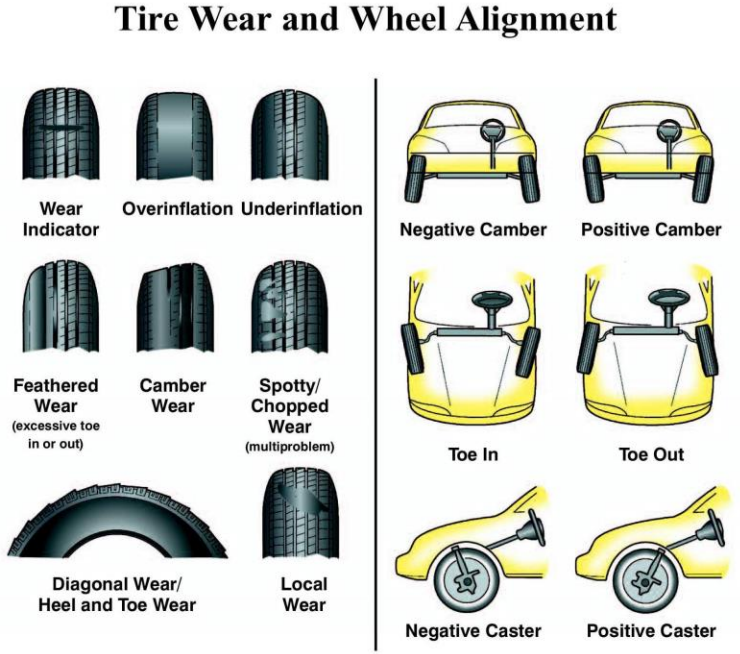
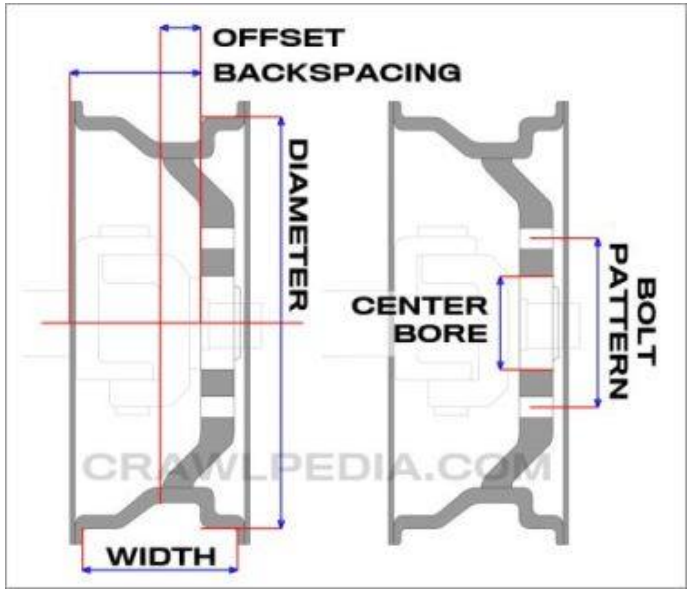
Miata Fitment

Anthony Devito

Car defintions

- Camber: Tilt around X (wheel leans in/out).
- Toe: Yaw around Z (wheel points in/out).
- Caster: Shifts wheel along X slightly forvisual/fitment alignment.
- Offset: Mounting face vs wheel centerline; affects Z position.
- Spacer: Adds to offset outward; Z position change.
- Ride height: Y position of car.





