2D and 3D Transformations

Joaquim Madeira

March 2024

Topics

- Motivation
- 2D Transformations
- 3D Transformations
- Transformations in Three.js
- The Scene Graph

MOTIVATION

CG Main Tasks

- Modeling
  - Construct individual models / objects
  - Assemble them into a 2D or 3D scene
- Animation
  - Static vs. dynamic scenes
  - Movement and / or deformation
- Rendering
  - Generate final images
  - Where is the viewer / camera ?
  - How is he / she looking at the scene?

Modeling vs Rendering

- Modeling
  - Create models
  - Apply materials to models
  - Place models around scene
  - Place lights in the scene
  - Place the camera

YouTube Demo

- Rendering
  - Take picture with the camera

[van Dam]

Three.js – A scene with a cube

```
// The SCENE

scene = new THREE.Scene();

// The MODEL --- A cube is added to the scene

var geometry = new THREE.BoxBufferGeometry( 200, 200, 200 );

var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );

model = new THREE.Mesh( geometry, material );

scene.add( model );
```

Three.js – The camera

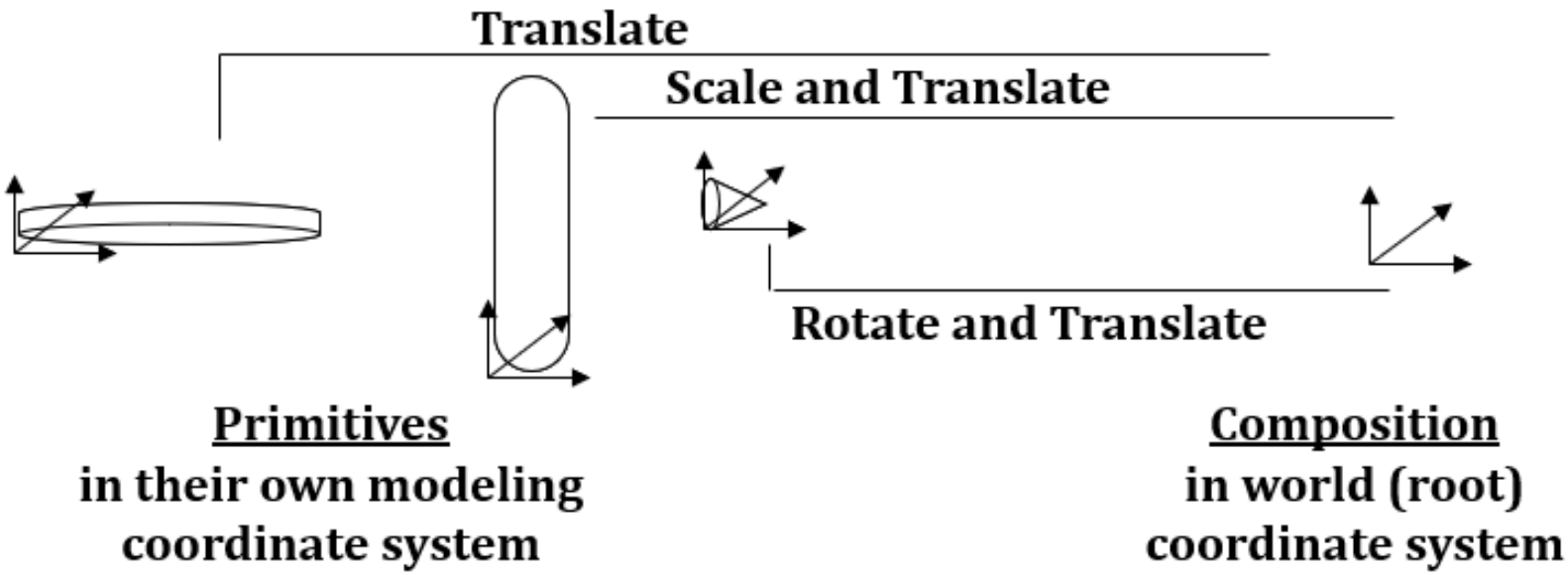
```
// The CAMERA

// --- Where the viewer is and how he is looking at the scene

camera = new THREE.PerspectiveCamera( 70,

                    window.innerWidth / window.innerHeight, 1, 1000 );

camera.position.z = 400;
```

Three.js – Animation

```
function animate() {
    requestAnimationFrame( animate );
    model.rotation.x += 0.005;
    model.rotation.y += 0.01;
    renderer.render( scene, camera );
}
```

Composition of a geometric model

- Assemble primitives to create final object
  - Apply affine transformations



[van Dam]

2D and 3D Transformations

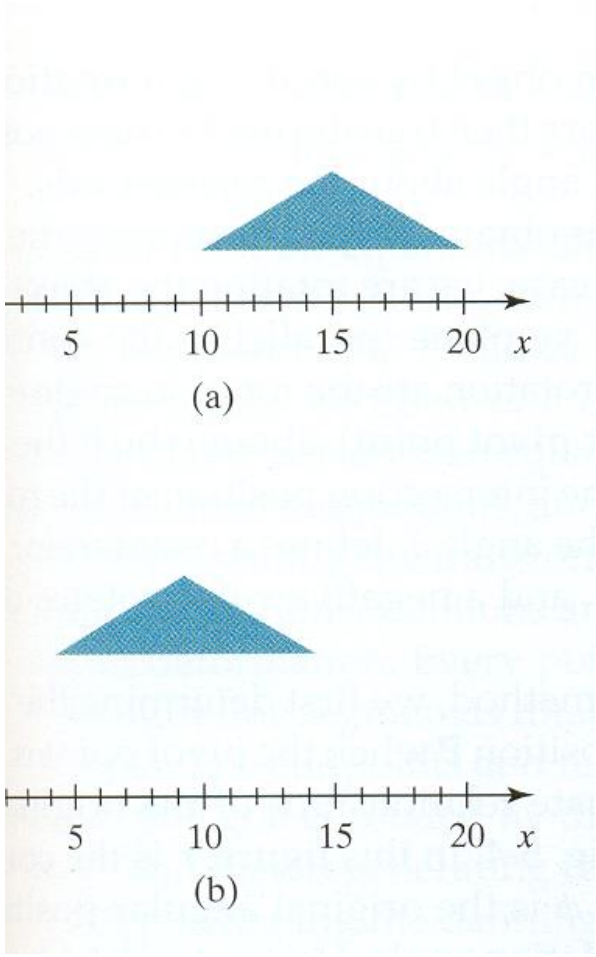
- Position, rotate and scale objects on the 2D plane or in 3D space
- Basic transformations
  - Translation
  - Rotation
  - Scaling
- Matricial representation
  - Homogeneous coordinates !!
  - Concatenation = Matrix products
- Complex transformations ?
  - Decompose into a sequence of basic transformations

