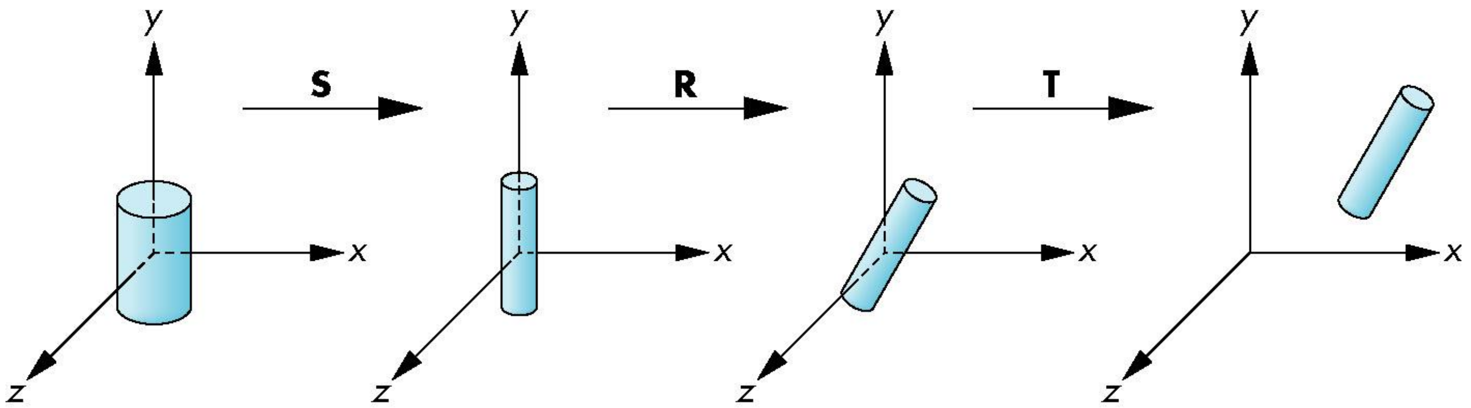
Hierarchical Modeling

14TH WEEK, 2022

Instance Transformation

- Start with a prototype object (a *symbol*)
- Each appearance of the object in the model is an *instance*
  - Must scale, orient, position
  - Defines instance transformation



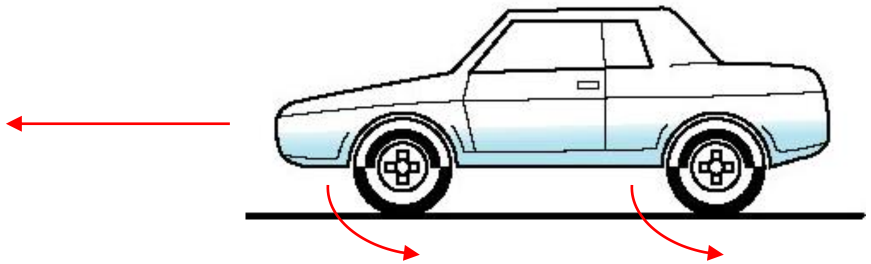
Symbol-Instance Table

- Can store a model by assigning a number to each symbol and storing the parameters for the instance transformation

Symbol	Scale	Rotate	Translate
1	s_x, s_y, s_z	$\theta_x, \theta_y, \theta_z$	d_x, d_y, d_z
2			
3			
1			
1			
.			
.			

Relationships in Car Model

- Symbol-instance table does not show relationships between parts of model
- Consider model of car
  - Chassis + 4 identical wheels
  - Two symbols



- Rate of forward motion determined by rotational speed of wheels

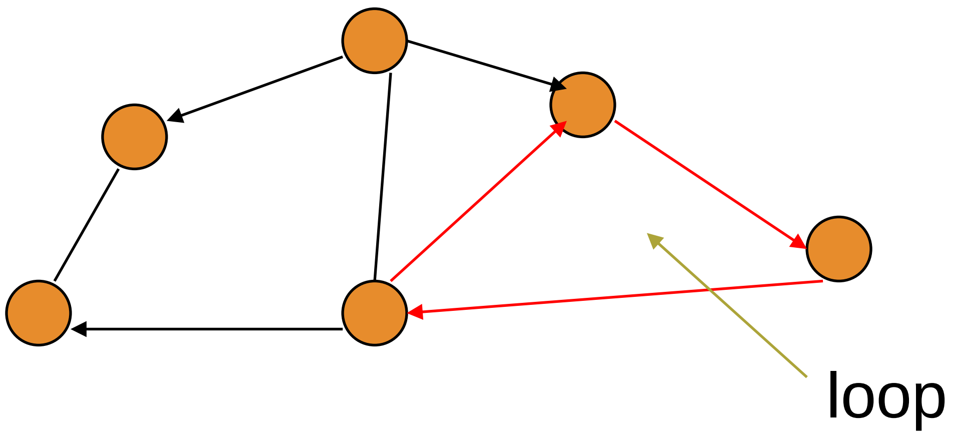
Structure Through Function Calls

```
car(speed)
{
    chassis()
    wheel(right_front);
    wheel(left_front);
    wheel(right_rear);
    wheel(left_rear);
}
```

- Fails to show relationships well
- Look at problem using a graph

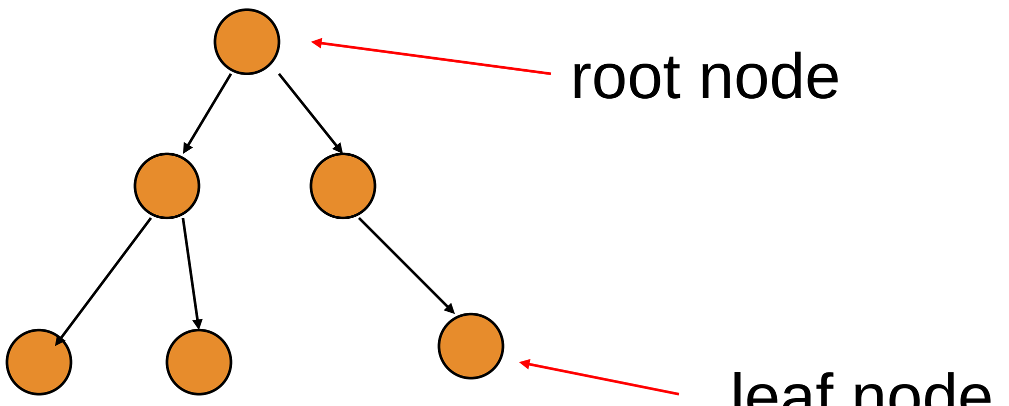
Graphs

- Set of *<u>nodes</u>* and *<u>edges</u>* (*links*)
- Each connects a pair of nodes
  - Directed or undirected
- Cycle: directed path that is a loop

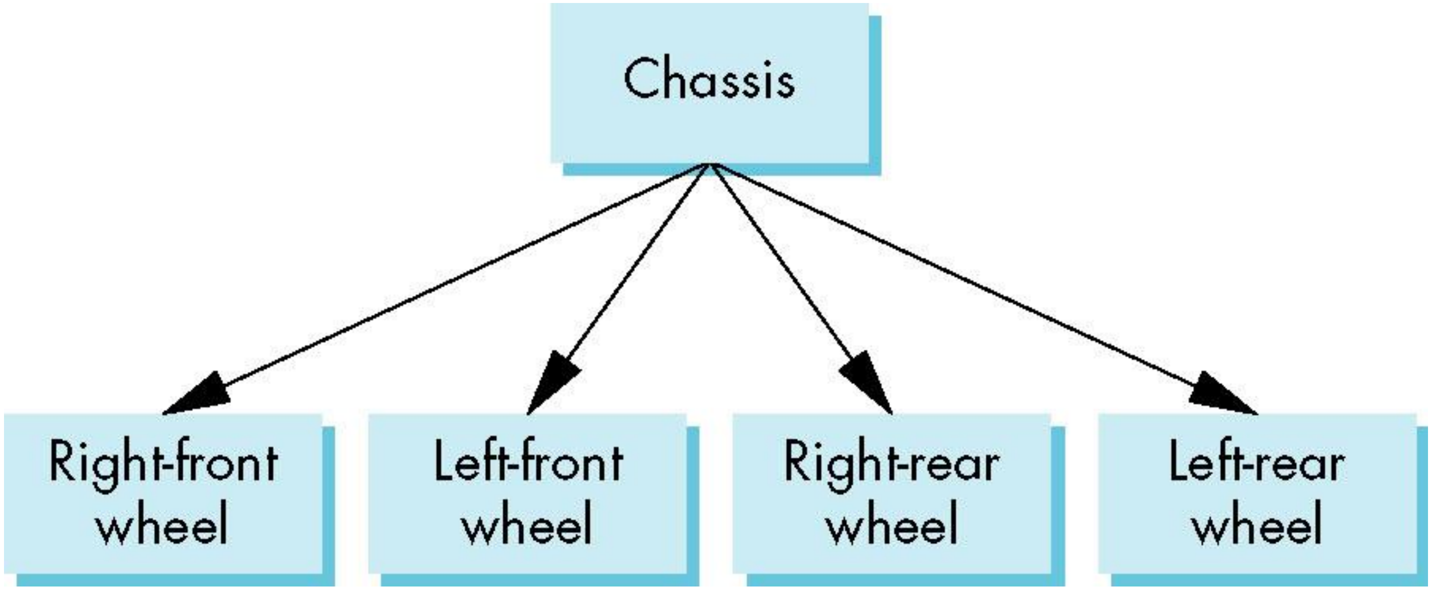


Trees

- Graph in which each node (except the root) has exactly one parent node
  - May have multiple children
  - Leaf or terminal node: no children

