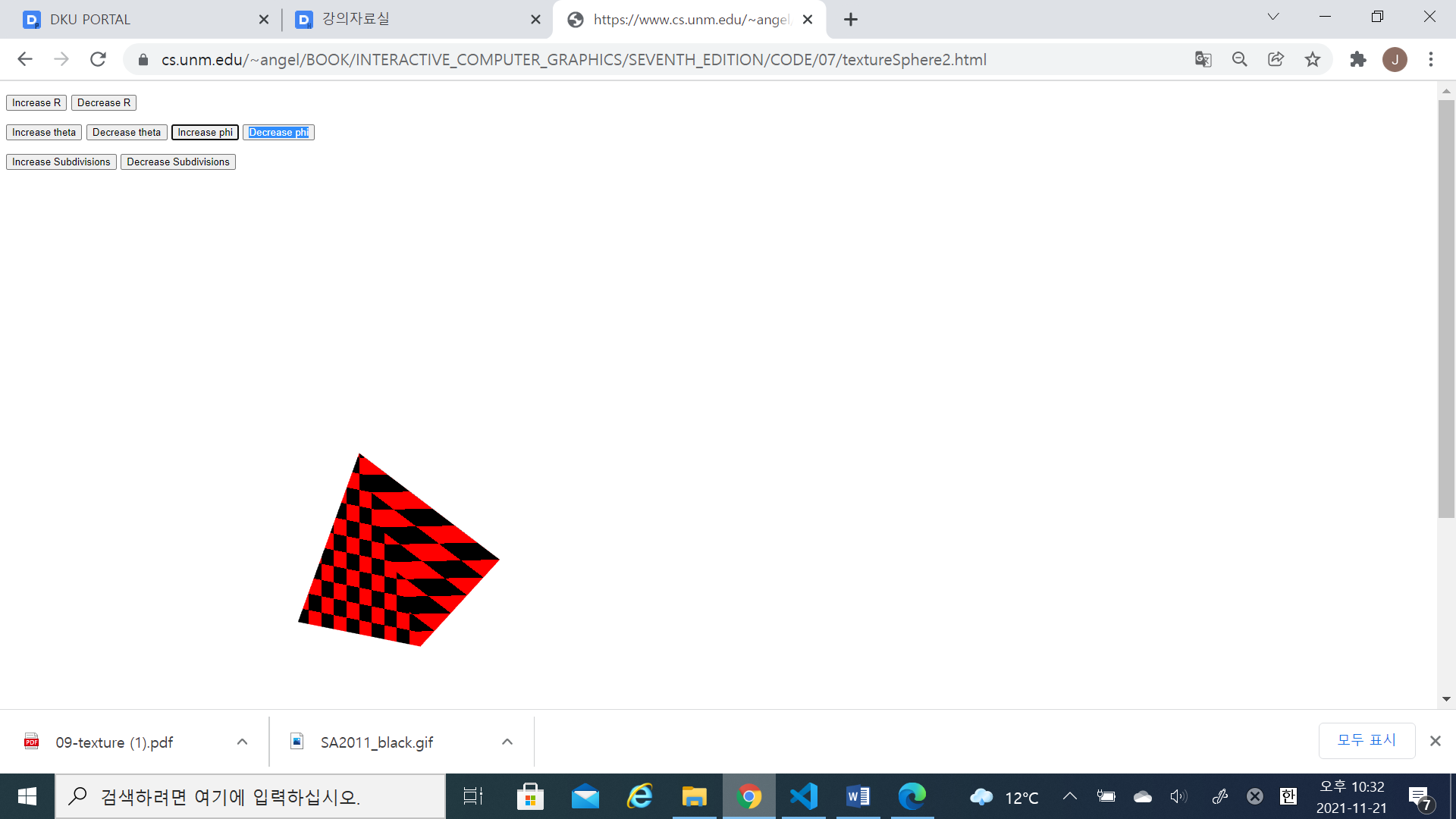
이를 javascript에 적용하여 texture on 버튼을 누르면 변수 o에 0이 대입되어 texture가 입혀지도록 아이디어를 냈다. 그러나 한 가지 문제가 발생하였다. **바로 html 파일 <script></script>내의 javascript 코드와 .js 파일 사이에서 변수가 서로 연동이 되지 않는다는 사실이었다.** 그래서 js 파일 내에서 다음과 같이 o = 0을 대입하여도 texture on 버튼은 작동하지 않았다.

document.getElementById("Button8").onclick = function(){o = 1;}

// texture 버튼 켜기

앞서 언급한 문제만 해결한다면 texture on 버튼으로 사용자가 texture를 입히고 벗길 수 있을 것이다. **지금 현재는 개발자가 변수 o 값을 1과 0으로 바꾸어 주는 것으로 texture적용 유무를 조정할 수 있다.**

또한 원래 아래와 그림과 같이 gif 파일을 적용하여 texture 이미지를 만들거나 직접 연산한 패턴을 적용하려고 했었다.



그러나 위 III. 브라우저 테스트 결과에서 볼 수 있듯이 texture가 까맣게 나온다. 그 원인에 대해서는 아직 연구가 필요하다.

**참고문헌**

Edward Angel, <Interactive Computer Graphics: A Top-down Approach with WebGL 7th edition>, Pearson Education, 2015

<https://www.cs.unm.edu/~angel/BOOK/INTERACTIVE_COMPUTER_GRAPHICS/SEVENTH_EDITION/CODE/07/textureSphere2.html>

<https://www.cs.unm.edu/~angel/BOOK/INTERACTIVE_COMPUTER_GRAPHICS/SEVENTH_EDITION/CODE/06/shadedSphere4.html>