Backstory on the problem at hand

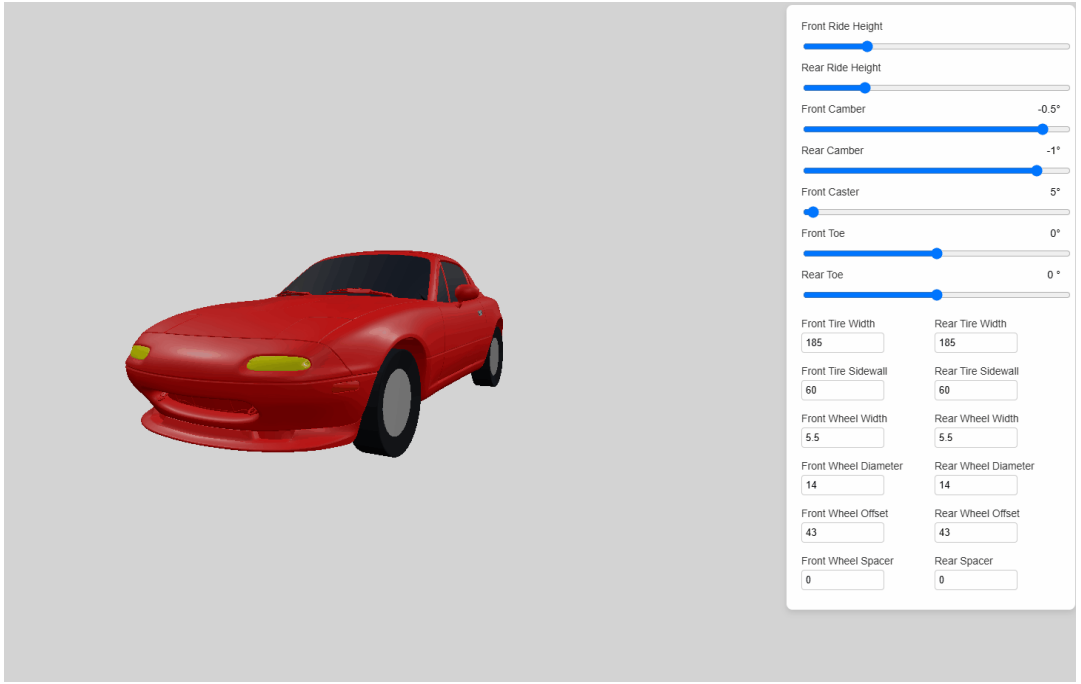
- Cars such as the Mazda Miata are super popular to modify
  - Spec Miata is the fastest growing racing class in the US
  - 1.2M Miatas sold since 1989, still one of the most popular sports cars
  - 87.5% of surveyed Miata owners modify their cars
- One of the most common modification are wheels, tires and suspension (coilovers, alignments, arms, bushings, etc)
  - The big problem lies here!

How a market was found

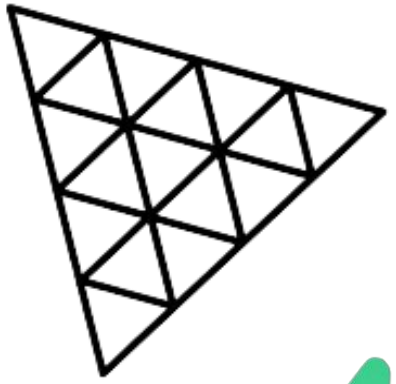
- One of the most common questions on Miata Forums: "Will this wheel setup fit my car?"
  - Technically anything fits with enough hammering and negative camber
- Because of this, countless threads, guides, and videos exist trying to answer the same question
  - One ClubRoadster fitment thread alone has 2.2 million view and 7k comments
- Traditionally, enthusiasts rely on photos and willtheyfit.com, but what if there were a better, more visual way

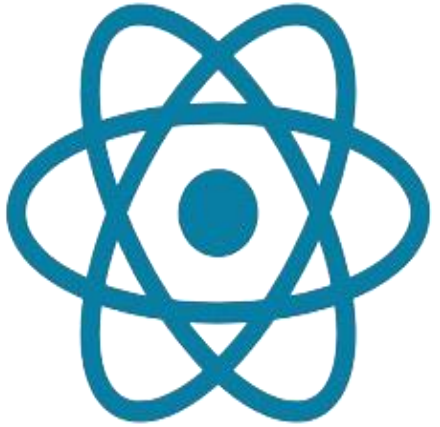
What Miata Fitment is and what it solves