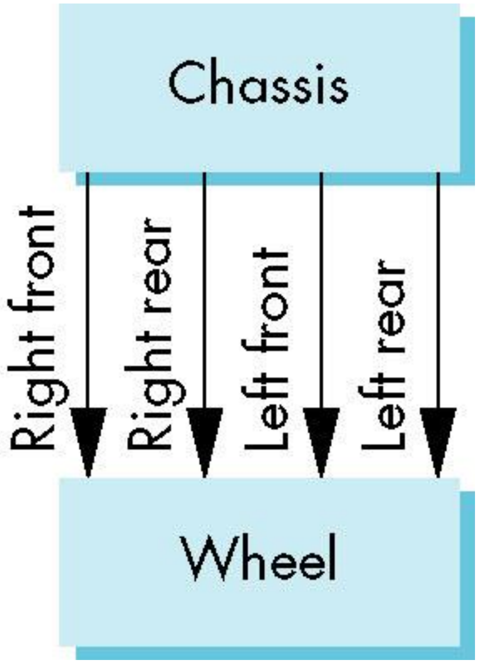


Tree Model of Car



DAG Model

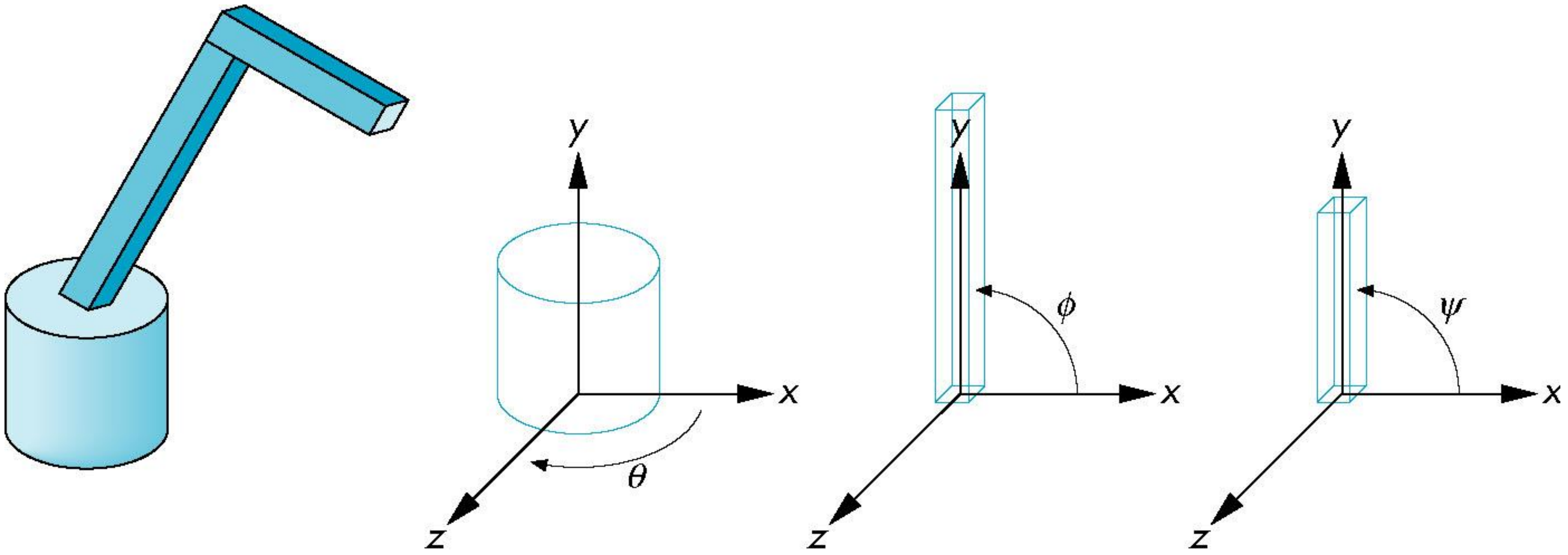
- If we use the fact that all wheels are identical, we get a directed acyclic graph
  - Not much different that dealing with a tree



Modeling with Trees

- Must decide what information to place in nodes and what to put in edges
- Nodes
  - What to draw
  - Pointers to children
- Edges
  - May have information on incremental changes to transformation matrices (can also store in nodes)

Robot Arm

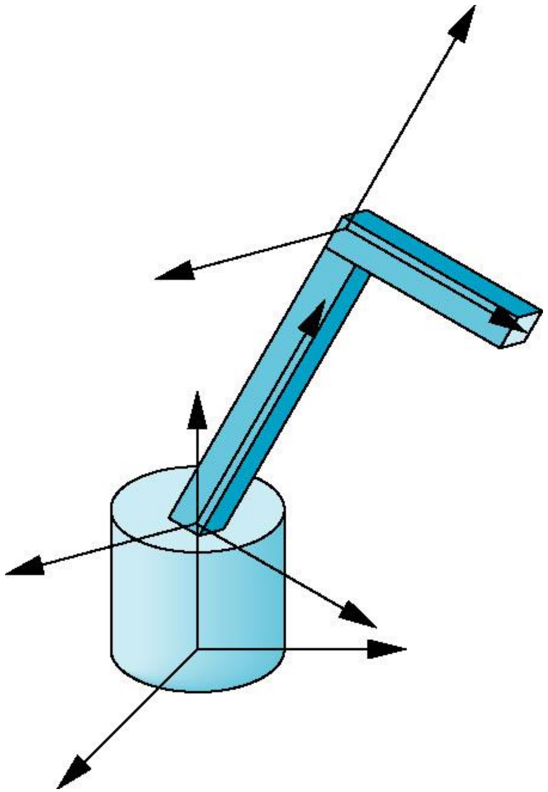


robot arm

parts in their own
coodinate systems

Articulated Models

- Robot arm is an example of an *articulated model*
  - Parts connected at joints
  - Can specify state of model by giving all joint angles



Relationships in Robot Arm

- Base rotate independently
  - Single angle determines position
- Lower arm attached to base
  - Its position depends on rotation of base
  - Must also translate relative to base and rotate about connecting joint
- Upper arm attached to lower arm
  - Its position depends on both base and lower arm
  - Must translate relative to lower arm and rotate about joint connecting to lower arm

Required Matrices

- Rotation of base: $\mathbf{R}_b$
  - Apply $\mathbf{M} = \mathbf{R}_b$ to base
- Translate lower arm relative to base: $\mathbf{T}_{lu}$
- Rotate lower arm around joint: $\mathbf{R}_{lu}$
  - Apply $\mathbf{M} = \mathbf{R}_b \mathbf{T}_{lu} \mathbf{R}_{lu}$ to lower arm
- Translate upper arm relative to lower arm: $\mathbf{T}_{uu}$
- Rotate upper arm around joint: $\mathbf{R}_{uu}$
  - Apply $\mathbf{M} = \mathbf{R}_b \mathbf{T}_{lu} \mathbf{R}_{lu} \mathbf{T}_{uu} \mathbf{R}_{uu}$ to upper arm

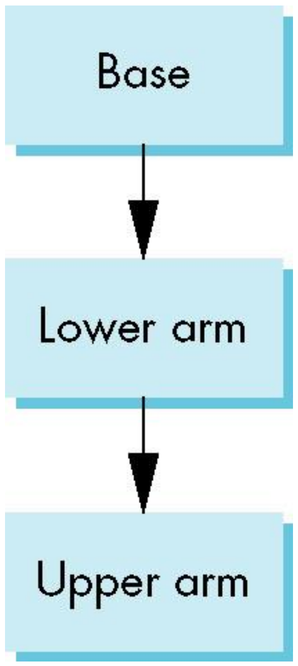
WebGL Code for Robot

```
var render = function() {
    gl.clear( gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT );

    modelViewMatrix = rotate(theta[Base], 0, 1, 0 );
    base();

    modelViewMatrix = mult(modelViewMatrix, translate(0.0, BASE_HEIGHT, 0.0));
    modelViewMatrix = mult(modelViewMatrix, rotate(theta[LowerArm], 0, 0, 1 ));
    lowerArm();

    modelViewMatrix  = mult(modelViewMatrix, translate(0.0, LOWER_ARM_HEIGHT, 0.0));
    modelViewMatrix  = mult(modelViewMatrix, rotate(theta[UpperArm], 0, 0, 1) );
    upperArm();

    requestAnimFrame(render);
}
```

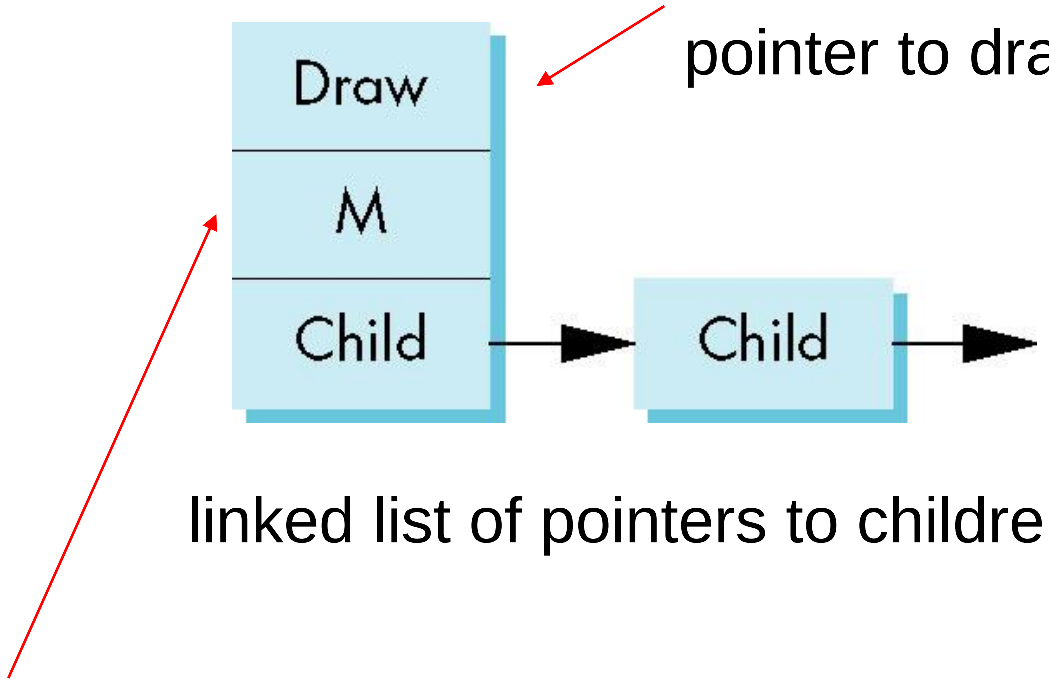
Tree Model of Robot

- Note code shows relationships between parts of model
    - Can change "look" of parts easily without altering relationships

- Simple example of tree model
- Want a general node structure for nodes



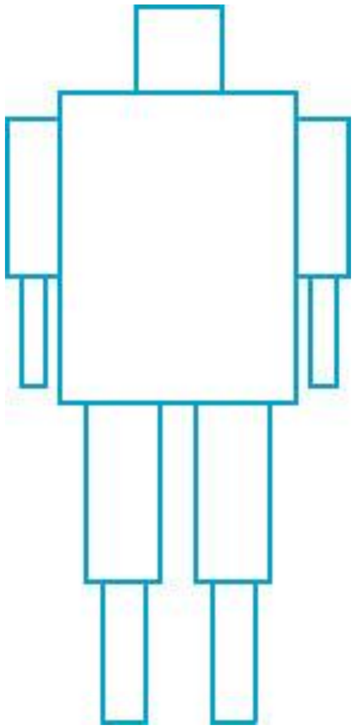
Possible Node Structure

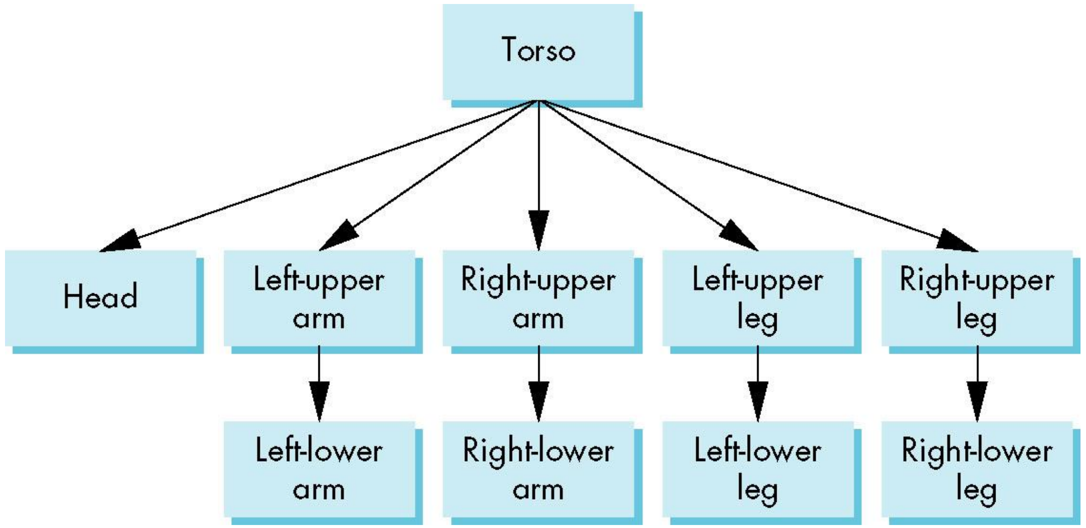


Generalizations

- Need to deal with multiple children
  - How do we represent a more general tree?
  - How do we traverse such a data structure?
- Animation
  - How to use dynamically?
  - Can we create and delete nodes during execution?

