WebGL Project

: 춤추는 꼭두각시

학번: 32193430

이름: 이재원

담당교수: 송 인 식 교수님

분반: 2분반

제출일: 2021. 12. 17

코드 편집은 Visual Studio Code 소프트웨어를 이용하였다.

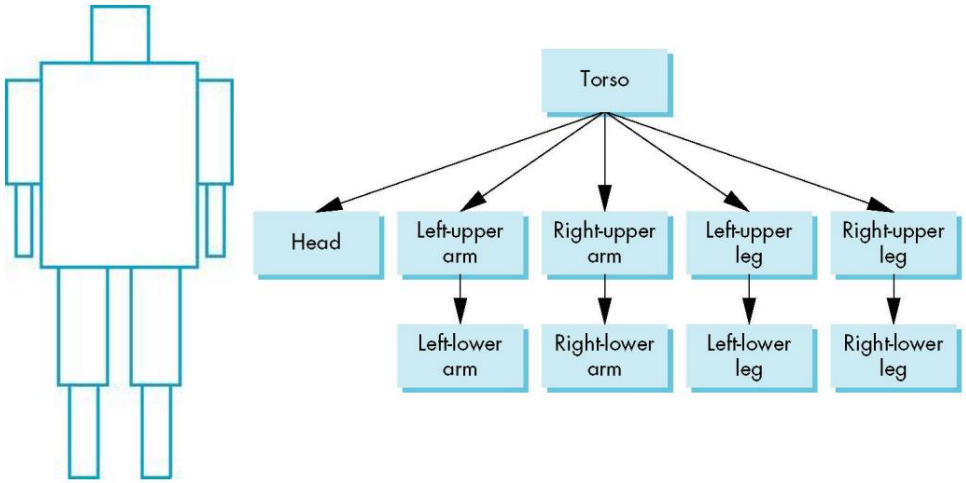
I. 주제 선정 이유

'Just a puppet on a lonely string' 필자가 가장 좋아하는 Coldplay의 노래 Viva la Vida의 가사 중 일부이다. 줄에 매달린 외로운 꼭두각시라는 의미이다. 이 가사를 듣는 순간 WebGL 프로그램으로 꼭두각시를 구현하자는 아이디어를 떠올렸다. 마침 수업에서 Humanoid 구현 관련 수업을 진행하였고, 몸의 각 부분이 회전하도록 만들어 역동적으로 춤추는 꼭두각시를 구현하였다. 꼭두각시의 움직임과 회전은 각종 버튼으로 조정할 수 있도록 만들었다.

II. 설계 및 구현 내용

Github 전체 코드, https://github.com/Jaewon0702/WebGL-Project

1. 참고한 코드: Humanoid Code



꼭두각시는 Torso(몸통)이라는 Parent node에 Head, Left-upper arm ... Right-upper leg와 같은 Child Node를 가지고 각각의 upper는 lower라는 Child Node를 가진다. 위 코드에서는 torso(), Head()와 같은 함수를 이용하여 각 부품에 접근하였다. 그리고 행렬을 이용하여 Parent node에 따라 상대적으로 달라지는 Child Node의 위치를 표현하였다.

```
 instanceMatrix = mat4();

    projectionMatrix = ortho(-10.0,10.0,-10.0, 10.0,-10.0,10.0);
    modelViewMatrix = mat4();
```

```
function traverse(Id) {

   if(Id == null) return;
   stack.push(modelViewMatrix);
   modelViewMatrix = mult(modelViewMatrix, figure[Id].transform);
   figure[Id].render();
   if(figure[Id].child != null) traverse(figure[Id].child);
    modelViewMatrix = stack.pop();
   if(figure[Id].sibling != null) traverse(figure[Id].sibling);
}
```

```
function torso() {

    instanceMatrix = mult(modelViewMatrix, translate(0.0, 0.5*torsoHeight,
0.0) );
    instanceMatrix = mult(instanceMatrix, scale4( torsoWidth, torsoHeight,
torsoWidth));
    gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(instanceMatrix));
    for(var i =0; i<6; i++) gl.drawArrays(gl.TRIANGLE_FAN, 4*i, 4);
}

function head() {

    instanceMatrix = mult(modelViewMatrix, translate(0.0, 0.5 * headHeight,
0.0 ));
    instanceMatrix = mult(instanceMatrix, scale4(headWidth, headHeight,
headWidth) );
    gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(instanceMatrix));
    for(var i =0; i<6; i++) gl.drawArrays(gl.TRIANGLE_FAN, 4*i, 4);
}
```

