CLAUDE CODE · THREE.JS · DESIGN

# How to build a particle theme that changes shape as you scroll

How I built the new dec5.tech homepage with Claude Code. Copy-paste prompts, a single-file working demo, and the bugs I hit along the way.

| TIME | LEVEL | STEPS | PROMPTS |
| --- | --- | --- | --- |
| 30-45 min | Intermediate | 5 | 6 |

The brain on my homepage scatters as you scroll and turns into a light bulb, a globe and finally my logo. A lot of you asked how in the comments. You can't copy a prompt from an Instagram comment, so everything lives here now. Hit the **Copy** button in the top-right corner of any block, paste, keep going.

## What we're making

Thousands of tiny triangles sit on the screen. Each one belongs to several shapes at once. As you scroll, they flow from one shape to the next. The idea comes from a site called Dala; I adapted it to mine.

You'll end up with three things:

1. A single-file demo you can open in the browser right away
2. A sequence of prompts to apply it to your own site

3. A way to swap particles for your own images

**INGREDIENTS**

- Claude Code. That's what I used, and the prompts are written for it.
- A website project. Mine is Next.js, but the demo works as plain HTML.
- The Playwright MCP connected to Claude Code so it can take screenshots. Without it, you'll do step 2 by hand.

**TOOLS**

Claude Code | Playwright MCP | three.js

## 01 Find a reference

Find a site you like. I use Refero Styles, which lists the colors, fonts and components of real sites.

But watch out, this was my first mistake. I applied Refero's color table, the site changed color, and it looked nothing like the reference. A site's real feel isn't in its colors, it's in its **motion**.

## 02 Make it watch the site first, then build

This is the most important step. Don't say "make it look like this site". Ask Claude to open the site, watch it, scroll it and tell you what happens first.

**PROMPT**

Open this site with Playwright: <SITE URL>

1. Watch it for 10 seconds without touching anything, take a screenshot every 3 seconds.
2. Then scroll from top to bottom step by step, taking a screenshot at every step.
3. Give me a table: at each scroll stage, which shape is on screen, where the text sits, and how the transitions happen.

Don't write code yet. Just tell me what you see.

Once you read the table, it's clear what to copy. For Dala, the flow was: brain → scatter → light bulb → globe → logo.

## 03 Try it in a single file first

See the mechanism before touching your project. Save the code below as `particles.html` and open it in the browser. No install; three.js loads from a CDN.

As you scroll, the sphere scatters and turns into a ring, then a cube. Live version here.