Humanoid Figure

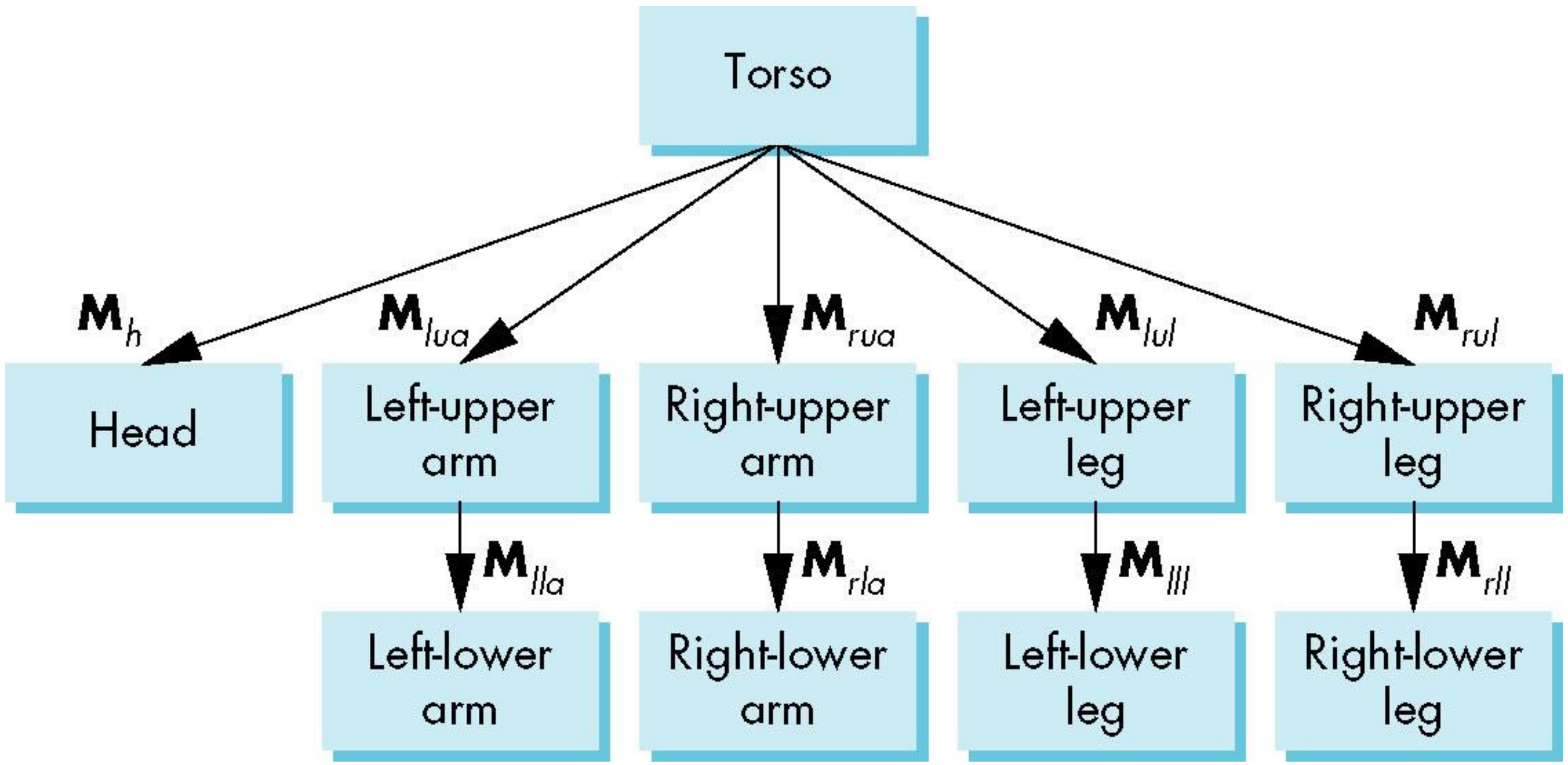




Building the Model

- Can build a simple implementation using quadrics: ellipsoids and cylinders
- Access parts through functions
  - **`torso()`**
  - **`leftUpperArm()`**
- Matrices describe position of node with respect to its parent
  - $\mathbf{M}_{\mathrm{lla}}$ positions left lower leg with respect to left upper arm

Tree with Matrices



Display and Traversal

- The position of the figure is determined by 11 joint angles (two for the head and one for each other part)
- Display of the tree requires a *graph traversal*
  - Visit each node once
  - Display function at each node that describes the part associated with the node, applying the correct transformation matrix for position and orientation

Transformation Matrices

- There are 10 relevant matrices
  - $\mathbf{M}$ positions and orients entire figure through the torso which is the root node
  - $\mathbf{M}_{\mathrm{h}}$ positions head with respect to torso
  - $\mathbf{M}_{\mathrm{lua}}$, $\mathbf{M}_{\mathrm{rua}}$, $\mathbf{M}_{\mathrm{lul}}$, $\mathbf{M}_{\mathrm{rul}}$ position arms and legs with respect to torso
  - $\mathbf{M}_{\mathrm{lla}}$, $\mathbf{M}_{\mathrm{rla}}$, $\mathbf{M}_{\mathrm{lll}}$, $\mathbf{M}_{\mathrm{rll}}$ position lower parts of limbs with respect to corresponding upper limbs

Stack-based Traversal

- Set model-view matrix to $\mathbf{M}$ and draw torso
- Set model-view matrix to $\mathbf{MM}_{\mathrm{h}}$ and draw head
- For left-upper arm need $\mathbf{MM}_{\mathrm{lua}}$ and so on
- Rather than recomputing $\mathbf{MM}_{\mathrm{lua}}$ from scratch or using an inverse matrix, we can use the matrix stack to store $\mathbf{M}$ and other matrices as we traverse the tree

Traversal Code

```
figure() {
    PushMatrix()
    torso();
    Rotate  (…);
    head();
    PopMatrix();
    PushMatrix();
    Translate(…);
    Rotate(…);
    left_upper_arm();
    PopMatrix();
    PushMatrix();
```

save present model-view matrix

update model-view matrix for head

recover original model-view matrix

save it again

update model-view matrix
for left upper arm

recover and save original
model-view matrix again

rest of code

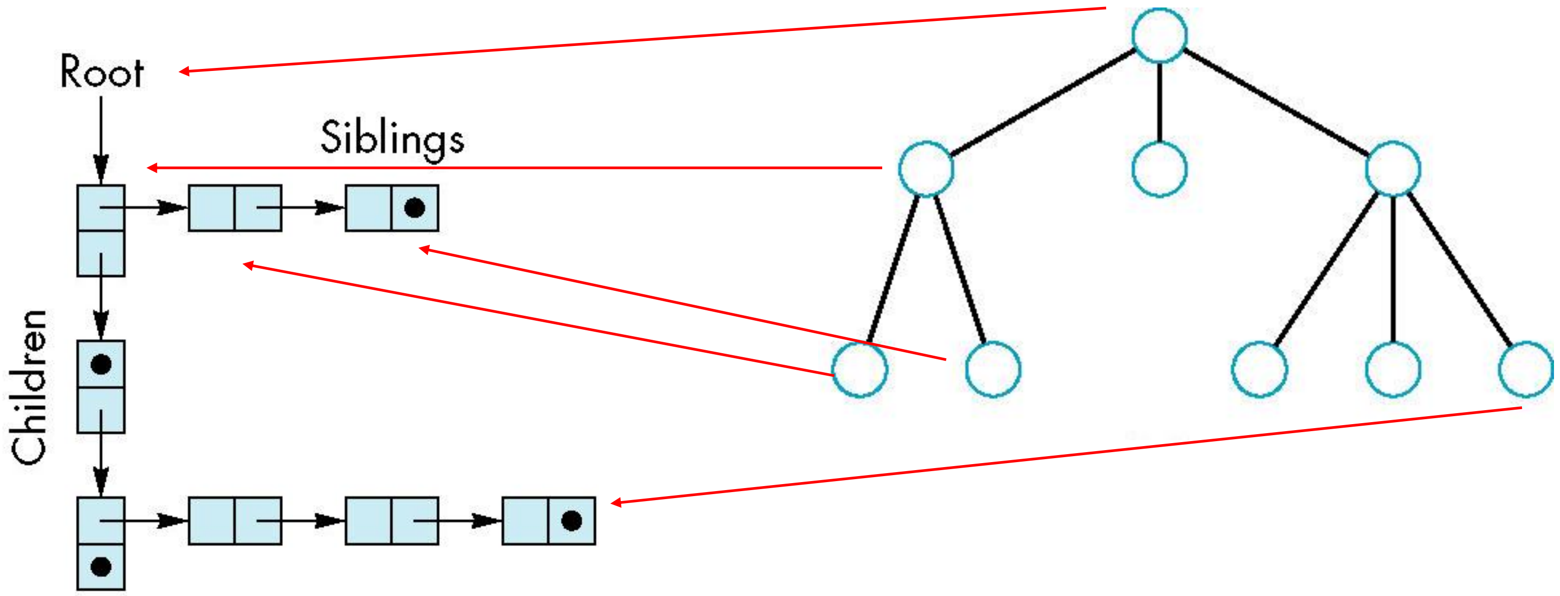
Analysis

- The code describes a particular tree and a particular traversal strategy
  - Can we develop a more general approach?
- Note that the sample code does not include state changes, such as changes to colors
  - May also want to push and pop other attributes to protect against unexpected state changes affecting later parts of the code

General Tree Data Structure

- Need a data structure to represent tree and an algorithm to traverse the tree
- We will use a *left-child right-sibling* structure
  - Uses linked lists
  - Each node in data structure is two pointers
  - Left: next node
  - Right: linked list of children

Left-Child Right-Sibling Tree



Tree Node Structure

- At each node we need to store
  - Pointer to sibling
  - Pointer to child
  - Pointer to a function that draws the object represented by the node
  - Homogeneous coordinate matrix to multiply on the right of the current model-view matrix
    - Represents changes going from parent to node
    - In WebGL this matrix is a 1D array storing matrix by columns

Creating a Tree Node

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

Initializing Nodes

```
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 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;
```

Notes

- The position of figure is determined by 11 joint angles stored in **`theta[11]`**
- Animate by changing the angles and redisplaying
- We form the required matrices using rotate and translate
- Because the matrix is formed using the model-view matrix, we may want to first push original model-view matrix on matrix stack

