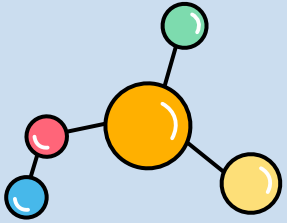
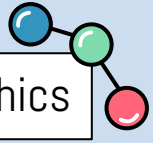
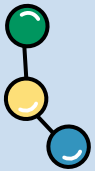


Computer Graphics



HIERARCHICAL MODELING SIMULATOR with WebGL

Team 14. Jinwon Jung & Geunho Lee



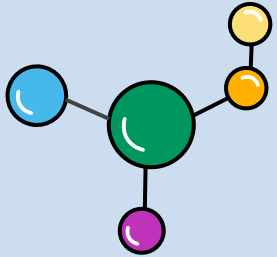
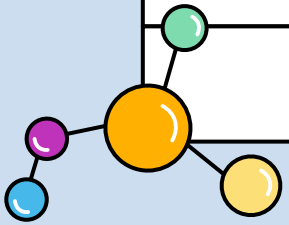


TABLE OF CONTENTS

01

WHAT TO IMPLEMENT

Introduce Subject and Purpose



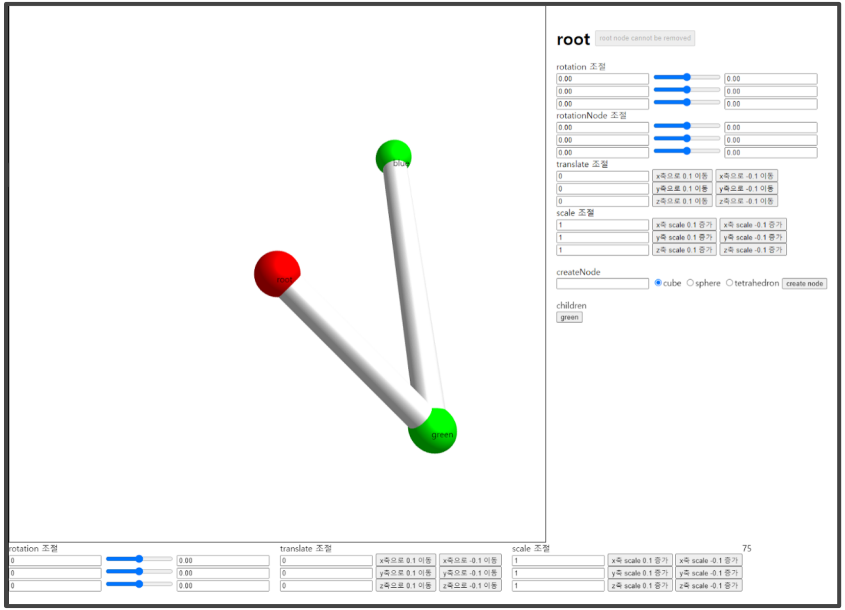
Subject and Purpose

We wanted to try something out of the ordinary.

However, there were no real objects that came to mind.
So, we came up with a virtual model!

In addition, we thought it would be good to learn the basic concept of hierarchical modeling through our virtual model.

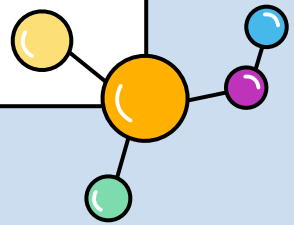
Therefore, the final aim was to become a hierarchical modeling simulator for education!



