Real-Time 3D/XR Visualization

Practical Work: The Robot in the Kitchen

March 2026

1 Introduction

In this practical work, you will consolidate your knowledge of 3D geometry, materials, lighting, and transformation hierarchies by creating a complete interactive scene. You will simulate a simplified humanoid robot performing tasks in a kitchen environment.

The final objective is to enable tele-operation of the robot's arms using VR controllers, bridging the gap between standard 3D graphics and eXtended Reality (XR) interactions.

Prerequisites

- Basic knowledge of HTML, JavaScript, and Three.js.
- Understanding of 3D coordinate systems and scene graphs.
- A WebXR-compatible browser (Chrome, Firefox, or Edge) and an XR device (or emulator).

Note: You must create a single-page application (SPA) per exercise.

2 Exercise 1: Setting up the Kitchen Environment

The first step is to create the stage for our robot. You will model a simple kitchen environment using basic geometry and physically based materials.

Tasks

1. **Scene Initialization**: Set up a basic Three.js scene with a perspective camera and a WebGL renderer. Enable shadow maps on the renderer (e.g., `renderer.shadowMap.enabled = true`).
2. **Room Structure**:
   - Create a floor using a `PlaneGeometry`. Apply a material that resembles tiles or wood.
   - Create simple walls using `BoxGeometry` or planes to enclose the space on two sides (e.g., back and left).
3. **The Workplan**:
   - Model a kitchen counter (workplan) using a `BoxGeometry`.
   - Place it in front of the camera.
   - Apply a distinct material (e.g., marble or stainless steel) to distinguish it from the floor and walls.
4. **Objects**: Place at least two interactable objects on the counter (e.g., a "Cup" using a cylinder and a "Plate" using a flattened cylinder). Give them different colors or materials.

3 Exercise 2: The Humanoid Robot (Hierarchical Modeling)

Now you will build the robot. The key challenge here is to create a correct hierarchy of meshes so that rotating a "parent" part correctly moves all "child" parts.

Concept: The Scene Graph

Recall that in Three.js, adding object B as a child of object A (`A.add(B)`) means proper transformations applied to A automatically affect B.

Tasks

1. **Root Node**: Create a simplified robot base (hips/legs or a wheeled base) and a torso using `BoxGeometry` or `CylinderGeometry`.
2. **Arm Hierarchy**: Implement at least one arm with the following hierarchy:
   - **Shoulder**: Attached to the torso. Rotates the entire arm.
   - **Upper Arm**: Attached to the shoulder.
   - **Elbow**: Attached to the lower end of the upper arm. Acts as a pivot.
   - **Forearm**: Attached to the elbow.
   - **Hand/Gripper**: Attached to the end of the forearm.
3. **Positioning**: Ensure that the pivot points (origins) of your geometries are at the joints. You may need to translate geometries within their local coordinate space or use wrapper `Group` objects to align pivots correctly.
4. **Visual Debugging**: Add an `AxesHelper` to each joint temporarily to verify the orientation of local axes.

4 Exercise 3: Lighting and Shadows

To make the scene realistic and help with depth perception (crucial for VR), accurate lighting is essential.

Tasks

1. **Ambient Light**: Add a soft `AmbientLight` to prevent pitch-black shadows.
2. **Main Light source**: Add a `SpotLight` or `PointLight` above the kitchen counter to simulate a ceiling lamp.
   - Enable shadows for this light (`castShadow = true`).
   - Adjust the shadow map size and bias to remove artifacts.
3. **Material Properties**: Ensure your materials (robot, counter, floor) are configured to react to light properly. Use `MeshStandardMaterial`.
4. **Shadow Casting/Receiving**:
   - The Robot and Objects on the table must cast shadows (`castShadow = true`).
   - The Table and Floor must receive shadows (`receiveShadow = true`).
5. **Material/lights properties controls**:
   - Add a `lil-gui` to control the material properties of the robot and objects, and the light properties.

5 Exercise 4: Animation and Interaction

A static robot is not very useful. Let's make it move.

Tasks

1. **FK Animation (Forward Kinematics)**: In your render loop, animate the rotation of the shoulder and elbow joints using `Math.sin(Date.now())` to create a smooth waving or "chopping" motion.
2. **Camera Controls**: Add `OrbitControls` to allow the user to rotate around the robot and inspect the hierarchy from different angles.
3. **GUI (Optional)**: Add a simple GUI (e.g., using `lil-gui`) to manually control the angles of the shoulder and elbow joints. This is very helpful for debugging your hierarchy.

6 Exercise 5: Tele-operation in VR/MR

This is the final and most advanced step. You will map the movement of your real hands (via VR controllers) to the robot's hands.

Tasks

1. **WebXR Setup**: Enable the 'VRButton' (or 'ARButton') in Three.js and configure the renderer for XR (`renderer.xr.enabled = true`).
2. **Controller Input**: Access the controller data.

```
const controller1 = renderer.xr.getController(0);
const controller2 = renderer.xr.getController(1);
scene.add(controller1, controller2);
```

3. **Tele-operation Mapping**:
   - **Option A (Direct Mapping)**: If the robot is free-floating or has a floating base, you can attach the robot's hands directly to the controller positions. *Note: This might break the arm if the reach is exceeded.*
   - **Option B (Remote Control)**: Use the controller's rotation/position to drive the rotation of the robot's shoulder/elbow relative to its body.
   - **Challenge (Inverse Kinematics)**: Calculate the necessary joint angles for the shoulder and elbow so that the robot's hand reaches the VR controller's position. You can use a simple 2-link IK analytical solution (Law of Cosines).
4. **Mixed Reality (Passthrough)**: If your device supports it, enable AR mode ('ARButton') with 'optionalFeatures: ['dom-overlay']'. Make the kitchen walls transparent but keep the table solid, so the robot appears to be working on your real-world table.

Submission

Submit your source code (HTML, JS, assets, etc.) as a web pages (one per exercise) and a git repository.