Preorder Traversal

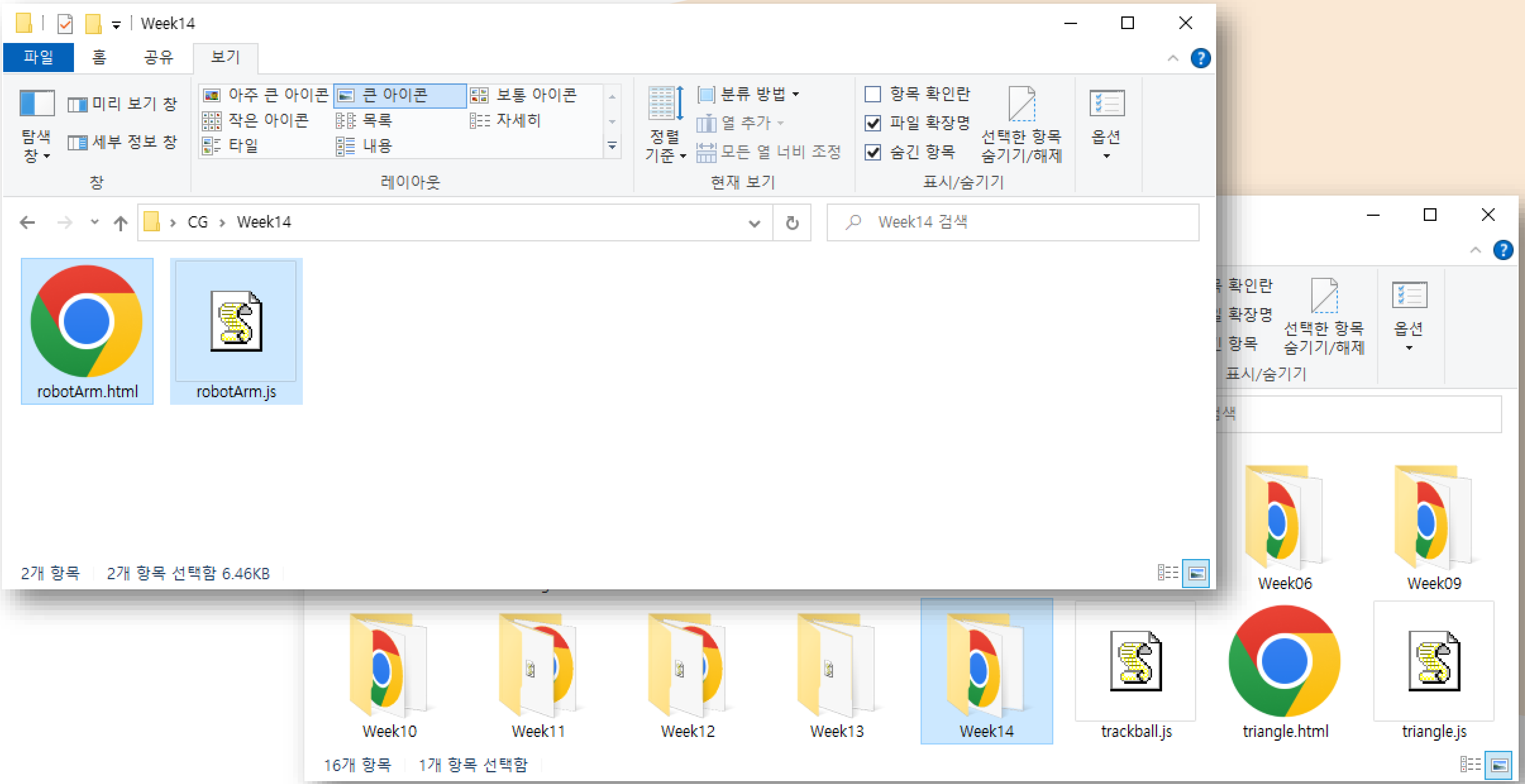
```
function traverse(Id) {
     if(Id == null) return;
     stack.push(modelViewMatrix);
     modelViewMatrix = mult(modelViewMatrix, figure[Id].transform);
     figure[Id].render();
     if(figure[Id].child != null) traverse(figure[Id].child);
     modelViewMatrix = stack.pop();
     if(figure[Id].sibling != null) traverse(figure[Id].sibling);
}
var render = function() {
    gl.clear( gl.COLOR_BUFFER_BIT );
    traverse(torsoId);
    requestAnimFrame(render);
}
```

Notes

- We must save model-view matrix before multiplying it by node matrix
  - Updated matrix applies to children of node but not to siblings which contain their own matrices
- The traversal program applies to any left-child right-sibling tree
  - The particular tree is encoded in the definition of the individual nodes
- The order of traversal matters because of possible state changes in the functions

Dynamic Trees

- Because we are using JS, the nodes and the node structure can be changed during execution
- Definition of nodes and traversal are essentially the same as before but we can add and delete nodes during execution
- In desktop OpenGL, if we use pointers, the structure can be dynamic


Week14
파일
홈
공유
보기
미리 보기 창
세부 정보 창
탐색 창
창
아주 큰 아이콘
큰 아이콘
보통 아이콘
작은 아이콘
목록
자세히
타일
내용
레이아웃
정렬 기준
분류 방법
열 추가
모든 열 너비 조정
현재 보기
항목 확인란
파일 확장명
숨긴 항목
선택한 항목 숨기기/해제
표시/숨기기
옵션
CG › Week14
Week14 검색
robotArm.html
robotArm.js
2개 항목
2개 항목 선택함 6.46KB
Week06
Week09
Week10
Week11
Week12
Week13
Week14
trackball.js
triangle.html
triangle.js
16개 항목
1개 항목 선택함

File Edit Selection View Go Run Terminal Help robotArm.html - Visual Studio Code

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

robotArm.html robotArm.js

C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > robotArm.html > ...

```html
<!DOCTYPE html>
<html>
    <head>
        <title>학번 이름 - Robot Arm</title>
        <script id="vertex-shader" type="x-shader/x-vertex">
        attribute vec4 vPosition;
        attribute vec4 vColor;
        varying vec4 fColor;

        uniform mat4 modelMatrix, viewMatrix, projectionMatrix;

        void main() {
            gl_Position = gl_Position = projectionMatrix * viewMatrix * modelMatrix * vPosition;
            fColor = vColor;
        }
        </script>

        <script id="fragment-shader" type="x-shader/x-fragment">
        precision mediump float;
        varying vec4 fColor;

        void main() {
            gl_FragColor = fColor;
        }
        </script>

        <script type="text/javascript" src="../Common/webgl-utils.js"></script>
        <script type="text/javascript" src="../Common/initShaders.js"></script>
        <script type="text/javascript" src="../Common/MV.js"></script>
        <script type="text/javascript" src="../trackball.js"></script>
        <script type="text/javascript" src="robotArm.js"></script>
    </head>
    <body>
        <div style="width:512px; text-align:center;">
            <button id="left">◀</button>
```

Restricted Mode ⊗ 0 ⚠ 0 Ln 1, Col 1 Spaces: 4 UTF-8 CRLF HTML

```html

        <script type="text/javascript" src="../Common/webgl-utils.js"></script>
        <script type="text/javascript" src="../Common/initShaders.js"></script>
        <script type="text/javascript" src="../Common/MV.js"></script>
        <script type="text/javascript" src="../trackball.js"></script>
        <script type="text/javascript" src="robotArm.js"></script>
    </head>
    <body>
        <div style="width:512px; text-align:center;">
            <button id="left">◀</button>
            <button id="up">▲</button>
            <button id="right">▶</button><br>
            <button id="down">▼</button>
        </div>
        <canvas id="gl-canvas" width="512" height="512">
            Oops... your browser doesn't support the HTML5 canvas element!
        </canvas>
    </body>
</html>
```

```
var gl;
var points = [];
var colors = [];

var modelMatrixLoc;

var trballMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
var vertCubeStart, numVertCubeTri, vertGroundStart, numVertGroundTri, numVertGroundLine;

// Parameters controlling the size of the Robot's arm
const BASE_HEIGHT      = 1.0;
const BASE_WIDTH       = 5.0;
const LOWER_ARM_HEIGHT = 5.0;
const LOWER_ARM_WIDTH  = 0.5;
const UPPER_ARM_HEIGHT = 5.0;
const UPPER_ARM_WIDTH  = 0.5;

// Array of rotation angles (in degrees) for each rotation axis
const Base = 0;
const LowerArm = 1;
const UpperArm = 2;

var theta= [0, 0, 0];

window.onload = function init()
{
    var canvas = document.getElementById("gl-canvas");

    gl = WebGLUtils.setupWebGL(canvas);
    if( !gl ) {
        alert("WebGL isn't available!");
    }

    generateGround(10);
    generateCube();
```

robotArm.js - Visual Studio Code

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

robotArm.html | robotArm.js

C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > JS robotArm.js > ...

```javascript

    // virtual trackball
    var trball = trackball(canvas.width, canvas.height);
    var mouseDown = false;

    canvas.addEventListener("mousedown", function (event) {
        trball.start(event.clientX, event.clientY);
        mouseDown = true;
    });

    canvas.addEventListener("mouseup", function (event) {
        mouseDown = false;
    });

    canvas.addEventListener("mousemove", function (event) {
        if (mouseDown) {
            trball.end(event.clientX, event.clientY);
            trballMatrix = mat4(trball.rotationMatrix);
        }
    });

    // Configure WebGL
    gl.viewport(0, 0, canvas.width, canvas.height);
    gl.clearColor(0.0, 0.0, 0.0, 1.0);

    // Enable hidden-surface removal
    gl.enable(gl.DEPTH_TEST);

    gl.enable(gl.POLYGON_OFFSET_FILL);
    gl.polygonOffset(0.01, 1);

    // Load shaders and initialize attribute buffers
    var program = initShaders(gl, "vertex-shader", "fragment-shader");
    gl.useProgram(program);

```

Restricted Mode ⊗ 0 ⚠ 0 Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} JavaScript

robotArm.js - Visual Studio Code

```
    // Load the data into the GPU
    var bufferId = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);

    // Associate our shader variables with our data buffer
    var vPosition = gl.getAttribLocation(program0, "vPosition");
    gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vPosition);

    // Create a buffer object, initialize it, and associate it with
    // the associated attribute variable in our vertex shader
    var cBufferId = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, cBufferId);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(colors), gl.STATIC_DRAW);

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

    var modelMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
    modelMatrixLoc = gl.getUniformLocation(program, "modelMatrix");
    gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(modelMatrix));

    var viewMatrix = lookAt(vec3(0.0, 3.0, 10.0), vec3(0.0, 3.0, 0.0), vec3(0.0, 1.0, 0.0));
    var viewMatrixLoc = gl.getUniformLocation(program, "viewMatrix");
    gl.uniformMatrix4fv(viewMatrixLoc, false, flatten(viewMatrix));
    /*
    // 3D orthographic viewing
    var viewLength = 2.0;
    var projectionMatrix;
    if (canvas.width > canvas.height) {
        var aspect = viewLength * canvas.width / canvas.height;
        projectionMatrix = ortho(-aspect, aspect, -viewLength, viewLength, -viewLength, 1000);
    }
```

robotArm.js - Visual Studio Code

```
    else {
        var aspect = viewLength * canvas.height / canvas.width;
        projectionMatrix = ortho(-viewLength, viewLength, -aspect, aspect, -viewLength, 1000);
    }
    */
    // 3D perspective viewing
    var aspect = canvas.width / canvas.height;
    var projectionMatrix = perspective(90, aspect, 0.1, 1000);

    var projectionMatrixLoc = gl.getUniformLocation(program, "projectionMatrix");
    gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));

    // Event listeners for buttons
    document.getElementById("left").onclick = function () {

    };
    document.getElementById("right").onclick = function () {

    };
    document.getElementById("up").onclick = function () {

    };
    document.getElementById("down").onclick = function () {

    };

    render();
};

window.onkeydown = function(event) {
    switch (event.keyCode) {
        case 37:    // left arrow
        case 65:    // 'A'
        case 97:    // 'a'
```

```javascript

            break;
        case 39:    // right arrow
        case 68:    // 'D'
        case 100:   // 'd'

            break;
        case 38:    // up arrow
        case 87:    // 'W'
        case 119:   // 'w'

            break;
        case 40:    // down arrow
        case 83:    // 'S'
        case 115:   // 's'

            break;
    }
    render();
};

var render = function() {
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    // draw the ground
    gl.drawArrays(gl.TRIANGLES, vertGroundStart, numVertGroundTri);
    gl.drawArrays(gl.LINES, vertGroundStart+numVertGroundTri, numVertGroundLine);

    // draw a cube
    gl.drawArrays(gl.TRIANGLES, vertCubeStart, numVertCubeTri);

    window.requestAnimationFrame(render);
}

function generateCube() {
```

robotArm.js - Visual Studio Code

C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > robotArm.js > ...

```javascript

function generateCube() {
    vertCubeStart = points.length;
    numVertCubeTri = 0;
    quad(1, 0, 3, 2);
    quad(2, 3, 7, 6);
    quad(3, 0, 4, 7);
    quad(4, 5, 6, 7);
    quad(5, 4, 0, 1);
    quad(6, 5, 1, 2);
}

function quad(a, b, c, d) {
    vertexPos = [
        vec4(-0.5, -0.5, -0.5, 1.0),
        vec4( 0.5, -0.5, -0.5, 1.0),
        vec4( 0.5,  0.5, -0.5, 1.0),
        vec4(-0.5,  0.5, -0.5, 1.0),
        vec4(-0.5, -0.5,  0.5, 1.0),
        vec4( 0.5, -0.5,  0.5, 1.0),
        vec4( 0.5,  0.5,  0.5, 1.0),
        vec4(-0.5,  0.5,  0.5, 1.0)
    ];

    vertexColor = [
        vec4(0.0, 0.0, 0.0, 1.0),   // black
        vec4(1.0, 0.0, 0.0, 1.0),   // red
        vec4(1.0, 1.0, 0.0, 1.0),   // yellow
        vec4(0.0, 1.0, 0.0, 1.0),   // green
        vec4(0.0, 0.0, 1.0, 1.0),   // blue
        vec4(1.0, 0.0, 1.0, 1.0),   // magenta
        vec4(1.0, 1.0, 1.0, 1.0),   // white
        vec4(0.0, 1.0, 1.0, 1.0)    // cyan
    ];