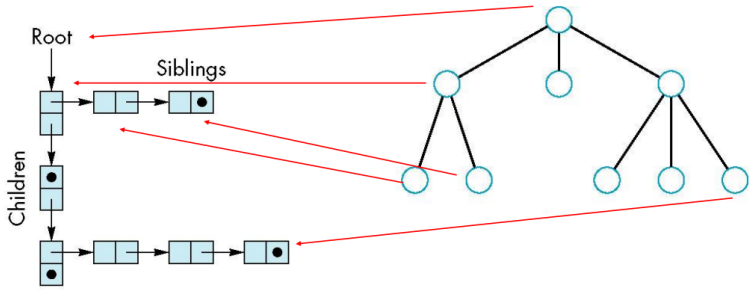
꼭두각시의 위치는 11개의 연결 각(머리에 2개, 나머지 부품에 대해 각 하나씩)에 의해 결정된다. 이 코드에서는 theta 배열에 처음에 선언한 torsoId, headId,... rightLowerLegId까지 숫자를 0~9까지 부여하여 인덱스에 맞게 접근할 수 있도록 하였다.

```
var theta = [0, 0, 0, 0, 0, 0, 180, 0, 180, 0, 0];
function initNodes(Id) {

    var m = mat4();

    switch(Id) {

    case torsoId:

    m = rotate(theta[torsoId], 0, 1, 0 );
    figure[torsoId] = createNode( m, torso, null, headId );
    break;

    case headId:
    case head1Id:
    case head2Id:


    m = translate(0.0, torsoHeight+0.5*headHeight, 0.0);
```

```
    m = mult(m, rotate(theta[head1Id], 1, 0, 0))
    m = mult(m, rotate(theta[head2Id], 0, 1, 0));
    m = mult(m, translate(0.0, -0.5*headHeight, 0.0));
    figure[headId] = createNode( m, head, leftUpperArmId, null);
    break;


    case leftUpperArmId:

    m = translate(-(torsoWidth+upperArmWidth), 0.9*torsoHeight, 0.0);
    m = mult(m, rotate(theta[leftUpperArmId], 1, 0, 0));
    figure[leftUpperArmId] = createNode( m, leftUpperArm, rightUpperArmId,
leftLowerArmId );
    break;
```

위 트리를 화면에 나타나게 하려면 각 노드를 한 번씩 방문하는 그래프 순회가 필요하다. 그리고 트리와 순회 알고리즘을 표현할 자료구조가 필요하다. 그래서 Left-child right-sibling 구조를 사용하였다. 여기서 Left는 다음 노드를 의미하고, Right는 자식 노드들의 연결 리스트를 의미한다.

Left-child Right-sibling 트리



각 노드에는 1) Sibling에 대한 포인터, 2) Child에 대한 포인터, 3) 노드가 표현하는 객체를 그리는 함수에 대한 포인터, 4) 현재 모델-뷰 행렬의 우측에 곱해질 동차 좌표계 행렬이 필요하다.

우선 다음과 같이 트리 노드를 생성하였다.

```
function createNode(transform, render, sibling, child){
    var node = {
    transform: transform,
```

```
    render: render,
    sibling: sibling,
    child: child,
    }
    return node;
}
```

그리고 다음과 같이 노드를 초기화 하였다.

```
function initNodes(Id) {

    var m = mat4();

    switch(Id) {

    case torsoId:

    m = rotate(theta[torsoId], 0, 1, 0 );
    figure[torsoId] = createNode( m, torso, null, headId );
    break;

    case headId:
    case head1Id:
    case head2Id:


    m = translate(0.0, torsoHeight+0.5*headHeight, 0.0);
    m = mult(m, rotate(theta[head1Id], 1, 0, 0))
    m = mult(m, rotate(theta[head2Id], 0, 1, 0));
    m = mult(m, translate(0.0, -0.5*headHeight, 0.0));
    figure[headId] = createNode( m, head, leftUpperArmId, null);
    break;
```

rotate와 translate를 이용하여 행렬을 구성한다. 여기서는 각도와 재 디스플레이를 통해 애니메이션을 처리한다.