2D TRANSFORMATIONS

2D Translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\mathbf{P}' = \mathbf{T}(t_x, t_y) \cdot \mathbf{P}$$



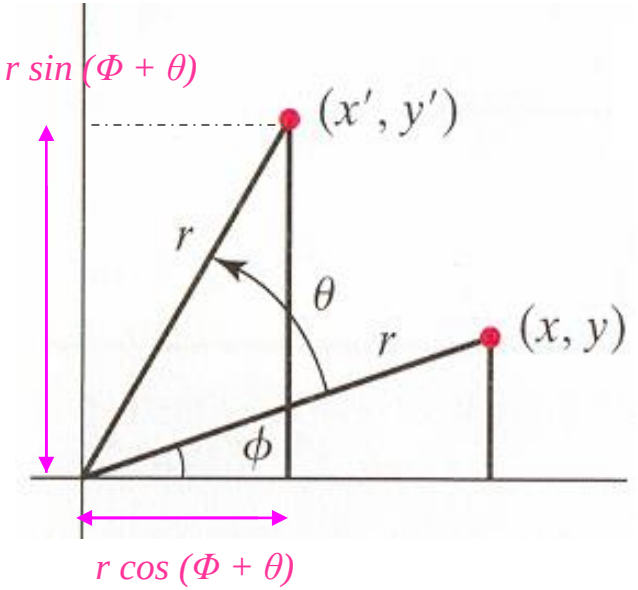
2D Scaling

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\mathbf{P}' = \mathbf{S}(s_x, s_y) \cdot \mathbf{P}$$

2D Rotation

$$x' = r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta$$
$$y' = r\sin(\phi+\theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta$$



$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\mathbf{P}' = \mathbf{R}(\theta) \cdot \mathbf{P}$$

Concatenation

$$\mathbf{P}' = \mathbf{T}(t_{2x}, t_{2y}) \cdot \{\mathbf{T}(t_{1x}, t_{1y}) \cdot \mathbf{P}\}$$
$$= \{\mathbf{T}(t_{2x}, t_{2y}) \cdot \mathbf{T}(t_{1x}, t_{1y})\} \cdot \mathbf{P}$$

$$\begin{bmatrix} 1 & 0 & t_{2x} \\ 0 & 1 & t_{2y} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_{1x} \\ 0 & 1 & t_{1y} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{1x} + t_{2x} \\ 0 & 1 & t_{1y} + t_{2y} \\ 0 & 0 & 1 \end{bmatrix}$$

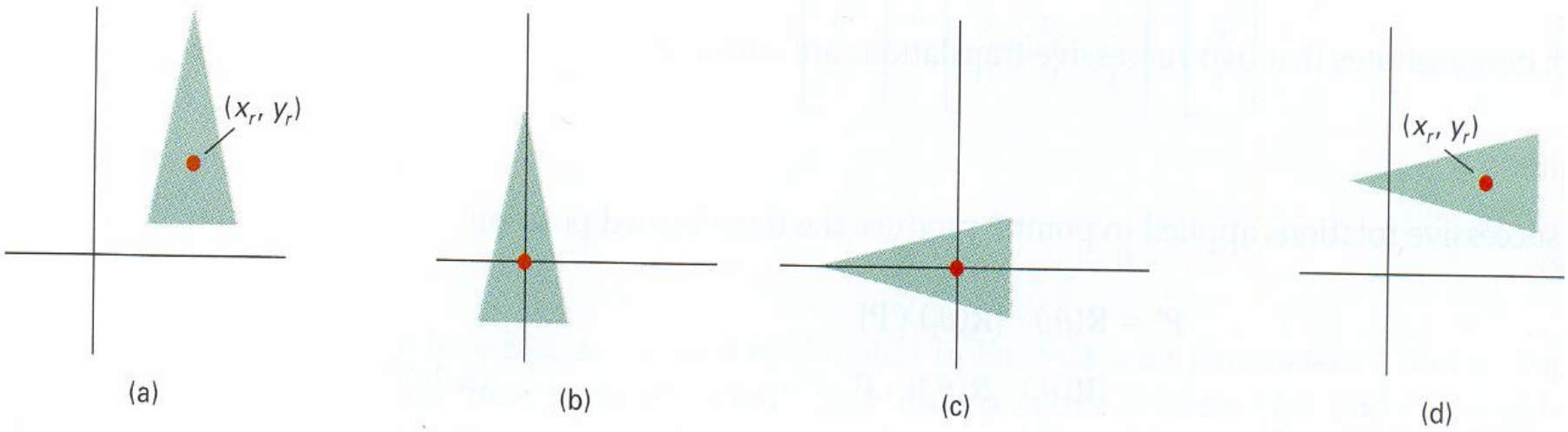
$$\mathbf{T}(t_{2x}, t_{2y}) \cdot \mathbf{T}(t_{1x}, t_{1y}) = \mathbf{T}(t_{1x} + t_{2x}, t_{1y} + t_{2y})$$

Concatenation

$$\begin{bmatrix} s_{2x} & 0 & 0 \\ 0 & s_{2y} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_{1x} & 0 & 0 \\ 0 & s_{1y} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{1x} \cdot s_{2x} & 0 & 0 \\ 0 & s_{1y} \cdot s_{2y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