```

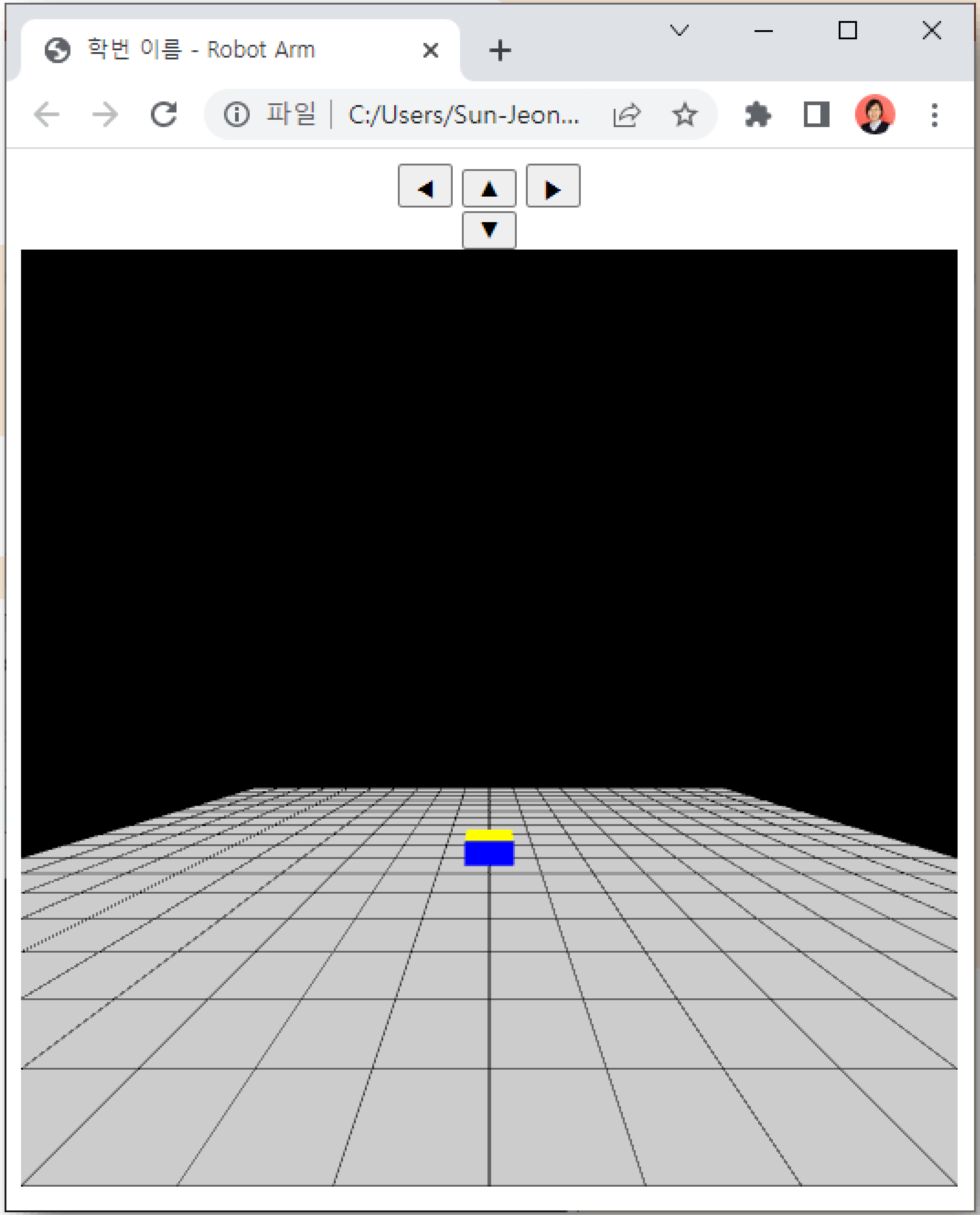
robotArm.js - Visual Studio Code

```javascript
        // We need to partition the quad into two triangles in order for WebGL
        // to be able to render it. In this case, we create two triangles from
        // the quad indices.
        var index = [ a, b, c, a, c, d ];
        for(var i=0; i<index.length; i++) {
            points.push(vertexPos[index[i]]);
            colors.push(vertexColor[a]);
            //colors.push(vertexColor[index[i]]);
            numVertCubeTri++;
        }
    }

    function generateGround(scale) {
        vertGroundStart = points.length;
        numVertGroundTri = 0;
        var color = vec4(0.8, 0.8, 0.8, 1.0);
        for(var x=-scale; x<scale; x++) {
            for(var z=-scale; z<scale; z++) {
                // two triangles
                points.push(vec4(x, 0.0, z, 1.0));
                colors.push(color);
                numVertGroundTri++;

                points.push(vec4(x, 0.0, z+1, 1.0));
                colors.push(color);
                numVertGroundTri++;

                points.push(vec4(x+1, 0.0, z+1, 1.0));
                colors.push(color);
                numVertGroundTri++;

                points.push(vec4(x, 0.0, z, 1.0));
                colors.push(color);
                numVertGroundTri++;
```

robotArm.js - Visual Studio Code

C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > robotArm.js > ...

```javascript

            points.push(vec4(x+1, 0.0, z+1, 1.0));
            colors.push(color);
            numVertGroundTri++;

            points.push(vec4(x+1, 0.0, z, 1.0));
            colors.push(color);
            numVertGroundTri++;
        }
    }

    // grid lines
    numVertGroundLine = 0;
    color = vec4(0.0, 0.0, 0.0, 1.0);
    for(var x=-scale; x<=scale; x++) {
        points.push(vec4(x, 0.0, -scale, 1.0));
        colors.push(color);
        numVertGroundLine++;

        points.push(vec4(x, 0.0, scale, 1.0));
        colors.push(color);
        numVertGroundLine++;
    }
    for(var z=-scale; z<=scale; z++) {
        points.push(vec4(-scale, 0.0, z, 1.0));
        colors.push(color);
        numVertGroundLine++;

        points.push(vec4(scale, 0.0, z, 1.0));
        colors.push(color);
        numVertGroundLine++;
    }
}

```


학번 이름 - Robot Arm
파일 | C:/Users/Sun-Jeon...

File Edit Selection View Go Run Terminal Help robotArm.js - Visual Studio Code

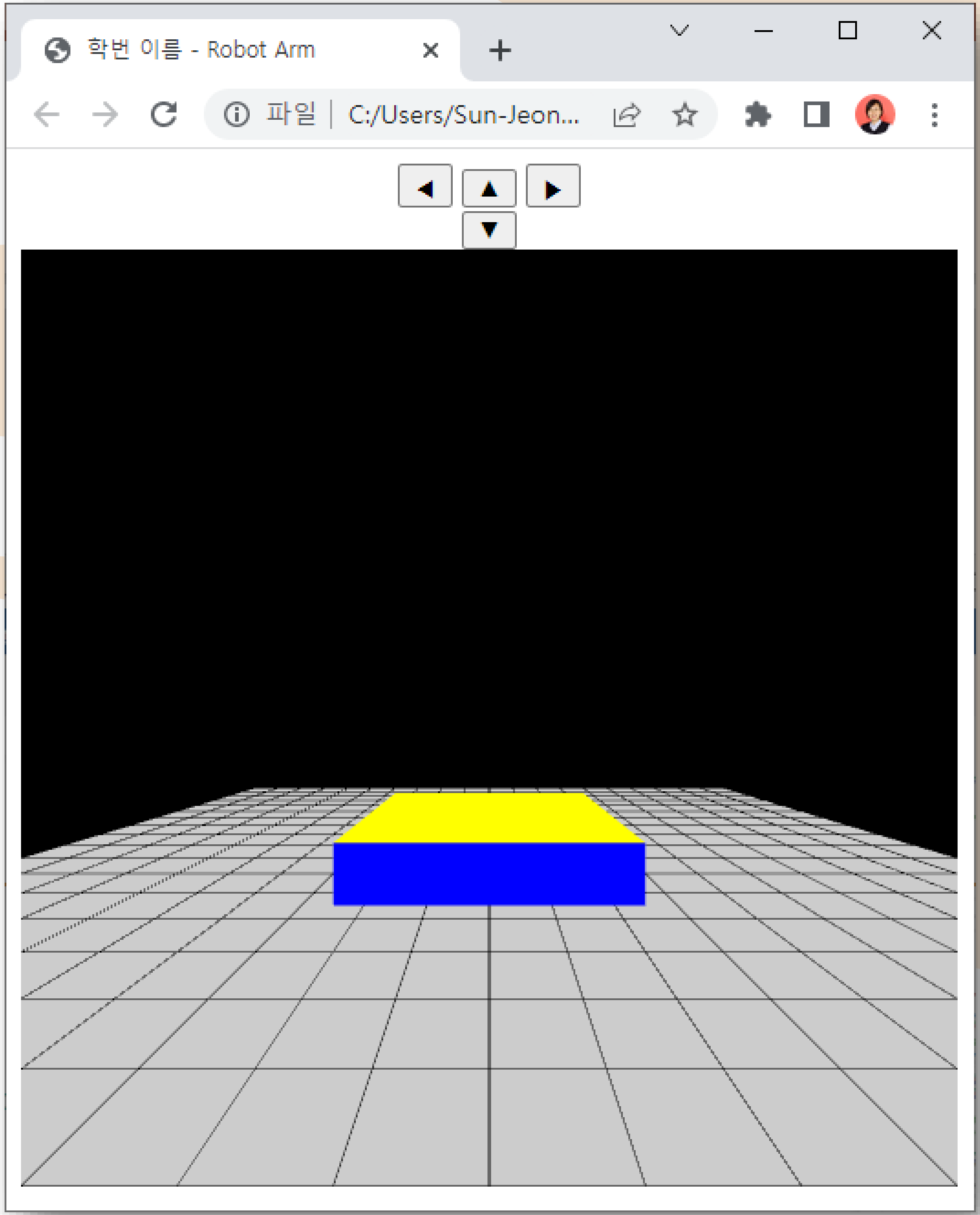
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

robotArm.html | robotArm.js

C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > robotArm.js > base

```javascript

var render = function() {
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    var modelMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);

    // draw the ground
    ground(modelMatrix);

    // draw a cube
    //gl.drawArrays(gl.TRIANGLES, vertCubeStart, numVertCubeTri);

    // draw the robot arm
    base(modelMatrix);

    window.requestAnimationFrame(render);
}

function ground(modelMatrix) {
    gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(modelMatrix));
    gl.drawArrays(gl.TRIANGLES, vertGroundStart, numVertGroundTri);
    gl.drawArrays(gl.LINES, vertGroundStart+numVertGroundTri, numVertGroundLine);
}

function base(modelMatrix) {
    var sMatrix = scalem(BASE_WIDTH, BASE_HEIGHT, BASE_WIDTH);
    var tMatrix = mult(translate(0.0, 0.5*BASE_HEIGHT, 0.0), sMatrix);
    var instanceMatrix = mult(modelMatrix, tMatrix);
    gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(instanceMatrix));
    gl.drawArrays(gl.TRIANGLES, vertCubeStart, numVertCubeTri);
}

function generateCube() {
    vertCubeStart = points.length;
    numVertCubeTri = 0;
```

