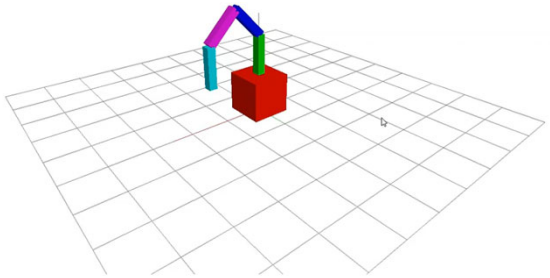
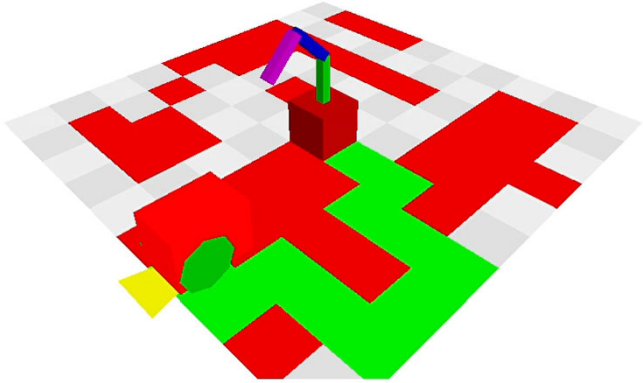
Robotics

Kyushu University
Ryo Kurazume

Contents

- Manipulator control
  - Direct Kinematics, Inverse Kinematics
  - Direct Dynamics, Inverse Dynamics
  - Force control, impedance control
  - Jacobian, Manipulability
- Rover control
  - Direct Kinematics, Inverse Kinematics
  - Dead reckoning, Feedback control
  - Dijkstra, Path planning
- 3D Simulation using WebGL





About this lecture

- Explain theories using PowerPoint

- Exercise using your own PC
  - 3D simulation using WebGL
  - Sample program is available at Moodle
  - Do not download from the beginning. Make your own program by yourself.

