THE FULL BUILD GUIDE // @HINUMPY

Build Your Own Human Atlas

A 3D anatomy explorer you can build with an AI coding agent this weekend

Time: 4 to 8 hours **Difficulty:** Intermediate **Stack:** React, TypeScript, Three.js

This is the companion to my reel. Human Atlas is an interactive 3D anatomy explorer that feels like a digital museum. You can rotate the body, toggle 8 anatomical systems, search any structure, and drag one slider to pull the entire body apart into an organized collection of real 3D specimens.

I saw a version of this on Twitter and wanted to see if I could remake it. I built mine with Codex, but the master prompt in this guide works with any AI coding agent, including Claude Code and Cursor.

Read the setup steps first, then paste the prompt. Do not skip Step 2. The real anatomy data is what makes this project look professional.

What You Are Building

- **A full screen 3D human body** you can orbit, pan, and zoom with smooth, damped camera movement
- **8 toggleable systems:** skeleton, muscles, heart, sensory organs, arteries, veins, nervous system, and respiratory
- **An explode slider** that moves every piece from the assembled body into a giant specimen atlas, with every piece still clickable
- **Hover labels and an info panel** for every structure, plus an isolate mode
- **Search** that flies the camera straight to any structure
- **Regional collections** for the hand, shoulder, thigh, foot, heart, and brain

What You Need

- **Node.js 20 or newer.** Check with node -v in your terminal.
- **An AI coding agent.** Codex, Claude Code, or Cursor all work.

- **A code editor.** VS Code is perfect.
- **The BodyParts3D dataset.** Free, open 3D anatomy models. Step 2 shows you where to get it.
- **A GitHub and Vercel account** if you want a live link to share (both free).

Step 1: Set Up the Project

Your agent can do this for you, but running it yourself takes two minutes and makes sure the project lives where you expect.

```
npm create vite@latest human-atlas -- --template react-ts
cd human-atlas
npm install
npm install three @react-three/fiber @react-three/drei framer-motion
npm install -D @types/three
npm run dev
```

If a page opens at localhost, you are ready. Create a folder at assets/anatomy/raw inside the project for the next step.

Step 2: Download the Anatomy Models

Every piece of anatomy in Human Atlas comes from BodyParts3D, an open 3D anatomy database built from full body MRI data by the Database Center for Life Science in Japan.

1. Go to the official download page: dbarchive.biosciencedbc.jp/en/bodyparts3d/download.html
2. Download the latest OBJ archive. Pick the version with **99% polygon reduction**. The files are much smaller and load faster in a browser, and they still look great.
3. Also download the **name and relationship tables** from the same archive. The OBJ files are named by ID number, so these tables are how your app knows that a file is the "Right Gracilis".
4. Unzip everything into assets/anatomy/raw.

> **Attribution Is Required**

BodyParts3D is free to use, but its license requires credit. The current license page lists this attribution:

"BodyParts3D, © The Database Center for Life Science licensed under CC Attribution 4.0 International"

Put it in your app's credits panel and your README. Check the license page before you publish in case the wording has changed.

Keep Raw Files Out of Git

The raw OBJ files are big. Add assets/anatomy/raw to your .gitignore and only commit the compressed GLB files your build script creates. GitHub blocks individual files over 100 MB.

Step 3: Paste the Master Prompt

Open your project folder in your AI coding agent and paste this entire prompt. It is long on purpose. Specific prompts get specific results, and vague prompts get generic Three.js demos.