Restricted Mode ⊗ 0 ⚠ 0 Ln 189, Col 64 Spaces: 4 UTF-8 CRLF {} JavaScript


학번 이름 - Robot Arm
파일 | C:/Users/Sun-Jeon...

robotArm.js - Visual Studio Code

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

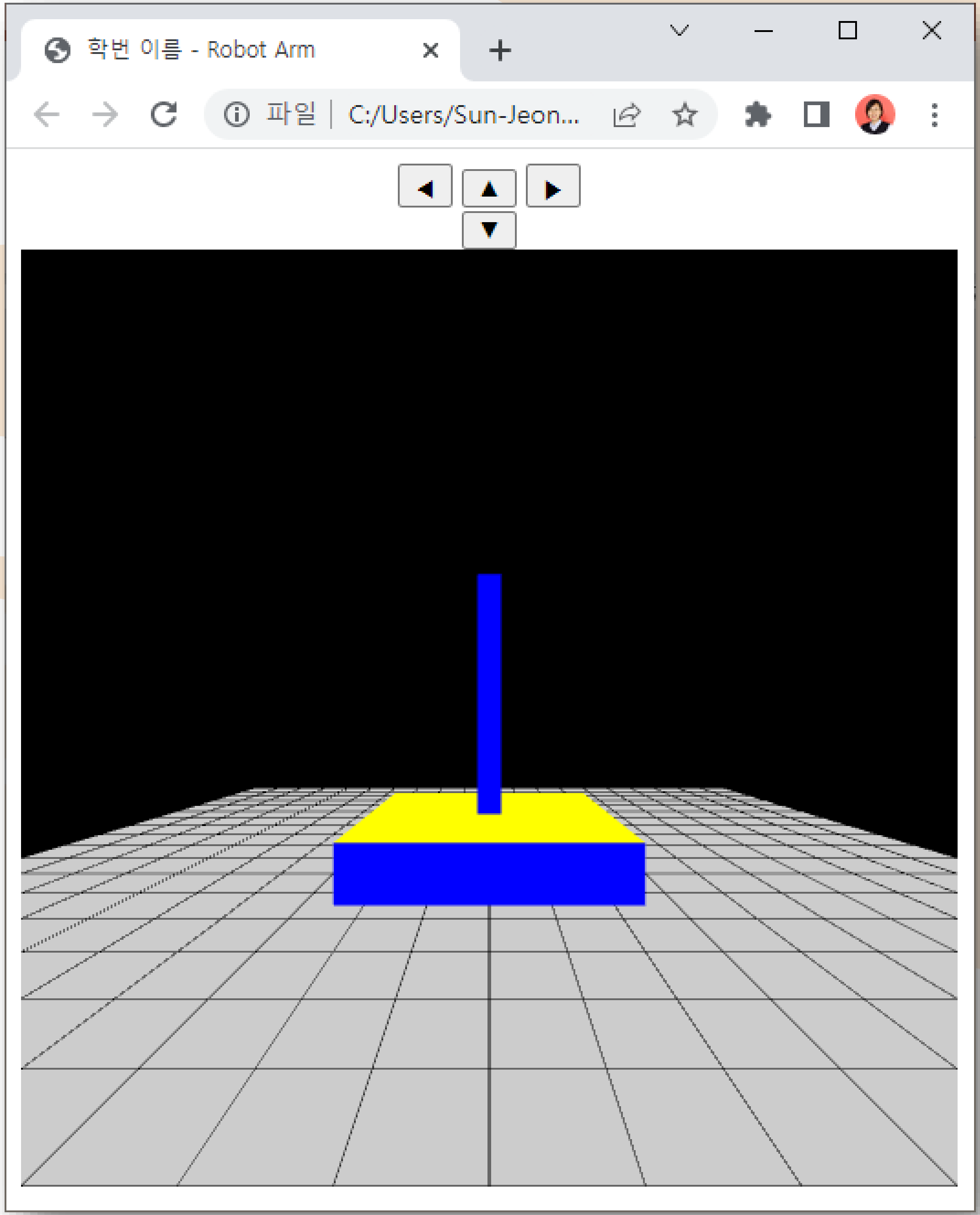
robotArm.html | robotArm.js

C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > robotArm.js > lowerArm

```javascript

var render = function() {
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    var modelMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);

    // draw the ground
    ground(modelMatrix);

    // draw a cube
    //gl.drawArrays(gl.TRIANGLES, vertCubeStart, numVertCubeTri);

    // draw the robot arm
    base(modelMatrix);

    modelMatrix = mult(modelMatrix, translate(0.0, BASE_HEIGHT, 0.0));
    lowerArm(modelMatrix);

    window.requestAnimationFrame(render);
}

function ground(modelMatrix) {
    gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(modelMatrix));
    gl.drawArrays(gl.TRIANGLES, vertGroundStart, numVertGroundTri);
    gl.drawArrays(gl.LINES, vertGroundStart+numVertGroundTri, numVertGroundLine);
}

function base(modelMatrix) {
    var sMatrix = scalem(BASE_WIDTH, BASE_HEIGHT, BASE_WIDTH);
    var tMatrix = mult(translate(0.0, 0.5*BASE_HEIGHT, 0.0), sMatrix);
    var instanceMatrix = mult(modelMatrix, tMatrix);
    gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(instanceMatrix));
    gl.drawArrays(gl.TRIANGLES, vertCubeStart, numVertCubeTri);
}

```

Restricted Mode ⓧ 0 ⚠ 0 Ln 200, Col 64 Spaces: 4 UTF-8 CRLF {} JavaScript

```javascript

function ground(modelMatrix) {
    gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(modelMatrix));
    gl.drawArrays(gl.TRIANGLES, vertGroundStart, numVertGroundTri);
    gl.drawArrays(gl.LINES, vertGroundStart+numVertGroundTri, numVertGroundLine);
}

function base(modelMatrix) {
    var sMatrix = scalem(BASE_WIDTH, BASE_HEIGHT, BASE_WIDTH);
    var tMatrix = mult(translate(0.0, 0.5*BASE_HEIGHT, 0.0), sMatrix);
    var instanceMatrix = mult(modelMatrix, tMatrix);
    gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(instanceMatrix));
    gl.drawArrays(gl.TRIANGLES, vertCubeStart, numVertCubeTri);
}

function lowerArm(modelMatrix) {
    var sMatrix = scalem(LOWER_ARM_WIDTH, LOWER_ARM_HEIGHT, LOWER_ARM_WIDTH);
    var tMatrix = mult(translate(0.0, 0.5*LOWER_ARM_HEIGHT, 0.0), sMatrix);
    var instanceMatrix = mult(modelMatrix, tMatrix);
    gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(instanceMatrix));
    gl.drawArrays(gl.TRIANGLES, vertCubeStart, numVertCubeTri);
}

function generateCube() {
    vertCubeStart = points.length;
    numVertCubeTri = 0;
    quad(1, 0, 3, 2);
    quad(2, 3, 7, 6);
    quad(3, 0, 4, 7);
    quad(4, 5, 6, 7);
    quad(5, 4, 0, 1);
    quad(6, 5, 1, 2);
}

function quad(a, b, c, d) {
```


학번 이름 - Robot Arm
파일 | C:/Users/Sun-Jeon...

File Edit Selection View Go Run Terminal Help robotArm.js - Visual Studio Code

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

robotArm.html | robotArm.js

C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > robotArm.js > upperArm

```
var render = function() {
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    var modelMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);

    // draw the ground
    ground(modelMatrix);

    // draw a cube
    //gl.drawArrays(gl.TRIANGLES, vertCubeStart, numVertCubeTri);

    // draw the robot arm
    base(modelMatrix);

    modelMatrix = mult(modelMatrix, translate(0.0, BASE_HEIGHT, 0.0));
    lowerArm(modelMatrix);

    modelMatrix = mult(modelMatrix, translate(0.0, LOWER_ARM_HEIGHT, 0.0));
    upperArm(modelMatrix);

    window.requestAnimationFrame(render);
}

function ground(modelMatrix) {
    gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(modelMatrix));
    gl.drawArrays(gl.TRIANGLES, vertGroundStart, numVertGroundTri);
    gl.drawArrays(gl.LINES, vertGroundStart+numVertGroundTri, numVertGroundLine);
}

function base(modelMatrix) {
    var sMatrix = scalem(BASE_WIDTH, BASE_HEIGHT, BASE_WIDTH);
    var tMatrix = mult(translate(0.0, 0.5*BASE_HEIGHT, 0.0), sMatrix);
    var instanceMatrix = mult(modelMatrix, tMatrix);
    gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(instanceMatrix));
```