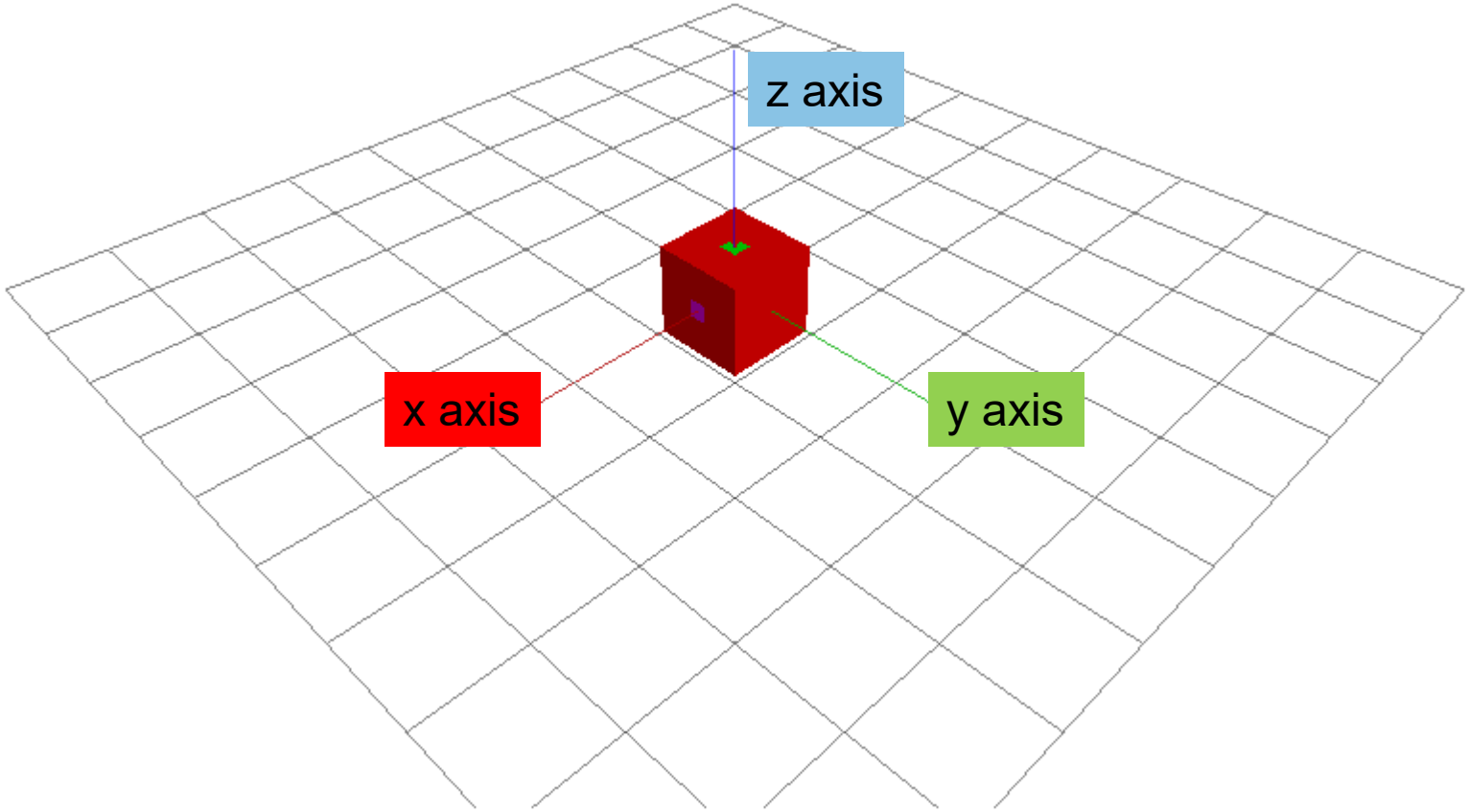
3D Programming with WebGL

3D Programming with WebGL

- Download and unzip manipulator.zip
- Open “manipulator” folder
- Double crick “manipulator.html”

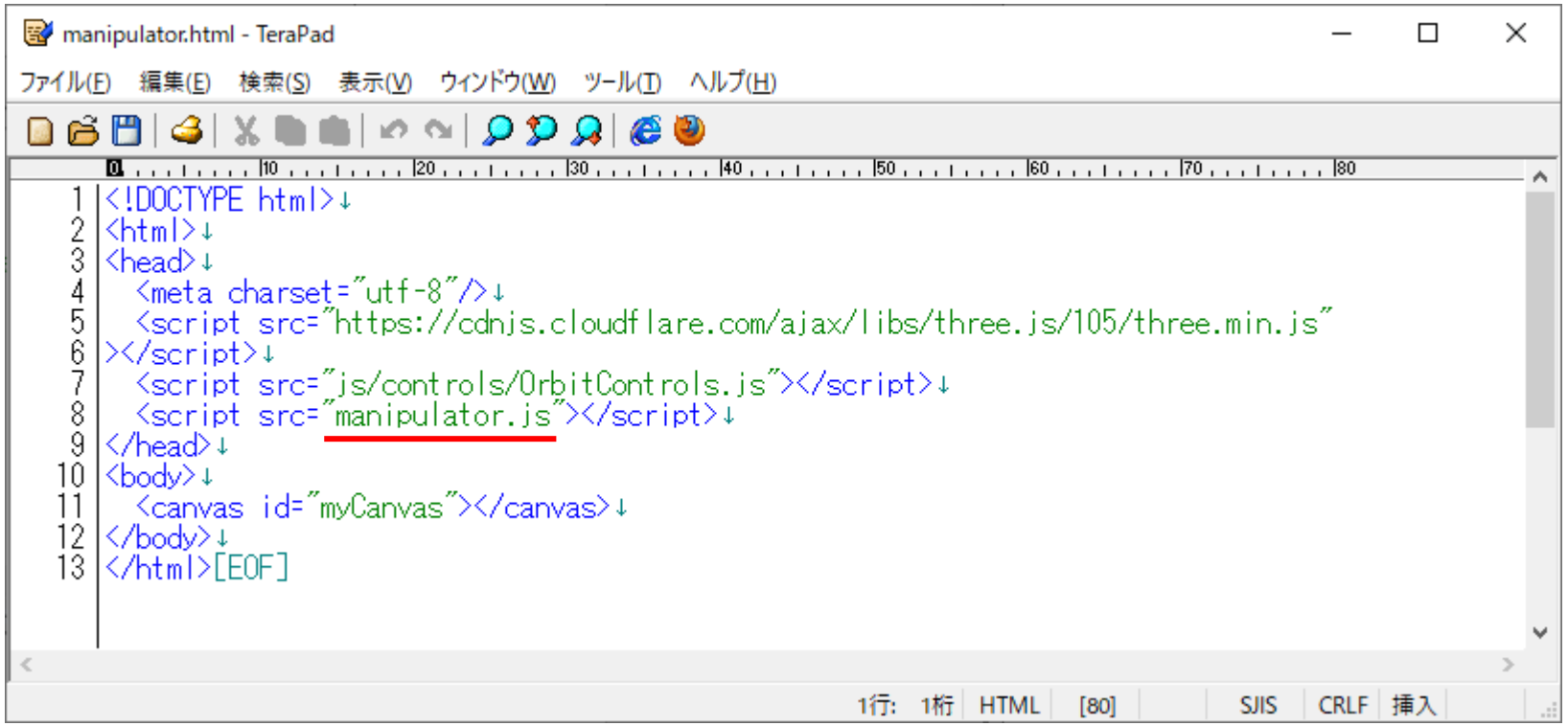
3D Programming with WebGL

Draw four boxes



3D Programming with WebGL

- Open “manipulator.html” with editor app.