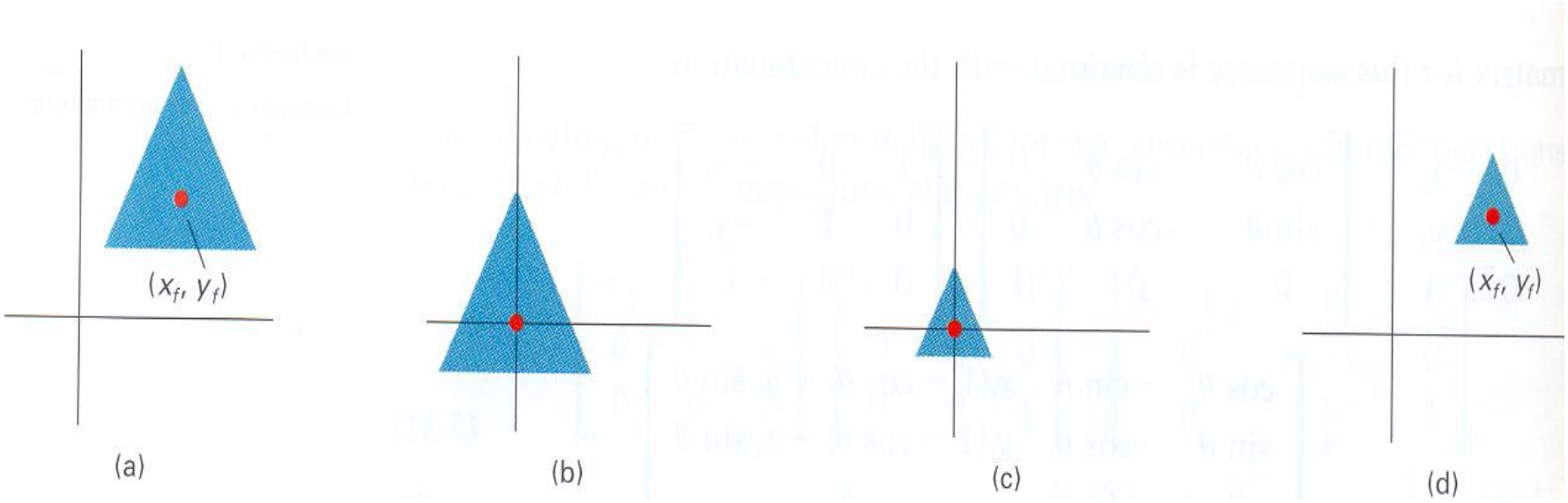
$$\mathbf{S}(s_{2x}, s_{2y}) \cdot \mathbf{S}(s_{1x}, s_{1y}) = \mathbf{S}(s_{1x} \cdot s_{2x}, \quad s_{1y} \cdot s_{2y})$$

Arbitrary Rotation



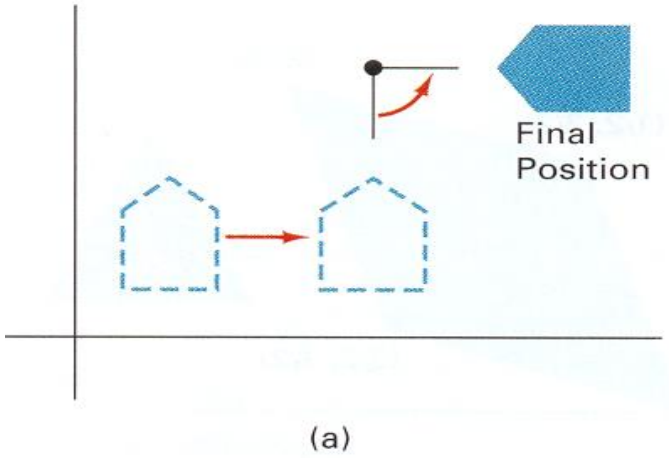
Translation + Rotation + Inverse Translation

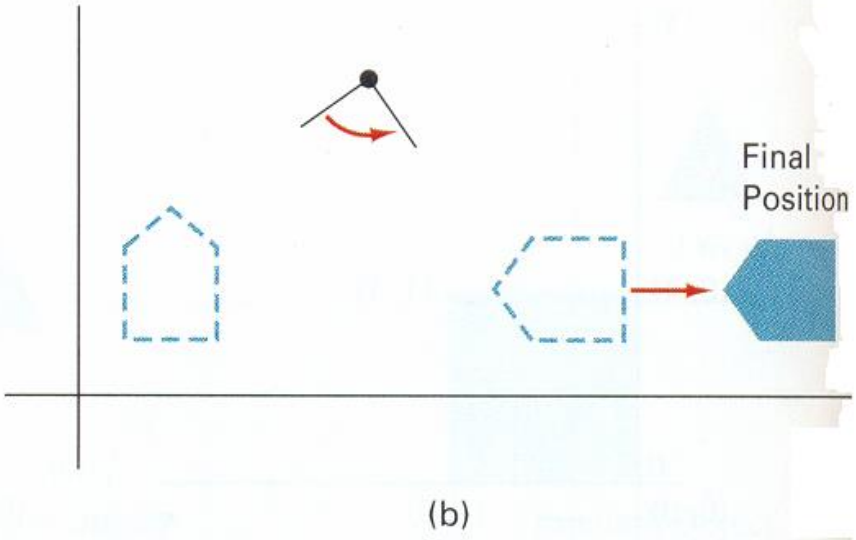
Arbitrary Scaling



Translation + Scaling + Inverse Translation

Order is important !





3D TRANSFORMATIONS

3D Transformations

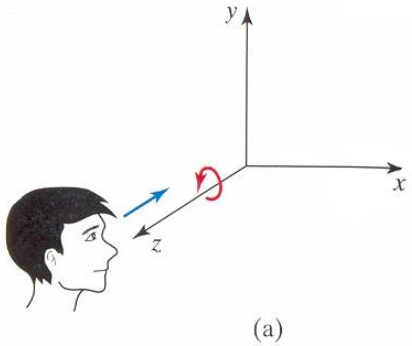
- Translation

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Scaling

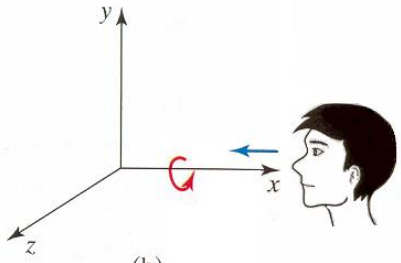
$$\mathbf{S} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Rotation



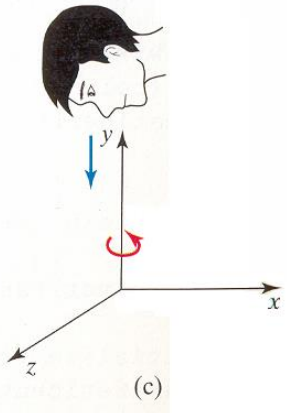
(a)

