Individual Assignment: Build an Interactive 3D Product Viewer (Basic Mesh Edition)

Objective

Create an interactive **3D Product Viewer Application** using Three.js, where users can explore a "product" built entirely using **basic geometries** (like boxes, cylinders, spheres, etc.). The application should allow for both **manual interaction** (mouse controls) and **automated camera rotation** around the object.

This project will help you apply foundational concepts from *Discover three.js* including scene setup, mesh composition, lighting, raycasting, animation, and camera controls.

Application Features

1. Scene Setup

- **Scene Initialization**: Set up a basic Three.js scene with:
  - `PerspectiveCamera`
  - `WebGLRenderer`
  - Canvas integration in the HTML

- **Responsiveness**: Make the renderer update on window resize.
- **Controls**: Add `OrbitControls` for zoom and pan (rotation can be disabled if using auto-rotate).

2. 3D Product Creation (Using Basic Meshes)

- **Product Composition**: Build a single "product" using multiple `THREE.Mesh` objects (e.g., a chair made from boxes and cylinders).
- **Materials**: Use `MeshStandardMaterial` or `MeshPhysicalMaterial` to show realistic lighting interaction.
- **Scene Centering**: Ensure the product is centered at the origin `(0, 0, 0)` for proper camera rotation.

3. Lighting

- **Ambient Light**: Add base ambient lighting for general illumination.
- **Directional or Spot Light**: Add light(s) to create highlights and shadows on the mesh.

- **Light Positioning**: Experiment with light placement for visual balance.

4. Mouse Interaction

- **Raycasting**: Detect mouse clicks on different parts of the product.
- **Feedback**: When a part is clicked:
    - Change color or scale briefly
    - Show a small panel with the part name (e.g., “Chair leg”)
- **Highlight Effect**: Provide subtle hover or click feedback (e.g., outline or scale effect).

5. Camera Animation

- **Automatic Rotation**: Animate the camera to orbit smoothly around the product over time (use p olar coordinates or tweening):
    - The camera should rotate horizontally around the Y-axis while always looking at the product.
    - Use a consistent rotation speed (e.g., based on time elapsed).
- **User Control Override** *(optional)*: Allow mouse control to override rotation (pause auto-rotation when user interacts).

6. Animation Loop

- **Render Loop**: Use `requestAnimationFrame` for smooth rendering.
- **Mesh Animations**: Optionally add floating or pulsing effects to the product for added life.

7. Code Structure

- Organize into modular files or functions:tg
  G
    - `initScene.js`
    - `createProduct.js`
    - `addLighting.js`
    - `interaction.js`
    - `cameraAnimation.js`
- Use clear, concise comments.
- Structure assetslogically (e.g., separate folders for scripts, styles, textures if used).