- “manilulator.js” is included.

3D Programming with WebGL

- Open “manipulator. js” with editor app.

```
window.addEventListener('load', init);
//
// Window size
const width = 960;
const height = 540;
//
var c0 = new THREE.Vector3(); // center of body
var c1 = new THREE.Vector3(); // center of arm1
var c2 = new THREE.Vector3(); // center of arm2
var c3 = new THREE.Vector3(); // center of arm3
var q01 = new THREE.Quaternion(); // 0A1 matrix
var q02 = new THREE.Quaternion(); // 0A2 matrix
var q03 = new THREE.Quaternion(); // 0A3 matrix
var p0 = new THREE.Vector3(); // position of base origin
var p1 = new THREE.Vector3(); // position of joint 1
var p2 = new THREE.Vector3(); // position of joint 2
var p3 = new THREE.Vector3(); // position of joint 3
var pe = new THREE.Vector3(); // position of hand
var l = [1, 1, 1, 1]; // arm length
var phi = [0, 45, 90];
//
function init() {
  // renderer
  const renderer = new THREE.WebGLRenderer({
    canvas: document.querySelector('#myCanvas')
  });
  renderer.setPixelRatio(window.devicePixelRatio);
  renderer.setSize(width, height);
//
  // scene
  const scene = new THREE.Scene();
```

3D Programming with WebGL

```
1  window.addEventListener('load', init);
2
3  // Window size
4  const width = 960;
5  const height = 540;
6
7  var c0 = new THREE.Vector3(); // center of body
8  var c1 = new THREE.Vector3(); // center of arm1
9  var c2 = new THREE.Vector3(); // center of arm2
10 var c3 = new THREE.Vector3(); // center of arm3
11 var q01 = new THREE.Quaternion(); // 0A1 matrix
12 var q02 = new THREE.Quaternion(); // 0A2 matrix
13 var q03 = new THREE.Quaternion(); // 0A3 matrix
14 var p0 = new THREE.Vector3(); // position of base origin
15 var p1 = new THREE.Vector3(); // position of joint 1
16 var p2 = new THREE.Vector3(); // position of joint 2
17 var p3 = new THREE.Vector3(); // position of joint 3
18 var pe = new THREE.Vector3(); // position of hand
19 var l = [1, 1, 1, 1]; // arm length
20 var phi = [0, 45, 90];
21
22 function init() {
23   // renderer
24   const renderer = new THREE.WebGLRenderer({
25     canvas: document.querySelector('#myCanvas')
26   });
27   renderer.setPixelRatio(window.devicePixelRatio);
28   renderer.setSize(width, height);
29
30   // scene
31   const scene = new THREE.Scene();
```

Global variables

3D Programming with WebGL

Global variables

```
var c0 = new THREE.Vector3(); // center of body
var c1 = new THREE.Vector3(); // center of arm1
var c2 = new THREE.Vector3(); // center of arm2
var c3 = new THREE.Vector3(); // center of arm3
```

Arm position

```
var q01 = new THREE.Quaternion(); // 0A1 matrix
var q02 = new THREE.Quaternion(); // 0A2 matrix
var q03 = new THREE.Quaternion(); // 0A3 matrix
```

Arm orientation

```
var p0 = new THREE.Vector3(); // position of base origin
var p1 = new THREE.Vector3(); // position of joint 1
var p2 = new THREE.Vector3(); // position of joint 2
var p3 = new THREE.Vector3(); // position of joint 3
var pe = new THREE.Vector3(); // position of hand
```

Joint position

```
var l = [1, 1, 1, 1]; // arm length
var phi = [0, 45, 90];
```