- Rotation around each one of the coordinate axis



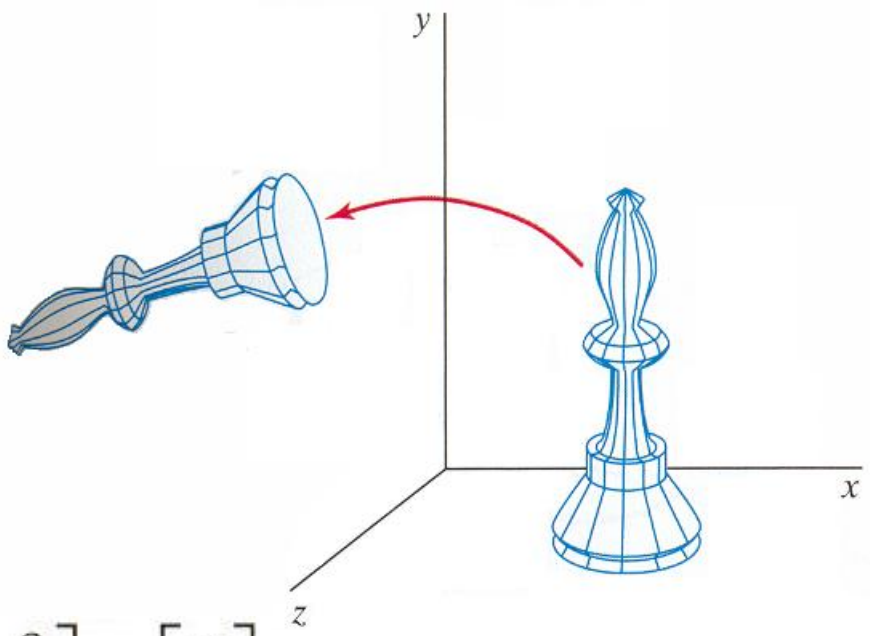
(b)

- Positive rotations are CCW !!



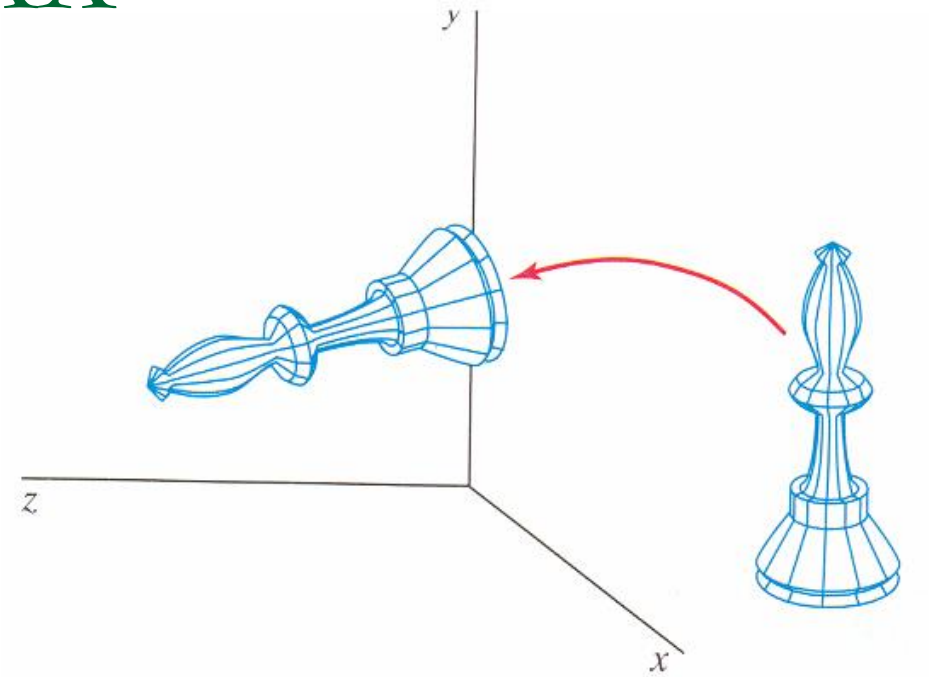
(c)

Rotation around ZZ'



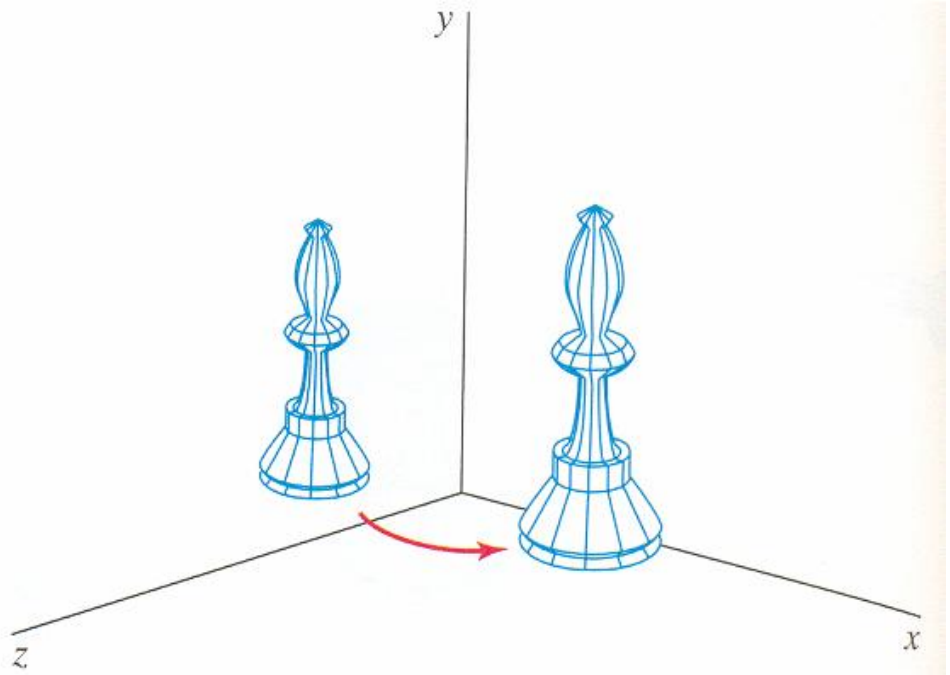
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation around XX'