위 트리를 화면에 나타나게 하기 위해 각 노드를 한 번씩 방문하는 전위 순회를 하였다.

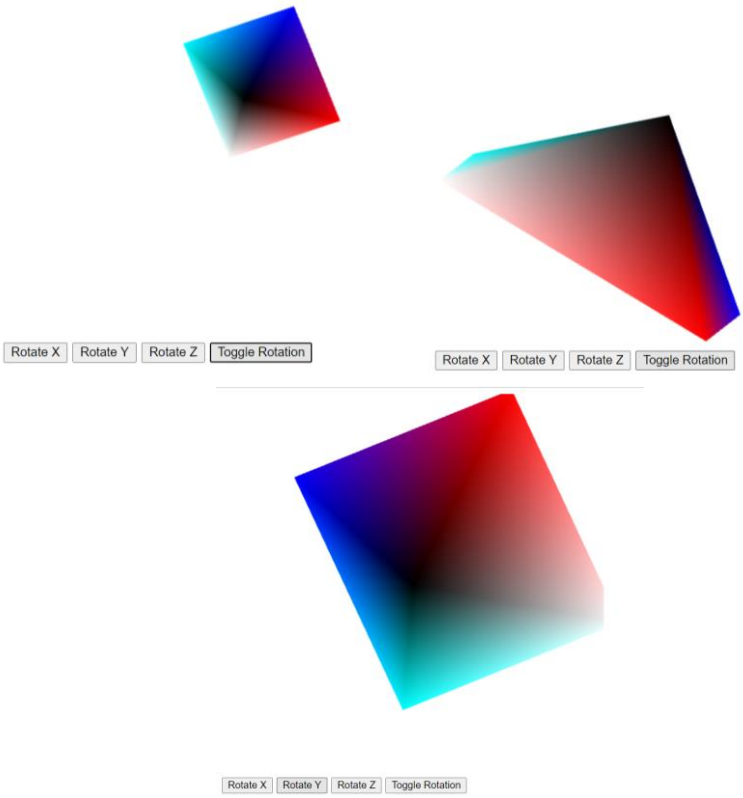
```
function traverse(Id) {

   if(Id == null) return;
   stack.push(modelViewMatrix);
   modelViewMatrix = mult(modelViewMatrix, figure[Id].transform);
   figure[Id].render();
   if(figure[Id].child != null) traverse(figure[Id].child);
```

```
    modelViewMatrix = stack.pop();
   if(figure[Id].sibling != null) traverse(figure[Id].sibling);
}
```

여기서 수정된 행렬은 Child Node에는 적용되지만 자체 행렬을 가지는 Sibling Node에는 적용되지 않으므로 modelViewMatrix를 stack에 push하였다.

2. 직접 구현한 코드: 역동적으로 움직이는 꼭두각시

앞에서 구현한 Head, Left-upper arm ... Right-upper leg와 같은 객체들이 각각 회전하도록 하여 좀 더 꼭두각시가 역동적으로 움직일 수 있도록 알고리즘을 수정하였다. 중간고사를 준비하면서 만들었던 회전하면서 점점 커지고 작아지고 반복하는 피라미드를 참고하였다.



일단 다음과 같이 axis, flag, dt, theta1, thetaLoc을 선언하였다. axis는 꼭두각시의 몸이 x축/y축/z축 어느 방향을 중심으로 회전하는지 결정하는 역할을 한다. flag는 true or false 값을 가져 회전을 멈출 수 있다. theta1은 회전하는 각을 저장한 것

으로, theta1 배열에 일정한 값을 더하면서 물체가 회전한다.

```
var axis = 0;
var theta1 = [0, 0, 0];

var thetaLoc;

var flag = true;
```

shader를 load하고 buffer의 초기 내용을 설정한다. 이는 인덱스를 GPU에 전달하는 역할을 한다.

```
vBuffer = gl.createBuffer();

    gl.bindBuffer( gl.ARRAY_BUFFER, vBuffer );
    gl.bufferData(gl.ARRAY_BUFFER, flatten(pointsArray), gl.STATIC_DRAW);

    var vPosition = gl.getAttribLocation( program, "vPosition" );
    gl.vertexAttribPointer( vPosition, 4, gl.FLOAT, false, 0, 0 );
    gl.enableVertexAttribArray( vPosition );
```

getUniformLocation 함수를 이용하여 회전에 따른 물체의 위치를 표현한다.

```
thetaLoc = gl.getUniformLocation(program, "theta");
```

그 다음 Rotate X, Y, Z 버튼을 누를 때마다 axis 값이 바뀌는 것을 이용하여 theta1의 값에 회전 속도를 더하고, Toggle Rotation을 누르며 flag 값이 true나 false로 바뀌는 것을 이용하여 회전을 멈춘다.