02

THE RESULT

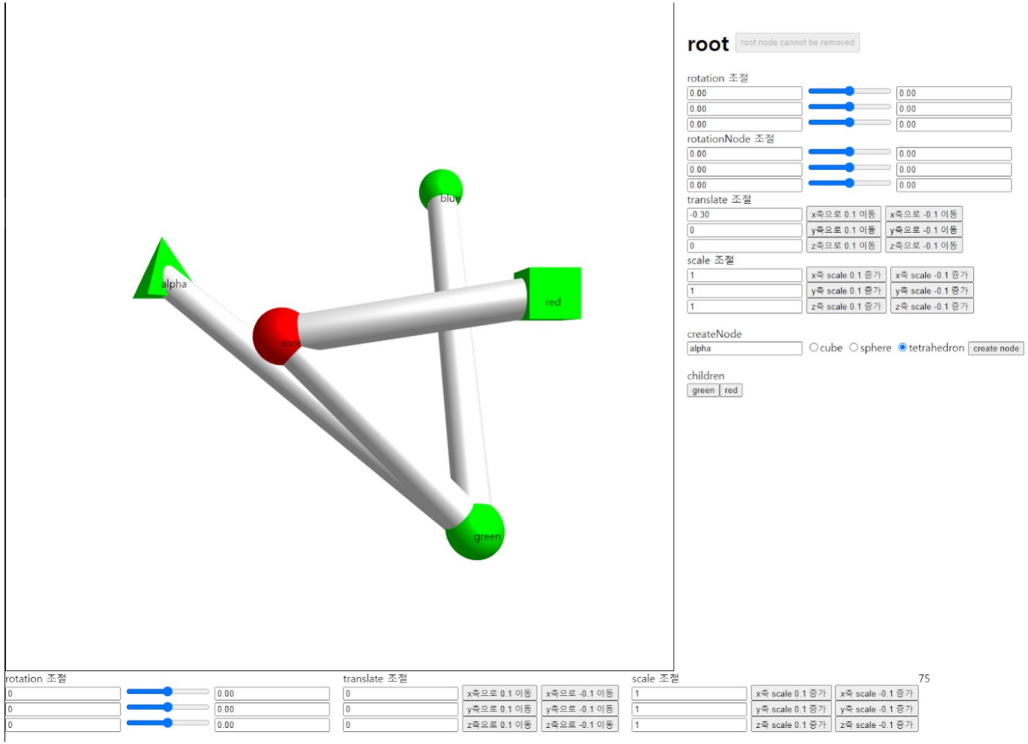
Show Capture images and Brief demonstration



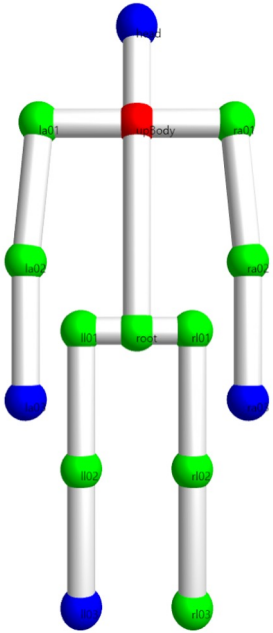
Simulator UI

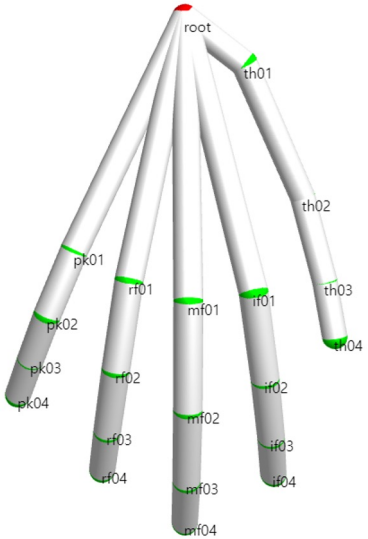
New nodes can be created and deleted.
We can also select a specific node and use the side button to transform it.

They have a hierarchy!



Hierarchical Modeling Example





It was not made with the addition of nodes and the transformation function of the simulator.

However, these are modeling examples made using the same sphere and cylinder.