Restricted Mode ⊗ 0 ⚠ 0 Ln 211, Col 64 Spaces: 4 UTF-8 CRLF {} JavaScript

File Edit Selection View Go Run Terminal Help   robotArm.js - Visual Studio Code

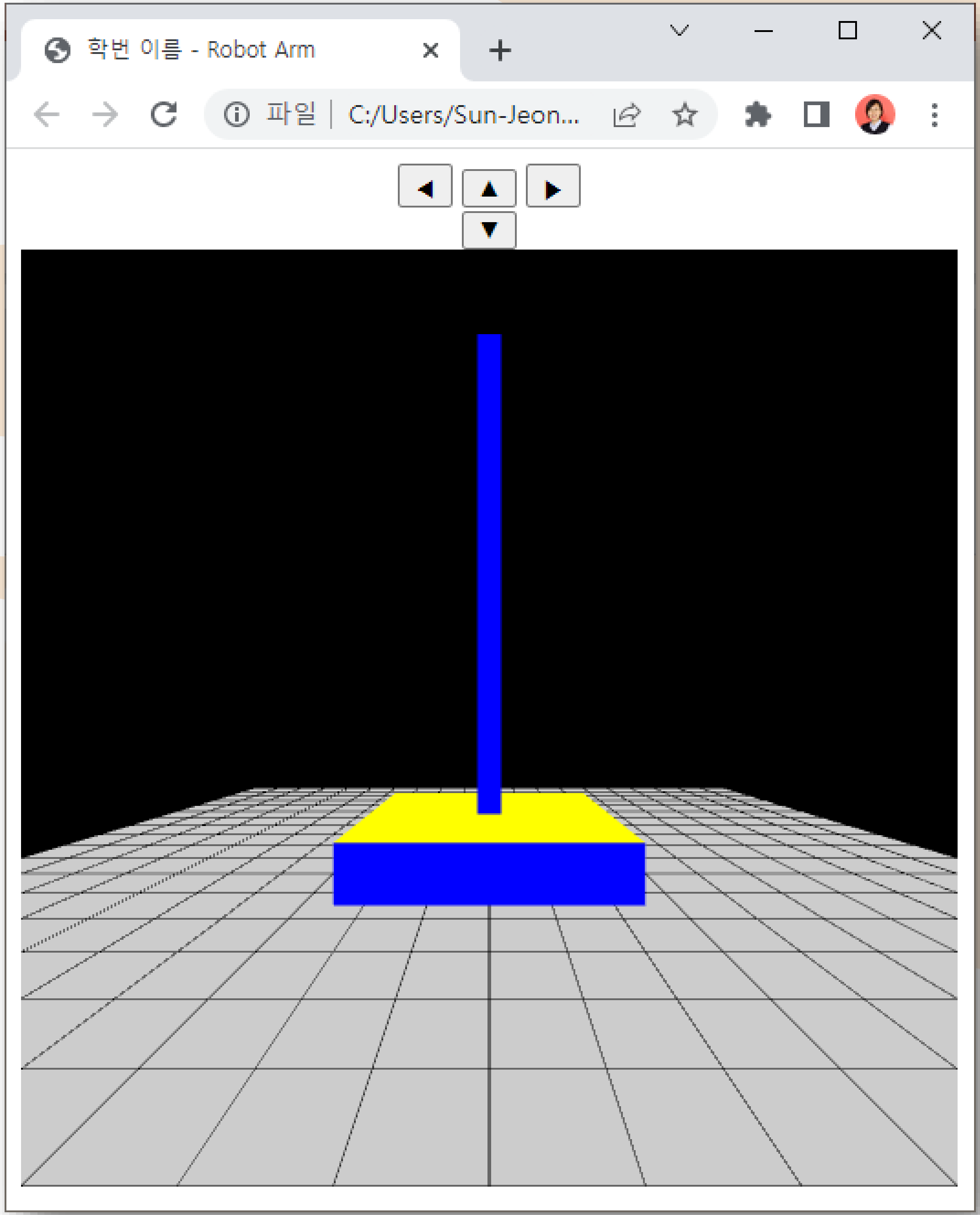
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

robotArm.html | robotArm.js

C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > robotArm.js > upperArm

```javascript

function ground(modelMatrix) {
    gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(modelMatrix));
    gl.drawArrays(gl.TRIANGLES, vertGroundStart, numVertGroundTri);
    gl.drawArrays(gl.LINES, vertGroundStart+numVertGroundTri, numVertGroundLine);
}

function base(modelMatrix) {
    var sMatrix = scalem(BASE_WIDTH, BASE_HEIGHT, BASE_WIDTH);
    var tMatrix = mult(translate(0.0, 0.5*BASE_HEIGHT, 0.0), sMatrix);
    var instanceMatrix = mult(modelMatrix, tMatrix);
    gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(instanceMatrix));
    gl.drawArrays(gl.TRIANGLES, vertCubeStart, numVertCubeTri);
}

function lowerArm(modelMatrix) {
    var sMatrix = scalem(LOWER_ARM_WIDTH, LOWER_ARM_HEIGHT, LOWER_ARM_WIDTH);
    var tMatrix = mult(translate(0.0, 0.5*LOWER_ARM_HEIGHT, 0.0), sMatrix);
    var instanceMatrix = mult(modelMatrix, tMatrix);
    gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(instanceMatrix));
    gl.drawArrays(gl.TRIANGLES, vertCubeStart, numVertCubeTri);
}

function upperArm(modelMatrix) {
    var sMatrix = scalem(UPPER_ARM_WIDTH, UPPER_ARM_HEIGHT, UPPER_ARM_WIDTH);
    var tMatrix = mult(translate(0.0, 0.5*UPPER_ARM_HEIGHT, 0.0), sMatrix);
    var instanceMatrix = mult(modelMatrix, tMatrix);
    gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(instanceMatrix));
    gl.drawArrays(gl.TRIANGLES, vertCubeStart, numVertCubeTri);
}

function generateCube() {
    vertCubeStart = points.length;
    numVertCubeTri = 0;
    quad(1, 0, 3, 2);
```

Restricted Mode | 0 0 | Ln 211, Col 64 | Spaces: 4 | UTF-8 | CRLF | JavaScript


학번 이름 - Robot Arm
파일
C:/Users/Sun-Jeon...

```javascript
window.onkeydown = function(event) {
    switch (event.keyCode) {
        case 65:    // 'A'
        case 97:    // 'a'
            theta[Base] -= 2.0;
            break;
        case 68:    // 'D'
        case 100:   // 'd'
            theta[Base] += 2.0;
            break;
        case 37:    // left arrow
            break;
        case 39:    // right arrow
            break;
        case 38:    // up arrow
        case 87:    // 'W'
        case 119:   // 'w'

            break;
        case 40:    // down arrow
        case 83:    // 'S'
        case 115:   // 's'

            break;
    }
    render();
};

var render = function() {
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    var modelMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);

    // draw the ground
    ground(modelMatrix);
```

robotArm.js - Visual Studio Code

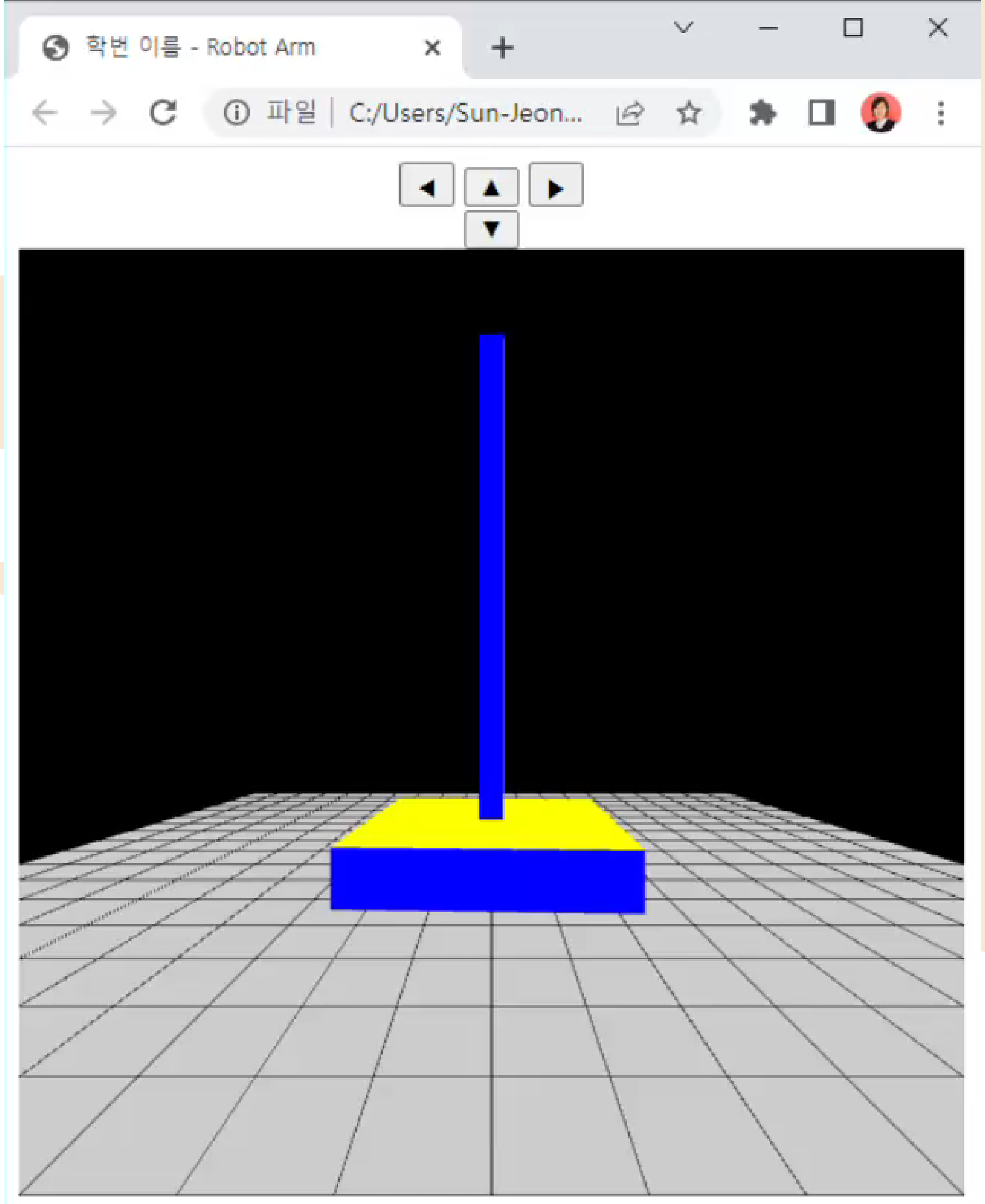
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

robotArm.html | robotArm.js

C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > robotArm.js > render

```javascript

var render = function() {
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    var modelMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);

    // draw the ground
    ground(modelMatrix);

    // draw a cube
    //gl.drawArrays(gl.TRIANGLES, vertCubeStart, numVertCubeTri);

    // draw the robot arm
    modelMatrix = mult(modelMatrix, rotate(theta[Base], 0, 1, 0 ));
    base(modelMatrix);

    modelMatrix = mult(modelMatrix, translate(0.0, BASE_HEIGHT, 0.0));
    lowerArm(modelMatrix);

    modelMatrix = mult(modelMatrix, translate(0.0, LOWER_ARM_HEIGHT, 0.0));
    upperArm(modelMatrix);

    window.requestAnimationFrame(render);
}

function ground(modelMatrix) {
    gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(modelMatrix));
    gl.drawArrays(gl.TRIANGLES, vertGroundStart, numVertGroundTri);
    gl.drawArrays(gl.LINES, vertGroundStart+numVertGroundTri, numVertGroundLine);
}

function base(modelMatrix) {
    var sMatrix = scalem(BASE_WIDTH, BASE_HEIGHT, BASE_WIDTH);
    var tMatrix = mult(translate(0.0, 0.5*BASE_HEIGHT, 0.0), sMatrix);
    var instanceMatrix = mult(modelMatrix, tMatrix);
```

Restricted Mode | 0 0 | Ln 175, Col 68 | Spaces: 4 | UTF-8 | CRLF | JavaScript


학번 이름 - Robot Arm
파일
C:/Users/Sun-Jeon...