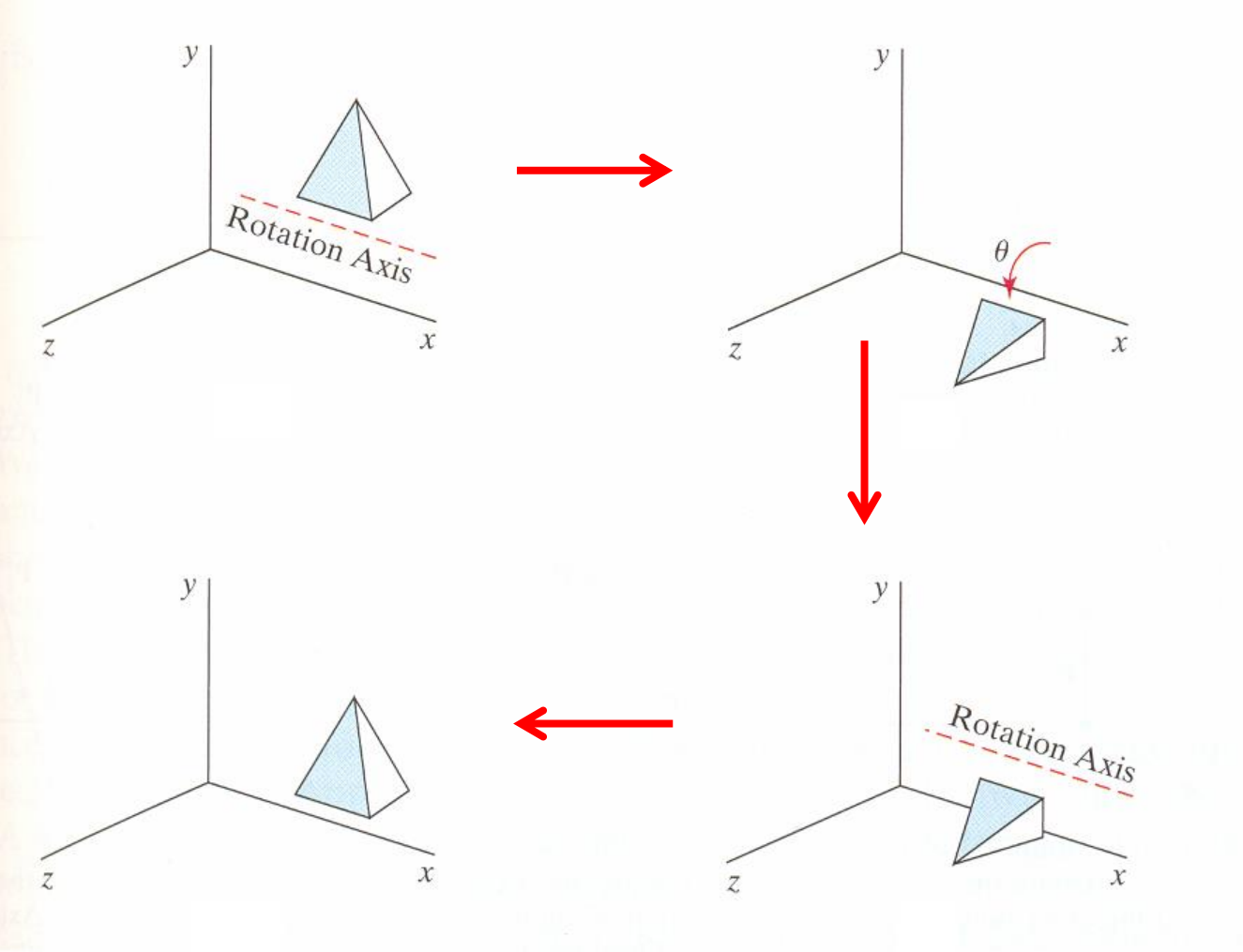
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation around YY'



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Example – Decomposition



Task – Application Problems

- Solve the first application problems !!
- See if you have any questions / difficulties

Possible References

- 2D and 3D transformations are presented in any Computer Graphics book
- E. Angel and D. Shreiner. Interactive Computer Graphics, 7th Ed., Addison-Wesley, 2015
- J M Pereira, et al. Introdução à Computação Gráfica. FCA, 2018

TRANSFORMATIONS IN THREE.JS

Three.js – position

- Position of a model/object relative to the position of its parent in the scene graph
  - Default: (0,0,0)
- The parent is usually a THREE.Scene object or a THREE.Object3D object

```
cube.position.x=10;
cube.position.y=3;
cube.position.z=1;

cube.position.set(10,3,1);

cube.postion=new THREE.Vector3(10,3,1)
```

Three.js – scale

- Scale factors of a model/object relative to its XX, YY or ZZ axis
  - Default: (1,1,1)

Three.js – rotation

- Rotation angle(s) for a model/object around any one of its XX, YY or ZZ axis
  - In radians !!
- And the order in which the rotations are carried out
  - Optional argument !!
- Default: (0,0,0,'XYZ')

```
cube.rotation.x = 0.5*Math.PI;
cube.rotation.set(0.5*Math.PI, 0, 0);
cube.rotation = new THREE.Vector3(0.5*Math.PI,0,0);
```

Three.js – 4 × 4 Matrix Class

Matrix4

A class representing a 4x4 matrix.

The most common use of a 4x4 matrix in 3D computer graphics is as a Transformation Matrix. For an introduction to transformation matrices as used in WebGL, check out this tutorial.

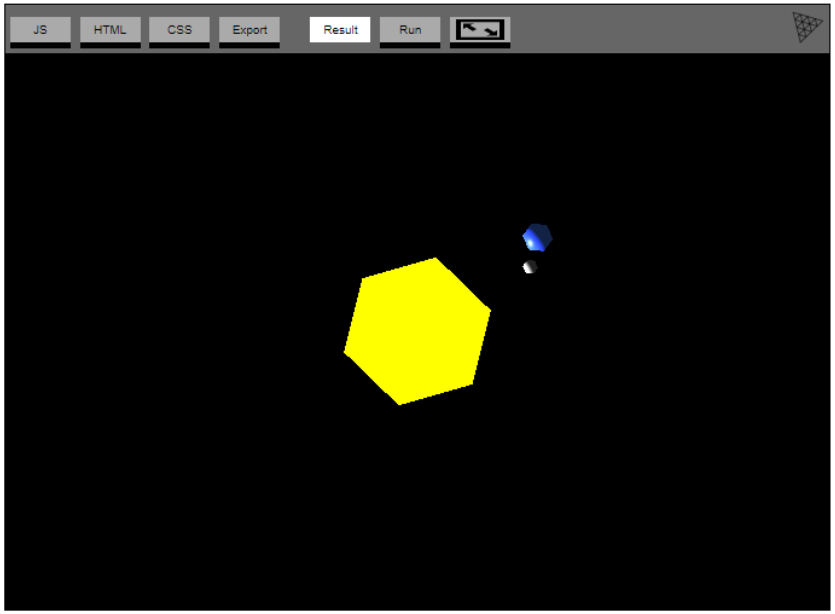
This allows a Vector3 representing a point in 3D space to undergo transformations such as translation, rotation, shear, scale, reflection, orthogonal or perspective projection and so on, by being multiplied by the matrix. This is known as *applying* the matrix to the vector.