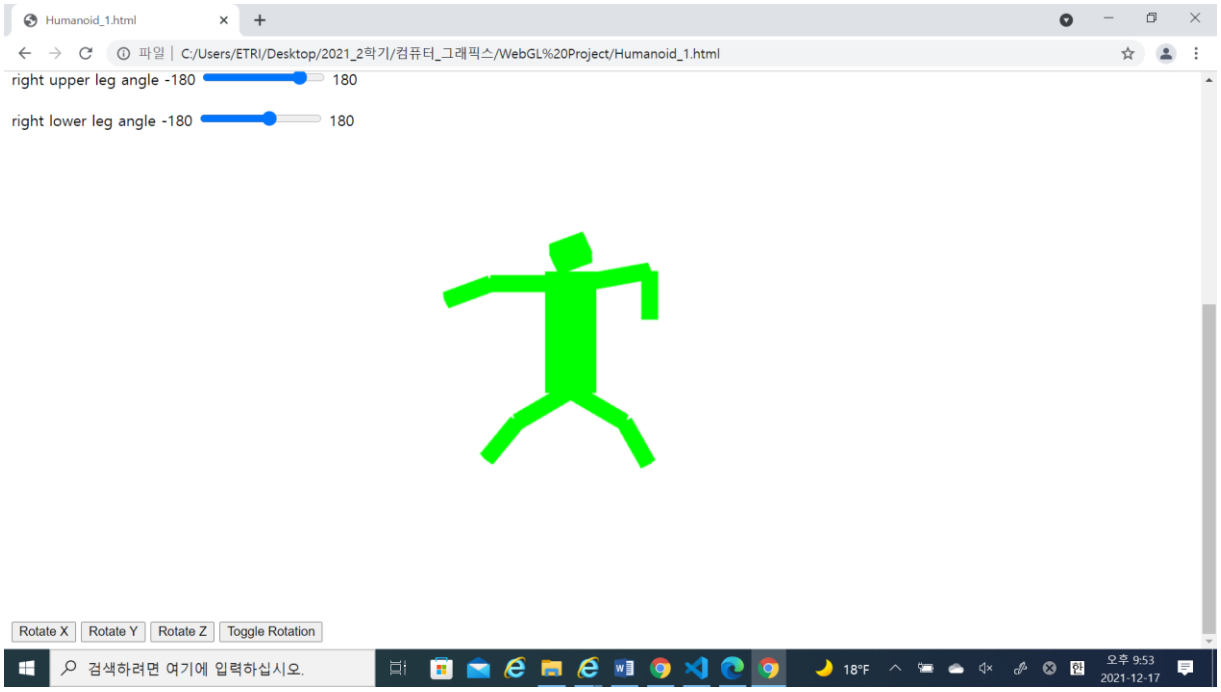
```
document.getElementById( "xButton" ).onclick = function () {
        axis = xAxis;
    };
    document.getElementById( "yButton" ).onclick = function () {
        axis = yAxis;
    };
    document.getElementById( "zButton" ).onclick = function () {
        axis = zAxis;
    };
    document.getElementById("ButtonT").onclick = function(){flag = !flag;};

    render();
```

```
var render = function() {

        gl.clear( gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
        traverse(torsoId);

    if(flag) theta1[axis] += 1;
// axis 에 따라서 x 축, y 축, z 축 방향으로 회전하는 게 정해진다/
    gl.uniform3fv(thetaLoc, theta1);
```