- Miata Fitment is a website that allows users to test fit any different wheel, tire, and suspension setup in a 3D visualizer
- What users can do:
  - Select their Miata Model
  - **Input exact wheel specs** – diameter, width, offset, bolt pattern
  - **Input tire specs**– sidewall height, stretch, and width
  - **Adjust suspension** – ride height and camber
  - See everything rendered in **real time** – 360° rotation, zoom, and camera angles
  - Make changes instantly to find their perfect setup



Overview of the techstack





- React
  - UseEffects, useStates, and useRefs to control how the wheels, tires, and suspension gets updated
  - Renders the UI
- ThreeJS
  - Renders the car, wheels, and tires
  - Controls scene variables such as lighting, and the camera
- SupaBase
  - Database, auth, other backend functions
- Vercel
  - Hosting

Technical Definitions

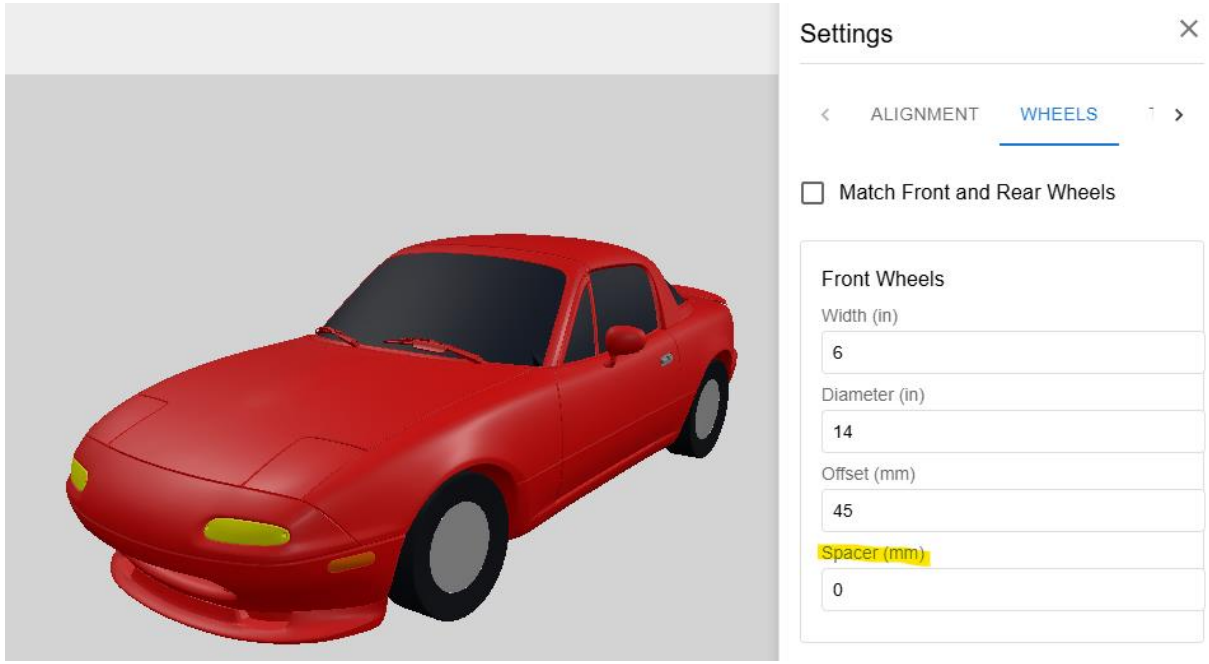
- UseState
  - Declarative UI state that triggers re-renders
- UseEffect
  - Side effects that run after render to sync React state to Three.js
- UseRef
  - Mutable, persistent handles that do not cause re-renders
- ThreeJS
  - Scene: Container for all objects you render.
  - Renderer (WebGLRenderer): Draws the scene to the canvas each frame.
  - Camera (PerspectiveCamera): Viewpoint used to render the scene.
  - Controls (OrbitControls): Mouse/touch camera navigation.
  - Object3D/Mesh: Things in the scene; position/rotation/scale mutate these.
  - Lighting: Makes materials visible with shading (ambient, directional, etc.).

