EVOKNOW.AI · AI Super Power Skills for Claude

Rebuild Any Image as a 3D Model

Getting started with img2threejs -- from a photo to code

img2threejs turns a single reference photo into a browser-ready, animation-ready 3D model -- written entirely as procedural Three.js code. No mesh files, no photogrammetry, no downloaded assets: just primitives, generated geometry, and shaders, gated through a staged pipeline that renders and visually compares its work at every step.

This guide covers what img2threejs is, how to install and run it, how its sculpting pipeline works pass by pass, and what you get at the end. If you can run Claude Code and have Python 3.10+, you can rebuild your first object today.

Repository: github.com/hoainho/img2threejs
Live gallery: hoainho.github.io/img2threejs-showcase
Published by EVOKNOW.AI · Guide version 1.0

1. What is img2threejs?

img2threejs is a skill for Claude Code (and other image-capable coding agents) that rebuilds an object from a reference image as a procedural Three.js model. Instead of extracting a mesh or running photogrammetry, it writes real code -- primitives, generated geometry, and shaders -- that reconstructs the object's shape, proportions, and colours.

The result is animation-ready and runs in the browser with zero external assets. Because the heavy, mechanical validation is handled by plain Python scripts, the agent spends its tokens where they matter: on visual judgement -- comparing each render against the reference and deciding what to fix.

2. What you need

- Claude Code (or another agent that can read images -- Codex, OpenCode, or your own screenshots all work; img2threejs is agent-agnostic).
- Python 3.10 or newer. The pipeline uses only the standard library -- no pip installs, no external dependencies.
- A browser and a Three.js runtime to view the generated model and comparison sheets.
- One clear reference image of the object, character, or hybrid you want to rebuild.

3. Installing img2threejs

Clone the repository straight into your Claude Code skills folder:

```
git clone https://github.com/hoainho/img2threejs.git ~/.claude/skills/img2threejs
```

That is the whole install -- no build step and nothing to configure.

4. Your first rebuild

In Claude Code, attach or point to an object image and invoke the skill:

```
/img2threejs Rebuild this object as a Three.js model, keep the
proportions, angles, and colours.
```

The skill first probes whether the image is a good candidate, then walks the full sculpting pipeline, rendering and scoring its work as it goes. You approve or let it self-correct until the model matches.

5. How the pipeline works

img2threejs uses a staged "sculpting" pipeline. Each phase is gated -- the model has to render and pass a visual score before the next stage unlocks:

- Suitability probe -- is this image a workable reference at all?
- Pre-spec assessment -- inventory every identity-defining detail before any code.
- Spec authoring -- write the structured ObjectSculptSpec (component tree, materials,

sockets).

Strict-quality validation -- a Python check enforces the spec is complete and sane.

Locked build passes -- the object is sculpted in ordered stages (see next section).

Browser render + comparison sheet -- render each pass beside the reference.

Agent vision review -- the agent scores the match and chooses continue, refine-spec, refine-code, request-input, or stop.

6. The build passes

Inside the build stage the model is constructed in a fixed, locked order -- each pass adds one layer of fidelity:

Blockout -- rough masses and overall proportions.

Structural -- the load-bearing geometry and how parts connect.

Form -- refined silhouettes and shape detail.

Material -- colours and material properties.

Surface -- surface detail and shading nuance.

Lighting -- how the model reads under light.

Interaction -- pivots, sockets, and moving parts.

Optimization -- trim cost while keeping the look.

7. What you get

A completed run produces a self-contained, animation-ready package:

A TypeScript factory -- createObjectNameModel(spec, options) -- that returns a THREE.Group you can drop into any Three.js scene.

An ObjectSculptSpec JSON: the full component tree, materials, sockets, and the review history from each pass.

A sculptRuntime hierarchy exposing pivots, sockets, colliders, and destruction groups -- so the model is ready to animate and interact with.

Browser renders and side-by-side comparison sheets for every pass.

8. Running the pipeline by hand

You can also drive each stage directly with the bundled Python scripts -- useful for debugging or scripting your own runs:

```
python3 forge/stage1_intake/probe_image.py <image>
python3 forge/stage2_spec/new_pre_spec_assessment.py "Name" \
--image <image> --out assessment.json
python3 forge/stage2_spec/new_sculpt_spec.py "Name" --image <image> \
--assessment assessment.json --out spec.json
python3 forge/stage2_spec/validate_sculpt_spec.py spec.json --strict-quality
python3 forge/stage3_build/generate_threejs_factory.py spec.json \
--out src/createObjectModel.ts
```

9. Objects, characters & honesty

img2threejs is strongest on hard-surface, mechanical objects -- earbuds, tools, weapons, chests. From v1.2 it also has an anatomy-aware character track with proportion-locking and facial landmarks, though characters are treated as stylized reconstructions rather than exact likenesses.

The skill also ships an explicit "honesty clause": when a result is approximate, low-poly, or simply cannot reach the fidelity you asked for, it says so rather than pretending otherwise -- so you always know what you are looking at.

10. Tips & troubleshooting

- Pick a clean, well-lit reference showing the object from a readable angle -- the suitability probe rejects images it cannot work from.
- Let the self-correction loop run; refine-code and refine-spec passes are where the match tightens.
- If /img2threejs is not recognized, confirm the clone landed at ~/.claude/skills/img2threejs and restart Claude Code.
- Browse the live showcase first -- seeing the earbuds, shotgun, and loot-chest rebuilds sets realistic expectations for your own objects.

Resources

- Repository & docs: github.com/hoainho/img2threejs
- Live demo gallery: hoainho.github.io/img2threejs-showcase
- More skills, reviews, and guides: evoknow.ai/ai-skills