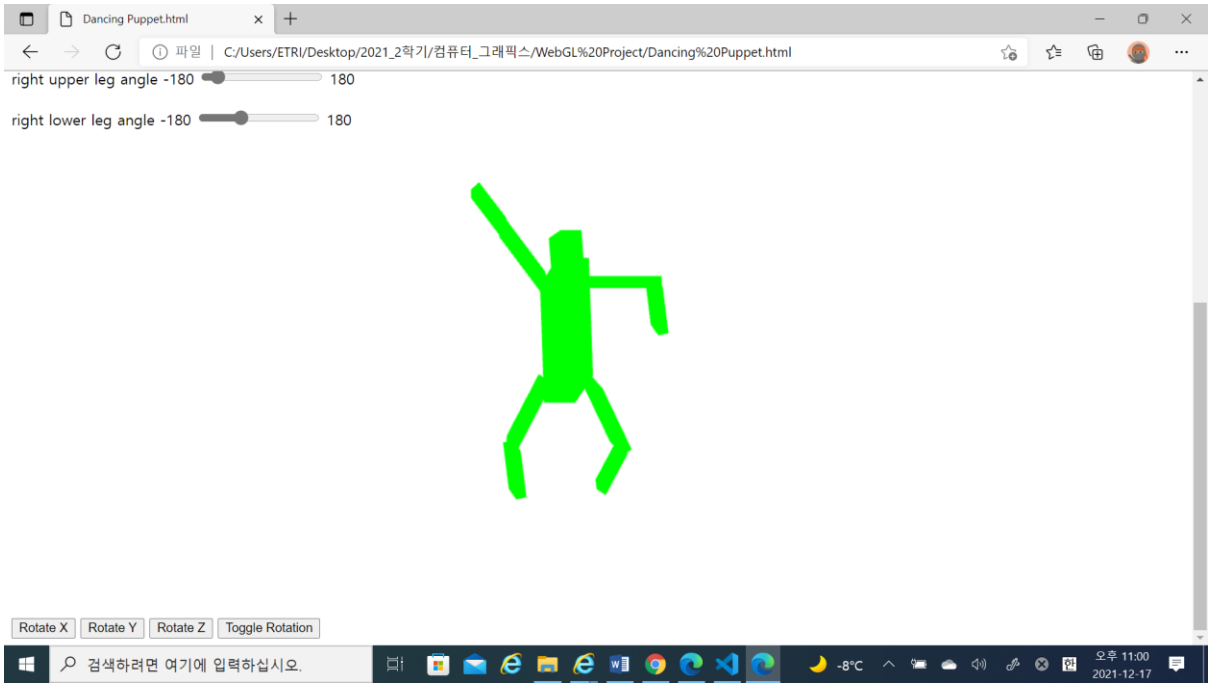
2. 브라우저 테스트 결과(Chrome, Edge)