03

HOW TO IMPLEMENT

Describe the Graphics and HTML techniques

Events in the Simulator I

Create

Add a new node
to the node object
with name, children, etc.

the structure of
node object

```
root: {
  isRoot: true,
  translate: [0, 0, 0],
  rotate: [0, 0, 0],
  rotateSpd: [0, 0, 0],
  rotateNode: [0, 0, 0],
  rotateNodeSpd: [0, 0, 0],
  scale: [1, 1, 1],
  children: ['green'],
  color: [0, 1, 0, 1],
  calcedCoord: [0, 0],
  shape: 'sphere',
},
```

the part of
node creation

```
const parent = renderObj.data[sltd];
const node = {
  translate: [0.2, 0.2, 0.2],
  rotate: [0, 0, 0],
  rotateSpd: [0, 0, 0],
  rotateNode: [0, 0, 0],
  rotateNodeSpd: [0, 0, 0],
  scale: [1, 1, 1],
  children: [],
  color: [0, 1, 0, 1],
  calcedCoord: [0, 0],
  shape: createNodeType,
};
renderObj.data[nodeName] = node;
parent.children.push(nodeName);
```

```
if (!renderObj.data[nodeName]) return;
if (nodeName === 'root') return;
if (sltd === nodeName) {
  selectNode(null);
}
renderObj.data[nodeName].children.forEach(removeNode);
delete renderObj.data[nodeName];
const names = document.getElementById('nodeNames');
const div = document.getElementById(nodeName);
names.removeChild(div);

Object.keys(renderObj.data).forEach(key => {
  const node = renderObj.data[key];
  node.children = node.children.filter(child => child !== nodeName);
});
```

Delete

Remove the selected node
from the node dictionary

Events in the Simulator 2

Select

Select a node on the canvas with a mouse click

```
const mat = [...calcedMatrix];
mulMatrix(mat, matNodeRotate);
const oriX = mat[12];
const oriY = mat[13];
const divV = mat[15];
const cordX = (1 + oriX / divV) * (gl.canvas.clientWidth / 2);
const cordY = (1 - oriY / divV) * (gl.canvas.clientHeight / 2);
node.calcedCoord = [cordX, cordY];
```

```
function renderDone() {
  Object.keys(renderObj.data).forEach((k) => {
    if (k === 'info') return;
    const elem = document.getElementById(k);
    elem.style.left = `${renderObj.data[k].calcedCoord[0]}px`;
    elem.style.top = `${renderObj.data[k].calcedCoord[1]}px`;
  });
```

```
axis.forEach(i => {
  const idx = axisIdx[i];
  elemMapper[`rot${i}S-N`].value = renderObj.data[k].rotateSpd[idx].toString();
  elemMapper[`rot${i}S-SPD`].value = renderObj.data[k].rotateSpd[idx].toFixed(2);
  elemMapper[`rot${i}SN-N`].value = renderObj.data[k].rotateNodeSpd[idx].toString();
  elemMapper[`rot${i}SN-SPD`].value = renderObj.data[k].rotateNodeSpd[idx].toFixed(2)

  elemMapper[`rot${i}D-N`].value = renderObj.data[k].rotate[idx].toFixed(2)
  elemMapper[`rot${i}DN-N`].value = renderObj.data[k].rotateNode[idx].toFixed(2)

  elemMapper[`tr${i}D-N`].value = renderObj.data[k].translate[idx].toString();
  elemMapper[`sc${i}D-N`].value = renderObj.data[k].scale[idx].toString();
});
```

```
rot_S_N.onchange = (ev) => {
  if (!sltd) return;
  renderObj.data[sltd].rotateSpd[idx] = Number(ev.target.value);
  rot_S_SPD.value = ev.target.value;
};

rot_S_SPD.onchange = (ev) => {
  if (!sltd) return;
  renderObj.data[sltd].rotateSpd[idx] = Number(ev.target.value);
  rot_S_N.value = ev.target.value;
};
```

Transformation

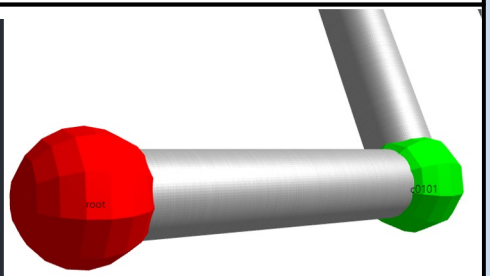
Transform the selected node using the side buttons