How the 3D simulator works

- React state drives the simulator
  - MainComponent holds settings and currentModel. Changing settings re-renders React and triggers effects that mutate the Three.js scene via refs.
- Three.js scene lifecycle
  - On mount: create Scene, Renderer, Camera/Controls, Lighting; build Car + Wheels/Tires; start an animation loop that renders every frame.
  - On model change: remove the old car and add the selected generation.
  - On settings change: update settings and rebuild wheel/tire meshes when sizes change.

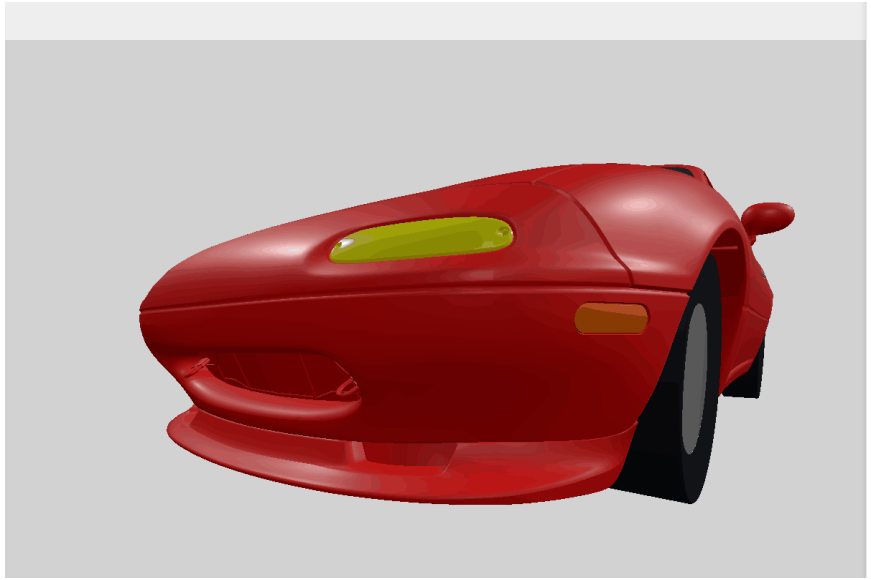
What happens when you change a setting

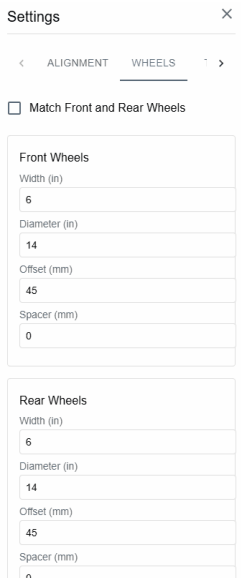
- What happens when a user changes the front wheel spacer value
  - Super common adjustment
  - This demonstrates React state management, Three.js scene updates, and real-time 3D rendering
- Let's follow the data flow from user input to visual update