Tip: if you have screenshots of a design you like, attach them with the prompt as visual reference.

```
Build a polished, interactive web app called Human Atlas: a full screen 3D human anatomy
explorer that feels like a digital museum or scientific atlas.

Use the written spec below as the source of truth. If I attach screenshots, use them as
visual reference for spacing, proportions, and feel.

Do not build a generic medical dashboard. Do not build a normal webpage with a 3D model in
the middle. The anatomy visualization IS the page, with a minimal floating UI on top of it.

Create the project in a new folder called human-atlas.

================================================
1. STACK
================================================
- React + TypeScript + Vite
- three, @react-three/fiber, @react-three/drei
- Framer Motion for interface animation
- Keep all per frame animation inside the render loop (useFrame). Never update React state
every frame.
```

```
================================================
2. OVERALL EXPERIENCE
================================================
Full screen canvas (100vw x 100vh). No navbar, no hero section, no card grids, no gradients,
no marketing copy.

It should feel: scientific, editorial, museum like, premium, minimal, spacious, slightly
futuristic but NOT sci fi.

Floating UI elements:
1. Branding, upper left
2. Systems panel, left
3. Specimen collections panel, left, below Systems
4. Search, upper right
5. Slim vertical camera toolbar, right edge
6. Explode anatomy control, bottom center
7. Information panel when a structure is selected
8. Tiny hint text along the bottom edge ("Drag to orbit · Scroll to zoom · Tap to inspect")
and a "Sources & credits" link bottom right

================================================
3. VISUAL STYLE
================================================
- Background: pale warm off white, around #F3F3F1. Never pure white.
- Panels: near white, 10 to 14px radius, border 1px solid rgba(0,0,0,0.05), shadow 0 8px
30px rgba(0,0,0,0.04), generous whitespace.
- Primary dark color: desaturated navy/charcoal around #13242D. Use it for active buttons,
toggles, tooltips, and primary actions.
- Type: Inter (or a similar neutral sans serif). Small, dark charcoal, understated. No
futuristic fonts.

================================================
4. BRANDING (upper left, about 24px from the edges, NOT in a card)
================================================
- Tiny letter spaced eyebrow: INTERACTIVE ANATOMY
- Title: Human Atlas, semibold, 22 to 28px, with a small "3D" superscript
- Metadata line: "[N] modeled pieces · BodyParts3D"

================================================
5. CENTRAL MODEL
================================================
- The human stands in anatomical position, centered, with lots of negative space around it.
- Soft neutral studio lighting, subtle ambient occlusion, no dramatic shadows. It should
never look like a game character.
- Perspective camera with OrbitControls: enableDamping = true, dampingFactor about 0.05.
- Orbit, zoom, and pan. Every camera move interpolates. Nothing snaps.

================================================
6. ANATOMY ASSETS (BodyParts3D)
================================================
First inspect the repo for .glb, .gltf, or .obj files and reuse them.

I downloaded the BodyParts3D OBJ files (Database Center for Life Science) into
/assets/anatomy/raw, along with the name and relationship tables from the same archive. Each
OBJ file is named by its element ID.

Write a build script (scripts/build-anatomy.ts) that:
```

```
- reads every OBJ file
- maps each ID to its English name using the tables
- assigns each piece to one of 8 systems: skeleton, muscles, heart, sensory organs,
arteries, veins, nervous system, respiratory
- detects side (left, right, midline) from the name
- converts and compresses the meshes to GLB (gltf-transform with meshopt or draco)
- writes a manifest JSON

Manifest entry shape:
{ id, name, system, region, side, meshFile, assembledPosition, explodedPosition }

Log any piece the script cannot classify to a file instead of guessing.

If assets are missing, build the full architecture with clearly isolated placeholder
geometry and a documented loader so real GLB files can be dropped in later. Do not hand
model anatomy.

================================================
7. SYSTEMS PANEL
================================================
Floating card: left about 24px, top about 140px, width 170 to 200px. It is a small detached
tool palette, never a full height sidebar.

- Title "Systems" with a small "8/8" count badge
- Segmented control: All | Skeleton | Organs (presets)
- One row per system: small colored bullet, name, piece count, tiny toggle on the right
- Toggles: dark when on, light gray when off, thumb animates smoothly
- Rows 27 to 34px tall, tiny compact type
- Footer: "[N] pieces visible" and a "Hide all" text button
- Systems fade in and out (opacity, 200 to 400ms) with a very slight stagger when several
change at once

================================================
8. SPECIMEN COLLECTIONS PANEL (below Systems)
================================================
- Title "Specimen collections" with a collapse chevron, subtitle "Study the body, region by
region."
- 2 column grid of region buttons with piece counts: Hand, Shoulder, Thigh, Foot, Heart,
Brain
- Segmented control: Both | Left | Right
- Selecting a region frames the camera on it and lays that region out as a specimen sheet

================================================
9. EXPLODED ANATOMY (THE MOST IMPORTANT FEATURE)
================================================
- Every mesh stores an assembledPosition and an explodedPosition.
- explodedPosition comes from an organized atlas layout: a huge specimen sheet with pieces
