Technical Assessment: Senior 3D Graphics Engineer (Electronic PCB Editor)

Project Overview

Your objective is to build a high-performance **3D PCB (Printed Circuit Board) Viewer & Editor core** using **React** and **Vanilla Three.js**. The tool must handle layered geometry, performant rendering of repetitive components, and precise interaction for board inspection.

Time Limit: 24 Hours
Technical Constraint: React Three Fiber (R3F) is strictly prohibited. All Three.js logic must be handled imperatively to demonstrate deep engine knowledge.

---

1. Engine Foundation & Layering

- **Manual Integration:** Initialize the Three.js environment (Renderer, Scene, Camera) inside a React lifecycle.
- **The Board Substrate:** Create a parametric "Board" (BoxGeometry) representing the FR4 substrate.
- **Layer Management:** Implement a system that renders objects on specific layers (e.g., **Top Copper**, **Bottom Copper**).
    - *Requirement:* Ensure no Z-fighting between the board surface and the copper elements using polygonOffset or discrete Z-spacing.
-
- **Resource Disposal:** Implement a strict cleanup routine. Deleting a component or unmounting the app must call .dispose() on all geometries and materials.

2. PCB Primitive Creation

Implement a system to generate and manage the following "Electronic Primitives":

- **SMD Pads (Instanced):** Use InstancedMesh to render 100+ rectangular or circular copper pads.
- **Through-Holes:** Render cylindrical holes that "drill" through the board.
- **Traces (Paths):** Implement a method to render a "Trace" (conductive path) between two points.

    - *Requirement:* Traces must have a defined width and be rendered as flat, manifold geometry on the board surface (not simple GL_LINES).
-

3. Custom Shader & Visual Style

- **Material System:** All copper elements must use a custom ShaderMaterial.
- **Face & Edge Rendering:**
    - **Faces:** Implement a GLSL shader that simulates a "Brushed Copper" effect or a procedural "Solder Mask" green tint.
    - **Edges:** Use EdgesGeometry or a barycentric wireframe shader to render distinct outlines for every pad and trace.
-
- **Interaction Uniforms:** The shader must support a uHovered and uSelected state. When a user hovers over a component, the shader must highlight the object (e.g., an emissive pulse or color shift).

4. Interaction & Precision Picking

- **Raycasting:** Implement a Raycaster to detect interaction with pads and traces.
- **Hover State:** Moving the mouse over a pad must trigger the "Edge Highlight" and change the cursor to a pointer.
- **Selection & Transformation:**
    - Clicking a pad selects it and attaches TransformControls for movement along the XZ plane.
    - **Dynamic Data:** Display the selected component's **World Coordinates** and **Surface Area** in a React-based sidebar.
-

5. Persistence & Schema

Serialization: Export the PCB layout to a JSON format.

```
 code JSON
downloadcontent_copy
expand_less
   {
  "board": { "width": 100, "height": 80, "thickness": 1.6 },
  "components": [
    { "id": "pad_1", "type": "smd_rect", "pos": [10, 0, 5], "size": [2, 4], "layer": "top" },
    { "id": "trace_1", "type": "path", "points": [[0,0], [10,10]], "width": 0.5 }
  ]
}
```

-
- **Hydration:** The application must be able to "Load" this JSON and reconstruct the entire 3D board state perfectly.

Submission Requirements

1. **Git Repository:** Clean, modular code. (Separation of Engine, Shaders, and React Components).
2. **README.md:**
    - Detailed explanation of your **performance strategy** for rendering many small pads (e.g., Instancing).
    - Documentation of your **Z-fighting mitigation** strategy.
3.
4. **Deployment:** A live URL for the reviewer to test the interactions.

Measurable Evaluation Criteria

Feature	Success Metric
Vanilla Integration	Efficient use of useRef and useEffect with a single requestAnimationFrame loop.
Performance	Use of InstancedMesh for pads to minimize draw calls.
Visual Fidelity	Clear distinction between Edges and Faces; no flickering (Z-fighting) on the board surface.
Memory Management	renderer.info.memory returns to baseline after clearing the board.
Shader Logic	Successful implementation of hover/selection logic via GLSL uniforms rather than material swapping.
State Sync	React UI accurately reflects the 3D position of components during transformation.