Initial values

3D Programming with WebGL

```
function init() {
  // renderer
  const renderer = new THREE.WebGLRenderer({
    canvas: document.querySelector('#myCanvas')
  });
  renderer.setPixelRatio(window.devicePixelRatio);
  renderer.setSize(width, height);

  // scene
  const scene = new THREE.Scene();

  // camera
  const camera = new THREE.PerspectiveCamera(45, width / height, 1, 10000);
  camera.up.x = 0; camera.up.y = 0; camera.up.z = 1;
  camera.position.set(7, 7, 7);

  // camera controller
  const controls = new THREE.OrbitControls(camera);
  controls.enableDamping = true;
  controls.dampingFactor = 0.2;

  // parallel light
  const directionalLight = new THREE.DirectionalLight( 0xFFFFFF, 0.7 );
  directionalLight.position.set(0, 1, 1);
  scene.add( directionalLight );

  // ambient light
  const ambientLight = new THREE.AmbientLight( 0x404040 ); // soft white light
  scene.add(ambientLight);

  // floor mesh
  createFloor();

  function createFloor() {
    var geometry = new THREE.Geometry();
    var N = 10;
    var w = 1;
    for( var i=0; i<N; i++){
      for( var j=0; j<=N; j++){
        geometry.vertices.push( new THREE.Vector3((i - N/2 ) * w, (j - N/2 ) * w,
```

Main function

3D Programming with WebGL

```
function init() {
  // renderer
  const renderer = new THREE.WebGLRenderer({
    canvas: document.querySelector('#myCanvas')
  });
  renderer.setPixelRatio(window.devicePixelRatio);
  renderer.setSize(width, height);

  // scene
  const scene = new THREE.Scene();

  // camera
  const camera = new THREE.PerspectiveCamera(45, width / height, 1, 10000);
  camera.up.x = 0; camera.up.y = 0; camera.up.z = 1;
  camera.position.set(7, 7, 7);

  // camera controller
  const controls = new THREE.OrbitControls(camera);
  controls.enableDamping = true;
  controls.dampingFactor = 0.2;

  // parallel light
  const directionalLight = new THREE.DirectionalLight( 0xFFFFFF, 0.7 );
  directionalLight.position.set(0, 1, 1);
  scene.add( directionalLight );

  // ambient light
  const ambientLight = new THREE.AmbientLight( 0x404040 ); // soft white light
  scene.add(ambientLight);

  // floor mesh
  createFloor();

  function createFloor() {
    var geometry = new THREE.Geometry();
    var N = 10;
    var w = 1;
    for( var i=0; i<N; i++){
      for( var j=0; j<=N; j++){
        geometry.vertices.push( new THREE.Vector3((i - N/2 ) * w, (j - N/2 ) * w,
```

Camera setting

Mouse control

Light setting

3D Programming with WebGL

```
49  const ambientLight = new THREE.AmbientLight( 0x404040 ); // soft white light
50  scene.add(ambientLight);
51
52  // floor mesh
53  createFloor();
54
55  function createFloor() {
56    var geometry = new THREE.Geometry();
57    var N = 10;
58    var w = 1;
59    for( var i=0; i<N; i++){
60      for( var j=0; j<=N; j++){
61        geometry.vertices.push( new THREE.Vector3((i - N/2 ) * w, (j - N/2 ) * w, 0) );
62        geometry.vertices.push( new THREE.Vector3(((i+1) - N/2 ) * w, (j - N/2 ) * w, 0) );
63      }
64      for( var j=0; j<=N; j++){
65        geometry.vertices.push( new THREE.Vector3((j - N/2 ) * w, (i - N/2 ) * w, 0) );
66        geometry.vertices.push( new THREE.Vector3((j - N/2 ) * w, ((i+1) - N/2 ) * w, 0) );
67      }
68    }
69    var material = new THREE.LineBasicMaterial({ color: 0xFFFFFF, transparent:true, opacity:0.5 });
70    lines = new THREE.LineSegments(geometry, material);
71    scene.add(lines);
72  }
73
74  // base
75  createBase();
76
77  function createBase() {
78    var geometry_base = new THREE.BoxGeometry(1, 1, l[0]);
79    var material_base = new THREE.MeshStandardMaterial({color: 0xff0000, side: THREE.DoubleSide});
80    base = new THREE.Mesh(geometry_base, material_base);
81    scene.add(base);
82  }
83
84  // arm 1
```