robotArm.js - Visual Studio Code

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

robotArm.html | robotArm.js

C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > robotArm.js > render

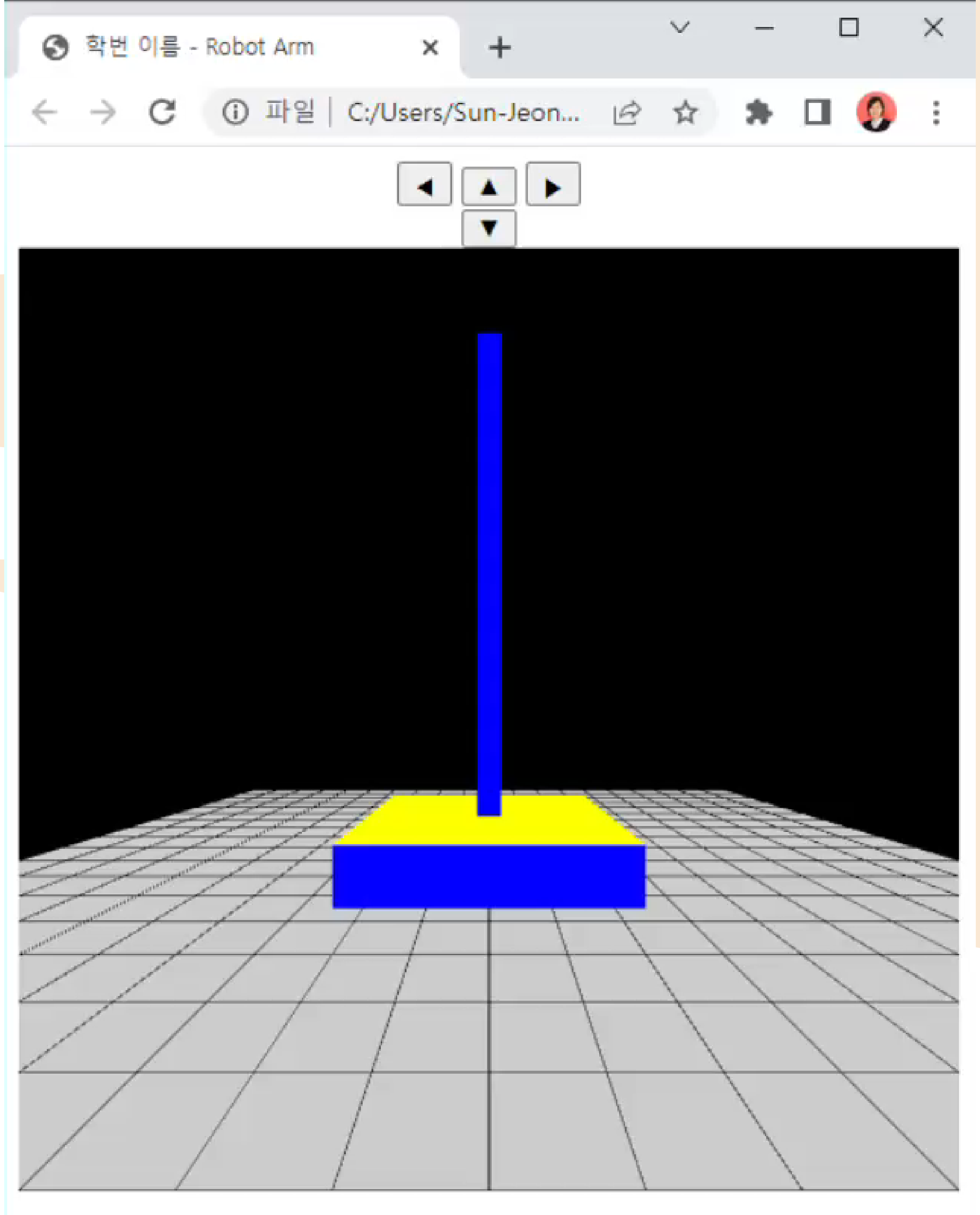
```javascript

    // Event listeners for buttons
    document.getElementById("left").onclick = function () {
        theta[LowerArm] += 2.0;
    };
    document.getElementById("right").onclick = function () {
        theta[LowerArm] -= 2.0;
    };
    document.getElementById("up").onclick = function () {

    };
    document.getElementById("down").onclick = function () {

    };

    render();
};

window.onkeydown = function(event) {
    switch (event.keyCode) {
        case 65:    // 'A'
        case 97:    // 'a'
            theta[Base] -= 2.0;
            break;
        case 68:    // 'D'
        case 100:   // 'd'
            theta[Base] += 2.0;
            break;
        case 37:    // left arrow
            theta[LowerArm] += 2.0;
            break;
        case 39:    // right arrow
            theta[LowerArm] -= 2.0;
            break;
        case 38:    // up arrow
```

Restricted Mode | 0 0 | Ln 181, Col 72 | Spaces: 4 | UTF-8 | CRLF | JavaScript

robotArm.js - Visual Studio Code

C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > robotArm.js > render

```javascript

var render = function() {
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    var modelMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);

    // draw the ground
    ground(modelMatrix);

    // draw a cube
    //gl.drawArrays(gl.TRIANGLES, vertCubeStart, numVertCubeTri);

    // draw the robot arm
    modelMatrix = mult(modelMatrix, rotate(theta[Base], 0, 1, 0 ));
    base(modelMatrix);

    modelMatrix = mult(modelMatrix, translate(0.0, BASE_HEIGHT, 0.0));
    modelMatrix = mult(modelMatrix, rotate(theta[LowerArm], 0, 0, 1 ));
    lowerArm(modelMatrix);

    modelMatrix = mult(modelMatrix, translate(0.0, LOWER_ARM_HEIGHT, 0.0));
    upperArm(modelMatrix);

    window.requestAnimationFrame(render);
}

function ground(modelMatrix) {
    gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(modelMatrix));
    gl.drawArrays(gl.TRIANGLES, vertGroundStart, numVertGroundTri);
    gl.drawArrays(gl.LINES, vertGroundStart+numVertGroundTri, numVertGroundLine);
}

function base(modelMatrix) {
    var sMatrix = scalem(BASE_WIDTH, BASE_HEIGHT, BASE_WIDTH);
    var tMatrix = mult(translate(0.0, 0.5*BASE_HEIGHT, 0.0), sMatrix);
```


학번 이름 - Robot Arm
파일 | C:/Users/Sun-Jeon...

File Edit Selection View Go Run Terminal Help robotArm.js - Visual Studio Code

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

robotArm.html robotArm.js

C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > robotArm.js > render

```javascript
    document.getElementById("up").onclick = function () {
        theta[UpperArm] += 2.0;
    };
    document.getElementById("down").onclick = function () {
        theta[UpperArm] -= 2.0;
    };

    render();
};

window.onkeydown = function(event) {
    switch (event.keyCode) {
        case 65:    // 'A'
        case 97:    // 'a'
            theta[Base] -= 2.0;
            break;
        case 68:    // 'D'
        case 100:   // 'd'
            theta[Base] += 2.0;
            break;
        case 37:    // left arrow
            theta[LowerArm] += 2.0;
            break;
        case 39:    // right arrow
            theta[LowerArm] -= 2.0;
            break;
        case 38:    // up arrow
            theta[UpperArm] += 2.0;
            break;
        case 40:    // down arrow
            theta[UpperArm] -= 2.0;
            break;
        case 87:    // 'W'
        case 119:   // 'w'
            break;
```

Restricted Mode 0 0 Ln 187, Col 68 Spaces: 4 UTF-8 CRLF JavaScript

robotArm.js - Visual Studio Code

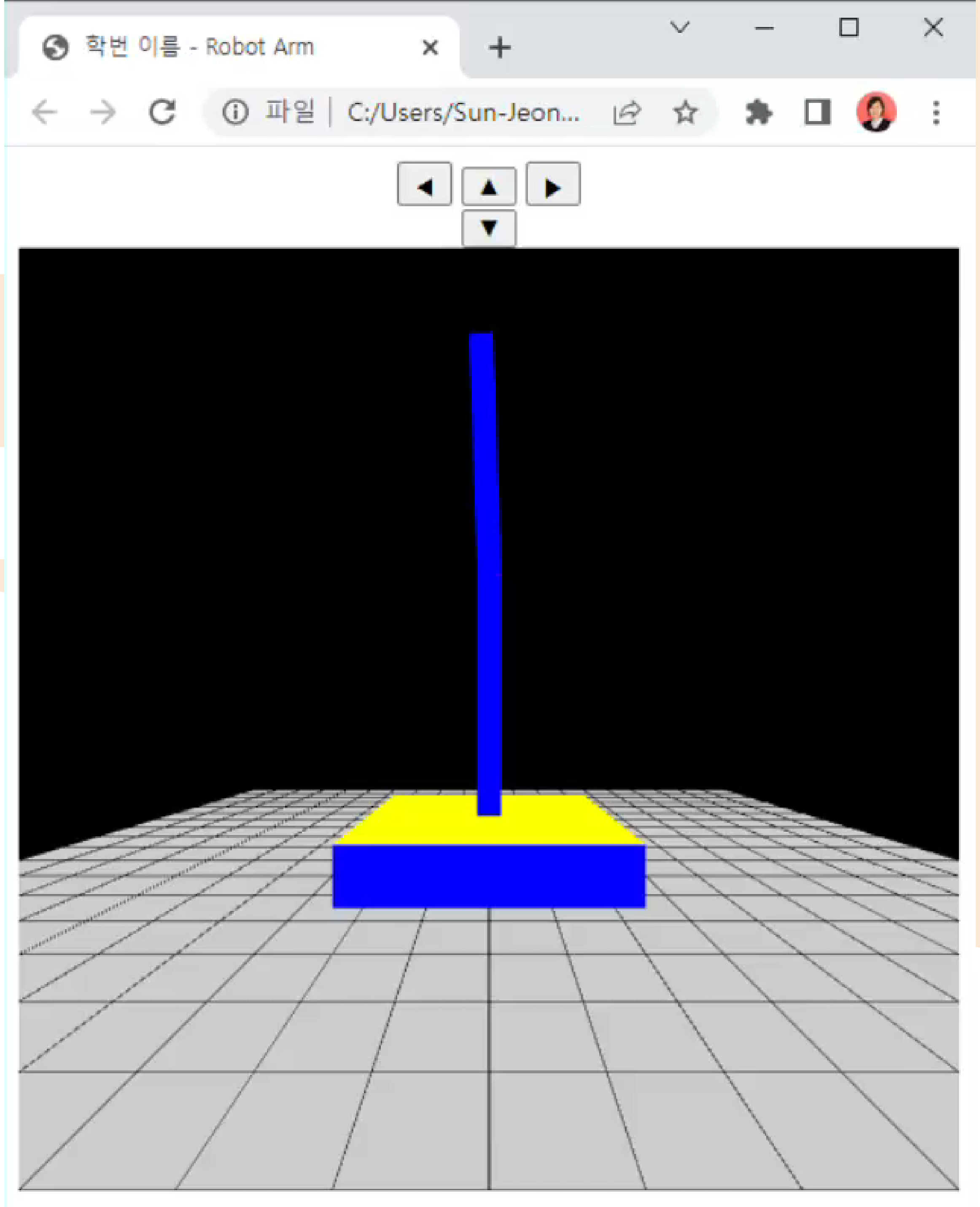
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

robotArm.html | robotArm.js

C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > robotArm.js > render

```javascript
                break;
            case 83:    // 'S'
            case 115:   // 's'
                break;
        }
        render();
};

var render = function() {
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    var modelMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);

    // draw the ground
    ground(modelMatrix);

    // draw a cube
    //gl.drawArrays(gl.TRIANGLES, vertCubeStart, numVertCubeTri);

    // draw the robot arm
    modelMatrix = mult(modelMatrix, rotate(theta[Base], 0, 1, 0 ));
    base(modelMatrix);

    modelMatrix = mult(modelMatrix, translate(0.0, BASE_HEIGHT, 0.0));
    modelMatrix = mult(modelMatrix, rotate(theta[LowerArm], 0, 0, 1 ));
    lowerArm(modelMatrix);

    modelMatrix = mult(modelMatrix, translate(0.0, LOWER_ARM_HEIGHT, 0.0));
    modelMatrix = mult(modelMatrix, rotate(theta[UpperArm], 0, 0, 1 ));
    upperArm(modelMatrix);

    window.requestAnimationFrame(render);
}

function ground(modelMatrix) {
```

Restricted Mode | 0 0 | Ln 187, Col 68 | Spaces: 4 | UTF-8 | CRLF | JavaScript


학번 이름 - Robot Arm
파일 | C:/Users/Sun-Jeon...

연습 문제

- ADSW 키를 이용하여 Base를 전후좌우로 이동(translation) 시키시오.
- Base의 회전은 PageUp(33), PageDown(34)키를 이용하시오.