Modeling and Texture in Examples

```
const { triangles, normals, length, texture } = shapeBufMapper[node.shape];
gl.enableVertexAttribArray(texCoordLocation);
gl.bindBuffer(gl.ARRAY_BUFFER, texture);
gl.vertexAttribPointer(texCoordLocation, 2, gl.FLOAT, false, 0, 0);

gl.enableVertexAttribArray(positionLocation);
gl.bindBuffer(gl.ARRAY_BUFFER, triangles);
gl.vertexAttribPointer(positionLocation, 3, gl.FLOAT, false, 0, 0);

gl.enableVertexAttribArray(normalLo
gl.bindBuffer(gl.ARRAY_BUFFER, norma
gl.vertexAttribPointer(normalLocati

gl.drawArrays(gl.TRIANGLES, 0, leng
```

```
for (let i = 0; i < node.children.length; i++) {
  const cnode = obj[node.children[i]];
  renderObject(locMmatrix, obj, cnode, matOri, dt);
}
```

```
texCoordLocation = gl.getAttribLocation(program, 'aTexCoord');
gl.vertexAttribPointer(
  texCoordLocation,
  num,
  type,
  normalize,
  stride,
  offset
);
gl.enableVertexAttribArray(texCoordLocation);

textureLoc = gl.getUniformLocation(program, 'uTexture');
gl.activeTexture(gl.TEXTURE0);
gl.bindTexture(gl.TEXTURE_2D, texture);
gl.uniform1i(textureLoc, 0);

useTexLoc = gl.getUniformLocation(program, 'uUseTexture');
gl.uniform1f(useTexLoc, 0);
```



Modeling

It is called recursively from the root node of the model.
If it is not a root, a cylinder is drawn, and a sphere is drawn.

We get data from the vertex and texture coordinate, and normal vector buffers.
It also sends its own color and computed transformation matrix using `uniform` function.

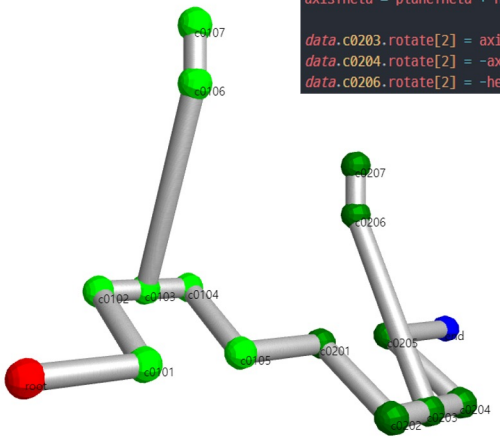
Steel Texture

The steel image was loaded with the `texImage2D` function, and the coordinates in the image were designated for each vertex.

Since we only need textures on the crank model, whether to apply the textures is passed to the vertex and fragment shaders.

Animation in Examples

```
data.root.rotate[2] -= data.info.rotSpeed * dt;

let planeTheta = -data.root.rotate[2];
let headTheta = Math.asin(data.info.planeLen * Math.sin(planeTheta) / data.info.axisLen);
let axisTheta = planeTheta - headTheta;

data.c0103.rotate[2] = axisTheta;
data.c0104.rotate[2] = -axisTheta;
data.c0106.rotate[2] = headTheta;

planeTheta = -data.root.rotate[2];
headTheta = Math.asin(data.info.planeLen * Math.sin(planeTheta) / data.info.axisLen);
axisTheta = planeTheta + headTheta;

data.c0203.rotate[2] = axisTheta;
data.c0204.rotate[2] = -axisTheta;
data.c0206.rotate[2] = -headTheta;
```



Crank :
We calculated the angles of other nodes that change according to the rotation angle of the root.
And the rotation angle is set to change to a value calculated for each frame.

Human Hand :
Angle change was applied to each finger joint of the initially set model.
In the form of a toggle, the finger moves when the button is pressed.
And it can be implemented by storing the toggle and the range of angles to rotate.

OUR TEAM

Jinwon Jung	Geunho Lee
Software / 201720706	Software / 202020767
- Node Create & Delete - Node Select & Transformation - Hierarchical Modeling	- Hierarchical Modeling - Texture Mapping - Animation

THANKS