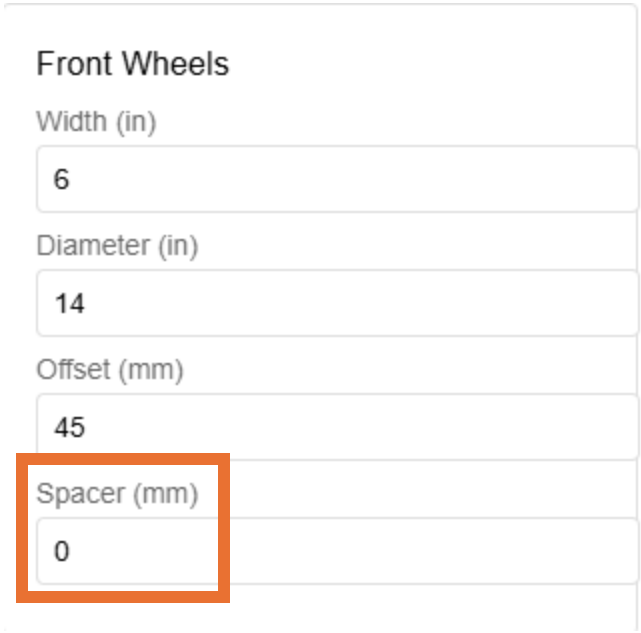


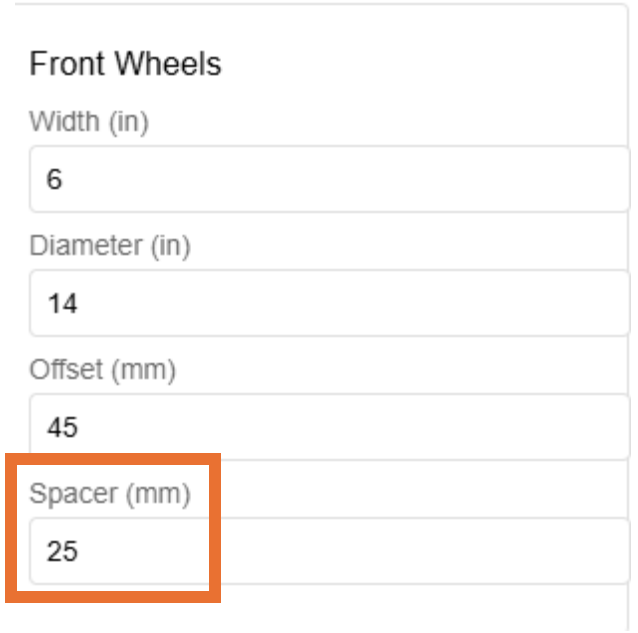
What happens when you change a setting

- User Inputs spacer value
  - 25mm







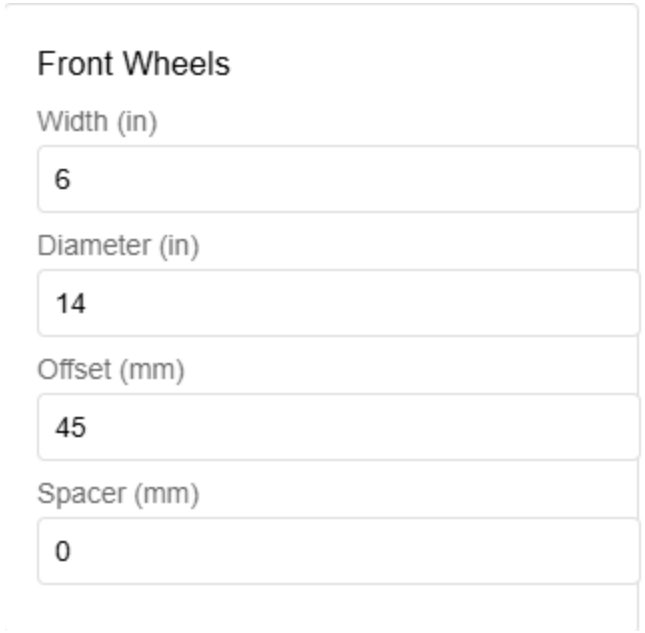


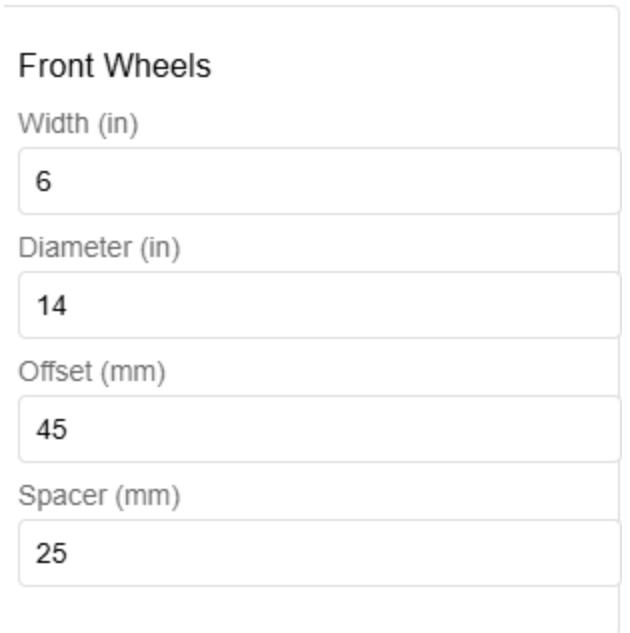

```
<StyledLabel>Spacer (mm)</StyledLabel>
<StyledInput
  type="number"
  value={frontWheelSpacer}
  onChange={(e) => setFrontWheelSpacer(parseFloat(e.target.value))}
/>
```