HTML

```
<!doctype html>
<html lang="tr">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-
scale=1" />
  <title>Parçacık Demo</title>
  <style>
    html, body { margin: 0; background: #000; color: #fff;
font-family: system-ui, sans-serif; }
    #scene { position: fixed; inset: 0; }
    section { min-height: 140vh; display: flex; align-items:
center; padding: 0 8vw; position: relative; }
    h2 { font-size: clamp(2rem, 6vw, 4.5rem); font-weight:
400; letter-spacing: -0.04em; max-width: 12ch; }
  </style>
</head>
<body>
  <div id="scene"></div>
  <section data-morph="0"><h2>Küre</h2></section>
  <section data-morph="1"><h2>Dağıl</h2></section>
  <section data-morph="2"><h2>Halka</h2></section>
  <section data-morph="3"><h2>Küp</h2></section>

  <script type="importmap">
    { "imports": { "three":
"https://cdn.jsdelivr.net/npm/three@0.184.0/build/three.modul
e.js" } }
  </script>
  <script type="module">
    import * as THREE from "three";

    const N = 5000;
    const COLORS = ["#8052ff", "#ffb829", "#ffffff",
"#e056fd", "#2fd3b5"];

    // Her şekil için N nokta üret. Aynı parçacık her şekilde
bir yer tutuyor.
    const sphere = (i) => {
      const u = Math.random() * 2 - 1, a = Math.random() *
```

```
Math.PI * 2, s = Math.sqrt(1 - u * u);
      return [s * Math.cos(a) * 1.6, u * 1.6, s * Math.sin(a)
* 1.6];
    };
    const chaos = () => [(Math.random() - 0.5) * 12,
(Math.random() - 0.5) * 7, (Math.random() - 0.5) * 7];
    const torus = () => {
      const a = Math.random() * Math.PI * 2, b =
Math.random() * Math.PI * 2;
      const x = (1.4 + 0.45 * Math.cos(b)) * Math.cos(a), y =
0.45 * Math.sin(b), z = (1.4 + 0.45 * Math.cos(b)) *
Math.sin(a);
      return [x, y * Math.cos(1.1) - z * Math.sin(1.1), y *
Math.sin(1.1) + z * Math.cos(1.1)]; // biraz eğ
    };
    const cube = () => {
      const p = [Math.random() * 2 - 1, Math.random() * 2 -
1, Math.random() * 2 - 1];
      p[Math.floor(Math.random() * 3)] = Math.random() < 0.5
? -1 : 1; // yüzeye yapıştır
      return p.map((v) => v * 1.2);
    };
    const shapes = [sphere, chaos, torus, cube];

    const geo = new THREE.BufferGeometry();
    shapes.forEach((fn, s) => {
      const arr = new Float32Array(N * 3);
      for (let i = 0; i < N; i++) arr.set(fn(i), i * 3);
      geo.setAttribute("aP" + s, new
THREE.BufferAttribute(arr, 3));
    });
    geo.setAttribute("position", geo.getAttribute("aP0"));
    const col = new Float32Array(N * 3), seed = new
Float32Array(N * 2), c = new THREE.Color();
    for (let i = 0; i < N; i++) {
      c.set(COLORS[i % COLORS.length]);
      col.set([c.r, c.g, c.b], i * 3);
      seed.set([Math.random() * 6.28, Math.random()], i * 2);
    }
    geo.setAttribute("aColor", new THREE.BufferAttribute(col,
3));
    geo.setAttribute("aSeed", new THREE.BufferAttribute(seed,
```

```
2));

    const uniforms = { uMorph: { value: 0 }, uTime: { value:
0 }, uPixel: { value: 1 } };
    const mat = new THREE.ShaderMaterial({
      uniforms,
      transparent: true,
      depthWrite: false,
      blending: THREE.AdditiveBlending,
      vertexShader: `
        attribute vec3 aP0; attribute vec3 aP1; attribute
vec3 aP2; attribute vec3 aP3;
        attribute vec3 aColor; attribute vec2 aSeed;
        uniform float uMorph; uniform float uTime; uniform
float uPixel;
        varying vec3 vColor; varying float vRot; varying
float vSize;
        vec3 shape(float i) {
          if (i < 0.5) return aP0; if (i < 1.5) return aP1;
if (i < 2.5) return aP2; return aP3;
        }
        void main() {
          float i0 = floor(uMorph), f = fract(uMorph);
          float t = smoothstep(0.0, 1.0, clamp(f * 1.6 -
aSeed.y * 0.6, 0.0, 1.0)); // parçacıklar sırayla kalkar
          vec3 p = mix(shape(i0), shape(min(i0 + 1.0, 3.0)),
t);
          float a = uTime * 0.15; p = vec3(p.x * cos(a) + p.z
* sin(a), p.y, -p.x * sin(a) + p.z * cos(a));
          p.x += 1.3; // metne yer aç: şekli sağa al
          vec4 mv = viewMatrix * vec4(p, 1.0);
          gl_PointSize = min(0.05 * uPixel / -mv.z, 64.0);
          gl_Position = projectionMatrix * mv;
          vColor = aColor; vRot = aSeed.x + uTime * 0.3;
vSize = gl_PointSize;
        }`,
      fragmentShader: `
        varying vec3 vColor; varying float vRot; varying
float vSize;
        float seg(vec2 p, vec2 a, vec2 b) { vec2 pa = p - a,
ba = b - a; return length(pa - ba * clamp(dot(pa, ba) /
dot(ba, ba), 0.0, 1.0)); }
```

```
        void main() {
          vec2 p = gl_PointCoord * 2.0 - 1.0; float c =
cos(vRot), s = sin(vRot); p = mat2(c, -s, s, c) * p;
          vec2 a = vec2(0.0, 0.82), b = vec2(-0.71, -0.41), d
= vec2(0.71, -0.41);
          float e = min(min(seg(p, a, b), seg(p, b, d)),
seg(p, d, a)); // içi boş üçgen
          float px = 2.0 / vSize, alpha = 1.0 - smoothstep(px
* 0.5, px * 1.5, e);
          if (alpha < 0.01) discard;
          gl_FragColor = vec4(vColor, alpha * 0.8);
        }`,
    });

    const scene = new THREE.Scene();
    scene.add(new THREE.Points(geo, mat));
    const camera = new THREE.PerspectiveCamera(40, 1, 0.1,
50);
    camera.position.z = 6;
    const renderer = new THREE.WebGLRenderer({ alpha: true
});

document.getElementById("scene").appendChild(renderer.domElem
ent);

    function resize() {
      const dpr = Math.min(devicePixelRatio, 2);
      renderer.setPixelRatio(dpr);
      renderer.setSize(innerWidth, innerHeight);
      camera.aspect = innerWidth / innerHeight;
      camera.updateProjectionMatrix();
      uniforms.uPixel.value = (innerHeight * dpr) / (2 *
Math.tan((camera.fov * Math.PI) / 360));
    }
    resize();
    addEventListener("resize", resize);

    // Kaydırma ⟶ hangi şekil: ekranın ortasına en yakın iki
bölüm arasında yumuşak geçiş.
    let target = 0;
    function onScroll() {
      const y = scrollY + innerHeight / 2;
```

```
      const marks = [...document.querySelectorAll("[data-
morph]")].map((el) => ({
        y: el.offsetTop + el.offsetHeight / 2,
        i: Number(el.dataset.morph),
      }));
      target = marks[0].i;
      for (let k = 0; k < marks.length - 1; k++) {
        if (y > marks[k].y) {
          const t = Math.min(1, Math.max(0, ((y - marks[k].y)
/ (marks[k + 1].y - marks[k].y) - 0.25) / 0.5));
          target = marks[k].i + (marks[k + 1].i - marks[k].i)
* t;
        }
      }
    }
    addEventListener("scroll", onScroll, { passive: true });
    onScroll();

    const clock = new THREE.Clock();
    renderer.setAnimationLoop(() => {
      uniforms.uTime.value = clock.getElapsedTime();
      uniforms.uMorph.value += (target -
uniforms.uMorph.value) * 0.06;
      renderer.render(scene, camera);
    });
  </script>
</body>
</html>
```

