Ok what's goin on in here

Alright, here's **what's going on in here**:

This is your **InteractiveBoard.tsx** React component for your **Xiangqi board**, and I can see:

---

1. Board rendering is grid-centric (10x9):

- **Cells** are **50x50px**, rendered using **flexbox rows and columns**.
- **Pieces** are **centered on the intersections** (via `` `position: absolute` ``, `` `transform: translate(-50%, -50%)` ``).

  Translation: You're working from a **vertex-based** coordinate system, where **pieces sit on intersections**, not in grid cells.

---

2. Palace diagonals are absolutely positioned:

- Using **divs with borders and background gradients**.
- **Top palace**:
  - Positioned at **top: 0px, left: 150px**.
  - Width/height: **100x100px**.
- **Bottom palace**:
  - Positioned at **top: 350px, left: 150px**.

  Translation:

- The **palace overlay** is **anchored to specific pixel coordinates**,
- But the **board grid is dynamic** (centered, flexbox-driven).
- So if **board layout flexes** (screen size, margins), the **palace doesn't comigrate**.

---

3. Why the palace and board don't sync:

- The **board grid** is **CSS-driven** (flexbox rows + 50px cells).
- The **palace overlay** is **absolutely positioned relative to a parent div with fixed dimensions** (`` `450px x 500px` ``).
- If the **board grid moves** (due to centering, margins, etc.), the **palace stays locked** at **absolute pixel positions**.

How to fix / anchor them together (conceptually glue the objects):

- Instead of **fixed pixel positioning** for the palace (`` `top: 0px; left: 150px` ``),

- **Calculate palace position based on cell size and grid offsets**.

Pseudo-fix:

```tsx
`const cellSize = 50; // Existing grid cell size

// Palace starts at (row: 0, col: 3)
const palaceTop = 0 * cellSize;
const palaceLeft = 3 * cellSize;
const palaceSize = 3 * cellSize; // 3x3 cells

<div style={{
  position: 'absolute',
  top: `${palaceTop}px`,
  left: `${palaceLeft}px`,
  width: `${palaceSize}px`,
  height: `${palaceSize}px`,
  border: '1px solid red',
}}>
  {/* diagonals */}
</div>
`
```

Why this works:

- Palace **moves with the grid** because it **derives its position from the board's cell size**.
- Any **changes to cell size or board offset** propagate to the **palace overlay**.

Want me to **refactor the palace logic** directly,
or build a **utility function** that **auto-calculates palace positions** based on **board dimensions**? 😎