25

What happens when you change a setting

- State update
  - UpdateModel
  - SetSettings is a useState





{FrontWheelSpacer: 25}

```
const updateModel = useCallback((newSettings: Partial<Settings>) => {
  setSettings((prev) => ({ ...prev, ...newSettings }));
}, []);
```

What happens when you change a setting

- UseEffect triggers an update
  - Updates wheel position on all four corners

```
useEffect(() => {
  if (sceneRef.current && wheelRefs.current.length > 0) {
    updateWheelPosition(
      wheelRefs.current[0],
      tireRefs.current[0],
      "FL",
      settings
    );
    updateWheelPosition(
      wheelRefs.current[1],
      tireRefs.current[1],
      "BL",
      settings
    );
    updateWheelPosition(
      wheelRefs.current[2],
      tireRefs.current[2],
      "BR",
      settings
    );
    updateWheelPosition(
      wheelRefs.current[3],
      tireRefs.current[3],
      "FR",
      settings
    );
    updateWheelAndTireSizes(settings);
  }
}, [settings, updateWheelPosition, updateWheelAndTireSizes]);
```

What happens when you change a setting

- Calculating Spacer offset
  - Extract spacer value

```
case WheelPosition.FRONT_LEFT:
  camberDeg = settings.frontCamber;
  offset = -mmToFeet(settings.frontWheelOffset); // Convert mm to feet
  spacer = mmToFeet(settings.frontWheelSpacer); // Convert mm to feet
  toe = settings.frontToe;
  baseX =
    WHEEL_POSITIONS.FRONT.LEFT.x +
    settings.frontCaster / WHEEL_POSITIONS.FRONT.LEFT.casterOffset;
  baseZ = WHEEL_POSITIONS.FRONT.LEFT.z;
  rideY = settings.rideHeightFront;
  break;
```

```
case WheelPosition.FRONT_RIGHT:
  camberDeg = settings.frontCamber;
  offset = mmToFeet(settings.frontWheelOffset); // Convert mm to feet
  spacer = -mmToFeet(settings.frontWheelSpacer)  // Convert mm to feet
  toe = -settings.frontToe;
  baseX =
    WHEEL_POSITIONS.FRONT.RIGHT.x +
    settings.frontCaster / WHEEL_POSITIONS.FRONT.RIGHT.casterOffset;
  baseZ = WHEEL_POSITIONS.FRONT.RIGHT.z;
  rideY = settings.rideHeightFront;
  break;
```