Draw floor

3D Programming with WebGL

```
71      scene.add(lines);
72    }
73
74    // base
75    createBase();
76
77    function createBase() {
78      var geometry_base = new THREE.BoxGeometry(1, 1, l[0]);
79      var material_base = new THREE.MeshStandardMaterial({color: 0xff0000, side: THREE.DoubleSide});
80      base = new THREE.Mesh(geometry_base, material_base);
81      scene.add(base);
82    }
83
84    // arm 1
85    createArm1();
86
87    function createArm1() {
88      var geometry = new THREE.BoxGeometry(0.2, 0.2, l[1]);
89      var material = new THREE.MeshStandardMaterial({color: 0x00ff00, side: THREE.DoubleSide});
90      arm1 = new THREE.Mesh(geometry, material);
91      scene.add(arm1);
92    }
93
94    // arm 2
95    createArm2();
96
97    function createArm2() {
98      var geometry = new THREE.BoxGeometry(l[2], 0.2, 0.2);
99      var material = new THREE.MeshStandardMaterial({color: 0x0000ff, side: THREE.DoubleSide});
100     arm2 = new THREE.Mesh(geometry, material);
101     scene.add(arm2);
102   }
103
104   // arm 3
105   createArm3();
```

Draw four boxes

3D Programming with WebGL

```
100     arm2 = new THREE.Mesh(geometry, material);↓
101     scene.add(arm2);↓
102   }↓
103 ↓
104   // arm 3↓
105   createArm3();↓
106 ↓
107   function createArm3() {↓
108     var geometry = new THREE.BoxGeometry(l[3], 0.2, 0.2);↓
109     var material = new THREE.MeshStandardMaterial({color: 0xff00ff, side: THREE.DoubleSide});↓
110     arm3 = new THREE.Mesh(geometry, material);↓
111     scene.add(arm3);↓
112   }↓
113 ↓
114   DK();↓
115 ↓
116   // rendering↓
117   tick();↓
118 ↓
119   // keyboad control↓
120   var RotSpeed = 1;↓
121 ↓
122   document.addEventListener("keydown", onDocumentKeyDown, false);↓
123   function onDocumentKeyDown(event) {↓
124     var keyCode = event.which;↓
125     if (keyCode == 90) {↓
126       // z↓
127       phi[0] += RotSpeed;↓
128     } else if (keyCode == 88) {↓
129       // x↓
130       phi[0] -= RotSpeed;↓
131     } else if (keyCode == 65) {↓
132       // a↓
133       phi[1] += RotSpeed;↓
134     } else if (keyCode == 83) {↓
135       // s↓
136       phi[1] -= RotSpeed;↓
```

Draw four boxes

Direct kinematics (later)

Rendering

Keyboard

3D Programming with WebGL

```
116   // rendering↓
117   tick();↓
118 ↓
119   // keyboad control↓
120   var RotSpeed = 1;↓
121 ↓
122   document.addEventListener("keydown", onDocumentKeyDown, false);↓
123   function onDocumentKeyDown(event) {↓
124     var keyCode = event.which;↓
125     if (keyCode == 90) {↓
126       // z↓
127       phi[0] += RotSpeed;↓
128     } else if (keyCode == 88) {↓
129       // x↓
130       phi[0] -= RotSpeed;↓
131     } else if (keyCode == 65) {↓
132       // a↓
133       phi[1] += RotSpeed;↓
134     } else if (keyCode == 83) {↓
135       // s↓
136       phi[1] -= RotSpeed;↓
137     } else if (keyCode == 81) {↓
138       // q↓
139       phi[2] += RotSpeed;↓
140     } else if (keyCode == 87) {↓
141       // w↓
142       phi[2] -= RotSpeed;↓
143     } else if (keyCode == 32) {↓
144       phi[0] =  0.0;↓
145       phi[1] = 45.0;↓
146       phi[2] = 90.0;↓
147     }↓
148     DK();↓
149   }↓
150 ↓
151 ↓
```

Keyboard

Key Code
http://faq.creasus.net/04/0131/CharCode.html

3D Programming with WebGL

```
152  function DK() {
153    //Insert code for DK here
154
155  }
156
157  function IK() {
158    //Insert code for IK here
159
160  }
161
162  function calcJacobi() {
163    //Insert code for jacobian calculation here
164
165  }
166
167  function tick() {
168
169    base.position.copy(c0);
170    arm1.position.copy(c1);
171    arm1.quaternion.copy(q01);
172    arm2.position.copy(c2);
173    arm2.quaternion.copy(q02);
174    arm3.position.copy(c3);
175    arm3.quaternion.copy(q03);
176
177    // update camera controller
178    controls.update();
179
180    // rendering
181    renderer.render(scene, camera);
182
183   // console.log("phi " + phi[0] + " " + phi[1] + " " + phi[2]);
184    requestAnimationFrame(tick);
185  }
186
```

Direct kinematics (later)