How it works, briefly:

- **Every particle has an address in every shape.** `aP0`, `aP1`, `aP2`, `aP3` are the same particle's position in four shapes.
- **Scrolling becomes a number.** `uMorph` is a sphere at 0, scattered at 1, a ring at 2. In between, two shapes blend.
- **Particles don't all leave at once.** `aSeed.y` gives each one a small delay, so the shape dissolves and re-forms instead of sliding.

- **The shader draws the triangles.** Each point is a small square on screen; the fragment shader draws an outlined triangle inside it.

## 04 Apply it to your site

If the demo works, move to your real project. Give the prompts in order. After each one, check the result in the browser before moving on.

**PROMPT**

Add a full-screen three.js particle scene to the homepage that stays fixed behind the page (position: fixed).

- Every particle should belong to several shapes at once. Shapes: brain, scattered chaos, light bulb, globe, and finally the site logo.
- Give every section a data-morph="<shape number>" attribute. Particles should flow to the shape of whichever section is closest to the middle of the screen.
- During a transition, particles should leave with small delays, not all at once.
- Particles are outlined triangles; colors: violet, amber, white, pink, teal.
- Particles close to the camera should blur (depth of field).
- If prefers-reduced-motion is on, no animation.

**PROMPT**

Build the logo shape from the logo file: <PATH TO LOGO FILE>
Draw the image onto a canvas, find the filled pixels, and scatter the particles across those pixels.

**PROMPT**

Open the page with Playwright on desktop (1440 wide) and mobile (390 wide).
Scroll to each data-morph section, take a screenshot and show me.
If shapes overlap the text or overflow the screen, fix it.

That last prompt matters a lot. When Claude says "done" without seeing its own result, something is usually missing.

## 05 Your own images instead of particles

On my site, my Instagram covers make up the brain. Ask for this as a separate step:

**PROMPT**

Turn some of the particles, alongside the triangles, into small cards made from the images in <IMAGE FOLDER>.

- Pack the images into a single texture atlas (each one small; keep the total under a few hundred KB).
- The cards should flow into the same shapes.
- Cards stay upright and don't spin. Slightly rounded corners.
- 150 cards on desktop and 80 on mobile is enough.

Don't load full-size images. My 40 covers were 6.8MB; the small versions came down to 300KB.

**TIPS**

**Everything is white, you can't see the shape.** Particles pile up and blow out to white. Lower the particle count, thin the lines, fade the particles on the back side.

PROMPT

The particles are too bright, the shape is unreadable. Reduce the particle count, make the triangle lines 1 pixel, and lower the opacity of particles on the back side of the shape.

**The scene doesn't show up at all.** If you pushed the canvas back with `z-index: -10`, the body's black background paints over it. Don't give the canvas a negative z-index; put it before the content in the page and give the content `position: relative`.

**On mobile it's one white blob.** The shape shrinks but the triangles stay the same size. On narrow screens, lower the particle size and opacity, and move the shape above the text.

DONE?

- [ ] I had the reference site watched and scrolled, and got the flow as a table
- [ ] I opened the single-file demo and it worked
- [ ] The scene sits behind my own site and the text is readable
- [ ] I checked desktop and mobile with screenshots
- [ ] With reduced motion on, the page stays still

Tell me on Instagram where you got stuck, and I'll add it to the doc.