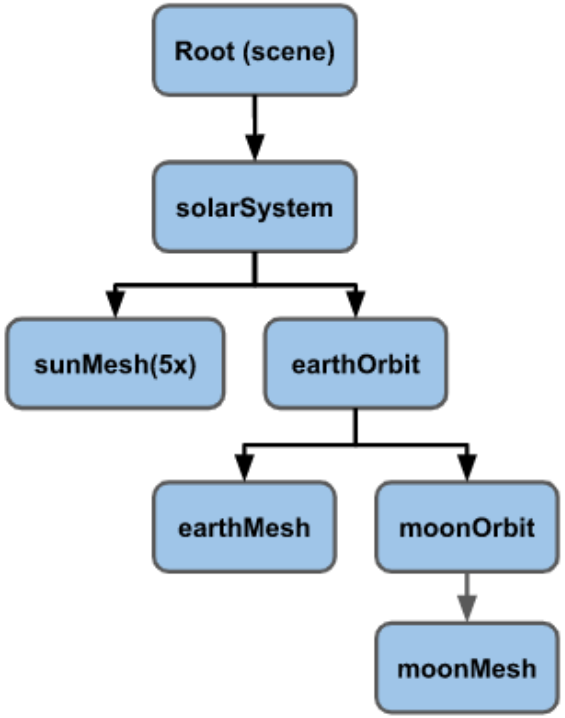
[https://threejs.org/docs/#api/en/math/Matrix4]

Local vs Global Transformations

- The local transformation matrix embodies the transformations defined on the object own coordinate system
  - .matrix attribute of a THREE.Object3D
- The global transformation matrix embodies all transformations applied to an object
  - .matrixWorld attribute of a THREE.Object3D
  - If the Object3D has no parent, it is identical to .matrix.

Local vs Global Transformations



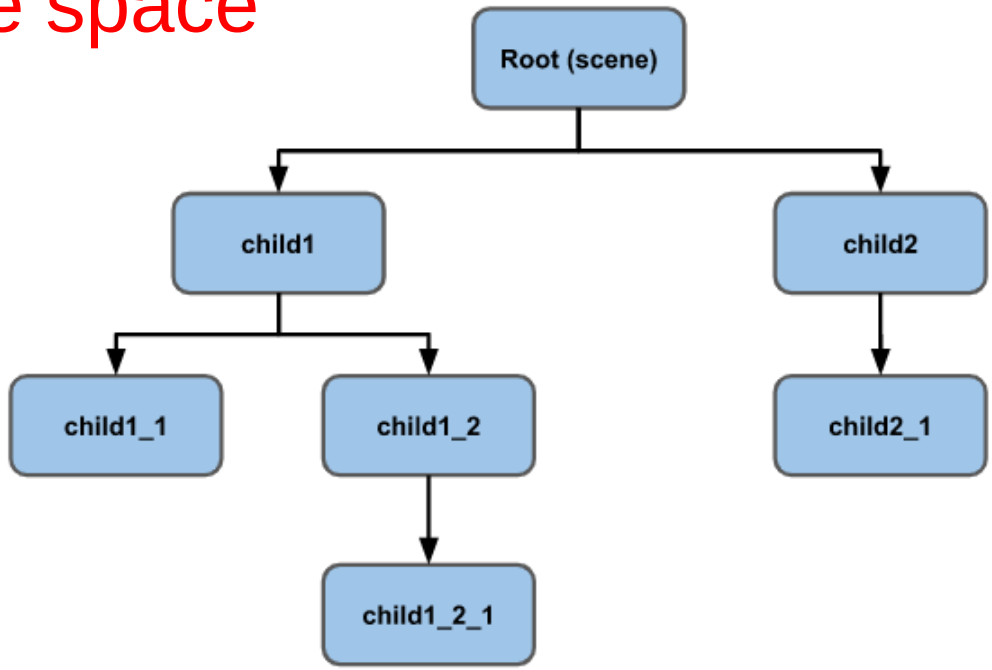


https://threejs.org/manual/examples/scenegraph-sun-earth-moon.html

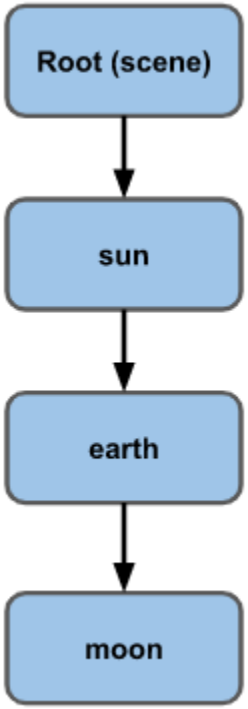
THE SCENE GRAPH

Scene Graph ?

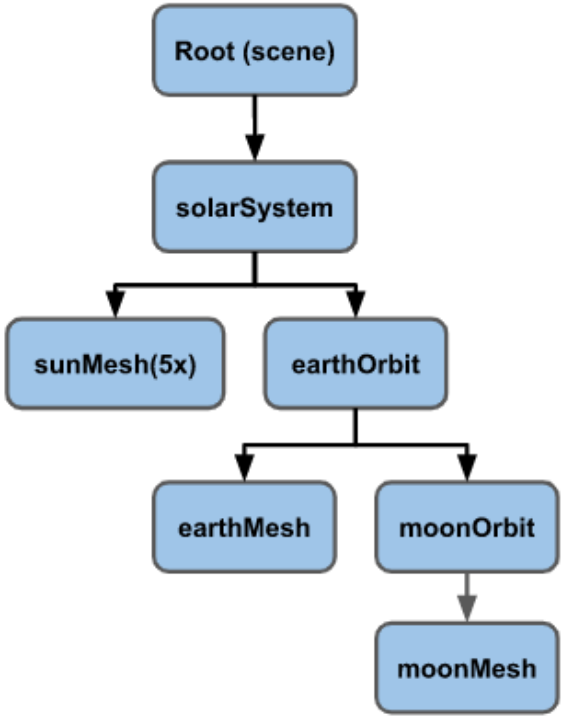
- A scene graph in a 3D engine is a hierarchy of nodes, where each node represents a local coordinate space



Simple solar system



?