in catalogued rows. Larger structures toward the top, smaller structures denser toward the
bottom. It can span several screen widths.
- Real 3D geometry only, never a flat image. Every piece stays hoverable, selectable,
focusable, and isolatable.
- The camera pulls back as the explode value increases.
- Interpolate with easeInOutCubic (or a low bounce spring). Full transitions take 1 to 2
seconds. Collapsing animates every piece back to its anatomical position.

================================================
10. EXPLODE CONTROL (bottom center)
```

```
================================================
Floating card: bottom about 24px, centered, width 300 to 440px.
- Small icon, label "Explode anatomy", percentage readout on the right
- Thin continuous slider. 0% = "Assembled" (label under the left end). 100% = "Every piece"
(label under the right end).
- Small Reset button separated by a thin vertical divider
- Tiny caption above the card: ANATOMICAL POSITION · FRONT

================================================
11. HOVER AND SELECTION
================================================
- Raycast every mesh.
- On hover: subtle highlight, pointer cursor, small dark navy tooltip with the structure
name and its category under it (for example "Long Head Of Right Biceps Brachii" / "Muscle").
- On click: emphasize the structure, slightly dim everything else, open the information
panel, and focus the camera on it.

================================================
12. INFORMATION PANEL
================================================
Floating card on the right, width 260 to 320px.
- Tiny category label (Bone, Muscle, Artery)
- Large bold structure name
- 2 to 4 lines of description
- Metadata rows with thin dividers: ID, System, Region, Side
- Relationships: for muscles with verified attachment data, list clickable attachment bones
and supporting vessels. Bones link back to their attached muscles. Label anything that is
only nearby as "nearby", never as a verified connection. Never invent anatomical
relationships.
- Full width dark "Isolate structure" button with an icon, plus a small "Clear selection"
text button
- On small screens this becomes a compact bottom sheet

================================================
13. ISOLATE MODE
================================================
Keep the selected structure visible, fade everything else, move the camera to center it, and
allow orbiting around it. Clearing restores everything with smooth transitions.

================================================
14. CAMERA TOOLBAR
================================================
Right edge about 20px, vertically centered, 36 to 42px wide, rounded container.
Small square icon buttons: zoom in, front view (F), zoom out, orbit mode, pan mode, reset
camera, fullscreen.
The active tool uses the dark navy background.

================================================
15. SEARCH
================================================
Upper right, 200 to 260px wide.
- Magnifying glass icon, placeholder "Find a structure", "/" keyboard shortcut hint
- Results appear in a small floating dropdown
- Selecting a result focuses the camera, highlights the structure, and opens the information
panel

================================================
```

```
16. CAMERA FOCUS
================================================
Create a reusable focusOnObject(mesh) function: compute the bounding box and center,
calculate a good viewing distance, and tween camera.position and controls.target over 600 to
1000ms.

================================================
17. CREDITS
================================================
Add a "Sources & credits" panel that shows the BodyParts3D attribution exactly as its
license requires, with a link to the license page.

================================================
18. PERFORMANCE
================================================
GLTF caching, memoization, frustum culling, instancing and LOD where useful, and no React
rerenders per frame. Target 60fps on a modern laptop.

================================================
19. MICROINTERACTIONS
================================================
Button hovers about 150ms, panel fades 200 to 300ms, springy toggles, tooltip fades, camera
easing.
Do NOT add bouncy animations, neon effects, large shadows, gradients, or oversized text.

================================================
20. INITIAL LOAD
================================================
Full body centered, all panels visible. The UI and model fade in subtly. Nothing dramatic.

================================================
21. IMPLEMENTATION ORDER
================================================
1. Inspect the repo and dependencies
2. Asset build script and manifest
3. Full screen canvas and layout
4. Scene, lighting, orbit camera
5. Systems panel and visibility
6. Hover and raycasting
7. Selection and information panel
8. Camera focus
9. Atlas layout
10. Explode slider and animation
11. Search
12. Specimen collections
13. Camera toolbar
14. Isolate mode
15. Credits panel
16. Microinteractions
17. Performance
18. Visual polish pass

================================================
22. QUALITY BAR
================================================
Do not stop when it merely works. Before finishing:
- run the app and fix all TypeScript and build errors
```

```
- check the console for errors
- test every system toggle, search, selection, isolate mode, explode and collapse, zoom and orbit
- check the layout at common desktop sizes and on mobile
- do a final polish pass on spacing, type sizes, corner radii, camera framing, shadows, animation speed, and atlas density

It should feel like a production quality product, not a hackathon prototype or a generic Three.js demo.
```