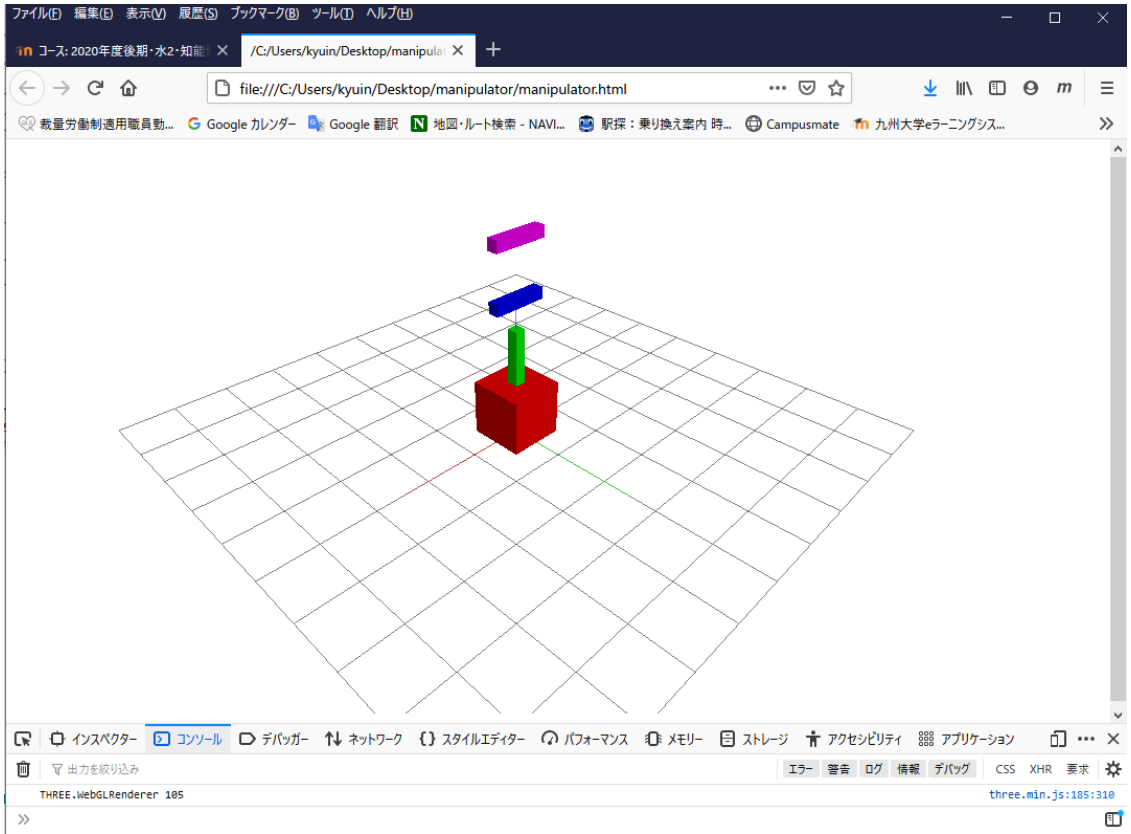
Inversekinematics (later)

RMRC control (later)

Rendering

Console (Firefox)

Control + Shift + “k”