The Sun

```
// an array of objects whose rotation to update
const objects = [];

// use just one sphere for everything
const radius = 1;
const widthSegments = 6;
const heightSegments = 6;
const sphereGeometry = new THREE.SphereGeometry(
    radius, widthSegments, heightSegments);

const sunMaterial = new THREE.MeshPhongMaterial({emissive: 0xFFFF00});
const sunMesh = new THREE.Mesh(sphereGeometry, sunMaterial);
sunMesh.scale.set(5, 5, 5);  // make the sun large
scene.add(sunMesh);
objects.push(sunMesh);
```

The Earth

```
const earthMaterial = new THREE.MeshPhongMaterial({color: 0x2233FF, emissive: 0x112244});
const earthMesh = new THREE.Mesh(sphereGeometry, earthMaterial);
earthMesh.position.x = 10;
scene.add(earthMesh);
objects.push(earthMesh);
```

```
objects.forEach((obj) => {
  obj.rotation.y = time;
});
```

- Sun and Earth rotate around each own YY axis
  - How to improve ?

The Solar System – Objects ?

```
+ const solarSystem = new THREE.Object3D();
+ scene.add(solarSystem);
+ objects.push(solarSystem);

const sunMaterial = new THREE.MeshPhongMaterial({emissive: 0xFFFF00});
const sunMesh = new THREE.Mesh(sphereGeometry, sunMaterial);
sunMesh.scale.set(5, 5, 5);
- scene.add(sunMesh);
+ solarSystem.add(sunMesh);
objects.push(sunMesh);

const earthMaterial = new THREE.MeshPhongMaterial({color: 0x2233FF, emissive: 0x112244});
const earthMesh = new THREE.Mesh(sphereGeometry, earthMaterial);
earthMesh.position.x = 10;
- sunMesh.add(earthMesh);
+ solarSystem.add(earthMesh);
objects.push(earthMesh);
```

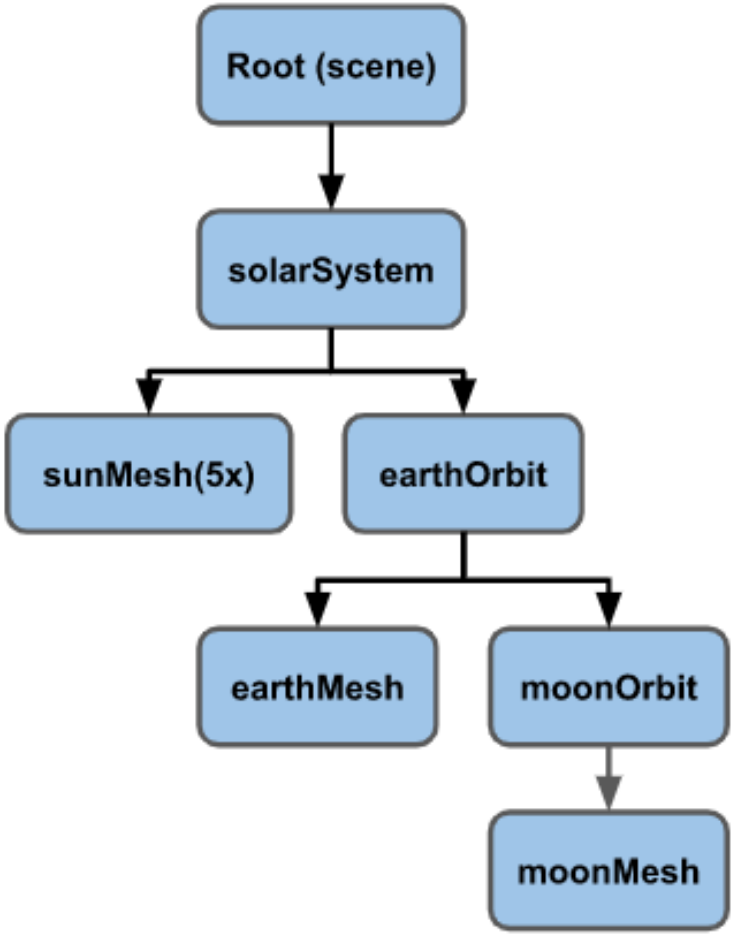
The Earth Orbit

```
+ const earthOrbit = new THREE.Object3D();
+ earthOrbit.position.x = 10;
+ solarSystem.add(earthOrbit);
+ objects.push(earthOrbit);

const earthMaterial = new THREE.MeshPhongMaterial({color: 0x2233FF, emissive: 0x112244});
const earthMesh = new THREE.Mesh(sphereGeometry, earthMaterial);
- earthMesh.position.x = 10; // note that this offset is already set in its parent's THREE
- solarSystem.add(earthMesh);
+ earthOrbit.add(earthMesh);
objects.push(earthMesh);
```

The Moon Orbit

```
+ const moonOrbit = new THREE.Object3D();
+ moonOrbit.position.x = 2;
+ earthOrbit.add(moonOrbit);

+ const moonMaterial = new THREE.MeshPhongMaterial({color: 0x888888, emissive: 0x222222});
+ const moonMesh = new THREE.Mesh(sphereGeometry, moonMaterial);
+ moonMesh.scale.set(.5, .5, .5);
+ moonOrbit.add(moonMesh);
+ objects.push(moonMesh);
```