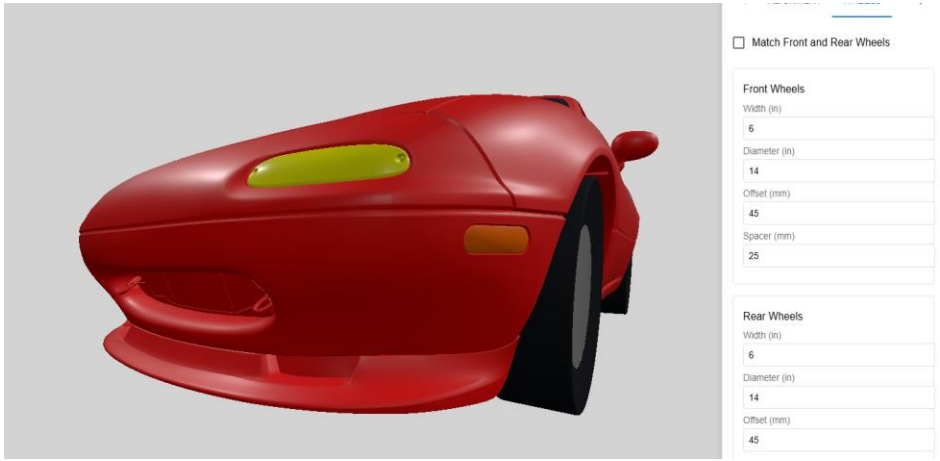
What happens when you change a setting

- Applying updated position
  - Add all variables that could effect the Z position
  - Set for both wheels and tires

```
const camberRad = (Math.min(Math.max(camberDeg, -20), 1) * Math.PI) / 180;
const toeRadiusComp =
  (rollingDiameter(
    position.startsWith("F")
      ? settings.frontWheelDiameter
      : settings.rearWheelDiameter,
    position.startsWith("F")
      ? settings.frontTireWidth
      : settings.rearTireWidth,
    position.startsWith("F")
      ? settings.frontTireSidewall
      : settings.rearTireSidewall
  ) * <- #103-113 rollingDiameter
    Math.sin(toe)) / <- #103-114 const toeRadiusComp =
  12;

const zPos = baseZ + offset + spacer;

const rotX = Math.PI / 2 + camberRad;
const rotZ = toeRadiusComp;

wheel.rotation.set(rotX, 0, rotZ);
wheel.position.set(baseX, rideY, zPos);

tire.rotation.set(rotX, 0, rotZ);
tire.position.set(baseX, rideY, zPos)
```

What happens when you change a setting

- ThreeJS rendering loop
  - Renders the new scene



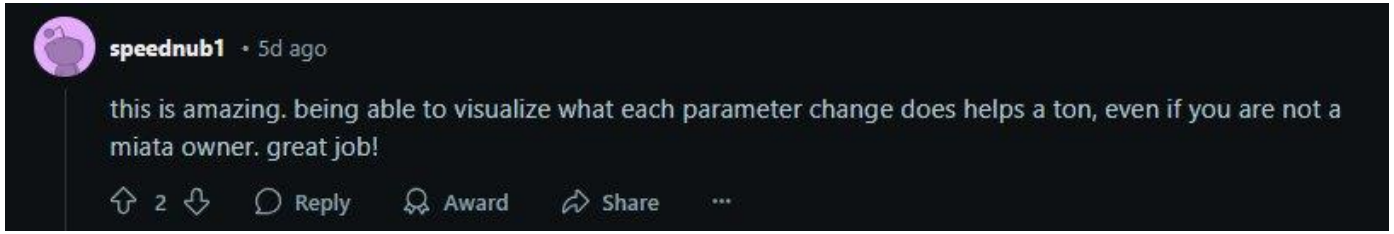

```
const animateLoop = () => {
  requestAnimationFrame(animateLoop);
  if (controlsRef.current) {
    controlsRef.current.update();
  }
  renderer.render(scene, camera);
}; <- #379-385 const animateLoop = ()
animateLoop();
```

Some random other tidbits

- Material UI
  - Easy to work with
  - Similar to BlueX
- SupaBase
  - Postgres + Row-Level Security
  - Easy auth
  - Batteries included
  - Cost
- Vercel
  - It fits my stack well
  - Zero-config
  - Cool features

Community reaction/ feedback

- Super positive feedback from the community
- Lots of constructive feedback
- ~1.8k unique vistors
  - 13% repeat visitors
  - 47 countries
- Investment/ buyout offers
- $2,000 in prizes



Future plans

- Recently added another developer to help
- UI has so many bugs that need to get fixed
- Redo the visualizers UI… again
  - Better support for Mobile
  - Make it a little easier to work with on desktop
- Add physics simulations
- Add the remaining three Miata models
  - Second gen is almost done being modeled
- Implement paywalls
- Sell stickers!!

Finish

- Thanks for listening