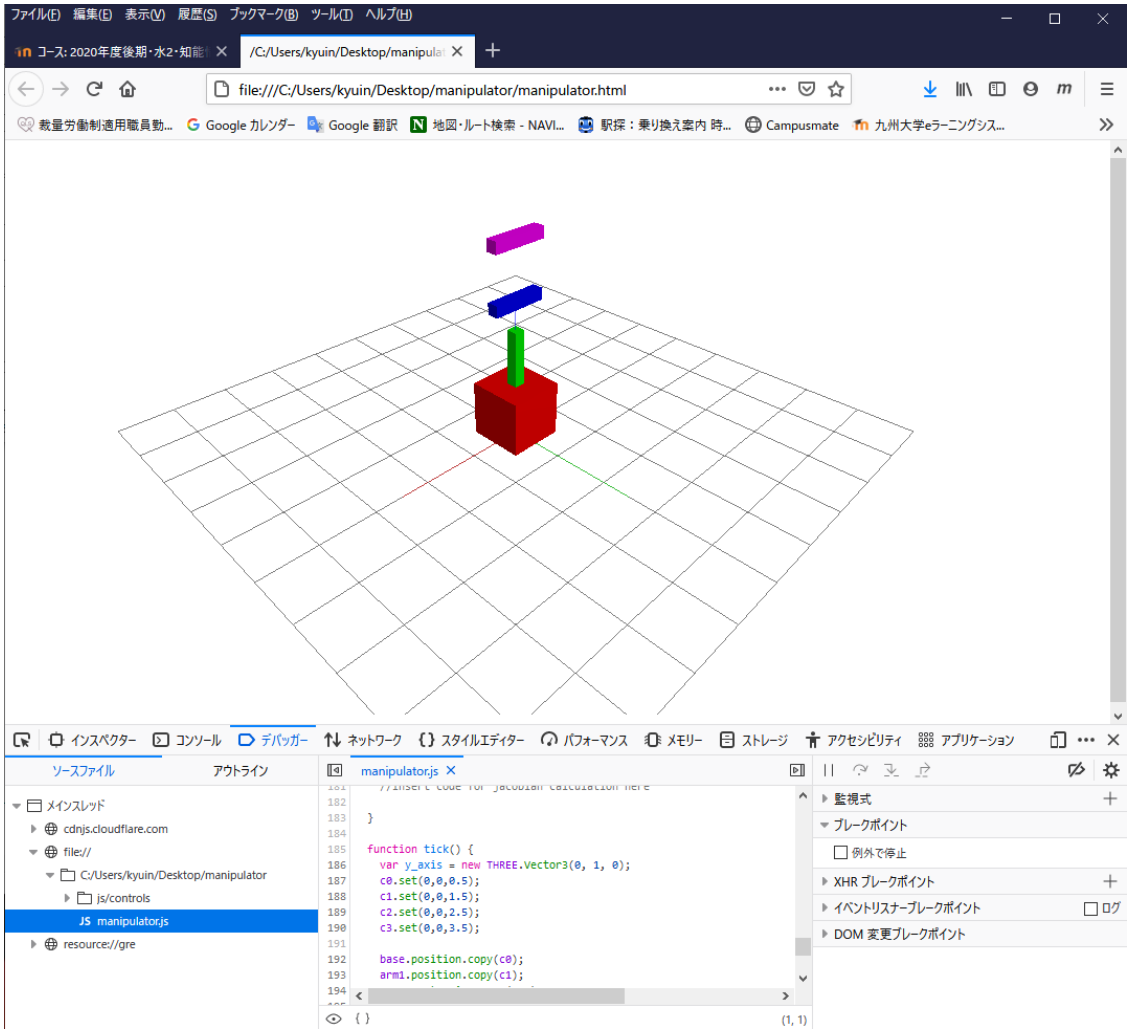


Variables can be shown here.

console.log("phi " + phi[0]);

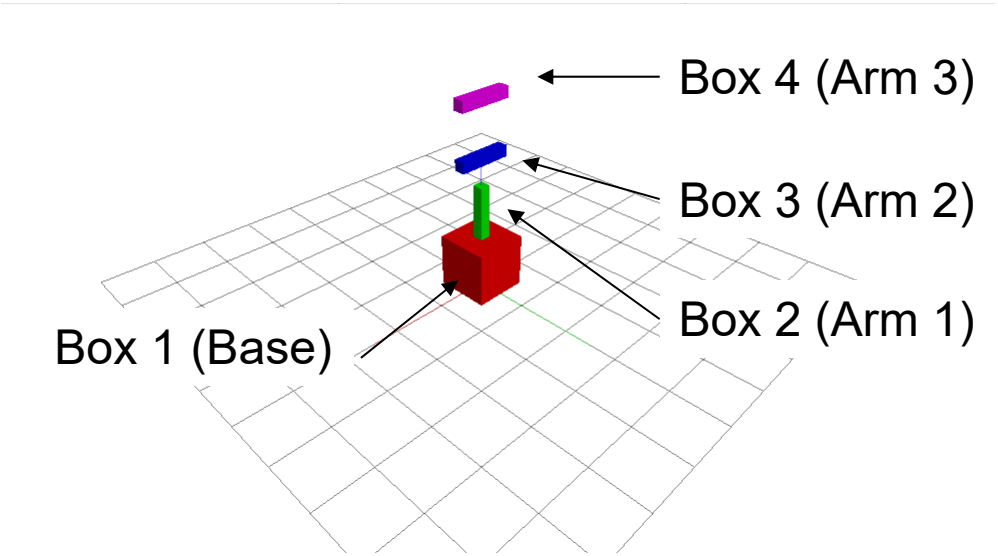
Debugger (Firefox)

Control + Shift + “I”