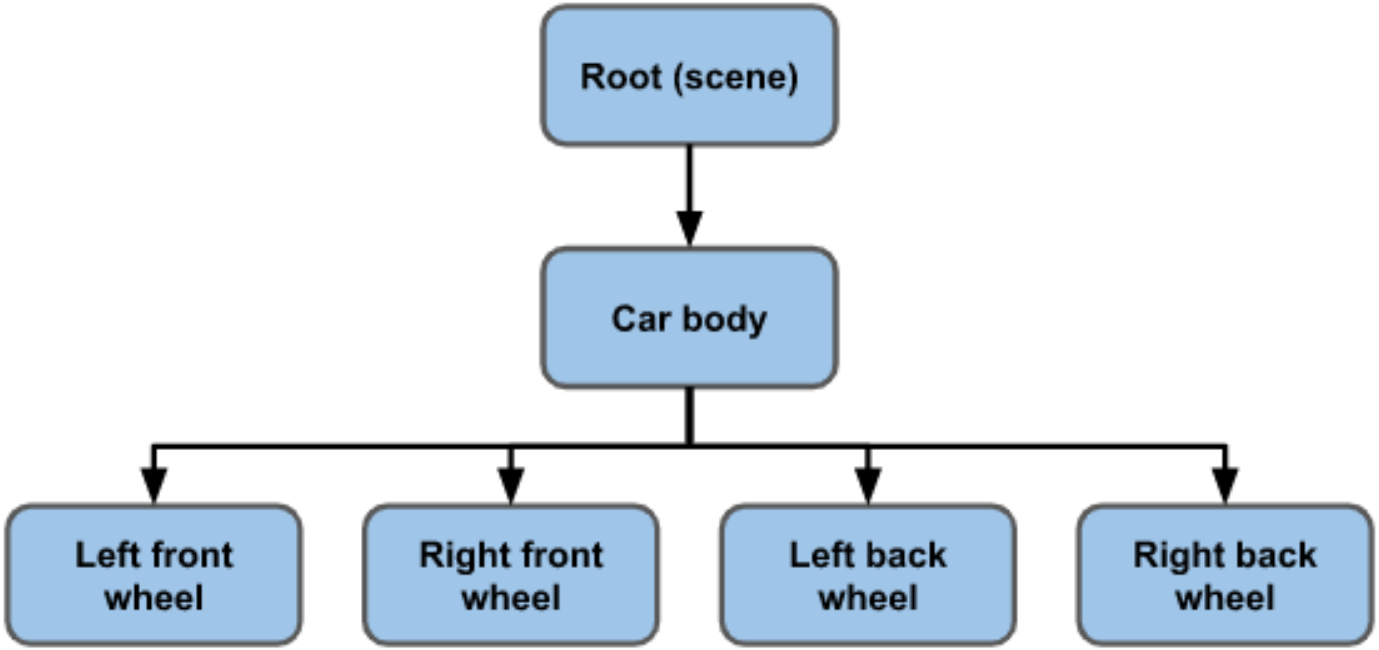
The Scene Graph



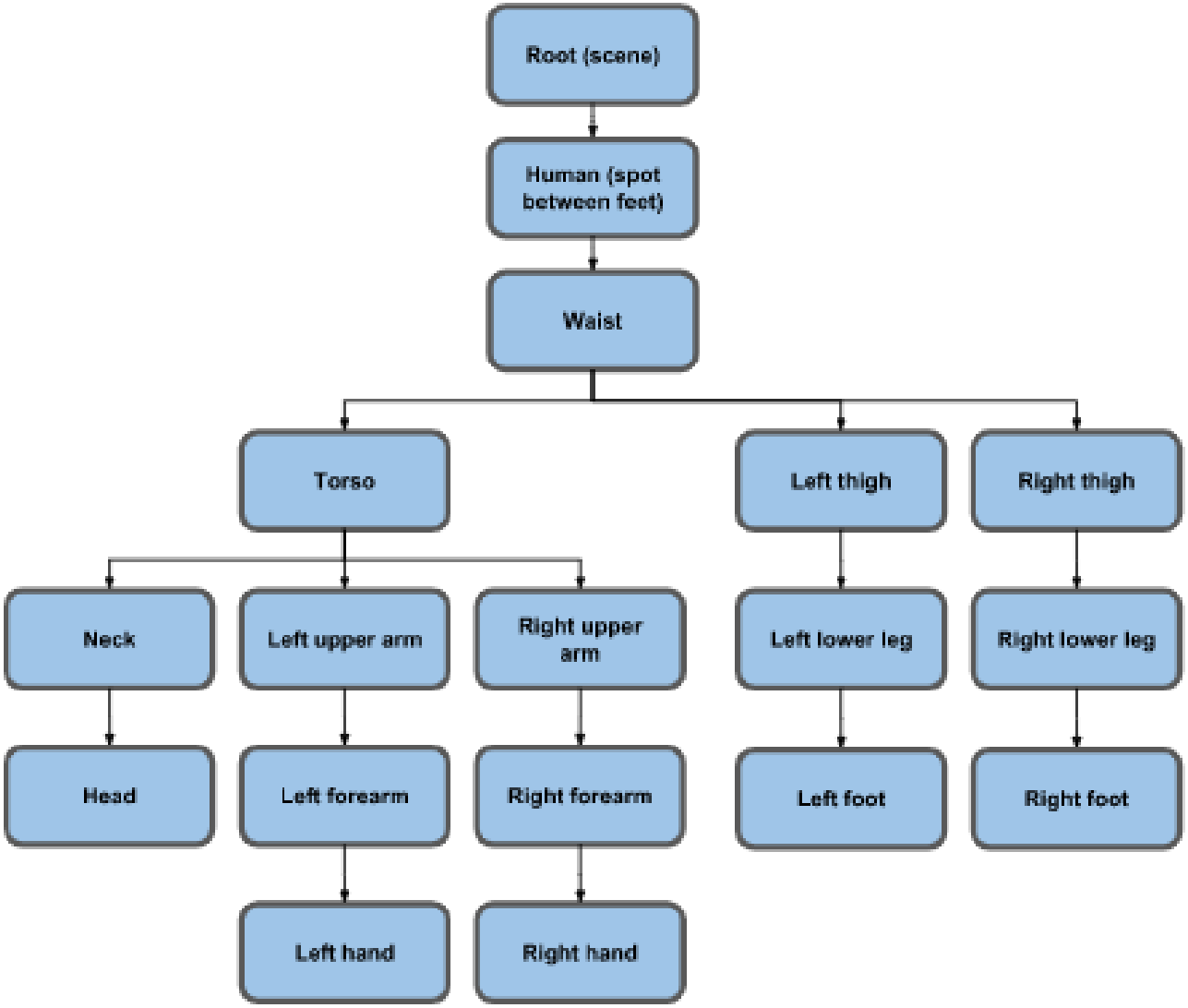
Task

- Revise the solar system example
  https://threejs.org/manual/#en/scenegraph
- See if you have any questions / difficulties !!

Another example – A car



Example – A human in a game world



Acknowledgment

- Example code and figures taken from https://threejs.org/manual/#en/scenegraph