Cuboid Grouping and Visualization at Runtime in Web Browser

🎯 Task Objective

Build a **runtime WebGL 2 or WebGPU visualization** of cuboid groups defined in a CSV file. This test assesses your ability to handle geometry, rendering, grouping logic, and Web graphics programming in the browser.

💡 **Setup Tip:** Since you are coming from a native development background, we highly recommend using **Vite** with **TypeScript** to skip the configuration boilerplate.

You can set up the project in 1 minute using this command:

```
npm create vite@latest my-cuboid-project -- --template vanilla-ts
```

📂 Input CSV File Format

The provided **CSV file** contains cuboid definitions with the following structure:

```
id,x1,y1,z1,x2,y2,z2
1,0,0,0,2,3,4
2,2,3,0,5,5,4
3,10,10,10,15,15,15
```

Where:

- **id** – Unique identifier for the cuboid.
- **(x1, y1, z1)** – Start point of the diagonal.
- **(x2, y2, z2)** – End point of the diagonal.

Assumptions:

- All cuboids are **axis-aligned**.
- Coordinates are **natural numbers**.
- **x1 < x2**, **y1 < y2**, **z1 < z2**.
- The provided dataset is small (5k items). In your solution, please assume the target dataset could be **100x larger (500k items)**. Design your architecture to handle that scale, or explain in README how you would scale it.

📊 Task Details

Grouping Cuboids by Adjacency

- **Group Definition:**
  Cuboids belong to the same group if they **touch at least one face** with another cuboid in the group.
  Touching at **corners or edges** does **not** count.
- **Grouping Rules:**
  - Groups must contain **at least two cuboids**.
  - **Isolated cuboids** (no touching cuboids) are **discarded**.

Visualization Requirements

- Load the CSV file **at runtime** (e.g. drag & drop or file input).
- Display all **valid cuboid groups** in 3D.
- Each group must use a **distinct material**, not just a color.
- Include **camera controls** (orbit, zoom, pan).
- Target **smooth performance.**.

⚙️ Technical Constraints

- Must run fully in the **browser**.
- Use either **WebGL 2** or **WebGPU** – your choice. Justify in the README.
- You **can use a framework** like Three.js or Babylon.js.
- **Custom shader material requirement**: write your own fragment (and optionally vertex) shader code for at least one material.
- You must include and use **at least one texture image asset**.

📦 Deliverables

- ✅ Complete working project in TypeScript (preferred), modern JavaScript (ES Modules) or WASM.
- ✅ README.md including:
  - Setup & run instructions
  - Explanation of your **grouping algorithm**
  - Description of the **overall architecture and code structure**
  - Justification for using WebGL 2 or WebGPU
  - Notes on performance, optimizations, or limitations
- ✅ Working build via npm run dev or link to GitHub Pages.