Exercises

- Change the variables “c0, c1, c2, c3” and show the following images

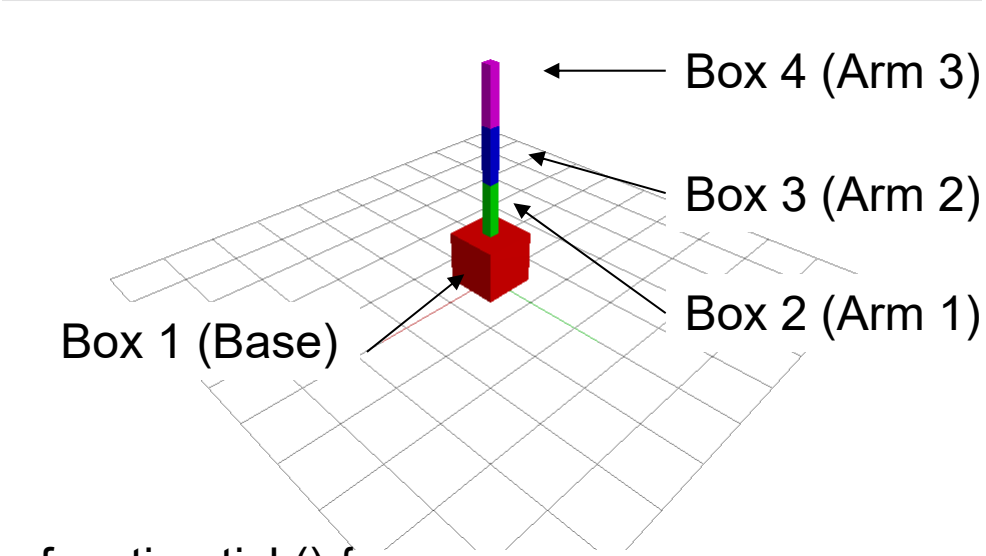


C1 is the center of box 2 (Arm 1)
C2 is the center of box 3 (Arm 2)
C3 is the center of box 4 (Arm 3)

```
function tick() {
  c0.set(0,0,0.5);
  c1.set(0,0,1.5);
  …
```

Exercises

- Change the variables “c0, c1, c2, c3” and “q02, q03”, and show the following images



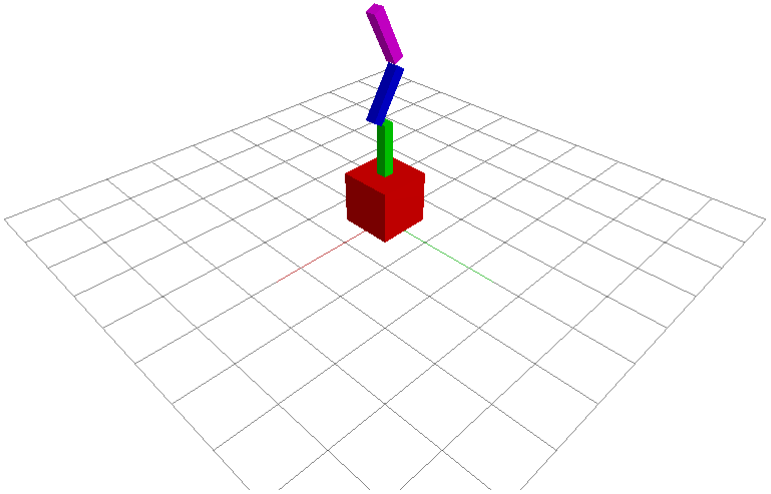
q02 is the attitude of box 3
q03 is the attitude of box 4

Rotate box 3 and 4 around y axis

```
function tick() {
  var y_axis = new THREE.Vector3(0, 1, 0);
  q02 = new THREE.Quaternion().setFromAxisAngle(y_axis, -Math.PI/2.0);
…
```

Exercises

- Change the variables “q01, q02, q03” so that the boxes rotate by key input



```
function tick() {
  var y_axis = new THREE.Vector3(0, 1, 0);
  q01 = new THREE.Quaternion().setFromAxisAngle(y_axis, phi[0]);
…
```

Exercises

- Change the lighting condition

Color (Hexadecimal)

```
// parallel light
const directionalLight = new THREE.DirectionalLight( 0xFFFFFF, 0.7 );
directionalLight.position.set(0, 1, 1);
scene.add( directionalLight );

// ambient light
const ambientLight = new THREE.AmbientLight( 0x404040 ); // soft white light
scene.add(ambientLight);
```

Color

Color : 0x(R value 00~FF)(G value 00~FF)(B value 00~FF)
0x000000 (Black) ~ 0xFFFFFF (White)

Submit your video

- Capture the video of your browser and submit the video (mpeg) on moodle by the next lecture.

- If you are unable to develop a program within this period, a sample program will be available on moodle next week. Please read them and submit your video.