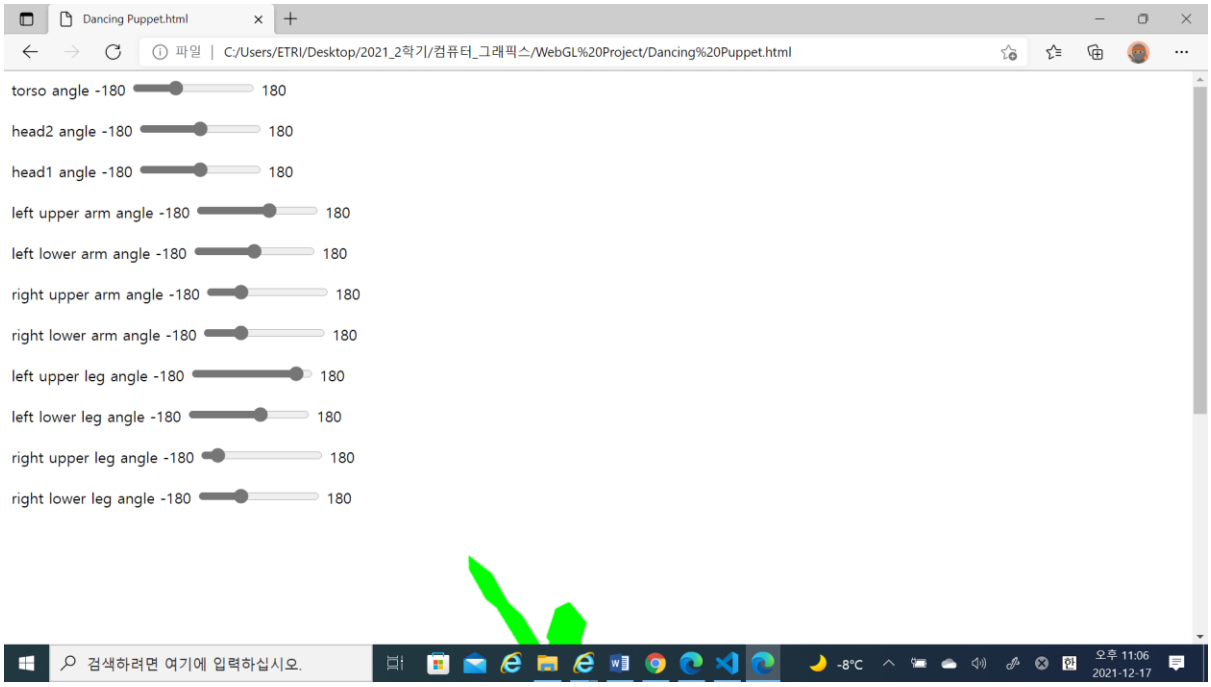
테스트 영상, https://youtu.be/s6d-paQYKS8

1) Chrome



2) Edge


Dancing Puppet.html
파일 | C:/Users/ETRI/Desktop/2021_2학기/컴퓨터_그래픽스/WebGL%20Project/Dancing%20Puppet.html
right upper leg angle -180 180
right lower leg angle -180 180
Rotate X
Rotate Y
Rotate Z
Toggle Rotation
검색하려면 여기에 입력하십시오.
-8°C
오후 11:00
2021-12-17


Dancing Puppet.html
파일 | C:/Users/ETRI/Desktop/2021_2학기/컴퓨터_그래픽스/WebGL%20Project/Dancing%20Puppet.html
torso angle -180 180
head2 angle -180 180
head1 angle -180 180
left upper arm angle -180 180
left lower arm angle -180 180
right upper arm angle -180 180
right lower arm angle -180 180
left upper leg angle -180 180
left lower leg angle -180 180
right upper leg angle -180 180
right lower leg angle -180 180
검색하려면 여기에 입력하십시오.
-8°C
오후 11:06
2021-12-17

III. 후기

평소에 듣던 음악에서 영감을 받아 webGL 프로젝트를 진행하였고, 춤추는 꼭두각시를 만들었다. 화면에 나타나게 하기 위해 노드를 한 번씩 방문하는 그래프 순회, Left-child right-sibling 구조를 이해하고 실제 webGL 코드로 구현하는 법, torso(), Head()와 같은 함수를 이용하여 각 부품에 접근하는 법, 그리고 행렬을 이용하여 Parent node에 따라 상대적으로 달라지는 Child Node의 위치를 표현하는 법, 물체를 회전시키는 법 등 다양한 알고리즘을 구현할 수 있었다. 이번 프로젝트와 컴퓨터 그래픽스 수업을 들으면서 오픈 소스의 중요성을 느꼈다. 대부분의 과제는 A부터 Z까지 전부 구현하는 것이 아니라, 70% 정도는 오픈 소스에서 가져오고, 나머지는 코드를 수정하고 추가하는 방식으로 진행하였다. 훨씬 효율적으로 알고리즘을 구현할 수 있었고, 다양한 코드를 접하면서 영감을 얻었다. 예를 들면, 광원의 위치가 달라지면서 물체에 가해지는 빛이 달라지는 애니메이션을 구현하는 과제가 인상적이었다. 최종무 교수님의 <시스템 프로그래밍> 수업에서 실제 회사에서는 직접 코드를 작성하는 일과 더불어 오픈 소스 코드를 찾고 그것을 수정하고 최적화하는 일을 많이 한다고 하셨다. 컴퓨터 그래픽스 과제를 하면서 검색하는 능력, 다른 사람의 코드를 이해하는 능력, 그것을 내가 원하는 대로 활용하는 능력을 기를 수 있었다. webGL에 대한 이해와 html, javascript에 대한 이해, 각종 알고리즘에 대한 이해는 덤이다. 데이터가 모두에게 자유롭게 흐르는, 오픈 소스를 활용하는 문화가 좀 더 활발해졌으면 좋겠다. 이를 위해 필자도 앞으로 코드를 구현할 때마다 github에 틈틈이 올릴 것을 다짐한다. 깊이 감사의 말씀을 드린다.

참고문헌

송인식, <컴퓨터 그래픽스 강의자료: 12. hierarchy>, 단국대학교, 2021

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/09/figure.html

https://www.cs.unm.edu/~angel/BOOK/INTERACTIVE_COMPUTER_GRAPHICS/SEVENTH_EDITION/CODE/04/cube.html