Step 4: Test and Iterate

Your first result will not be perfect, and that is normal. The skill here is knowing what to ask for next. These follow up prompts fix the most common problems.

If the explode animation snaps or stutters

```
Move all position interpolation into useFrame. Each mesh should ease from assembledPosition to explodedPosition based on the slider value, without updating React state every frame.
```

If pieces land in the wrong system

```
Open the unclassified log and the manifest. Show me the 20 most suspicious system classifications and explain your rules, then fix them.
```

If it lags in the exploded view

```
Profile the draw calls in the exploded view. Add instancing or merge static meshes where it is safe, and keep every piece selectable.
```

For the final polish pass

```
Compare the current UI against the spec section by section. List every mismatch in spacing, sizes, colors, and animation, then fix them one by one.
```

Step 5: Deploy It

1. Push your project to a new GitHub repository (without the raw OBJ folder).

2. Go to vercel.com, click Add New Project, and import your repo.
3. Vercel detects Vite automatically. Click Deploy.
4. Open your live link on your laptop and your phone and test every feature.

The Part That Actually Matters

An AI agent can write the code. What makes this project yours is understanding what it wrote. These five concepts are your interview talking points. Make sure you can explain each one without looking at the code.

Raycasting

When you hover, the app shoots an invisible ray from the camera through your mouse position and checks which mesh it hits first. That is how it knows which of 1,600+ pieces you mean.

Linear Interpolation (Lerp)

Every piece stores two positions: where it sits in the body and where it sits in the atlas. The slider is just a number from 0 to 1, and every frame each piece moves to the eased point between its two positions. One number controls the whole body.

The Render Loop vs React State

Updating React state 60 times a second for over a thousand meshes would freeze the page. So animation lives inside the Three.js render loop, and React only handles the interface. This is the performance decision that makes the whole thing smooth.

Bounding Boxes and Camera Focus

To fly to a structure, the app draws an invisible box around it, finds its center and size, and backs the camera up just enough to fit it on screen. Then it animates there instead of teleporting.

The Data Pipeline

The hidden work is turning thousands of raw OBJ files named by ID into a clean, labeled, classified manifest. That is a real data engineering story, and it is often the most impressive part to talk about.

How You Know You Are Done

- Every system toggles on and off independently with a smooth fade
- The slider goes from 0% to 100% smoothly and every exploded piece is clickable
- Searching "gracilis" flies the camera straight to it
- The console is clear of errors and the explode animation runs smoothly
- The credits panel shows the BodyParts3D attribution
- You can explain raycasting and the explode interpolation to a friend

Stretch Goals

- **Quiz mode:** highlight a random structure and make the user guess its name
- **Share links:** put the selected structure in the URL so you can send someone straight to it
- **Left vs right compare:** show mirrored structures side by side
- **Study mode:** save structures to a personal flashcard list

Resume Bullet Template

Fill in your own real numbers. Do not make them up. Run the app and measure.

> **Template**
> *"Built an interactive 3D anatomy explorer in React, TypeScript, and Three.js rendering [n] labeled meshes from the open BodyParts3D dataset, with system layering, raycast selection, search, and an animated explode view that interpolates every mesh between anatomical and atlas layouts at [x] fps"*

The Shipping Checklist

A project nobody can open is not a project. Before you call it done:

- **A README with a GIF at the top.** Show the explode slider. Recruiters spend seconds on a repo.
- **A live link** at the top of the README and in your LinkedIn featured section.

- **One command to run it.** npm install, then npm run dev.
- **Attribution** for BodyParts3D in the app and the README.
- **Credit for your inspiration.** If you remade someone's idea, link to them. It is the right thing to do and creators love seeing remakes.
- **Three sentences on what you would do differently.** Interviewers quote this section back to you.

What To Do Once It Is Built

- Record a 30 second screen recording of the explode slider in action
- Write one post about the hardest bug you hit. The bug proves you built it yourself
- Add the resume bullet with your real numbers filled in

Go Build Something <3

Made by @hinumpy · Hinna

Anatomy data: BodyParts3D, © The Database Center for Life Science, CC Attribution 4.0 International